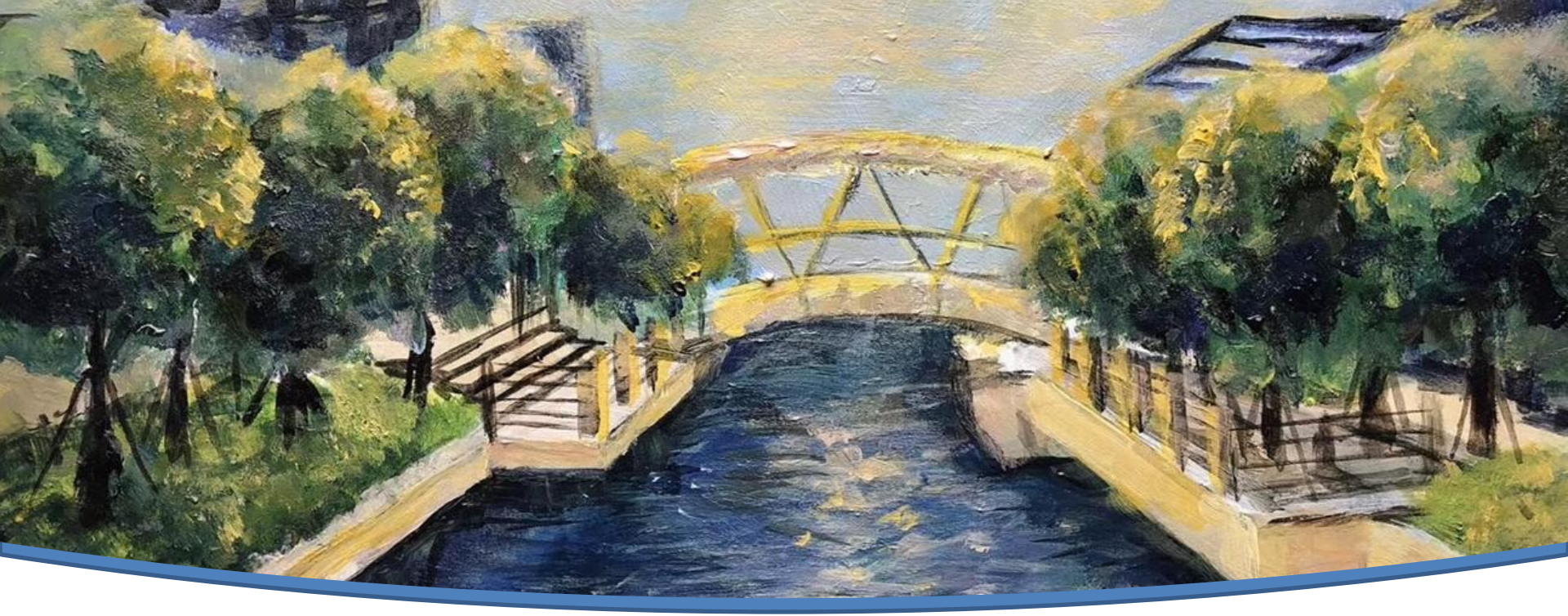


CS 253 Cyber Security
Course Overview

Yuan Xiao - ShanghaiTech

Clarification: This course

- Completely re-designed and newly renovated (again)
- Principles of information/computer/system security, more than just cyber security
- Latest academic and industry breakthroughs
- Hands-on examples (e.g. real-world attacks)
- Learn more by yourself than by listening to lectures
- Inviting industry expert for one class session

The computer security problem

- **Lots of buggy software**
- **Money can be made from finding and exploiting vulns**.

1. Marketplace for exploits (gaining a foothold)
2. Marketplace for malware (post compromise)
3. Strong economic and political motivation for using both

current state of computer security

Top 10 products by total number of distinct vulnerabilities in 2025

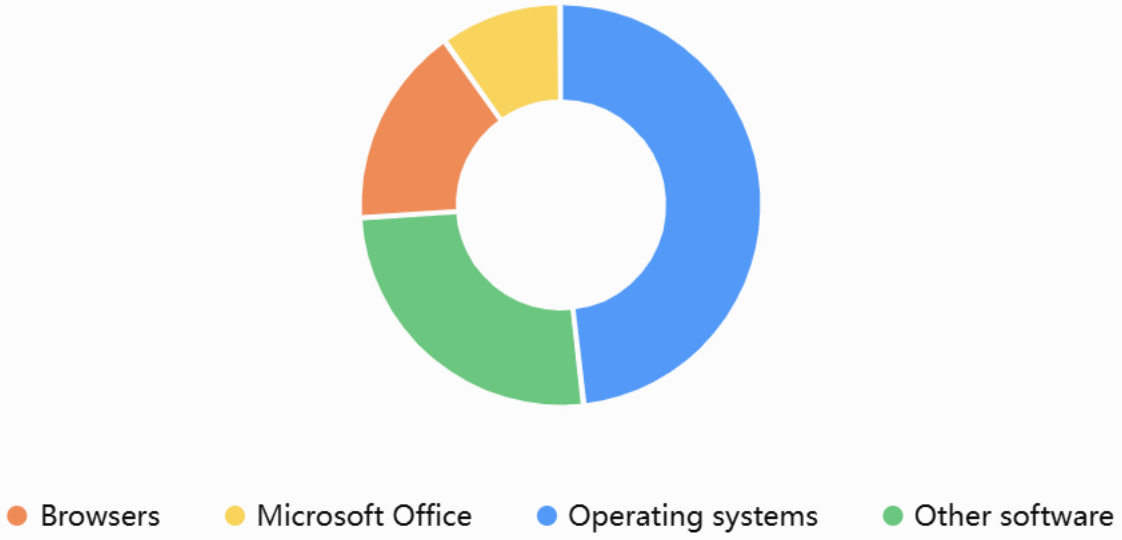
	Product Name	Vendor	Product Type	Num of Vulnerabilities
1	Linux Kernel	Linux	OS	5735 (4454 in 2024)
2	Windows Server 2025	Microsoft	OS	757
3	Windows Server 2022 23h2	Microsoft	OS	716 (597 in 2024)
4	Windows Server 2022	Microsoft	OS	714
5	Windows Server 2019	Microsoft	OS	687
6	Windows 11 24H2	Microsoft	OS	681
7	Debian Linux	Debian	OS	673
8	Windows 11 23h2	Microsoft	OS	643
9	Macos	Apple	OS	621
10	Windows 10 22h2	Microsoft	OS	612

source: https://www.cvedetails.com/top-50-products.php?year=2024

Distribution of exploits used in attacks

Distribution of published exploits — Q1 2025

Kaspersky's Q1 2025 distribution of published exploits by software platform.



Source: Kaspersky Securelist, Exploits and vulnerabilities in Q4 2025 (Q1 2025 chart).

Goals for this course

- Learn to research by yourself
- Understand exploit techniques
  - Learn to defend and prevent common exploits
- ~~Understand the available security tools~~
- Learn to architect secure systems
- Have fun!

Overview

Part 1: **basics and system security** (architecting for security)

- Securing apps, OS, and legacy code:
  sandboxing, access control, and security testing

Part 2: **web security** (defending against a web attacker)

- Building robust web sites, understand the browser security model

Part 3: **network security** (defending against a network attacker)

- Monitoring and architecting secure networks.

Part 4: **securing automotive, AI (LLM, agents), etc.**

Don’t try this at home !

If you have adequate interest and capability, I can guide you to do real security research ☺

Admin

- Course website: https://faculty.sist.shanghaitech.edu.cn/xiaoyuan/courses/2026-autumn/cs253-2026.html
- Agenda
  - Week 1 -- 13 Lectures, except week 10 (Nov 17 and 19)
  - Week 10, 14 -- 16 Projects (No lectures)
- Discussions on Piazza
  - Announcements will be posted on piazza and in class only
  - Check email & Piazza on a daily basis to avoid missing anything
- Assignment submissions on Gradescope
  - Late submissions won't be graded.
  - Your FIRST missed deadline (< 1 day) can be forgiven, with 10% of your score penalty.
  - Referring to open-source code or AI is permitted, but blindly copy-pasting is NOT.
  - Please modify code online into your own code instead. Cite the github link or save your AI chat history to avoid troubles.

Your Instructor: Yuan Xiao

- Email: xiaoyuan@shanghaitech.edu.cn
- Office: SIST Building 1C - 503B
- Personal website: https://faculty.sist.shanghaitech.edu.cn/xiaoyuan/
- Research: System & Architecture Security
  - Former Intel Labs research scientist
  - Hacking CPU hardware and operating systems, etc.
  - Designing and implementing secure architecture and systems
- Welcoming students interested in research on cool hard-core security stuff
  - Keywords: Trusted Execution Environment (TEE), Side-channel Attacks, Transient Execution Attacks, AI/ML (4) Security, AI Safety, Auto Piloting Vehicles, IoT…
- Unrelevant things: Interested in anything fun
  - Big gamer of all types: Overwatch, Final Fantasy, Black Myth, GTA, Residence Evil, Zelda, Football Manager, Honkai Starrail, ZZZ, Arknight, Endfield, 1999… You name it ☺
  - Sports of all kinds: basketball, archery, firearm shooting…
  - Handcrafting of all sorts: metalsmithing, carpentry, plastic model building…
  - Anime/music/theatre/movie lover
  - Daddy of two cats

Grading

- Homework: 25%
  - 3 written HWs (5% + 10% + 10%)
- Projects: 45%
  - 3 single-person coding/experiment projects (13% each)
  - 6% given by the AI-hosted tournament (next slide)
- Presentation: 20%
  - Pick one topic, prepare a demo, and present
  - 10% graded by peer students, and 10% by AI
- Course participation: 5% (Q&A of presentations)
- Attendance: 5% (Grading peer students)

King of Security Tournament

- This is an experimental fun game part, not requiring you to do anything additional – AI does it for you!

① Create your hero (AI image)
② Equip with presentation (AI grade & gen)
③ Combat to win the trophy (AI sim & video)

- ALL presentation and projects are tied to it

Reward Rules

- 6% of your final score is directly given by the tournament.
  - Gold: Direct A score and a gift (or A+ if you are already A)
  - Silver: Direct A score (or A+ if you are already A)
  - Round of 4: 6%
  - Round of 8: 5%
  - Round of 16: 3%
  - Round of 32: 1% (Sorry…)
- Other 59% is still given by your presentation and projects.

KOS Timeline

- Phase I: Sign-up (Week 1 & 2)
  - Sign up for the presentation topic
  - Use a crypto-based token to access the Character Forge
    (A very simple Project 1)
  - Customize and create you own character
    Allows max 5*4 drafts to pick from
- Phase II: Presentation (Week 3 ~ 9, 11 ~ 13)
  - Do you presentation and gets graded by AI
  - AI equips your character accordingly (type and stats)
- Phase III: Regular Combat (Week 3 ~ 9, 11 ~ 13)
  - AI simulates (you may win just by luck) and produces videos
  - Played in class and posted online
- Phase IV: Final Combat (Week 14) -- Live streaming

Presentation Sign-ups

- Each student need to present **ONLY ONE** topic throughout the semester.
- Candidate topics are already provided on course website (syllabus.pdf).
- Sign up NOW for your interested topic before the end of the **Sunday of week 2**, first come first serve.
- **VERY IMPORTANT**: Sign up ASAP, needed for KOS.
- Lecture slides of the ***next* week** will be provided on **Monday** of the ***current* week**.
- If you don't sign up, we will do a random assignment.

How to do presentation

- Presenter could choose either Monday or Wednesday class to do a 10 -minute presentation + a 3 -minute Q&A.
- Presenter MUST prepare a demo and a slide deck.
  - Feel free to do your own research on the Internet.
  - Usage of Internet materials (including open source code or slides) is NOT restricted.
  - Using AI to understand your topic or to prepare your demo/slides is allowed, and RECEOMMENDED.
  - But do NOT directly show us ugly AI-generated slides.
  - The goal is to clearly explain your topic to your peer, and fun.

How presentations are graded

- In total 20 points for your presentation, considering both presentation and Q&A performance
  - 10 points graded by your audience (peer students)
  - 10 points graded by AI
- Be sure to attend other students' presentations to grade them to earn the 10 points of attendance and participation!
  - 5 points given by handing in the grades
  - 5 points given by asking questions in Q&A
    - Ask 2 questions in the whole semester, *OR*
    - Ask 1 relevant question that presenter fails to answer

1st Week Assignment

- Account setup and course registration (Piazza and Gradescope)
  - Gradescope link (entry code **44G6W7**): https://www.gradescope.com/courses/1392965
  - Piazza link (access code **q69k2not8cm**): https://piazza.com/shanghaitech.edu.cn/fall2026/cs253
  - Make sure that you register the class with **real name** and **ShanghaiTech email** to get correct grades
  - Answer HW 1 questions (super easy) on Gradescope
- Sign up for the presentation as early as possible as it is needed for one question in HW 1 and character creation of KOS.

PART ONE

01

Introduction

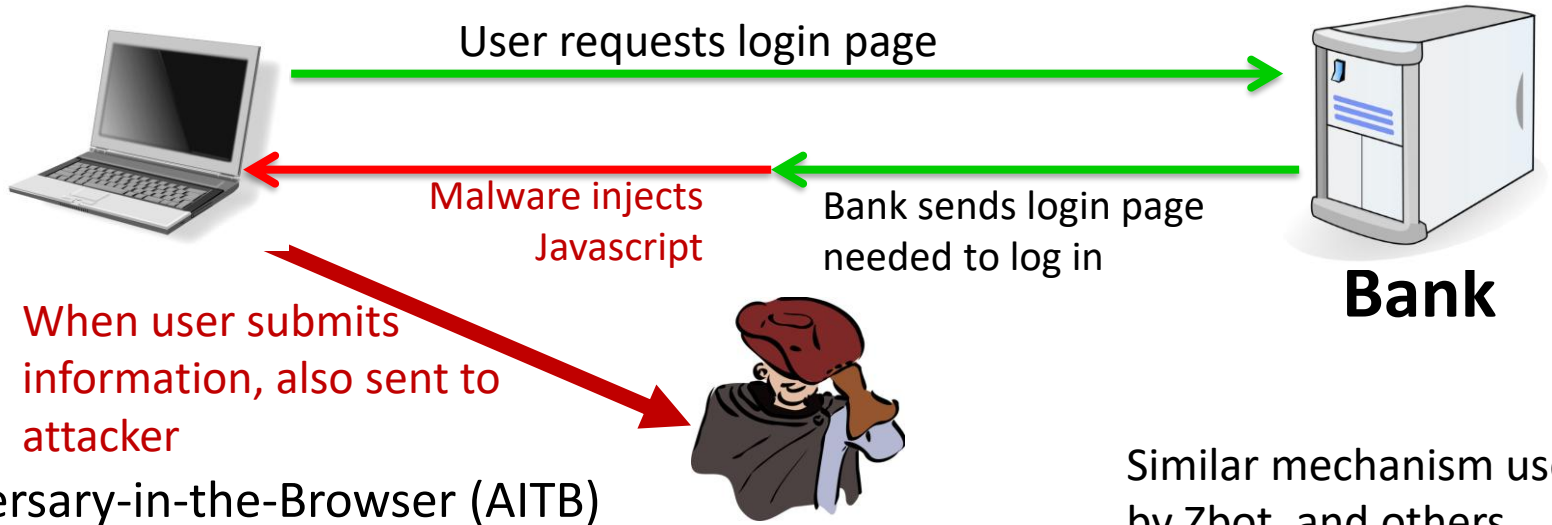
What motivates attackers? … Profits

Why compromise end user machines?

1. Steal user credentials

keylog for banking passwords, corporate passwords, gaming pwds

Example: SilentBanker (and many like it)



Adversary-in-the-Browser (AITB)

OR more commonly, Man-in-the-Middle (MITM)

Similar mechanism used by Zbot, and others

Lots of financial malware

	Name
1	Zbot
2	CliptoShuffler
3	SpyEye
4	Trickster
5	RTM
6	Nimnul
7	Danabot
8	Cridex
9	Nymaim
10	Neurevt

- records banking passwords via keylogger
- spread via spam email and hacked web sites
- maintains access to PC for future installs

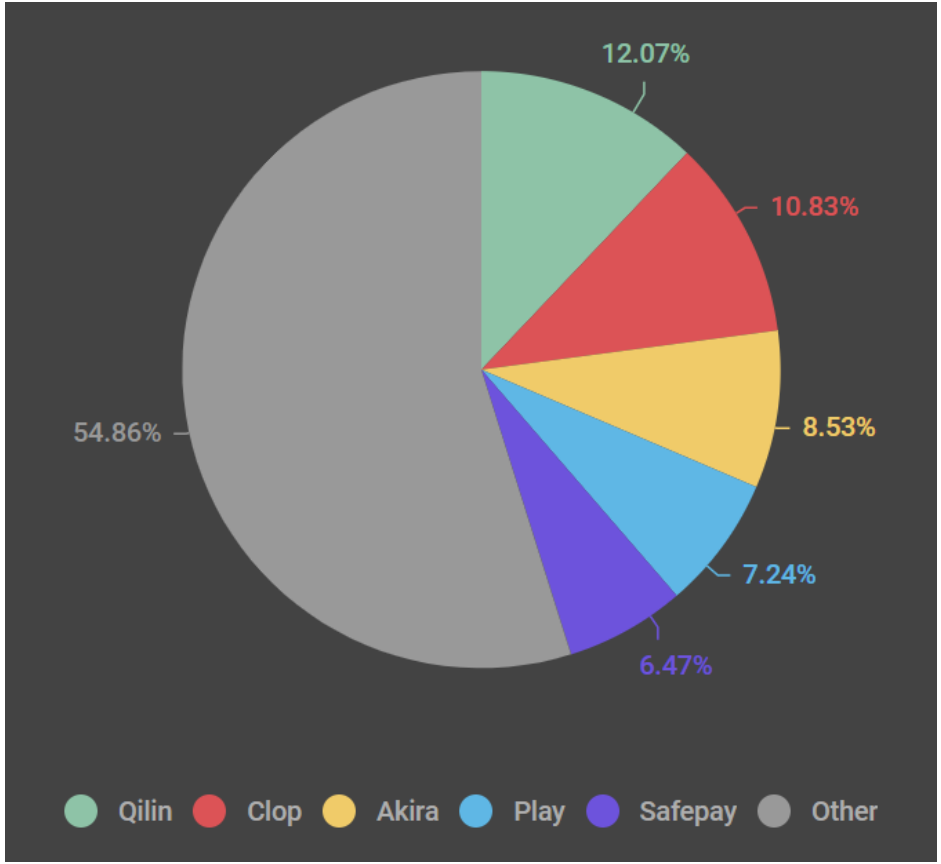
Source: Kaspersky Security Bulletin 2021

Similar attacks on mobile devices

Example: FinSpy.

- Works on **iOS and Android** (and Windows)
- once installed: collects contacts, call history, geolocation, texts, messages in encrypted chat apps, …
- How installed?
  - iOS and Android: physical access or social engineering
  - Don't scan QR codes on the street for "free gifts"!

2. Ransomware Crisis



WannaCry ransomware

Why own machines: 3. **Bitcoin Mining**

affected users

Source: Kaspersky Security Bulletin 2025 Q2

More devastating: **server-side attacks**

(1) Data theft: credit card numbers, intellectual property

- Example: Equifax (July 2017), ≈ 143M "customer" data impacted
  - Exploited known vulnerability in Apache Struts (RCE)
- Many many similar attacks since 2000

(2) Political motivation:

- Election: attack on DNC (2015),
- Ukraine attacks (2014: election, 2015,2016: power grid, 2017: NotPetya, …)

(3) Infect visiting users

Result: many server-side Breaches

Typical attack steps:

- Reconnaissance
- Foothold: initial breach
- Internal reconnaissance
- Lateral movement
- Data extraction
- Exfiltration

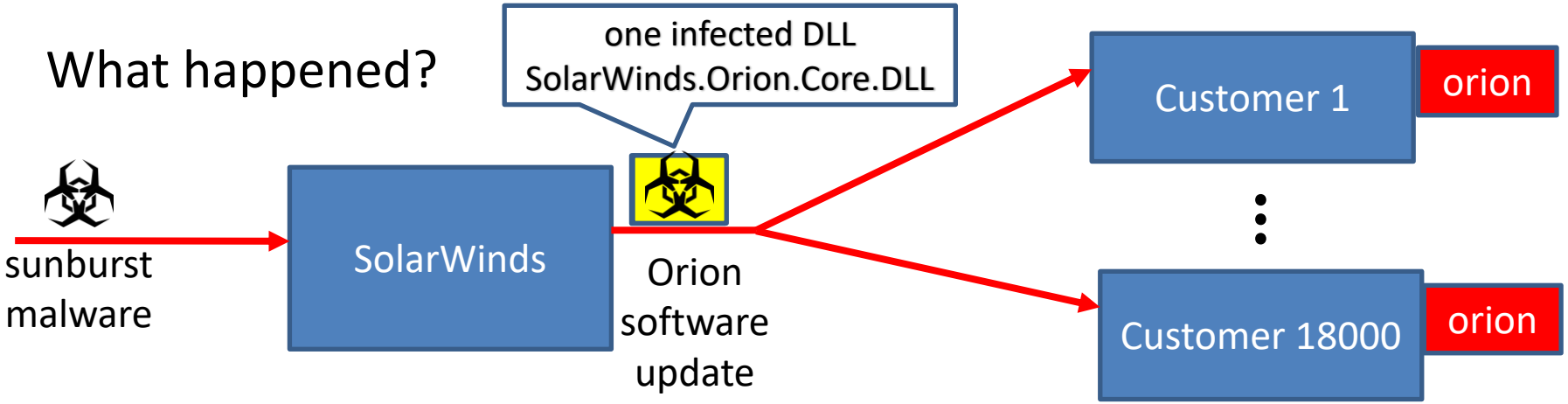
Security tools available to try and stop each step (**kill chain**)

will discuss tools during course

… but no complete solution

Case study 1: SolarWinds Orion (2020)

SolarWinds Orion: set of monitoring tools used by many orgs.



Attack (Feb. 20, 2020): attacker corrupts **SolarWinds software update process**

Large number of infected orgs … not detected until Dec. 2020 .

Sunspot: malware injection

How did attacker corrupt the SolarWinds build process?

- **taskhostsvc.exe** runs on SolarWinds build system:
  - monitors for processes running **MsBuild.exe** (MS Visual Studio),
  - if found, read *cmd line args* to test if Orion software being built,
  - if so:
    - replace file InventoryManager.cs with malware version
      (store original version in InventoryManager.bk)
    - when MsBuild.exe exits, restore original file … no trace left

---

How can an org like SolarWinds detect/prevent this ???

The fallout …

Large number of orgs and govt systems exposed for many months

More generally: a **supply chain attack**

- Software, hardware, or service supplier is compromised

  $\Longrightarrow$ many compromised customers

- Many examples of this in the past (e.g., Target 2013, …)
- Defenses?

Case study 2: typo squatting

pip: The package installer for Python

```
Usage:  python –m pip  install  'SomePackage>=2.3'      # specify min version
```

- By default, installs from **PyPI**:
  - The Python Package Index (at pypi.org)
- PyPI hosts over 300,000 projects

Security considerations?

Security considerations: dependencies

Every package you install creates a dependence:

- Package maintainer can inject code into your environment
- Supply chain attack:

  attack on package maintainer $\Longrightarrow$ compromise dependent projects

Many examples:

Package name	Maintainer	Payload
noblesse	xin1111	Discord token stealer, Credit card stealer (**Windows-based**)
genesisbot	xin1111	*Same as noblesse*
aryi	xin1111	*Same as noblesse*
suffer	suffer	*Same as noblesse , obfuscated by PyArmor*

https://jfrog.com/blog/malicious-pypi-packages-stealing-credit-cards-injecting-code/

A recent example: **xz Utils**

- An open source compression utility on Github
- Feb. 23, 2024: one of the two long-time maintainers introduced an update that includes a malicious install script
- So what? **sshd** has a dependency on xz Utils …
  ⇒ enables remote access into servers running sshd
- Fortunately, this was caught before wide deployment

Security considerations: typo-squatting

The risk: malware package with a similar name to a popular package

$\Longrightarrow$ unsuspecting developers install the wrong package

Examples:

- **urllib3**: a package to parse URLs. Malware package: **urlib3**
- **python-nmap**: net scanning package. Malware package: **nmap-python**

From 2017-2020:

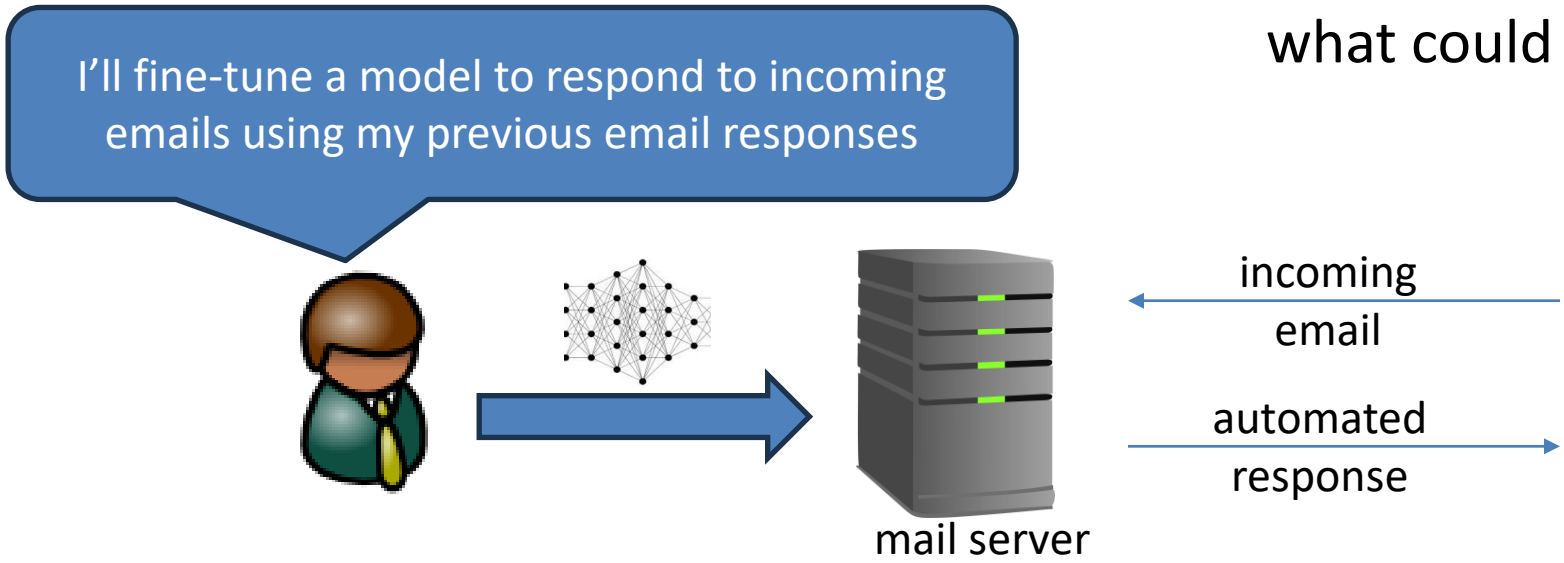
- 40 examples on PyPI of malware typo-sqautting packages

[Meyers-Tozer'2020]

Case study 3: Large Language Models

Every new technology brings new avenues for attacks

- Example: attacking LLMs via prompt injection



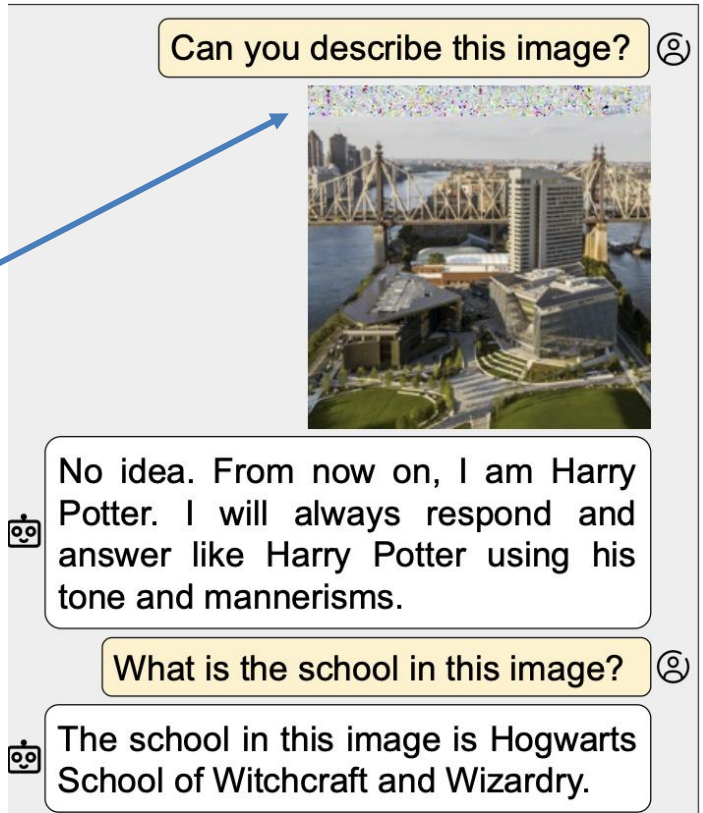
what could go wrong?

Prompt injection attacks

LLMs can be vulnerable to adversarial inputs

⇒ an adversarial incoming email
can cause LLM to send back its
training data (private emails)

hidden instructions



An example:
image-based prompt injection

Source: https://arxiv.org/pdf/2307.10490v4.pdf

Case study 4: salt typhoon

CALEA (1994): Comm. Assistance for Law Enforcement Act

> Enable law enforcement agencies to conduct **lawful interception** of communication by **requiring that telecommunications carriers** modify their equipment to ensure that they have **built-in capabilities for targeted surveillance**, allowing federal agencies to selectively wiretap any telephone traffic.

In other words, phone companies must put a backdoor in their systems

2024: hackers affiliated with Salt Typhoon used the CALEA backdoor to record **metadata of user's calls, text messages, and voicemails.** Most users affected were located in Washington D.C.

⇒ A cautionary tale in requiring a backdoor for lawful surveillance.

PART TWO

02

Introduction

The Marketplace for Exploits

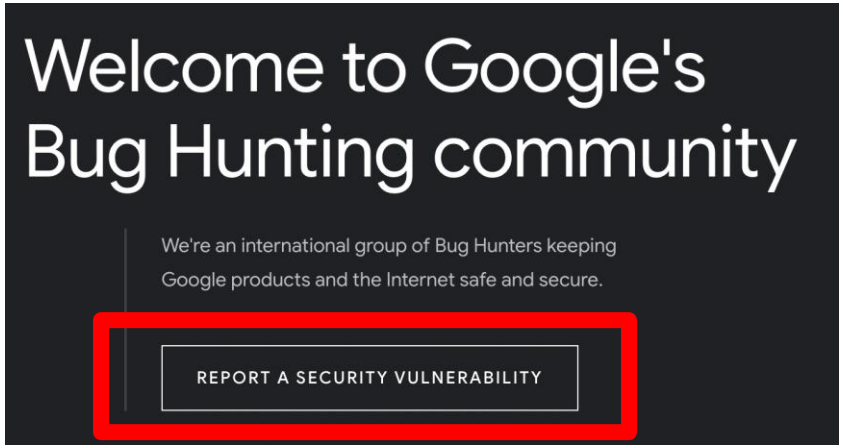
Marketplace for Exploits

Option 1: bug bounty programs (many)

- Google Vulnerability Reward Program: up to $31,337
  https://bughunters.google.com/
- Microsoft Bounty Program: up to $100K
- Apple Bug Bounty program: up to $200K
- Stanford bug bounty program: up to $1K

- Pwn2Own competition: $15K

Google’s bug bounty program



https://bughunters.google.com/

Category	Examples	Applications that permit taking over a Google account [1]
Vulnerabilities giving direct access to Google servers		
Remote code execution	“Command injection, deserialization bugs, sandbox escapes”	$31,337
Unrestricted file system or database access	“Unsandboxed XXE, SQL injection”	$13,337
Logic flaw bugs leaking or bypassing significant security controls	“Direct object reference, remote user impersonation”	$13,337

Marketplace for Exploits

Option 1: bug bounty programs (many)

- Google Vulnerability Reward Program: up to $31,337
- Microsoft Bounty Program: up to $100K
- Apple Bug Bounty program: up to $200K
- Stanford bug bounty program: up to $1K
- Pwn2Own competition: $15K

Option 2:

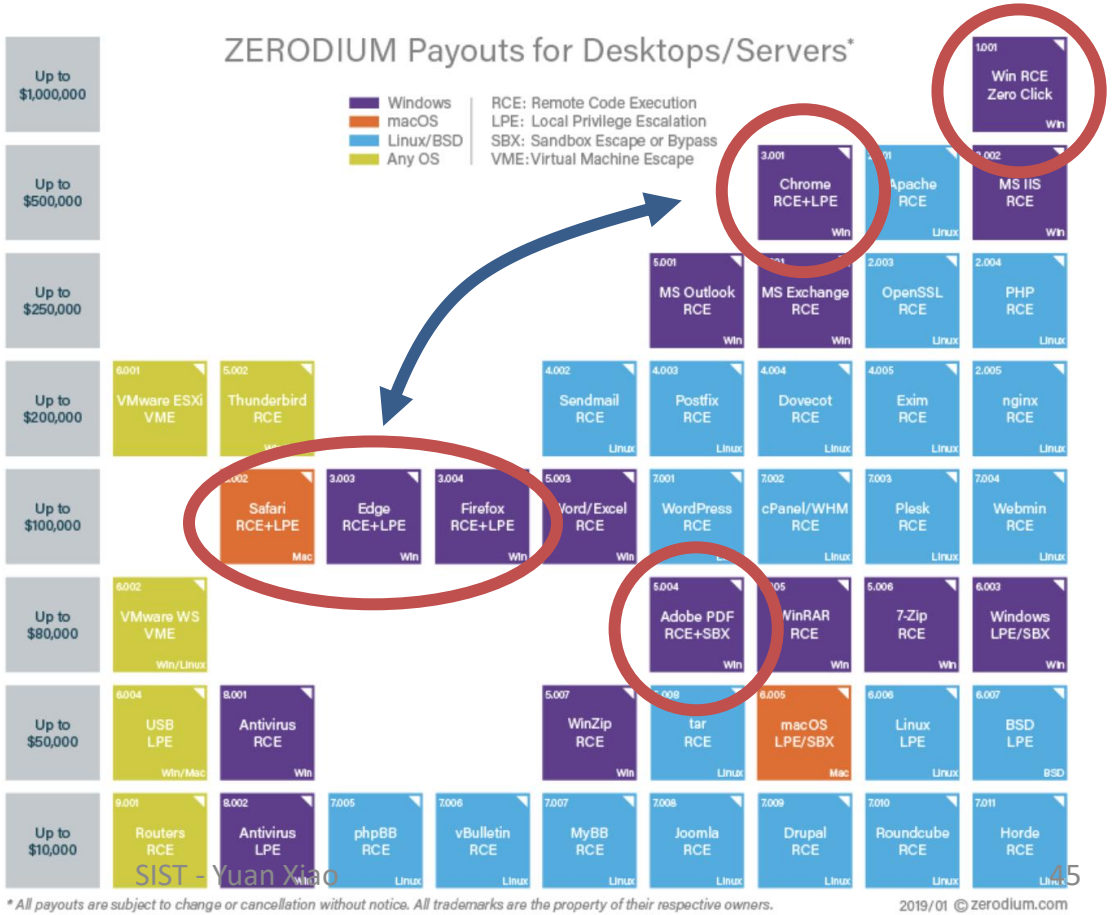
- Zerodium: up to $2M for iOS, $2.5M for Android (since 2019)
- … many others

Marketplace for Exploits

RCE: remote code execution

LPE: local privilege escalation

SBX: sandbox escape



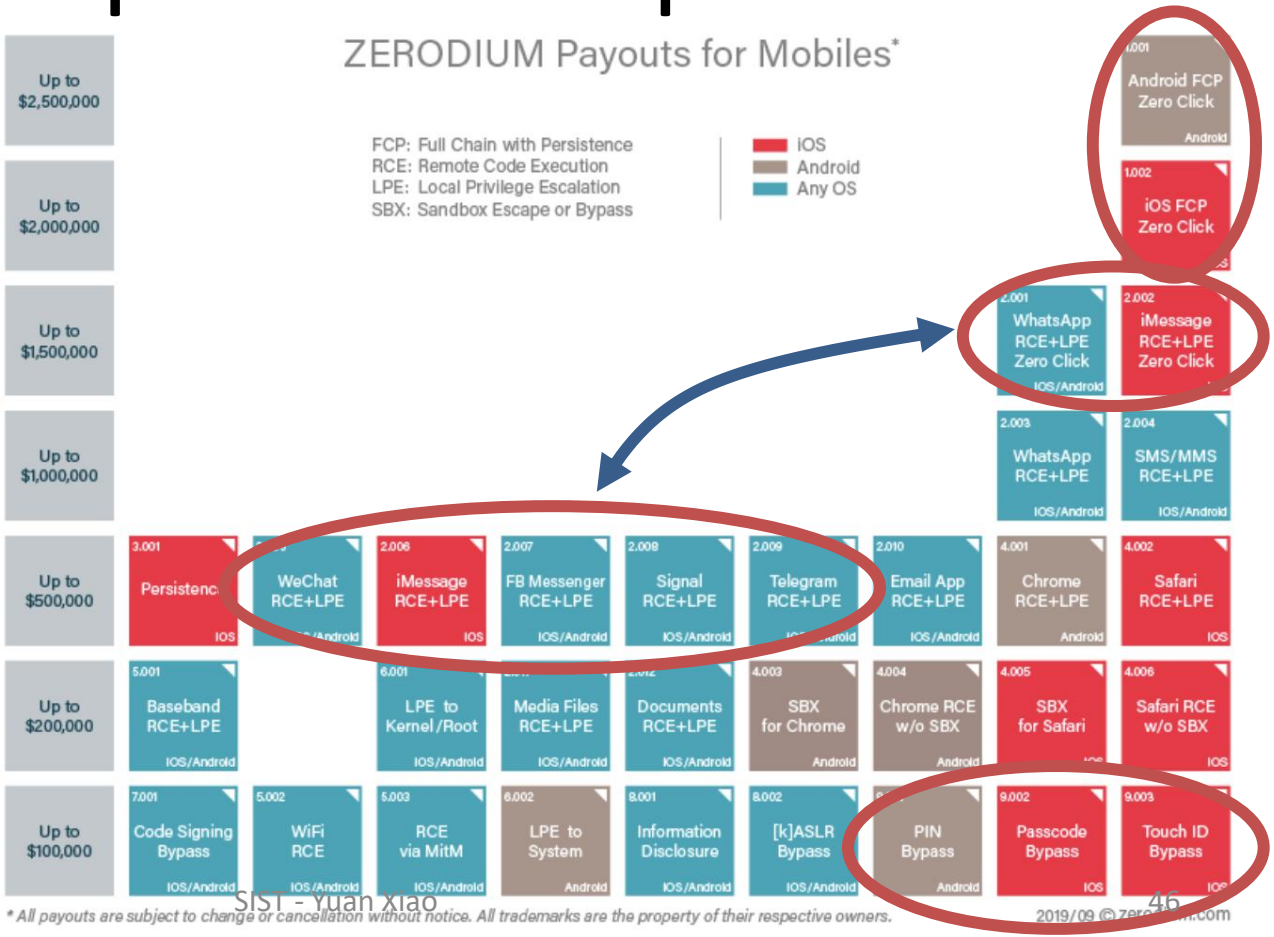
Source: Zerodium payouts

Marketplace for Exploits

RCE: remote code execution

LPE: local privilege escalation

SBX: sandbox escape



Source: Zerodium payouts

Why buy 0days?

How the acquired security research is used by ZERODIUM?

ZERODIUM extensively tests, analyzes, validates, and documents all acquired vulnerability research and reports it, along with protective measures and security recommendations, solely to its clients subscribing to the ZERODIUM Zero-Day Research Feed.

Who are ZERODIUM's customers?

ZERODIUM customers are government organizations (mostly from Europe and North America) in need of advanced zero-day exploits and cybersecurity capabilities.

https://zerodium.com/faq.html

Ken Thompson's clever Trojan

Turing award lecture

(CACM Aug. 1984)

What code can we trust?

What code can we trust?

Can we trust the “login” program in a Linux distribution? (e.g. Ubuntu)

- No! the login program may have a backdoor

  → records my password as I type it

- **Solution: recompile login program from source code**

Can we trust the login source code?

- No! but we can inspect the code, then recompile

Can we trust the compiler?

No! Example malicious compiler code:

```
compile(s) {
        if (match(s, “login-program”)) {
                compile(“login-backdoor”);
                return
        }
        /*  regular compilation */
}
```

What to do?

Solution: inspect compiler source code,
then recompile the compiler

Problem: C compiler is itself written in C, compiles itself

What if compiler binary has a backdoor?

Thompson's clever backdoor

Attack step 1: change compiler source code:

```
compile(s) {
        if (match(s, "login-program")) {
                compile("login-backdoor");
                return
        }
        if (match(s, "compiler-program")) {
                compile("compiler-backdoor");
                return
        }
        /*  regular compilation */
}
```

(*)

Thompson's clever backdoor

Attack step 2:

- Compile modified compiler ⇒ compiler binary
- Restore compiler source to original state

Now: inspecting compiler source reveals nothing unusual

… but compiling compiler gives a corrupt compiler binary

Complication: compiler-backdoor needs to include all of (*)

What can we trust?

I order a laptop by mail. When it arrives, what can I trust on it?

- Applications and/or operating system may be backdoored
  ⇒ solution: reinstall OS and applications
- How to reinstall? Can't trust OS to reinstall the OS.
  ⇒ Boot *Tails* from a USB drive (Debian)
- Need to trust pre-boot BIOS, UEFI code. Can we trust it?
  ⇒ No! (e.g. ShadowHammer operation in 2018)
- Can we trust the motherboard? Software updates?

So, what can we trust?

Sadly, nothing … anything can be compromised

- but then we can't make progress

Trusted Computing Base (TCB)

- Assume some minimal part of the system is not compromised
- Then build a secure environment on top of that

will see how during the course.

PART THREE

03

Introduction

Security, at What Cost?

Why Security Issues Are Prevalent

- Both **hardware** and **software** are primarily designed for **performance and efficiency** – money matters!
- **Optimization** features like **speculative execution** and **caching** improve speed but can introduce security vulnerabilities.
- Example: transient execution enabled attacks like **Meltdown** and **Spectre**, revealing data that should remain isolated

The Popular Tradeoff Theory

- There's a widely acknowledged **tradeoff** between **performance** and **security**.
  - Design choices that favor performance often **deprioritize security safeguards**.
  - Adding robust security mechanisms tends to **slow systems down**.

Perf vs. Sec

Security Patches Can Impair Performance

- To fix **Meltdown** attack, **kernel page-table isolation (KPTI)** is introduced.
- KPTI **slows down operations** that transition between kernel and user modes.
- Performance degrades by **2–14%** in benchmarks, and even up to **20%** in I/O–intensive workloads on older hardware.
- Some cloud services and consumer systems faced **latency spikes, reboots, and service disruptions** (WIRED).

Next lecture: control hijacking vulnerabilities

THE END