A Guide to CTFs for the Uninitiated

Portland State University's Security Club
void* vikings

ctf-team@pdx.edu
Twitter: @VoidStarVikings

Allison Evan Robin Tristan

Who are the void * vikings?

Our mission is to promote security culture, ethics, research, ongoing education, and development of safer code through playing in Capture the Flag competitions.

- Portland State University student club
- Formed in February 2020
- 3 meetings a week + CTF events
- Majority Women in Leadership
  - Team ~ 50/50

Feb 2020 damctf, OSU ...

What have we been up to?

2020/2021 School Year

- BsidesPDX 2020
  - $25 e-gift card No Starch Press
- UTCTF 2021
- WeCTF 2021
  - $200 Digitalocean credit
  - (Top 10)
- picoCTF 2021
- DarkCON 2021
- DiceCTF 2021
- WPICTF 2021
- DawgCTF 2021
- JustCTF 2021
- Mentored by Samurai
  - PlaidCTF 2021
  - DEFCON 29 quals

Community Engagement

- Collaboration with students from UO, NYU, PCC, and high school students
- Participation in a panel discussion for high school students at NW CyberCamp in 2020 and 2021
- Presentation at BSides PDX in October 2020
- Exchanging mentorship with Samurai members

Who you are

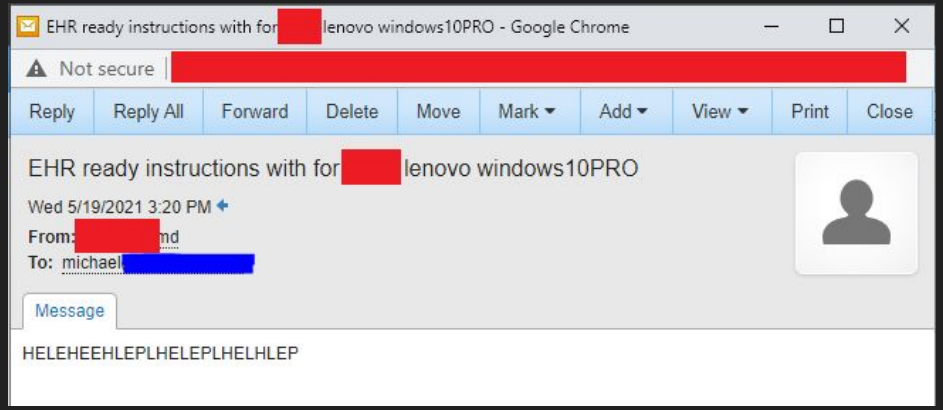
- You might not be sure what a CTF is...
- You might not played in a CTF before...
- You might be new to infosec/cybersecurity...
- You might be new to tech at large…
- You might not be the conventional candidate for CTFs…
  - You might have tried one and had a bad time...

OR

- You might be trying to understand and mentor someone who is.
  - And it might not be going so well...

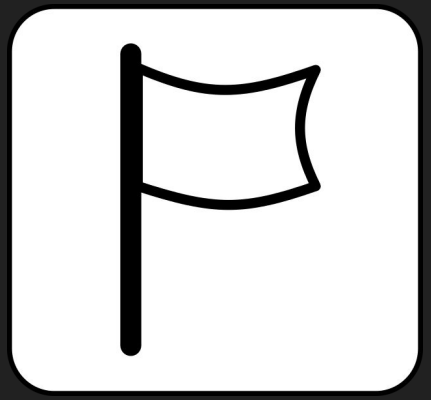
Security Games: Why Capture the Flag?

- Very Important skill for all developers!
- Education
- Career exploration
  - https://www.cyberseek.org/
- Networking and recruiting
- Experience / Resume building
  - Can count as ‘work experience’
  - Good skills for developers
  - Bug Identification
- Fun & Prizes



What is a Capture the Flag (CTF)?

- A security game or series of games that require a range of skills and knowledge to solve a puzzle or find a piece of hidden information.
- Often you are looking for a specific string or phrase (the “flag”) to prove that you have completed the challenge.
- Began in the 1990s in industry



CTF Game Types

- Timed Games: 1 hour - 4 weeks
  - 48 hours is common
- Jeopardy
  - (the most common type)
  - Series of individual challenges across a variety of categories
  - Qualifiers tend to be this form
- Attack/Defense
  - Identical servers for each team to patch and exploit
  - Finals are often A/D
- Mission Based
  - ‘Scavenger hunt’, ‘hackquest’, ‘role play’
  - Vary wildly
  - Usually working through a storyline instead of explicit stages
- Evergreen CTFs / “Wargames”
  - No end time, play through on your schedule

Challenge Categories

- **Web**
- **Forensics**
- Open Source Intelligence (OSINT)
- Social Engineering (SE)
- Steganography
- Cryptography (Crypto)
- Reverse Engineering (RE)
- Binary Exploitation (pwn)
  - Shellcode, shellcraft
- ML/AI
- Misc/random
- Challenges (challs) may cross multiple categories

Scoring

Scoring

- *Friendly*: points stay the same regardless of how many teams solve it or when
- *Dynamic/Fixed Time Balancing*: score decays, points locked in when you submit flag
- *Full Self Balancing*: as the value of the challenge decreases, your final score decreases
- *Golf*: more points for smallest ‘something’ like size, opcodes, etc
- Final Scoreboard is often hidden in final hours of the game

Example

Game Start: challenge C1 worth 500 pts
Game Play: 5 teams solve C1, you solve C1, then 5 more teams solve C1.

Scoring type	Your submission value	C1 value at end of game	Your final score
Friendly	500	500	500
Dynamic	300	50	300
Full Self Balancing	300	50	50

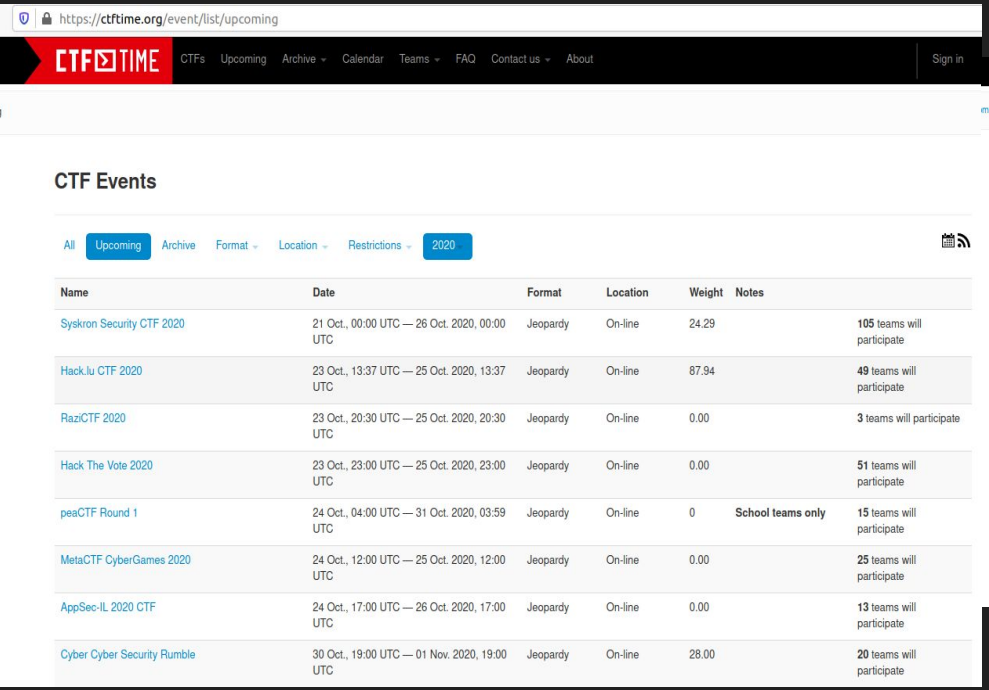
The Adventure Continues After The Game

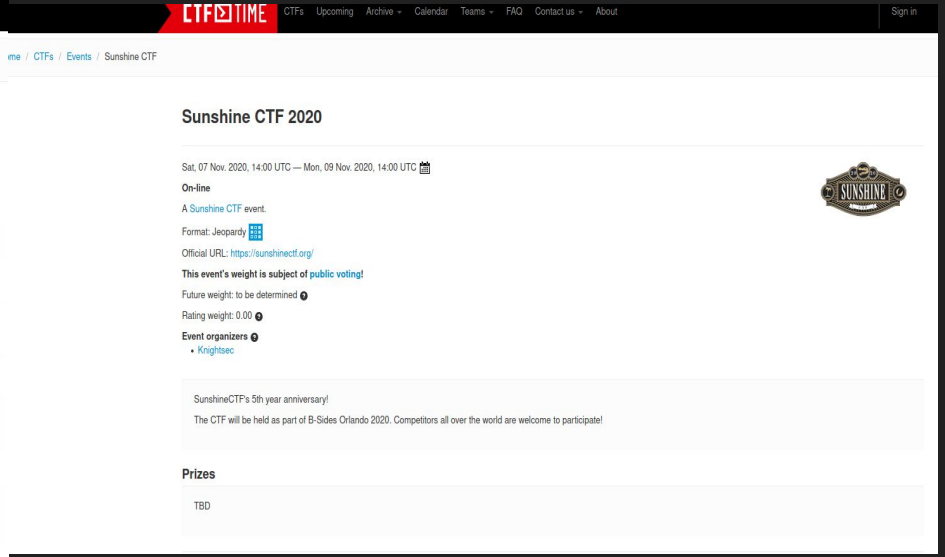
- Try to do your own write up while it's top of mind
  - Your notes may not be as coherent as you think
- Look at write-ups
  - These can vary in usefulness
  - CTFTime.org, blog posts
  - Sometimes it can take a month or more for one to surface
  - https://isopach.dev/Redpwn-CTF-2020/ ← example of a great writeup set
  - If you're lucky, the organizers will release their intended solutions
- Try to write your own challenge
  - Can you recreate it?
  - You are not too new to try your hand at challenge creation

Etiquette (a.k.a stay classy)

- Read the Rules
- Respect the organizers, even if other teams don't
  - Usually [experienced] volunteers. CTF creation can take hundreds of hours of work
  - There will be mistakes and infrastructure problems. Stay Classy!
- Stay in touch with the organizers
  - You'll probably need a Discord account for communication from the organizers.
  - There will be updates! You'll want them minty fresh!
  - New challenges will usually be released intermittently during the game.
  - If the comms are a mess, designate one player to watch it and relay updates
- Respect the infrastructure, even if other teams don't
  - DON'T DDOS unless you're given explicit permission to do so
  - E.g. sqlmap, brute force solvers, etc.
- Don't give into the negativity

Where do I find CTFs?





That’s overwhelming?! Where do I even start?

1. Know thy Goals
2. Know thy Limits
3. Know thy Needs
4. Know thy Tools

1. Know thy Goals

Skillbuilding? Competition?

If the competitive spirit doesn't motivate you, **you can ignore it**

- Don't check the scoreboard

...but if you thrive on competition, there are teams for you too!

If networking is your goal, behave like it's a professional space

- Be courteous
- Thank the organizers; congratulate other teams
- Try hosting or collaborating with other groups to host events, publish walkthroughs, etc.
  - Future referrals have more weight when you've worked together

2. Know thy Limits

If you want to learn, but a whole CTF is too much time…

Pick one problem and just work on that until you hit a wall

Set a time limit & schedule it out on calendar. i.e., Fri 5pm-dinner: CTF play

It is okay to walk away when it stops being fun!

3. Know thy Needs:

Teams and Community: If prefer not to work alone, an important thing you can do is find your community

- How much time do they expect? How competitive are they?
- Size? Culture?
- If you *are* a lone wolf, there are also lots of CTFs for you

If you can't find a team, make one with some friends!

- You only need 1-2 others
- You don't need to be affiliated with a school or company
- CTFs often have 'looking for team' channels
- Active teams will also recruit

What kind challenges work for you?
Do you like to specialize?

- "I'm a man of many almost-specializations" - *samczsun*

4. Know thy Tools

- **The best tool is the one that works for you!**

Web - Browser developer tools - Python requests library	**Reverse Engineering** - gdb, objdump - Ghidra, Cutter, IDA Pro, Binary Ninja	**Stego** - file - binwalk - Image editor
Pwn - Python pwntools - capstone - unicorn - netcat	**Forensics** - Wireshark - Python - grep - Your favorite text editor	**OSINT** - Web searches - Reverse image search (tineye, Google, etc.) - whois
Crypto - PyCryptodome - CyberChef	**Misc.** - Linux CLI familiarity - Basic scripting skills (python is very common)	**Social Engineering** - Google - Roleplay - OSINT

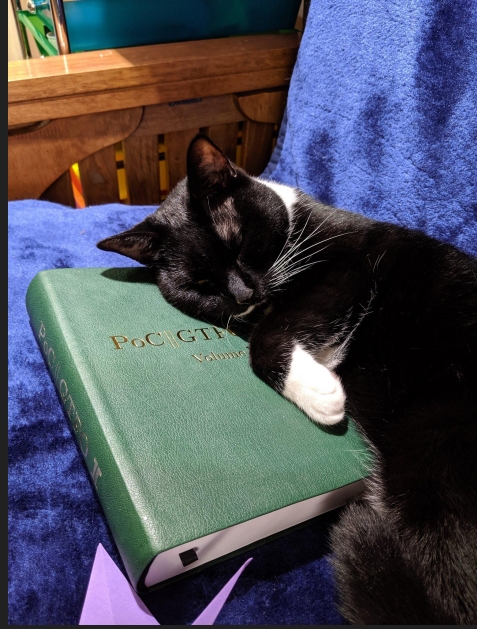
Misc. Pro Tips

- Flags are usually exact string matches, example: `ctf{i_w1sh_I_w4s_4_fl4g}`
  - Watch your case, trailing whitespace, l33tspeak, etc.
- If a file is provided with a challenge, assume you'll need it to solve!
  - The prompt text may or may not be useful
- If a binary file and a **`nc`** invocation is provided in the challenge, you need both.
  - You figure out what to do locally, then repeat it on the challenge server provided to get the flag.
  - **`nc`** == netcat, i.e. `nc chals.damctf.xyz 1337`
- Take notes during the CTF
  - It's easy to get lost down your own rabbit hole
  - Also, if you win a prize during a ctf, the organizers might ask you to provide a write-up as proof that you did in fact solve it

“The only way to get good at CTFs is by playing CTFs”

- *Nopnopgoose*

(it’s a skill, not an innate trait!)



“I wish someone had told me/our group that CTFing would be really difficult for a long time (we were all total beginners) but that you **just keep going**, learn what you can, and the skills eventually start to build on each other in useful ways”

- *Jaime*

Good CTFs for Beginners

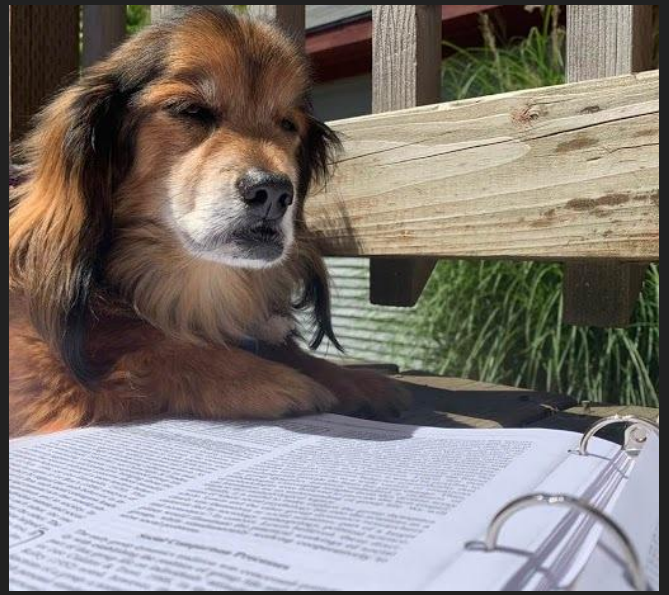
Evergreen CTFs

- Portswigger Web Academy (web)
- Over The Wire (net)
  - Includes Natus
- Try Hack Me (all)
  - Good for students
- PicoCTF (all)
  - Now year round, previous years
  - Good calibration
- Microcorruption (RE)
  - Good calibration
- OWASP Juice Shop (web)
  - Some assembly required
  - Good calibration
- ...and many more!

Regularly Scheduled

- PicoCTF
- NCL (2 seasons: Fall & Spring)
  - Fee required
  - *Note: public write-ups are NOT PERMITTED*
  - However, there is a fair bit of support/infrastructure
- BSides PDX CTF (postponed until 2022)

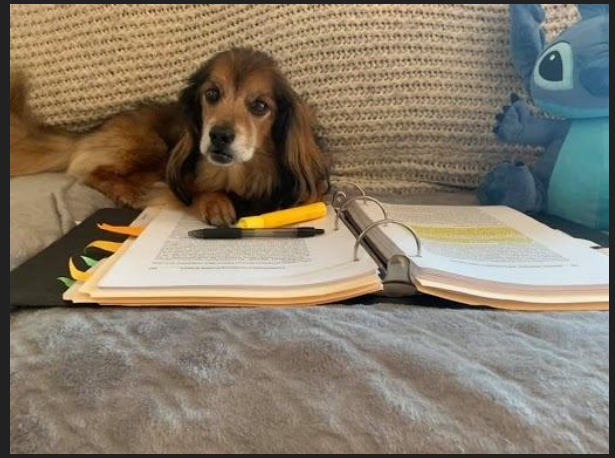
I need a game plan!



1. Pick, form or join a team
2. Pick a CTF (or 3) to start
    a. Read the rules
3. Try a challenge from each category at least once
    a. Take notes: what you liked, disliked
    b. Take notes: what you tried
4. Celebrate
5. Look for writeups
    a. What tool classes did they use?
    b. What techniques did they use?
6. Practice with them!
7. If you feel called to specialize, specialize
8. **Repeat . Consistent practice matters more than anything else**

Suggested Reading

- Some of the many guides for getting started
  - https://jaimelightfoot.com/blog/so-you-want-to-ctf-a-beginners-guide/
  - https://tinyurl.com/CTFfornoobz
  - https://trailofbits.github.io/ctf/intro/
  - https://primer.picoctf.com
- https://clark.center/c/nccp
  - (Especially if NCL is interesting to you)
- https://www.cyberseek.org/
- https://github.com/Naetw/CTF-pwn-tips



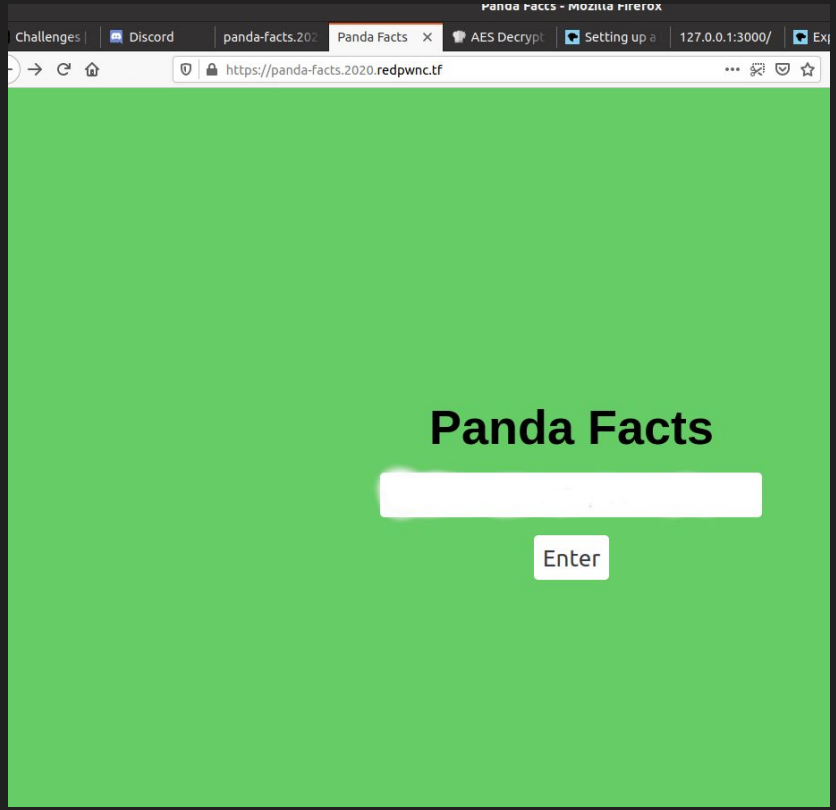
Walkthroughs

- Web
  - Panda Facts I - Allison
  - Natas 26 - Tristan
- Reverse Engineering
  - PicoCTF - Evan

Panda Facts I
(redpwn 2020)

Prompt:
sneak into
this panda hate site

Poking around..



On the other side of login

Welcome, a! Here are some panda facts!

- The binomial name of the giant panda is *Ailuropoda melanoleuca*
- Though giant pandas survive exclusively on bamboo, they process it extremely inefficiently
- The giant panda is a conservation-reliant vulnerable species
- Giant pandas will never be able to survive on their own
- Giant pandas are useless animals that are unimportant in their ecosystem
- Billions of dollars have already been spent on keeping them alive
- Giant pandas are very ungrateful: they generally refuse to breed in captivity
- Money spent on keeping giant pandas alive is money not spent on species that are actually important

Click to see a member-only fact!

Source code supplied

index.js

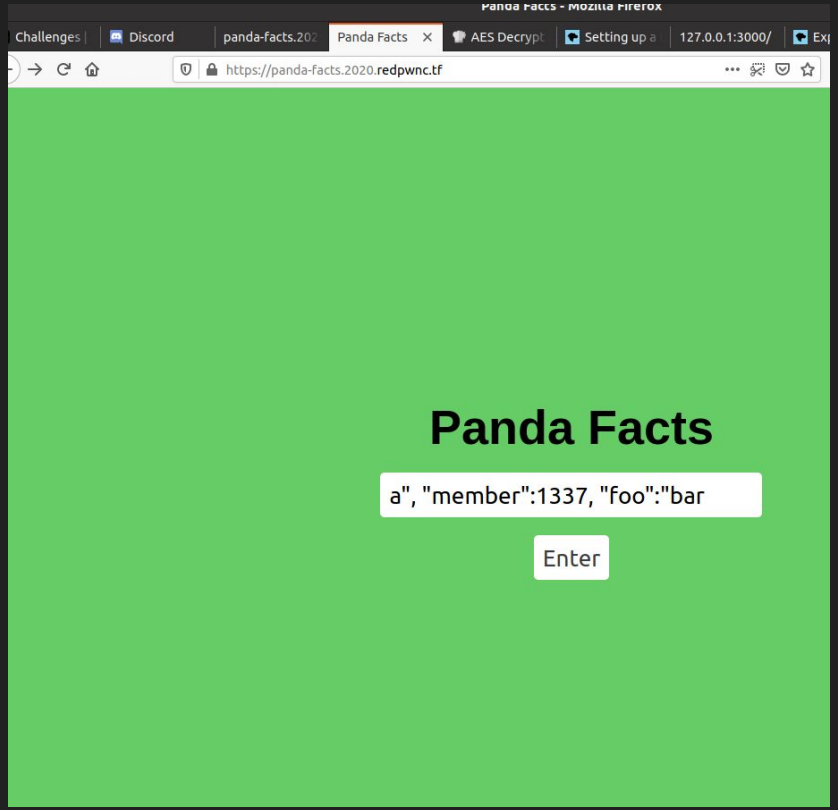
home > amn > Documents > CTFs > redpwn2020 > pandas > index.js > decodeToken

```
110   });
111
112 function generateToken(username) {
113     const algorithm = 'aes-192-cbc';
114     // from their server
115     //const key = Buffer.from(process.env.KEY, 'hex');
116     // hardcoded key can't be good crypto?
117     const key = Buffer.from("12370cc0f387730fb3f273e4d46a94e5", 'hex');
118     console.log("key is "+ key);
119     // Predictable IV doesn't matter here
120     // iv is filled with 0s
121     const iv = Buffer.alloc(16, 0);
122
123     // creates cipher, key probably has salt?
124     const cipher = crypto.createCipheriv(algorithm, key, iv);
125
126     // originally 0
127     const token = `{"integrity":"${INTEGRITY}","member":0,"username":"${username}"}`
128
129     let encrypted = '';
130
131     // switch from utf8 to base64 cipher
132     encrypted += cipher.update(token, 'utf8', 'base64');
133     // any remaining cipher contents? huh?
134     encrypted += cipher.final('base64');
135     return encrypted;
136 }
```

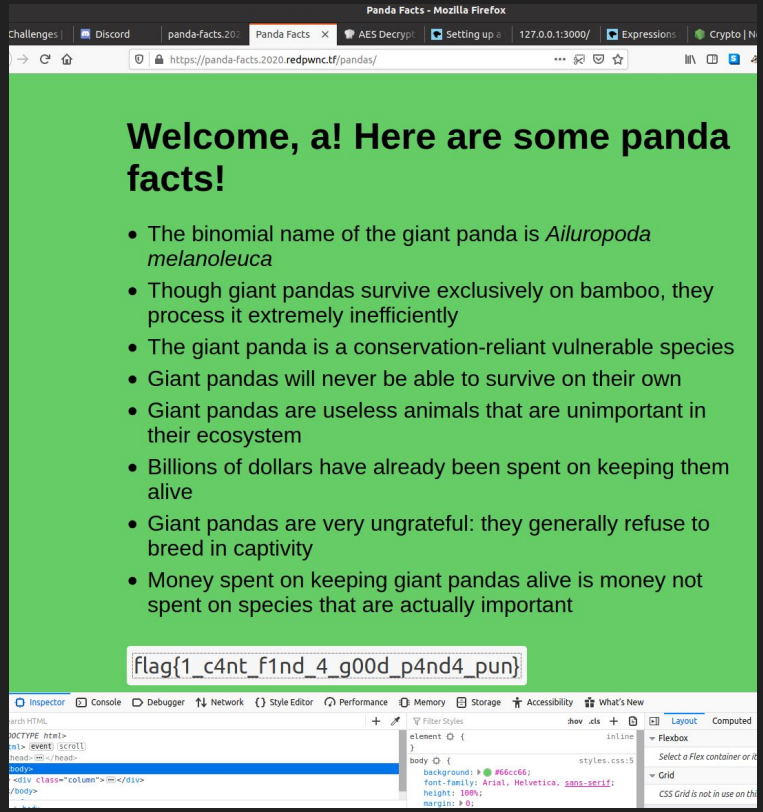
JSON has injection problems

- I did remember that JSON has some structural problems, but not quite what they were…
  - ended up doing some googling
  - It's not a bad thing if you spend a lot of your time during a CTF researching things
- Tl;dr JSON allows duplicate keys, even though it shouldn't
  - Parsers typically just take the last one if there are duplicates
- So, if I can inject my own string, I can override id and replace it with a value of my choice
  - My payload, injected into $username
  - `a", "member":1337, "foo":"bar`
- Injection happens when executable instructions co-mingle with data
  - The attacker injects executable instructions via user controlled data

Injection


Panda Facts - Mozilla Firefox
Challenges
Discord
panda-facts.202
Panda Facts
AES Decrypt
Setting up a
127.0.0.1:3000/
https://panda-facts.2020.redpwnc.tf
Panda Facts
a", "member":1337, "foo":"bar
Enter

Sweet, sweet success



[Back to walkthroughs]

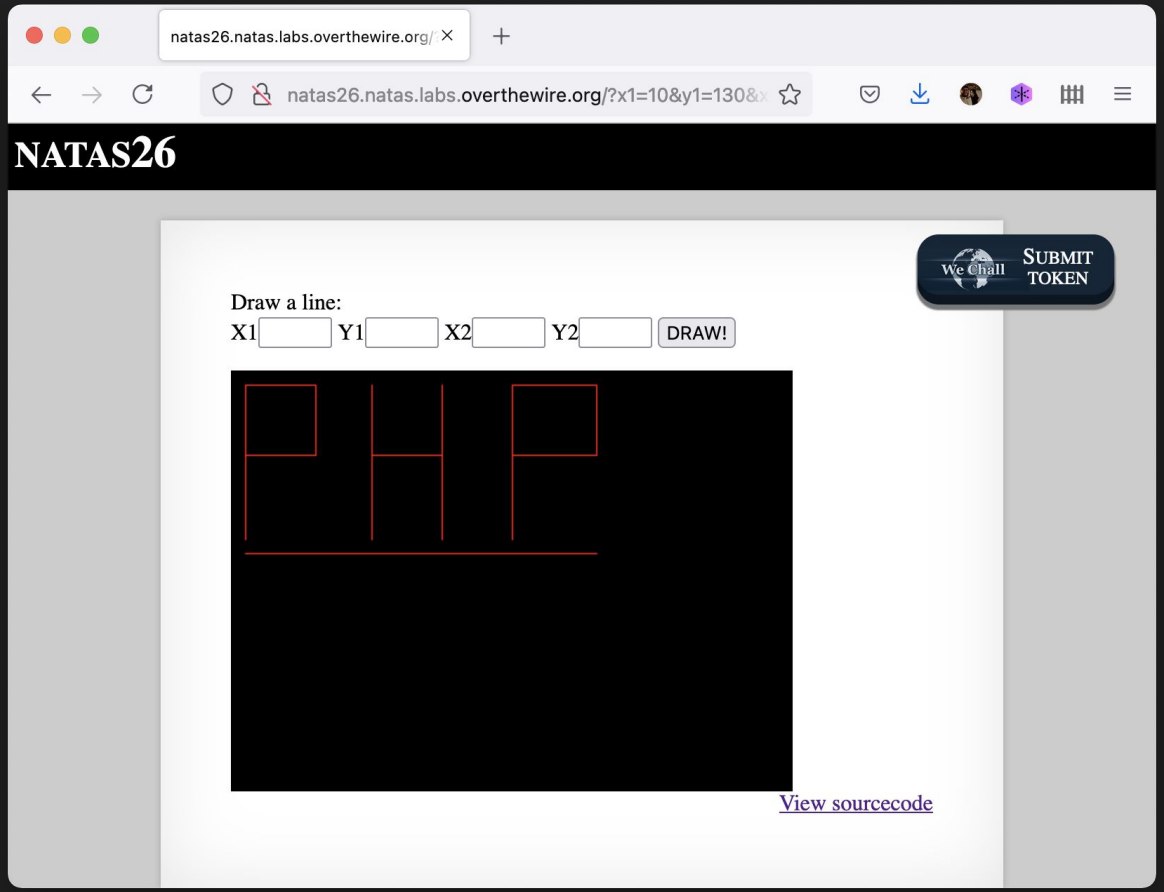
Natas 26

What is Natas

- Evergreen CTF/Wargames on overthewire.org
- Natas teaches the basics of server side web-security.
- 35 Challenges
- We’re going over level 26
- Password for next challenge is stored in /etc/natas_webpass/natas*

https://overthewire.org/wargames/natas/

Getting Started



Checking out the PHP

Via the “View sourcecode” button we are given some server side code.

Program flow:

1. Check for the existence of the “drawing” cookie or newly entered coordinates.
2. Calls drawImage()
3. Calls showImage()
4. Calls storeData();

```php
<?php
    session_start();

    if (array_key_exists("drawing", $_COOKIE) ||
        (   array_key_exists("x1", $_GET) && array_key_exists("y1", $_GET) &&
            array_key_exists("x2", $_GET) && array_key_exists("y2", $_GET))){
        $imgfile="img/natas26_" . session_id() .".png";
        drawImage($imgfile);
        showImage($imgfile);
        storeData();
    }

?>
```

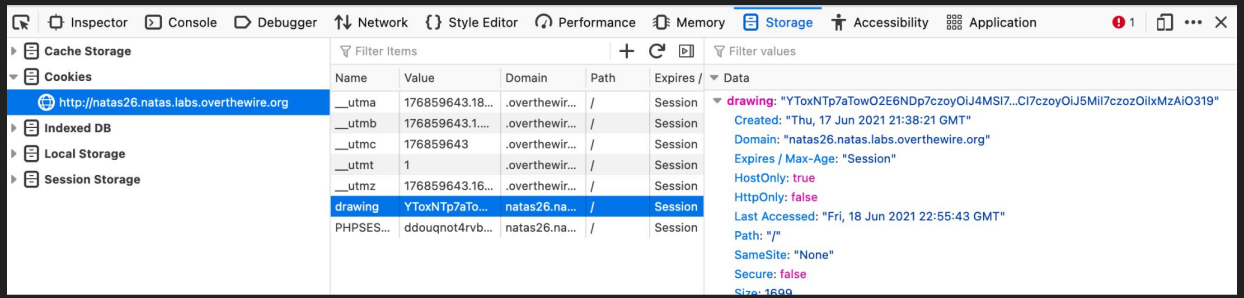
Under the hood

- Creates file using the session ID
- Draws a line from user input and then adds lines stored in the serialized “drawing” cookie.
- Displays the image using the filename specified by the session ID
- Creates a new “drawing” cookie with the newly created lines.

```php
function showImage($filename){
    if(file_exists($filename))
        echo "<img src=\"$filename\">";
}

function drawImage($filename){
    $img=imagecreatetruecolor(400,300);
    drawFromUserdata($img);
    imagepng($img,$filename);
    imagedestroy($img);
}

function drawFromUserdata($img){
    if( array_key_exists("x1", $_GET) && array_key_exists("y1", $_GET) &&
        array_key_exists("x2", $_GET) && array_key_exists("y2", $_GET)){

        $color=imagecolorallocate($img,0xff,0x12,0x1c);
        imageline($img,$_GET["x1"], $_GET["y1"],
                        $_GET["x2"], $_GET["y2"], $color);
    }

    if (array_key_exists("drawing", $_COOKIE)){
        $drawing=unserialize(base64_decode($_COOKIE["drawing"]));
        if($drawing)
            foreach($drawing as $object)
                if( array_key_exists("x1", $object) &&
                    array_key_exists("y1", $object) &&
                    array_key_exists("x2", $object) &&
                    array_key_exists("y2", $object)){

                    $color=imagecolorallocate($img,0xff,0x12,0x1c);
                    imageline($img,$object["x1"],$object["y1"],
                            $object["x2"] ,$object["y2"] ,$color);

                }
    }
}

function storeData(){
    $new_object=array();

    if(array_key_exists("x1", $_GET) && array_key_exists("y1", $_GET) &&
        array_key_exists("x2", $_GET) && array_key_exists("y2", $_GET)){
        $new_object["x1"]=$_GET["x1"];
        $new_object["y1"]=$_GET["y1"];
        $new_object["x2"]=$_GET["x2"];
        $new_object["y2"]=$_GET["y2"];
    }

    if (array_key_exists("drawing", $_COOKIE)){
        $drawing=unserialize(base64_decode($_COOKIE["drawing"]));
    }
    else{
        // create new array
        $drawing=array();
    }

    $drawing[]=$new_object;
    setcookie("drawing",base64_encode(serialize($drawing)));
}
```

The “drawing” cookie

- Stores a serialized version of the coordinates of our lines
- Base64 encoded
- Decoding the Base64 gives us the PHP serialized string
  - a:size:elements
  - i:int
  - s:length:string
  - etc.
- What good does this do us?




```
b'a:15:{i:0;a:4:{s:2:"x1";s:2:"10";s:2:"y1";s:2:"10";s:2:"x2";s:2:"10";s:2:"y2";
s:2:"60";}i:1;a:4:{s:2:"x1";s:2:"10";s:2:"y1";s:2:"10";s:2:"x2";s:2:"60";s:2:"y2
";s:2:"10";}i:2;a:4:{s:2:"x1";s:2:"60";s:2:"y1";s:2:"10";s:2:"x2";s:2:"60";s:2:"
y2";s:2:"60";}i:3;a:4:{s:2:"x1";s:2:"10";s:2:"y1";s:2:"60";s:2:"x2";s:2:"60";s:2
:"y2";s:2:"60";}i:4;a:4:{s:2:"x1";s:2:"10";s:2:"y1";s:2:"60";s:2:"x2";s:2:"10";s
:2:"y2";s:3:"120";}i:5;a:4:{s:2:"x1";s:3:"100";s:2:"y1";s:2:"10";s:2:"x2";s:3:"1
00";s:2:"y2";s:3:"120";}i:6;a:4:{s:2:"x1";s:3:"100";s:2:"y1";s:2:"60";s:2:"x2";s
:3:"150";s:2:"y2";s:2:"60";}i:7;a:4:{s:2:"x1";s:3:"150";s:2:"y1";s:2:"10";s:2:"x
2";s:3:"150";s:2:"y2";s:3:"120";}i:8;a:4:{s:2:"x1";s:3:"200";s:2:"y1";s:2:"10";s
:2:"x2";s:3:"200";s:2:"y2";s:3:"120";}i:9;a:4:{s:2:"x1";s:3:"200";s:2:"y1";s:2:"
10";s:2:"x2";s:3:"260";s:2:"y2";s:2:"10";}i:10;a:4:{s:2:"x1";s:3:"200";s:2:"y1";
s:2:"60";s:2:"x2";s:3:"260";s:2:"y2";s:2:"60";}i:11;a:4:{s:2:"x1";s:3:"260";s:2:
"y1";s:2:"10";s:2:"x2";s:3:"260";s:2:"y2";s:2:"60";}i:12;a:4:{s:2:"x1";s:2:"10";
s:2:"y1";s:3:"130";s:2:"x2";s:3:"260";s:2:"y2";s:3:"130";}i:13;a:4:{s:2:"x1";s:2
:"10";s:2:"y1";s:3:"130";s:2:"x2";s:3:"260";s:2:"y2";s:3:"130";}i:14;a:4:{s:2:"x
1";s:2:"10";s:2:"y1";s:3:"130";s:2:"x2";s:3:"260";s:2:"y2";s:3:"130";}}'
```

Logger

- The logger class is never used
- Writes a message at the beginning and end of a session to a specified file on server using the constructor, destructor, and log
- So, what is the vulnerability?

```php
class Logger{
    private $logFile;
    private $initMsg;
    private $exitMsg;

    function __construct($file){
        // initialise variables
        $this->initMsg="#--session started--#\n";
        $this->exitMsg="#--session end--#\n";
        $this->logFile = "/tmp/natas26_" . $file . ".log";

        // write initial message
        $fd=fopen($this->logFile,"a+");
        fwrite($fd,$initMsg);
        fclose($fd);
    }

    function log($msg){
        $fd=fopen($this->logFile,"a+");
        fwrite($fd,$msg."\n");
        fclose($fd);
    }

    function __destruct(){
        // write exit message
        $fd=fopen($this->logFile,"a+");
        fwrite($fd,$this->exitMsg);
        fclose($fd);
    }
}
```

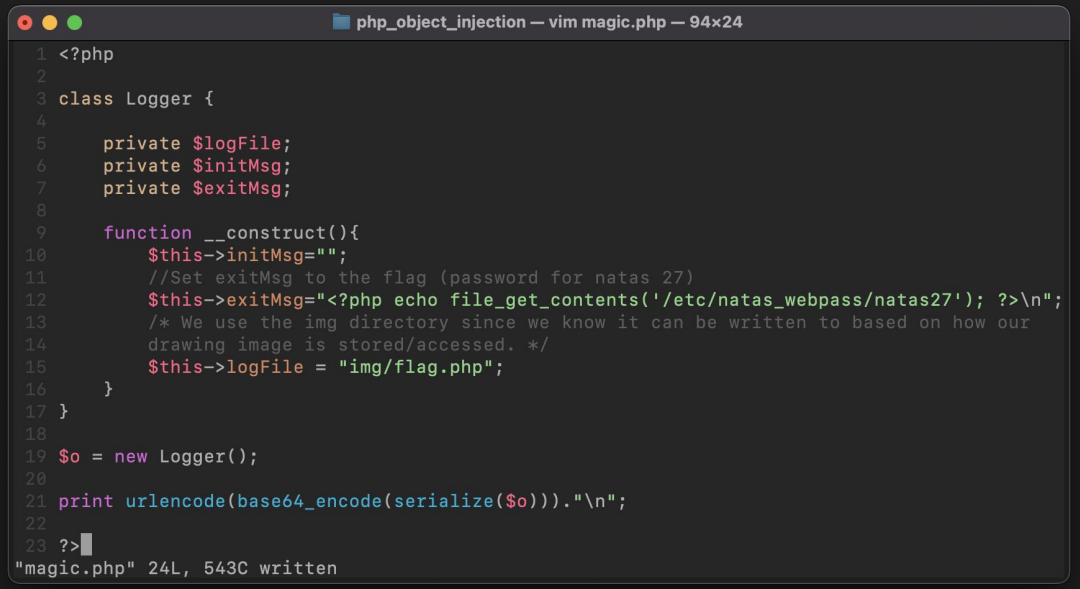
__Magic()

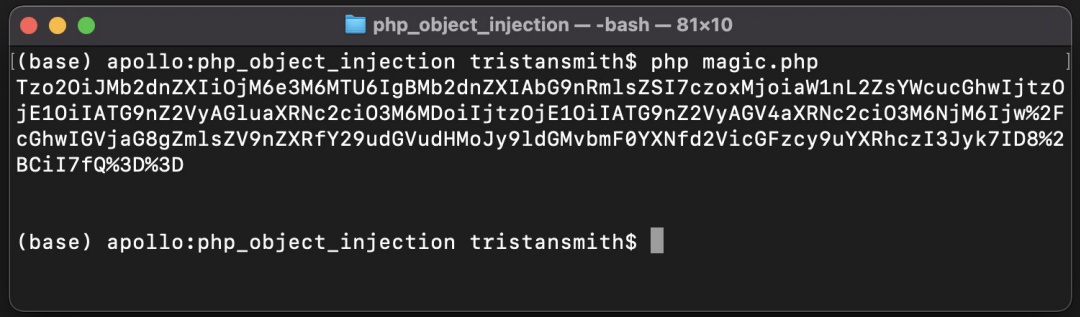
- Magic methods are special methods that override default PHP behavior.
  - Prefixed with “__”
  - __construct(), __destruct(), __wakeup(), etc.
- Since unserialize() doesn’t sanitize input we can use PHP Object Injection and create a serialized Logger object disguised as a “drawing” cookie
- Using our magic methods in Logger we can set member variables as we see fit.

https://www.php.net/manual/en/language.oop5.magic.php#object.unserialize

Serialized logger object

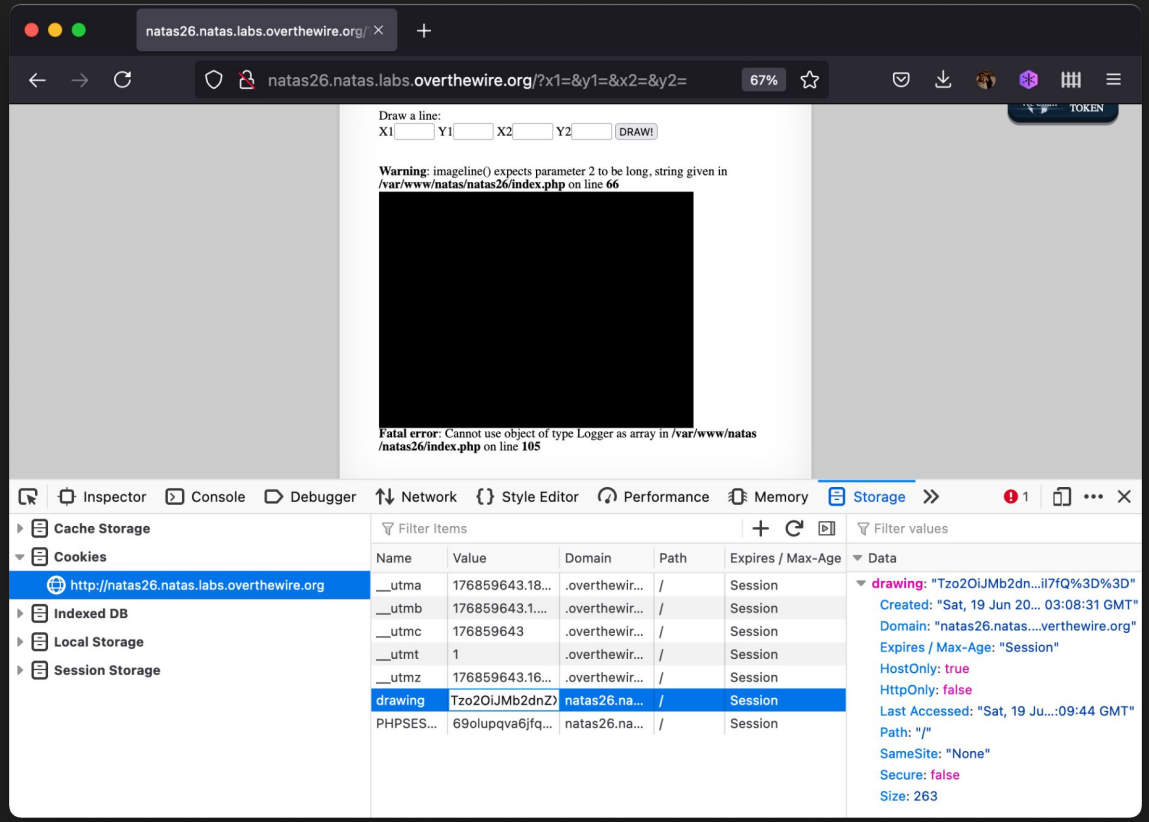
- PHP Script that sets our exit message to the flag.
- Sets the logFile to “img/flag.php” since we know this subdirectory can be accessed.
- Relies on the destructor to create flag.php since it doesn’t exist, and write our exit message to it.
- Serialize, b64 encode, and urlencode!
- Time to inject our malicious object!





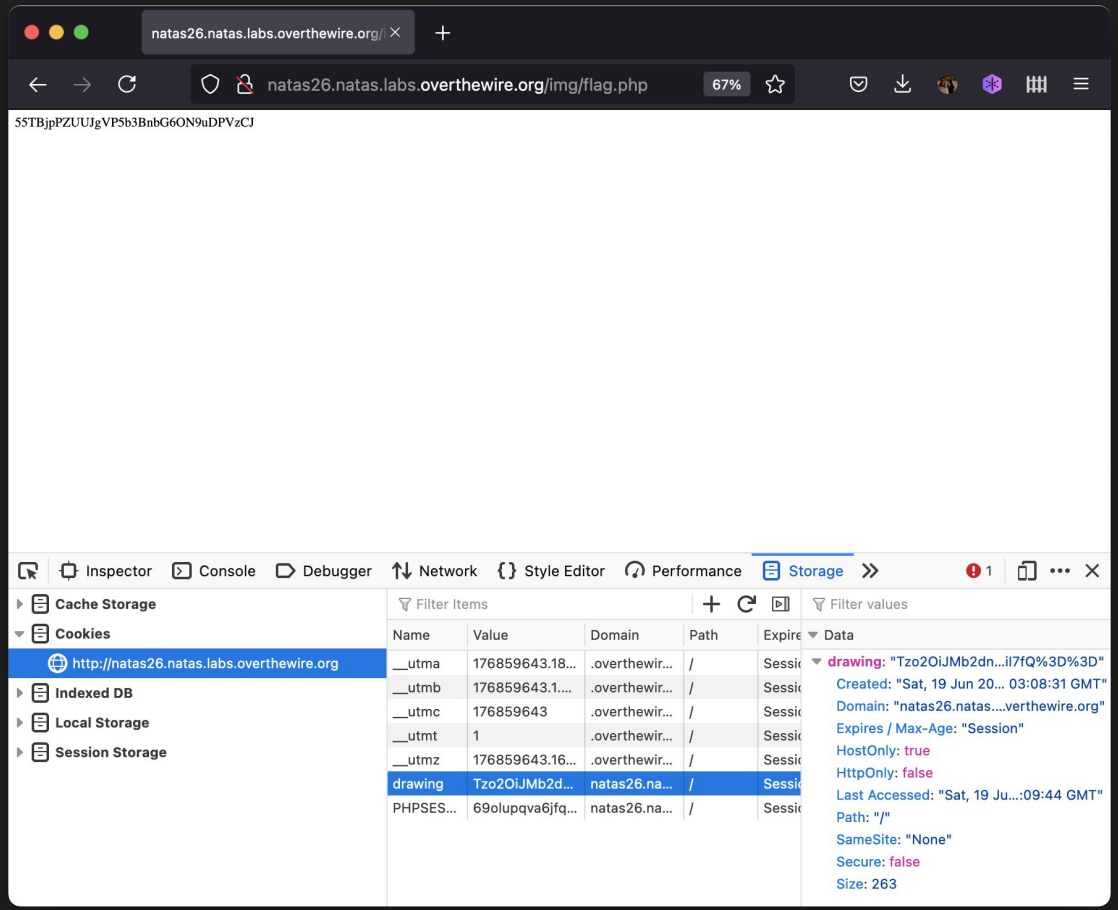
Attack

- Copying the encoded serialized object into our drawing cookie and refreshing the page shows success!
- Fatal error indicates the logger object was successfully ran and destructed.
- Since we control the member variables (i.e. exitMsg and logFile) the destructor should have wrote our flag to the specified logFile (img/flag.php).



The flag!

- The log file was successfully created and our flag is visible!
- Moral of the story, user input and unserialize() don't mix.



Live Walkthroughs

Some Assembly Required 1

Web, but actually Rev
(from the PicoCTF Gym)

https://play.picoctf.org/practice/challenge/152?page=1&search=some%20assembly%20required

WebAssembly (Wasm)

- Stack machine language
- Compiled code in the browser! C, Rust, C++, etc.

example:

i32.const 1 *put a 1 on the stack*

i32.const 2 *put a 2 on the stack - now it's [1, 2]*

i32.add *take the top two values off the stack, add them, put the result on the stack*

Wasm tools

Command line: WABT, emscripten, and more

Browser:

- Dev tools!
- Firefox and Chrome support debugging with breakpoints
  - You may need to reload after opening the Wasm binary in dev tools to see it in text form
- Console access to Wasm memory spaces
  - `memory0.buffer` is the default memory ArrayBuffer in Firefox
  - `memories.$memory.buffer` in Chrome