BLG460E - Secure Programming

Assignment 2 : Canonicalization Issues

Due Date: 05.04.2022, 23:00 PM

1 Introduction

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 "naive" 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.

2 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 decided not to take the Secure Programming course 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.

3 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. In addition to these, argue whether your code will still be functional if the operating system of the host changes. (Hint: case sentivity, 8.3 file naming convention, etc.). For this part of the assignment, create two text files *naivepasswordfile.txt* and *naivecreditcarddetails.txt* and apply your tests accordingly.

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

void processFile(const char *filename) {

  printf("Process file executes\n");
  FILE *filepointer;
  int c;
  filepointer = fopen(filename, "r");

  if (filepointer == NULL) {
    printf("Could not open the file\n");
    return;
  }

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

  fclose(filepointer);
  return;
}

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

  const char *protectedfiles[] = { "naivepasswordfile.txt","naivecreditcarddetails.txt"
    };

  char *fn = argv[1];

  if (strcmp(fn, protectedfiles[0]) != 0 && strcmp(fn, protectedfiles[1]) != 0) {
    processFile(fn);

  }
  else {
    printf("Access to the file specified is not permitted!\n");
    return;
  }

  return;
}
```

Important Notes:

- Date and the time of the submission will not be changed. Please be aware of the deadline.
- Interactions among individuals are prohibited.
- You should upload two modified code files and a report that explains all attack scenarios and the codes.
- Please use screenshots and do not put any photograph of the screen or any handwritten note in the report.
- You should show and explain how you managed to achieve the attacks. You should prove your work by screenshots and meaningful explanations. Do not just tell the story please.
- Send an email to **sayinays@itu.edu.tr** for your questions.