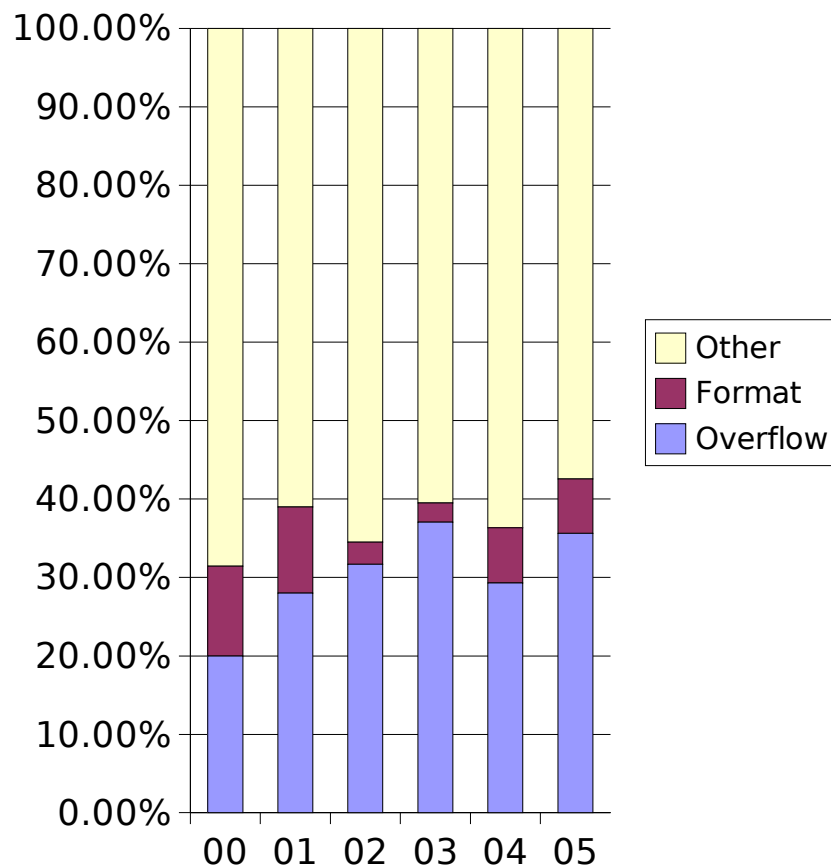


Taking the fun out of smashing the stack

Sean Burford
sean.burford@ultri.cx
linux.conf.au 2005
security miniconf

Stack, Heap and Format String Bugs

Debian Advisories '00-'05



- Questionable statistics 😊
- 874 bugs
- 274 stack or heap
- 53 format string
- 45 integer overflow

Exploit Requirements

- A bug can be found and triggered
- Code can be inserted or located and run to perform the desired action
- Program execution can be redirected to this exploit code
- Arguments may be needed for the exploit code (eg. `execve` takes the command to execute as an argument)

Improving Userland Resistance

- Remove the bugs and do not add more
 - use safer programming languages
 - audit code, with help from eg. [flawfinder](#)
 - audit memory usage with tools eg. [valgrind](#)
- Modify compilers and libraries to warn about bad code or detect exploitation
 - stack canaries (eg. [Immunix StackGuard](#) or [IBM ProPolice](#))
 - harden libraries (eg. [Immunix AppArmor](#))

Improving Kernel Resistance

- Modify the kernel to prevent execution of code injected into processes
 - mark data memory as non executable
 - randomise memory space
 - load programs into addresses that are incompatible with certain exploit constraints
 - limit potential for damage via mandatory access control

Non-executable memory

- 32 bit Intel architecture problem
- Other architectures offer support execute permissions on memory
- Two IA32 solutions
 - Utilise the split data/instruction TLBs
 - Segment the memory space between instructions and data
 - Both apply rules during page faults

Address Space Randomisation

- Some memory is readily relocatable
 - Dynamic Shared Libraries
 - Stack & Heap Space
- Code and Data
 - Choice of two methods:
 - Dynamic executables compiled with position independent code
 - Mirror code at random location and normal location, and check for attack signature during page faults

Kernel Stack Randomisation

- Randomise the location of the userland to kernel stack with every system call
- Exploit code injected onto the kernel stack no longer has predictable address

ASCII Armour

- Load code into addresses containing 0x00 (0x00***** - 0x0100*****)
- Makes it hard to construct address or pass arguments by exploiting string functions
- Position dependent code can be placed into ASCII armoured addresses at compile time with a linker patch

Mandatory Access Control

- Control privilege at a finer level to strengthen “minimum privileges” model
 - eg. Allow network services to bind some privileged ports but not others
- Can be applied to user shells
- Protects from lucky or brute force attackers who get past randomisation

PaX kernel patch

✓ No Execute

- Segmentation based + CPU NX page based
- Kernel page protection also supported
- mprotect() restricted to enforce W^X

✓ Address Space Randomisation

- Text for ET_DYN executables
- All libraries, stack and heap

✓ Kernel Stack Randomisation

PaX kernel patch

❑ ASCII Armour

- Not supported?

✗ Mandatory Access Control

- Not in scope of this project

✓ Features can be enabled/disabled on a per binary basis

Exec-Shield kernel patch

✓ No Execute

- Segmentation based + CPU NX page based
- No restriction on mprotect

✓ Address Space Randomisation

- Text for Position Independent Executables
- All Libraries, Stack and Heap

✗ Kernel Stack Randomisation

- Not supported

Exec-Shield kernel patch

✓ ASCII Armour

- PIE executables are mapped inside the 1-16M ASCII Armour range where possible

✗ Mandatory Access Control

- Not in scope of this project

✓ Most features can be enabled/disabled on a system wide, per binary or per invocation basis

- setarch i386 disables most features

GRSecurity

- Collection of kernel patches including PaX
- Same protection as PaX plus:
 - ✓ Mandatory Access Control via GRSecurity ACL system
 - Various hardening, randomisation and auditing improvements

Hardened Distributions

- **Hardened Gentoo**
 - Gentoo based with PaX, GRSecurity, SE Linux and **RSBAC** (Rule Set Based Access Control)
- **Adamantix**
 - Debian based with PaX, ProPolice, RSBAC
- **RedHat** Fedora Core and RHEL 3+
 - Include Exec-Shield and SE Linux

Summary

- Basic protection requires no configuration beyond using a suitable kernel
- Advanced protection requires applications be compiled as PIE or ET_DYN
- Further protection can be provided by configuring MAC profiles

Thank You

Sean Burford
sean.burford@ultri.cx
linux.conf.au 2005
security miniconf

Additional Resources

- Exec-Shield
 - [Linux Exec Shield Overflow Protection](#)
 - [Security Enhancements in RHEL](#)
- PAX
 - [Documentation](#)
 - [The Guaranteed end of arbitrary code execution](#)
 - [PaXtest](#) regression test suite
- SELinux
 - [LWN article on SCC TE patent](#)
 - [SCC patent Statement of Assurance](#)