



zafenity.com

WEB APPLICATION PENETRATION TEST V1.0

SUMMARY OF WORK PERFORMED	1
METHODOLOGY.....	2
TOOLS USED.....	3
EXECUTIVE SUMMARY.....	4-5
INFORMATION GATHERING.....	6
SUMMARY OF FINDINGS.....	7
WEB APPLICATION PENETRATION TEST.....	8-23



SUMMARY OF WORK PERFORMED

A web application penetration test was performed on the zafenity.com site. This application consists of a web portal that provides access to account information and the ability to manage agents and services.

- <https://zafenity.com/>



METHODOLOGY

An Application Penetration Test is designed to identify vulnerabilities in an application which could negatively impact the organization if exploited by an attacker. Manual and automated assessment will be conducted using a testing methodology based on expert knowledge, in combination with information provided by zafenity. Application testing is conducted in accordance with OWASP Top 10, application security best practices, and internal checklists developed to ensure thorough coverage. CrusherslabQA Application Penetration Testing consists of the following phases:

- **Pre-Assessment** – CrusherslabQA will request access to the systems in advance of the test and guide the customer through the testing process on a pre-assessment Call.
- **Enumeration** – Using resources such as DNS, Google, and Bing, CrusherslabQA will look for information that is available that may be helpful to an attacker.
- **Unauthenticated Testing** – CrusherslabQA will evaluate the application from the perspective of an attacker who does not have authenticated access to the application.
- **Authentication Testing** – A large number of vulnerabilities result from weaknesses in the authentication process. This phase examines the various authentication mechanisms in place.
- **Authenticated Testing** – This phase simulates an attacker who has access, or maliciously obtains access, to credentials to log in to the application.
- **Reporting** – A report outlining all identified issues will be prepared which focuses on presenting identified vulnerabilities in a manner which makes them effective to remediate.

TOOLS USED

In addition to a variety of open source tools, exploits, and utilities used on an as-needed basis, the following tools may be used when conducting an Application Penetration Test:

- Google or other search engines
- Mozilla Firefox
- Nessus Network Vulnerability Scanner
- Nikto
- Nmap
- Burp Suite
- SQLMap
- Zap
- dirbuster/gobuster
- Wireshark



EXECUTIVE SUMMARY

During the application penetration test, 10 vulnerabilities were discovered, there is one high risk vulnerability, the medium risk vulnerabilities are three, low risk vulnerabilities are six. Their most high risk is successful brute force attack. There are cryptographic weaknesses that allow an attacker to bypass the password. Here some problems are in code, some misconfiguration and server.Content Security Policy (CSP) Header Not Set, Missing Anti-clickjacking Header

There are some low risks like Cross-Domain JavaScript Source File Inclusion. The page includes one or more script files from a third-party domain. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s), Timestamp Disclosure - Unix, X-Content-Type-Options Header Missing, Strict-Transport-Security Header Not Set

The following chart shows the distribution of findings across all severity levels:

Severity	Count of Findings
High	1
Medium	3
Low	6

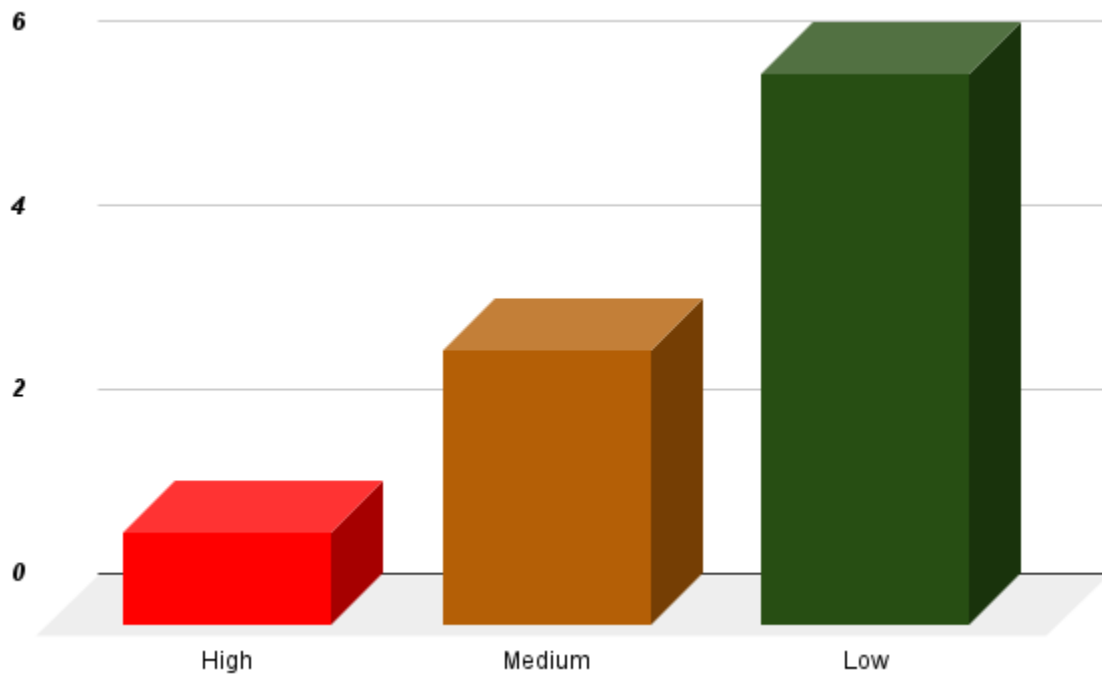


Chart1: Severity and findings

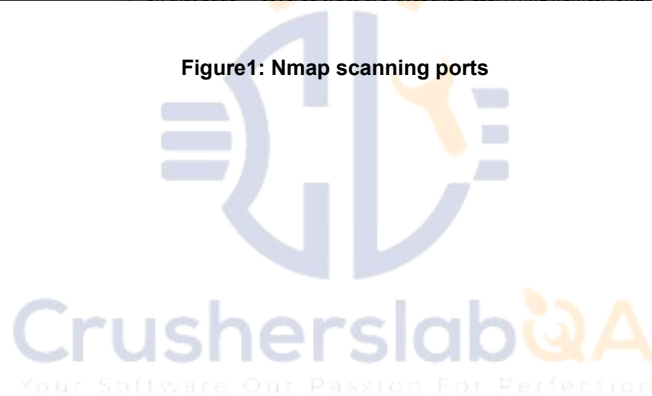


INFORMATION GATHERING

Here we found some ports:

```
Starting Nmap 7.94 ( https://nmap.org ) at 2023-08-02 20:17 +06
Nmap scan report for zafenity.com (104.21.4.70)
Host is up (0.094s latency).
Other addresses for zafenity.com (not scanned): 172.67.131.192 2606:4700:3035::6815:446 2606:4700:3037::ac43:83c0
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
443/tcp    open  https
8080/tcp   open  http-proxy
8443/tcp   open  https-alt
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: firewall
Running (JUST GUESSING): Fortinet embedded (87%)
Aggressive OS guesses: Fortinet FortiGate-50B or 310B firewall (87%)
No exact OS matches for host (test conditions non-ideal).
Content Security Policy (CSP) Header Not Set (3)
OS detection performed: Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 26.25 seconds
```

Figure1: Nmap scanning ports



SUMMARY OF FINDINGS

Finding information in this report is presented in order of severity. Severity ratings within this report are presented without the knowledge of the business risk that the vulnerabilities present to zafenity or its customers.

The following vulnerabilities were discovered during the Application Penetration Test:

#	Finding	Severity
1	Successful Brute Force Attack	High
2	Content Security Policy (CSP) Header Not Set	Medium
3	Cross-Domain Misconfiguration	Medium
4	Missing Anti-clickjacking Header	Medium
5	Cross-Domain JavaScript Source File Inclusion	Low
6	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low
7	Server Leaks Version Information via "Server" HTTP Response Header Field	Low
8	Strict-Transport-Security Header Not Set	Low
9	Timestamp Disclosure - Unix	Low
10	X-Content-Type-Options Header Missing	Low

Request ^	Payload	Status code	Error	Timeout	Length	Comment
0		401	<input type="checkbox"/>	<input type="checkbox"/>	914	
1		401	<input type="checkbox"/>	<input type="checkbox"/>	912	
2	test	401	<input type="checkbox"/>	<input type="checkbox"/>	920	
3	test1	401	<input type="checkbox"/>	<input type="checkbox"/>	920	
4	test3	401	<input type="checkbox"/>	<input type="checkbox"/>	918	
5	test33	401	<input type="checkbox"/>	<input type="checkbox"/>	912	
6	test12	401	<input type="checkbox"/>	<input type="checkbox"/>	912	
7	test123	401	<input type="checkbox"/>	<input type="checkbox"/>	918	
8	test1235	401	<input type="checkbox"/>	<input type="checkbox"/>	918	
9	test1234	200	<input type="checkbox"/>	<input type="checkbox"/>	1032	
10	test4321	401	<input type="checkbox"/>	<input type="checkbox"/>	912	
11	test3222	401	<input type="checkbox"/>	<input type="checkbox"/>	914	

Figure3: Successful Brute Force Attack

Impact:

Once attackers gain access to the system and network of the user account of interest, they can steal valuable personal information like bank and credit account details, personal identity details, health information, etc.

Solution:

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. If your web site requires user authentication, you are a good target for a brute-force attack.

An attacker can always discover a password through a brute-force attack, but the downside is that it could take years to find it. Depending on the password's length and complexity, there could be trillions of possible combinations.

To speed things up a bit, a brute-force attack could start with dictionary words or slightly modified dictionary words because most people will use those rather than a completely random password. These attacks are called dictionary attacks or hybrid brute-force attacks. Brute-force attacks put user accounts at risk and flood your site with unnecessary traffic.

Hackers launch brute-force attacks using widely available tools that utilize wordlists and smart rulesets to intelligently and automatically guess user passwords. Although such attacks are easy to detect, they are not so easy to prevent.

For example, many HTTP brute-force tools can relay requests through a list of open

proxy servers. Since each request appears to come from a different IP address, you cannot block these attacks simply by blocking the IP address. To further complicate things, some tools try a different username and password on each attempt, so you cannot lock out a single account for failed password attempts.

Other techniques you might want to consider are:

- For advanced users who want to protect their accounts from attack, give them the option to allow login only from certain IP addresses.
- Assign unique login URLs to blocks of users so that not all users can access the site from the same URL.
- Use a CAPTCHA to prevent automated attacks
- Instead of completely locking out an account, place it in a lockdown mode with limited capabilities.

2. Content Security Policy (CSP) Header Not Set

Severity : Medium

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. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

Evidence:

```
HTTP/1.1 200 OK
Date: Wed, 02 Aug 2023 14:30:35 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
X-Powered-By: Next.js
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=9xCeT6dVeH%2BHG%2F7K%2FSheiJriMVZ9Yt%2BA5BZ0C60mE1%2FoevYswrtD1xZg5Y6Wt5Q%2Fst25%2B5m68uG%2BF9nhicqYJXp4UENBcFRriInPvvcZ4Ig%2B40VLzN%2Fz1I9yNf0VPo%3D"}], "group": "cf-nel", "max_age": 604800}
NEL: {"success_fraction": 0, "report_to": "cf-nel", "max_age": 604800}
Server: cloudflare
CF-RAY: 7f070503d8d8a069-SIN
alt-svc: h3=":443"; ma=86400
content-length: 3798
```

Figure4: Content Security Policy (CSP) Header Not Set

```
<!DOCTYPE html><html><head><meta charset="utf-8"/><title></title><script src="https://js.pusher.com/beams/1.0/push-notifications-cdn.js" async=""></script><meta name="google-site-verification" cont
```

Figure5: Content Security Policy (CSP) Header Not Set

Impact:

Content Security Policy (CSP) adds a layer of security which helps to detect and mitigate certain types of attacks such as Cross Site Scripting (XSS) and data injection attacks. Hackers use XSS attacks to trick trusted websites into delivering malicious content. The browser executes all code from trusted origin and can't differentiate between legitimate and malicious code, so any injected code is executed as well.

Solution:

To fix Content Security Policy (CSP) Header Not Set you need to configure your web server to return the Content-Security-Policy HTTP Header and giving it values to control what resources the browser is allowed to load for your page.

The syntax is:

Content-Security-Policy: <policy-directive>; <policy-directive>

Where:

<policy-directive>

consists of: <directive> <value> with no internal punctuation.

Example:

Content-Security-Policy: default-src 'self' <http://example.com>;

For a full list of possible directives and more examples please check <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>.

3. Cross-Domain Misconfiguration

Severity : Medium

Description:

Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server.

Evidence:

```
HTTP/1.1 200 OK
Date: Wed, 02 Aug 2023 14:31:13 GMT
Content-Type: text/css; charset=utf-8
Connection: keep-alive
access-control-allow-origin: *
cache-control: public, max-age=31536000
last-modified: Sat, 26 Oct 1985 08:15:00 GMT
etag: W/"37c0-cW5oWHzFcgrzuKuBtMixbFPjmt4"
via: 1.1 fly.io
fly-request-id: 01G8E0BJ897NZBB8W467YXR869-sin
CF-Cache-Status: HIT
Age: 1123142
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
X-Content-Type-Options: nosniff
Server: cloudflare
CF-RAY: 7f0705f72f55bc2b-DAC
content-length: 14272
```

Figure6: Cross-Domain Misconfiguration

```

/* required styles */

.leaflet-pane,
.leaflet-tile,
.leaflet-marker-icon,
.leaflet-marker-shadow,
.leaflet-tile-container,
.leaflet-pane > svg,
.leaflet-pane > canvas,
.leaflet-zoom-box,
.leaflet-image-layer,
.leaflet-layer {
    position: absolute;
    left: 0;
    top: 0;
}
.leaflet-container {
    overflow: hidden;
}
.leaflet-tile,
.leaflet-marker-icon,
.leaflet-marker-shadow {
    -webkit-user-select: none;
    -moz-user-select: none;
    user-select: none;
    -webkit-user-drag: none;
}
/* Prevents IE11 from highlighting tiles in blue */
.leaflet-tile::selection {
    background: transparent;
}

```

Figure7: Cross-Domain Misconfiguration

Impact:

A CORS misconfiguration can leave the application at a high risk of compromises resulting in an impact on the confidentiality and integrity of data by allowing third-party sites to carry out privileged requests through your website's authenticated users such as retrieving user setting information or saved payment card data.

On the other hand, the risk is low for applications that deal with public data and require that resources are sent to other origins. The configuration could be expected behavior and it would need to be up to the penetration tester to identify the appropriate risk and the organization to understand and mitigate, or accept the risk.

Solution:

To mitigate the risk of CORS, we always recommend whitelisting your Access-Control-Allow-Origin instead of wildcarding. Using a subdomain such as subdomain.yoursite.com makes it more difficult for the attackers given they would need to find a vulnerability (such as cross-site scripting or cross-site request forgery) to issue the cross-origin request. However, it is frowned upon because it does not provide the critical need-to-know security control. With whitelisting, the scope of your Access-Control-Allow-Origin will be limited to only the sites that deal directly with your primary site or API and exclude any of your sites that do not.



4. Missing Anti-clickjacking Header

Severity: Medium

Description:

The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

Evidence:

```
HTTP/1.1 200 OK
Date: Wed, 02 Aug 2023 14:30:35 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
X-Powered-By: Next.js
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=9xCeT6dVeH%2BHJG%2F7K%2FSheiJriMVZ9Yt%2BA5BZ0C60mE1%2FoeVyswrtD1xzg5Y6Wt5Q%2Fst25%2B5m68uG%2BF9nhicqYJXp4UENBcFRRIInPvvcZ4Ig%2B40VLzN%2Fz1I9yNf0VP0%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 7f070503d8d8a069-SIN
alt-svc: h3=":443"; ma=86400
content-length: 3798
```

Figure8: Missing Anti-clickjacking Header

Impact:

Clickjacking is a type of attack that tricks users into clicking on something they didn't intend to click on. The attacker usually accomplishes this by overlaying a transparent layer on top of a legitimate website, so when the user tries to click on something, they're actually clicking on the attacker's hidden content.

One way to protect your web application against clickjacking attacks is to add an anti-clickjacking header to your HTTP responses. This header tells web browsers to block any attempts to embed your website in a frame or iframe, which is the most common way that clickjacking attacks are carried out.

If your vulnerability scanner has identified a missing anti-clickjacking header, it means that your website is not currently protected against clickjacking attacks. In this guide, we'll walk you through the steps to add an anti-clickjacking header to your website's HTTP responses.

Solution:

Adding an anti-clickjacking header to your website is an important security measure, but it's not the only one you should take. You should also update your website's security policy to reflect your new security measures.

Your security policy should include details about how you're protecting your website from clickjacking attacks, as well as any other security measures you've implemented. You should also include instructions for your developers on how to maintain and update your website's security measures in the future.

5. Cross-Domain JavaScript Source File Inclusion

Severity: Low

Description:

The page includes one or more script files from a third-party domain.

Evidence:

```
HTTP/1.1 200 OK
Date: Wed, 02 Aug 2023 14:30:35 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
X-Powered-By: Next.js
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https://a.ne1.cloudflare.com/report/v37s=9xCeT6dVeh%28HJG%2F7K%2FShe1Jr1MVZ9Yt%28ASBZ0C60mE1%2FoevYswrtd1xzg5Y6Wt5Q%2Fst25%285m68u0%28F9nhicqfJXp4UEncFRRIinPvvcZ4Ig%2840VLz%2F2119yNf0Vp0%3D"}], "group": "cf-nel", "max_age": 604800}
NEL: {"success_fraction": 0, "report_to": "cf-nel", "max_age": 604800}
Server: cloudflare
CF-RAY: 7f070503d8d8a069-SIN
alt-svc: h3=":443"; ma=86400
content-length: 3798

<!DOCTYPE html><html><head><meta charset="utf-8"/><title></title><script src="https://js.pusher.com/beams/1.0/push-notifications-cdn.js" async=""></script><meta name="google-site-verification" content="..."></head><body></body></html>
```

Figure9: Cross-Domain JavaScript Source File Inclusion

Impact:

Cross-Domain JavaScript Source File Inclusion is a security vulnerability that can be exploited by attackers to execute malicious code on your web application. This vulnerability occurs when your web application loads JavaScript files from an external domain without proper validation, allowing an attacker to inject their own code and potentially take control of the application. In this guide, we will cover how to fix this vulnerability in a step-by-step manner.

Solution:

Cross-Domain JavaScript Source File Inclusion is a serious vulnerability that can be exploited by attackers to execute malicious code on your web application. By following the steps outlined in this guide, you can fix this vulnerability and ensure that your web application is secure from this type of attack. Remember to always keep your software up-to-date and to sanitize user input to prevent attackers from injecting malicious code into your web application. Use Content Security Policy to restrict the domains that are allowed

6. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description:

The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Evidence:

```
HTTP/1.1 200 OK
Date: Wed, 02 Aug 2023 14:30:35 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
X-Powered-By: Next.js
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=9xCeT6dVeH%2BHG%2F7K%2FShelJr1MVZ9Yt%2BA5BZ0C60mE1%2FoevYswrtd1xzg5Y6wt5Q%2Fst25%2B5m68u6%2BF9nhicqYJXp4UENBCFRRiInPvcZ4Ig%2B40VLzN%2Fz1I9yNf0VPo%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 7f070503d8d8a069-SIN
alt-svc: h3=":443"; ma=86400
content-length: 3798
```

Figure10: Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Impact:

Why “Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)” can be dangerous. The issue means that the web server returns one or more X-Powered-By HTTP headers. The X-Powered-By header is one of the HTTP response headers that can be returned by the web server.

Solution:

Ensure that your web server, application server, load balancer, etc. is configured to suppress 'X-Powered-By' headers

7. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description:

The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Evidence:

```
HTTP/1.1 200 OK
Content-Type: application/javascript
Content-Length: 83225
Connection: keep-alive
Date: Tue, 01 Aug 2023 17:05:00 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, HEAD
Last-Modified: Thu, 10 Sep 2020 10:59:53 GMT
ETag: "7b1eadae70451cf223f5e9e211565809"
Accept-Ranges: bytes
Server: AmazonS3
Vary: Accept-Encoding
X-Cache: Hit from cloudfront
Via: 1.1 ae495479ab117e6473f411eb6dd0ba98.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: SIN5-C1
X-Amz-Cf-Id: hJYpW0WA3UQeNX9Fg7h7jA0tMGjjo0VpeeKcElpYohonbpYQQSusFA==
Age: 77172
```

Figure11: Server Leaks Version Information via "Server" HTTP Response Header Field

Impact:

Why "Server Leaks Version Information via "Server" HTTP Response Header Field" can be dangerous. If your application leaks web server version details via "Server" HTTP response header field the attacker may use it to find and exploit security vulnerabilities present specifically in the reported web server information.

Solution:

How to fix "Server Leaks Version Information via "Server" HTTP Response Header Field" You should configure your web server and other HTTP transport software like proxy servers and load balancers to remove the Server field from HTTP response header or replace it with a generic value.


8. Strict-Transport-Security Header Not Set

Severity: Low

Description:

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standard track protocol and is specified in RFC 6797.

Evidence:



```

HTTP/1.1 200 OK
Content-Type: application/javascript
Content-Length: 83225
Connection: keep-alive
Date: Tue, 01 Aug 2023 17:05:00 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, HEAD
Last-Modified: Thu, 10 Sep 2020 10:59:53 GMT
ETag: "7b1eadae70451cf223f5e9e211565809"
Accept-Ranges: bytes
Server: AmazonS3
Vary: Accept-Encoding
X-Cache: Hit from cloudfront
Via: 1.1 ae495479ab117e6473f411eb6dd0ba98.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: SIN5-C1
X-Amz-Cf-Id: hJYpW0WA3UQeNX9Fg7h7jA0tMGjjo0VpeeKcElpYohonbpYQQSusFA==
Age: 77172
  
```

Figure12: Strict-Transport-Security Header Not Set

Impact:

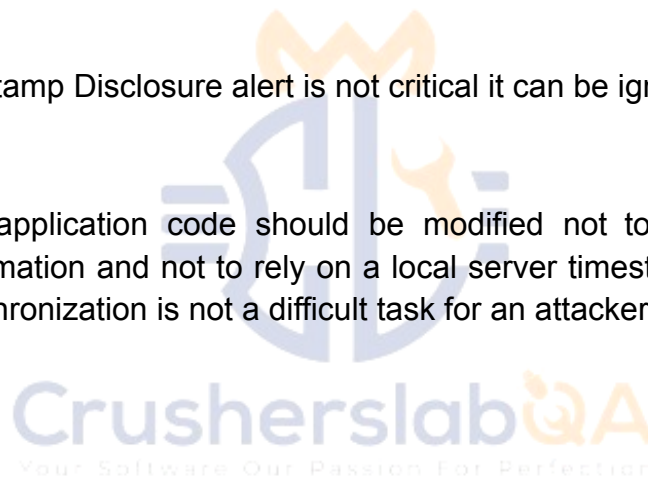
A timestamp disclosed by the application server or web server can be used to retrieve other sensitive information e.g. when used as a salt or a token during authentication or encryption.

Solution:

Any Timestamp Disclosure alerts should be manually reviewed to confirm that a) these are actual server timestamp leaks, b) the disclosed timestamp data is not sensitive as it is not used in any form to generate any sensitive information on the server side.

If a given Timestamp Disclosure alert is not critical it can be ignored.

Otherwise the application code should be modified not to disclose current timestamp information and not to rely on a local server timestamp as generally timestamp synchronization is not a difficult task for an attacker.



10. X-Content-Type-Options Header Missing

Severity: Low

Description:

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy

versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

Evidence:

```
HTTP/1.1 200 OK
Date: Wed, 02 Aug 2023 14:30:35 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
X-Powered-By: Next.js
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https://a.ne1.cloudflare.com/report/v3?ts=9xCeT6dVeh%28HJG%2F7K%2FSheijrIMVZ9Yt%28A5BZ0C60mE1%2FoeVYswrtd1xzg5Y6Wt5Q%2Fst25%2B5m68u6%28F9hnicqYJXp4UENBcFRRiInPvvcZ4Igs2B4OVLzN%2Fz1I9ytf8Vp0x3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 7f070503d8d8a069-SIN
alt-svc: h3=":443"; ma=86400
content-length: 3798

<!DOCTYPE html><html><head><meta charset="utf-8"/><title></title><script src="https://js.pusher.com/beams/1.0/push-notifications-cdn.js" async=""></script><meta name="google-site-verification" content="
```

Figure14: X-Content-Type-Options Header Missing

Impact:

The 'X-Content-Type-Options Header Missing' vulnerability is a common security issue in web applications. This vulnerability arises when a web server doesn't set the 'X-Content-Type-Options' header in its response, allowing attackers to perform content-type sniffing attacks.

Solution:

The 'X-Content-Type-Options Header Missing' vulnerability can lead to serious security issues in web applications. By adding the 'X-Content-Type-Options' header and implementing CSP, you can protect your web application from content injection attacks like XSS. It is essential to regularly scan and test your web application to ensure that it is free from vulnerabilities and to fix any vulnerabilities that are detected promptly.