



EuSecWest/core06
February 20-21, 2006

Lessons in Open Source Security: The Tale of a 0-Day Incident

Andrea Barisani

Inverse Path Ltd

Chief Security Engineer

<andrea@inversepath.com>

Gentoo Foundation

Infrastructure Developer

<lcars@gentoo.org>

INVERSE  PATH





Introduction

DISCLAIMER:

All the scripts and/or commands and/or configurations provided in the presentation must be treated as examples, you use them at your own risk. Please review all the code before using it in any environment.

Copyright 2006 Andrea Barisani <andrea@inversepath.com>.

This work is released under the terms of the *Creative Commons Attribution-NonCommercial-NoDerivs License* available at <http://creativecommons.org/licenses/by-nc-nd/2.5>.

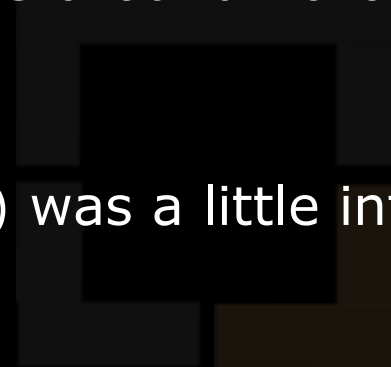


Topics - What's on the menu

- 0-day incident -> the *gentoo.org* rsync compromise
- global threats and real world trends
- Open Source project's infrastructure security and attack vectors
- paranoid and effective real world security practices that can save your day:
 - ♦ security monitoring
 - ♦ security hardening
- Open Source and security: myths and facts, what works and what does not (flamewar-proof suit not required but recommended)

The incident

- Gentoo/Linux uses rsync mirrors for propagating the portage tree, a ports repository used for building and installing all packages
- rsync1.it.gentoo.org (rsync.gentoo.org rotation member) suddenly froze on [2 Dec, 2003](#) somewhere between 05:30 and 08:00 (CET)
- the box was a fully updated Gentoo/Linux server running a 2.4.21-ac4 kernel (believed to be secure at the time)
- the server was power cycled and booted on 2 Dec, 2003 around 10:00 CET
- luckily enough the box administrator (your speaker ;)) was a little into security...



Traces of a compromise

- ...while starting up the box, email reports from the two (yes I'm paranoid) integrity checkers reported a new friend on the filesystem:

```
ADDITION: ["rsync1"] /usr/bin/chfgk
```

Inode	Permissions	NLink	Uid	Gid	Size	Created	On
242193	-rwsr-xr-x	1	0	0	4467	Dec 01 23:26	2003

- system logs before the reboot confirmed the compromise:

```
Dec 2 05:16:34 rsync1 kernel: eth1: Promiscuous mode enabled.
```

- so there was clear evidence of intrusion, at this point every administrator should follow the First Golden Rule of Incident Handling (Tm):



Traces of a compromise

- ...while starting up the box, email reports from the two (yes I'm paranoid) integrity checkers reported a new friend on the filesystem:

```
ADDITION: ["rsync1"] /usr/bin/chfgk
```

Inode	Permissions	NLink	Uid	Gid	Size	Created	On
242193	-rwsr-xr-x	1	0	0	4467	Dec 01 23:26	2003

- system logs before the reboot confirmed the compromise:

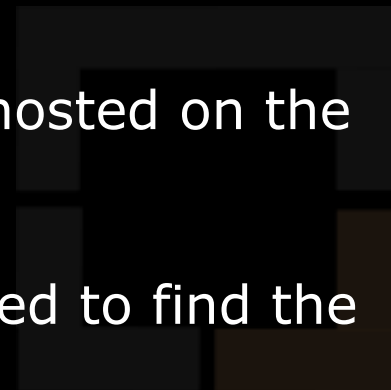
```
Dec 2 05:16:34 rsync1 kernel: eth1: Promiscuous mode enabled.
```

- so there was clear evidence of intrusion, at this point every administrator should follow the First Golden Rule of Incident Handling (Tm):

DON'T PANIC!

Containment

- ♦ as soon as the compromise is confirmed a series of containment actions must be performed depending on the server's function:
 - ♦ perform a preliminary “live” analysis of the system looking for the data that might be lost in a shutdown
 - ♦ evaluate the visibility of other systems in your network and the potential risk of further compromises
 - ♦ shutdown the affected services and possibly the whole box for off-line forensic analysis on a read-only disk image
 - ♦ revoke any private key (i.e. openssh, gpg) that was hosted on the filesystem
- since the server was supposedly fully patched we needed to find the currently unknown attack vector



- point of entry was not immediately clear from IDS (Snort) logs, since entries might have been deleted a quick 'n dirty forensic felt appropriate:

```
ws1 # strings /images/sda1 | grep -e "Dec 2.*snort"
```

- unsurprisingly deleted log entries showed up:

```
Dec 2 04:39:06 sole snort: [1:1322:5] BAD-TRAFFIC bad frag bits  
[Classification: Misc activity] [Priority: 3]: {TCP} 196.40.59.24 ->  
140.105.134.1
```

```
Dec 2 04:43:33 sole snort: [1:1322:5] BAD-TRAFFIC bad frag bits  
[Classification: Misc activity] [Priority: 3]: {TCP} 196.40.59.24 ->  
140.105.134.1
```

```
Dec 2 04:43:40 sole snort: [1:498:4] ATTACK-RESPONSES id check  
returned root [Classification: Potentially Bad Traffic] [Priority: 2]:  
{TCP} 140.105.134.1:873 -> 196.40.59.24:1084
```

Digging deeper

- rsync seemed to be the attack vector, no vulnerability for it was known at the time (**rsync 2.5.6**)
- Snort binary logs with packets dump were left intact (only logs were quickly deleted):

```

12/02-04:43:40.034532 140.105.134.1:873 -> 196.40.59.24:1084
TCP TTL:64 TOS:0x0 ID:43532 IpLen:20 DgmLen:76 DF
***AP*** Seq: 0x61885836 Ack: 0xCBC3F650 Win: 0xF8E0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 186318604 1350045231
0x0000: AA 00 04 00 71 96 00 80 C8 6E 6B 86 08 00 45 00   ....q....nk...E.
0x0010: 00 4C AA 0C 40 00 40 06 7E F4 8C 69 86 01 C4 28   .L..@.@.~..i...(
0x0020: 3B 18 03 69 04 3C 61 88 58 36 CB C3 F6 50 80 18   ;..i.<a.X6...P..
0x0030: F8 E0 64 3A 00 00 01 01 08 0A 0B 1A FF 0C 50 78   ..d:.....Px
0x0040: 0E 2F 75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69   ./uid=0(root) gi
0x0050: 64 3D 30 28 72 6F 6F 74 29 0A                          d=0(root).

```

- we can see the worst payload your server could possibly return to an external host: `uid=0(root) gid=0(root)`

- traffic just prior the “smoking gun” packet was consistent with an exploit attempt pattern:

```

196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18181 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18339 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18351 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18361 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18461 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18464 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18477 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18488 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18554 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18568 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:18676 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:21481 IpLen:20 DgmLen:1500 DF MF
...
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:24465 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:24879 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:25126 IpLen:20 DgmLen:1500 DF MF
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:25131 IpLen:20 DgmLen:1500 DF MF
140.105.134.1:873 -> 196.40.59.24:1084 TCP TTL:64 TOS:0x0 ID:43532 IpLen:20 DgmLen:76

```

- and let's go see the payload...



Nasty Fragments: payload

```
12/02-04:43:05.059659 AA:0:4:0:71:96 -> 0:80:C8:6E:6B:86 type:0x800 len:0x5EA
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:24402 IpLen:20 DgmLen:1500 DF MF
Frag Offset: 0x0000 Frag Size: 0x05C8
.8.i....`I.\...}xM.....Px.;...M@`..@`..@`..@`..@`..@`..@`..@`..
@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..
@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..
@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..@`..
```

```
12/02-04:43:33.577065 AA:0:4:0:71:96 -> 0:80:C8:6E:6B:86 type:0x800 len:0x5EA
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:25126 IpLen:20 DgmLen:1500 DF MF
Frag Offset: 0x0000 Frag Size: 0x05C8
.<.i....a.V...}x.....Px.^...n4p..4p..4p..4p..4p..4p..4p..4p..
4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..
4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..
4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..
```

```
12/02-04:43:33.577065 AA:0:4:0:71:96 -> 0:80:C8:6E:6B:86 type:0x800 len:0x5EA
196.40.59.24 -> 140.105.134.1 TCP TTL:50 TOS:0x0 ID:25131 IpLen:20 DgmLen:1500 DF MF
Frag Offset: 0x0000 Frag Size: 0x05C8
.<.i....a.V...}x.....Px.^...n4p..4p..4p..4p..4p..4p..4p..4p..
4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..
4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..
4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..4p..
```



Backdoor analysis

- the added file is a setuid binary with no debugging symbols

```
ws1 # file chfgk
chfgk: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for
GNU/Linux 2.2.0, dynamically linked (uses shared libs), stripped
```

- strings gives us a clue

```
ws1 # cat /images/rsync1/usr/bin/chfgk |strings|uniq|grep -v -e "^\.."
```

```
Password incorrect while getting initial ticket
```

```
Unable to identify user from password file
```

```
getpwuid
```

```
setgid
```

```
setuid
```

```
/bin/sh
```

```
getting initial ticket
```

```
HOME
```

```
getting principal from ccache
```

```
/etc
```

```
initializing kerberos library
```

```
/var/tmp
```

```
kadmin/changepw
```

```
SHELL
```

```
krb5_cc_default
```

```
Enter it again:
```

```
krb5_change_password
```

```
Enter new password:
```

```
krb5_get_init_creds_opt_init
```

```
GCC: (GNU) 3.3.2 20030831 (Debian prerelease)
```

```
krb5_get_init_creds_opt_set_fwd
```

```
PTRh
```

```
krb5_prompter_posix
```

```
PTRhP
```

Backdoor analysis

- the backdoor turned out to be a wrapper for spawning a root shell when a predefined environment variable is set:

```
sandbox1 $ export HOME=/etc
sandbox1 $ cd /var/tmp
sandbox1 $ /usr/bin/chkfg
sh-3.0#
sh-3.0# id
uid=0(root) gid=0(root)
```

- it was most likely left by the attacker for future logins on the box using a legit user
- there was no evidence of new users added to the system or old ones being modified, no other filesystem changes were detected
- looks like the box froze before giving the attacker the chance to complete his/her work

Pinpointing the attack vector

- everything pointed to the rsync daemon but no vulnerability for it was known at the time
- the box was a fully updated Gentoo/Linux server running a 2.4.21-ac4 kernel, a bug in the `do_brk()` function was silently fixed *28 Nov, 2003* in kernel 2.4.23 but no association with a security problem was reported
- *that same night* (while poor sysadmin was sleeping) the bug was linked to a *local privilege escalation* vulnerability (*CVE-2003-0961*)
- so the attack vector looked like it related to either rsync or the kernel bug or possibly both of them
- the Gentoo core team was immediately contacted with preliminary analysis, an advisory was released that *same day* informing our users about the problem
<http://www.gentoo.org/security/en/glsa/glsa-200312-01.xml>
- the rsync team was also immediately contacted for investigating a possible rsync vulnerability

Investigating rsync

- in the following 48 hours **Martin Pool**, **Andrew Tridgell**, **Paul "Rusty" Russel** and **Dave Dykstra** from the rsync team, **Mike Warfield** from Internet Security Systems and myself worked together exchanging all available data and packet dumps for auditing and finding the issue
- a few hours after sending initial email report I was on the phone with Australia talking to rsync folks
- an international group of people who didn't know each other (except by fame in some cases) started a collaboration process on their own time trying to solve this potentially critical issue
- **rsync 2.5.7** was released along with an advisory **4 Dec 2003, 05:55 GMT**, within 48 hours of the compromise

<http://lists.samba.org/archive/rsync-announce/2003/000011.html>

Heap Overflow

- a heap overflow was being used in combination with the `do_brk()` vulnerability to escalate privileges
- missing sanity checking or validation allows an attacker to convince rsync to allocate huge chunks of memory, this is simply a DoS attack, the server will run out of RAM/swap, however over in kernel land...
- ...kernel fails to check that a call to `brk()` won't map parts of the kernels memory space into the applications', combined with the error in rsync which allows an attacker to force rsync to call `brk()` an attacker may gain root privileges
- rsync now places a limit on the size of any one chunk of memory during it's allocation
- learned lesson: never underestimate vulnerabilities advertised as local



~~Press~~ Incident Handling

- having to handle the press is probably much worse than handling the 0-day itself

<http://it.slashdot.org/article.pl?sid=03/12/03/1921235>

http://news.com.com/Hacked+Gentoo+Linux+server+taken+offline/2100-7349_3-5113227.html

<http://news.zdnet.co.uk/software/linuxunix/0,39020390,39118285,00.htm>

<http://www.eweek.com/article2/0,3959,1402934,00.asp?kc=EWRSS03119TX1K0000594>

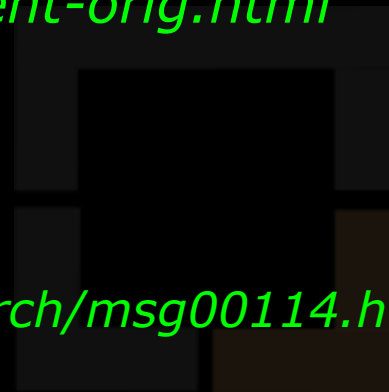
http://news.zdnet.com/2100-1009_22-5113227.html

and many more...

- on-line press has the power to deeply affect the perception of your project in the Internet community
- news like this spreads as fast as a nasty Worm
- journalist are not always technical-savvy and tend to misquote you
- when dealing with the press be *very careful* about the words you use and always ask to review any article before publication if possible

Do we see a trend here?

- for anyone that thinks these are isolated incidents:
 - 6 Nov, 2003 – attempted backdoor commit on linux kernel sources
<http://kerneltrap.org/node/1584/>
 - 21 Nov, 2003 - *debian.org* compromised
<http://lists.debian.org/debian-devel-announce/2003/11/msg00012.html>
 - 1 Dec, 2003 – *savannah.gnu.org* compromise announced
(**compromised 2 Nov, 2003!**)
<http://http://savannah.gnu.org/statements/statement-orig.html>
 - 2 Dec, 2003 – *rsync1.it.gentoo.org* compromised
 - 23 Mar, 2004 – *www.gnome.org* compromised
<http://mail.gnome.org/archives/gnome-announce-list/2004-March/msg00114.html>
 - 15 Jul, 2005 – *spreadfirefox.com* compromised



- Open Source Software related projects are major targets due to their exposure, visibility and relatively close connection with the hacking (and cracking) community
- other targets:
 - University's networks are often compromised due to the wide variety of systems and networks, usually with a very poor security level
 - some private companies are affected by industrial espionage issues, chances of full disclosure in that scenario are close to none
- in this context the major concerns are dedicated attackers and not script kiddies

OSS Projects attack vectors

- apart from standard vulnerabilities in running services OSS projects features an interesting range of secondary but yet effective attack vectors
- accounts creation/validation/revocation management and procedures
- shell accounts
- social engineering: IRC channels, forums
- developers competence and security awareness ranges from high to none, education and training is hard to accomplish and developers workstation security is a primary threat
- no “corporate network”, global overview of all resources and proper monitoring, resources and competences are spread and quality control of system administration is difficult



(Not so) Paranoid security best practices

- there are lots of OSS security solutions that can help us in monitoring and pro-actively defending our networks from potential compromises
- security is a process not a product as you all know, software can help but it's no substitute for security administration and awareness
- security monitoring is the front and most important line of defense
- no pro-active hardening is 100% safe, assuming that chances of intrusion are not null *awareness about a successful intrusion is vital*
- not performing integrity checking on your filesystems, network traffic analysis and log monitoring is just like being blind
- post-intrusion recovery times and severity grow exponentially from the time of the compromise



Monitoring: filesystem integrity checkers

- filesystem integrity checkers are more and more becoming full HIDS (Host Intrusion Detection Systems)
- basic requirements:
 - ◊ scalability, centralized management
 - ◊ off-site db, pull system for updates
 - ◊ support for the relevant OS of your network
 - ◊ architecture should be tamper resistant
- stealth features are appreciated but don't count on them, a secondary integrity checker exposed via crontab is normally an effective decoy
- software you should check:
 - ◊ **samhain** (<http://la-samhna.de/samhain>)
 - ◊ **osiris** (<http://www.hostintegrity.com/osiris/>)

Monitoring: log analysis

- your system usually provides a consistent amount of logs, a centralized collection and analysis is a basic layer for incidents detection and post compromise activity
- catch the alert before the intruder has the chance to delete it
- using **syslog-ng** and **stunnel** it's very easy to collect all logs to a central location
- lots of log analysis software are then available for reporting, depending on personal tastes and habits **logwatch** (<http://www.logwatch.org>), **SEC** (<http://kodu.neti.ee/~risto/sec>), **swatch** (<http://swatch.sf.net>)
- while we want you to make the choice we just can't help but advertise **tenshi** (<http://dev.inversepath.com/trac/tenshi>) as an elegant, simple and effective solution...



Monitoring: log analysis - tenshi

- **tenshi** is an Open Source log monitoring daemon written in perl
- it's capable of parsing more than 1.000.000 messages per day
- it can watch one or more log files (or a FIFO if syslog-ng is used) and matches and *summarizes* lines against user defined regular expressions
- regular expressions are assigned to *queues* with different reporting intervals and recipients
- special queues allows parsing of "last message repeated" logs, group matching/skipping for increased performance, pager friendly reports
- allows custom prefixes for non-syslog log messages
- sysadmin friendly: privileges dropping, signal handling for reload/restart/queues flush, multiple configuration files support (like xinetd.d)
- allows report filtering for colourization, gpg signing, etc...



Monitoring: log analysis - tenshi

```
set uid tenshi
set gid tenshi
set hidepid on
set logfile /var/log/messages
set logfile /var/log/tenshi.log
set mailserver localhost
set subject tenshi report
set queue main      tenshi@localhost sysadmin@localhost      [0 */6 * * *]
set queue critical  tenshi@localhost sysadmin@localhost      [now]
set queue pager     tenshi@localhost pager:93384@localhost [now] alert
repeat ^(?:last message repeated|above message repeats) (\\d+) time
trash ^Initializing USB
critical,pager ^Oops
group ^sshd(?:\\(pam_unix\\))?:
main ^sshd: fatal: Timeout before authentication for (.+)
critical ^sshd: Illegal user
main ^sshd: Connection from (.+)
main ^sshd: Connection closed (.+)
main ^sshd: Accepted keyboard-interactive/pam for (.+) from (.+) port (.+)
critical ^sshd\\(pam_unix\\): session opened for user root by root\\(uid=0\\)
critical ^sshd\\(pam_unix\\): session opened for user root by \\(uid=0\\)
main ^sshd\\(pam_unix\\): session closed for user (.+)
main ^sshd\\(pam_unix\\): session opened for user (.+)
main ^sshd\\(pam_unix\\): authentication failure; logname=
group_end
misc .*
```



Monitoring: log analysis - tenshi

```
From: tenshi@localhost
To: admin@localhost
Date: Tue, 22 Jan 2006 12:53:20 +0100
X-tenshi-version: 0.4
X-tenshi-hostname: localhost
X-tenshi-report-start: Tue Jan 31 12:37:51 +0100 2006
Subject: tenshi report [main]
```

host1:

```
20: sshd: Accepted publickey ____
20: sshd(pam_unix): session opened for user ____
20: sshd(pam_unix): session closed for user ____
1: sshd(pam_unix): sessione opened for user root by (uid=0)
```

firewall:

```
4: arpwatch: bogon 0.0.0.0 0:11:24:2a:d6:e2
1: sshd: Did not receive identification string from 10.7.92.140
```

mail:

```
34: sm-mta: ____: from=____,____relay=____
29: sm-mta: ____: to=____,____delay=____
22: clamd: SelfCheck: Database status OK.
14: pop3d-ssl: LOGOUT, user=____, ip=____
14: pop3d-ssl: LOGIN, user=____, ip=____
1: clamd: stream: Worm.Mytob.BZ FOUND
```

Monitoring: network traffic

- network traffic should be passively monitored on a dedicated box
- **Snort** (<http://www.snort.org>) is pretty much the standard in Network Intrusion Detection
- a huge amount of commercial and open source products are available for aiding large deployments (but remember *Keep It Short and Simple*)
- lots of documentation is available on the subject, but we take to chance to give you some advice:
 - tune the ruleset but if you don't get any false positives at all it means that something is wrong, don't try to avoid them at all costs
 - keep binary logs for easy tcpdump parsing and replaying
 - don't restrict addresses range too specifically, attack attempts on unused IP space is a valuable piece of information
 - spend efforts on proper deployment before messing with fancy GUIs

- there are many projects devoted to pro-active hardening of UNIX systems
- the cost/benefits ratio in terms of manageability and effectiveness varies among the different technologies
- deployment of these systems depends on your production environments characteristics, system administration resources and skills, upgrade policies and of course OS versions/flavours
- due to its nature "OpenBSD hardening" is a redundant phrase, we won't cover it not because it's not interesting or good but simply because there's no confusion and other hardening options would make no sense
- many of the concepts/patches we are going to see for Linux systems are already built-in on OpenBSD

<http://www.openbsd.org/papers/ven05-deraadt/index.htm>

Hardened Gentoo

- Linux systems lacks the consistency and centralized development of *BSD systems due to the massive '*bazaar*' style development involved
- Linux distributions attempt to provide several consistent projects for software packaging and maintenance
- however security hardening is a secondary concern for most distributions, despite availability of many hardening projects for the Linux kernel their use often involves manual installation (we are talking about invasive patches here, not a few applications)
- the **Hardened Gentoo** project extends the Gentoo portage system in order to provide reasonable installation and maintenance of all mainstream hardening techniques for Linux systems
- **no technology is Gentoo-specific**, everything can be installed on other Linux (and possibly *NIX) systems with patience and experience

Hardened Gentoo: PaX

- **PaX** is a kernel patch that tries to protect against buffer and heap overflow (OpenBSD uses W^X , rivers of blood were shed comparing the two, taking sides is cowardly left as an exercise to the reader)
- when a buffer is overrun an attacker can inject shellcode, PaX mitigates this problem with **Address Space Layout Randomization**
- ASLR enables random offsets for functions and buffers causing problems for an attacker that wants to craft return codes
- it's most effective when combined with **PIE** (Position Independent Executables), which can be easily enabled with Hardened Gentoo
- PaX also offers **paxexec** which allows executable segments to be executable and not writable and writable segments to be writable and not executable (software emulated on x86 hardware)
- it's available as a standalone project or included in Grsecurity patch, Hardened Gentoo ships it as hardened-sources and pax-utils packages



Hardened Gentoo: PIE/SPP

- the hardened toolchain (gcc, binutils, glibc) builds all code with **PIE** enabled and also provides **SSP** (Smashing Stack Protection)
`CFLAGS="-fPIE -fstack-protector-all" LDFLAGS="-Wl, -z,now -Wl,-z,relro"`
- the Gentoo portage tree patches (bad) applications that don't work with PIE/SSP enabled, that usually relates to poor code quality
- they are two separate userland layers and they don't require kernel patches, PIE is not a security measure per se but it take advantage of ASLR with very little overhead
- SSP is a very cost effective technology which protects against stack smashing by allocating a 'canary' for checking overflows
- all of this is completely transparent to the user
- OpenBSD implements SSP with the (mostly) same implementation

<http://www.trl.ibm.com/projects/security/ssp>



Hardened Gentoo: Mandatory Access Control

- Mandatory Access Control is the last layer of protection you would need
- it generally takes much time to properly configure and tune it, maintaining good MAC policies is no easy task
- on the other hand it's very effective in containing intruders and local users
- Hardened Gentoo allows relatively easy installation of **SELinux**, **Grsecurity** and **RSBAC**
- all three options can co-exist in your installation, the project goal is to provide you an easy choice and installation path for these technologies

<http://hardened.gentoo.org/selinux>

<http://hardened.gentoo.org/grsecurity.xml>

<http://hardened.gentoo.org/rsbac>

<http://www.nsa.gov/selinux>

<http://www.grsecurity.net>

<http://www.rsbac.org>



Hardened Gentoo: SELinux, RSBAC, Grsecurity

- SELinux is implemented as a **LSM** (Linux Security Module, rivers of blood here as well), a library (libselinux) and userland utilities for compiling the policies. It implements types and roles for enforcing policy.
- RSBAC (Rule Set Based Access Control) is a modular access control framework. Modules includes MAC, Privacy Model (PM), a replacement for Linux user management (UM), ACL, Linux Capabilities (CAP) control and process jail (JAIL)
- Grsecurity is a kernel patch that provides PaX, a MAC implementation and additional features for hardening the kernel like sane chroot(), TCP/IP stack values randomization where possible, proc fs restrictions, extensive auditing logs



Hardened Gentoo: problematic binaries

- outstanding problems related to ELF binaries are dealt with
- **RPATH**: some applications creates ELF's with hard coded library path pointing to '.' or with relative ones, such apps are patched
- **TEXTREL**: a relocation is an operation that substitutes a reference to a shared object with the real address in case the object is loaded in memory, text relocations happens in the text segment where executable code is placed. In order to allow non-writable text segments text relocations are eliminated with **PIC** (Position Independent Code).
- lazy bindings are removed by telling the dynamic linker to process all relocations before transferring control to the program (done on SUID files also by non-hardened Gentoo), it prevents Global Offset Table poisoning <http://www.gentoo.org/proj/en/hardened/pic-guide.xml>
<http://www.gentoo.org/proj/en/hardened/pic-fix-guide.xml>



OSS myths & facts: the good

- we've seen how working with Open Source results in effective collaborations as in the 0-day incident (and many more examples out there), something that you would hardly find in closed source software (WMF anyone?)
- having so many choices in OSS is a good thing and it's a 'good' competition that enhances quality of all projects and satisfies all tastes
- BSD vs Linux is not a matter of absolute truth, and that's why there's no point in actually comparing them directly, the development model is radically different. The comparison and choice belongs to the needs and requirements of your environment.
- projects like Hardened Gentoo (but this is not the only example) tries to fill the issues in the apparent chaos of Linux environments and exploit all the benefits and powerful technologies available



OSS myths & facts: the bad and the ugly

- it's finally time to put on the flamewar-proof suit ;)
- politics and bad attitude of some developers results in public flames which often blinds both development efforts from learning from each other's implementations/mistakes
- outstanding examples of OSS myths:
 - Sendmail and its security
 - is LSM the answer to everything?
- one example of problematic applications that everyone uses: [phpbb](#)
- one example of ugly applications that everyone uses: [OpenLDAP](#)
- one example of a wonderful application that for silly reasons is not widely used on Linux: [systrace](#)
- cross-application security bugs makes maintainers blame each other
- unnecessary forks, majority of forks die or disappear, wasted effort

Questions?

:-)

(shameless plug)

<http://www.inversepath.com>

[peace to phpbb and openldap developers. We do appreciate your efforts]
[thanks goes to: rob@inversepath.com, solar@gentoo.org]
[kudos to Dragos Ruiu and all CanSecWest/PacSec/EuSecWest folks]