ВАРИАНТ 1 **9-10 класс**

РЕШЕНИЯ ЗАДАЧ XII МЕЖРЕГИОНАЛЬНОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

2017 год

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 size:integer;
```

```
  const int p=31;
  const int Nstr=32;
  const int Hsize=Nstr/2;
  unsigned short int hash=0, p_pow=1;
  char buf[Nstr+1]={0};
  int size=0;
  char *res=new char[Hsize+1];
  while (str[size]!='\0' && size<Nstr)
  {
    buf[size]=str[size];
    size++;
  }

  for (int i=size; i<Nstr; i++)
  {
    buf[i]='A'+(i-size);
  }
  for (int i=0; i<Nstr; i+=2)
  {
    hash+=(buf[i]-'a'+1)*p_pow;
    res[i/2]=letters[hash%strlen(letters)];
    p_pow*=p;
  }
  res[Hsize]='\0';
  return res;
}
```

```
var letters:string[100];
var res:string[32];
begin
  letters:='abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890';
  p:=31;
  Nstr:=32;
  Hsize:=Nstr div 2;
  p_hash:=0;
  p_pow:=1;
  size:=1;
  while ( (size<=length(str)) and
          (size<=Nstr)) do
  begin
    buf:=buf+str[size];
    size:=size+1;
  end;
  for i:=size to Nstr+1 do
    buf:= buf+chr(ord('A')+(i-size));
  i:=1;
  repeat
    p_hash:=p_hash+(ord(buf[i])-
          ord('a')+1)*p_pow;
    res:=res+letters[p_hash mod
              length(letters)+1];
    p_pow:=p_pow*p;
    i:=i+2;
  until i>Nstr;
  Hash:=res;
end;
```

В базе пользователей присутствуют следующие записи:

```
admin     rUZxHUUfw9oJSFNm
user      vYhkB2klurVYYPtq
operator  pS8HIUqmrJ60ZOrk
manager   ng3JfGZ0TQzmCtS5
root      sQUsDRwnsAf90HN0
```

В ходе проверки администратор выявил случаи возможного нарушения политики безопасности.

Определите:

а) пользователей, для которых были обнаружены нарушения;

б) причины возникновения нарушений.

Решение:

Проанализировав функцию `CheckUser`(), можно сделать вывод о том, что она действительно не позволяет создавать пользователей, пароль которых совпадает со значением имени его учетной записи. Следовательно, можно сделать вывод, что причина возникновения нарушений связана с ошибкой в вычислении хеш-значения пароля. Действительно, из анализа кода

```
1.for (int i = 0; i < Nstr; i+=2)
2.{
3.   hash += (buf[i] - 'a' + 1) * p_pow;
4.   res[i/2] = letters[hash % strlen(letters)];
5.   p_pow *= p;
6.}
```

видно, что в вычислении результирующего хеш-значения (переменная *res*) участвуют только четные (начиная с 0) символы пароля (i = 0 … i+=2 … buf[i] …).

Таким образом, вычислив хеш-значения пароля для имен учетных записей всех пользователей, получаем, что для пользователей operator и manager произошло нарушение.

Ответ: Нарушение произошло для пользователей operator и manager из-за возникновения коллизии, связанной с тем, что в вычислении хеш-значения пароля участвуют только четные (начиная с 0) символы.

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 1-байтную метку, после которой идет сообщение размером до 10 байт:

Метка (1 байт)	Сообщение (10 байт)

Какое сообщение было внедрено в файл?

К задаче прилагается: исполняемый файл PROG.EXE.

Решение:

Для однозначного определения внедренного сообщения необходимо в файле найти байт, который встречается в нем ровно 1 раз. Для оптимизации процесса поиска необходимо реализовать программу подсчета встречаемости байтов в файле. Единственный байт, который может выступать в роли метки, имеет значение 0xAD. Первые 10 байтов после метки — сообщение. После извлечения соответствующих байтов переводим их в символы по ASCII-таблице.

Ответ: document

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;
```

Какое количество файлов размером 7 байт, состоящих только из цифр от 0 (код 0x30) до 9 (код 0x39) включительно, может содержать трех-байтовую сигнатуру «0x313231» хотя бы один раз, но успешно проходить проверку антивирусом? Ответ обоснуйте.

Решение:

Из анализа функции `CheckFile()` видно, что антивирус анализирует файл, начиная с нулевого байта, блоками данных размером по 3 байта, без пересечений. Поэтому, если начало сигнатуры в одном блоке, а конец – в другом, то она не будет обнаружена.

Возможные варианты размещения сигнатуры в файлах размером 7 байт, содержащих допустимые ASCII-символы (где X – любая цифра от 0 до 9):

1) X12 1XX X – X12 121 X
2) XX1 21X X – 121 21X X
3) XXX X12 1 – X12 112 1 – 121 X12 1 – XX1 212 1 + 121 212 1 (т.к. этот файл исключен дважды в вариантах 121 X12 1 и XX1 212 1)

Для оптимизации процесса подсчета количества файлов можно написать программу, которая реализует корректный поиск сигнатур с использованием функции, полученной из `CheckFile()`, например, добавлением строки

```
fseek(fin, -(read - 1), SEEK_CUR);
```

в конец цикла while.

Разница между числом файлов, проходящих проверку антивирусами на основе функции `CheckFile()` и ее исправленной версии, – ответ к задаче.

Ответ: 29681

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 двух байтов ключа «0xB69D».

Зашифрованные файлы не запускаются, а при попытке запуска незашифрованных файлов, некоторые из них блокируются антивирусом из-за наличия в них подозрительной сигнатуры «0x0A0B0C0F».

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.

Решение:

На первом этапе необходимо выделить множество файлов, которые корректно запускаются: Clockres.exe; plink.exe; pslist.exe. Остальные файлы архива являются зашифрованными. Заметим, что у всех незашифрованных файлов первые два байта одинаковые: «0x4D 0x5A», что соответствует началу заголовка любого файла формата .EXE. Следовательно, можно предположить, что остальные файлы после их расшифрования должны содержать аналогичные 2 первых байта. Т.к. для шифрования применялся метод «двоичного гаммирования» с длиной ключа 2 байта, то первых двух байтов будет достаточно для определения самого ключа. Для этого необходимо выполнить операцию «побитового исключающего ИЛИ» между двумя первыми байтами любого из файлов Clockres.exe; plink.exe;

pslist.exe и двумя первыми байтам любого зашифрованного файла (например puttygen_enc.exe): `4D 5A ^ FB C7 = B6 9D`.

Таким образом, в качестве ключа использовалась последовательность – B6 9D. Далее необходимо применить операцию «побитового исключающего ИЛИ» между всеми байтами зашифрованных файлов и байтами, полученными циклическим повторением последовательности байтов ключа, т.е. B6 9D B6 9D … B6 9D, для чего необходимо написать программное средство, позволяющее выполнить данную операцию в автоматическом режиме. Результатом работы данного программного средства являются файлы формата .EXE, которые успешно запускаются.

На втором этапе необходимо определить файлы, содержащие сигнатуру «`0x0A 0x0B 0x0C 0x0F`»: plink.exe, whois.exe. Следовательно, остальными программами школьник сможет воспользоваться, не отключая антивирус.

Ответ: все файлы, **кроме** plink.exe и whois.exe.

Задача 5. Контроль версий

В системе развернута инкрементная система контроля версий, хранящая исходные значения контролируемых файлов и их изменения в виде контрольных точек, по которым возможно восстановить текущее содержимое файлов.

Для файла `config.txt` в системе сохранено 5 контрольных точек:

1) `35 57 38 64 39 6F 3A 77 3B 73 3C 37`
2) `D3 54 D4 52 D5 55 D6 45 D7 22 D8 00`
3) `DA 35`
4) `D8 0D D9 0A DA 75 DB 73 DC 65 DD 72 DE 3D DF 22 E0 72 E1 6F E2 6F E3 74 E4 22`
5) `96 38 98 32 C2 38 C4 31`

Содержимое файла после 3-й контрольной точки приведено (`config.txt.backup.3`). Восстановите содержимое файла после 5-й контрольной точки.

<u>К задаче прилагается:</u> исходный файл CONFIG.TXT и файл после третьей контрольной точки CONFIG.TXT.BACKUP.3.

Решение:

Проанализировав содержимое файла после третьей контрольной точки, можно определить формат хранения изменений:

*номер_байта*1 *новое_значение*1 *номер_байта*2 *новое_значение*2 …

Таким образом, каждая контрольная точка (КТ) отражает следующие изменения:

КТ1: значение поля *name* изменено на «Windows7».
КТ2: значение поля *installed* изменено на «TRUE»; обнулен последний байт.
КТ3: значение поля *gateway* изменено на «12.5.2.5».
КТ4: в конец файла добавлена строка «user = “root”».
КТ5: значение поля *address* изменено на «12.5.8.2», *gateway* – «12.5.8.1».

Ответ: Содержание файла после 5-ой контрольной точки:

```
encoding="windows-1251"
UID="2"
```

```
current="2"
name="Windows7"
createTime="347261"
numDisks="1"
disk0.node="ide0:0"
network="IPv4"
address="12.5.8.2"
subnet="255.255.255.0"
gateway="12.5.8.1"
installed="TRUE"
user=”root”
```