Advanced OpenBSD Hardening

Wrongun & DC

June 2005

ssh://root:uncon@192.168.1.5

Lab Challenge

- Join the wifi net and ssh into the box using the account specified in the footer
- Try to pwn the box by adding an account for yourself or backdooring sshd

“Only one remote hole in the default install, in more than 8 years! “

So OpenBSD is uber secure, right?

- Actually, no… The default install has nothing enabled (except ssh)

  "No wonder it's secure, it's powered off!"

- Source-only patching strategy makes it difficult to roll out fixes to platforms w/o compilers (i.e. diskless firewalls, etc.)

Brief History of OpenBSD Vulnerabilities

- 30 March 05: Bugs in the cp(4) stack can lead to memory exhaustion or processing of TCP segments with invalid SACK options and cause a system crash.
- 14 Dec 04: On systems running sakmpd(8) it is possible for a local user to cause kernel memory corruption and system panic by setting psec(4) credentials on a socket
- 20 Sept 04: radius authentication, as implemented by ogin_radius(8), was not checking the shared secret used for replies sent by the radius server. This could allow an attacker to spoof a reply granting access to the attacker. Note that OpenBSD does not ship with radius authentication enabled

Brief History of OpenBSD Vulnerabilities

- Jun 2002: Apache chunked encoding vulnerability (remote uid=nobody) (Apache-nosejob.c)

** Your high priced security consultant's plane ticket: $1500 * Your high priced security consultant's time: $200/hour * RealSecure nodes all over your company: $200,000 * Getting owned by 0day: Priceless*

- *Gobbles June '02*

Proactive Approach to Security

- Source Code Audits
- Privilege separation
- Privilege revocation
- Chroot jailing
- New uids
- ProPolice
- strlcpy() and strlcat() size-bounded string copying and concatenation
- Memory protection
  - W^X
  - .rodata segment
  - Guard pages
  - Randomized malloc()
  - Randomized mmap()
  - atexit() and stdio protection

ProPolice

- Modifies GCC to catch many stack overflow issues at compilation time
- Re-orders objects on stack for safety
- Better than StackGuard
  - Works on more than just i386

W ^ X

- Memory pages shouldn't be both writable and executable
- w/o hardware support (i.e. 64bit Intels or various SPARC/RiSC) this may have serious performance considerations

Randomized Memory Management

- Malloc()
  - When you need to allocate less than a page
- Mmap()
  - A page or greater
- Result: each time you perform a memory allocation, you get a different address.
- Note: this breaks A LOT of apps, and the Obsd team blames app developers for writing rubbish code

Randomized Memory Management

- StackGap
  - Random 8 byte alignment for top of stack
- Randomize shared library order
  - May break stuff if loading lots of libraries
- Not insurmountable for attacker, but makes it difficult enough that many won't bother

Privilege Revocation

- Many progs run w/ 'revoked' privs:
  - Ping, portmap, traceroute, rwalld, pppd, spamd, httpd, named, authpf, etc.
  - Once process kicks off, it runs as unprivileged user. Attacks against setuid binaries running w/ privilege revocation won't succeed (unless they do prior to revocation!)

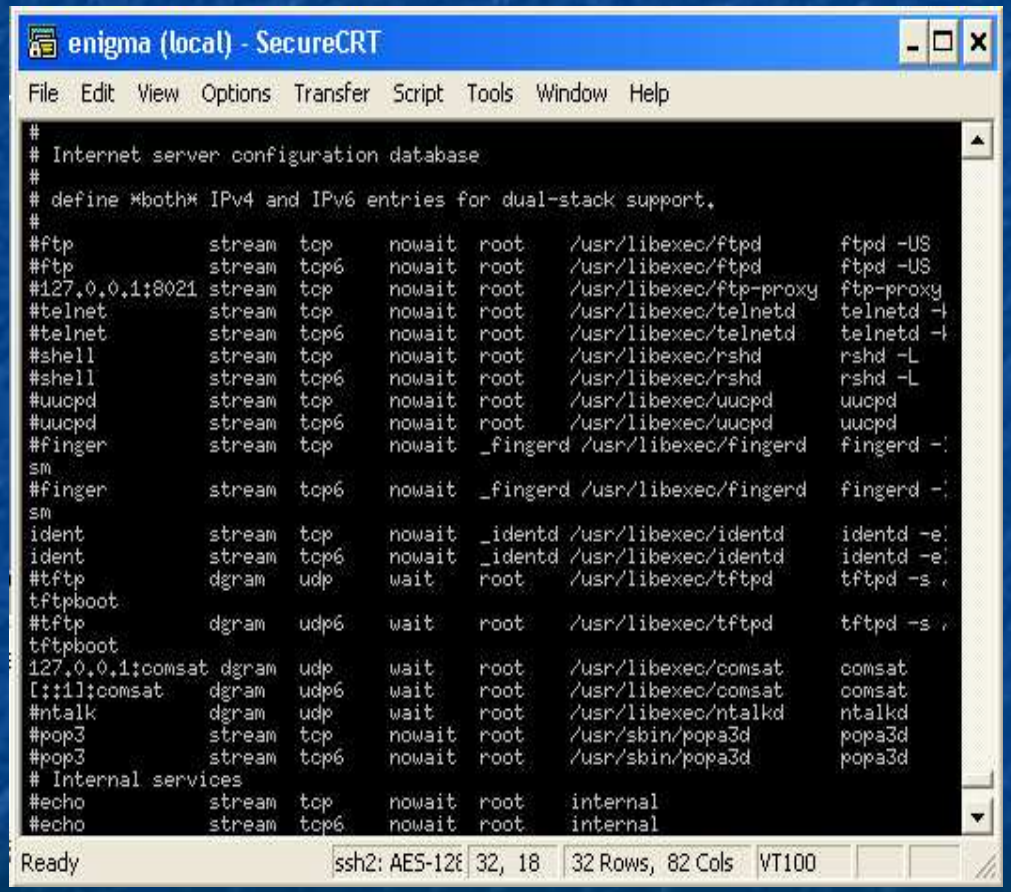
Privilege Separation

- Many progs run w/ 'separated' privs:
  - Ftpd, Sshd, syslogd, pflogd, isakmpd, bgpd, tcpdump, etc.
  - Once process kicks off, it forks. Most work is done by larger unprivileged process. Priv'd work is done by smaller process that retains privileges.
  - Inter-process communication accomplished by socketpair()
  - Non-trivial to code, however even Linux has adopted this (for sshd, and maybe some other tasks)

Hardening Basics

- Kill ftp-proxy
- Kill identd
- Kill daytime, time

- Hell, just kill inetd



ssh://root:uncon@192.168.1.5

Hardening Basics (cont)

- Disable root login via ssh
- Disable SSH prot ver 1
- pf (makes iptables look like a kludge)
  - Egress filtering is a pain, but will stop 99% of remote shells
  - PF AUTH can grant outbound perms to specific users
- Setup off-box logging w/ syslog-ng

Hardening Advanced Topics (chflags)

- sappnd set the system append-only flag (superuser only)
- schg set the system immutable flag (superuser only)
- uappnd set the user append-only flag (owner or superuser only)
- uchg set the user immutable flag (owner or superuser only)

Best practices:

- Flag binaries immutable w/ Schg
- Flag log files append only w/ sappend
  - Note this breaks newsyslog… deal w/ it ☺
- Note system must be in single user mode to unset these flags

Hardening Advanced Topics (Stephanie)

- Enable Trusted Path Execution (TPE)
  - based on code Mike Schiffman wrote for OpenBSD 2.4)
  - Only files owned by root are executable
  - Only users in trusted group can execute arbitrary non-root 0wned binaries
    - kern.security.trust_gid=666
  - Root can turn function on/off via sysctl
    - kern.security.tpe=1
- Note daemon users needs to be added to trusted group if their binaries are owned by !root

Hardening Advanced Topics (Stephanie)

- Enable VEXEC
  - Integrity verification of executed programs, memory mapped objects, and opened files. Uses hash tables. Supports MD5, SHA1, SHA256, SHA384, SHA512, and RMD160.

- Creates the Vexec pseudo-device
- Creates a fingerprint list of binaries listed in /etc/vexec.conf (using desired hash)
- Turn on via sysctl
  - kern.security.vexec.op=1
  - kern.security.vexec.verbose=1
  - kern.security.vexec.strict=0 (set this to 1 for extra fun!)

Hardening Advanced Topics (Stephanie)

- **VEXEC is essentially *realtime* TRIPWIRE**

```
thirtysix# Jun  2 11:28:29 thirtysix /bsd: vexec_verify: Fingerprint matches. (f
ile=/usr/bin/clear, inode=144518, dev=3)
Jun  2 11:28:29 thirtysix /bsd: vexec_verify: Fingerprint matches. (file=/usr/bi
n/clear, inode=144518, dev=3)

thirtysix# ^[
[A: Command not found.
thirtysix# w
vexec_verify: Fingerprint matches. (file=/usr/bin/w, inode=144538, dev=3)
Jun  2 11:28:33 thirtysix /bsd: vexec_verify: Fingerprint matches. (file=/usr/bi
n/w, inode=144538, dev=3)
Jun  2 11:28:33 thirtysix /bsd: vexec_verify: Fingerprint matches. (file=/usr/bi
n/w, inode=144538, dev=3)
vexec_openchk: Fingerprint matches. (file=/usr/lib/libkvm.so.8.0, dev=3, inode=3
09156)
11:28AM  54 secs, 1 user, load averages: 0.18, 0.06, 0.02
USER    TTY FROM              LOGIN@  IDLE WHAT
root     C0 -                11:28AM     0 w
thirtysix# Jun  2 11:28:33 thirtysix /bsd: vexec_openchk: Fingerprint matches. (
file=/usr/lib/libkvm.so.8.0, dev=3, inode=309156)
Jun  2 11:28:33 thirtysix /bsd: vexec_openchk: Fingerprint matches. (file=/usr/l
ib/libkvm.so.8.0, dev=3, inode=309156)

thirtysix# _
```

ssh://root:uncon@192.168.1.5

Hardening Advanced Topics (Stephanie)

- Enables userland privacy
- Finger
- Last
- Netstat
- W
- Who

Last version (for 3.6) at
http://www.innu.org/~brian/Stephanie/

Hardening Advanced Topics (Securelevel)

- The OpenBSD kernel provides four levels of system security:

- -1: Permanently insecure mode
  - init(8) will not attempt to raise the securelevel
  - may only be set with sysctl(8) while the system is insecure

Hardening Advanced Topics (Securelevel)

- 0: Insecure mode
  - used during bootstrapping and while the system is single-user
  - all devices may be read or written subject to their permissions
  - system file flags may be cleared

Hardening Advanced Topics (Securelevel)

- 1: Secure mode
  - default mode when system is multi-user
  - securelevel may no longer be lowered except by init
  - /dev/mem and /dev/kmem may not be written to
  - raw disk devices of mounted file systems are read-only
  - system immutable and append-only file flags may not be removed
  - kernel modules may not be loaded or unloaded
  - the fs.posix.setuid sysctl variable may not be raised
  - the net.inet.ip.sourceroute sysctl variable may not be raised

Hardening Advanced Topics (Securelevel)

- 2: Highly secure mode
  - all effects of securelevel 1
  - raw disk devices are always read-only whether mounted or not
  - settimeofday and clock_settime may not set the time backwards or close to overflow
  - pfctl may no longer alter filter or nat rules
  - the ddb.console and ddb.panic sysctl variables may not be raised

Breaking Out of Securelevels

- Non trivial, but we've found 3 ways
  - If /etc/ (dir) is schg, 2 methods
  - If /boot (file) is schg, but /etc is not…

Breaking Out of Securelevels [/etc not SCHG'd] : Method 1

- Lazyman's way (no style points tho ☹)
  - Simply mv /etc/ /etc.off
  - mkdir /etc && cd /etc.off; tar cf - . \
  - | (cd /etc; tar xpf -)

Breaking Out of Securelevels [/etc not SCHG'd]

Method 2 :: Stylepointz++

- If /etc/boot.conf doesn't exist... (and it doesn't by default !)
- Create /etc/boot.conf, and "reboot" from your own "customised" kernel ;-)
  - Set option INSECURE in the kernel config
  - Remove stephanie/vexec/etc
  - Remove securelevel code
  - Add in an openbsd rootkit backdoor
- Hell, it's your kernel, and hence your box!

Breaking Out of Securelevels

[/etc *is* SCHG'd, but /boot is writable]

- usr/mdec/biosboot
- first stage bootstrap
- /boot
- system bootstrap
- /etc/boot.conf
- system bootstraps startup file
- If we can write to the bootloader, we can install our own bootloader which looks for the boot.conf wherever we want to put it. And we would have gotten away with it too if it wasn't for those pesky kernels...

Quick and Dirty PPTP VPN

- Quick PPTPD setup using poptop & userland GRE, mppe & mschapv2
  - Echo "net.inet.gre.allow=1" >> /etc/sysctl.conf
  - Echo "net.inet.ip.forwarding=1" >> /etc/sysctl.conf
  - Cd /usr/ports/net/poptop; Make && make install
  - Echo "pptp:" >> /etc/ppp/ppp.conf
  - Echo "enable MSChapV2" >> /etc/ppp/ppp.conf
  - Echo "set ifaddr 10.0.0.254 10.0.1.69-10.0.1.79" >> /etc/ppp/ppp.conf
  - Echo "dc password * *" >> /etc/ppp/ppp.secret
  - /usr/local/sbin/pptpd;
  - Configure PF to taste

Quick and Dirty PPTP VPN

- This will give you a quick and effective PPTP (mschapv2) VPN server, compatible w/ WinXP native clients
- Good for l-users that need to publish web-content or pop/imap mail
- Not strong enough for proper system administration

Quick and Dirty PPTP VPN

- X.509 based IPSEC VPN is still the "real" way to do VPN
- See Schneier and Mudge on how well M$ cleaned up mschap to create MSChapV2 and decide for yourself
- http://www.schneier.com/paper-pptpv2.html

Strong Authentication for Remote Administration

- SSH with private key on smartcard
- OpenSSH client and server has support
- Userspace program 'sectok' to read/write to Cyberflex smartcards (and possibly others)

- If we want to use X.509 based auth we'll either need to patch sshd, or run a commercial sshd.
- Popular win32 ssh clients already support X.509 cert on smartcard (i.e. SecureCRT)
- Passwords are going away – even latest Debian installer disables ssh-pw-auth by default!

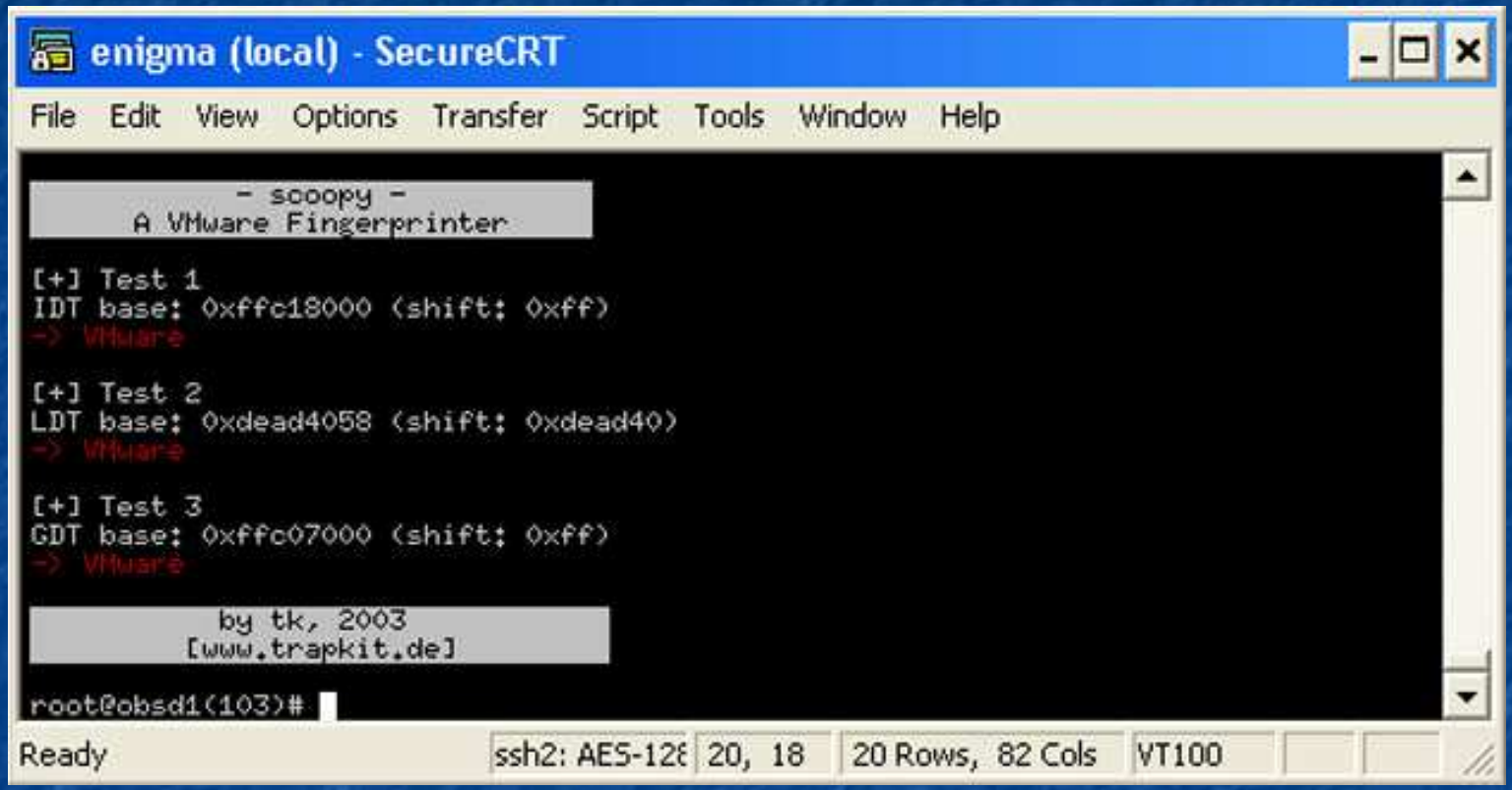
IPSEC

- OpenBSD has touted "native" IPSEC since 2.x.
- We can create site to site IPSEC tunnels with kernel IPSEC support and userspace isakmpd.
- No need for freeswan/openswan or kernel hacking like on Linux
- See man vpn for details

VMWare Detection

- On Intel architectures, it's possible to tell with some degree of certainty if an OpenBSD system we're using is "real" or "memorex"
- This can be done by attempting to write to Sensitive Register Instructions: SGDT, SLDT and SIDT
- VMWare systems write predictable values to the IDTR, LDTR, and GDTR (interrupt descriptor, local descriptor and global descriptor registers)

VMWare Detection



- See ttp://www.trapkit.de/research/vmm for more info

Questions?

- So is a locked down OpenBSD box actually usable?
- Do I sleep better at night knowing I run OpenBSD?

- ??????

References

- Theo de Raadt, CansecWest '03
  - http://www.openbsd.org/papers/csw03/index.html
- Stephanie
  - http://www.innu.org/~brian/Stephanie/
- Mudge & Bruce on M$PPTP (mschapv1&v2)
  - http://www.schneier.com/pptp-faq.html (pre sellout)
  - http://www.schneier.com/pptp.html (post-sellout!)
- VMware detection
  - http://invisiblethings.org/papers/redpill.html
    - Joanna you rock!!!
  - http://www.trapkit.de/#research