# WEB APPLICATION PENTESTING REPORT

*BY*

*M.SEENUVASAN*

# *OWASP Top 10 Application Security Risks - 2017*

## *A1:2017-Injection*

*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.*

## *A2:2017-Broken Authentication*

*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.*

## *A3:2017-Sensitive Data Exposure*

*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.*

## *A4:2017-XML External Entities (XXE)*

*Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.*

## *A5:2017-Broken Access Control*

*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.*

## *A6:2017-Security Misconfiguration*

*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.*

***A7:2017-Cross-Site Scripting (XSS)***

*XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijackuser sessions, deface web sites, or redirect the user to malicious sites.*

***A8:2017-Insecure Deserialization***

*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.*

***A9:2017-Using Components with Known Vulnerabilities***

*Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.*

***A10:2017-Insufficient Logging & Monitoring***

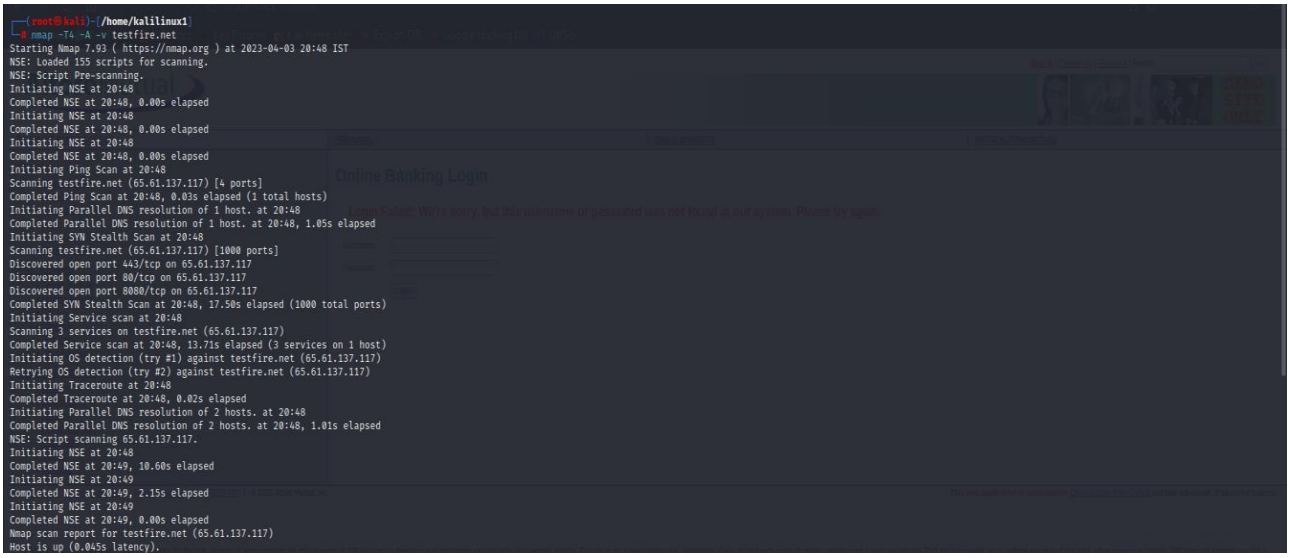
*Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.*

*TARGET URL: http://www.testfire.com*

*TARGET IP: 65.61.137.117*

# *SCAN REPORT:*

*Tools used of scanning the target website is nmap & whatweb*



```
┌──(root㉿kali)-[/home/kalilinux1]
└─# nmap -T4 -A -v testfire.net
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-03 20:48 IST
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 20:48
Completed NSE at 20:48, 0.00s elapsed
Initiating NSE at 20:48
Completed NSE at 20:48, 0.00s elapsed
Initiating NSE at 20:48
Completed NSE at 20:48, 0.00s elapsed
Initiating Ping Scan at 20:48
Scanning testfire.net (65.61.137.117) [4 ports]
Completed Ping Scan at 20:48, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:48
Completed Parallel DNS resolution of 1 host. at 20:48, 1.05s elapsed
Initiating SYN Stealth Scan at 20:48
Scanning testfire.net (65.61.137.117) [1000 ports]
Discovered open port 443/tcp on 65.61.137.117
Discovered open port 80/tcp on 65.61.137.117
Discovered open port 8080/tcp on 65.61.137.117
Completed SYN Stealth Scan at 20:48, 17.50s elapsed (1000 total ports)
Initiating Service scan at 20:48
Scanning 3 services on testfire.net (65.61.137.117)
Completed Service scan at 20:48, 13.71s elapsed (3 services on 1 host)
Initiating OS detection (try #1) against testfire.net (65.61.137.117)
Retrying OS detection (try #2) against testfire.net (65.61.137.117)
Initiating Traceroute at 20:48
Completed Traceroute at 20:48, 0.02s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 20:48
Completed Parallel DNS resolution of 2 hosts. at 20:48, 1.01s elapsed
NSE: Script scanning 65.61.137.117.
Initiating NSE at 20:48
Completed NSE at 20:49, 10.60s elapsed
Initiating NSE at 20:49
Completed NSE at 20:49, 2.15s elapsed
Initiating NSE at 20:49
Completed NSE at 20:49, 0.00s elapsed
Nmap scan report for testfire.net (65.61.137.117)
Host is up (0.045s latency).
```

```
80/tcp   open   http      Apache Tomcat/Coyote JSP engine 1.1
|_http-server-header: Apache-Coyote/1.1
|_http-title: Altoro Mutual
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
443/tcp  open   ssl/http  Apache Tomcat/Coyote JSP engine 1.1
|_http-server-header: Apache-Coyote/1.1
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_ssl-date: 2023-04-03T15:19:08+00:00; 0s from scanner time.
| ssl-cert: Subject: commonName=demo.testfire.net
| Subject Alternative Name: DNS:demo.testfire.net, DNS:altoromutual.com
| Issuer: commonName=Sectigo RSA Domain Validation Secure Server CA/organizationName=Sectigo Limited/stateOrProvinceName=Greater Manchester/countryName=GB
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2022-06-15T00:00:00
| Not valid after:  2023-07-16T23:59:59
| MD5:   96412f2943fb2c8e48008447da3e32f6
|_SHA-1: 97d684a78ac4047882bbead767b6d54ff6e79afe
|_http-title: Altoro Mutual
8080/tcp open   http      Apache Tomcat/Coyote JSP engine 1.1
|_http-server-header: Apache-Coyote/1.1
|_http-title: Altoro Mutual
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
8443/tcp closed https-alt
Device type: bridge|general purpose
Running (JUST GUESSING): Oracle Virtualbox (97%), QEMU (92%)
OS CPE: cpe:/o:oracle:virtualbox cpe:/a:qemu:qemu
Aggressive OS guesses: Oracle Virtualbox (97%), QEMU user mode network gateway (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=17 (Good luck!)
IP ID Sequence Generation: Incremental

TRACEROUTE (using port 80/tcp)
HOP RTT     ADDRESS
1   0.72 ms 10.0.2.2
2   0.82 ms 65.61.137.117

NSE: Script Post-scanning.
Initiating NSE at 20:49
Completed NSE at 20:49, 0.00s elapsed
Initiating NSE at 20:49
Completed NSE at 20:49, 0.00s elapsed
Initiating NSE at 20:49
Completed NSE at 20:49, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 56.41 seconds
```

```
┌──(root㉿kali)-[/home/kalilinux]
└─# whatweb -v www.testfire.net
WhatWeb report for http://www.testfire.net
Status    : 200 OK
Title     : Altoro Mutual
IP        : 65.61.137.117
Country   : UNITED STATES, US

Summary   : Apache, Cookies[JSESSIONID], HTTPServer[Apache-Coyote/1.1], HttpO
nly[JSESSIONID], Java

Detected Plugins:
[ Apache ]
        The Apache HTTP Server Project is an effort to develop and
        maintain an open-source HTTP server for modern operating
        systems including UNIX and Windows NT. The goal of this
        project is to provide a secure, efficient and extensible
        server that provides HTTP services in sync with the current
        HTTP standards.

        Google Dorks: (3)
        Website     : http://httpd.apache.org/

[ Cookies ]
        Display the names of cookies in the HTTP headers. The
        values are not returned to save on space.

        String      : JSESSIONID

[ HTTPServer ]
        HTTP server header string. This plugin also attempts to
        identify the operating system from the server header.

        String      : Apache-Coyote/1.1 (from server string)

[ HttpOnly ]
        If the HttpOnly flag is included in the HTTP set-cookie
        response header and the browser supports it then the cookie
        cannot be accessed through client side script - More Info:
        http://en.wikipedia.org/wiki/HTTP_cookie

        String      : JSESSIONID

[ Java ]
        Java allows you to play online games, chat with people
        around the world, calculate your mortgage interest, and
        view images in 3D, just to name a few. It's also integral
        to the intranet applications and other e-business solutions
        that are the foundation of corporate computing.

        Website     : http://www.java.com/
```

```
HTTP Headers:
        HTTP/1.1 200 OK
        Server: Apache-Coyote/1.1
        Set-Cookie: JSESSIONID=1CE4C76343F83907036BAC90A7C3D89D; Path=/; Http
Only
        Content-Type: text/html;charset=ISO-8859-1
        Transfer-Encoding: chunked
        Date: Fri, 07 Apr 2023 11:59:30 GMT
        Connection: close
```

*This gives the details about open port and server software that host this website*

# *MANUAL VULNERABLITY ASSEMENT*

*Based on owasp top 10 vulnerablity we scan the website manually .During this assement we found sever vulnerability like that we mentioned above in owasp top 10*

# *LIST OF VULNERABLITIES WE THAT FOUNDED*

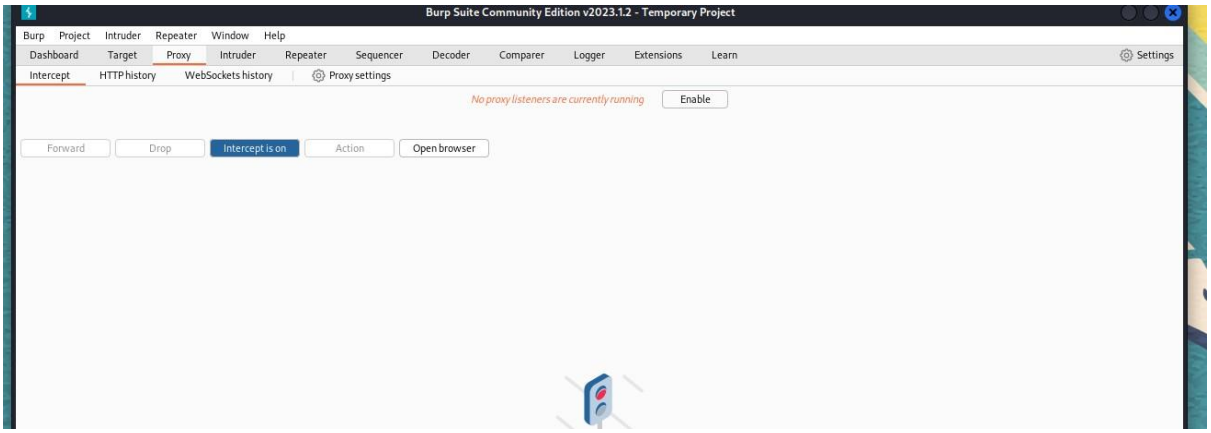
- *SQL INJECTION*
- *BROKEN AUTHENTICATION*
- *SENSITIVE DATA EXPOSURE*
- *SECURITY MISCONFIGURATION*
- *CROSS SITE SCRIPTING(XSS)*

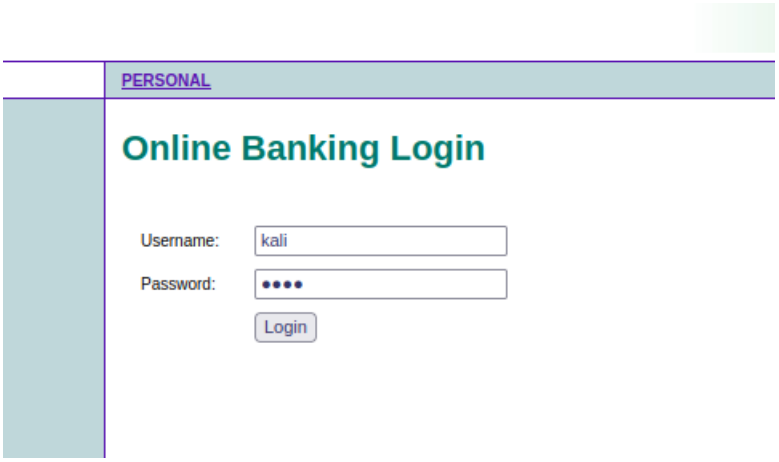
# ***SQL INJECTION***

*The basic form of SQL injection describes direct insertion of attacker-controlled data into variables that are used to construct SQL commands. As a result, an attacker can tamper with the original query by permanently terminating the string, appending new commands etc.*

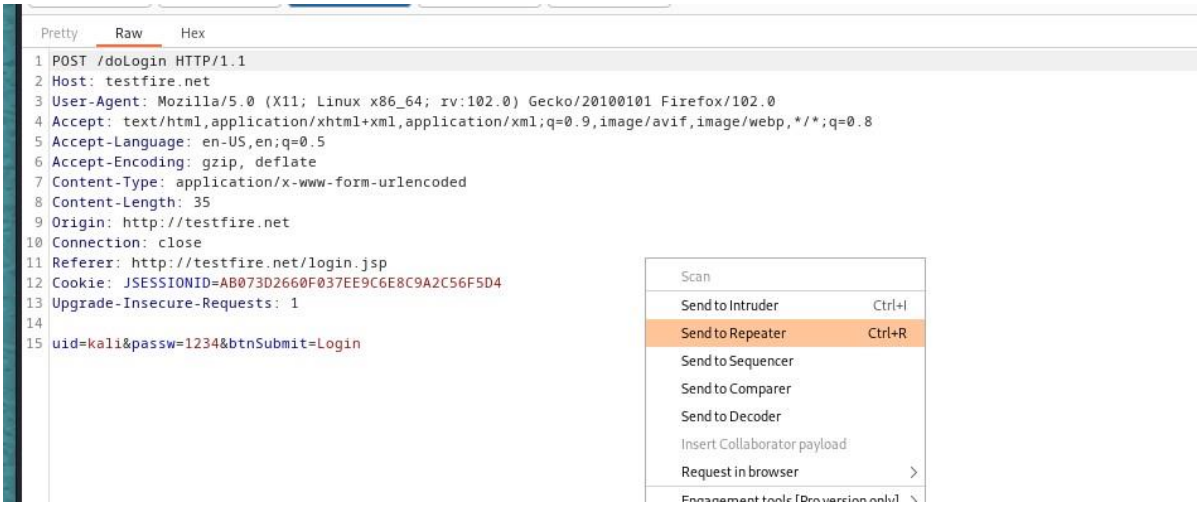
## *POC*

***Step1**: Open the browser and go to the http://testfire.netWebsite download foxyproxy and set proxy ip address = 127.0.0.1 and port =8080. Open burp suite and turn on the proxy and burp suite .Enter any credentials in the login page and press enter now you captured the Request in burpsuite*





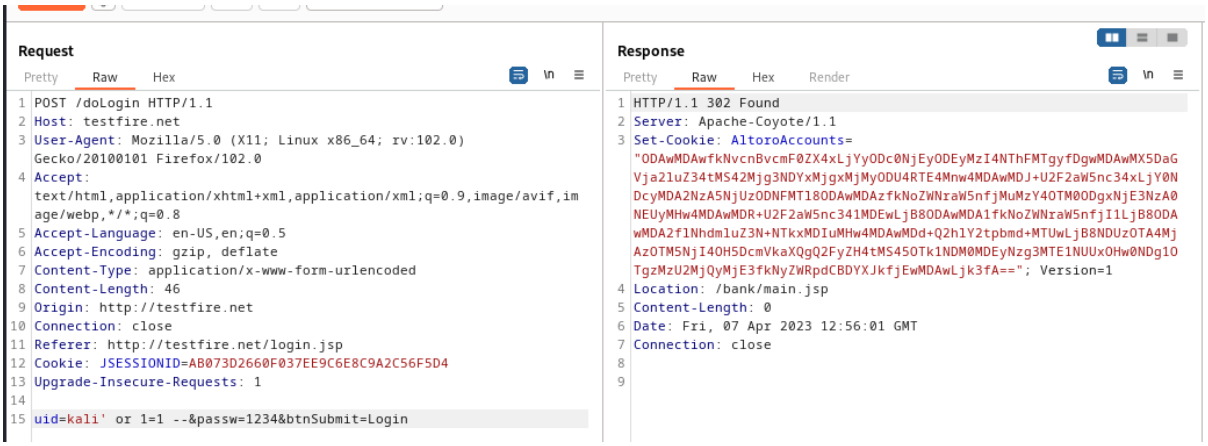

***Step2:*** *Send that Request to repeater for injecting the sql payload*



***Step3:*** *In UID after username enter this payload*

*(‘ or 1=1 --) and press send button  you get a cookie with the session token id and press forward redirection you will find the*

*Admin logged in page will be shown*




## *PREVENTION*

*Primary Defenses:*

*Option 1: Use of Prepared Statements (with Parameterized Queries)*

*Option 2: Use of Properly Constructed Stored Procedures*

*Option 3: Allow-list Input Validation*

*Option 4: Escaping All User Supplied Input*

*Additional Defenses:*

*Also: Enforcing Least Privilege*

*Also: Performing Allow-list Input Validation as a Secondary* Defense

## *REFERENCE:*

*https://www.w3schools.com/sql/sql_injection.asp*

*https://portswigger.net/web-security/sql-injection*

# *BROKEN AUTHENTICATION:*

*In simple words, Broken Authentication and Session Management attacks are anonymous attacks with the intention to try and retrieve passwords, user account information, IDs and other details.*

*Read more at: https://www.appknox.com/blog/understanding-the-owasp-top-10-broken-authentication-session-management*

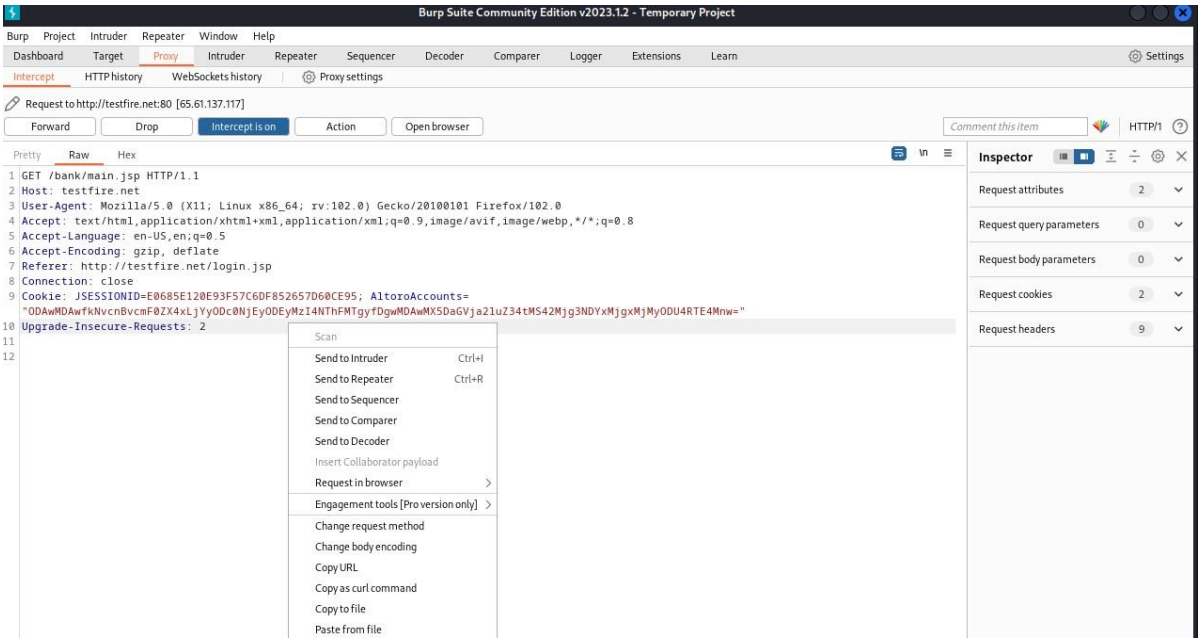
# *POC*

***Step1**: Login with that admin credentials in that website and*

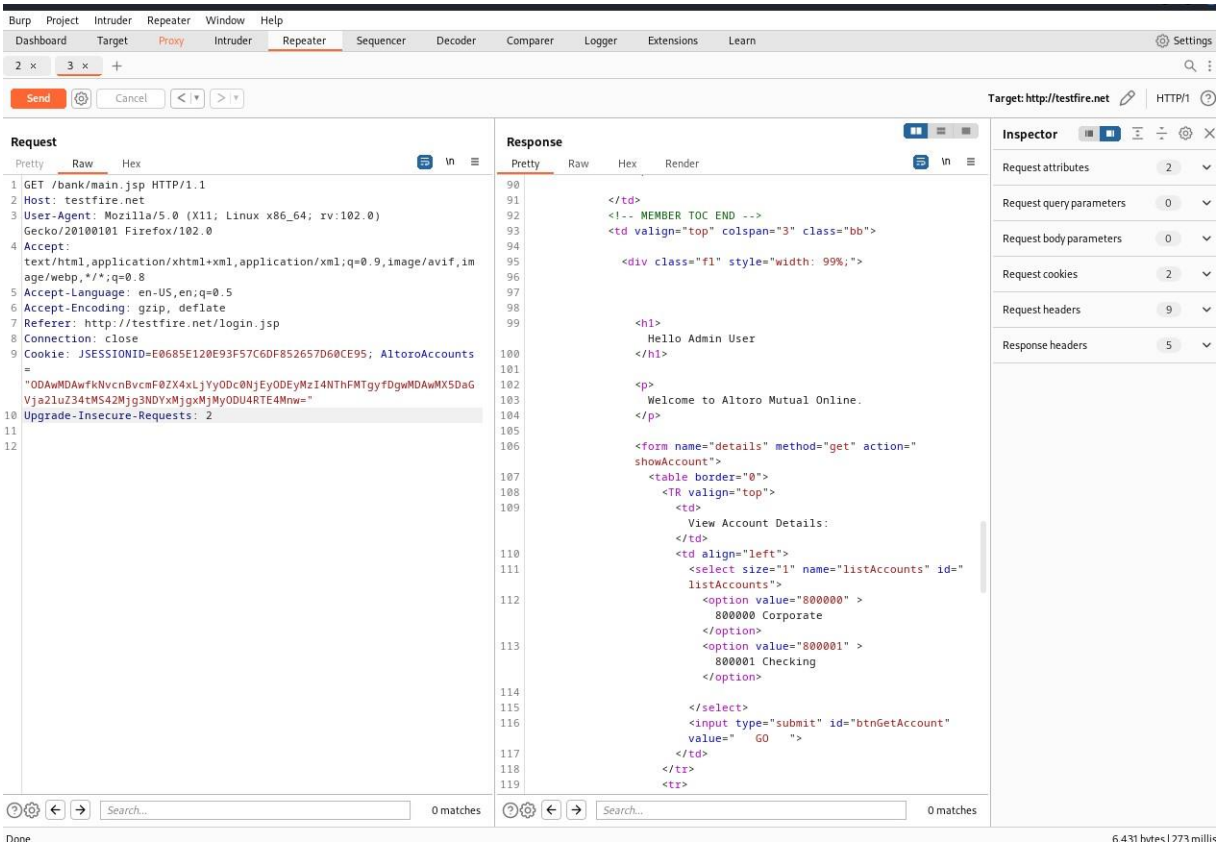
*Turn on the proxy and interception refresh the page you will*

*Get session id of the logged in page . logout that page after the*

*Request is recived*



***Step2**: Send that request to the repeater and press send button and forward redirection you will login to that webpage without Credentials it will get the admin logged in page*




# *PREVENTION*

- *Here are some of OWASP's technical recommendations to make sure your application is safe from these broken authentication vulnerabilities:*

- *Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login.*

- *Session IDs should not be in the URL. IDs should also be securely stored and invalidated after logout, idle, and absolute timeouts.*

- *Wherever possible, implement multi-factor authentication to prevent automated, credential stuffing, brute force, and stolen credential re-use attacks.*

- *Do not ship or deploy with any default credentials, particularly for administrator users.*

- *Implement weak-password checks, such as testing new or changed passwords against a list of the top 10000 worst passwords.*

- *Align password length, complexity and rotation policies with NIST 800-63 B's guidelines in section 5.1.1 for Memorized Secrets or other modern, evidence-based password policies.*

- *Ensure registration, credential recovery, and API pathways are hardened against account enumeration attacks by using the same messages for all outcomes. e.g. Authentication failure responses should not indicate which part of the authentication data was incorrect. Instead of "Invalid username" or "Invalid password", just use "Invalid username and/or password" for both. Error responses must be truly identical in both display and source code.*

- *Limit or increasingly delay failed login attempts. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected. The use of lists of known passwords is a common attack. If an application does not implement automated threat or credential stuffing protection, the application can be used as a password oracle to determine if the credentials are valid.*

## *REFERENCE*

*https://crashtest-security.com/broken-authentication-and-session-management/*

*https://www.sitelock.com/blog/owasp-top-10-broken-authentication-session-management/*

# *SENSITIVE DATA EXPOSURE*

*Data exposure vulnerability depends on how we handle certain information. If we store sensitive data in plain text documents, we make our application vulnerable to this attack.*

*Similarly, If we don't use SSL and don't have HTTPS security on web pages that store information, there is a risk of data being exposed.*

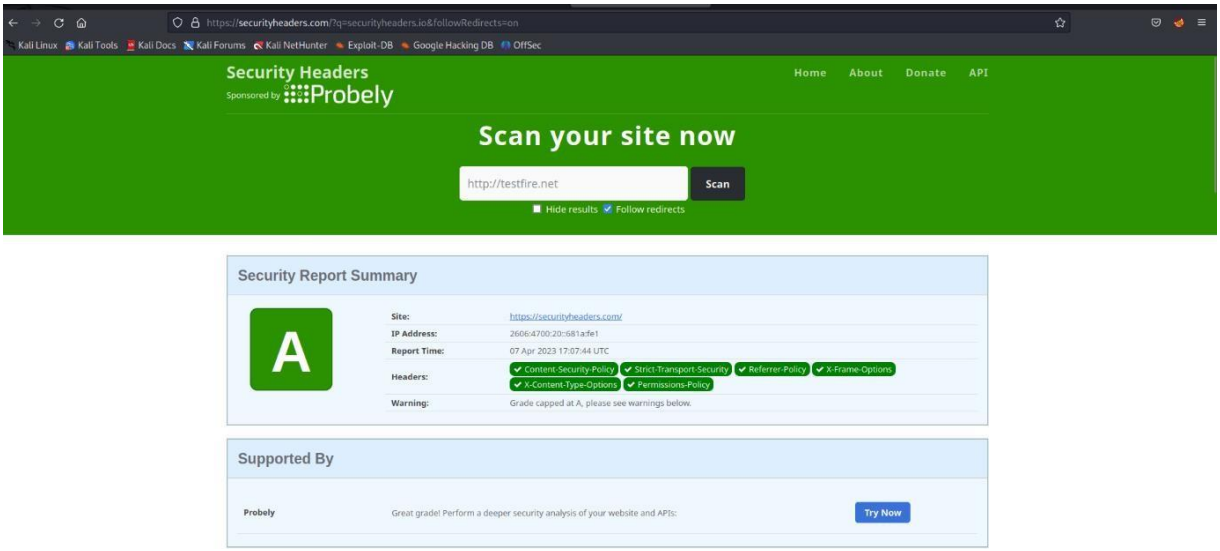
*Another thing to watch out for is storing data in a database that may be compromised by SQL injection. We have explained SQL injection and how to prevent that in our article OWASP Top 10 – Injection*

*If we are using weak cryptographic algorithms, keys, etc or not implementing hashed and salted passwords, our application becomes vulnerable to this type of attack.*

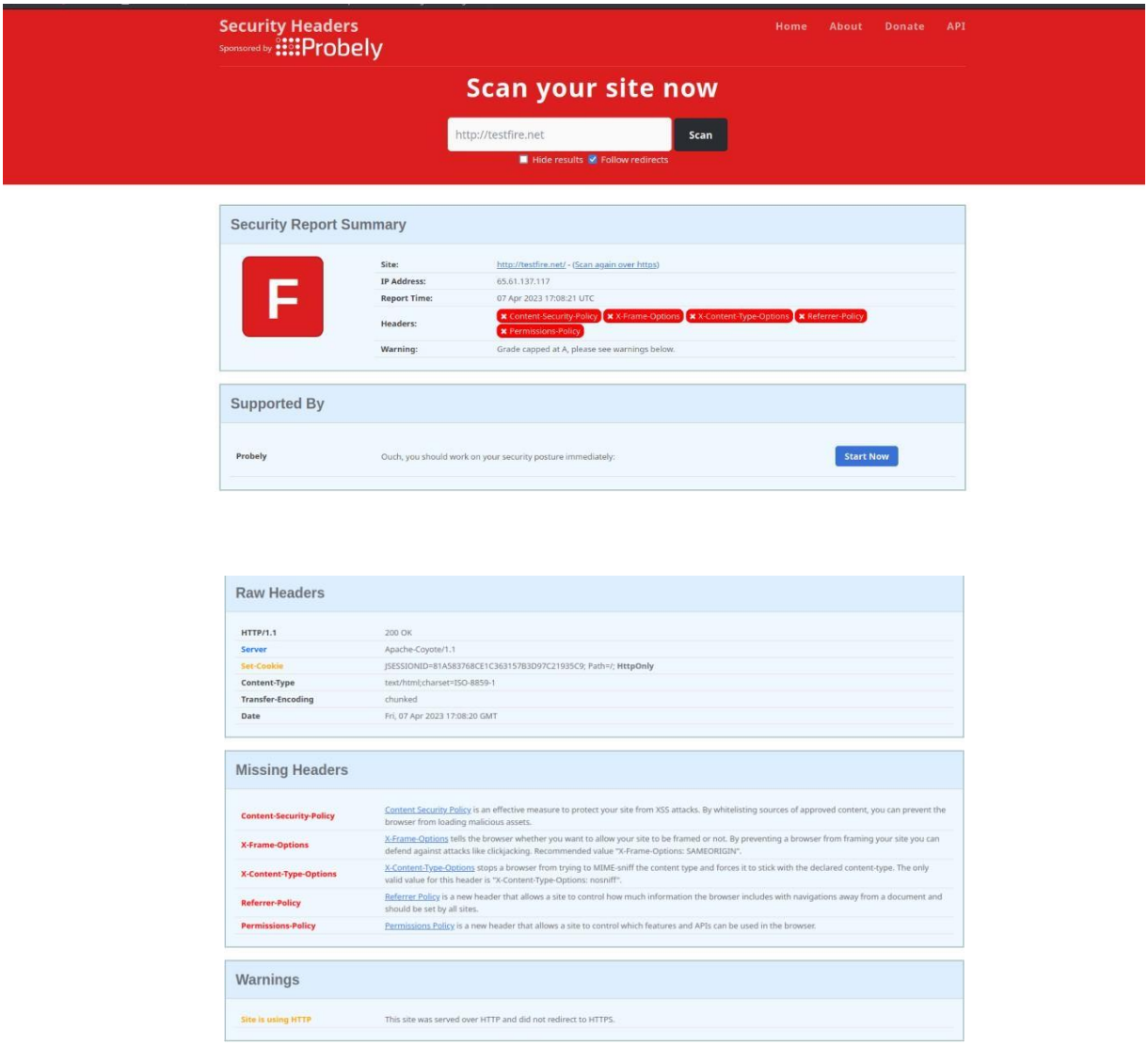
# *POC*

***Step1**: To check the security certificate go to the [www.securityheader.com](http://www.securityheader.com)*

*And enter the target url and enter scan*




***Step 2**: Now you will see the security report summary is F because this website does not have security header certificate like TSL & SSL*


Security Headers
Sponsored by Probely
Home
About
Donate
API
Scan your site now
http://testfire.net
Scan
Hide results
Follow redirects
Security Report Summary
F
Site:
http://testfire.net/ - (Scan again over https)
IP Address:
65.61.137.117
Report Time:
07 Apr 2023 17:08:21 UTC
Headers:
Content-Security-Policy
X-Frame-Options
X-Content-Type-Options
Referrer-Policy
Permissions-Policy
Warning:
Grade capped at A, please see warnings below.
Supported By
Probely
Ouch, you should work on your security posture immediately:
Start Now
Raw Headers
HTTP/1.1
200 OK
Server
Apache-Coyote/1.1
Set-Cookie
JSESSIONID=81A583768CE1C363157B3D97C21935C9; Path=/; HttpOnly
Content-Type
text/html;charset=ISO-8859-1
Transfer-Encoding
chunked
Date
Fri, 07 Apr 2023 17:08:20 GMT
Missing Headers
Content-Security-Policy
Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets.
X-Frame-Options
X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking. Recommended value "X-Frame-Options: SAMEORIGIN".
X-Content-Type-Options
X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
Referrer-Policy
Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.
Permissions-Policy
Permissions Policy is a new header that allows a site to control which features and APIs can be used in the browser.
Warnings
Site is using HTTP
This site was served over HTTP and did not redirect to HTTPS.

## *PREVENTION*

*first step is to figure out what data can be considered sensitive and therefore important to protect.*

- *When that is done, go over each of these data points and make sure that:*
- *The data is never stored in clear text.*
- *The data is never transmitted in clear text. Example between database and server, or over the internet.*
- *The algorithms used to encrypt the data are considered strong enough.*
- *The generation of the keys is secure.*
- *Browser headers are set to not cache when the sensitive data is presented to end-user.*
- *There are more things to look for when securing data, but what matters most is understanding what data is considered sensitive, and make sure it is treated as such in every instance.*

## *REFERENCE*

*https://blog.detectify.com/2016/07/01/owasp-top-10-sensitive-data-exposure-6/*

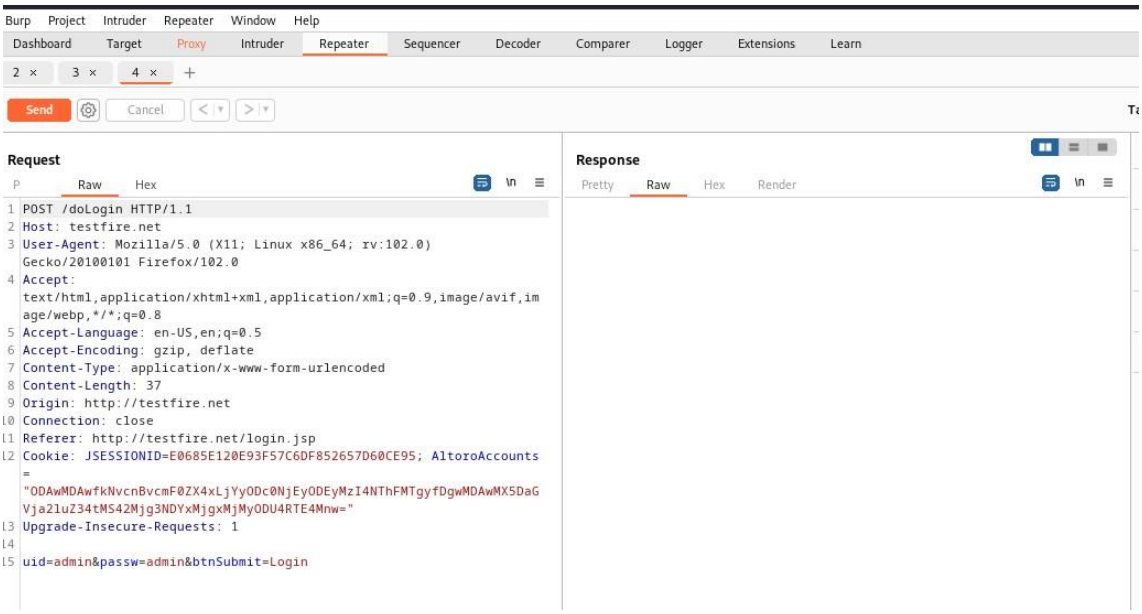
*https://owasp.org/www-project-top-ten*

# *SECURITY MISCONFIGURATION*

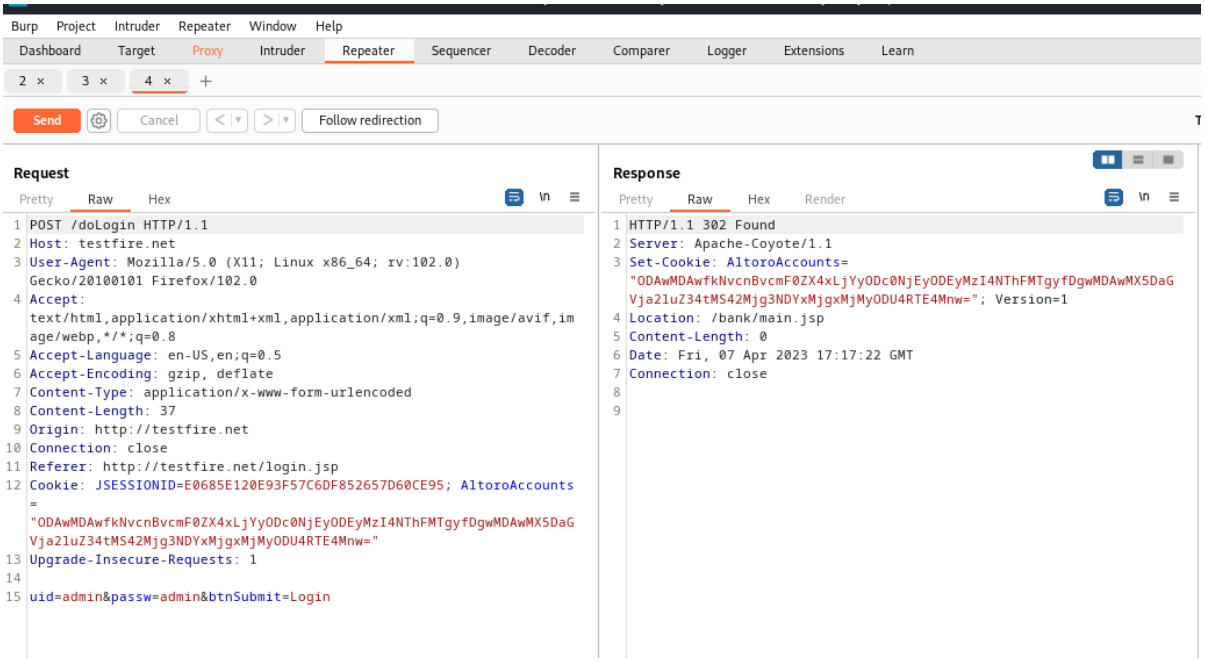
- *Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.*
- *Unnecessary features are enabled or installed (e.g., unnecessary ports, services, pages, accounts, or privileges).*
- *Default accounts and their passwords are still enabled and unchanged.*
- *Error handling reveals stack traces or other overly informative error messages to users.*
- *For upgraded systems, the latest security features are disabled or not configured securely.*
- *The security settings in the application servers, application frameworks (e.g., Struts, Spring, ASP.NET), libraries, databases, etc., are not set to secure values.*
- *The server does not send security headers or directives, or they are not set to secure values.*
- *The software is out of date or vulnerable (see A06:2021-Vulnerable and Outdated Components).*

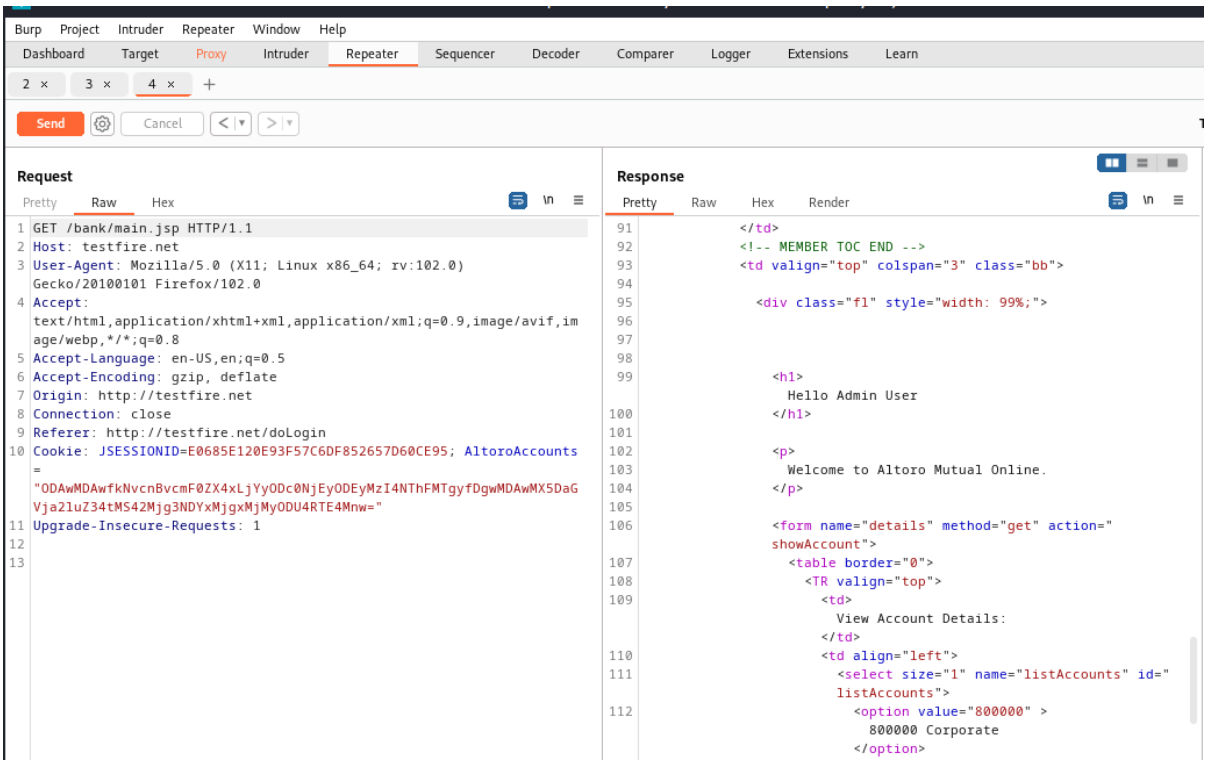
# *POC*

***Step1**: Go to the target website and capture the login request using burp suite*



***Step2**: change the UID=admin & passw=admin send that request and forward redirect that request now you can log into that web page easily because of default admin ,admin was not changed*



# *PREVENTION*

*Secure installation processes should be implemented, including:*

- *A repeatable hardening process makes it fast and easy to deploy another environment that is appropriately locked down. Development, QA, and production environments should all be configured identically, with different credentials used in each environment. This process should be automated to minimize the effort required to set up a new secure environment.*
- *A minimal platform without any unnecessary features, components, documentation, and samples. Remove or do not install unused features and frameworks.*
- *A task to review and update the configurations appropriate to all security notes, updates, and patches as part of the patch management process (see A06:2021-Vulnerable and Outdated Components). Review cloud storage permissions (e.g., S3 bucket permissions).*
- *A segmented application architecture provides effective and secure separation between components or tenants, with segmentation, containerization, or cloud security groups (ACLs).*
- *Sending security directives to clients, e.g., Security Headers.*
- *An automated process to verify the effectiveness of the configurations and settings in all environments.*

*REFERENCE*

*https://owasp.org/Top10/A05_2021-Security_Misconfiguration*

*https://www.thesslstore.com/blog/how-to-prevent-security-misconfiguration*

# *CROSS SITE SCRIPTING*

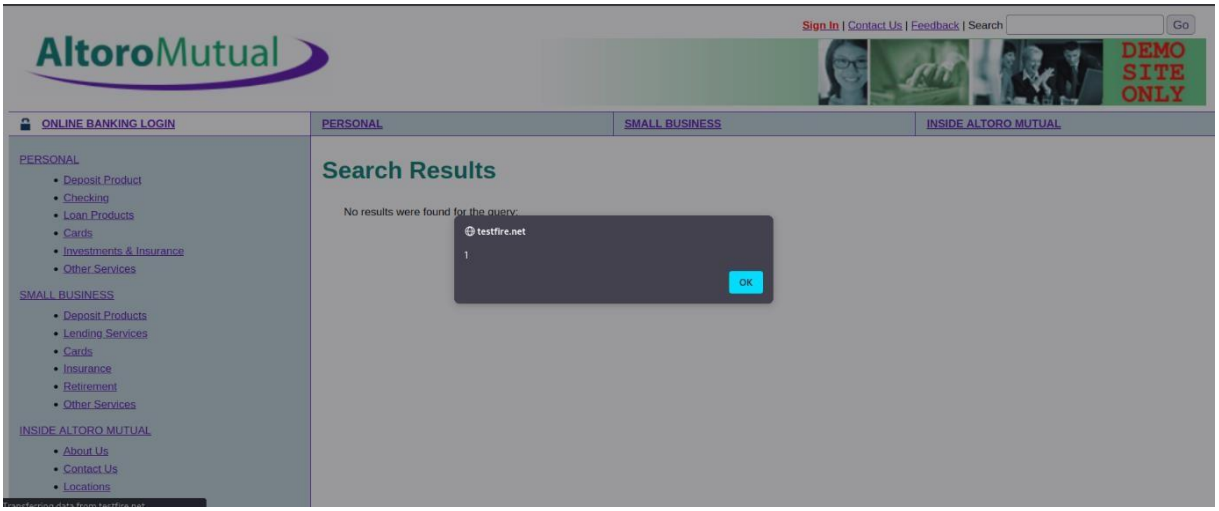
*Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.*

*An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. For more details on the different types of XSS flaws*

# *POC*

***Step1**: In the top right side of the website the will be a search bar enter the reflected xss payload (<script>alert(1)</script>)*


***Step 2*** *: Press enter there will be pop that shows alert of 1*




# *PREVENTION*

*Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data.*

*In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:*

- ***Filter input on arrival.*** *At the point where user input is received, filter as strictly as possible based on what is expected or valid input.*
- ***Encode data on output.*** *At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.*
- ***Use appropriate response headers.*** *To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can usethe Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.*
- ***Content Security Policy.*** *As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.*

# *REFERENCE*

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

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