



CAPTURE THE FLAG!!

C R A C K , L E A R N , S E C U R E :
S T E P I N T O C Y B E R S E C U R I T Y !



Index

01

What are
CTFs?

03

Pirating
Games (Rev)

05

Hiding Secret
Information

07

What now?

02

Hacking
Facebook

04

Cryptography

06

How to get
hired by the
CIA



Who we are..

We are a group of Open Source Enthusiasts who focus on something more preliminary and relevant, "A Love for Coding".

Every year KOSS holds events for familiarizing students with UNIX tools, Linux environment, Git Development Workflow, besides conducting workshops on GUI programming, Web development, and Open Source Programs.





Why CTFs?

Apart from learning, CTFs are fun! And the fun doubles when you take part in CTFs as a team!

CTFs help you learn how systems operate, identify weaknesses, and think like an attacker. This perspective is key in protecting systems. They also open up career opportunities in digital security, helping you gain hands-on experience in the field.





What are CTFs

kossCTF{babys_first_ctf_!!!}

CTFs are (primarily online) competitions, that are essentially hacking tournaments.

- The idea is that you have to recover a string (the flag) from a (supposedly) secure system.

We will be conducting a very simple CTF alongside this presentation for you to follow along with. Please make your accounts.

Some rules:

- Don't share flags
- Don't attack the website directly
- You are allowed to use any resources available to you, but try not to GPT (or Deepseek). There's no special prize for solving them all
- You will receive plenty of hints on solving these during the presentation



10.147.8.22:8000/



Web Exploitation



01
What happens
at google.com?

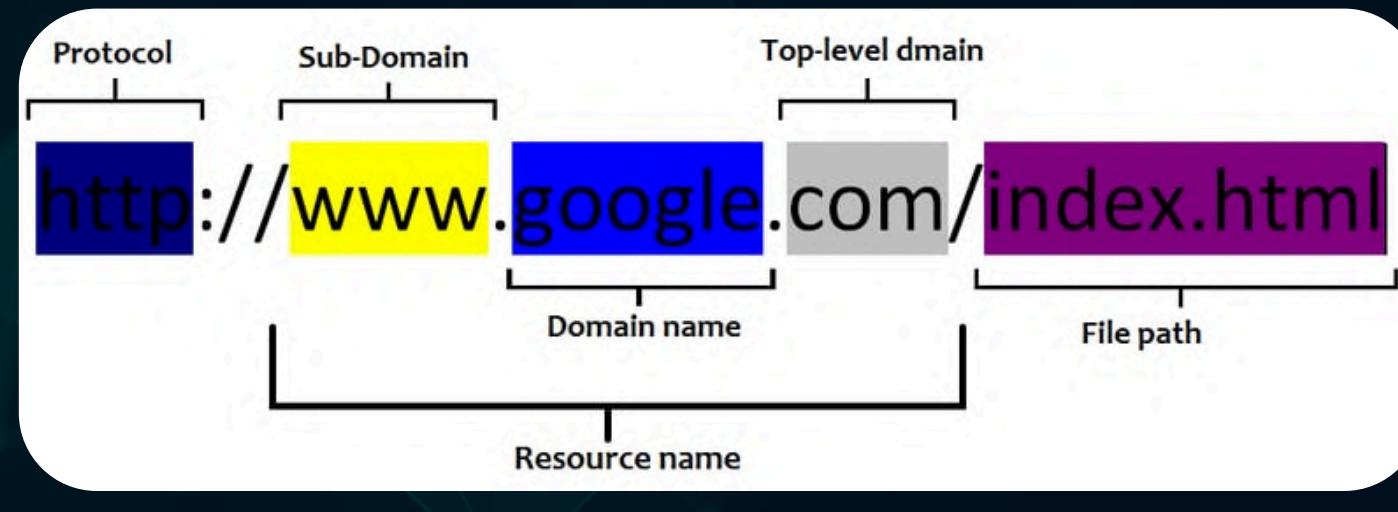
03
Messing with
web servers

02
Browser tools

04
Famous
Vulnerabilities

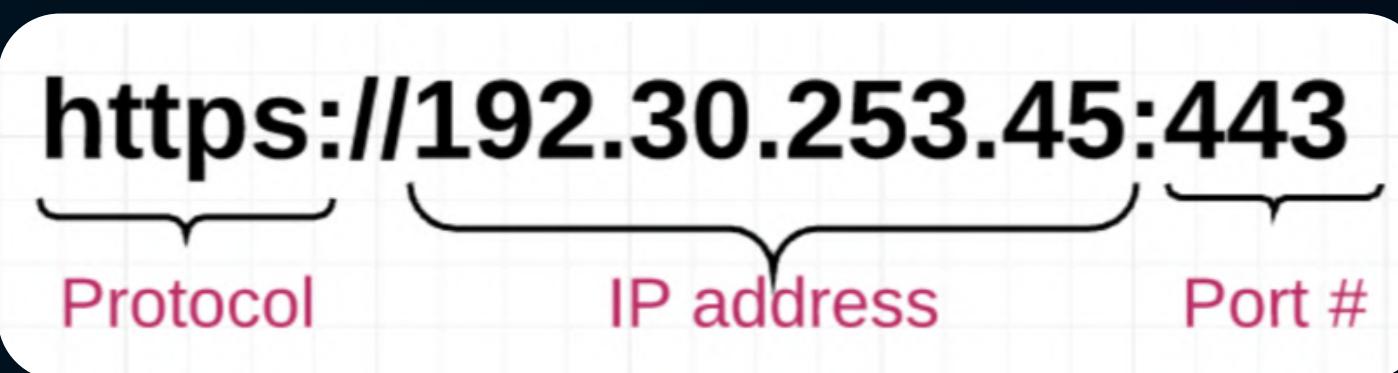


What happens when you enter "google.com"



Request

- When you type a URL in the browser, it parses the information contained in the URL and a request is sent to your internet service provider (ISP)

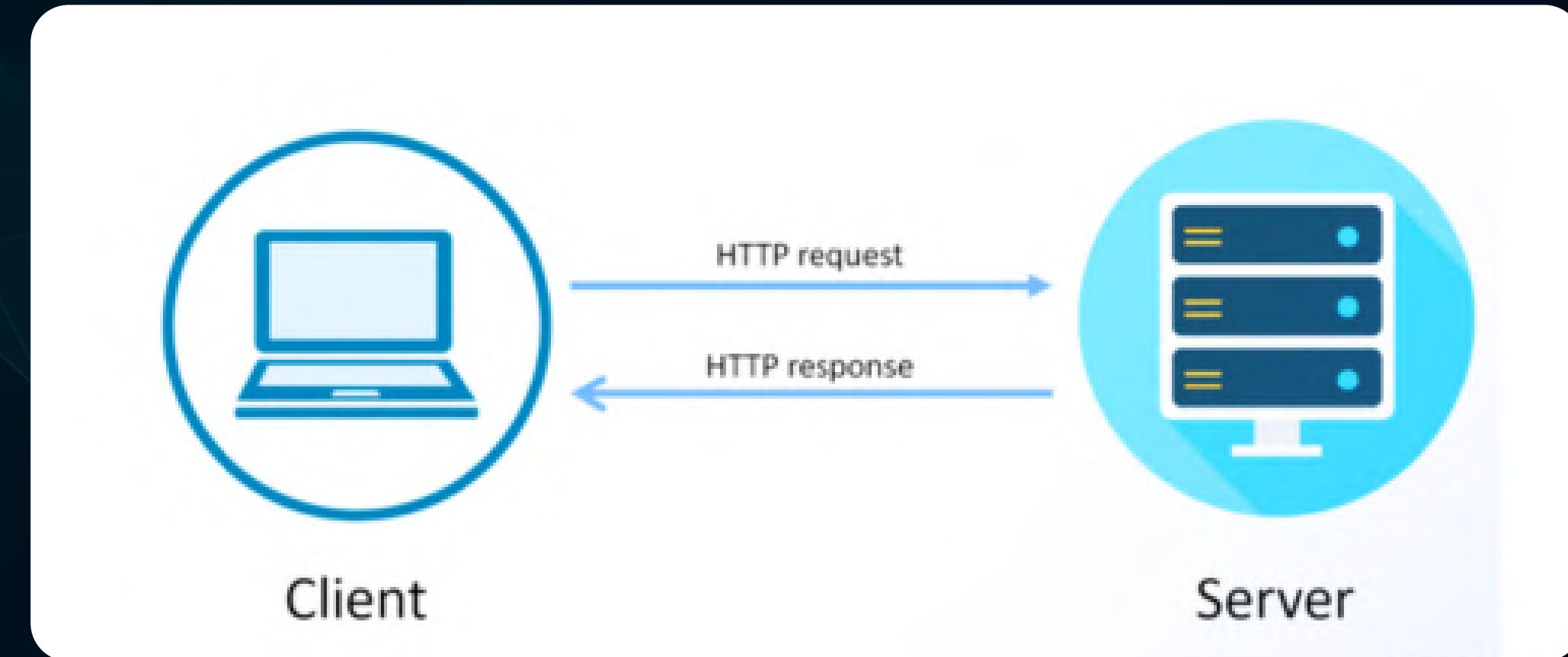


Response

- The server does a quick lookup and returns the IP address of the websites which redirects you to the main pages of the site



What happens when you enter "google.com"



A request is made by client to the server asking for resources, like a webpage or data. A response is the server's reply, containing the requested data or error message.



How to use your browser tools?

Web Console

The screenshot shows the 'Console' tab selected in the developer tools. The toolbar includes icons for Inspector, Debugger, Network, Style Editor, Performance, and Memory. Below the toolbar, there's a 'Filter Output' section with dropdowns for Errors, Warnings, Logs, Info, and Debug. A prominent yellow warning message at the top states: '⚠ Some cookies are misusing the recommended "SameSite" attribute 6'. The main content area is currently empty.

- In the Inspect Console of a browser, it's possible for malicious code to be executed through injections, such as JavaScript.(An example of Cross-site Scripting)

Storage

The screenshot shows the 'Storage' tab selected in the developer tools. The sidebar on the left lists storage categories: Cache Storage, Cookies, Indexed DB, Local Storage, and Session Storage. Under 'Cookies', there are two entries for 'https://developer.mozilla.org': '_ga' and 'csrftoken'. The main area is a table with columns: Name, Domain, Path, Expires on, Last accessed on, Value, table.h..., and sameSite. The table data is as follows:

Name	Domain	Path	Expires on	Last accessed on	Value	table.h...	sameSite
_ga	.mozilla.org	/	Tue, 23 Feb 2021 1...	Sun, 23 Jun 2019 1...	GA1.2.1812154749.1...	false	Unset
csrftoken	.developer....	/	Mon, 08 Jun 2020 ...	Sun, 23 Jun 2019 1...	65MCOsFkEef3AFEo...	false	Unset

- Shows the data a website stores in the browser like cookies and session data.



How to use your browser tools?

Network Monitor

The screenshot shows a browser's developer tools Network tab. The interface includes a toolbar with tabs like Console, Inspector, Debugger, Network, Style Editor, Performance, and a status bar at the bottom. The main area displays a table of network activity:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
302	GET	locked developer...	/	document	html	7.34 KB	28.9...	161 ms
200	GET	locked developer...	/en-US/	document	html	8.90 KB	28.9...	15 ms
200	GET	locked developer...	ZillaSlab-Boldsubset.0beac26b.woff2	font	octet...	33.87 KB (ra...	33.2...	16 ms
204	POST	locked www.bin...	xls.aspx	URK8juAgKq...	plain	10.27 KB	0 B	109 ms
204	POST	locked www.bin...	xls.aspx	URK8juAgKq...	plain	3.55 KB	0 B	109 ms
200	GET	locked developer...	main.f9e86cd5.chunk.css	stylesheet	css	10.41 KB	52.0...	0 ms
200	GET	locked developer...	runtime-main.ef8b3670.js	script	js	2.43 KB	3.92...	15 ms
200	GET	locked developer...	4.7bcccd08.chunk.js	script	js	51.77 KB	166....	15 ms
200	GET	locked developer...	main.79bfcd27.chunk.js	script	js	18.15 KB	61.7...	15 ms
200	GET	cdn.spee...	lux.js?id=108906238	script	js	6.62 KB (rac...	17.3...	544 ms
304	GET	locked developer...	ga.js	script	js	cached	1.64...	16 ms
GET	GET	locked developer...	ZillaSlab-Regularsubset.ce3a756d.woff2	font	octet...	33.02 KB (ra...	33.0...	0 ms

At the bottom, the stats are: 19 requests, 432.28 KB / 194.58 KB transferred, Finish: 2.58 s, DOMContentLoaded: 887 ms, load: 2.24 s.

- Used to analyze a website's network traffic, to view requests, API calls, and responses.
- Interception of these requests, which is useful for identifying vulnerabilities.



How to mess with web servers?

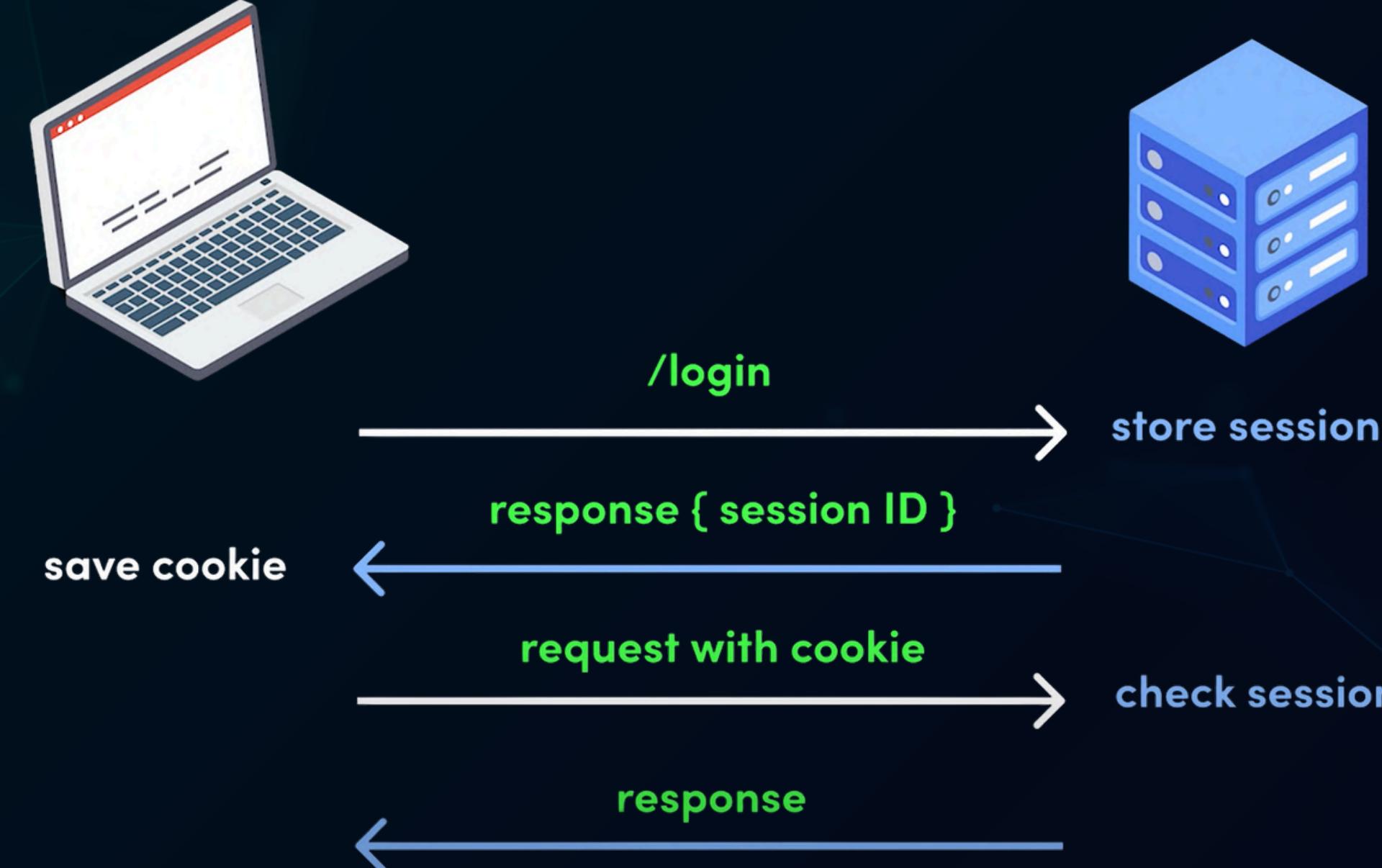
01 Cookie based
authentication

02 Path traversal





Cookie-based Authentication



Cookie based authentication can be bypassed if the attacker intercepts or modifies the session cookie. Let's try it out!

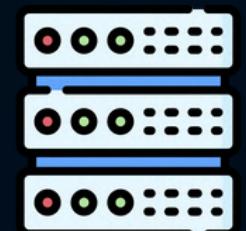


Path Traversal



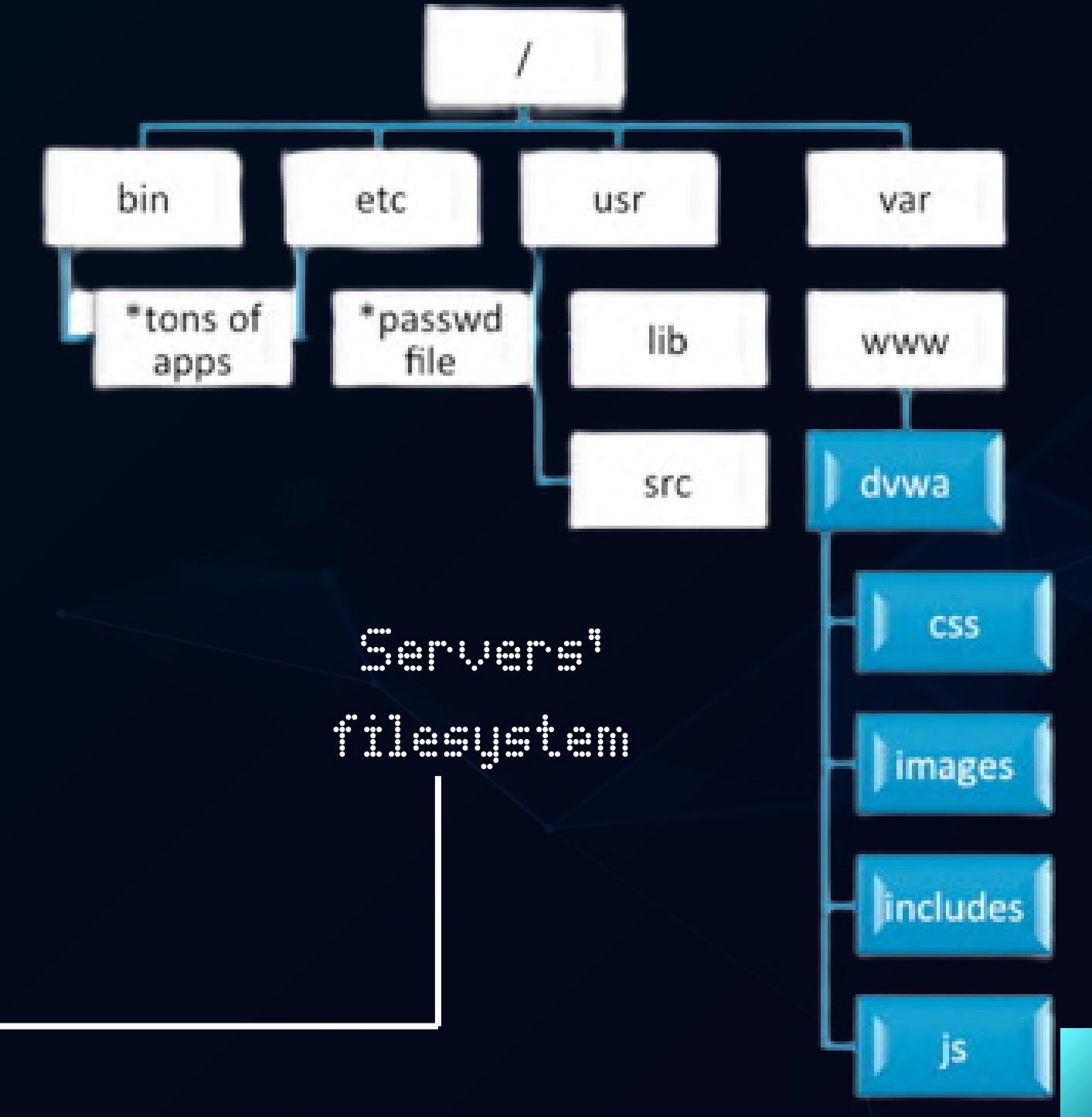
Hacker

`https://example.com/?filename=fastly.png`
`https://example.com/?filename=../../../../etc/passwd`



Request to
the server

```
[malsaid@HAL ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
```





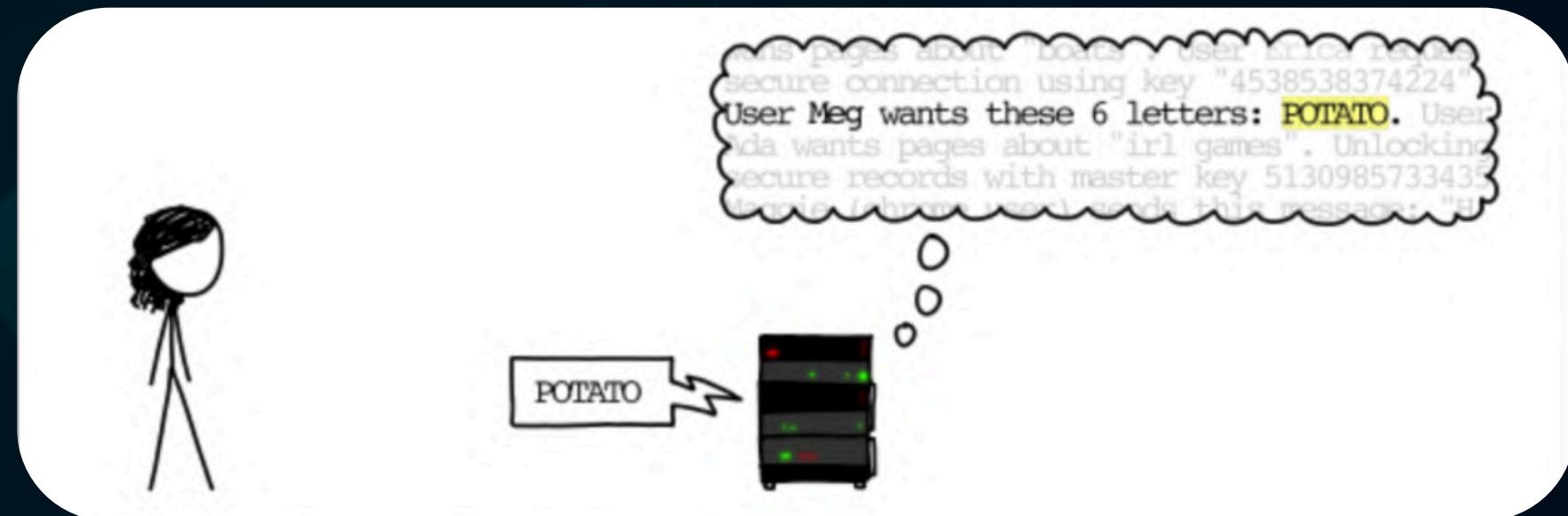
Famous Web Vulnerabilities

01
Heartbleed

02
Large scale
DDoS



Heart Bleed



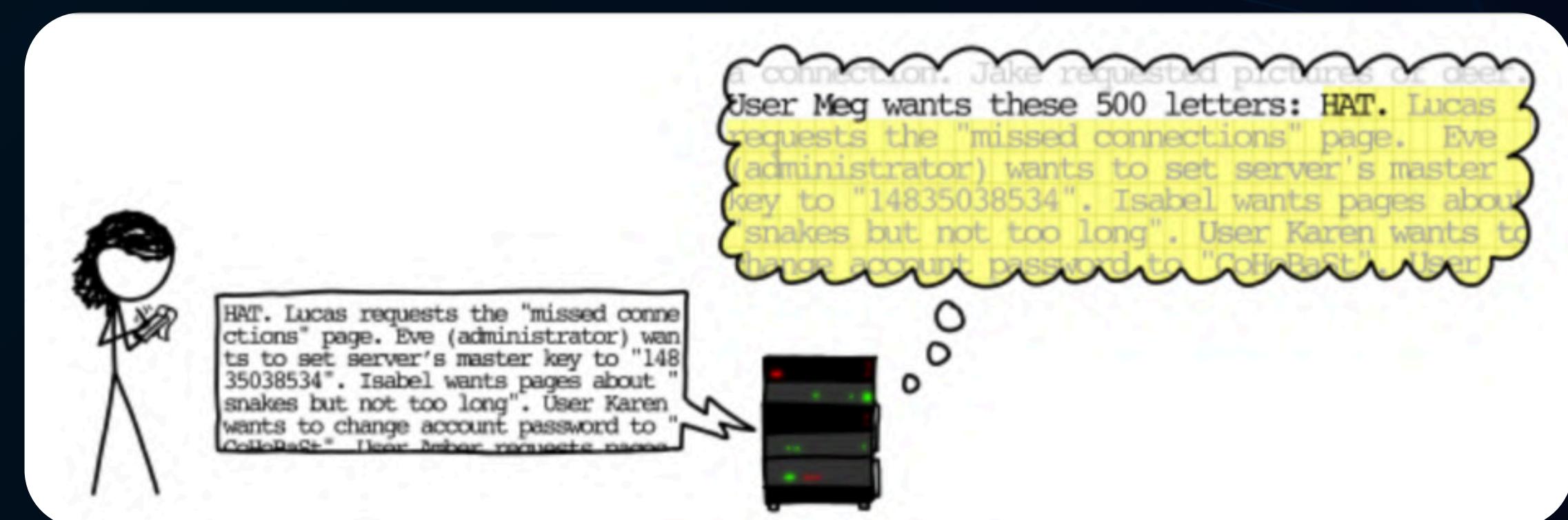
- This bug allows attackers to perform man-in-the-middle attacks on websites, by taking advantage of a loophole in TLS encryption provided by OpenSSL.



- This consists of attacker sending a heartbeat request with higher payload size.

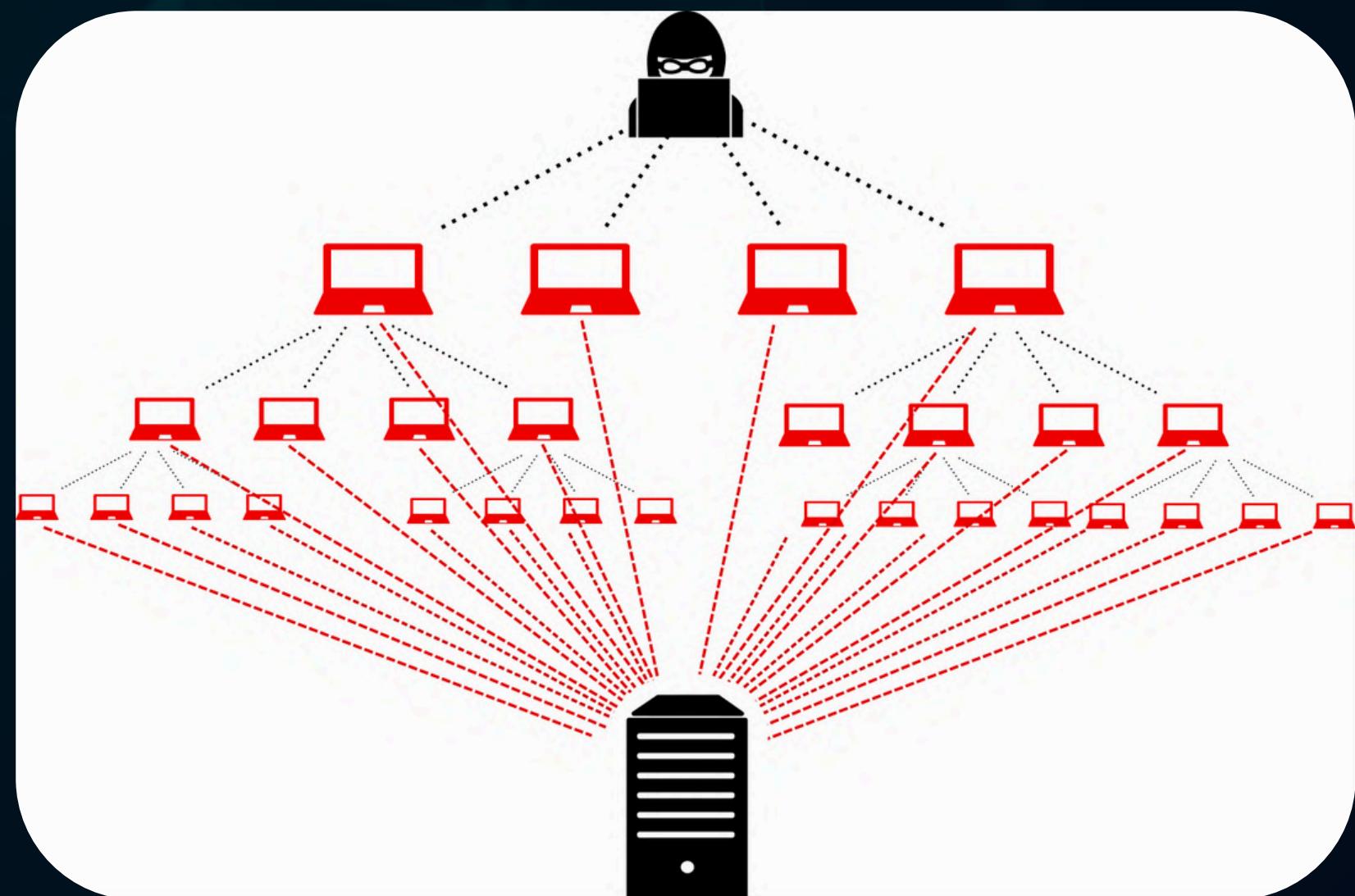


Heart Bleed





Large Scale DDoS



- In a DDoS attack many computers (bots/zombies forming a botnet) controlled by a hacker flood a server with fake requests avoiding legitimate users from accessing it.
- Can occur over TCP/UDP/HTTP networks



Cryptography



Q1
A way to send
secrets

Q3
Actually
secure Crypto

Q5
Gravity Falls!!!

Q2
Common
Ciphers

Q4
Quantum
Cryptography



A way to send secrets

What it is: Fancy word for making sure your secrets stay secret

Why it matters:

- Keep it Private: Only the right people can see the info.
- No Tampering: Making sure no one messes with your stuff.
- Prove It's You: Confirming who you are.



Common Ciphers and Encryptions

01

Caesar Cipher

02

Vigenere Cipher

03

XOR

04

base64

05

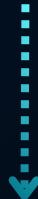
Hashing



Back in the days of Julius Caesar

Julius Caesar was already encrypting his messages using the Caesar Cipher. It's basically shifting letters around to make messages harder to read.

Some Cool Text



Uqog Eqgn Ugzv

But here is the thing, even a child can crack this now.



Cryptography now

But we have come a long way

Some Cool Text



b767991c582edb5b51216c02e6fdfde5993b4547df4621582112cc

98a796860a

Getting back the text from these scrambled letter is next to impossible.



Vigenère Cipher

The Vigenère Cipher is a method of encrypting text by using a series of Caesar ciphers based on a keyword. It is a polyalphabetic substitution cipher, which means it uses multiple alphabets to encrypt the message, making it more secure than a simple Caesar cipher.



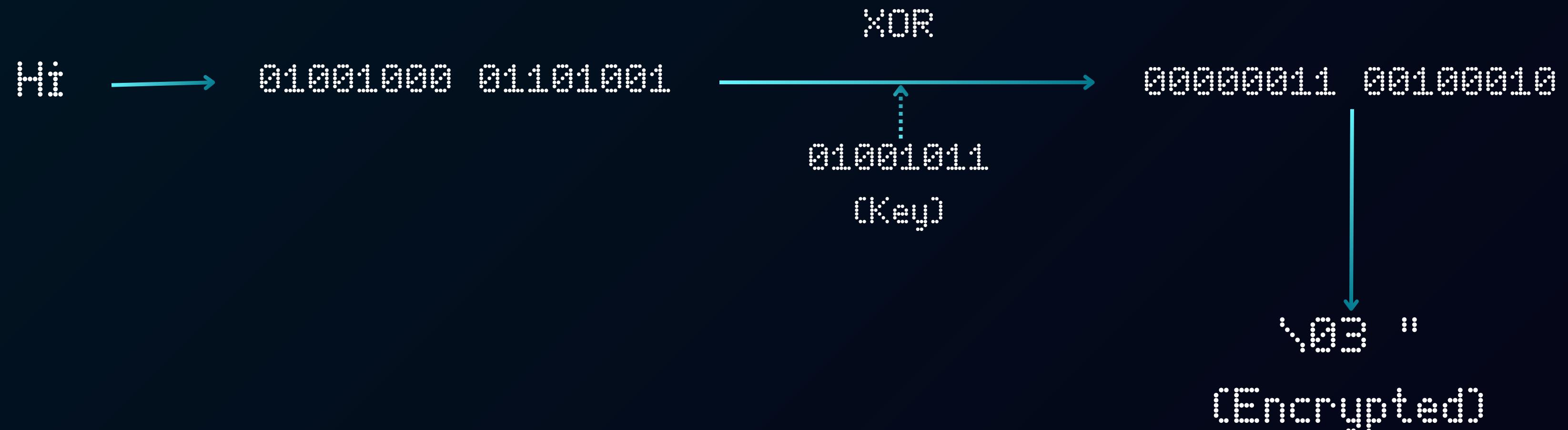
It shifts each alphabet just like in Caesar Cipher but this time, each letter is shifted by a different number, based on a key provided to it.



XOR

I guess you do know how information is stored in computers. It's stored in binary i.e., it consists of only zeroes and ones.

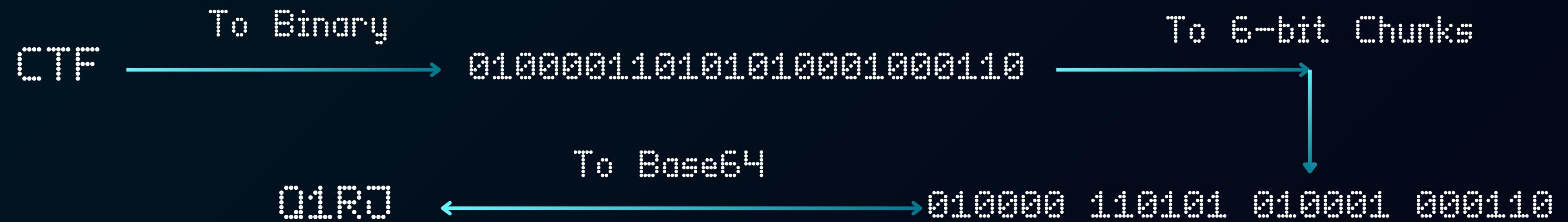
Our next encryption trick works on the binary representation of data and encrypts it using a provided key.





Base64

Base64 represents binary data in an ASCII string format. It encodes data into Base64 characters (6 bits each).



This is used for encoding API keys and Tokens. However, this is NOT an encryption. It can be easily decoded and hence not used to store sensitive data. If there are not enough bytes to make 4 chunks, '=' is added as padding.



Hashing

Now, this is where it gets interesting

Hashing is a process that converts data into a fixed-length string using a mathematical function. The output, called a "Hash", is unique to the input, meaning even a small change in the original data produces a completely different hash.

Remember the freaky encryption shown at the starting. That is an example of Hashing.



Hashing

Some Cool Text



b767991c582edb5b51216c02e6fdfde5993b4547df4621582112cc
98a796860a

This is an example of SHA-256, an industry level encryption.

Here is the cool part. This cannot be reversed. Hashing is a one-way process and is used for storing passwords



Modern Cryptography

The last encryption we took a look at was awesome...
But what if I want a way to decrypt my data.

RSA : This is one of the first public-key cryptosystems and is foundational to modern cryptography. It uses two keys: a public key (for encryption) and a private key (for decryption).

RSA encryption can be theoretically reversed, but with the current compute power we have, it will take thousands of years to decrypt it without the private key.



Modern Cryptography

AES: A symmetric encryption algorithm that encrypts data using a secret key. But here's the catch, the same key is used to encrypt and decrypt data.

Like RSA, it is also reversible, but is difficult to crack with the present compute power.



Quantum Cryptography: The Future of Security

Quantum cryptography leverages the principles of quantum mechanics to create unbreakable encryption.

Unlike traditional cryptography, which relies on mathematical complexity, quantum cryptography is based on the laws of physics, making it immune to computational attacks—even from quantum computers.

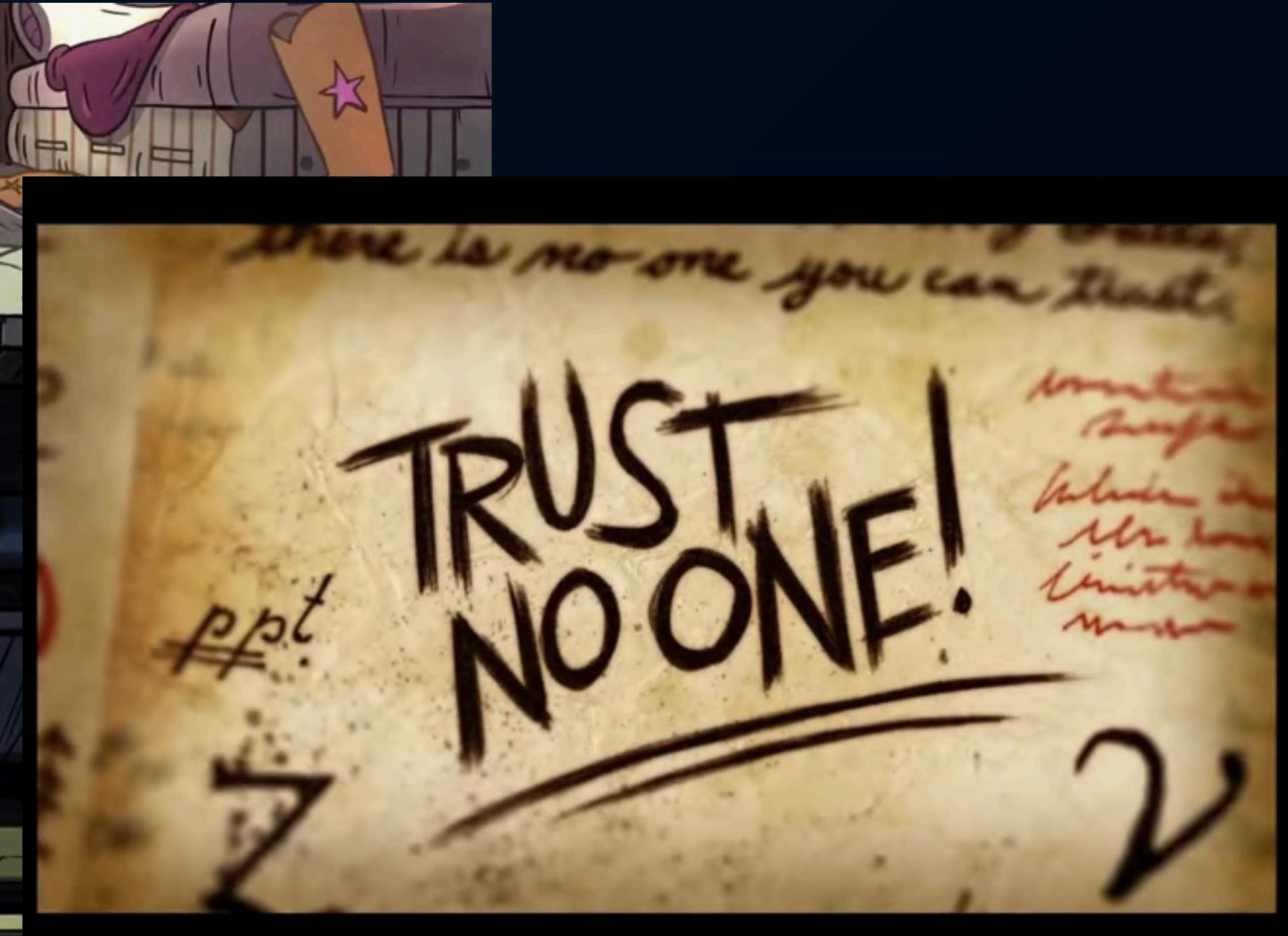


Heisenberg
Uncertainty
Principle

That's right... Even a quantum computer bows in front of the Laws of physics.



Gravity Falls!!! :3



5-19-23-6-21-16 18-9-6 4-16-19 22-12-15-10-20-19-25-19



Gravity Falls!!! :3



YM'KL ECN PPK WFOM UBR KQVXNLK,
DCI SIK'U VDA JFTOTA AYQ BWL VVCT "EBTGGB BHWKGZH" HVV:
TMEASZFA LOS YCDT PRWKTIYEKGL DBV XQDTYRDGVI





Reverse Engineering

Q1

What is a
binary? (PDS)

Q3

How can you
reverse it?

Q5

Getting our
hands dirty

Q2

C is not a
safe language

Q4

Quick
pre-requisite
knowledge



Story of a binary file

When a developer writes a program, they make it machine readable by compiling it with a compiler, which converts the code to a file containing 0s and 1s. The output file is called our "Binary File".

Source code ➡️ Compiler ➡️ Linker ➡️

User written
code in any high
level language

Converts the
source code to
machine code

Adds the
dependencies
and libraries



A single
executable
binary file

.....

Windows have .exe/.dll
files

Unix like systems have
ELFs .out/.so files

Android APKs has DEX
files

Vulnerabilities in a binary file

Why is C unsafe?

C is considered unsafe because it lacks built-in memory safety features. It allows direct memory access, meaning you can read or write beyond allocated memory, leading to buffer overflows. Attackers exploit this to overwrite critical data, execute arbitrary code, or crash programs.

Unsafe memory access can lead to serious security exploits, including game hacking, DRM bypassing, router firmware modifications, and privilege escalation

Vulnerabilities in a binary file

Why is C unsafe?

```
int main() {
    char buffer[8];
    int secret = 1234;

    printf("Enter input: ");
    scanf("%s", buffer);
    //overwrites secret if input>8chars
    printf("Buffer: %s\n", buffer);
    printf("Secret: %d\n", secret);

    return 0;
}
```

```
int main() {
    unsigned int x = UINT_MAX;
    printf("Before overflow: %u\n", x);
    x = x + 1; // No warning
    printf("After overflow: %u\n", x);
    return 0;
}
```

```
int main() {
    int arr[7] = {1, 2, 3, 4, 5, 6, 7};
    printf("Valid: %d\n", arr[6]);
    printf("Invalid: %d\n", arr[8]); // Unsafe access
    return 0;
}
```

```
int main() {
    int *ptr = malloc(sizeof(int));
    *ptr = 69;
    printf("Before free: %d\n", *ptr);
    free(ptr);

    printf("After free: %d\n", *ptr);
    return 0;
}
```



Reversing a binary file

Well...it's easy to generate binary from source code, it's not that easy to get the source code from the pool of 0s and 1s.

Heyy, tension not when Ghidra
is here...



Ladies and gentlemen, gather 'round! Allow me to introduce Ghidra, the powerhouse of software reverse engineering!

For quick decompilation,
Use: dogbolt.org



meet Ghidra—the NSA's powerful, open-source reverse engineering tool. It's free, user-friendly, and perfect for unraveling complex code.





Quick prerequisite knowledge

Static Analysis : Inspects binaries without execution.

Dynamic Analysis : Inspects binaries with execution, to see how it's interacting with system. Unsafe if it contains malicious code. (Run in sandboxed environment)

Static Analysis tools: Ghidra, IDA pro

Dynamic Analysts tools: GDB, Ltrace

- file <pathToFile>: Get file info
- ls -l <pathToFile>: Get file permissions
- chmod +x <pathToFile>: Give executable permission



Getting our hands dirty



Forensics



01
Sniffing for
information

03
Image
forensics

05
Get Netflix
movies for
free!!!

02
Common file
formats

04
Network
traffic

06
What real digital
forensics looks like



What is Digital Forensics?

Everything you do online leaves a footprint

- Digital Forensics involves tracking that footprint

You can hide all sorts of information in all sorts of files.
In this section we're going to look at this aspect of forensics called Steganography

- images, pdfs, audio files, and so much more



Common File formats

```
/ctfs/mm  
$ file ben10.txt  
ben10.txt: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, Exif  
Standard: [TIFF image data, big-endian, direntries=4, xresolution=62, yresolution=70], progressive,  
precision 8, 338x601, components 3
```

in steganography you're often working with images, audio files or archives.

- But beware, you can't always trust the file extension!
- This is when you use the 'file' command!

How does 'file' work???



Image Steganography

Metadata Analysis

Every file in your filesystem has some metadata associated with it. And certain file formats allow you to modify this metadata with user defined values

```
> exiftool meow.jpg
File Type          : JPEG
File Type Extension : jpg
MIME Type         : image/jpeg
...
User Comment      : a29zc0NURntsb29rX2Zvcl90aGVfZGF0YV91bmRlc190aGVfZGF0YX0=
```

Strings

```
> strings hackme
libc.so.6
exit
_isoc99_sscanf
...
Enter the password:
kossCTF{remeber_to_strip_your_binaries_kids}
...
```



Image Steganography

File carving

Sometimes you can add a file INSIDE another file.

```
> binwalk -aye torrent.pcap  
for/torrent/extractions/torrent.pcap
```

DECIMAL	HEXADECIMAL	DESCRIPTION
8861	0x229D	ZIP archive, file count: 2, total size: 192209460 bytes
[+] Extraction of zip data at offset 0x229D completed successfully		

```
file1 start  
...  
...  
...  
file1 end (supposedly)  
—  
file2 start  
...  
...  
...  
file2 end (file actually ends now)
```



Computer Networks

Network Forensics

As you saw in the web module, all communication happens through "requests". These requests are sent through "packets" and given the scale of the internet, you can imagine there must be MANY packets being sent every second.

- Network forensics deals with understanding these packets

Packets and Protocols

This is a very overly simplified explanation, but just remember that these packets follow certain protocols that define exactly the contents of the packet and any other required details. You saw this a little in heartbleed. Knowledge about various protocols and their common vulnerabilities becomes essential after a point.



Wireshark

Computer Networks

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
199	0.119097	192.168.23.221	10.0.0.1	BitTor...	71	Request, Piece (Idx:0xa,Begin:0x9fa7e,Len:0xffffd3)
200	0.119097	10.0.0.1	192.168.76.41	BitTor...	65462	Continuation data
201	0.119097	10.0.0.1	192.168.122.78	BitTor...	65462	Continuation data
202	0.119097	10.0.0.1	192.168.102.46	BitTor...	65462	Continuation data
203	0.124240	10.0.0.1	192.168.102.46	TCP	65462	[TCP Retransmission] 37630 → 6881 [SYN] Seq=0 Win=8192 Len=65408
204	0.119097	10.0.0.1	192.168.64.200	BitTor...	65462	Continuation data
205	0.122642	10.0.0.1	192.168.64.200	TCP	65462	[TCP Retransmission] 62683 → 6881 [SYN] Seq=0 Win=8192 Len=65408
206	0.119097	10.0.0.1	192.168.118.189	BitTor...	65462	Continuation data
207	0.119097	192.168.88.196	10.0.0.1	BitTor...	65462	Continuation data
208	0.119097	192.168.158.190	10.0.0.1	BitTor...	65462	Continuation data
209	0.119097	192.168.158.190	10.0.0.1	BitTor...	65462	Continuation data
210	0.119097	192.168.118.189	10.0.0.1	BitTor...	71	Request, Piece (Idx:0xb,Begin:0xaf9f1,Len:0xffffd3)
211	0.119097	192.168.102.46	10.0.0.1	BitTor...	71	Request, Piece (Idx:0xb,Begin:0xaf9f1,Len:0xffffd3)
212	0.119097	10.0.0.1	192.168.76.41	BitTor...	65462	Continuation data
213	0.119097	10.0.0.1	192.168.204.136	BitTor...	65462	Continuation data
214	0.126993	10.0.0.1	192.168.204.136	TCP	65462	[TCP Retransmission] 2661 → 6881 [SYN] Seq=0 Win=8192 Len=65408
215	0.134737	10.0.0.1	192.168.242.7	BitTor...	65462	Continuation data
216	0.143162	10.0.0.1	192.168.242.7	TCP	65462	[TCP Retransmission] 43162 → 6881 [SYN] Seq=0 Win=8192 Len=65408
217	0.134737	10.0.0.1	192.168.242.7	BitTor...	65462	Continuation data
218	0.134737	10.0.0.1	192.168.41.253	BitTor...	65462	Continuation data
219	0.143030	10.0.0.1	192.168.41.253	TCP	65462	[TCP Retransmission] 54165 → 6881 [SYN] Seq=0 Win=8192 Len=65408
220	0.134737	10.0.0.1	192.168.184.230	BitTor...	65462	Continuation data
221	0.136406	10.0.0.1	192.168.184.230	TCP	65462	[TCP Retransmission] 62335 → 6881 [SYN] Seq=0 Win=8192 Len=65408

Frame 209: 65462 bytes on wire (523696 bits), 65462 bytes captured (523696 bits)
Ethernet II, Src: WistronNeweb_0c:4a:20 (40:1a:58:0c:4a:20), Dst: ServercomPri_56:6b:9e (f0:ed:b8:56:6b:9e)
Internet Protocol Version 4, Src: 192.168.158.190, Dst: 10.0.0.1
Transmission Control Protocol, Src Port: 58609, Dst Port: 6881, Seq: 0, Len: 65408
BitTorrent
[Community ID: 1:AnpY4KB7qW1MqvjZp61rhtzWa0E=]
TRANSUM RTE Data

0000 f0 ed b8 56 6b 9e 40 1a 58 0c 4a 20 08 00 45 00
0010 ff a8 00 01 00 00 40 06 11 e7 c0 a8 9e be 0a 00
0020 00 01 e4 f1 1a e1 00 00 00 00 00 00 00 50 02
0030 20 00 44 45 00 00 00 00 ff 7c 07 00 00 00 0a 00 .DE.
0040 09 fa 7e 90 48 df 8d 01 a5 26 5c 3e ae 40 31 5e .~. H . . &> @1^ .
0050 9a d1 29 9d 90 5a 3d 0b 73 16 cf a3 e4 f5 25 .) . Z= . s . N % .
0060 03 ab 66 6a b5 6e 17 83 dc 6c 0a e8 6e 32 9d 84 .fj. n . l . n2 . .
0070 0d 09 36 98 78 84 00 f5 d1 c2 87 f6 e7 e0 64 .6. x d .
0080 a5 e9 26 9a 92 07 20 c3 8e 90 21 aa 32 ec 1d ec .&. . . . ! 2 . .
0090 fe 58 51 3f 7b 67 f8 15 e2 88 f0 b4 19 17 3e 87 .XQ?{g > .
00a0 27 95 2a 35 2d 54 78 17 c8 3c 23 b9 66 6d fe d0 '.*5-Tx . <# fm . .
00b0 4e f4 aa 12 89 05 82 d0 c4 71 92 fa 28 e5 97 5a N q .(. Z .
00c0 19 21 7c 93 0e e4 cc c3 c4 5b 0f 4f 3a 5b c4 06 !| . . . [0:[.
00d0 d6 35 9b 08 0f 35 f6 ce ed f0 79 25 c1 ba 1c 04 .5 . . 5 . . y% . .
00e0 90 22 e6 d0 5e 95 c2 5c 1a 57 3a be 9c bd b4 67 .". A . \ . W . . g .
00f0 b6 9d 2c 4d 85 b5 db 11 1e f4 ad 11 18 a0 39 0e . , M 9 . .
0100 d9 df 78 e0 00 6a 17 18 2f 9c 88 0f 04 5c 72 a3 .x . j . / . . . \ r . .
0110 9c 7c 2c ea cb f9 d3 44 41 12 04 79 8c 39 35 83 |, . . . D A . y . 95 .
0120 97 0d 36 42 b9 c5 a0 d1 d5 95 a9 37 d9 f7 ac 5a .6B 7 . . Z . .
0130 f5 d4 81 47 e5 c9 19 7d b9 0a 0a 31 fc 49 e6 e6 .G . . } . 1 . I . .
0140 68 f0 5a 5e 0e 93 7c 50 19 de e3 a7 c1 29 a1 e4 h . Z ^ . | P . .) . .
0150 15 cb 85 11 79 53 8b 06 89 46 ea 8f cb ce 40 65 .y S . . F . . @e . .
0160 2f a8 fa 93 9e 86 94 15 8b 48 34 72 17 2f 8d c2 ./. . . . H4r / . .
0170 09 01 a5 ba dc 7f e1 3b a3 ef 6a 03 c9 f5 2e be .; . j
0180 30 3d bc e7 1c bd 82 6b c9 db 11 e3 70 23 fd 8a .0= . . k . . p# . .

Packets: 3414 Profile: Default

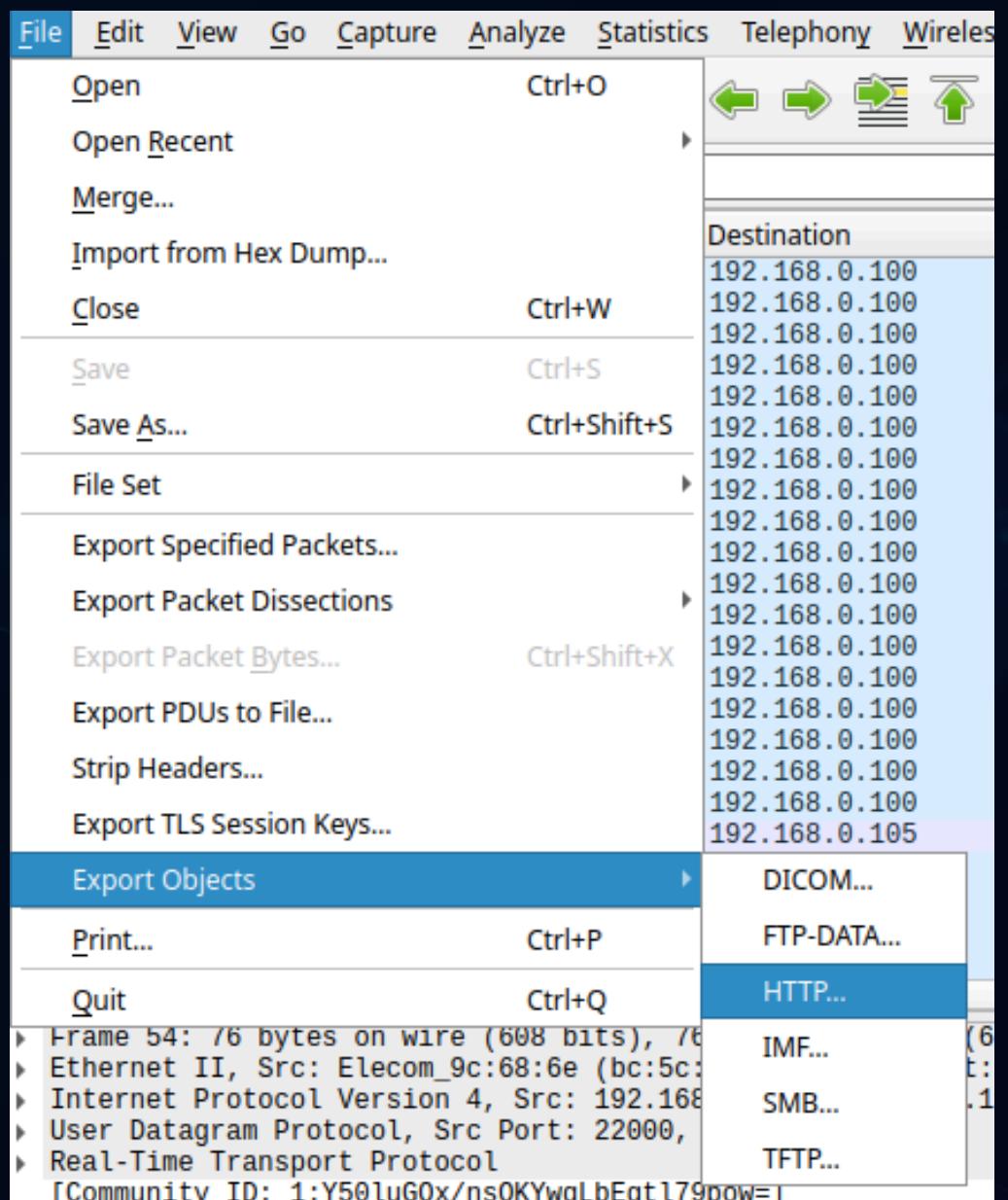


Computer Networks

Free Netflix Movies?

What happens when you watch some content?

- The video is being sent to your machine via packets
- If you capture these packets (with wireshark) could you extract this video?
 - Maybe not for Netflix movies (there's something called DRM protected content)
 - But you can extend this idea to get other cool stuff
- Live streams must be sent via unencrypted packets (RTP protocol), you can extract them from pcaps.
- Similarly files shared via "weaker" protocols like HTTP can be extracted.
- Other examples I've seen:
 - Reconstructing a minecraft world from the server packets
 - Recreating a file from the torrent packets





What does Real DFIR Look like?

Like we mentioned in the beginning of this section, everything leaves a footprint.

What are some examples of recovering footprints?

- recovering deleted files?
- breaking password protection?
- or slogging through log files large enough to crash your text editor?

All of the above! And a lot more. DFIR is more defense oriented. Going through very large .pcaps or logs to find suspicious activity or analysing file systems and recovering data etc is what these guys do.

-rw-r--r-- 1 [REDACTED] [REDACTED] 14M Jan 5 09:33 dns.txt
-rw-r--r-- 1 [REDACTED] [REDACTED] 217M Jan 4 05:41 logs.json
-rw-r--r-- 1 [REDACTED] [REDACTED] 7.5K Jan 5 09:33 meow.txt
-rw-r--r-- 1 [REDACTED] [REDACTED] 263M Jan 4 08:35 new_logs.json

opening
these
ate half
of my
ram



Open Source Intelligence (OSINT)

OSINT is about gathering and analyzing information that's out there for everyone to see. Imagine you're a detective. But instead of dusting for fingerprints, you're combing through social media posts, public records, and online forums. That's OSINT in action!

Let's dive into some basic OSINT challenges!





How to get better at CTFs?

use linux

learn a scripting language

read writeups

write writeups

ctftime.org



References and Resources

GEN:

- [Careers in cybersecurity](#)
- [Beginner's guide](#)

Web:

- [What happens at "google.com"](#)
- [Browser Tools](#)
- [Heartbleed XKCD](#)
- [Heartbleed](#)

CRYPTO:

- [Gravity falls \(available on KGP's DC++\)](#)

FORENSICS:

- [Image Steganography](#)

OSINT:

- [Beginner's guide to OSINT](#)



—

Official Solutions

<https://github.com/kossiitkgp/KossCTF-2025>

Feel free to add your writeups and solutions if you think you did something unique or cool

Feedback Form

We value your feedback and would appreciate you filling out this form



Feedback





—

THANK YOU

R E A C H O U T T O U S :

