<u>TOPIC</u>: Vulnerability Assessment and Penetration Testing

---

<u>PROJECT</u>: Vulnerability Assessment and Penetration Testing in Web Applications and Websites.

<u>Problem Statement</u>: Find the vulnerabilities and submit the report to the stakeholders – VAPT Team.

Reported Issued: 2 – 8 – 2024

Submitted By: D Vardhan Reddy
B Anjali

Table Of Contents

Executive Summary:

We performed a security assessment on three different web applications. The purpose of this assessment was to discover and identify vulnerabilities in the four websites' infrastructure suggest methods to remediate thevulnerabilities and identify a total of three vulnerabilities within the scope ofthe engagement which are broken down by severity in the table below.

Vulnerability Severity	No. Of Vulnerability found
Critical	0
High	4
Medium	1
Low	3

Scope:

Security assessment includes testing for security loopholes in the scope defined below nothing was assumed at the start of the security assessment. The following are the websites on which testing is done.

Website 1: https://www.5movierulz.pro/

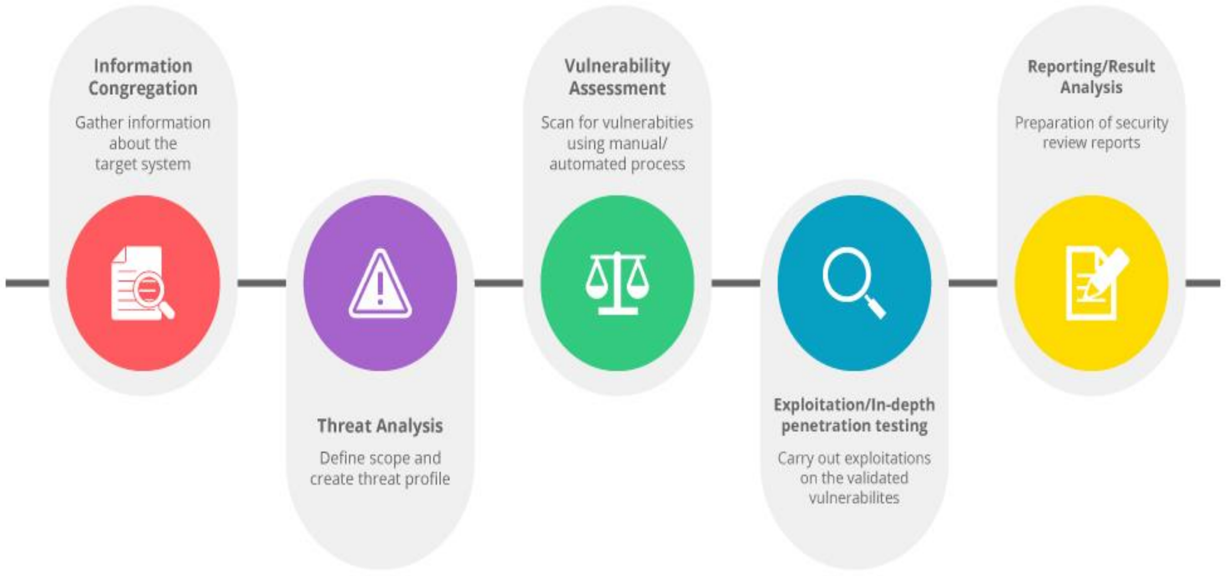
Website 2: https://www.mygov.in/

Website 3: https://ghostvolt.com/

Website 4: https://temp-mail.org/en/

Testing Methodology

The following image is a graphical representation of the methodology.


Information Congregation
Gather information about the target system
Threat Analysis
Define scope and create threat profile
Vulnerability Assessment
Scan for vulnerabities using manual/ automated process
Exploitation/In-depth penetration testing
Carry out exploitations on the validated vulnerabilites
Reporting/Result Analysis
Preparation of security review reports

1. Information Congregation (Reconnaissance):

It is also called as information-gathering phase. The information-gathering phase is a crucial step in the VAPT methodology. It involves collecting as much relevant data about the target system or network as possible. This phase sets the foundation for subsequent phases, as it helps in understanding the target environment and identifying potential vulnerabilities.

2. Threat Analysis:

The Threat Analysis phase in Vulnerability Assessment and Penetration Testing (VAPT) involves evaluating the gathered information to identify and assess potential threats to the target system or network. This phase aims to understand the nature of possible attacks, their impact, and the likelihood of their occurrence.

3. Vulnerability Assessment:

The Vulnerability Assessment phase in Vulnerability Assessment and Penetration Testing (VAPT) involves identifying, evaluating, and prioritizing vulnerabilities within the target environment. This phase aims to provide a comprehensive understanding of the security weaknesses that exist in systems, networks, and applications.

4. Exploitation/ In-depth penetration testing:

The Exploitation or In-depth Penetration Testing phase in Vulnerability Assessment and Penetration Testing (VAPT) involves actively attempting to exploit identified vulnerabilities to determine the potential impact and effectiveness of security controls. This phase helps in understanding the real-world implications of vulnerabilities and provides valuable insights into the security posture of the target environment.

5. Reporting/result analysis:

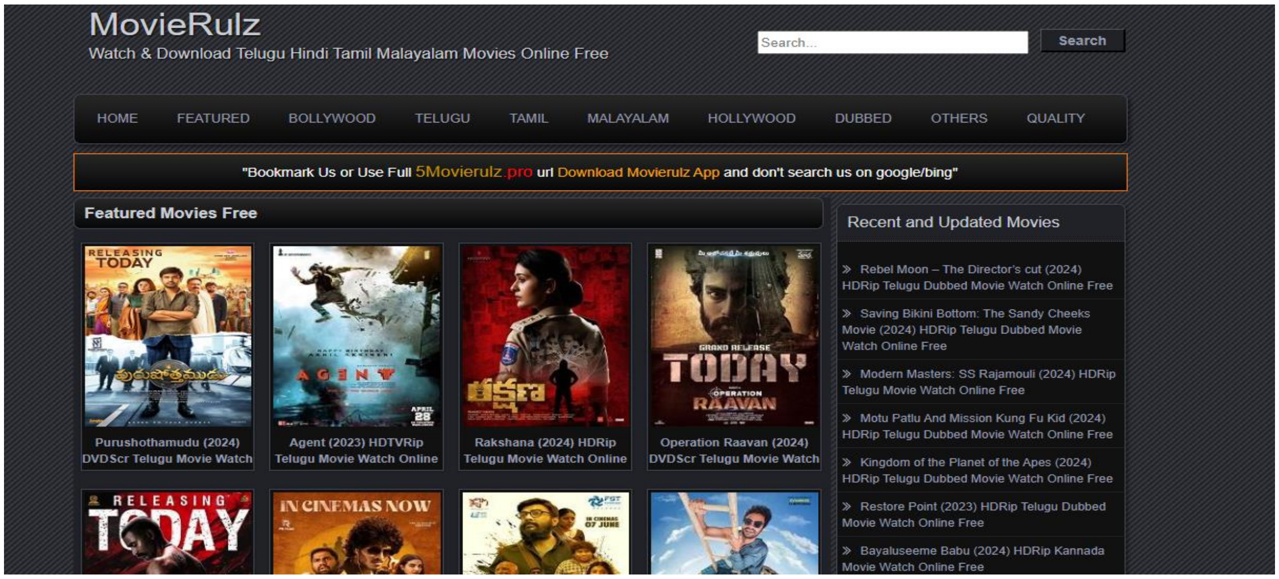
It is the last step. The Reporting/Result Analysis phase in Vulnerability Assessment and Penetration Testing (VAPT) is essential for documenting the findings, analyzing the results, and communicating the outcomes to stakeholders. This phase provides a comprehensive overview of the vulnerabilities discovered, their potential impact, and recommendations for remediation.

Assessment Findings

Vulnerability 1:

High Risk	
Name of Vulnerability	XSS (Cross-site scripting)

URL: https://www.5movierulz.pro/

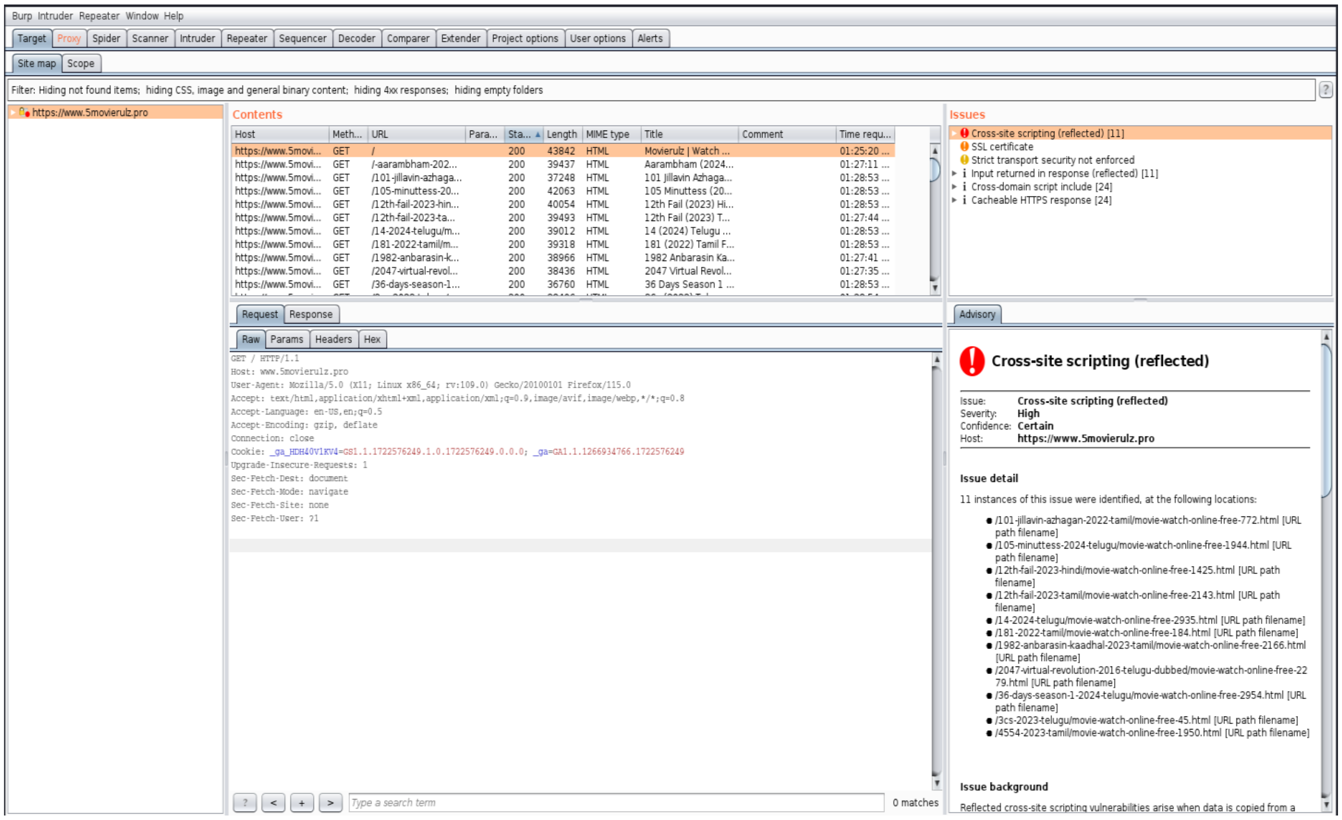

MovieRulz
Watch & Download Telugu Hindi Tamil Malayalam Movies Online Free
Search...
Search
HOME
FEATURED
BOLLYWOOD
TELUGU
TAMIL
MALAYALAM
HOLLYWOOD
DUBBED
OTHERS
QUALITY
"Bookmark Us or Use Full 5Movierulz.pro url Download Movierulz App and don't search us on google/bing"
Featured Movies Free
Purushothamudu (2024) DVDScr Telugu Movie Watch
Agent (2023) HDTVRip Telugu Movie Watch Online
Rakshana (2024) HDRip Telugu Movie Watch Online
Operation Raavan (2024) DVDScr Telugu Movie Watch
Recent and Updated Movies
Rebel Moon – The Director's cut (2024) HDRip Telugu Dubbed Movie Watch Online Free
Saving Bikini Bottom: The Sandy Cheeks Movie (2024) HDRip Telugu Dubbed Movie Watch Online Free
Modern Masters: SS Rajamouli (2024) HDRip Telugu Movie Watch Online Free
Motu Patlu And Mission Kung Fu Kid (2024) HDRip Telugu Dubbed Movie Watch Online Free
Kingdom of the Planet of the Apes (2024) HDRip Telugu Dubbed Movie Watch Online Free
Restore Point (2023) HDRip Telugu Dubbed Movie Watch Online Free
Bayaluseeme Babu (2024) HDRip Kannada Movie Watch Online Free

(XSS) Cross-Site Scripting Vulnerability:

Cross-site scripting (XSS) is a security vulnerability typically found in web applications. It allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can then be executed in the context of the user's browser, potentially leading to various malicious activities such as stealing cookies, session tokens, or other sensitive information or defacing websites

Movierulz is a website known for providing unauthorized access to movies and TV shows, often involving illegal distribution of copyrighted content.

We are using the **Burp suite professional tool** for performing the Vulnerability Assessment.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Site map Scope
Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders
https://www.5movierulz.pro
Contents
Request Response
Raw Params Headers Hex
Type a search term
0 matches
Advisory

Host	Meth...	URL	Para...	Sta...	Length	MIME type	Title	Comment	Time requ...
https://www.5movi...	GET	/		200	43842	HTML	Movierulz | Watch ...		01:25:20 ...
https://www.5movi...	GET	/-aarambham-202...		200	39437	HTML	Aarambham (2024...		01:27:11 ...
https://www.5movi...	GET	/101-jillavin-azhaga...		200	37248	HTML	101 Jillavin Azhaga...		01:28:53 ...
https://www.5movi...	GET	/105-minuttess-20...		200	42063	HTML	105 Minuttess (20...		01:28:53 ...
https://www.5movi...	GET	/12th-fail-2023-hin...		200	40054	HTML	12th Fail (2023) Hi...		01:28:53 ...
https://www.5movi...	GET	/12th-fail-2023-ta...		200	39493	HTML	12th Fail (2023) T...		01:27:44 ...
https://www.5movi...	GET	/14-2024-telugu/m...		200	39012	HTML	14 (2024) Telugu ...		01:28:53 ...
https://www.5movi...	GET	/181-2022-tamil/m...		200	39318	HTML	181 (2022) Tamil F...		01:28:53 ...
https://www.5movi...	GET	/1982-anbarasin-k...		200	38966	HTML	1982 Anbarasin Ka...		01:27:41 ...
https://www.5movi...	GET	/2047-virtual-revol...		200	38436	HTML	2047 Virtual Revol...		01:27:35 ...
https://www.5movi...	GET	/36-days-season-1...		200	36760	HTML	36 Days Season 1 ...		01:28:53 ...

```
GET / HTTP/1.1
Host: www.5movierulz.pro
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: _ga_HDH40V1KV4=GS1.1.1722576249.1.0.1722576249.0.0.0; _ga=GA1.1.1266934766.1722576249
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

Issues

- Cross-site scripting (reflected) [11]
- SSL certificate
- Strict transport security not enforced
- Input returned in response (reflected) [11]
- Cross-domain script include [24]
- Cacheable HTTPS response [24]

Cross-site scripting (reflected)

Issue:	**Cross-site scripting (reflected)**
Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.5movierulz.pro**

Issue detail

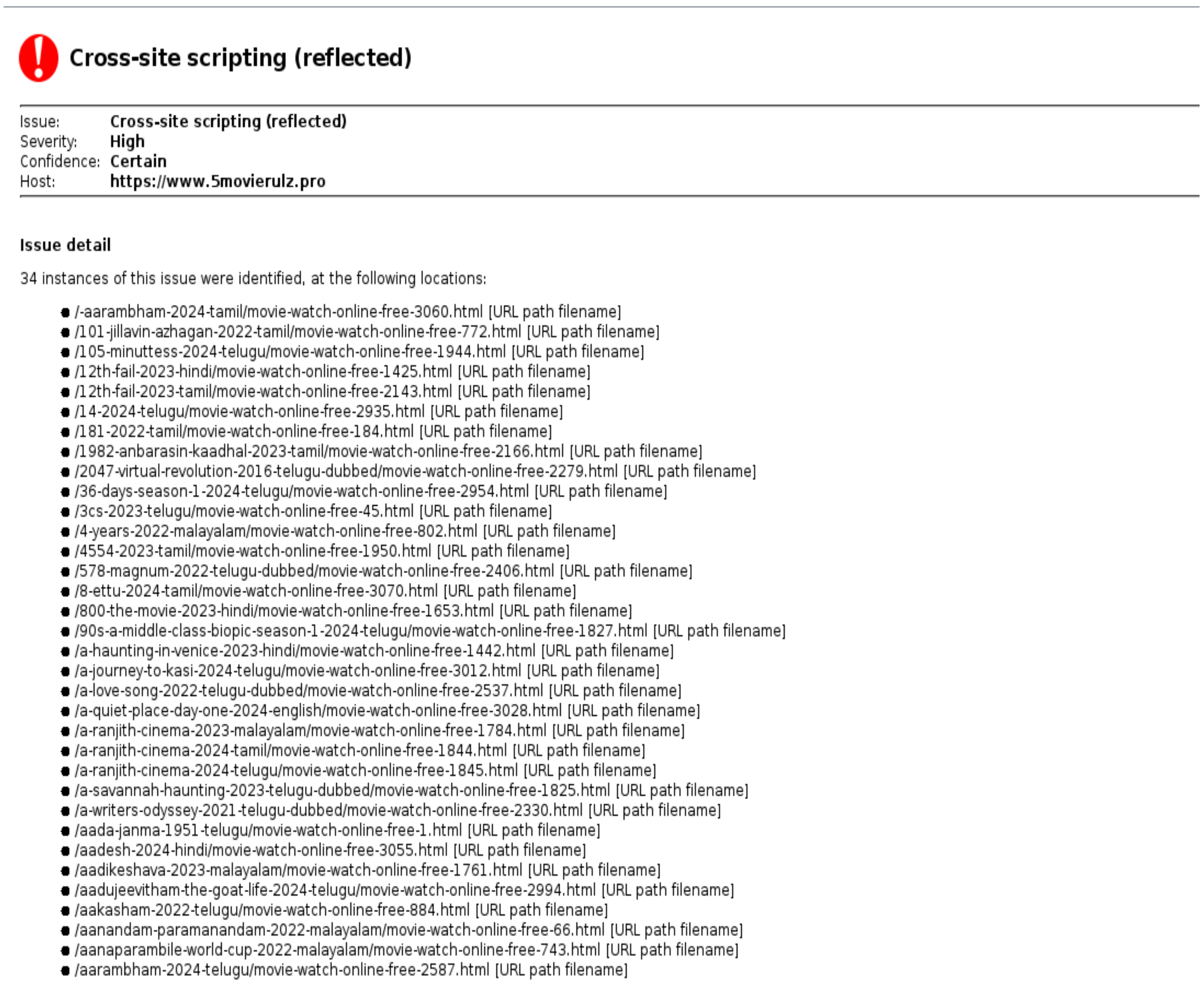
11 instances of this issue were identified, at the following locations:

- /101-jillavin-azhagan-2022-tamil/movie-watch-online-free-772.html [URL path filename]
- /105-minuttess-2024-telugu/movie-watch-online-free-1944.html [URL path filename]
- /12th-fail-2023-hindi/movie-watch-online-free-1425.html [URL path filename]
- /12th-fail-2023-tamil/movie-watch-online-free-2143.html [URL path filename]
- /14-2024-telugu/movie-watch-online-free-2935.html [URL path filename]
- /181-2022-tamil/movie-watch-online-free-184.html [URL path filename]
- /1982-anbarasin-kaadhal-2023-tamil/movie-watch-online-free-2166.html [URL path filename]
- /2047-virtual-revolution-2016-telugu-dubbed/movie-watch-online-free-2279.html [URL path filename]
- /36-days-season-1-2024-telugu/movie-watch-online-free-2954.html [URL path filename]
- /3cs-2023-telugu/movie-watch-online-free-45.html [URL path filename]
- /4554-2023-tamil/movie-watch-online-free-1950.html [URL path filename]

Issue background

Reflected cross-site scripting vulnerabilities arise when data is copied from a

On the Movierulz website, a total of 34 distinct instances of cross-site scripting (XSS) vulnerabilities were identified. These vulnerabilities span various parts of the site, indicating widespread security issues that could potentially affect user data and overall site integrity.



Cross-site scripting (reflected)

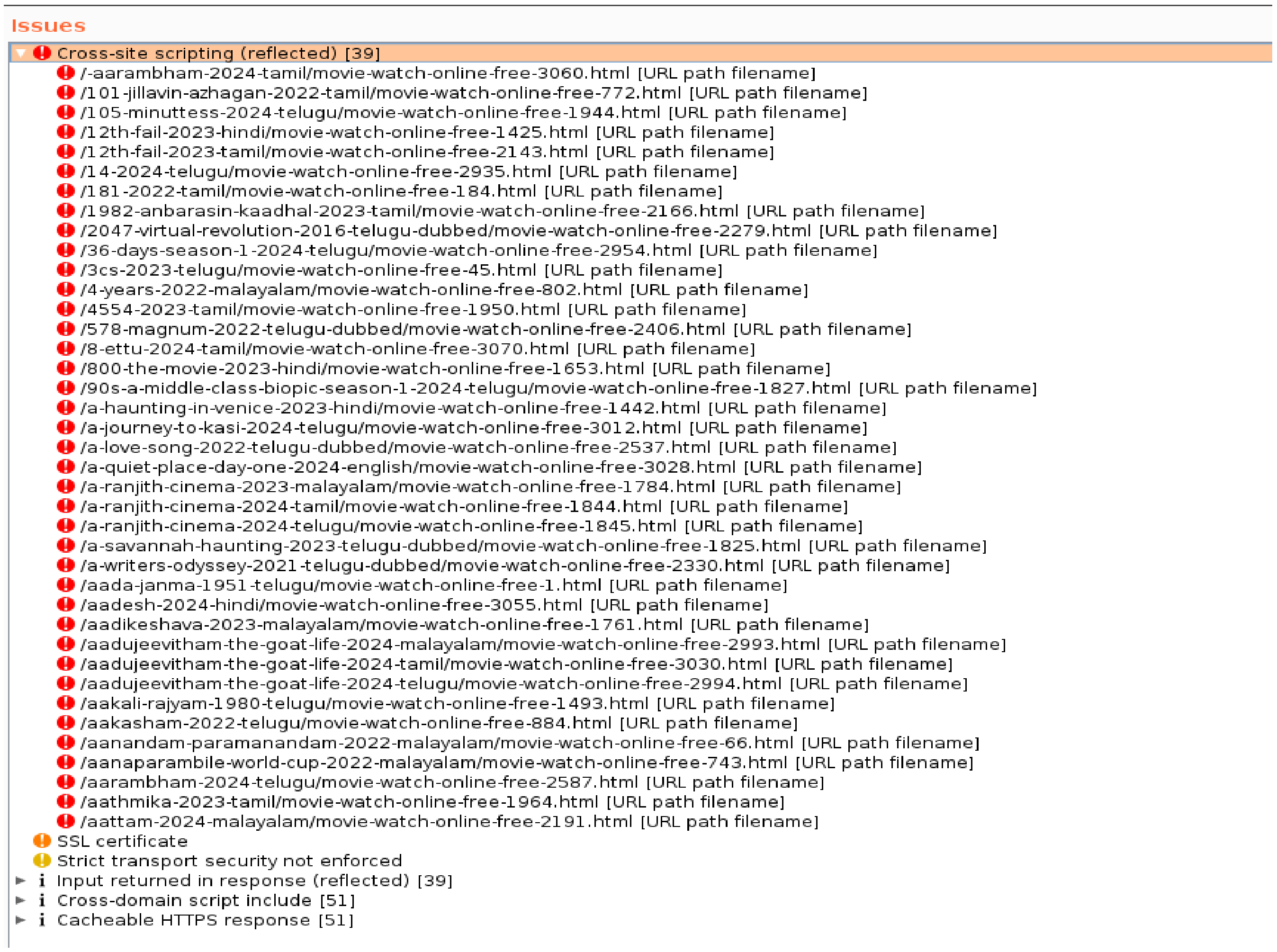
Issue:	**Cross-site scripting (reflected)**
Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.5movierulz.pro**

Issue detail

34 instances of this issue were identified, at the following locations:

- /-aarambham-2024-tamil/movie-watch-online-free-3060.html [URL path filename]
- /101-jillavin-azhagan-2022-tamil/movie-watch-online-free-772.html [URL path filename]
- /105-minuttess-2024-telugu/movie-watch-online-free-1944.html [URL path filename]
- /12th-fail-2023-hindi/movie-watch-online-free-1425.html [URL path filename]
- /12th-fail-2023-tamil/movie-watch-online-free-2143.html [URL path filename]
- /14-2024-telugu/movie-watch-online-free-2935.html [URL path filename]
- /181-2022-tamil/movie-watch-online-free-184.html [URL path filename]
- /1982-anbarasin-kaadhal-2023-tamil/movie-watch-online-free-2166.html [URL path filename]
- /2047-virtual-revolution-2016-telugu-dubbed/movie-watch-online-free-2279.html [URL path filename]
- /36-days-season-1-2024-telugu/movie-watch-online-free-2954.html [URL path filename]
- /3cs-2023-telugu/movie-watch-online-free-45.html [URL path filename]
- /4-years-2022-malayalam/movie-watch-online-free-802.html [URL path filename]
- /4554-2023-tamil/movie-watch-online-free-1950.html [URL path filename]
- /578-magnum-2022-telugu-dubbed/movie-watch-online-free-2406.html [URL path filename]
- /8-ettu-2024-tamil/movie-watch-online-free-3070.html [URL path filename]
- /800-the-movie-2023-hindi/movie-watch-online-free-1653.html [URL path filename]
- /90s-a-middle-class-biopic-season-1-2024-telugu/movie-watch-online-free-1827.html [URL path filename]
- /a-haunting-in-venice-2023-hindi/movie-watch-online-free-1442.html [URL path filename]
- /a-journey-to-kasi-2024-telugu/movie-watch-online-free-3012.html [URL path filename]
- /a-love-song-2022-telugu-dubbed/movie-watch-online-free-2537.html [URL path filename]
- /a-quiet-place-day-one-2024-english/movie-watch-online-free-3028.html [URL path filename]
- /a-ranjith-cinema-2023-malayalam/movie-watch-online-free-1784.html [URL path filename]
- /a-ranjith-cinema-2024-tamil/movie-watch-online-free-1844.html [URL path filename]
- /a-ranjith-cinema-2024-telugu/movie-watch-online-free-1845.html [URL path filename]
- /a-savannah-haunting-2023-telugu-dubbed/movie-watch-online-free-1825.html [URL path filename]
- /a-writers-odyssey-2021-telugu-dubbed/movie-watch-online-free-2330.html [URL path filename]
- /aada-janma-1951-telugu/movie-watch-online-free-1.html [URL path filename]
- /aadesh-2024-hindi/movie-watch-online-free-3055.html [URL path filename]
- /aadikeshava-2023-malayalam/movie-watch-online-free-1761.html [URL path filename]
- /aadujeevitham-the-goat-life-2024-telugu/movie-watch-online-free-2994.html [URL path filename]
- /aakasham-2022-telugu/movie-watch-online-free-884.html [URL path filename]
- /aanandam-paramanandam-2022-malayalam/movie-watch-online-free-66.html [URL path filename]
- /aanaparambile-world-cup-2022-malayalam/movie-watch-online-free-743.html [URL path filename]
- /aarambham-2024-telugu/movie-watch-online-free-2587.html [URL path filename]

The following list provides a comprehensive overview of the various cross-site scripting (XSS) issues currently identified on the website. These issues include multiple instances where malicious scripts can be injected and executed, affecting user input fields, URL parameters, and dynamically generated content. Each identified vulnerability represents a potential security risk that could allow attackers to steal sensitive information, manipulate user sessions, or compromise overall site integrity.



```
Issues
▼ Cross-site scripting (reflected) [39]
    /-aarambham-2024-tamil/movie-watch-online-free-3060.html [URL path filename]
    /101-jillavin-azhagan-2022-tamil/movie-watch-online-free-772.html [URL path filename]
    /105-minuttess-2024-telugu/movie-watch-online-free-1944.html [URL path filename]
    /12th-fail-2023-hindi/movie-watch-online-free-1425.html [URL path filename]
    /12th-fail-2023-tamil/movie-watch-online-free-2143.html [URL path filename]
    /14-2024-telugu/movie-watch-online-free-2935.html [URL path filename]
    /181-2022-tamil/movie-watch-online-free-184.html [URL path filename]
    /1982-anbarasin-kaadhal-2023-tamil/movie-watch-online-free-2166.html [URL path filename]
    /2047-virtual-revolution-2016-telugu-dubbed/movie-watch-online-free-2279.html [URL path filename]
    /36-days-season-1-2024-telugu/movie-watch-online-free-2954.html [URL path filename]
    /3cs-2023-telugu/movie-watch-online-free-45.html [URL path filename]
    /4-years-2022-malayalam/movie-watch-online-free-802.html [URL path filename]
    /4554-2023-tamil/movie-watch-online-free-1950.html [URL path filename]
    /578-magnum-2022-telugu-dubbed/movie-watch-online-free-2406.html [URL path filename]
    /8-ettu-2024-tamil/movie-watch-online-free-3070.html [URL path filename]
    /800-the-movie-2023-hindi/movie-watch-online-free-1653.html [URL path filename]
    /90s-a-middle-class-biopic-season-1-2024-telugu/movie-watch-online-free-1827.html [URL path filename]
    /a-haunting-in-venice-2023-hindi/movie-watch-online-free-1442.html [URL path filename]
    /a-journey-to-kasi-2024-telugu/movie-watch-online-free-3012.html [URL path filename]
    /a-love-song-2022-telugu-dubbed/movie-watch-online-free-2537.html [URL path filename]
    /a-quiet-place-day-one-2024-english/movie-watch-online-free-3028.html [URL path filename]
    /a-ranjith-cinema-2023-malayalam/movie-watch-online-free-1784.html [URL path filename]
    /a-ranjith-cinema-2024-tamil/movie-watch-online-free-1844.html [URL path filename]
    /a-ranjith-cinema-2024-telugu/movie-watch-online-free-1845.html [URL path filename]
    /a-savannah-haunting-2023-telugu-dubbed/movie-watch-online-free-1825.html [URL path filename]
    /a-writers-odyssey-2021-telugu-dubbed/movie-watch-online-free-2330.html [URL path filename]
    /aada-janma-1951-telugu/movie-watch-online-free-1.html [URL path filename]
    /aadesh-2024-hindi/movie-watch-online-free-3055.html [URL path filename]
    /aadikeshava-2023-malayalam/movie-watch-online-free-1761.html [URL path filename]
    /aadujeevitham-the-goat-life-2024-malayalam/movie-watch-online-free-2993.html [URL path filename]
    /aadujeevitham-the-goat-life-2024-tamil/movie-watch-online-free-3030.html [URL path filename]
    /aadujeevitham-the-goat-life-2024-telugu/movie-watch-online-free-2994.html [URL path filename]
    /aakali-rajyam-1980-telugu/movie-watch-online-free-1493.html [URL path filename]
    /aakasham-2022-telugu/movie-watch-online-free-884.html [URL path filename]
    /aanandam-paramanandam-2022-malayalam/movie-watch-online-free-66.html [URL path filename]
    /aanaparambile-world-cup-2022-malayalam/movie-watch-online-free-743.html [URL path filename]
    /aarambham-2024-telugu/movie-watch-online-free-2587.html [URL path filename]
    /aathmika-2023-tamil/movie-watch-online-free-1964.html [URL path filename]
    /aattam-2024-malayalam/movie-watch-online-free-2191.html [URL path filename]
  SSL certificate
  Strict transport security not enforced
► i Input returned in response (reflected) [39]
► i Cross-domain script include [51]
► i Cacheable HTTPS response [51]
```

How XSS Attacks Work:

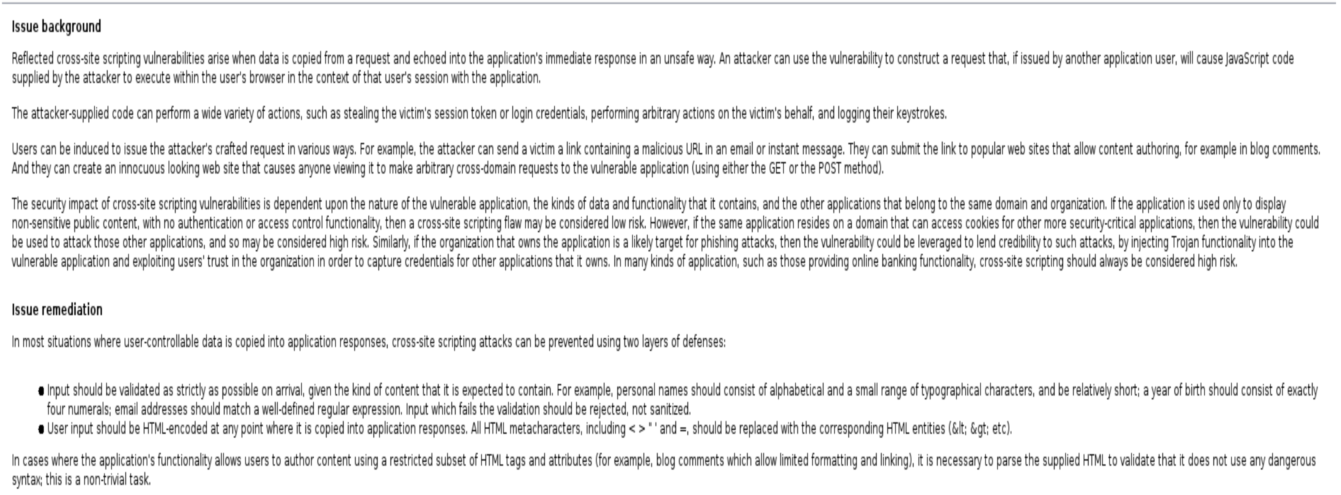
1. **Injection**: An attacker injects malicious code (typically JavaScript) into a web application.
2. **Execution**: The malicious code is executed in the context of a victim's browser when they visit a compromised page or click on a malicious link.
3. **Impact**: The attacker can steal session tokens, cookies, or other sensitive information, manipulate the content displayed to the user, or perform actions on behalf of the user.

Preventing XSS:

1. **Input Validation**: Ensure that all user inputs are validated and sanitized. This involves checking input data against a whitelist of acceptable values.
2. **Output Encoding**: Encode output data, especially when displaying user input. Use functions provided by web frameworks to automatically escape HTML, JavaScript, and URL contexts.
3. **Content Security Policy (CSP)**: Implement CSP headers to restrict the sources from which scripts can be loaded and executed.
4. **HTTPOnly Cookies**: Set the HttpOnly flag on cookies to prevent access to cookie data through JavaScript.
5. **Secure Coding Practices**: Follow secure coding guidelines and use web application frameworks that provide built-in protection against XSS.

Remedies:

The following are the remedies which are given by the burp suite tool itself.



Issue background

Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.

The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

Users can be induced to issue the attacker's crafted request in various ways. For example, the attacker can send a victim a link containing a malicious URL in an email or instant message. They can submit the link to popular web sites that allow content authoring, for example in blog comments. And they can create an innocuous looking web site that causes anyone viewing it to make arbitrary cross-domain requests to the vulnerable application (using either the GET or the POST method).

The security impact of cross-site scripting vulnerabilities is dependent upon the nature of the vulnerable application, the kinds of data and functionality that it contains, and the other applications that belong to the same domain and organization. If the application is used only to display non-sensitive public content, with no authentication or access control functionality, then a cross-site scripting flaw may be considered low risk. However, if the same application resides on a domain that can access cookies for other more security-critical applications, then the vulnerability could be used to attack those other applications, and so may be considered high risk. Similarly, if the organization that owns the application is a likely target for phishing attacks, then the vulnerability could be leveraged to lend credibility to such attacks, by injecting Trojan functionality into the vulnerable application and exploiting users' trust in the organization in order to capture credentials for other applications that it owns. In many kinds of application, such as those providing online banking functionality, cross-site scripting should always be considered high risk.

Issue remediation

In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defenses:

- Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.
- User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (&lt; &gt; etc).

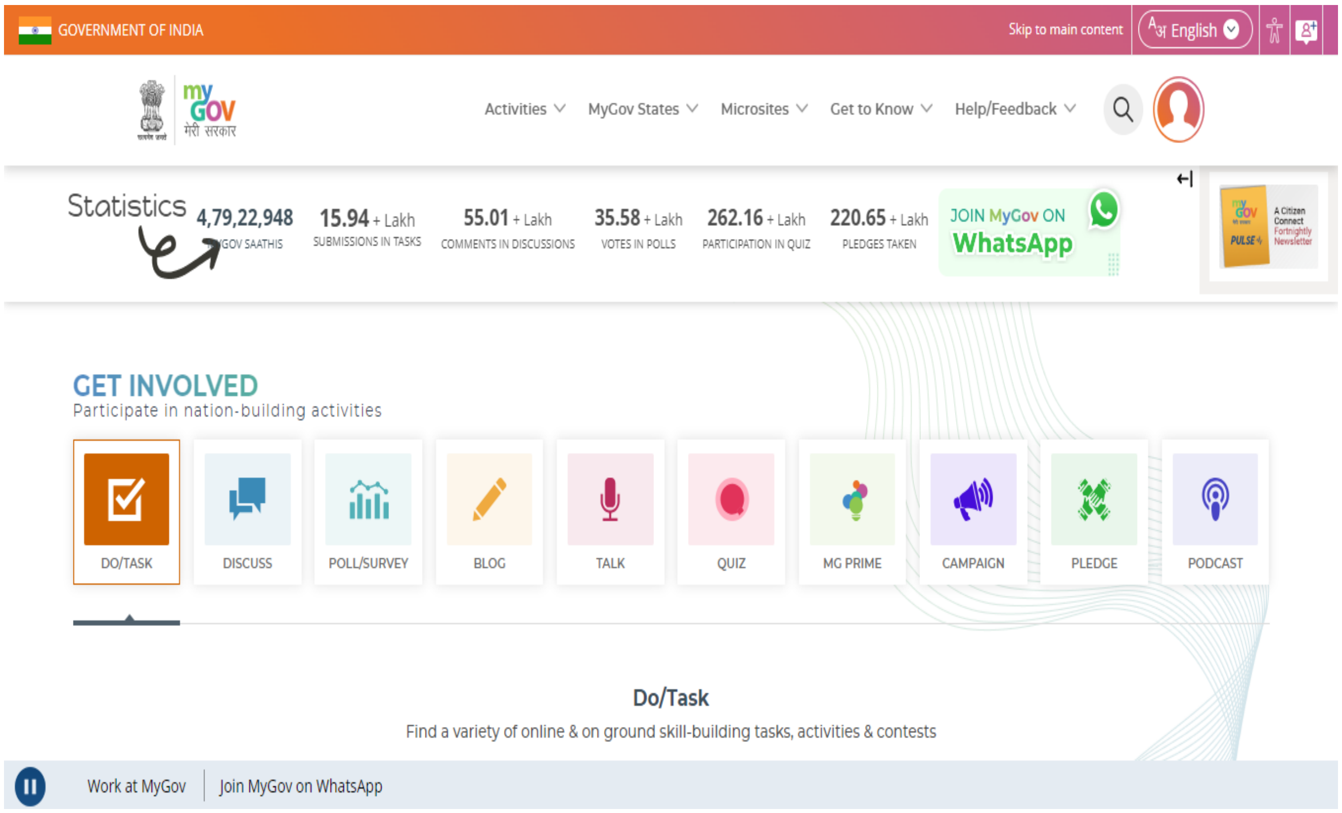
In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax; this is a non-trivial task.

<u>Vulnerability 2:</u>

On this website, there are 2 high risks which are represented below:

High Risk	
Name of Vulnerability	External Service Interaction (DNS)
Name of Vulnerability	XSS (Cross-site scripting)

URL: https://www.mygov.in/



GOVERNMENT OF INDIA
Skip to main content
English
my GOV मेरी सरकार
Activities
MyGov States
Microsites
Get to Know
Help/Feedback
Statistics
4,79,22,948 MYGOV SAATHIS
15.94 + Lakh SUBMISSIONS IN TASKS
55.01 + Lakh COMMENTS IN DISCUSSIONS
35.58 + Lakh VOTES IN POLLS
262.16 + Lakh PARTICIPATION IN QUIZ
220.65 + Lakh PLEDGES TAKEN
JOIN MyGov ON WhatsApp
A Citizen Connect Fortnightly Newsletter
GET INVOLVED
Participate in nation-building activities
DO/TASK
DISCUSS
POLL/SURVEY
BLOG
TALK
QUIZ
MG PRIME
CAMPAIGN
PLEDGE
PODCAST
Do/Task
Find a variety of online & on ground skill-building tasks, activities & contests
Work at MyGov
Join MyGov on WhatsApp

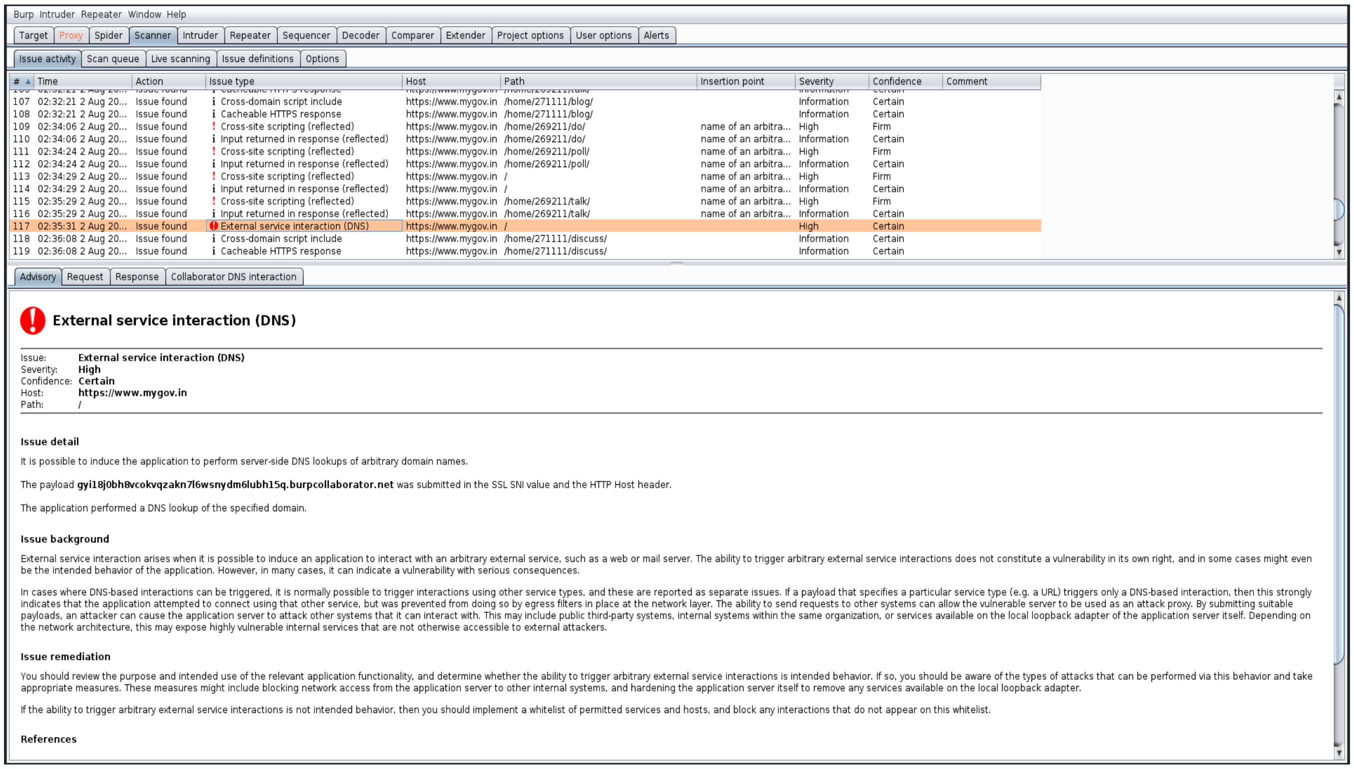
External Service Interaction (DNS):

External Service Interaction (DNS) Vulnerability in the context of Vulnerability Assessment and Penetration Testing (VAPT) involves examining how a web application interacts with external services, particularly Domain Name System (DNS) services, to identify potential security risks. This type of vulnerability can lead to various security issues, including unauthorized data access, domain hijacking, and data exfiltration.

MyGov is an Indian government initiative designed to facilitate citizen engagement and participation in governance. The platform provides a range of services and resources, including information about government schemes, public consultations, and digital services.

We are using the **Burp suite professional tool** for performing the Vulnerability Assessment.

DNS: DNS is a system that translates human-readable domain names (like www.example.com) into IP addresses.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Issue activity Scan queue Live scanning Issue definitions Options
Advisory Request Response Collaborator DNS interaction

#	Time	Action	Issue type	Host	Path	Insertion point	Severity	Confidence	Comment
107	02:32:21 2 Aug 20...	Issue found	Cross-domain script include	https://www.mygov.in	/home/271111/blog/		Information	Certain	
108	02:32:21 2 Aug 20...	Issue found	Cacheable HTTPS response	https://www.mygov.in	/home/271111/blog/		Information	Certain	
109	02:34:06 2 Aug 20...	Issue found	Cross-site scripting (reflected)	https://www.mygov.in	/home/269211/do/	name of an arbitra...	High	Firm	
110	02:34:06 2 Aug 20...	Issue found	Input returned in response (reflected)	https://www.mygov.in	/home/269211/do/	name of an arbitra...	Information	Certain	
111	02:34:24 2 Aug 20...	Issue found	Cross-site scripting (reflected)	https://www.mygov.in	/home/269211/poll/	name of an arbitra...	High	Firm	
112	02:34:24 2 Aug 20...	Issue found	Input returned in response (reflected)	https://www.mygov.in	/home/269211/poll/	name of an arbitra...	Information	Certain	
113	02:34:29 2 Aug 20...	Issue found	Cross-site scripting (reflected)	https://www.mygov.in	/	name of an arbitra...	High	Firm	
114	02:34:29 2 Aug 20...	Issue found	Input returned in response (reflected)	https://www.mygov.in	/	name of an arbitra...	Information	Certain	
115	02:35:29 2 Aug 20...	Issue found	Cross-site scripting (reflected)	https://www.mygov.in	/home/269211/talk/	name of an arbitra...	High	Firm	
116	02:35:29 2 Aug 20...	Issue found	Input returned in response (reflected)	https://www.mygov.in	/home/269211/talk/	name of an arbitra...	Information	Certain	
117	02:35:31 2 Aug 20...	Issue found	External service interaction (DNS)	https://www.mygov.in	/		High	Certain	
118	02:36:08 2 Aug 20...	Issue found	Cross-domain script include	https://www.mygov.in	/home/271111/discuss/		Information	Certain	
119	02:36:08 2 Aug 20...	Issue found	Cacheable HTTPS response	https://www.mygov.in	/home/271111/discuss/		Information	Certain	

External service interaction (DNS)

Issue:	**External service interaction (DNS)**
Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.mygov.in**
Path:	/

Issue detail

It is possible to induce the application to perform server-side DNS lookups of arbitrary domain names.

The payload **gyi18j0bh8vcokvqzakn7l6wsnydm6lubh15q.burpcollaborator.net** was submitted in the SSL SNI value and the HTTP Host header.

The application performed a DNS lookup of the specified domain.

Issue background

External service interaction arises when it is possible to induce an application to interact with an arbitrary external service, such as a web or mail server. The ability to trigger arbitrary external service interactions does not constitute a vulnerability in its own right, and in some cases might even be the intended behavior of the application. However, in many cases, it can indicate a vulnerability with serious consequences.

In cases where DNS-based interactions can be triggered, it is normally possible to trigger interactions using other service types, and these are reported as separate issues. If a payload that specifies a particular service type (e.g. a URL) triggers only a DNS-based interaction, then this strongly indicates that the application attempted to connect using that other service, but was prevented from doing so by egress filters in place at the network layer. The ability to send requests to other systems can allow the vulnerable server to be used as an attack proxy. By submitting suitable payloads, an attacker can cause the application server to attack other systems that it can interact with. This may include public third-party systems, internal systems within the same organization, or services available on the local loopback adapter of the application server itself. Depending on the network architecture, this may expose highly vulnerable internal services that are not otherwise accessible to external attackers.

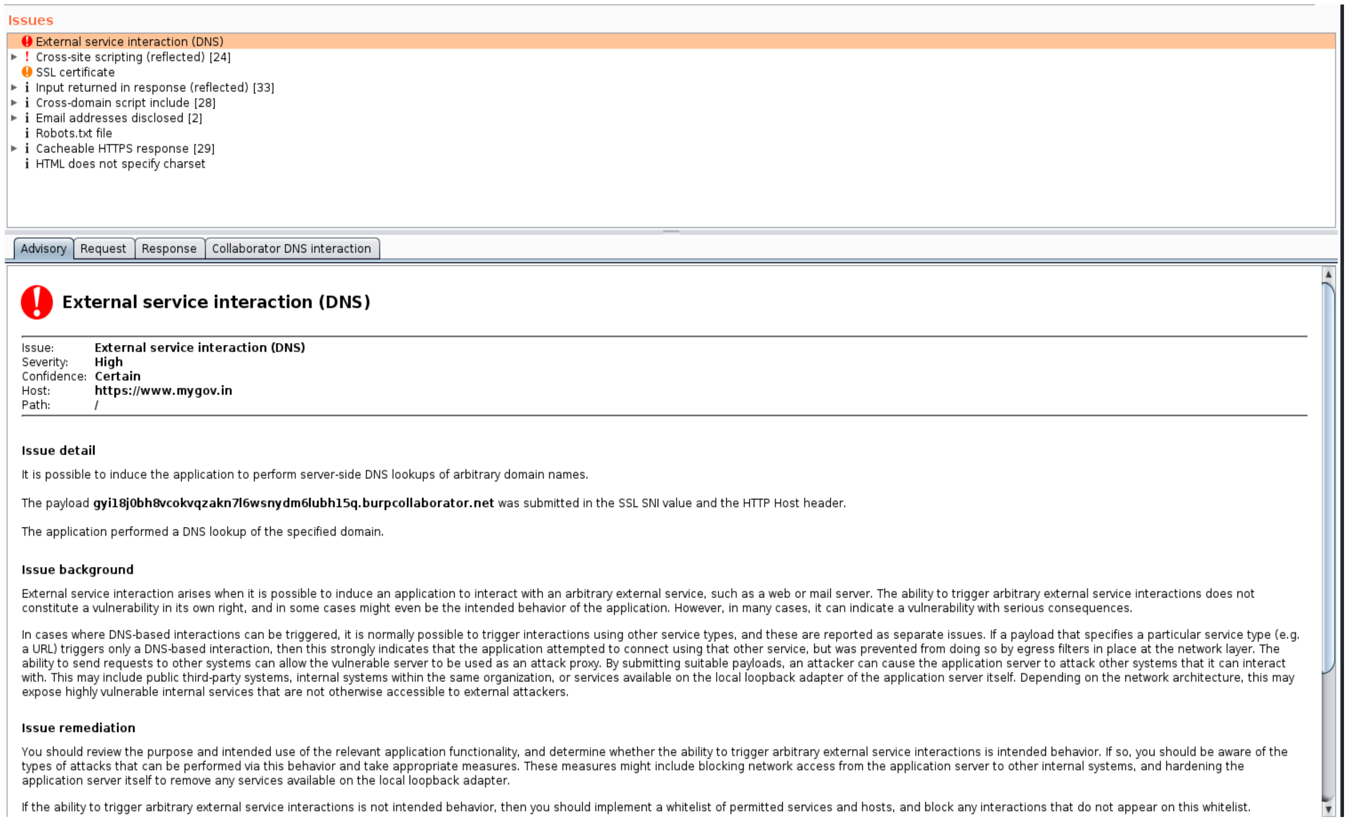
Issue remediation

You should review the purpose and intended use of the relevant application functionality, and determine whether the ability to trigger arbitrary external service interactions is intended behavior. If so, you should be aware of the types of attacks that can be performed via this behavior and take appropriate measures. These measures might include blocking network access from the application server to other internal systems, and hardening the application server itself to remove any services available on the local loopback adapter.

If the ability to trigger arbitrary external service interactions is not intended behavior, then you should implement a whitelist of permitted services and hosts, and block any interactions that do not appear on this whitelist.

References

The following is a detailed list of issues currently identified in the external service interaction with DNS, including various vulnerabilities and misconfigurations that could impact the security and functionality of the web application. These issues encompass potential risks such as DNS spoofing, insecure configurations, and data exfiltration, which need to be addressed to ensure robust protection and reliable performance.



Issues

- External service interaction (DNS)
- Cross-site scripting (reflected) [24]
- SSL certificate
- Input returned in response (reflected) [33]
- Cross-domain script include [28]
- Email addresses disclosed [2]
- Robots.txt file
- Cacheable HTTPS response [29]
- HTML does not specify charset

External service interaction (DNS)

Issue:	**External service interaction (DNS)**
Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.mygov.in**
Path:	/

Issue detail

It is possible to induce the application to perform server-side DNS lookups of arbitrary domain names.

The payload **gyi18j0bh8vcokvqzakn7l6wsnydm6lubh15q.burpcollaborator.net** was submitted in the SSL SNI value and the HTTP Host header.

The application performed a DNS lookup of the specified domain.

Issue background

External service interaction arises when it is possible to induce an application to interact with an arbitrary external service, such as a web or mail server. The ability to trigger arbitrary external service interactions does not constitute a vulnerability in its own right, and in some cases might even be the intended behavior of the application. However, in many cases, it can indicate a vulnerability with serious consequences.

In cases where DNS-based interactions can be triggered, it is normally possible to trigger interactions using other service types, and these are reported as separate issues. If a payload that specifies a particular service type (e.g. a URL) triggers only a DNS-based interaction, then this strongly indicates that the application attempted to connect using that other service, but was prevented from doing so by egress filters in place at the network layer. The ability to send requests to other systems can allow the vulnerable server to be used as an attack proxy. By submitting suitable payloads, an attacker can cause the application server to attack other systems that it can interact with. This may include public third-party systems, internal systems within the same organization, or services available on the local loopback adapter of the application server itself. Depending on the network architecture, this may expose highly vulnerable internal services that are not otherwise accessible to external attackers.

Issue remediation

You should review the purpose and intended use of the relevant application functionality, and determine whether the ability to trigger arbitrary external service interactions is intended behavior. If so, you should be aware of the types of attacks that can be performed via this behavior and take appropriate measures. These measures might include blocking network access from the application server to other internal systems, and hardening the application server itself to remove any services available on the local loopback adapter.

If the ability to trigger arbitrary external service interactions is not intended behavior, then you should implement a whitelist of permitted services and hosts, and block any interactions that do not appear on this whitelist.

<u>How DNS attack works</u>

DNS Cache Poisoning (DNS Spoofing):

- **Injection of Malicious Data**: An attacker sends forged DNS responses to a DNS resolver, which then caches these incorrect responses.
- **Cache Compromise**: The attacker manipulates the cache so that the DNS resolver returns malicious IP addresses for domain names.
- **User Redirection**: When users request the compromised domain, the resolver returns the malicious IP address, redirecting users to attacker-controlled sites.

DNS Tunneling:

- **Data Exfiltration**: Attackers use DNS queries and responses to transmit data between an infected system and an attacker-controlled server.
- **Encoded Data**: Sensitive data is encoded within DNS requests or responses, effectively bypassing network security controls.
- **Communication Channel**: DNS acts as a covert channel for command and control communications.

Preventing External Service Interaction (DNS):

a. Secure DNS Resolvers

- **Use DNSSEC**: Implement DNS Security Extensions (DNSSEC) to protect against cache poisoning and ensure the integrity of DNS responses.
- **Configure Resolvers Properly**: Ensure DNS resolvers are configured securely and do not accept unsolicited or malicious DNS responses.

b. Input Validation

- **Sanitize Inputs**: Validate and sanitize all user inputs that are used for DNS lookups or domain resolution to prevent abuse and information disclosure.
- **Restrict DNS Requests**: Limit DNS requests to trusted and necessary domains, avoiding exposure of internal network details.

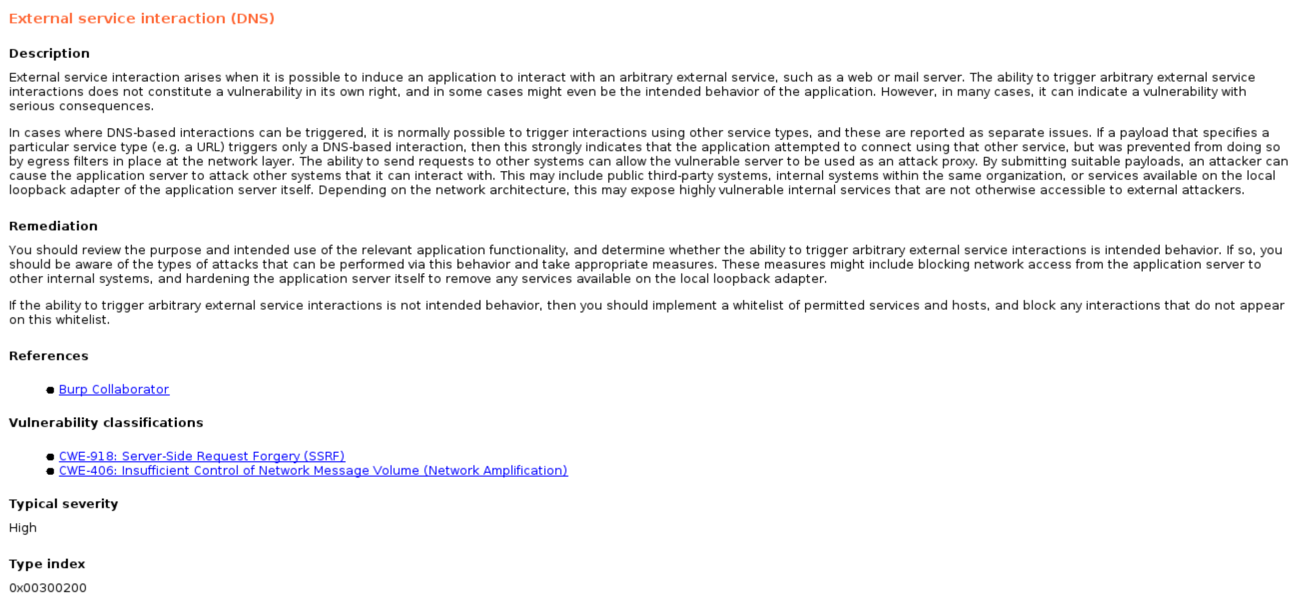
c. Network Security

- **Use VPNs and Firewalls**: Implement VPNs or firewalls to restrict access to internal network resources and reduce exposure to DNS rebinding attacks.
- **Monitor Traffic**: Continuously monitor network traffic for unusual DNS queries or responses that might indicate attacks.

d. Secure Code Practices

- **Avoid Direct DNS Lookups**: Avoid using user-supplied data for direct DNS lookups unless necessary and validated.
- **Employ Security Controls**: Implement additional security controls and checks to mitigate risks associated with external service interactions.

Remedies:



External service interaction (DNS)

Description

External service interaction arises when it is possible to induce an application to interact with an arbitrary external service, such as a web or mail server. The ability to trigger arbitrary external service interactions does not constitute a vulnerability in its own right, and in some cases might even be the intended behavior of the application. However, in many cases, it can indicate a vulnerability with serious consequences.

In cases where DNS-based interactions can be triggered, it is normally possible to trigger interactions using other service types, and these are reported as separate issues. If a payload that specifies a particular service type (e.g. a URL) triggers only a DNS-based interaction, then this strongly indicates that the application attempted to connect using that other service, but was prevented from doing so by egress filters in place at the network layer. The ability to send requests to other systems can allow the vulnerable server to be used as an attack proxy. By submitting suitable payloads, an attacker can cause the application server to attack other systems that it can interact with. This may include public third-party systems, internal systems within the same organization, or services available on the local loopback adapter of the application server itself. Depending on the network architecture, this may expose highly vulnerable internal services that are not otherwise accessible to external attackers.

Remediation

You should review the purpose and intended use of the relevant application functionality, and determine whether the ability to trigger arbitrary external service interactions is intended behavior. If so, you should be aware of the types of attacks that can be performed via this behavior and take appropriate measures. These measures might include blocking network access from the application server to other internal systems, and hardening the application server itself to remove any services available on the local loopback adapter.

If the ability to trigger arbitrary external service interactions is not intended behavior, then you should implement a whitelist of permitted services and hosts, and block any interactions that do not appear on this whitelist.

References

- Burp Collaborator

Vulnerability classifications

- CWE-918: Server-Side Request Forgery (SSRF)
- CWE-406: Insufficient Control of Network Message Volume (Network Amplification)

Typical severity

High

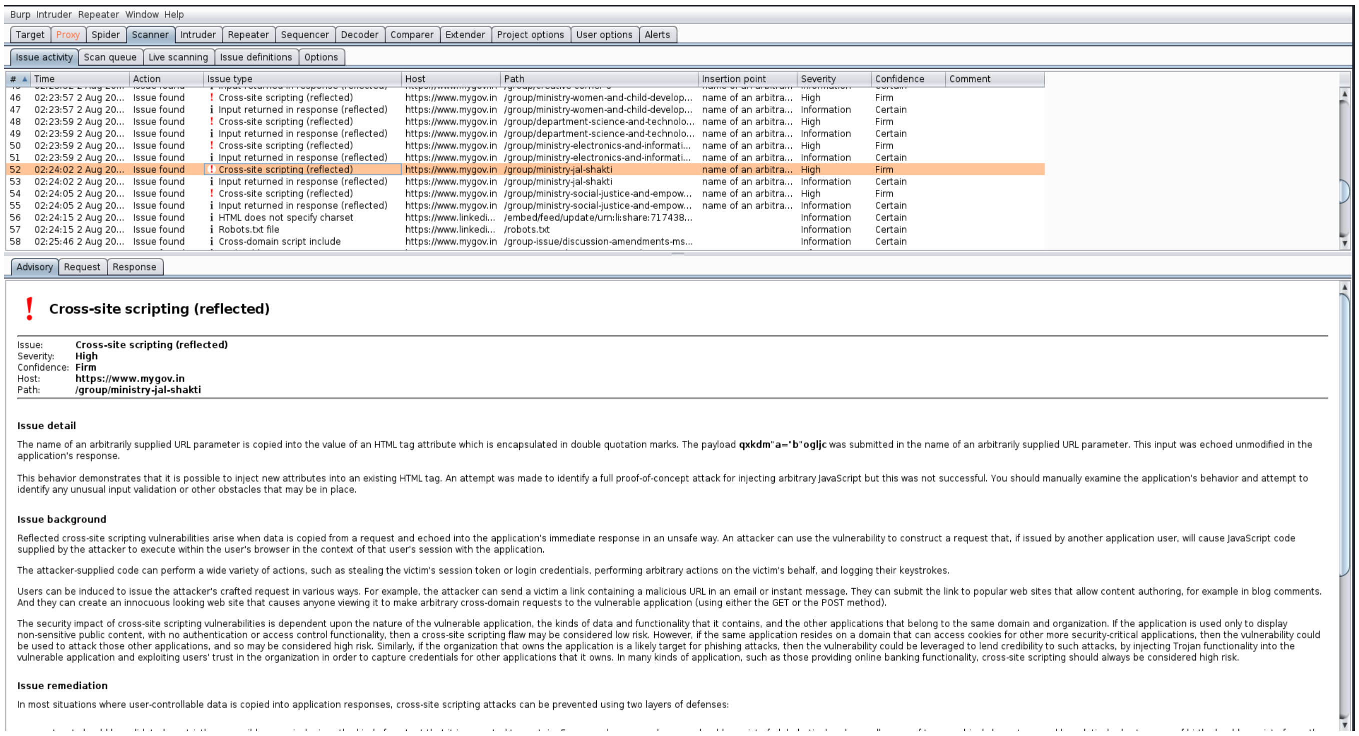
Type index

0x00300200

(XSS) Cross-Site Scripting Vulnerability:

Cross-site scripting (XSS) is a type of security vulnerability typically found in web applications. It allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can then be executed in the context of the user’s browser, potentially leading to a variety of malicious activities such as stealing cookies, session tokens, or other sensitive information, defacing websites, or redirecting users to malicious sites.

We are using the **Burp suite professional tool** for performing the Vulnerability Assessment.



Burp Intruder Repeater Window Help

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts

Issue activity | Scan queue | Live scanning | Issue definitions | Options

#	Time	Action	Issue type	Host	Path	Insertion point	Severity	Confidence	Comment
46	02:23:57 2 Aug 20...	Issue found	! Cross-site scripting (reflected)	https://www.mygov.in	/group/ministry-women-and-child-develop...	name of an arbitra...	High	Firm	
47	02:23:57 2 Aug 20...	Issue found	i Input returned in response (reflected)	https://www.mygov.in	/group/ministry-women-and-child-develop...	name of an arbitra...	Information	Certain	
48	02:23:59 2 Aug 20...	Issue found	! Cross-site scripting (reflected)	https://www.mygov.in	/group/department-science-and-technolo...	name of an arbitra...	High	Firm	
49	02:23:59 2 Aug 20...	Issue found	i Input returned in response (reflected)	https://www.mygov.in	/group/department-science-and-technolo...	name of an arbitra...	Information	Certain	
50	02:23:59 2 Aug 20...	Issue found	! Cross-site scripting (reflected)	https://www.mygov.in	/group/ministry-electronics-and-informati...	name of an arbitra...	High	Firm	
51	02:23:59 2 Aug 20...	Issue found	i Input returned in response (reflected)	https://www.mygov.in	/group/ministry-electronics-and-informati...	name of an arbitra...	Information	Certain	
52	02:24:02 2 Aug 20...	Issue found	! Cross-site scripting (reflected)	https://www.mygov.in	/group/ministry-jal-shakti	name of an arbitra...	High	Firm	
53	02:24:02 2 Aug 20...	Issue found	i Input returned in response (reflected)	https://www.mygov.in	/group/ministry-jal-shakti	name of an arbitra...	Information	Certain	
54	02:24:05 2 Aug 20...	Issue found	! Cross-site scripting (reflected)	https://www.mygov.in	/group/ministry-social-justice-and-empow...	name of an arbitra...	High	Firm	
55	02:24:05 2 Aug 20...	Issue found	i Input returned in response (reflected)	https://www.mygov.in	/group/ministry-social-justice-and-empow...	name of an arbitra...	Information	Certain	
56	02:24:15 2 Aug 20...	Issue found	i HTML does not specify charset	https://www.linkedi...	/embed/feed/update/urn:li:share:717438...		Information	Certain	
57	02:24:15 2 Aug 20...	Issue found	i Robots.txt file	https://www.linkedi...	/robots.txt		Information	Certain	
58	02:25:46 2 Aug 20...	Issue found	i Cross-domain script include	https://www.mygov.in	/group-issue/discussion-amendments-ms...		Information	Certain	

Advisory | Request | Response

! Cross-site scripting (reflected)

Issue:	**Cross-site scripting (reflected)**
Severity:	**High**
Confidence:	**Firm**
Host:	**https://www.mygov.in**
Path:	**/group/ministry-jal-shakti**

Issue detail

The name of an arbitrarily supplied URL parameter is copied into the value of an HTML tag attribute which is encapsulated in double quotation marks. The payload **qxkdm"a="b"ogljc** was submitted in the name of an arbitrarily supplied URL parameter. This input was echoed unmodified in the application's response.

This behavior demonstrates that it is possible to inject new attributes into an existing HTML tag. An attempt was made to identify a full proof-of-concept attack for injecting arbitrary JavaScript but this was not successful. You should manually examine the application's behavior and attempt to identify any unusual input validation or other obstacles that may be in place.

Issue background

Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.

The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

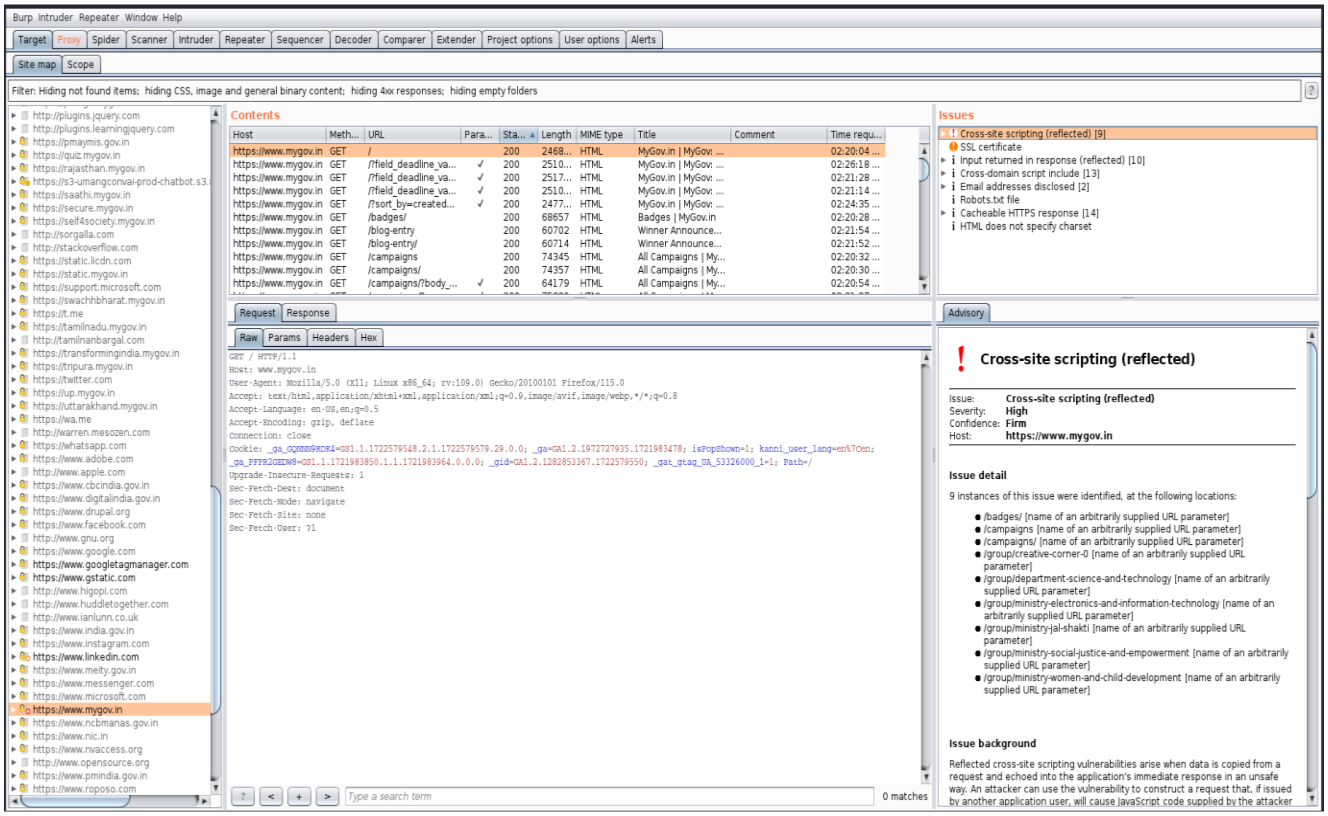
Users can be induced to issue the attacker's crafted request in various ways. For example, the attacker can send a victim a link containing a malicious URL in an email or instant message. They can submit the link to popular web sites that allow content authoring, for example in blog comments. And they can create an innocuous looking web site that causes anyone viewing it to make arbitrary cross-domain requests to the vulnerable application (using either the GET or the POST method).

The security impact of cross-site scripting vulnerabilities is dependent upon the nature of the vulnerable application, the kinds of data and functionality that it contains, and the other applications that belong to the same domain and organization. If the application is used only to display non-sensitive public content, with no authentication or access control functionality, then a cross-site scripting flaw may be considered low risk. However, if the same application resides on a domain that can access cookies for other more security-critical applications, then the vulnerability could be used to attack those other applications, and so may be considered high risk. Similarly, if the organization that owns the application is a likely target for phishing attacks, then the vulnerability could be leveraged to lend credibility to such attacks, by injecting Trojan functionality into the vulnerable application and exploiting users' trust in the organization in order to capture credentials for other applications that it owns. In many kinds of application, such as those providing online banking functionality, cross-site scripting should always be considered high risk.

Issue remediation

In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defenses:

The following are the issues that are present in the mygov website under cross-site scripting(xss).



Burp Intruder Repeater Window Help

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts

Site map | Scope

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

- http://plugins.jquery.com
- http://plugins.learningjquery.com
- https://pmaymis.gov.in
- https://quiz.mygov.in
- https://rajasthan.mygov.in
- https://s3-umangconvai-prod-chatbot.s3.
- https://saathi.mygov.in
- https://secure.mygov.in
- https://self4society.mygov.in
- http://sorgalla.com
- http://stackoverflow.com
- https://static.licdn.com
- https://static.mygov.in
- https://support.microsoft.com
- https://swachhbharat.mygov.in
- https://t.me
- https://tamilnadu.mygov.in
- http://tamilnanbargal.com
- https://transformingindia.mygov.in
- https://tripura.mygov.in
- https://twitter.com
- https://up.mygov.in
- https://uttarakhand.mygov.in
- https://wa.me
- http://warren.mesozen.com
- https://whatsapp.com
- https://www.adobe.com
- http://www.apple.com
- https://www.cbcindia.gov.in
- https://www.digitalindia.gov.in
- https://www.drupal.org
- https://www.facebook.com
- http://www.gnu.org
- https://www.google.com
- https://www.googletagmanager.com
- https://www.gstatic.com
- http://www.higopi.com
- http://www.huddletogether.com
- http://www.ianlunn.co.uk
- https://www.india.gov.in
- https://www.instagram.com
- https://www.linkedin.com
- https://www.meity.gov.in
- https://www.messenger.com
- https://www.microsoft.com
- https://www.mygov.in
- https://www.ncbmanas.gov.in
- https://www.nic.in
- https://www.nvaccess.org
- http://www.opensource.org
- https://www.pmindia.gov.in
- https://www.roposo.com

Contents

Host	Meth...	URL	Para...	Sta...	Length	MIME type	Title	Comment	Time requ...
https://www.mygov.in	GET	/		200	2468...	HTML	MyGov.in | MyGov: ...		02:20:04 ...
https://www.mygov.in	GET	/?field_deadline_va...	✓	200	2510...	HTML	MyGov.in | MyGov: ...		02:26:18 ...
https://www.mygov.in	GET	/?field_deadline_va...	✓	200	2517...	HTML	MyGov.in | MyGov: ...		02:21:28 ...
https://www.mygov.in	GET	/?field_deadline_va...	✓	200	2510...	HTML	MyGov.in | MyGov: ...		02:21:14 ...
https://www.mygov.in	GET	/?sort_by=created...	✓	200	2477...	HTML	MyGov.in | MyGov: ...		02:24:35 ...
https://www.mygov.in	GET	/badges/		200	68657	HTML	Badges | MyGov.in		02:20:28 ...
https://www.mygov.in	GET	/blog-entry		200	60702	HTML	Winner Announce...		02:21:54 ...
https://www.mygov.in	GET	/blog-entry/		200	60714	HTML	Winner Announce...		02:21:52 ...
https://www.mygov.in	GET	/campaigns		200	74345	HTML	All Campaigns | My...		02:20:32 ...
https://www.mygov.in	GET	/campaigns/		200	74357	HTML	All Campaigns | My...		02:20:30 ...
https://www.mygov.in	GET	/campaigns/?body_...	✓	200	64179	HTML	All Campaigns | My...		02:20:54 ...

Request | Response

Raw | Params | Headers | Hex

```
GET / HTTP/1.1
Host: www.mygov.in
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: _ga_GQNNN9RDK4=GS1.1.1722579548.2.1.1722579579.29.0.0; _ga=GA1.2.1972727935.1721983478; isPopShown=1; kanni_user_lang=en%7Cen;
_ga_PFPR2GEDW8=GS1.1.1721983850.1.1.1721983964.0.0.0; _gid=GA1.2.1282853367.1722579550; _gat_gtag_UA_53326000_1=1; Path=/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

? < + > Type a search term 0 matches

Issues

- ! Cross-site scripting (reflected) [9]
- ! SSL certificate
- i Input returned in response (reflected) [10]
- i Cross-domain script include [13]
- i Email addresses disclosed [2]
- i Robots.txt file
- i Cacheable HTTPS response [14]
- i HTML does not specify charset

Advisory

! Cross-site scripting (reflected)

Issue:	**Cross-site scripting (reflected)**
Severity:	**High**
Confidence:	**Firm**
Host:	**https://www.mygov.in**

Issue detail

9 instances of this issue were identified, at the following locations:

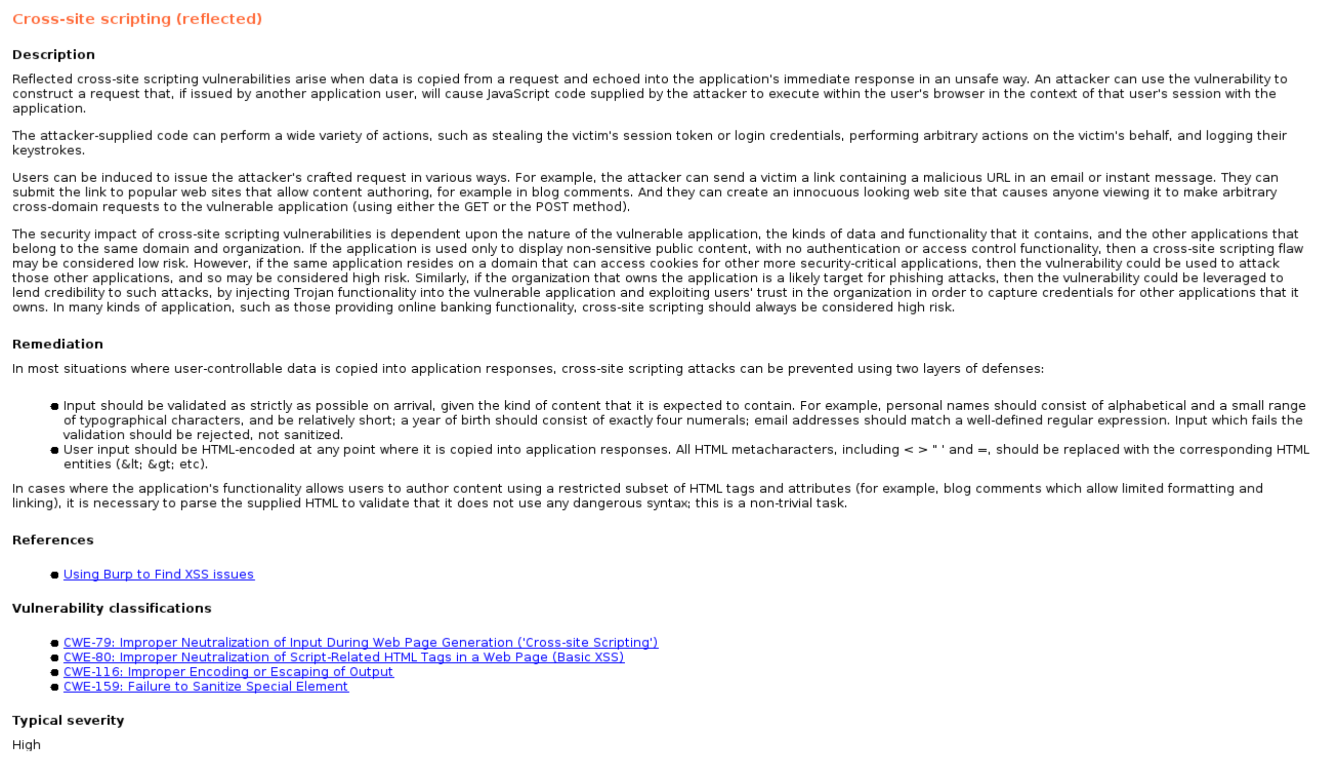
- /badges/ [name of an arbitrarily supplied URL parameter]
- /campaigns [name of an arbitrarily supplied URL parameter]
- /campaigns/ [name of an arbitrarily supplied URL parameter]
- /group/creative-corner-0 [name of an arbitrarily supplied URL parameter]
- /group/department-science-and-technology [name of an arbitrarily supplied URL parameter]
- /group/ministry-electronics-and-information-technology [name of an arbitrarily supplied URL parameter]
- /group/ministry-jal-shakti [name of an arbitrarily supplied URL parameter]
- /group/ministry-social-justice-and-empowerment [name of an arbitrarily supplied URL parameter]
- /group/ministry-women-and-child-development [name of an arbitrarily supplied URL parameter]

Issue background

Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker

Remedies:

The following are the remedies provided by the Burp Suite professional tool. To prevent the cross-site scripting.



Cross-site scripting (reflected)

Description

Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.

The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

Users can be induced to issue the attacker's crafted request in various ways. For example, the attacker can send a victim a link containing a malicious URL in an email or instant message. They can submit the link to popular web sites that allow content authoring, for example in blog comments. And they can create an innocuous looking web site that causes anyone viewing it to make arbitrary cross-domain requests to the vulnerable application (using either the GET or the POST method).

The security impact of cross-site scripting vulnerabilities is dependent upon the nature of the vulnerable application, the kinds of data and functionality that it contains, and the other applications that belong to the same domain and organization. If the application is used only to display non-sensitive public content, with no authentication or access control functionality, then a cross-site scripting flaw may be considered low risk. However, if the same application resides on a domain that can access cookies for other more security-critical applications, then the vulnerability could be used to attack those other applications, and so may be considered high risk. Similarly, if the organization that owns the application is a likely target for phishing attacks, then the vulnerability could be leveraged to lend credibility to such attacks, by injecting Trojan functionality into the vulnerable application and exploiting users' trust in the organization in order to capture credentials for other applications that it owns. In many kinds of application, such as those providing online banking functionality, cross-site scripting should always be considered high risk.

Remediation

In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defenses:

- Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.
- User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (&lt; &gt; etc).

In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax; this is a non-trivial task.

References

- Using Burp to Find XSS issues

Vulnerability classifications

- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
- CWE-116: Improper Encoding or Escaping of Output
- CWE-159: Failure to Sanitize Special Element

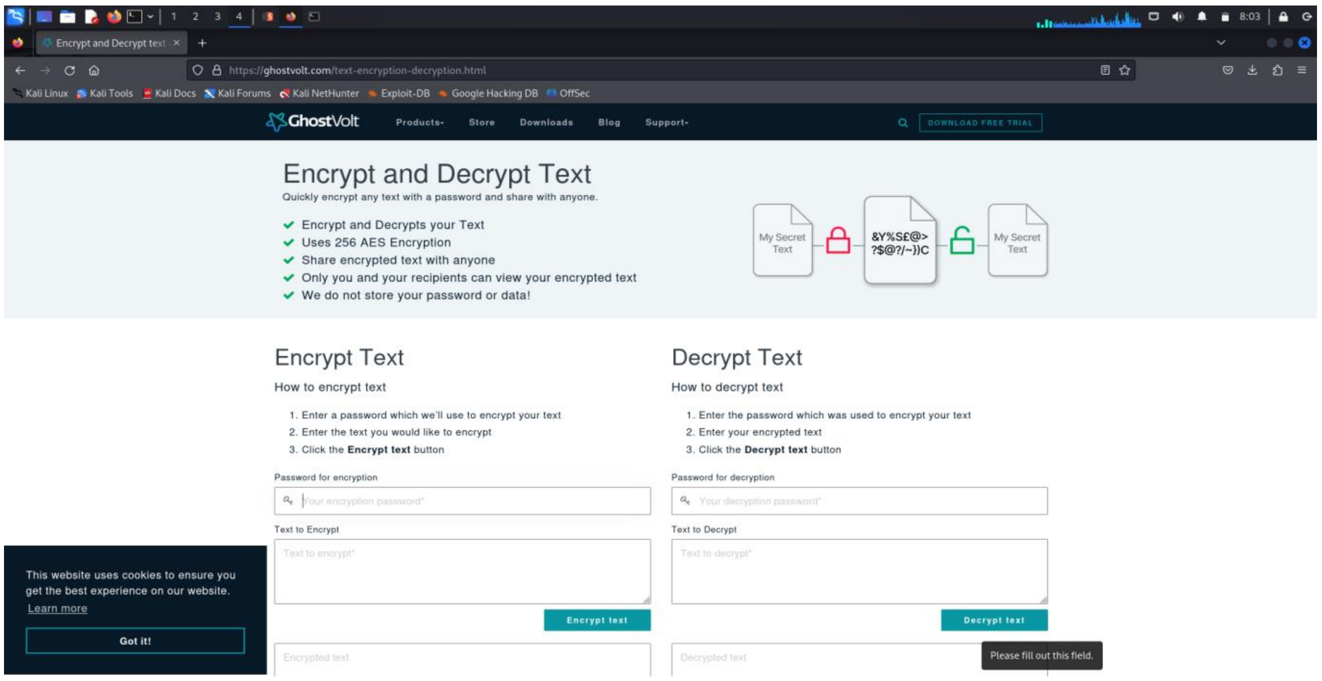
Typical severity

High

Vulnerability 3:

low Risk	
Name of Vulnerability	Strict transport security not enforced
Name of Vulnerability	Password submitted using GET method

URL: https://ghostvolt.com/



Encrypt and Decrypt Text

Quickly encrypt any text with a password and share with anyone.

- ✓ Encrypt and Decrypts your Text
- ✓ Uses 256 AES Encryption
- ✓ Share encrypted text with anyone
- ✓ Only you and your recipients can view your encrypted text
- ✓ We do not store your password or data!

Encrypt Text

How to encrypt text

1. Enter a password which we'll use to encrypt your text
2. Enter the text you would like to encrypt
3. Click the **Encrypt text** button

Password for encryption: Your encryption password*

Text to Encrypt: Text to encrypt*

Encrypt text

Encrypted text

Decrypt Text

How to decrypt text

1. Enter the password which was used to encrypt your text
2. Enter your encrypted text
3. Click the **Decrypt text** button

Password for decryption: Your decryption password*

Text to Decrypt: Text to decrypt*

Decrypt text

Decrypted text

Please fill out this field.

This website uses cookies to ensure you get the best experience on our website. Learn more

Got it!

GhostVolt protects your files, data, and folders with advanced encryption to prevent unauthorized access, data breaches, and cybercrime. Only authorized users can access your data, keeping it secure and private.

Strict Transport Security not enforced:

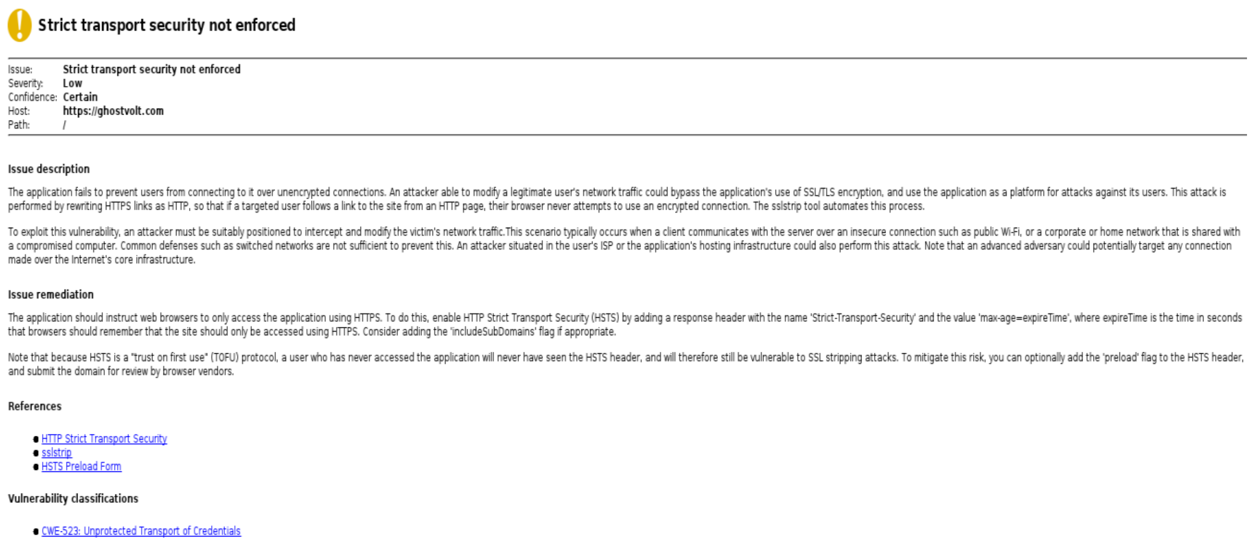
Strict Transport Security Not Enforced is a vulnerability related to web security and specifically to the HTTP Strict Transport Security (HSTS) mechanism. When HSTS is not enforced, it means that a website is not properly implementing or utilizing this security feature, which can expose users to certain types of attacks.

HTTP Strict Transport Security (HSTS):

HSTS is a web security policy mechanism that helps protect websites against man-in-the-middle attacks, such as session hijacking and cookie theft, by ensuring that browsers only connect to the site over HTTPS. This is achieved by instructing the browser to automatically convert any HTTP requests to HTTPS and to prevent any form of HTTP communication.

we are using the **Burp suite professional tool** for performing the Vulnerability Assessment.

The following are the issues that have been identified regarding the lack of enforcement of Strict Transport Security (HSTS) on the Ghostvolt website: this includes the absence of the HSTS header in HTTP responses, failure to enforce HTTPS across all subdomains, insufficient max-age directives, and potential vulnerabilities to man-in-the-middle attacks due to unencrypted communication over HTTP, all of which compromise the site's overall security and user data protection.



Strict transport security not enforced

Issue:	**Strict transport security not enforced**
Severity:	**Low**
Confidence:	**Certain**
Host:	**https://ghostvolt.com**
Path:	/

Issue description

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic.This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Issue remediation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

References

- HTTP Strict Transport Security
- sslstrip
- HSTS Preload Form

Vulnerability classifications

- CWE-523: Unprotected Transport of Credentials

How HSTS attack works:

- **No HSTS Header**: If a web server does not send the HSTS header or does not include the appropriate directives, the browser may not enforce HTTPS for the site. This allows users to connect over HTTP, which is not secure.
- **Potential Attacks**:
  - **Man-in-the-Middle (MitM) Attacks**: Attackers can intercept and modify HTTP traffic between the user and the server.
  - **Session Hijacking**: Attackers may steal session cookies sent over unencrypted HTTP.
  - **Cookie Theft**: Sensitive cookies transmitted over HTTP can be captured by attackers.

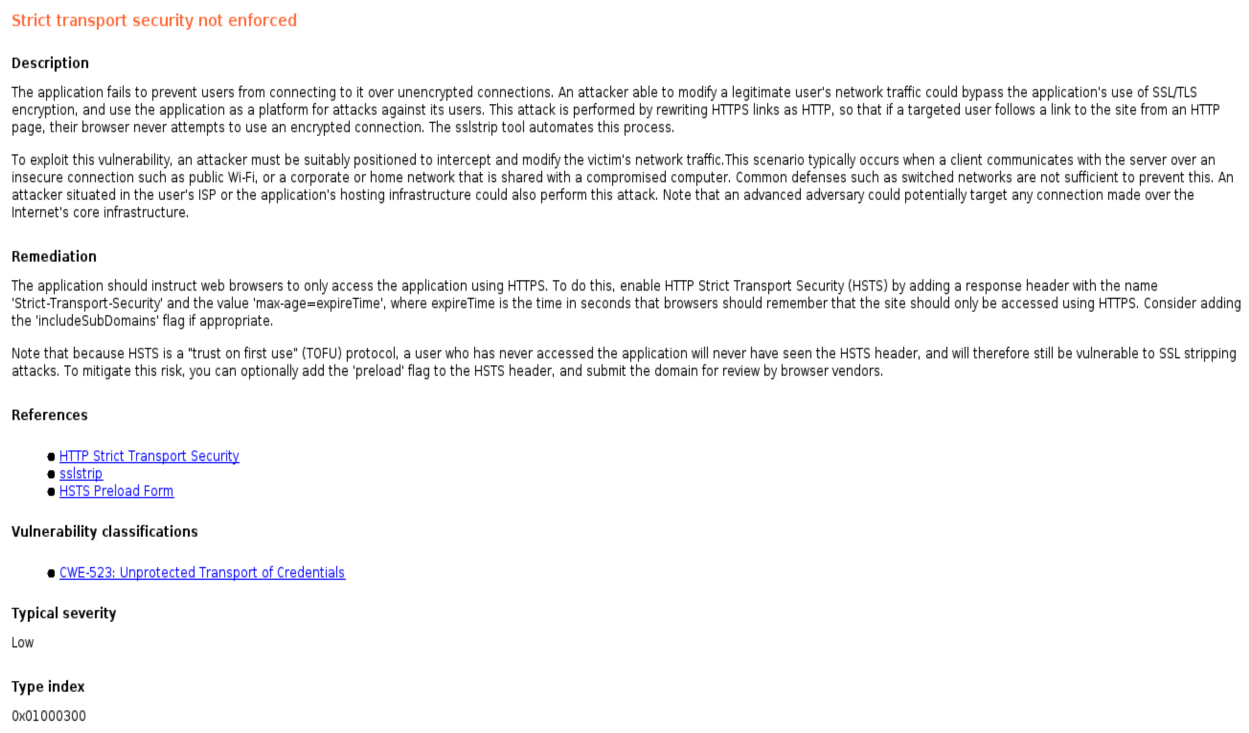
How to prevent HSTS attack:

1. **Implement HSTS**:
   - **Server Configuration**: Configure the web server to include the Strict-Transport-Security header in its responses.
2. **Enable HSTS Preloading**:
   - **Preloading List**: Submit your domain to the HSTS preload list maintained by major browser vendors to ensure that browsers enforce HTTPS before the first connection.
3. **Redirect HTTP to HTTPS**:
   - **Server Configuration**: Ensure that all HTTP traffic is redirected to HTTPS to enforce secure communication.
   - **Example Redirect**:

4. **Regular Testing and Monitoring**:
   - **Security Scans**: Regularly scan your website for security vulnerabilities, including HSTS enforcement.
   - **Compliance Checks**: Ensure that security policies are properly implemented and up-to-date.

Remedies:

The following are the remedies which are given by the burp suite tool itself.



Strict transport security not enforced

Description

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic.This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Remediation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

References

- HTTP Strict Transport Security
- sslstrip
- HSTS Preload Form

Vulnerability classifications

- CWE-523: Unprotected Transport of Credentials

Typical severity

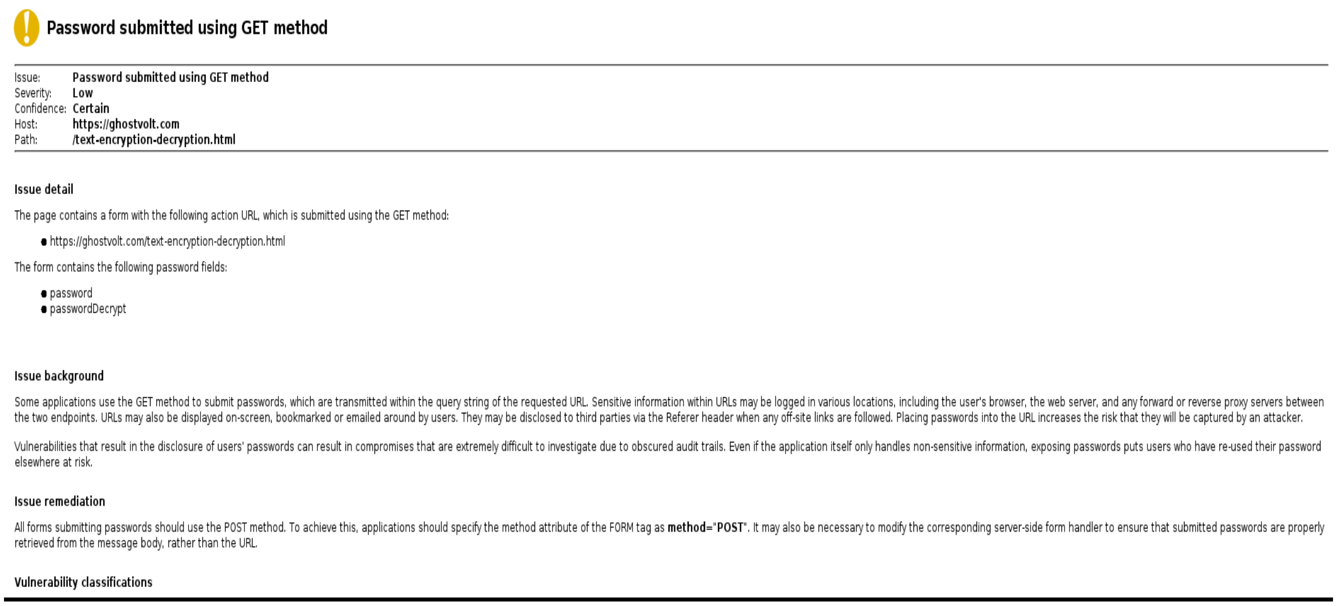
Low

Type index

0x01000300

Password Submitted Using GET Method:

In a Vulnerability Assessment and Penetration Testing (VAPT) context, discovering that passwords are submitted using the GET method indicates a critical security flaw, as it exposes sensitive information through URLs. This practice can lead to passwords being stored in browser history, server logs, or cache, making them accessible to unauthorized parties. The GET method transmits data in the URL, which is inherently insecure compared to POST, as it is visible in network traffic and can be intercepted if HTTPS is not enforced.



Password submitted using GET method

Issue:	**Password submitted using GET method**
Severity:	**Low**
Confidence:	**Certain**
Host:	**https://ghostvolt.com**
Path:	**/text-encryption-decryption.html**

Issue detail

The page contains a form with the following action URL, which is submitted using the GET method:

- https://ghostvolt.com/text-encryption-decryption.html

The form contains the following password fields:

- password
- passwordDecrypt

Issue background

Some applications use the GET method to submit passwords, which are transmitted within the query string of the requested URL. Sensitive information within URLs may be logged in various locations, including the user's browser, the web server, and any forward or reverse proxy servers between the two endpoints. URLs may also be displayed on-screen, bookmarked or emailed around by users. They may be disclosed to third parties via the Referer header when any off-site links are followed. Placing passwords into the URL increases the risk that they will be captured by an attacker.

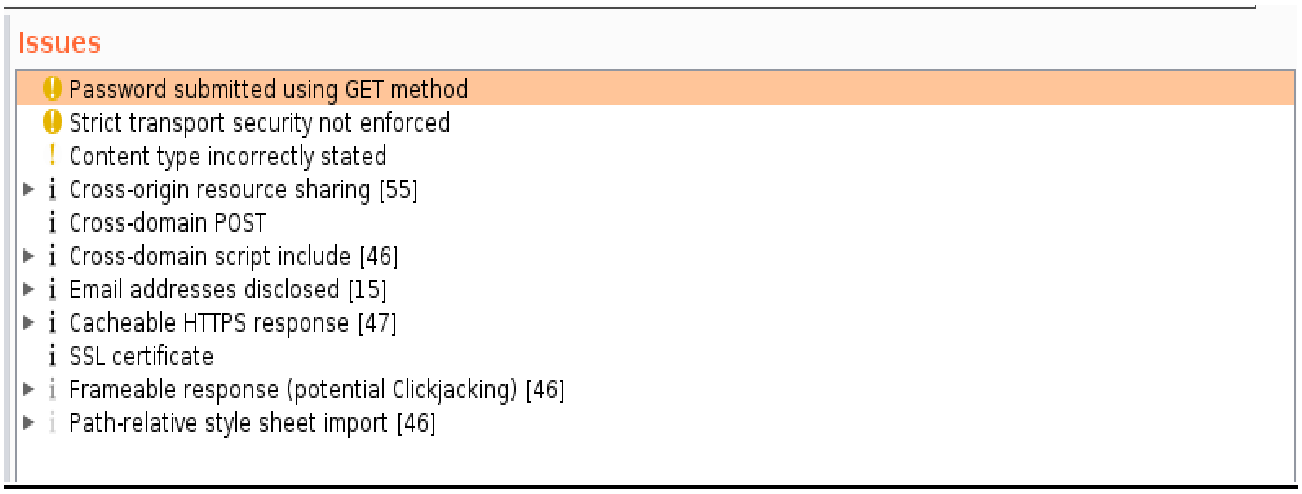
Vulnerabilities that result in the disclosure of users' passwords can result in compromises that are extremely difficult to investigate due to obscured audit trails. Even if the application itself only handles non-sensitive information, exposing passwords puts users who have re-used their password elsewhere at risk.

Issue remediation

All forms submitting passwords should use the POST method. To achieve this, applications should specify the method attribute of the FORM tag as **method="POST"**. It may also be necessary to modify the corresponding server-side form handler to ensure that submitted passwords are properly retrieved from the message body, rather than the URL.

Vulnerability classifications

The following are the issues which are issues which are present.


Issues

- Password submitted using GET method
- Strict transport security not enforced
- Content type incorrectly stated
- Cross-origin resource sharing [55]
- Cross-domain POST
- Cross-domain script include [46]
- Email addresses disclosed [15]
- Cacheable HTTPS response [47]
- SSL certificate
- Frameable response (potential Clickjacking) [46]
- Path-relative style sheet import [46]

<u>How Password Submitted Using GET Method Works:</u>

When a web application submits passwords using the GET method, sensitive information is transmitted as part of the URL in query parameters. Here's how it works:

1. **Form Submission**: A user submits a form where the HTTP method is set to GET. For example, the URL might look like this:
2. **Data Exposure**:
   - **URL Inclusion**: The password is included in the URL as a query parameter
   - **Visibility**: This data is visible in browser history, server logs, and network traffic. It can be intercepted by attackers if HTTPS is not used, or it can be inadvertently shared through logs or URL sharing.

3. **Caching**:
   - **Browser Cache**: The URL with the sensitive data might be cached by the browser, leading to potential exposure if the cache is accessed by others.

How to prevent this attack:

Secure Transmission: Ensure that all forms and data submissions are conducted over HTTPS. This encrypts the data transmitted between the client and server, protecting it from interception.

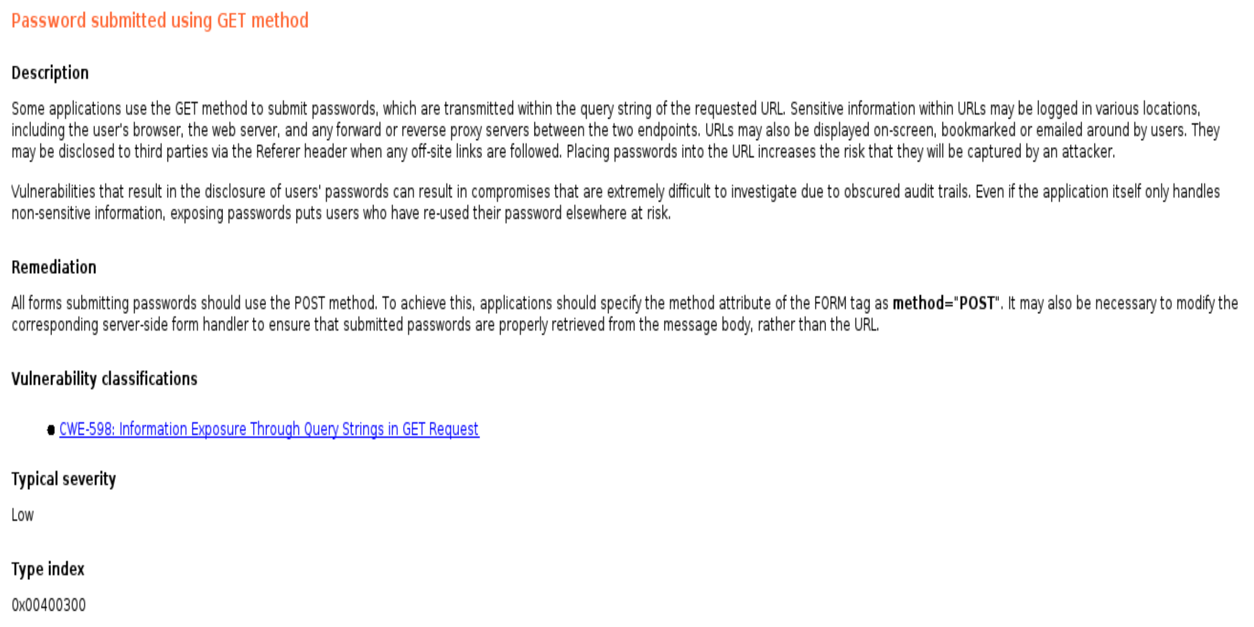
Redirect HTTP to HTTPS: Configure the server to redirect all HTTP requests to HTTPS to enforce secure communication.

Security Best Practices: Train developers on secure coding practices and the importance of using POST for sensitive data to avoid similar vulnerabilities in the future.

Conduct Audits: Perform regular security assessments and code reviews to identify and address potential vulnerabilities, including improper use of HTTP methods.

Remedies:

The following are the remedies which are given by the burp suite tool itself.



Password submitted using GET method

Description

Some applications use the GET method to submit passwords, which are transmitted within the query string of the requested URL. Sensitive information within URLs may be logged in various locations, including the user's browser, the web server, and any forward or reverse proxy servers between the two endpoints. URLs may also be displayed on-screen, bookmarked or emailed around by users. They may be disclosed to third parties via the Referer header when any off-site links are followed. Placing passwords into the URL increases the risk that they will be captured by an attacker.

Vulnerabilities that result in the disclosure of users' passwords can result in compromises that are extremely difficult to investigate due to obscured audit trails. Even if the application itself only handles non-sensitive information, exposing passwords puts users who have re-used their password elsewhere at risk.

Remediation

All forms submitting passwords should use the POST method. To achieve this, applications should specify the method attribute of the FORM tag as **method="POST"**. It may also be necessary to modify the corresponding server-side form handler to ensure that submitted passwords are properly retrieved from the message body, rather than the URL.

Vulnerability classifications

- CWE-598: Information Exposure Through Query Strings in GET Request

Typical severity

Low

Type index

0x00400300

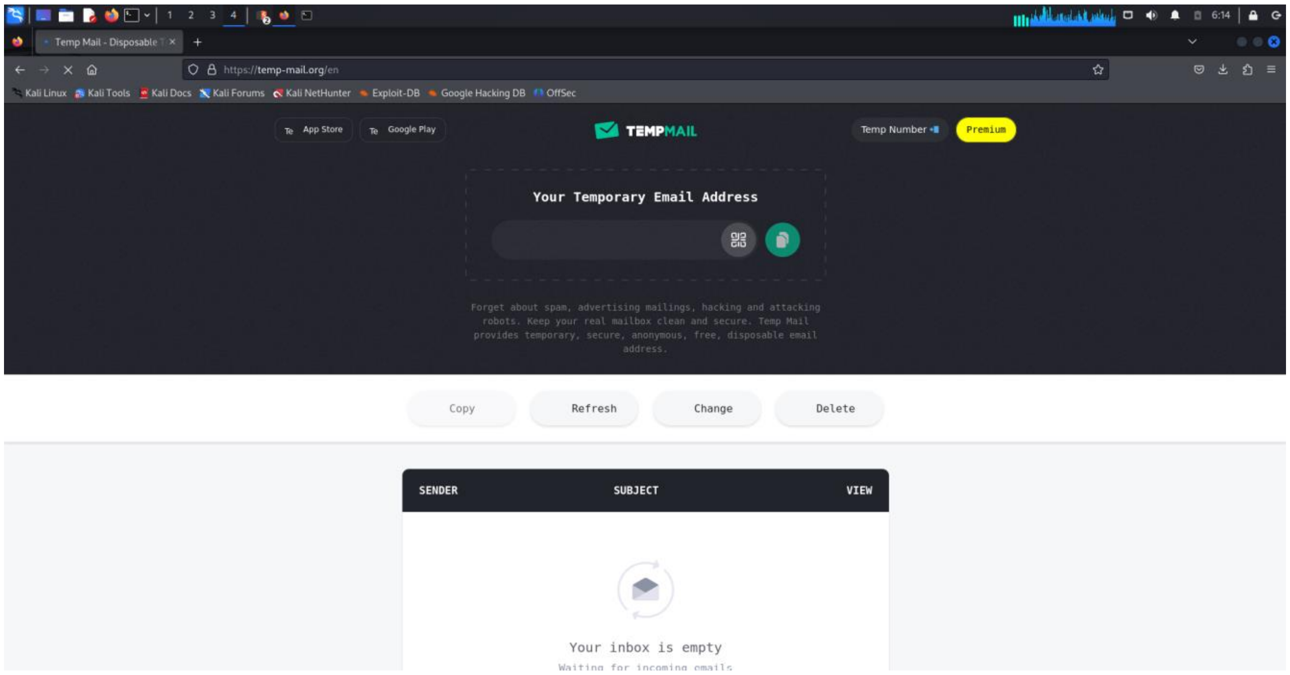
Vulnerability 4:

High Risk	
Name of Vulnerability	Cross-Site Scripting (DOM Based)

Medium Risk	
Name of Vulnerability	SSL cookie without secure flag set

low Risk	
Name of Vulnerability	Source Code Disclosure

URL: https://temp-mail.org/en/



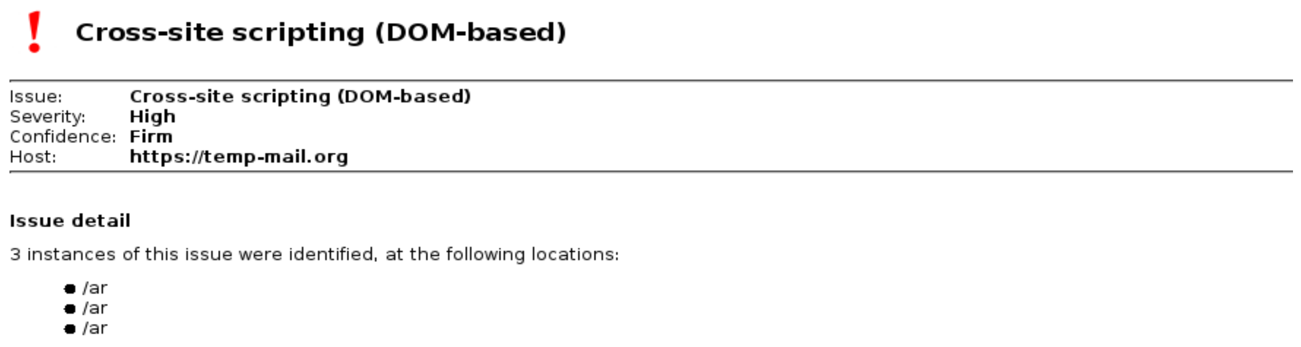
Forget about spam, advertising mailings, hacking and attacking robots. Keep your real mailbox clean and secure. Temp Mail provides temporary, secure, anonymous, free, disposable email address.

A temporary email website provides users with a disposable email address that lasts for a short period, typically ranging from 10 minutes to a few hours. These services are useful for signing up for websites or services without revealing your real email address, helping to avoid spam and unwanted emails. They are often used for quick, anonymous interactions online

Cross-Site Scripting (DOM Based):

Cross-site scripting (XSS) is a security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. **DOM-Based XSS** is a subtype of XSS that occurs when the client-side script in a webpage processes user input in an unsafe manner, leading to the execution of malicious code. Unlike traditional XSS, where the server directly injects the malicious payload, in DOM-based XSS, the vulnerability exists entirely within the client-side code.

We are using the **Burp Suite professional tool** to perform the vulnerability assessment.



Cross-site scripting (DOM-based)

Issue:	**Cross-site scripting (DOM-based)**
Severity:	**High**
Confidence:	**Firm**
Host:	**https://temp-mail.org**

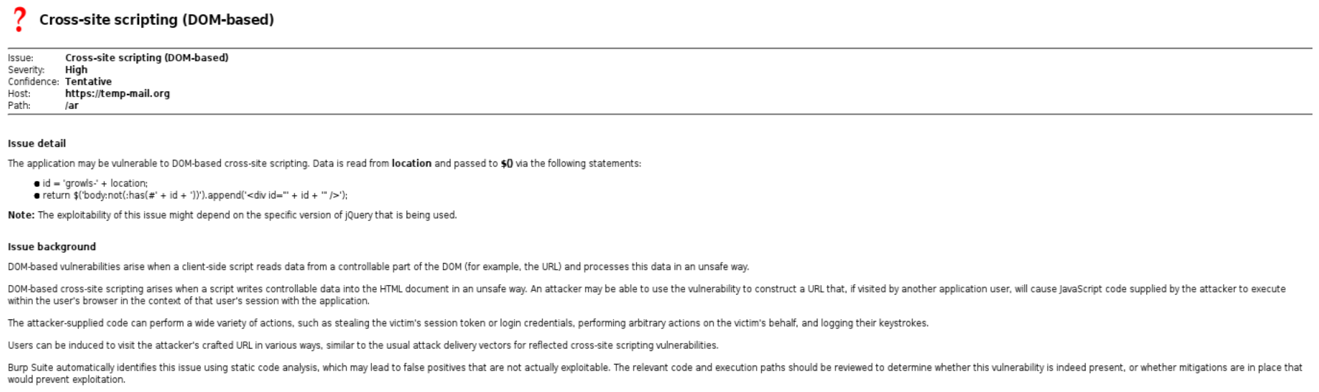
Issue detail

3 instances of this issue were identified, at the following locations:

- /ar
- /ar
- /ar

If a temporary email website has a **DOM -based cross-site Scripting (XSS)** vulnerability, it could potentially allow attackers to inject and execute malicious scripts directly within the user's browser by exploiting the client-side code of the website.

The following are the issues which are issues which are present.



Cross-site scripting (DOM-based)

Issue:	**Cross-site scripting (DOM-based)**
Severity:	**High**
Confidence:	**Tentative**
Host:	**https://temp-mail.org**
Path:	**/ar**

Issue detail

The application may be vulnerable to DOM-based cross-site scripting. Data is read from **location** and passed to **$()** via the following statements:

- id = 'growls-' + location;
- return $('body:not(:has(#' + id + '))').append('<div id="' + id + '" />');

Note: The exploitability of this issue might depend on the specific version of jQuery that is being used.

Issue background

DOM-based vulnerabilities arise when a client-side script reads data from a controllable part of the DOM (for example, the URL) and processes this data in an unsafe way.

DOM-based cross-site scripting arises when a script writes controllable data into the HTML document in an unsafe way. An attacker may be able to use the vulnerability to construct a URL that, if visited by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.

The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

Users can be induced to visit the attacker's crafted URL in various ways, similar to the usual attack delivery vectors for reflected cross-site scripting vulnerabilities.

Burp Suite automatically identifies this issue using static code analysis, which may lead to false positives that are not actually exploitable. The relevant code and execution paths should be reviewed to determine whether this vulnerability is indeed present, or whether mitigations are in place that would prevent exploitation.

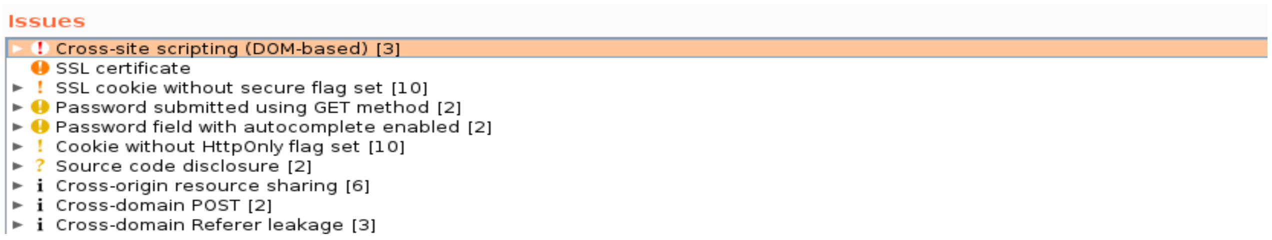
How Cross-Site Scripting (DOM Based) works:

1. **User Input Processed by Client-Side JavaScript**: The website includes JavaScript code that dynamically updates the DOM based on user input or URL parameters. For example, it might read values from the URL, cookies, or form inputs to personalize content, display data, or perform other actions.

2. **Manipulation of the DOM**: The JavaScript code uses methods like document.write(), innerHTML, eval(), or location.href to insert or modify content in the DOM. If this content is derived from user input without proper sanitization, it creates an opportunity for XSS.

3. **Malicious Input Injection**: An attacker crafts a malicious input, such as a specially crafted URL or form data, containing JavaScript code. For instance, the attacker might insert a script tag or a snippet of JavaScript code into a URL parameter or form field.

4. **Execution of Malicious Script**: When the user interacts with the website (e.g., clicking a link or submitting a form), the vulnerable JavaScript processes the malicious input and dynamically updates the DOM. Since the input wasn't properly sanitized or validated, the embedded script is executed within the user's browser.



Issues

```
► ! Cross-site scripting (DOM-based) [3]
  ! SSL certificate
► ! SSL cookie without secure flag set [10]
► ! Password submitted using GET method [2]
► ! Password field with autocomplete enabled [2]
► ! Cookie without HttpOnly flag set [10]
► ? Source code disclosure [2]
► i Cross-origin resource sharing [6]
► i Cross-domain POST [2]
► i Cross-domain Referer leakage [3]
```

How to prevent this attack:

1. Sanitize and Validate User Input

- **Sanitize Input**: Ensure that any user input inserted into the DOM is properly sanitized to remove any potential malicious code. Use libraries like DOMPurify to clean up user input before inserting it into the DOM.

2. Avoid Insecure DOM Manipulation

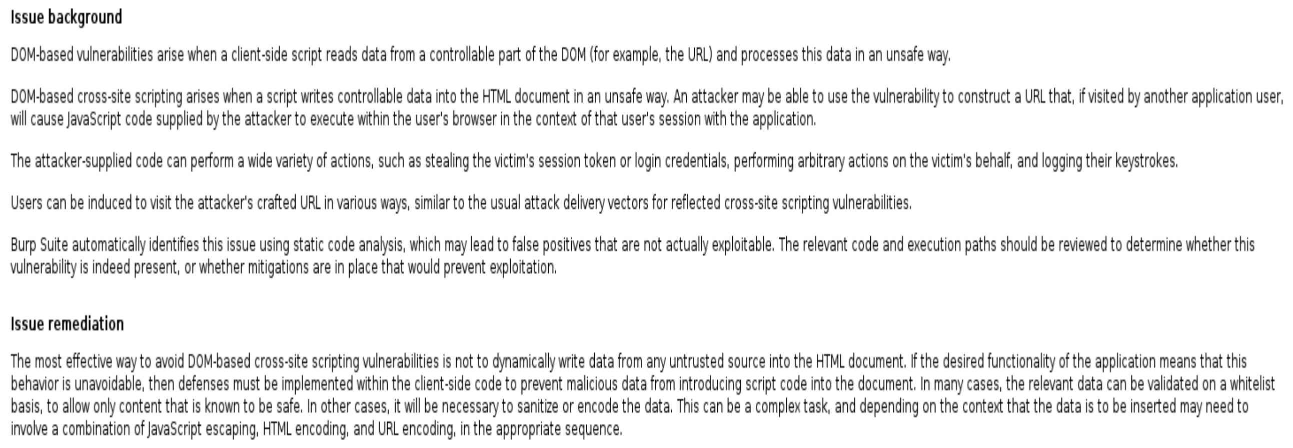
- **Avoid innerHTML and document.write()**: These methods directly inject content into the DOM and can execute HTML and JavaScript. Instead, methods like textContent, innerText, or setAttribute() insert text without interpreting it as HTML or JavaScript.

3. Use Content Security Policy (CSP)

- **Implement a Strict CSP**: CSP is a browser feature that helps prevent XSS by restricting the sources from which scripts can be loaded and executed. By implementing a strong CSP, you can mitigate the impact of XSS attacks even if they occur. 4. Use Security Libraries and Frameworks

Remedies:

The following are the remedies which are given by the burp suite tool itself.



Issue background

DOM-based vulnerabilities arise when a client-side script reads data from a controllable part of the DOM (for example, the URL) and processes this data in an unsafe way.

DOM-based cross-site scripting arises when a script writes controllable data into the HTML document in an unsafe way. An attacker may be able to use the vulnerability to construct a URL that, if visited by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.

The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

Users can be induced to visit the attacker's crafted URL in various ways, similar to the usual attack delivery vectors for reflected cross-site scripting vulnerabilities.

Burp Suite automatically identifies this issue using static code analysis, which may lead to false positives that are not actually exploitable. The relevant code and execution paths should be reviewed to determine whether this vulnerability is indeed present, or whether mitigations are in place that would prevent exploitation.

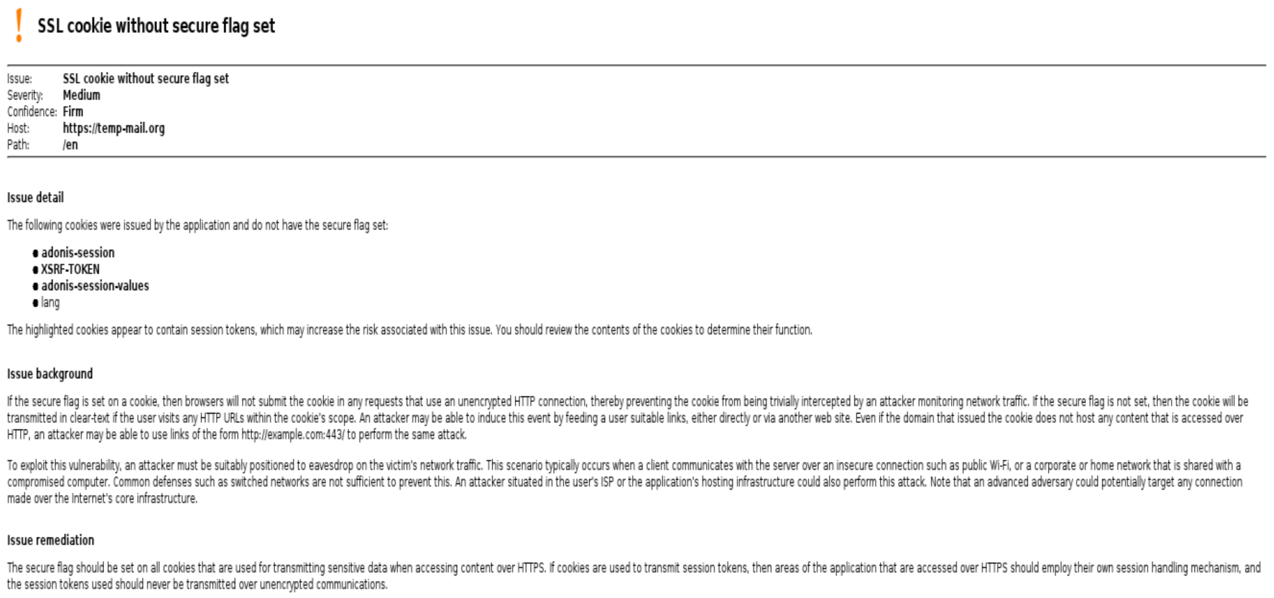
Issue remediation

The most effective way to avoid DOM-based cross-site scripting vulnerabilities is not to dynamically write data from any untrusted source into the HTML document. If the desired functionality of the application means that this behavior is unavoidable, then defenses must be implemented within the client-side code to prevent malicious data from introducing script code into the document. In many cases, the relevant data can be validated on a whitelist basis, to allow only content that is known to be safe. In other cases, it will be necessary to sanitize or encode the data. This can be a complex task, and depending on the context that the data is to be inserted may need to involve a combination of JavaScript escaping, HTML encoding, and URL encoding, in the appropriate sequence.

SSL cookie without secure flag set:

An SSL cookie is a cookie transmitted over a Secure Sockets Layer (SSL) or Transport Layer Security (TLS) connection, which ensures that the data sent between the user's browser and the web server is encrypted. However, if this cookie is not set with the Secure flag, it could be exposed to potential security risks, even if the connection itself is secure.

We are using the **Burp Suite professional tool** to perform the vulnerability assessment.



SSL cookie without secure flag set

Issue:	**SSL cookie without secure flag set**
Severity:	**Medium**
Confidence:	**Firm**
Host:	**https://temp-mail.org**
Path:	**/en**

Issue detail

The following cookies were issued by the application and do not have the secure flag set:

- **adonis-session**
- **XSRF-TOKEN**
- **adonis-session-values**
- lang

The highlighted cookies appear to contain session tokens, which may increase the risk associated with this issue. You should review the contents of the cookies to determine their function.

Issue background

If the secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic. If the secure flag is not set, then the cookie will be transmitted in clear-text if the user visits any HTTP URLs within the cookie's scope. An attacker may be able to induce this event by feeding a user suitable links, either directly or via another web site. Even if the domain that issued the cookie does not host any content that is accessed over HTTP, an attacker may be able to use links of the form http://example.com:443/ to perform the same attack.

To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Issue remediation

The secure flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications.

If a temporary email website (temp mail) uses cookies to manage user sessions or store other sensitive data but fails to set the Secure flag on these cookies, it exposes users to significant security risks.

How SSL cookie without a secure flag set works:

When a cookie is transmitted over a Secure Sockets Layer (SSL) or Transport Layer Security (TLS) connection (i.e., an HTTPS connection), it is encrypted during transit, providing confidentiality and integrity to the data being exchanged. However, if a cookie does not have the Secure flag set, it can also be sent over an unencrypted HTTP connection.

1. **Setting the Cookie**:
   - The server sends the cookie to the browser with the Set-Cookie HTTP header. If the Secure flag is **not** included, the cookie is considered a regular cookie and can be sent over both HTTPS and HTTP connections.

2. **Transmission Over HTTPS**:

- When the user interacts with the website over HTTPS, the cookie is transmitted securely. The data, including the cookie, is encrypted, protecting it from eavesdroppers and ensuring its integrity.

3. **Man-in-the-Middle (MitM) Attack**:

- In an environment where an attacker has control over the network (e.g., public Wi-Fi), they can perform a MitM attack, intercepting the HTTP request and extracting the cookie.

4. **Session Hijacking**:

- The intercepted cookie, if it contains a session ID or authentication token, can be used by the attacker to hijack the user's session. The attacker can then perform actions as if they were the user, such as accessing email, accounts, or other sensitive services.

<u>How to prevent this attack:</u>

1. **Set the Secure Flag on Cookies**

- The Secure flag ensures that the cookie is only sent over HTTPS connections. This prevents the cookie from being transmitted in plaintext over an insecure HTTP connection, which could be intercepted by an attacker.

2. **Enforce HTTPS Across the Entire Site**

   - Ensuring that your entire website operates over HTTPS eliminates the risk of cookies being sent over HTTP. This can be enforced by redirecting all HTTP requests to HTTPS and using HSTS (HTTP Strict Transport Security).

3. Use the HttpOnly Flag

- The HttpOnly flag prevents client-side scripts (e.g., JavaScript) from accessing the cookie. This mitigates the risk of Cross-Site Scripting (XSS) attacks stealing cookies.

4. **Implement the SameSite Attribute**

- The SameSite attribute controls whether cookies are sent with cross-site requests. This helps prevent Cross-Site Request Forgery (CSRF) attacks by ensuring that cookies are only sent in a first-party context.

5. **Regular Security Audits and Penetration Testing**

- Regular security audits and penetration testing help identify and address vulnerabilities, such as cookies being set without the Secure flag. Automated tools and manual reviews should be used to ensure all cookies are correctly configured.

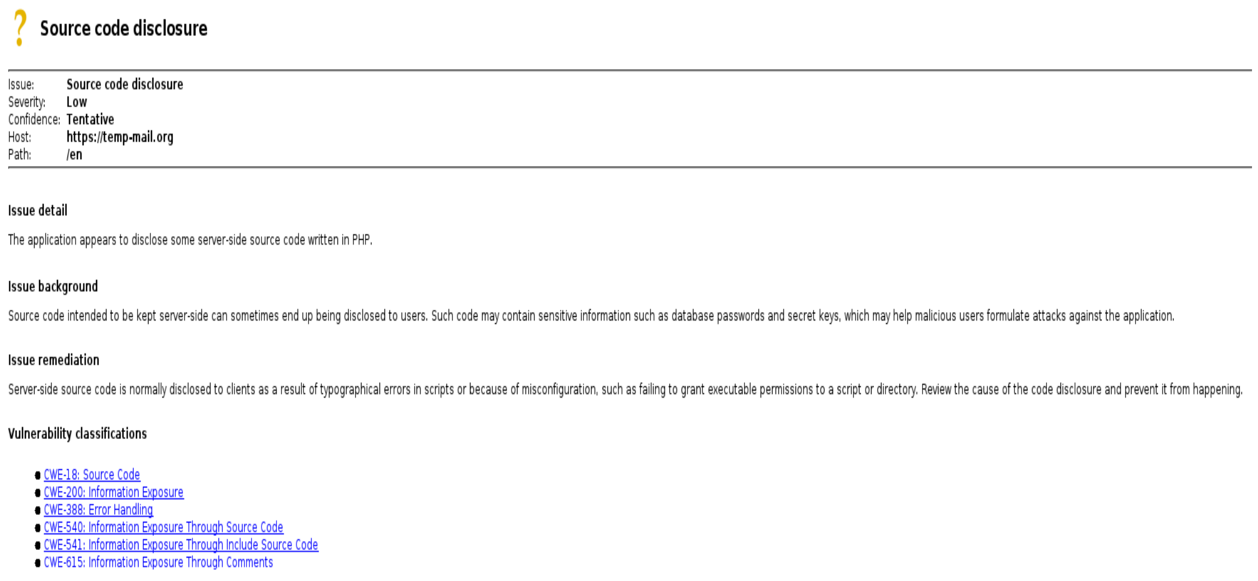
Remedies:

- **Sanitize Input:** Always sanitize and validate input data before processing it. Avoid using data directly from user input without proper filtering.
- **Secure JavaScript APIs:** Avoid using dangerous APIs like innerHTML, document.write(), and eval(). Use safer alternatives like textContent, innerText, or createElement().
- **Output Encoding:** Encode data before inserting it into the DOM. For instance, use appropriate encoding functions (e.g., JavaScript encoding, HTML encoding) based on the context.
- **Content Security Policy (CSP):** Implement CSP to restrict the sources from which scripts can be loaded and executed.
- **Avoid Inline JavaScript:** Avoid inline event handlers (e.g., onclick) and script blocks. Keep JavaScript code separate from HTML to reduce XSS risks.

Source Code Disclosure:

A Source Code Disclosure attack occurs when an attacker gains unauthorized access to the source code of a web application, software, or server. This access can reveal sensitive information, such as application logic, configuration details, API keys, database credentials, and other critical data that should not be publicly available. This type of attack can severely compromise the security of an application.

We are using the **Burp Suite professional tool** to perform the vulnerability assessment.



? Source code disclosure

Issue:	**Source code disclosure**
Severity:	**Low**
Confidence:	**Tentative**
Host:	**https://temp-mail.org**
Path:	**/en**

Issue detail

The application appears to disclose some server-side source code written in PHP.

Issue background

Source code intended to be kept server-side can sometimes end up being disclosed to users. Such code may contain sensitive information such as database passwords and secret keys, which may help malicious users formulate attacks against the application.

Issue remediation

Server-side source code is normally disclosed to clients as a result of typographical errors in scripts or because of misconfiguration, such as failing to grant executable permissions to a script or directory. Review the cause of the code disclosure and prevent it from happening.

Vulnerability classifications

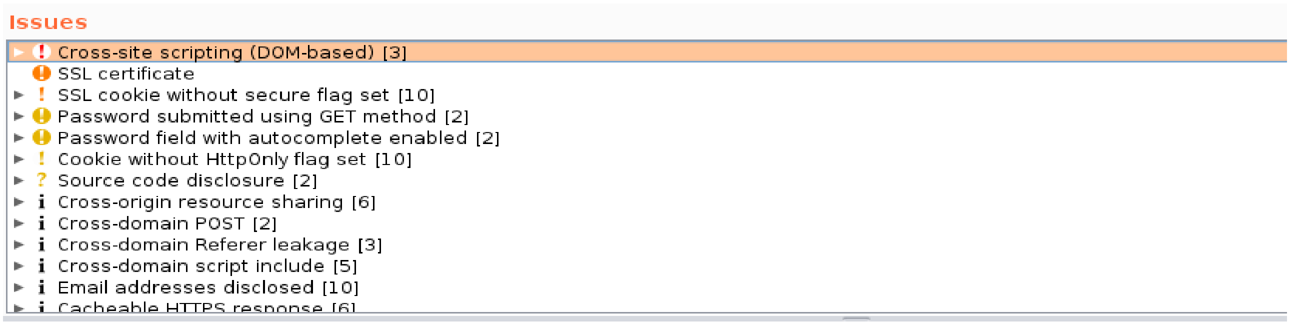
- CWE-18: Source Code
- CWE-200: Information Exposure
- CWE-388: Error Handling
- CWE-540: Information Exposure Through Source Code
- CWE-541: Information Exposure Through Include Source Code
- CWE-615: Information Exposure Through Comments

A Source Code Disclosure attack on a tempmail website involves the exposure of the website's source code, which can include sensitive information such as configuration files, API keys, database credentials, and the underlying logic of the application. This vulnerability can arise from improper server configurations, insecure coding practices, or flaws in web server software.

How Source Code Disclosure attack works:

A Source Code Disclosure attack works by exploiting vulnerabilities or misconfigurations in a web application, server, or version control system to gain access to the application's source code.

- **Finding Vulnerabilities:** Attackers look for misconfigured servers, exposed backup files, accessible version control directories (.git, .svn), or detailed error messages.
- **Exploiting File Access:** They directly access files through URLs (e.g., index.php.bak) or leverage Local/Remote File Inclusion (LFI/RFI) vulnerabilities to view source code.
- **Analyzing Code:** The exposed code is analyzed to find sensitive information (e.g., credentials, API keys) or flaws that can be further exploited.
- **Further Attacks:** Using the disclosed information, attackers can perform more severe attacks like database breaches, logic bypasses, or code injections.



Issues

```
▶ ! Cross-site scripting (DOM-based) [3]
  ! SSL certificate
▶ ! SSL cookie without secure flag set [10]
▶ ! Password submitted using GET method [2]
▶ ! Password field with autocomplete enabled [2]
▶ ! Cookie without HttpOnly flag set [10]
▶ ? Source code disclosure [2]
▶ i Cross-origin resource sharing [6]
▶ i Cross-domain POST [2]
▶ i Cross-domain Referer leakage [3]
▶ i Cross-domain script include [5]
▶ i Email addresses disclosed [10]
▶ i Cacheable HTTPS response [6]
```

<u>How to prevent this attack:</u>

- **Secure Server Configuration:** Disable directory listing, set proper file permissions, and ensure servers execute rather than serve source files.
- **Remove Sensitive Files:** Regularly delete backup, old, and temporary files, and keep sensitive files outside web-accessible directories.
- **Hide Error Details:** Disable debug mode in production, use generic error messages, and prevent exposure of code snippets in errors.
- **Protect Version Control:** Ensure .git and similar directories are inaccessible from the web.
- **Regular Audits and Code Reviews:** Conduct security audits and code reviews to identify and fix vulnerabilities.

- **Remedies**:
- **Fix Server Configurations:** Disable directory listing and ensure code files execute, not serve as text.
- **Remove Exposed Files:** Delete backup, temporary, and old files; keep sensitive files out of the web root.
- **Restrict Version Control Access:** Block access to `.git`, `.svn`, etc., using server rules.
- **Secure Error Handling:** Disable debug mode and hide detailed error messages.
- **Implement Access Controls:** Set strict file permissions and protect sensitive directories.
- **Regular Security Audits:** Perform regular audits and scans to identify and fix vulnerabilities.