

Hands-on SQL Injection in the classroom: Lessons Learned*

Jens Mache¹, Carlos García Morán¹, Nic Richardson¹,
Wyeth Greenlaw Rollins¹, Richard Weiss²

¹Lewis & Clark College, Portland, OR 97219

{jmache, carlos, nrichardson, wyethg}@lclark.edu

²The Evergreen State College, Olympia, WA 98505

weissr@evergreen.edu

Abstract

SQL injections remain a serious security threat to applications using databases. In this experience paper, we report on teaching SQL injection hands-on using the EDURange platform in two different undergraduate courses, Web Development and Databases. We analyze the results from a voluntary survey with answers from 17 students who took the Web Development course and from 8 students who took the Database course. We focus our discussion around several lessons we learned, including the importance of guiding questions, covering unions and padding, and how to deal with the possibility of students adversely modifying the learning environment.

1 Introduction

Web-facing applications are common targets for attackers who seek to expose sensitive information such as passwords or credit card information. The list of the top ten security risks in 2021 at the Open Web Application Security Project (OWASP) [6] shows injection vulnerabilities as number three. This category includes SQL injection. Injection attacks can occur when unsanitized user

*Copyright ©2022 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

input is included in executed statements. These types of attacks are often used to gain access to sensitive information in SQL databases. Taylor et al. [8] have surveyed seven popular database textbooks and found that most of them do not cover this topic adequately. To address this problem, we have developed an exercise to teach students how SQL injections work and how to prevent them, which were the learning goals for the exercise we created.

One could discuss SQL injection attacks in class, but hands-on exercises are critical for learning and engagement. In this experience paper, we explore hands-on teaching platforms and discuss our experience teaching SQL injection using the EDURange platform [10, 2]. Our main goal was to explore how to teach SQL injection and what the obstacles might be.

2 Related Work

Numerous hands-on exercises have been created to aid the teaching of SQL injection. Yuan [11] describes eight web security labs, three of which deal with SQL injection. However, they are primarily focused on vulnerability detection and testing. They found that the “real-world” aspects of the exercises made the material more interesting to students. Du [3] outlines the labs of the SEED project, one of which focuses on SQL injections. The EDURange version goes beyond these attacks and can also run in the cloud or locally. Most important is that EDURange exercises are extensible by the instructor.

Li et al [4] outline eight security labs that can be played in two modes: war mode and peace mode. War mode has students work as ethical attackers while peace mode has students attempt to prevent attacks. However, the war mode has not been developed for the SQL injection lab.

The Web Application Hacker’s Handbook and the accompanying Portswigger website [7] have an extensive set of exercises. They provide an overview of SQL injection that includes an injection cheat sheet and a video explanation of the topic. They also provide injection samples to accomplish different goals such as retrieving hidden data, subverting application logic, UNION attacks, examining the database, and blind SQL injection. Alongside the explanatory information, they provide 16 labs. Each lab contains the author’s solution as well as community solutions in the form of videos. A few of the labs require the user to accomplish the same goal on different SQL implementations (MySQL, Oracle, etc). While focusing on multiple implementations might be appropriate for deeper dives into SQL, this can cause confusion in introductory SQL exercises. The Portswigger labs [7] require Burp Suite or other tools to complete the later labs. Burp Suite is a web security testing tool with the ability to intercept and modify HTTP requests made by the client. This capability is required in order to complete Portswigger’s blind SQL injection labs. However,

this could cause unnecessary friction during in-class activities. Portswigger contains a large body of work and might be too much to be included in Web or database classes. The EDURange containers provide all of the necessary software tools and tries to pare down the scope of the exercise while including some complex but important examples.

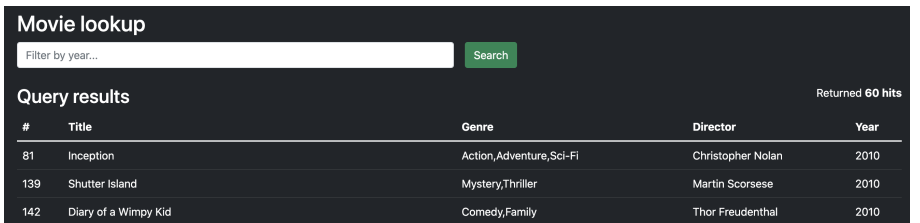
Basit [1] created a platform with 12 challenges to introduce students to SQL injection. Unfortunately, the levels hosted at www.databases.cs.virginia.edu/sqlinject/ are no longer available. They attempt to structure the levels in a way that each level builds on the previous. Level one asks students to create a SQL injection that will expose all usernames contained within a “users” table. The hint gives students the answer. Level two has the student create a SQL injection that exposes the names of other tables in the database. This is a big leap. It requires the student to be familiar with the implementation-specific SQL table called `information_schema.tables`. EDURange addresses some of the problems of bridging the gaps between levels by using guiding questions. In addition, EDURange is flexible. Instructors can easily modify exercises by adding their own questions or by inserting/deleting levels based on classroom data that is collected and student feedback.

3 Levels and courses

The goal of the EDURange SQL CTF (capture the flag) exercise is to teach students why SQL injections are important, how they work, and how to prevent them. It applies to any web application with a database back end.

Our exercise has three levels. Students are given the following information: the queries for level 1 (`SELECT * FROM countries WHERE name='<ARG>;'`) and level 2 (`SELECT * FROM books WHERE author LIKE '%<ARG>%';`), and a hint for level 3 ("count the number of columns in the table").

Fig. 1 shows level 3 after input without SQL injection. Fig. 2 shows level 3 after a successful SQL injection.



#	Title	Genre	Director	Year
81	Inception	Action,Adventure,Sci-Fi	Christopher Nolan	2010
139	Shutter Island	Mystery,Thriller	Martin Scorsese	2010
142	Diary of a Wimpy Kid	Comedy,Family	Thor Freudenthal	2010

Figure 1: Level 3 after ‘2010’ has been entered.

Movie lookup

' UNION SELECT *, null, null FROM users --

Search

Returned 5 hits

Query results

#	Title	Genre	Director	Year
1	admin	0192023a7bbd73250516f069df18b500		
2	tom	8a24367a1f46c141048752f2d5bbd14b		
3	backdoor	737c04bbf91056509f2fd3e25b7b3dc8		
4	31337	238e96d5b62a1aec3739730469f27192		
5	flag	FLAG{1M_NOT_4_H4SH3D_H4X0R}		

Figure 2: Level 3 after a successful SQL injection.

Students were also given the following guiding questions:

1. What is a "comment" symbol?
2. What boolean operator did you use to dump the table from level 1?
3. What is the flag for level 1?
4. What special character is the LIKE query using in level 2?
5. What is the flag for level 2?
6. What keyword could you use to query two combined tables?
7. What is the flag for level 3?
8. What is the password for user "backdoor"?

In Spring of 2022, we taught SQL injection in two undergraduate courses, web development and databases, using the EDURange platform. Students in the database class had spent several weeks writing SQL queries, while those in web development only had about one week. The same instructor introduced the same exercise in both classes.

In the web development course, students had an hour long class period on one day to mostly independently try to answer the questions. They were asked

to spend an additional hour after class to continue exploration. The next two class days provided more time and review, but also introduced another topic.

In the database course, one of the co-authors of this paper (who had been sitting in the back for most of the term) was invited to give a guest appearance. The same SQL injection exercise was used. Since the time available was limited to about 40 minutes of one class day, the instructor showed a guided walk-through in which the students could follow along.

4 Student survey

In both courses, in an anonymous survey, students were asked six questions on a 5-point Likert scale. They were also asked the two following open-ended questions: "What problems did you encounter in completing the lab?" and "What changes could be made to the lab to enhance your learning?"

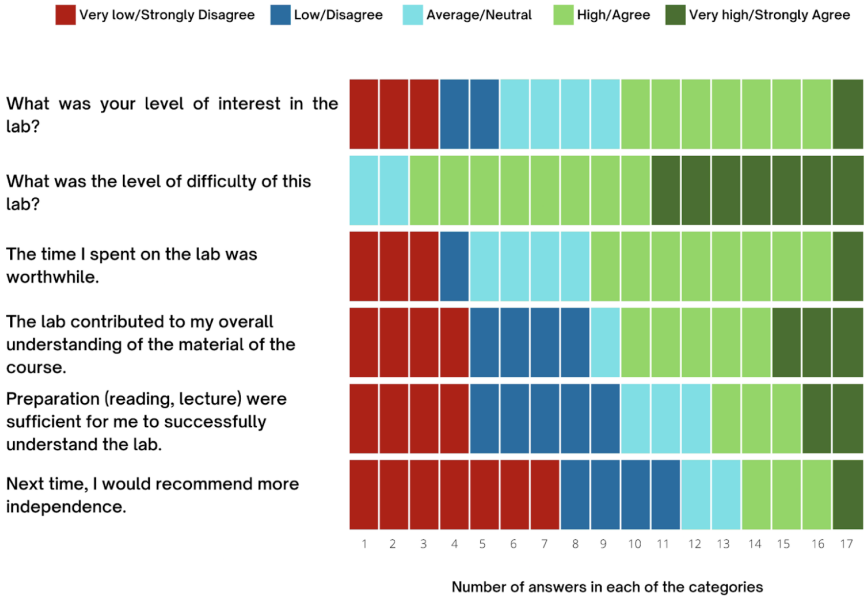


Figure 3: Survey results from the web development course.

We received responses from 17 students in the Web Development course and 8 students in the Database course. Most students in the Web Development course found the activity to be interesting and challenging. Looking at the graph, a theme emerges suggesting students in this course wanted more

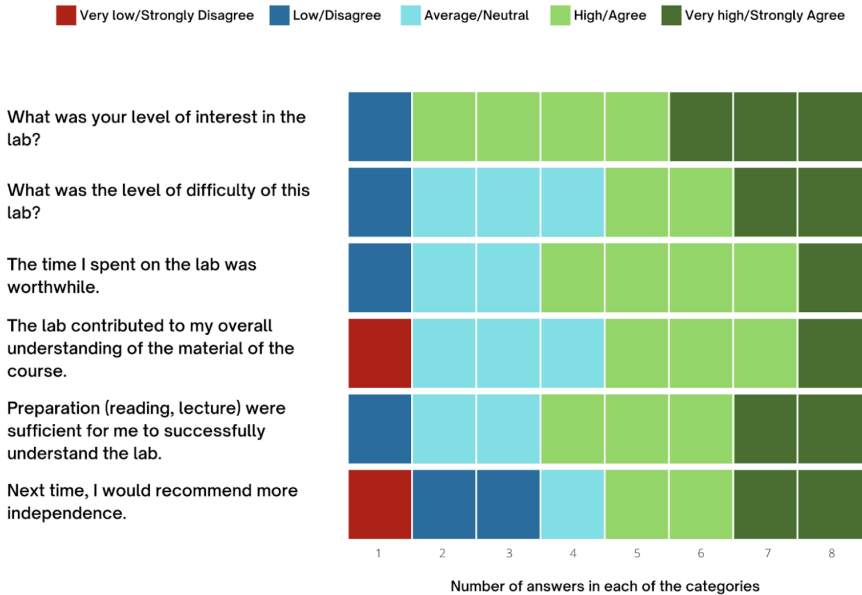


Figure 4: Survey results from the databases course.

guidance before and during the activity. In the two open-ended questions, Web Development students reported that they felt confused or “unsure what to do a lot of the time.”

In contrast, more students in the databases course agreed that the activity was interesting and helpful to build their understanding of the course material. The database students’ answers to the open-ended questions suggest they were less confused about how to complete the exercises. There was a higher demand for independence rather than guidance during the exercises. One student responded saying they would have liked “[m]ore time to sit and think about how to solve the problems, [and] less time spent being told how to do them.”

5 Lessons Learned

5.1 Guiding questions

This feedback from the students helped to inform our lessons learned. Specifically, it helped to demonstrate the importance of guiding questions in leading students through the exercises. Where the web development class could have used better hints and guiding questions, the database students could have ben-

efited from more variety in the types of SQL injections used and more time to solve the problems on their own. It is clear that platforms used to teach SQL injection must be flexible. Instructors should be able to adjust questions to match the level of difficulty and guidance which will suit the students best. The EDURange platform [2] now includes an editable YAML file which instructors can use to change and rearrange the guiding questions associated with the exercise. Guiding questions are especially useful for advancing to levels where it is difficult to get the solution by trial and error, as in the case of padding (to force matching numbers of columns in an UNION statement).

5.2 UNION attacks and padding

The UNION operator is used to combine the result-set of two or more SELECT statements. It can be used to spill secrets from tables other than the one in the original query. However, to use UNION, each SELECT statement must produce the same number of columns with the same data types. The former can require padding, a topic that some students may be unfamiliar with and find difficult to grasp. We have seen multiple approaches to padding. Some exercises avoid padding altogether by only using a single column while others try to demonstrate the best methods for finding a table's dimensions. We believe that it is important to address padding as it is often needed to display data on the site that is being attacked. We feel that it is most beneficial to start with a simplified example using only one or two columns (to avoid the need for padding), before demonstrating methods to count columns for padding.

5.3 Progression of levels

Level 1 of our exercise introduces comments, quotes and logic. These rudimentary parts of SQL injections are easy to utilize. With this knowledge students can start to retrieve unintended information and gain unwanted access. After students have used these ideas, they are ready to bypass common prevention measures such as blocking all comment symbols. This leads to an opportunity for students to learn that different representations (like ASCII) for these characters can be used in their place. Next, students can be introduced to queries that use the LIKE clause and how wildcards can be used to match common names of tables, columns, etc. After these concepts, students can begin forming injections that query information about the database itself. This may include the version, table names, and the number of rows or columns. At this point, padding should be discussed as it is common for the number or type of columns returned by the injection to not match those of the original query. It is our opinion these introductory exercises should only use one SQL flavor as the use of multiple may cause unnecessary confusion. Guiding questions

should be provided for each level.

5.4 How to handle "DROP TABLE"

After students learn these new and intriguing techniques, our hope is that they will get satisfaction out of putting them into practice and experimenting with new ideas. Unfortunately, the place students may attempt this experimentation is on our site (that hosts the hands-on exercise). This can cause problems if the appropriate precautions are not taken. One of the more harsh injections a testbed needs to be ready for is students attempting to drop tables. This is something that we experienced during the first iteration of our exercise. Fig. 5 shows the apologetic email we got from a student.

I dropped the main data table from my webfu page. Whoops. I couldn't figure out what I was supposed to do, so I thought I should try breaking it! I guess I tried too hard.

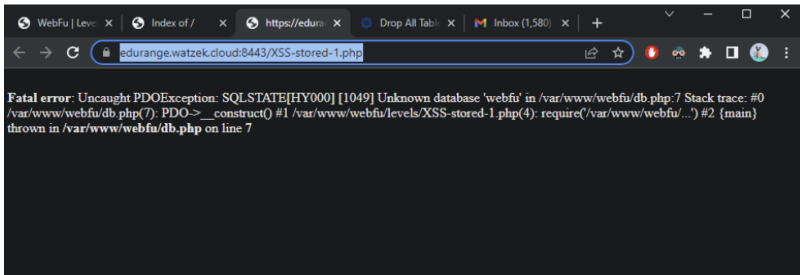


Figure 5: Email from a student.

How to combat this while still giving students the opportunity to experiment with SQL injections? One resource-intensive option would be to have individual containers or databases for every student. This would allow students to experience destruction without damaging the learning environment for other students. In our current setup, we try to detect and prevent the DROP command and similar statement that alter the state of the database. We do notify the student when we detect a potentially destructive command. At any rate, it is recommended to be ready (with scripts) to rebuild the learning environment to its original state.

5.5 Learning Environment

We believe an important part of a learning environment includes hands-on activities. Hands-on activities give students many opportunities to actively participate in the classroom and aid their learning. With interactive platforms, we can allow students to test their knowledge immediately and help them retain the knowledge and get assistance if needed. Cloud-based learning platforms eliminate barriers to entry for students as well as many compatibility issues which could limit students. Often with local environments, a lot of the student’s time could easily be taken up by attempting to set up the environment instead of being able to practice the material at hand. Additionally, we believe that gamification is a very strong way to maintain student interest and help them enjoy the material. Some common ways we implemented gamification is by giving the students tasks to find specific information within the site such as flags and passwords. Future work would be to turn this into a Red Team/Blue Team exercise by allowing students to modify code.

5.6 Logging student activity

We wanted to log user input within the scenario to see how students were approaching problems and to identify common misunderstandings or struggles for students within the testbed. EDURange has the capability to capture student actions [5]. Theoretically students could be individually identified by their session. However, this can be blocked by using incognito browser tabs. Indeed, a browser assigns a new session to each new incognito tab that the student may open, causing the logger into treating each session as a distinct user. Once we are able to consistently distinguish between users, we will be able to assign logged activities to participants accurately. This can help us improve feedback to students that are struggling with any part of the activity. In the future, we plan to extend our previous work [9] and use machine learning to identify these patterns and determine the best time to help students through the activity with a helpful hint.

336	eyJFZnlhZ2gicSQL-1	Cuba	{[\"id\": \"57\", \"name\": \"Cuba\", \"code\": \"CU\"]}	
337	eyJFZnlhZ2gicSQL-1	Cuba\\'-	[]	PDOException: SQLSTATE[42C 2022-04-22T13:01:01-07:00
338	eyJFZnlhZ2gicSQL-1	Cuba\\' OR 1-1	[]	PDOException: SQLSTATE[42C 2022-04-22T13:01:05-07:00
339	eyJk1kA0FSQL-1		NULL	2022-04-22T13:06:33-07:00
340	eyJk1kA0FSQL-1	\\' OR 1=1;--	{[\"id\": \"1\", \"name\": \"Afghanistan\", \"code\": \"AF\"]}	
341	eyJZ0sFKAzFSQL-2		NULL	2022-04-22T13:06:59-07:00
342	eyJZ0sFKAzFSQL-2	%	{[\"id\": \"1\", \"title\": \"Brave New World\", \"author\": \"Aldous Huxley\"]}	
343	eyJt0B1kAzFSQL-1		NULL	2022-04-22T13:07:17-07:00
344	eyJt0B1kAzFSQL-1	Cuba	{[\"id\": \"57\", \"name\": \"Cuba\", \"code\": \"CU\"]}	
345	eyJZ0s1kAzFSQL-3		NULL	2022-04-22T13:07:23-07:00
346	eyJZ0s1kAzFSQL-3	\\' UNION SELECT table_name FROM information_schema.tables;--	[]	PDOException: SQLSTATE[21C 2022-04-22T13:07:29-07:00

Figure 6: Excerpt from the log file showing line number, session id (truncated), exercise level, user input, output (if any), error (if any), timestamp.

6 Conclusion and Future Work

Using hands-on exercises in the EDURange platform, we integrated the security topic of SQL injection into two different non-security undergraduate courses, Web Development and Databases. We collected feedback from student activity logs, a voluntary survey, and from direct communication with the students.

Many students had questions about UNION attacks. These attacks are often used to retrieve data from other tables, e.g. passwords of users. In general, one needs to use padding for this. Based on student feedback, we plan to improve the exercise in two ways. We would add a preparatory level that uses UNION without padding, and we would add more guiding questions on this topic.

Even if the exercise doesn't call for it, curious students may experiment with the "DROP TABLE" statement and thus destroy the learning environment. We discussed several ways in which this could be handled.

Future work includes using machine learning to identify students having trouble, especially with padding and to evaluate which hints to give and when.

7 Additional Information

We contacted our IRB and got exempt status. We plan to make some of the artifacts available to the community. We would like to especially thank Alain Kaegi. This work was supported by the National Science Foundation under grants 2216485 and 2216492.

References

- [1] Nada Basit et al. "A Learning Platform for SQL Injection". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE '19. Minneapolis, MN, USA: Association for Computing Machinery, 2019, pp. 184–190. ISBN: 9781450358903. DOI: 10.1145/3287324.3287490. URL: <https://doi.org/10.1145/3287324.3287490>.
- [2] Jack Cook et al. "An authoring process to construct docker containers to help instructors develop cybersecurity exercises". In: *Journal of Computing Sciences in Colleges* 37.10 (2022), pp. 37–47.
- [3] Wenliang Du. "SEED: Hands-On Lab Exercises for Computer Security Education". In: *IEEE Security & Privacy* 9.5 (2011), pp. 70–73. DOI: 10.1109/MSP.2011.139.

- [4] Lei Li et al. “Developing Hands-on Labware for Emerging Database Security”. In: *Proceedings of the 17th Annual Conference on Information Technology Education*. SIGITE '16. Boston, Massachusetts, USA: Association for Computing Machinery, 2016, pp. 60–64. ISBN: 9781450344524. DOI: 10.1145/2978192.2978225. URL: <https://doi.org/10.1145/2978192.2978225>.
- [5] Jelena Mirkovic et al. “Using terminal histories to monitor student progress on hands-on exercises”. In: *Proceedings of the 51st ACM technical symposium on computer science education*. 2020, pp. 866–872.
- [6] OWASP. 2022. URL: <https://owasp.org/www-project-top-ten/>.
- [7] Portswigger. 2022. URL: <https://portswigger.net/web-security/sql-injection>.
- [8] Cynthia Taylor and Saheel Sakharkar. “’);DROP TABLE Textbooks;--: An Argument for SQL Injection Coverage in Database Textbooks”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE '19. Minneapolis, MN, USA: Association for Computing Machinery, 2019, pp. 191–197. ISBN: 9781450358903. DOI: 10.1145/3287324.3287429. URL: <https://doi.org/10.1145/3287324.3287429>.
- [9] Quinn Vinlove, Jens Mache, and Richard Weiss. “Predicting student success in cybersecurity exercises with a support vector classifier”. In: *Journal of computing sciences in colleges* 36.1 (2020).
- [10] Richard Weiss et al. “Finding the balance between guidance and independence in cybersecurity exercises”. In: *2016 USENIX Workshop on Advances in Security Education (ASE 16)*. 2016.
- [11] Xiaohong Yuan et al. “Hands-on Laboratory Exercises for Teaching Software Security”. In: *Proceedings of the 16th Colloquium for Information System Security Education* (2012).