Raspberry PI Pen Testing

The aim of this lab is to introduce you to the world of penetration testing with a basic demo. Following this lab sheet, you will learn about SQL injection (As well as some basic SQL) and buffer overflows in a C program. **Beware**, trying any of these techniques on any system without permission is a **crime and you will be caught and prosecuted for it**. With that out of the way, at the end of the cheat sheet is some helpful resources to safely and legally practice pen testing in your own time.

Task 1: Network Mapper

First, we need to find out what services are running on the system. To do this, open a new console and scan 'localhost'. Use the results to answer the following questions…

1) Name the services running on port 80 and 3306

_____

Task 2: SQL Injection

Having scanned the localhost, you've found there's two services running on this system. One of them is a website being hosted on port 80. Type the following address into the browser:

localhost:80

Once you're finished exploring, travel to the search page. Here there is a search bar that corresponds to the database running on the system. It's important to verify that this search bar is secured against SQL injection attacks. To test this, type the following into the search bar and hit search:

' UNION (SELECT 1,2,3 FROM dual);#

The results don't look good. The dual table is present in databases for testing, and the fact we can access this means there is no protection against and SQL injection attack. The way this works is as follows. Using the search normally would place the term into an SQL query as such where Search_Name is what we put in the search box:

SELECT * FROM ? WHERE ? LIKE '%<Search_Name>%';

However, by inserting a single quotation mark, writing our code then commenting out anything after that using the hash (#) character we can manipulate the results to show us other tables.

SELECT * FROM ? WHERE ? LIKE '%' UNION (SELECT 1,2,3 FROM dual);#%';

Using this, search for and create SQL queries and answer the questions below. (There are hints available on the hint sheet if need be)

1) Find the name of the third, secret table

_____

2) Find the name from the table and decode the password inside

_____

Task 3: Buffer Overflow

A more modern and constantly occurring example of a vulnerability is programs susceptible to a buffer overflow attack. The basic idea of these attacks is to input a value to a program that's bigger than the variable it is being assigned to. In C programming languages, without careful consideration these vulnerabilities can go unnoticed. Included on this system is a basic example to demonstrate this attack.

To find this program, first open a new console and navigate to the 'BufferOverflow' folder found in the documents folder. In this folder there is a program called 'Vuln' containing the vulnerability. To use this program, type the following into the console (Remember the . at the start):

./vuln hello

You'll notice that it outputs a before and after of a variable B. What's happening here is the program is copying your input (hello) into an array called buffer. Before it does this it outputs B, carries out the copy then outputs it again.

The buffer array is only big enough to store 8 characters. Try replacing hello by inputting enough characters to exceed this.

1) What number was B after inputting enough characters to overflow the buffer?

_____

The variables are declared next to each other in code, therefore when the buffer overflows it does so into the neighbouring B variable. This is a tame example; however, it can be exploited to devastating effect resulting in access to the whole system. With a larger input and a bit of trial and error, the program's own instructions can be overwritten with a payload containing malicious code that gets executed instead.