https://brown-csci1660.github.io

CS1660: Intro to Computer Systems Security
Spring 2025

Lecture 16: OS IV

Co-Instructor: **Nikos Triandopoulos**

March 20, 2025

BROWN

CS1660: Announcements

- Course updates
  - Project 3 is out and due Thursday, April 3
  - Where we are
    - **Part I: Crypto**
    - **Part II: Web** (with demos coming soon)
    - **Part III: OS**
    - Part IV: Network
    - Part V: Extras

Today

- OS security

Malware

Malware

- Programs planted by an agent with malicious intent
  - to cause unanticipated or undesired effects
- Virus
  - a program that can replicate itself
    - pass on malicious code to other non-malicious programs by modifying them
- Worm
  - a program that spreads copies of itself through a network
- Trojan horse
  - code that, in addition to its stated effect, has a second, nonobvious, malicious effect

Types of malware

Code Type	Characteristics
Virus	Code that causes malicious behavior and propagates copies of itself to other programs
Trojan horse	Code that contains unexpected, undocumented, additional functionality
Worm	Code that propagates copies of itself through a network; impact is usually degraded performance
Rabbit	Code that replicates itself without limit to exhaust resources
Logic bomb	Code that triggers action when a predetermined condition occurs
Time bomb	Code that triggers action when a predetermined time occurs
Dropper	Transfer agent code only to drop other malicious code, such as virus or Trojan horse
Hostile mobile code agent	Code communicated semi-autonomously by programs transmitted through the web
Script attack, JavaScript, Active code attack	Malicious code communicated in JavaScript, ActiveX, or another scripting language, downloaded as part of displaying a web page

Types of malware (cont.)

Code Type	Characteristics
RAT (remote access Trojan)	Trojan horse that, once planted, gives access from remote location
Spyware	Program that intercepts and covertly communicates data on the user or the user's activity
Bot	Semi-autonomous agent, under control of a (usually remote) controller or "herder"; not necessarily malicious
Zombie	Code or entire computer under control of a (usually remote) program
Browser hijacker	Code that changes browser settings, disallows access to certain sites, or redirects browser to others
Rootkit	Code installed in "root" or most privileged section of operating system; hard to detect
Trapdoor or backdoor	Code feature that allows unauthorized access to a machine or program; bypasses normal access control and authentication
Tool or toolkit	Program containing a set of tests for vulnerabilities; not dangerous itself, but each successful test identifies a vulnerable host that can be attacked
Scareware	Not code; false warning of malicious code attack

History of malware

Year	Name	Characteristics
1982	Elk Cloner	First virus; targets Apple II computers
1985	Brain	First virus to attack IBM PC
1988	Morris worm	Allegedly accidental infection disabled large portion of the ARPANET, precursor to today's Internet
1989	Ghostballs	First multipartite (has more than one executable piece) virus
1990	Chameleon	First polymorphic (changes form to avoid detection) virus
1995	Concept	First virus spread via Microsoft Word document macro
1998	Back Orifice	Tool allows remote execution and monitoring of infected computer
1999	Melissa	Virus spreads through email address book
2000	IloveYou	Worm propagates by email containing malicious script. Retrieves victim's address book to expand infection. Estimated 50 million computers affected.
2000	Timofonica	First virus targeting mobile phones (through SMS text messaging)
2001	Code Red	Virus propagates from 1st to 20th of month, attacks whitehouse.gov web site from 20th to 28th, rests until end of month, and restarts at beginning of next month; resides only in memory, making it undetected by file-searching antivirus products

History of malware (cont.)

Year	Name	Characteristics
2001	Code Red II	Like Code Red, but also installing code to permit remote access to compromised machines
2001	Nimda	Exploits known vulnerabilities; reported to have spread through 2 million machines in a 24-hour period
2003	Slammer worm	Attacks SQL database servers; has unintended denial-of-service impact due to massive amount of traffic it generates
2003	SoBig worm	Propagates by sending itself to all email addresses it finds; can fake From: field; can retrieve stored passwords
2004	MyDoom worm	Mass-mailing worm with remote-access capability
2004	Bagle or Beagle worm	Gathers email addresses to be used for subsequent spam mailings; SoBig, MyDoom, and Bagle seemed to enter a war to determine who could capture the most email addresses
2008	Rustock.C	Spam bot and rootkit virus
2008	Conficker	Virus believed to have infected as many as 10 million machines; has gone through five major code versions
2010	Stuxnet	Worm attacks SCADA automated processing systems; zero-day attack
2011	Duqu	Believed to be variant on Stuxnet
2013	CryptoLocker	Ransomware Trojan that encrypts victim's data storage and demands a ransom for the decryption key

Harm from malicious code

- Harm to users and systems
  - Sending email to user contacts
  - Deleting or encrypting files
  - Modifying system information, such as the Windows registry
  - Stealing sensitive information, such as passwords
  - Attaching to critical system files
  - Hide copies of malware in multiple complementary locations
- Harm to the world
  - Some malware has been known to infect millions of systems, growing at a geometric rate
  - Infected systems often become staging areas for new infections

Transmission and propagation

- Setup and installer program
- Attached file
- Document viruses
- Autorun
- Using non-malicious programs:
  - appended viruses
  - viruses that surround a program
  - integrated viruses and replacements

Malware activation

- One-time execution (implanting)
- Boot sector viruses
- Memory-resident viruses
- Application files
- Code libraries

Virus effects

Virus Effect	How It Is Caused
Attach to executable program	• Modify file directory • Write to executable program file
Attach to data or control file	• Modify directory • Rewrite data • Append to data • Append data to self
Remain in memory	• Intercept interrupt by modifying interrupt handler address table • Load self in non-transient memory area
Infect disks	• Intercept interrupt • Intercept operating system call (to format disk, for example) • Modify system file • Modify ordinary executable program
Conceal self	• Intercept system calls that would reveal self and falsify result • Classify self as “hidden” file
Spread infection	• Infect boot sector • Infect systems program • Infect ordinary program • Infect data ordinary program reads to control its execution
Prevent deactivation	• Activate before deactivating program and block deactivation • Store copy to reinfect after deactivation

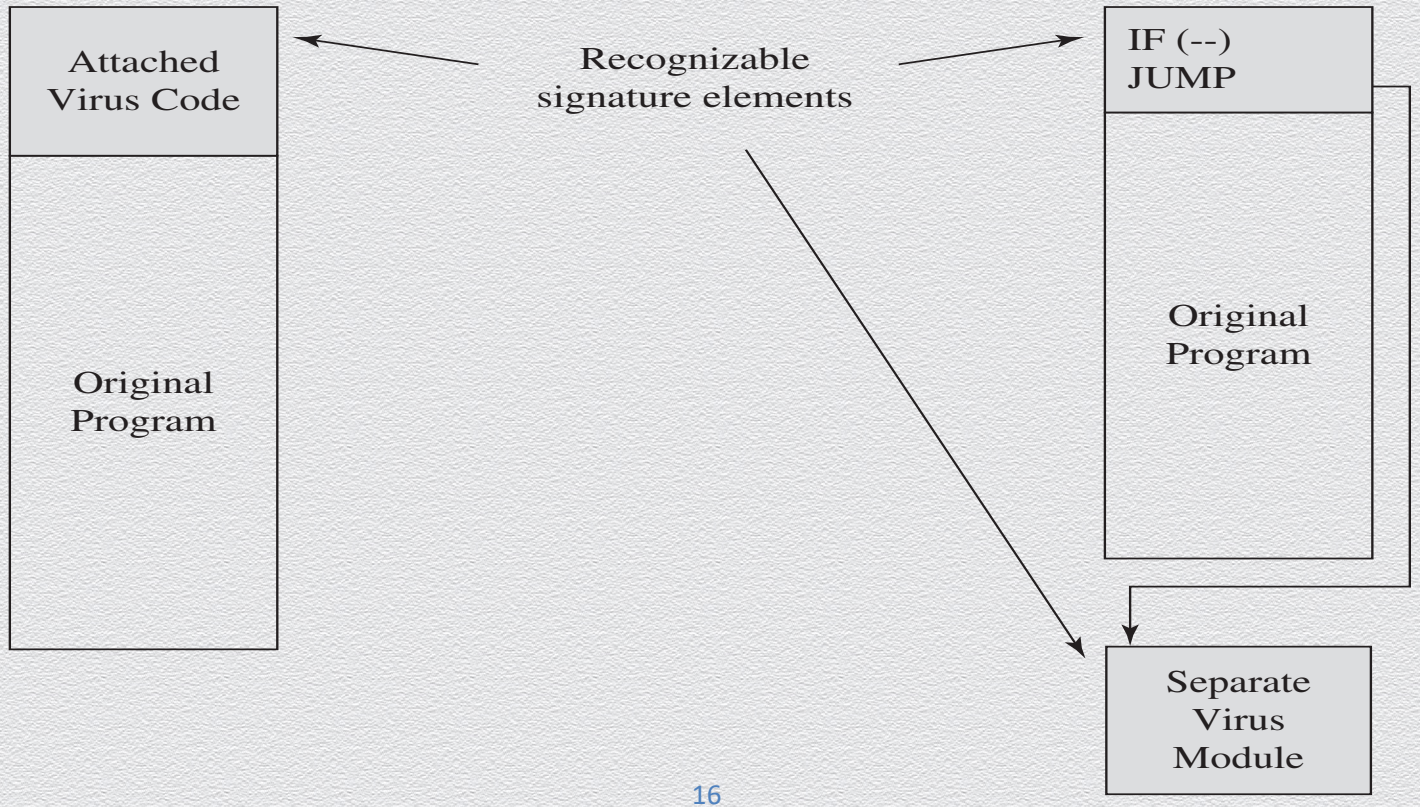
Countermeasures for users

- Use software acquired from reliable sources
- Test software in an isolated environment
- Only open attachments when you know them to be safe
- Treat every website as potentially harmful
- Create and maintain backups

Virus detection

- Virus scanners look for signs of malicious code infection using signatures in program files and memory
- Traditional virus scanners have trouble keeping up with new malware—detect about 45% of infections
- Detection mechanisms
  - Known string patterns in files or memory
  - Execution patterns
  - Storage patterns

Virus signatures



Countermeasures for developers

- Modular code: Each code module should be
  - Single-purpose
  - Small
  - Simple
  - Independent
- Encapsulation
- Information hiding
- Mutual suspicion
- Confinement
- Genetic diversity

Code testing

- Unit testing
- Integration testing
- Function testing
- Performance testing
- Acceptance testing
- Installation testing
- Regression testing
- Penetration testing

Design principles for security

- Least privilege
- Economy of mechanism
- Open design
- Complete mediation
- Permission based
- Separation of privilege
- Least common mechanism
- Ease of use

Other countermeasures

- Good
  - Proofs of program correctness—where possible
  - Defensive programming
  - Design by contract
- Bad
  - Penetrate-and-patch
  - Security by obscurity

Summary

- Buffer overflow attacks can take advantage of the fact that code and data are stored in the same memory in order to maliciously modify executing programs
- Programs can have a number of other types of vulnerabilities, including off-by-one errors, incomplete mediation, and race conditions
- Malware can have a variety of harmful effects depending on its characteristics, including resource usage, infection vector, and payload
- Developers can use a variety of techniques for writing and testing code for security

So setuid/setgid is dangerous...

setuid/setgid is dangerous...

In modern times: only for programs that really need it

- System programs that changing passwords/users, legacy programs
  - Don't do this yourself!
- Very very bad idea for shell scripts

What else can we do?

In the shell: su, sudo

- Run as another user (if you have permissions)

```
user@shell:~$ su -c "command" other user
```

- Run commands as root (or another user) based on system config file (/etc/sudoers)
  - Can restrict to specific commands, environment, ...

```
user@shell:~$ sudo whoami
root
```

```
/etc/sudoers:
%wheel ALL=(ALL) NOPASSWD: ALL

. . .
```

From man page on /etc/sudoers: (aka sudoers(5))

```
ALL             CDROM = NOPASSWD: /sbin/umount /CDROM,\
                /sbin/mount -o nosuid\,nodev /dev/cd0a /CDROM


Any user may mount or unmount a CD-ROM on the machines in the CDROM
Host_Alias (orion, perseus, hercules) without entering a password.
```

sudo has a LOT of features, see
`man sudoers` for details!

From sudo’s man page…

```
       -E, --preserve-env
                   Indicates to the security policy that the user wishes to
                   preserve their existing environment variables.  The
                   security policy may return an error
                   if the user does not have permission to preserve the
                   environment.


       --preserve-env=list
                   Indicates to the security policy that the user wishes to
                   add the comma-separated list of environment variables to
                   those preserved from the user's environment.  The
security
                   policy may return an error if the user does not have
                   permission to preserve the environment.  This option may
                   be specified multiple times.
```

Why is this better?

- Leaves the tricky code that deals with privileges to one program (sudo)
  => Maintained by professionals, like with crypto libraries

- Application developers don’t need to decide how to elevate permissions

- One common system to decide how to authenticate and set policies
  => System users/passwords, /etc/sudoers rules

CVE-2021-3156: Heap-Based Buffer Overflow in Sudo (Baron Samedit)

However, there can still be problems... eg. CVE-2021-3156 (more info)

Himanshu Kathpal, Senior Director, Product Management, Qualys Platform and Sensors.
January 26, 2021 - 12 min read

418

Last updated on: *December 23, 2022*

Sudo is a powerful utility that's included in most if not all Unix- and Linux-based OSes. It allows users to run programs with the security privileges of another user. The vulnerability itself has been hiding in plain sight for nearly 10 years. It was introduced in July 2011 (commit 8255ed69) and affects all legacy versions from 1.8.2 to 1.8.31p2 and all stable versions from 1.9.0 to 1.9.5p1 in their default configuration.

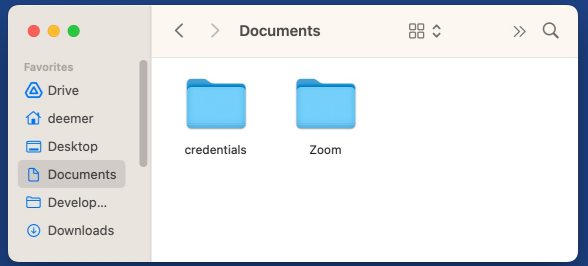
Successful exploitation of this vulnerability allows any unprivileged user to gain root privileges on the vulnerable host. Qualys security researchers have been able to independently verify the vulnerability and develop multiple variants of exploit and obtain full root privileges on Ubuntu 20.04 (Sudo 1.8.31), Debian 10 (Sudo 1.8.27), and Fedora 33 (Sudo 1.9.2). Other operating systems and distributions are also likely to be exploitable.

Taking a step back…

Is this enough?

Linux Default: Discretionary Access Control

- Owner of a resource decides on how it can be used
- Privileges depend on current user (and some groups)
- To elevate: admin user (root) vs. other users



=> How many of these can read your browser history?

…. all of them?!?!

```
deemer@ceres$ ls la ~/.mozilla/firefox/Standard/cookies.sqlite
-rw-r--r-- 1 deemer deemer 524288 Jan 10  2023 cookies.sqlite

deemer@ceres$ sqlite3 ~/.mozilla/firefox/Standard/cookies.sqlite
SQLite version 3.44.2 2023-11-24 11:41:44
Enter ".help" for usage hints.
sqlite> .tables
moz_cookies
```

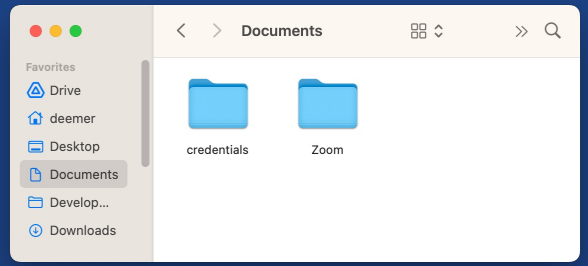
…. all of them?!?!

```
deemer@ceres$ ls la ~/.mozilla/firefox/Standard/cookies.sqlite
-rw-r--r-- 1 deemer deemer 524288 Jan 10  2023 cookies.sqlite

deemer@ceres$ sqlite3 ~/.mozilla/firefox/Standard/cookies.sqlite
SQLite version 3.44.2 2023-11-24 11:41:44
Enter ".help" for usage hints.
sqlite> .tables
moz_cookies
```

=> Just a syscall! Works as long as permissions check out 😮

```
deemer@ceres:~$ strace -- sqlite3 cookies.sqlite
. . .
access("cookies.sqlite", F_OK)          = 0
openat(AT_FDCWD, "cookies.sqlite", O_RDONLY) = 3
. . .
```



How many of these <u>should</u> be able to read your browser history?

Why?

- File permissions are very coarse
- Apps might not be trusted
- Apps might get compromised

Why?

- File permissions are very coarse
- Apps might not be trusted
- Apps might get compromised

=> Would like a more secure design: restrict application privileges so they can only access what they need

✨ ✨ ✨ ***Principle of Least Privilege*** ✨ ✨ ✨

An application should only be able to perform the operations necessary for its intended purpose

How? Depends on the context

Affects design of different systems/abstractions

One way: finer-grained permissions

Linux: can we do better than just root vs. non-root?

=> Capabilities: more precise permissions for certain actions, can be bestowed per-process

```
CAPABILITIES (7)

DESCRIPTION
       Starting with Linux 2.2, Linux divides the privileges traditionally
associated with superuser into distinct units, known as capabilities, which can be
independently enabled and  disabled
   Capabilities list

       CAP_AUDIT_WRITE (since Linux 2.6.11)
              Write records to kernel auditing log.
       CAP_NET_ADMIN
              Perform various network-related operations
       CAP_SYS_BOOT
              Use reboot(2) and kexec_load(2).
                                          . . .
```

API to start processes/threads with or without certain capabilities
=> Possible to “drop” permissions for unsafe operations
=> Once you drop permissions, process can’t get them back

```
CAPABILITIES (7)

DESCRIPTION
       Starting with Linux 2.2, Linux divides the privileges traditionally
associated with superuser into distinct units, known as capabilities, which can be
independently enabled and  disabled
   Capabilities list

       CAP_AUDIT_WRITE (since Linux 2.6.11)
              Write records to kernel auditing log.
       CAP_NET_ADMIN
              Perform various network-related operations
       CAP_SYS_BOOT
              Use reboot(2) and kexec_load(2).
                                          . . .
```

API to start processes/threads with or without certain capabilities
=> Possible to “drop” permissions for unsafe operations
=> One you drop permissions, process can’t get them back

Examples: webservers, sshd, etc.
=> Servers that operate on untrusted inputs

Another way: Process separation

- System service runs as privileged user
- Client program run by unprivileged users

Separation of processes

- System service runs as privileged user
- Client program run by unprivileged users
- Some API for how these programs communicate
  - Local network connection
  - Unix socket
  - dbus or other IPC mechanism
  - ...

One way: Separation of processes

- System service runs as privileged user
- Client program run by unprivileged users
- Some API for how these programs communicate
  - Local network connection
  - Unix socket
  - dbus or other IPC mechanism
  - ...

=> Better control over *how* privileged code runs
=> Interface between privileged/unprivileged defined more clearly

Example: docker

```
[root@ceres run]# ls -la /run/docker.sock
srw-rw---- 1 root docker 0 Jan  4 07:26 /run/docker.sock


deemer@ceres$ id
uid=1000(deemer) gid=1000(deemer) groups=1000(deemer),...,966(docker),...
```

Example: docker

```
[root@ceres run]# ls -la /run/docker.sock
srw-rw---- 1 root docker 0 Jan  4 07:26 /run/docker.sock


deemer@ceres$ id
uid=1000(deemer) gid=1000(deemer) groups=1000(deemer),...,966(docker),...
```

```
[root@ceres run]# ps aux | grep docker
root        1417  0.0  0.1 4350944 80252 ?       Ssl  Jan04  87:22
        /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
. . .

deemer     309604  0.0  0.0  12300   512 ?        S+   Feb26   0:00 /bin/bash
        /home/deemer/cs1660/env/run-container
```

One way: Isolation within OS

Linux namespaces (+ related features): give processes/users separate views of userspace components

Example: chroot (1980s)

- "Change root"
- Run command with separate root directory
- All child processes inherit this root directory

Example: chroot (1980s)

- "Change root"
- Run command with separate root directory
- All child processes inherit this root directory

- Implications?

If you need to do this in practice: look up "schroot"

One way: Isolation within OS

Linux namespaces (+ related features): give processes/users separate views of userspace components

- chroot (separate filesystem trees)
- Processes trees
- UIDs/GIDs
- cgroups (Resource limits/quotas)
- Network connections
- Time
- . . .

One way: Isolation within OS

Linux namespaces (+ related features): give processes/users separate views of userspace components

- chroot (separate filesystem trees)
- Processes trees
- UIDs/GIDs
- cgroups (Resource limits/quotas)
- Network connections
- Time
- . . .

Not a security feature *per se*, but can help...

Containers (ie, Docker) [ON LINUX]

Automated way to run applications

- Leverages lots of Linux namespaces at once
- Super great for deploying software!!

What do we notice?

- Separate filesystem
- Separate UIDs/GIDs
  - Can be root in the container => does it matter?
- Separate network interfaces, etc.

- When running the container, we decide what resources are shared with the host (files, network, etc)

Isolation mediated by Docker, OS kernel

What does this mean?

- Easy to "punch holes" depending on configuration
  - Shared directories, "privileged containers", ...
- Namespaces are growing all the time
- Docker has lots of permissions levels for what privileges containers can use

A lot of "knobs"...

- What if the configuration is incorrect?
- What if the kernel has a bug?

But…

What if the container config is incorrect?

What if the kernel has a bug?

What if you don’t trust the software you’re running?

Another way: Virtual Machines (VMs)

Isolated way to run an entire system (hardware, kernel, ...)

Another way: Virtual Machines (VMs)

Isolated way to run an entire system (hardware, kernel, ...)

- A whole OS could run as a program
- Modern systems: hardware support for isolating memory, page tables, etc. and preserving performance
  - Curious? Take CS1670.
- Virtual hardware/drivers to interact with host

Another way: Virtual Machines (VMs)

Isolated way to run an entire system (hardware, kernel, ...)

- A whole OS could run as a program
- Modern systems: hardware support for isolating memory, page tables, etc. and preserving performance
  - Curious? Take CS1670.
- Virtual hardware/drivers to interact with host

=> "Stronger" isolation, possibly more overhead for configuration/performance vs. containers

So where should we run our untrusted code?

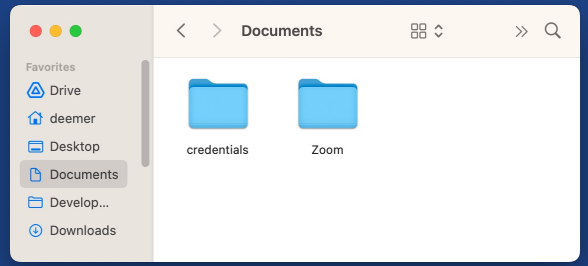
- Functionality: What privileges should the code (or the user) have?
- Threat model: What are the attacker's capabilities?

Docker on Windows, Mac?

Windows/Mac don't have Linux namespaces...

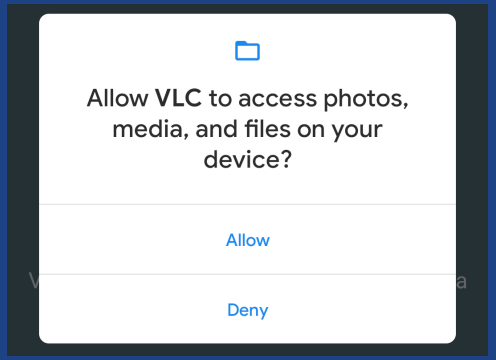
Comparing isolation mechanisms

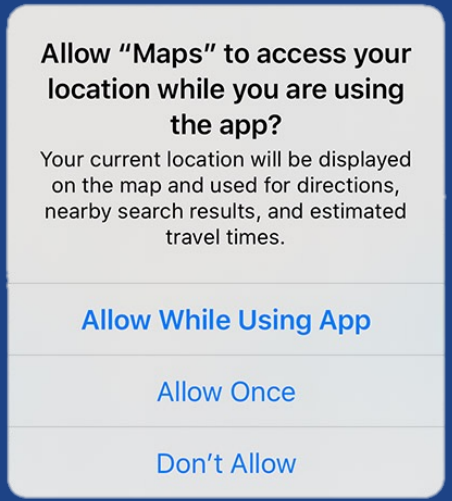
Mechanism	"Interface" to privileged operations
setuid/setgid application	Application code
Process isolation (client/server process)	API between client program and service (network protocol, socket file, IPC calls, ...)
Container	OS kernel (+ any host features turned on by container author)
VM	Virtualization Platform (hypervisor, virtual device drivers, shared folders, ...)



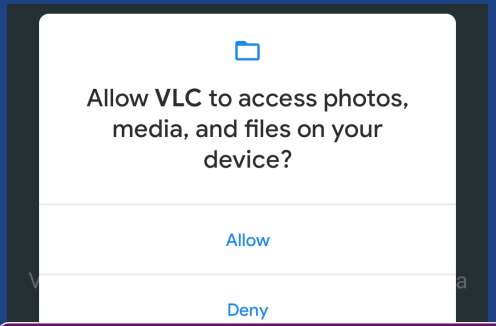
How many of these <u>should</u> be able to read your browser history?

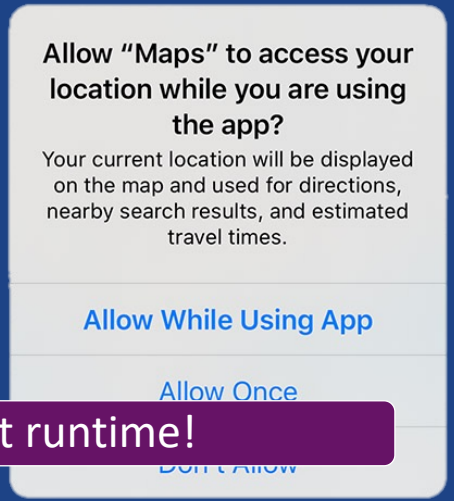
```
access("cookies.sqlite", F_OK)          = 0
openat(AT_FDCWD, "cookies.sqlite", O_RDONLY) = 3
```





```
access("cookies.sqlite", F_OK)          = 0
openat(AT_FDCWD, "cookies.sqlite", O_RDONLY) = 3
```





=> Fine-grained permissions at runtime!

…at compile time?

Other ways?

- What does it mean for the user to be "unprivileged"?
- What does it mean for code run by a user to be "unprivileged"?

- What do we want that code to be able to do?

  => How much do we trust the user? The code?

Other ways?

- What does it mean for the user to be "unprivileged"?
- What does it mean for code run by a user to be "unprivileged"?

- What do we want that code to be able to do?

  => How much do we trust the user? The code?

- sudo is pretty coarse-grained…

Viruses, Worms, Trojans, Rootkits

- Malware :
  - A software that is specifically designed to disrupt, damage, or gain unauthorized access to a computer system
  - It can be classified into several categories, depending on propagation and concealment
- Propagation
  - Virus: human-assisted propagation (e.g., open email attachment)
  - Worm: automatic propagation without human assistance
- Concealment
  - Rootkit: modifies operating system to hide its existence
  - Trojan: provides desirable functionality but hides malicious operation (i.e. *payload*)
- Various types of payloads, ranging from annoyance to crime, breaks of Confidentiality, Integrity, and Availability

A Brief History of Malware

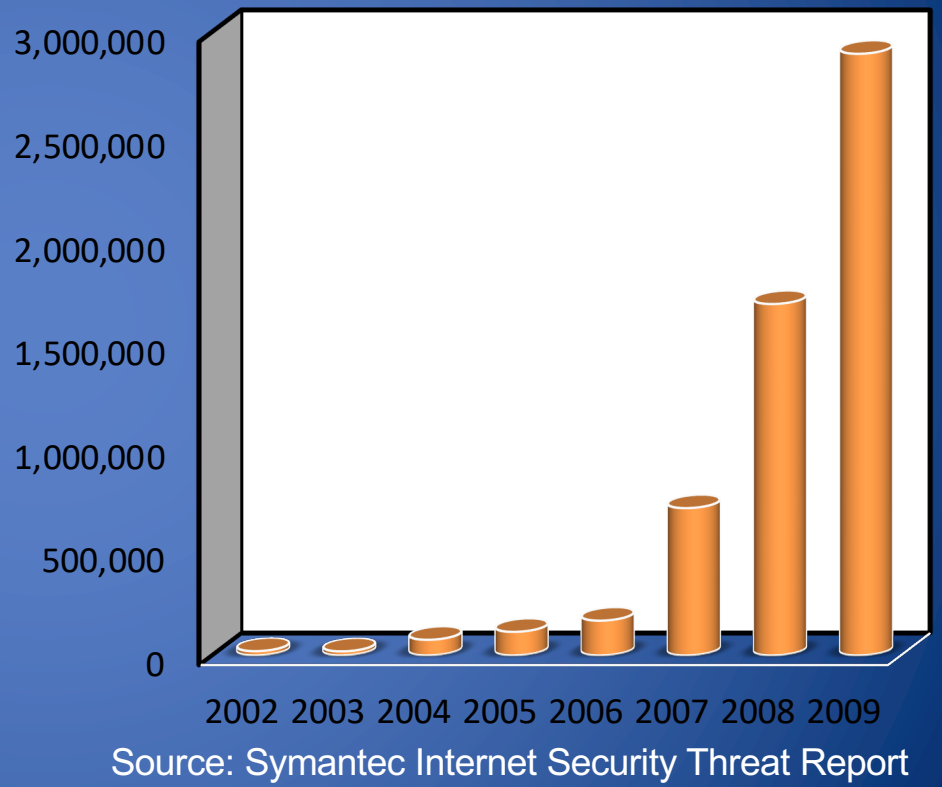
Early History

- 1972: sci-fi novel “When HARLIE Was One” features self-reproducing computer program called VIRUS
- 1982: high-school student Rich Skrenta wrote first virus released in the wild, Elk Cloner, a boot sector virus
- 1984: first academic use of “virus” by PhD student Fred Cohen, who credits advisor Len Adleman
- 1986: (c)Brain, by Basit and Amjood Farooq Alvi, credited with being first virus to infect PCs
- 1987: CHRISTMA EXEC targeting IBM VM/CMS systems was first email worm
- 1988: first internet worm, Morris Worm by Cornell student Robert Tappan Morris

Source: Wired, https://www.wired.com/2011/07/0726first-computer-fraud-indictment/

Previous Decade 2000-2009

- New malware threats have grown from 20K to 3M in the period 2002-2009
- Most of the growth has been from 2006 to 2009
- Growth in professional cybercrime and online fraud led to demand for professionally developed malware
- New malware often a custom-designed variation of known one
- Most notable:MELISSA, ILOVEYOU, CODE RED, NIMDA, etc.
- Let see the modern malwares…



Source: Symantec Internet Security Threat Report

Malware Vectors, Propagation and Concealment

Some Malware Vectors

- Compromised Legitimate Websites
  - Theft of credentials
  - Malicious downloads Mobile Apps
  - Exfiltration of personal information
  - Invasive ads
- IoT Devices
  - Rarely patched
  - Provide access to private networks of homes and offices
- Email through phishing or spamming/spoofing
  - Includes malicious links or attachments
  - Tricks users to send money or reveal passwords with social engineering
  - Mass distribution or targeted to specific users
  - About 50% of email volume is malware-related

A malware vector: Phishing

- Attempt to fraudulently acquire sensitive information
  - —Passwords, credit card numbers, etc.
- Usually copies the HTML of a website and tries to pass off as a sub-site of that page.
- Phishers create a page or e-mail (spam) that appears to be from another source
- Usually relies on the user not exploring the page in depth
- Famous phishing attempts are PayPal and eBay scams
- Examples on
  - www.phishtank.com, openphish.com

From: PayPal Security Department [service@paypal.com]
Subject: [SPAM:99%] Your PayPal Account

PayPal The way to send and receive money online

Security Center Advisory!

We recently noticed one or more attempts to log in to your PayPal account from a foreign IP address and we have reasons to belive that your account was hijacked by a third party without your authorization. If you recently accessed your account while traveling, the unusual log in attempts may have been initiated by you.

If you are the rightful holder of the account you must **click the link below** and then complete all steps from the following page as we try to verify your identity.

Click here to verify your account

http://211.248.156.177/.PayPal/cgi-bin/webscrcmd_login.php

If you choose to ignore our request, you leave us no choise but to temporaly suspend your account.

Thank you for using PayPal!

Please do not reply to this e-mail. Mail sent to this address cannot be answered. For assistance, log in to your PayPal account and choose the "Help" link in the footer of any page.

To receive email notifications in plain text instead of HTML, update your preferences here.

PayPal Email ID PP697

Protect Your Account Info

Make sure you never provide your password to fraudulent persons.

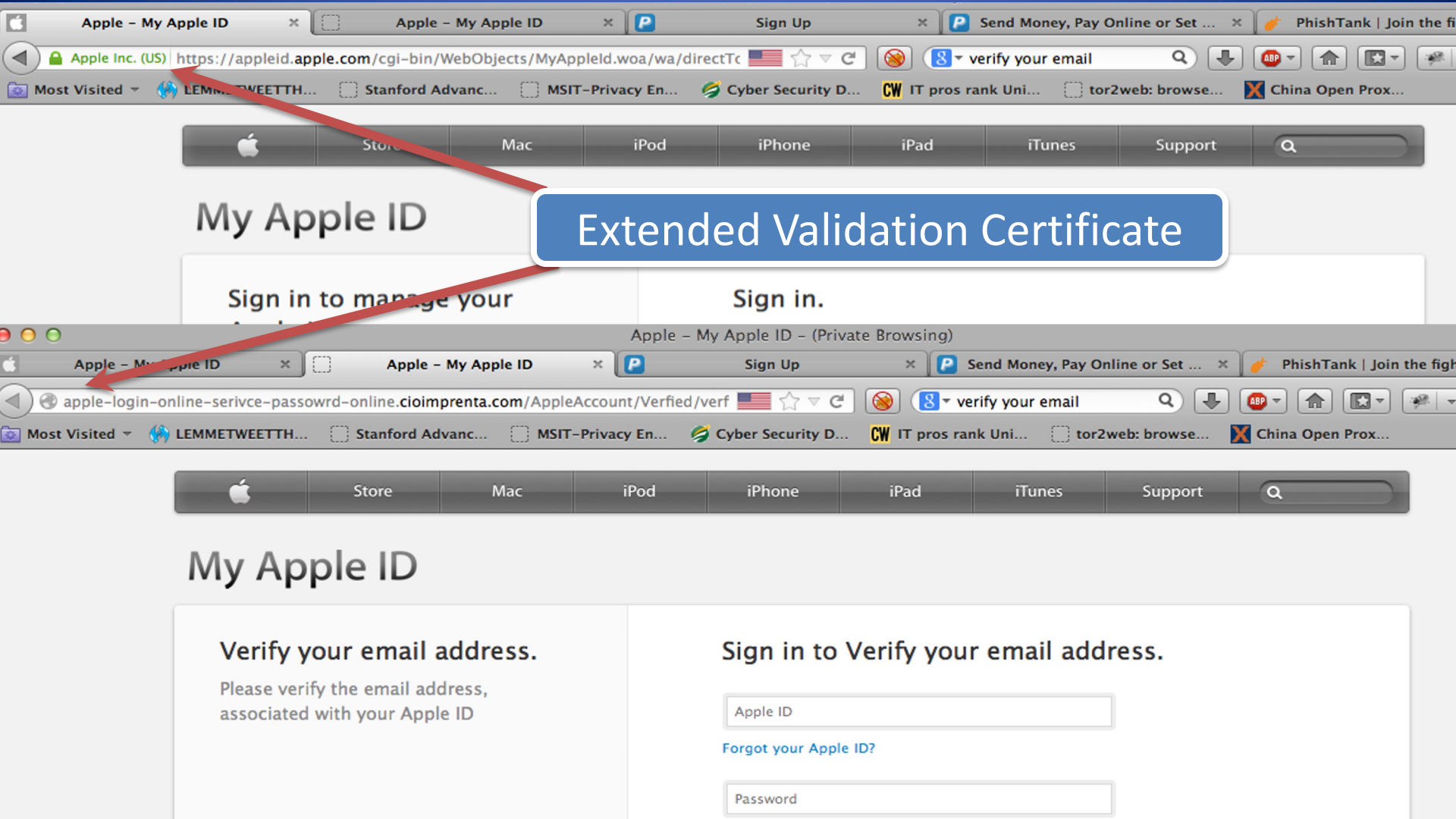
PayPal automatically encrypts your confidential information using the Secure Sockets Layer protocol (SSL) with an encryption key length of 128-bits (the highest level commercially available).

PayPal will never ask you to enter your password in an email.

For more information on protecting yourself from fraud, please review our Security Tips at http://www.paypal.com/securitytips

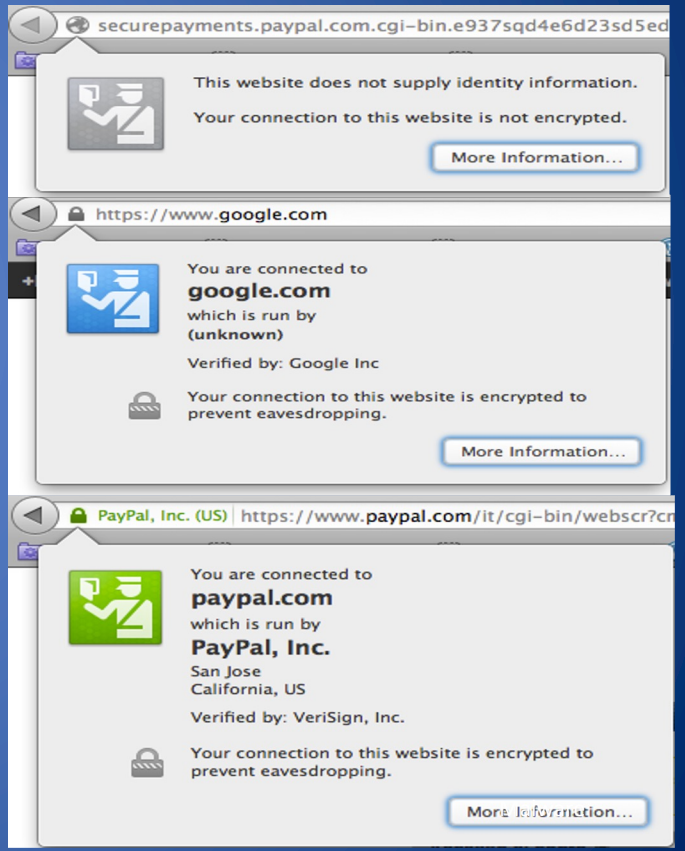
Protect Your Password

You should never give your PayPal password to anyone, including PayPal employees.


Apple – My Apple ID
Apple – My Apple ID
Sign Up
Send Money, Pay Online or Set ...
PhishTank | Join the fi
Apple Inc. (US)
https://appleid.apple.com/cgi-bin/WebObjects/MyAppleId.woa/wa/directTc
verify your email
Most Visited
Stanford Advanc...
MSIT-Privacy En...
Cyber Security D...
IT pros rank Uni...
tor2web: browse...
China Open Prox...
Store
Mac
iPod
iPhone
iPad
iTunes
Support
My Apple ID
Extended Validation Certificate
Sign in to manage your
Sign in.
Apple – My Apple ID – (Private Browsing)
Apple – My Apple ID
Sign Up
Send Money, Pay Online or Set ...
PhishTank | Join the figh
apple-login-online-serivce-passowrd-online.cioimprenta.com/AppleAccount/Verfied/verf
verify your email
Most Visited
LEMMETWEETTH...
Stanford Advanc...
MSIT-Privacy En...
Cyber Security D...
IT pros rank Uni...
tor2web: browse...
China Open Prox...
Store
Mac
iPod
iPhone
iPad
iTunes
Support
My Apple ID
Verify your email address.
Please verify the email address,
associated with your Apple ID
Sign in to Verify your email address.
Apple ID
Forgot your Apple ID?
Password

Extended Validation Certificate: Firefox

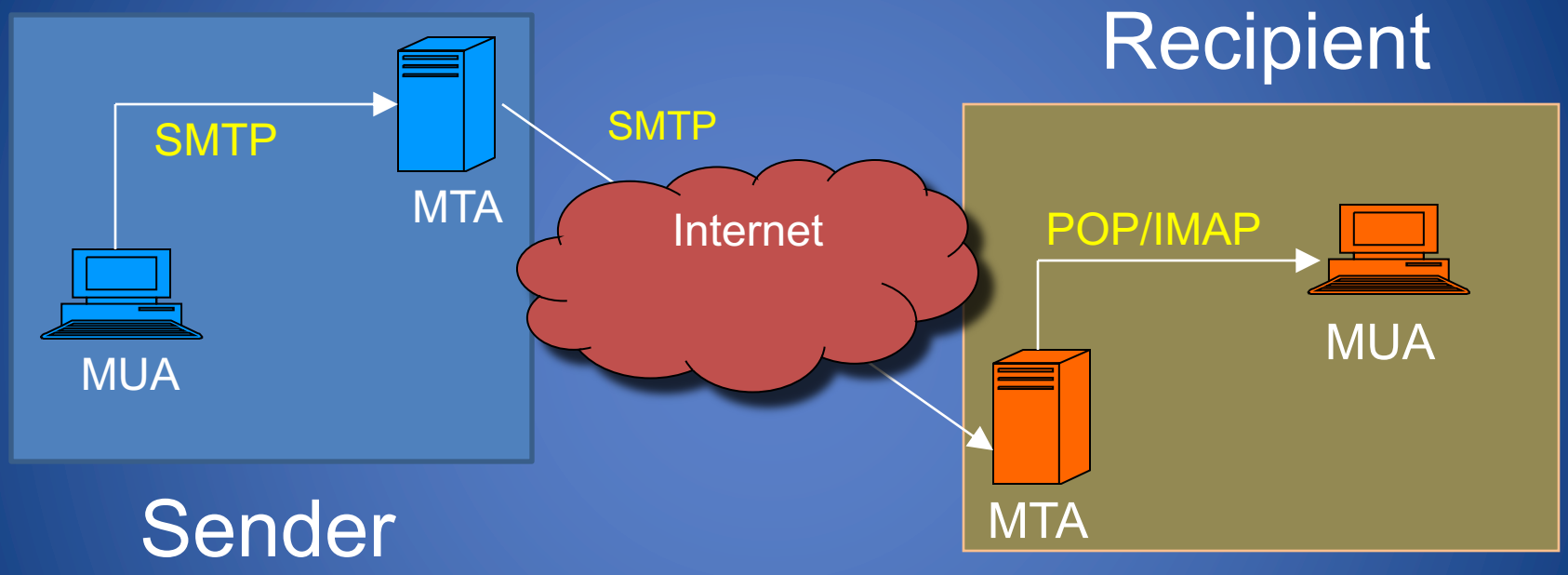
- Instant Website ID
  - —A color-coded system makes it easy to check on suspicious sites and avoid Web forgeries.
- Anti-Phishing & Anti-Malware
  - —Firefox protects you from trojan horses and spyware, and warns you away from fraudulent sites.



“why would anyone give their personal data to a phisher?”

- Spear Phishing
  - Phishing attempts directed at specific individuals or companies
  - Attackers may gather personal information about their target to increase their probability of success
- Whaling
  - Attacks directed specifically at senior executives and other high profile targets within businesses,
- These attacks are very difficult to undertstand and usually use email system

E-mail Transport



- MUA: mail user agent, *aka* mail client
- MTA: mail transport agent, *aka* mail server

SMTP

- Simple Mail Transfer Protocol
  - Client connects to server on TCP port 25
  - RFC 821 (1982) – 2821 (2001)
  - Client sends commands to server
  - Server acks or notifies of error
- Security issues
  - Sender not authenticated
  - Message and headers transmitted in plain text
  - Message and header integrity not protected
  - Spoofing and Spamming trivial to accomplish

- Example SMTP session

HELO mail.cs.brown.edu

MAIL FROM:<joe_biden@whitehouse.gov>

RCPT TO:<bernardo_palazzi@brown.edu>

DATA

Subject: Executive order

Date: Tue, March 21, 2023

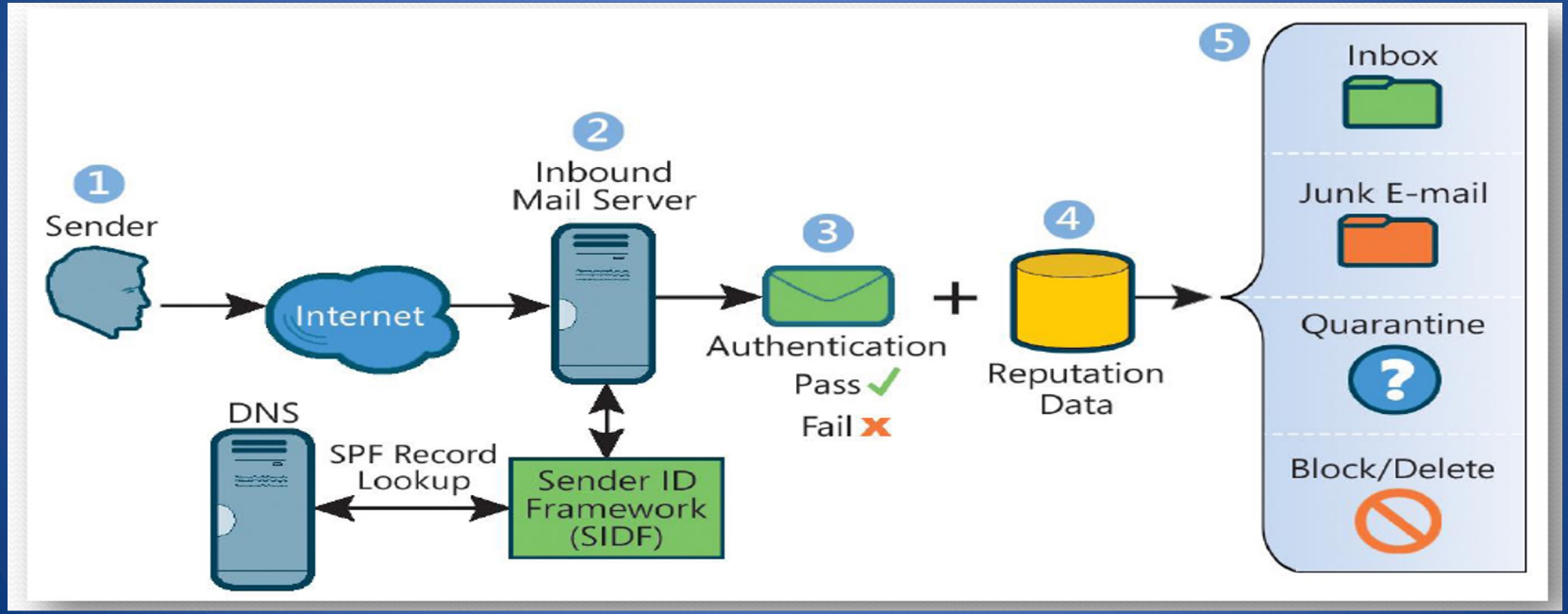
You are hereby ordered to grade all the students of CS 166 class with A.

The President of the United States

.

Sender ID and Sender Policy Framework (SPF)

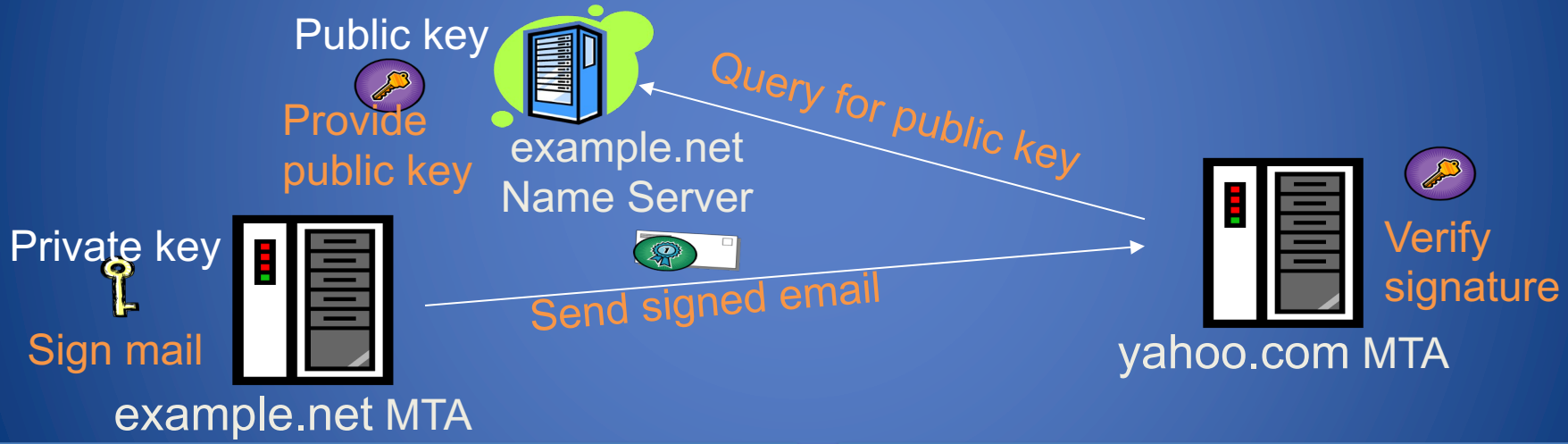
- Store DNS records about servers authorized to send mail for a given domain
- Look up domain in From header to find IP address of authorized mail server



Source: Microsoft

DomainKeys Identified Mail (DKIM)

- Sender's mail server signs email to authenticate domain
- Public key of server available in DNS record
- To be used in conjunction with other spam filtering methods

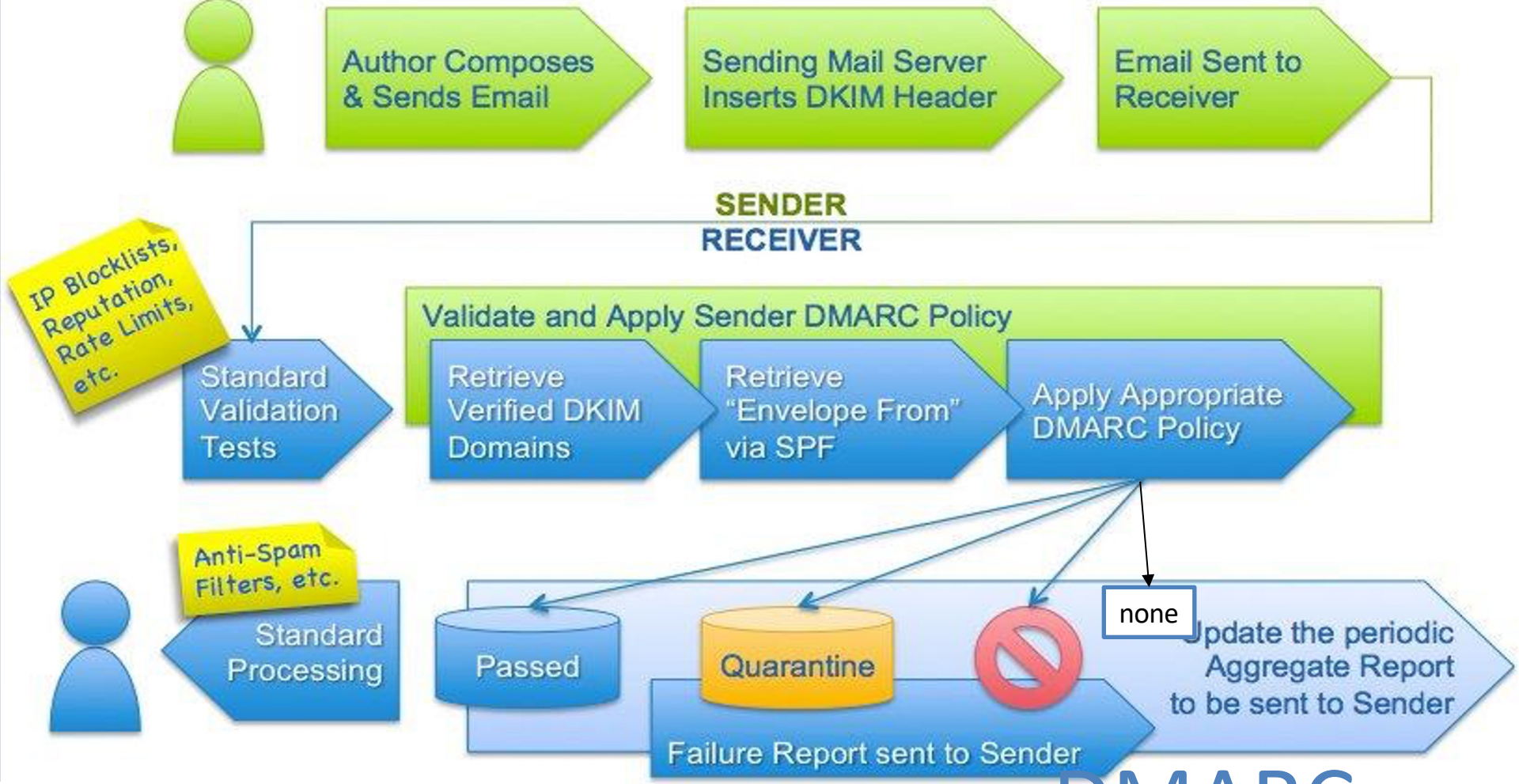


DomainKey-Signature: a=rsa-sha1; s=mail;
d=example.net; c=simple; q=dns;
b=Fg…5J

Authentication-Results: example.net
from=bob@example.net;
domainkeys=pass;

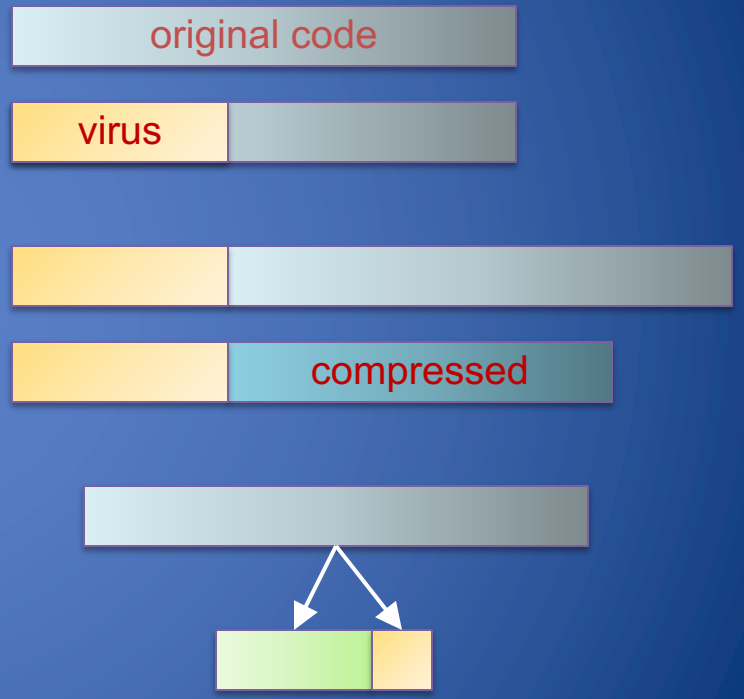
DMARC

- Domain-based Message Authentication, Reporting & Conformance
- Allows you to get reports back on the effectiveness of your SPF and DKIM investments
- Validates that the “From” header is the same as the domains validated by SPF and DKIM
- Provides clear instructions to the receiving server on what to do with emails that fail SPF or DKIM
- Google message header validator:
  - https://toolbox.googleapps.com/apps/messageheader/


Author Composes & Sends Email
Sending Mail Server Inserts DKIM Header
Email Sent to Receiver
SENDER
RECEIVER
IP Blocklists, Reputation, Rate Limits, etc.
Validate and Apply Sender DMARC Policy
Standard Validation Tests
Retrieve Verified DKIM Domains
Retrieve "Envelope From" via SPF
Apply Appropriate DMARC Policy
Anti-Spam Filters, etc.
Standard Processing
Passed
Quarantine
none
Update the periodic Aggregate Report to be sent to Sender
Failure Report sent to Sender

Infection Types

- Overwriting
  - Destroys original code
- Pre-pending
  - Keeps original code, possibly compressed
- Infection of libraries
  - Allows virus to be memory resident
  - E.g., kernel32.dll
- Macro viruses
  - Infects MS Office documents
  - Often installs in main document template



Worm Development

- Identify vulnerability still unpatched
- Write code for
  - Exploit of vulnerability
  - Generation of target list
    - Random hosts on the internet
    - Hosts on LAN
    - Divide-and-conquer
  - Installation and execution of payload
  - Querying/reporting if a host is infected
- Initial deployment on botnet

- Worm template
  - Generate target list
  - For each host on target list
    - Check if infected
    - Check if vulnerable
    - Infect
    - Recur
- Distributed graph search algorithm
  - Forward edges: infection
  - Back edges: already infected or not vulnerable

Concealment

- Encrypted virus
  - Decryption engine + encrypted body
  - Randomly generate encryption key
  - Detection looks for decryption engine
- Polymorphic virus
  - Encrypted virus with random variations of the decryption engine (e.g., padding code)
  - Detection using CPU emulator
- Metamorphic virus
  - Different virus bodies
  - Approaches include code permutation and instruction replacement
  - Challenging to detect

Rootkits

- A rootkit modifies the operating system to hide its existence
  - E.g., modifies file system exploration utilities (e.g., ls, cd, …)
  - Hard to detect using software that relies on the OS itself
- RootkitRevealer for Windows
  - By Bryce Cogswell and Mark Russinovich (Sysinternals)
  - Two scans of file system
  - High-level scan using the Windows API
  - Raw scan using disk access methods
  - Discrepancy reveals presence of rootkit
  - Could be defeated by rootkit that intercepts and modifies results of raw scan operations

What We Have Learned

- Types of malware
- Historical evolution of malware
- Modeling malware propagation:
  - phishing and email spoofing
- Concealment techniques

Ransomware

- Ransomware takes control of the victim machine as in a botnet
- Ransomware encrypts a victim's data (local hard-disk, or networked file-system)
- Attacker requests a ransom in exchange for the decryption key
  - Usually with hard-to-trace cryptocurrencies
  - No guarantee that you actually get the key
- WannaCry worm (May 2017)
  - 200,000 computers infected in 150 countries, including
    - Large network of hospitals in UK
    - Mobile carrier in Spain
    - TSMC in Taiwan
  - Propagated through an exploit in Windows 7 and older
  - Microsoft had released a security update 1 month earlier

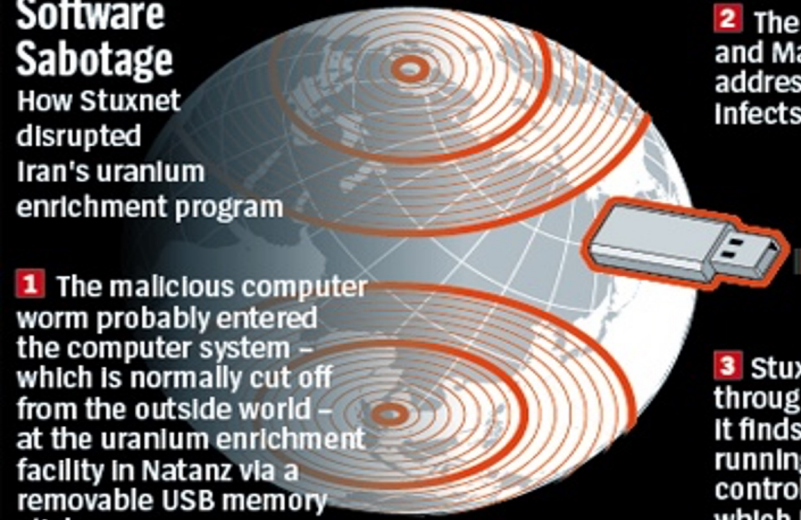
Malwares for a Cyberwar

Cyberweapons

- Starting from 2010 several viruses acted as a sort of weapons in international relationships
- Usually is not confirmed by governments
- Most famous:
  - 2010 Stuxnet
  - 2012 Flame
  - 2020 Orion Solarwinds ???

Software Sabotage

How Stuxnet disrupted Iran's uranium enrichment program

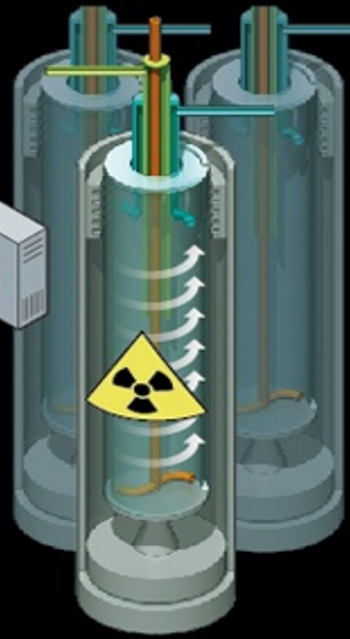


1 The malicious computer worm probably entered the computer system – which is normally cut off from the outside world – at the uranium enrichment facility in Natanz via a removable USB memory stick.

2 The virus is controlled from servers in Denmark and Malaysia with the help of two Internet addresses, both registered to false names. The virus infects some 100,000 computers around the world.

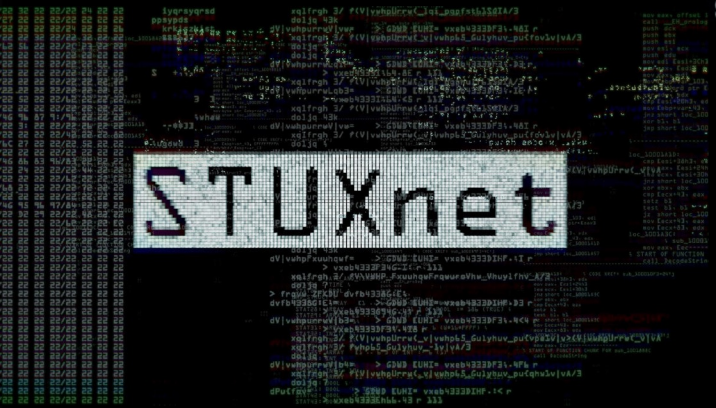
3 Stuxnet spreads through the system until it finds computers running the Siemens control software Step 7, which is responsible for regulating the rotational speed of the centrifuges.

4 The computer worm varies the rotational speed of the centrifuges. This can destroy the centrifuges and impair uranium enrichment.

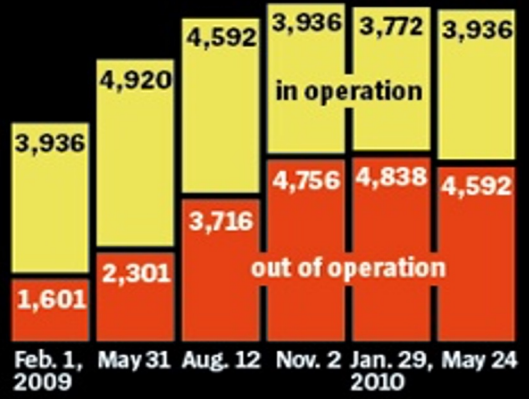


Iranian centrifuges for uranium enrichment

DER SPIEGEL



5 The Stuxnet attacks start in June 2009. From this point on, the number of inoperative centrifuges increases sharply.



Source: IAEA, ISIS, FAS, World Nuclear Association, FT research

Stuxnet: Command & Control (C&C)

- Once a system was infected Stuxnet checked two fake web domains:
  - mypremierfutbol.com
  - todaysfutbol.com
  - Registered with two fake names and credit cards
  - Servers pointed to Denmark and Malaysia
- Virus sent encrypted information about the infected target
  - Windows version
  - Internal IP address
  - Targeted Siemens sw installed
- If target does not have Siemens sw installed
  - The payload does not start
  - The worm spreads to other target

NEW "FLAME" CYBER WEAPON

Article in 2012

Kaspersky Lab, one of the world's largest makers of anti-virus software, discovered a new malware codenamed "Worm.Win32.Flame," or simply "Flame," the most complex piece of malicious software yet found.

COMPLEXITY

Comprising almost 20 MB in size and some 20 modules of code when fully deployed, Flame is one of the biggest examples of malicious software ever discovered

BREADTH

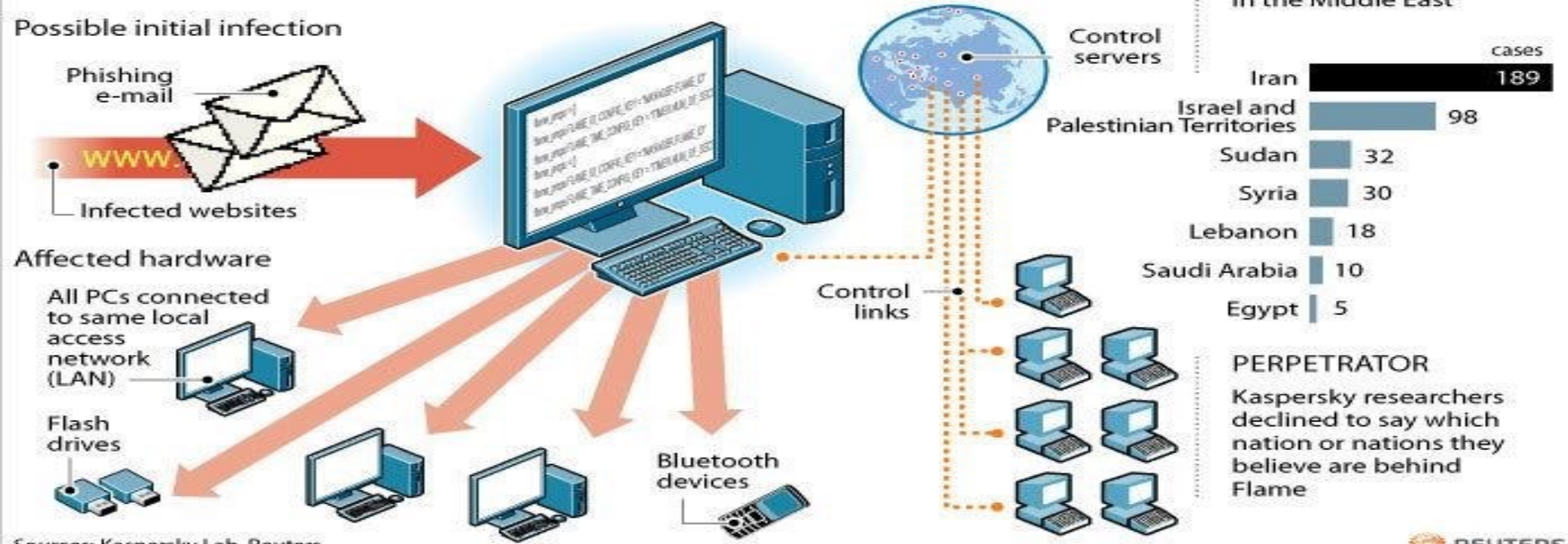
Virus can record sounds, access Bluetooth communications, capture screenshot images and log internet messaging conversations

NETWORK

The creators of the virus used a network of some 80 servers across Asia, Europe and North America to remotely control infected machines

VICTIMS

Researchers estimate that altogether between 1,000 and 5,000 machines are infected worldwide, with the larger number of infected computers found in the Middle East



PERPETRATOR

Kaspersky researchers declined to say which nation or nations they believe are behind Flame

Sources: Kaspersky Lab, Reuters

REUTERS

Flame details

- >80 Domains for C&C
- Innovative attack Vector: Flame used a rogue Microsoft signed update based on a md5 hash collision
  - More to come with Orion Solarwinds attack…
- Flame targets different office files (e.g. word, excel) and also AutoCAD
  - Usually the malware extracted 1 KB of text from each file and transmitted it back to the C&C, where there was probably a supercomputer to elaborate which file could be interesting
- Patient zero?
  - Difficult to establish, the first infection uncovered was dated December 2007 in Europe, but Flame could potentially alter the timestamp to prevent researchers from dating the work

Data breach and finance

The Equifax Breach

- Equifax is a leading credit reporting agency
- Keeps personal information and credit history for virtually every American
- In the summer of 2017, sensitive personal information about 148 million people was stolen
  - Name
  - Date of birth
  - Address
  - Social security number
- Attackers exploited vulnerability in popular web server software
  - Apache Struts code for Java web applications was vulnerable to remote code execution
  - Attacker only needed a browser
- Vulnerability had existed for years
  - Variants reported in March and September 2017
- First patch available in March 2017

The Equifax Breach One Year Later

- The CEO resigned shortly after the revelation of the breach
  - Forfeited a $3M bonus
  - Kept $18M in pension benefits
- Other executives also resigned
- The stock shed about 1/3 of its value in the following month but recouped most of the loss after one year
- No significant action taken for consumer reparation and no substantive regulatory changes since the breach
- US senators Elizabeth Warren and Mark Warner introduced a bill to hold credit agencies accountable for data theft

Source: Sentieo, Inc.

https://qz.com/1383810/equifax-data-breach-one-year-later-no-punishment-for-the-company/
http://fortune.com/2018/09/07/equifax-data-breach-one-year-anniversary/

Impossibility of Malware Detection

Undecidability

- Undecidable problem:
  A yes/no problem for which there exists no algorithm that always returns an answer.

- Halting Problem:
  Will this arbitrary program eventually return?
  Alan Turing (1936)

- We can prove that problems are undecidable.

Proof of Undecidability of the Halting Problem

- Suppose algorithm *halts(P)* can decide if any program P halts.
- We can show by contradiction that no such algorithm exists.

```
def prog():
        if halts(prog):
                loop_forever()
```

- *halts* returns True: prog loops forever
- *halts* returns False: prog terminates

Contradiction: no algorithm *halts* can exist.

Virus Detection is Undecidable

- Theoretical result by Fred Cohen (1987)
- Virus abstractly modeled as program that eventually executes infect
- Code for infect may be generated at runtime
- Proof by contradiction similar to that of the halting problem

- Suppose program isVirus(P) determines whether program P is a virus

```
def prog():
        if (not isVirus(prog)):
                infect
```

Running isVirus on the code of prog achieves a contradiction

Virus Detection is Undecidable

- Another example
- Define program prog() as:

```
{
    foo();     // harmless code
    infect
}
```

- Let's run isVirus(prog)
  - If foo() can return, isVirus should return True
  - If foo() never returns (eg infinite loop), then isVirus should return False because infect will never execute
- isVirus must determine whether foo() can ever halt. This is the **halting problem**, which is known to be undecidable.

Question

Which of the following statements summarizes what it means to say that virus detection is undecidable?

A. Virus detection is theoretically possible but exceedingly difficult to program
B. Assuming the existence of a virus detection program leads to a logical contradiction
C. Virus detection is a problem whose solution requires an exponential time algorithm
D. None of the above

Question - Answer

Which of the following statements summarizes what it means to say that virus detection is undecidable?

A. Virus detection is theoretically possible but exceedingly difficult to program
B. Assuming the existence of a virus detection program leads to a logical contradiction
C. Virus detection is a problem whose solution requires an exponential time algorithm
D. None of the above

Other Undecidable Detection Problems

- Detection of a virus
  - by its appearance
  - by its behavior
- Detection of a triggering mechanism
  - by its appearance
  - by its behavior
- Detection of a virus detector
  - by its appearance
  - by its behavior
- Detection of an evolution of
  - a known virus
  - a known triggering mechanism
  - a virus detector

Malware Detection

Detection Works Where Prevention Fails

- Detection is the act of noticing or discovering something

Detection by its appearance

- Detects specific malicious signatures
- Often uses fast pattern matching techniques
- Problems?
  - False negative
    Signature Evasion

Detection by its behavior

- Detects anomalies on a normal system/network activity
- Often uses machine learning
- Problems?
  - False positive
    Legitimate behavior could be not standard

High Cost of Errors

- False Positives FP require expensive analysis time
- False Negatives can be catastrophic
- Examples?
  - **Airport Security**: FP is when ordinary items such as keys or coins get mistaken for weapons (machine goes "beep")
  - **Quality Control**: FP is when a good quality item gets rejected, and a FN is when a poor quality item gets accepted
  - **Presumption of innocence:** "It is better that ten guilty persons FN escape than that one innocent suffer FP"
  - **Antivirus software**: a FP is when a normal file is thought to be a virus

Signatures

- Scan compares the analyzed object with a database of signatures
- A signature is a virus fingerprint
  - E.g., a string with a sequence of instructions specific for each virus
  - Different from a digital signature
- A file is infected if there is a signature inside its code
  - Fast pattern matching techniques to search for signatures
- All the signatures together create the malware database that usually is proprietary

Heuristic Analysis

- Useful to identify new and “zero day” malware
- Code analysis
  - Based on the instructions, the antivirus can determine whether or not the program is malicious, i.e., program contains instruction to delete system files,
- Execution emulation (sandbox)
  - Run code in isolated emulation environment
  - Monitor actions that target file takes
  - If the actions are harmful, mark as virus
- Heuristic methods can trigger false alarms

Online vs Offline Anti Virus Software

Online

- Free browser plug-in
- Authentication through third party certificate (i.e. VeriSign)
- No shielding
- Software and signatures update at each scan
- Poorly configurable
- Scan needs internet connection
- Report collected by the company that offers the service

Offline

- Paid annual subscription
- Installed on the OS
- Software distributed securely by the vendor online or a retailer
- System shielding
- Scheduled software and signatures updates
- Easily configurable
- Scan without internet connection
- Report collected locally and may be sent to vendor

Anti Malware Software Today

- In addition to signature-based scanning, behavior-based detection and sandboxing, anti malware software may also rely on reputation-based systems with information about current malware in the wild
- Symantec's STAR malware protection technologies rely on the following:
  - **File-Based Protection** continues to play a major protection role due to new innovations in static file heuristics.
  - **Network-Based Protection** can detect when both known and unknown vulnerabilities are used to enter a user's system.
  - **Behavior-Based Protection** looks at the dynamic behavior of malicious activity rather than static characteristics.
  - **Reputation-Based Protection** examines the meta information of a file – its age, origin, how it travels, where it exists, etc.
  - **Remediation** is a set of technologies that can help clean up an infected system.

https://searchsecurity.techtarget.com/definition/antimalware
https://www.symantec.com/theme/star

Quarantine/Virus Chest

- A suspicious file can be isolated in a folder or database called quarantine:
  - E.g., if the result of the heuristic analysis is positive and you are waiting for updates of the signatures
- The suspicious file is not deleted but made harmless: the user can decide when to remove it or eventually restore it in case of a false positive
  - Interacting with a file in quarantine is possible only through the antivirus program
- A file in quarantine is often stored encrypted to prevent its execution
- The quarantine system architecture is typically proprietary

Static vs. Dynamic Analysis

- Static Analysis
- Check the code without execution
- Filtering: scan with different antivirus and check if they return same result with different name
- Weeding: remove the correct part of files as junk to better identify the virus
- Code analysis: check binary code to understand if it is an executable
- Disassembling: check if the byte code shows something unusual

- Dynamic Analysis
- Check the execution of codes inside a virtual sandbox
- Monitor
  - File changes
  - Registry changes
  - Processes and threads
  - Network ports

How to Check if AV Software is Running?

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

- www.caro.org
- www.eicar.org

AntiVirus evaluation

- Shield
  - Background process (service/daemon)
  - Scans each time a file is touched (open, copy, execute, etc.)
- On-demand
  - Scan on explicit user request or according to regular schedule
  - On a suspicious file, directory, drive, etc.

Performance test of scan techniques

- Comparative/Performance: check the number of already known viruses that are found and the time to perform the scan
- False alarm test: number of false viruses detected
- Heuristic / Behaviour Tests: measure the proactive protection capabilities

Anti-viruses are ranked using both parameters: http://www.av-comparatives.org/

Resources

- Symantec's Internet Security Threat Report
  - —Published annually
- Countdown to zero day by Kim Zetter, 2014
- Art of Computer Virus Research and Defense by Peter Szor
- http://virus.wikidot.com/

Responsible Disclosure

- What happens if someone discovers a vulnerability in software?
  - 2008: MBTA sued three MIT students to prevent them from giving a talk about vulnerabilities in the subway fare system
  - 2019: researcher Jonathan Leitschuh discovered a vulnerability in Zoom, which they did not fix until he publicly disclosed it
- Today, many companies have bug bounty programs in place to encourage responsible disclosure of vulnerabilities
- Disclosure deadlines: amount of time researchers give companies to patch vulnerabilities before disclosure
  - Often varies by company and by how critical the vulnerability is

SolarWinds Hack

- **What happened?**
  - Texas-based IT management company
  - hackers able to compromise networks of many other companies and deliver malware
    - supply-chain attack
  - malware inserted into update of Orion system
    - Orion: allows companies to see what is going on in their network
    - Hackers used AWS as a disguise
  - White house says at least 100 companies impacted (Microsoft included) + US government agencies
- **Response**
  - FireEye, a cybersecurity company impacted discovered the hack
  - SolarWinds issued a security advisory + what defensive measures could be taken
  - FBI Investigation to find the actors
- **Why did this happen?**
  - Bad security practices
    - “solarwinds123” used as a password for secure server (security researcher already warned SolarWinds of this!)

Underinvestment in Cybersecurity

- Why were basic security practices not followed at SolarWinds?
- “Employees say that under [CEO] Mr. Thompson ... every part of the business was examined for cost savings and **common security practices were eschewed because of their expense**. His approach helped almost triple SolarWinds’ annual profit margins to more than $453 million in 2019 from $152 million in 2010.”
- Bruce Schneier: “The market does not reward security, safety or transparency. It doesn't reward reliability past a bare minimum, and it doesn't reward resilience at all.”
- Core problem: limited economic incentives to invest in cybersecurity
  - Expense with diminishing returns
  - Limited legal liability
  - Small factor in customers’ decisions => small effect on share price
  - Supply chain security

Investment under ideal circumstances

- Gordon-Loeb model: even with optimal incentives, firms will never invest more than 37% of expected damage from security breaches in cybersecurity
- Caused by cybersecurity not generating profit and having diminishing returns on investment
- If you expect fire damage to cause $10,000 damage, it doesn't make sense to purchase a $10,000 device that reduces the probability of fire damage
- Result: total damage caused by cybersecurity will always significantly exceed investment in cybersecurity

Legal liability

- Having poor cybersecurity is legal
- Limited laws regulating cybersecurity standards
- Federal Trade Commission relies on “unfair or deceptive acts” to press charges
- Customers and shareholders need extreme cases of negligence or false statements
  - Class-action lawsuit against SolarWinds by shareholders, but only because they allege false and misleading statements
- As long as an honest effort is made, very little legal risk in having bad cybersecurity

Lack of business consequences

- Over time, customers tend to forgive and forget data breaches
- Equifax, eBay, Adobe, and Marriott all recovered from their breaches
- In corporate context, incentives in procurement favour functionality and cost over possible cybersecurity risks
- Difficult to evaluate cybersecurity between companies
- Share prices usually drop heavily after a data breach, but studies show a negligible long-term effect
- More recent data breaches have had smaller share price drops due to “breach fatigue”

Supply chain issues

- Centralization of software has created points of vulnerability that dramatically reduce hacker effort
  - SolarWinds allowed hackers to access 18,000 systems
- Vulnerabilities in one company's product have cascading effects beyond their immediate customers
  - CISA: 30% of SolarWinds victims did not use SolarWinds
- Example: 2017 NotPetya attack
  - Malware deployed by a malicious automatic update in MeDoc, Ukrainian tax preparation software
  - Caused $10 billion damage
  - Damaged pharmaceutical production, global shipping, hospital systems

What We Have Learned

- Types of malware
- Historical evolution of malware
- Modeling malware propagation
- Concealment techniques
- Undecidability of malware detection
- Heuristic techniques for malware detection