EECS 388

Introduction to Computer Security

Lecture 15:

Malware

Oct. 31, 2023

Prof. Ensafi

What is Malware?

Malware (malicious software)

is any software **intentionally designed** to surreptitiously run on client devices, servers, or networks and **cause harm**.

Frequent malware goals:

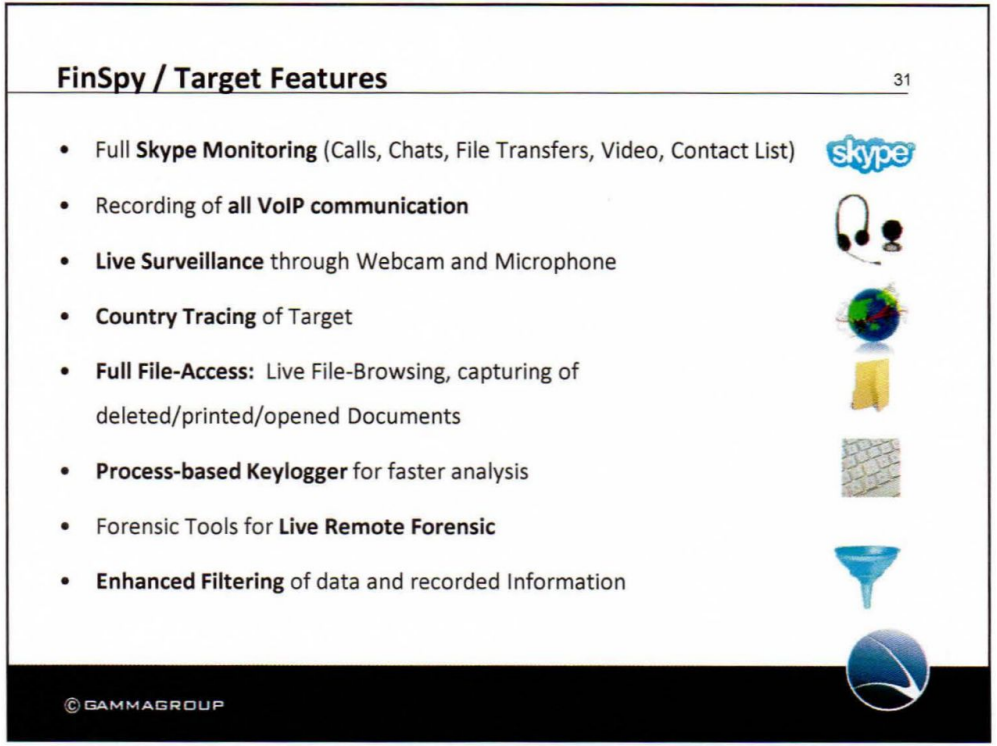
- Stealing private data
- Spying on the user
- Granting unauthorized access
- Stealing computing resources
- Damaging devices
- Congesting networks
- Attacking other systems
- Displaying ads
- Committing fraud
- Extorting the user

Kinds of Malware: Spyware

Spyware is malware that provides remote access to data and sensors available from the device.

Common spyware features:

- Log keystrokes, targeting passwords and credit card data
- Capture screenshots, browsing history
- Exfiltrate files
- Track network or location data
- Access microphone, camera



Example: **FinFisher** spyware has been used by repressive governments to target activists.

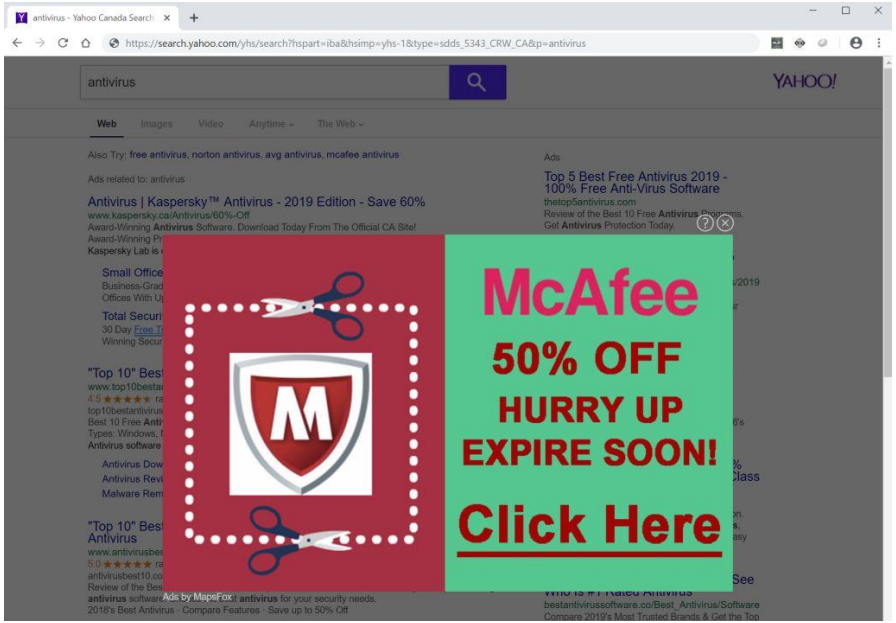
Kinds of Malware: Adware

Adware is unwanted software that displays paid advertisements.

Ads may take the form of pop-ups, interstitial ads, or replacing ads on sites with ads sold by the adware maker.

Sometimes will change home page or default search engine.

Sometimes tracks users' browsing for ad targeting.



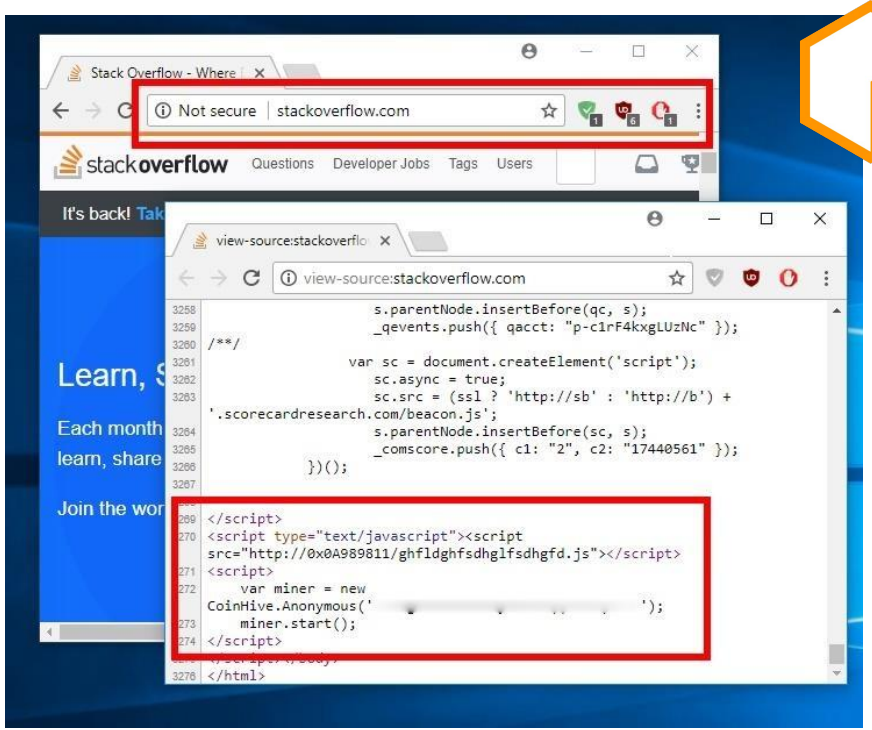
Example: **MapsFox** adware is sometimes quietly bundled with freeware apps to increase revenue.

Kinds of Malware: Cryptojacking

Cryptojacking malware uses the user's device to mine for cryptocurrency.

May take several forms:

- Client-side apps or browser extensions
- Code placed on sites via server-side compromise
- Code injected into sites by malicious network components



Example: **CoinHive** cryptojacker script injected into a page via a MITM attack.

Kinds of Malware: Ransomware

Ransomware encrypts files on the device and demands payment to decrypt them.

Typically works like this:

1. Malware generates random key **k**.
2. Encrypts user's files with **k** and securely erases the originals.
3. Encrypts **k** with attacker's public key. Demands payment.
4. Upon receiving payment, attacker decrypts **k** and sends to user.

Example: **WannaCry** (2017) spread using a leaked NSA exploit and infected over 300,000 machines.

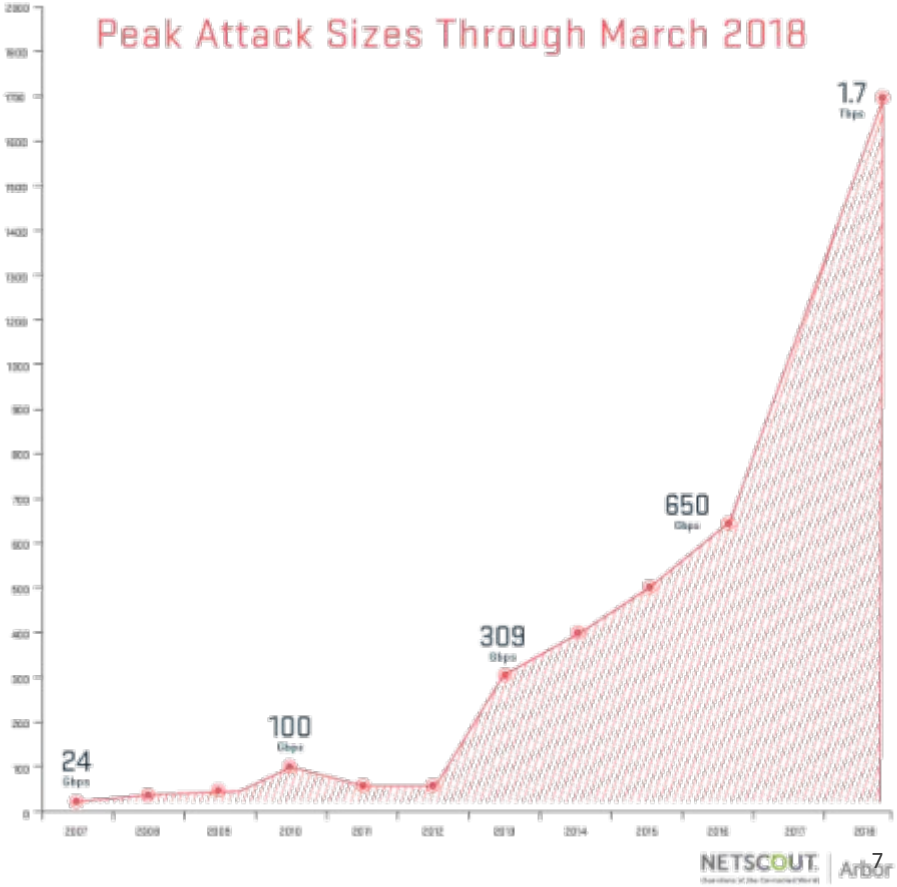
Kinds of Malware: DDoS

Malware can be used to conduct **distributed denial-of-service (DDoS)** attacks that overwhelm target site with traffic.

Large number of infected clients makes traffic more difficult to filter out.

Attackers may demand extortion payment to make traffic stop.

Today, large DDoS attacks can exceed **1 Tbps** of traffic, threatening even large CDNs and core Internet infrastructure.



Infection Methods

Many ways malware might infiltrate a system:

Software exploits: e.g., buffer overflow in network service or device OS

Drive-by downloads: site exploits client vulns. to install malware.

Malicious networks: malicious code inserted into unencrypted web page.

Compromised server: source code or binary altered before download.

Social engineering: attacker tricks user into installing/running malware.

Supply chain attacks: malicious functions added to a system component during development, manufacturing, or shipping.

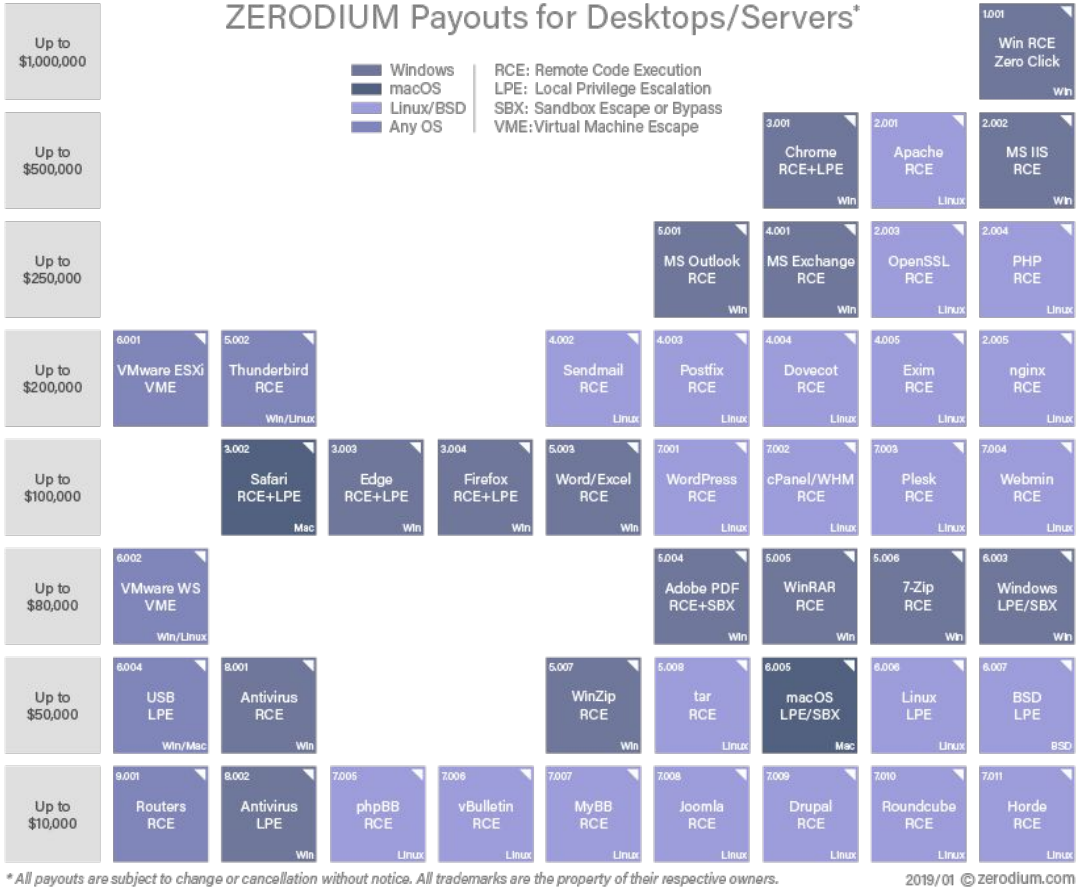
Insider attacks: attacker with local access downloads/runs it directly.

Infection: Software Exploits

Thriving, legal market for "**zero-day**" (novel, unpatched) exploits in popular software

Example:

Zerodium is a company that buys zero-days from discoverers and sells to government clients.



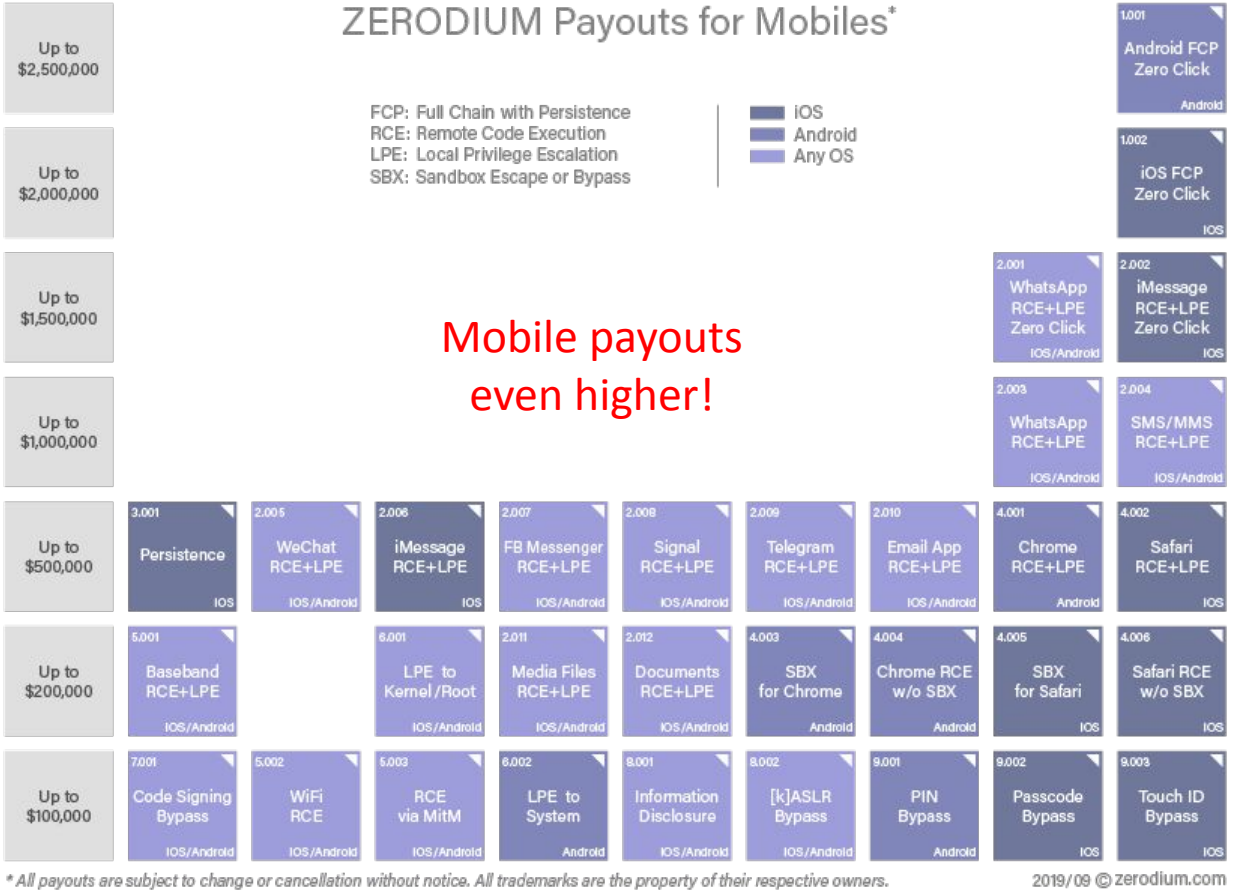
What software you use is on this list?

Infection: Software Exploits

Thriving, legal market for "**zero-day**" (novel, unpatched) exploits in popular software

Example:

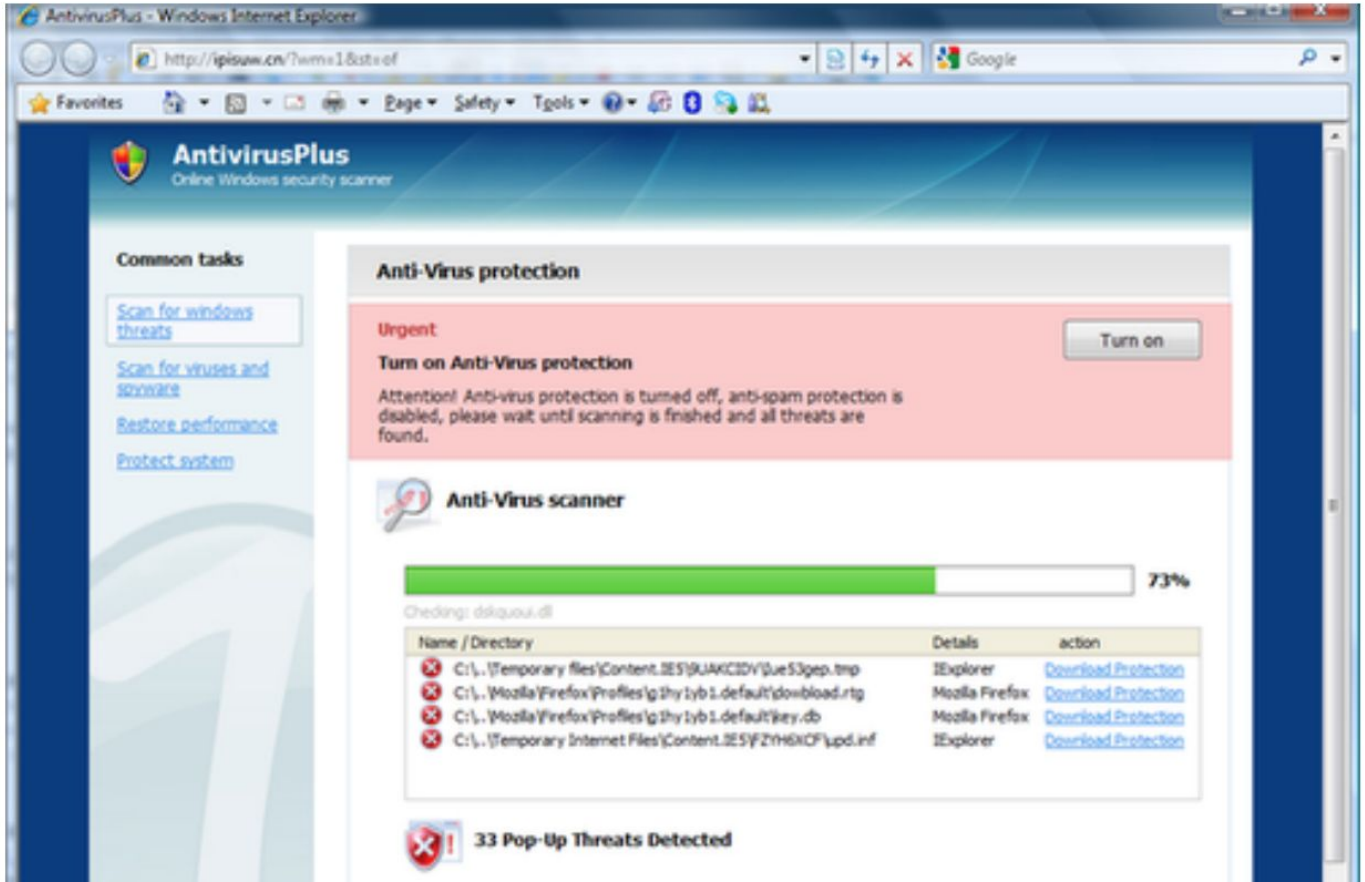
Zerodium is a company that buys zero-days from discoverers and sells to government clients.



Infection: Social Engineering

Example:

Fake antivirus, pops up a warning that machine is infected and offers to clean for a fee



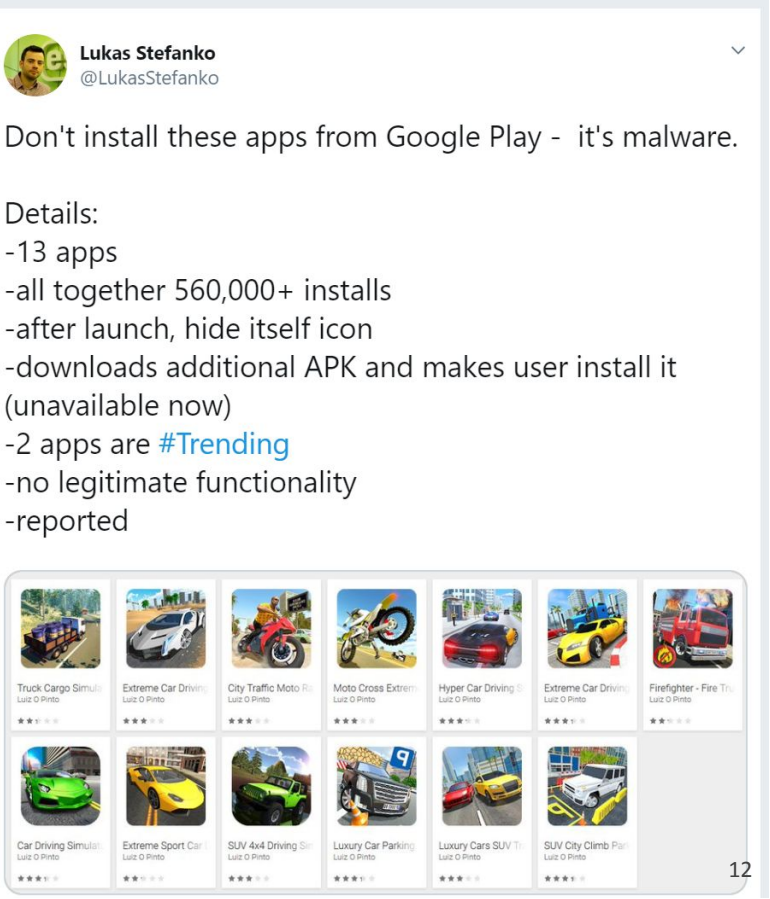
Infection Methods: Trojan Horses

A **Trojan horse** appears to perform a desirable function but actually is designed to perform **undisclosed malicious functions**.

A form of social engineering.

Many diverse forms. Two examples:

- **FakeAV Trojans:** Fake antivirus programs falsely claim to detect malware and requests payment to remove it.
- **App Repackaging:** Clones of popular apps that actually contain malicious code. Frequent problem in app stores.



Infection: Social Engineering

Example:

Exploit USB autorun functionality

Users Really Do Plug in USB Drives They Find

Matthew Tischer[†] Zakir Durumeric[‡†] Sam Foster[†] Sunny Duan[†]
Alec Mori[†] Elie Bursztein[◇] Michael Bailey[†]

[†] University of Illinois, Urbana Champaign [‡] University of Michigan [◇] Google, Inc.
{tischer1, sfoster3, syduan2, ajmori2, mdbailey}@illinois.edu
zakir@umich.edu elieb@google.com

(a) Unlabeled drive (b) Drive with keys (c) Drive with return label (d) Confidential drive (e) Exam solutions drive

Fig. 1: **Drive Appearances**—We dropped five different types of drives. We chose two appearances (keys and return label) to motivate altruism and two appearances (confidential and exam solutions) to motivate self-interest, as well as an unlabeled control.

Infection: Drive-by Downloads

Example:

U.S. Government search warrants describe installing malware on a target's computer as a "network investigative technique"

ATTACHMENT B

This warrant authorizes a remote network technique in which law enforcement agents will transmit to each of the TARGET COMPUTERS described in Attachment A code and/or commands intended make the following information available to officers authorized to execute this warrant:

(1) the TARGET COMPUTER's actual IP address, and the date and time that the IP address is determined;

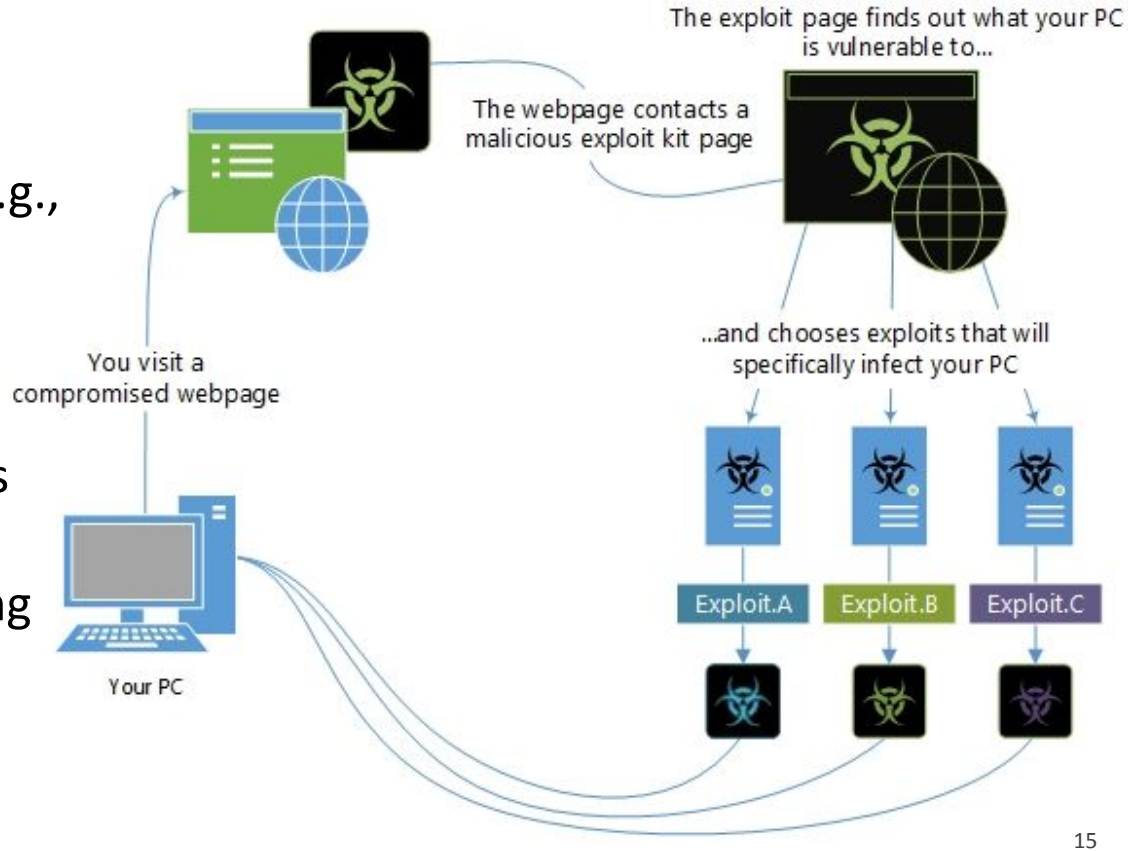
(2) The TARGET COMPUTER'S Computer Name and Media Access Control Address; and

(3) a unique identifier (e.g., a series of numbers, letters, and/or special characters) for the TARGET COMPUTER.

Infection: Exploit Kits

A **drive-by download** exploits vulnerabilities in client software, such as the browser or plugins (e.g., PDF reader, Java, Flash) to infect your device.

Exploit kits automate the process by scanning your device for vulnerable software and deploying corresponding exploits to install malware.



Infection: Supply-chain Attacks

Example:

Fake Cisco equipment sold in China contained malware (2008)

Infection: Supply-chain Attacks

Example:

NSA supply chain interdiction to insert backdoors into Cisco products (2014)

(TS//SI//NF) Such operations involving **supply-chain interdiction** are some of the most productive operations in TAO, because they pre-position access points into hard target networks around the world.

(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A "load station" implants a beacon

Infection: Software Supply-chain Attacks

Example:

Infiltrate software provider. Juniper code base compromised in 2012 and 2014, discovered in 2015

2015-12 Out of Cycle Security Bulletin: ScreenOS: Multiple Security issues with ScreenOS (CVE-2015-7755, CVE-2015-7756)

[JSA10713] Show Article Properties

PRODUCT AFFECTED:
Please see below for details.

PROBLEM:
During an internal code review, two security issues were identified.

Administrative Access (CVE-2015-7755) allows unauthorized remote administrative access to the device. Exploitation of this vulnerability can lead to complete compromise of the affected device.

This issue only affects ScreenOS 6.3.0r17 through 6.3.0r20. **No other Juniper products or versions of ScreenOS are affected by this issue.**

Upon exploitation of this vulnerability, the log file would contain an entry that '**system**' had logged on followed by password authentication for a username.

Example:

Normal login by user **username1**:

```
2015-12-17 09:00:00 system warn 00515 Admin user username1 has logged on via SSH from …..
2015-12-17 09:00:00 system warn 00528 SSH: Password authentication successful for admin
user 'username1' at host …
```

Compromised login by user **username2**:

```
2015-12-17 09:00:00 system warn 00515 Admin user system has logged on via SSH from …..
2015-12-17 09:00:00 system warn 00528 SSH: Password authentication successful for admin
user 'username2' at host …
```

Note that a skilled attacker would likely remove these entries from the local log file, thus effectively eliminating any reliable signature that the device had been compromised.

Hardware Trojans

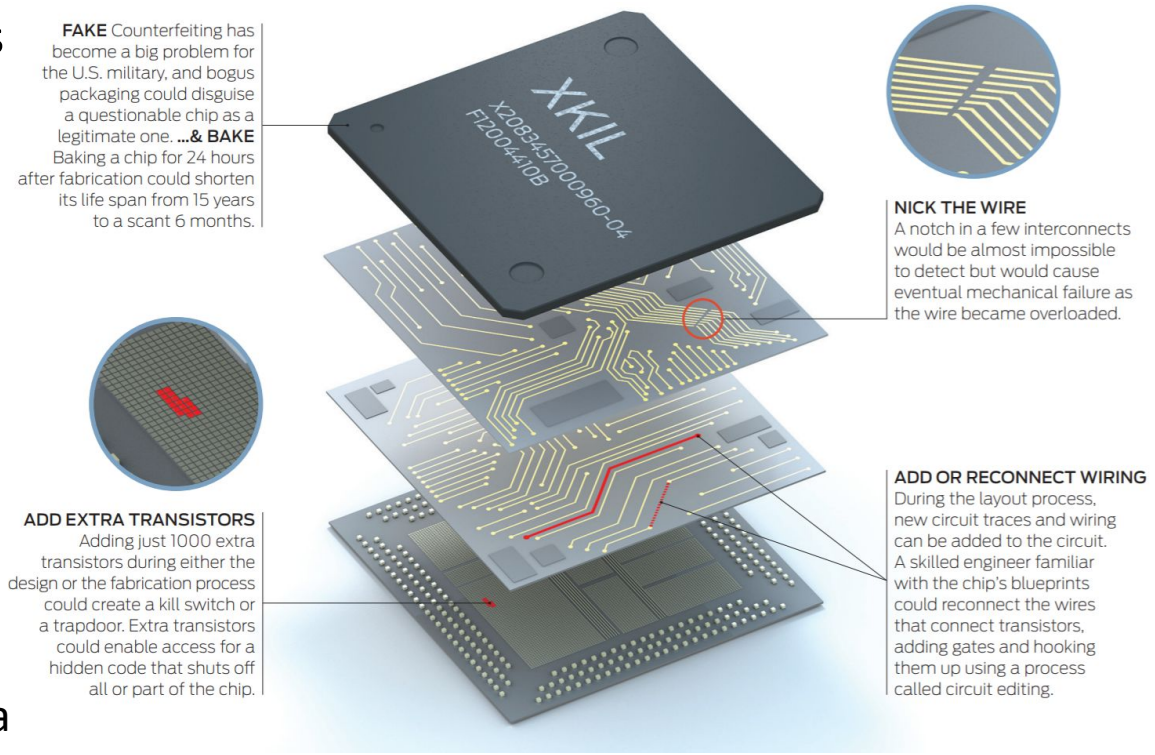
A **hardware Trojan** is a malicious change to circuits or chips.

Can be introduced during design, manufacturing, or distribution.

Possible payloads:

- Introduce malicious code
- Disable security features
- Leak data over networks or via radio emanations
- Disable or destroy device

Can be designed to be triggered by a secret pattern of data on a bus.

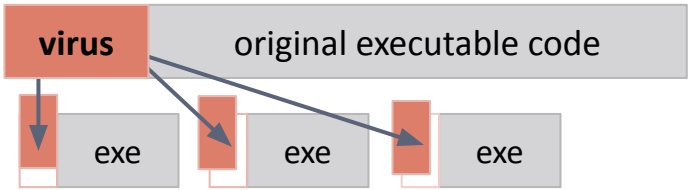


Possible to insert a Trojan with only *chemical* changes to dopants in gates of a chip, making it impossible to detect by optical inspection.

Self-Replicating Malware: Viruses

A **virus** spreads by **modifying other programs** to include copies of itself.

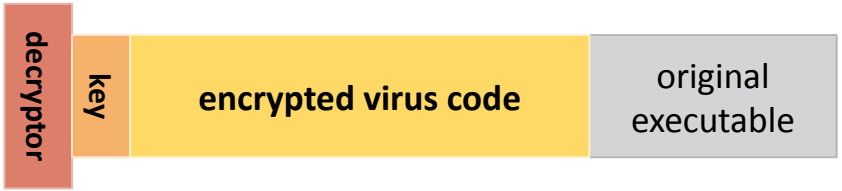
1. A virus infects a program by prepending its code to an executable, so the virus will be executed when the program runs.
2. If an infected program is copied to another system and run, the virus will covertly infect other programs there:



Viruses can **infect document files** too, by exploiting scripting languages (e.g. MS Office macros) or application vulnerabilities.

Viruses can use **self-modification** to "mutate" and avoid detection.

An **encrypted virus** contains a short function that decrypts its main code into memory and runs it:

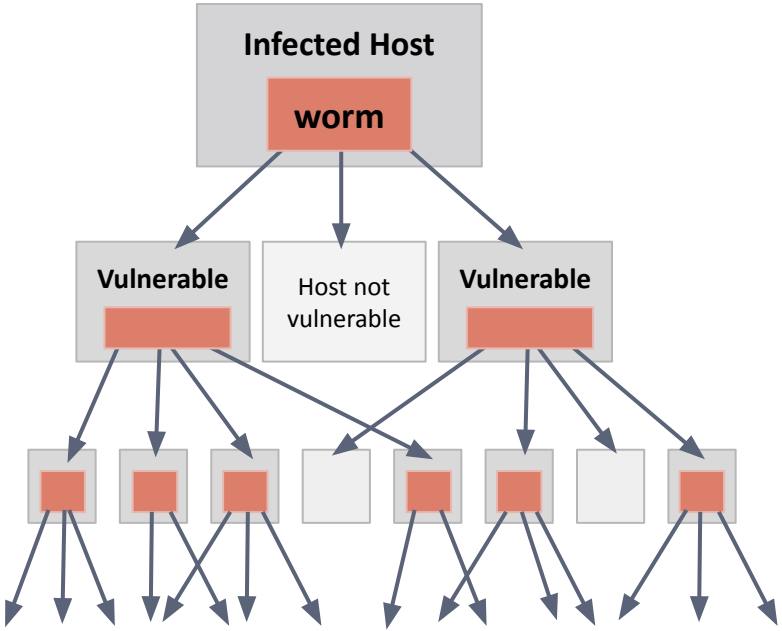


Polymorphic viruses generate a new key and re-encrypt their main code each time they replicate. Only the decryptor remains the same.

Metamorphic viruses rewrite themselves completely each time they replicate, e.g. randomly replacing instructions with equivalent operations.

Self-Replicating Malware: Worms

A **worm** spreads by **exploiting network vulnerabilities** to run copies of itself on remote systems.



Typical worms spread by selecting potential victims at random. Each new host allows worm to spread faster, and **infection can grow exponentially** until vulnerable population is saturated.

Examples of ways worms have spread:

- **Spreading by remote code-injection**

SQL Slammer (2003) exploited buffer-overflow in MS SQL Server. Spread by a single UDP packet, infected entire global population in <10 mins! Took down ATMs, 911.

- **Spreading by cross-site scripting**

Samy Worm (2005) exploited XSS vulnerability in MySpace social network to add itself to users' profiles. In 20 hours, over a million users were infected.

- **Spreading by password guessing**

Mirai Botnet (2016) guessed common passwords for IOT devices. At peak, over 600,000 devices were infected.

History: The Morris Worm

Worms date to November 2, 1988: **The Morris Worm**

Employed a suite of tricks to infect systems and to find victims.

Infected ~10% of computers on the Internet (6000 machines).

Bug caused worm to infect each machine many times, causing downtime and up to $10M in damages.

Created by Cornell graduate student Robert Tappan Morris.

He claims his intent was to highlight security problems.

First prosecution under **Computer Fraud and Abuse Act**.
Sentenced to community service and $10,050 fine.

Now a Professor at MIT (and co-founder of Y Combinator).

More on the Morris Worm

The Morris Worm (1988) exploited a buffer overflow in the fingerd utility, also propagated itself via rsh and cracked passwords.

- Bogged down infected machines by uncontrolled spawning.

More details:

office is, the number of their phone extension and so on. The Berkeley[3] version of the *finger* server is a really trivial program: it reads a request from the originating host, then runs the local *finger* program with the request as an argument and ships the output back. Unfortunately the *finger* server reads the remote request with *gets*(), a standard C library routine that dates from the dawn of time and which does not check for overflow of the server's 512 byte request buffer on the stack. The worm supplies the finger server with a request that is 536 bytes long; the bulk of the request is some VAX machine code that asks the system to execute the command interpreter *sh*, and the extra 24 bytes represent just enough data to write over the server's stack frame for the main routine. When the main routine of the server exits, the calling function's program counter is supposed to be restored from the stack, but the worm wrote over this program counter with one that points to the VAX code in the request buffer. The program jumps to the worm's code and runs the command interpreter, which the worm uses to enter its bootstrap.

Distributed Malware: Bots and Botnets

Botnets are collections of compromised machines (**bots**) under the unified control of an attacker (**botmaster**).

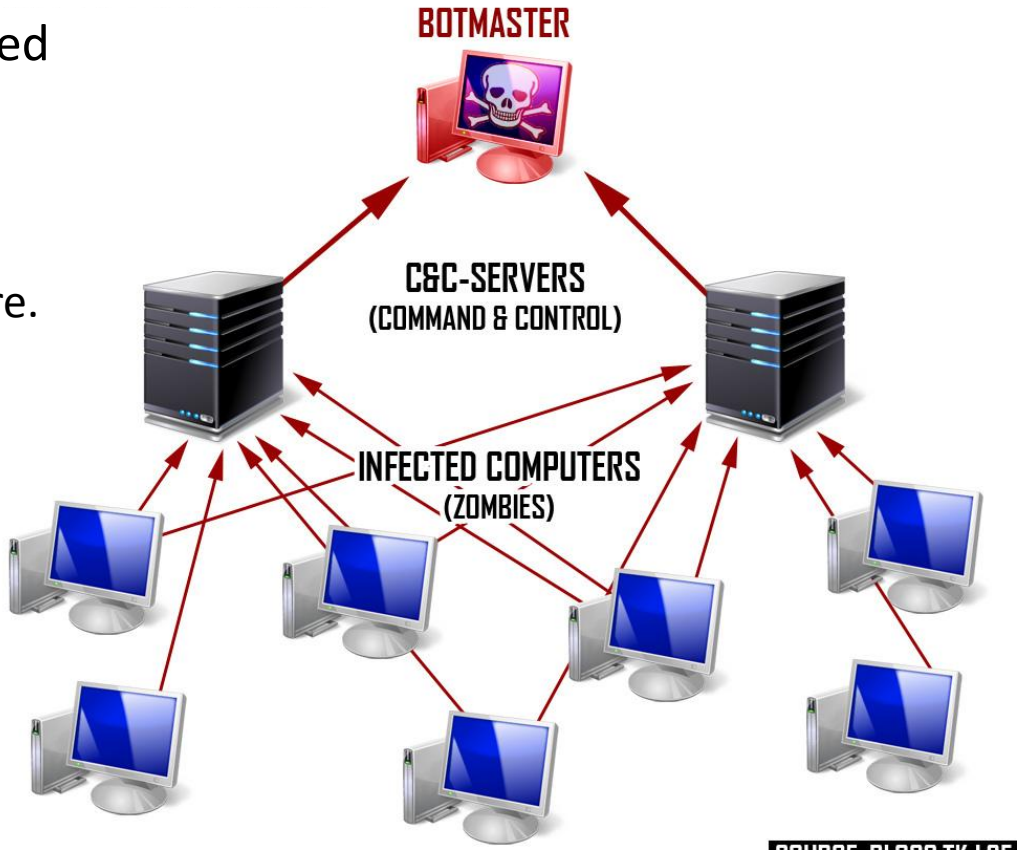
Remotely commanded via **command and control** (C&C) infrastructure.

Method of compromise (any of the above) decoupled from method of control.

Largest grow to 100,000s of bots.

Frequently have a financial motive:
Malware-as-a-service (MaaS)

Typical payloads:
DDoS / spam / infecting other hosts



Botnet Command and Control

Upon infection, new bots “phones home” to rendezvous w/ botnet C&C.

Lots of ways to architect C&C:

- Single server
  (simple but easy to disable)
- Hierarchical or peer-to-peer topology
  (complex but more robust)
- Encrypted/stealthy communication

Botmaster uses C&C to push out commands and updates.

Example of C&C Messages

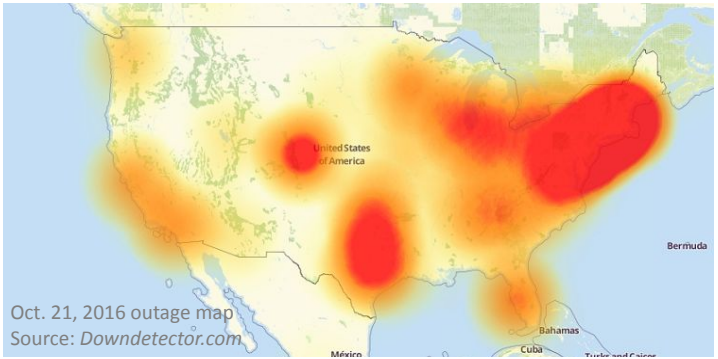
1. Activation
   (report from bot to botmaster)
2. Email address harvests
3. Spamming instructions
4. Delivery reports
5. DDoS instructions
6. Fast Flux instructions
   (rapidly changing C&C DNS)
7. HTTP proxy instructions
8. Sniffed passwords report
9. IFRAME injection/report

From Storm Botnet (2008)

Example: The Mirai Botnet

Mirai botnet infected IoT devices, like smart light bulbs, home routers, webcams

October 2016: Launched massive DDoS against Dyn (large DNS provider), caused Internet outages



Oct. 21, 2016 outage map
Source: *Downdetector.com*

Matthew Green
@matthew_d_green

Follow

Today we answered the question "what would happen if we connected a vast number of cheap, crummy embedded devices to broadband networks?"

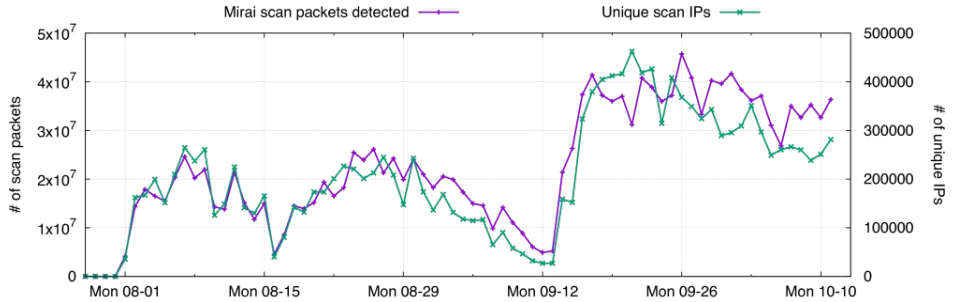
Simple but effective design: Bots telnet to random IP addresses, try 60 common user/pw combos.

root	admin	root	123456	root	12345	admin	smcadmin
admin	admin	root	54321	user	user	admin	1111
root	888888	support	support	admin	(none)	root	666666
root	xmhdipc	root	(none)	root	pass	root	password
root	default	admin	password	admin	admin1234	root	1234
root	juantech	root	root	root	1111	root	klv123

If success, botmaster uploads bot binary (many CPU architectures).

Bots also phone home periodically for attack instructions.

Research tracked Mirai activity using a **network telescope**
(a region of unused IP address space that logs all incoming packets)



Telescope data helped law enforcement find and convict botmaster.

Concealing Malware: Rootkits

A **rootkit** is a malware component that uses stealth to maintain persistent and undetected presence on a machine.

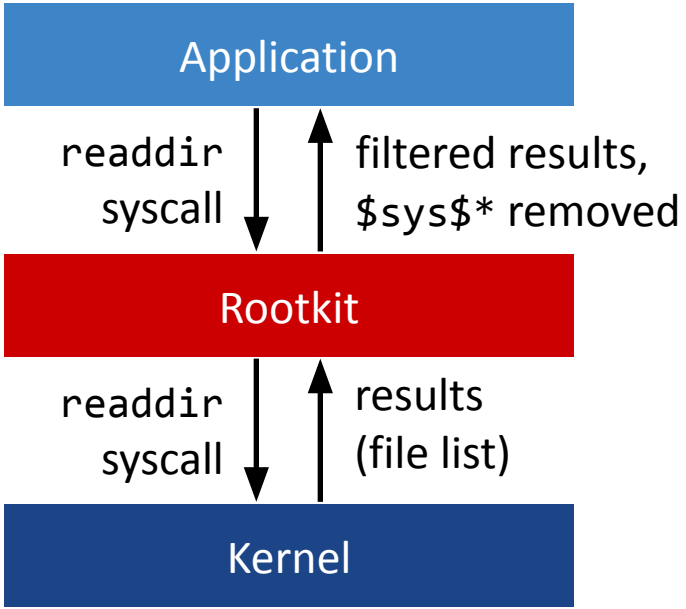
- For instance, modifies OS's file system introspection routines
- Hard to detect using software that relies on the OS itself

Example of a kernel-based rootkit:

1. Intercept system calls for listing files, processes, etc.
2. Act as a filter, removing malware's files and processes

[How can you detect a kernel-based rootkit?]

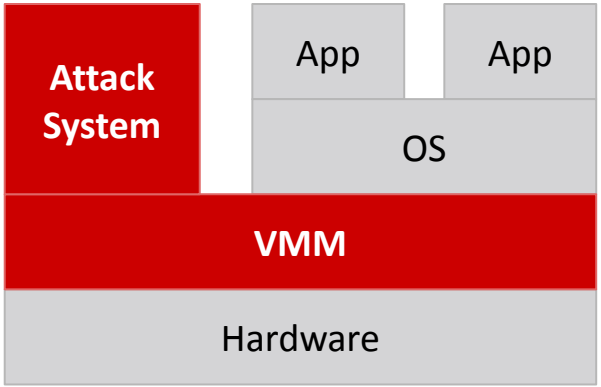
Example: Magic prefix, `$sys$filename`



If call is from rootkit application (e.g. `$sys$rootkit.exe`), don't filter!

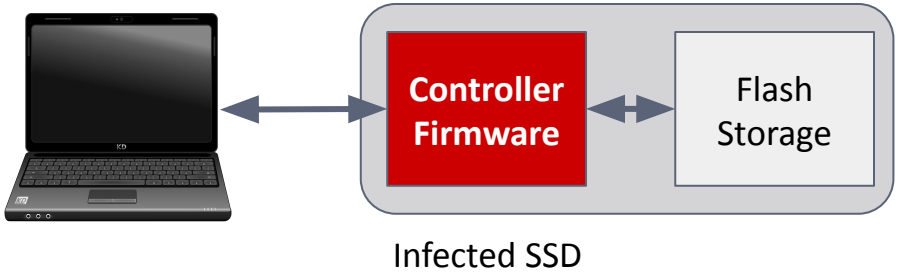
Concealing Malware: Even Greater Stealth

Attacker can install **virtual machine-based rootkit** *below* operating system.



Attack system runs in separate VM (invisible to OS kernel) and has total control of system.

Firmware-based malware can infect e.g., UEFI, IME, disk, or network controller firmware and remain completely invisible from software.



Malicious disk can return different (possibly malicious) data at different times, or in response to different access patterns.

Malware Defenses

Secure programming

Avoid flaws with safe languages and practices.

Security testing

Test for and fix security bugs (e.g., **fuzzing**).

Secure architectures

Apply principles of access control, isolation, and least privilege.

(Example: Compare Android/iOS app isolation model to traditional OSes.)

Run-time detection

Detect malicious code and exploitation attempts.

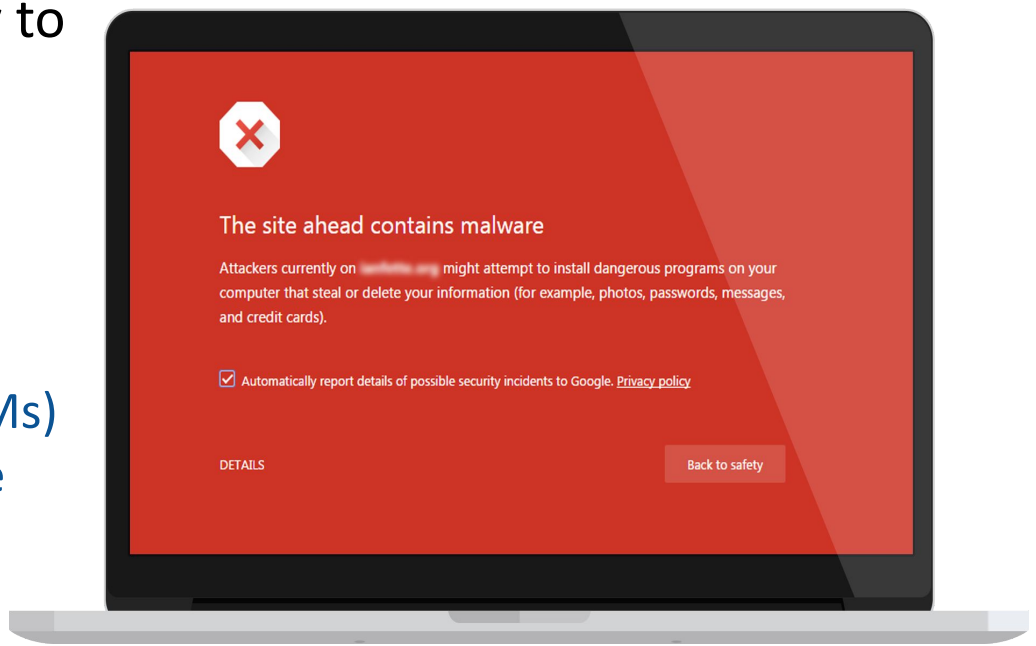
Common sense

Avoid suspicious sites, attachments, and downloads.

Malware Defenses: Safe Browsing

Modern browsers try to warn users when they visit sites that might try to exploit vulnerabilities.

Google Safe Browsing uses a collection of **honeypot** **web crawlers** (vulnerable clients in instrumented VMs) to detect exploitation attempts before users are exposed.



Malware Defenses: Antiviruses and IPSes

Antiviruses inspect software on a host and attempt to identify and disable malware.

Intrusion-prevention systems (IPSes) inspect network traffic and attempt to block exploitation.

Signature detection:

- Analysts find a string that can identify a known virus
- AV or IPS compares against signature database
- Won't work against zero-day attacks or mutating viruses

Heuristic detection:

- Analyze program behavior to identify harmful activity
  e.g. suspicious network/file access, modifies firmware
- Tradeoff between false positives and false negatives

[Pros and cons of using an AV?]

Theorem: Perfect virus detection is impossible

Assume P is a perfect virus detector, V is a possible virus.

V calls P on itself:

```
if P(V) == false:
    spread()
else:
    halt()
```

P(V) gives the wrong answer, so P is not a perfect detector.

Even for programs we write ourselves, cannot trust that binaries are non-malicious. What if compiler is evil? See Ken Thompson, "Reflections on Trusting Trust" (1984).

Summary of countermeasures

- Signature-based detection
  - Look for bytes corresponding to virus code.
  - Antivirus software is a multibillion dollar industry.
- AV arms race:
  - Virus writers change viruses to evade detection.
  - One idea: Virus encrypts its code. Static code detection works less well; decryption code is small, generic.
- Cleanup:
  - Best way: rebuild from original media/backups
  - Some malware contains rootkits: Kernel patches to hide its continuous presence
- Analysis:
  - Run in VM/sandboxed environment
  - Modern malware tries to detect if it runs in VM/fresh install and acts less maliciously

Coming Up

Reminders:

Thursday, Nov. 2 Lab 4 due 6 pm

Thursday

Access Control & Isolation

Isolation, sandboxing, virtual machines,...

Tuesday

Election Cybersecurity

Vulnerabilities, defenses, policy