

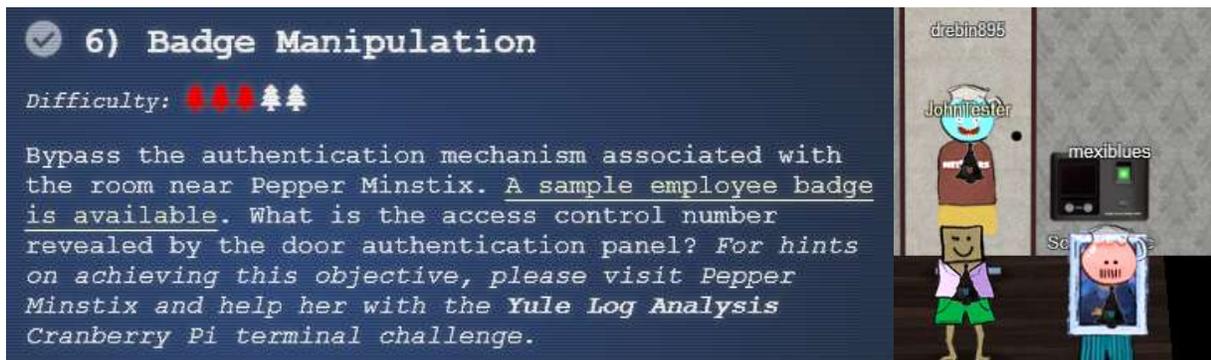
Objective--Badge Manipulation (Part 1)

What you can learn from this

People have been injecting commands into SQL databases through web sites for a long time. Despite intense education efforts, there are still many web sites that are vulnerable to SQL Injection (SQLI). SQLI has been used in many famous breaches, so anyone who works in IT security should have a basic understanding of SQLI. Researchers have discovered that badge systems often make queries to databases, and those are often vulnerable to SQLI as well. This challenge will take you through the basics of SQLI and help you develop an injection that will allow you to bypass the scanner and enter the secret room

Getting Started

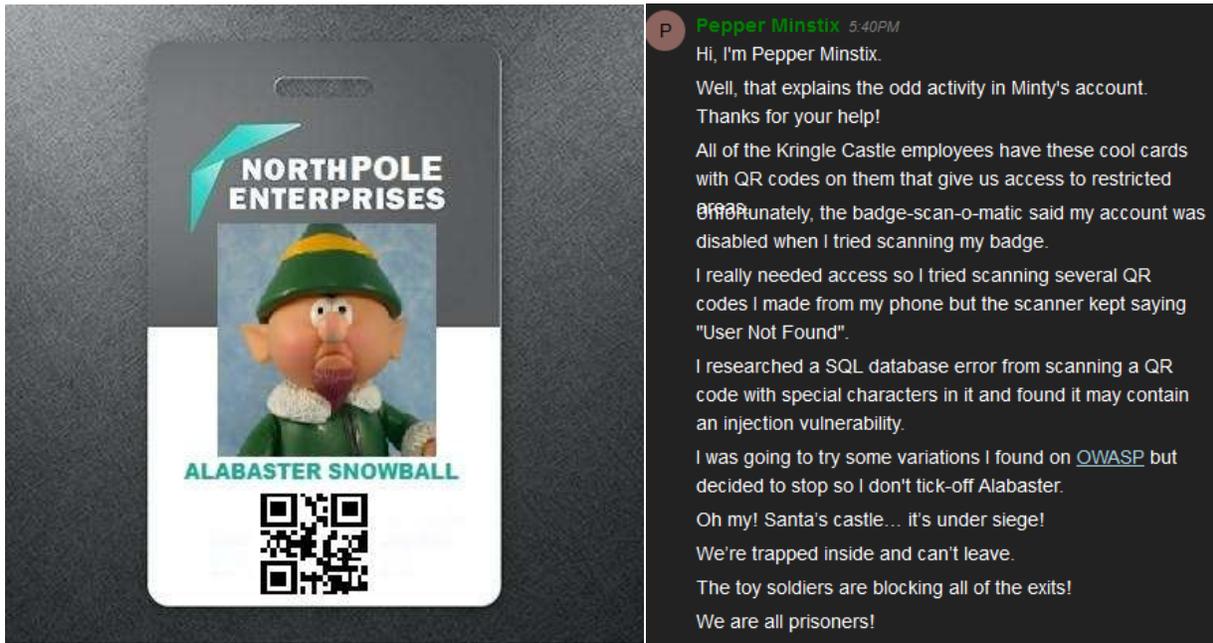
The scanner is farther down the hall, past Pepper Minstix and the Yule Log terminal on the second floor, right side.



Here are some hints from Pepper. The links will appear later in the document.

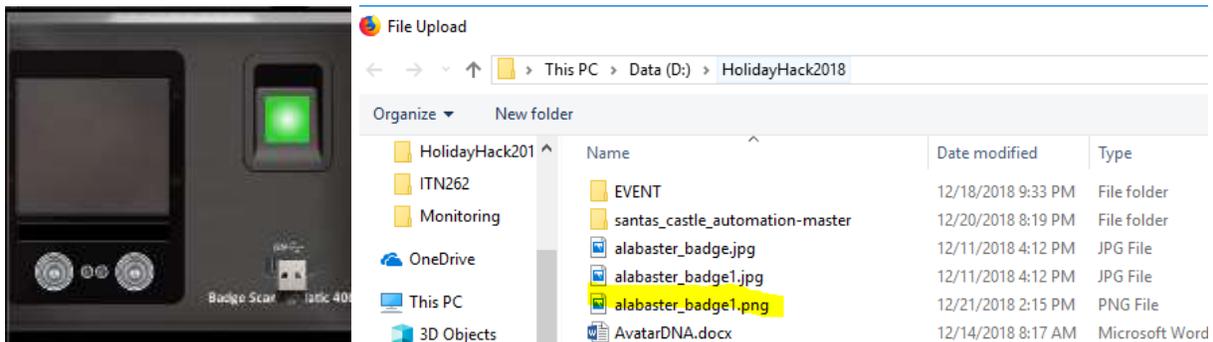


This is the [link with the sample employee badge](#) and here is the badge itself. You can tell from Pepper's hints that we are going to be working on SQLI.



The hint about SQLI in your badge from Pepper goes to [this link](#).

The QR code at the bottom of the badge is what the scanner reads. Fortunately for us, the "USB drive" on the scanner will accept .png files with the QR code. Unfortunately, it won't accept the picture we have of Alabaster's badge, as is. The scanner wants just the QR code, with only white space around it. This picture below was the result of cropping everything but Alabaster's QR code. Let's see if we can get in with Alabaster's code.





Rats, Authorized user account has been disabled.

Let's put Alabaster's code in to a QR decoder to see what it is.

→ ↻ <https://zxing.org/w/decode>

Decode Succeeded	
Raw text	oRfjg5uGHmbduj2m
Raw bytes	41 06 f5 26 66 a6 73 57 54 74 86 d6 26 47 56 a3 26 d0 ec
Barcode format	QR_CODE
Parsed Result Type	TEXT
Parsed Result	oRfjg5uGHmbduj2m

It must be a random string that identifies Alabaster in the database.

Poking about the database

A good first step in SQLI is to try to get the database to generate an error. Some error messages are informative, telling us how we should proceed with the attack. Instead of a string like the one in Alabaster's badge, we can create our own QR code with some special characters in hopes of getting an error. The old standard for SQLI is `OR 1=1`, so we might as well try that. Try anything you like, just make sure that it includes a special character or two.

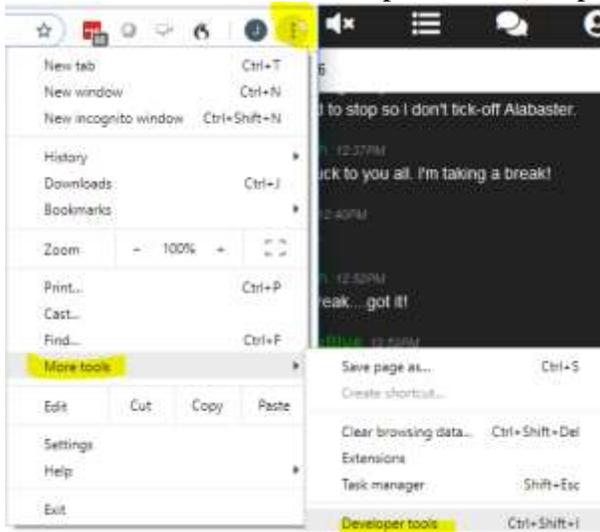
To make a QR code, go to the other hint Pepper put in your badge called [Bar Code Creation](#). That site will accept a string of your choosing and generate a QR code in a .png file. You can present the .png file to the scanner. Be careful not to click on the advertising links near the

bottom, you want the SAVE link at the top right.

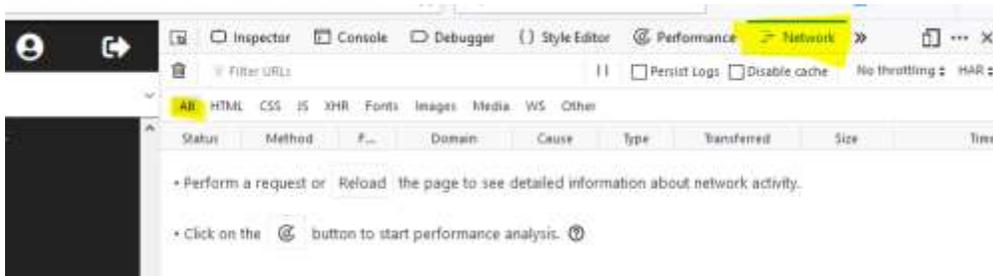


Hmm, the OR 1=1 just gave us the error “No Authorized User Found.” Let’s try something else, like aaa' OR 1=1

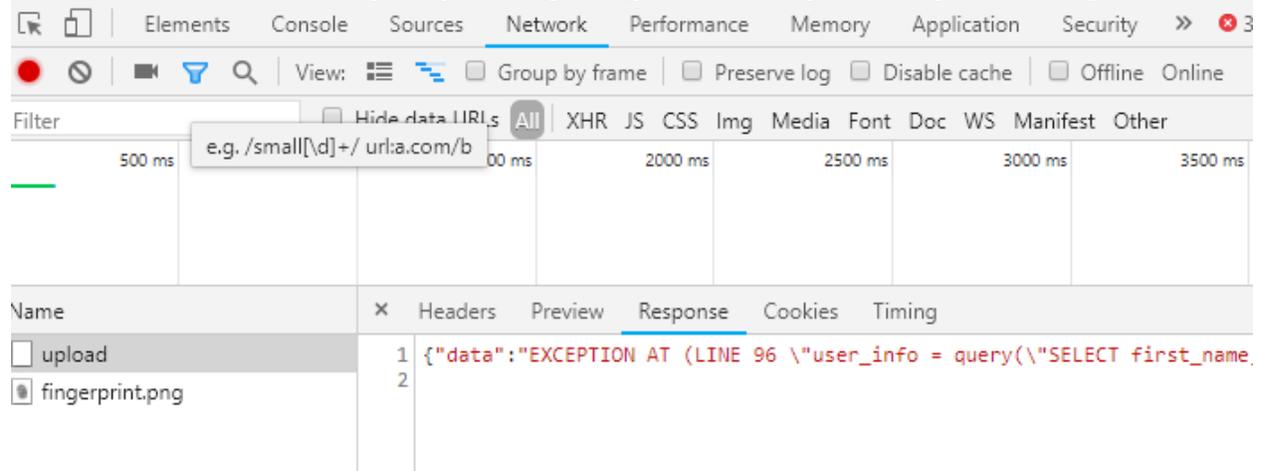
If that works, you should see a long error message scroll through the scanner display window. If you like, you can write very quickly as the error goes past, take pictures of it with your phone, or let Chrome developer tools do the work for you. Before you submit the QR .png file to the scanner, More tools > Developer tools. (Or, press F12.)



Then select Network and All.



Once you open the scanner and submit your QR code, you should see traffic between your browser and the site. Since you uploaded your QR code, the upload entry is the one you want.



Hand In

- 1) What text did you inject to generate the long SQL error?
- 2) What is the error message?
- 3) What is the SQL query the scanner uses? Strip away all the error message text until only the query remains.