Course Introduction

and some motivation

Mainack Mondal

CS60112
Spring 2025

Today’s Class

- Course logistics
- How to learn security?
- How to (not) nuke your system?
- Epilogue

Instructor

Mainack Mondal: usable security and privacy, system security and privacy, operationalizing privacy theories

Office: CSE 316

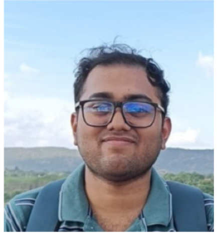
TA’s

Shiladitya De

SHILADITYA dot DE at kgpian.iitkgp.ac.in

Nimish Mishra

neelam dot nimish at gmail.com

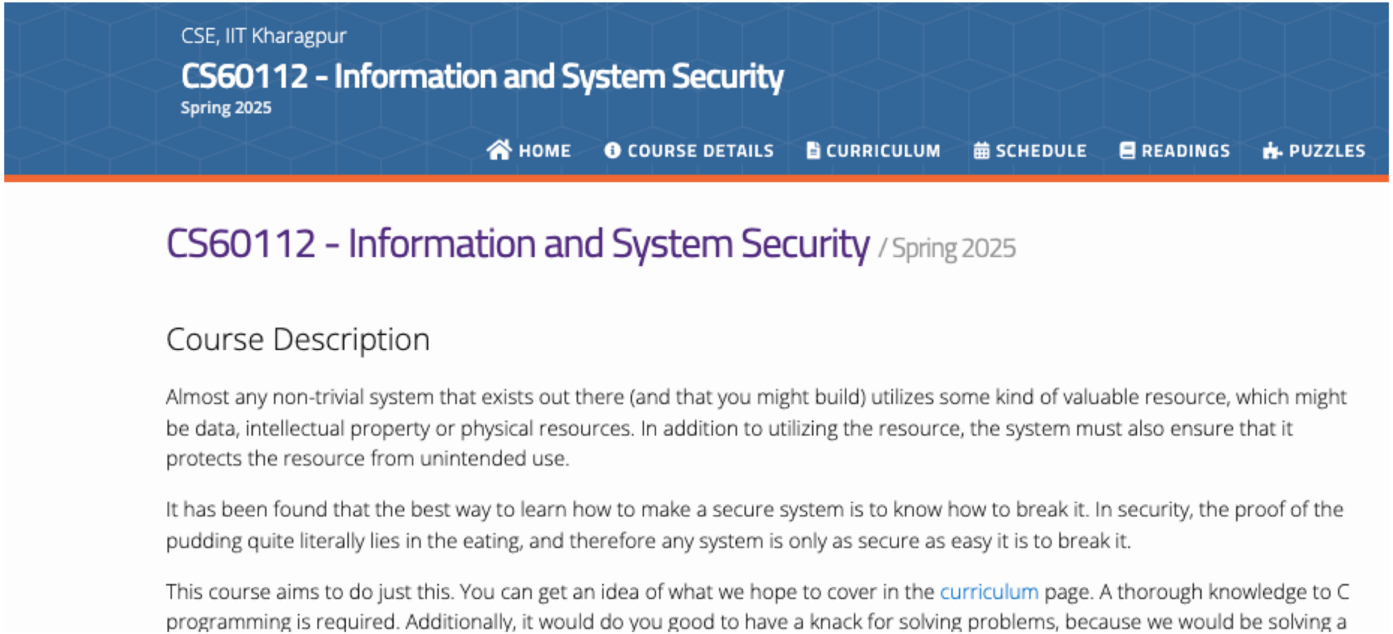


Sourabh Soumyakanta Das

Dassourabh103 at kgpian.iitkgp.ac.in

Website

https://kronos-192081.github.io/InfoSec-2025/



CSE, IIT Kharagpur

CS60112 - Information and System Security

Spring 2025

CS60112 - Information and System Security / Spring 2025

Course Description

Almost any non-trivial system that exists out there (and that you might build) utilizes some kind of valuable resource, which might be data, intellectual property or physical resources. In addition to utilizing the resource, the system must also ensure that it protects the resource from unintended use.

It has been found that the best way to learn how to make a secure system is to know how to break it. In security, the proof of the pudding quite literally lies in the eating, and therefore any system is only as secure as easy it is to break it.

This course aims to do just this. You can get an idea of what we hope to cover in the curriculum page. A thorough knowledge to C programming is required. Additionally, it would do you good to have a knack for solving problems, because we would be solving a

Course Timings

- Credit: 3-0-0

- Wednesday 10:00 am - 10:55 am
- Thursday 9:00 am - 9:55 am
- Friday 11:00 am - 11:55 am

Mode of Teaching

- Offline lectures
  - Please come to class (no recordings)
- (occasional) Pre-recorded lectures for special topics
  - We will upload the recorded lectures via MS Teams
- Two exams + Assignments

MS Teams

Link:

https://teams.microsoft.com/l/team/19%3AlF59SxbMNlF8qcEtIaabtg4wZyRFj8kfPVO_yfyZftU1%40thread.tacv2/conversations?groupId=86e1aa23-cbd8-477b-9b38-2d6f7b6f5dec&tenantId=71dbb522-5704-4537-9f25-6ad2dcd4278d

Or

https://shorturl.at/5UUW7

Code: **m6axutv**

Course evaluation: Exam

Two exams (55%)

- Syllabus: Everything until that point
- Dates will be in the webpage and announced in academic calendar

Course evaluation: Assignments

Regular assignments (45%)

- You will learn this course mostly through the assignments
- Assignments will comprise CTF style problems
- Start forming teams! (size <= 3)

Course logistics

Questions?

Ethical Considerations

Source: https://myozonelayer.com/2016/11/22/the-4th-monkey-do-no-evil/

Ethical Considerations

- Don’t do evil
- If you feel it is wrong, it is wrong
- Cyber offenses are punishable by law
- Use your tools responsibly (“It was an accident, Milord!” won’t hold up)

Today’s Class

- Course logistics
- How to learn security?
- How to (not) nuke your system?
- Epilogue

How to learn Security?

You first learn how to break systems!

How to learn Security?

Then you'd know what to <u>not</u> do!

CTF

Capture the Flag (CTF) is a kind of security exercise, where the aim is to “break” the system in order to retrieve a piece of text, called the “flag”.

CTF rules

- The flag must be submitted **as-is** (no modifications)
- Checking will be **automated**
- Flags will be **randomized** on a per-team basis
- Sometimes the flag will have a specified format, such as “drapeau{<flag-text>}”. You need to submit the **whole thing**.

Leaderboard

- We will have a leaderboard for assignments.
- All points add up.
- Top three teams to get cash prizes!

Topics

- Web security
- Reverse Engineering
- Pwning
- Cryptography
- Hardware

Topic 1: Web security

- Perhaps the source tells you something
- Perhaps you can create a url which will lead you to a secret
- …
- Goal: find vulnerabilities

Topic 2: Reverse engineering

- Auditing binaries (static, dynamic)
- Understand the binary code / file
- ...
- Goal: check if you can find vulnerabilities using that knowledge

Topic 3: Pwning

- Pwning -> **O**wning
- Find and exploit vulnerabilities to obtain access to a system

Topic 3: Pwning

- Demo?

Topic 3: Pwning

- https://www.programiz.com/c-programming/online-compiler/

```
#include <stdio.h>
int main() {
    char *buf = "iss2024";
    puts(buf);
    return 0;
}
```

What would it print?

Topic 3: Pwning

- https://www.programiz.com/c-programming/online-compiler/

```
#include <stdio.h>
int main() {
    char *buf = "iss2024";
    puts(buf);
    return 0;
}
```

What would it print? **Make it a bit more interesting?**

Topic 4 / 5: Cryptography / HW

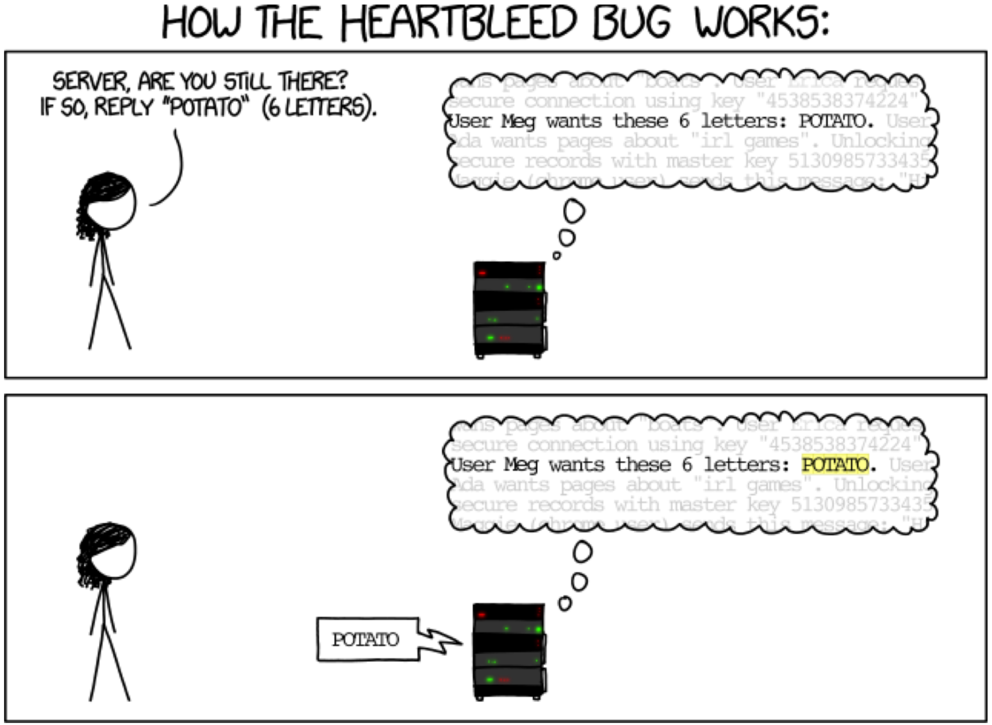
- Will cross the bridge when we get there

Topic 4 / 5: Cryptography / HW

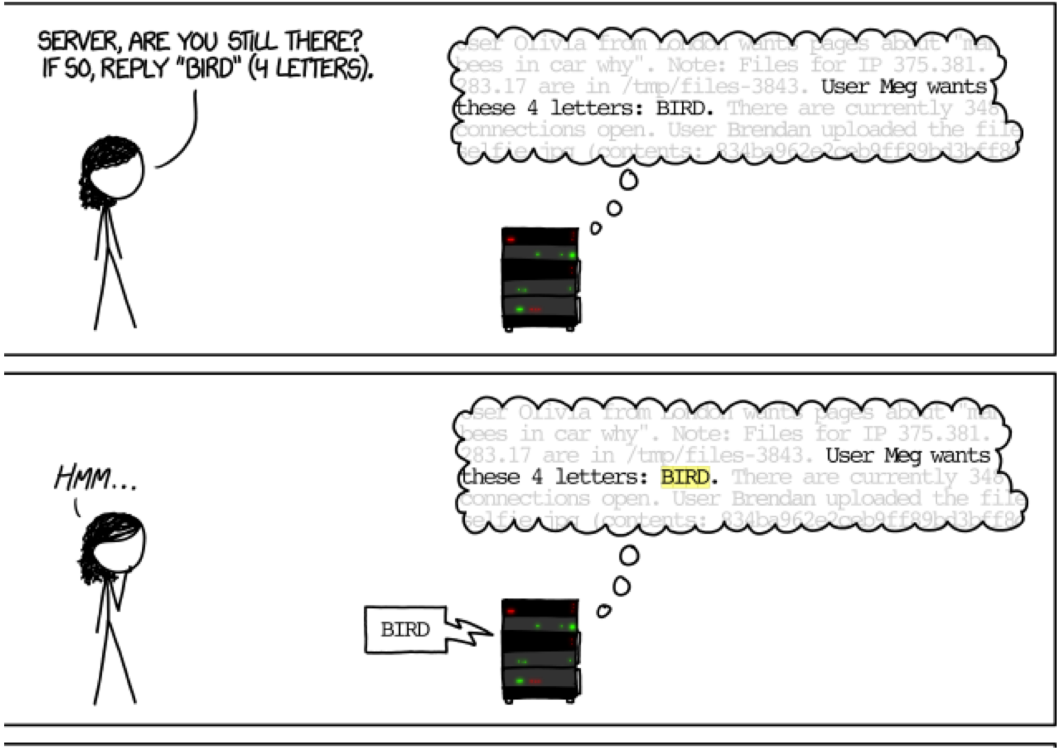
break

- Will ~~cross~~ the bridge when we get there

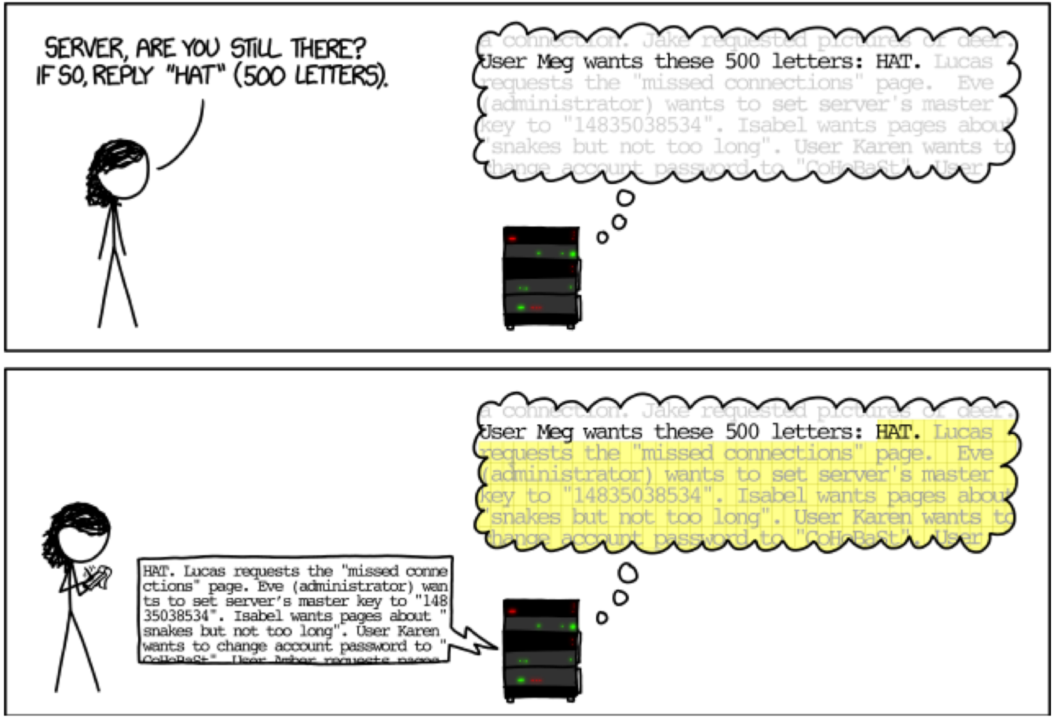
Case study 1: Heartbleed (1/3)

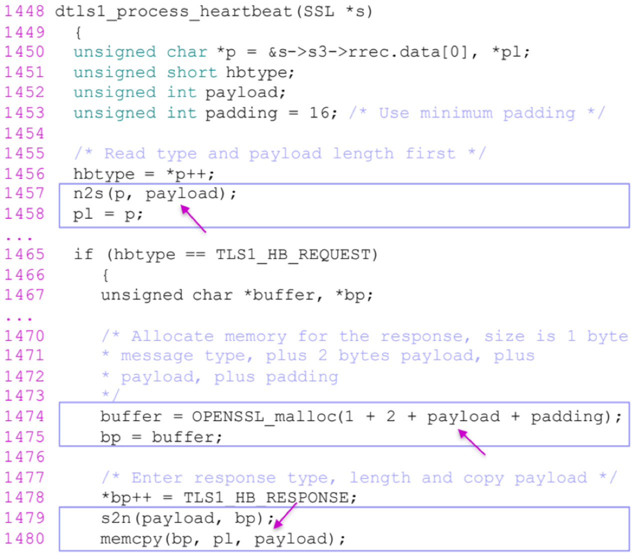

HOW THE HEARTBLEED BUG WORKS:
SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
ans pages about "boats". User Erica reques secure connection using key "4538538374224". User Meg wants these 6 letters: POTATO. User Ada wants pages about "irl games". Unlocking secure records with master key 5130985733435 Maggie (chrome user) sends this message: "H
ans pages about "boats". User Erica reques secure connection using key "4538538374224". User Meg wants these 6 letters: POTATO. User Ada wants pages about "irl games". Unlocking secure records with master key 5130985733435 Maggie (chrome user) sends this message: "H
POTATO

Case study 1: Heartbleed (2/3)


SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Olivia from London wants pages about "ma bees in car why". Note: Files for IP 375.381. 283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 346 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
HMM...
User Olivia from London wants pages about "ma bees in car why". Note: Files for IP 375.381. 283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 346 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
BIRD

Case study 1: Heartbleed (3/3)


SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User Amber requests pages

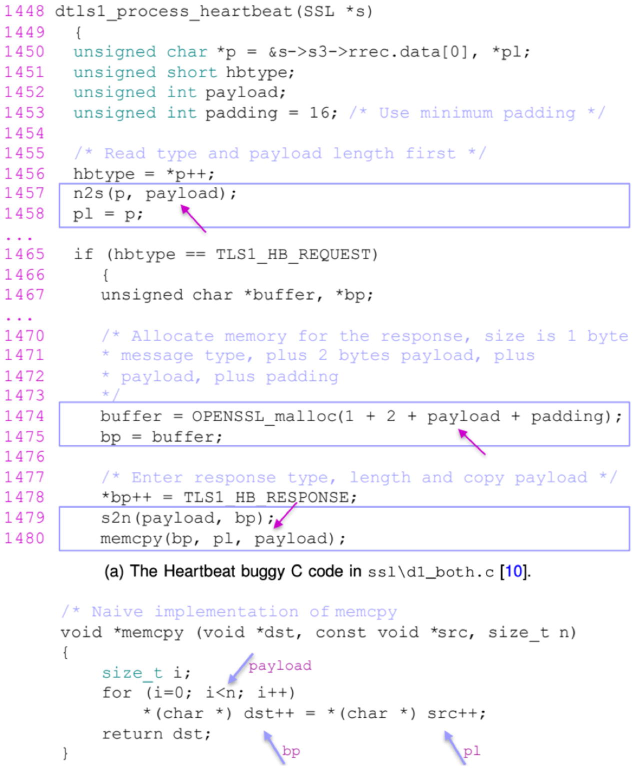


```
dtls1_process_heartbeat(SSL *s)
  {
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */

  /* Read type and payload length first */
  hbtype = *p++;
  n2s(p, payload);
  pl = p;
...
  if (hbtype == TLS1_HB_REQUEST)
     {
     unsigned char *buffer, *bp;
...
     /* Allocate memory for the response, size is 1 byte
     * message type, plus 2 bytes payload, plus
     * payload, plus padding
     */
     buffer = OPENSSL_malloc(1 + 2 + payload + padding);
     bp = buffer;

     /* Enter response type, length and copy payload */
     *bp++ = TLS1_HB_RESPONSE;
     s2n(payload, bp);
     memcpy(bp, pl, payload);
```

(a) The Heartbeat buggy C code in ssl\d1_both.c [10].



```
dtls1_process_heartbeat(SSL *s)
  {
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */

  /* Read type and payload length first */
  hbtype = *p++;
  n2s(p, payload);
  pl = p;
...
  if (hbtype == TLS1_HB_REQUEST)
     {
     unsigned char *buffer, *bp;
...
     /* Allocate memory for the response, size is 1 byte
      * message type, plus 2 bytes payload, plus
      * payload, plus padding
      */
     buffer = OPENSSL_malloc(1 + 2 + payload + padding);
     bp = buffer;

     /* Enter response type, length and copy payload */
     *bp++ = TLS1_HB_RESPONSE;
     s2n(payload, bp);
     memcpy(bp, pl, payload);
```

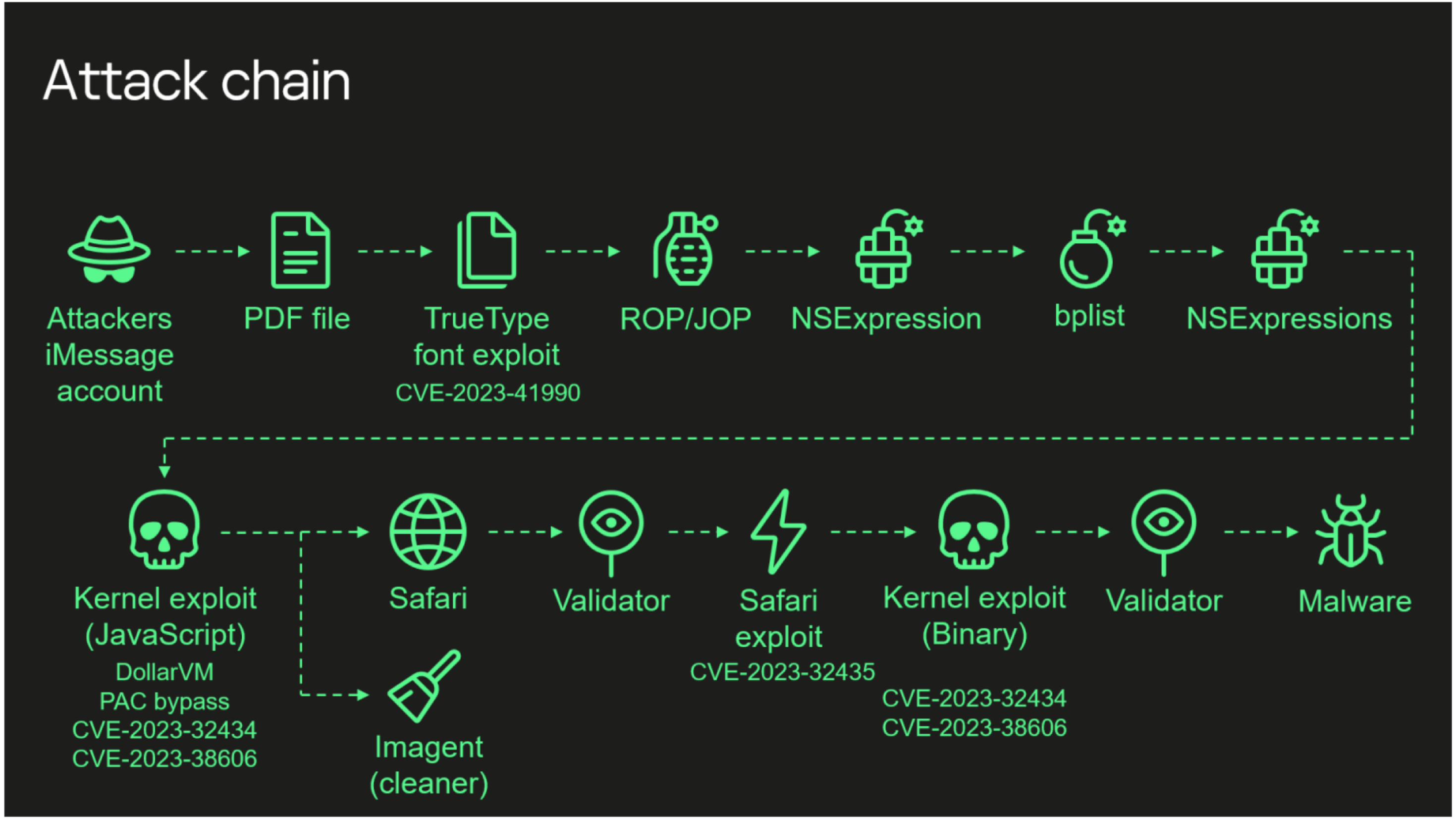
(a) The Heartbeat buggy C code in `ssl\d1_both.c` [10].

```
/* Naive implementation of memcpy
void *memcpy (void *dst, const void *src, size_t n)
{
    size_t i;
    for (i=0; i<n; i++)
        *(char *) dst++ = *(char *) src++;
    return dst;
}
```

(b) A naive C implementation of the `memcpy()` function.

FIGURE 1: Analysis of Heartbleed.

Case study 2: Operation Triangulation (1/2)



Case study 2: Operation Triangulation (2/2)

However apple had hardware protection against writing random memory addresses

Case study 2: Operation Triangulation (2/2)

- However apple had hardware protection
  - Against writing random memory addresses
- Bypassed using undocumented registers
  - possibly there for debugging

https://securelist.com/operation-triangulation-the-last-hardware-mystery/111669/

Today’s Class

- Course logistics
- How to learn security?
- How to (not) nuke your system?
- Epilogue

How to nuke your system?

- Run untrusted executables/programs on your system
- Turn off kernel security features
- Stick “sudo” before random commands
- Run “sudo rm -rf --no-preserve-root /”

How to NOT nuke your system?

Ignore all the previous advice and instead:

- Use Docker (has caveats)
- Use virtualization
  - VirtualBox
  - QEMU

Docker

- Lightweight “virtualization” software, with the caveat being it requires root to run (so be careful with mounted volumes).
- Kernel shared with host with namespace based isolation.
- Install Docker Engine (required).

VirtualBox

- Full-system virtualization software, with a nice GUI.
- Install from here (required).
- Also install the extension pack and guest additions for nicer integration.
- We will provide VirtualBox VM files (with extension .ovf) when required.

QEMU

- Full-system virtualization + architecture emulation software, CLI only.
- Install from here (required).
- QEMU typically requires a long list of flags and a disk-image file to run. We will provide both.
- Helpful in emulating non-native ISAs (e.g., MIPS, RISC-V)

Today’s Class

- Course logistics
- How to learn security?
- How to (not) nuke your system?
- Epilogue