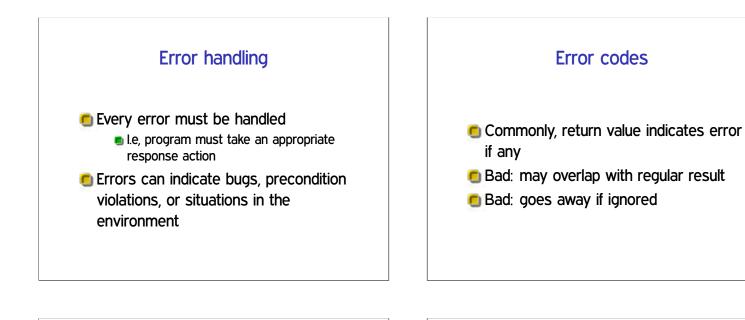
CSci 5271 Introduction to Computer Security Day 8: Defensive programming and design, part 2

Stephen McCamant University of Minnesota, Computer Science & Engineering

Outline

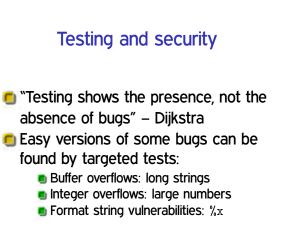
Software engineering for security

- Secure use of the OS
- Announcements intermission
- Bernstein's perspective
- Techniques for privilege separation



Exceptions

- Separate from data, triggers jump to handler
- Good: avoid need for manual copying, not dropped
- May support: automatic cleanup (finally)
- Bad: non-local control flow can be surprising



Fuzz testing

- Random testing can also sometimes reveal bugs
- Original 'fuzz' (Miller): program </dev/urandom</p>
- Modern: small random changes to a benign input

Outline

Software engineering for security

Secure use of the OS

Announcements intermission

Bernstein's perspective

Techniques for privilege separation

Avoid special privileges

- Require users to have appropriate permissions

 Rather than putting trust in programs

 Anti-pattern 1: setuid/setgid program
- Anti-pattern 2: privileged daemon
- But, sometimes unavoidable (e.g., email)

One slide on setuid/setgid

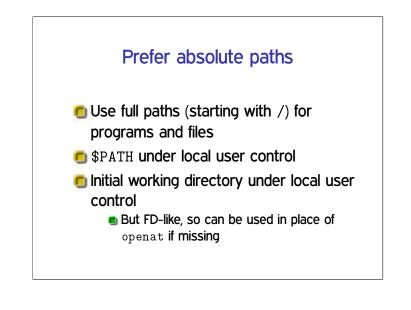
- Unix users and process have a user id number (UID) as well as one or more group IDs
- Normally, process has the IDs of the use who starts it
- A setuid program instead takes the UID of the program binary

Don't use shells or Tcl

- … in security-sensitive applications
- String interpretation and re-parsing are very hard to do safely
- Eternal Unix code bug: path names with spaces

Prefer file descriptors

- Maintain references to files by keeping them open and using file descriptors, rather than by name
- References same contents despite file system changes
- Use openat, etc., variants to use FD instead of directory paths



Prefer fully trusted paths

- Each directory component in a path must be write protected
- Read-only file in read-only directory can be changed if a parent directory is modified

Don't separate check from use

Avoid pattern of e.g., access then open
 Instead, just handle failure of open
 You have to do this anyway
 Multiple references allow races
 And access also has a history of bugs

Be careful with temporary files

 Create files exclusively with tight permissions and never reopen them
 See detailed recommendations in Wheeler
 Not quite good enough: reopen and check matching device and inode
 Fails with sufficiently patient attack

Give up privileges

Using appropriate combinations of set*id functions

 Alas, details differ between Unix variants
 Best: give up permanently
 Second best: give up temporarily
 Detailed recommendations: Setuid Demystified (USENIX'02)

Whitelist environment variables

- Can change the behavior of called program in unexpected ways
- Decide which ones are necessary As few as possible
- 🖲 Save these, remove any others

Outline

Software engineering for security

Secure use of the OS

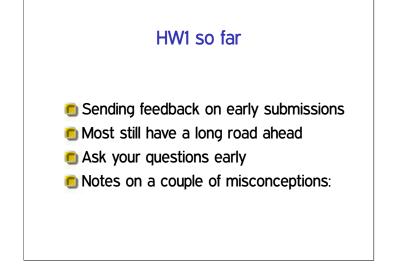
Announcements intermission

Bernstein's perspective

Techniques for privilege separation



- Project progress reports: Wednesday night
- 🖲 HW1 final submission: Friday night
- Exercise set 2: week from Thursday



Stack and non-stack buffers

```
char global[100];
void f(void) {
    char stack[100];
    char *heap = malloc(100);
}
```

printf and sprintf

Buffer overflow and format strings
printf("%s\n", untrusted)

sprintf(buf, "%s", untrusted)

Changed office hours this Thu/Fri

- John will be traveling
 Should have already submitted HW1 groups
 Thursday 10-11am will substituted by
- Stephen in his office 4-225E
- Friday will substituted by Mike, 3-4pm in 2-209

Outline

Software engineering for security

Secure use of the OS

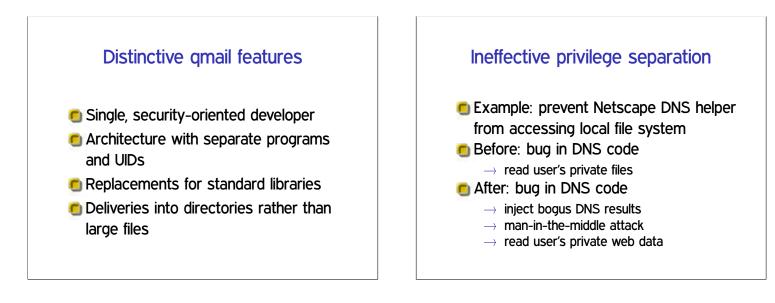
Announcements intermission

Bernstein's perspective

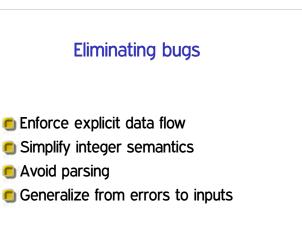
Techniques for privilege separation

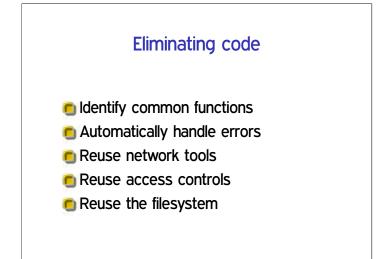
Historical background

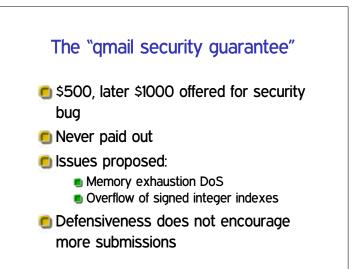
- Traditional Unix MTA: Sendmail (BSD)
 - Monolithic setuid root program
 - Designed for a more trusting era
 - In mid-90s, bugs seemed endless
- Spurred development of new,
 - security-oriented replacements
 - Bernstein's qmail
 - Venema et al.'s Postfix











qmail today Originally had terms that prohibited modified redistribution Now true public domain Latest release from Bernstein: 1998; netqmail: 2007 Does not have large market share All MTAs, even Sendmail, are more secure now

Outline

Software engineering for security

Secure use of the OS

Announcements intermission

Bernstein's perspective

Techniques for privilege separation

Restricted languages

- Main application: code provided by untrusted parties
- Packet filters in the kernel
- JavaScript in web browsers Also Java, Flash ActionScript, etc.

SFI

- Software-based Fault Isolation
- Instruction-level rewriting like (but predates) CFI
- Limit memory stores and sometimes loads
- Can't jump out except to designated points
- 🖲 E.g., Google Native Client

Separate processes

- OS (and hardware) isolate one process from another
- Pay overhead for creation and communication
- System call interface allows many possibilities for mischief

System-call interposition

- Trusted process examines syscalls made by untrusted
- Implement via ptrace (like strace, gdb) or via kernel change
- 🖲 Easy policy: deny

Interposition challenges Argument values can change in memory (TOCTTOU) OS objects can change (TOCTTOU) How to get canonical object identifiers? Interposer must accurately model kernel behavior Details: Garfinkel (NDSS'03)

chroot

- Unix system call to change root directory
- Restrict/virtualize file system access
- Only available to root
- 🖲 Does not isolate other namespaces

OS-enabled containers

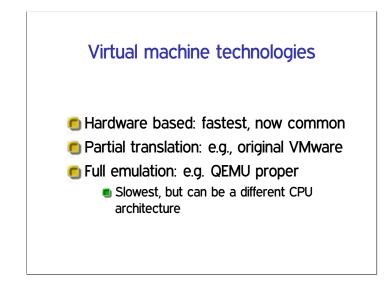
- One kernel, but virtualizes all namespaces
- FreeBSD jails, Linux LXC, Solaris zones, etc.
- Quite robust, but the full, fixed, kernel is in the TCB

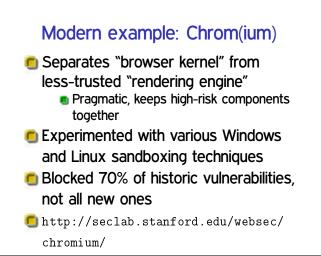
(System) virtual machines

- Presents hardware-like interface to an untrusted kernel
- Strong isolation, full administrative complexity
- I/O interface looks like a network, etc.

Virtual machine designs

- (Type 1) hypervisor: 'superkernel' underneath VMs
- 🖲 Hosted: regular OS underneath VMs
- Paravirtualization: modify kernels in VMs for ease of virtualization





Next time

Protection and isolation
 Basic (e.g., classic Unix) access control