Foundations of Network and Computer Security

John Black

Lecture #26
Dec 2nd 2004

CSCI 6268/TLEN 5831, Fall 2004

Announcements

- Proj #3 – Due Today
- FCQs – At end of class today
- Quiz #4: Discuss Solutions
- Next Week: firewalls and a final review
- Following week: Final Exam (Mon, Dec 13th)

Intrusion Detection (IDS)

- An intruder is either external or internal
  - External breaks into your system
    - Gains access to files or completely takes over the machine
  - Internal is already on your system, but wants to escalate privileges or gain access to unauthorized areas
    - Normal user becomes root, eg
- An IDS attempts to detect these events and alert the proper authority

Anomalies and Misuse

- Anomaly Detection
  - Static form is defined
    - Code and data must conform to some precise set of rules; easy to specify this
  - Dynamic behavior is defined
    - Behavior of users is evaluated as acceptable or anomalous; much harder to define
- Misuse Detection
  - Monitors for specific types of penetration attempts
    - Virus scanners, eg
    - Other “pattern checkers” (more later)

Anomaly Detection

- Static part is easy
  - Certain code or data on the disk should never change; make sure it doesn't
- Dynamic part is hard
  - Typical approach: scan audit logs
    - Audit logs are produced by the OS
    - IDS looks at log entries and uses various technologies to evaluate whether there is something bad going on
      - Statistical tests, neural nets, machine learning, genetic algorithms, AI, etc.

TripWire – Static Analysis

- Unix Based
  - File structure and inodes are recorded along with meta-data
  - meta-data specify what can change in the inode of a given file
  - Uses hashing and signatures to check whether a file has changed or not
- Self-NonSelf is not as famous but its inventor may be coming to CU

Dynamic Anomaly Detection

- Typically need to specify “normal” behavior
  - Get a “base profile”
    - For each user, typical log-in time/location, favorite editor, average bandwidth consumed, length of interactive session, and common sequences of actions
    - Older approaches measure statistical deviation from this profile and flag if deviations are too great
    - Some systems gradually update the profile over time as user behavior changes

NIDES

- Next-generation Intrusion Detection Expert System (SRI)
  - Statistical system with three measure classes:
    - Audit record distribution – tracks the types of audit records generated over some interval
    - Categorical – transaction-specific information (user name, file names, machine names accessed by user)
    - Continuous – a count of any event such as elapsed user CPU time, number of open files, number of pages read from disk, etc

NIDES – Continuous

- Continuous measures are sorted into "bins"
  - A bin might be (eg) total memory size of a user's processes; profile is the distribution of these measures over time
  - Each bin is compared with its corresponding base distribution

NIDES – Statistics

- NIDES stores basic statistics
  - frequencies, means, variances, and covariances
    - Storing the audit data in detail is too space-consumptive
  - Given a profile with n measures, NIDES characterizes any point in the n-space of measures as anomalous if sufficiently far from an expected value
    - Eg, two standard deviations
- All statistical measures are exponentially decayed
  - Older measures have less weight than newer measures

UNM – Pattern Matching

- UNM – University of New Mexico
  - Forrest is currently there
- For privileged processes, profile which system calls are made
  - Eg, sendmail might call:
    - open, read, mmap, mmap, open, read, mmap
    - Parameters are ignored
    - Profile varies depending on what sendmail is doing
      - Forwarding is different from bouncing, is different from sending to multiple recipients
      - All these are profiled

UNM – Typical database size

- sendmail 1318 Kbytes
- lpr 198 Kbytes
- ftpd 1017 Kbytes

- Of course not ALL legal usages are captured here
- Each program above was known to have vulnerabilities (3 for sendmail, 1 each for lpr and ftpd)

UNM – Results

- All intrusions using these vulnerabilities were detected
  - Suppose an attacker exploits a sendmail buffer overflow and adds a backdoor to the password file, spawns a new shell that listens on port 80
  - Sequence of calls might be: open, write, close, socket, bind, listen, accept, read, fork
  - Highly suspicious sequence would be caught by UNM
- Does this mean it works?
- pH is a similar system by the UNM people we'll see in a minute

Attacking IDS's

- Assume attacker knows how the IDS works
- Assume "normal" behavior for a system
  - An approximation could be obtained by running the IDS on a "normal" system for a while
- Assume initial penetration leaves no system call trace
  - Eg, a buffer overflow
- Assume IDS is watching what attacker does after initial penetration
  - This is what most of them are looking for—anomalous behavior

Slipping under the Radar

- Don’t execute any system calls
  - Web pages defacement
  - Changing emails dynamically
  - Some (rare) exploits that don’t require system calls
    - Old Solaris bug: cause a divide by zero trap and you get to be root!
- Probably the harm caused without any system calls is limited, however

Be Patient

- Attacker simulates the IDS and waits until the malicious sequence won't be tagged as anomalous
  - This might not ever happen, but then again it might!
  - This might happen, but the subsequent events after the exploit might trigger an alarm
    - Perhaps it's too late then?
    - Perhaps we can crash the application and a sys admin would be used to this?
      - Blue Screen of Death

Parameter Replacement

- Since parameters are ignored in system calls on (almost) all IDS's
  - Replace open("/lib/libc.so", O_RDONLY)
  - with open("/etc/shadow", O_RDWR)

- Since open() call is expected, no anomaly is reported
- Subsequent read() and write() calls are similarly expected

Insert no-ops

- Between calls to create malicious behavior the attacker does useless stuff
  - Attacker wants to call bind() and listen() but this will trigger an alarm
  - Attacker instead does bind(), getpid(), open(), close(), etc, on some arbitrary file name
  - Then calls listen()
  - IDS looks only a sequences of calls of some fixed length (typically 6)

Mimicry Attacks: Best Known Attack

- Model the IDS
  - In pH we need to know the last 5 system calls in order to know whether the next one will trigger an alert
  - Build a FSM M to tell us what we can do to avoid detection given the current state
  - Find a path through M such that we get all our system calls without raising an alarm

pH: Process Homeostasis

- Like other UNM system but
  - pH slows down the application by delaying each system call for 2^m time units
    - m is the number of mismatched length-6 subtraces in the last 128 system calls

An Attack

- Wagner and Soto attacked pH using an off-the-shelf exploit
  - wuftp FTP server and autowux.c exploit
  - autowux.c exploits a format string vulnerability in wuftp and then
    - calls setreuid(0,0)
    - break out of chroot jail
    - calls execve() with /bin/sh

Digression: chroot jail

- On Unix you can use chroot() to set the root directory of a process to some other directory
  - Example: chroot("/etc/apache/home");
    - Now if process executes chdir("/") it goes to the apache home directory
    - Note that "/bin/sh" won't work!
  - Breaking out is easy if we're root:
    - make a temp directory
    - chroot to that directory
    - Use chdir("..") a bunch of times to move up to the real root
    - chroot(".")

Back to the Attack

- Wagner and Soto trained pH to understand “normal” wuftpd operation
  - Hundreds of files downloaded over two day period
- Tested the exploit and pH caught the behavior
  - Have to be careful not to train pH that the exploit is “normal”!

How to get around pH?

- At the point the exploit occurs, the last 5 calls are
  - fstat, mmap, lseek, close, write
  - Now we know what calls we can make next
- Exploit wants to make these calls
  - setreuid(0,0), dup2(1,2), mkdir(“sh”), chroot(“sh”), 9 x chdir(“..”), chroot(“.”), execve(“/bin/sh”)
    - can collapse 9 chdir’s into chdir(“../../../../../../../../..”)

Attack Target

- Need to build
  - N* setreuid N* dup2 N* mkdir N* chroot N* chdir N* chroot N* execve
    - N* indicates 0 or more no-ops, where the no-ops come from a pre-defined list
- Wagner and Soto's tool found there was no path through M that found a way to get this sequence
  - But they noted another attack squence they COULD get

Alternate Attack

- First, use an existing dir instead of creating a new one
- Second, put a backdoor into the passwd file instead of spawning a shell
- New sequence:
  - setreuid(0,0), chroot(“pub”), chdir(“../../../../../../../../../..”), chroot(“.”), open(“etc/passwd”, O_APPEND | O_WRONLY), write(fd, “toor:AAaaaaaaaaaa:0:0::/:/bin/sh”, 33), close(fd), exit(0)

read() write() close() munmap() sigprocmask() wait4()
sigprocmask() sigaction() alarm() time() stat() read()
alarm() sigprocmask() *setreuid()* fstat() getpid()
time() write() time() getpid() sigaction() socketcall()
sigaction() close() flock() getpid() lseek() read()
kill() lseek() flock() sigaction() alarm() time()
stat() write() open() fstat() mmap() read() open()
fstat() mmap() read() close() munmap() brk() fcntl()
setregid() open() fcntl() *chroot()* *chdir()* setreuid()
lstat() lstat() lstat() lstat() open() fcntl() fstat()
lseek() getdents() fcntl() fstat() lseek() getdents()
close() write() time() open() fstat() mmap() read()
close() munmap() brk() fcntl() setregid() open() fcntl()
chroot() chdir() setreuid() lstat() lstat() lstat()
lstat() *open()* fcntl() brk() fstat() lseek() getdents()
lseek() getdents() time() stat() *write()* time() open()
getpid() sigaction() socketcall() sigaction() umask()
sigaction() alarm() time() stat() read() alarm()
getrlimit() pipe() fork() fcntl() fstat() mmap() lseek()
close() brk() time() getpid() sigaction() socketcall()
sigaction() chdir() sigaction() sigaction() write()
munmap() munmap() munmap() *exit()*

Moral of the Story

- IDS systems are useful but still not perfect
- Any new IDS system should provide implementations for evasion testing
- Perhaps parameters should be looked at as well?