(A9) – Security Logging Failures

(2)– Log Spoofing Challenge

🎯 Objective:

Make it appear in the logs that the username `admin` successfully logged in, even though it didn't.

✅ Steps:

1. **Submit a normal account** and observe the message:

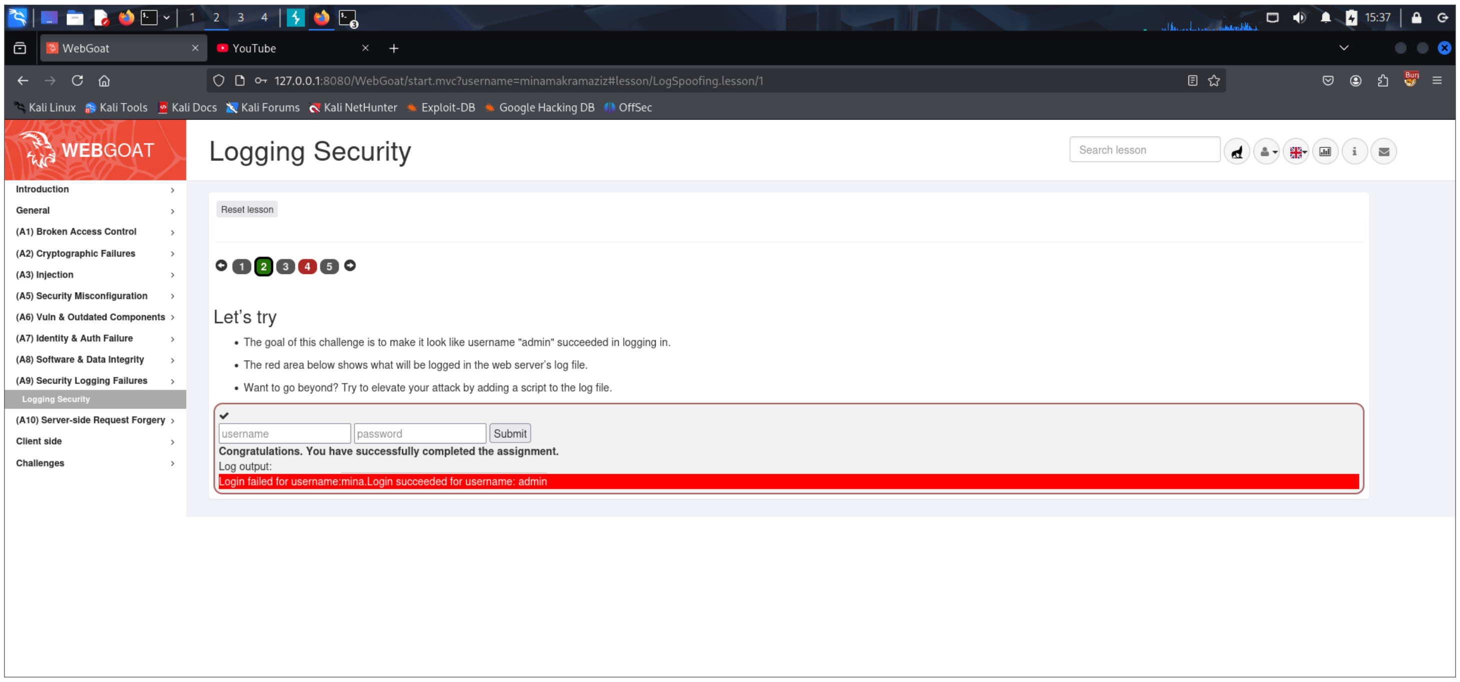
   ```
   Login failed for username: [username]
   ```

2. **Change the username** input field to:

   ```
   [username]. Login succeeded for username: admin.
   ```

   This input manipulates the logs to simulate a successful login for `admin`.

3. **Click Submit** to send the manipulated request.



127.0.0.1:8080/WebGoat/start.mvc?username=minamakramaziz#lesson/LogSpoofing.lesson/1

Logging Security

Let's try

- The goal of this challenge is to make it look like username "admin" succeeded in logging in.
- The red area below shows what will be logged in the web server's log file.
- Want to go beyond? Try to elevate your attack by adding a script to the log file.

Congratulations. You have successfully completed the assignment.

Log output:

```
Login failed for username:mina.Login succeeded for username: admin
```

---

(4)– Log Bleeding and Credential Leak

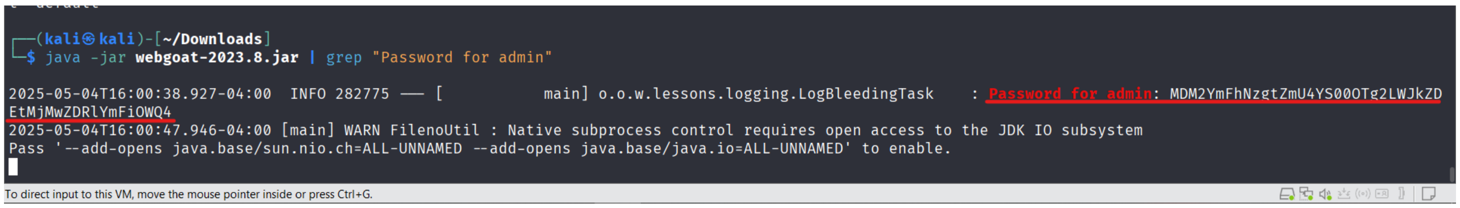
Objective:

Extract and decode a leaked admin password from WebGoat's application logs.

Steps:

1. **Open the Command Prompt** (or Terminal) where the WebGoat server is running.
2. **Look into the logs** and find a line like:

```
Password for admin: MDM2YmFhNzgtZmU4YS00OTg2LWJkZDEtMjMwZDRlYmFiOWQ4
```



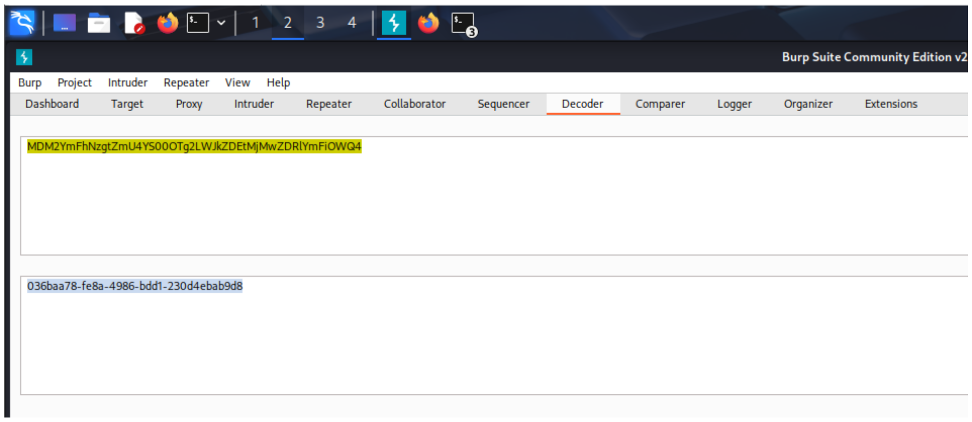
```
┌──(kali㉿kali)-[~/Downloads]
└─$ java -jar webgoat-2023.8.jar | grep "Password for admin"

2025-05-04T16:00:38.927-04:00  INFO 282775 --- [           main] o.o.w.lessons.logging.LogBleedingTask    : Password for admin: MDM2YmFhNzgtZmU4YS00OTg2LWJkZD
EtMjMwZDRlYmFiOWQ4
2025-05-04T16:00:47.946-04:00 [main] WARN FilenoUtil : Native subprocess control requires open access to the JDK IO subsystem
Pass '--add-opens java.base/sun.nio.ch=ALL-UNNAMED --add-opens java.base/java.io=ALL-UNNAMED' to enable.
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

This is an **encoded password**, usually a UUID or Base64 string.

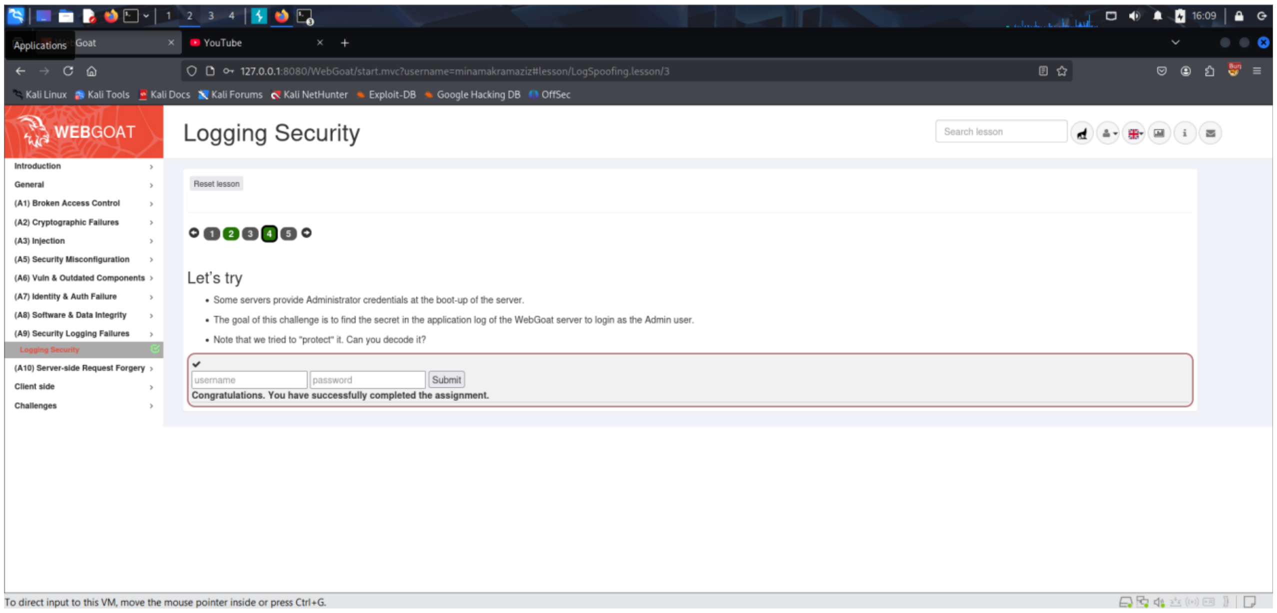
3. **Copy the encoded password**.
4. Open a tool like **Burp Suite** ogo to the **Decoder** section.
5. **Paste the encoded password** into the decoder and **select Base64 decode**


Burp Suite Community Edition v2
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions

```
MDM2YmFhNzgtZmU4YS00OTg2LWJkZDEtMjMwZDRlYmFiOWQ4
```

```
036baa78-fe8a-4986-bdd1-230d4ebab9d8
```

6. Use the decoded value as the **password**, and use the **username: Admin**


Applications
Goat
YouTube
127.0.0.1:8080/WebGoat/start.mvc?username=minamakramaziz#lesson/LogSpoofing.lesson/3
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
WEBGOAT
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
Logging Security
(A10) Server-side Request Forgery
Client side
Challenges
Reset lesson
username
password
Submit

Logging Security

Let's try

- Some servers provide Administrator credentials at the boot-up of the server.
- The goal of this challenge is to find the secret in the application log of the WebGoat server to login as the Admin user.
- Note that we tried to "protect" it. Can you decode it?

Congratulations. You have successfully completed the assignment.

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

(Ensure "Admin" is capitalized correctly – it's case-sensitive)

7. **Submit the login form**.

⚠️ Security Risks Highlighted:

1. **Log Bleeding:**
   - Sensitive credentials or secrets may be written to logs.
   - Attackers with access to logs can escalate privileges.
2. **Improper Encoding/Obfuscation:**
   - Using Base64 to "hide" secrets is not secure—it's easily reversible.

✅ Recommendations for Developers:

- Never log sensitive data (like passwords, secrets, tokens).
- Use secure vaults or environment variables for credential storage.
- Regularly audit logs for leaked secrets.
- Implement log access control to limit who can read them.
