BLG460E - Secure Programming

Take-Home Exam 2 : Canonicalization Issues

Due Date: Mar 13, 2019, 23.59 PM

This assignment aims to provide hands-on experience for students on canonicalization issues. You are required to research the canonicalization issues, analyze the code examples, try to bypass the security checks and do the necessary modifications on the code segments to defeat such issues as much as you can. This assignment also requires students to check the issues regarding with different operating systems (i.e., GNU/Linux, Windows). You are required to write a report which explains the weaknesses with the code examples, sample attack strings which can bypass the checks, explanations of your modifications and further discussions on your concerns. You should submit your report along with the code examples.

Part1: Directory Traversal Issues

A program is needed which requires all the files to be processed to be residing only in the /home directory. Assume that a friend of you who says "Security last" wrote the following code to validate a path name whether it is in the */home* directory or not.

```
#include <stdio.h>
#include <stdlib.h>

void processFile(const char *path){

  printf("Process file executes\n");
  char samplecommandtoexecute[100]={"ls -la "};
  strcat(samplecommandtoexecute,path);
  int status = system(samplecommandtoexecute);
  return;

}

void main(int argc, char *argv[]){

  const char *safepath = "/home";
  size_t spl = strlen(safepath);

  char *fn = argv[1];

  if(!strncmp(fn,safepath, spl)){
    processFile(fn);
  }
  else {
    printf("Path specified is not valid!\n");
    return;
  }

  return;
}
```

Analyze the code and determine sample file paths which will let an attacker to skip the validation performed in the code. Modify the code to make it resilient against directory traversal issues.

In the report, you should give sample attack scenarios with explanations. You can put screenshots if necessary. You should also explain how you mitigate the problems in the modified code.

Part2: File Name Handling

Consider the code segment below. Analyze the code and figure out whether the given code segment will able to defeat any canonicalization issues regarding with the file name handling. Do the necessary corrections on the given code and verify its functionality against file name handling based canonicalization issues.

You should try the code both in Windows and in Linux. List and explain all the attack scenarios in the report. Argue whether your code will still be functional if the operating system of the host changes. For this part of the assignment, create the text file *mysecretfile.txt* and apply your tests accordingly.

Hint: case sensitivity, 8.3 file naming convention, file path issues ([..], [.\], full path name, using Windows Environment Path Variables, accessing local computer using UNC share).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

void readFile(const char *filename) {
  printf("Reading the file : %s\n", filename);
  FILE *filepointer;
  int c;
  filepointer = fopen(filename, "r");

  if (filepointer == NULL) {
    printf("Error: %s\n", strerror(errno));
    return;
  }

  while ((c = fgetc(filepointer)) != EOF)
  printf("%c", (char)c);

  fclose(filepointer);
  return;
}

void main(int argc, char *argv[]) {
  const char protectedfile[] = "mysecretfile.txt";

  char *fn = argv[1];

  if (strcmp(fn, protectedfile) != 0) {
    readFile(fn);
  }
  else {
    printf("Access to the file specified is not permitted!\n");
    return;
  }

  return;
}
```

Requirements

You should upload two modified code files and a report that explains all attack scenarios and the codes.

Do not put any photograph of the screen or any handwritten note in the report. You can use screenshots.

You should show and explain how you managed to achieve the attacks. Do not just tell the story, prove it by screenshots and meaningful explanations.