Threats, Vulnerabilities, and Attacks

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

   Different kinds: loss of services, compromise of information or functions, technical failure, …
   Different origins: deliberate, accidental, environmental, …

A vulnerability is a weakness that makes a threat possible

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

   Active vs. passive
   Insider vs. outsider
Threats, Vulnerabilities, and Attacks

Threat Classification and Risk Assessment

Classification example: 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: 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

Set of assumptions about possible attacks that 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, 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?
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
  Detection
  Recovery
  Awareness
Threat Actors

‘90s: script kiddies
‘00s: criminals
‘10s: nations  *(OK, much earlier, but now we talk about it)*

Different motives
   $$$$$$$$$$$
Honest but curious individuals
Political or social ends
Bribed or angry insiders
Espionage
Military *

Different resources: $$$$$$$$$$$, skills, infrastructure, …

Know your enemy!
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

SDL: design, implementation, operation, maintenance

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

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, “responsible” vs. full disclosure, …

Multiple vulns. 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
  Increases with complexity (more logic, features, dependencies, …)
Intrusions
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 host
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, internal components, external antennas/cameras/sensors), …

Remote

Internet
Local network (Ethernet, WiFi, 3/4G, bluetooth, …)
Infected media (disks, CD-ROMs, USB sticks, …)
Phone (social engineering)
Intrusion Method

Social engineering  (phishing, spam, scareware, …)

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

Network traffic interception  (access credentials, keys, …)

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

Physical access  (reboot, keylogger, screwdriver, …)

Software vulnerability exploitation
Just This Month’s News…
Google patched 10 critical remote code execution bugs in its August Android Security Bulletin issued Monday. It warned the most severe RCE vulnerabilities could enable a remote attacker, using a specially crafted file, to execute arbitrary code within the
Microsoft patched more than two dozen remote code execution vulnerabilities today, many of them rated critical. One was a RCE bug that allowed an attacker to take complete control of a server or workstation via Windows Search.
Juniper Networks warned customers Thursday of a high-risk vulnerability in the GD graphics library that could allow a remote attacker to take control of systems running certain versions of the Junos OS.
Despite a marked decrease in activity, exploit kits haven’t completely disappeared just yet. The Neptune, or Terror Exploit Kit, is alive and well; during the last month, researchers have observed the kit as part of a campaign to abuse a legitimate popup ad service to drop cryptocurrency miners.

Researchers with FireEye said Tuesday the kit has been redirecting victims with popups from fake hiking ads to exploit kit landing pages and in turn to HTML and Adobe Flash exploits. Researchers elected not to disclose the name of the popup ad service, but stressed that it’s within Alexa’s top 100.

The landing pages run a handful of exploits, including three targeting Internet Explorer (CVE-2016-7250, CVE-2016-7254, and CVE-2017-0078) and a fourth targeting Flash Player (CVE-2017-0118).
Russian-speaking cyberespionage group APT28, also known as Sofacy, is believed to be behind a series of attacks last month against travelers staying in hotels in Europe and the Middle East. APT28 notably used the NSA hacking tool EternalBlue as part of its scheme to steal credentials from business travelers, according to a report released Friday by security firm FireEye.

One of the goals of the attack is to trick guests to download a malicious document masquerading as a hotel reservation form that, if opened and macros are enabled, installs a dropper file that ultimately downloads malware called Gamefish. Gamefish establishes a foothold in targeted systems as a way to install the open source tool called Responder, according to FireEye.

“Once inside the network of a hospitality company, APT28 sought out machines that controlled both guest and internal Wi-Fi networks,” wrote authors of the report Lindsay Smith and Benjamin Read.
Remote Exploitation: Server-side vs. Client-side
(Very Simple) Buffer Overflow Exploitation

![Diagram of stack and code injection]

Code injection

**Shellcode**

- spawn shell
- listen for connections
- add user account
- download and execute malware
Malware and Botnets

click fraud
port scanning
phishing
illegal content
DDoS
code injection
malicious websites
spam
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
Many more threats...

Password Attacks
Information Leakage
Spoofing
Repudiation
Privilege escalation
Information gathering
Session hijacking
Social engineering
Denial of Service
Tampering
Information disclosure
Sniffing
Spoofing

...subject of future lectures
Basic Security Principles

In the 1970s, J. H. Saltzer and M. D. Schroeder had been working on Multics

Identified a set of design principles intended to help designers of time-sharing OSs 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 architectures. Section III relates the material of the earlier sections to the architecture.

Authorize

To grant a principal access to certain information.

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.

To check the accuracy, correctness, and
Economy of Mechanism
Economy of Mechanism

*Security mechanisms should be as simple as possible*

Simpler 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

Default action should be to deny access, unless privileges have been explicitly granted

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

Oversights regarding handling corner cases are a common cause of vulnerabilities

Deny by default \(\rightarrow\) denial of service
Will be reported by legitimate users and corrected quickly

Allow by default \(\rightarrow\) 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

Forgot?

Cancel  OK
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 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 overhead
Open Design

Education works best when all the parts are working.
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
   Kekhoff’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, …

Confinement: virtualization, containers, sandboxing, …
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: running as 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)

Main challenge: identify the minimal set of privileges
Least Common Mechanism
Least Common Mechanism

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

More shared state ➞ more possibilities 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
Psychological Acceptability

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.

View Certificate  Cancel  OK
Psychological Acceptability

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

Certificate mismatch security identification administrator communication intercept liliputian snotweasel foxtrot omegafone.
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: e.g., 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, …
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. NSA, …

Quite challenging in practice due to advances in technology and 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 be 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**