Malware

Malware is software that intentionally behaves against a user's wishes

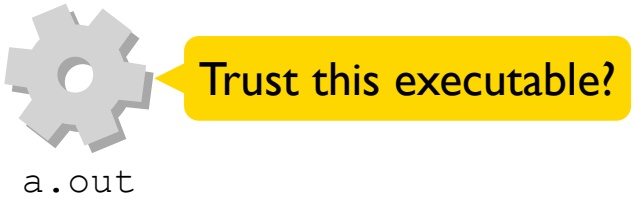
 **Trojan horse** — actual behavior different than advertised

 **Virus** — uses a host program to run and propagate itself

 **Worm** — creates own processes to replicate itself

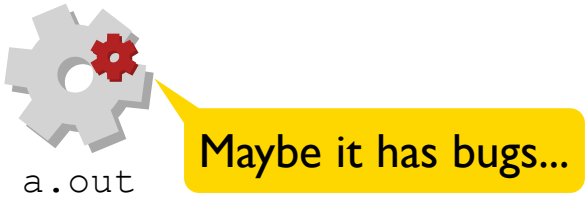
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture

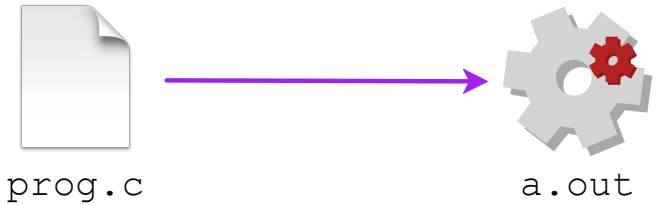
`a.out`

Maybe it has bugs...

Maybe it even intentionally behaves against a user's wishes

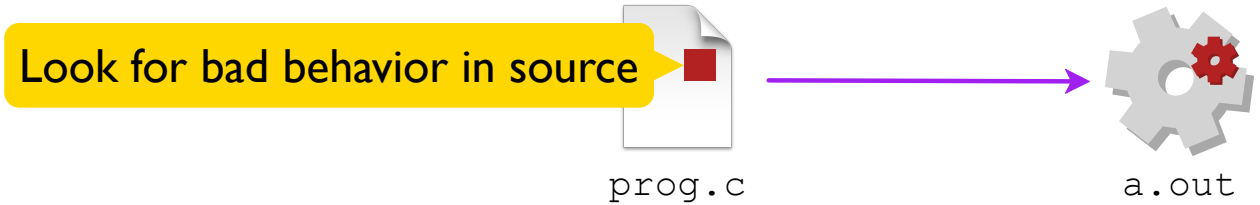
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



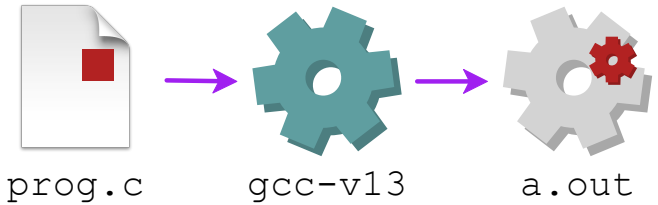
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



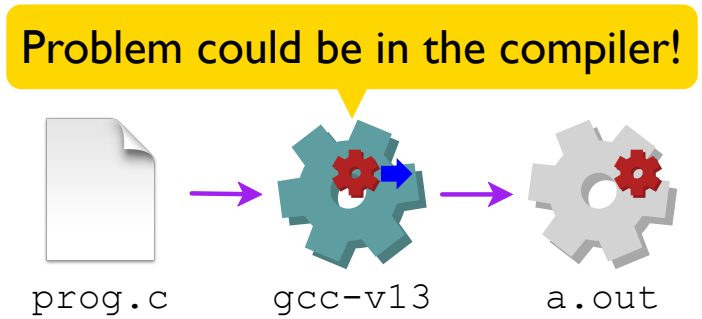
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



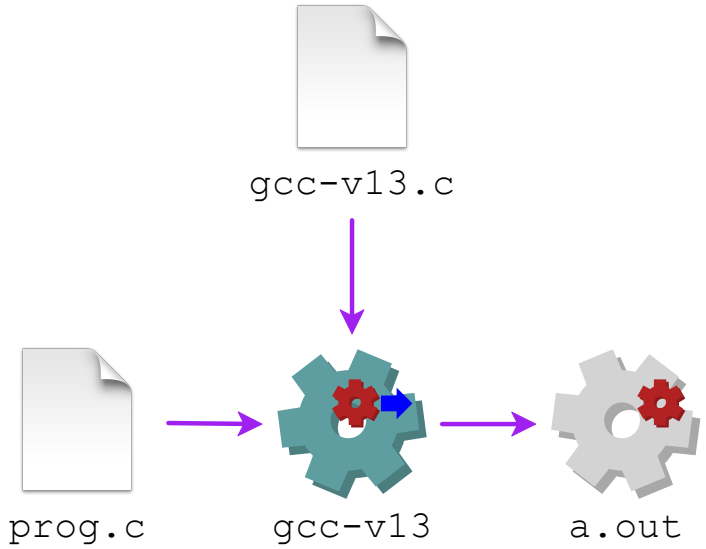
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



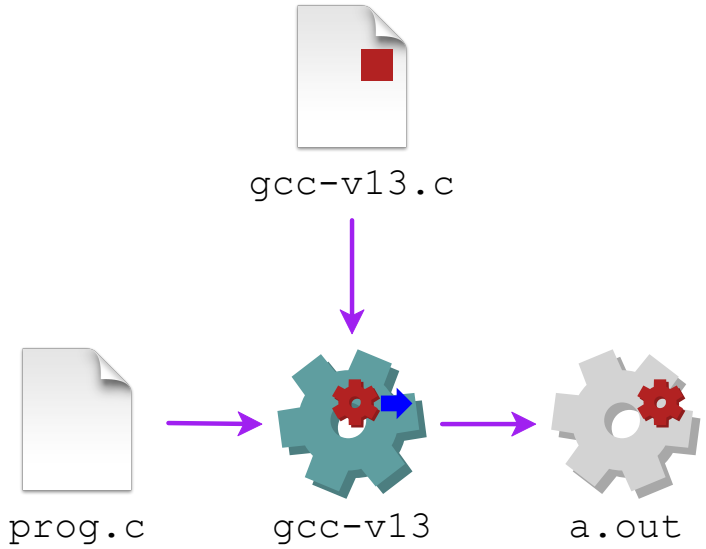
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



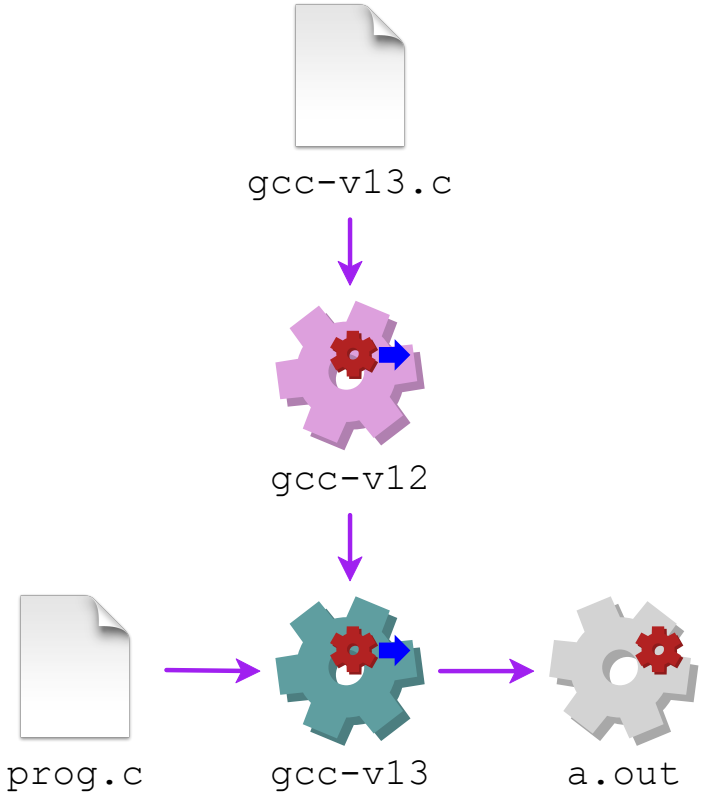
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



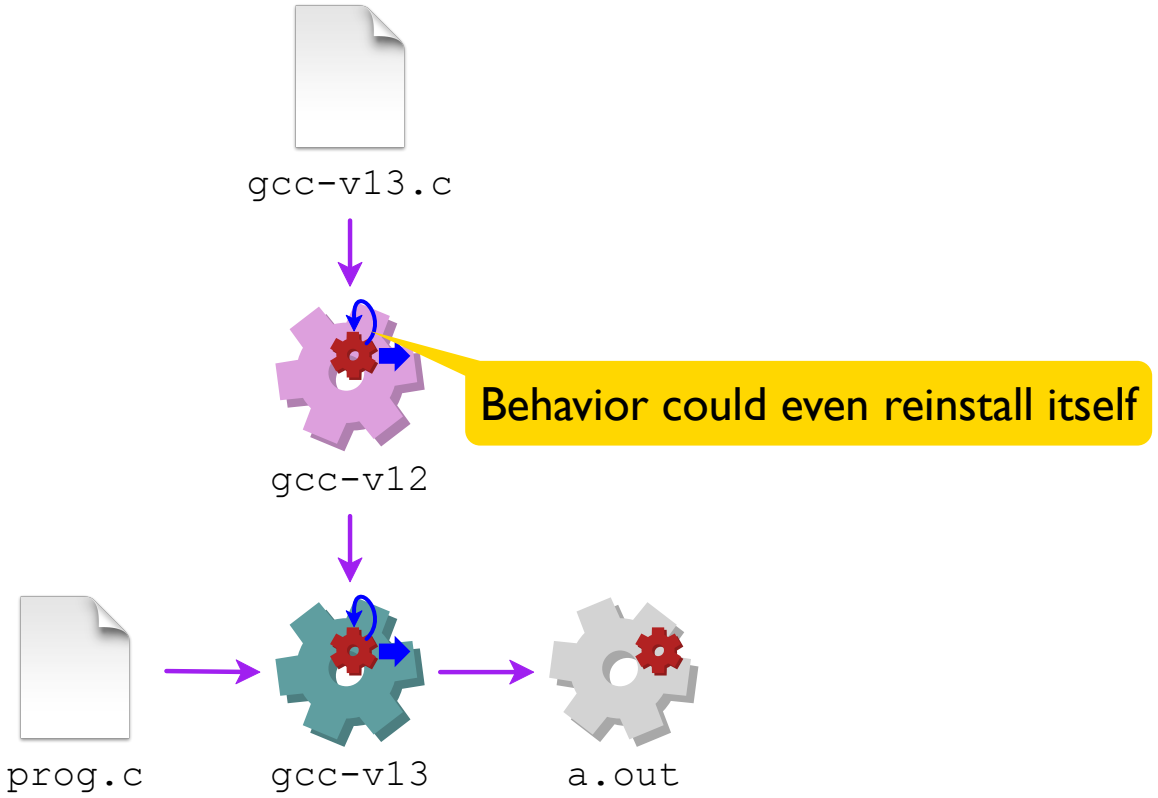
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



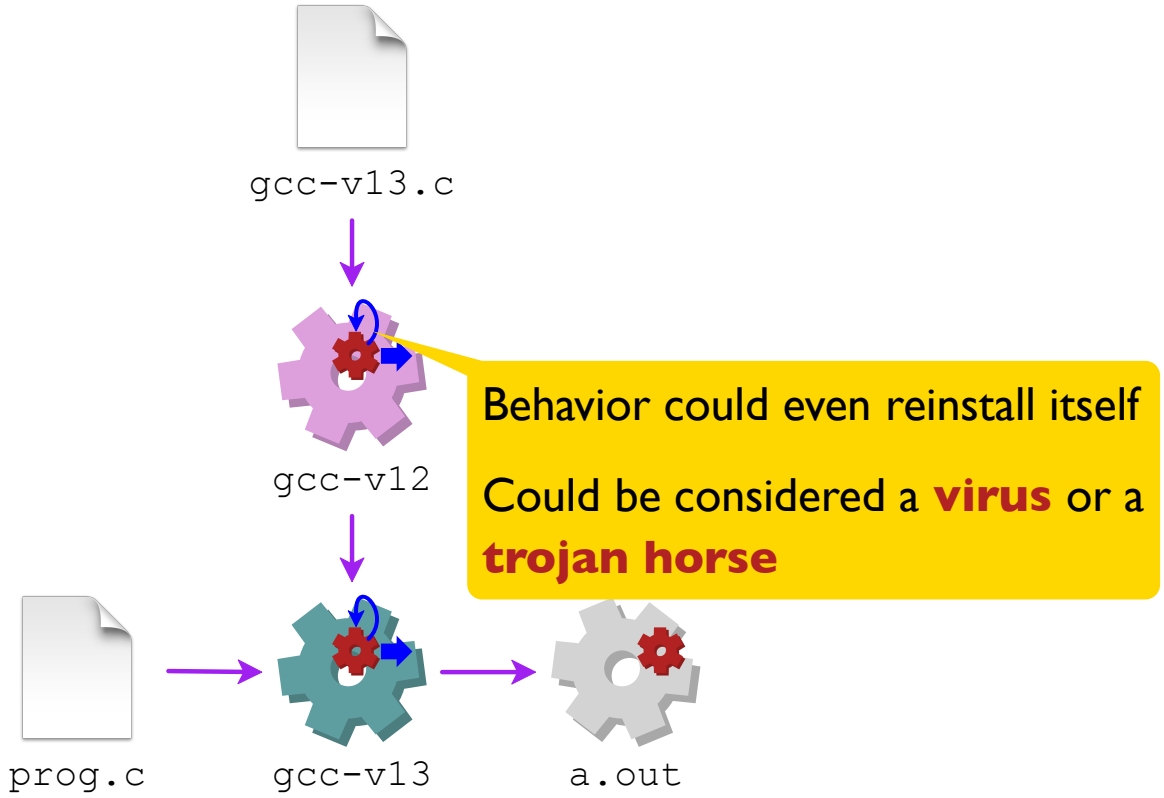
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



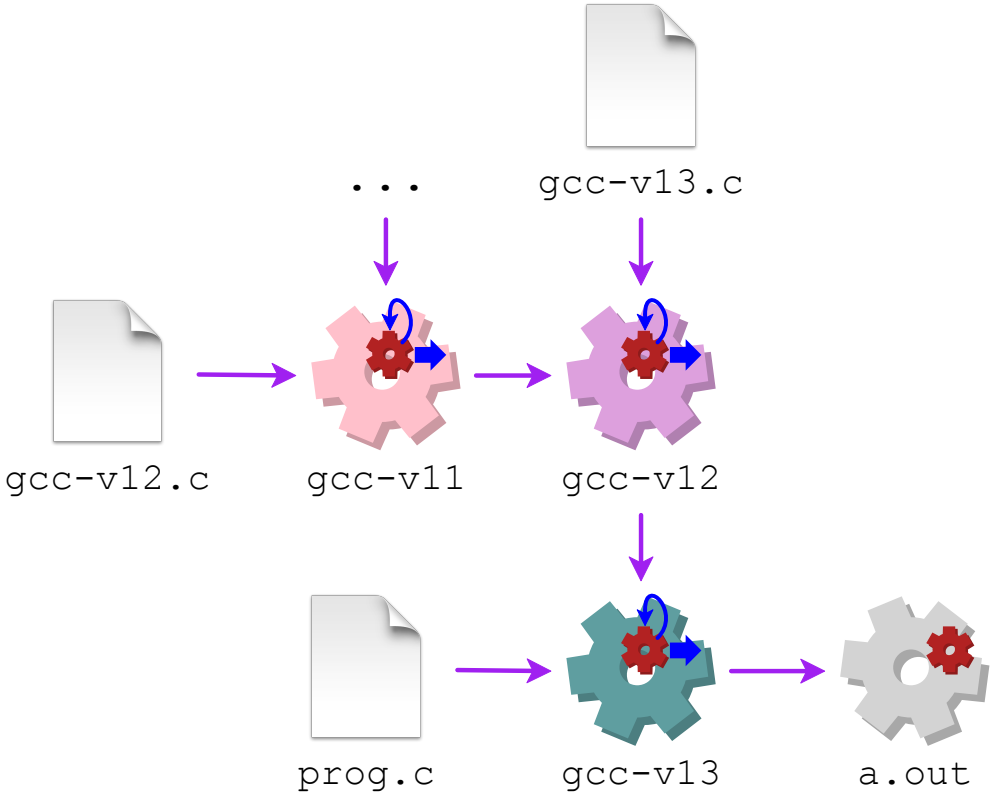
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



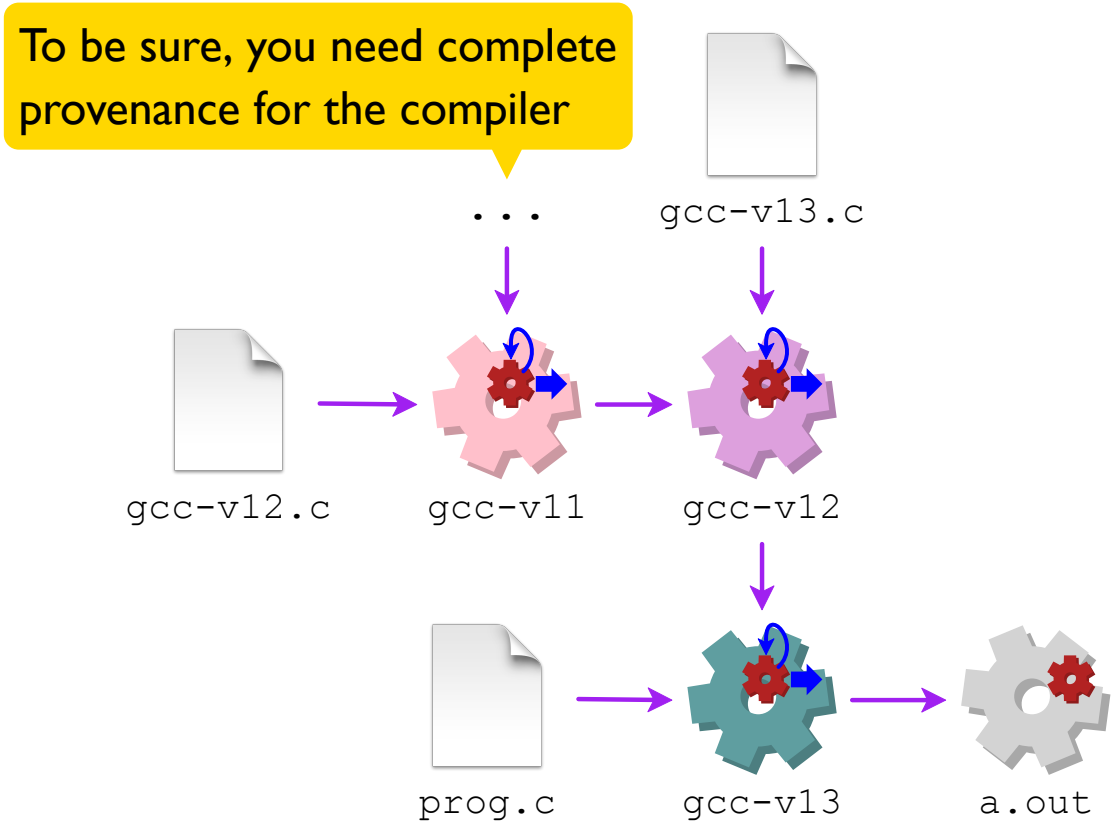
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



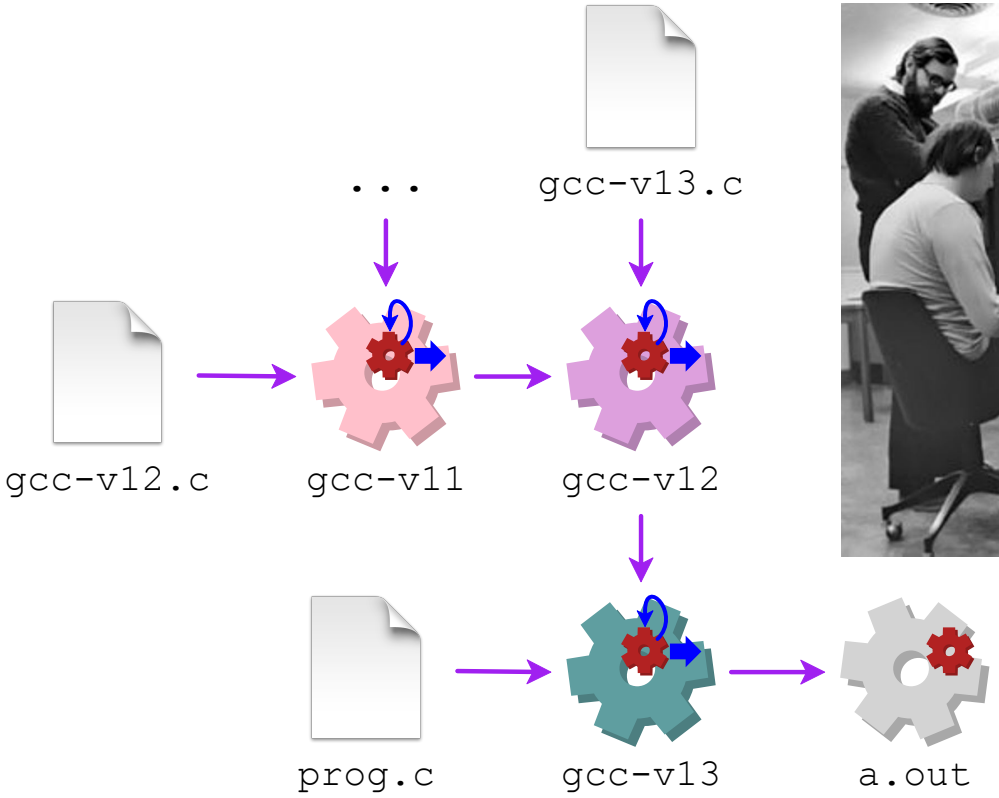
Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award Lecture



Context of Thompson's speech:
new laws to clarify that virtual spaces are private property

Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**

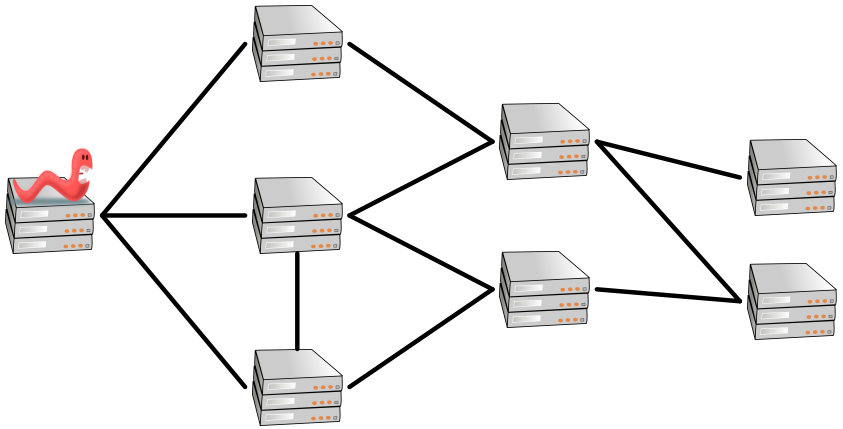
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



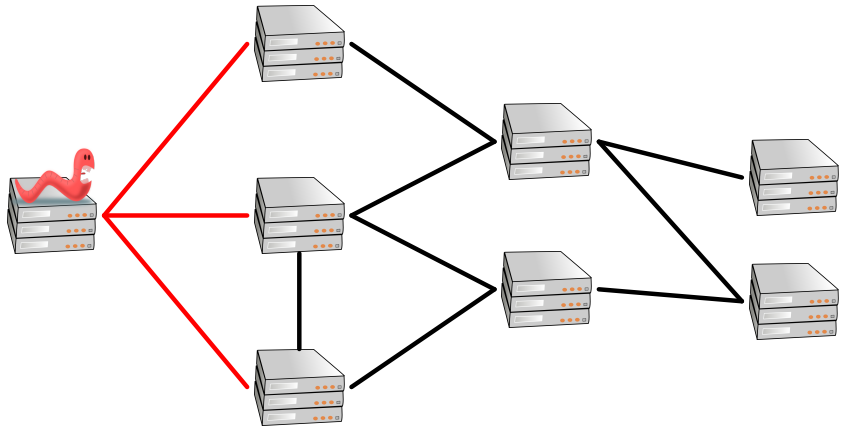
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



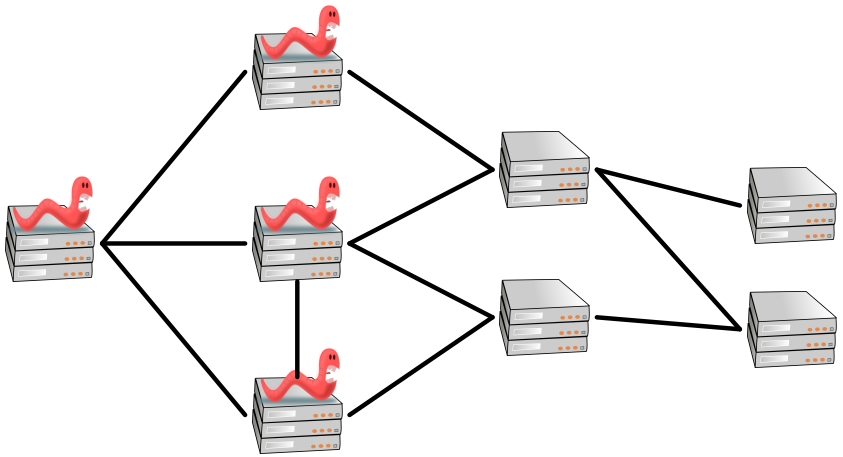
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



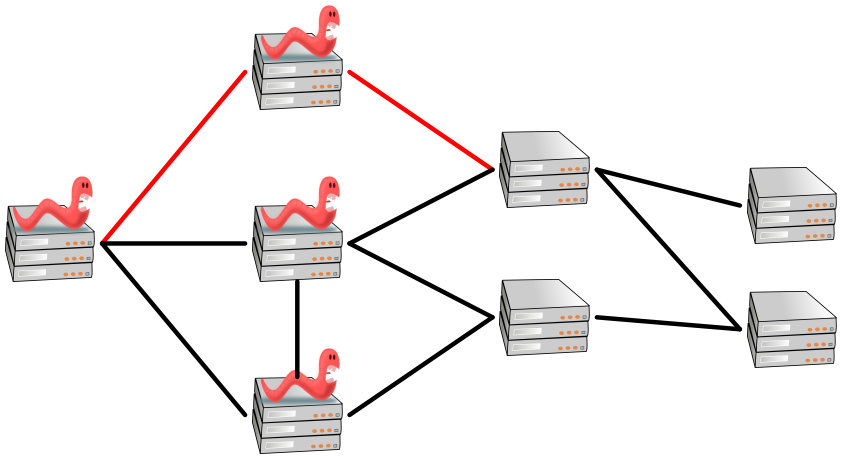
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



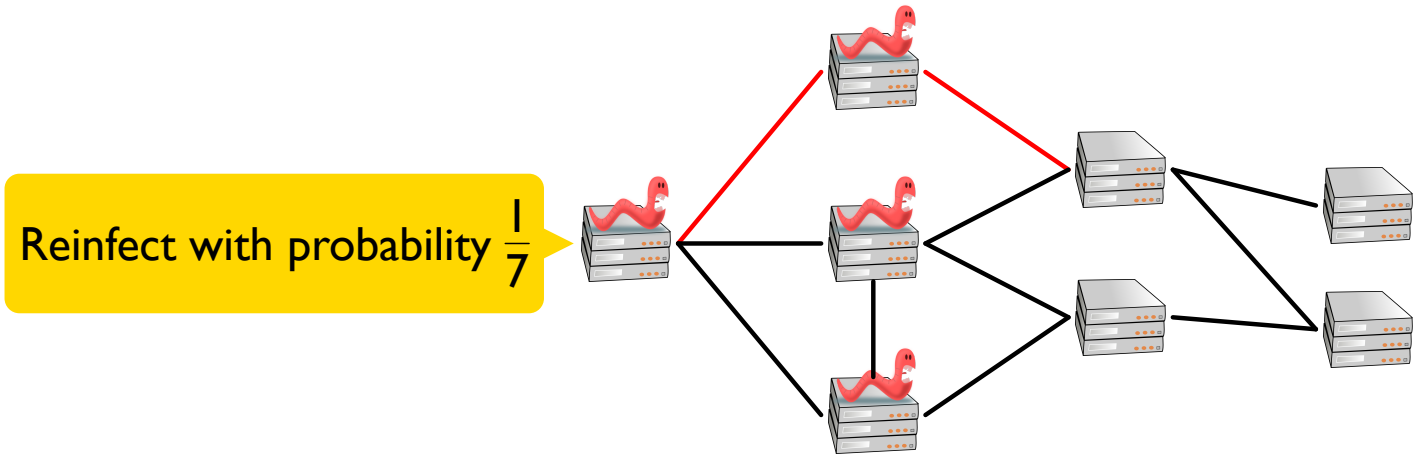
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



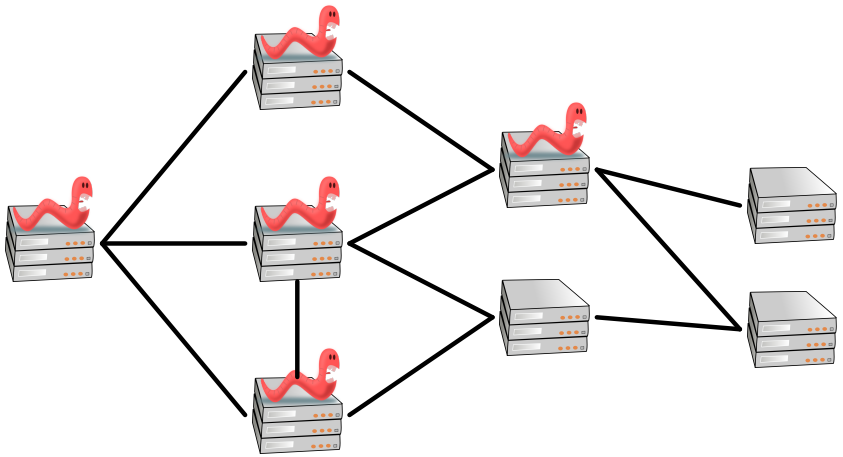
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



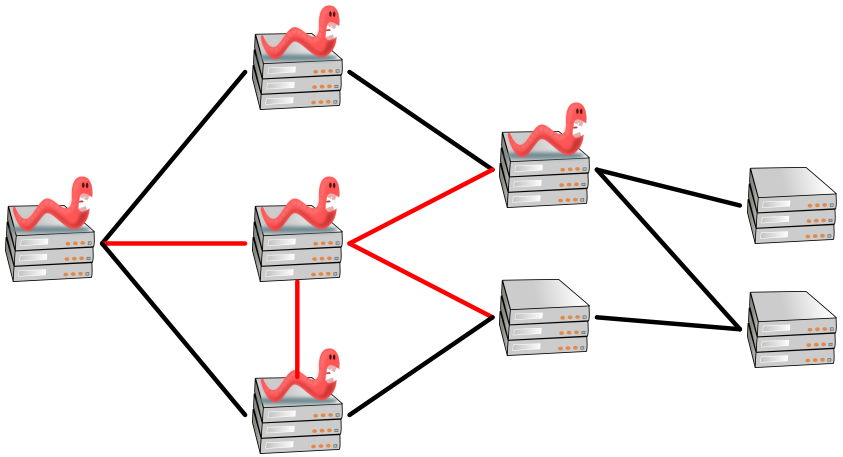
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



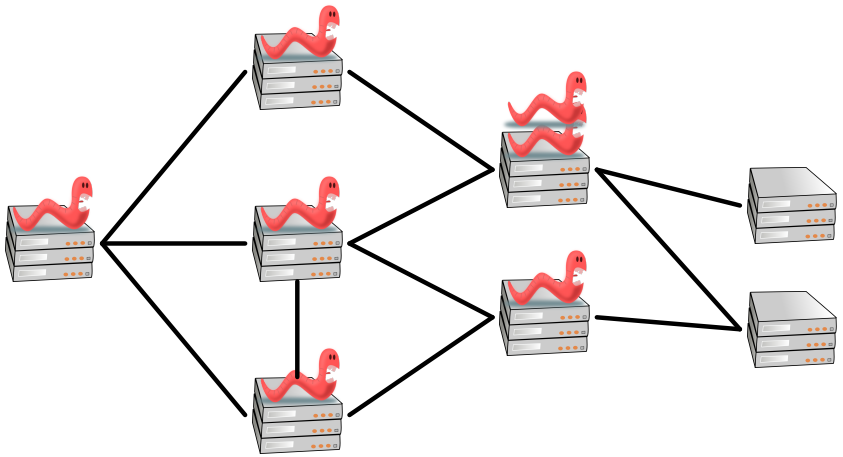
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



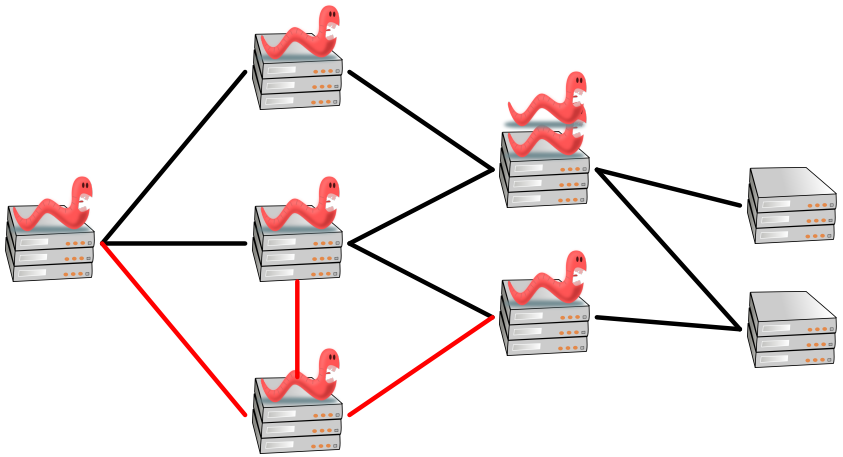
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



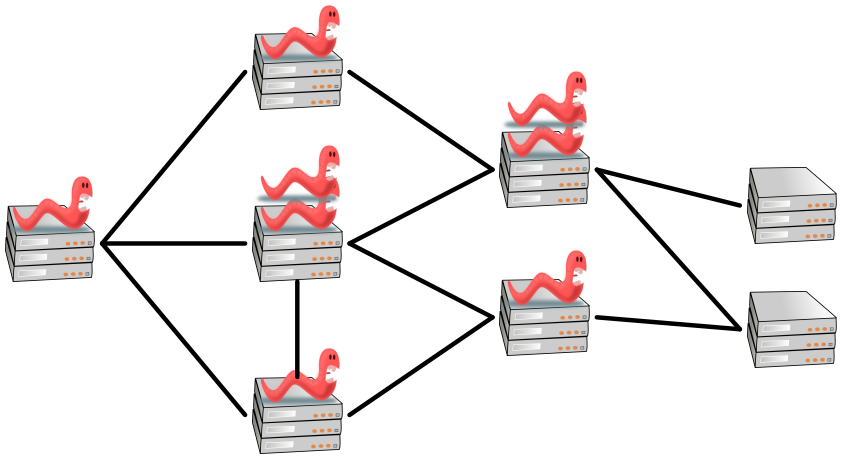
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



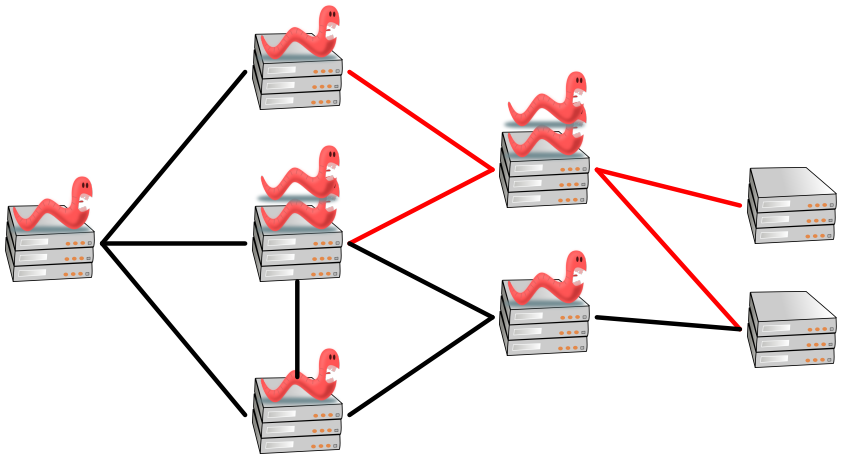
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



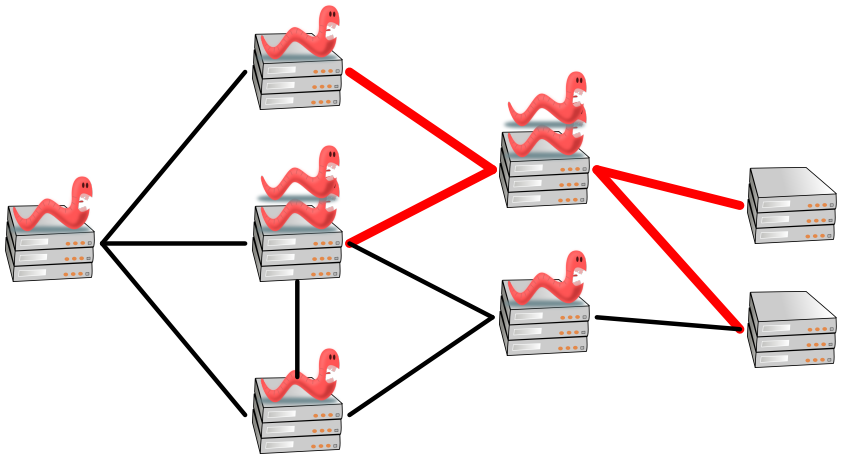
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



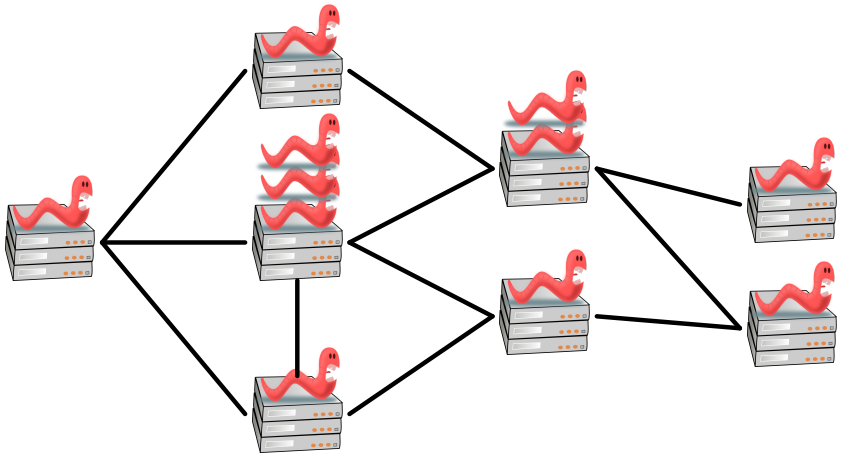
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



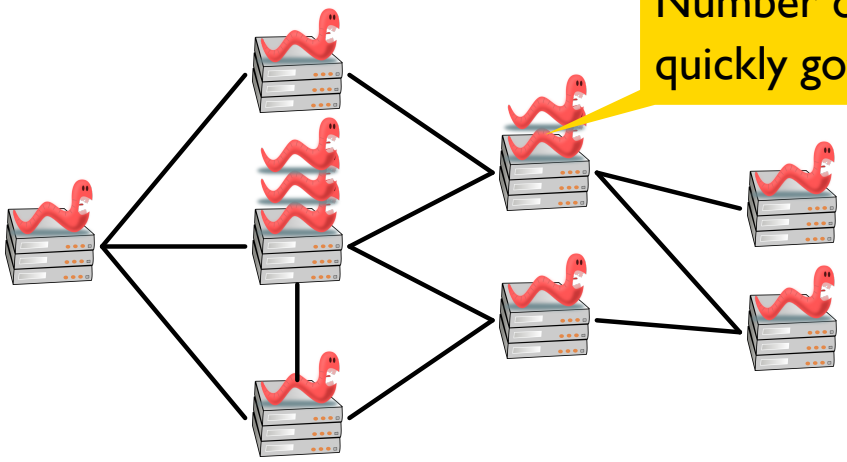
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



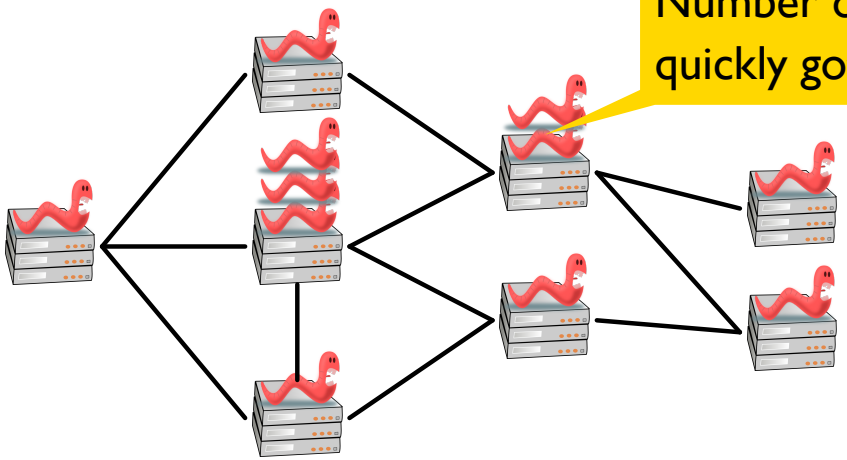
Early Internet Worm

Robert T. Morris, 1988

Grad student at Cornell tests internet security:

- exploit debug-mode hole in `sendmail`
- exploit buffer-overflow bug in `fingerd`
- exploit weak passwords

with botnet-like propagation, making it a **worm**



Number of worms on each machine quickly got out of hand!

Fallout: felony for the worm's author

Defense Gone Wrong

Sony BMG, 2005

Sony's problem: Easy to copy digital music files from a CD on Windows

Defense Gone Wrong

Sony BMG, 2005

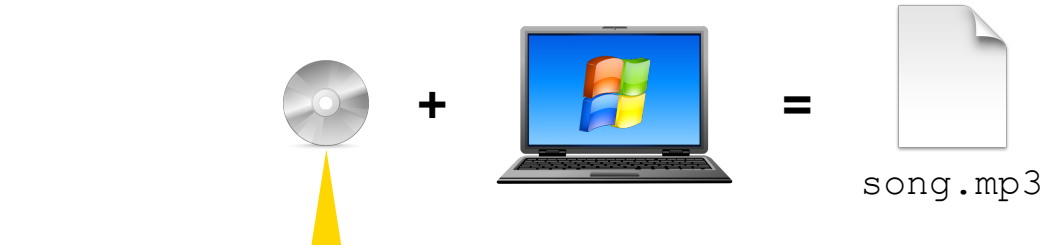
Sony's problem: Easy to copy digital music files from a CD on Windows

Some CDs have music,
others have games,
...

Defense Gone Wrong

Sony BMG, 2005

Sony's problem: Easy to copy digital music files from a CD on Windows



Windows runs `autorun.exe` from CD to let it pick behavior

Defense Gone Wrong

Sony BMG, 2005

Sony's problem: Easy to copy digital music files from a CD on Windows

Sony's idea: include `autorun.exe` on music CDs, and make it patch the operating system to disable song copying

Defense Gone Wrong

Sony BMG, 2005

Sony's problem: Easy to copy digital music files from a CD on Windows

Sony's idea: include `autorun.exe` on music CDs, and make it patch the operating system to disable song copying

A trojan horse that acts like a **rootkit**

Defense Gone Wrong

Sony BMG, 2005

Sony's problem: Easy to copy digital music files from a CD on Windows

Sony's idea: include `autorun.exe` on music CDs, and make it patch the operating system to disable song copying

Also, to thwart uninstall attempts

Defense Gone Wrong

Sony BMG, 2005

Sony's problem: Easy to copy digital music files from a CD on Windows

Sony's idea: include `autorun.exe` on music CDs, and make it patch the operating system to disable song copying

Also, to thwart uninstall attempts

Hide file names that start with `$sys$`

Defense Gone Wrong

Sony BMG, 2005

Sony's problem: Easy to copy digital music files from a CD on Windows

Sony's idea: include `autorun.exe` on music CDs, and make it patch the operating system to disable song copying

Also, to thwart uninstall attempts

Hide file names that start with `$sys$` — which was great for *other* attackers

Defense Gone Wrong

Sony BMG, 2005

Sony's problem: Easy to copy digital music files from a CD on Windows

Sony's idea: include `autorun.exe` on music CDs, and make it patch the operating system to disable song copying

Also, to thwart uninstall attempts

Hide file names that start with `$sys$` — which was great for *other* attackers

Fallout: many lawsuits against Sony

Spy Games Gone Wrong

WannaCry, 2017

- Early 2010s: Windows network vulnerability
- Mid 2010s: NSA creates **EternalBlue** exploit
  does not alert Microsoft
- sometime before March 2017: EternalBlue is stolen
- March 2017: NSA alerts Microsoft, bug is patched
- April 2017: **WannaCry** worm attacks unpatched installations

Spy Games Gone Wrong

WannaCry, 2017

WannaCry operation:

- encrypt many files to new `.wncry` files
- delete unencrypted originals
- get user to pay ransom in Bitcoin

- infect other machines using EternalBlue exploit

Spy Games Gone Wrong

WannaCry, 2017

WannaCry operation:

- encrypt many files to new `.wncry` files
- delete unencrypted originals
- get user to pay ransom in Bitcoin
- check `iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com`
- infect other machines using EternalBlue exploit

Spy Games Gone Wrong

WannaCry, 2017

WannaCry operation:

- encrypt many files to new `.wncry` files
- delete unencrypted originals
- get user to pay ransom in Bitcoin
- check `iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com`
- infect other machines using EternalBlue exploit

Security researcher registered this domain name, accidentally stopping the worm's spread!

Malware

All of those examples are **malware**: software that intentionally behaves against a user's wishes

Who

- **Script kiddies** — use published malware
- **Skilled criminals** — mostly modify existing malware
- **State-sponsored groups** — develop new malware

Malware

All of those examples are **malware**: software that intentionally behaves against a user's wishes

Why

- Information and identity theft
- Encrypt/steal data for ransom
- Inject/falsify data
- Break things controlled by software
- Steal resources (e.g., for spamming)

Kinds of Malware Delivery

 **Trojan horse** — actual behavior different than advertised

example: Sony copy protection

 **Virus** — uses a host program to run and propagate itself

examples: VisualBasic scripts in a Word document,
malformed PDF that injects code into viewer

 **Worm** — creates own processes to replicate itself

examples: Mirai, WannaCry

Malware Payload

Payload is the part of malware that performs the bad action

as delivered by a virus, worm, or trojan horse

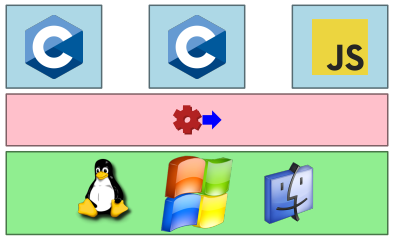
Example payloads:

- copying data
- deleting files
- encryping files
- creating backdoor access for future commands

A rootkit is normally considered a payload

Rootkits

A **rootkit** modifies the OS to intercept and change system actions



Sits between applications and the kernel, so it can control applications

e.g., log keystrokes, redirect TCP connections, disable TLS

Can change or lie to system tools to hide its own existence

e.g., change `ls` or `ps`

May change the bootloader to ensure that the rootkit stays in place

Malware Lifecycle

Before intrusion:
- network reconnaisance
- determine software versions
- gather information on users

Intrusion:
- buffer overflow
- code injection
- password guessing
- phishing / social engineering

If we can't stop malware, which of these steps might be detected?

Privilege escalation:
- access more powerful machine
- access root user

Exploitation:
- theft
- destruction
- surveilance

Access maintenance:
- turn off updates
- create accounts
- steal credentials
- install back door

Cover up:
- delete files
- erase logs

Intrusion Detection

Red flags that are relatively simple to detect:

- port scanning
- modifications to system files
- frequency of unusual system calls

More generally:

- **Anomaly-based systems** watch for unusual activity based on a model of normal activity
- **Signature-based systems** watch for specific activity based on patterns for known malware

Intrusion Detection

Red flags that are relatively simple to detect:

- port scanning
- modifications to system files
- frequency of unusual system calls

More generally:

- **Anomaly-based systems** watch for unusual activity based on a model of normal activity
- **Signature-based systems** watch for specific activity based on patterns for known malware

Effective against script-kiddie attacks

Intrusion Detection

Red flags that are relatively simple to detect:

- port scanning
- modifications to system files
- frequency of unusual system calls

More generally:

- **Anomaly-based systems** watch for unusual activity based on a model of normal activity
- **Signature-based systems** watch for specific activity based on patterns for known malware

Snort software and database is a popular choice

Package Repositories

A concern for programmers: *Can you trust that library?*

Recently in the news: attempted `xz` back door

Package repos have various mechanisms to prevent attacks and back doors

Guix package manager uses Git-like hashing to ensure whole build and dependency chains are secure

small Scheme interpreter in C
- ➡ tiny C compiler in Scheme ↖
- ➡ `tinycc` compiler
- ➡ very old `gcc`
- ➡ modern `gcc`
- ➡ ...

Goal is a small and auditable starting point

Summary

Malware does what you don't want, and on purpose

Malware delivery mechanisms: **trojan horse**, **virus**, **worm**

A **rootkit** owns a machine

Even if you write bug-free software, you have to build on complex layers that will still have bugs for the forseeable future, so you'll need security in commensurate depth