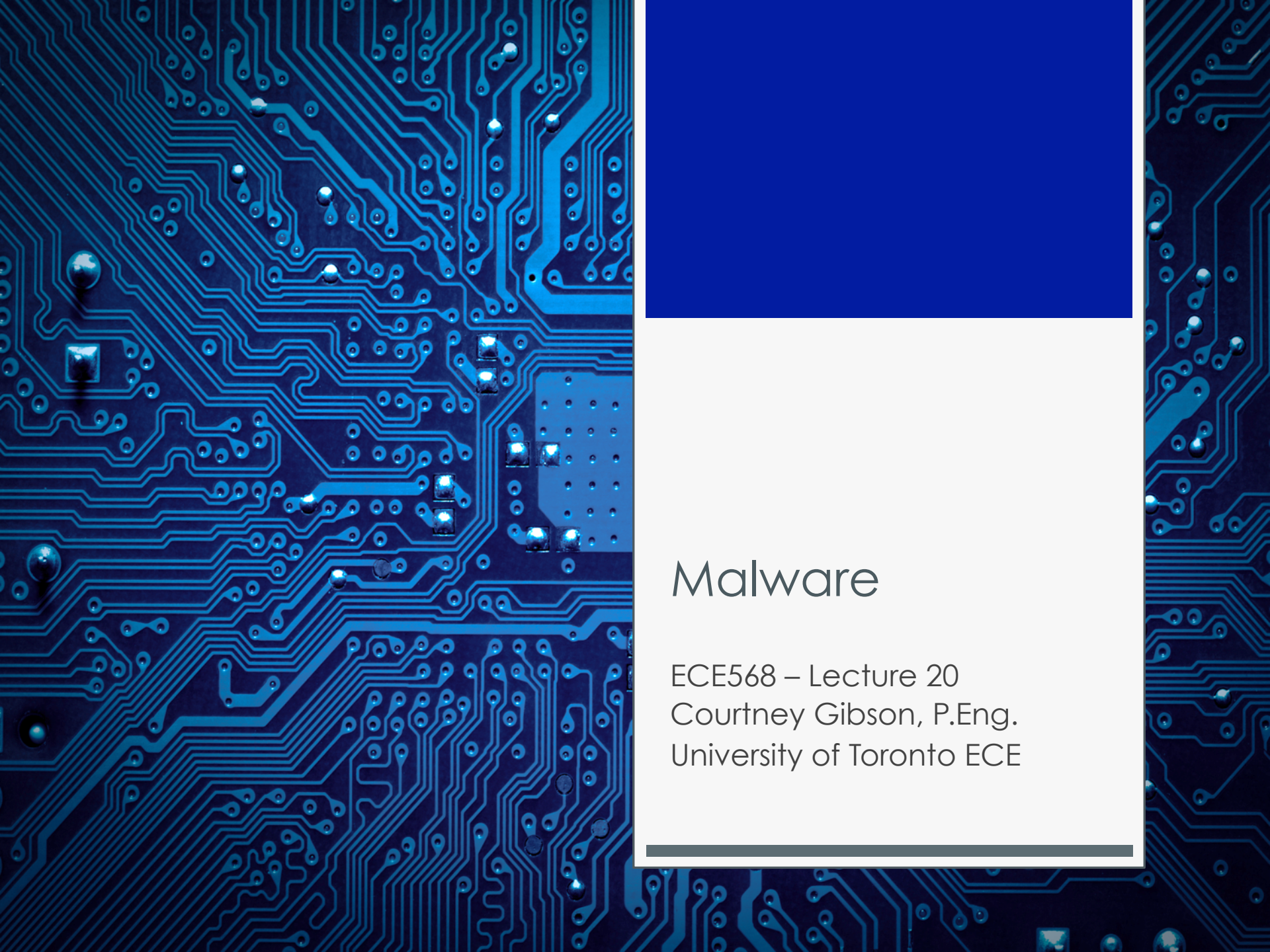


Malware

ECE568 – Lecture 20

Courtney Gibson, P.Eng.
University of Toronto ECE

Outline

Malware

Viruses

- Insertion Points
- Polymorphic Viruses
- Metamorphic Viruses

Worms

- Morris Worm
- Worm Scanning Techniques
- Worm Migration
- Botnets

Rootkits

Malware

Malware

Malware is a shortened term for **malicious software**

- Software that is hostile, intrusive or annoying, and is designed to infiltrate or damage a computer system, typically installed without the owner's informed consent

- Malware includes
  - **Virus and Worm:** malicious replicating programs
  - **Rootkit:** software designed to hide or obscure the fact that a system has been compromised
  - **Trojan:** software that appears to be desirable, but in fact performs malicious actions
  - **Backdoor:** a method that allows bypassing normal authentication procedures
  - **Spyware:** software installed surreptitiously to intercept the user's interaction with the computer

Viruses

Viruses vs. Worms

What are the differences?

- Both replicate automatically
- Both consume system resources, can be destructive

Virus	Worm
▫ Spreads secretly, makes a lot of effort to avoid detection ▫ Needs a host program to infect, is not a stand-alone program ▫ Slow spreading, often requires human help	▫ Goal is usually to spread as quickly as possible ▫ Stand-alone program, does not infect other programs ▫ Spreads automatically without human intervention

Viruses

Viruses usually have two main goals

- To hide their presence
- To replicate themselves

They work by inserting their own instructions into existing programs

- The virus is executed when the infected program is run
- On execution, the virus may **propagate** to other programs

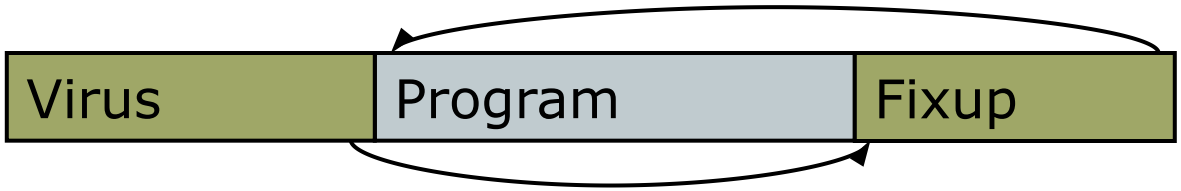
Early viruses often infected the disk **boot sector**

- A boot sector virus would load in memory when the disk is inserted and copy itself to other disks that are inserted
- Prevalent in the days when people booted off floppies and copied data between machines using floppies

Virus Insertion Points

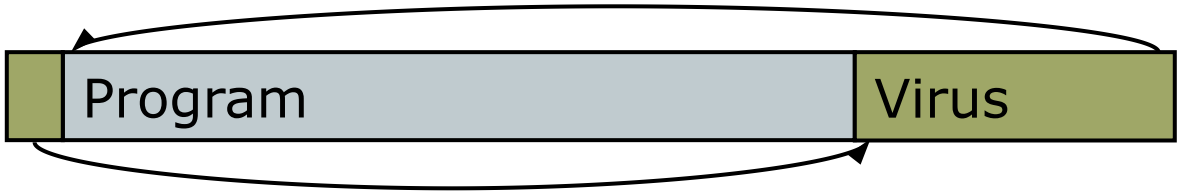
At the beginning of the program

- Virus **overwrites** the start of the program, and then inserts some fixup code to replicate the code that it overwrote
- Virus length is limited as it can't overwrite too much of the program before causing problems



At the end of the program

- Virus overwrites first instruction of program with a jump to virus code, and then jumps back to the beginning of program
- Virus only has to overwrite one instruction at the start of the program, and virus length is not limited



```
_start_main:
  goto virus_123; // normally 'goto main'

main:
  existing code;

virus_123:
  goto infect_executable;
  if (pull_trigger) do_damage else goto main;

infect_executable:
  while (some_condition) {
    file = get-random-executable-file;
    if (first-line-of-file == 'goto virus_123')
      continue;
    // propagate virus
    prepend 'goto virus_123' to file;
    append virus code to file;
  }

pull_trigger:
  return true if some condition holds;
do_damage:
  remove files…;
```

Overwrite first instruction

Existing program code

Virus code

Virus Detection

Virus scanners look for **signatures**, which are strings of bits corresponding to instructions found in known viruses

- A virus company identifies new viruses by using a honeypot farm to gather malware
- The malware is analyzed manually to build signatures
- Sometimes, automated signature generation techniques are possible
- These signatures are sent to customers periodically and the virus scanner looks for the existence of these signatures in programs on the customer's machines

Designing Signatures

Signatures should be long enough so that legitimate code is not mistakenly identified to be infected

- A false alarm or a false positive

Signatures that are too long lead to scanners missing variants of viruses

- A false negative

Often signatures are created for each variant found as well (A,B,C…), which means more signatures, but better accuracy

EICAR Anti-Virus Test

Researchers, anti-virus software vendors, system administrators and end-users all have a need to test their anti-virus software

- Not a good idea to test with real viruses on real-world systems
- EICAR standardized a common test signature that all anti-virus programs (should) recognize if they are properly working
- When put into a file, should trigger a virus alert:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Polymorphic Viruses

The easiest way to evade detection is to vary the virus payload from which a signature is generated

- Modern viruses use **polymorphism**, in which a small decryption engine decrypts the rest of the virus

- Virus uses a simple encryption scheme to defeat a signature scan (usually XOR)
  - Encryption key is changed when virus propagates to a new file, so the encrypted body is never the same
- The decryption engine is constant, but it is short and simple, making it hard to build a signature

Defeating Polymorphic Viruses

To detect polymorphic viruses, advanced virus scanners will run files they suspect are infected inside an emulator

- As the decryptor is run, the virus body will be decrypted
- A signature scanner will then be able to detect the virus body because the decrypted code does not change
- Since the scanner assumes the virus will try to run early in the program's execution, the scanner only needs to run for a short time
- If no signature matches after some time, the virus scanner declares the file clean

Metamorphic Viruses

The most advanced viruses today are using a more sophisticated technique called **metamorphism**

- Metamorphic viruses change their code on every infection by rewriting themselves
  - How is this different from polymorphic viruses?

Typical code changes include:

- Changing register allocations
- Using equivalent instruction sequences
- Changing the order of blocks of code
- Some also integrate themselves into different portions of the infected program, and not just at the beginning
  - They may not be always executed, slowing the infection rate
  - However, it makes it harder to detect their presence

Detecting Metamorphic Viruses

Detection is very difficult

- One possibility is to run in an emulator and then look for sequences of executed instructions that indicate a virus
- Often viruses leave **markers** in infected files so they know not to infect them again, so it may be possible to look for these markers

- All methods so far have problems and are not perfect
- Anti-virus companies are not releasing much info on how to detect these viruses: seen as a competitive advantage

Worms

Worms

Worms spread automatically by identifying and exploiting vulnerabilities in hosts

- After finding a new vulnerable machine, the worm installs itself on the machine and searches for another machine
- Worms can spread very fast because they require no human intervention, computers have become interconnected and networks have become fast

Example: the Slammer worm infected 90% of vulnerable hosts worldwide in under 10 min

- Eventually Slammer limited its own spread because existing infected machines were scanning so aggressively that they were starving out other infection attempts
- Full analysis of Slammer on the course web site

Case Study: Morris Worm

First worm, released in 1988

- Infected 6000 machines, severely disrupting Internet
- Damage was estimated to be $10M-$100M
- Most worms today operate in much the same way

Morris Worm operation:

- The worm consisted of two parts, a main **server** program, and a bootstrap or **vector** program
- The server looked for vulnerable remote target machines and tried to exploit a vulnerability on them
- If successful, it created a shell on the target, uploaded the vector (a short C program), compiled it on the target and then ran the vector program
- The vector downloaded the rest of the worm from the server, and started the server on the newly infected host

Case Study: Morris Worm

The Morris worm exploited four vulnerabilities:

- Vulnerable versions of fingerd program had a buffer overflow
- sendmail on many systems had been compiled with a DEBUG option. By connecting to the sendmail port and sending the string "DEBUG", the attacker received a root shell
- The worm would try popular passwords, and at the time password hashes were stored in `/etc/passwd` which is readable by any user
- The worm would try to connect to hosts in `/etc/hosts.equiv`. The hosts in this file are other machines in which users can log into without a password. The reverse is also usually true as the machines trust each other.

Case Study: Morris Worm

The worm also took precautions to hide itself

- Tried to limit infecting already-infected machines
  - However, a miscalculation meant that the limit was not effective, causing aggressive re-infection, slowing machines and networks dramatically
- Deleted files after they were loaded in memory, obscured program arguments, killed unneeded parent processes, so it was hard to analyze or detect the worm

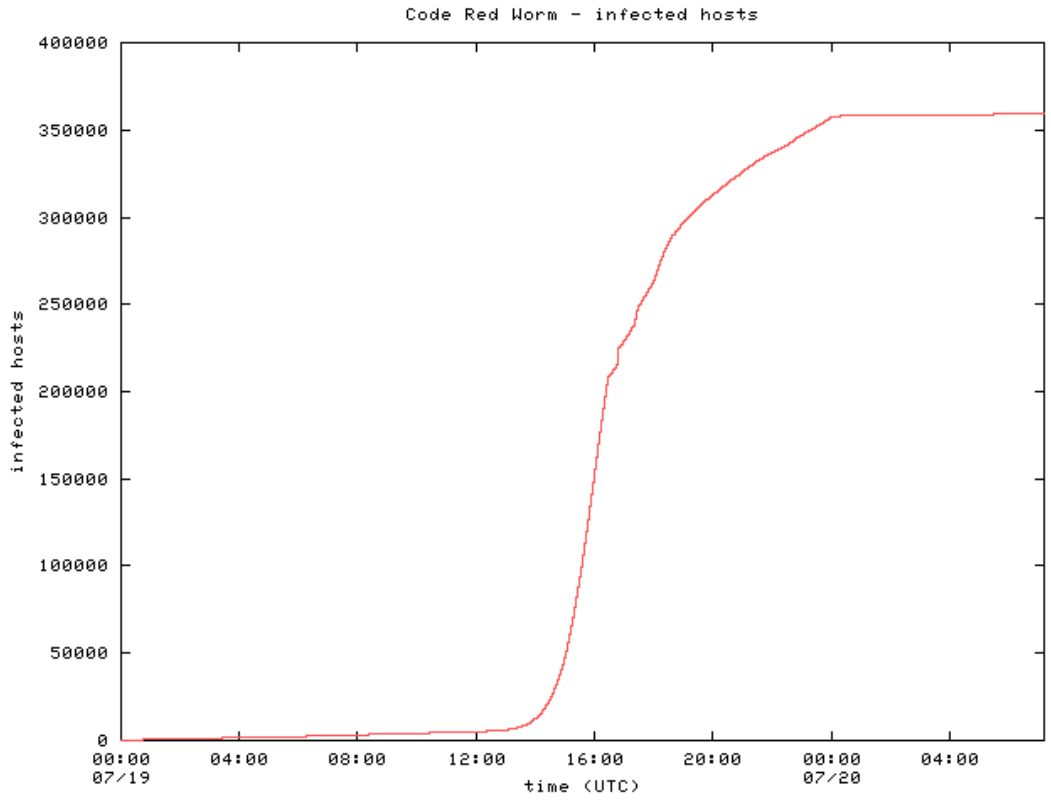
Modern Worms

Modern worms aim to increase spreading speed.

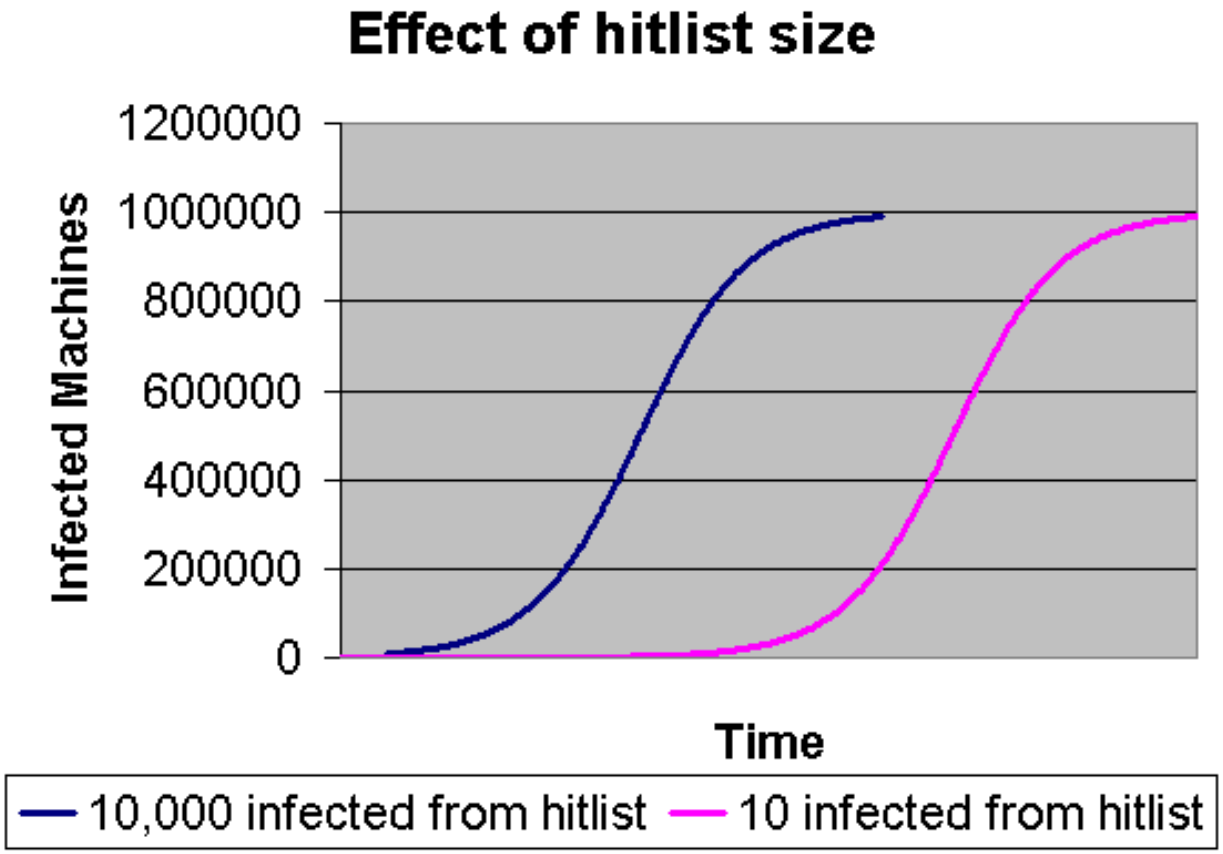
- Improve scanning speed for vulnerable machines
  - **Hit list scanning:** Worm builder pre-seeds worms with hosts that are potentially vulnerable (*e.g.*, hosts running a specific version of a program) and, ideally, with high bandwidth (university hosts, other large institutions, etc), so that initially the worm has a lot of scanning bandwidth
  - **Local scanning vs. random scanning:** A worm can try infecting local hosts first since connecting to them will be faster
- Use UDP instead of TCP
  - Slammer was able to exploit MS SQL Server using a single UDP packet, and since UDP does not use acknowledgements, Slammer can send an attack packet without having to wait for an ACK, making it bandwidth limited rather than latency limited

Worm Propagation

Worms start slowly, then propagate rapidly until most vulnerable machines have been exploited



Using Hitlist Scanning



Stuxnet

Released in January 2009 and detected in June 2010, the *Stuxnet* worm marked a significant shift in electronic attacks

- Appears to have been specifically written to destroy uranium-enrichment centrifuges at Natanz plant in Iran
- Appears to have been moderately successful

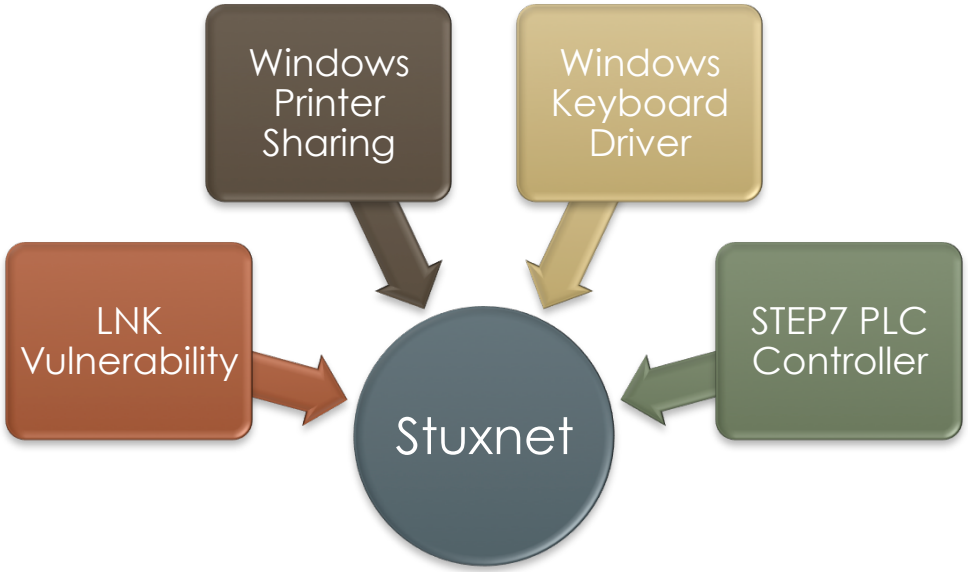
Zero-Day Exploits

A previously-unknown, exploitable security vulnerability is referred to as a **Zero-Day Exploit**

- Depending on its nature, can be quite valuable: sell for anywhere from $50k to $1M on black-market hacking websites
- Value is quickly lost if exploits are used (and consequently revealed), as software creators and security maintainers will react quickly to eliminate the vulnerability
- Out of more than 12 million pieces of malware discovered per year, less than a dozen (0.0001%) use a zero-day exploit

Stuxnet: Zero-Day Exploits

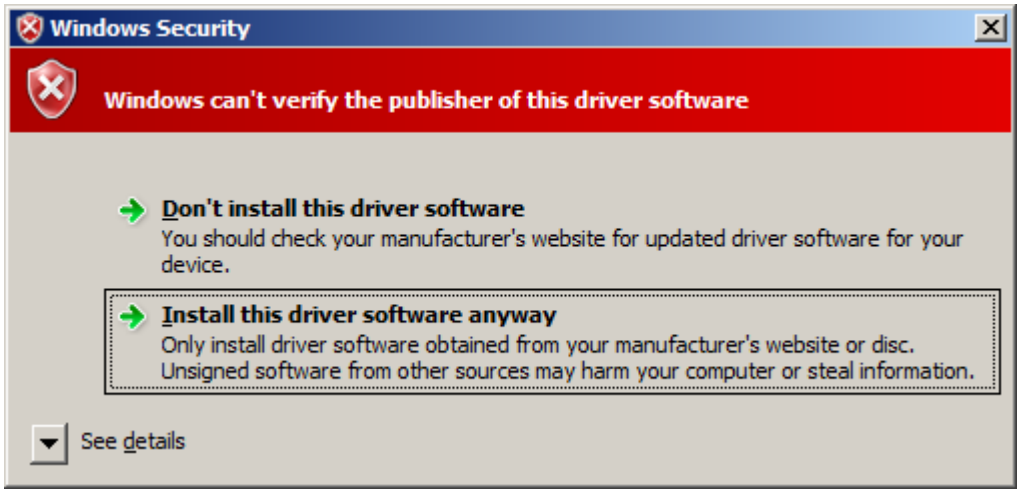
Was immediately clear that Stuxnet was viewed by its creators as high-worth, as it exploited at least four zero-day exploits



Driver Signing

Windows uses a signing mechanism to validate device driver and OS files

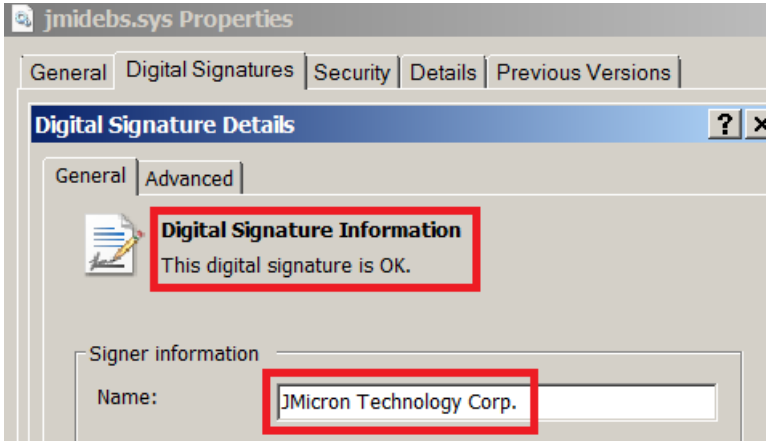
- Key files are signed by their author, using a public-key signature, to establish the authenticity of a file
- Can be used to detect changes to the file, or unauthorized software (*e.g.*, viruses, worms)



Stuxnet: Driver Signing

The files used by Stuxnet to spread were digitally signed using certificates stolen from two major hardware manufacturers (JMicron and RealTek)

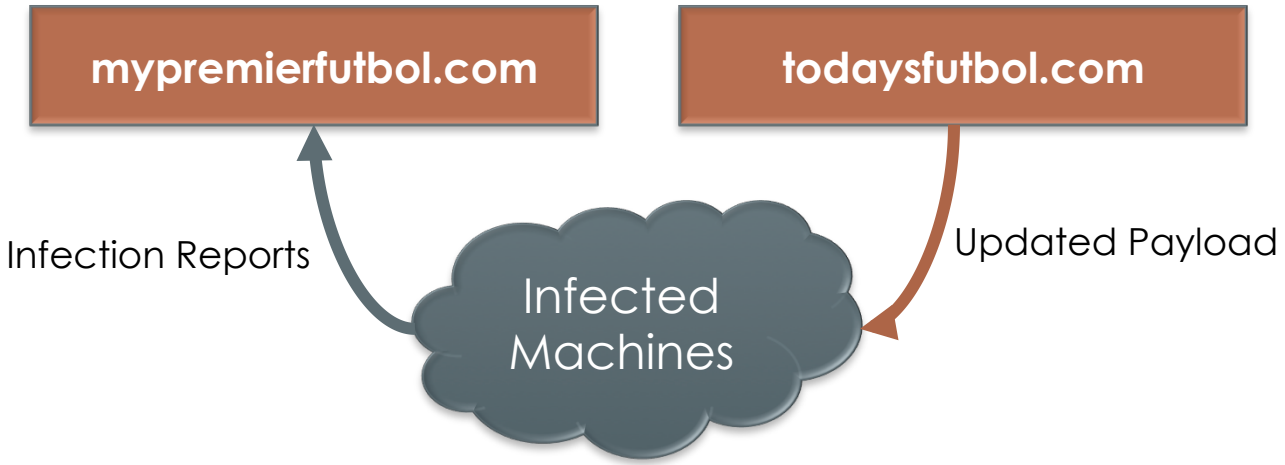
- Headquarters are adjacent, in the same business park in Taiwan



Stuxnet: Command and Control

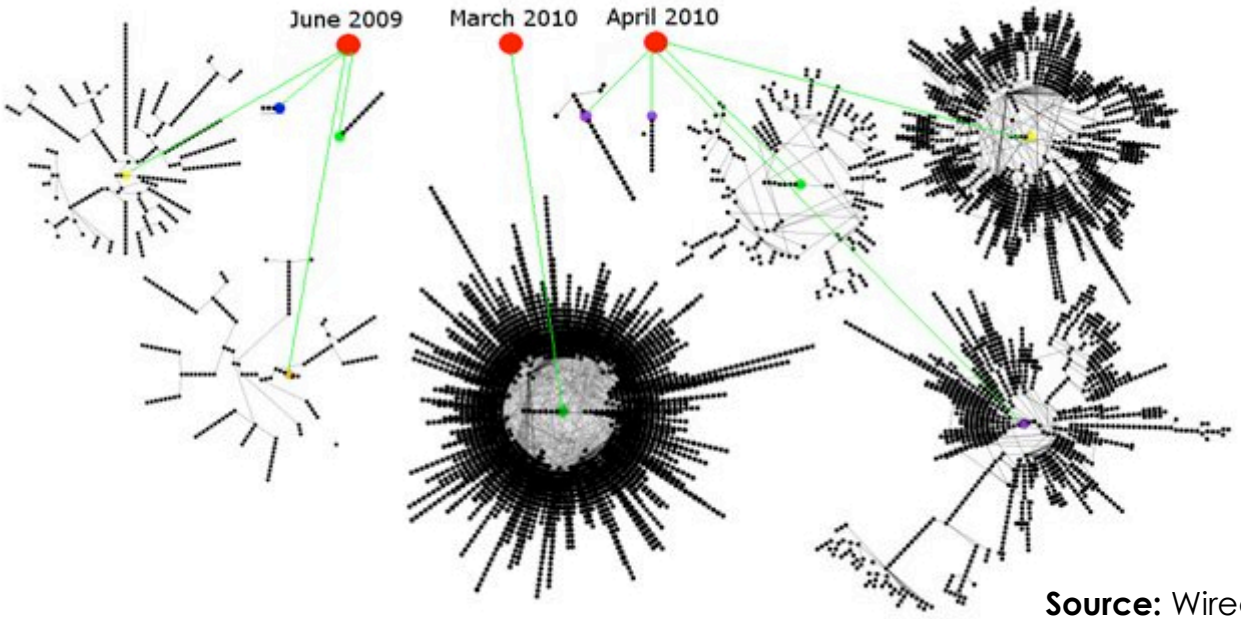
Stuxnet was controlled and monitored via a central pair of websites

- Received reports of infected machines
- Distributed updated commands/payload to the running copies of the worms



Stuxnet: Infection Pattern

Further evidence for the argument that Stuxnet was a targeting Iran is the pattern of infected hosts: can be traced back to five primary infections, all related to Iranian nuclear production

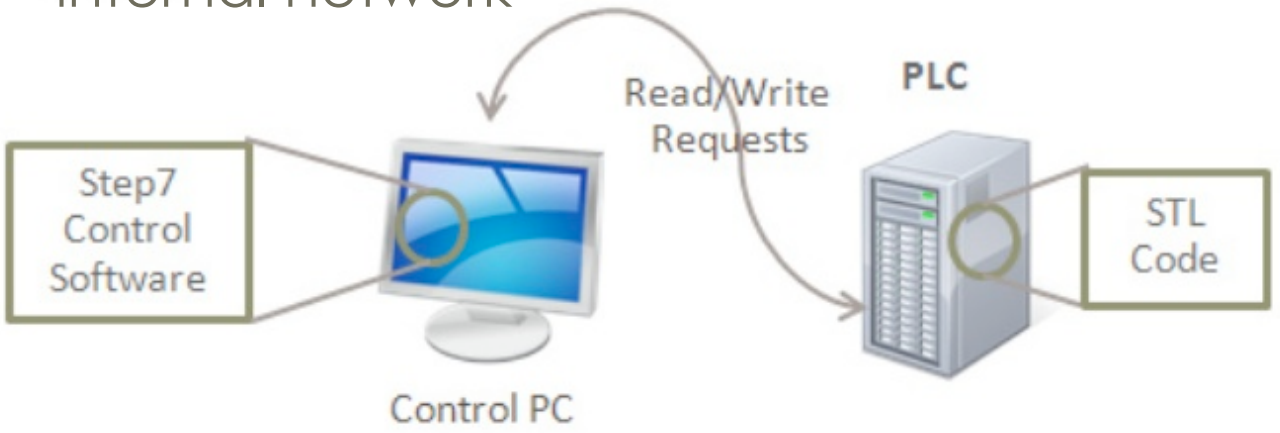


Source: Wired.com

PLCs

PLC (Programmable Logic Controllers) are small programmable controllers, designed to automate mechanical systems (assembly robots, valves, etc.)

- Typically poor security in these systems, as it's often assumed that they are isolated on an internal network



Stuxnet: What Was it Targetting?

It was written to target one specific network:

- Infected thousands of industrial control systems around the world (reactors, public utilities, oil rigs, etc.), but did nothing to change their operation
- Was checking for a specific combination of frequency controllers made in Finland and Iran
- Would only deploy its payload if the facility had exactly 33 frequency counters installed, operating at a frequency of 1,064Hz
  - IAED confirmed that was the exact layout at the Iranian production facility

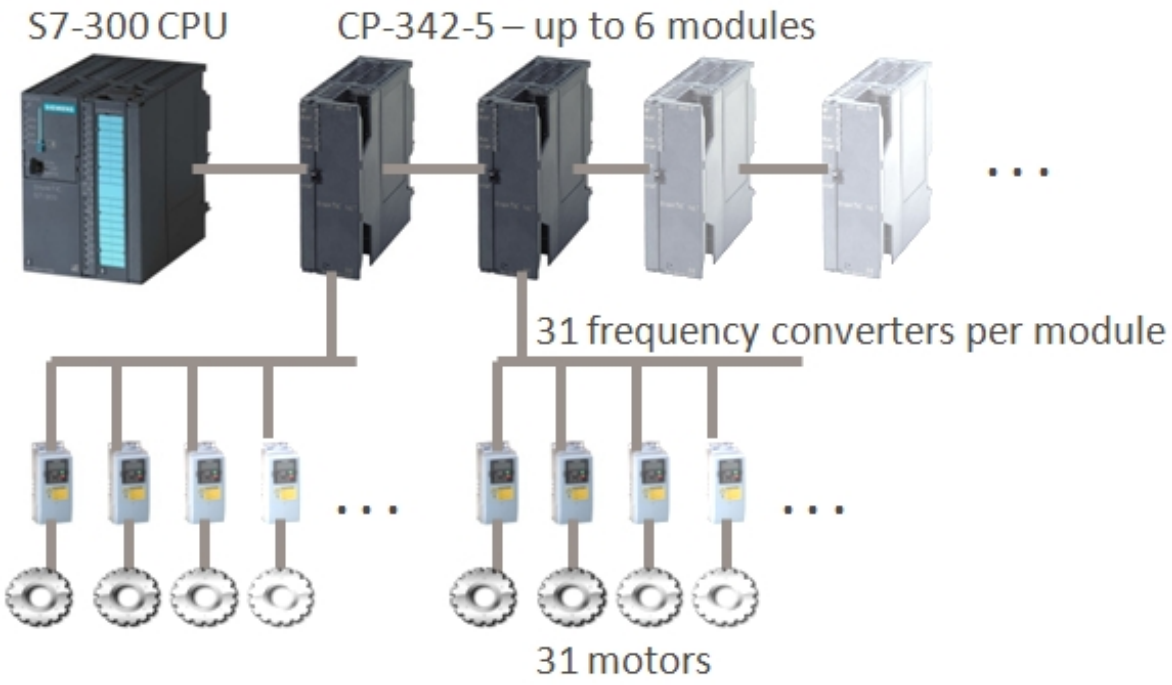
Stuxnet: What Was it Doing?

It was written to reprogram a specific PLC (Programmable Logic Controller) produced by Siemens

- Designed to send malicious commands over the control bus to rapidly speed up / slow down the centrifuges
- Disabled monitoring alarms that would normally report anomalies in the system
- Actively intercepted and changed status reports from the centrifuges, to misreport their current state of operation

Stuxnet: What Was it Doing?

It was written to reprogram a specific PLC (Programmable Logic Controller) produced by Siemens



Stuxnet: What Was it Doing?

It was written to reprogram a specific PLC (Programmable Logic Controller) produced by Siemens

- Would lay dormant for several weeks
- Would suddenly increase the centrifuge speed for 15 minutes, then restore it to normal
- Would then lay dormant for another several weeks
- Would suddenly drop the centrifuge speed to almost zero for 50 minutes
- Would then repeat…

Stuxnet: How Effective Was It?

- IAEA reports that Iran normally replaces 10% of its centrifuges per year, due to mechanical failures, electrical defects, etc.
- During period that Stuxnet was active, the failure rate increased to between 60% and 70% per yer
- It was discovered and the control structure was dismantled before several of the other known payloads were deployed
- Concern for the future: the Stuxnet code has been widely disclosed and dissected
  - Iran has been openly forming a hacking group within the Iranian Revolutionary Guard

Stuxnet: Fall-out

- Son of Stuxnet Found in the Wild on Systems in Europe (Wired, Oct 2011)
  - Appears to exist primarily to allow remote access to data on infected systems
- Richard Clarke on Who Was Behind the the Stuxnet Attack (Smithsonian, Apr 2012)
  - "*… you can take it apart and you can say, 'Oh, let's change this over here, let's change that over there.' Now I've got a really sophisticated weapon. So thousands of people around the world have it and are playing with it. And if I'm right, the best cyberweapon the United States has ever developed, it then gave the world for free.*"

Stuxnet: Further Reading

- How Digital Detectives Deciphered Stuxnet, the Most Menacing Malware in History (Wired, Jul 2011)

Worm Defenses

Prevent Attacks

- Worms exploit vulnerabilities, so patch systems regularly
- Disable unnecessary services to reduce the possibility of running a vulnerable service
- Services that do no need to be externally visible should be firewalled

After worm is released

- Shut down vulnerable service, so that it cannot be infected
- Create a signature for detected worms and filter them at the network layer
  - This approach is similar to a host-based virus scanner
  - People speculate that worms will soon start to use polymorphism and metamorphism to evade such scanners

Worm Defenses

Research ideas

- Earlybird (UC San Diego)
  - As a worm starts spreading, its code will appear in an increasing number of network messages
  - Thus worms can be identified by looking for commonly occurring substrings in packets
  - To reduce false positives, filter by only flagging substrings that appear in many packets with highly diverse src/dest pairs
- Shields (Microsoft Research)
  - Vulnerabilities often lie in obscure paths (*i.e.*, some rarely used feature) in network protocols that aren't heavily tested
  - Worms exercise these paths so if a protocol is in one of these paths, then block
  - Works as a quick fix until a patch can be released so you don't have to shut down vulnerable service

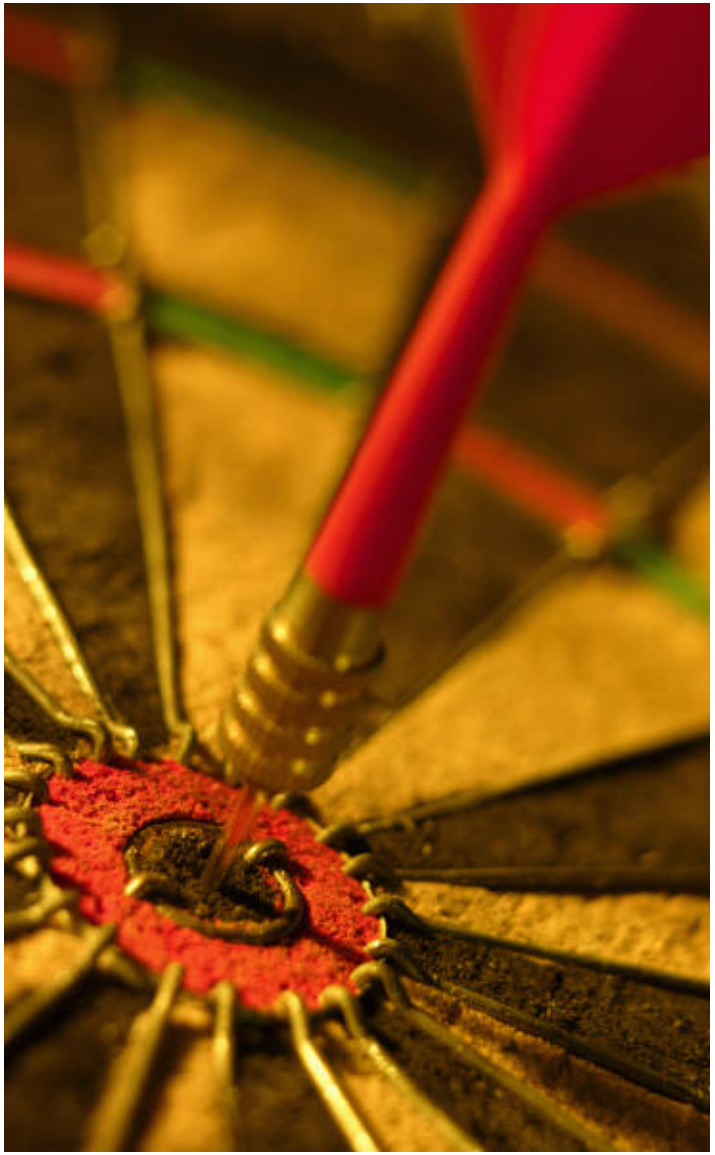
Botnets

A **Botnet** is a collection of compromised machines running malicious software under a common command-and-control infrastructure

- The software is typically installed using worms, trojans, or backdoors
- Used for DoS, spamming, key logging, spyware, etc.

Characteristics of botnets

- Remote control facility for coordinating bot machines
  - *e.g.*, use IRC, HTTP, covert channels, with optional encryption
  - Newer botnets provide general purpose remote execution
- Spreading mechanism similar to worms
  - Counter-measures are similar to worms



Rootkits

Rootkit

A **rootkit** is any software designed to hide the fact that a system has been compromised

- Typically, it subverts the mechanisms that report on processes, files, registry entries, etc.
  - As a result, is hard to detect

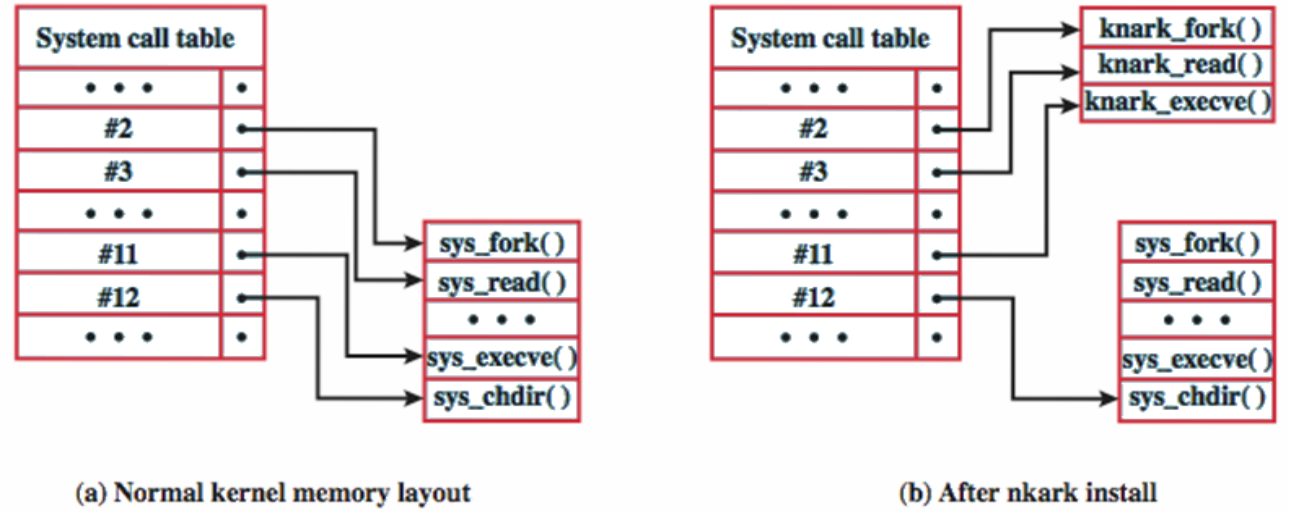
A rootkit may be:

- Memory-based
  - Does not survive a reboot
- Persistent
  - *e.g.*, stored in config file and runs on each boot

Kernel Rootkits

A rootkit may be

- **User mode**
  - Intercepts library or system calls at the user level and modifies returned results
- **Kernel mode**
  - Intercepts system calls in the kernel, changes kernel code or data structures, and modifies returned results



(a) Normal kernel memory layout

(b) After nkark install

Sony Rootkit Scandal

In 2005 Sony BMG included copy protection measures on 102 albums

- Two enforcement mechanisms, each of which included Windows and OS X rootkits
  - XCP (eXtended Copy Protection)
  - MediaMax CD-3
- Automatically installed when the CD was inserted, and interfered with the normal way that the OSes play CDs
  - Installed automatically, with no notice in the EULA or agreement from the user

Sony Rootkit Scandal

Additional problems quickly surfaced:

- Code included a number of open source libraries (LAME, VLC) in a way that violated their licenses
- Copy protection programs constantly scanned the system all the time, eating system resources
- Race condition in the code could lead to Windows crashes ("blue screens")
- Would cripple systems if uninstalled by many of the usual anti-rootkit tools available at the time
- Code contained several vulnerabilities that are exploitable by other malware and viruses

Sony Rootkit: Fallout

Sony eventually recalled and replaced the effected CDs, offered an uninstaller

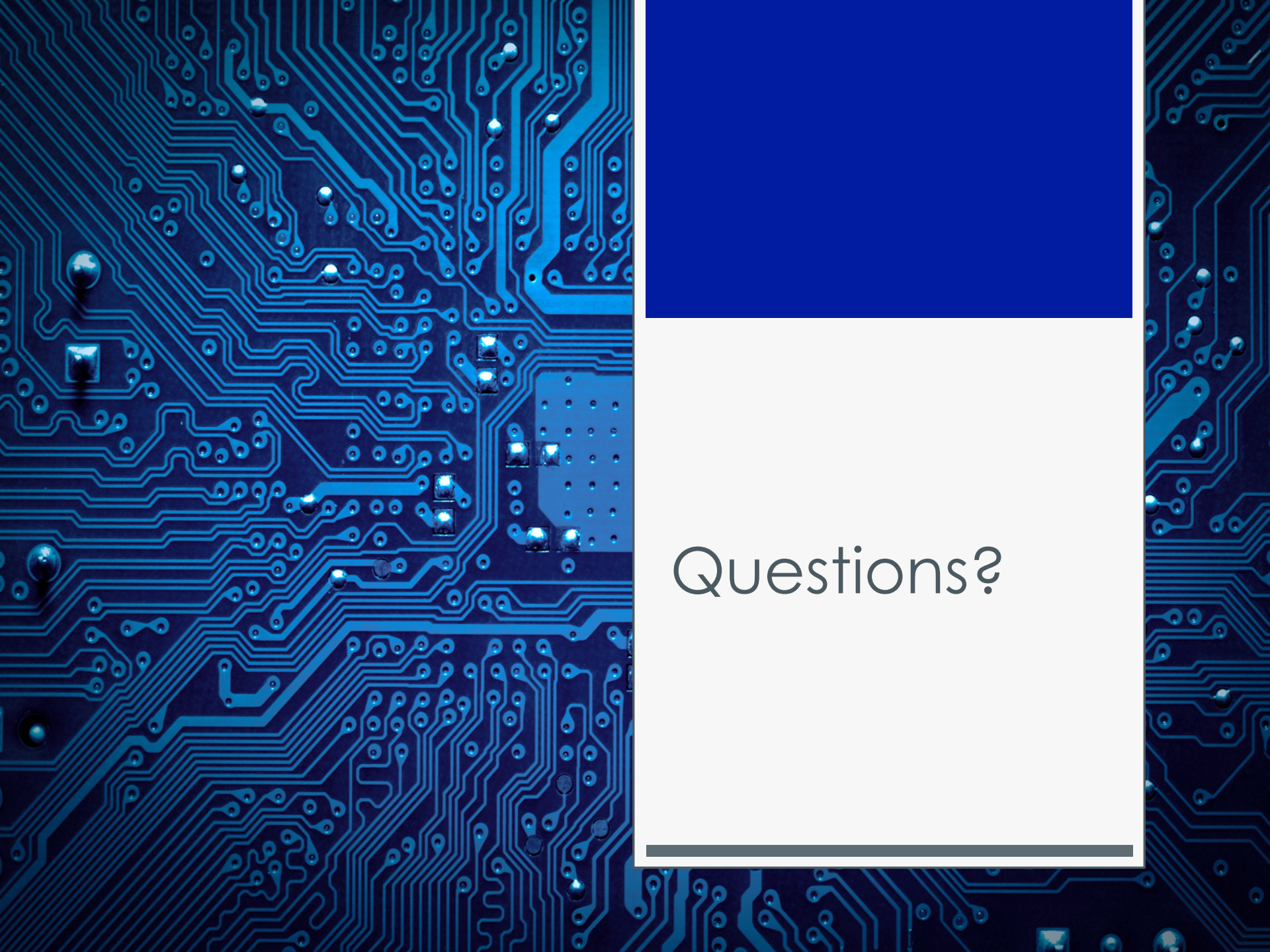
- Uninstaller introduced a new ActiveX security hole on the user's computer: allowed malicious websites to execute commands

Several class-action lawsuits and investigations by state Attorneys General

Rootkit Defenses

Rootkits can be installed using viruses, worms, trojans, backdoors, or an intruder on the system

- Defenses are similar to virus, worm defenses
  - Host and network based signature detection
  - File integrity checks that bypass system call
    - *e.g.*, access disk directly


Questions?