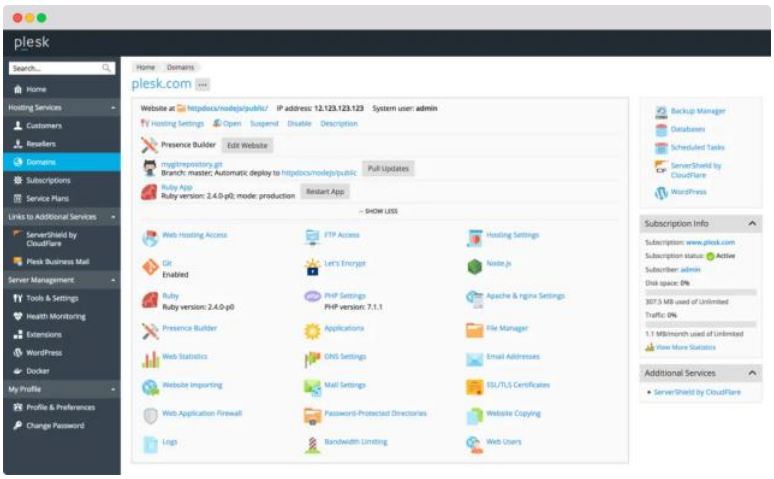
Search ...

SEARCH

CATEGORIES

COMPROMISING PLESK VIA ITS REST API



Plesk Dashboard

INTRO

Plesk is a commercial web hosting and server data center automation software developed for Linux and Windows-based retail hosting service providers. It's the main choice of web hosting providers these days being used by 86.7% of the websites that use a web panel for administration. This is 4.4% of all websites and there around 2M Plesk installations in the US alone. As expected there are many interesting features to attack as an administrator, however we couldn't find anything really exploitable and also it isn't that interesting to begin with, if you're already an administrator, right? We tried to see if we can escalate our privileges from one of the limited roles, but these seem solid. In the end we discovered a cookieless CSRF, which is basically a design issue in this case, because it affects all the POST requests and we could abuse most of the APIs with it.

Generally speaking, it's obvious that Plesk has been through quite a few pentests and overall has a good security posture. The last thing we wanted to focus our attention on was the REST API. We were wondering, has the REST API the same level of security as the other Plesk components? Also we need to mention that the source code Plesk is highly obfuscated with a custom PHP extension.

Misconfigured CORS policy

While reviewing the REST API, we quickly noticed that there was a completely open CORS policy:

Using a wildcard to allow everything

So, we thought, wow this is definitely a good sign and we were on the right track testing the REST API. This will basically allow us to send any request with any methods/headers from any origin and read the response back. Or so we thought.

Cookieless CSRF #1 – add secret token

The administrator can call the REST APIs from the browser and there's no CSRF protection either (partially correct as you will see next). Let's try to CSRF an Administrator then. But what API endpoint shall we target? After reviewing the REST API, we discovered that there is the /api/v2/auth/keys endpoint which we thought of abusing by adding a secret key to give us admin access.

API endpoint to add a secret key for Administrator

This key will allow us to authenticate and call any endpoints after that, no more CSRF needed 🙂 Perform CSRF once, add a backdoor token and get full access forever. Sweet, let's try this.

CSRF first attempt failed

We have used Burp's CSRF POC generator and it seems that the request fails due to "=" added at the end, the server can't parse this JSON. This was no big issue, this old dog knows some old tricks and by reformatting our payload a bit we were able to generate a valid JSON.

Name field of the input will hold most of the JSON field

The input's name is html encoded and basically decodes to the following. The input's value field will hold 'test"}' so that when it's concatenated with the name it will result in a valid JSON payload. Nothing new here.

Name field html decoded

HTML POC looks like this:

```
<html>
  <!-- CSRF PoC -- FORTBRIDGE -- add a secret  key-->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://VICTIMIP:8443/api/v2/auth/keys" method="POST" enctype="text/plain">
      <input type="hidden"
name='&#123;&#10;&#32;&#32;&quot;ips&quot;&#58;&#32;&#91;&#10;&#32;&#32;&#32;&#32;&quot;86&#46;189
&#46;139&#46;122&quot;&#10;&#32;&#32;&#93;&#44;&#10;&#32;&#32;&quot;login&quot;&#58;&#32;&quot;adm
in&quot;&#44;&#10;&#32;&#32;&quot;description&quot;&#58;&#32;&quot;' value='test"}' />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Now we were able to create a valid CSRF payload and generate a secret key as you can see bellow. We also have a miconsfigured CORS policy, so we can steal it, right?

CSRF attack on the Administrator

Well, not really. What we've missed initially is that the Authorization header gets added automatically by the browser when using an html form to perform the CSRF attack using the POST method. However, when using the javascript XHR object to create the same request and get the token from the response, the Authorization header isn't added anymore. Bummer, our CSRF fails in this case as we can't read the key! But wait, there's still some hope left. Maybe we can find some other REST endpoints where we can trigger a CSRF attack using HTML forms (POST based) that will be impactful enough to give an attacker what they need without reading the response. And we sure found quite a few of these endpoints that we've exploited with various degree of severity as you'll see next.

Cookieless CSRF #2 – add DB user

We found a REST endpoint that allows us to add a db user which can connect from ANY remote host to ANY database.

Add DB user (CSRF-able)

Result looks like bellow:

DB user added

We thought of injecting an UDF for a quick RCE, however the mysql port is not accessible. Lesson learned, check the port first.

Mysql port filtered

Cookieless CSRF #3 – add FTP user

We managed to add a new FTP user with access to any existing domain we wanted.

Add FTP user (CSRF-able)

This actually worked, you can connect via FTP, inspect the client's home dir and conduct further attacks from there.

FTP connect OK

Cookieless CSRF #3 – add a malicious Plesk extension

We didn't know what Plesk extensions were initially, so we did some quick research, backdoored an existing one and added it via CSRF upload using the REST API

Plesk malicious extension added via CSRF in the REST API

The problem was that you still need to authenticate in Plesk to access the backdoored extension. Certainly there must be an easier way, right?

Cookieless CSRF #4 – compromise the admin

Plesk implements some custom commands internally (a "command" is an abrastraction layer if you will in the code which calls various cli tools to do the heavy lifting) in order to manage the server as you can see bellow. One of these, is the "admin" command that allows you to do multiple things, including changing the Administrator's password.

Plesk custom commands callable via REST API

And that is exactly what we needed. We ended up compromising Plesk via its REST API (CSRF) and change the Admin's password. Game over!

CSRF-ing the Administrator to change his password.

HTML POC used to trigger the above request:

```
<html>
  <!-- CSRF PoC - FORTBRIDGE -  change admin password -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://VICTIMIP:8443/api/v2/cli/admin/call" method="POST" enctype="text/plain">
      <input type="hidden"
name='&#123;&#10;&#9;&quot;params&quot;&#58;&#32;&#91;&#32;&quot;&#45;&#45;set&#45;admin&#45;passw
ord&quot;&#44;&quot;&#45;passwd&quot;&#32;&#44;&quot;'
value='F0rTBr1DG3&#33;&#64;&#35;&quot;&#93;&#10;&#125;' />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

The documentation around this feature isn't great so it took a while to figure out the above payload. All POST requests can be CSRF-ed , because the browser adds the Authorization header automatically when using html forms. We noticed that there is a common misconception that an application using the Authorization header is protected against CSRF. As long as you don't have to read the response back using javascript everything just works.

Timeline

10/05/22 FORTBRIDGE reaches out to PLESK for vuln disclosure

11/05/22 Plesk confirms receipt and starts investigation

19/05/22 Plesk confirms the vulnerability and starts working on a fix, no ETA offered

03/06/22 Plesk fixes the issue, but requests ***delaying*** the disclosure, FORTBRIDGE accepts

25/07/22 FORTBRIDGE requests update on the patching progress

27/07/22 Plesk requests ***delaying*** the disclosure again, FORTBRIDGE accepts

03/08/22 Plesk estimates "no longer than 2 weeks" for disclosure

16/09/22 FORTBRIDGE requests update

16/09/22 Plesk requests ***delaying*** the disclosure, FORTBRIDGE agrees to delay 2 months max

08/11/22 FORTBRIDGE releases blogpost with technical details of the Plesk Admin takeover issue

10/11/22 blog post updated with Plesk Call to Action

References

https://support.plesk.com/hc/en-us/articles/8497233114514 Plesk Call to Action

RCE & Privesc in cPanel/WHM

Mass account takeover in Yunmai smartscale by abusing the REST API

CSRF POCs sent to Plesk

About Post Author

Adrian Tiron

Cloud Application Security Consultant/ Pentester

See author's posts

Posted in Research Tagged CORS-vulnerability, csrf, json-csrf

< A CSRF vulnerability in the popular csurf package

Would you like to be updated every time we post something cool on our website?
Please subscribe to our newsletter.

Enter your email SUBSCRIBE NOW

Copyright FORTBRIDGE 2020

Branding and Web development secured by INOVEO