CSCI 493-81
Lab 10
Security Lab
Topic: SQL Injection

Prof. Dietrich

April 30, 2025

Group size: 4 — Setup needed: Windows/macOS, Unix hosts, SPHERE.[1,2]

1 General description

In this lab, we are experimenting with SQL injection. This lab is based on a SPHERE exercise by Peter A. H. Peterson and Peter Reiher.

2 File with samples and results

The files are in lab-sqli.zip. The files there provide more background on SQL injection attacks, as well as patching.

3 Setting up for the experiment

1. Login to SPHERE.
2. Create an instance using the lab name `sqli`.
3. After the experiment has finished swapping in, log in to the `sqli` node via ssh.

Make sure that you save your work as you go. See the instructions in the submission section of this exercise for information about save and restore scripts. Save any changes you make to the source code, your patches, memos, etc. in your home directory.

You will probably want to set up port forwarding for tunnelling HTTP over ssh so you can test the web applications with a browser on your own desktop.

3.1 Requirement

- You use the instructions provided with this lab. You should use SPHERE, as this experiment can do damage.
- Your output should demonstrate that you have done the experiments, such as screenshots, Unix typescripts, and the traffic logs showing network packets.

4 Tasks

The tasks are related to performing a SQL injection.

4.1 Database security – dealing with SQL

FrobozzCo has its own internal company credit union, FrobozzCo Community Credit Union (FCCU). FCCU has an Internet-accessible web-based application that allows employees to access their paychecks and pay bills via a money wiring system. There are very few bank employees, and they use a a special administrative interface that runs on a different system that is not network accessible. In true FrobozzCo fashion, the public banking software was written in house by the CTO's nephew (who is a nice kid but not the brightest candle on the cake).

As it turns out, a lot of money has been disappearing from the credit union while you've been gone. It looks like someone has figured out how to force other accounts to wire money... to an anonymous bank account in the Cayman Islands! Worse yet, several employees have had serious identity theft problems of late – clearly someone has access to personal information and you have a hunch it's all coming from this server. To top it all off, the company itself is showing a deficit of $32,767 and it looks like it was somehow drawn through FCCU.

In a surprising display of foresight, your predecessor installed a network monitor watching the FCCU server. However, you are shocked to find out (from the network monitor and server logs) that nobody has logged into the server directly – in fact, the only interaction that anyone has had with the server has come through the Internet facing web interface. It looks like insecure software is to blame, again.

You assume that there must be one or more vulnerabilities in the code that interfaces with the SQL database – in the FCCU software, the directory, or both – and that somehow a malicious user is able to make the system do something it's not supposed to, like write checks. Worse yet, it seems like the attacker has managed to view private information like social security numbers, birthdates, and so on. You have heard about a class of attacks called "SQL Injection," and it seems likely that this is the kind of exploit being used.

Surprisingly, your boss agrees with you and instructs you to produce a one-page memo, a detailed transcript demonstrating the exploit, and a patch for the software. Additionally, he also wants to know how to clean up this mess – how severe is the compromise? How can we restore the system to a safe state?

4.2 SQL Injection

1. Load your `sqli` image in SPHERE. (You don't need to reload it if it is already active.)
    - The source code is located at `/usr/lib/cgi-bin/FCCU.php`

- If you have set up ssh tunnelling via port 8118 (a good idea), the memo application can be accessed at `http://localhost:8118/cgi-bin/FCCU.php`.

2. Exercise a remote SQL-Injection vulnerability to perform these unauthorized tasks on the SQL server:
   (a) Show how you can log into a single account without knowing any id numbers ahead of time.
   (b) Show how you can log into any account you like (without knowing any id numbers ahead of time).
   (c) Make some account (your choice) wire its total balance to the bank with routing number: 314159265 and account number: 271828182845
   (d) Explain why you can't create a new account or arbitrarily update account balances (or show that you can).
3. Create an exploit transcript in the file /root/submission/exploit.txt, including your answers and how you completed the above attacks.
   (a) document all SQL Injection strings you used, including which pages and in what order.
   (b) This is not an executable script, but should be a step-by-step "walkthrough" of the attack that a colleague could follow without assistance.
4. Fix the vulnerability in the FCCU application by adding input validation and either character escaping or prepared statements.
5. Create a patch against the original source.
6. Quoting as little source code as possible, write a 1-page memo, including:
   - A description of the security flaw in the FCCU application
   - A description of how you fixed the flaw. How does your fix solve the problem?
   - Considering the FCCU application alone, describe a recovery plan for the server, answering:
     – How serious was this breach? Could attackers gain root access through this vulnerability?
     – What should be done with the server in order to secure it?
     – Include any other observations or thoughts you might have.
7. Store the following files in `/root/submission`:
   - your memo
   - your exploit walkthrough
   - your patch
8. Use the scripts described in the section for creating a submission tarball.

4.3 Using the scripts submit.sh and restore.sh

`submit.sh` will back up:

Note: do not run submit.sh and restore.sh as sudo!

- The FCCU code in /usr/lib/cgi-bin/FCCU.cgi
- everything in /root/submission, which should include:
  - exploit.txt
  - your memo
  - your patch

Note: do not run submit.sh and restore.sh as sudo!

Submit your tarball to your instructor.

`restore.sh` will restore those files to their original locations, automatically overwriting whatever is there.

5 Word Problems

1. Describe how the problems above could be mitigated if not eliminated completely. Sketch out a plan for containing such attacks.
2. How useful are debuggers or tools like IDA Pro in assessing vulnerabilities or building exploits? What other approaches can you think of for discovering vulnerabilities?

6 Deliverables

1. A report describing all your findings above.
2. A zip file containing:
   - The tar ball that resulted from the `submit.sh` script.
   - Answers from the word problems.
3. Submit by the class on May 7, 2025.

7 Grading

Points will be subtracted if any of the pieces of the deliverables are missing or incomplete. The late submission policy applies.