Scan Comparison

Acunetix Security Audit

16 November 2020

Generated by Acunetix

Scan comparison

Scan details

Start URL	
First scan	https://teache-me-front.herokuapp.com/authenticate.html
Second scan	https://teache-me-front.herokuapp.com/authenticate.html

Threat levels

First scan	Second scan
Acunetix Threat Level 2	Acunetix Threat Level 2

Alert counts

First scan		Second scan	
Total alerts found	17	Total alerts found	14
High	0	High	0
Medium	6	Medium	4
Low	5	Low	4
Informational	6	Informational	6

Unchanged issues

HTML form without CSRF protection

Severity	**Medium**
Reported by module	/Crawler/12-Crawler_Form_NO_CSRF.js

Description

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

Impact

An attacker could use CSRF to trick a victim into accessing a website hosted by the attacker, or clicking a URL containing malicious or unauthorized requests.

CSRF is a type of 'confused deputy' attack which leverages the authentication and authorization of the victim when the forged request is being sent to the web server. Therefore, if a CSRF vulnerability could affect highly privileged users such as administrators full application compromise may be possible.

Affected items

/index.html
Details
Form name: <empty> Form action: <empty> Form method: GET Form inputs: • <empty> [unknown]

TLS 1.0 enabled

Severity	**Medium**
Reported by module	/Scripts/PerServer/SSL_Audit.script

Description

The web server supports encryption through TLS 1.0. TLS 1.0 is not considered to be "strong cryptography" as defined and required by the PCI Data Security Standard 3.2(.1) when used to protect sensitive information transferred to or from web sites. According to PCI, "30 June 2018 is the deadline for disabling SSL/early TLS and implementing a more secure encryption protocol – TLS 1.1 or higher (TLS v1.2 is strongly encouraged) in order to meet the PCI Data Security Standard (PCI DSS) for safeguarding payment data.

Impact

An attacker may be able to exploit this problem to conduct man-in-the-middle attacks and decrypt communications between the affected service and clients.

Affected items

Web Server
Details
The SSL server (port: 443) encrypts traffic using TLSv1.0.

Clickjacking: X-Frame-Options header missing

Severity	**Low**
Reported by module	/Scripts/PerServer/Clickjacking_X_Frame_Options.script

Description

Clickjacking (User Interface redress attack, UI redress attack, UI redressing) is a malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages.

The server didn't return an **X-Frame-Options** header which means that this website could be at risk of a clickjacking attack. The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page inside a frame or iframe. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

Impact

The impact depends on the affected web application.

Affected items

Web Server
Details

Login page password-guessing attack

Severity	**Low**
Reported by module	/Scripts/PerScheme/Html_Authentication_Audit.script

Description

A common threat web developers face is a password-guessing attack known as a brute force attack. A brute-force attack is an attempt to discover a password by systematically trying every possible combination of letters, numbers, and symbols until you discover the one correct combination that works.

This login page doesn't have any protection against password-guessing attacks (brute force attacks). It's recommended to implement some type of account lockout after a defined number of incorrect password attempts. Consult Web references for more information about fixing this problem.

Impact

An attacker may attempt to discover a weak password by systematically trying every possible combination of letters, numbers, and symbols until it discovers the one correct combination that works.

Affected items

/authenticate.html
Details
The scanner tested 10 invalid credentials and no account lockout was detected.
/signup.html
Details
The scanner tested 10 invalid credentials and no account lockout was detected.

Possible sensitive files

Severity	**Low**
Reported by module	/Scripts/PerFolder/Possible_Sensitive_Files.script

Description

A possible sensitive file has been found. This file is not directly linked from the website. This check looks for common sensitive resources like password files, configuration files, log files, include files, statistics data, database dumps. Each one of these files could help an attacker to learn more about his target.

Impact

This file may expose sensitive information that could help a malicious user to prepare more advanced attacks.

Affected items

/debug.log
Details

Content Security Policy (CSP) not implemented

Severity	**Informational**
Reported by module	/httpdata/CSP_not_implemented.js

Description

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 the 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.

Affected items

/authenticate.html
Details

Subresource Integrity (SRI) not implemented

Severity	**Informational**
Reported by module	/RPA/SRI_Not_Implemented.js

Description

Subresource Integrity (SRI) is a security feature that enables browsers to verify that third-party resources they fetch (for example, from a CDN) are delivered without unexpected manipulation. It works by allowing developers to provide a cryptographic hash that a fetched file must match.

Third-party resources (such as scripts and stylesheets) can be manipulated. An attacker that has access or has hacked the hosting CDN can manipulate or replace the files. SRI allows developers to specify a base64-encoded cryptographic hash of the resource to be loaded. The integrity attribute containing the hash is then added to the <script> HTML element tag. The integrity string consists of a base64-encoded hash, followed by a prefix that depends on the hash algorithm. This prefix can either be sha265, sha384 or sha512.

The script loaded from the external URL specified in the Details section doesn't implement Subresource Integrity (SRI). It's recommended to implement Subresource Integrity (SRI) for all the scripts loaded from external hosts.

Impact

An attacker that has access or has hacked the hosting CDN can manipulate or replace the files.

Affected items

/authenticate.html
Details
/selectclass.html
Details

TLS 1.1 enabled

Severity	**Informational**
Reported by module	/Scripts/PerServer/SSL_Audit.script

Description

The web server supports encryption through TLS 1.1. When aiming for Payment Card Industry (PCI) Data Security Standard (DSS) compliance, it is recommended (although at the time or writing not required) to use TLS 1.2 or higher instead. According to PCI, "30 June 2018 is the deadline for disabling SSL/early TLS and implementing a more secure encryption protocol – TLS 1.1 or higher (TLS v1.2 is strongly encouraged) in order to meet the PCI Data Security Standard (PCI DSS) for safeguarding payment data.

Impact

An attacker may be able to exploit this problem to conduct man-in-the-middle attacks and decrypt communications between the affected service and clients.

Affected items

Web Server
Details
The SSL server (port: 443) encrypts traffic using TLSv1.1.

Newly discovered issues

HTML form without CSRF protection

Severity	**Medium**
Reported by module	/Crawler/12-Crawler_Form_NO_CSRF.js

Description

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

Impact

An attacker could use CSRF to trick a victim into accessing a website hosted by the attacker, or clicking a URL containing malicious or unauthorized requests.

CSRF is a type of 'confused deputy' attack which leverages the authentication and authorization of the victim when the forged request is being sent to the web server. Therefore, if a CSRF vulnerability could affect highly privileged users such as administrators full application compromise may be possible.

Affected items

/authenticate.html

Details

Form name: <empty>
Form action: <empty>
Form method: POST

Form inputs:

- your_email [text]
- your_pass [password]
- <empty> [button]

/signup.html

Details

Form name: <empty>
Form action: <empty>
Form method: POST

Form inputs:

- name [text]
- last_name [text]
- email [email]
- pass [password]
- description [textarea]
- <empty> [button]

Password type input with auto-complete enabled

Severity	**Informational**
Reported by module	/Crawler/12-Crawler_Password_Input_Autocomplete.js

Description

When a new name and password is entered in a form and the form is submitted, the browser asks if the password should be saved.Thereafter when the form is displayed, the name and password are filled in automatically or are completed as the name is entered. An attacker with local access could obtain the cleartext password from the browser cache.

Impact

Possible sensitive information disclosure.

Affected items

Web Server
Details
Form name: <empty> Form action: <empty> Form method: POST Form input: • your_pass [password]
Web Server
Details
Form name: <empty> Form action: <empty> Form method: POST Form input: • pass [password]

Undetected issues

HTML form without CSRF protection

Severity	**Medium**
Reported by module	/Crawler/12-Crawler_Form_NO_CSRF.js

Description

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

Impact

An attacker could use CSRF to trick a victim into accessing a website hosted by the attacker, or clicking a URL containing malicious or unauthorized requests.

CSRF is a type of 'confused deputy' attack which leverages the authentication and authorization of the victim when the forged request is being sent to the web server. Therefore, if a CSRF vulnerability could affect highly privileged users such as administrators full application compromise may be possible.

Affected items

/authenticate.html

Details

Form name: <empty>
Form action: <empty>
Form method: GET

Form inputs:

- your_email [text]
- your_pass [password]
- <empty> [button]

/signup.html

Details

Form name: <empty>
Form action: <empty>
Form method: GET

Form inputs:

- name [text]
- last_name [text]
- email [email]
- pass [password]
- description [textarea]
- <empty> [button]

Password field submitted using GET method

Severity	**Medium**
Reported by module	/RPA/Password_In_Get.js

Description

This page contains a form with a password field. This form submits user data using the GET method, therefore the contents of the password field will appear in the URL.Sensitive information should not be passed via the URL. URLs could be logged or leaked via the Referer header.

Impact

Possible sensitive information disclosure.

Affected items

/authenticate.html
Details
Form name: <empty> Form action: <empty> Form method: GET
/signup.html
Details
Form name: <empty> Form action: <empty> Form method: GET

Sensitive page could be cached

Severity	**Low**
Reported by module	/RPA/Cacheable_Sensitive_Page.js

Description

This page contains possible sensitive information (e.g. a password parameter) and could be potentially cached. Even in secure SSL channels sensitive data could be stored by intermediary proxies and SSL terminators. To prevent this, a Cache-Control header should be specified.

Impact

Possible sensitive information disclosure.

Affected items

/signup.html
Details

Password type input with auto-complete enabled

Severity	**Informational**
Reported by module	/Crawler/12-Crawler_Password_Input_Autocomplete.js

Description

When a new name and password is entered in a form and the form is submitted, the browser asks if the password should be saved.Thereafter when the form is displayed, the name and password are filled in automatically or are completed as the name is entered. An attacker with local access could obtain the cleartext password from the browser cache.

Impact

Possible sensitive information disclosure.

Affected items

Web Server
Details
Form name: <empty> Form action: <empty> Form method: GET Form input: • your_pass [password]

Web Server
Details
Form name: <empty> Form action: <empty> Form method: GET Form input: • pass [password]