מטלת מנחה (ממ"ן) 12

הקורס: תכנות מערכות דפנסיבי - 20937

חומר הלימוד למטלה: יחידה 3 – חולשות אבטחה בשפת C++

מספר השאלות: 2 **משקל המטלה:** 4

סמסטר: 2025ג **מועד אחרון להגשה:** 20.8.2025

שימו לב:
את המטלה יש להגיש באמצעות מערכת המטלות המקוונת באתר הבית של הקורס **בלבד**
את התשובה יש להגיש בקבצים בהתאם למפורט בשאלות.
את ניתוחי החולשות יש לכתוב בהתאם לפורמט המופיע בספר הלימוד בפרק 4, תחת Documentation and analysis, הוא נמצא גם בלשונית יחידה 3 באתר הקורס.

שאלה 1 (20%)

צפו בסרטון בקישור הבא, המתאר מנגנון אבטחה במעבדי ARM, וענו על השאלות הבאות בעברית:

https://www.youtube.com/watch?v=VSXljcPDMic

א. הסבירו את מנגנון האבטחה המתואר בסרטון.

ב. האם המנגנון דומה לאחד המנגנונים שנלמדו ביחידה 3, ואם כן לאיזה?

ג. מה צריך המתכנת או המשתמש לעשות כדי להפעיל את המנגנון במערכת הפעלה Debian?

הגשה: מסמך בפורמט pdf או word, אורך התשובה צריך להיות כעמוד אחד לכל השאלה.

שאלה 2 (80%)

לפניכם תוכנה המדפיסה ל- stdout את הקלט שלה. הקוד זמין גם בקובץ mmn02-q2.cpp באתר הקורס.

1. קמפלו את הקוד, הריצו אותו והבינו כיצד הוא עובד.
2. מצאו חולשה והשתמשו בה על מנת לקרוא לפונקציה unreachable. יש להציג קלט מתאים.
3. תקנו את הקוד כך שההתקפה לא תעבוד.
4. כתבו מסמך מחקר עם הסבר על החולשה, ההתקפה וההגנה.

```
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <string>
```

```
////////////////////////////////////////////////////////////////////////////
///////////
//
// -- IMPORTANT! --
//
// for this exercise to run correctly do the following:
//
// a. Disable ASLR:
//            VS:    Configuration Properties->Linker->Advanced ->
"Randomized Base Address"
//            g++: disabled by default in gdb
//
// b. Set the target binary to x86
//            VS: Build -> Configuration Manager -> Active solution platform
-> X86
//            g++: -m32 flag   (if fails try: sudo apt-get install gcc-
multilib g++-multilib)
//
// c. Debug mode:
//            VS: Build -> Configuration Manager -> Active solution
configuration -> Debug
//            g++: -g3 flag   (maximal debug information)
//
////////////////////////////////////////////////////////////////////////////
///////////


#define PROGRAM_NAME "echoutil"
#define VERSION "1.0"

#define VERY_SECRET_PASSWORD "Cowabunga!"

class Handler
{
   virtual void unreachable()
   {
      printf("%s", VERY_SECRET_PASSWORD);
      exit(0);
   }

   virtual void helper(const char *str)
   {
      std::string s = "0" + std::string(str);
      unsigned int x = std::stoul(s, nullptr, 16);
      printf("%c", x);
   }

public:

   void interpret(const char* str)
   {
      helper(str);
   }
};
```

```c
void usage(int status)
{
   fputs("Echo the STRING(s) to standard output\n"
      "\n"
      "\t-n   do not output the trailing newline\n"
      "\t-e   enable interpretation of backslash escapes\n"
      "\n"
      "\tIf - e is in effect, the following sequences are recognized : \n"
      "\t\t\\xHH     byte with hexadecimal value HH(1 to 2 digits)\n"
      , stdout);

   exit(status);
}

void handle_escape(const char* str)
{
   struct
   {
      char buffer[16] = { 0 };
      Handler h;
   } l;

   // copy only the characters after the escape char
   const char* s = str;
   char* p = l.buffer;
   s++;
   while (*s)
      *p++ = *s++;

   // handle different options
   switch (l.buffer[0])
   {
   case 'x':
      l.h.interpret(l.buffer);
      break;

   default:
      fputs(str, stdout);
   }
}

char* dupenv(const char* varname)
{
#if defined(_WIN32)

   char* buff = NULL;
   size_t cnt;
   if (_dupenv_s(&buff, &cnt, varname) != 0)
      return NULL;
   return buff;

#elif defined(__linux__)
```

```
   const char* s = getenv(varname);
   if (!s)
      return NULL;
   return strdup(s);

#endif
}
int main(int argc, char** argv)
{
   bool display_return = true;
   bool do_escape = false;

   char* env = dupenv("ECHOUTIL_OPT_ON");
   bool allow_options = env != NULL;
   free(env);

   if (allow_options && argc == 2)
   {
      if (strcmp(argv[1], "--help") == 0)
         usage(EXIT_SUCCESS);

      if (strcmp(argv[1], "--version") == 0)
      {
         fprintf(stdout, "%s version %s\n", PROGRAM_NAME, VERSION);
         exit(EXIT_SUCCESS);
      }
   }

   --argc;
   ++argv;

   if (allow_options)
   {
      while (argc > 0 && *argv[0] == '-')
      {
         const char* temp = argv[0] + 1;
         size_t i;
         for (i = 0; temp[i]; i++)
            switch (temp[i])
            {
            case 'e': case 'n':
               break;
            default:
               goto just_echo;
            }
         if (i == 0)
            goto just_echo;

         // options are valid
         while (*temp)
            switch (*temp++)
            {
            case 'e':
```

```
                do_escape = true;
                break;

            case 'n':
                display_return = false;
                break;
            }

        argc--;
        argv++;
    }
}

just_echo:

    while (argc > 0)
    {
        const char* s = argv[0];

        if(do_escape && s[0] == '\\')
            handle_escape(s);
        else
            fputs(argv[0], stdout);

        argc--;
        argv++;
        if (argc > 0)
            putchar(' ');
    }

    if (display_return)
        putchar('\n');

    exit(EXIT_SUCCESS);
}
```

דגשים:

א. עבור תרגיל זה יש **לבטל את מנגנון ה- ASLR** ולבנות את הקוד ב- **32 סיביות (x86)**
ב. קמפלו את הקוד בקונפיגורצית **debug** ועשו שימוש בדבאגר (מספיק שההתקפה תעבוד עם דבאגר).
ג. עבודתכם תיבדק במ"ה לינוקס (Ubuntu), באמצעות g++ ולכן מומלץ לעבוד עם סביבה זו.

הגשה: קובץ עם הקוד המתוקן ומסמך מחקר בפורמט pdf או word. רצוי לחבר אותם ביחד לקובץ zip.

בהצלחה!