Worms

- What are they?
- How do they spread?
- What can be done about them?

Worms in Science Fiction

"Let me put it another way. You have a computer with an auto-dial phone link. You put the VIRUS program into it and it starts dialing phone numbers at random until it connects to another computer with an auto-dial. The VIRUS program then *injects* itself into the new computer. Or rather, it reprograms the new computer with a VIRUS program of its own and erases itself from the first computer. The second machine then begins to dial phone numbers at random until it connects with a third machine. . . .

When Harlie Was One, David Gerrold, 1972

Viruses

■ "Infected" program (or floppy)

■ When program is executed, it performs its normal function

■ It also infects some other programs

■ It may carry an extra "payload" that performs other functions

Worms

- Similar to viruses, but they spread *between* machines
- Some are fully automatic; some require manual intervention to spread
- Some exploit bugs; others use social engineering
- Name from John Brunner's *The Shockwave Rider*, 1975
- Viruses are covered in more detail in COMS W4187

Classic Worms

Early Worms

■ IBM Christmas Card "Virus", December 1987

■ Morris Internet Worm, November 1988

■ Most worms since then have emulated one or both of those

Christmas Card Virus

- Infected EARN, BITNET, and IBM's VNET
- (Old, pre-TCP/IP network for IBM mainframes)
- Spread by *social engineering*

What Users Saw

```
            X
           X X
          X X X
         X X X X
        X X X X X
       X X X X X X
      X X X X X X X
           X
           X
           X
```

A very happy Christmas and my best wishes for the
next year. Let this run and enjoy yourself.
Browsing this file is no fun at all.
Just type Christmas.

What Happened

- A file transfer mechanism (not quite email, though it could have been) delivered a short script to users
- It was written in REXX, a shell script-like language for IBM's VM/CMS system
- The script displayed the Christmas card; it also looked through the (equivalent of) the user's email alias file and the file transfer log
- It transmitted a copy of itself to any usernames it found
- People trusted it, because it was coming from a regular correspondent. . .

Essential Elements

- Self-replicating executable
- Apparently from a trusted source
- Request that the recipient execute the program
- Using the email alias file to find new victims
- These characterize most current email worms

The Damage

- The worm itself wasn't malicious
- However, it had exponential growth patterns
- It clogged servers, communication paths, spool directories, etc.
- In other words, it was an unintentional denial of service attack

The Internet Worm

- Got much more mainstream publicity
- Estimated to have taken out 6000 hosts — 10% of the Internet
- Arguably, the first time the Internet made the evening news

Characteristics

- Much more sophisticated
- Exploited buggy code — spread *without* human intervention
- Exploited trust patterns among computers
- Multiple attack vectors
- Multiple architectures (Vax and Sun 3)
- Intended to demonstrate the insecurity of the Internet. . .

Attack Vectors

- Back door in `sendmail`
- Buffer overflow in `fingerd`
- Password-guessing
- Pre-authenticated login via `rsh`

Sendmail Back Door

- The author of `sendmail` wanted continued access to the production version installed at Berkeley
- The system administrator wouldn't permit this
- He put a deliberate back door into `sendmail`, to give himself continued access
- Production systems shipped with this option enabled. . .

Buffer Overflow

- The `finger` daemon call `gets()`, a now-deprecated library routine
- Unlike `fgets()`, there was no buffer length parameter
- By sending a long-enough string over the network as input, the attacking program

  1. Injected some assembler-language code, and
  2. Overwrote the return address in the stack frame so that `gets()` branched to that code instead of back to the caller

Buffer Overflows Shouldn't Happen!

"The first principle was security: ...A consequence of this principle is that every occurrence of every subscript of every subscripted variable was on every occasion checked at run time against both the upper and the lower declared bounds of the array. ...I note with fear and horror that even in 1980, language designers and users have not learned this lesson. In any respectable branch of engineering, failure to observe such elementary precautions would have long been against the law."

Turing Award Lecture, C.A.R. Hoare

Password Guessing

- It looked up a list of usernames in the password file
- It used easy transformations of the login name and the user's name, plus a dictionary of common passwords
- Ironic note: the author of the worm, Robert T. Morris, drew upon a technique first described by his father, Robert H. Morris. . .

Pre-Authenticated Login

- Exploit trust patterns: `/etc/hosts.equiv` and per-user `.rhosts` files list trusted machines
- If machine A trusts machine B (if only for a particular user), machine B usually trusts machine A
- This provided two things: an infection path and a list of other machines to attack

Spread Patterns

■ It looked at a variety of sources to find other machines to attack:

   ◆ `rsh/rlogin` trust sources
   ◆ Machines listed in `.forward` files

■ Routers (in 1988, most routers were general-purpose computers)

■ Randomly-generated addresses on neighboring nets

Hiding

- The worm used a variety of techniques to hide
- It was named `sh`
- It forked frequently, to change processID
- It unlinked its own executable
- Text strings were (lightly) encrypted

Essential Elements

■ Self-spreading, via buggy code

■ Self-spreading, via trust patterns

■ Combination of directed and random targets
for next attack

■ Stealth characteristics

Modern Worms

Modern Worms

■ Most resemble either the Christmas card worm or the Internet worm

■ Today's email worms try to trick the user with tempting `Subject:` lines — nude pictures, software "updates", etc.

■ A notable one: "Osama bin Laden Captured", with an attached "video"

■ Some pose as anti-virus software updates. . .

■ Can get through many firewalls

Stealthiness

- Deceptive filenames for the attachments
- Add a phony extension before the real one: `kournikova.jpg.exe`
- Hide in a `.zip` file
- Hide in an encrypted `.zip` file, with the password in the body of the email
- Many strategies for hiding on hosts, including strange filenames, tinkering with the registry, etc.

Trust Patterns

- Preferentially attack within the same network — may be on the inside of a firewall
- Exploit shared disks
- Mass-mailing worms rely on apparent trustworthy source

Spreading Via Buggy Code

- Exploit many different (Windows) bugs
- Can spread much more quickly
- Slammer spread about as far is it could in just 15 minutes, and clogged much of the Internet

The Slammer Worm

- Exploited a bug in Microsoft's SQL server
- Used UDP, not TCP — a single 376-byte packet to UDP port 1434 could infect a machine!
- Use of UDP instead of TCP let it spread much faster — one packet, from a forged source address, instead of a three-way handshake, payload transmission, and a three-packet `close()` sequence
- No direct damage, but it clogged network links very quickly

The Welchi Worm

- Attempted to do good
- Used the same Microsoft RPC bug as the Nachi worm
- Removes certain other worm infections
- Installs Microsoft's fix for the hole
- Deletes itself after January 1, 2004

Was it a Good Idea?

- No — unauthorized
- No — not well-tested
- No — generates a lot of network traffic, more than the worm it was trying to cure

Worm Effects

- Seriously clogged networks
- Slammer affected some ATM and air traffic control networks
- CSX Railroad's signaling network was affected

Sobig.F

- Part of a family of worms
- High-quality code
- Primary purpose: spamming
- Turned infected machines into spambots
- Marked the turning point in worm design — now, it's done for profit instead of fun

Worm Spread Patterns

Spread Patterns

- Worms tend to exhibit *exponential growth* patterns
- They start slow, but get very big quite quickly
- Equation: $y = e^{kt}$, where t is time
- If k is small, it spreads more slowly — but it still grows

Exponential Growth

There's a Ceiling

- Worms run out of vulnerable hosts
- Doesn't matter much if a machine is infected twice (and worms often prevent that)
- Actual graph is a *logistic curve*: $y = a\frac{1+me^{-t/\tau}}{1+ne^{-t/\tau}}$

Warhol Worms

■ *"In the future everyone will be famous for 15 minutes"* —Andy Warhol, 1960s

■ As we've seen, it's possible for a worm to spread *very* quickly

■ (Note that this paper was published before Slammer hit)

■ Suppose it had a malicious payload.

■ It could do tremendous damage before any human had a chance to react

Scanning Patterns

- Older worms used clumsy random scans
- New ones use different probabilities for local versus remote networks
- Often have built-in lists of useful IP address ranges
- Some have exclusion lists for known honeynets

Detection and Prevention

Detecting Worms

■ How are worms detected?

■ Initially, by honeypots and by people sending samples of suspicious code to anti-virus comapanies

■ A/V companies build worm *signatures*

■ Signatures are byte patterns that match that file

■ Every new worm or worm variant needs its own signature, which is why anti-virus scanners need weekly updates

Encrypted and Polymorphic Worms

- Some worms generate variants of themselves
- Others encrypt much of themselves
- Anti-virus programs look for complex patterns and/or decryption code

■ Application firewalls can do anti-worm scanning

■ Good packet filters can deflect many buggy code attacks

■ But — some worms spread from web servers to web browsers, which then go on to attack other web servers

More Science Fiction

"It's fun to think about, but it was hell to get out of the system. The guy who wrote it had a few little extra goodies tacked onto it – well, I won't go into any detail. I'll just tell you that he also wrote a second program, only this one would cost you – it was called VACCINE.

When Harlie Was One, David Gerrold, 1972