Security Attacks

Network Mgmt/Sec.

Jim Binkley

Outline

- the methodology
- the original worm and the lessons we learned (sic)
- DOS attacks of late
- network analysis and passwords
- ip address authentication
- Melissa and her friends (viruses/java)
- tcp and sequence numbers
- any lessons here ?
  - well, yes

the methodology of the attackers

- surveillance
  - find hosts (IP address search)
  - find type of host (os fingerprint), firewalls too
- find KNOWN bugs (known to them)
- exploitation post break-in
  - escalation of privilege, user attacks root
- hiding their tracks post Or pre break-in
  - root shells on UNIX

1988 - the Morris worm: problems included:

- fingerd gets does not check buffer-length on input - results in root shell for attacker
  - » **buffer-overflow** attack
- an idiot bug in sendmail that allowed attacker to fire up shell
  - – DEBUG opttion not turned off
- using rsh/rcp/rshd .rhost scheme (IP address authentication) to break into nearby sites (exploitation post break-in)

cont.

- password attacks
  - try built-in dictionary, try idiot guesses (no passwords), try /usr/dict/words
  - read /etc/passwd “result” and try to match
- fanout attacks included
  - looking at .forward since if we cracked this system, maybe user has same password on that system?
- worm tried to hide (fork and kill parent)

fingerd program BEFORE

- char line[512];   /* automatic storage */
- line[0] = ‘\0’
- gets(line); /* user to be fingered from stdin*/
- Morris fed it a carefully constructed program that caused a root shell to be executed

VAX buffer attack code

```
pushl $68732f '/sh\0'
pushl $6e69622f '/bin'
movl  sp, rl0
pushrl $0
pushrl $0
pushrl r10
pushrl $3
movl sp, ap
chmk $3b
```

result equivalent to:

- execve(“/bin/sh”, 0, 0);
- so root shell executed when main returned
- attacking system would have TCP connection to root shell and could proceed with other attacks
- lesson: **buffer overflows attempt to gain a privilege via an already privileged server**
  - UNIX attacks/exec or write (e.g., write passwd)

here's the fix, but it doesn't take

- **fgets(line,sizeof(line),stdin);** /* the fix */
- other fixes include:
  - no stack execution
  - Crispin Cowan's stackguard
- defence mechanisms may include:
  - staying patched (hard especially if 1000 systems)
  - firewalls
- don't use C, after all, java doesn't have security problems (sic)

lessons learned

- explicit check for stack overflow!?
  - yes, but the program was the 1st java applet?
- passwords should not be in the dictionary
  - /etc/passwd should not be readable by the world (cut down on brute-force “crack” attempts)
- permissions of daemons should not be root
  - limited to that daemon only (least privilege)

cont.

- weak authentication mechanisms based on ip address (.rhosts) should be viewed with circumspection
  - lots of cases of this though, NFS, DNS, rsh of course, ACL firewall mechanisms
- sendmail too (check out qmail)
- Ref: The Internet Worm Program: An Analysis, Eugene H. Spafford, Purdue, CSD-TR-823, November 1988.

note two-kinds of attacks in general (from net, from user)

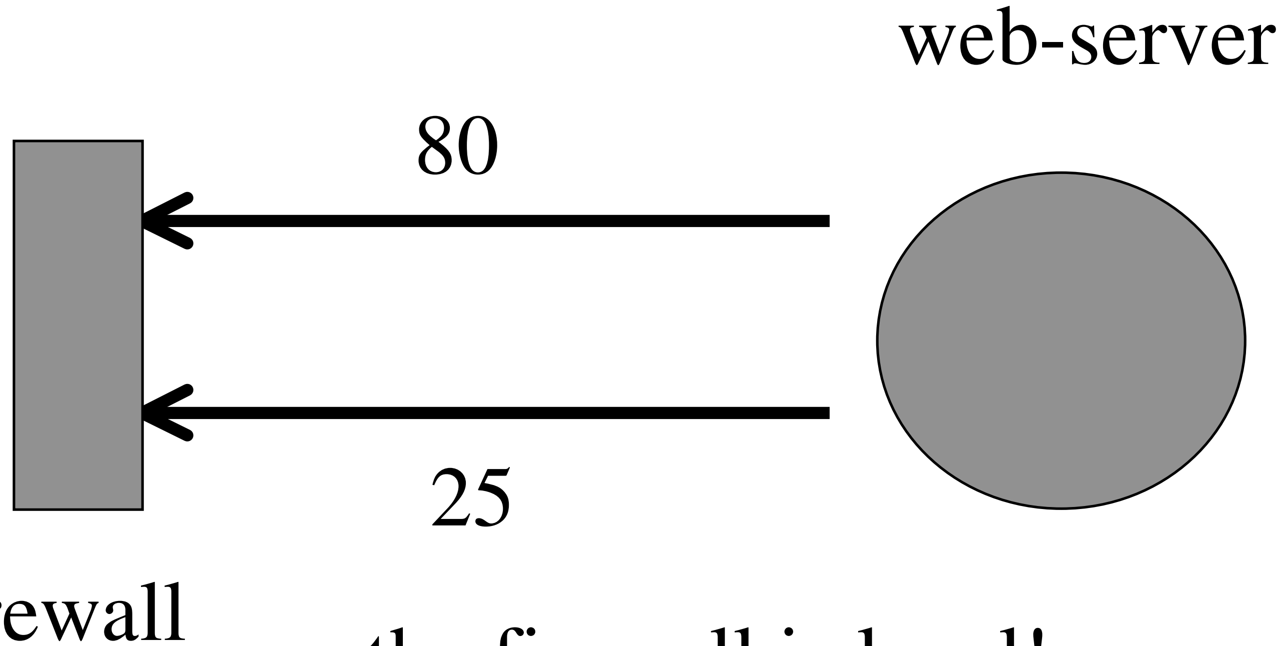
- attacks **over the network** (e.g., buffer overflow, or password guessing)
- may result NOT in root penetration, but in user account penetration
- which in turn may lead to attempt to take over system from "inside" - **multi-user user attacks**
  - lots of root-exploitation attacks on UNIX possible from access to local system

web-server/cgi/classic - phf

- phone book cgi script came with early NCSA apache web servers
- due to bug, could be used to execute any command locally
- e.g., send /etc/passwd away for computational crack attack
- fire up xterm or telnet to get “back-channel” from server out
- solution: remove the script (or all cgi)

consider back-channel hack

telnet to /bin/sh to telnet



the firewall is local!

DOS attacks (denial of service)

- not Disk Operating System Stupid ...:->
- famous ping of death
- winnuke (handout)
- land (handout)
- smurf (handout)
- ip fragmentation attacks (teardrop, etc.)
- note: often one-way and may use ip spoof
- new DDOS attacks, combine some of these
- what next?

network analyzers and ASCII passwords

- assume box X is hacked and bad bart has root access
- box X is on traditional network X
- bart installs tcpdump (or whatever) and starts sniffing for telnet passwords/ftp (non-anon) passwords
  - allows bart to attack other systems
- death of promiscuous mode is not good for bart ... (not universal either)
- keep this in mind in terms of rmon probe security

ip address authentication

- arp spoofing
  - bart is root on box X (129.1.2.1)
  - bart knows you are user U on box Z, same subnet (129.1.2.2)
  - # ifconfig eth0 129.1.2.2 (i.e., bart's machine claims to be you)
- bart can then setup an account as you on his box (same userid), and rlogin in sans password on some other box with .rhosts

viruses and worms

- executable programs written by others and accidentally executed by you
  - are not a good idea for you
- melissa - one of latest Usoft macro viruses
  - basic program
- java applet security problems
  - interpreted code
- so what would be safe actually?
- how do you know HTML cookies are safe?

all those nasty C programs ...

- “if we didn’t use C, we wouldn’t have those problems ...”
- this explains the problems Java has with a theoretical type-safe virtual machine
- some bottom lines:
  - don’t do interpreted code
    - » you don’t execute “destroy.exe” found on a BBS, do you?
  - don’t mail .doc files ... (ASCII remains unexecutable)

tcp sequence # spoofing attacks

- Bellovin and others have pointed out that a TCP session may be hijacked
- e.g., see Phrack article Vol7/Issue48/File14
- **ip spoofing** - use of false ip src address
- **ip splicing** - injection of packets with guessed/correct tcp sequence # into stream in order to attack remote system

sequence-attack paradigm

- assume two hosts, target A, and trusted confederate B (e.g., assume B has ./rhosts access on A
- assume attacking host X
- the attacker on X has root access and wants root access on A
- X must query A to learn a reasonable sequence space
- X then must shut B up say with TCP syn attack or other means

sequence attack paradigm, cont

- X must then simultaneously attack A with a guessed sequence #, (send SYN, SYN-ACK, won't get initial ACK)
- and X must use B's ip address (ip spoof)
- if rlogin/rsh etc., open X may gain root access to A
  - plop down backdoor files
- what are defenses against this?

more attacks

- previous are representative ... but not all
- e.g., DNS attacks known
  - return false binding or overload (add false) binding to reply
  - cause cache to contain (victim DNS, man-in-the-middle IP)
  - **man-in-the-middle** hence possible
  - map site A to competitor ... or what?!

lessons

- hosts and routers exposed to the Internet have to be "patched" or have binaries kept current
- "the fundamental rule/s apply": (partly from Chapman/Zwicky":
  - **least privilege** - don't give it away unless you have to (e.g., daemons should have own UID)
  - **defense in depth** - router/host/protocol
    - » you have a great firewall ... use ssh too, ipfw on unix host

lessons, cont

- **choke point** - force the bad guys to come in the gate and sign in
- **weakest link** - security is as good as the worst security component
  - it's not just bugs, bugs, bugs, but the worst password in your group, the hole in the firewall ...
  - the ones you don't know about can kill you
  - every virtual machine MAY have a fatal security flaw you don't know about (consider halting problem ...)
  - hacker rule: if manual says don't do X, do X

lessons, cont:

- **fail-safe**: if component fails (goes down), it should fail so that it denies access, not permits access (does not **fail-open**)
- **consider default deny vs. default permit** as a rather crucial decision (least privilege again)
- **no exceptions**
  - see weakest link
- **link security systems where possible**
  - show two picture IDs ..., permit IPSEC to bastion host X, or you only use ssh to get to bastion host X

and

- simplicity - KISS
  - security is often anti-user convenience though
- passwords are basic - improve authentication
- openness (and common sense)
  - it is widely believed that secure crypto algorithms are not secure unless open and publicly reviewed (and subject to test of time)

meta- lesson: fuzz revisited

- consider Microsoft or Sun or anyone who sells only binaries but not open src; i.e.,
- **OPEN CODE IS A MUST**
  - **code for tcp/ip stacks should be available ...**
  - **all security code should be available for review**
- ftp://grilled.cs.wisc.edu/technical_papers/fuzz-revisited.ps.Z
- "An Empirical Study of the Reliability of UNIX Utilities", Miller, Fredrickson, and So, ACM Communications, Volume 33, issue 12, pp. 32-44

fuzz, and not just lint

- Barton Miller subjected UNIX utilities to random input (“line noise”) and learned:
- too many crashes due to random inputs
  - EOF in middle of input line
- vendor programs had failure rates at 15-43%
- bugs didn’t change from 1990 to 1995
  - although failure ids/test results made available

and here's the clincher

- two lowest failure rates:
- 1. Free Software Foundation's GNU utilities (7%)
  - FSF does not allow fixed-length buffers
- 2. Linux utilities (%9)
  - lots of GNU stuff in here as Stallman is quick to point out
- See Cathedral and Bazaar url (on class home page)

point to ponder

- software will have bugs ... and still be useable
- security takes one and only one (possibly unknown up until now) fatal flaw
  - said flaw needs to be fixed ASAP -
  - not stone-walled
  - not put off until the next release

left unvisited

- security policies
- host os security
  - prevention of holes in multi-user system
  - logging/auditing (we'll briefly touch on that later)
  - passwords all over again
  - physical host security
- post attack measures
  - backups and forensics