VULNERABILITY REPORT OF MAIN WEBSITE

TEAM – 2.6 , TOPIC – MALWARE DETECTION AND CLASSIFICATION

1. VULNERABILITY NAME: HTTP METHODS ALLOWED (PER DIRECTORY)

CWE : CWE-650 (https://cwe.mitre.org/data/definitions/650.html)
Description: By calling the OPTIONS method, it is possible to determine which HTTP methods are allowed on each directory.
Business Impact: The vulnerability related to "HTTP METHODS ALLOWED (PER DIRECTORY)" can have significant business impacts, including data breaches, damage to your reputation, and potential legal and regulatory consequences due to the unauthorized access to sensitive data on your web server.
Vulnerability Path : https://opensourcefootball.com(75.2.60.5)
Vulnerability Parameter: https://75.2.60.5/opensourcefootball.com / HTTP Methods Allowed
About this vulnerability:
By calling the OPTIONS method, it is possible to determine which HTTP methods are allowed on each directory.
The following HTTP methods are considered insecure: PUT, DELETE, CONNECT, TRACE, HEAD
Many frameworks and languages treat 'HEAD' as a 'GET' request, albeit one without any body in the response. If a security constraint was set on 'GET' requests such that only 'authenticatedUsers' could access GET requests for a particular servlet or resource, it would be bypassed for the 'HEAD' version. This allowed unauthorized blind submission of any privileged GET request.

Ways to exploit vulnerabilities: Using the PUT method, you can upload any file on the server. This can be used to perform Cross Site Scripting (XSS). How you do this is explained below.

PUT /XSS.html HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.myblog.com

Accept-Language: en-us

Connection: Keep-Alive

Content-type: text/html

Content-Length: 182

(Input your XSS script here)

The server responds back with a 201 status code which says “file was created successfully”.

HTTP/1.1 201 Created

Date: Mon, 05 May 2014 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Content-type: text/html

Content-length: 30

Connection: Closed

Now we can try to access this uploaded XSS.html file in browser. As soon as you access this page, you get an XSS pop-up.

Likewise, this can be further exploited to perform Command Injection as well, though I haven't tried this yet. If application uses XML, then XML External Entity attack can also be performed. Havent done this too yet. Directory Traversal attack may be possible, too.

Recommendation/solution :

- As this list may be incomplete, the plugin also tests - if 'Thorough tests' are enabled or 'Enable web applications tests' is set to 'yes' in the scan policy - various known HTTP methods on each directory and considers them as unsupported if it receives a response code of 400, 403, 405, or 501.

2. VULNERABILITY NAME: HSTS MISSING FROM HTTPS SERVER

CWE: 523

Description: The remote HTTPS server is not enforcing HTTP Strict Transport Security (HSTS). HSTS is an optional response header that can be configured on the server to instruct the browser to only communicate via HTTPS. The lack of HSTS allows downgrade attacks, SSL-stripping man-in-the-middle attacks, and weakens cookie-hijacking protections.

Business Impact: In brief, the business impact of the "HSTS Missing from HTTPS Server" vulnerability includes potential data exposure, the risk of man-in-the-middle attacks, a loss of customer trust, possible regulatory compliance issues, and negative SEO impact. This can result in data breaches, reputation damage, reduced website traffic, legal consequences, and lower search engine rankings.

Vulnerability Path : https://opensourcefootball.com(75.2.60.5)

Vulnerability Parameter: https://75.2.60.5/opensourcefootball.com/HSTS Missing From HTTPS Server

Threats:

HSTS addresses the following threats:

- User bookmarks or manually types http://example.com and is subject to a man-in-the-middle attacker.
- HSTS automatically redirects HTTP requests to HTTPS for the target domain.
- Web application that is intended to be purely HTTPS inadvertently contains HTTP links or serves content over HTTP.
- HSTS automatically redirects HTTP requests to HTTPS for the target domain.
- A man-in-the-middle attacker attempts to intercept traffic from a victim user using an invalid certificate and hopes the user will accept the bad certificate.
- HSTS does not allow a user to override the invalid certificate message.

Ways to exploit vulnerabilities:

- One of the example is you log into a free Wi-Fi access point at an airport and start surfing the web, visiting your online banking service to check your balance and pay a couple of bills. Unfortunately, the access point you're using is actually a hacker's laptop, and they're intercepting your original HTTP request and redirecting you to a clone of your bank's site instead of the real thing. Now your private data is exposed to the hacker.Strict Transport Security resolves this problem; as long as you've accessed your bank's website once using HTTPS, and the bank's website uses Strict Transport Security, your browser will know to automatically use only HTTPS, which prevents hackers from performing this sort of man-in-the- middle attack.
- All present and future subdomains will be HTTPS for a max-age of 1 year. This blocks access to pages or subdomains that can only be served over HTTP.
- HTTP Copy to Clipboard.
- Strict-Transport-Security: max-age=31536000; includeSubDomains
- Although a max-age of 1 year is acceptable for a domain, two years is the recommended value as explained on https://hstspreload.org.
- In the following example, max-age is set to 2 years, and is suffixed with preload, which is necessary for inclusion in all major web browsers' HSTS preload lists, like Chromium, Edge, and Firefox.
- Strict-Transport-Security: max-age=63072000; includeSubDomains; preload

Recommendation/solution :

- Configure the remote web server to use HSTS.

• To fix the HSTS Missing from HTTP Server error, follow the 5 steps below.

1. Create a Full Website Backup before adding the HTTP Transport Security Header
2. Use an HTTP to HTTPS Redirect with 301 Status Code
3. Add the HSTS Header to the Web Server for Forcing the Usage of HTTPS
4. Add the Website to the HSTS Preload List of Google for Protection
5. Audit and Validate the HSTS Header from the Website

Fixation and solution of HSTS Missing from HTTPS Server provides a better trust for the websites from search engines and web users.

3. VULNERABILITY NAME: SSL CERTIFICATE 'COMMONNAME' MISMATCH

CWE: 297 (https://cwe.mitre.org/data/definitions/297.html)
OWASP 2017-A3

Description: The service running on the remote host presents an SSL certificate for which the 'commonName' (CN) attribute does not match the hostname on which the service listens.
Vulnerability Path : https://opensourcefootball.com(99.83.231.61)
Vulnerability Parameter: https://75.2.60.5/opensourcefootball.com/SSL Certificate 'commonName' Mismatch

Business Impact:

A "Common Name" (CN) mismatch in an SSL certificate can lead to:

- Trust issues.
- Security risks.
- User loss.
- Reputation damage.
- Legal and compliance problems.

Common Consequences:

Scope	Impact	Likelihood
Access Control	Technical Impact: *Gain Privileges or Assume Identity* The data read from the system vouched for by the certificate may not be from the expected system.	
Authentication Other	Technical Impact: *Other* Trust afforded to the system in question - based on the malicious certificate - may allow for spoofing or redirection attacks.	

Example 1

The following OpenSSL code obtains a certificate and verifies it.
(bad code)
Example Language: **C**

```
cert = SSL_get_peer_certificate(ssl);
if (cert && (SSL_get_verify_result(ssl)==X509_V_OK)) {
// do secret things
}
```

Even though the "verify" step returns X509_V_OK, this step does not include checking the Common Name against the name of the host. That is, there is no guarantee that the certificate is for the desired host. The SSL connection could have been established with a malicious host that provided a valid certificate.

Solution: If the machine has several names, make sure that users connect to the service through the DNS hostname that matches the common name in the certificate.

4. VULNERABILITY NAME: SSL CERTIFICATE SIGNED USING WEAK HASHING ALGORITHM (KNOWN CA)

CWE-310 (https://cwe.mitre.org/data/definitions/310.html)

OWASP 2017-A3

Description: This vulnerability poses a significant risk to IT infrastructure as it can be used to compromise the security of the SSL certificate. Attackers can exploit the weak hashing algorithm to gain access to sensitive data or to launch malicious attacks on the system (OWASP Testing

Guide, 2020). As a result, the system may be vulnerable to malicious attacks, data leakage, and other security incidents.

Vulnerability Path : https://opensourcefootball.com(99.83.231.61)

Vulnerability Parameter: https://75.2.60.5/opensourcefootball.com/SSL Certificate 'commonName' Mismatch

Business Impact:

In brief, the business impact of having an SSL certificate signed using a weak hashing algorithm (known CA) includes:

- Data Breach Risk: Increased vulnerability to data breaches.
- Loss of Customer Trust: Decreased trust from users, potentially leading to reduced engagement and sales.
- Regulatory Compliance Issues: Non-compliance with data protection regulations, resulting in fines and legal repercussions.
- Reputation Damage: Negative publicity and damage to the organization's reputation.
- Financial Consequences: Potential financial losses due to breach-related costs and customer churn.

Addressing this vulnerability is crucial to maintaining a secure and trusted online presence.

Exploits: an example of the vulnerability The following example code illustrates the use of a weak
hashing algorithm to sign an SSL certificate.

```
openssl req -new -newkey rsa:2048 -nodes -keyout server.key -out server.csr

openssl x509 -req -days 365 -in server.csr -signkey server.key -sha1
```

In the code above, the command “sha1” is used to sign the SSL certificate using the SHA-1 algorithm.
This is an example of a weak hashing algorithm and should be avoided in order to protect the SSL certificate from being compromised (OWASP Testing Guide, 2020).

Solution: The most effective solution to this vulnerability is to use a stronger algorithm to sign the SSL certificate. Specifically, SSL certificates should be signed with a SHA-2 algorithm, which is more secure than the SHA-1 algorithm (OWASP Testing Guide, 2020). Additionally, the system should be regularly monitored for any signs of potential security incidents such as suspicious activity, data leakage, or malicious attacks.