HTB: 2Million Machine Walkthrough - Linux Easy

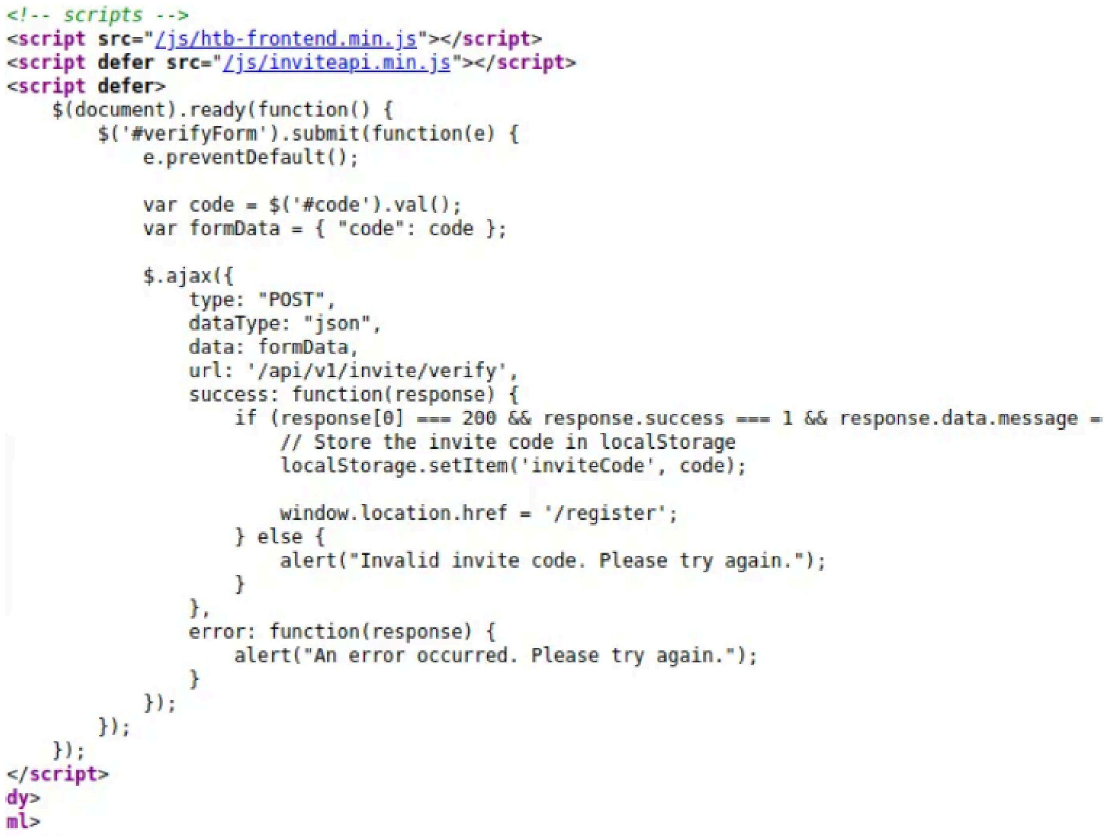
Completed 5/17/2024 through Guided Mode.

Note* Documenting this 6 months after I completed the machine, so this is less of a walkthrough and more of a documentation of the process I went through to get the flags.

```
>> nmap -sC -sV -pN -v 10.10.11.221
```

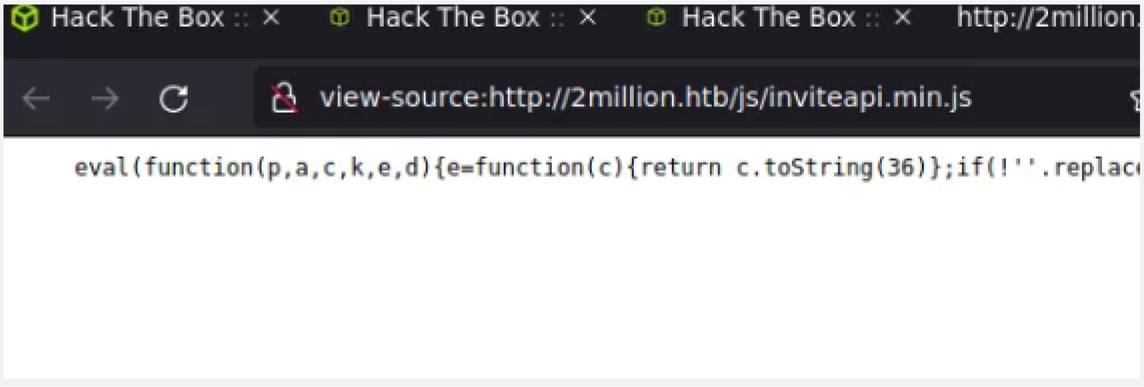
Where there are 2 open ports one running ssh on port 22 and the other running http on port 80. Since we have no ssh user or password, we went over to the website http://10.10.11.221 which redirects you to http://2million.htb/.

First by looking at the source code, we saw that a POST request was being made towards `/api/v1/invite/verify.`



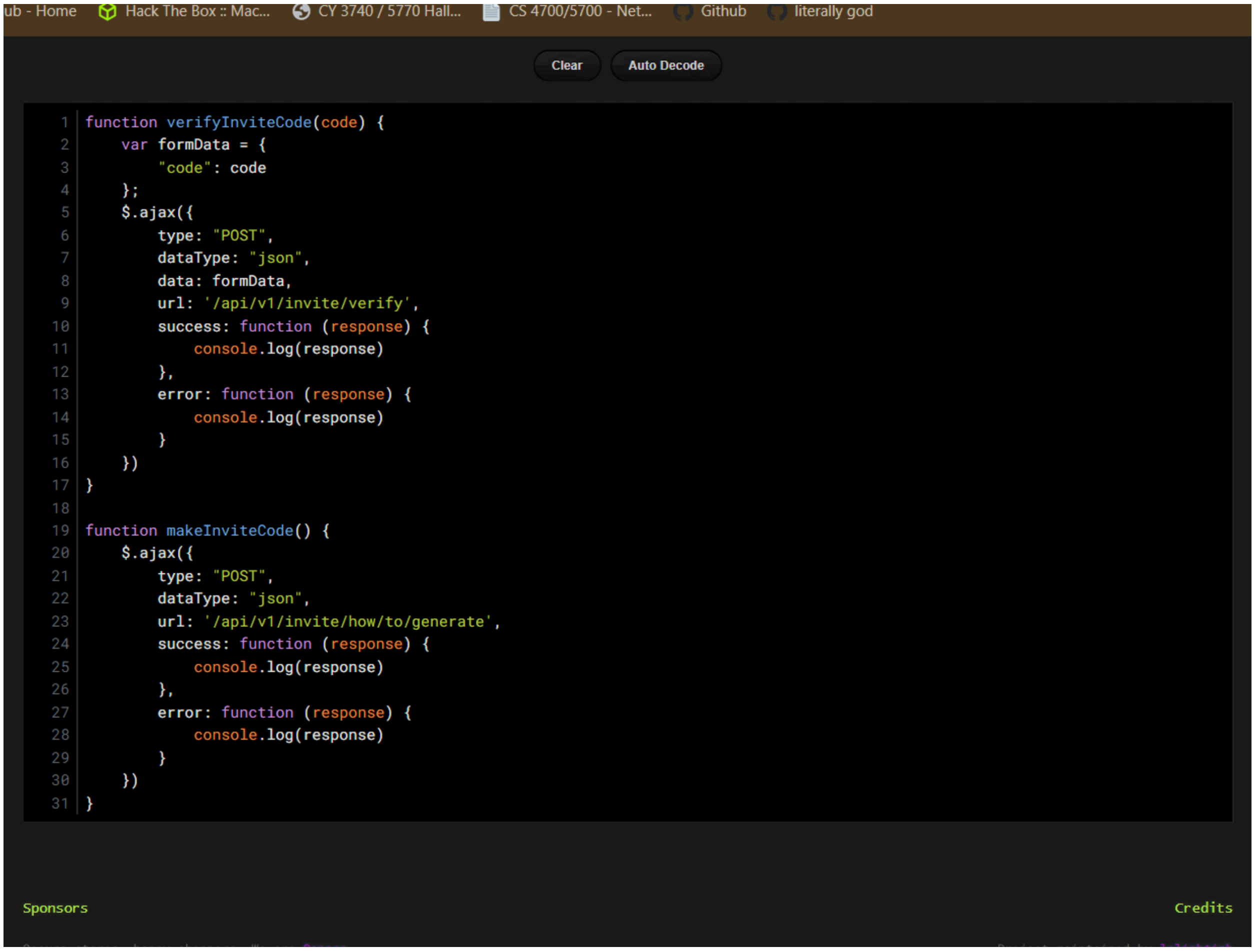
```
<!-- scripts -->
<script src="/js/htb-frontend.min.js"></script>
<script defer src="/js/inviteapi.min.js"></script>
<script defer>
    $(document).ready(function() {
        $('#verifyForm').submit(function(e) {
            e.preventDefault();

            var code = $('#code').val();
            var formData = { "code": code };

            $.ajax({
                type: "POST",
                dataType: "json",
                data: formData,
                url: '/api/v1/invite/verify',
                success: function(response) {
                    if (response[0] === 200 && response.success === 1 && response.data.message =
                        // Store the invite code in localStorage
                        localStorage.setItem('inviteCode', code);

                        window.location.href = '/register';
                    } else {
                        alert("Invalid invite code. Please try again.");
                    }
                },
                error: function(response) {
                    alert("An error occurred. Please try again.");
                }
            });
        });
    });
</script>
dy>
ml>
```

We additionally found a script called `inviteapi.min.js` if you click on the script, it takes you to an obfuscated script.



```
eval(function(p,a,c,k,e,d){e=function(c){return c.toString(36)};if(!''.replac
```

We put this in a deobfuscator and got



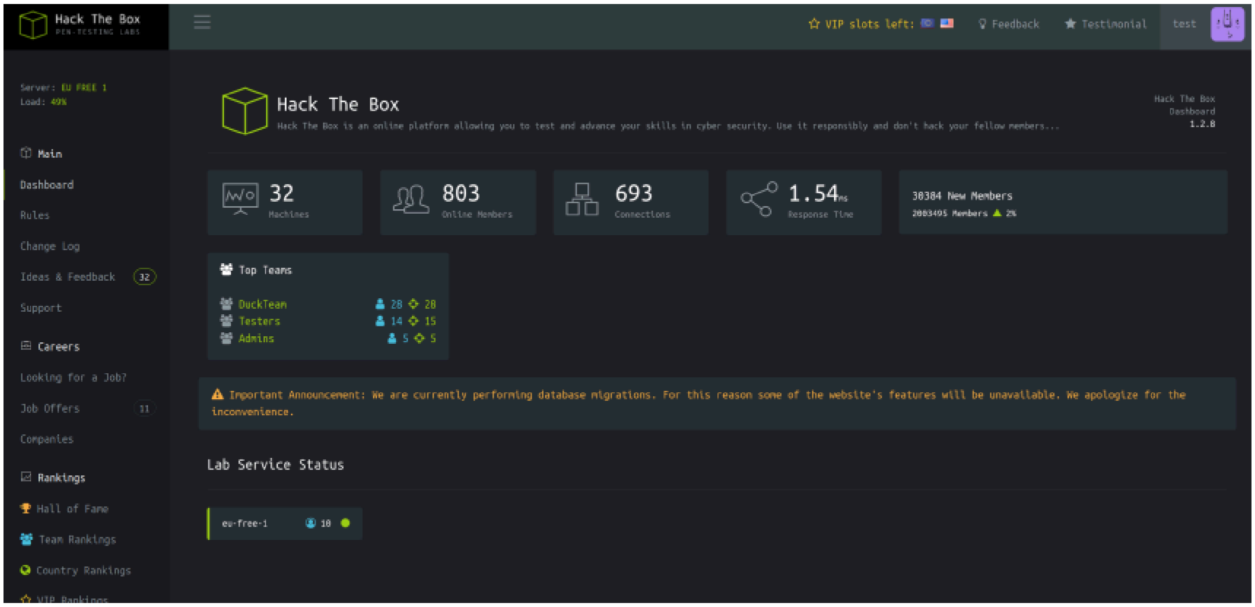
```
function verifyInviteCode(code) {
    var formData = {
        "code": code
    };
    $.ajax({
        type: "POST",
        dataType: "json",
        data: formData,
        url: '/api/v1/invite/verify',
        success: function (response) {
            console.log(response)
        },
        error: function (response) {
            console.log(response)
        }
    })
}

function makeInviteCode() {
    $.ajax({
        type: "POST",
        dataType: "json",
        url: '/api/v1/invite/how/to/generate',
        success: function (response) {
            console.log(response)
        },
        error: function (response) {
            console.log(response)
        }
    })
}
```

From burpsuite, we fuzzed for various api endpoints. In Burpsuite, if you simply sent in a request to api/v1 it would return all the available endpoints. From this we were able to put in `/api/v1/invite/how/to/generate` into the repeater and we saw that we got an encrypted ROT13 cipher, put it into cyberchef and got the hint “In order to generate the invite code, make a POST request to `/api/v1/invite/generate`”

So we edited a post request through our repeater POST `/api/v1/invite/generate HTTP/1.1`. Our response was a base64 encoded string “U1oyN0ktUTVaOVMtREtOSVgtUDNDU00=”

Which we decoded in cyberchef to get the invide code: SZ27I-Q5Z9S-DKNIX-P3CSM

After logging in, we were redirected to home.



Hack The Box is an online platform allowing you to test and advance your skills in cyber security. Use it responsibly and don't hack your fellow members...

Important Announcement: We are currently performing database migrations. For this reason some of the website's features will be unavailable. We apologize for the inconvenience.

In the Access page, we noticed that the button allows a user to Download and Regenerate their VPN file to be able to access the HTB infrastructure in the same way we did to originally access the box. Upon clicking on the button a GET request is sent out to `/api/v1/users/vpn/generate` and in return the VPN file for our current user is downloaded.

We found out that by requesting /api we could see versions of the VPN and furthermore by requesting `/api/v1` through burpsuite. we could enumerate all our endpoints.

We then set a get request to `/api/v1/admin/auth` (one of our enumerated endpoints) and noticed that the admin authentication is a simple JSON data with a message variable.

From here we simply changed the Content-Type of the request and sent it back through the repeater where our response was that we were missing an email parameter. We sent in our email and then our new response was that we were missing the is_admin variable, which we then set to true and sent back to the repeater.

Once we were admin we had to utilize a command injection, to change our guid. We then were able to successfully send in commands such as ‘whoami’ to find the user www-data and then set up a reverse shell listener, and send in a base64 encoded payload to access the machine through www-data.

When we explored the filesystem we noticed that it has a .env file which shows passwords for users. We can use password reuse to gain users. And were able to get admin's password. With this we exited the shell and then ssh through port 22 back into the machine with admin's credentials.

After logging into the machine it notified us that we had mail. So we cd'd /var/mail/admin and this is where we got the hint to utilize the CVE in the box description. So we cloned the CVE into our machine and then ran it as instructed in the README.md (CVE-2023-0386 ./exp) After running it we get root and receive the thankyou.json text that we simply decoded through cyberchef to get our final flag