Ira A. Fulton Schools of Engineering
Arizona State University
Engineering | School of Computing, Informatics, and Decision Systems Engineering

CSE 545: Software Security

Backdoor Web Server Assignment

Purpose

A critical part of establishing persistence on a system is to leave a “backdoor” that allows the hacker access to the system at a later date, without exploiting the same vulnerabilities (they may be fixed in the meantime). In this assignment, you will explore writing a backdoor that pretends to be a web server. A web server makes a great pretense for a backdoor, because web traffic is so prevalent it does not raise alarms and ports 80 and 443 are frequently permitted through firewalls. In addition, you will become familiar with network programming in C with only the C standard library as well as reading technical specifications.

Your goal is to create, in C, a minimal HTTP 1.1 server, based on RFC 2616 from scratch, without using any libraries except for the C standard library.

Objectives

Students will be able to:

- Develop a network server in C
- Analyze a complex technical specification
- Implement a protocol from a technical specification
- Develop a minimally compliant HTTP 1.1 server in C

Technology Requirements

Ubuntu 18.04 64-bit with the default packages installed. Highly recommended to create a virtual machine running Ubuntu 18.04.

Project Description

The name of your backdoor executable will be `normal_web_server`.

You must implement the following command-line interface for your server:

```
./normal_web_server <port>
```

Your server should listen for incoming connections to the given port, and respond to most requests with a valid HTTP 1.1 response with the 404 HTTP response code.

It is important that your server support valid HTTP 1.1 requests from HTTP clients (otherwise your backdoor will be detected), and your server should not cause the client to hang or otherwise malfunction.

The backdoor functionality is: when your server receives a GET request for a URL in the form of `/exec/<command>`, then your server should take `<command>` and execute it using the `system` libc function and the HTTP response will be the stdout of the executed command. The HTTP status code of the response should be 200. Note that there are no limitations to the characters in `<command>`, in other words, your program should capture the rest of the requested URL from the `/` after `/exec` to the end of the URL.

For instance, an HTTP GET of `/exec/ls` will return an HTTP response with the body of the output of the execution of the `ls` command on the server. An HTTP GET of `/exec/ls%20-la` will return an HTTP response with the body of the output of `ls -la`.

When the server is killed (Control-C via command prompt or the SIGINT signal is sent to the program), the server should release the port and safely terminate.

You will need to submit your source code, along with a Makefile and a README. The Makefile must create your executable, when the command `make` is run. Your README file must contain your name, ID, and a description of how your program works.

Your submission will be tested on Ubuntu 18.04, and that is the only platform that is supported. If your program exhibits different behavior on your system and Ubuntu 18.04, the behavior on Ubuntu 18.04 will be the one that is graded.

Submission Directions for Project Deliverables

In the submission space in the course, you will need to submit a `.tar.gz` file that contains your source code, Makefile, and README. Your README file must contain your name, ID, and a description of how your program works.