Lab 4 - Injection

CSE 4481 4.0 Computer Security Lab, Winter 2013

Due: Sunday, Feb 24th, 2013, 11:59pm.

Format: Individual

Learning Objective: To understand various injection problems such as command injection, code injection etc. Injection attack is the exploitation of software bugs caused by processing invalid data. The goal of this lab is to study ways to exploit different injection vulnerabilities and demonstrate the damage that can be achieved by such an attack.

Task 1

The following program runs with root privileges and is supposed to execute the `/bin/ls` command. However, the programmer only uses the relative path for the `ls` command, rather than the absolute path:

```
int main()
{
        system("ls");
        return 0;
}
```

Utilizing this program, devise a way to run arbitrary code with root privileges. Also, devise a way to get a root shell.

Report: Describe the process you followed to achieve the two goals above.

Task 2

Bob works for an auditing agency, and he needs to investigate a company for suspected fraud. For the purposes of the investigation, Bob needs to be able to read all the files in the company's Unix system. On the other hand, to protect the integrity of the system, Bob should not be able

to modify any file. To achieve this goal, Vince, the superuser of the system, wrote a special set-root-uid program (see below), and then gave the executable permission to Bob. This program requires Bob to type a file name at the command line, and then it will run `/bin/cat` to display the specified file.

Since the program is running as root, it can display any file Bob specifies. However, since the program has no write operations, Vince is very sure that Bob cannot use this special program to modify any file. Your goal is to check whether Bob can compromise the integrity of the system. If the program has security bugs, you should fix them.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
  char *v[3];
  if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
  }
  v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;
  char *command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
  sprintf(command, "%s %s", v[0], v[1]);
  system(command);
  return 0 ;
}
```

Report: Explain any attacks you devised, and provide the modified source code that fixes the problems.

Task 3

Let us assume that a developer wrote the following code.

```
int main()
{
  sleep(1);
  return 0;
}
```

How can you make this application print "I am not sleeping" without changing the code? Hint: `LD_PRELOAD` adjusts the runtime linking process by searching for shared libraries at alternative locations.

How can you prevent such an attack (without using static linkage)? Hint: investigate the conditions for the run-time linker to ignore `LD_PRELOAD`.

Report: Provide answers to the questions above.

Task 4

On the course website, you can download a simple Java application that checks whether a user is allowed to enter a casino. Your goal is to:

1. Hack the application in such a way that a user can log-on into the casino with any password.
2. Discover the original password(s) of the casino application.

Hint: Review the Java code signature tutorials and the obfuscated code and Java decompiler *Jode*.

Report: Describe both attacks.

Task 5 - scan4481

Develop a simple security scanner called `scan4481`. The scanner attempts to find vulnerabilities in the server component of the chat application you developed for the term project. The scanner will connect as at least two clients and then send a series of generated input to the server. The scanner must generate both valid and invalid messages, and send the messages before and after

authentication. Also, the scanner must simulate a DoS attack. Finally, the scanner must verify that the server handles all input as expected.

The scanner must run from the command line and it may include any set of parameters you believe is appropriate.

Report: Explain all the test cases you implemented. Also, describe any additional test cases you would like to implement but were unable to. Discuss the power of your scanner based on your experiments.
Installation instructions for your scanner must also be part of the report.

Submit: The code for your scanner and a script that executes the scanner with various parameters.

What to Submit

Also, drop off a hard copy of the report into the CSE 4481 assignment dropoff box located on the first floor of CSEB. The hard copy will be the one to be marked. The electronic copy will be used for record keeping.

If the source code you would like to submit resides in the Attack Lab, place it in the following directory in the `Submit` virtual machine: `C:/CSE4481/<cse username>/PhaseX`. You will probably need to create this directory.