Sri Lanka Institute of Information Technology

Web Security - IE2062

Bug Bounty Report 5

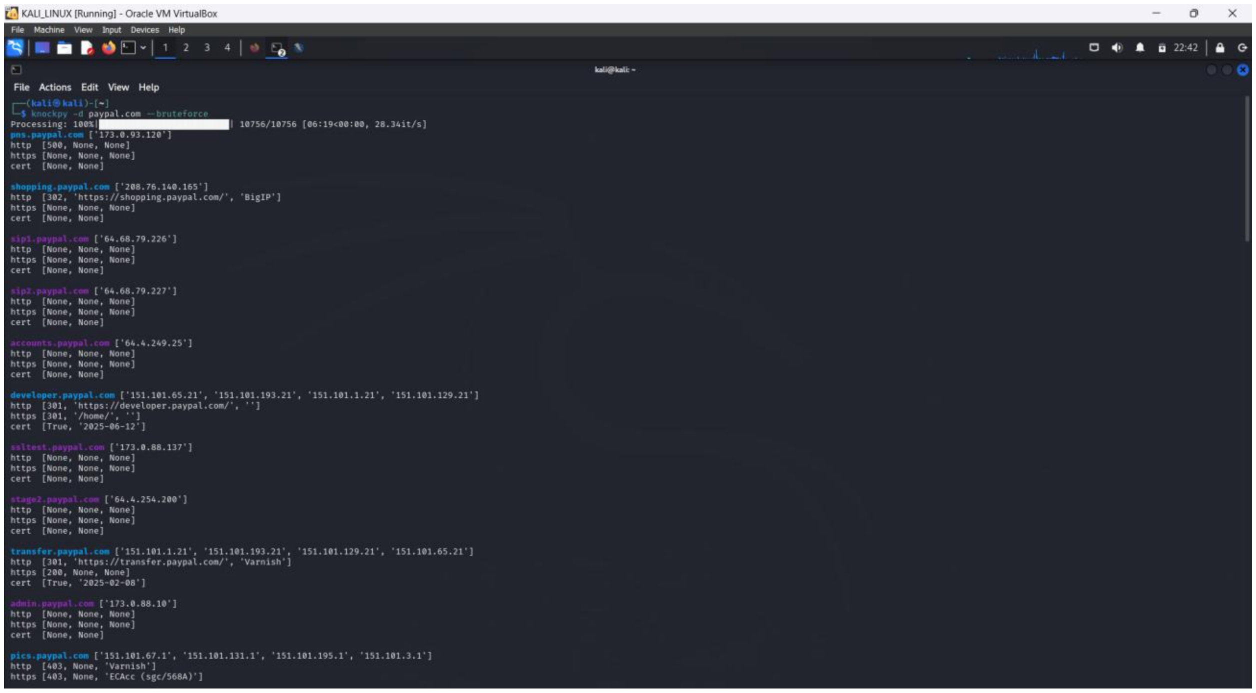
PERERA A.P.J

IT22280992

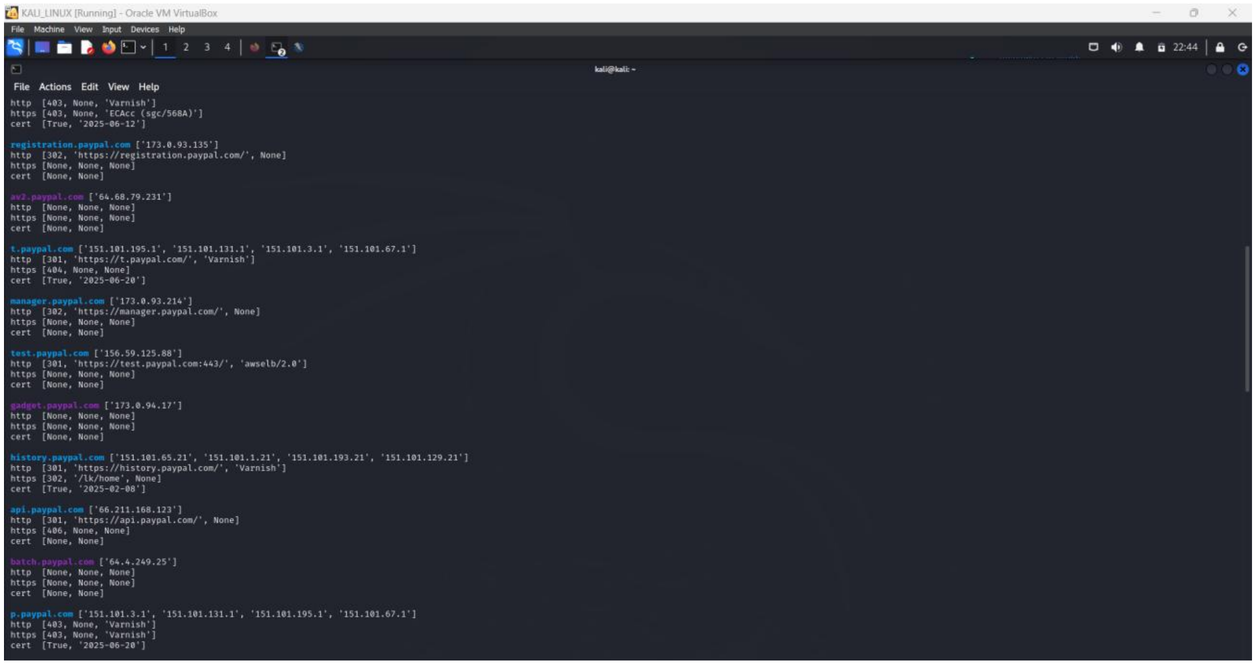
Group Y2S2.CS

Table of Contents

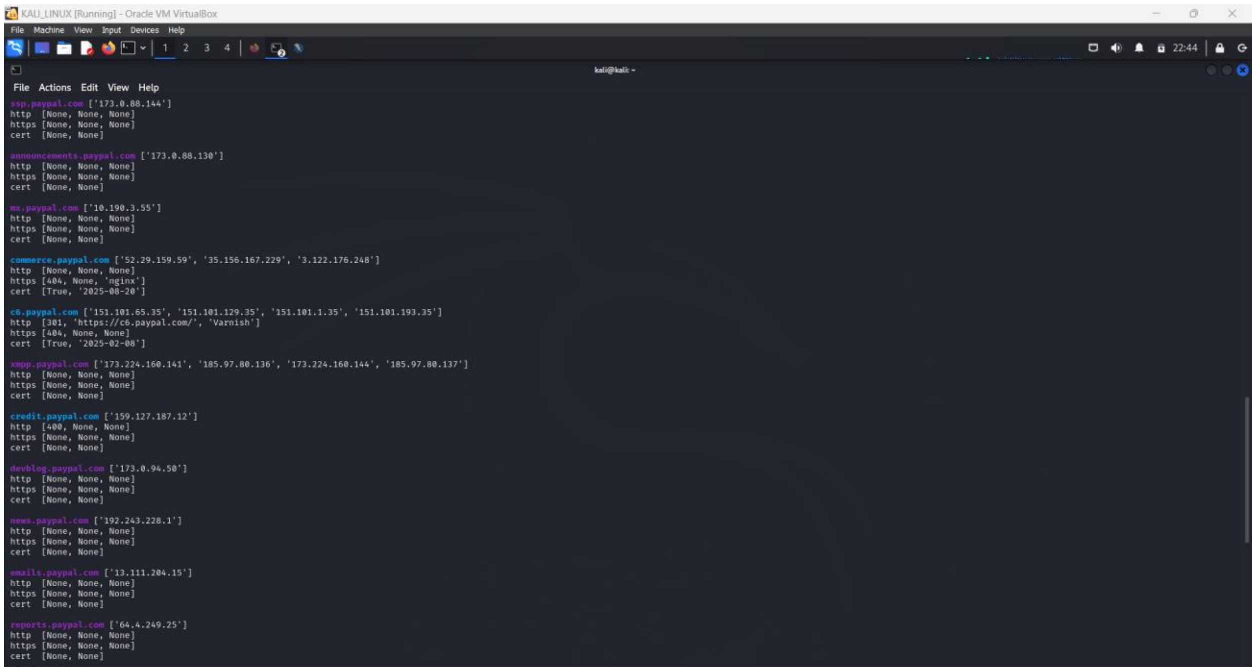
Using Knockpy – d paypal.com –bruteforce found 41 domains.



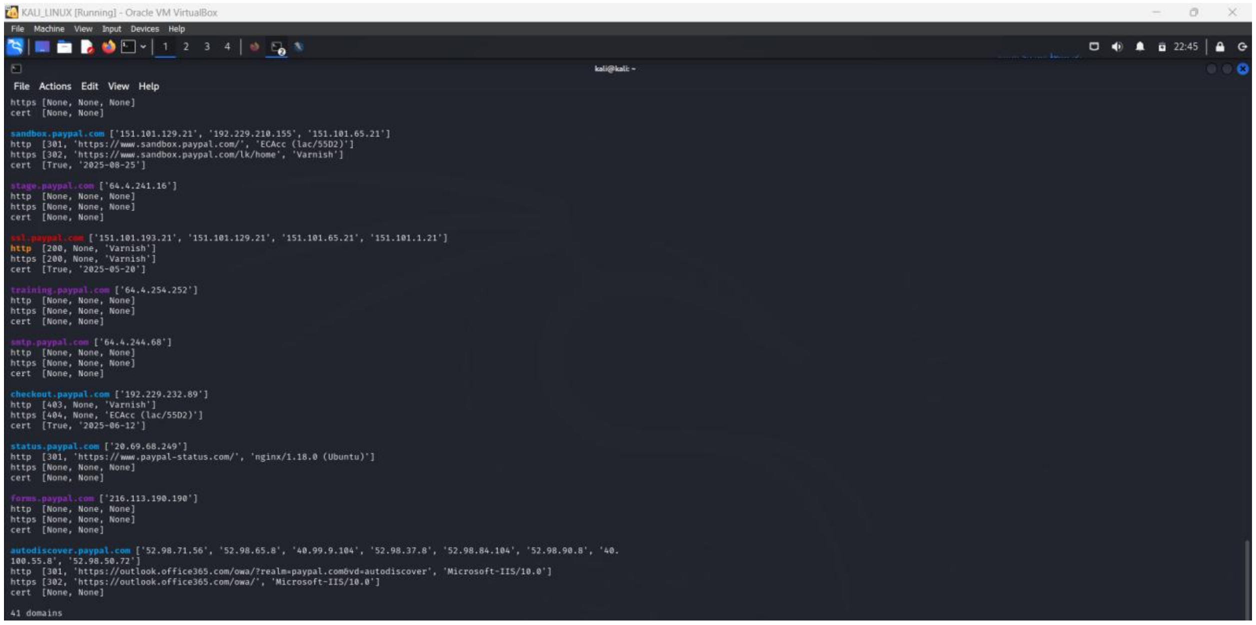
```
┌──(kali㉿kali)-[~]
└─$ knockpy -d paypal.com --bruteforce
Processing: 100%|████████████████████| 10756/10756 [06:19<00:00, 28.34it/s]
pns.paypal.com ['173.0.93.120']
http  [500, None, None]
https [None, None, None]
cert  [None, None]

shopping.paypal.com ['208.76.140.165']
http  [302, 'https://shopping.paypal.com/', 'BigIP']
https [None, None, None]
cert  [None, None]

sip1.paypal.com ['64.68.79.226']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

sip2.paypal.com ['64.68.79.227']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

accounts.paypal.com ['64.4.249.25']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

developer.paypal.com ['151.101.65.21', '151.101.193.21', '151.101.1.21', '151.101.129.21']
http  [301, 'https://developer.paypal.com/', '']
https [301, '/home/', '']
cert  [True, '2025-06-12']

ssltest.paypal.com ['173.0.88.137']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

stage2.paypal.com ['64.4.254.200']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

transfer.paypal.com ['151.101.1.21', '151.101.193.21', '151.101.129.21', '151.101.65.21']
http  [301, 'https://transfer.paypal.com/', 'Varnish']
https [200, None, None]
cert  [True, '2025-02-08']

admin.paypal.com ['173.0.88.10']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

pics.paypal.com ['151.101.67.1', '151.101.131.1', '151.101.195.1', '151.101.3.1']
http  [403, None, 'Varnish']
https [403, None, 'ECAcc (sgc/568A)']
```



```
http  [403, None, 'Varnish']
https [403, None, 'ECAcc (sgc/568A)']
cert  [True, '2025-06-12']

registration.paypal.com ['173.0.93.135']
http  [302, 'https://registration.paypal.com/', None]
https [None, None, None]
cert  [None, None]

av2.paypal.com ['64.68.79.231']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

t.paypal.com ['151.101.195.1', '151.101.131.1', '151.101.3.1', '151.101.67.1']
http  [301, 'https://t.paypal.com/', 'Varnish']
https [404, None, None]
cert  [True, '2025-06-20']

manager.paypal.com ['173.0.93.214']
http  [302, 'https://manager.paypal.com/', None]
https [None, None, None]
cert  [None, None]

test.paypal.com ['156.59.125.88']
http  [301, 'https://test.paypal.com:443/', 'awselb/2.0']
https [None, None, None]
cert  [None, None]

gadget.paypal.com ['173.0.94.17']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

history.paypal.com ['151.101.65.21', '151.101.1.21', '151.101.193.21', '151.101.129.21']
http  [301, 'https://history.paypal.com/', 'Varnish']
https [302, '/lk/home', None]
cert  [True, '2025-02-08']

api.paypal.com ['66.211.168.123']
http  [301, 'https://api.paypal.com/', None]
https [406, None, None]
cert  [None, None]

batch.paypal.com ['64.4.249.25']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

p.paypal.com ['151.101.3.1', '151.101.131.1', '151.101.195.1', '151.101.67.1']
http  [403, None, 'Varnish']
https [403, None, 'Varnish']
cert  [True, '2025-06-20']
```

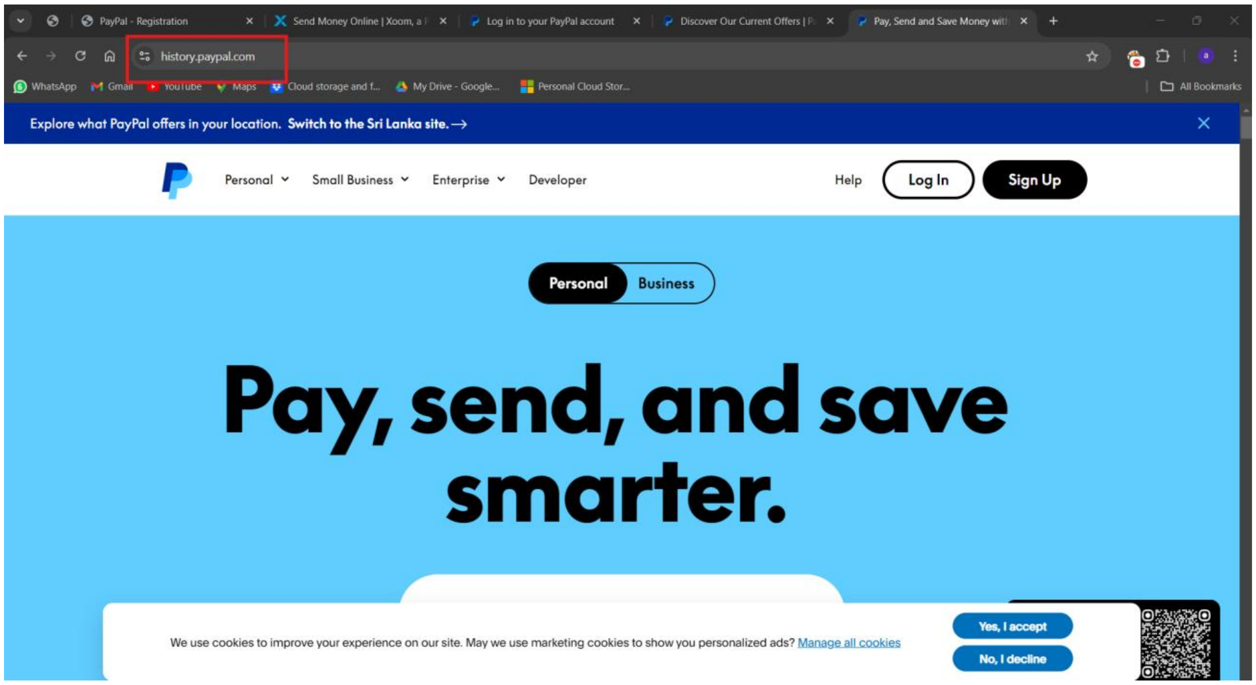


```
ssp.paypal.com ['173.0.88.144']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

announcements.paypal.com ['173.0.88.130']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

mx.paypal.com ['10.190.3.55']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

commerce.paypal.com ['52.29.159.59', '35.156.167.229', '3.122.176.248']
http  [None, None, None]
https [404, None, 'nginx']
cert  [True, '2025-08-20']

c6.paypal.com ['151.101.65.35', '151.101.129.35', '151.101.1.35', '151.101.193.35']
http  [301, 'https://c6.paypal.com/', 'Varnish']
https [404, None, None]
cert  [True, '2025-02-08']

xmpp.paypal.com ['173.224.160.141', '185.97.80.136', '173.224.160.144', '185.97.80.137']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

credit.paypal.com ['159.127.187.12']
http  [400, None, None]
https [None, None, None]
cert  [None, None]

devblog.paypal.com ['173.0.94.50']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

news.paypal.com ['192.243.228.1']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

emails.paypal.com ['13.111.204.15']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

reports.paypal.com ['64.4.249.25']
http  [None, None, None]
https [None, None, None]
cert  [None, None]
```



```
https [None, None, None]
cert  [None, None]

sandbox.paypal.com ['151.101.129.21', '192.229.210.155', '151.101.65.21']
http  [301, 'https://www.sandbox.paypal.com/', 'ECAcc (lac/55D2)']
https [302, 'https://www.sandbox.paypal.com/lk/home', 'Varnish']
cert  [True, '2025-08-25']

stage.paypal.com ['64.4.241.16']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

ssl.paypal.com ['151.101.193.21', '151.101.129.21', '151.101.65.21', '151.101.1.21']
http  [200, None, 'Varnish']
https [200, None, 'Varnish']
cert  [True, '2025-05-20']

training.paypal.com ['64.4.254.252']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

smtp.paypal.com ['64.4.244.68']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

checkout.paypal.com ['192.229.232.89']
http  [403, None, 'Varnish']
https [404, None, 'ECAcc (lac/55D2)']
cert  [True, '2025-06-12']

status.paypal.com ['20.69.68.249']
http  [301, 'https://www.paypal-status.com/', 'nginx/1.18.0 (Ubuntu)']
https [None, None, None]
cert  [None, None]

forms.paypal.com ['216.113.190.190']
http  [None, None, None]
https [None, None, None]
cert  [None, None]

autodiscover.paypal.com ['52.98.71.56', '52.98.65.8', '40.99.9.104', '52.98.37.8', '52.98.84.104', '52.98.90.8', '40.
100.55.8', '52.98.50.72']
http  [301, 'https://outlook.office365.com/owa/?realm=paypal.com&vd=autodiscover', 'Microsoft-IIS/10.0']
https [302, 'https://outlook.office365.com/owa/', 'Microsoft-IIS/10.0']
cert  [None, None]

41 domains
```

Target: http://history.paypal.com



Explore what PayPal offers in your location. Switch to the Sri Lanka site. →

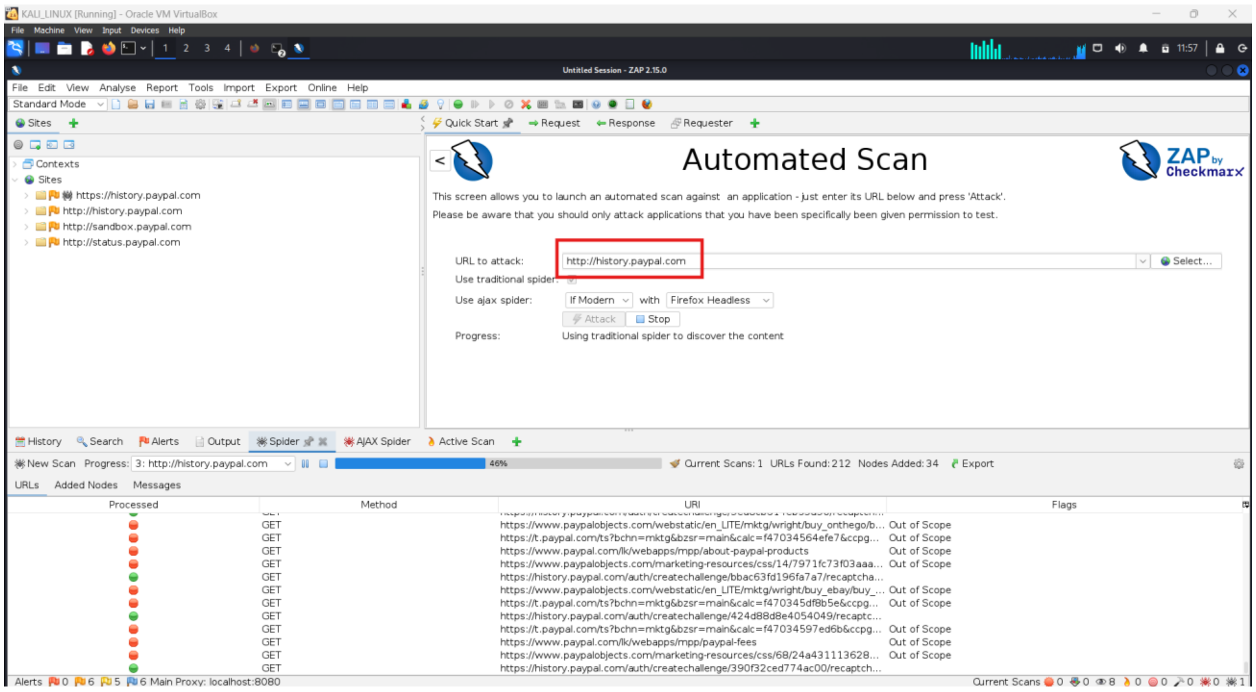
Pay, send, and save smarter.

We use cookies to improve your experience on our site. May we use marketing cookies to show you personalized ads? Manage all cookies

1.Vulnerability

1.1 Vulnerability title

- Spring4Shell.



KALI_LINUX [Running] - Oracle VM VirtualBox

Untitled Session - ZAP 2.15.0

Sites
- Contexts
- Sites
  - https://history.paypal.com
  - http://history.paypal.com
  - http://sandbox.paypal.com
  - http://status.paypal.com

Automated Scan

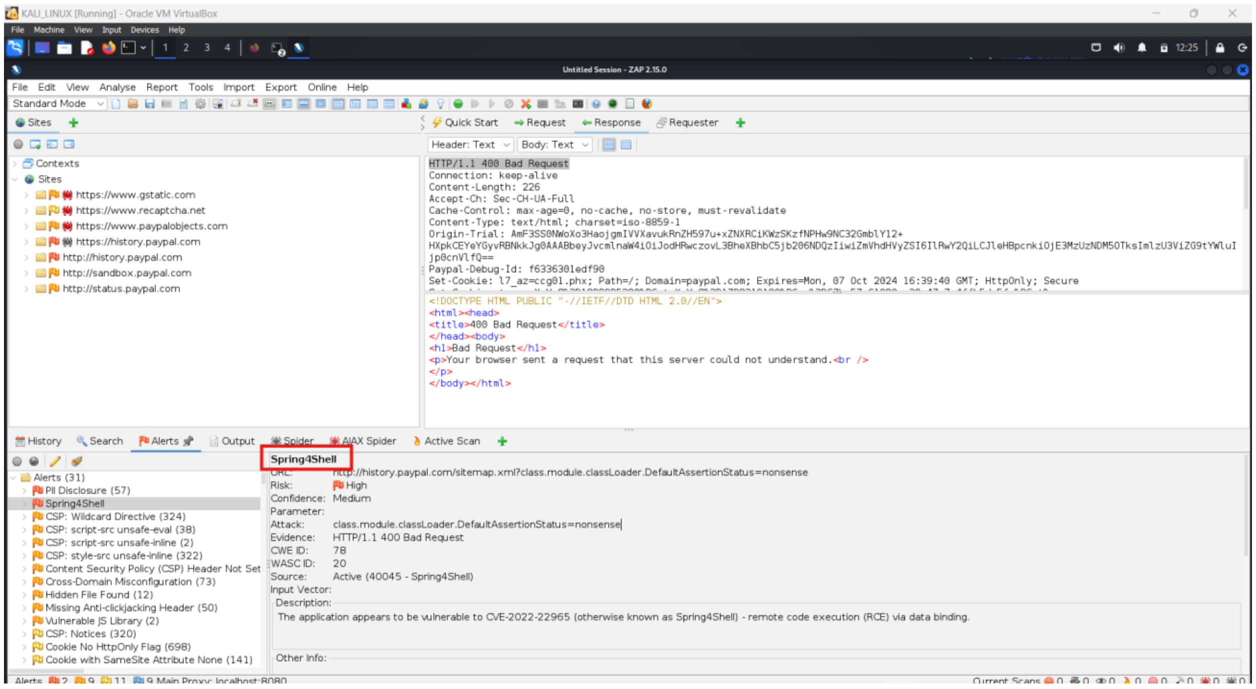
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://history.paypal.com
Use traditional spider: ✓
Use ajax spider: If Modern with Firefox Headless
Progress: Using traditional spider to discover the content

New Scan Progress: 3: http://history.paypal.com 46% Current Scans: 1 URLs Found: 212 Nodes Added: 34 Export

Processed	Method	URI	Flags
🟥	GET	https://www.paypalobjects.com/webstatic/en_LITE/mktg/wright/buy_onthego/b...	Out of Scope
🟥	GET	https://t.paypal.com/ts?bchn=mktg&bzsr=main&calc=f47034564efe7&ccpg...	Out of Scope
🟥	GET	https://www.paypal.com/lk/webapps/mpp/about-paypal-products	Out of Scope
🟥	GET	https://www.paypalobjects.com/marketing-resources/css/14/7971fc73f03aaa...	Out of Scope
🟩	GET	https://history.paypal.com/auth/createchallenge/bbac63fd196fa7a7/recaptcha...	
🟥	GET	https://www.paypalobjects.com/webstatic/en_LITE/mktg/wright/buy_ebay/buy_...	Out of Scope
🟥	GET	https://t.paypal.com/ts?bchn=mktg&bzsr=main&calc=f470345df8b5e&ccpg...	Out of Scope
🟩	GET	https://history.paypal.com/auth/createchallenge/424d88d8e4054049/recaptc...	
🟥	GET	https://t.paypal.com/ts?bchn=mktg&bzsr=main&calc=f47034597ed6b&ccpg...	Out of Scope
🟥	GET	https://www.paypal.com/lk/webapps/mpp/paypal-fees	Out of Scope
🟥	GET	https://www.paypalobjects.com/marketing-resources/css/68/24a431113628...	Out of Scope
🟩	GET	https://history.paypal.com/auth/createchallenge/390f32ced774ac00/recaptch...	

Alerts 0 6 5 6 Main Proxy: localhost:8080


KALI_LINUX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Untitled Session - ZAP 2.15.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Header: Text
Body: Text
History
Search
Alerts
Output
Spider
AJAX Spider
Active Scan

Contexts
Sites
- https://www.gstatic.com
- https://www.recaptcha.net
- https://www.paypalobjects.com
- https://history.paypal.com
- http://history.paypal.com
- http://sandbox.paypal.com
- http://status.paypal.com

```
HTTP/1.1 400 Bad Request
Connection: keep-alive
Content-Length: 226
Accept-Ch: Sec-CH-UA-Full
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Content-Type: text/html; charset=iso-8859-1
Origin-Trial: AmF3SS0NWoXo3HaojgmIVVXavukRnZH597u+xZNXRCiKWzSKzfNPHw9NC32GmblY12+
HXpkCEYeYGyvRBNkkJg0AAABbeyJvcmlnaW4iOiJodHRwczovL3BheXBhbC5jb206NDQzIiwiZmVhdHVyZSI6IlRwY2QiLCJleHBpcnkiOjE3MzUzNDM5OTksImlzU3ViZG9tYWluI
jp0cnVlfQ==
Paypal-Debug-Id: f6336301edf90
Set-Cookie: l7_az=ccg01.phx; Path=/; Domain=paypal.com; Expires=Mon, 07 Oct 2024 16:39:40 GMT; HttpOnly; Secure
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
</body></html>
```

- Alerts (31)
  - PII Disclosure (57)
  - Spring4Shell
  - CSP: Wildcard Directive (324)
  - CSP: script-src unsafe-eval (38)
  - CSP: script-src unsafe-inline (2)
  - CSP: style-src unsafe-inline (322)
  - Content Security Policy (CSP) Header Not Set
  - Cross-Domain Misconfiguration (73)
  - Hidden File Found (12)
  - Missing Anti-clickjacking Header (50)
  - Vulnerable JS Library (2)
  - CSP: Notices (320)
  - Cookie No HttpOnly Flag (698)
  - Cookie with SameSite Attribute None (141)

Spring4Shell

URL:	http://history.paypal.com/sitemap.xml?class.module.classLoader.DefaultAssertionStatus=nonsense
Risk:	High
Confidence:	Medium
Parameter:	
Attack:	class.module.classLoader.DefaultAssertionStatus=nonsense
Evidence:	HTTP/1.1 400 Bad Request
CWE ID:	78
WASC ID:	20
Source:	Active (40045 - Spring4Shell)
Input Vector:	

Description:
The application appears to be vulnerable to CVE-2022-22965 (otherwise known as Spring4Shell) - remote code execution (RCE) via data binding.

Other Info:

Alerts 2 9 11 9 Main Proxy: localhost:8080

1.2 Vulnerability description

- Spring4Shell is a security issue identified as CVE-2022-22965 that constitutes a security risk to users of the common java spring-core framework especially those whose applications are deployed on JDK 9 and above to run as WAR files on tomcat servers. With respect to its domain, this vulnerabilities enables attackers to execute any code of their choice on the compromised systems including the capability of taking over the entire servers in question. This vulnerability is caused by an inherent problem in spring's data binding mechanism in its backend layers within the application. This weakness, when provoked allows the attackers to send specially crafted requests to the server and tamper with the properties of certain objects. By doing that, the attackers are able send code that alters with the servers classes, thus causing code to execute on the machine and in some cases allowing the person to upload virus, get data from the machine or even sit on the machine without permission for a long period. Because of its destructive nature, Spring4Shell is not the only vulnerability that has drawn this much concern and is even referred to as the Log4Shell vulnerability, hence the need for organizations utilizing affected Spring Core framework versions to patch and mitigate as soon as possible.

1.3 Affected components

The Spring4Shell vulnerability primarily affects applications with the following components and configurations:

1. Spring Core Framework:
   - Specifically, versions prior to 5.3.18 and 5.2.20 are affected. Applications using these older versions of the Spring Core library are vulnerable.

2. Java Development Kit (JDK):
   - The vulnerability affects applications running on JDK 9 or higher. Earlier versions of the JDK are not susceptible due to differences in Java class structures and class loader behaviors.

3. Apache Tomcat:
   - The application must be deployed as a WAR file (Web Application Archive) on an Apache Tomcat server. This setup is common in enterprise environments where web applications are served by Tomcat.
   - Standalone Spring Boot applications (e.g., those using embedded Tomcat) are generally not vulnerable unless they have specific configurations that allow for the exploitation.

4. Exposed Endpoints for Parameter Binding:
   - The vulnerability targets endpoints that use @RequestMapping or @PostMapping annotations, where parameter binding is performed on an object.
   - Applications that do not directly expose data-binding endpoints are at a lower risk but could still be indirectly affected.

1.4 Impact assessment

The impact of the Spring4Shell vulnerability (CVE-2022-22965) is significant due to its potential to enable remote code execution (RCE) on affected systems. Below are key areas of impact:

1. Remote Code Execution (RCE)

- Attackers can exploit this vulnerability to execute arbitrary code on the server. With RCE, attackers can install malware, modify files, or gain persistent access to the server.
- This access can allow them to deploy backdoors or control the application and the underlying server, putting other applications and services at risk.

2. Data Breach and Confidentiality Risks

- Attackers with RCE capabilities can access sensitive data stored on the server, such as user information, database credentials, and configuration files.
- Sensitive business data or personally identifiable information (PII) could be exfiltrated, leading to potential privacy violations and compliance issues (e.g., with GDPR, HIPAA, etc.).

3. Availability Risks

- Attackers could disrupt the availability of the application or server by deploying malware or using resources maliciously.
- This could lead to Denial-of-Service (DoS) conditions or make the application inaccessible, affecting business operations and causing financial losses.

4. Integrity Risks

- Since attackers can potentially write files to the server, they could tamper with legitimate application files or insert malicious ones.
- This could result in altered application behavior, unauthorized changes to data, or the introduction of Trojanized versions of the software, making it difficult to trust the integrity of the system.

5. Privilege Escalation

- If exploited in conjunction with other vulnerabilities, Spring4Shell could allow attackers to escalate privileges, gaining higher-level access than they initially had.

- Privilege escalation could give attackers control over system-level resources, allowing for more extensive attacks on the network or connected systems.

6. Reputation and Financial Impact

- A successful exploit of Spring4Shell can lead to significant reputational damage, as organizations may face backlash from customers, partners, and the public.
- In addition to the potential financial costs of remediation and recovery, affected organizations may face regulatory fines and legal expenses if sensitive data is compromised.

7. Compliance and Legal Risks

- Organizations that fail to address this vulnerability could face non-compliance issues with regulatory standards (like GDPR, CCPA, etc.), especially if sensitive data is compromised.
- Legal actions may be taken if the attack affects customers or partners, leading to potential lawsuits, fines, and penalties.

8. Long-Term Security Risks

- Once exploited, attackers might establish persistent access or install backdoors, making it easier for them to re-compromise the server even after initial remediation.
- This could lead to further attacks in the future, especially if an attacker remains undetected for extended periods.

1.5 Steps to reproduce

1. Environment Setup:
    - Java Development Kit (JDK) version 9 or higher (Spring4Shell only affects JDK 9+).
    - A vulnerable Spring Core Framework version (prior to 5.3.18 or 5.2.20).
    - Apache Tomcat server with an application deployed as a WAR file (Web Application Archive).
2. Sample Spring Application:
    - Create a simple Spring MVC application that uses @RequestMapping or @PostMapping in a controller to handle HTTP requests.
3. Testing Tools:
    - Use a tool like cURL or Postman to send HTTP requests. Alternatively, you can use Burp Suite or similar tools for crafting HTTP payloads.

Step-by-Step Exploitation Process

Note: Follow these steps in a controlled test environment. Unauthorized exploitation or testing on live systems is illegal and unethical.

Step 1: Set Up the Vulnerable Spring Application

1. Deploy the sample Spring MVC application on an Apache Tomcat server using a vulnerable Spring Core version.
2. Ensure the application is accessible via a web browser to confirm it is up and running.

Step 2: Identify an Exposed Endpoint with Parameter Binding

1. Identify an endpoint in the Spring application that uses @RequestMapping or @PostMapping annotations.
2. This endpoint should bind request parameters to a Java object. In the above example, DataBindingObject is an object whose properties will be set based on request parameters.

Step 3: Send a Malicious HTTP Request

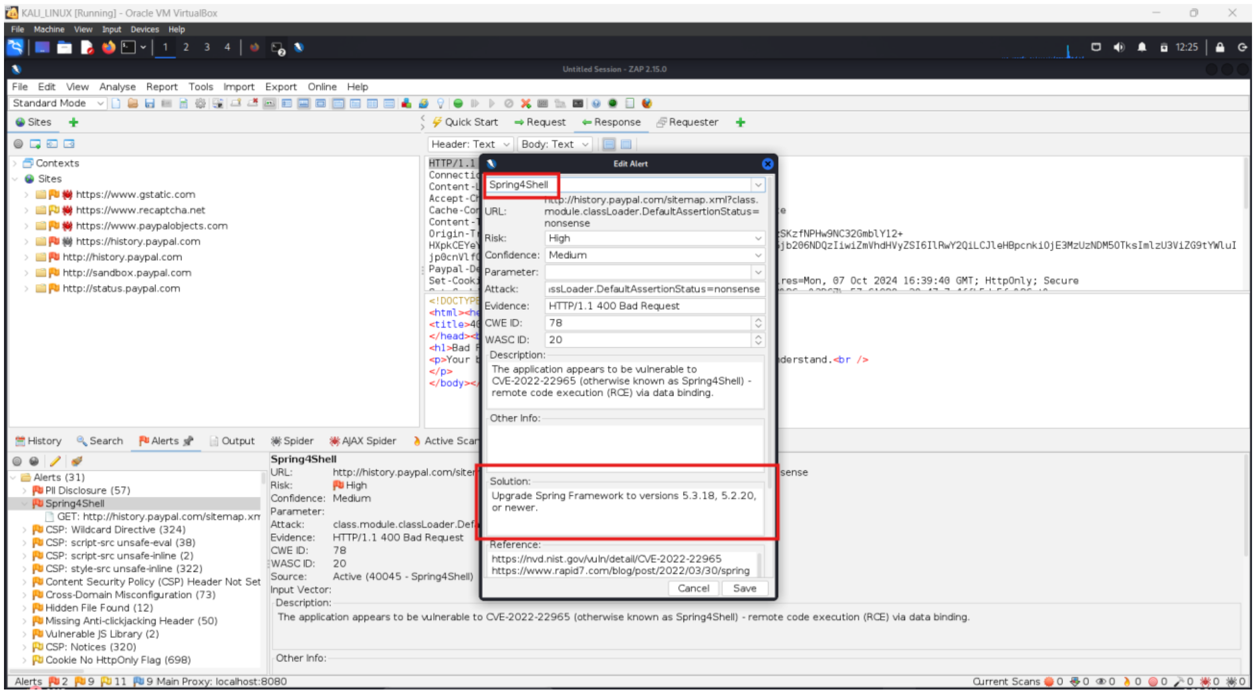
1. Use a tool like cURL to send a crafted request that exploits the vulnerability. In this example, you will try to modify the classLoader property:

This example attempts to manipulate the Spring application's classLoader resources and overwrite certain properties. You might see different variations of the payload depending on your exact setup and version.

Step 4: Check for Successful Exploitation

1. If successful, the crafted payload will allow an attacker to create a file, execute commands, or manipulate resources within the server.
2. Examine your server logs or the /tmp directory for evidence that the payload was executed.
3. For more precise control and debugging, a tool like Burp Suite may assist in refining the payload until it triggers the intended behavior.

1.6 Proof of concept (if applicable)



Untitled Session - ZAP 2.15.0

Edit Alert

Spring4Shell

URL: http://history.paypal.com/sitemap.xml?class.module.classLoader.DefaultAssertionStatus=nonsense

Risk: High

Confidence: Medium

Parameter:

Attack: ıssLoader.DefaultAssertionStatus=nonsense

Evidence: HTTP/1.1 400 Bad Request

CWE ID: 78

WASC ID: 20

Description:
The application appears to be vulnerable to CVE-2022-22965 (otherwise known as Spring4Shell) - remote code execution (RCE) via data binding.

Other Info:

Solution:
Upgrade Spring Framework to versions 5.3.18, 5.2.20, or newer.

Reference:
https://nvd.nist.gov/vuln/detail/CVE-2022-22965
https://www.rapid7.com/blog/post/2022/03/30/spring

Alerts (31)
- PII Disclosure (57)
- Spring4Shell
  - GET: http://history.paypal.com/sitemap.xm
- CSP: Wildcard Directive (324)
- CSP: script-src unsafe-eval (38)
- CSP: script-src unsafe-inline (2)
- CSP: style-src unsafe-inline (322)
- Content Security Policy (CSP) Header Not Set
- Cross-Domain Misconfiguration (73)
- Hidden File Found (12)
- Missing Anti-clickjacking Header (50)
- Vulnerable JS Library (2)
- CSP: Notices (320)
- Cookie No HttpOnly Flag (698)

Spring4Shell
URL: http://history.paypal.com/siter
Risk: High
Confidence: Medium
Parameter:
Attack: class.module.classLoader.Def
Evidence: HTTP/1.1 400 Bad Request
CWE ID: 78
WASC ID: 20
Source: Active (40045 - Spring4Shell)
Input Vector:
Description:
The application appears to be vulnerable to CVE-2022-22965 (otherwise known as Spring4Shell) - remote code execution (RCE) via data binding.
Other Info:

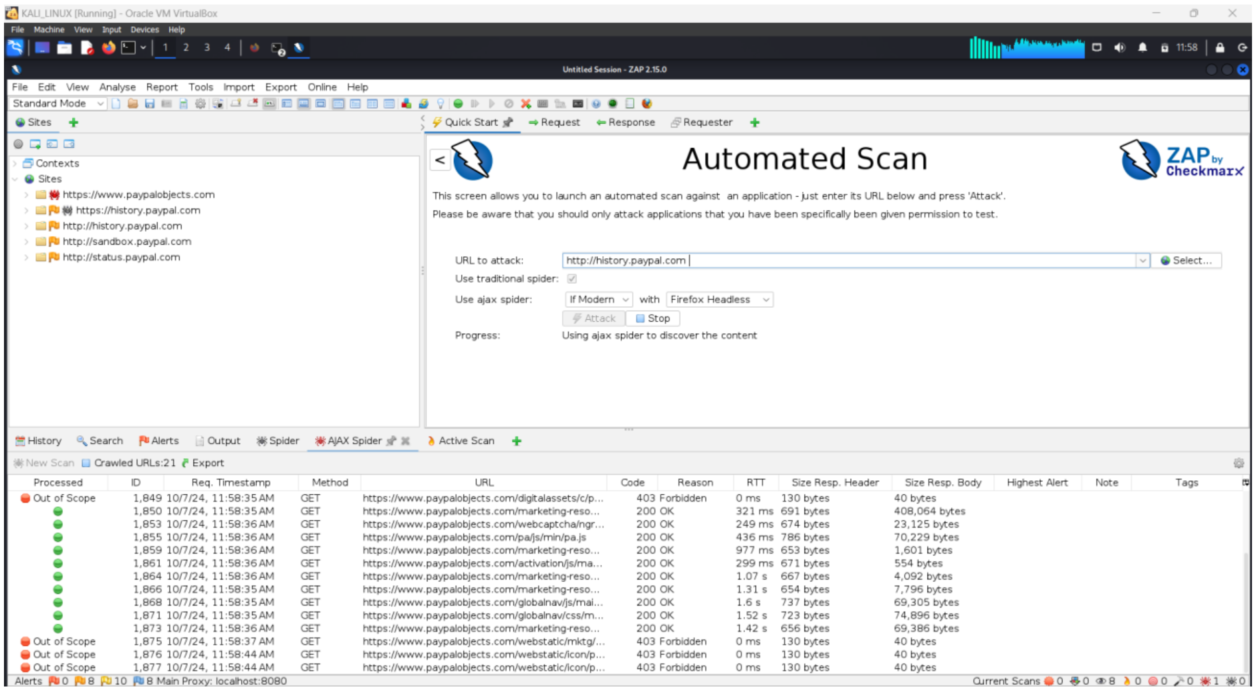
1.7 Proposed mitigation or fix

- Upgrade spring Framework to versions 5.3.8.18,5.2.20, or newer.
- Upgrade java Versoin.
- Use Alternative Deployment Methods.
- Apply security Patches to Apache Tomcat.
- Use a web Application Firewall(WAF).
- Implement Strong Access Control.
- Disable Binding.
- Of Specific Properties.
- Use security tools and Conduct Regular Scans.
- Monitor Logs Malicious Activity.
- Isolate in Vulnerable systems.

2. Vulnerability

2.1 Vulnerability title

- X-Content-type-option Header Missing.



KALI_LINUX [Running] - Oracle VM VirtualBox

Untitled Session - ZAP 2.15.0

Sites:
- Contexts
- Sites
  - https://www.paypalobjects.com
  - https://history.paypal.com
  - http://history.paypal.com
  - http://sandbox.paypal.com
  - http://status.paypal.com

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://history.paypal.com

Use traditional spider: ✓

Use ajax spider: If Modern with Firefox Headless

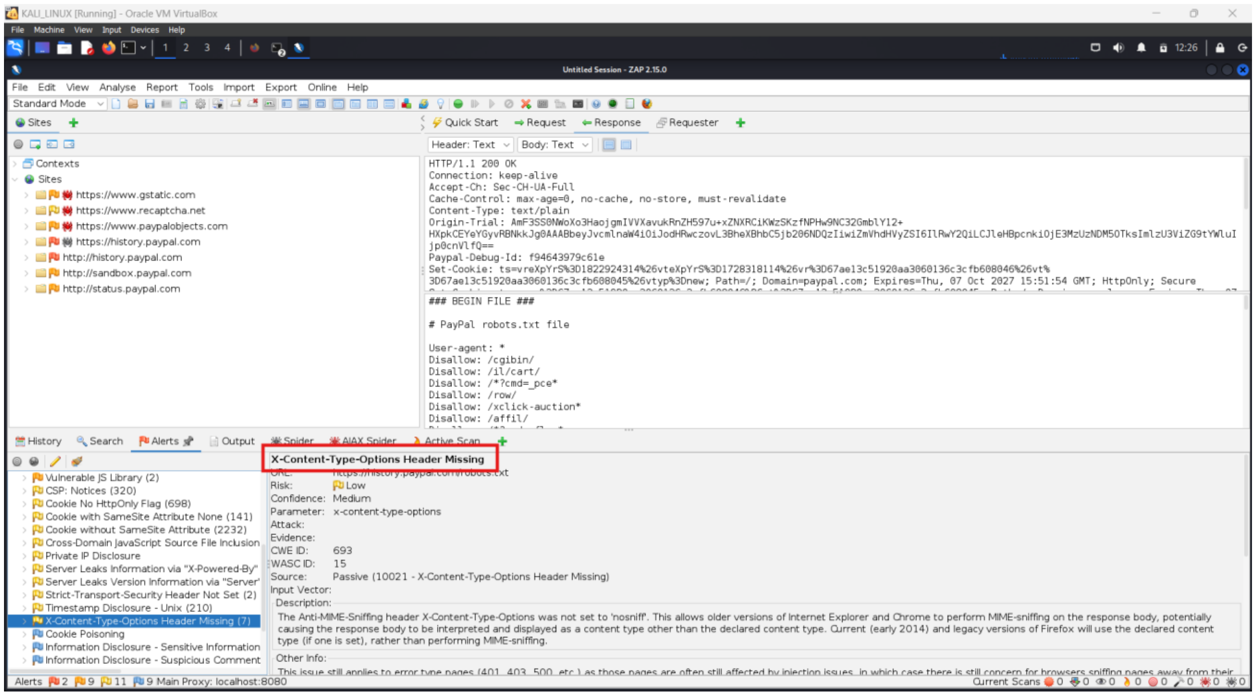
Progress: Using ajax spider to discover the content

Crawled URLs:21

Processed	ID	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	Note	Tags
🟥 Out of Scope	1,849	10/7/24, 11:58:35 AM	GET	https://www.paypalobjects.com/digitalassets/c/p...	403	Forbidden	0 ms	130 bytes	40 bytes			
🟩	1,850	10/7/24, 11:58:35 AM	GET	https://www.paypalobjects.com/marketing-reso...	200	OK	321 ms	691 bytes	408,064 bytes			
🟩	1,853	10/7/24, 11:58:36 AM	GET	https://www.paypalobjects.com/webcaptcha/ngr...	200	OK	249 ms	674 bytes	23,125 bytes			
🟩	1,855	10/7/24, 11:58:36 AM	GET	https://www.paypalobjects.com/pa/js/min/pa.js	200	OK	436 ms	786 bytes	70,229 bytes			
🟩	1,859	10/7/24, 11:58:36 AM	GET	https://www.paypalobjects.com/marketing-reso...	200	OK	977 ms	653 bytes	1,601 bytes			
🟩	1,861	10/7/24, 11:58:36 AM	GET	https://www.paypalobjects.com/activation/js/ma...	200	OK	299 ms	671 bytes	554 bytes			
🟩	1,864	10/7/24, 11:58:36 AM	GET	https://www.paypalobjects.com/marketing-reso...	200	OK	1.07 s	667 bytes	4,092 bytes			
🟩	1,866	10/7/24, 11:58:35 AM	GET	https://www.paypalobjects.com/marketing-reso...	200	OK	1.31 s	654 bytes	7,796 bytes			
🟩	1,868	10/7/24, 11:58:35 AM	GET	https://www.paypalobjects.com/globalnav/js/mai...	200	OK	1.6 s	737 bytes	69,305 bytes			
🟩	1,871	10/7/24, 11:58:35 AM	GET	https://www.paypalobjects.com/globalnav/css/m...	200	OK	1.52 s	723 bytes	74,896 bytes			
🟩	1,873	10/7/24, 11:58:36 AM	GET	https://www.paypalobjects.com/marketing-reso...	200	OK	1.42 s	656 bytes	69,386 bytes			
🟥 Out of Scope	1,875	10/7/24, 11:58:37 AM	GET	https://www.paypalobjects.com/webstatic/mktg/...	403	Forbidden	0 ms	130 bytes	40 bytes			
🟥 Out of Scope	1,876	10/7/24, 11:58:44 AM	GET	https://www.paypalobjects.com/webstatic/icon/p...	403	Forbidden	0 ms	130 bytes	40 bytes			
🟥 Out of Scope	1,877	10/7/24, 11:58:44 AM	GET	https://www.paypalobjects.com/webstatic/icon/p...	403	Forbidden	0 ms	130 bytes	40 bytes			

Alerts 0 8 10 8 Main Proxy: localhost:8080

Current Scans 0 0 8 0 0 0 1 0


KALI_LINUX [Running] - Oracle VM VirtualBox
Untitled Session - ZAP 2.15.0
Standard Mode
Sites
Quick Start
Request
Response
Requester
Header: Text
Body: Text
Contexts
Sites
https://www.gstatic.com
https://www.recaptcha.net
https://www.paypalobjects.com
https://history.paypal.com
http://history.paypal.com
http://sandbox.paypal.com
http://status.paypal.com
History
Search
Alerts
Output
Spider
AJAX Spider
Active Scan

```
HTTP/1.1 200 OK
Connection: keep-alive
Accept-Ch: Sec-CH-UA-Full
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Content-Type: text/plain
Origin-Trial: AmF3SS0NWoXo3HaojgmIVVXavukRnZH597u+xZNXRCiKWzSKzfNPHw9NC32GmblY12+
HXpkCEYeYGyvRBNkkJg0AAABbeyJvcmlnaW4iOiJodHRwczovL3BheXBhbC5jb206NDQzIiwiZmVhdHVyZSI6IlRwY2QiLCJleHBpcnkiOjE3MzUzNDM5OTksImlzU3ViZG9tYWluI
jp0cnVlfQ==
Paypal-Debug-Id: f94643979c61e
Set-Cookie: ts=vreXpYrS%3D1822924314%26vteXpYrS%3D1728318114%26vr%3D67ae13c51920aa3060136c3cfb608046%26vt%
3D67ae13c51920aa3060136c3cfb608045%26vtyp%3Dnew; Path=/; Domain=paypal.com; Expires=Thu, 07 Oct 2027 15:51:54 GMT; HttpOnly; Secure
```

```
### BEGIN FILE ###

# PayPal robots.txt file

User-agent: *
Disallow: /cgibin/
Disallow: /il/cart/
Disallow: /*?cmd=_pce*
Disallow: /row/
Disallow: /xclick-auction*
Disallow: /affil/
```

- Vulnerable JS Library (2)
- CSP: Notices (320)
- Cookie No HttpOnly Flag (698)
- Cookie with SameSite Attribute None (141)
- Cookie without SameSite Attribute (2232)
- Cross-Domain JavaScript Source File Inclusion
- Private IP Disclosure
- Server Leaks Information via "X-Powered-By"
- Server Leaks Version Information via "Server"
- Strict-Transport-Security Header Not Set (2)
- Timestamp Disclosure - Unix (210)
- X-Content-Type-Options Header Missing (7)
- Cookie Poisoning
- Information Disclosure - Sensitive Information
- Information Disclosure - Suspicious Comment

X-Content-Type-Options Header Missing

URL:	https://history.paypal.com/robots.txt
Risk:	Low
Confidence:	Medium
Parameter:	x-content-type-options
Attack:	
Evidence:	
CWE ID:	693
WASC ID:	15
Source:	Passive (10021 - X-Content-Type-Options Header Missing)
Input Vector:	

Description:

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

Other Info:

This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their

Alerts 2 9 11 9 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

2.2 Vulnerability description

- The risk associated with Missing the X-Content-Type-Options Header in web applications arises when there is no X-Content-Type-Options HTTP header sent with a value of nosniff in a web application. This header is necessary not to allow the browser to guess the kind of file it is based on its contents which is called trying to sniff the MIME type. In the absence of this type of header, the browser may try to treat files differently from what the server intended, which in turn may lead to vulnerabilities such as XSS. For instance, an intruder would be able to inject arbitrary code into the browser by masking the code as an image file and including it within the page thus compromising the application and its users. The instruction X-Content-Type-Options: nosniff directs the web browser to use only the associated content type defined in the Content-Type header without any further guesses. Implementation of this simple feature disallowing MIME type interpretation will prevent most of the common types of security vulnerabilities.

2.3 Affected components

- When the X-Content-Type-Options Header Missing vulnerability exists, several components of a web application may be affected. The lack of this header can lead to unintended behaviors in various areas:

1. Web Browsers

   - MIME Type Sniffing: Browsers will attempt to determine the content type of files by analyzing their content rather than relying solely on the Content-Type header specified by the server. This can lead to browsers interpreting files in unintended ways, which could result in the execution of scripts or other hazardous content.
   - Cross-Site Scripting (XSS): Without X-Content-Type-Options: nosniff, browsers might treat files as executable scripts even if the Content-Type header indicates they are non-executable (like treating a text file as JavaScript). This is especially risky for user-uploaded content, which could be executed by other users' browsers.
   - File Download Vulnerabilities: Browsers may allow unsafe file downloads by misinterpreting MIME types. For instance, files intended as images could be interpreted as executables, increasing the risk of malware downloads.

2. Web Application Frontend

   - Static Resources: Resources like JavaScript, CSS, and images can be treated as executable code, potentially leading to unintended script execution or other harmful behavior.
   - Cross-Origin Resource Sharing (CORS): If other websites link to the application's resources, they may attempt to execute or misuse these resources, leading to security issues, particularly if the browser doesn't enforce the correct MIME type.

3. Content Delivery Networks (CDNs)

- Cached Content: CDNs may cache and deliver content with incorrect MIME types, allowing potentially dangerous content to be cached globally, which could harm users on a large scale.
- Shared Resources: The vulnerability may also affect shared resources across multiple sites, increasing the risk of cross-origin attacks if the content is misinterpreted.

4. Application Backend

- File Upload and Download Functionality: If users can upload content (like images, text files, etc.), attackers might upload files that the application will misinterpret. When other users access these files, their browsers could execute unintended scripts or code, leading to XSS or other attacks.
- API Responses: API responses, especially those delivering JSON or XML data, can be misinterpreted by the client if the X-Content-Type-Options header is missing, allowing potentially harmful content to be mistakenly treated as executable scripts.

5. Web Servers

- Content-Type Spoofing: The absence of X-Content-Type-Options: nosniff on a server can enable attackers to spoof the Content-Type header. For instance, they may disguise malicious code as an image file, which the browser will then execute.
- Error Pages and Custom Scripts: Many web servers use custom error pages and scripts. Without nosniff, these pages and scripts can be exploited, especially if any sensitive information is present or if they are accessible to users.

2.4 Impact assessment

- The impact assessment of a missing X-Content-Type-Options header depends on how the web application handles and serves its content, as well as the overall security posture of the application. Here's a breakdown of potential impacts:

1. Cross-Site Scripting (XSS) Attacks

- Impact: XSS vulnerabilities can allow attackers to inject and execute malicious scripts in users' browsers, leading to session hijacking, data theft, and the ability to impersonate the user.
- Risk Level: High, especially if user-uploaded content or dynamic content is improperly handled and could be interpreted as executable code.
- Example: An attacker uploads a file disguised as a plain text file or image, but the browser interprets it as JavaScript, allowing it to run malicious code.

2. MIME Type Confusion

- Impact: MIME type confusion can lead to misinterpretation of resources, where the browser treats non-executable content as executable. This can cause benign files, like images, to be executed as scripts.
- Risk Level: Medium to High, depending on the content types served by the application. Static files such as JavaScript, CSS, and images are particularly vulnerable.
- Example: A server mislabels a malicious JavaScript file as an image file, and without nosniff, a browser might still execute it as JavaScript.

3. Data Leakage and Privacy Risks

- Impact: In cases where sensitive information is present in files, MIME sniffing could lead to unintended exposure. Additionally, CORS could allow unauthorized access to resources if MIME types are misinterpreted.
- Risk Level: Medium, especially for applications handling sensitive data like personal, financial, or medical information.

- Example: A user downloads a confidential report marked as a text file, but the browser interprets it incorrectly, allowing unintended access or execution in ways that expose sensitive data.

4. File Download Vulnerabilities

- Impact: Improper MIME handling could lead to users inadvertently downloading and executing malicious files, potentially infecting their system with malware or ransomware.
- Risk Level: High for applications that provide downloadable content, particularly if the content is generated dynamically.
- Example: A file meant to be a PDF is sniffed and executed as an executable, potentially installing malware on the user’s device.

5. Compromise of Content Delivery Networks (CDNs)

- Impact: If the web application uses a CDN for content delivery, cached content may be served with incorrect MIME types to multiple users globally, spreading malicious or misinterpreted content at scale.
- Risk Level: High, as a CDN compromise could lead to mass distribution of malicious content.
- Example: A CDN caches a JavaScript file that has been misidentified as an image file. All users retrieving the cached content may have it interpreted and executed as malicious code.

6. Impact on SEO and Browser Compatibility

- Impact: Some browsers and search engines may penalize sites with insufficient security headers. This can affect the site's credibility and visibility in search results, leading to decreased traffic and user trust.
- Risk Level: Medium, primarily affecting user perception and SEO.
- Example: A search engine might flag the site as insecure, reducing its ranking and visibility in search results.

2.5 **Steps to reproduce**

Step 1: Identify the Target Application

- Choose a web application that you want to test. Ensure you have permission to perform security testing on it, especially if it is a production application.

Step 2: Analyze HTTP Responses

1. Open Developer Tools:

- In your web browser (e.g., Chrome, Firefox), open the Developer Tools (usually by right-clicking and selecting "Inspect" or pressing F12).

2. Navigate to the Network Tab:

- Go to the "Network" tab to monitor network requests.

3. Load the Application:

- Refresh the page or navigate to different parts of the application to trigger various HTTP requests.

4. Select a Request:

- Click on one of the requests made to the server (usually the main page request).

5. Inspect the Response Headers:

- In the details pane, look for the "Response Headers" section. Check for the presence of the X-Content-Type-Options header.

Step 3: Verify the Absence of the Header

- Check for the Header:
    - If you do not see the X-Content-Type-Options header listed in the response, the application is vulnerable to this issue.

Step 4: Test Content-Type Handling

1. Find or Create a Test File:

- Locate a file on the server that could be misinterpreted, such as a JavaScript file or a file upload functionality (e.g., an image upload).

2. Upload or Access the File:

- If you are testing file uploads, try uploading a file with a .txt or .jpg extension but containing malicious JavaScript code.

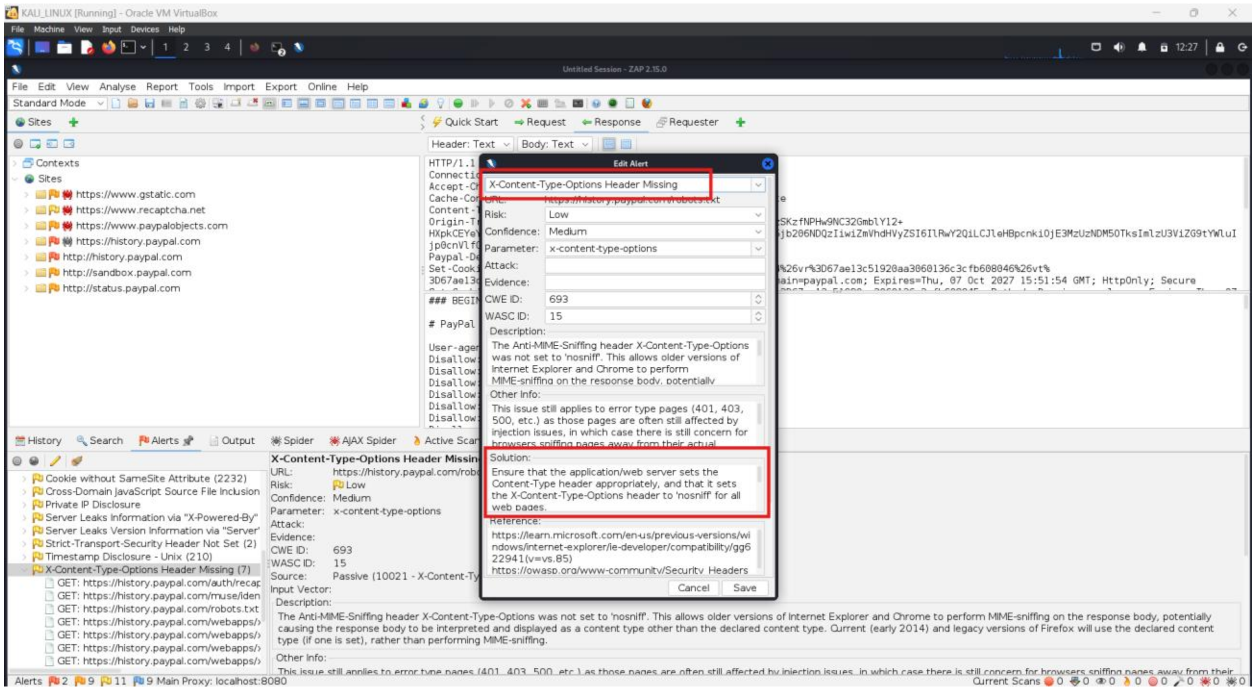
3. Check the Response for Content-Type:

- Once the file is uploaded or accessed, inspect the response headers again to see if the X-Content-Type-Options header is present or absent.

Step 5: Document Findings

- Record Your Observations:
- Document the steps taken, the HTTP responses received, and the absence of the X-Content-Type-Options header, noting any implications for security based on the observed behavior.

2.6 <u>Proof of concept (if applicable)</u>


KALI_LINUX [Running] - Oracle VM VirtualBox
Untitled Session - ZAP 2.15.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Header: Text
Body: Text
History
Search
Alerts
Output
Spider
AJAX Spider
Active Scan

Contexts
Sites
https://www.gstatic.com
https://www.recaptcha.net
https://www.paypalobjects.com
https://history.paypal.com
http://history.paypal.com
http://sandbox.paypal.com
http://status.paypal.com

Edit Alert

X-Content-Type-Options Header Missing

URL:	https://history.paypal.com/robots.txt
Risk:	Low
Confidence:	Medium
Parameter:	x-content-type-options
Attack:	
Evidence:	
CWE ID:	693
WASC ID:	15

Description:
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially

Other Info:
This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual

Solution:
Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.

Reference:
https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)
https://owasp.org/www-community/Security_Headers

Cancel Save

Cookie without SameSite Attribute (2232)
Cross-Domain JavaScript Source File Inclusion
Private IP Disclosure
Server Leaks Information via "X-Powered-By"
Server Leaks Version Information via "Server"
Strict-Transport-Security Header Not Set (2)
Timestamp Disclosure - Unix (210)
X-Content-Type-Options Header Missing (7)
GET: https://history.paypal.com/auth/recap
GET: https://history.paypal.com/muse/iden
GET: https://history.paypal.com/robots.txt
GET: https://history.paypal.com/webapps/
GET: https://history.paypal.com/webapps/
GET: https://history.paypal.com/webapps/
GET: https://history.paypal.com/webapps/

X-Content-Type-Options Header Missin

URL:	https://history.paypal.com/robo
Risk:	Low
Confidence:	Medium
Parameter:	x-content-type-options
Attack:	
Evidence:	
CWE ID:	693
WASC ID:	15
Source:	Passive (10021 - X-Content-Ty
Input Vector:	

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

Other Info:
This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their

Alerts 2 9 11 9 Main Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0

2.7 <u>Proposed mitigation or fix</u>

- Ensure that the application server sets the content-type header appropriately , and that it sets the x-content-Type-Option header to ‘nosniff’ for all web pages.
- Configure Web server Stings.
- Update application code.
- Verify the configuration.
- Review other security Headers.
- Conduct Regular Security Audits.