The Case For grsecurity

Brad Spengler
Open Source Security, Inc.
2012

Overview

- What is grsecurity?
- History
- Why grsecurity exists
- Recent advances
- Response strategy
- Future improvements

What is grsecurity?

- Kernel patch for Linux 2.6.32, 3.2, and the current “stable” Linux
- Provides access control, auditing, chroot hardening, anti-bruteforcing, anti-infoleaking
- Includes PaX for defense against exploitation of memory corruption vulns (and more)

What is grsecurity? (cont.)

- Goals of detection, prevention, containment
- Drive up exploit development costs, hopefully require specific targeting
- Psychology of uncertainty – attempt using 0day and risk losing not only the vuln but exploit vectors used?

What is grsecurity? (cont.)

- Ideal for webhosting environments
  - First work was in webhosting, so I experienced the problems first-hand
  - Very difficult security environment, can't just throw Apache in a VM
- Generally years ahead of mainstream security
  - See http://forums.grsecurity.net/viewtopic.php?f=7&t=2574for some examples

History

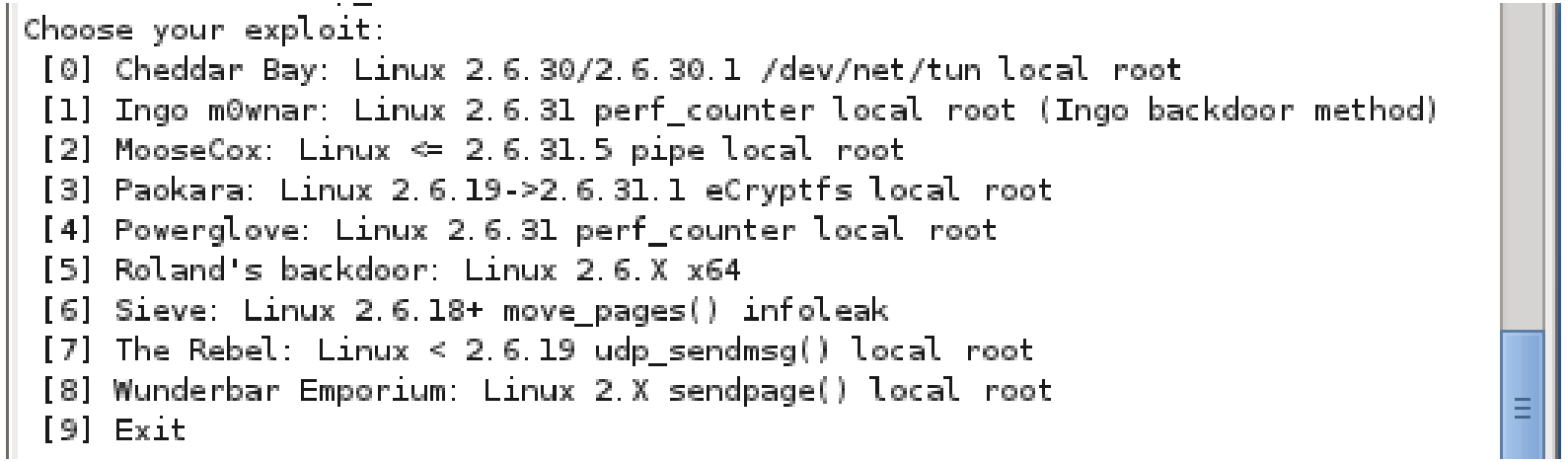
- Feb 18, 2000 - First release
  - then called “GRKERNSEC”
- Poor port of Openwall to 2.4 kernels
  - 2.4 unsupported by Openwall at the time
- 2001 – Included PaX
- 2001 – Michael Dalton creates “Oblivion” ACL system for grsecurity

History (cont.)

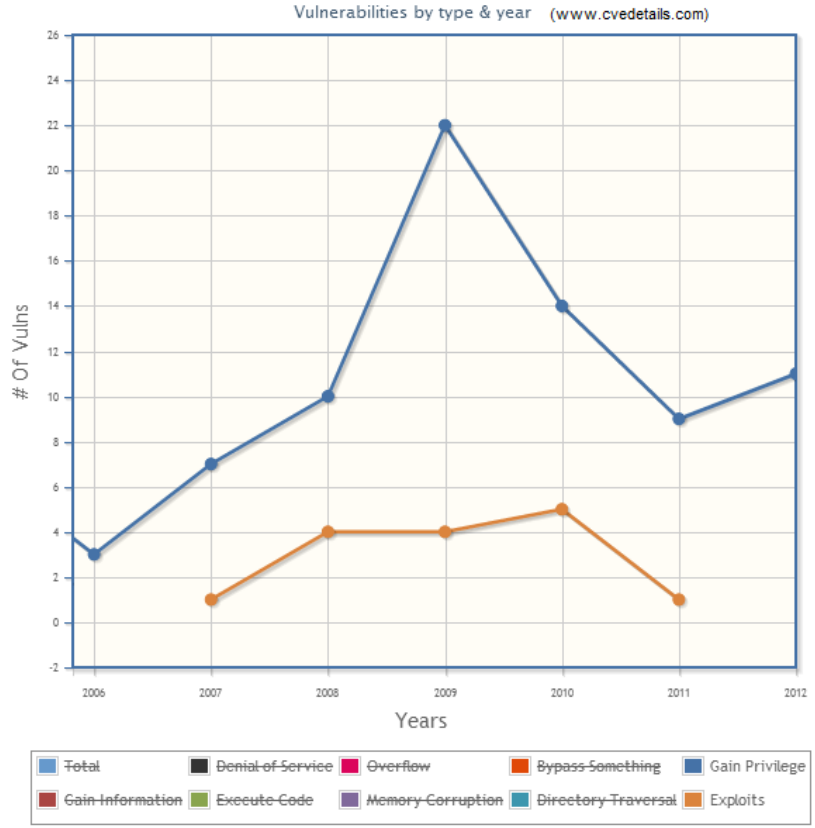
- Aug 3, 2002 – I create learning mode for ACL system
- Sept 2002 – Anti-bruteforcing, IP tagging/tainting
- April 6, 2003 – RBAC system, more advanced learning (full system policies)
- 2004 - HIDESYM
- 2009 – USERCOPY, limited size overflow prevention, MODHARDEN, fptr constifying
  - See http://grsecurity.net/news.php#develup

Why grsecurity Exists

- Because a few hours over a couple months nets:



Why grsecurity Exists (cont.)



"I'll be curious to see what the CVE statistics are like for the kernel this year when they get compiled next year -- I'm predicting that when someone's watching the sleepy watchers, a more personal interest is taken in doing the job that you're paid to do correctly." – exp_moosecox.c, 2009

Why grsecurity Exists (cont.)

- Culture of anti-security upstream
  - “I literally draw the line at anything that is simply greppable for. If it's not a very public security issue already, I don't want a simple "git log + grep" to help find it.” – Linus Torvalds, LKML
  - “I just committed this to mainline, and it should also go into stable. It's a real DoS fix, for a trivial oops (see the security list for example oopser program by Oleg), even if I didn't want to say that in the commit message ;)” – Linus Torvalds, not LKML
  - “I have tried to camouflage the security fix a bit by calling it a PROT_NONE fix and using pte_read(), not pte_user() (these are the same on x86). Albeit there's no formal embargo on it, please consider it embargoed until the fix gets out.” – Ingo Molnar, 2005, private bugtraq for RHEL

Why grsecurity Exists (cont.)

- Vendor-sec compromised at least twice
  - 2005, 2011 (finally shut down)
  - No accountability, sat on IA64 hardware DoS for two years
  - Embargoed vulns basically guaranteed head-start for blackhats
- Replacement list is better, but lessons learned from vendor-sec show failure of reactive security
- Users disempowered when information is controlled by a few (see http://blog.xen.org/index.php/2012/08/23/disclosure-process-poll-results/, decision to pre-release to "genuine cloud providers")

Why grsecurity Exists (cont.)

- Eight "stable" kernel trees
- Upstream focus is on adding new features (with new vulns)
  - From series of infoleak vulns found by Mathias Krause (minipli):
    - 11 affected 2.6.32 (released 2010)
    - 15 affected 3.2 (released Jan 2012)
    - 17 affected 3.5 (released July 2012)

Why grsecurity Exists (cont.)

- Vuln is DoS if not clever enough to exploit
  - See sudden spike in 2009 of privesc
- Generally no defense in depth on the kernel level
  - beyond copying grsecurity, that is
- Find bug / patch bug cycle
  - Whitelist vs blacklist
  - Exploit vectors vs vulnerabilities
- The “many eyes” of open source are blind, uninterested, or selling to governments for profit (it’s not the 1992 AD scene anymore)

Why grsecurity Exists (cont.)

- 3.x uname stack infoleak fixed in grsec Sept 19th, mentioned in both grsec and PaX changelogs
  - "Fix 3.x uname emulation infoleak" in grsec
  - "fixed kernel stack disclosure in sys_newuname affecting linux 3.x" in PaX
  - Not spotted for several weeks by anyone else, notified Google
  - Patch submitted recently, finally in Linus tree Oct 19
- Many eyes, right?

Recent advances

- Since 2011:
  - GRKERNSEC_BRUTE
    - Bruteforce deterrence for suid/sgid binaries
  - GRKERNSEC_MODHARDEN
    - mount via root can only auto-load filesystem modules
    - Netdev code can only auto-load netdev modules
    - No udisks auto-load

Recent advances (cont.)

- Since 2011:
  - GRKERNSEC_KERN_LOCKOUT
    - Attack by uid 0 or in interrupt handler, panic()
    - Attack by non-priv user, ban until reboot
  - PAX_USERCOPY
    - Whitelisting of slab caches that can be used for copies to/from userland
    - Ex: no copying to/from cred, task, dentry structs

Recent advances (cont.)

- Since 2012:
  - GRKERNSEC_PTRACE_READEXEC
    - Disallow ptracing unreadable binaries
  - GRKERNSEC_SETXID
    - Uid 0 setuid to non-root, change performed across all threads
    - Required per-arch changes
  - GRKERNSEC_SYMLINKOWN
    - Race-free implementation of Apache's SymLinksIfOwnerMatch

Recent advances (cont.)

- Since 2012:
  - GRKERNSEC_PROC_MEMMAP
    - Per-CPU, non-overflowable exec ID to ensure sensitive /proc entries can only be read/written by the same process that opened them
    - Arg/env pages limited to 512KB for suid/sgid binaries (defuse entropy reduction)
    - RLIMIT_STACK bounded, 3GB personality cleared to prevent alternate memory layout for suid/sgid binaries

Recent advances (cont.)

- Since 2012:
  - GRKERNSEC_HIDESYM
    - Reused PAX_USERCOPY slab cache whitelisting code, made generic caches
    - Made seqfile code allocate out of whitelisted generic cache
    - Added check to *printf() that sanitizes kernel pointers printed with %p in buffers allowed to be copied to userland
    - Prevented useful leak via /proc/net/ptype (hi Dan!)

Recent advances (cont.)

- Backported ~110 security fixes to the 2.6.32.59 kernel in 2012 that upstream missed
  - Notified maintainer, who added ~70 of these to 2.6.32.60 based on my changelogs
  - Number of backports are even higher for newer kernels, as many vulns are in code recently introduced

Response strategy

- Motivation for many advances: spite
- Scorched-earth exploit response
  - "A scorched earth policy is a military strategy or operational method which involves destroying anything that might be useful to the enemy while advancing through or withdrawing from an area." - Wikipedia
  - Upstream kills the vulnerability exploited, we kill exploit vectors found along the way
  - Must be weighed against produced disincentive to publish, as this harms reactive security users more than us

Response strategy (cont.)

- Stackjacking (2011)
  - 30 minutes advance notice, killed in a week before repeat presentation
  - Original presentation "demo" needed an artificial, best-case arbitrary-write and infoleak vuln
  - 6 enhancements made to grsecurity/PaX which have been improved further since
  - A year later, still presenting on the same techniques that were "promptly demolished by the PaX Team" – Jon Oberheide

Response strategy (cont.)

- Sudo format string vuln (VNSecurity, 2012)
  - 6 improvements made to grsecurity/PaX
    - Most already mentioned
    - Increased heap randomization in higher order bits
    - Increased stack randomization in lower order bits on x64
    - Small randomization in gap between program stack and arg/env strings
  - Despite all this, however, VNSecurity still able to create a one-shot exploit, aided by some unique sudo characteristics
    - Very nice work! See the progression here: http://www.vnsecurity.net/2012/02/exploiting-sudo-format-string-vunerability/
    - Short term vs long term strategy

Future improvements

- Kernel self-protection in place pushes many exploits into the code-reuse + infoleak space
- Drive up complexity of code reuse, force some data attacks into this space (e.g., cred struct modification)
- Eliminate known offsets/heuristic scanning as a technique against important kernel targets (GCC plugin)

Future improvements (cont.)

- Make it easier – official, unique kernel packages without distro kernel drawbacks
- RBAC improvements
- Improved learning system using real machine learning algorithms instead of heuristics
  - Not just reduction of path accesses to directories, but regular expression learning for more usable policies across software updates
- Automatically mark PaX flags for problem apps with a simple configurable daemon

Questions/Requests?

- Feel free to email me at spender@grsecurity.net
- http://www.grsecurity.net

- Thanks to my sponsors for their support

- Most of all, thanks to pipacs and Emese ☺