

CS 370

Introduction to Security

Software Attacks



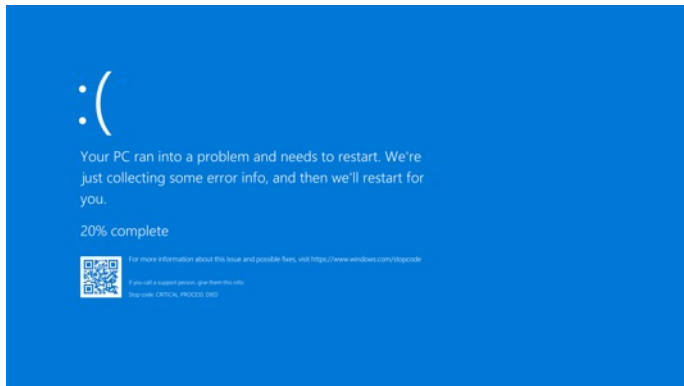
Oregon State
University

Topics For Today

- Compiler tools to detect/prevent software vulnerabilities
- How to disclose software vulnerabilities
 - Responsible disclosure of security vulnerabilities
 - Bug Bounty Program
 - Competitions (Pwn2Own)
- Cyberweapons and possible attack scenarios

Software Vulnerabilities Recap

- Modern software is complex
 - 34M+ lines of code in Google Chrome
 - 27M+ lines of code in Linux Kernel
 - 25M+ lines of code in Android
- Mistake happens



```
if (gidsetsize <= NGROUPS_SMALL)
    group_info->blocks[0] = group_info->small_blocki
else /
```

Software Vulnerabilities Recap

- Buffer overflow vulnerabilities
 - `char buf[100]; strcpy(buf, data);`

```
int main() {  
    char buf[512];  
    fgets(buf, 512, stdin);  
    printf(buf, 1, 2, 3);  
}
```

- Format String vulnerabilities

- `printf(buf);` where `buf = "%d %d %s %p %n ..."`

```
$ ./format  
%p %p %p %p %p %p %p  
0x1 0x2 0x3 0x7ff852cffaf8 0x0 0x7025207025207025 0x2520702520702520
```

- Use-after-free vulnerabilities

- `Object *o = new Object(); delete o; o->toString();`

- Other logic bugs

- `goto fail;`
- `goto fail;`

```
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)  
    goto fail;  
    goto fail; /* MISTAKE! THIS LINE SHOULD NOT BE HERE */  
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)  
    goto fail;
```

```
err = sslRawVerify(...);
```

A Good Practice: Compiler Options

- Don't ignore compiler warnings in using C/C++

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
int main() {
    char buf[10];
    fgets(buf, 512, stdin);
    printf(buf);
    return 0;
}
```

```
$ gcc -o a a.c
a.c: In function 'main':
a.c:8:12: warning: format not a string literal and no format arguments [-Wformat-security]
      8 |     printf(buf);
        |           ^^^
a.c:7:5: warning: 'fgets' writing 512 bytes into a region of size 10 overflows the destination [-Wstringop-overflow=]
      7 |     fgets(buf, 512, stdin);
        |     ^^^^^^^^^^^^^^^^^^^^^
a.c:6:10: note: destination object 'buf' of size 10
      6 |     char buf[10];
        |           ^^^
In file included from a.c:1:
/usr/include/stdio.h:592:14: note: in a call to function 'fgets' declared with attribute 'access(write_only, 1, 2)'
    592 | extern char *fgets (char *__restrict __s, int __n, FILE *__restrict __stream)
        |                  ^~~~~~
```

A Good Practice: Compiler Options

- -O2 -D_FORTIFY_SOURCE=2
 - Prevent buffer overflow and format string attacks

```
blue9057@blue9057-gram /tmp
$ gcc -oa a.c -O2 -fno-stack-protector -D_FORTIFY_SOURCE=0
blue9057@blue9057-gram /tmp
$ ./a
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[1] 964567 segmentation fault (core dumped) ./a
blue9057@blue9057-gram /tmp
$ ./a
139 ↵
%n
[1] 964966 segmentation fault (core dumped) ./a
blue9057@blue9057-gram /tmp
$
139 ↵
```

```
blue9057@blue9057-gram /tmp
$ gcc -oa a.c -O2 -fno-stack-protector -D_FORTIFY_SOURCE=2
blue9057@blue9057-gram /tmp
$ ./a
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
*** buffer overflow detected ***: terminated
[1] 966139 IOT instruction (core dumped) ./a
blue9057@blue9057-gram /tmp
$ ./a
134 ↵
%n
*** %n in writable segment detected ***
[1] 966260 IOT instruction (core dumped) ./a
blue9057@blue9057-gram /tmp
$
134 ↵
```

A Good Practice: Compiler Options

- -fsanitize=address
- Apply Address Sanitizer
 - <https://github.com/google/sanitizers/wiki/AddressSanitizer>

```
blue9057@MBP /tmp
$ gcc -o a a.c -fsanitize=address

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char *buf = (char *) malloc(10);
    fgets(buf, 512, stdin);      // heap buffer overflow
    free(buf);
    printf("%s\n", buf);        // use-after-free
}
```


Address Sanitizer

- Type AAAAAAAAAAAAAAAAAAAAAA to trigger heap buffer overflow

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
=====
==7327==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x6020000000ba at pc 0x00010b5b002c bp 0x7ff7b4d90a30 sp 0x7ff7b4d901f0
WRITE of size 45 at 0x6020000000ba thread T0
#0 0x10b5b002b in wrap_memcpy+0x2ab (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x1d02b)
#1 0x7ff81477b94e in fgets+0xf1 (libsystem_c.dylib:x86_64+0x2594e)
#2 0x10b5b2199 in wrap_fgets+0x49 (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x1f199)
#3 0x10b171e54 in main a.c:7
#4 0x110ead52d in start+0x1cd (dyld:x86_64+0x552d)

0x6020000000ba is located 0 bytes to the right of 10-byte region [0x6020000000b0,0x6020000000ba)
allocated by thread T0 here:
#0 0x10b5db4f0 in wrap_malloc+0xa0 (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x484f0)
#1 0x10b171e01 in main a.c:6
#2 0x110ead52d in start+0x1cd (dyld:x86_64+0x552d)

SUMMARY: AddressSanitizer: heap-buffer-overflow (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x1d02b) in wrap_memcpy+0x2ab
Shadow bytes around the buggy address:
 0x1c03ffffffc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x1c03ffffffd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x1c03ffffffe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x1c03fffffff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x1c0400000000: fa fa fd fd fa fa 00 00 fa fa 00 00 fa fa 00 fa
=>0x1c0400000010: fa fa 00 04 fa fa 00[02]fa fa fa fa fa fa fa fa
 0x1c0400000020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x1c0400000030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x1c0400000040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x1c0400000050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x1c0400000060: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char *buf = (char *) malloc(10);
    fgets(buf, 512, stdin); // heap buffer overflow
    free(buf);
    printf("%s\n", buf);    // use-after-free
}
```


Address Sanitizer

- Type A to trigger use-after-free bug

```
A
=====
==7307==ERROR: AddressSanitizer: heap-use-after-free on address 0x6020000000b0 at pc 0x00010ee3b93d bp 0x7ff7b150c1d0 sp 0x7ff7b150b950
READ of size 2 at 0x6020000000b0 thread T0
#0 0x10ee3b93c in printf_common(void*, char const*, __va_list_tag*)+0x8ec (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x2393c)
#1 0x10ee3c32c in wrap_printf+0xdc (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x2432c)
#2 0x10e9f6e6f in main a.c:9
#3 0x11ab2952d in start+0x1cd (dyld:x86_64+0x552d)

0x6020000000b0 is located 0 bytes inside of 10-byte region [0x6020000000b0,0x6020000000ba)
freed by thread T0 here:
#0 0x10ee60639 in wrap_free+0xa9 (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x48639)
#1 0x10e9f6e5d in main a.c:8
#2 0x11ab2952d in start+0x1cd (dyld:x86_64+0x552d)

previously allocated by thread T0 here:
#0 0x10ee604f0 in wrap_malloc+0xa0 (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x484f0)
#1 0x10e9f6e01 in main a.c:6
#2 0x11ab2952d in start+0x1cd (dyld:x86_64+0x552d)

SUMMARY: AddressSanitizer: heap-use-after-free (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x2393c) in printf_common(void*, char const*, __va_list_tag*)+0x8ec
Shadow bytes around the buggy address:
 0x1c03ffffffc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x1c03ffffffd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x1c03ffffffe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x1c03fffffff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x1c0400000000: fa fa fd fd fa fa 00 00 fa fa 00 00 fa fa 00 fa
=>0x1c0400000010: fa fa 00 04 fa fa[fd]fd fa fa fa fa fa fa fa fa
 0x1c0400000020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x1c0400000030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x1c0400000040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x1c0400000050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x1c0400000060: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char *buf = (char *) malloc(10);
    faets(buf, 512, stdin); // heap buffer overflow
    free(buf);
    printf("%s\n", buf); // use-after-free
}
```

How CVE Works? (Recap)

- Developers
 - Find vulnerabilities in their software (e.g., nginx v1.0.7 ~ 1.0.14 has a BOF)
 - Fix that
 - Announce that to CVE

Vulnerability Details : [CVE-2012-2089](#)

Buffer overflow in ngx_http_mp4_module.c in the ngx_http_mp4_module module in nginx 1.0.7 through 1.0.14 and 1.1.3 through 1.1.18, when the mp4 directive is used, allows remote attackers to cause a denial of service (memory overwrite) or possibly execute arbitrary code via a crafted MP4 file.

Publish Date : 2012-04-17 Last Update Date : 2021-11-10

- System operators
 - Watch the CVE list and update vulnerable software

How CVE Works (Recap)

- White Hat Hackers
 - Analyze software using testing methods
 - Fuzzing, symbolic execution, manual testing, code auditing, reverse engineering, etc
 - Find a bug
 - Exploit the bug
 - Vendor reports that to NIST/MITRE CVE

- **syslog**

Available for: iPhone 4s and later, iPod touch (5th generation) and later, iPad 2 and later

Impact: A local user may be able to change permissions on arbitrary files

Description: syslogd followed symbolic links while changing permissions on files. This issue was addressed through improved handling of symbolic links.

CVE-ID

CVE-2014-4372 : Tielei Wang and YeongJin Jang of Georgia Tech Information Security Center (GTISC)

Vulnerability Disclosure is Important

- A good model of vulnerability discovery/disclosure/patch cycle



1. A white hat hacker finds a vulnerability

Vulnerability Disclosure is Important

- A good model of vulnerability discovery/disclosure/patch cycle



1. A white hat hacker finds a vulnerability



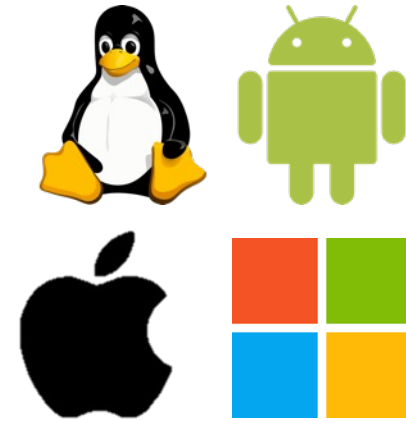
2. Let affected vendors know about the vulnerability;
Where in the code it is, why it happens,
and preferably with “how to fix it”

Vulnerability Disclosure is Important

- A good model of vulnerability discovery/disclosure/patch cycle



1. A white hat hacker finds a vulnerability



2. Let affected vendors know about the vulnerability;
Where in the code it is, why it happens,
and preferably with "how to fix it"

3. Vendors start to fix the vulnerabilities

Vulnerability Disclosure is Important

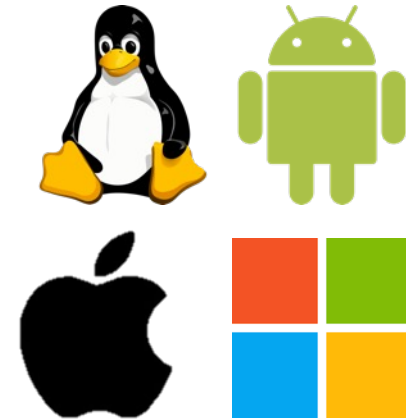
- A good model of vulnerability discovery/disclosure/patch cycle



1. A white hat hacker finds a vulnerability



4. Wait until vendors fixing the vulnerability (90~180 days)

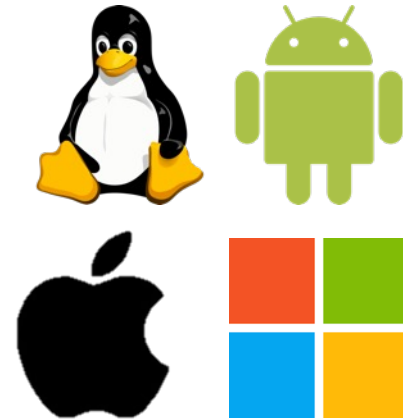


2. Let affected vendors know about the vulnerability;
Where in the code it is, why it happens,
and preferably with "how to fix it"

3. Vendors start to fix the vulnerability

Vulnerability Disclosure is Important

- A good model of vulnerability discovery/disclosure/patch cycle



1. A white hat hacker finds a vulnerability



4. Wait until vendors fixing the vulnerability (90~180 days)

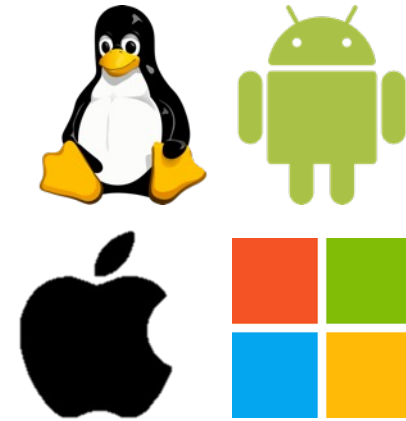
2. Let affected vendors know about the vulnerability;
Where in the code it is, why it happens,
and preferably with “how to fix it”

3. Vendors start to fix the vulnerability

5. The vulnerability has been fixed after 75 days..

Vulnerability Disclosure is Important

- A good model of vulnerability discovery/disclosure/patch cycle



1. A white hat hacker finds a vulnerability



4. Wait until vendors fixing the vulnerability (90~180 days)

2. Let affected vendors know about the vulnerability;
Where in the code it is, why it happens,
and preferably with “how to fix it”

3. Vendors start to fix vulnerability

5. The vulnerability has been fixed after 75 days..

6. The hacker publishes their work (paper, presentation, open-source, etc.)

Report -> Patch -> CVE -> UPDATE

- Vulnerability Patch Cycles

- Vendor gets a secret report from researchers/hackers/developers/etc
- Vendor fixes it, and release a patch
- At the same time, archive the vulnerability in the CVE database
- Users are running updates to be immune to the vulnerability

Patch Tuesday

From Wikipedia, the free encyclopedia

Patch Tuesday^[1] (also known as **Update Tuesday**^{[1][2]}) is an unofficial term used to refer to when [Microsoft](#), [Adobe](#), [Oracle](#) and others regularly release software patches for their software products.^[3] It is widely referred to in this way by the industry.^{[4][5][6]}

Microsoft formalized Patch Tuesday in October 2003.^{[1][7]} Patch Tuesday is known within Microsoft also as the “**B**” **release**, to distinguish it from the “C” and “D” releases that occur in the third and fourth weeks of the month, respectively.^[1]

Patch Tuesday occurs on the second Tuesday of each month^[8] in North America. Critical security updates are occasionally released outside of the normal Patch Tuesday cycle; these are known as "Out-of-band" releases. As far as the integrated [Windows Update](#) (WU) function is concerned, Patch Tuesday begins at 10:00 a.m. [PST](#).^[9] Vulnerability information is immediately available in the [Security Update Guide](#)[↗]. The updates show up in Download Center before they are added to WU, and the [KB](#) articles are unlocked later.

Report -> Patch -> CVE -> UPDATE

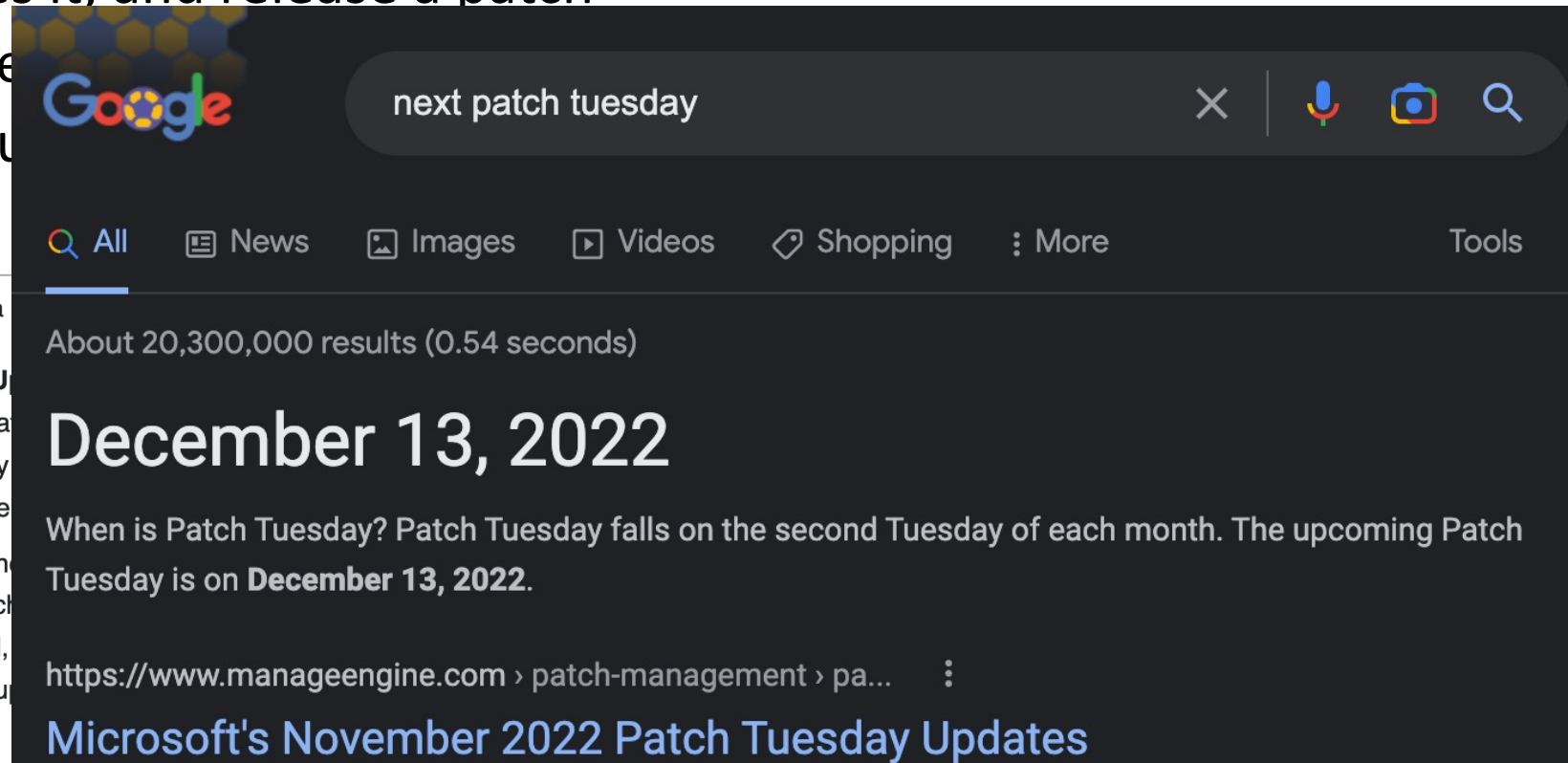
- Vulnerability Patch Cycles

- Vendor gets a secret report from researchers/hackers/developers/etc
- Vendor fixes it, and release a patch
- At the same time, the vendor releases a CVE
- Users are notified of the vulnerability

Patch Tuesday

From Wikipedia, the free encyclopedia

Patch Tuesday^[1] (also known as **Update Tuesday**) is a term used by Microsoft and other software vendors to describe the regular release of software patches. Microsoft formalized Patch Tuesday in 2002 to distinguish it from the "C" and "D" releases. Patch Tuesday occurs on the second Tuesday of each month, and patches are released outside of the normal Patch Tuesday cycle. The Windows Update (WU) function is concerned, the [Security Update Guide](#). The update is unlocked later.



A screenshot of a Google search interface. The search bar contains the text "next patch tuesday". Below the search bar, the results show "About 20,300,000 results (0.54 seconds)". The main result is titled "December 13, 2022" and includes the text "When is Patch Tuesday? Patch Tuesday falls on the second Tuesday of each month. The upcoming Patch Tuesday is on **December 13, 2022**." Below this, there is a link to "https://www.manageengine.com > patch-management > pa..." and a snippet titled "Microsoft's November 2022 Patch Tuesday Updates".

Responsible Vulnerability Disclosure

- Zero-day vulnerabilities are dangerous
 - 0-day vulns: vulnerabilities that do not have any patch yet
 - i.e., nobody other than who just discover the vulnerability knows about it
- But we need to report/disclose vulnerabilities
 - For vendors to release software patches to fix them
 - For users to update them
- Most importantly, we need to reduce the number of victims of the vulnerability disclosure
 - User software update takes time (do you update Windows right away on Tuesday??)
 - 90~180 days are norm in these days

How to Report Vulnerabilities?

- Contact to vendor directly
- In old days

Hey, I found a vuln in your product.
Here's the detail and way to fix it, so please do so!
It's free, just leaving my id, 'blue9057' for the credit.



What..the...
Hackers are bad. Criminals!
Let's call the police!!



Why?

- DMCA (Digital Millennium Copyright Act)
 - Enacted since 1998
- Section 1201 of DMCA

Seventh Triennial Section 1201 Proceeding, 2018

The Digital Millennium Copyright Act (“DMCA”), codified in part in 17 U.S.C. § 1201, makes it unlawful to circumvent technological measures used to prevent unauthorized access to copyrighted works, including copyrighted books, movies, video games, and computer software. Section 1201, however, also directs the

- Reverse engineering / hacking is prohibited by law

Sony vs Hotz

- Geohot and others hacked Sony Playstation 3
 - A complete jailbreak
 - Let users install any OS/apps, do whatever with PS3
 - Leaked a private key



Sony Computer Entertainment America, Inc. v. Hotz

From Wikipedia, the free encyclopedia

SCEA v. Hotz was a lawsuit in the [United States](#) by [Sony Computer Entertainment of America](#) against [George Hotz](#) and associates of the group fail0verflow. It was about jailbreaking and reverse engineering the PlayStation 3.

Contents [\[hide\]](#)

- 1 [Timeline](#)
- 2 [Alleged Anonymous attacks in response to lawsuit](#)
- 3 [See also](#)
- 4 [References](#)
- 5 [External links](#)

Timeline [\[edit \]](#)

On January 11, 2011, Sony sued George Hotz and fail0verflow members [Hector Martin](#) and Sven Peter on 8 claims,^[2] including violation of the [Digital Millennium Copyright Act](#), computer fraud, and copyright infringement.^[2] The law firm used by Sony was [Kilpatrick Townsend & Stockton LLP](#).^[3] In response to the suit, [Carnegie Mellon University](#) professor [David S. Touretzky](#) mirrored Hotz's writings and issued a statement supporting that Hotz's publication is within his right to free speech.^[4]

Sony Computer Entertainment America, Inc v. Hotz



Court	United States District Court for the Northern District of California
Full case name	<i>Sony Computer Entertainment America, Inc. v. George Hotz, Hector Martin Cantero, Sven Peter, and Does 1 through 100</i>
Started	January 11, 2011
Docket nos.	3:11-cv-00167

geohot's home on the internet...ain't my web design skills sick?

buy an Apple, Microsoft, LG, Samsung, Nintendo, not Axel Nix, ...

don't buy a Sony, [Axel Nix](#), (sue me, and your name can get added here)

check out [my startup](#)

i also have a poorly written [blog](#)

Lessons

- Don't do illegal stuff with computers
 - Even if you have a good will, you could get punished by jurisdiction
 - Releasing private key to public -> BAD
 - Release the exploit without having patch, let other users pirate apps
 - Yes, it is a crime
- But DMCA could be too harsh to the security community
 - Reverse engineering the device that you own
 - Hack the device that you own, not harm any others
 - Attack the 'test' user account that you have created
 - All such could be punished by DMCA

Lessons

- Don't

- Ev

- Re

- Re



LEGAL BOOP —

Judge drops DMCA claims that Bungie reverse-engineered *Destiny 2* cheats

- But D

- Re

- Ha

If you're going to reverse-sue under DMCA and CFAA, you'd better prove it.

KEVIN PURDY - 11/15/2022, 10:59 AM

- Attack the 'test' user account that you have created

- All such could be punished by DMCA



waypoint

GAMES BY VICE

Cheat Maker Sues Bungie for Hacking Its 'Destiny 2' Hacks

- But D

- Re

- Ha

- Attack the

- All such cc

If you're
it.

KEVIN PUFF

Bungie, has been countersued by the popular cheat maker AimJunkies over claims that Bungie violated the DMCA on AimJunkies cheats, hacked one of their contractor's computers, and violated copyright law by reverse engineering the software to build countermeasures against it.

This suit is only possible because of evidence presented by Bungie in its amended suit against AimJunkies for developing cheats. News of the countersuit was first reported by [TorrentFreak](#).

that
d

n

ops

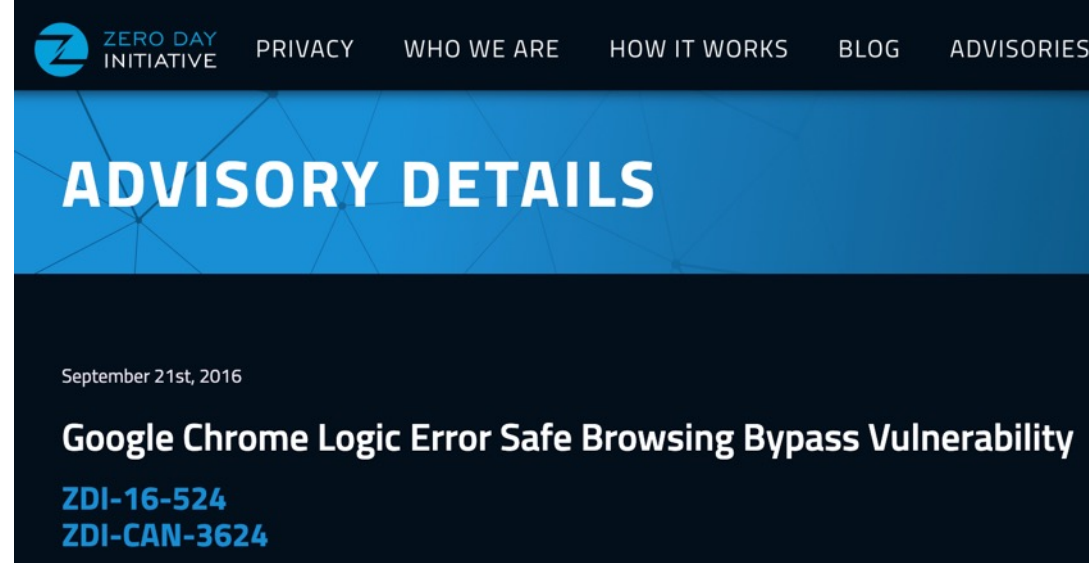
Then, How Can We Protect Ourselves from Lawsuit?

- Coordinated Vulnerability Disclosure
- There are companies that helps communication between
 - Hackers and vendors
 - The company offers full protection to the hackers
 - Anonymity of the submission, no lawsuit from the vendor, etc.
 - The company helps the vendor to fix the issue
 - The company gets compensation from the vendor
 - Pays back the hacker
- No lawsuit, hacker gets \$\$\$, vendor fixes vuln, all good!
 - How such a company earns \$\$\$? Is the company blackmails the vendor? I don't know...



ZDI Example

- ~6 month to be get public



DISCLOSURE TIMELINE

2016-03-19 - Vulnerability reported to vendor

2016-09-21 - Coordinated public release of advisory

CREDIT

Anonymous

This vulnerability allows remote attackers to bypass restrictions on vulnerable installations of Google Chrome. User interaction is required to exploit this vulnerability in that the target must visit a malicious page or open a malicious file.

The specific flaw exists within the handling of Safe Browsing. The issue lies in failure to properly check URLs. An attacker can leverage this in conjunction with other vulnerabilities to execute code in the context of the current process.

Pwn2Own

- ZDI also offers a hacking competition that targets real programs
 - VMM: VirtualBox, VMWare, Hyper-V
 - Web browser: Chrome, Edge, Safari, Firefox
 - Documents processor: MS Office and Adobe PDF
 - Many other categories, cellphone modems, etc.
- All discovered vulnerabilities will be reported to
 - Vendors via ZDI



The graphic is a vertical leaderboard titled "MASTER OF PWN" on the left and "LEADERBOARD" on the right. It features a table with five rows, each representing a participant. Each row has a rank number in a blue hexagon, the participant's name, a "PRIZE \$" column, and a "POINTS" column. The background is dark blue with a subtle pattern of white dots.

		PRIZE \$	POINTS
1	STAR Labs	\$270,000	27
2	Hector "P3rr0" Peralta	\$150,000	15
3	Masato Kinugawa	\$150,000	15
4	Manfred Paul	\$150,000	15
5	Synacktiv	\$75,000	7.5

Report to the Vendor

- Nowadays, many software vendors offer a 'bug bounty' program
- Meta: <https://m.facebook.com/whitehat/info/>
- Alphabet (Google):
<https://bughunters.google.com/about/rules/6625378258649088/google-and-alphabet-vulnerability-reward-program-vrp-rules>
- Apple: <https://security.apple.com/bounty/>
- MS: <https://www.microsoft.com/en-us/msrc/bounty>
- Discord: <https://discord.com/security>

Bounty range? Could be from \$50 to \$250,000, depending on severity



Public Bug Bounty Program

- Hacker One
 - <https://hackerone.com/bug-bounty-programs>

The screenshot displays the HackerOne platform's interface for finding public bug bounty programs. It is divided into two main sections: 'Programs in the Internet & Online Services industry' and 'Programs with updated bounties'. Each section contains a grid of program cards, each with a company logo, a star icon for favorites, and a status label (Updated or New). The cards provide details such as the program name, triaging status, response efficiency, and a list of supported platforms (e.g., Executable, Domain, iOS, Android, Play Store). Each card also includes a 'See details' button.

Programs in the Internet & Online Services industry

- Acronis** (Updated): Bug Bounty Program. Retesting, Bounty splitting. Platforms: Executable (7), Domain (7), iOS: App Store (4), Android: Play Store (2). Response efficiency: 99%. Range: \$1k - \$6k.
- ZeroBounce** (New): Bug Bounty Program. Triaged by HackerOne, Retesting. Platforms: Domain (2). Response efficiency: 97%. Range: \$150 - \$3k.
- CS Money** (Updated): Bug Bounty Program. Retesting. Platforms: Domain (4). Response efficiency: 67%. Range: \$100 - \$5k.
- Trustpilot** (Updated): Bug Bounty Program. Triaged by HackerOne, Retesting. Platforms: Domain (11), iOS: App Store (1). Response efficiency: 100%. Range: \$100 - \$2k.

Programs with updated bounties

- A.S. Watson Group** (Updated): Bug Bounty Program. Triaged by HackerOne, Retesting. Platforms: Other (19). Response efficiency: 97%. Range: \$100 - \$4k.
- Priceline** (Updated): Bug Bounty Program. Triaged by HackerOne, Retesting. Platforms: Domain (9), Android: Play Store (1), iOS: App Store (1). Response efficiency: 98%. Range: \$50 - \$5k.
- Expedia Group Bug Bounty** (New): Bug Bounty Program. Triaged by HackerOne, Retesting, Bounty splitting. Platforms: Android: Play Store (5), Domain (5), iOS: App Store (5). Response efficiency: 97%. Range: \$100 - \$5k.
- EPAM Bounty** (Updated): Bug Bounty Program. Triaged by HackerOne, Retesting. Platforms: Domain (10), Other (2), Android: Play Store (1), iOS: App Store (1). Response efficiency: 99%. Range: \$50 - \$10k.

Vulnerability Markets

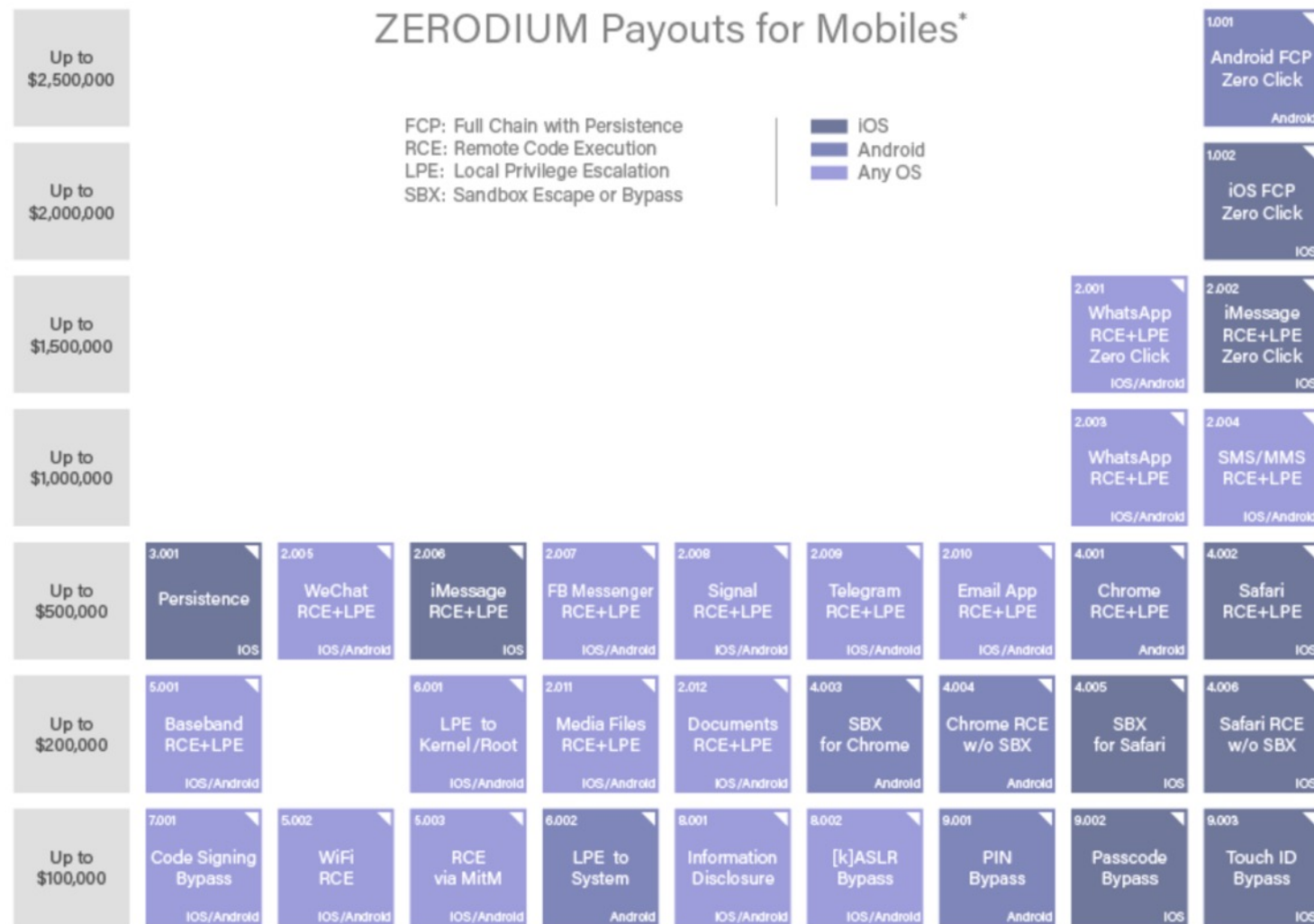
- Zerodium
 - Big bounties, how big?
- A leading exploit acquisition platform
 - What are advanced zero-day research?
 - What are cybersecurity capabilities?
 - Zerodium is for them...



Zerodium Payouts

- \$2.5M: Android 0-click
- \$2M: iPhone 0-click
- \$1.5M: iMessage 0-click
- \$1.5M: WhatsApp 0-click
- \$1M: WhatsApp/SMS...

- 0-click
- No user's interaction is required for the successful hack, so users cannot notice that their device got hacked or not



* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.

2019/09 © zerodium.com

0-click Attack

- 1. Suppose an attacker discovered a vulnerability in iMessage
- 2. Attacker writes an exploit that does:
 - 1) send an attack payload message to the victim
 - 2) the payload, before it triggers the push notification, hacks the device complete, suppress the push notification of the attack message
(The user will never see this...)
 - 3) make the attack persistent (survive after the reboot)
 - Attack iMessage app, break the application sandbox in iOS, attack iOS kernel to get root, and then, edit the root partition to make the attack persistent...
 - Usually it requires exploiting multiple vulnerabilities, say, 3~7-ish...

Think About the Business Model

- Who will pay that much amount of money?
 - Vendors? No... then do that in their own bug bounty program
 - Vendors does not need 0-click exploits; just need the crash input
- Who needs 0-click exploits?
 - Someone who wants to secretly peeping on others
 - Smartphone
 - Laptop/Desktop
 - E-mail/messages
 - Web accounts, passwords, etc....

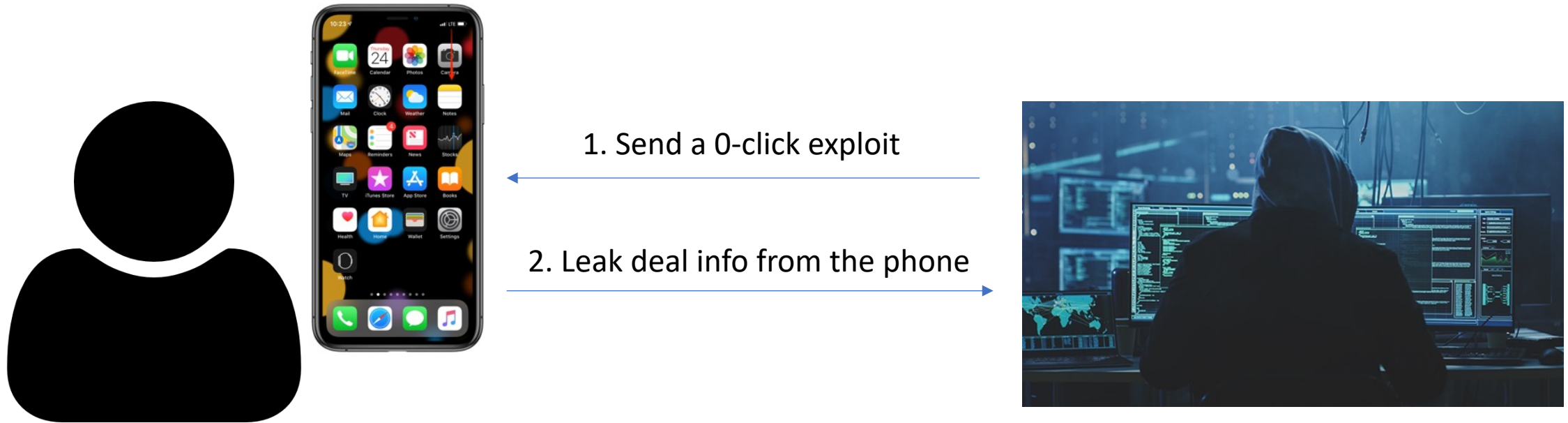
Software Attack Could be a CyberWeapon

- A fake scenario
 - You are a president of the United Counties of Earth, and you will have a huge deal with the Greasy Oil Company tomorrow. But you would like to know the minimum price that the Greasy Oil company accepts.
- In 1970



Software Attack Could be a CyberWeapon

- In 2022

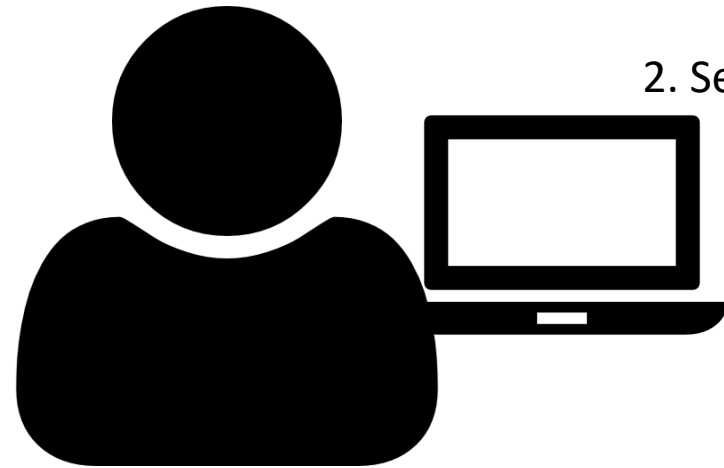


GOC executive
iMessage id: blue9057@gmail.com

Software Attack Could be a CyberWeapon

- In 2022

1. Prepare an exploit PDF document as a contract document



2. Send it as a contract docs for tomorrow

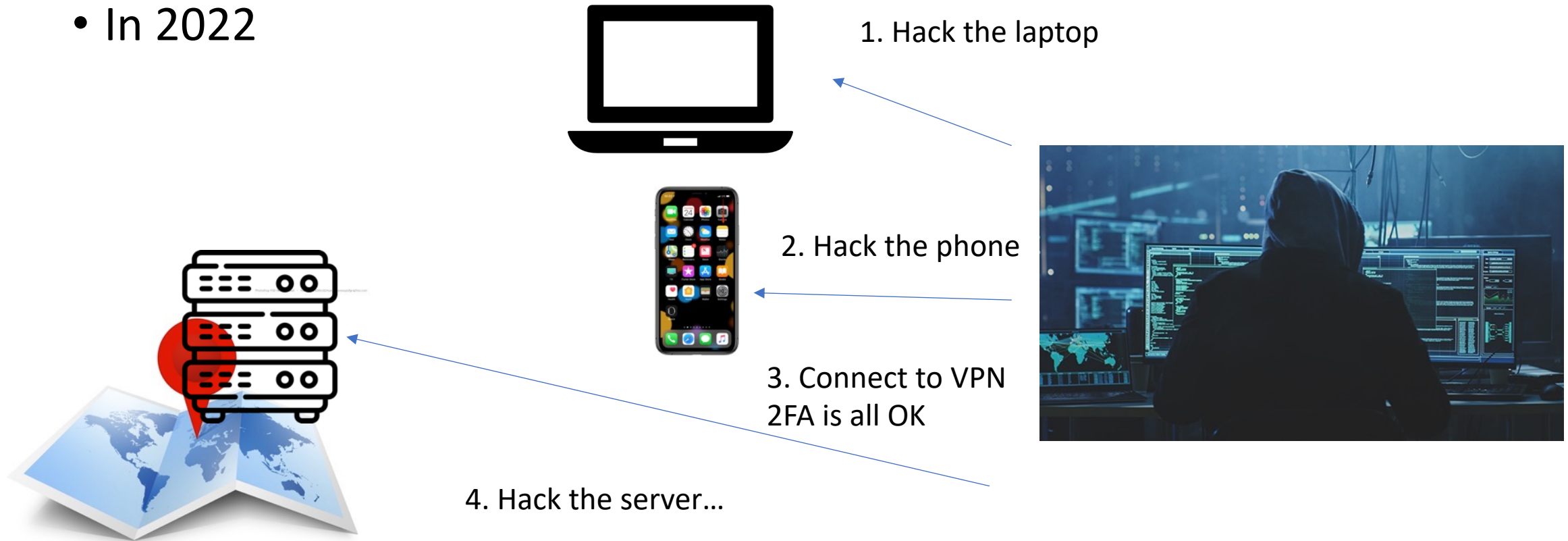
3. Leak deal info from the laptop



GOC executive
iMessage id: blue9057@gmail.com

Software Attack Could be a CyberWeapon

- In 2022



Software Attacks are Powerful

- No systems comes w/o vulnerabilities
 - I have a belief that all systems can be hacked (for now)
 - That's why my domain name is unexploitable.systems
- To assess the threat of software attack, think like
 - Anything connected to the Internet can be controlled by hackers
 - Anything receive inputs without physical touch can be controlled by hackers
 - Typing keyboards, mouse, etc...



Not Just for That...

- We are living in the era of Ubiquitous and Pervasive Computing
 - Computers are everywhere
- Smartphone – GPS, gyrometer, camera, mic, speaker, Internet, etc...
- Smartwatch – GPS, gyrometer, laser to measure heartrate, breathe, sleep, Internet, etc...
- Smartspeaker – Mic, speaker, Internet, etc...
- Smartpoweroutlet – capability to monitor/manipulate power, Internet, etc..

Totally Possible Attack Scenario

- 1. Hack a smartphone of a user
- 2. Record voice all time in background; you can turn a smartphone as an eavesdropping bug
- 3. When the user gets to home (locate via GPS on Smartphone), play an ultrasonic sound to command Smartspeaker
 - Alexa, order 100 copies of Yeongjin's security program from Amazon!

DolphinAttack: Inaudible Voice Commands

- FYI, human ear can hear audio in range of
- 20Hz—20KHz

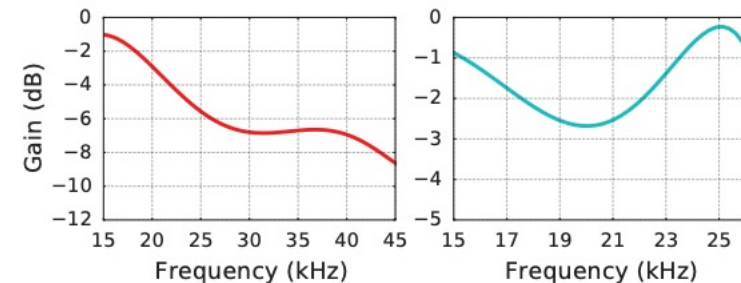
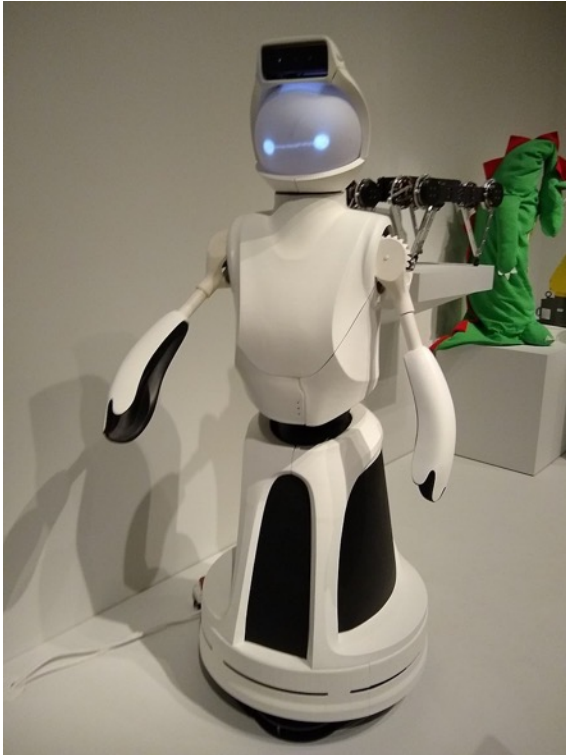


Figure 10: The frequency responses of the ADMP401 MEMS microphone (left) and the Samsung Galaxy S6 Edge speaker (right).

Totally Possible Attack Scenario

- What if you can hack a robot, or a drone?
 - Software attack -> Physical attack



Summary

- Compiler options can detect/prevent software attacks
 - Use them, and don't ignore their warnings and reports
- There are several ways to responsibly disclose vulnerabilities
 - Now community give rewards for finding/fixing their problems
 - Many companies opens bug bounty program -> report vulns here!!
 - If not, we can go through coordinated vulnerability disclosure (e.g., via ZDI)
- In the ubiquitous computing era, software attacks are serious matter. There is a huge cyberweapon market under the hood
- Thursday, we will learn about malware, for how such weapons could be used to launch a targeted attack (Nuclear power plant) and a mass attack (DDOS and ransomware)