

Root Me Challenge Writeup

Luis Cruz 03/2025

I. Presentation

- Overview:

In these challenges, I explored various security vulnerabilities across different protocols and technologies. Each challenge provided hands-on experience in identifying, exploiting, and mitigating potential security risks. The key takeaways include understanding cryptographic weaknesses, network-level attacks, DNS misconfigurations, JavaScript bundling risks, and server-side XSS exploitation. These insights are valuable for improving security awareness and strengthening defenses against real-world threats.

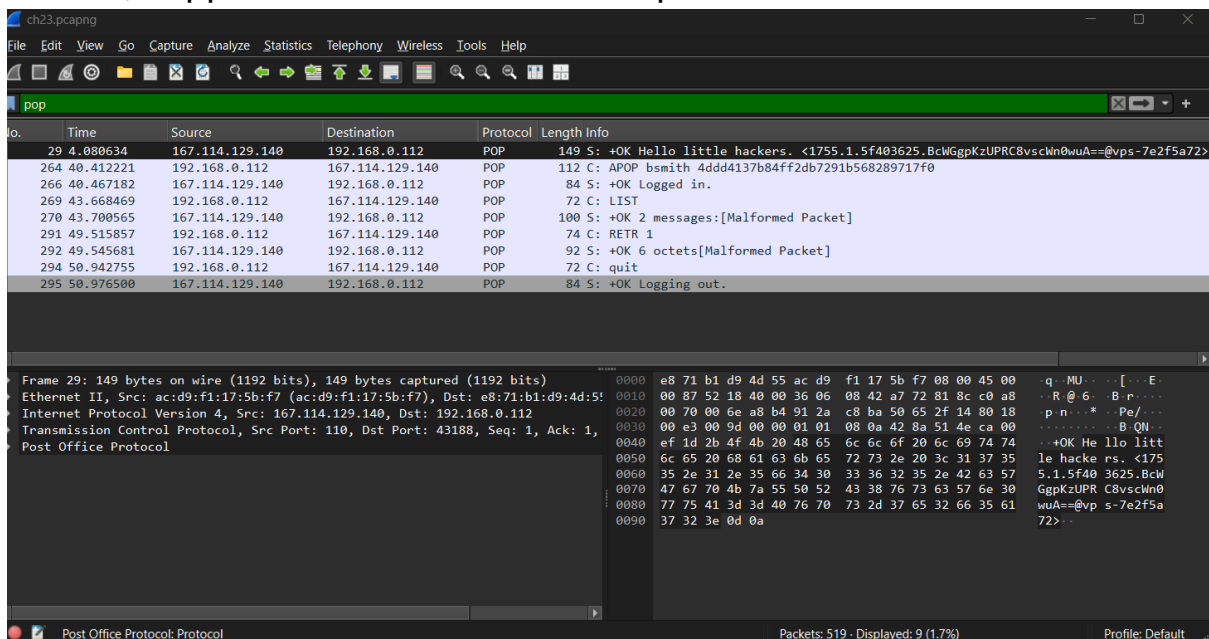
- Challenge 1: POP – APOP
This challenge focused on the Post Office Protocol (POP) and its authentication mechanism, APOP (Authenticated Post Office Protocol). I learned about how APOP hashes passwords with a timestamp but remains vulnerable to replay attacks if the timestamp is predictable. The key takeaway is that cryptographic protocols relying on weak hash-based authentication can be exploited, and modern email security should use stronger authentication mechanisms like OAuth or TLS-encrypted IMAP/POP3.
- Challenge 2: TCP – Back to College
This challenge involved analyzing TCP communication, by intercepting and reconstructing messages. I gained insights into how TCP streams can be manipulated or observed through packet capture tools like Wireshark. The main lesson here is that sensitive data transmitted over unencrypted channels can be intercepted, reinforcing the importance of encryption protocols like TLS for data security.
- Challenge 3: DNS-zone-transfer
This challenge dealt with DNS zone transfer vulnerabilities, where misconfigured DNS servers allow unauthorized access to internal records. I learned how an attacker could retrieve detailed domain information using the dig or host command. The key takeaway is that administrators should restrict zone transfers to trusted sources and implement proper access controls to prevent data leaks.
- Challenge 4: JavaScript – Webpack
This challenge revolved around JavaScript bundling with Webpack and potential security risks. I discovered how improperly configured Webpack setups could expose sensitive data, such as environment variables or debug information. The lesson here is that developers must review their build configurations and avoid including unnecessary information in client-side JavaScript to minimize exposure.
- Challenge 5: XSS – Server Side

This challenge focused on Server-Side XSS, where user input is improperly handled on the server, leading to script execution on the backend. I learned how this differs from traditional client-side XSS and how it can be used to access sensitive server-side resources. The primary takeaway is that developers must sanitize and validate user input on both client and server sides to prevent exploitation.

II. Solution Steps

Challenge 1: POP - APOP

I began by downloading and extracting the provided file. After unzipping, I found a pcapng file, which I opened in the Wireshark app. The challenge name hinted at its focus, so I researched the POP (Post Office Protocol) to understand its functionality. Using Wireshark's search feature, I applied a filter to isolate POP packets.



Next, I researched APOP authentication and discovered that it encrypts passwords using MD5 with salt to encrypt passwords. Additionally, I found examples of APOP-encrypted passwords, which helped me recognize the encrypted string in Wireshark. The challenge-provided resources further clarified POP mechanics with examples and relevant strings.

A quick Google search suggested using Hashcat to crack the password. Since I had never used Hashcat before, I watched some YouTube tutorials and looked up relevant commands to understand the process.

Using Kali Linux, I installed the rockyou.txt wordlist and created a file named hash.txt to store the hash. Initially, I encountered multiple CPU-related errors, which required increasing my virtual machine's CPU allocation. Once adjusted, I returned to Linux and successfully cracked the password using the following command:

```
hashcat -a 0 -m 20 hash.txt /usr/share/wordlists/rockyou.txt --show
```

-a (Attack Mode)

-m (Hash Type)

--show (Display cracked passwords)

Executing this command revealed the password, allowing me to complete the challenge successfully.



```
(kali@kali)-[~/Documents]
$ hashcat -a 0 -m 20 hash.txt /usr/share/wordlists/rockyou.txt --show
4ddd4137b84ff2db7291b568289717f0:<1755.1.5f403625.BcWGgpKzUPRC8vscWn0wuA=@vp
s-7e2f5a72>:100%popprincess
```

Challenge 2: TCP – Back to School

Start with reading the provided resources and learning more about TCP. Developed in the early 1970s, Transmission Control Protocol (TCP) is a transmission protocol used on IP networks. It is described in detail in IETF RFC 793.

I also learned Python is one of the programming languages that connect to TCP and there were some notable examples of code to connect to a TCP server.

Next, I proceeded to set up a connection on Linux.

First, I updated my Linux, installed python3 and confirmed the download using the following commands:

`sudo apt update -y, sudo apt install python3 -y, & Python --version`

I needed to find a python script that.

Next, I created a folder using a text editor (nano) so that I could add a python script that would create a TCP connection to the server and run some computations and return the results back to the server.

```
1  import socket
2  import re
3  import math
4
5  def main():
6      host = "challenge01.root-me.org"
7      port = 52002
8
9      # Create a TCP socket
10     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
11         s.connect((host, port))
12
13         # Receive data from the server
14         data = s.recv(1024).decode()
15         print("Received:", data)
16
17         # Extract numbers using regex
18         match = re.search(r"([0-9]+) .* ([0-9]+)", data)
19         if match:
20             num1 = int(match.group(1))
21             num2 = int(match.group(2))
22
23             # Perform calculation
24             result = round(math.sqrt(num1) * num2, 2)
25             print(f"Computed result: {result}")
26
27             # Send result back
28             s.sendall(f"{result}\n".encode())
29
30             # Receive final response
31             response = s.recv(1024).decode()
```

I asked ChatGPT to create a python script for me that would make a connection, provide the calculation given and complete it in 2 seconds before the program times out. I also used Github to look at some prewritten python scripts.

Next, I ran the script to gain my flag.

```
kali@kali: ~  
File Actions Edit View Help  
└─$ python3 script.py  
Received: _____  
GO BACK TO COLLEGE  
_____  
You should tell me the answer of this math operation in less than 2 seconds !  
  
Calculate the square root of 340 and multiply by 1967 =  
Error: Could not extract two numbers.  
  
└─(kali@kali)-[~]  
└─$ sudo nano script.py  
  
└─(kali@kali)-[~]  
└─$ python3 script.py  
Received: _____  
GO BACK TO COLLEGE  
_____  
You should tell me the answer of this math operation in less than 2 seconds !  
  
Calculate the square root of 910 and multiply by 5604 =  
Sending: 169051.42  
Response: [+] Good job ! Here is your flag: RM{TCP_C0nnecT_4nD_m4Th}
```

Challenge 3: DNS-zone-transfert

Start by reading provided resources to learn about DNS zone transfer, dig and nslookup.

Next a simple Google search confirms that nslookup and dig are both tools capable of interacting with DNS.

Next, I opened the URL provided `ch11.challenge01.root-me.org` and realized that it changes over to the host page <http://challenge01.root-me.org/>. One of these sites is the nameserver for the other.

A nslookup on Linux confirms the nameserver and provides the IP address.

```

(kali㉿kali)-[~]
$ nslookup
> set port=54011
> set type=ANY
> SERVER challenge01.root-me.org
Default server: challenge01.root-me.org
Address: 212.129.38.224#54011
Default server: challenge01.root-me.org
Address: 2001:bc8:35b0:c166::151#54011
>

```

Next, I wanted to find a command that I could use with the dig command in Linux.

```

(kali㉿kali)-[~]
$ dig -h
Usage: dig [@global-server] [domain] [q-type] [q-class] {q-opt}
        {global-d-opt} host [@local-server] {local-d-opt}
        [ host [@local-server] {local-d-opt} [ ... ] ]
Where: domain    is in the Domain Name System
q-class    is one of (in,hs,ch, ... ) [default: in]
q-type     is one of (a,any,mx,ns,soa,hinfo,axfr,txt, ... ) [default:a]
           (Use ixfr=version for type ixfr)
q-opt      is one of:
            -4                      (use IPv4 query transport only)
            -6                      (use IPv6 query transport only)
            -b address[#port]      (bind to source address/port)
            -c class                (specify query class)
            -f filename            (batch mode)
            -k keyfile              (specify tsig key file)
            -m                      (enable memory usage debugging)
            -p port                 (specify port number)
            -q name                 (specify query name)
            -r                      (do not read ~/.digrc)
            -t type                 (specify query type)
            -u                      (display times in usec instead of msec)
            -x dot-notation         (shortcut for reverse lookups)
            -y [hmac:]name:key      (specify named base64 tsig key)
d-opt      is of the form +keyword[=value], where keyword is:
            +[no]aaflag             (Set AA flag in query (+[no]aaflag))

```

I tried several commands with no luck, it was trial and error until I found a command that worked.

```

(kali㉿kali)-[~]
$ dig TXT @challenge01.root-me.org -p 54011 ch11.challenge01.root-me.org

; <<>> DiG 9.20.2-1-Debian <<>> TXT @challenge01.root-me.org -p 54011 ch11.ch
allenge01.root-me.org
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 23123
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 2e37098183895fd40100000067bba5c01d6956aee2b1af9c (good)
;; QUESTION SECTION:
;ch11.challenge01.root-me.org. IN      TXT

;; ANSWER SECTION:
ch11.challenge01.root-me.org. 604800 IN TXT      "DNS transfer secret key : CB
kFRwfNMMtRjHY"

```

`dig TXT @challenge01.root-me.org -p 54011 ch11.challenge01.root-me.org`

Here is a breakdown of the command I used:

dig= The dig (Domain Information Groper) command is a DNS query tool used to retrieve DNS records for a domain.

TXT= Specifies the type of DNS record to query. In this case, a TXT record, which often contains arbitrary text or verification keys.

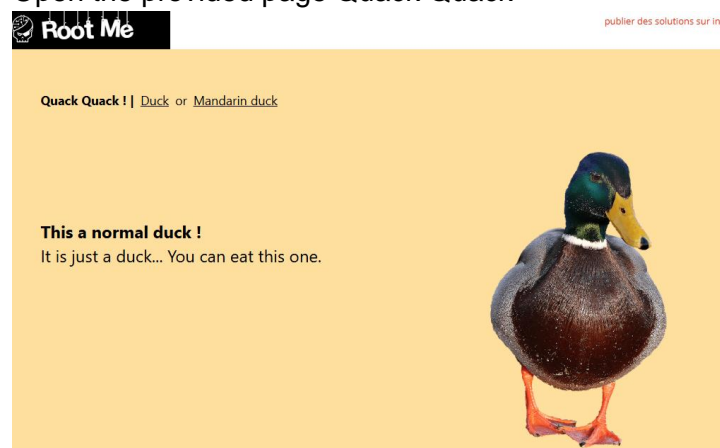
Followed by the DNS server to Query= @challenge01.root-me.org

-p 54011= pointing at the port being used provided by the challenge

Followed by ch11.challenge01.root-me.org which is the domain name for which TXT record is being queried.

Challenge 4: JavaScript – Webpack

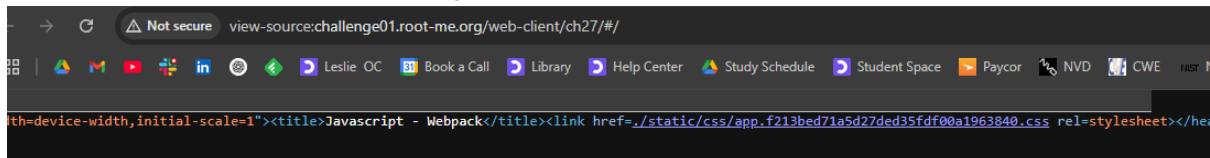
Open the provided page Quack Quack



Enter the sources tab in the site Dev Tools (F12)

As the title of the challenge is called webpack, I went straight to the webpack folder and started going through the file. I came across a file *YouWillNotFindThisRouteBecauseItsHidden.vue*.

I then went to the page source and noticed the HTML source referenced a CSS file with the following path:



This revealed that CSS files were stored under /static/css/ Since CSS files were inside /static/css/ files it is likely that JavaScript files would be inside /static/js/ since that's the typical structure of a standard directory.

Name	Date modified	Type	Size
css	14-Mar-19 6:44 PM	File folder	
fonts	20-Jun-18 12:46 AM	File folder	
images	05-May-20 9:35 PM	File folder	
js	14-Mar-19 6:44 PM	File folder	
scss	20-Jun-18 12:46 AM	File folder	
.DS_Store	21-Mar-19 7:10 PM	DS_STORE File	11 KB
index - Copy.html	06-May-20 2:21 PM	Chrome HTML Do...	13 KB
index.html	05-May-20 9:38 PM	Chrome HTML Do...	13 KB
prepros-6.config	20-Mar-19 1:06 AM	CONFIG File	18 KB

Assuming this I tested the URL:

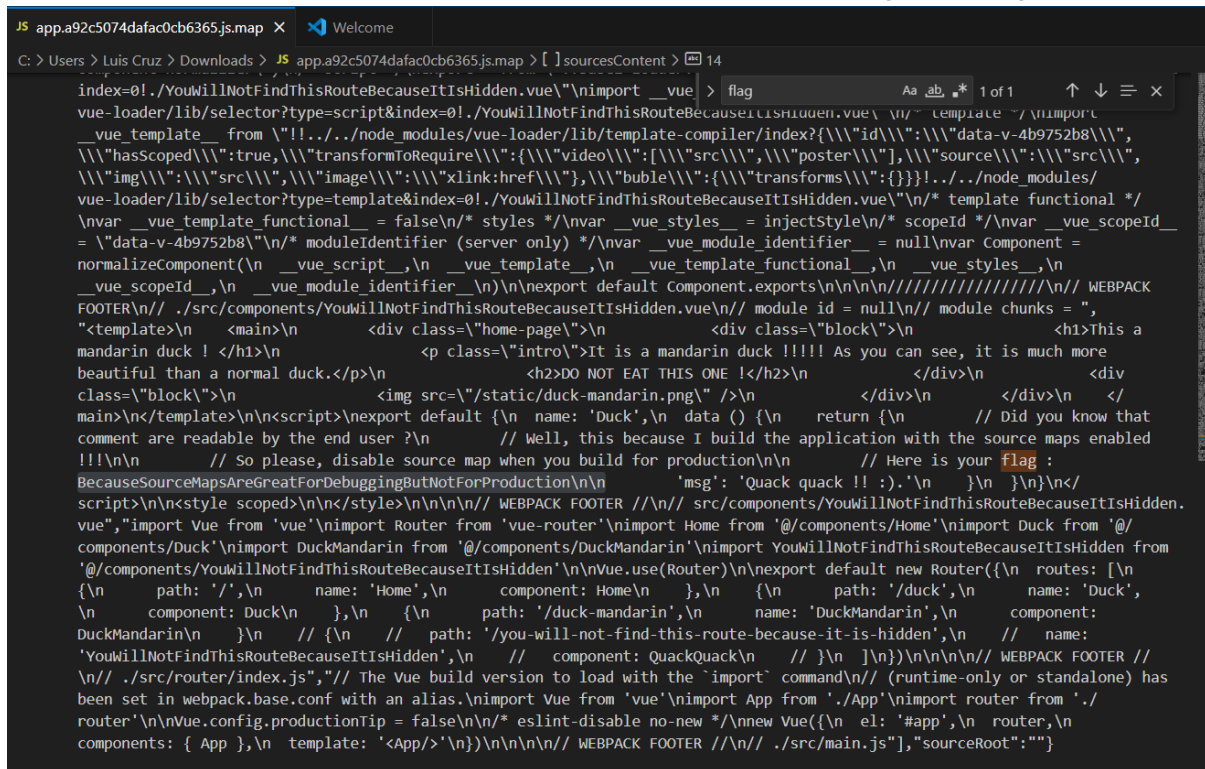
A screenshot of the Root Me website. The URL in the address bar is 'challenge01.root-me.org/web-client/ch27/static/js/'. The page shows a directory listing for '/web-client/ch27/static/js/'. The listing includes a table with columns for File Name, File Size, and Date.

/web-client/ch27/static/js/		
File Name ↓	File Size ↓	Date ↓
Parent directory/	-	-
app.a92c5074dafac0cb6365.js	3.6 KiB	2021-Dec-10 19:12
app.a92c5074dafac0cb6365.js.map	24.9 KiB	2021-Dec-10 19:12
manifest.2ae2e69a05c33dfc65f8.js	857 B	2021-Dec-10 19:12
manifest.2ae2e69a05c33dfc65f8.js.map	4.9 KiB	2021-Dec-10 19:12
vendor.458c9f5863b8f28e5570.js	118.4 KiB	2021-Dec-10 19:12
vendor.458c9f5863b8f28e5570.js.map	598.5 KiB	2021-Dec-10 19:12

This directory contained JavaScript files including the source map (.map) file.

Webpack often generates source maps which allow developers to debug JavaScript.

I downloaded [app.a92c5074dafac0cb6365.js.map](#) and used Visual Studio to view the contents of the download, revealing the flag.

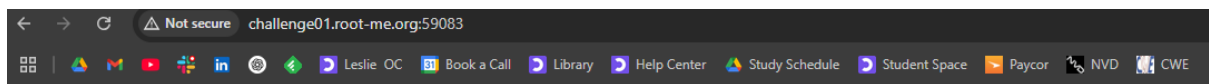


```
index=0!./YouWillNotFindThisRouteBecauseItIsHidden.vue\n\nimport __vue__ from 'vue-loader/lib/selector?type=script&index=0!./YouWillNotFindThisRouteBecauseItIsHidden.vue'\nimport __vue_template__ from '!!../node_modules/vue-loader/lib/template-compiler/index?{\\\"id\\\":\\\"data-v-4b9752b8\\\",\\\"hasScoped\\\":true,\\\"transformToRequire\\\":{\\\"video\\\":{\\\"src\\\",\\\"poster\\\"},\\\"source\\\":{\\\"src\\\",\\\"img\\\":{\\\"src\\\",\\\"image\\\":{\\\"xlink:href\\\"},\\\"buble\\\":{\\\"transforms\\\":{}}}}!../node_modules/vue-loader/lib/selector?type=template&index=0!./YouWillNotFindThisRouteBecauseItIsHidden.vue'\nimport __vue_template_functional__ from 'vue-loader/lib/selector?type=template&index=0!./YouWillNotFindThisRouteBecauseItIsHidden.vue'\nimport __vue_styles__ from 'vue-loader/lib/selector?type=styles?index=0!./YouWillNotFindThisRouteBecauseItIsHidden.vue'\n\nconst __vue__ = {\n  __vue_template__: __vue_template__,\n  __vue_template_functional__: __vue_template_functional__,\n  __vue_styles__: __vue_styles__,\n  __vue_module_identifier__: 'data-v-4b9752b8',\n  __vue_module_identifier__: null,\n  __vue_component__: 'YouWillNotFindThisRouteBecauseItIsHidden.vue',\n  __vue_module_id__: null,\n  __vue_module_chunks__: [],\n  __vue_module_exports__: {\n    __esModule: true,\n    default: {\n      name: 'YouWillNotFindThisRouteBecauseItIsHidden',\n      components: {\n        Duck: {\n          name: 'Duck',\n          data: () => {\n            return {\n              // Did you know that comment are readable by the end user ?\n              // Well, this because I build the application with the source maps enabled\n              // So please, disable source map when you build for production\n              // Here is your flag :\n              msg: 'Quack quack !! :).'\n            }\n          }\n        },\n        DuckMandarin: {\n          name: 'DuckMandarin',\n          data: () => {\n            return {\n              // Did you know that comment are readable by the end user ?\n              // Well, this because I build the application with the source maps enabled\n              // So please, disable source map when you build for production\n              // Here is your flag :\n              msg: 'Quack quack !! :).'\n            }\n          }\n        }\n      }\n    }\n  }\n}\n\nexport default __vue__\n\n// WEBPACK FOOTER\n//\n// src/components/YouWillNotFindThisRouteBecauseItIsHidden.vue\n//\n// import Vue from 'vue'\n// import Router from 'vue-router'\n// import Home from '@components/Home'\n// import Duck from '@components/Duck'\n// import DuckMandarin from '@components/DuckMandarin'\n// import YouWillNotFindThisRouteBecauseItIsHidden from '@components/YouWillNotFindThisRouteBecauseItIsHidden'\n//\n// const routes = [\n//   {\n//     path: '/',\n//     name: 'Home',\n//     component: Home\n//   },\n//   {\n//     path: '/duck',\n//     name: 'Duck',\n//     component: Duck\n//   },\n//   {\n//     path: '/duck-mandarin',\n//     name: 'DuckMandarin',\n//     component: DuckMandarin\n//   },\n//   {\n//     path: '/you-will-not-find-this-route-because-it-is-hidden',\n//     name: 'YouWillNotFindThisRouteBecauseItIsHidden',\n//     component: YouWillNotFindThisRouteBecauseItIsHidden\n//   }\n// ]\n//\n// const router = new Router({\n//   routes\n// })\n//\n// Vue.config.productionTip = false\n//\n// eslint-disable no-new\n//\n// new Vue({\n//   el: '#app',\n//   router,\n//   components: { App },\n//   template: '<App/>'\n// })\n//\n// WEBPACK FOOTER\n//\n// ./src/main.js\n//\n// sourceRoot: ''
```

Challenge 5: XSS – Server Side

I started this challenge by inspecting the page and looking through the sources tab only to find false flags that I wasted time trying to decrypt.

Next, I went on the webpage and noticed that if I typed characters into the empty box, it would generate a download (attestation pdf file).



Home

Root-Me attestations generator

Enter the message you want to certify

Hello

Generate

attestation (3)	2/25/2025 3:35 PM	Microsoft Edge P...	23 KB
attestation	2/25/2025 3:35 PM	Microsoft Edge P...	23 KB
app.a92c5074dafac0cb6365.js.map	2/24/2025 9:21 PM	MAP File	25 KB

Instead of displaying my input as plain text, the application executed an iframe element.

I studied the contents of the download with the Visual Studio app but did not find anything. I searched on GitHub for payloads that I could potentially add to the generator as an attack.

Product Solutions Resources Open Source Enterprise Pricing

Search or jump to...

payloadbox / xss-payload-list Public

Sponsor

Code Issues Pull requests Actions Projects Security Insights

master 6 Branches 0 Tags

Go to file

Code

About

Cross Site Scripting (XSS) Vulnerability Payload List

ismailtasdelen.medium.com

xss xss-vulnerability xss-scanners

bugbounty xss-scanner xss-exploitation

xss-detection payload payloads

xss-attacks xss-injection websecurity

dom-based xss-poc cross-site-scripting

reflected-xss-vulnerabilities

website-vulnerability xss-payloads

self-xss xss-payload

Readme

MIT license

Activity

Custom properties

6.7k stars

141 watching

1.7k forks

Report repository

Releases

No releases published

Sponsor this project

liberapay.com/ismailtasdelen

Cross Site Scripting (XSS) Vulnerability Payload List

awesome 6.7k Stars 1.7k Forks repo size 270 KB license MIT issue/pull request issue, pull request or repo not found

Overview :

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. For more details on the different types of XSS flaws, see: [Types of Cross-Site Scripting](#).

XSS Vulnerability Scanner Tool's :

Root-Me attestations generator

Enter the message you want to certify

```
<script>alert(123)</script>
```

Generate

The new download generated did not provide anything different.
I decided to try the same payload in the user sign in and sign-up options with no results.
I tried many payloads and decided to look for a payload for iframe.
Next, I created a user and logged in and the results were unchanged.
Finally, I decided to create a new user with the payload, I entered the payload in every field at the same time.

Sign up

Login

```
<iframe src=file:///flag.txt></iframe>
```

First name

```
<iframe src=file:///flag.txt></iframe>
```

Last name

```
<iframe src=file:///flag.txt></iframe>
```

Password

.....

Submit

To my luck it created a user with the payload username.
`<iframe src="file:///flag.txt"></iframe>`
`<iframe>` is an HTML element that embeds another webpage inside the current page.
The `src` attribute specifies the content to load inside the `iframe`.

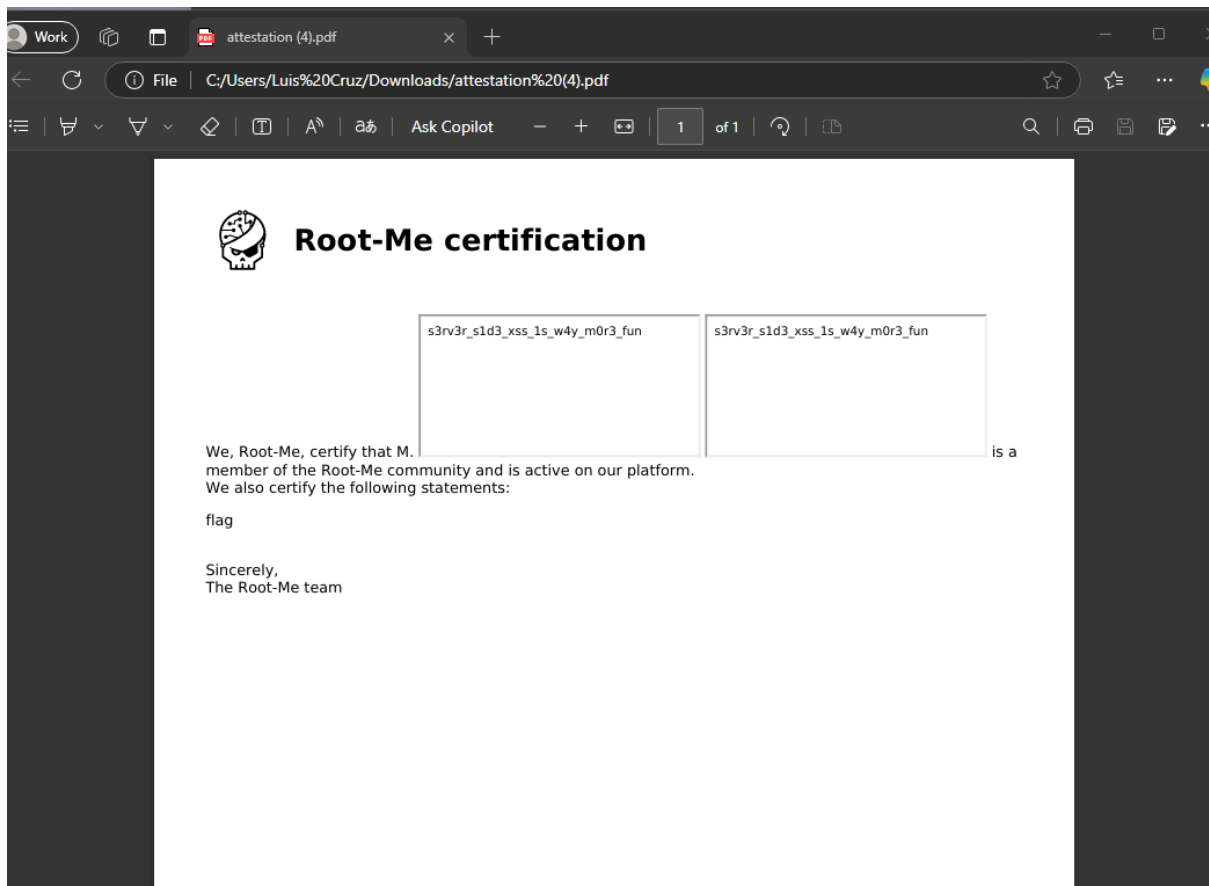
[File:///flag.txt](#) attempts to load a local file from the server's filesystem.

Welcome, <iframe src=file:///flag.txt> </iframe> <iframe src=file:///flag.txt> </iframe>!

Logout

Root-Me attestations generator

I tried to input the same payload name into the generator. I also tried typing flag.txt with no luck until I entered flag in the generator and the downloaded contents revealed the flag.






III. Confirmation of Completion

Insert here the screenshots confirming completion of each of the five challenges (flag).

- Challenge 1:



POP - APOP






15 Points

Secured authentication

Author
lutzenfried, 11 November 2020

Level 


Validations
7723 Challengers 

Note 
 387 Votes

I like


I don't like

Statement

Find the user password in this network frame.

Download the challenge


1 related ressource(s)

-  rfc1939 (RFC)

Validation




Well done but you've already won the 15 Points

Don't forget to give your opinion on the challenge by voting ;-)


 tweet it!

- Challenge 2:

TCP - Back to school






5 Points





Network programming

Author
Nishacid, M4tou, 19 June 2023

Level 


Validations
6713 Challengers  2%

Note 
 567 Votes

I like

I don't like

Statement


To start this test using the TCP protocol, you need to connect to a program on a network socket.

- › Calculate the square root of number 1 and multiply by number 2.
- › Then round the result to two decimal places.
- › You have 2 seconds to send the correct answer from the moment the program sends you the calculation.

Challenge connection informations

Host	challenge01.root-me.org
Protocol	TCP
Port	52002


Vulnerability sheet(s)

 [Programming - TCP \[EN\]](#)

Validation




Well done, you won 5 Points

Don't forget to give your opinion on the challenge by voting ;-)


 [tweet it!](#)

- Challenge 3:

DNS - zone transfert






15 Points





Network service

Author
g0uZ, 20 May 2013

Level 


Validations
25773 Challengers  8%

Note 
 1626 Votes

I like

I don't like


Statement

A not really dutiful administrator has set up a DNS service for the "ch11.challenge01.root-me.org" domain...






Challenge connection informations

Host	challenge01.root-me.org
Protocol	DNS
Port	54011

Vulnerability sheet(s)

 [DNS - Zone Transfer \[EN\]](#)


5 related ressource(s)

-  [Faiblesse des serveurs DNS par transfert de zone. \(Application\)](#)
-  [Master Slave XFR Transfert de Zone \(Réseau\)](#)
-  [rfc1035 \(RFC\)](#)
-  [rfc5936 \(RFC\)](#)
-  [rfc1034 \(RFC\)](#)

Validation

Well done, you won 15 Points

Don't forget to give your opinion on the challenge by voting ;-)

 [tweet it!](#)

- Challenge 4:

Javascript - Webpack

15 Points

Do you know webpack?

Author
CanardMandarin, 11 August 2020

Level

Validations
26968 Challengers

Note
★★★★★ 1309 Votes

Statement
Find the password.

1 related ressource(s)
▪ Webpackjs - Devtool (Exploitation - Web)

Validation

Well done, you won 15 Points

Don't forget to give your opinion on the challenge by voting :)

tweet it!

- Challenge 5:

XSS - Server Side

20 Points

Who said XSS was only for the client side?

Author
Elv, 23 June 2023

Level

Validations
2747 Challengers

Note
★★★★★ 173 Votes

Statement

Challenge created during the "HackDay 2023" CTF

This platform for issuing certificates of participation has just gone live. The developers assure you that they have followed best practices and escaped all user inputs before using them in their code...

The flag is located in the ``/flag.txt`` file.

Validation

Well done, you won 20 Points

Don't forget to give your opinion on the challenge by voting :)

tweet it!

IV. Reflections and Conclusions

Briefly describe your experience while completing the challenges. Answer the following questions:

- *Were there any difficulties you encountered? How did you overcome them?*

One of the main difficulties I faced was identifying where to start with each challenge. After having a challenging time with the first challenge I realized that the resources provided were a huge helping point.

- *Were some challenges easier or harder than others? Why?*

The hardest challenge was the easiest for me since I have done challenges in the past that required payloads. The Pop-APOP challenge was also not as hard as the others since I have worked with Wireshark in the past. The challenges were harder than I thought they would be, and I spent a lot of time trying to capture the flags and doing the write-ups.

- *Were the solutions obvious to you or did you have to work to figure them out? If so, how did you arrive at the solutions?*

It seemed like none of the solutions were obvious. The closest I got to one being obvious is the first challenge where the encrypted flag showed up right way and all I had to do was decrypt it. Thinking creatively and trying new ways is what got me through the challenges.

- *Did you use any outside resources to find information for your solutions? If so, which ones? How did you choose them and how did you use them?*

I tried to use outside sources, but nothing is available online for these challenges. When I was doing the write ups, I used ChatGPT to help me write some of them up and even ChatGPT said my write up was not possible for the XSS Server challenge.

- *If you did not encounter any difficulties, why do you think this is?*

I encountered difficulties.

V. Optional - ChatGPT Use

You may use ChatGPT to generate your writeup if you wish. If you do so, please list here any prompts you used, as well as a description of how you used the information from the scans and Parts II and III of the report to generate the summary.

A. Prompts:

1. Help me re-write this write up for XSS Server Side root-me.org challenge.
2. Help me get started with root-me.org challenge JavaScript Webpack.

B. Description:

1. I did not use the write-up, instead I tried to explain to the app how I completed the challenge.
2. Provided useful information detailing Webpack, Dev Tools, js.map, and SourceMapVisualizer.