CSE508 Network Security

Stony Brook University

2024-01-30

Threat Landscape and Basic Security Principles

Michalis Polychronakis

Stony Brook University

Threats, Vulnerabilities, and Attacks

A ***threat*** is a potential cause of an incident, malicious or otherwise, that could harm an asset

- Loss of service, compromise of information or functions, technical failure, …
- Different origins: deliberate, accidental, environmental, …

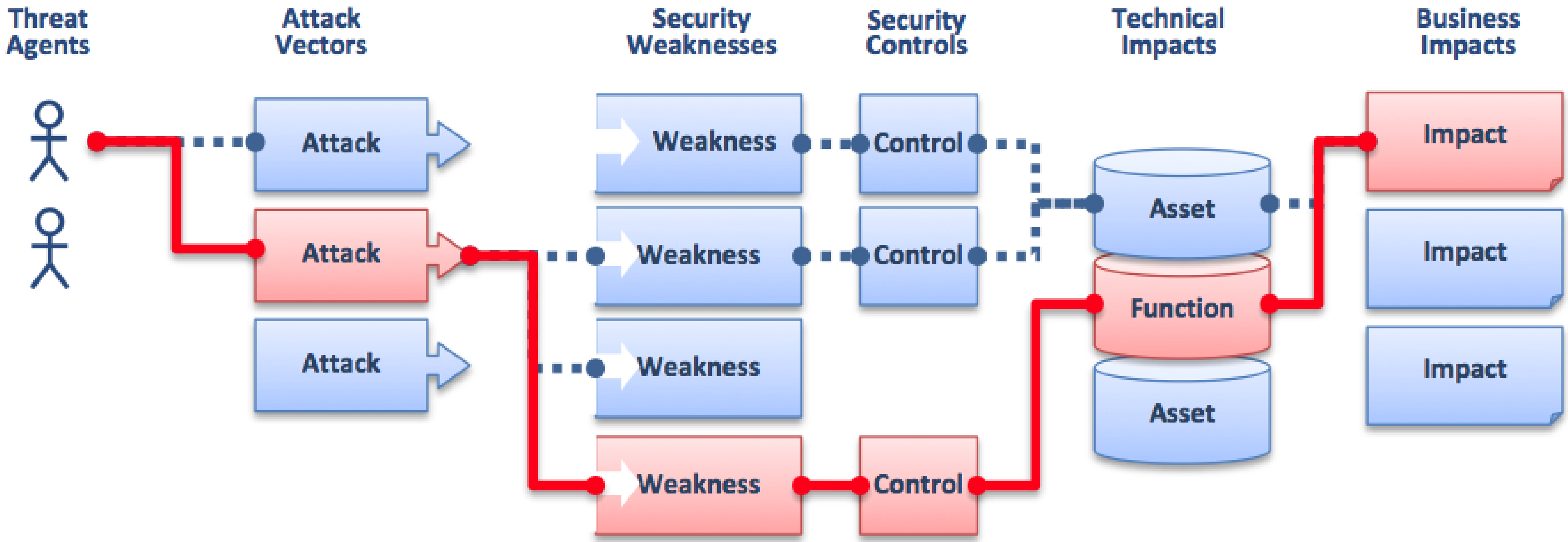
A ***vulnerability*** is a weakness that makes a threat possible

- Hardware, software, protocol, network, personnel, physical, organizational, …

An ***attack*** is an action that exploits a vulnerability or enacts a threat

- *Active* (observable effect) vs. *passive* (imperceptible collection of information)
- *Inside* (initiated by an authorized "insider" entity) vs. *outside* ("outsider" entity)

Threats, Vulnerabilities, and Attacks



© Neil Smithline – https://commons.wikimedia.org/wiki/File:2010-T10-ArchitectureDiagram.png

Threat Classification

Example classification scheme: Microsoft's STRIDE

Spoofing: *TCP/IP, identity, HTTP headers, email address, poisoning, …*

Tampering: *network traffic, code, HTTP cookies/URLs/parameters, …*

Repudiation: *deniability, audit log scrubbing/modification, …*

Information disclosure: *unauthorized data access, data leakage, …*

Denial of Service: *crashing, flooding, resource stagnation, …*

Elevation of privilege: *gain admin access, jailbreaking, …*

Risk Assessment

Example risk assessment scheme: Microsoft’s DREAD

Damage: *how bad would an attack be?*

Reproducibility: *how easy is it to reproduce the attack?*

Exploitability: *how much work is it to launch the attack?*

Affected users: *how many people will be impacted?*

Discoverability: *how easy is it to discover the threat?*

Threat Model

Assumptions about possible attacks a system tries to protect against

- Understanding potential threats is crucial for taking appropriate measures

Various threat modeling approaches: attacker-centric, software-centric, asset-centric, …

Example: data flow approach

- *View the system as an adversary:* identify entry/exit points, assets, trust levels, defenses, usage patterns, …
- *Characterize the system:* identify usage scenarios, roles, objectives, components, dependencies, security alerts, implementation assumptions, …
- *Identify threats:* what can the attacker do? How? What is the associated risk? How can the respective vulnerabilities be resolved?

Threat Actors

’90s: script kiddies

’00s: criminals

’10s: states *(OK, much earlier, but now we talk about it)*

Different motives

$$$$$$$$$$$

Honest but curious individuals

Political or social ends

Bribed or angry insiders

Espionage/sabotage/military

Different resources: $$$$, skills, infrastructure → **Know your enemy!**

Policies and Mechanisms

Threat model ➔ security policy ➔ security mechanisms

Security policy: a definition of what it means for a system/organization/entity to be secure

- Access control, information flow, availability, …
- Computer, information, network, application, password, …

Enforced through security mechanisms

- Prevention: *antivirus, firewall, email filtering, 2-factor authentication, …*
- Detection: *intrusion detection system (IDS/IPS, SIEM, EDR), honeypots, …*
- Recovery: *backup, forensics, configuration management, software provisioning, …*
- Awareness: *training, monitoring, asset inventory, …*

Vulnerability

“A property of a system or its environment which, in conjunction with an internal or external threat, can lead to a security failure, which is a breach of the system’s security policy.” [Anderson]

Various classifications based on…

SDL: design, implementation, operation, maintenance

Abstraction level: low vs high level, OSI network layers, system vs. process, hardware/firmware/OS/middleware/application, …

Type of error/condition/bug: memory errors, range and type errors, input validation, race conditions, synchronization/timing errors, access-control problems, environmental/system problems (e.g., authorization or crypto failures), protocol errors, logic flaws, …

Disclosure process: zero-day vs. known, private vs. public, “coordinated” vs. full disclosure, …

Multiple vulnerabilities are often combined for a single purpose

Vulnerability (Another Definition)

"The intersection of a system susceptibility or flaw, access to the flaw, and the capability to exploit the flaw." [AFRL ATSPI]

System Susceptibility: focus on what's critical

Reduce access points to only those that are absolutely necessary

Access to the flaw: move it out of band

Make critical access points and associated security elements less accessible to the adversary

Capability to exploit the flaw: prevent, detect, react

Appropriate response upon detection of an attack

Related term: ***attack surface***

The different points through which an attacker can interact with the system/environment

Zero-Day Vulnerabilities/Exploits

A previously unknown vulnerability discovered before the vendor becomes aware of it

0-day exploits: become known once they are detected "in the wild"

- Vendors then rush to release a patch
- Intrusion detection systems are updated to detect the threat

N-day exploits: developed immediately after a patch is released

- Vulnerability is discovered and fixed by the vendor (not the attacker)
- Once the patch is released, attackers reverse engineering it (e.g., using binary diffing) to build an exploit
- "Window of vulnerability" is open until most vulnerable systems are patched

Intrusions

“Any set of actions that attempt to compromise the integrity, confidentiality or availability of information resources” [Heady et al.]

“An attack that exploits a vulnerability which results to a compromise of the security policy of the system” [Lindqvist and Jonsson]

Most intrusions…

- Are carried out remotely
- Exploit software vulnerabilities
- Result in arbitrary code execution or unauthorized data access on the compromised system

Attack Source

Local

Unprivileged access ➔ privilege escalation

Physical access ➔ I/O ports (launch exploits), memory (cold boot attacks), storage (just remove it), shoulder surfing (steal credentials), dumpster diving (steal information), bugging (e.g., keylogger, antennas/cameras/sensors, HW parts), …

Remote

Internet

Local network (Ethernet, WiFi, cellular, bluetooth, NFC, …)

Phone (social engineering, SMS, cellular interception, …)

Infected media (~~disks~~, ~~CD-ROMs~~, USB sticks, …)

Pre-infected SW/HW components (libraries, third-party services, BIOS, NIC, router, …)

Intrusion Method

Social engineering (phishing, spam, scareware, …)

Viruses (~~disks~~, ~~CD-ROMs~~, USB sticks, downloads, …)

Network traffic interception (access credentials, keys, phishing, …)

Password guessing/leakage (brute force, root:12345678, …)

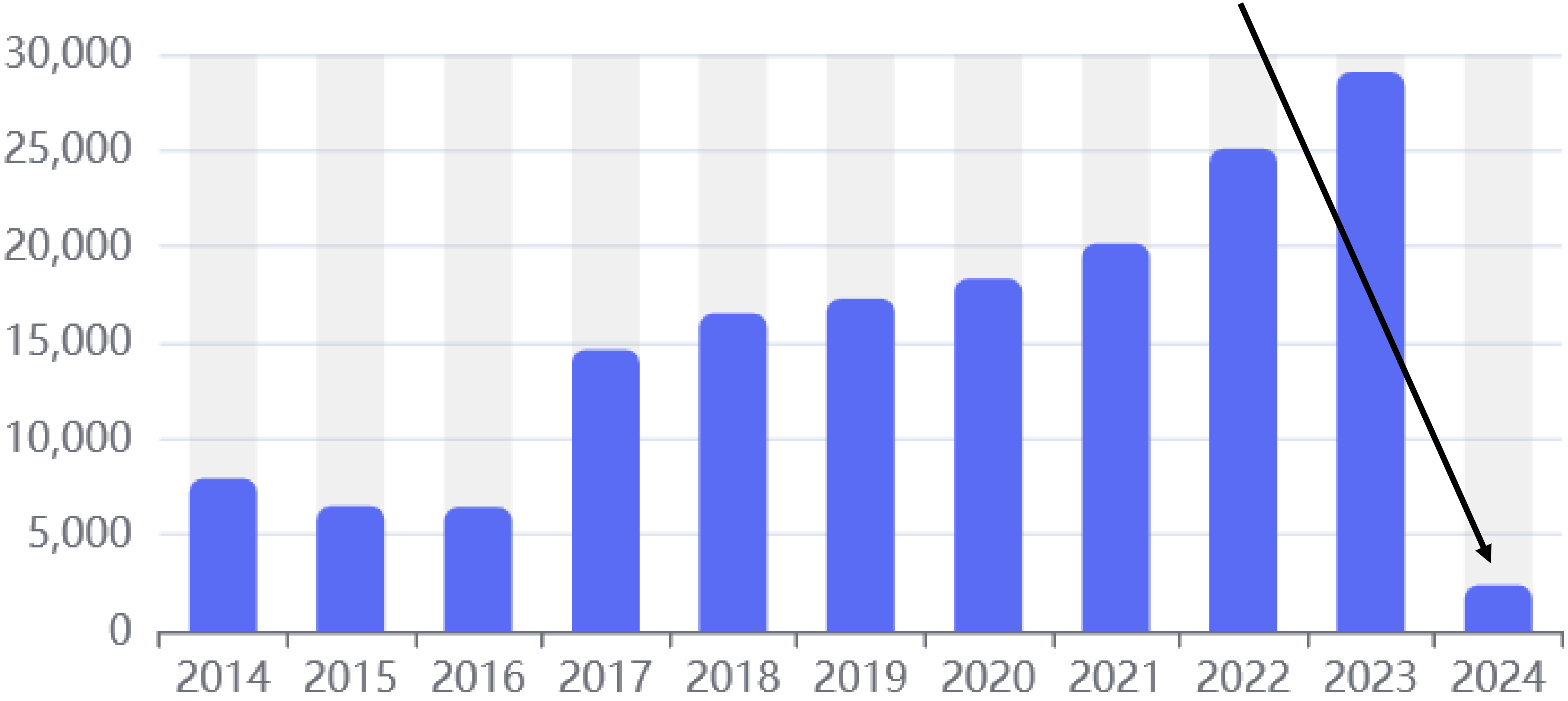
Physical access (reboot, keylogger, screwdriver, …)

Supply chain compromise (backdoor, infected update, …)

Software vulnerability exploitation

…

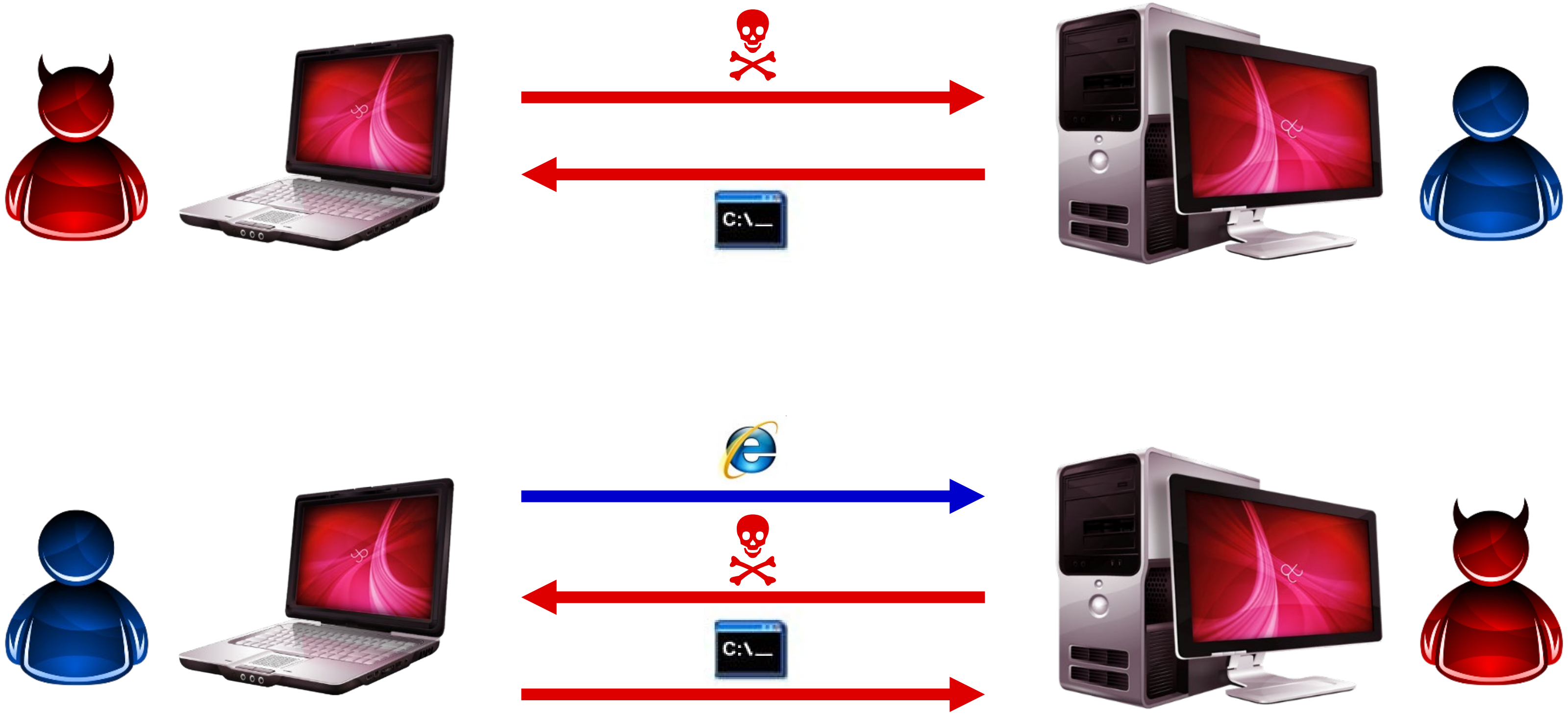
Number of CVEs by year



Just this past month: 2,392 CVEs

© CVE Details – https://www.cvedetails.com/browse-by-date.php

Remote Exploitation: Server-side vs. Client-side



’00s: Server-side Exploitation

Web, database, email, and other servers

PC OS/software included many default Internet-accessible services

- Windows XP: 135 (RPC), 137–139 (NetBIOS over TCP/IP), 445 (SMB), …
- **No firewall (!)** Default firewall was enabled in Windows XP SP2 in 2004

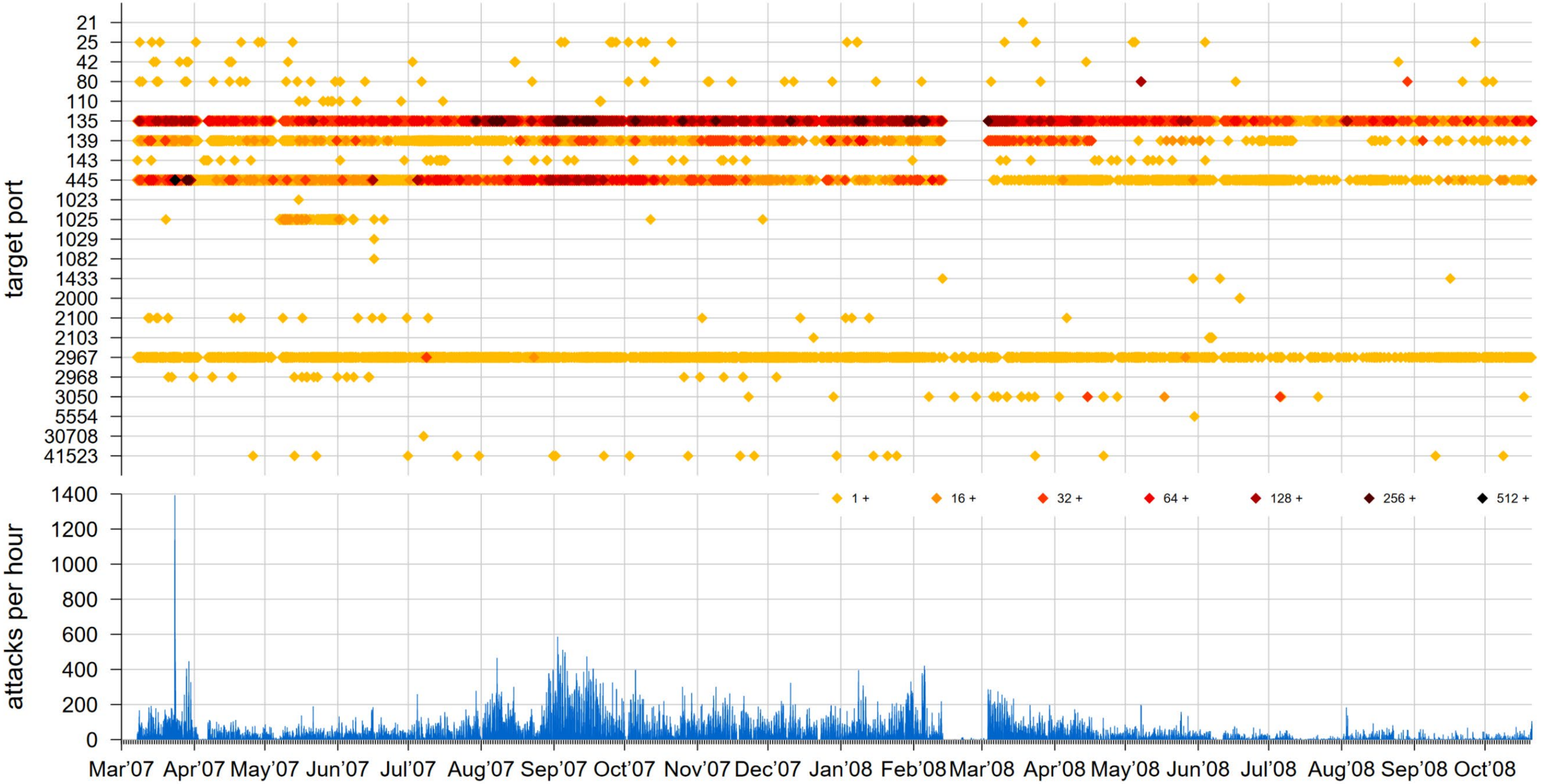
’10s: Client-side Exploitation

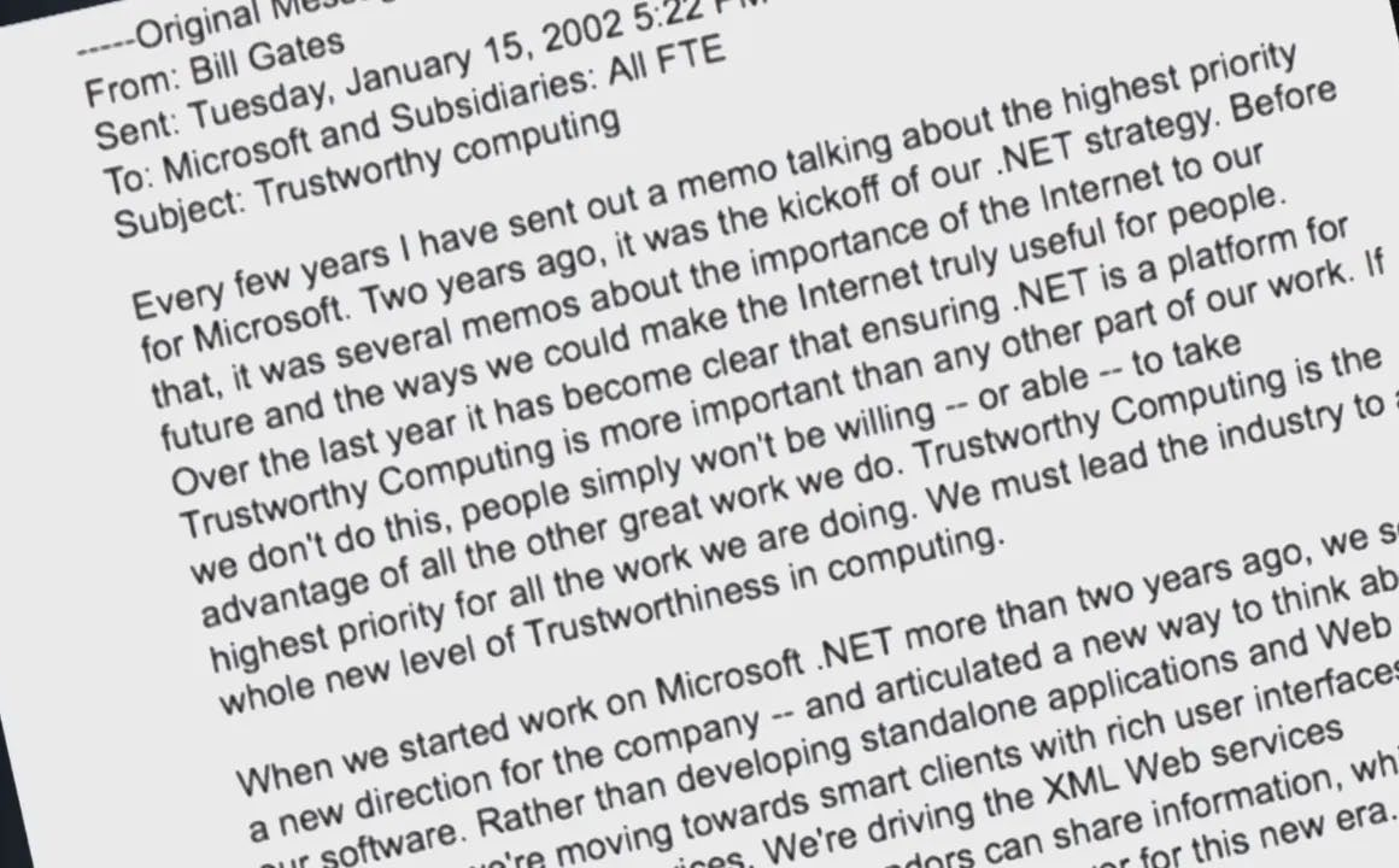
PC OS/software services gradually became unreachable from outside

- Operating systems shipping with strict firewall configurations
- Residential networks started using NAT (network address translation)

Attackers moved to *client-side* software that receives untrusted input

- Browsers, PDF viewers, office applications, media players, …


target port
21
25
42
80
110
135
139
143
445
1023
1025
1029
1082
1433
2000
2100
2103
2967
2968
3050
5554
30708
41523
attacks per hour
1400
1200
1000
800
600
400
200
0
◆ 1 +
◆ 16 +
◆ 32 +
◆ 64 +
◆ 128 +
◆ 256 +
◆ 512 +
Mar'07
Apr'07
May'07
Jun'07
Jul'07
Aug'07
Sep'07
Oct'07
Nov'07
Dec'07
Jan'08
Feb'08
Mar'08
Apr'08
May'08
Jun'08
Jul'08
Aug'08
Sep'08
Oct'08

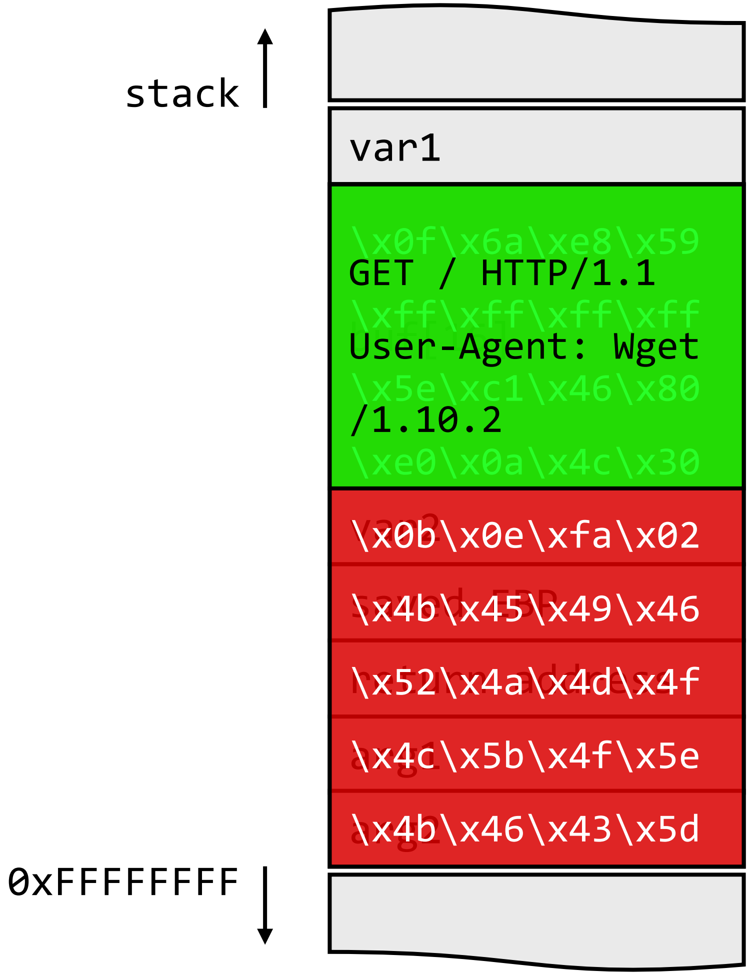


-----Original Mes
From: Bill Gates
Sent: Tuesday, January 15, 2002 5:22 P
To: Microsoft and Subsidiaries: All FTE
Subject: Trustworthy computing

Every few years I have sent out a memo talking about the highest priority for Microsoft. Two years ago, it was the kickoff of our .NET strategy. Before that, it was several memos about the importance of the Internet to our future and the ways we could make the Internet truly useful for people. Over the last year it has become clear that ensuring .NET is a platform for Trustworthy Computing is more important than any other part of our work. If we don't do this, people simply won't be willing -- or able -- to take advantage of all the other great work we do. Trustworthy Computing is the highest priority for all the work we are doing. We must lead the industry to a whole new level of Trustworthiness in computing.

When we started work on Microsoft .NET more than two years ago, we set a new direction for the company -- and articulated a new way to think about our software. Rather than developing standalone applications and Web
we're moving towards smart clients with rich user interfaces
ices. We're driving the XML Web services
ndors can share information, while
er for this new era.

(Very Simple) Buffer Overflow Exploitation



stack

var1

\x0f\x6a\xe8\x59
GET / HTTP/1.1
\xff\xff\xff\xff
User-Agent: Wget
\x5e\xc1\x46\x80
/1.10.2
\xe0\x0a\x4c\x30

\x0b\x0e\xfa\x02

\x4b\x45\x49\x46

\x52\x4a\x4d\x4f

\x4c\x5b\x4f\x5e

\x4b\x46\x43\x5d

0xFFFFFFFF

← **Code injection**

Shellcode

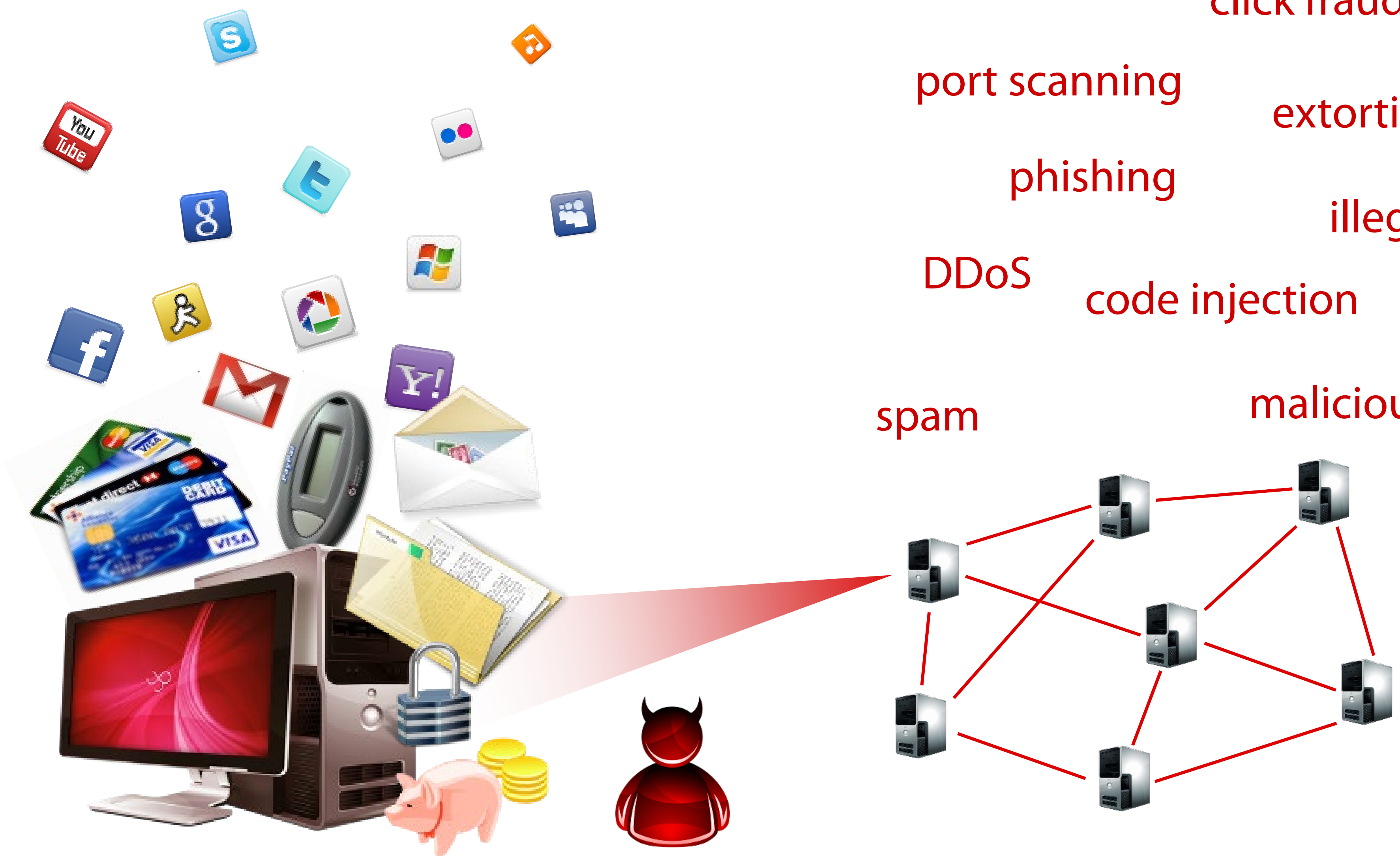
spawn shell

listen for connections

add user account

download and execute malware

Malware and Botnets



click fraud

port scanning

extortion

phishing

illegal content

DDoS

code injection

spam

malicious websites

Basic Phases of a Typical Targeted Attack

Reconnaissance and information gathering

Exploitation

Privilege Escalation

Persistent access

Internal reconnaissance

Lateral movement

Data exfiltration/damage/other goal

…subject of future lectures

Many more threats…

Password attacks

Information leakage

Repudiation

Privilege escalation

Information gathering

Session hijacking

Supply chain attacks

Social engineering

Denial of service

Tampering

Information disclosure

Sniffing

Spoofing

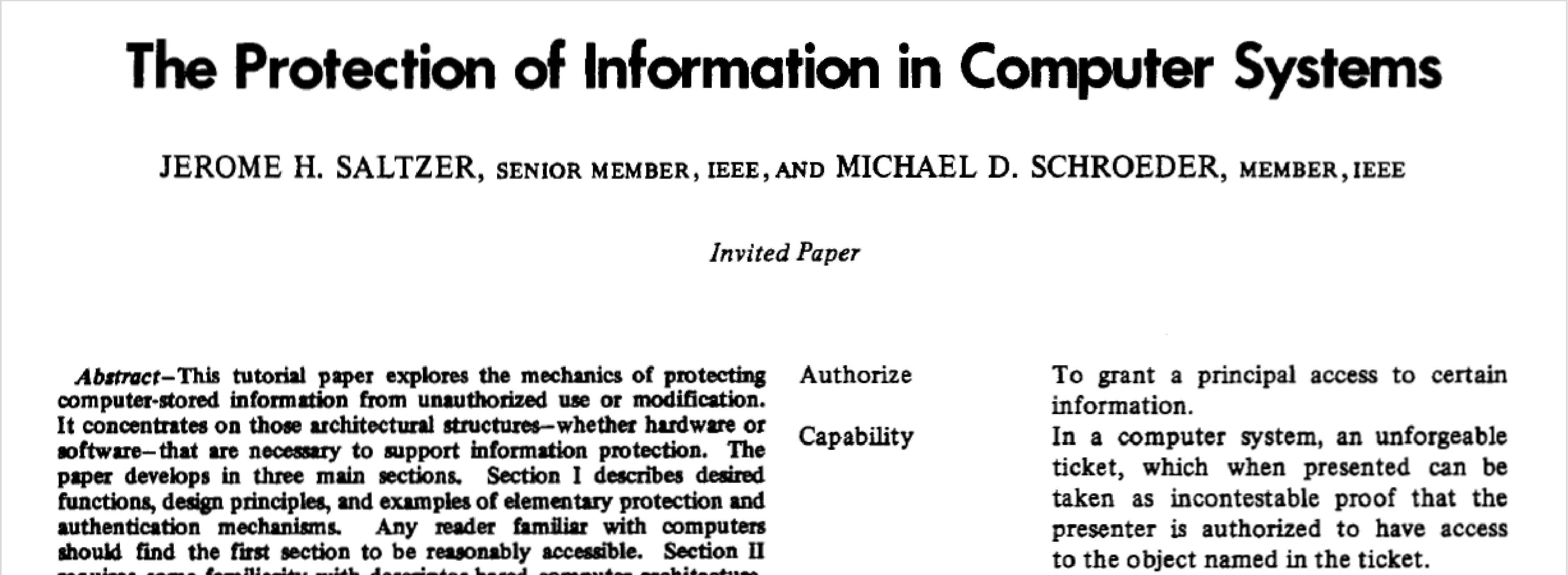
…subject of future lectures

Basic Security Principles

In the 1970s, Saltzer and Schroeder had been working on Multics

Identified a set of design principles intended to help designers of time-sharing operating systems protect information

- Some of the earliest thinking on building secure systems



The Protection of Information in Computer Systems

JEROME H. SALTZER, SENIOR MEMBER, IEEE, AND MICHAEL D. SCHROEDER, MEMBER, IEEE

Invited Paper

Abstract–This tutorial paper explores the mechanics of protecting computer-stored information from unauthorized use or modification. It concentrates on those architectural structures–whether hardware or software–that are necessary to support information protection. The paper develops in three main sections. Section I describes desired functions, design principles, and examples of elementary protection and authentication mechanisms. Any reader familiar with computers should find the first section to be reasonably accessible. Section II requires some familiarity with descriptor-based computer architecture

Authorize — To grant a principal access to certain information.

Capability — In a computer system, an unforgeable ticket, which when presented can be taken as incontestable proof that the presenter is authorized to have access to the object named in the ticket.

Economy of Mechanism

Economy of Mechanism

Security mechanisms should be as simple as possible

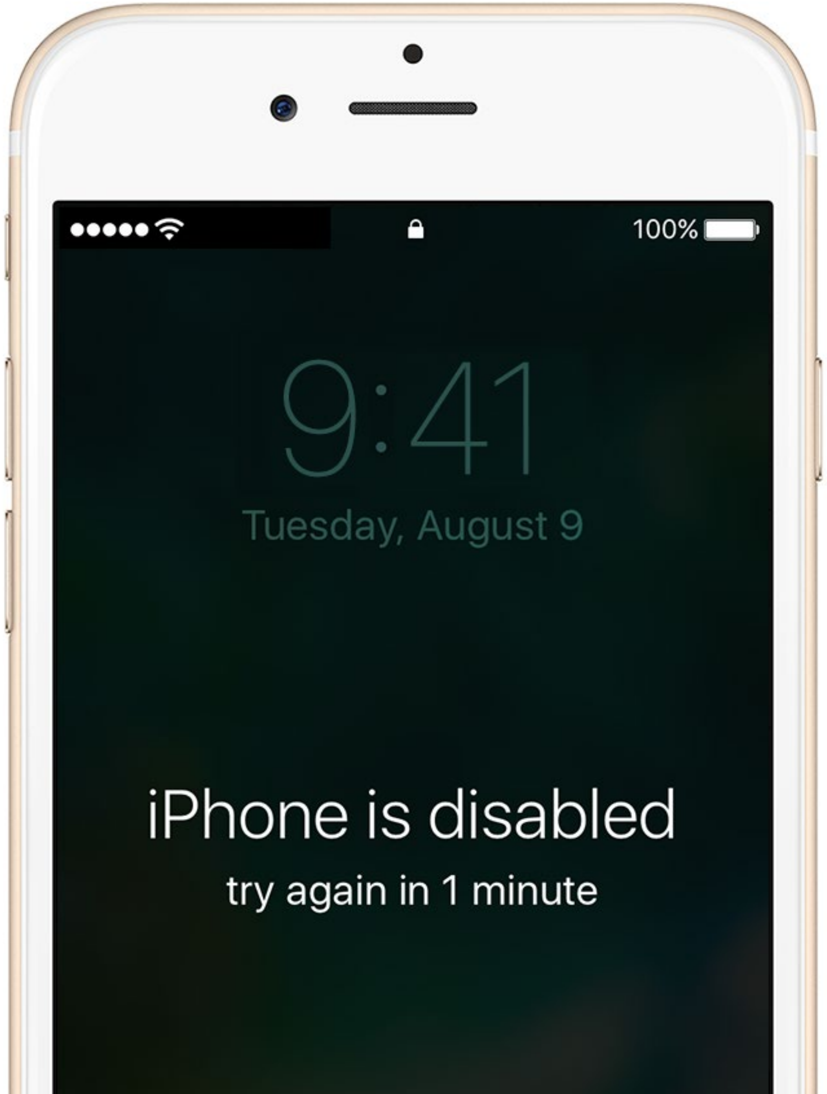
Simper design and implementation ➔ fewer possibilities for flaws

- Facilitates understanding by developers and users
- Facilitates careful review and verification
- Minimizes interfaces and interdependencies

Trusted computing base (TCB)

- Those portions of the system that are critical to its security
- Vulnerabilities in the TCB may jeopardize the security of the entire system
- The TCB should be as small as possible

Fail-safe Defaults



iPhone is disabled

try again in 1 minute

Fail-safe Defaults

"Deny" should be the default, unless privileges have been explicitly granted

- E.g., default user group has minimal access rights

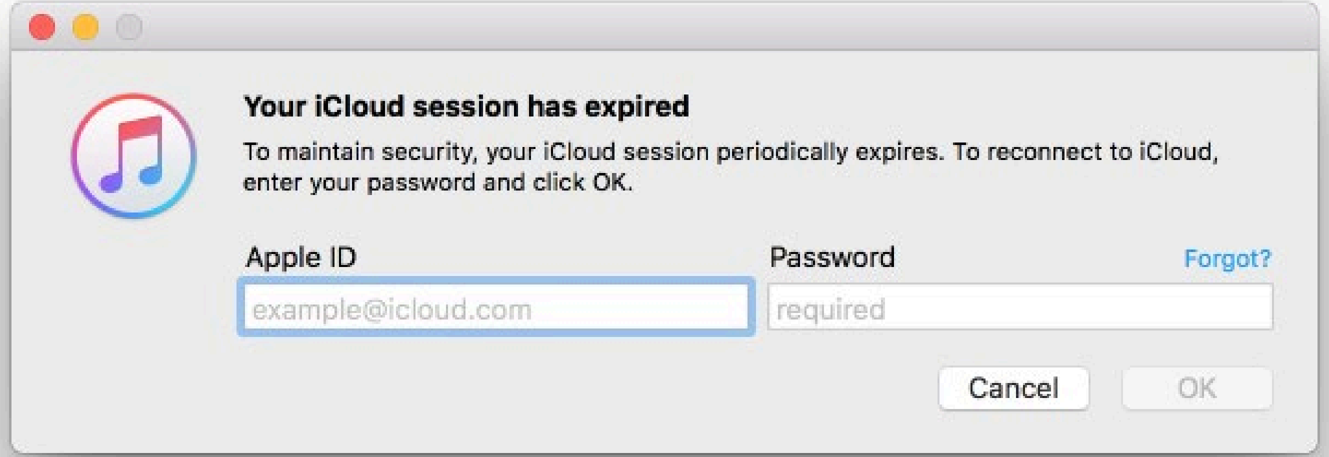
Oversights regarding handling of corner cases are a common cause of vulnerabilities

- Deny by default → denial of service
  Will be reported by legitimate users and corrected quickly
- Allow by default → potential for unauthorized access
  Will not be detected and turn into a vulnerability

Main challenge: usability vs. security

- Logging in as root, disabling Windows' UAC, jailbreaking, …
- Striking the right balance is not always easy

Complete Mediation



Your iCloud session has expired

To maintain security, your iCloud session periodically expires. To reconnect to iCloud, enter your password and click OK.

Apple ID: example@icloud.com

Password: required

Complete Mediation

Every access should be checked to ensure it is allowed

E.g., each transaction on an ATM requires re-entering the PIN

The mediation mechanism should be part of the TCB

E.g., the OS kernel mediates access to memory, files, devices

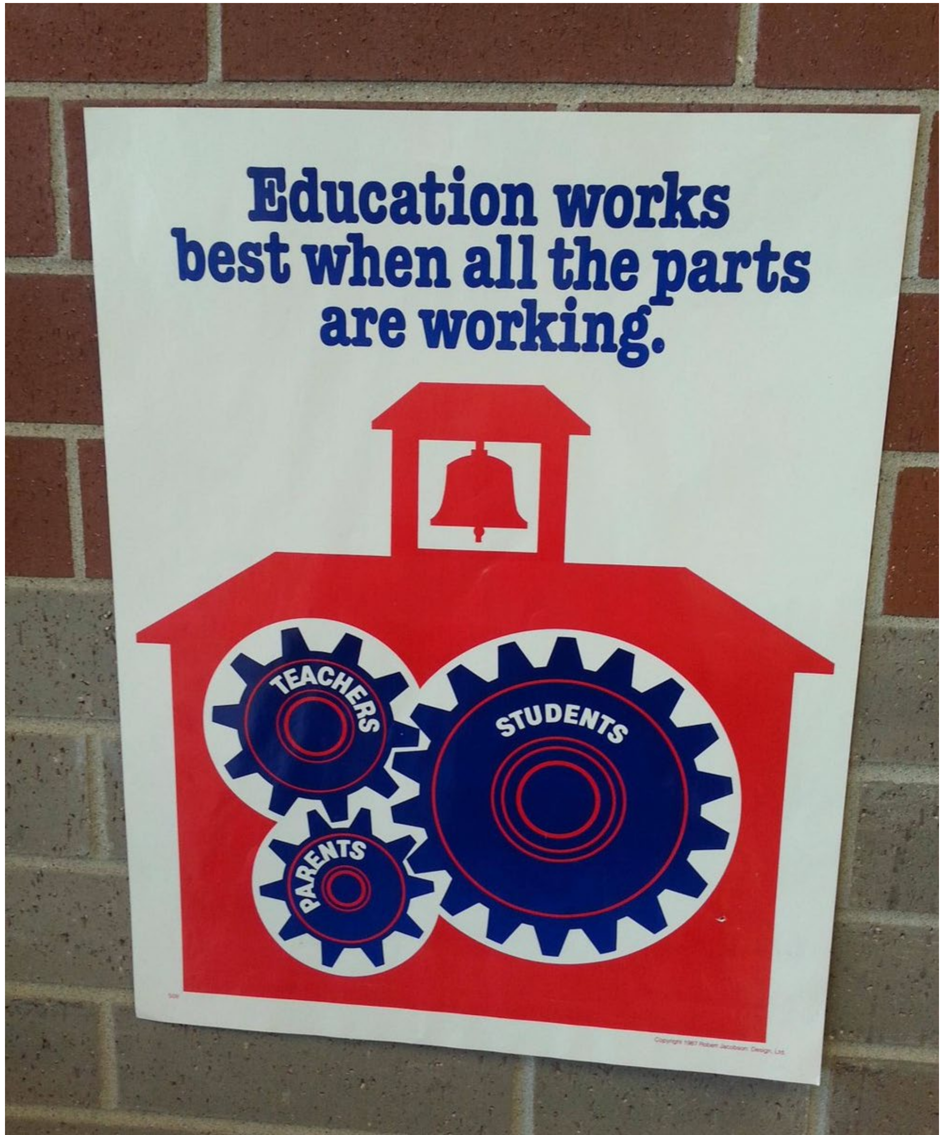
Main challenge: performance vs. security

Checking file permissions before opening a file vs. on every access:
permissions may change after opening

Caching DNS responses vs. always asking the authority:
an attacker may be able to poison the cache

More frequent checks → higher runtime overhead

Open Design


Education works best when all the parts are working.
TEACHERS
STUDENTS
PARENTS
Copyright 1987 Robert Jacobson Design, Ltd.

Open Design

The security of a mechanism should not rely on the secrecy of its design or implementation

Open design encourages scrutiny by multiple parties

- Earlier discovery of potential design or implementation errors

Security through obscurity is fragile

- Secrets may leak (e.g., insiders, neglect, theft), reverse engineering, …

Especially true in cryptography

- Kerkhoff's principle: *a cryptosystem should be secure even if everything about the system, except the key, is public knowledge*
- Secret keys/passwords are not algorithms: easily *replaceable*

Separation of Privilege

Separation of Privilege

It is more secure to grant permission based on multiple conditions instead of a single one

E.g., transfers of $50K or more must be signed off by two officers

Two-factor authentication

Attackers have to achieve more than simply stealing a password

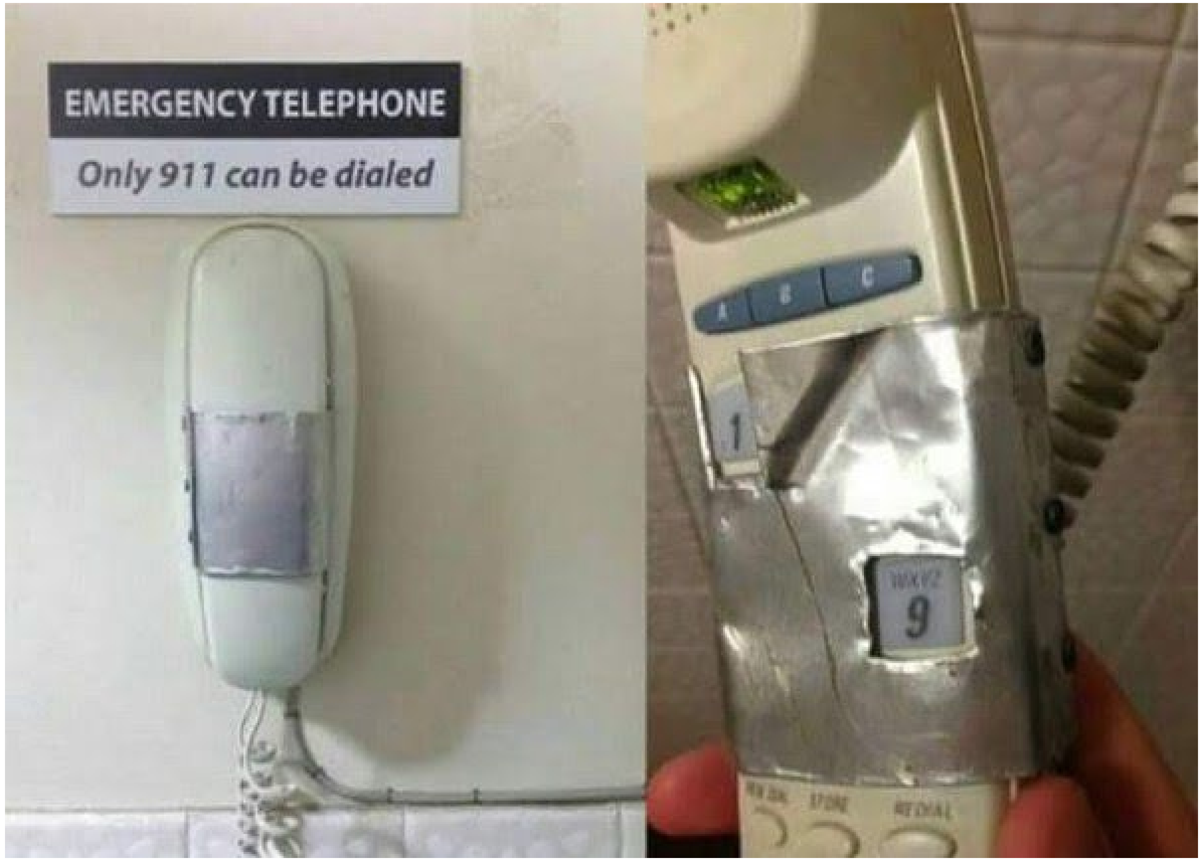
Related implication: *system compartmentalization*

Limit the damage caused by a compromise of any individual component

Separation: Monolithic OS kernel vs. microkernel, single process vs. multiple cooperating processes, single flat network vs. VLANs, …

Confinement: virtualization, containers, sandboxing, DMZ, VLANs, …

Least Privilege



Least Privilege

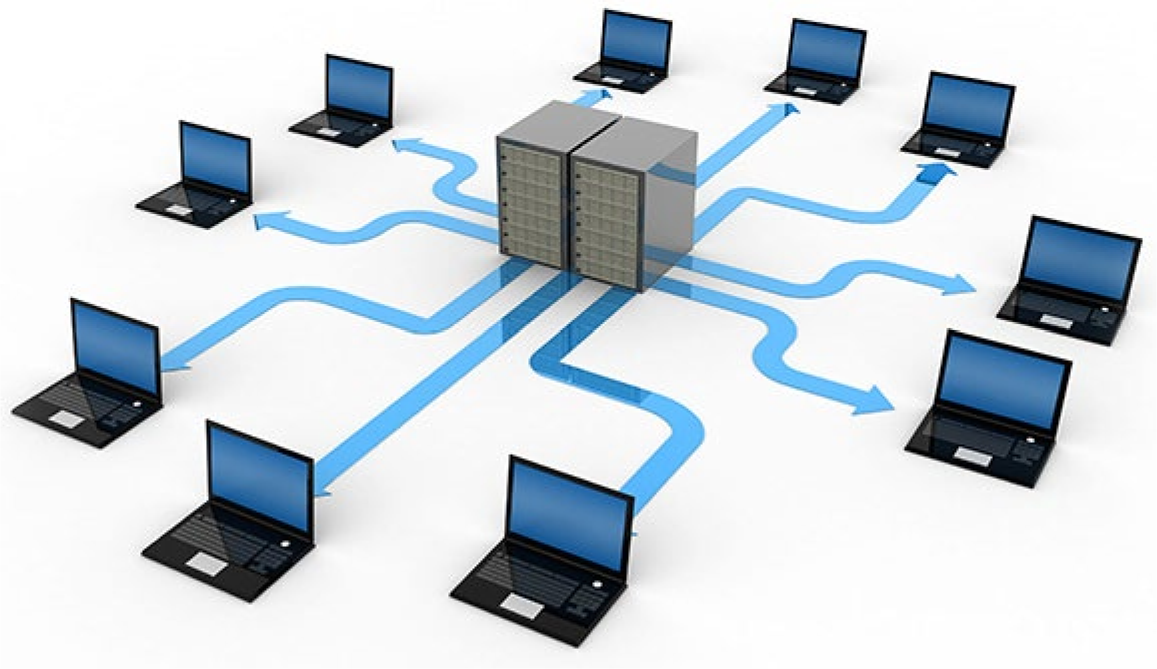
The system should grant the bare minimum set of privileges necessary to complete a given task

- Fewer privileges ➔ smaller damage upon compromise

Granularity matters

- All or nothing (e.g., root or non-root) vs. fine-grained permissions (e.g., capabilities, seccomp, access control lists)
- Poor design: root just for a single activity ➔ full system access when compromised
- Permissions may be needed only temporarily: start as root (e.g., for binding to a port <1024) and drop privileges right after
- Another example: Android app permissions (used to be all-or-nothing, now can be modified individually, and granted temporarily)
- Main challenge: identify the minimal set of privileges needed

Least Common Mechanism



Least Common Mechanism

Mechanisms allowing resources to be shared by multiple processes or users should be minimized

More shared state ➔ more ways for inadvertent information flows

- Shared system surfaces are attractive targets for attackers
- Confinement and compartmentalization can help

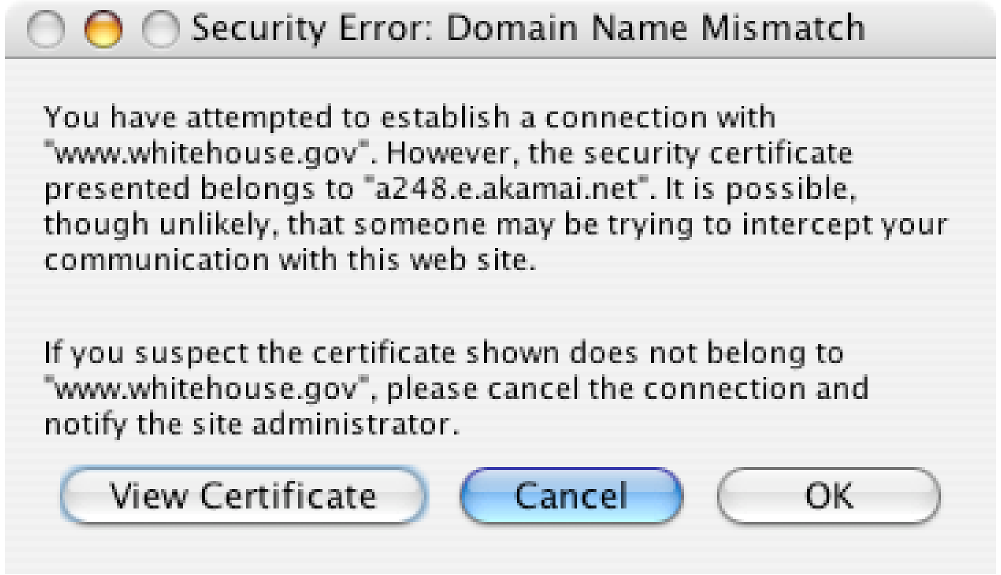
Main challenge: less state requires more careful (and potentially more complex) design

- Structured programming: avoid global state, avoid a single DB table for everything, …

Additional challenge: side channels

- Cryptographic algorithm implementations, microarchitectural attacks, …

Psychological Acceptability



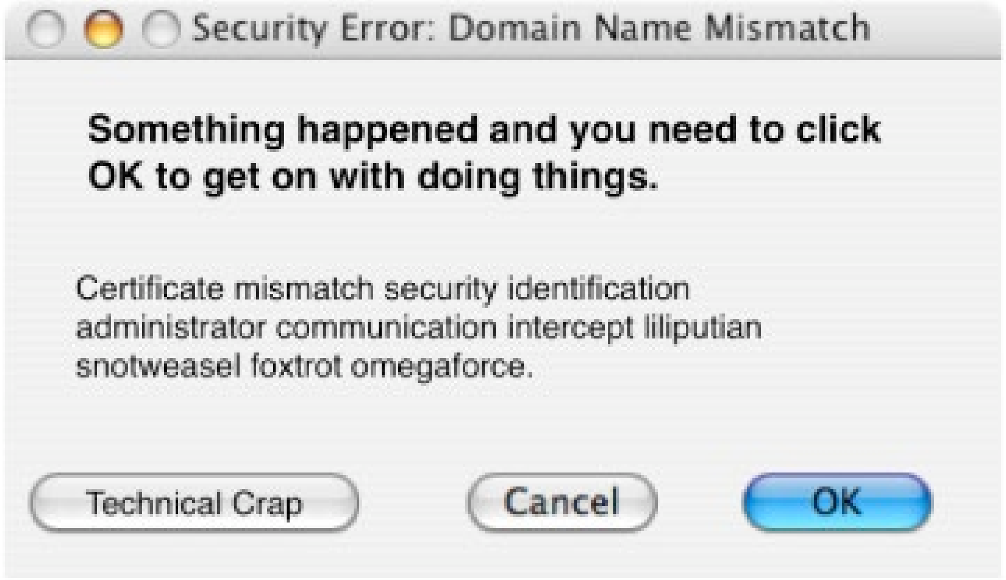
Security Error: Domain Name Mismatch

You have attempted to establish a connection with "www.whitehouse.gov". However, the security certificate presented belongs to "a248.e.akamai.net". It is possible, though unlikely, that someone may be trying to intercept your communication with this web site.

If you suspect the certificate shown does not belong to "www.whitehouse.gov", please cancel the connection and notify the site administrator.

© Johnathan Nightingale – http://blog.johnath.com/2007/10/11/todo-break-internet/

Psychological Acceptability



Security Error: Domain Name Mismatch

Something happened and you need to click OK to get on with doing things.

Certificate mismatch security identification administrator communication intercept liliputian snotweasel foxtrot omegaforce.

© Johnathan Nightingale – http://blog.johnath.com/2007/10/11/todo-break-internet/

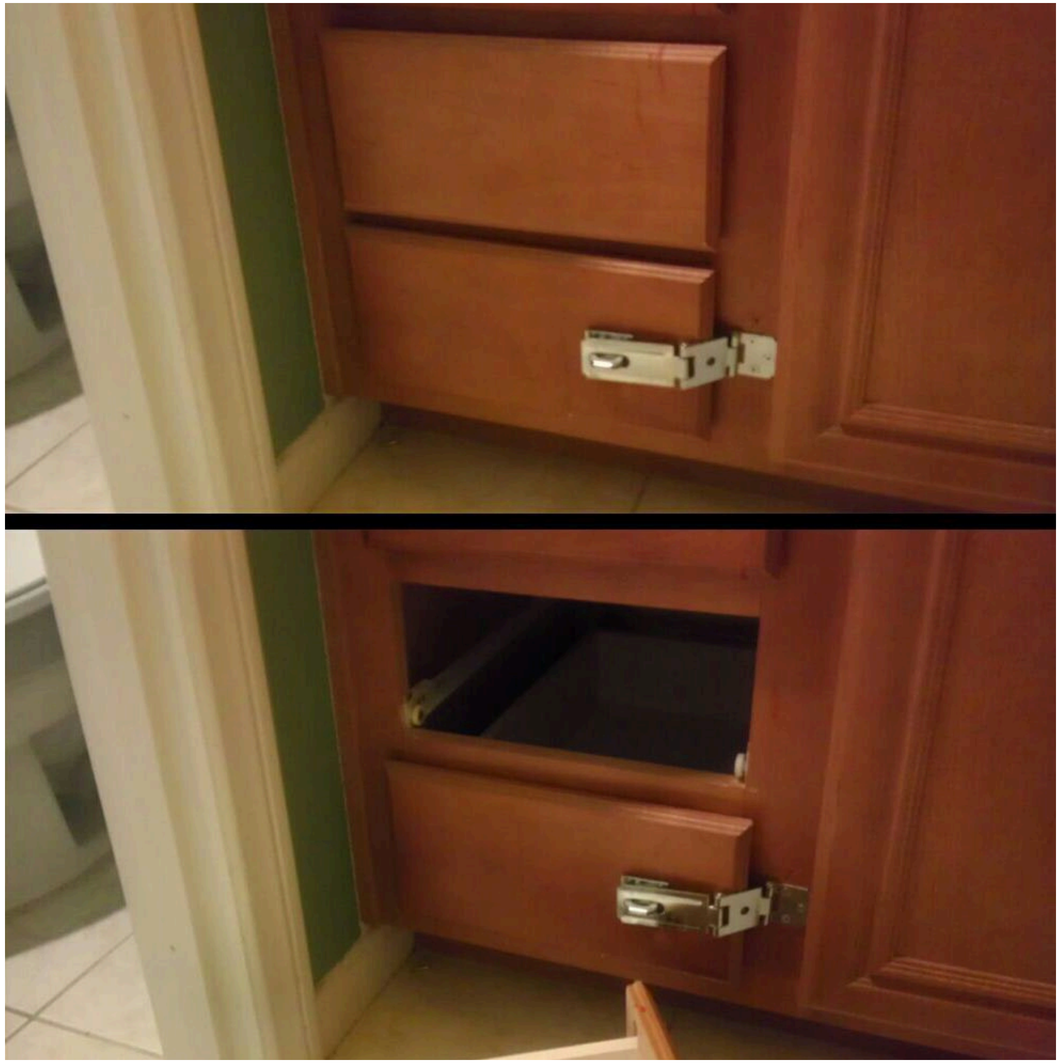
Psychological Acceptability

User interfaces should be intuitive and adhere to ordinary users' expectations

If users (including administrators) can't understand the system, they won't use it correctly

- Increased complexity leads to misconfigurations and mistakes: TLS certificates, PGP, Tor onion services, …
- Too much interruption leads to annoyance: ignore flood of IDS alerts, turn off AV, …
- Too much burden leads to workarounds: use a VPN to bypass firewall rules, write password on post-it note due to complex password requirements, …
- Repeated friction leads to weakened attention: training users to mindlessly clicking on cookie banners

Work Factor



Work Factor

The cost of bypassing a security mechanism should be compared with the resources an attacker must spend

Know your enemy: different threat models require different security mechanisms

- Online vs. offline password cracking, script kiddie vs. government agency, …

Quite challenging in practice due to advances in the state of the art

- Encryption key sizes that were considered safe are not anymore
- Code reuse replaced code injection

Elusive goal: "raise the bar for successful exploitation"

- The work factor is often hard to quantify

Compromise Recording

Compromise Recording

Detection and logging is equally important

Defense in depth

  If prevention mechanisms fail, detection mechanisms can provide an additional layer of defense

Intrusion detection

  Monitor networks or hosts for malicious activities or policy violations

Situational awareness

  Have a clear understanding of what is happening on the network and in the IT environment

Audit logs facilitate incident response and forensics