XII МЕЖРЕГИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

2017 год

ВАРИАНТ 2

Задача 1. Хеш-значения

Политика безопасности ОС не позволяет задавать для учетных записей пользователей пароли, совпадающие с их именами. Для этого перед добавлением нового пользователя в базу вызывается функция `CheckUser()`. При ее успешном выполнении (возвращаемое значение = 0) в базу добавляется новая запись, содержащая имя учетной записи пользователя и хеш-значение пароля, полученное с помощью функции `Hash()`.

Си

```
int CheckUser(char* username,
              char* password)
{
  for (int i=0; i<strlen(username); i++)
  {
    if ( (username[i] >= 0x30 &&
          username[i] <= 0x39) ||
         (username[i] >= 0x41 &&
          username[i] <= 0x5A) ||
         (username[i] >= 0x61 &&
          username[i] <= 0x7A)
       )
      continue;
    else
      return 1;
  }
  for(int i=0; i<strlen(password); i++)
  {
    if (password[i] >= 0x23 &&
        password[i] <= 0x7D)
      continue;
    else
      return 1;
  }
  if (stricmp(username, password) != 0)
    return 0;
  else
    return 1;
}
```

Паскаль

```
function CheckUser(
  var username: string;
  password: string) : integer;
var i:integer;
begin
  for i:=1 to Length(username) do
  begin
    if( ((username[i] >= #48) and
         (username[i] <= #57)) or
        ((username[i] >= #65) and
         (username[i] <= #90)) or
        ((username[i] >= #97) and
         (username[i] <= #122))
      )
    then continue
    else
    begin
      Result := 1;
      Exit;
    end;
  end;
  for i:=1 to Length(password) do
    if( (password[i] >= #35) and
        (password[i] <= #125) )
    then continue
    else
    begin
      Result := 1;
      Exit;
    end;
  if( UpperCase(username)<>
      UpperCase(password) )
  then Result :=0
  else Result :=1;
end;
```

```
const char letters[] =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP
QRSTUVWXYZ01234567890";

char* Hash(char* str)
{
```

```
function Hash(str:string[255]):
string[20];
var Nstr,hsize:integer;
var i,p_hash,p_pow,p:word;
var buf:string[255];
var str_size, hash_size:integer;
```

```
   const int p = 19;
   int Nstr = 32;
   const int Hsize = Nstr / 2;
   unsigned short int hash = 0, p_pow = 1;
   char *buf = new char[Nstr + 1];
   int StrSize = 0;
   int HashSize = 0;
   char *res = new char[Hsize + 1];

   while (str[StrSize] != '\0')
   {
     if (HashSize >= Nstr)
       HashSize = Nstr - 1;
     buf[HashSize] = str[StrSize];
     StrSize++;
     HashSize++;
   }
   for (int i = HashSize; i < Nstr; i++)
   {
     buf[i] = 'A' + (i - HashSize);
   }
   buf[Nstr] = '\0';
   for (int i = 0; i < Nstr; i += 2)
   {
     hash += (buf[i + 1] - 'A' + 1) * p_pow;
     p_pow *= p;
     hash += (buf[i] - 'A' + 1) * p_pow;
     p_pow *= p;
     res[i / 2] = letters[hash %
strlen(letters)];
   }
   res[Hsize] = '\0';
   return res;
}
```

```
var letters:string[100];
var res:string[32];
begin
  letters:='abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890';
  p:=19;
  Nstr:=32;
  Hsize:=Nstr div 2;
  p_hash:=0;
  p_pow:=1;
  str_size:=1;
  hash_size:=1;
  while ( str_size<=length(str) ) do
  begin
    if (hash_size > Nstr) then
    begin
      delete(buf,hash_size,1);
      hash_size:=Nstr;
    end;
    buf:=buf+str[str_size];
    str_size:=str_size+1;
    hash_size:=hash_size+1;
  end;
  for i:=hash_size to Nstr+1 do
    buf:= buf+chr(ord('A')+(i-
hash_size));
  i:=1;
  repeat
    p_hash:=p_hash+(ord(buf[i+1])-
          ord('A')+1)*p_pow;
    p_pow:=p_pow*p;
    p_hash:=p_hash+(ord(buf[i])-
          ord('A')+1)*p_pow;
    p_pow:=p_pow*p;
    res:=res+letters[p_hash mod
               length(letters)+1];
    i:=i+2;
  until i>Nstr;
  Hash:=res;
end;
```

Администратор периодически выполняет проверку базы пользователей и блокирует учетные записи, хеш-значения от имени которых совпадают с хеш-значениями их паролей. Приведите пример имени и пароля для учетной записи, которая удовлетворяет заданной политике безопасности, но будет заблокирована администратором в ходе проверки, и обоснуйте почему.

Задача 2. Секретное сообщение

В исполняемый файл PROG.EXE было внедрено секретное текстовое сообщение. При этом сам файл корректно выполняет все функции. Известно, что для того, чтобы отметить место внедрения информации, нарушитель использовал метку размером 1 байт:

Метка (1 байт)	Сообщение	Метка (1 байт)

Какое сообщение было внедрено в файл?

К задаче прилагается: исполняемый файл PROG.EXE.

Задача 3. Антивирус

Для выявления вредоносного кода некоторым антивирусом применяется только сигнатурный метод анализа, позволяющий выполнять поиск известных сигнатур в файле путем побайтового сравнения. Файл считается вредоносным при наличии в нем участка данных, точно совпадающего с одной из сигнатур. Реализация поиска сигнатур описана в функции `CheckFile()`, которая возвращает TRUE при отсутствии сигнатур в файле и FALSE в противном случае.

Си

```
/* ВХОДНЫЕ ПАРАМЕТРЫ:
 * FNAME     - имя анализируемого файла
 * SIGNATURE - сигнатура (массив байтов)
 * SIZE      - размер сигнатуры в байтах
 *
 * ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ:
 * TRUE      - файл не содержит сигнатуры
 * FALSE     - файл содержит хотя бы одну
 *             сигнатуру
 */
bool CheckFile(char* fname,
               char* signature,
               int size)
{
  FILE *fin=NULL;
  char *buf=NULL;
  int read, check_count=0;
  buf=new char[size];
  fin=fopen(fname, "rb");
  if (!fin)
    return false;
  while(!feof(fin))
  {
    read=fread(buf, 1, size, fin);
    if(read!=size)
      break;
    check_count=0;
    for (int j=0; j<size; j++)
    {
      if (buf[j]==signature[j])
           check_count++;
      else
           break;
    }
    if (check_count==size)
      return false;
  }
  return true;
}
```

Паскаль

```
/* ВХОДНЫЕ ПАРАМЕТРЫ:
 * FNAME     - имя анализируемого файла
 * SIGNATURE - сигнатура (массив байтов)
 * SIZE      - размер сигнатуры в байтах
 *
 * ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ:
 * TRUE      - файл не содержит сигнатуры
 * FALSE     - файл содержит хотя бы одну
 *             сигнатуру
 */
function CheckFile (fname,
          signature: string;
          size: integer): boolean;
var
  fin: file;
  ch: char;
  count_read, check_count, i: integer;
  buf: string;
begin
  assign(fin, fname);
  reset(fin);
  while true do
  begin
    buf:='';
    count_read:=0;
    for i:=1 to size do
    begin
      if (not eof(fin)) then
      begin
        read(fin, ch);
        buf:=buf+ch;
        count_read:=count_read+1;
      end;
    end;
    if count_read <> size then
      break;
    check_count:=0;
    for i:=1 to size do
    begin
      if(buf[i]=signature[i]) then
        check_count:=check_count+1
      else
        break;
    end;
    if check_count=size then
    begin
      CheckFile:=false;
      Exit;
    end;
  end;
  close(fin);
  CheckFile:=true;
end;
```

База сигнатур содержит две записи:
`0x313132`
`0x3131`

Проверка файла осуществляется последовательным вызовом функции `CheckFile()` для каждой сигнатуры из базы.

Какое количество файлов размером 7 байт, состоящих только из цифр от 0 (код 0x30) до 9 (код 0x39) включительно, может содержать хотя бы одну из указанных сигнатур хотя бы один раз, но успешно проходить проверку имеющимся антивирусом? Ответ обоснуйте.

Задача 4. Архив

Школьник скачал с некоторого Интернет-ресурса архив PROGS.RAR, который, согласно приведенному на сайте описанию, содержит пакет простых утилит (каждой утилите соответствует ровно один исполняемый файл формата .EXE). После распаковки архива школьник обнаружил, что часть файлов из архива зашифрована методом «двоичного гаммирования», т.е. путем выполнения операции «побитового исключающего ИЛИ» между байтами исходного файла и байтами, полученными циклическим повторением последовательности некоторого ключа.

Зашифрованные файлы не запускаются, а при попытке запуска незашифрованных файлов, некоторые из них блокируются антивирусом из-за наличия в них подозрительной сигнатуры «`0x0B0A0C0F`».

Помогите школьнику получить из архива максимальное количество программ, которыми он сможет воспользоваться, не отключая антивирус.

К задаче прилагается: архив PROGS.RAR, скаченный школьником с Интернет-ресурса.

Задача 5. Exploit

Имеется программа на языке С:

```
void main()
{
    char a[10];
    printf("Введите строку:");
    gets(a);
    printf("Вы ввели %d символов", strlen(a));
}
```

При компиляции получился следующий программный код:

Адрес памяти	Байты в памяти	Их содержание
230D123D	C2 E2 E5 E4 E8 F2 E5 20 F1 F2 F0 EE EA F3 00	Строка «*Введите строку*» с нулевым байтом в конце
230D124C	C2 FB 20 E2 E2 E5 EB E8 20 25 64 20 F1 E8 EC E2 EE EB EE E2 00	Строка «*Вы ввели %d символов*» с нулевым байтом в конце
230D1261	00 00 00 00 00 00 00 00 00 00	массив *a* (10 элементов типа char)
230D126B	68 3D 12 0D 23	Машинная команда, передающая адрес строки «*Введите строку*» (230D123D) в подпрограмму *printf* как параметр

230D1270	FF 15 14 72 0D 23	Машинная команда, вызывающая подпрограмму *printf*, расположенную по адресу 230D7214
230D1276	83 C4 04	Вспомогательная машинная команда, выполняемая после возвращения из подпрограммы *printf*, имеющей один параметр
230D1279	68 61 12 0D 23	Машинная команда, передающая параметр *a*, расположенный по адресу 230D1261, в подпрограмму *gets*
230D127E	FF 15 10 72 0D 23	Машинная команда, вызывающая подпрограмму *gets*, расположенную по адресу 230D7210
230D1284	83 C4 04	Вспомогательная машинная команда, выполняемая после возвращения из подпрограммы *gets*, имеющей один параметр
230D1287	68 61 12 0D 23	Машинная команда, передающая параметр *a*, расположенный по адресу 230D1261, в подпрограмму *strlen*
230D128C	E8 D5 FD FF FF	Машинная команда, вызывающая подпрограмму *strlen*
230D1291	83 C4 04	Вспомогательная машинная команда, выполняемая после возвращения из подпрограммы *strlen*, имеющей один параметр
230D1294	50	Машинная команда, передающая возвращаемое значение функции *strlen* в подпрограмму *printf* как параметр
230D1295	68 4C 12 0D 23	Машинная команда, передающая адрес строки «*Вы ввели %d символов*» (230D124C) в подпрограмму *printf* как параметр
230D129A	FF 15 14 72 0D 23	Машинная команда, вызывающая подпрограмму *printf*, расположенную по адресу 230D7214
230D12A0	83 C4 08	Вспомогательная машинная команда, выполняемая после возвращения из подпрограммы *printf*, имеющей два параметра
230D12A3	C3	Машинная команда, завершающая программу

Применяемая реализация подпрограммы *gets* принимает на вход любые данные без ограничений, в том числе специальные и непечатаемые символы (нулевой байт, символ конца строки и т.п.). Окончанием входного потока считается комбинация байтов «0x0D0A».

Укажите входную последовательность байтов (в шестнадцатеричном формате), после ввода которой программа выведет строку «*Информационная культура*».