'; DROP TABLE 'portfolios'; --

Timothy Aveni

Spring 2013

0 Table of Contents

1 To the Reader

2 The Author; Dedication

3 SQL Injection WNB

My beautiful reader: The eighteenth of June, two thousand and thirteen

I can't guarantee that you'll particularly like my work. I don't anticipate that you'll appreciate my writing style very much. I can tell you that I definitely don't. That's not the point, though, because at this point you've already read a solid four sentences of my words. If you disliked them so greatly, you'd have been gone by now.

I like to pretend to be sane, but the façade is rather difficult to maintain. When you walk by me on the street or have a short conversation with me in the hallway, you hopefully leave with the belief that I am only mildly insane. This portfolio, however, contains my ramblings during those times that I was not in the mood to contain myself, and I don't suppose that your opinion of me will be favorable once you have finished reading whichever portion of this work that you intend to read.

I am glad that you are reading this, though. I did not compose it with the intent of throwing it in the garbage. I am not of the opinion that this portfolio is garbage. I want you to analyze my work. I want you to realize everything I did wrong, and I want you to judge me for it. This portfolio might well be the death of your respect for me, but I anticipate that it will be the birth of your understanding of my underlying existence.

The title of this book is a silly reference to a programming concept that I discuss on page 3 of this book. I chose it with the intent of demonstrating the lack of seriousness in the nature of the portfolio as well as my interest in programming.

I have composed a very small portion of music for each page of this portfolio. The music was probably written on a whim, though I intended to portray through the music the emotion that came out while I was writing the text. I wrote some of this writing, like my thesis paper, with the intent of being emotionless. In some instances, I capture this lack of emotion by writing little more than a major scale on the page. In others, I free myself from the text and compose whatever music I'd like to compose.

I have recorded each slice of music on whichever instrument pleased me. You can access the recordings by navigating to the online version of this portfolio, which you are able to access at http://timothyaveni.com/portfolio.

I should warn you that there might be some inconsistencies with the spelling conventions I use for words. In casual situations, I tend to use British English in most of my spelling. My teachers dislike that, and prefer that I use American English. I have tried to eradicate most of the British English in this portfolio in favour of American English, but I cannot guarantee perfection.

Yours,
Timothy Aveni

The Author

Timothy Aveni, born on the fifteenth of February nineteen ninety-eight, is a young male who has found himself in an interesting position. He is respected by those around him, though he is not a particularly kind fellow. He prefers to avoid interacting with others when he can, with the exception of a few close friends who do their best to keep him sane. Still, Timothy finds himself content when he is alone, and is willing to spend as much time as is necessary to gather his thoughts away from others.

In his spare time, he programs. He loves lots of different sorts of programming, so he is never bored. He frequently explores new programming concepts, professing their greatness (or lack thereof) to anyone who will listen. His interest lies in computer science, and he interests himself specifically in the construction and population of neural networks.

He greatly enjoys playing, singing, and creating music, no matter what mood he is in. He plays his trumpet in the wind ensemble, though he enjoys playing his flute just as much. He is not a particularly good singer (don't tell him, though), but he will loudly sing whatever song is in his head without shame.

His lack of reluctance to make himself look silly is what brings success forth to him. In eighth grade French class, when tasked with presenting a poem in front of the class, he fitted a tune to the poem and sang the entire thing. He is not obsessed with identifying as an outcast, but he will violate convention when it does not suit him. He prefers to think through his actions before executing them, and occasionally finds himself in opposition to those around him. He doesn't really mind. He just kind of does whatever he wants.

Timothy is a pretty happy guy.

Dedication

There are a lot of people to whom I *should* dedicate this book; its dedication belongs to teachers, those whom I love, those who love me, and those who have helped me. I have dedicated this portfolio to none of those people.

No, I dedicate this book to a concept. You might think that this is cheating or disallowed, but I argue that nothing in our minds is more than a concept. I have seen people dedicate books to their God; is this God not a concept, as well?

I hereby feel completely justified in dedicating this portfolio to freedom. In this sense I do not mean personal freedom, as in the freedom to spontaneously jump into a lake, or economic freedom, as in free tacos, or independence, which allows one to be free of restraint. I refer specifically to the freedom granted by one party to another that gives permission to use content in a specific manner, as in Free Open-Source Software.

me@timothyaveni.com

Writer's Notebook: SQL Injection 24 March 2013

SQL injection is an interesting sort of software vulnerability. It is relatively common in websites, and I've actually found a few vulnerabilities myself that were based around SQL injection.

Let me start by describing SQL. SQL, of Structured Query Language, is a language used to store and retrieve data in databases. Each database can have multiple named tables. Tables have multiple named columns. Tables can them have entries (rows) inserted that use the columns as their definition.

For a website, I might have a table called "users" with columns "username", "email", and "password". "password" would likely not be stored in plain text, but rather in a hash. for more information about hashes, refer to MP2 page 13.

To insert a new user into the "users" table, I'll write:

```
1     INSERT INTO 'users' VALUES('george', 'george@gmail.com', 'a5de69f4');
```

When a website gets a hit for the "create user" page, it might run some code like this:

```
1     userName = userInput("username")
2     passWord = userInput("password").hash()
3     email = userInput("email")
4
5     query = "INSERT INTO 'users' VALUES('" + userName + "', '" + email + "',
'" + password + "');"
6     mysql_execute(query)
```

This would take user input and shove it into the database. Remember this code example, because it is important.

To obtain data from a database, I write:

```
1     SELECT * FROM 'users' WHERE username='george';
```

Which will return with a row that contains George's username, email, and hashed password. I can use similar code as before to dynamically form any SQL query.

Another thing I should mention now is the concept of SQL 'comments'. When the combination of characters "--" is located somewhere in an SQL statement, everything after the "--" is not executed. This is useful for writing comments in SQL without throwing an error. This:

```
1     SELECT * FROM 'users'; hi I like tacos
```

would give an error, because "hi I like tacos" is not valid SQL. This:

```
1     SELECT * FROM 'users'; -- hi I like tacos
```

would not, because everything after -- is ignored.

An attacker can use these concepts to his advantage.

Imagine a "log-in" procedure in your code that goes like this:

```
1     userName = userInput("username")
2     passWord = userInput("password")
3
4     query = "SELECT * FROM 'users' WHERE username='" + userName + "' AND
password='" + passWord + "';"
5     mysql_execute(query)
```

The code just inserts the user's username input and password input into the SQL statement. Say I wanted to log in with username "george" and a password hash of "ef6dbe3". The statement would look something like this:

```
1     SELECT * FROM 'users' WHERE username='george' AND password='ef6dbe3';
```

If the username exists in the databases, but the password is WRONG, the SQL statement will not return a user row- there is no row that exists in which the username is 'george' and the password is some random, incorrect hash.

Think about this. What if an attacker were to enter this:

```
george';--
```

into the username field, and leave the password field blank? Go on, think about it.

The resulting statement is devious:

```
1     SELECT * FROM 'users' WHERE username='george';--' AND password='';
```

After the SQL comment is applied, that will evaluate to:

```
1     SELECT * FROM 'users' WHERE username='george';
```

Now you don't even need a working password to log in! If you can sneak it by the website, you can log into anyone's account. Worse, imagine this input:

```
'; DROP TABLE 'users';--
```

Which evaluates to:

```
1     SELECT * FROM 'users' WHERE username=''; DROP TABLE 'users';
```

Heh... there goes your entire table of user accounts.

Prevention of SQL injection attacks is easy. First, databases generally support individual database access accounts with different permissions. Create a user without the "DROP TABLE" permission, and at least your tables will be okay.

It's also possible to "sanitize" your input strings to safely be added into SQL without the possibility of code execution. That's not the best solution, but it's rather popular in basic websites.

As much as I'd love to pretend I'd like writing, I've hit the twenty-eighth page. Goodbye until the fourth marking period.

7 minus two