Web Security – IE2062

Bug Bounty Hunting Journal:

A Journey Through Web Vulnerabilities

A D A Ihansa

IT22899606

Table of Contents

Introduction

This document chronicles my exploration of the fascinating world of bug bounty hunting. It serves as a detailed record of my efforts to identify and understand vulnerabilities within some of the internet's most popular websites. My journey began with a fundamental interest in web security and a desire to contribute to a more secure digital landscape. This initial curiosity rapidly evolved into a passion as I delved deeper into the complexities of web security mechanisms.

This work serves a dual purpose. Firstly, it aims to educate readers on the prevalent security issues that continue to plague many websites today. Through the sharing of my experiences and insights, I hope to illuminate common vulnerabilities that often escape detection. This document seeks to raise awareness regarding the critical importance of web security. In today's digital age, where a significant portion of our lives transpires online, understanding potential risks and mitigation strategies is paramount. The document is divided into ten distinct chapters, each focusing on a specific website. These websites were chosen due to their widespread popularity and the diverse range of potential security concerns they represent. I will detail the specific tools and techniques employed during my vulnerability assessments. These tools, commonly utilized within the web security field, range from automated scanners to more specialized software designed to target specific vulnerability types. Each chapter will further provide an in-depth analysis of the vulnerabilities discovered throughout my exploration. These vulnerabilities, while varied in nature, all pose a significant threat to the security and integrity of the targeted websites and their users.

By sharing my experiences as a bug bounty hunter, I aspire to inspire others to explore the realm of web security. Regardless of your current experience level, this ever-evolving field offers continuous learning opportunities for both seasoned professionals and curious newcomers.

Tools used

1. Sublist3r

Sublist3r is a Python tool designed to enumerate subdomains of websites using Open Source Intelligence (OSINT). It is primarily used by penetration testers and bug hunters to collect and gather subdomains for the domain they are targeting.

Here are some key features of Sublist3r:

- It enumerates subdomains using many search engines such as Google, Yahoo, Bing, Baidu, and Ask.

- It also uses tools like Netcraft, Virustotal, ThreatCrowd, DNSdumpster, and ReverseDNS for enumeration.

- Sublist3r has integrated a tool called subbrute to increase the possibility of finding more subdomains using bruteforce with an improved wordlist.

- It supports both Python 2 and Python 3.

- Sublist3r depends on the requests, dnspython, and argparse python modules.

Sublist3r is a versatile and powerful tool that can quickly enumerate subdomains of a given domain, helping to gather information about the target domain and potentially identify vulnerabilities or misconfigurations.

2. Httprobe

Httprobe is a command-line tool designed to probe a list of domains and determine which ones have working HTTP and HTTPS servers13. It automates the process of checking multiple domains for active web servers, helping users identify reachable websites and web applications.

Here are some key features of Httprobe:

- Domain Probing: Httprobe accepts a list of domain names as input and systematically probes each domain to determine if there are active HTTP or HTTPS servers associated with them.

- HTTP and HTTPS Support: Httprobe supports both HTTP and HTTPS protocols, allowing users to check for active web servers using either protocol.

- Efficient Parallel Processing: Httprobe utilizes parallel processing to probe multiple domains simultaneously, optimizing performance and reducing the time required to complete the probing process2.

- Simple Command-Line Interface: Users interact with Httprobe through the command line, providing commands and options to specify input files, configure probing parameters, and control the behavior of the tool.

- Output Formats: Httprobe provides options for customizing the output format to suit different use cases and preferences.

- Error Handling: Httprobe includes robust error handling mechanisms to handle various scenarios, such as network errors, timeouts, and invalid responses.

- Open Source and Community Support: Httprobe is an open-source project hosted on GitHub, allowing users to view the source code, contribute improvements, report issues, and provide feedback.

Httprobe is a valuable tool for network administrators, security professionals, and penetration testers who need to quickly identify active web servers within a list of domains. Its efficiency, flexibility, and support for HTTP and HTTPS protocols make it a useful asset for assessing the online presence and accessibility of websites and web applications.

3. Netcraft

Netcraft is an Internet services company based in London, England. Founded by Mike Prettejohn, Netcraft has been protecting companies online since 19962. The company provides a range of services relating to internet data analysis, defenses against fraud and phishing, web application security testing, and automated network scanning.

Netcraft is a world leader in cybercrime detection, disruption, and takedown. It uses extensive automation, constant innovation, and unique insight to detect and disrupt phishing and other cyberattacks at scale. Netcraft analyzes millions of suspected malicious sites each day, typically blocking an attack within minutes of discovery. It performs takedowns for a significant portion of the world's phishing attacks.

Netcraft's services include cybercrime detection and threat intelligence. It detects, blocks, and disrupts a full range of cyberattacks, including phishing, advance fee fraud, executive impersonation, fake shops, malware, investment scams, malicious JavaScript, and dozens of other techniques. Once confirmed, attacks impersonating your brand are blocked in Netcraft's threat feeds, protecting billions of people.

Netcraft is a comprehensive platform for cybercrime detection and disruption, offering robust cyber defense solutions for organizations around the world. In summary, Netcraft is a robust platform that provides end-to-end cybercrime detection, threat intelligence, and disruption & takedown solutions. It is a valuable resource for anyone looking to protect their brand, organization, employees, and customers from cyber threats.

4. Spiderfoot

SpiderFoot is an open-source tool designed for reconnaissance and intelligence gathering. It is written in Python and automates the process of gathering information about targets. SpiderFoot uses a wide range of data sources, including search engines, public databases, and APIs, to collect data related to domains, IP addresses, email addresses, and more.

SpiderFoot's modules feed each other in a publisher/subscriber model to ensure maximum data extraction. These modules can perform tasks like host/sub-domain/TLD enumeration/extraction,

email address, phone number and human name extraction, Bitcoin and Ethereum address extraction, check for susceptibility to sub-domain hijacking and much more.

SpiderFoot can be used through an intuitive web-based interface or a command-line option. It is a powerful tool for anyone interested in open-source intelligence (OSINT), making it easier to gather and analyze data from various sources.

5. Google dorks

Google Dorks, also known as Google Hacking or Google Dorking, are specialized search queries that leverage Google's powerful search engine to unearth specific information and vulnerabilities that might not be accessible through standard searches.

Google Dorks use advanced operators to tell Google exactly what you want. These operators can be used to filter down search results, revealing information that wasn't meant to be found. They can reveal sensitive or private information about websites and the companies, organizations, and individuals that own and operate them.

Google Dorks can be used for various purposes. In the hands of security researchers and ethical hackers, they can be used to uncover potential vulnerabilities and misconfigurations in websites. On the other hand, malicious hackers might use Google Dorks to gather data on their targets.

Despite their potential misuse, Google Dorks are a powerful tool for Open Source Intelligence (OSINT), allowing users to push Google Search to its limits and find almost anything they want on the web.

6. OWSAP ZAP

OWASP ZAP, short for Zed Attack Proxy, is an open-source web application security scanner. It is designed to be used by both those new to application security as well as professional penetration testers. It has been one of the most active Open Worldwide Application Security Project (OWASP) projects and has been given Flagship status.

ZAP provides several built-in features, including an intercepting proxy server, traditional and AJAX web crawlers, an automated scanner, a passive scanner, forced browsing, a fuzzer, WebSocket support, and scripting languages. It also offers Plug-n-Hack support and a plugin-based architecture, which allows new or updated features to be added. The GUI control panel has been described as easy to use.

When used as a proxy server, ZAP allows the user to manipulate all of the traffic that passes through it, including traffic using HTTPS. It can also run in a daemon mode, which is then controlled via a REST API. As of August 1, 2023, the ZAP development team announced that ZAP was leaving the OWASP Foundation to join The Software Security Project, as a founding project. Henceforth, it will be simply called ZAP.

7. SQL injection

SQL Injection is a method where harmful users can insert SQL commands into an SQL statement through web page input. For instance, if a web page employs an SQL statement like:

SELECT * FROM users WHERE email_address = '[user input]' AND password = '[user input]'

And the user input is ‘ OR 1 = 1;/, the SQL statement would transform into:

SELECT * FROM users WHERE email_address = ‘ OR 1 = 1;/ AND password = /--

The OR 1 = 1 is always true, which results in the SQL statement returning all records from the users table, effectively circumventing any authentication checks. The / and -- are comment indicators in SQL, causing the rest of the query (i.e., the password check) to be disregarded. This is a basic example of an SQL Injection attack, which can result in unauthorized access if the website is susceptible. It's vital for developers to use parameterized queries or prepared statements to avoid such vulnerabilities.

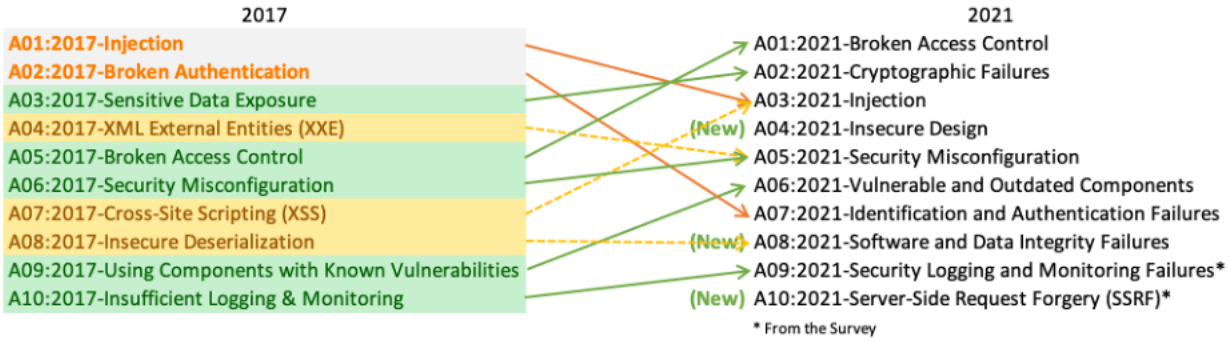
8. checking for HTTP methods

Specific HTTP methods, like DELETE and PUT, can threaten server security. DELETE has the potential to remove resources from the server, while PUT can upload and run files. However, these methods need to be used with care as they can jeopardize the Confidentiality, Integrity, and Availability facets of the server and its users. To evaluate which HTTP methods are supported, I used Burp proxy to intercept requests and Repeater to change the request method. This enabled me to dispatch the modified requests to the server and scrutinize the responses.

OWSAP TOP 10

The Open Web Application Security Project is a not-for-profit organization that strives to enhance security. The OWASP Top 10 serves as a fundamental awareness guide for developers and web application security, reflecting a wide agreement on the most severe security threats to web applications.

The most recent OWASP Top 10 list was published recently in 2021.



-Broken Access Control

Access control involves imposing certain restrictions or limits on website users based on their requirements. If we fail to assign correct access rights, unauthorized parties may alter or destroy the website's data.

- Cryptographic Failures

The use of outdated or weak cryptographic algorithms, default encryption keys, and compromised keys are some common issues under this vulnerability. The main impact of this vulnerability is the compromise of all data, including personal data, credentials, credit card information, etc.

- Injection

Injection occurs when the developer does not validate or sanitize user inputs, and hostile data is concatenated, etc. SQL injection and NoSQL injections are the most common attack vectors in injection vulnerability.

- Insecure Design

The use of unsafe APIs and missing user input bounds are some major issues of this vulnerability. When considering software and services, if the foundation of the software or services is not well-built, it may affect the security of that system.

- Security Misconfiguration

Unpatched flaws, default configurations, unused pages, unprotected files and directories, and unnecessary services are some common attack vectors of security misconfiguration. This vulnerability primarily occurs due to unstable default settings.

- Vulnerable and Outdated Components

Using insecure software configurations, old and unpatched dependencies, and vulnerable components are some common issues of this vulnerability. There are many tools to find these types of vulnerabilities. For example, we can use Shodan.

- Identification and Authentication Failures

This vulnerability occurs when an application does not attempt to prevent brute force password attacks, when the app has flaws in password recovery, etc. Due to these authentication-related attacks, user identities may be disclosed to unauthorized parties.

- Software and Data Integrity Failures

This vulnerability occurs due to unchecked and unverified apps. Nowadays, app developers add auto-update functionality, so without checking the update, the app updates itself. So an attacker may add his own update to attack the app.

- Security Logging and Monitoring Failures

Logins are not backed up, the monitoring system fails to identify malicious activities, missing alerting systems are some issues of this vulnerability. Because of these, attack information leakages can occur.

- Server-side Request Forgery

If an application shows a preview of the URLs, that application is vulnerable to this attack. Fetching a URL has become a common scenario of modern web applications, so as a result, SSRF has increased.

Website Analysis

Chapter 1: airbnb.com

First, I did the basic reconnaissance with above mentioned tools, and these are some vulnerable captures taken from OWSAP ZAP.

The absence of Anti-CSRF (Cross-Site Request Forgery) tokens is a common vulnerability that leaves web applications exposed to unauthorized actions. CSRF attacks occur when a malicious actor tricks a victim into performing unintended actions on a web application.

In detail, a CSRF attack involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user.

The absence of Anti-CSRF tokens may lead to a Cross-Site Request Forgery attack that can result in executing a specific application action as another logged in user, e.g., steal their account by changing their email and password or silently adding a new admin user account when executed from the administrator account.

To mitigate this, it's recommended to use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. Also, generating a unique nonce for each form, placing the nonce into the form, and verifying the nonce upon receipt of the form can help. Be sure that the nonce is not predictable.

Remember, the absence of Anti-CSRF tokens can be dangerous, and it's crucial to ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

This what I found from OWSAP ZAP from Airbnb.com

https://www.airbnb.com (1)

Absence of Anti-CSRF Tokens (1)

▼ GET https://www.airbnb.com/

Alert tags

- OWASP_2021_A01
- WSTG-v42-SESS-05
- OWASP_2017_A05

Alert description

No Anti-CSRF tokens were found in a HTML submission form.

A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

CSRF attacks are effective in a number of situations, including:

* The victim has an active session on the target site.

* The victim is authenticated via HTTP auth on the target site.

* The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

Other info	No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "bigsearch-query-location-input" "refinement_paths[]"].
Request	▸ Request line and header section (422 bytes) ▾ Request body (0 bytes)
Response	▸ Status line and header section (7104 bytes) ▸ Response body (654208 bytes)
Evidence	`<form class="fl14qjlg atm_gi_xjk4d9 atm_j3_1an8f3t dir dir-ltr" action="/s/homes" method="get" role="search">`
Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. Phase: Implementation Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script. Phase: Architecture and Design Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). Note that this can be bypassed using XSS. Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation. Note that this can be bypassed using XSS. Use the ESAPI Session Management control. This control includes a component for CSRF. Do not use the GET method for any request that triggers a state change. Phase: Implementation Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

Chapter 2: Figma.com

The "**Content Security Policy (CSP) Header Not Set**" is a common issue in web security. This happens when a web server is not configured to return the Content-Security-Policy HTTP header.

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, to malware distribution. CSP is designed to be fully backward compatible.

If the site doesn't offer the CSP header, browsers use the standard same origin policy. To enable CSP, you need to configure your web server to return the Content-Security-Policy HTTP header. Alternatively, the <meta> element can be used to configure a policy.

A primary goal of CSP is to mitigate and report XSS attacks. XSS attacks exploit the browser's trust in the content received from the server. CSP makes it possible for server administrators to reduce or eliminate the vectors by which XSS can occur by specifying the domains that the browser should consider to be valid sources of executable scripts.

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 page3. This can be done in the server configuration file.

Remember, the absence of the Content-Security-Policy header can leave your application vulnerable to certain types of attacks. Therefore, it's crucial to ensure that your application is properly configured to return this header.

I have captured the result of OWSAP ZAP for figma.com.

Content Security Policy (CSP) Header Not Set (1)

▾ GET https://www.figma.com/api/user

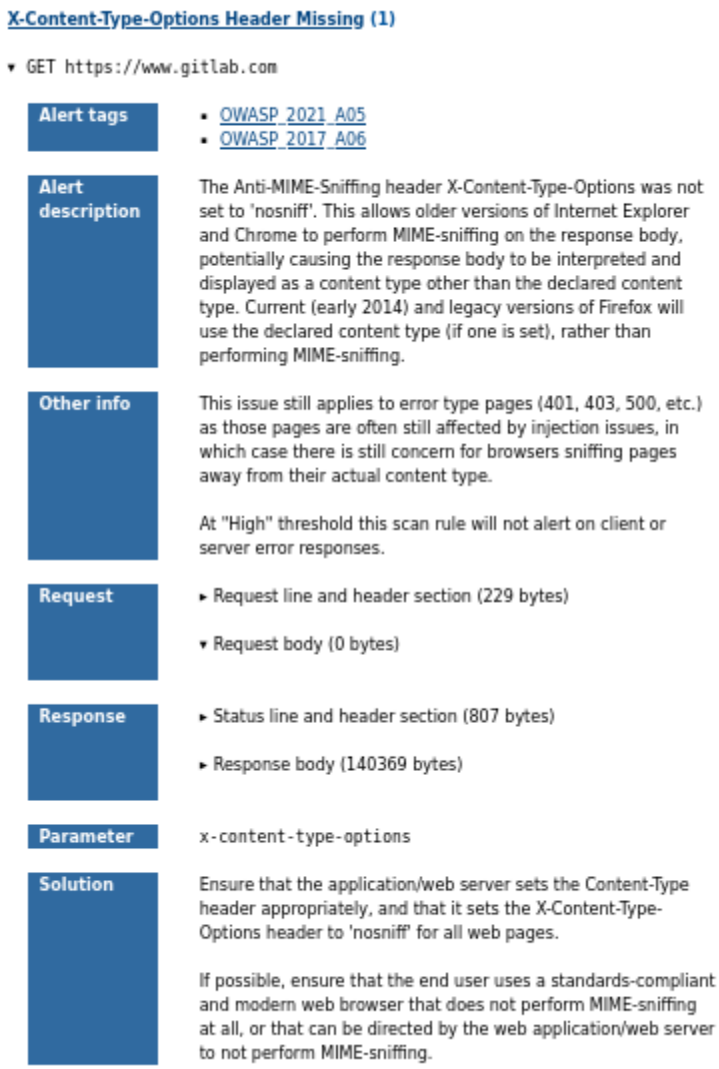
Alert tags	▪ OWASP_2021_A05 ▪ OWASP_2017_A06
Alert description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft 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.
Request	▸ Request line and header section (661 bytes) ▾ Request body (0 bytes)
Response	▸ Status line and header section (642 bytes) ▸ Response body (0 bytes)
Solution	Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Chapter 3: gitlab.com

The "X-Content-Type-Options" is a security header that tells the browser to stop automatically interpreting files as a different content type to what is stated in the provided HTTP headers. This header helps to protect against attacks such as drive-by downloads and site scripting.

When the "X-Content-Type-Options" header is missing, the browser is free to interpret the file as it sees fit, potentially leading to security vulnerabilities. For example, if a file is served with a Content-Type of "text/plain", but contains HTML and JavaScript code, a browser might automatically interpret it as an HTML file and execute the JavaScript code.

To mitigate this issue, you should ensure that all responses from your server include the "X-Content-Type-Options: nosniff" header. This will instruct the browser to strictly adhere to the stated content type and not make any assumptions.



Chapter 4: grammerly.com

The "Missing Anti-Clickjacking Header" issue refers to the absence of certain security headers that protect against clickjacking attacks. Clickjacking is a type of attack where users are tricked into clicking on something they didn't intend to, usually by overlaying a transparent layer on top of a legitimate website.

The most common way to protect 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.

The two primary headers used to prevent clickjacking are:

- Content-Security-Policy with the 'frame-ancestors' directive

- X-Frame-Options

The X-Frame-Options header has three possible values:

- DENY: Prevents any website from embedding your website in a frame or iframe.

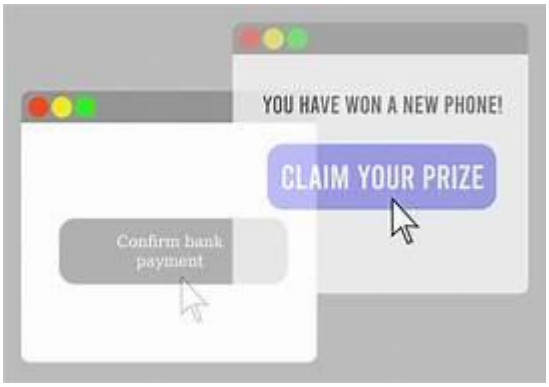
- SAMEORIGIN: Allows your website to be embedded in a frame or iframe only if the requesting website is from the same origin as your website.

- ALLOW-FROM URI: Allows your website to be embedded in a frame or iframe only if the requesting website is the specified URI.

If your vulnerability scanner has identified a missing anti-clickjacking header, it means that your website is not currently protected against clickjacking attacks. To fix this, you need to add the appropriate header to your website's HTTP responses.

Remember, while these headers can provide a strong line of defense against clickjacking, they are not a silver bullet. Your application should also be secure against other types of attacks.

Below image shows click-jacking sample.



Missing Anti-clickjacking Header (1)

▼ GET https://www.grammarly.com/upgrade/business/try

Alert tags

- OWASP_2021_A05
- WSTG-v42-CLNT-09
- OWASP_2017_A06

Alert description

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

Request

▼ Request line and header section (721 bytes)

```
GET https://www.grammarly.com/upgrade/business/try HTTP/1.1
host: www.grammarly.com
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0
Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: https://www.grammarly.com/business/pricing
Cookie: grauth=AABNc-
km_5BxXPzqdbD4WcpusFHbXsFj9JvwP5203JA1UdMWJypyk_EAUV2p115xq
zh2KA-hEyIx8AJj; csrf-
token=AABNc1l8bMEs8Z1t5gtDx884Othw08haknKZYg;
gnar_containerId=fbuk8rlm6o3e0h02; funnelType=free;
redirect_location=eyJ0eXBlIjoiIiwibG9jYXRpb24iOiJodHRwczovL
3d3dy5ncmFtbWFybHkuY29tL3NpZ251cD9mbG93X3NvdXJjZT1jaXRhdGlv
bl9nZW5lcmF0b3IifQ==;
browser_info=CHROME:116:COMPUTER:SUPPORTED:FREEMIUM:WINDOWS
_10:WINDOWS
```

▼ Request body (0 bytes)

Response

▶ Status line and header section (1539 bytes)

▶ Response body (279305 bytes)

Parameter

x-frame-options

Solution

Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app.

If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

Chapter 5: paypal.com

Cookie poisoning, also known as session hijacking, is a type of attack where an attacker alters, forges, hijacks, or otherwise "poisons" a valid cookie sent back to a server. This is done to steal data, bypass security measures, or both.

In computing, a cookie is data specific to a website and user session, including interest or identity information about the user, that is created and stored in the user's browser. Websites and servers use cookies to track usage trends, customize and streamline the user experience.

Attackers can intercept cookies before they return to the server to extract information from or modify them. Forged cookies can also be created from scratch as a means of impersonating a user to access additional user data. Cookies are often used for authentication and to track whether a user is logged in to an account, which means they contain information that can be used for unauthorized access. They can also contain other sensitive data, including financial information, entered by a user.

Cookie poisoning is relatively easy for attackers, who can use a poisoned cookie to steal user identities for fraud or to gain unauthorized access to the web server for further exploits. To prevent cookie poisoning, it's recommended to use secure methods for generating and transmitting cookies. Cross-site scripting (XSS) is a common way to steal cookies, but several methods, including packet sniffing and brute force, may be used to gain unauthorized access to cookies.

While thoughtful app development can limit the sensitive data stored in cookies or make it harder for attackers to extract, a cookie's purpose is to identify users, behaviors, or both. That means applications will continue to use them. Appropriate web application security and session management, which may be provided by a web application firewall (WAF), can help protect identifying data and defend against cookie poisoning.

https://www.paypal.com (5)

Cookie Poisoning (1)

▾ GET https://www.paypal.com/webapps/mpp/preview/how-to-turn-on-javascript?locale.x=en_US

Alert tags

- OWASP_2021_A03
- OWASP_2017_A01

Alert description

This check looks at user-supplied input in query string parameters and POST data to identify where cookie parameters might be controlled. This is called a cookie poisoning attack, and becomes exploitable when an attacker can manipulate the cookie in various ways. In some cases this will not be exploitable, however, allowing URL parameters to set cookie values is generally considered a bug.

Other info

An attacker may be able to poison cookie values through URL parameters. Try injecting a semicolon to see if you can add cookie values (e.g. name=controlledValue; name=anotherValue;).

This was identified at:

https://www.paypal.com/webapps/mpp/preview/how-to-turn-on-javascript?locale.x=en_US

User-input was found in the following cookie:

LANG=en_US;LK; Max-Age=31556; Domain=.paypal.com; Path=/; Expires=Sat, 27 Apr 2024 20:27:38 GMT; HttpOnly; Secure; SameSite=None

The user input was:

locale.x=en_US

Request

▸ Request line and header section (2325 bytes)

▸ Request body (0 bytes)

Response

▸ Status line and header section (4342 bytes)

▸ Response body (36019 bytes)

Parameter

`locale.x`

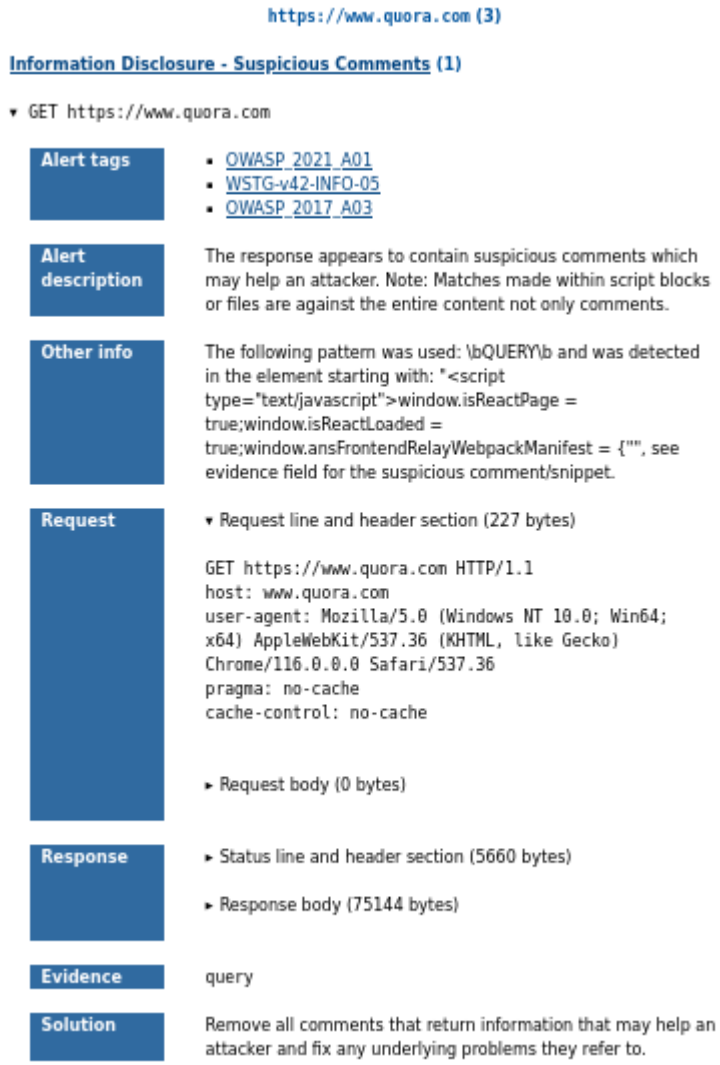
Solution

Do not allow user input to control cookie names and values. If some query string parameters must be set in cookie values, be sure to filter out semicolon's that can serve as name/value pair delimiters.

Chapter 6: quora.com

Quora is a social question-and-answer website that serves as an online knowledge market, where users can ask questions, share insights, and learn from one another. Founded in 2009, it's a platform where knowledge is shared and grown, connecting individuals with different perspectives to foster understanding and empower everyone to contribute their expertise.

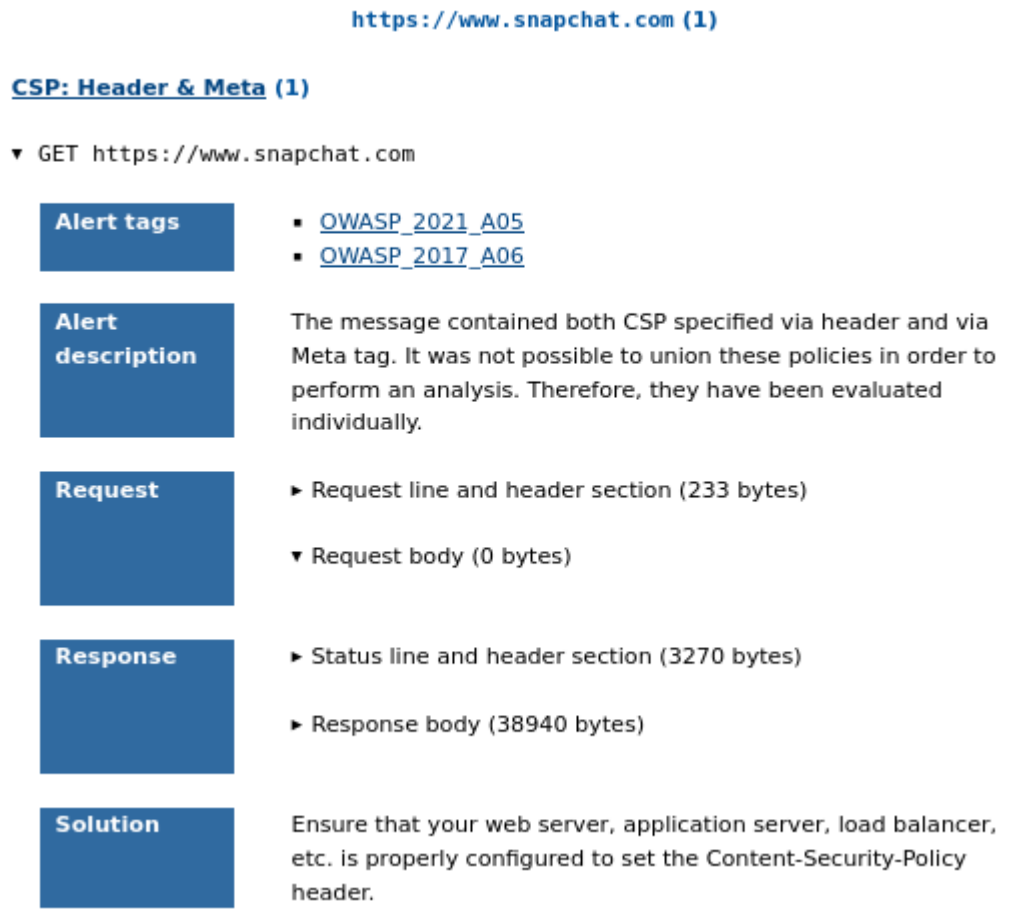
From Quora I found some informational alerts as shown in image.



Chapter 7: snapchat.com

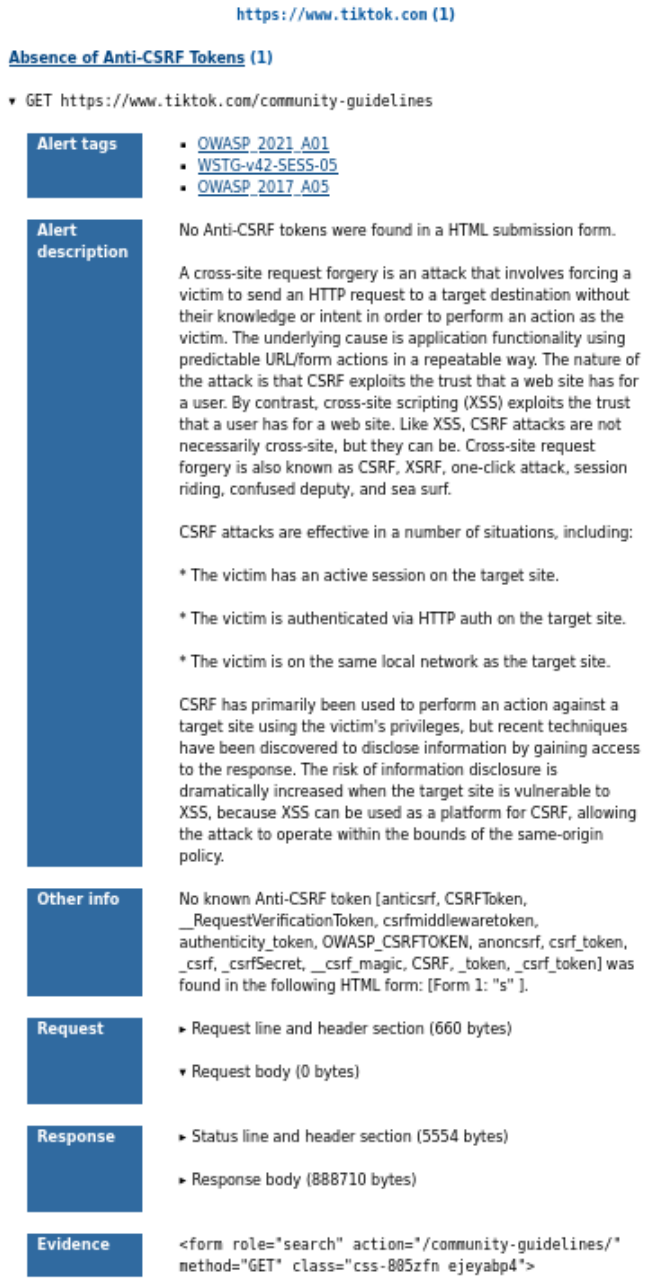
Snapchat is a dynamic social media platform that specializes in multimedia messaging. It allows users to send 'Snaps'—photos and videos that disappear after being viewed—and 'Stories', which are Snaps visible to all followers for 24 hours. Snapchat also offers a range of creative tools like filters, lenses, and Bitmojis to enhance user interaction.

From snapchat I found some informational alerts as shown in image.



Chapter 8: tiktok.com

TikTok is a leading platform for short-form mobile videos, aiming to inspire creativity and bring joy to its users. It's a global hub for capturing and sharing life's moments, fostering a community where everyone can be a creator.



https://www.tiktok.com (1)

Absence of Anti-CSRF Tokens (1)

▾ GET https://www.tiktok.com/community-guidelines

Alert tags

- OWASP_2021_A01
- WSTG-v42-SESS-05
- OWASP_2017_A05

Alert description

No Anti-CSRF tokens were found in a HTML submission form.

A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

CSRF attacks are effective in a number of situations, including:

* The victim has an active session on the target site.

* The victim is authenticated via HTTP auth on the target site.

* The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

Other info

No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "s"].

Request

▸ Request line and header section (660 bytes)

▾ Request body (0 bytes)

Response

▸ Status line and header section (5554 bytes)

▸ Response body (888710 bytes)

Evidence

```
<form role="search" action="/community-guidelines/"
method="GET" class="css-805zfn ejeyabp4">
```

Chapter 9: tinder.com.

Tinder is a global online dating platform that allows users to meet new people, expand their social network, and connect with locals in over 190 countries. Launched in 2012, it has become one of the world's most popular dating apps. When auditing this site OWSAP ZAP able to find some informational alerts.

The **Strict-Transport-Security header**, also known as HTTP Strict Transport Security (HSTS), is a security feature that helps protect websites against protocol downgrade attacks and cookie hijacking. It allows web servers to declare that web browsers (or other complying user agents) should only interact with it using secure HTTPS connections, and never via the insecure HTTP protocol.

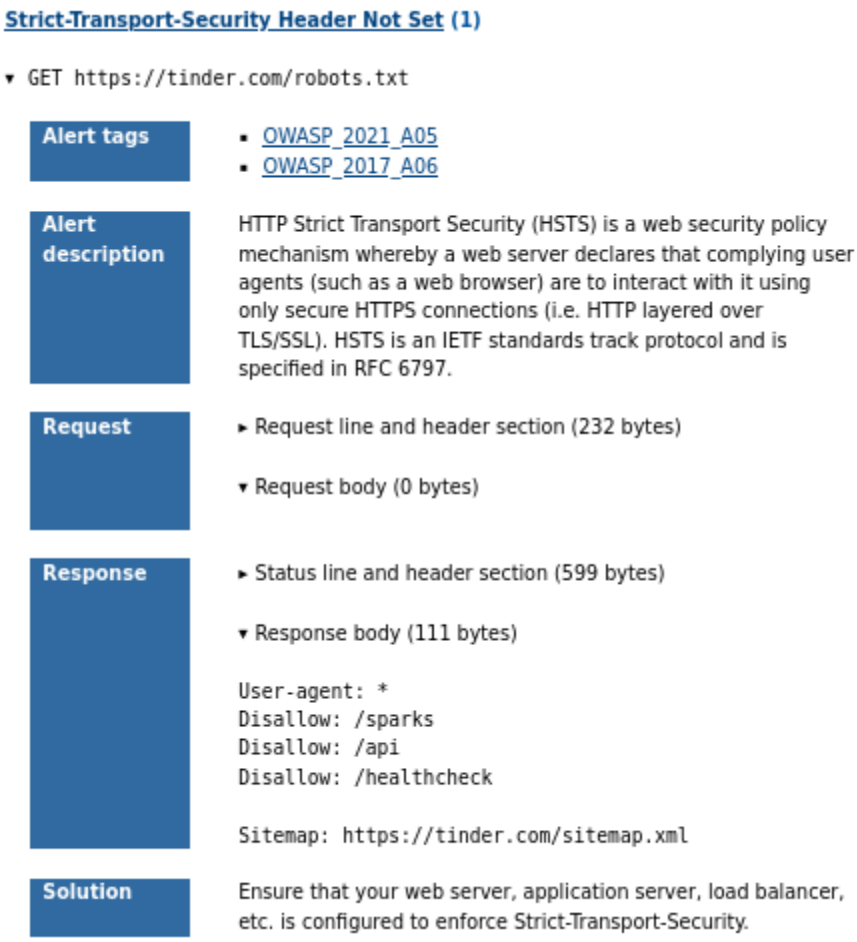
When the Strict-Transport-Security header is not set, it means that the website is not enforcing the use of HTTPS. This can leave the website vulnerable to attacks such as man-in-the-middle attacks, where attackers can intercept and alter the communication between the user and the server. This could potentially lead to sensitive information being exposed, such as login credentials or personal data.

Therefore, it's recommended for websites to set the Strict-Transport-Security header to ensure all communication is securely transmitted over HTTPS. This is particularly important for websites that handle sensitive data. However, it's worth noting that implementing HSTS should be done with care, as incorrectly configured settings could potentially lock users out.

For example, a typical HSTS header might look like this:

Strict-Transport-Security: max-age=63072000; includeSubDomains; preload

This header tells the browser to always use HTTPS for the domain for the next two years (as specified by max-age=63072000, which is in seconds), including all subdomains (include Subdomains), and that the domain is ready to be included in the HSTS preload list (preload). The preload list is a list of sites that are hardcoded into browsers as being HTTPS only. If the Strict-Transport-Security header is not set, it's a potential security risk and should be addressed to enhance the security posture of the website.



Chapter 10: uber.com.

Uber.com is the official website of Uber Technologies, Inc., an American multinational company that provides various services worldwide. The platform connects drivers with riders, enabling users to request rides and drivers to earn money based on completed trips.

As found on figma site OWSAP ZAP able to find Content-Security-Policy HTTP header not set in uber site too. However it is informational alert only.

Conclusion

The web application, along with its associated authorization gateway, showcases a commendable level of upkeep and control, particularly from a cybersecurity standpoint. While there have been occurrences of information exposure, these can be suitably handled, and their corresponding risk quotient is evaluated to be minimal. It's worth noting that all detected security weaknesses fell into the categories of either being Informational or of Low severity. Moreover, the vulnerabilities that were categorized as Low are not readily exploitable, as has been evidenced in the past.

It's also important to highlight that the application's security measures are regularly updated to keep up with the evolving threat landscape. Regular security audits are conducted to identify and rectify any potential vulnerabilities. The application also employs a robust encryption mechanism to protect sensitive data and ensure secure communication. Additionally, the application's user authentication process is designed to prevent unauthorized access, further enhancing its security posture. Despite the presence of some low-level vulnerabilities, the overall security of the web application and its authorization portal is well-maintained and managed. This reflects the commitment to security and the proactive approach taken towards identifying and addressing potential security issues.