



Acunetix Threat Level 4

One or more critical-severity type vulnerabilities have been discovered by the scanner. A malicious user can exploit these vulnerabilities and compromise the backend database and/or deface your website.

Scan Detail

Target	http://192.168.1.18:80
Scan Type	Full Scan
Start Time	Nov 6, 2025, 7:57:58 AM GMT
Scan Duration	56 seconds
Requests	5446
Average Response Time	3ms
Maximum Response Time	2652ms
Application Build	v24.6.240626115
Authentication Profile	-

2**2****3****2****3**

Critical

High

Medium

Low

Informational

Severity	Vulnerabilities	Instances
⚠ Critical	1	2
⚠ High	1	2
⚠ Medium	3	3
✗ Low	2	2
ⓘ Informational	3	3
Total	10	12

Critical Severity



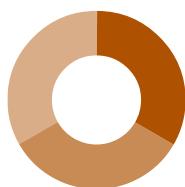
	Instances
SQL Injection	2

High Severity



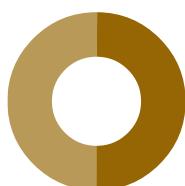
	Instances
Cross-site Scripting	2

Medium Severity



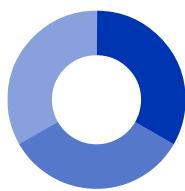
	Instances
Insecure HTTP Usage	1
Password transmitted over HTTP	1
SSL/TLS Not Implemented	1

Low Severity



	Instances
Apache mod_negotiation filename brutefo...	1
Programming Error Messages	1

Informational



	Instances
Content Security Policy (CSP) Not Implemented	1
Error page web server version disclosure	1
Permissions-Policy header not implemented	1

Impacts

SEVERITY	IMPACT
 Critical	 SQL Injection
 High	 Cross-site Scripting
 Medium	 Insecure HTTP Usage
 Medium	 Password transmitted over HTTP
 Medium	 SSL/TLS Not Implemented
 Low	 Apache mod_negotiation filename bruteforcing
 Low	 Programming Error Messages
 Informational	 Content Security Policy (CSP) Not Implemented
 Informational	 Error page web server version disclosure
 Informational	 Permissions-Policy header not implemented

SQL Injection

SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.

Impact

An attacker can use SQL injection to bypass a web application's authentication and authorization mechanisms and retrieve the contents of an entire database. SQLi can also be used to add, modify and delete records in a database, affecting data integrity. Under the right circumstances, SQLi can also be used by an attacker to execute OS commands, which may then be used to escalate an attack even further.

<http://192.168.1.18/login.php>

URL encoded POST input `email` was set to `testing@example.com'"`

Error message found:

You have an error in your SQL syntax

Request

```
POST /login.php HTTP/1.1
Referer: http://192.168.1.18/
Content-Type: application/x-www-form-urlencoded
Content-Length: 55
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive

email=testing%40example.com' "&password=u]H[ww6KrA9Fx-F"
```

<http://192.168.1.18/login.php>

URL encoded POST input `password` was set to `u]H[ww6KrA9Fx-F"`

Error message found:

You have an error in your SQL syntax

Request

```
POST /login.php HTTP/1.1
Referer: http://192.168.1.18/
Content-Type: application/x-www-form-urlencoded
Content-Length: 55
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive

email=testing%40example.com&password=u]H[ww6KrA9F.x-F'"
```

Recommendation

Use parameterized queries when dealing with SQL queries that contain user input. Parameterized queries allow the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.

References

[SQL Injection \(SQLi\) - Acunetix](https://www.acunetix.com/websitemanagement/sql-injection/)

<https://www.acunetix.com/websitemanagement/sql-injection/>

[Types of SQL Injection \(SQLi\) - Acunetix](https://www.acunetix.com/websitemanagement/sql-injection2/)

<https://www.acunetix.com/websitemanagement/sql-injection2/>

[Prevent SQL injection vulnerabilities in PHP applications and fix them - Acunetix](https://www.acunetix.com/blog/articles/prevent-sql-injection-vulnerabilities-in-php-applications/)

<https://www.acunetix.com/blog/articles/prevent-sql-injection-vulnerabilities-in-php-applications/>

[SQL Injection - OWASP](https://www.owasp.org/index.php/SQL_Injection)

https://www.owasp.org/index.php/SQL_Injection

[Bobby Tables: A guide to preventing SQL injection](https://bobby-tables.com/)

<https://bobby-tables.com/>

[SQL Injection Cheat Sheets - Pentestmonkey](http://pentestmonkey.net/category/cheat-sheet/sql-injection)

<http://pentestmonkey.net/category/cheat-sheet/sql-injection>

Cross-site Scripting

Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

Impact

Malicious JavaScript has access to all the same objects as the rest of the web page, including access to cookies and local storage, which are often used to store session tokens. If an attacker can obtain a user's session cookie, they can then impersonate that user.

Furthermore, JavaScript can read and make arbitrary modifications to the contents of a page being displayed to a user. Therefore, XSS in conjunction with some clever social engineering opens up a lot of possibilities for an attacker.

<http://192.168.1.18/login.php>

Verified

URL encoded POST input **email** was set to **testing@example.com'"()&%<zzz><ScRiPt >cnRv(9817)</ScRiPt>**

Request

```
POST /login.php HTTP/1.1
Referer: http://192.168.1.18/
Content-Type: application/x-www-form-urlencoded
Content-Length: 98
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

email=testing%40example.com'"()&%<zzz><ScRiPt %20>cnRv(9817)</ScRiPt>&password=u]H[ww6KrA9F.x-F

<http://192.168.1.18/login.php>

Verified

URL encoded POST input **password** was set to **u]H[ww6KrA9F.x-F'"()&%<zzz><ScRiPt >cnRv(9620)</ScRiPt>**

Request

```
POST /login.php HTTP/1.1
Referer: http://192.168.1.18/
Content-Type: application/x-www-form-urlencoded
Content-Length: 98
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

email=testing%40example.com&password=u]H[ww6KrA9F.x-F'"()&%<zzz><ScRiPt %20>cnRv(9620)</ScRiPt>

Recommendation

References

Cross-site Scripting (XSS) Attack - Acunetix

<https://www.acunetix.com/websitemanagement/cross-site-scripting/>

Types of XSS - Acunetix

<https://www.acunetix.com/websitemanagement/xss/>

XSS Filter Evasion Cheat Sheet

https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

Excess XSS, a comprehensive tutorial on cross-site scripting

<https://excess-xss.com/>

Cross site scripting

https://en.wikipedia.org/wiki/Cross-site_scripting

Insecure HTTP Usage

It was detected that your web application uses HTTP protocol, but doesn't automatically redirect users to HTTPS.

Impact

In some circumstances, it could be used for a man-in-the-middle (MitM) attack

<http://192.168.1.18/>

Request

```
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

Recommendation

It's recommended to implement best practices of HTTP Redirection into your web application.
Consult web references for more information

References

HTTP Redirections

https://infosec.mozilla.org/guidelines/web_security#http-redirections

Password transmitted over HTTP

User credentials are transmitted over an unencrypted channel. This information should always be transferred via an encrypted channel (HTTPS) to avoid being intercepted by malicious users.

Impact

A third party may be able to read the user credentials by intercepting an unencrypted HTTP connection.

http://192.168.1.18/

Forms with credentials sent in clear text:

- http://192.168.1.18/

```
Form name: <empty>
Form action: login.php
Form method: POST
Password input: password
```

- http://192.168.1.18/index

```
Form name: <empty>
Form action: login.php
Form method: POST
Password input: password
```

- http://192.168.1.18/index.html

```
Form name: <empty>
Form action: login.php
Form method: POST
Password input: password
```

Request

```
GET / HTTP/1.1
Referer: http://192.168.1.18/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

Recommendation

Because user credentials are considered sensitive information, should always be transferred to the server over an encrypted connection (HTTPS).

SSL/TLS Not Implemented

This scan target was connected to over an unencrypted connection. A potential attacker can intercept and modify data sent and received from this site.

Impact

Possible information disclosure.

<http://192.168.1.18/>

Verified

Request

```
GET / HTTP/1.1
Referer: http://192.168.1.18/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

Recommendation

The site should send and receive data over a secure (HTTPS) connection.

Apache mod_negotiation filename bruteforcing

mod_negotiation is an Apache module responsible for selecting the document that best matches the clients capabilities, from one of several available documents. If the client provides an invalid Accept header, the server will respond with a 406 Not Acceptable error containing a pseudo directory listing. This behaviour can help an attacker to learn more about his target, for example, generate a list of base names, generate a list of interesting extensions, look for backup files and so on.

Impact

Possible information disclosure: directory listing, filename bruteforcing, backup files.

http://192.168.1.18/

Pattern found:

```
<title>406 Not Acceptable</title>
```

Request

```
GET /index HTTP/1.1
Accept: zeablmhb/waom
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

Recommendation

Disable the MultiViews directive from Apache's configuration file and restart Apache.
You can disable MultiViews by creating a **.htaccess** file containing the following line:

Options -Multiviews

References

[mod_negotiation: directory listing, filename bruteforcing](#)

http://www.ush.it/2008/07/02/mod_negotiation-directory-listing-filename-bruteforcing/

[Multiviews Apache, Accept Requests and free listing](#)

<http://www.wisec.it/sectou.php?id=4698ebdc59d15>

[Apache Module mod_negotiation](#)

http://httpd.apache.org/docs/2.2/mod/mod_negotiation.html

Programming Error Messages

This alert requires manual confirmation

Acunetix found one or more error/warning messages. Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

These messages may also contain the location of the file that produced an unhandled exception.
Consult the 'Attack details' section for more information about the affected page(s).

Impact

Error messages may disclose sensitive information which can be used to escalate attacks.

http://192.168.1.18/

Application error messages:

- <http://192.168.1.18/login.php>

You have an error in your SQL syntax

Request

POST /login.php HTTP/1.1

Referer: http://192.168.1.18/

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive

email=12345' \"\"");|]*%00{%-d%0a<%00>%bf%27' ¤&password=u]H[ww6KrA9F.x-F
```

Recommendation

Verify that these page(s) are disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.

References

[PHP Runtime Configuration](#)

<https://www.php.net/manual/en/errorfunc.configuration.php#ini.display-errors>

[Improper Error Handling](#)

https://www.owasp.org/index.php/Improper_Error_Handling

Content Security Policy (CSP) Not Implemented

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks.

Content Security Policy (CSP) can be implemented by adding a **Content-Security-Policy** header. The value of this header is a string containing the policy directives describing your Content Security Policy. To implement CSP, you should define lists of allowed origins for all of the types of resources that your site utilizes. For example, if you have a simple site that needs to load scripts, stylesheets, and images hosted locally, as well as from the jQuery library from their CDN, the CSP header could look like the following:

```
Content-Security-Policy:
default-src 'self';
script-src 'self' https://code.jquery.com;
```

It was detected that your web application doesn't implement Content Security Policy (CSP) as the CSP header is missing from the response. It's recommended to implement Content Security Policy (CSP) into your web application.

Impact

CSP can be used to prevent and/or mitigate attacks that involve content/code injection, such as cross-site scripting/XSS attacks, attacks that require embedding a malicious resource, attacks that involve malicious use of iframes, such as clickjacking attacks, and others.

<http://192.168.1.18/>

Paths without CSP header:

- <http://192.168.1.18/>
- <http://192.168.1.18/index>
- <http://192.168.1.18/index.html>
- <http://192.168.1.18/login.php>

Request

```
GET / HTTP/1.1
Referer: http://192.168.1.18/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

Recommendation

It's recommended to implement Content Security Policy (CSP) into your web application.

Configuring Content Security Policy involves adding the **Content-Security-Policy** HTTP header to a web page and giving it values to control resources the user agent is allowed to load for that page.

References

[Content Security Policy \(CSP\)](#)

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>

[Implementing Content Security Policy](#)

<https://hacks.mozilla.org/2016/02/implementing-content-security-policy/>

Error page web server version disclosure

Application errors or warning messages may disclose sensitive information about an application's internal workings to an attacker.

Acunetix found the web server version number and a list of modules enabled on the target server. Consult the 'Attack details' section for more information about the affected page.

Impact

Error messages information about an application's internal workings may be used to escalate attacks.

http://192.168.1.18/

Request

```
GET /CkRmYJXz2y HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

Recommendation

Properly configure the web server not to disclose information about an application's internal workings to the user. Consult the 'Web references' section for more information.

References

[Custom Error Responses \(Apache HTTP Server\)](#)

<https://httpd.apache.org/docs/current/custom-error.html>

[server_tokens \(Nginx\)](#)

http://nginx.org/en/docs/http/ngx_http_core_module.html#server_tokens

[Remove Unwanted HTTP Response Headers \(Microsoft IIS\)](#)

<https://blogs.msdn.microsoft.com/varunm/2013/04/23/remove-unwanted-http-response-headers/>

Permissions-Policy header not implemented

The Permissions-Policy header allows developers to selectively enable and disable use of various browser features and APIs.

Impact

<http://192.168.1.18/>

Locations without Permissions-Policy header:

- http://192.168.1.18/
- http://192.168.1.18/login.php
- http://192.168.1.18/index
- http://192.168.1.18/index.html

Request

```
GET / HTTP/1.1
Referer: http://192.168.1.18/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/125.0.0.0 Safari/537.36
Host: 192.168.1.18
Connection: Keep-alive
```

References

[Permissions-Policy / Feature-Policy \(MDN\)](#)

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Feature-Policy>

[Permissions Policy \(W3C\)](#)

<https://www.w3.org/TR/permissions-policy-1/>

Coverage

 http://192.168.1.18

 index

 index.html

 login.php

 Inputs

 POST email, password