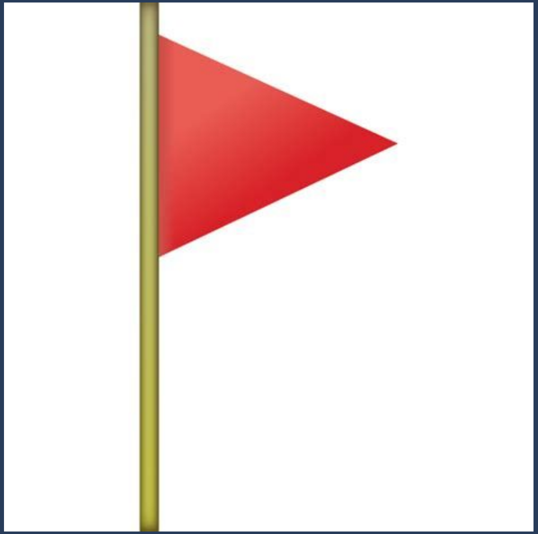
Introduction

What is CTF?

- Team-based hacking competition
- Solve problems, get flags:
  - picoCTF{i_4m_fl4g}
- Categories:
  - Web
  - Crypto
  - Reverse engineering
  - PWN / Binary exploitation
  - Forensics
  - Misc



RoyalRoppers

- CTF team
- Förening / Club / Association
- Stuff we do:
  - Play CTFs
  - Have meetups
  - Summer barbecue
  - Company events
  - Have fun
  - Make friends
  - Learn

Meetups

- Thursdays 18:00-20:00
- Usually a lecture first hour
- Solve problems second hour
    - picoctf.org
- Get “homework” problems
- You can present as well!

Omegapoint event

omega point.

28 of September or 2 of November

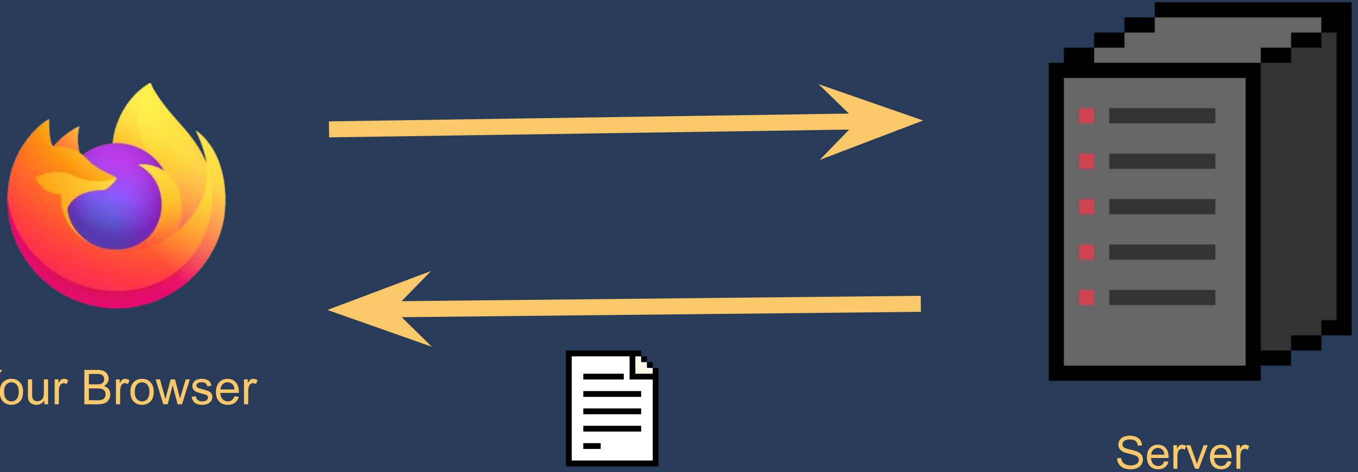
Web security

Slides by:
Asta Olofsson
Axel Nilsson

Table of contents

- Overview of the web
  - Client server
  - URLs
  - HTTP
  - HTML
- Vulnerability overview
- Tools
- Depending on time
  - Burp Demo
  - SQLi
  - XSS

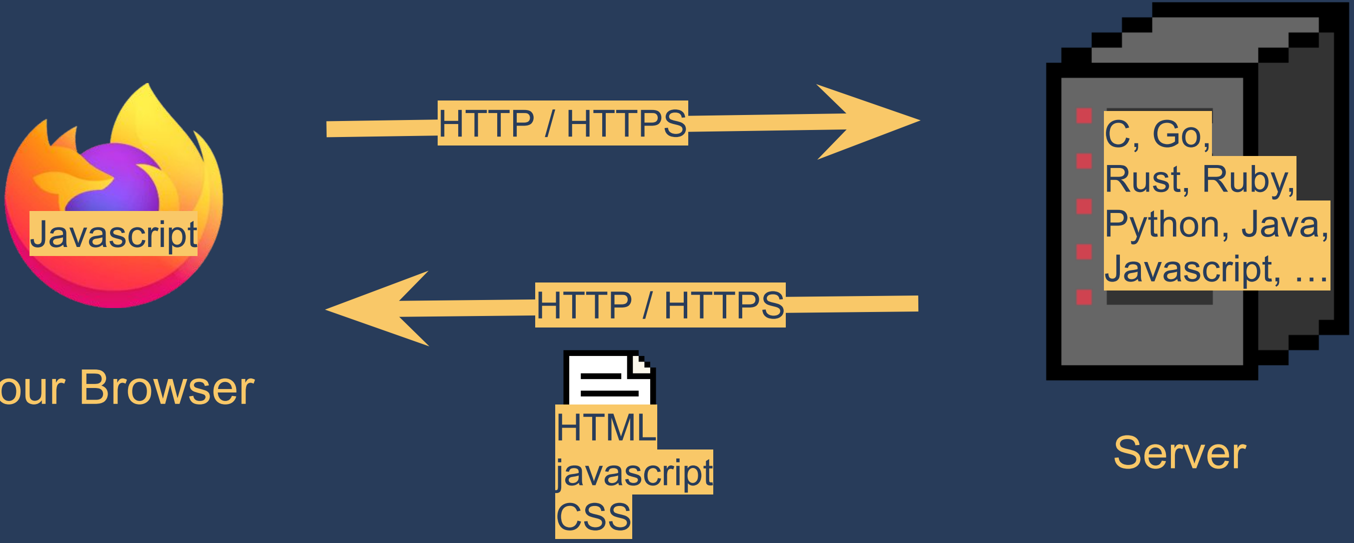
Overview



Your Browser

Server

Overview



Javascript

Your Browser

HTTP / HTTPS

HTTP / HTTPS

HTML
javascript
CSS

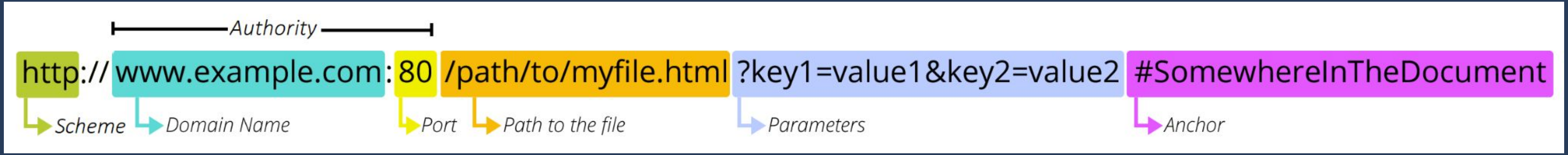
C, Go,
Rust, Ruby,
Python, Java,
Javascript, …

Server

URL

- Protocol
- Domain name / IP
- Port
- Path to file
- Parameters
- Anchor

When someone tries to “Rick Roll” you but you’ve memorized the URL:
You can't trick me anymore.
ifunny.co



Authority

http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument

Scheme

Domain Name

Port

Path to the file

Parameters

Anchor

HTTP (HyperText Transfer Protocol)

- Request-response protocol
  - Send request
  - Get response

Requests

Request

```
GET /hello.html?q=something HTTP/1.1
Host: example.com
Content-Length: 5
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-us
Accept-Encoding: gzip, deflate

hello
```

Request Line

- Method - GET
- Request URI - /hello.html?q=something
- HTTP version - HTTP/1.1

HTTP Methods

- GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH…
- GET and POST most used
- GET
  - Used to request data
  - Content can be cached by browser and proxy
  - Should not be used for sensitive data
- POST
  - Content is not cached
  - Used for handling sensitive data
  - Your password is sent using POST request

Requests

Request

```
GET /hello.html?q=something HTTP/1.1
Host: example.com
Content-Length: 5
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-us
Accept-Encoding: gzip, deflate

hello
```

Headers

- Key-Value pairs

Requests

Request

```
GET /hello.html?q=something HTTP/1.1
Host: example.com
Content-Length: 5
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-us
Accept-Encoding: gzip, deflate

hello
```

Body

- Extra data
- Example: POST requests can contain password here

Responses

Response

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Status Line

- HTTP version + status code
- Contains a status code
  - 1xx - Information
  - 2xx - OK (success)
  - 3xx - Redirection
  - 4xx - Client side error
  - 5xx - Server side error

Responses

Response

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello!</h1>
</body>
</html>
```

Headers

- Same as for requests
- Responses and Requests share some headers
- There are request-unique headers and response-unique headers

Problems

GET aHEAD

Responses

Response

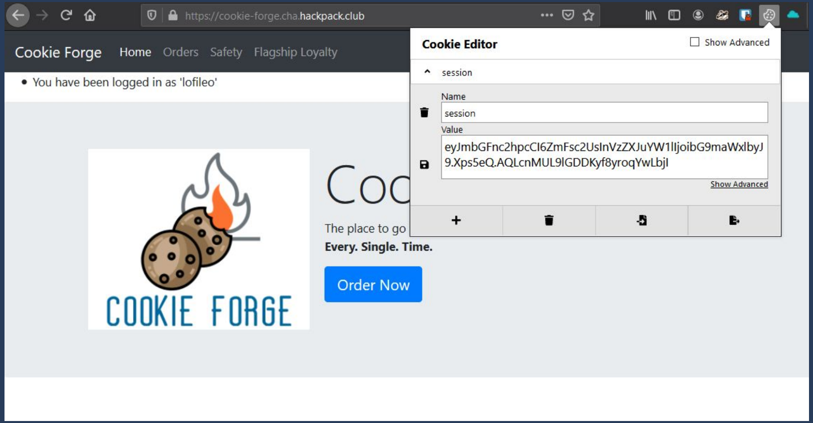
```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello!</h1>
</body>
</html>
```

Body

- Same as for requests - contains data
- In this case, HTML data for browser to display!

HTTP Cookies

- Stores information about a user
- Authentication cookies
- Tracking cookies
- Cookie Headers
  - Response: Set-Cookie: session=secret
  - Request: Cookie: session=secret



https://cookie-forge.cha.hackpack.club

Cookie Forge   Home   Orders   Safety   Flagship Loyalty

- You have been logged in as 'lofileo'

COOKIE FORGE

Coo

The place to go

Every. Single. Time.

Order Now

Cookie Editor   □ Show Advanced

session

Name: session

Value: eyJmbGFnc2hpcCI6ZmFsc2UsInVzZXJuYW1lIjoibG9maWxlbyJ9.Xps5eQ.AQLcnMUL9lGDDKyf8yroqYwLbjI

Show Advanced

Problems

Cookies

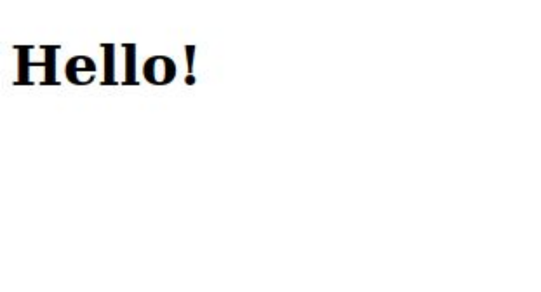
Logon

Power Cookie

HTML (HyperText Markup Language)

```
<html>
        <body>
                <h1>Hello!</h1>
        </body>
</html>
```

HTML Code



Hello!

Browser

- Consists of tags
- Tags can have attributes

```
<img src="cat.png" width="200" height="300">
```

- Usually a tag has an opening and a closing tag, but some does not.

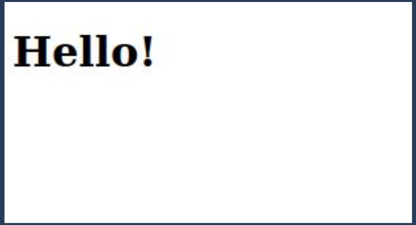
Problems

Insp3ct0r

Scavenger Hunt

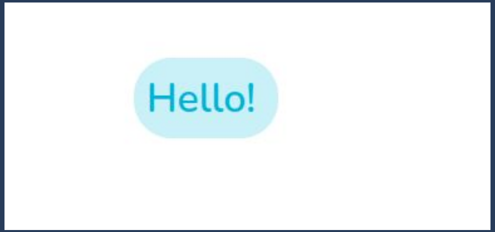
HTML (HyperText Markup Language)

- CSS (Cascading Style Sheets) - makes the HTML beautiful~~~



Hello!

Without CSS



Hello!

With CSS

Vulnerabilities - Injections

- General type of vulnerability
- Can appear in many different contexts

Vulnerabilities - Injections

- When code and data are mixed, injections can happen
- Command injection example:

```
os.system(“convert “ + path + ” profile.png”)
```

- Data can be treated as code:

```
os.system(“convert --help; evil-cmd; # profile.png”)
```

- Separate code and data
- Be very strict about the data

```
if not safePath(path):
    throw Exception(“bad”)
os.system(“convert %s profile.png” % path)
```

Vulnerabilities

- There are many types of vulnerabilities
- Too many to go through them all in detail
- Links so you can read more on your own

Vulnerabilities

- Client Side
    - XSS - Cross Site Scripting
    - CSRF - Cross Site Request Forgery
    - …
- Server side
    - SQL injection
    - Command Injection
    - Deserialization
    - Path Traversal
    - SSRF - Server Side Request Forgery
    - SSTI - Server Side Template Injection
    - XXE - XML External Entity
    - …
- Conclusion: user input is dangerous!

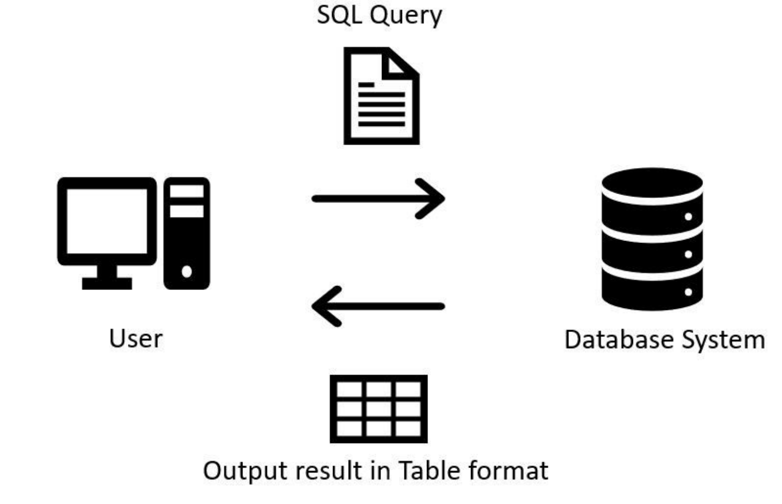
Tooling

- Burpsuite
- Python `requests`
- Curl
- Developer console (CTRL + SHIFT + i or F12)
- Ngrok
- Sqlmap

Burp Demo

SQL

- SQL - language used to access and manipulate databases
- DML - Data Manipulation statements
  - SELECT, INSERT, UPDATE, DELETE
  - Ex: SELECT * FROM MyTable
- DDL - Data Definition statements
  - CREATE, ALTER, DROP, TRUNCATE
  - Ex: CREATE TABLE Persons (ID int, Name varchar(255));



Problems

SQL Direct

SQL Injections (SQLi)

- Execute SQL scripts on the server that steals information from the server
- Examples
  - Authentication forms
  - Search engines
  - E-Commerce sites
  - Blog
  - Anything with SQL database

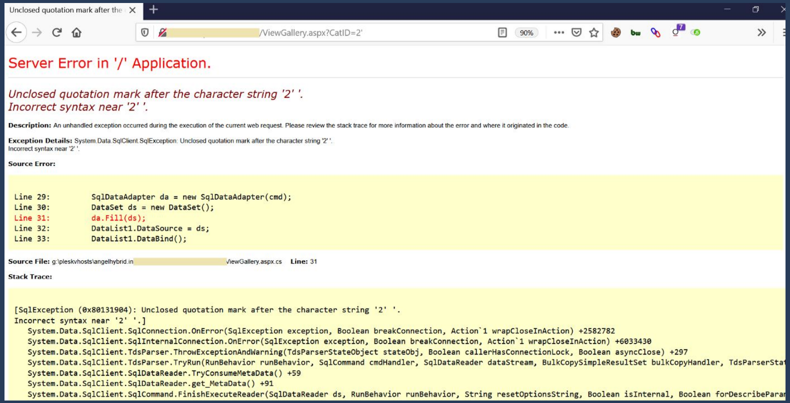
```
sqlQuery(“SELECT info FROM products WHERE name=’”+input+”’”)
```

Approach

- Detect
- Fingerprint
- Enumerate
- Exploit

Error based SQLi

- Utilizes the error output
- Verification and exploitation
  - Break out of SQL query statements using for example single-quotes, double-quotes, backticks or semi-colons in the input field
  - Look for any error-messages or misbehavior in the application



[redacted]/ViewGallery.aspx?CatID=2'

Server Error in '/' Application.

Unclosed quotation mark after the character string '2' '.
Incorrect syntax near '2' '.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string '2' '.
Incorrect syntax near '2' '.

Source Error:

```
Line 29:         SqlDataAdapter da = new SqlDataAdapter(cmd);
Line 30:         DataSet ds = new DataSet();
Line 31:         da.Fill(ds);
Line 32:         DataList1.DataSource = ds;
Line 33:         DataList1.DataBind();
```

Source File: g:\pleskvhosts\angelhybrid.in[redacted]ViewGallery.aspx.cs **Line:** 31

Stack Trace:

```
[SqlException (0x80131904): Unclosed quotation mark after the character string '2' '.
Incorrect syntax near '2' '.]
   System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +2582782
   System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +6033430
   System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose) +297
   System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserSta
   System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() +59
   System.Data.SqlClient.SqlDataReader.get_MetaData() +91
   System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternal, Boolean forDescribePara
```

How to detect SQLi

- Different injection points

```
SELECT info FROM products WHERE name=’text’
```

```
SELECT info FROM products WHERE name=”text”
```

- Detect with `‘te’||’xt’`

```
SELECT info FROM products WHERE id=number
```

- Detect with `2+2`

Error based SQLi

- Input field is a **String** and the query is **SELECT * FROM Table WHERE id = ‘1’**
  - **If ‘ gives false then ‘ ‘ must give true**
  - **If “ gives false then “ “ must give true**
  - **If \ gives false then \\ must give true**
- Input field is a **Numeric** and the query is **SELECT * FROM Table WHERE id = 1;**
  - AND 1 = True, AND 0 = False
  - AND true = True, AND false = False
  - 1*50 returns 50 if vulnerable and 1 if not vulnerable

Error base SQLi

- Input field is **LOGIN** and query is **SELECT * FROM Table WHERE username = ‘’;**
  - **‘ OR ‘1**
  - **‘ OR 1 --**
  - **“ OR “” = “**
  - **‘LIKE’**
  - **‘=’**

Union based SQLi

```
SELECT title, text FROM news WHERE id=0
```

Breaking News
Something something

```
SELECT title, text FROM news WHERE id=0 UNION SELECT ‘a’, ‘b’
```

Breaking News
Something something

a
b

Union based SQLi

```
SELECT title, text FROM news WHERE id=0
UNION SELECT username, password FROM users
```

```
Breaking News
Something something

alice
password123

bob
coolcool
```

Very powerful type of SQLi!

Union based SQLi

- Extract information from the database by extending the results returned by the original query
- We can use the **Union** operator for this
- SELECT * FROM profiles WHERE id=$id UNION SELECT 1,2,3 -- LIMIT 0,1
- SELECT header, txt FROM news UNION ALL SELECT name, pass FROM members

Blind SQLi

- No error message is displayed by the application but its behaviour changes
- Boolean-based
  - Result varies depending on whether the sql-query is true or false
  - Ex. error message when login in disappear when we inject our payload
- Time-based
  - Sql-query that makes the database wait before returning the result. The difference in time can be used to leak data.

Blind SQLi - Boolean based

- Example: Checking available usernames

```
SELECT name FROM users WHERE name=’alice’
```

SQLi problems

- SQLiLite - PicoCTF
- Irish-Name-Repo 1 - PicoCTF
- Irish-Name-Repo 2 - PicoCTF
- Irish-Name-Repo 3 - PicoCTF
- https://sqlilabs.carelessfinch.me/ - bunch of SQLi challs
- https://www.wechall.net/challs/MySQL/by/chall_score/ASC/page-1

Cross Site Scripting (XSS)

- XSS is when a user of an application can send javascript that is executed by the browser of another user of the same application
- Javascript can
  - Modify the page (DOM)
  - Send HTTP requests
  - Access cookies
- This can be combined to form an xss attack. Ex. extract cookies, send to server of attacker

Reflected XSS

- XSS through URL parameters
- https://example.com?data=<script>alert(1)</script>
  - <html>
  - <body>
  - <script>alert(1)</script>
  - </body>
  - </html>

Stored XSS

- The XSS payload is provided from the website itself
- Ex. posting a XSS payload as a comment or blog post that is displayed to other users

DOM XSS

- The browser itself is injecting an XSS payload into the DOM
- The server might not be injectable but the client side javascript files are causing the issue
- Ex. dynamic javascript code such as eval() or innerHTML
- Exploit in URL which is accessed with the window.location object

http://www.example.com/userdashboard.html#context=<script>SomeFunction(somevariable)</script>.

XSS Problems

- https://xss.challenge.training.hacq.me/
- https://xss-game.appspot.com/
- https://xss.pwnfunction.com/

Upcoming CTFs

- DownunderCTF - 23 sept 2022, 11:30– 25 sept 2022, 11:30

Upcoming meetups

- Week 37 tuesday: SSRF deep dive
- Week 38 thursday: Reverse engineering

What do you know about reversing?
What do you want to know?

- Have you written C code?
- Do you know some assembly language?
- Calling conventions?
- Virtual memory?
- Have you used Ghidra / IDA?
- Have you used gdb?
- Static vs. dynamic linking?
- Stripped vs. non-stripped binaries?

Link to these slides:

royalroppers.team/meetups/web_2022-09-08.pdf