Ben Gamliel

Pentrastion test report AltoroJ

18/05/22

Table of content

FINDING OVERVIEW

The web application runs in a safe environment on a local VM inside a container, while conducting the penetration test, there were several critical vulnerabilities discovered with the AltoroJ web application, I was able to expose numerous vulnerabilities that could be leveraged into different attack vectors such as XSS, SQL Injection, DDoS, ReDDos, and many more

This was possible since the web application was written with consideration of app functionality but not app security most of the app was written in Java and Javascript.

SEVERITY SCALE

CRITICAL Severity Issue: Poses an immediate danger to systems, networks, and/or data security and should be addressed as soon as possible. Exploitation requires little to no special knowledge of the target. Exploitation doesn't require highly advanced skills, training, or tools.

HIGH Severity Issue: Poses a significant danger to systems, networks, and/or data security. Exploitation commonly requires some advanced knowledge, training, skill, and/or tools. Issue(s) should be addressed promptly.

MEDIUM Severity Issue: Vulnerabilities should be addressed promptly. Exploitation is usually more difficult to achieve and requires special knowledge or access. Exploitation may also need social engineering as well as special conditions.

LOW Severity Issue: The danger of exploitation is unlikely as vulnerabilities offer little to no opportunity to compromise the system, network, and/or data security. It can be handled as time permits.

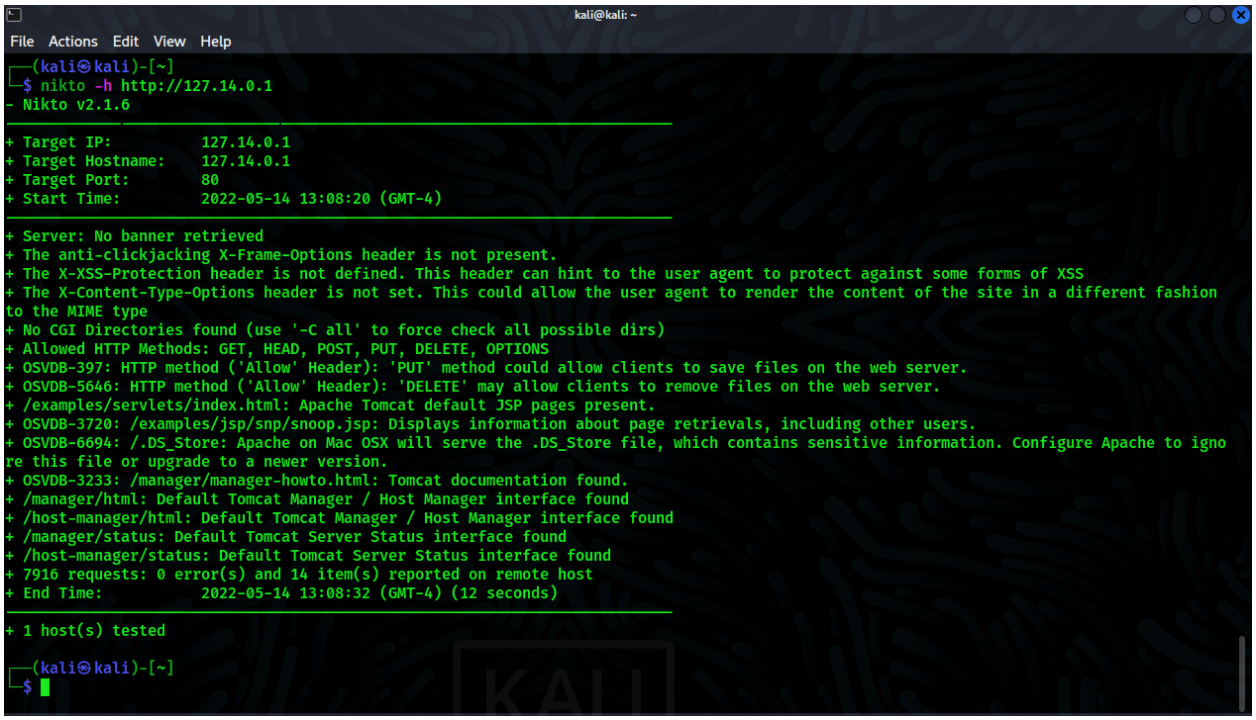
INFORMATIONAL Issue: Meant to increase the client's knowledge. Likely no actual threat.

METHODOLOGY

I used testing methods that are widely adopted and explained in the cyber-security assessment industry, which includes 5 phases: **Information Gathering, Enumeration, Vulnerability Assessment, Exploitation,** and **Reporting/Mitigation.**

INFORMATION GATHERING

I was given the web application git repo and a public server, after setting up the running environment I started communicating with the predefined IP address with burp suite for my manual fuzzing, I also used some automated fuzzing tools installed on kali including Nmap, Nikto, wapiti, free web scanners such as observatory and the SAST tool nodejsscan for the javascript code on the server



[Wapiti scan report]

Raw Headers

HTTP/1.1	200 OK
Server	Apache-Coyote/1.1
Set-Cookie	JSESSIONID=88CAAD54C8DD1AEE614363CEC022FCC9; Path=/; **HttpOnly**
Content-Type	text/html;charset=ISO-8859-1
Transfer-Encoding	chunked
Date	Thu, 12 May 2022 20:32:32 GMT
Connection	close

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.

Upcoming Headers

Cross-Origin-Embedder-Policy	Cross-Origin Embedder Policy allows a site to prevent assets being loaded that do not grant permission to load them via CORS or CORP.
Cross-Origin-Opener-Policy	Cross-Origin Opener Policy allows a site to opt-in to Cross-Origin Isolation in the browser.
Cross-Origin-Resource-Policy	Cross-Origin Resource Policy allows a resource owner to specify who can load the resource.

Additional Information

Server	This Server header seems to advertise the software being run on the server but you can remove or change this value.
Set-Cookie	This is not a SameSite Cookie.

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✗	-25	Content Security Policy (CSP) header not implemented	ⓘ
Cookies	✓	0	All cookies use the `Secure` flag and all session cookies use the `HttpOnly` flag	ⓘ
Cross-origin Resource Sharing	✓	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	ⓘ
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header cannot be set, as site contains an invalid certificate chain (optional)	ⓘ
HTTP Strict Transport Security	✗	-20	HTTP Strict Transport Security (HSTS) header cannot be set, as site contains an invalid certificate chain	ⓘ
Redirection	✗	-20	Does not redirect to an HTTPS site	ⓘ
Referrer Policy	–	0	Referrer-Policy header not implemented (optional)	ⓘ
Subresource Integrity	–	0	Subresource Integrity (SRI) is not needed since site contains no script tags	ⓘ
X-Content-Type-Options	✗	-5	X-Content-Type-Options header not implemented	ⓘ
X-Frame-Options	✗	-20	X-Frame-Options (XFO) header not implemented	ⓘ
X-XSS-Protection	✗	-10	X-XSS-Protection header not implemented	ⓘ

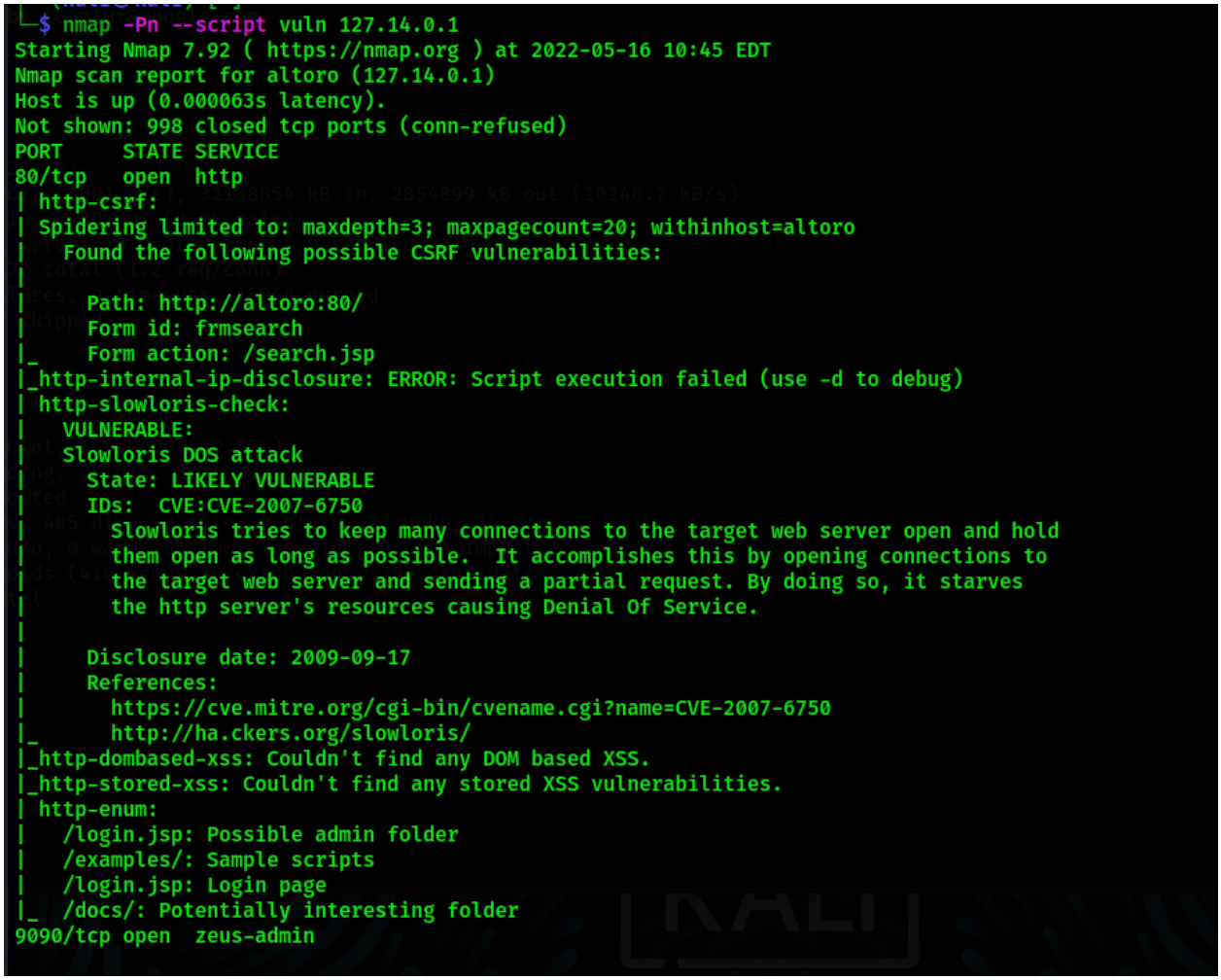
ENUMERATION

I performed service enumeration to discover information about the services provided by AltoroJ such as port scanning with Nmap to determine which services were open

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 1-65535 127.14.0.1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-16 10:35 EDT
Nmap scan report for altoro (127.14.0.1)
Host is up (0.000056s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT     STATE SERVICE
80/tcp   open  http
9090/tcp open  zeus-admin

Nmap done: 1 IP address (1 host up) scanned in 1.24 seconds

┌──(kali㉿kali)-[~]
└─$
```



VULNERABILITY ASSESSMENT

The vulnerability assessment was done manually and with the aid of Burpsuite and a few other automated tools such as Nmap

Vulnerability: SQL injection

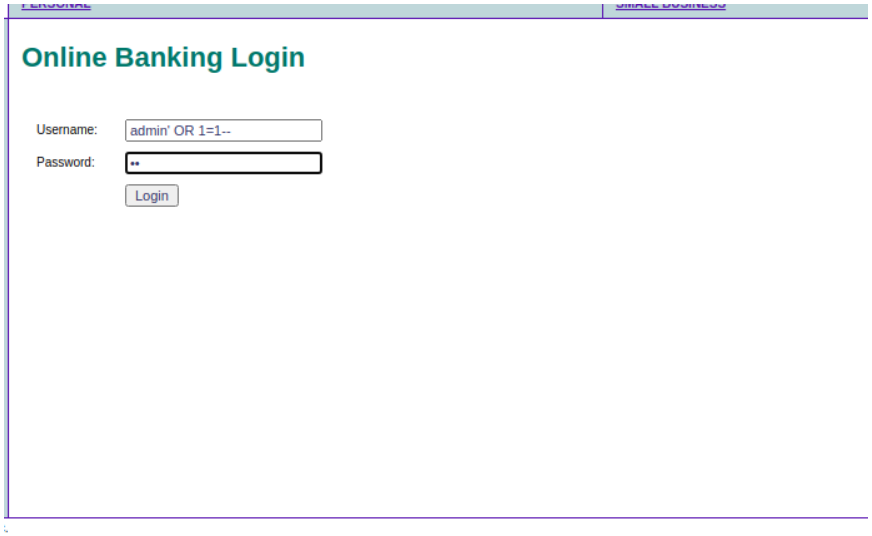
Vulnerability Explanation: using dynamic SQL statements controlled by the user without any validation and sanitation could allow an attacker to modify the statement's meaning or execute arbitrary SQL commands

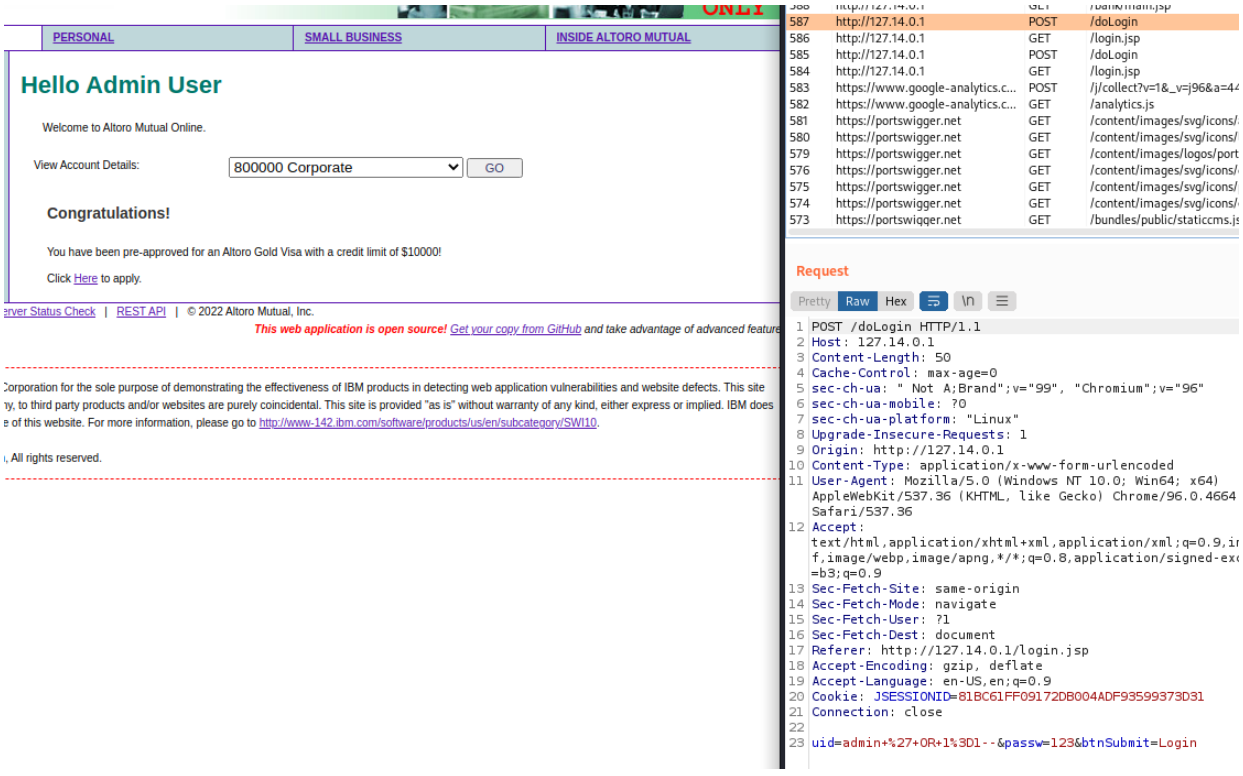
Mitigation: Implement Prepared Statements with Parameterized Queries, Implement User Input Whitelisting. Injection attacks remain the most common attacks leveraged against web applications. One of the most effective mitigation strategies for preventing SQL

Injection attacks is the implementation of Prepared Statements with Parameterized Queries.

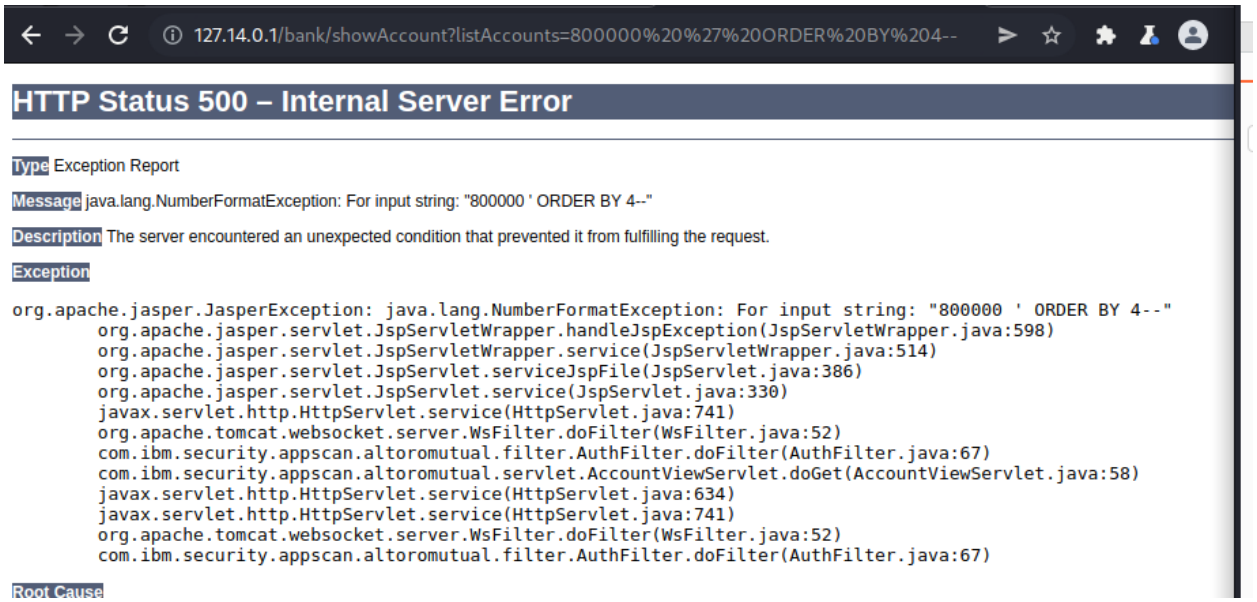
Severity: High CEW https://cwe.mitre.org/data/definitions/89.html

Usage Authentication Bypass





Another vector was in listAccounts but eventually, I did not complete the drill-down



Vulnerability: Cross-site scripting (XSS)

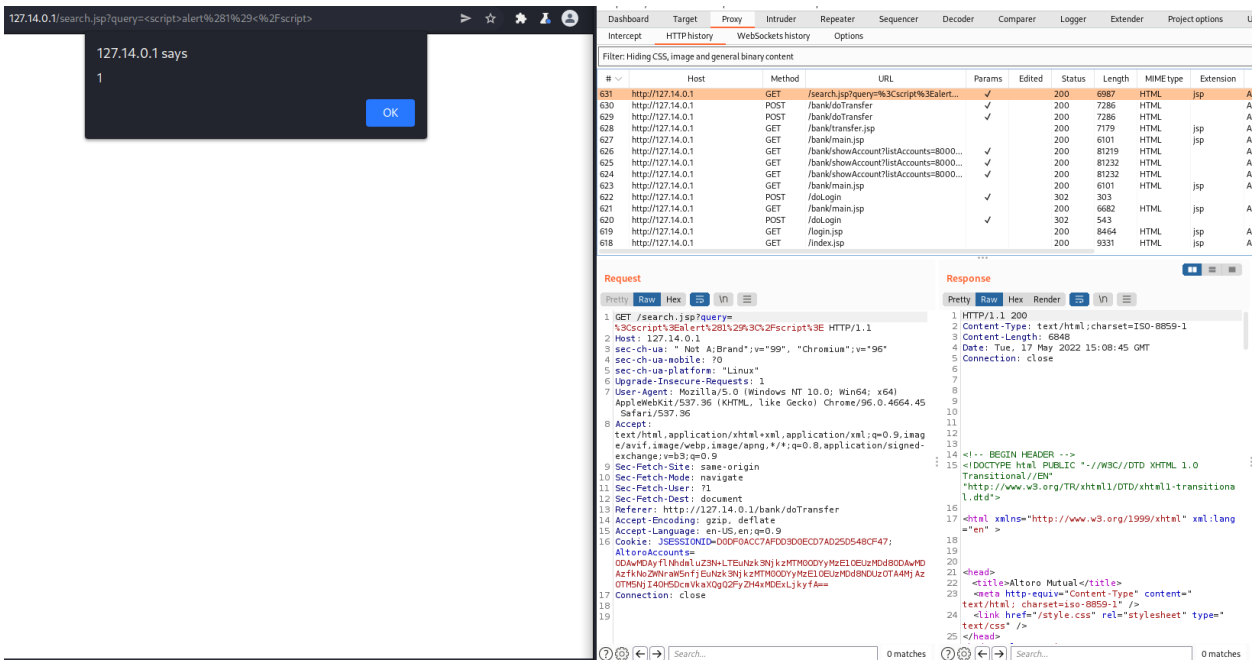
Vulnerability Explanation: XSS is a security vulnerability (that executes on the client-side)that allows an attacker to compromise the interactions of the web application by allowing the attacker to circumvent the same-origin policy (sop) that is designed to segregate different websites from each other (user data\cookies\information etc..) the malicious (reflected) URL or persistent XSS(stored on the server) will be executed in the context of the victim's web browser doing the action the attacker intended for example carry out any action that the user is able to perform

Vulnerability Mitigation: there are a few things you could do to mitigate an XSS vulnerability, first you could filter the user input as strictly as possible (white list) based on what is expected or valid input, it's possible to encode data on output, this will prevent the output data to be interpreted as active content, also it's required to use appropriate response headers using the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend, last you could use Content Security Policy(CSP) to reduce the severity of any XSS vulnerabilities that still occur

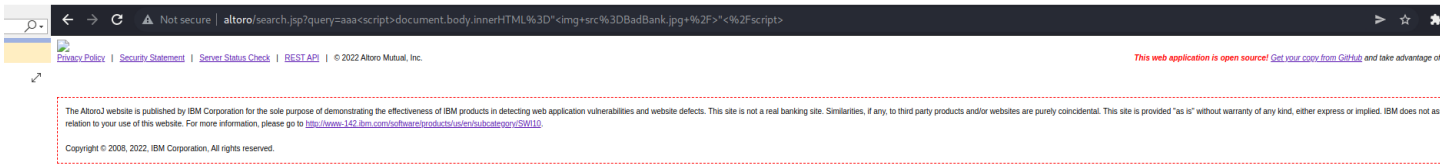
Severity: High depending on the type and business logic

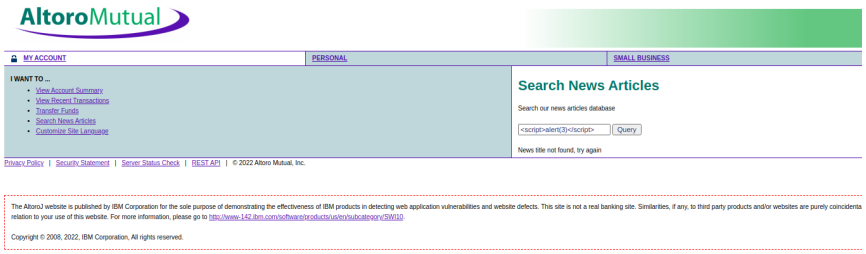
CWE: https://cwe.mitre.org/data/definitions/79.html

Usage: there were a few entry points that were valid for an XSS attack, reflected and Persistent, an attacker can craft a script of his choice and then send it to the victim, all the victim needs to do is click on the link, an attacker could encode the URL in a way that the link will look suspicious, also a skilled attacker could leverage this so a user will log in to his account and then exploit him with a CSRF attack



Another example



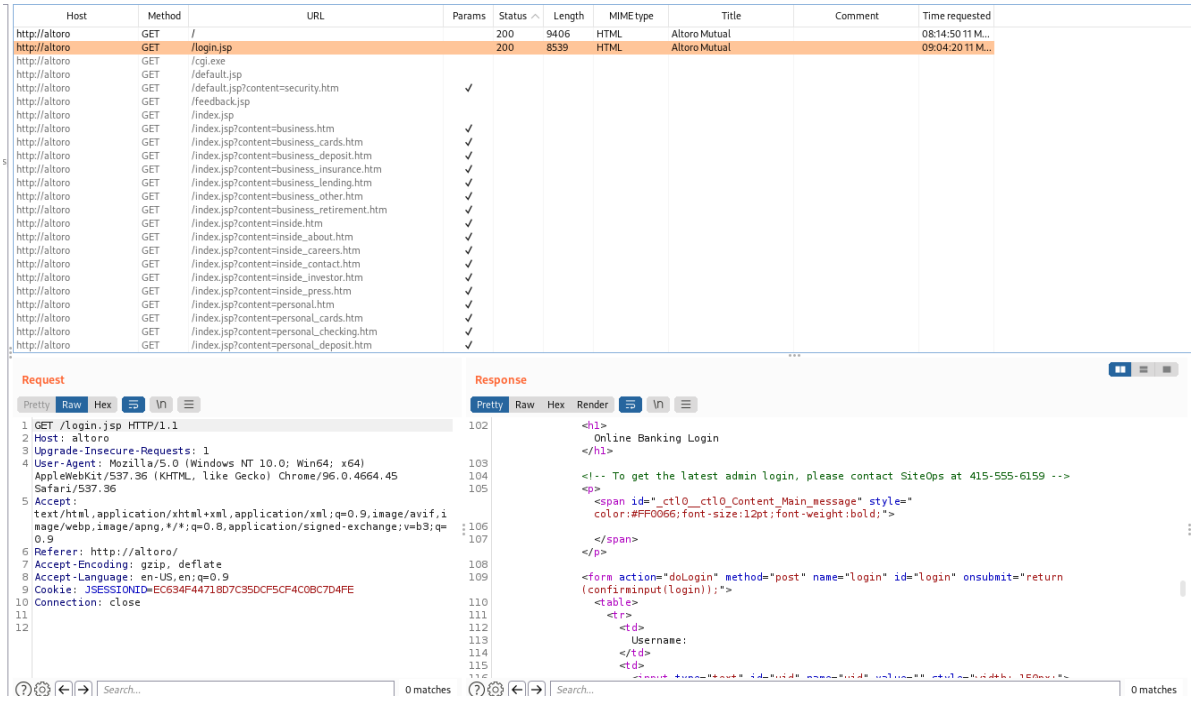


Vulnerability: Social Engineering

Vulnerability Explanation: in this case, an attacker could trick the telemarketer to believe that he is the system admin so he would provide the attacker with the latest admin login, maybe other critical information could be exposed in the same matter

Vulnerability Mitigation: employed social engineering awareness, secret information before information disclosure, 2 steps authentication (who you are, what you have, what you know)

Severity: low but in some cases, this could range from minimal damage to high(critical)



Vulnerability: Unencrypted login request

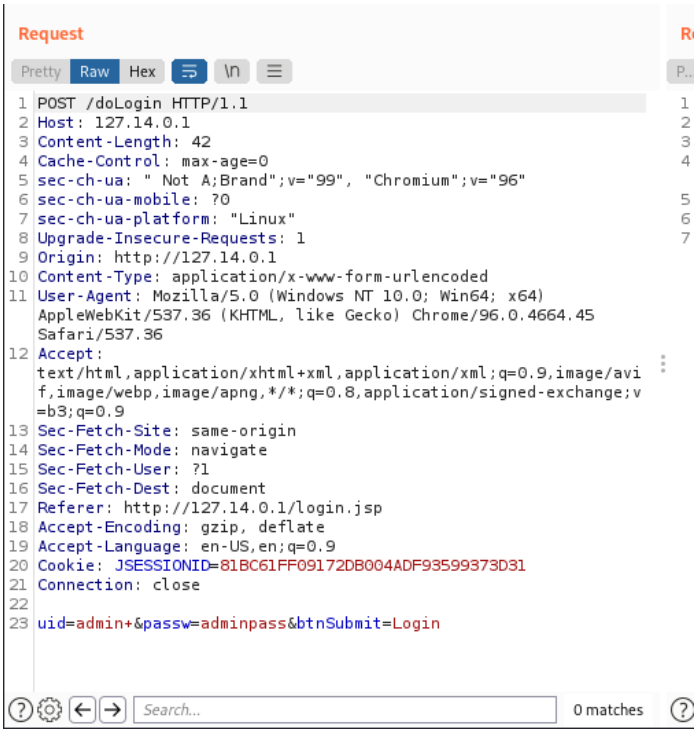
Vulnerability Explanation: unencrypted sensitive login information sent over the web could be exposed to a MIM attack

Vulnerability Mitigation: encrypt sensitive information before sending it over the web, and use a secure connection such as SSL when sending sensitive information

Severity: high

cwe: https://cwe.mitre.org/data/definitions/319.html

Usage:



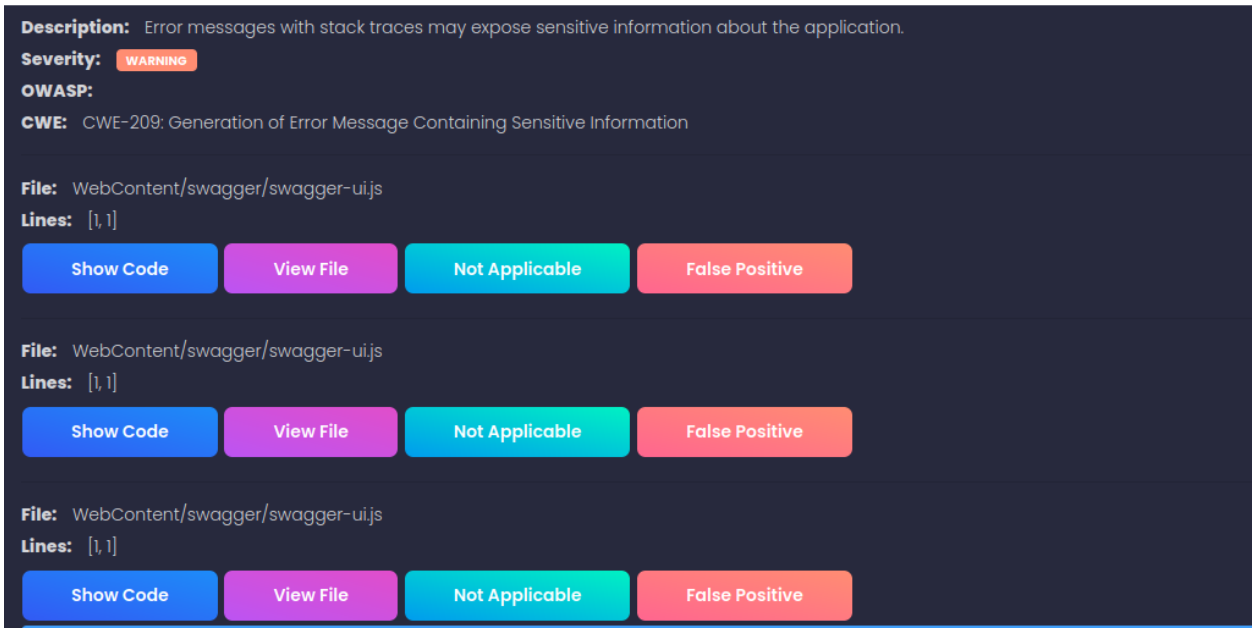
Vulnerability: Error messages with stack traces may expose sensitive information about the application

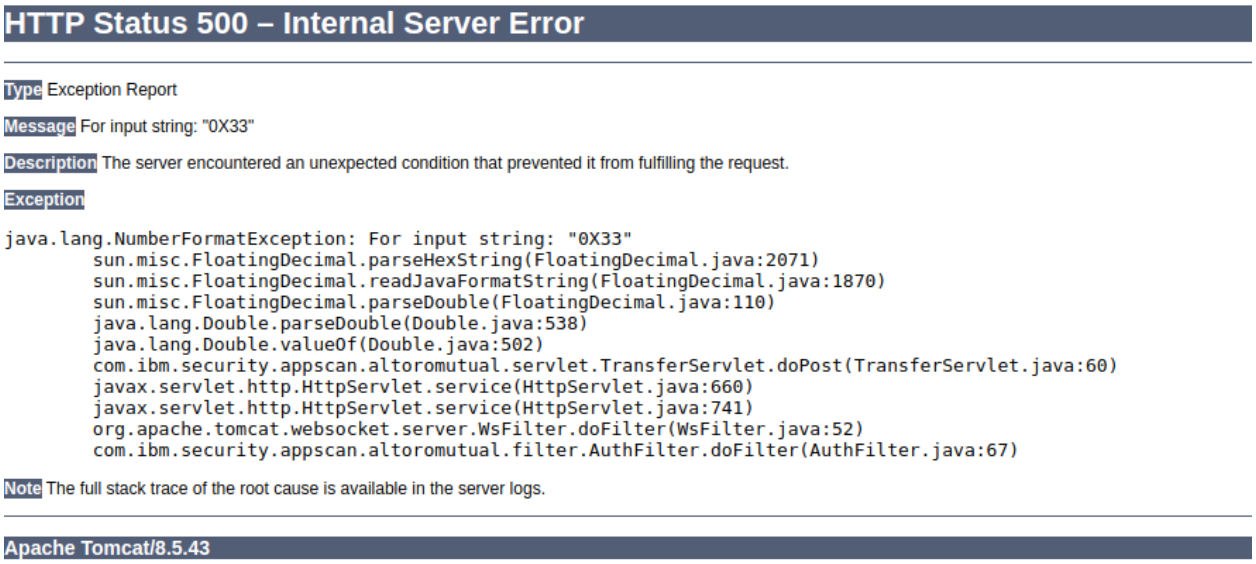
Explanation: information leakage may help an attacker with the reconnaissance stage meaning, An attacker may use the contents of error messages to help launch another, more focused attack for

Mitigation: ensure that an error message only the minimal details that are useful to the intended audience and no one else

Severity - High, Cwe: https://cwe.mitre.org/data/definitions/209.html

Attack usage: expose of internal information to leverage current knowledge on the application\server for example an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application





HTTP Status 500 – Internal Server Error

Type Exception Report

Message For input string: "0X33"

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

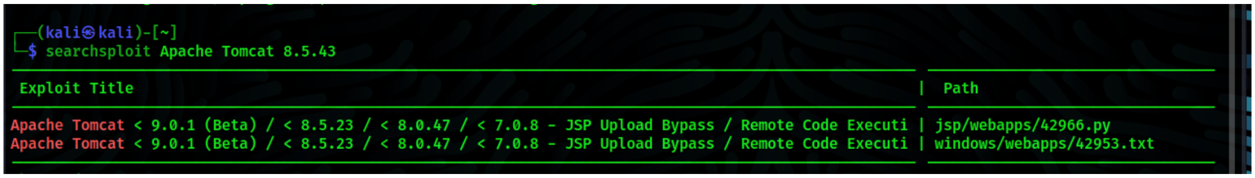
Exception

```
java.lang.NumberFormatException: For input string: "0X33"
        sun.misc.FloatingDecimal.parseHexString(FloatingDecimal.java:2071)
        sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1870)
        sun.misc.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        java.lang.Double.parseDouble(Double.java:538)
        java.lang.Double.valueOf(Double.java:502)
        com.ibm.security.appscan.altoromutual.servlet.TransferServlet.doPost(TransferServlet.java:60)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:660)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
        com.ibm.security.appscan.altoromutual.filter.AuthFilter.doFilter(AuthFilter.java:67)
```

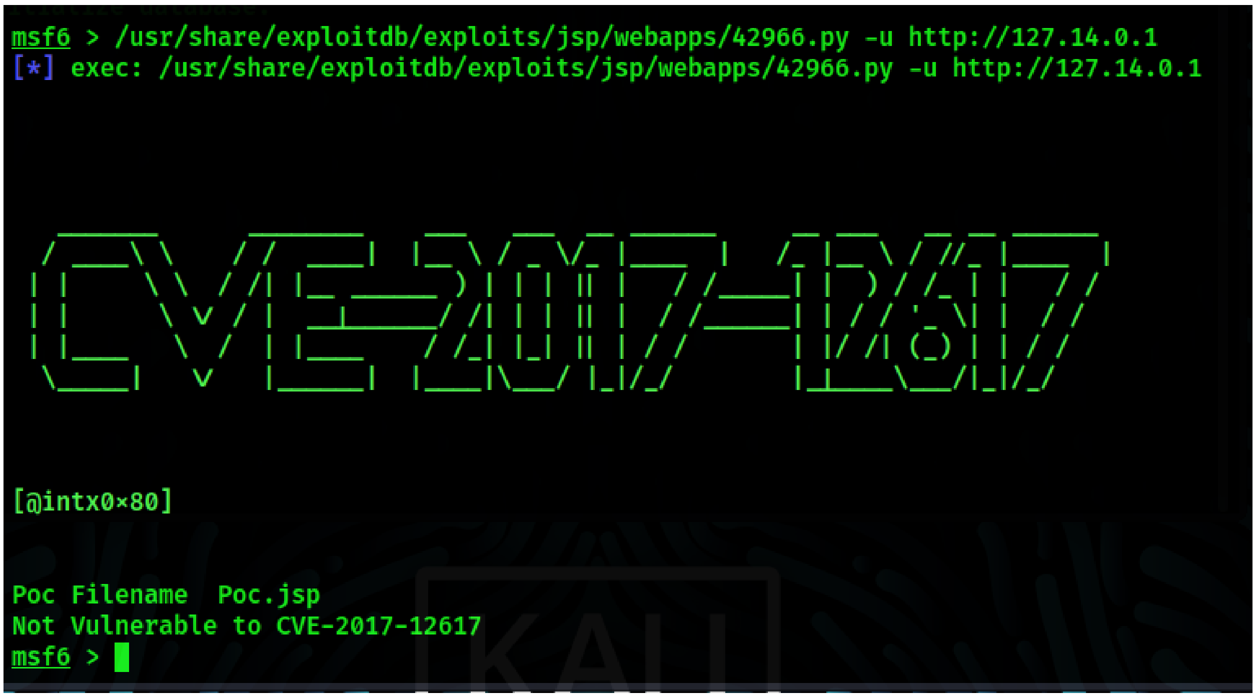
Note The full stack trace of the root cause is available in the server logs.

Apache Tomcat/8.5.43

With this information at hand, I was able to use searchsploit-DB for known CVE related to Apache Tomcat 8.5.43



Unfortunately, the application wasn't Vulnerable to the known CVE



Additional untested vulnerabilities and attack vectors

Vulnerability: unlimited user input size

Vulnerability Explanation: (untested)in some cases legit user input size wasn't limited this could lead to exhausting system resources resulting in DoS, and additionally, this could be leveraged to a Heap spraying attack, also if the variable responsible for holding the input size data, for example, is an int its possible to create an int overflow vulnerability and then leverage the attack to different paths

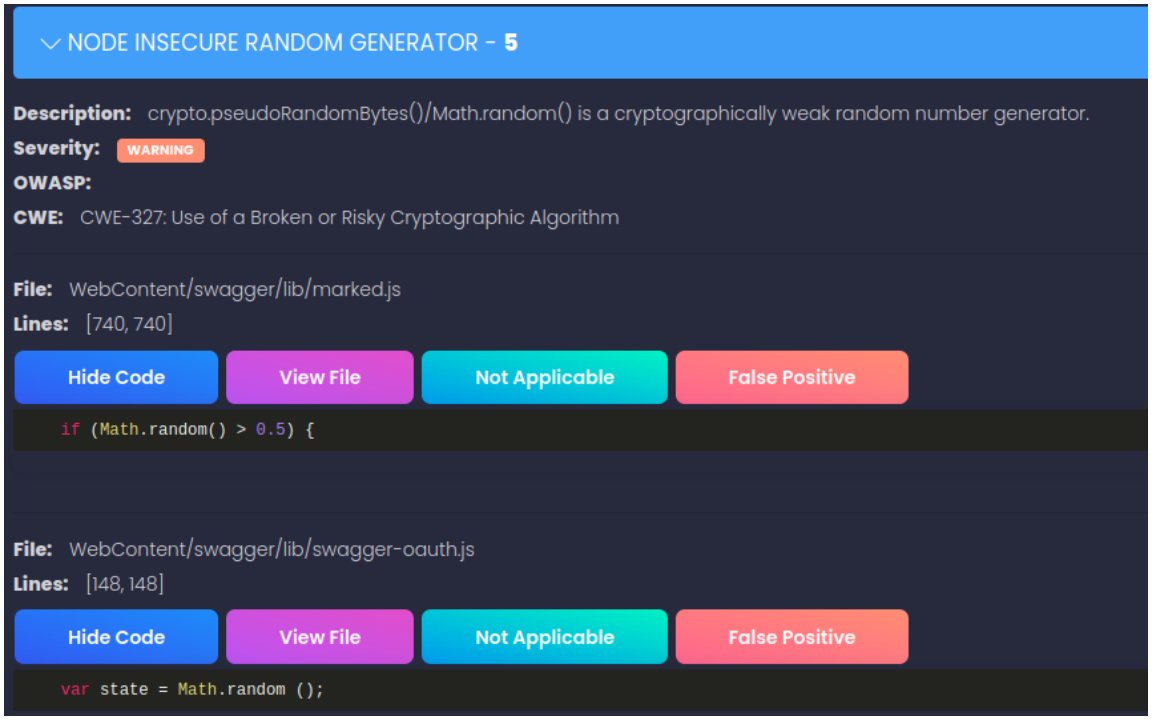
Vulnerability Mitigation: limit input size

Severity: warning

cwe:https://cwe.mitre.org/data/definitions/20.html

https://cwe.mitre.org/data/definitions/770.html

With almost similar properties the Regular expression for the user input in some cases combined with the fact that the input is unlimited could result in a denial of service



Mitigation: change to a stronger random algorithm

Attack usage: if an attacker knows the bound of the random function he could try to mimic its results

Many Headers with security properties were not defined\used in this application such as

- This application does not have API rate limiting controls.**CWE-770: Allocation of Resources Without Limits or Throttling - INFO**
- Description: This application does not have anti CSRF protection which prevents cross-site request forgery attacks. - **CWE-352: Cross-Site Request Forgery (CSRF) - INFO**
- Helmet X Permitted Cross-Domain Policies header is not configured for this application.
- Helmet XSS Protection header is not configured for this application
- Helmet IE No Open header is not configured for this application.
- Helmet Expect CT header is not configured for this application.
- Helmet DNS Prefetch header is not configured for this application
- Helmet Feature Policy header is not configured for this application.
- Helmet X Powered By header is not configured for this application.
- Helmet No Sniff header is not configured for this application.
- Helmet HSTS header is not configured for this application.
- Helmet X Frame Options header is not configured for this application.request
- Helmet Referrer-Policy header is not configured for this application.
- Helmet Content Security Policy header is not configured for this application
- This application does not have API rate limiting controls.
- **CWE-693: Protection Mechanism Failure**

  Severity: info

CONCLUSION

The web application contains many possible attack vectors, These issues should be addressed as soon as possible and I would recommend taking the website off the web until the production of a robust web application with additional security features even at the cost of damaging the customer's immediate usage