

2024 Security Assessment
Report Prepared For
securityarchive.net



Report Issued: 03/08/2024

Table of Contents

Confidentiality Notice.....	3
Disclaimer.....	3
EXECUTIVE SUMMARY.....	3
HIGH LEVEL ASSESSMENT OVERVIEW.....	4
Short Term Recommendations.....	4
Long Term Recommendations.....	5
SCOPE.....	6
TESTING METHODOLOGY.....	6
CLASSIFICATION DEFINITIONS.....	8
1 - Cross Site Request Forgery.....	10
2 - Cross Site Scripting.....	15
3 - Rate Limitation.....	18
4 - Weak Password Policy.....	21
5 - User Enumeration.....	23
6 - Weak Security Headers.....	26
APPENDIX A - TOOLS USED.....	28
APPENDIX B - ENGAGEMENT INFORMATION.....	28

Confidentiality Notice

This report contains sensitive, privileged, and confidential information. Precautions should be taken to protect the confidentiality of the information in this document. Publication of this report may cause reputational damage to securityarchive.net or facilitate attacks against securityarchive.net. Security Archive shall not be held liable for special, incidental, collateral or consequential damages arising out of the use of this information.

Disclaimer

Note that this assessment may not disclose all vulnerabilities that are present on the systems within the scope of the engagement. This report is a summary of the findings from a “point-in-time” assessment made on securityarchive.net’s environment. Any changes made to the environment during the period of testing may affect the results of the assessment.

EXECUTIVE SUMMARY

Security Archive performed a security assessment of the internal corporate network of securityarchive.net from 03/05/2024 to 03/08/2024. Security Archive's penetration test simulated an attack from an external threat actor attempting to gain access to systems within the securityarchive.net corporate network. The purpose of this assessment was to discover and identify vulnerabilities in securityarchive.net’s infrastructure and suggest methods to remediate the vulnerabilities. Security Archive identified a total of **6** vulnerabilities within the scope of the engagement which are broken down by severity in the table below.

CRITICAL	High	Medium	Low	Informational
0	2	0	2	2

The highest severity vulnerabilities give potential attackers the opportunity to access, change, and delete user’s stored account information. In order to ensure data confidentiality, integrity, and availability, security remediations should be implemented as described in the security assessment findings.

Note that this assessment may not disclose all vulnerabilities that are present on the systems within the scope. Any changes made to the environment during the period of testing may affect the results of the assessment.

HIGH LEVEL ASSESSMENT OVERVIEW

Areas for Improvement

Security Archive recommends securityarchive.net take the following actions to improve the security of the network. Implementing these recommendations will reduce the likelihood that an attacker will be able to successfully attack securityarchive.net's information systems and/or reduce the impact of a successful attack.

Short Term Recommendations

Security Archive recommends securityarchive.net take the following actions as soon as possible to minimize business risk.

Implementation of CSRF Token

- Due to the lack of CSRF token, multiple actions can be made for users to gain unauthorized access to the admin panel, company pages, and other areas of interest, as well as increases the risk of other found vulnerabilities. This breaks the secure trust between clients and Security Archive.
- We recommend securityarchive.net implement a CSRF token into user session data, that checks whether a user carried out an action, or was redirected.

Account Lockout

- During the testing period, it was noticed that securityarchive.net has no rate limiting, or account lockout in place. These preventative measures being implemented helps limit the chances of successful attacks, such as brute forcing.

Admin input sanitization

- While normal user input seemed to be sanitized and displayed in a secure manner, admin input seemed to have little to no sanitization. On top of this, information displayed in the admin panel displayed stored information with no regulation, which allowed for Cross-Site Scripting to be carried out.

Long Term Recommendations

Security Archive recommends the following actions be taken over the next few months to fix hard-to-remediate issues that do not pose an urgent risk to the business.

Authentication

- It is recommended to have multiple forms of authentication for users. Implementing Multi-Factor Authentication would assist in preventing successful account takeover, even in the case of a data breach.

SCOPE

All testing was based on the scope as defined in the Request For Proposal (RFP) and official written communications. The items in scope are listed below.

Networks

Network	Note
*.securityarchive.net	Security Archive web application

Provided Credentials

securityarchive.net provided Security Archive with the following credentials and access to facilitate the security assessment listed below.

Item	Note
N/A	N/A

TESTING METHODOLOGY

Security Archive's testing methodology was split into three phases: *Reconnaissance*, *Target Assessment*, and *Execution of Vulnerabilities*. During reconnaissance, we gathered information about securityarchive.net's network systems. Security Archive used port scanning and other enumeration methods to refine target information and assess target values. Next, we conducted our targeted assessment. Security Archive simulated an attacker exploiting vulnerabilities in the securityarchive.net network. Security Archive gathered evidence of vulnerabilities during this phase of the engagement while conducting the simulation in a manner that would not disrupt normal business operations.

The following image is a graphical representation of this methodology.



CLASSIFICATION DEFINITIONS

Risk Classifications

Level	Score	Description
Critical	10	The vulnerability poses an immediate threat to the organization. Successful exploitation may permanently affect the organization. Remediation should be immediately performed.
High	7-9	The vulnerability poses an urgent threat to the organization, and remediation should be prioritized.
Medium	4-6	Successful exploitation is possible and may result in notable disruption of business functionality. This vulnerability should be remediated when feasible.
Low	1-3	The vulnerability poses a negligible/minimal threat to the organization. The presence of this vulnerability should be noted and remediated if possible.
Informational	0	These findings have no clear threat to the organization, but may cause business processes to function differently than desired or reveal sensitive information about the company.

Exploitation Likelihood Classifications

Likelihood	Description
Likely	Exploitation methods are well-known and can be performed using publicly available tools. Low-skilled attackers and automated tools could successfully exploit the vulnerability with minimal difficulty.
Possible	Exploitation methods are well-known, may be performed using public tools, but require configuration. Understanding of the underlying system is required for successful exploitation.
Unlikely	Exploitation requires deep understanding of the underlying systems or advanced technical skills. Precise conditions may be required for successful exploitation.

Business Impact Classifications

Impact	Description
Major	Successful exploitation may result in large disruptions of critical business functions across the organization and significant financial damage.
Moderate	Successful exploitation may cause significant disruptions to non-critical business functions.
Minor	Successful exploitation may affect few users, without causing much disruption to routine business functions.

Remediation Difficulty Classifications

Difficulty	Description
Hard	Remediation may require extensive reconfiguration of underlying systems that is time consuming. Remediation may require disruption of normal business functions.
Moderate	Remediation may require minor reconfigurations or additions that may be time-intensive or expensive.
Easy	Remediation can be accomplished in a short amount of time, with little difficulty.

ASSESSMENT FINDINGS

	Number	Finding	Risk Score	Risk
	1	Cross Site Request Forgery	8	High
	2	Cross Site Scripting	7	High
	3	Rate Limitation	3	Low
	4	Weak Password Policy	2	Low
	5	User Enumeration	0	Informational
	6	Missing Security Headers	0	Informational

1 - Cross Site Request Forgery

HIGH RISK (8/10)	
Exploitation Likelihood	Likely
Business Impact	Severe
Remediation Difficulty	Moderate

Security Implications

This vulnerability allows an attacker to change usernames, passwords, assigned companies, and admin status of themselves, and others.

Analysis

In the admin panel, verified administrators have the ability to change a user's username, password, email, company, and admin status. Due to a lack of a CSRF token, an attacker can craft a malicious payload that, if visited by a logged in administrator, would be able to submit this information without the administrator, or the user, knowing. After finding out the user's assigned id, an attacker could craft the following payload:

```
CSRF HTML:
1 <html>
2 <!-- CSRF PoC - generated by Burp Suite Professional -->
3 <body>
4   <form action="https://securityarchive.net/[REDACTED]" method="POST">
5     <input type="hidden" name="id" value="[REDACTED]" />
6     <input type="hidden" name="[REDACTED]" value="[REDACTED]" />
7     <input type="submit" value="[REDACTED]" />
8   </form>
9   <script>
10    history.pushState('', '', '/');
11    document.forms[0].submit();
12  </script>
13 </body>
14 </html>
15
```

Figure 1.1: CSRF payload making user “attacker” an admin

Before the payload is ran, you see the following photos, showing that “attacker” is not an administrator:

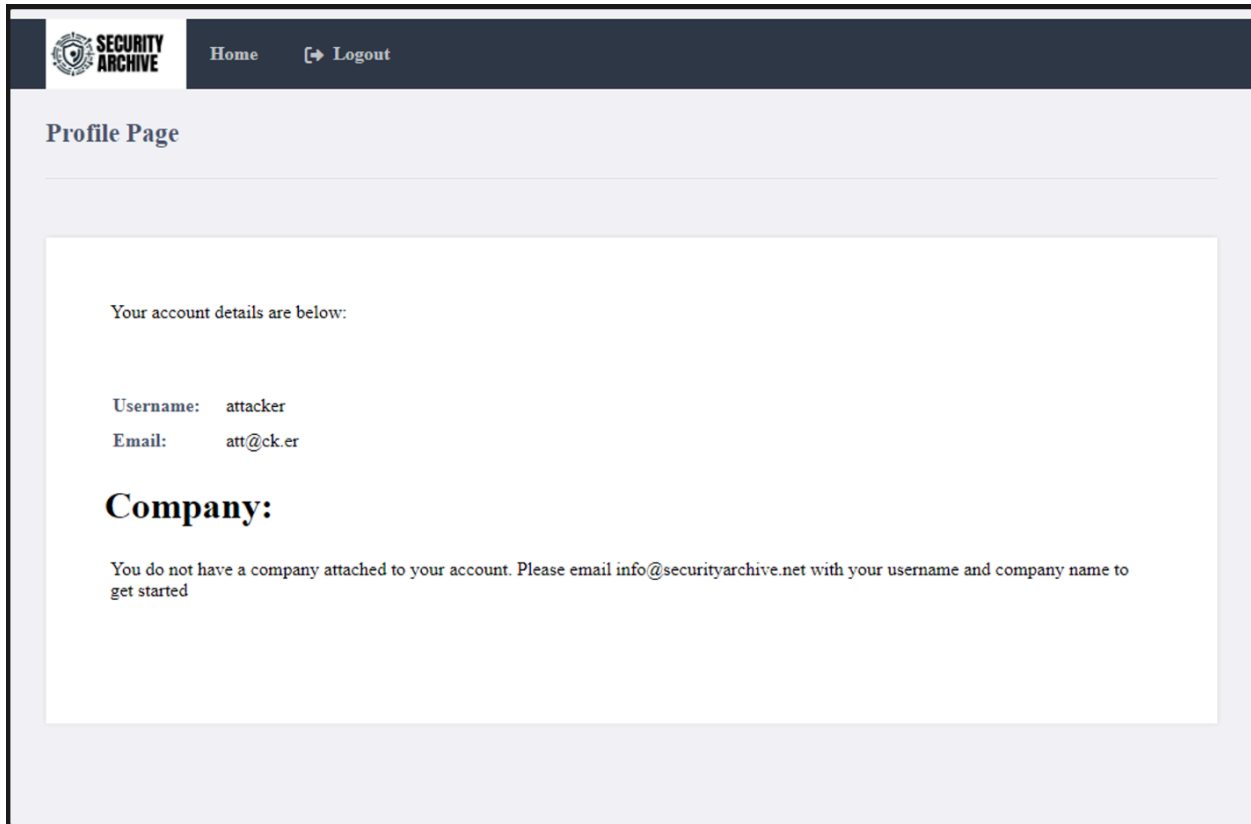


Figure 1.2: Non admin attacker page

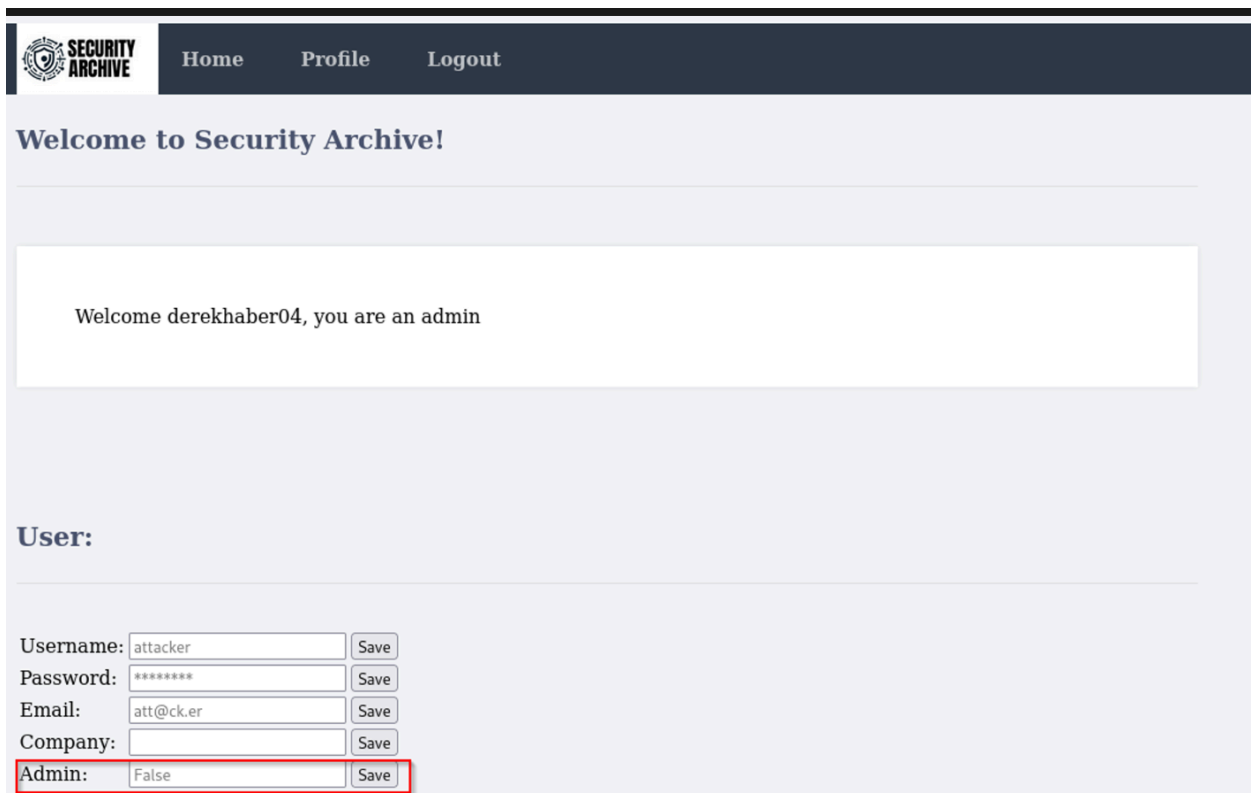
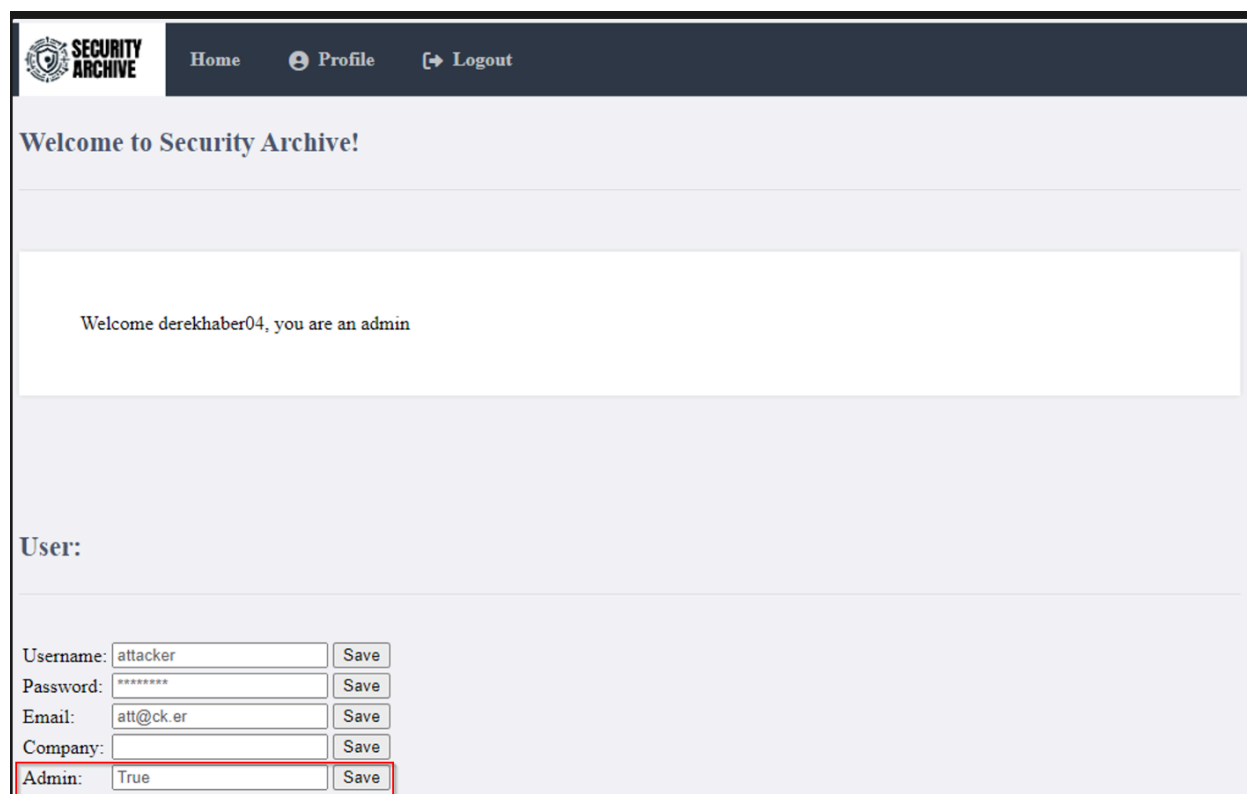


Figure 1.3: User edit page showing attacker isn't an admin

Now, running the malicious CSRF payload as an HTML webpage, the admin is redirected to the admin panel home screen. When looking at the attacker's account, and the manage user page, you can see that the attacker is now an administrator:



SECURITY ARCHIVE Home Profile Logout

Welcome to Security Archive!

Welcome derekhaber04, you are an admin

User:

Username:	<input type="text" value="attacker"/>	Save
Password:	<input type="password" value="*****"/>	Save
Email:	<input type="text" value="att@ck.er"/>	Save
Company:	<input type="text"/>	Save
Admin:	<input type="text" value="True"/>	Save

Figure 1.4: Manage user page showing “attacker” as an admin

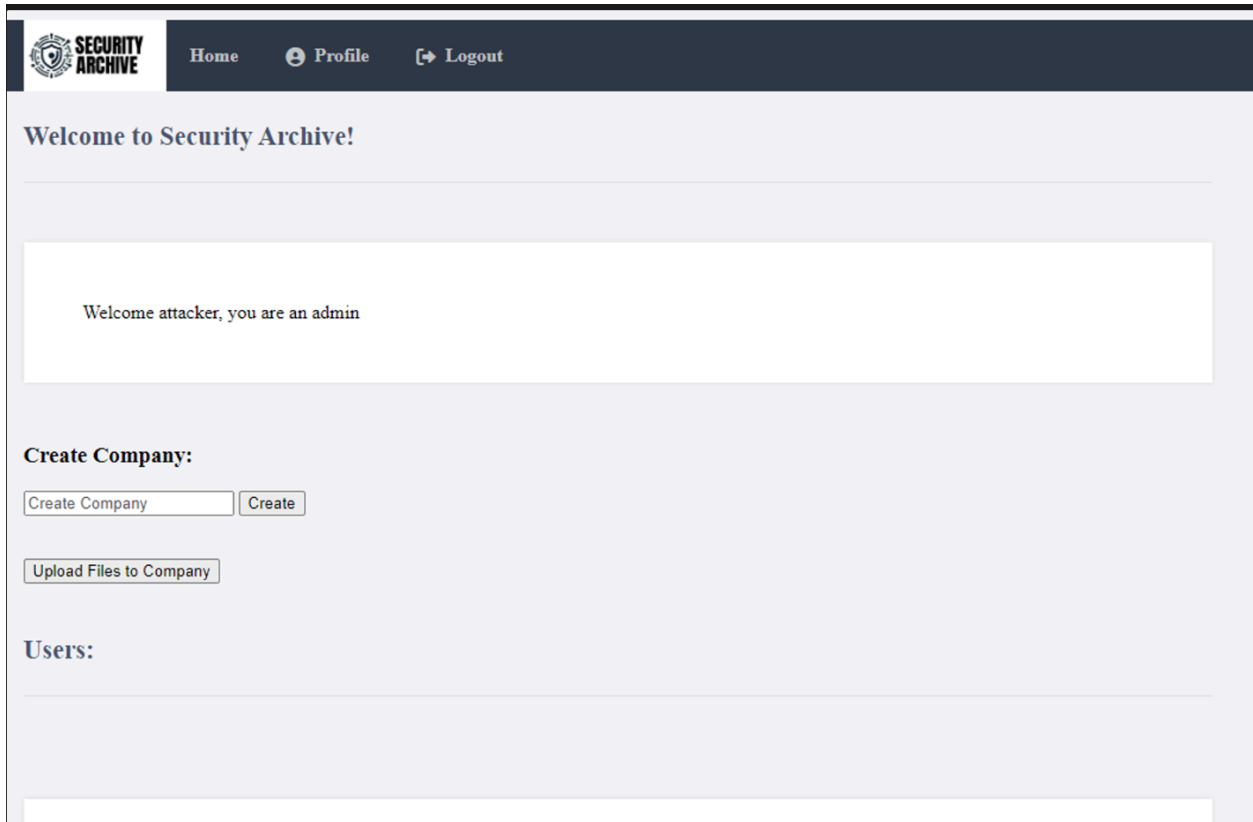


Figure 1.5: “attacker” logged in on admin page

Recommendations

To implement a random CSRF token, you need to add both a session for the token, as well as a hidden input for submitting the token. Since Security Archive is a LAMP project, we can do this in PHP.

In the beginning, add a session labeled ‘token’ with a random value. This can be set as a random md5 value using the following line:

```
$_SESSION['token'] = md5(uniqid(mt_rand(), true));
```

In any forms contained in the web app, add a hidden input containing the token:

```
<input type="hidden" name="token" value="<?php echo $_SESSION['token'] ?? ' ' ?>">
```

In the PHP file that processes the form, add the following block of code:

```
$token = filter_input(INPUT_POST, 'token', FILTER_SANITIZE_STRING);

if (!$token || $token !== $_SESSION['token']) {
    // return 405 http status code
    header($_SERVER['SERVER_PROTOCOL'] . ' 405 Method Not Allowed');
    exit;
} else {
    // process the form
}
```

2 - Cross Site Scripting

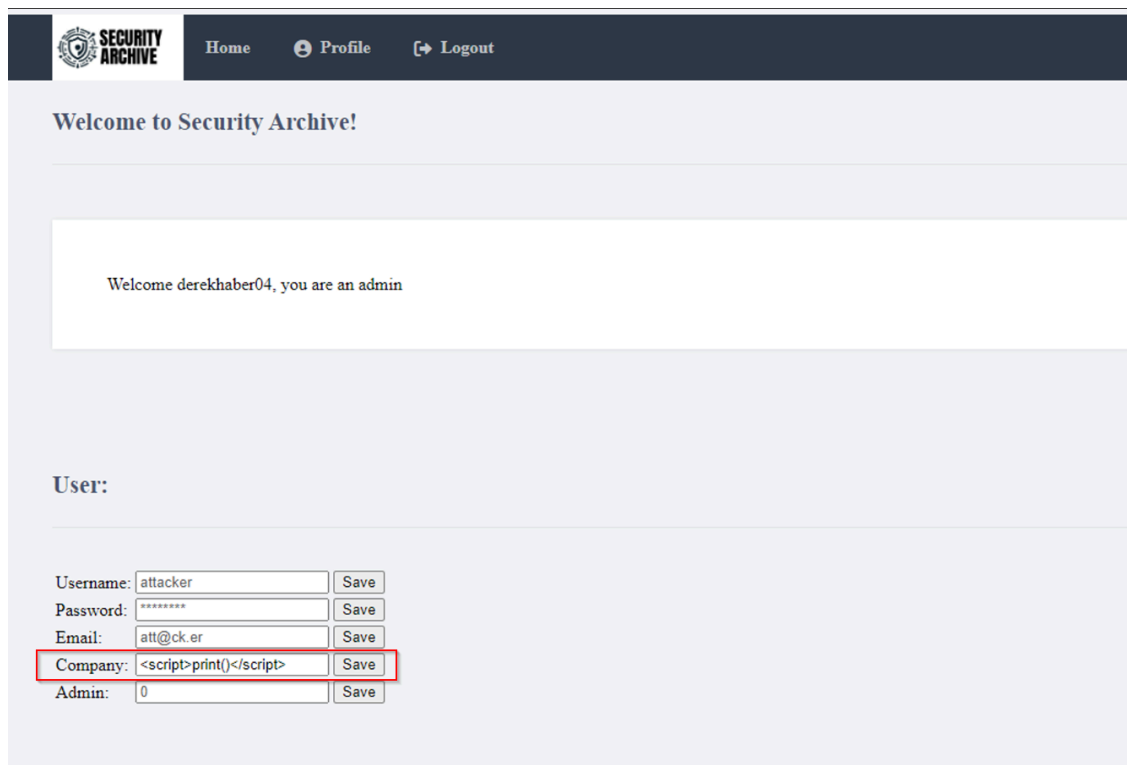
HIGH RISK (7/10)	
Exploitation Likelihood	Likely
Business Impact	Severe
Remediation Difficulty	Easy

Security Implications

Exploitation of cross site scripting (XSS) can be very dangerous, as it can run javascript in a user's browser. This can be used for stealing user cookies, running a javascript keylogger, and anything else that can be crafted in javascript.

Analysis

User input displayed throughout securityarchive.net is generally sanitized, and secured from being used maliciously for normal users, however, the admin panel showed to have weak filtering. This is demonstrated below using the set company function, running a print() function on the admin page.



SECURITY ARCHIVE

Home Profile Logout

Welcome to Security Archive!

Welcome derekhaber04, you are an admin

User:

Username: Save

Password: Save

Email: Save

Company: Save

Admin: Save

Figure 2.1: Setting XSS payload as attacker's company

Viewing the attacker profile page, this is filtered, and didn't run:

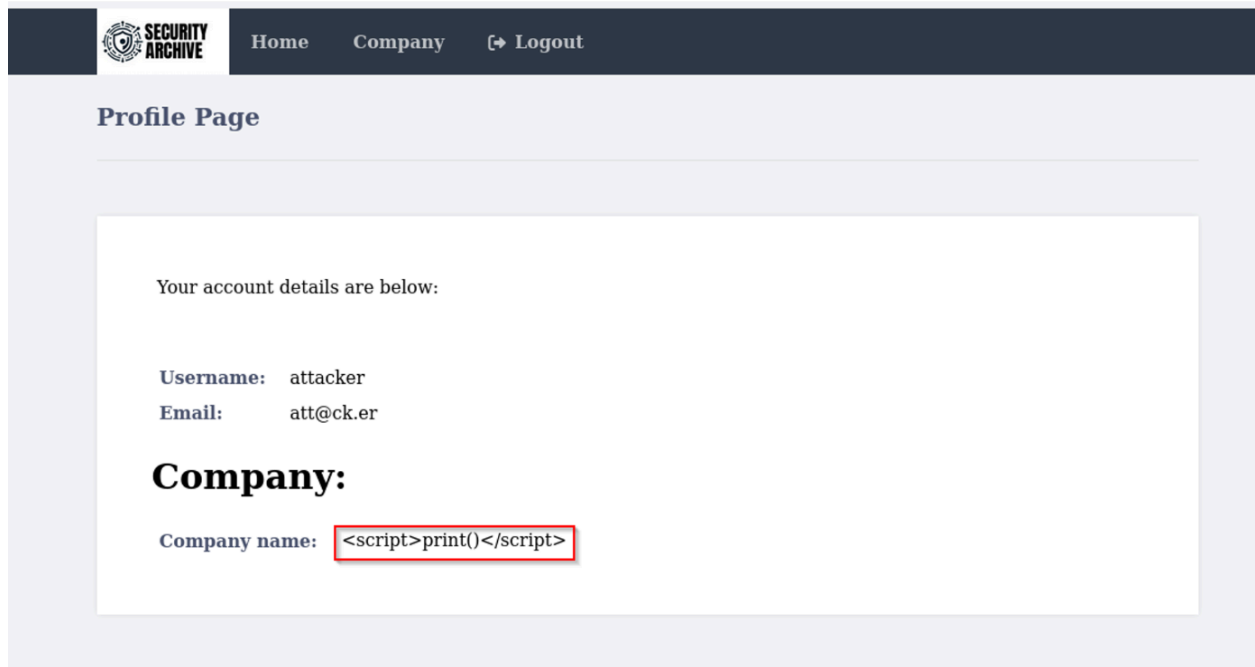


Figure 2.2: XSS payload not running on profile page

However, on the admin page, the XSS payload ran, which caused the print screen to pop up:

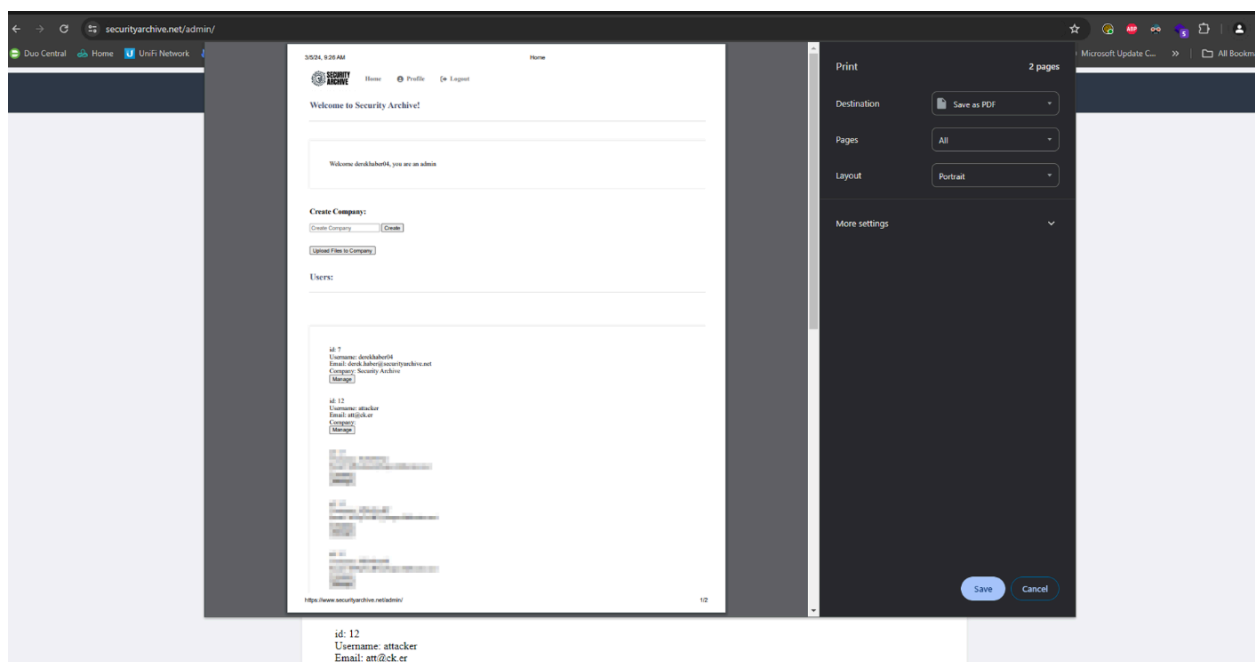


Figure 2.3: Print screen popping up due to XSS payload

Source code:

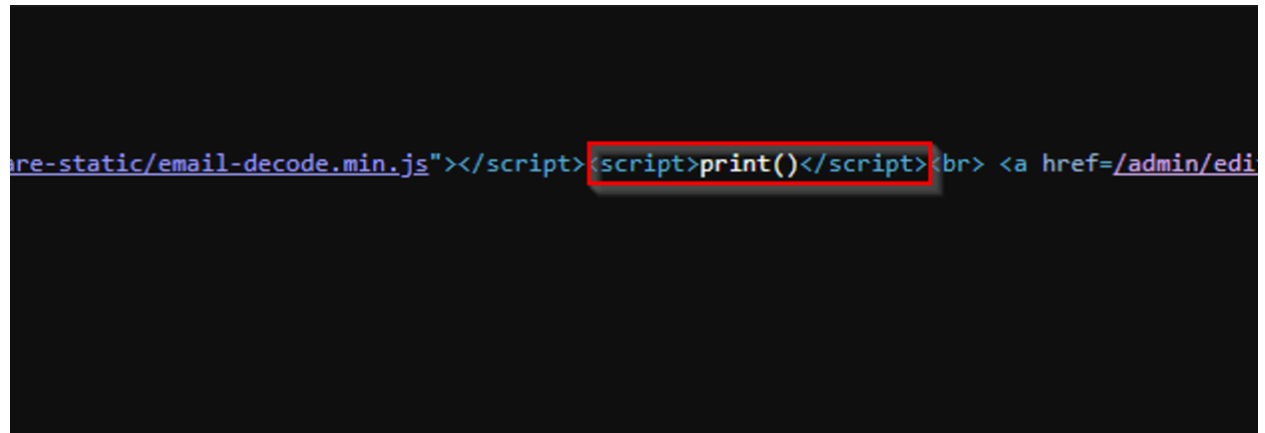
A screenshot of a dark-themed code editor showing a line of HTML source code. The code is: `are-static/email-decode.min.js"></script><script>print()</script>
 <a href=/admin/edi`. The `<script>print()</script>` portion is highlighted with a red rectangular box.

Figure 2.4: HTML source code showing the script tag

Recommendations

Similarly to how the profile page information is processed, the admin panel should HTML encode the responses. In PHP, there is an easy way to encode user inputs using the `htmlspecialchars` function:

```
htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
```

Another thing that can be done to help prevent the attempt at attacks would be to implement rules preventing the input of banned characters (such as `<>#|`- etc). This would help prevent bad characters from being saved in the first place.

3 - Rate Limitation

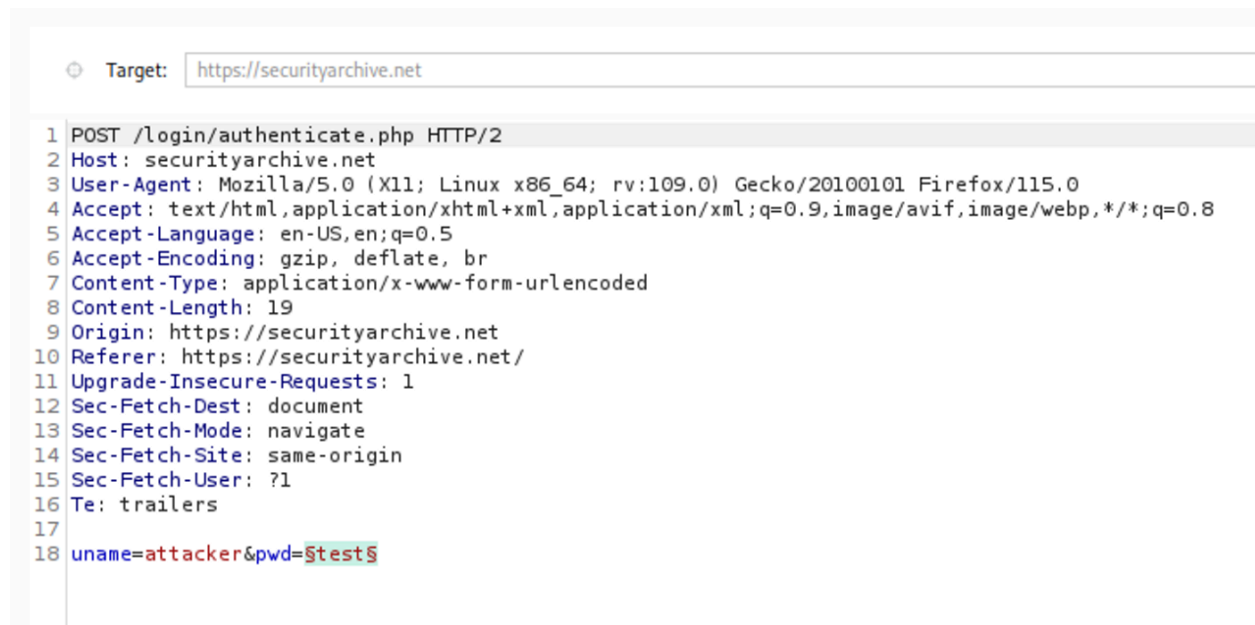
LOW RISK (3/10)	
Exploitation Likelihood	Likely
Business Impact	Low
Remediation Difficulty	Easy

Security Implications

Having no rate limiting rules, your web app is opened up to the risk of password brute forcing, brute forcing of payloads, username enumeration, and possibly overloading the server.

Analysis

There are no limitations to how many requests can be made to securityarchive.net. This allows an attacker to brute force user logins. The below example attempted just below 3,500 attempts in less than 30 seconds, ending in the attacker's password being found at the very end.



```
1 POST /login/authenticate.php HTTP/2
2 Host: securityarchive.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 19
9 Origin: https://securityarchive.net
10 Referer: https://securityarchive.net/
11 Upgrade-Insecure-Requests: 1
12 Sec-Fetch-Dest: document
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-User: ?1
16 Te: trailers
17
18 uname=attacker&pwd=5test$
```

Figure 3.1: Brute force payload

6. Intruder attack of https://securityarchive.net

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment
3425	attacker01	302			802	
0		200			800	
1	!@#%\$%	200			800	
2	!@#%\$%^	200			808	
3	!@#%\$%^&	200			802	
4	!@#%\$%^&*	200			800	
5	!root	200			800	
6	\$SRV	200			802	
7	\$secure\$	200			800	
8	*3noguru	200			796	
9	@#%\$%^&	200			798	
10	A.M.I	200			798	
11	ARC123	200			800	

Request Response

Pretty Raw Hex Render

```

8 Set-Cookie: PHPSESSID=j9dmq2crtutv15rfgsdpemmtvq; path=/
9 Location: /account/profile.php
10 Cf-Cache-Status: DYNAMIC
11 Report-To:
  {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=nH7aySJnZ9cNmc43Y4sUeiCQEeoh0mpZhjeHoTKXqHJuc4GweoF5pHMBI%2BUZSbB7NptHVS1AdYU3yMSf3AZVVcsHwB%2B%2F"}], "group": "cf-nel", "max_age": 604800}
12 Nel: {"success_fraction": 0, "report_to": "cf-nel", "max_age": 604800}
13 Server: cloudflare
14 Cf-Ray: 85fc4168ffa5e599-DFW
15 Alt-Svc: h3=":443"; ma=86400
16
17 <br>
18 <br>
19 <a href="/">
  Return to home
20 </a>

```

Figure 3.2: Successful brute force attempt

This can also be used to speed up enumeration tactics, such as username enumeration.

Recommendations

Implemented account lockout policy can prevent successfully brute forcing login attempts. Add a column to the SQL database containing an integer. This can be done in MySQL by running ALTER TABLE accounts ADD attempt INT NOT NULL DEFAULT 0;

In authenticate.php, when the web app responds to an incorrect login request, add a line that runs the following: UPDATE accounts SET attempt = attempt + 1 WHERE username = \$user;

Finally, in the authentication PHP, add a statement checking if “attempt” is 3 or less:

```

if ($attempt > 3){
Echo "Account is locked. Please contact info@securityarchive.net to get it unblocked.
} else {
// Authentication
}

```

As for rate limitation, there are multiple libraries that could be used. One in PHP is REDIS. Below is an example of what code could be added to the web app to prevent over usage:

```
<?php
$redis = new Redis();
$redis->connect('127.0.0.1', 6379);
$redis->auth(' REDIS_PASSWORD ');

$max_calls_limit = 3;
$time_period      = 10;
$total_user_calls = 0;

if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
    $user_ip_address = $_SERVER['HTTP_CLIENT_IP'];
} elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
    $user_ip_address = $_SERVER['HTTP_X_FORWARDED_FOR'];
} else {
    $user_ip_address = $_SERVER['REMOTE_ADDR'];
}

if (!$redis->exists($user_ip_address)) {
    $redis->set($user_ip_address, 1);
    $redis->expire($user_ip_address, $time_period);
    $total_user_calls = 1;
} else {
    $redis->INCR($user_ip_address);
    $total_user_calls = $redis->get($user_ip_address);
    if ($total_user_calls > $max_calls_limit) {
        echo "User " . $user_ip_address . " limit exceeded.";
        exit();
    }
}

echo "Welcome " . $user_ip_address . " total calls made " . $total_user_calls . " i
```

4 - Weak Password Policy

HIGH RISK (2/10)	
Exploitation Likelihood	Possible
Business Impact	Low
Remediation Difficulty	Easy

Security Implications

Weak password policies put users at risk, as it allows easy to guess, insecure passwords to be set during registration, that can make attacks like a brute force more likely to succeed. A good practice is to assume user's will only be as secure as you force them to be.

Analysis

When registering for securityarchive.net, there are some password policies in place. The web app forces it's user to set a password with a minimum of 8 characters:

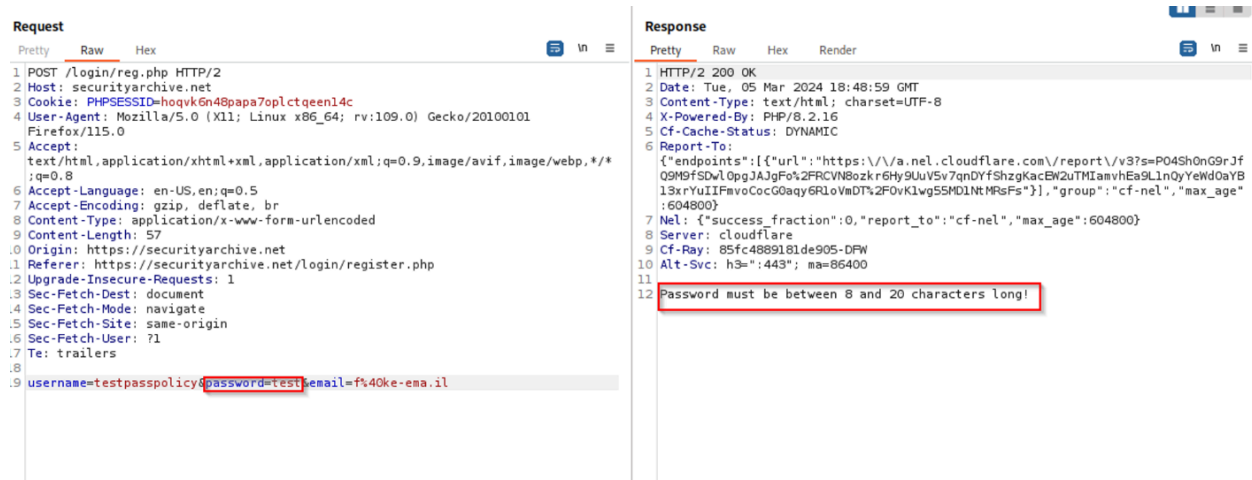


Figure 4.1: Request showing short password from being used

While this was the start to a good password policy, securityarchive.net had no other measures in place to prevent insecure passwords. Below the password "testtest" is used without and issue:

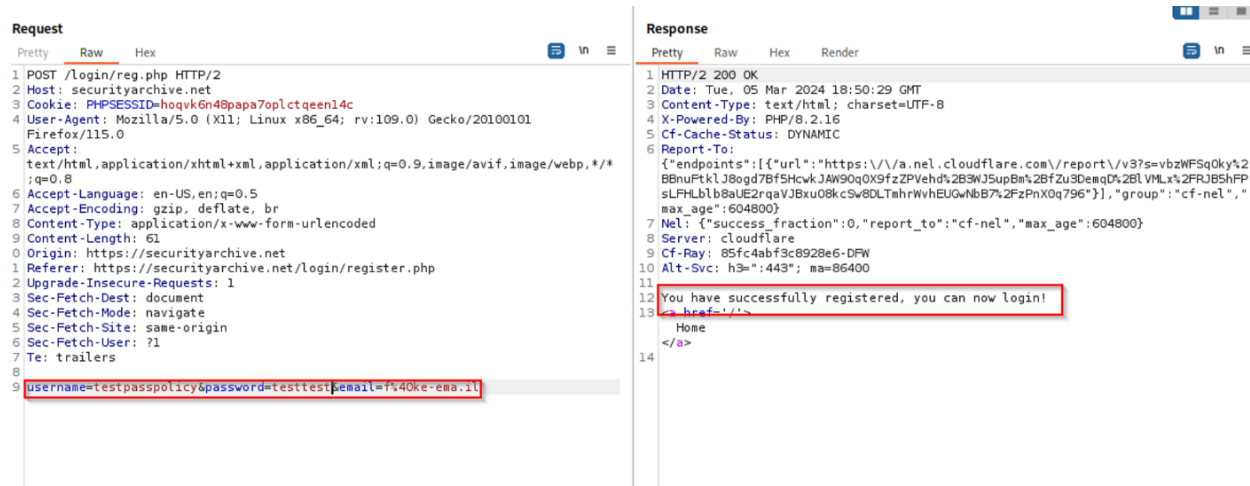


Figure 4.2: Weak password over 8 characters

Recommendations

When registering, before inputting data into the SQL database, check information included in the password field using code similar to the following:

```
$password = $_POST['pwd'];
```

```
// Validate password strength
```

```
$uppercase = preg_match('@[A-Z]@', $password);
```

```
$lowercase = preg_match('@[a-z]@', $password);
```

```
$number = preg_match('@[0-9]@', $password);
```

```
$specialChars = preg_match('@^[^w]@', $password);
```

```
if(!$uppercase || !$lowercase || !$number || !$specialChars || strlen($password) < 8) {
```

```
    echo 'Password should be at least 8 characters in length and should include at least one
    upper case letter, one number, and one special character.';
```

```
}else{
```

```
// Registration script here
```

```
}
```

5 - User Enumeration

INFORMATIONAL (7/10)	
Exploitation Likelihood	N/A
Business Impact	N/A
Remediation Difficulty	Easy

Security Implications

While having methods available for an attacker to enumerate usernames isn't necessarily dangerous, it allows for an attacker to better prepare an attack. For example, if John Doe's account at example.com got leaked, and his account used an email and password to authenticate, an attacker could see this, and try to brute force, and enumerate what usernames John Doe could be using, which allows the attacker to try different variations of the other leaked password.

Analysis

When registering a new account, the script prevents a new user from using an existing username. This is an important feature to have, however, its response gave a bit too much away:

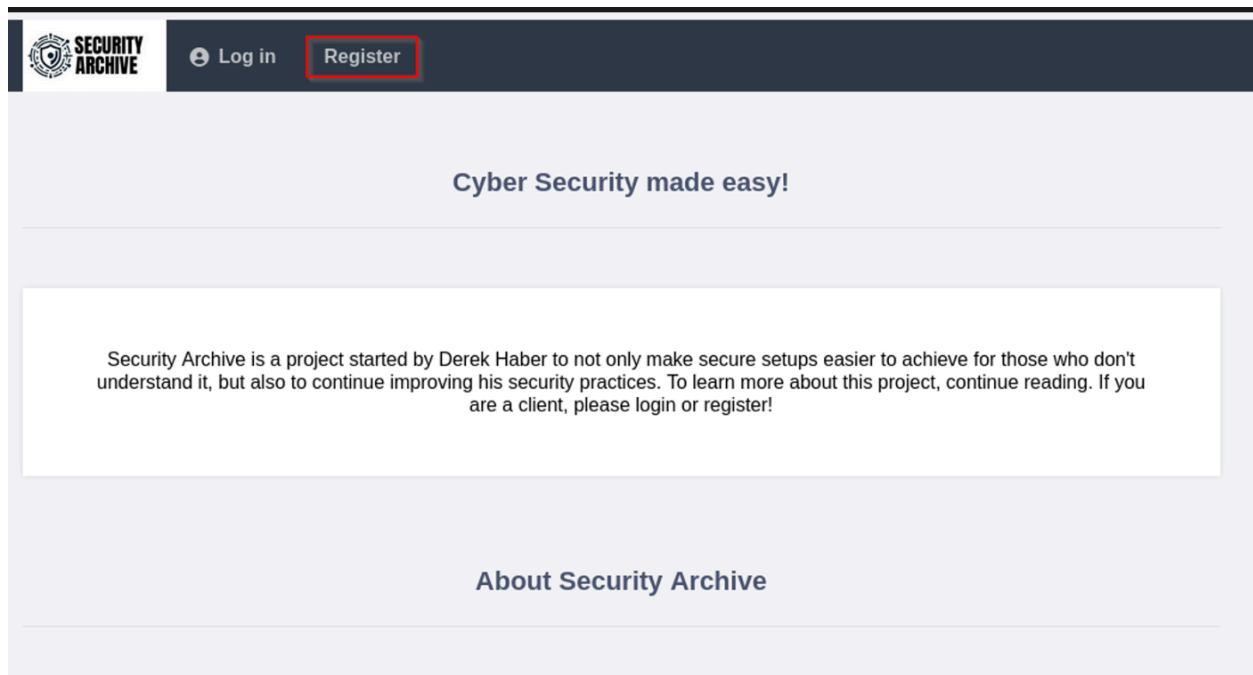


Figure 5.1: Home page showing “Register” option

Register

Enumerating the "attacker" username.
 attacker

.....

test@test.te

Register

Figure 5.2: Entering information to enumerate

Request	Response
<pre> 1 POST /login/reg.php HTTP/2 2 Host: securityarchive.net 3 Cookie: PHPSESSID=hoqvk6n48papa7oplctqueen14c 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/* ;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate, br 8 Content-Type: application/x-www-form-urlencoded 9 Content-Length: 56 10 Origin: https://securityarchive.net 11 Referer: https://securityarchive.net/login/register.php 12 Upgrade-Insecure-Requests: 1 13 Sec-Fetch-Dest: document 14 Sec-Fetch-Mode: navigate 15 Sec-Fetch-Site: same-origin 16 Sec-Fetch-Site: 71 17 Te: trailers 18 19 username=attacker&password=testtest&email=test%40test.te </pre>	<pre> 1 HTTP/2 200 OK 2 Date: Tue, 05 Mar 2024 18:56:14 GMT 3 Content-Type: text/html; charset=UTF-8 4 X-Powered-By: PHP/8.2.16 5 Cf-Cache-Status: DYNAMIC 6 Report-To: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=Pc04B958cGyb pz5yQDvE%2Fn1dxjUbZh6KpB%2FeDyJdKgfm3%2Ba9b66IEiE0KkpFy6czsUvlrNeMEjz26fBT0ladw KlnEMLWSGnEtzvSF80QxNeb2CSh3fN5ML4KJbXMOJBMLGWNsfy"}],"group":"cf-nel","max_ag e":604800} 7 Nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800} 8 Server: cloudflare 9 Cf-Ray: 85fc532a1e2ae836-DFW 10 Alt-Svc: h3=":443"; ma=B6400 11 12 Username exists, please choose another! 13 Home 14 </pre>

Figure 5.3: Request showing that the username exists

Recommendations

It is recommended to change the response message to something more generic, such as "Invalid username".

On top of this, implementing a rate limitation feature can prevent the speed at which this could be carried out, greatly limiting the success rate.

6 - Weak Security Headers

INFORMATIONAL (0/10)	
Exploitation Likelihood	N/A
Business Impact	N/A
Remediation Difficulty	N/A


Security Implications

Security headers communicate with the browser to aid in protecting the user from certain attacks, such as cross-site scripting, code injection, and clickjacking.

Analysis

When checking securityarchive.com's web security headers using <https://securityheaders.com>, we noticed that it scored an F overall

Security Report Summary



Site: <https://securityarchive.net/>

IP Address: 2606:4700:3036::6815:3c44

Report Time: 05 Mar 2024 18:35:06 UTC

Headers: ✖ Strict-Transport-Security ✖ Content-Security-Policy ✖ X-Frame-Options ✖ X-Content-Type-Options ✖ Referrer-Policy ✖ Permissions-Policy

Advanced: Ouch, you should work on your security posture immediately: [Start Now](#)

Missing Headers

Strict-Transport-Security	HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS. Recommended value "Strict-Transport-Security: max-age=31536000; includeSubDomains".
Content-Security-Policy	Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets.
X-Frame-Options	X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking. Recommended value "X-Frame-Options: SAMEORIGIN".
X-Content-Type-Options	X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
Referrer-Policy	Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.
Permissions-Policy	Permissions Policy is a new header that allows a site to control which features and APIs can be used in the browser.

Figure 6.1: Missing security headers

Recommendations

It is recommended to go through and add all these security headers. This is done on the web server. Since this is using apache, you can edit your `/etc/httpd/conf/httpd.conf` file, and add the following to the end of the file:

Header always set X-XSS-Protection: "1; mode=block"

Header always set X-Content-Type-Options: "nosniff"

Header always set X-Frame-Options: "SAMEORIGIN"

Header always set Referrer-Policy: "strict-origin-when-cross-origin"

Header always set Strict-Transport-Security: "max-age=31536000; includeSubDomains; preload"

You can then re-check, and add any other headers they recommend.

APPENDIX A - TOOLS USED

TOOL	DESCRIPTION
BurpSuite Pro Edition	Used for testing of web applications.
Nmap	Used for scanning ports on hosts.
securityheaders.com	Used to check the website's security headers.

Table A.1: Tools used during assessment

APPENDIX B - ENGAGEMENT INFORMATION

Client Information

Client	Security Archive
Primary Contact	Derek Haber, derek.haber@securityarchive.net
Approvers	The following people are authorized to change the scope of engagement and modify the terms of the engagement <ul style="list-style-type: none">Derek Haber

Version Information

Version	Date	Description
1.0	03/08/2024	Initial report to client

Contact Information

Name	Security Archive Consulting
Phone	
Email	derek.haber@securityarchive.net