GFI LANguard 9

Scripting manual

By GFI Software

http://www.gfi.com
Email: info@gfi.com

Last updated: 4TH September 2009

Version: LANSS-SCP-EN-01.00.00

Table of contents

1. Introduction

GFI LANguard allows the user to write custom scripts that check for vulnerabilities. The scripts can be platform dependent or platform independent.

Platform dependent: Unix scripts run through SSH: The remote machine must be a Unix machine and allow remote connections via SSH. These scripts are run on the scanned machine.

Platform independent:

Visual Basic scripts: This manual provides extensive information on how to write, debug and setup Visual Basic custom vulnerability checks.

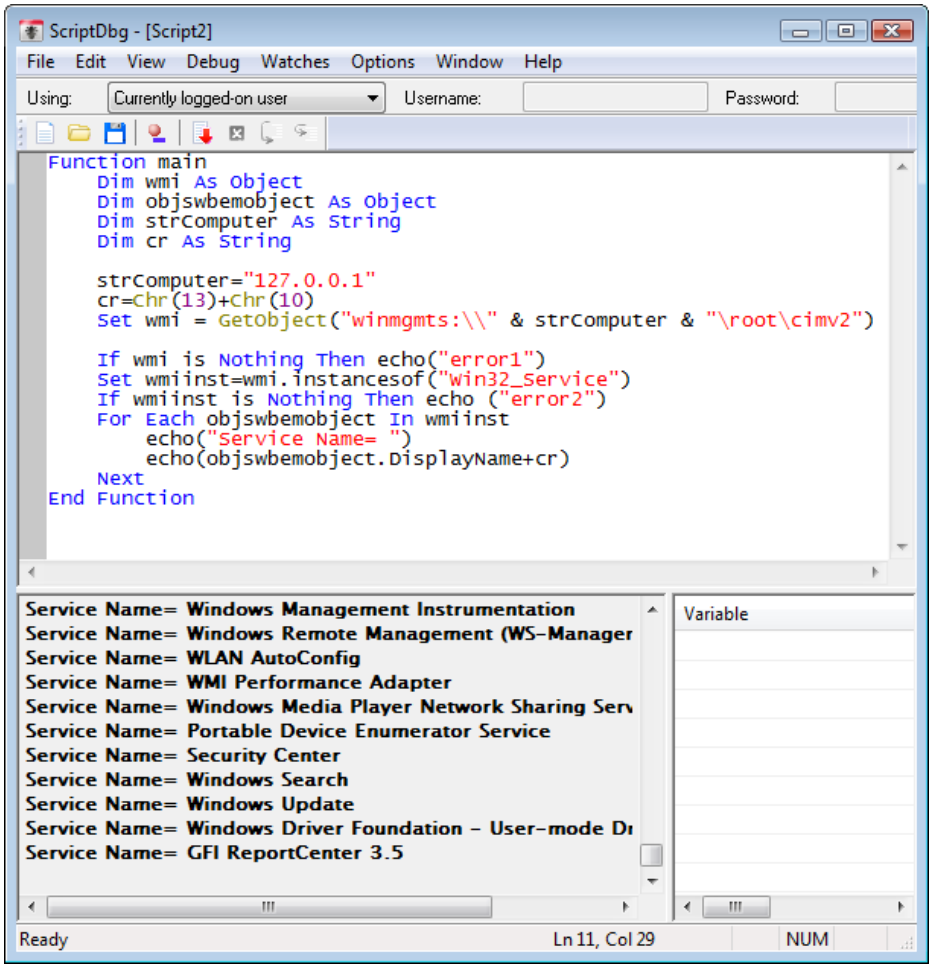
Python scripts: More details are available in the Python Scripting chapter in this manual.

1.1 Scripting Editor/Debugger

Although not necessary to develop in the editor, it is safer, faster and better to create both correct and actually working scripts. This will ensure proper functioning of the script when inserted into the vulnerability scans unit of the GFI LANguard security scanner tool. In the GFI LANguard editor/debugger you can run the script under development in the same conditions as if it was running under the security scanner in a real life situation. You have a controlled environment in which you can create, analyze, refine and investigate problems prior to putting the script into action on a live security scan. The GFI LANguard editor/debugger has all of the supporting functionality like breakpoints, step into, step over, as well as capabilities to specify parameters to be passed to the script to enable proper testing without the need to scan a machine over and over again.

WARNING: Running and debugging the scripts with the scanner is not recommended and should never be allowed since if there is anything wrong in the script, that script will never work and also the user will have no way to see why it is not working. The scanner will automatically ignore incorrect/non functioning scripts.

WARNING: If a script is not properly scripted and debugged there can be a high probability that it will go into infinite loop situations which can stall some parts of the scanner. Scripts are executed in order one after the other. There is no timeout capability. If a script needs 10 minutes or more to execute it will not be stopped. The scripting engine will wait for it to finish before moving to the next script.



Screenshot 1 Editor Debugger

1.1.1 Feature list present in the GFI LANguard debugger

Support for variable watches: Monitor in real time the changing values in variables used.

Step into/step over functionality to debug your scripts line by line while monitoring what is going on during execution of the script.

Syntax highlighting: Easier to program scripts and locate problems

Comprehensive error messaging: Indicates the type and location of the errors. Helps detecting variable type based errors.

Breakpoint: The debugger supports breakpoints which will shift into line by line debugging at a particular point of interest.

Capability to debug and run the script under alternative credentials during the script development and debugging process.

1.2 Important standards to follow

- In every script created, there must be a Function named "Main". GFI LANguard will look for and start from this function when executing any script.
- The return value of the "Main" function to the scripting engine is Boolean (true of false). This return value is specified by assigning the result value to a variable which has the same name of the function name (e.g. If the function is named MyFunction, the return value is specified as

MyFunction = true). This return value is generally specified at the end of the function, e.g.

```
Function Main
      'Show some text in the scanner activity window
      echo "Script has run successfully"
      'return the result
      Main = true
End Function
```

1.3 Native VBScript Functions supported in GFI LANguard

NOTE: ALL VBScript functions and scripting methods supported.

Other VBScript resources:

http://www.w3schools.com/VBScript/VBScript_ref_functions.asp

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vsgrpfeatures.asp

http://www.programmersheaven.com/zone1/mh50.htm

http://www.visualbasicforum.com/index.php?

1.4 Common mistakes and pitfalls

In VBScript there are two types of variables: simple types and objects. Simple Type variables are the ones of type integer, Boolean, string etc. Objects are complex items whose functionality is exposed by the automation objects interface.

It is important to declare the automation object types as **Object** before assigning them values.

NOTE: It is recommended you assign all variables/objects a type, for e.g.

```
Function Main
      'declare the object to be used
      Dim nameStr As String
      'assign a value to the variable
      nameStr = "This is a test assignment of text"
      'display the result in the scanner activity window
      of the assignment
      echo nameStr
      'return the result
      Main = true
End Function
```

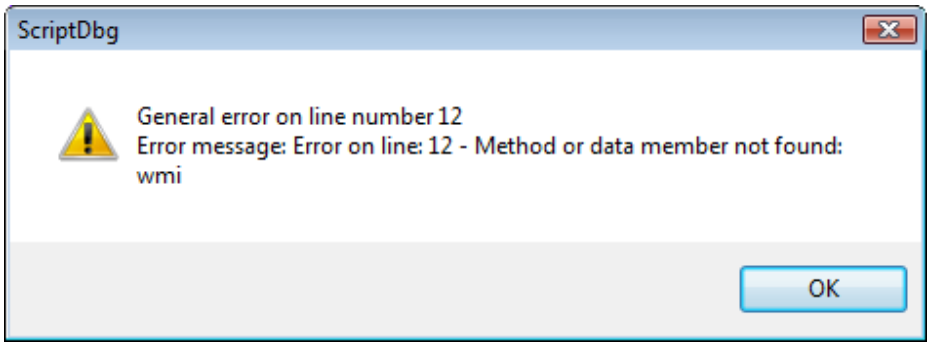
For a more advanced example, the script below will list which services are installed on the target machine (localhost = 127.0.0.1). Copy paste the following text in the script editor/debugger and run it (**F5**). In the debug window you will see the list of installed services on the local machine.

```
Function main
      Dim wmi As Object'declare the objects we will need
      to use
      Dim objswbemobject As Object
```

```
    Dim strComputer As String 'declare other variables
    we need.
    Dim cr As String
    strComputer = "127.0.0.1"
    cr = Chr(13) + Chr(10) 'carriage return
    Set wmi = GetObject("winmgmts:\\" & strComputer &
    "\root\cimv2") ' hook with the wmi object
    If wmi is Nothing Then echo ("error1") 'check that
    hook was successful
    Set    wmiinst=wmi.instancesof("Win32_Service")    '
    return the services instance of the wmi
    If wmiinst is Nothing Then echo ("error2") ' check
    to see that instance is available
    For Each objswbemobject In wmiinst ' loop true each
    instance
        echo("Service Name= ")
        echo(objswbemobject.DisplayName+cr)'display
        services
    Next
End Function
```

NOTE: If you try to use an automation object without declaring it as an object, it will cause the script to fail in execution. As an example consider the same piece of code but with a missing full declaration of the object variable **wmi**. The instant you try to run the script you will be presented with an error message as well as a clear indication of the line on which the error occurred:

```
Function main
    Dim wmi 'WARNING : missing "as object"
    Dim objswbemobject As Object
    …
    …
    …
End Function
```



Screenshot 2 Error produced as a result of not declaring an object before it is used.

1.5 Tips & Tricks

To display progress information in the Scanner activity window (the bottom window of the scanner tool, or the bottom left window of the editor/debugger) use the **echo** command e.g.

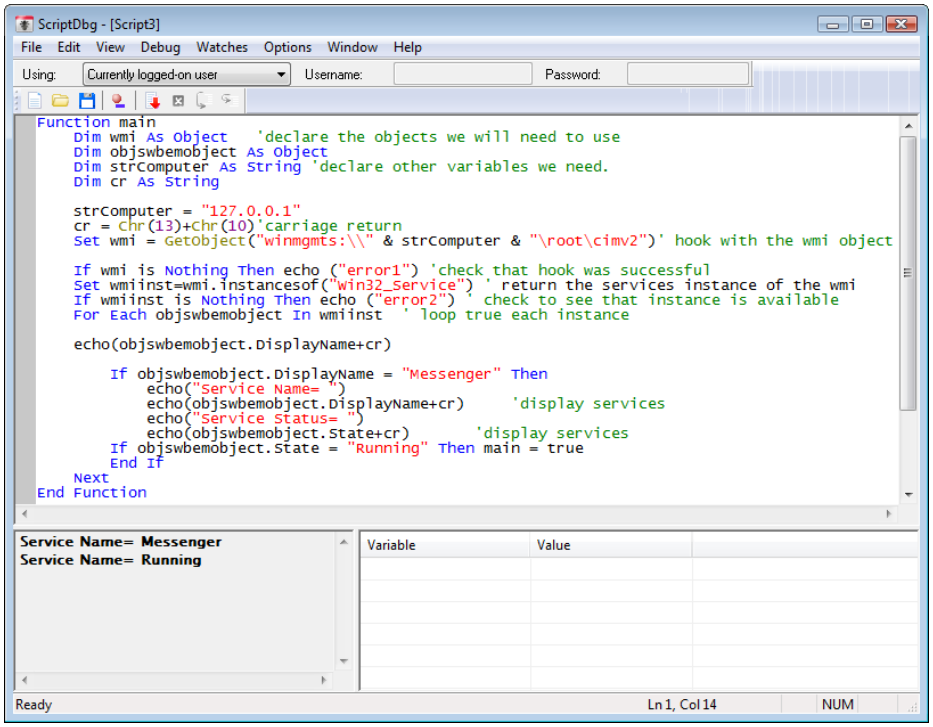
```
Function Main
      'Show some text in the scanner activity window
      echo "Script has run successfully"
      'return the result
      Main = true
End Function
```

1.6 Developing a script in the GFI LANguard debugger

In this section we will develop a script which will inform us whether the messenger service is running or not on the local machine, we will be developing it in the editor/debugger while demonstrating its various features.

```
Function main
      Dim wmi As Object'declare the objects we will need
      to use
      Dim objswbemobject As Object
      Dim strComputer As String 'declare other variables
      we need.
      Dim cr As String
      strComputer = "127.0.0.1"
      cr = Chr(13) + Chr(10) 'carriage return
      Set wmi = GetObject("winmgmts:\\" & strComputer &
      "\root\cimv2") ' hook with the wmi object
      If wmi is Nothing Then echo ("error1") 'check that
      hook was successful
      Set    wmiinst=wmi.instancesof("Win32_Service")    '
      return the services instance of the wmi
      If wmiinst is Nothing Then echo ("error2") ' check
      to see that instance is available
      For Each objswbemobject In wmiinst ' loop true each
      instance
            If  objswbemobject.DisplayName  =  "Messenger"
            Then
            echo("Service Name= ")
            echo(objswbemobject.DisplayName+cr)'display
            services
            echo("Service Status= ")
            echo(objswbemobject.State+cr)'display
            services
            If objswbemobject.State = "Running" Then main
            = true
            If objswbemobject.State = "Running" Then main
            = true
```

```
            End If
        Next
End Function
```



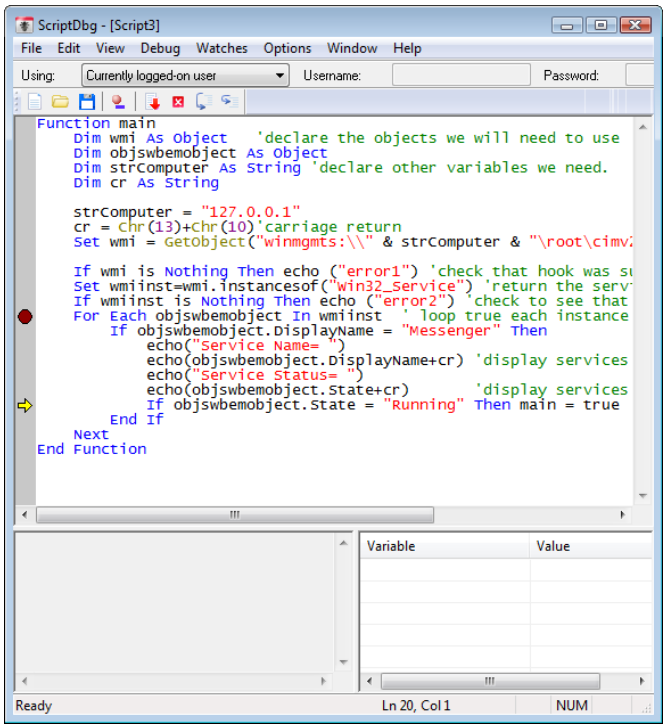
Screenshot 3 Script syntax highlighting.

1.6.1 Running a script

Once you entered your script and want to try it out use the **F5** button or use the pull down menu **Debug ► Go.**

1.6.2 Debugging breakpoints / Step In functionality

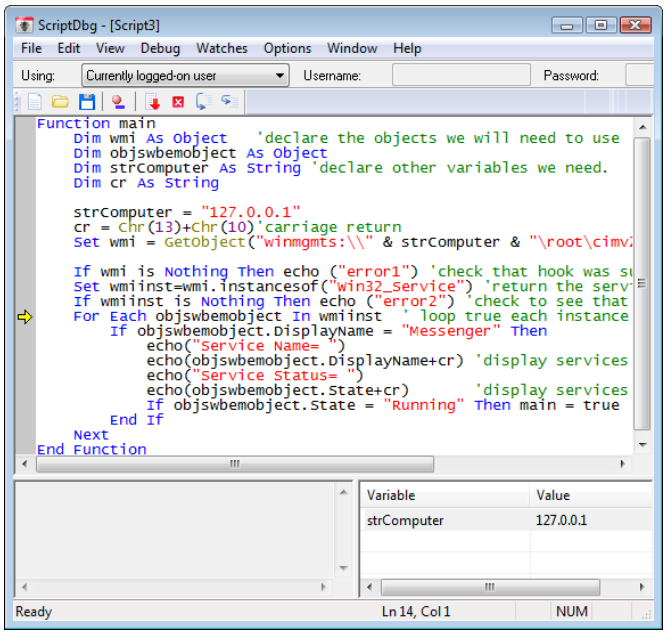
These editor/debugger features allow you to pause the execution when the interpreter gets to a certain line in the code and allow you to continue execution step by step under your supervision. For example, you could set a break point to execute when the variable containing the display name of the service is “Messenger”. To do this in the example script, you would go to line 17 (“echo("Service Name= ")”), and put the break point (Press **F9**) just under the “if” statement. During execution use the **F10** key (step in) to execute the remaining code line by line.



Screenshot 4 Breakpoints & line by line execution of the script

1.6.3 Monitoring the values inside variables

To monitor the values contained in variables you have to add a watch for that variable. For example if you want to monitor the contents of the variable named “myvalue” double Click top free line of the bottom right area of the script editor / debugger and add the name of the value you want to monitor.

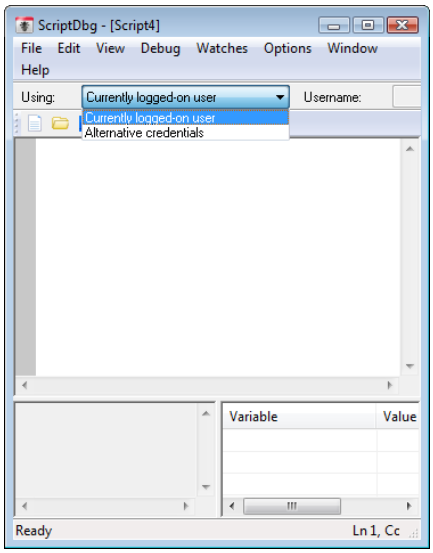


Screenshot 5 Watches

1.6.4 Debugging under alternative user sessions

To test the script under alternative credentials:

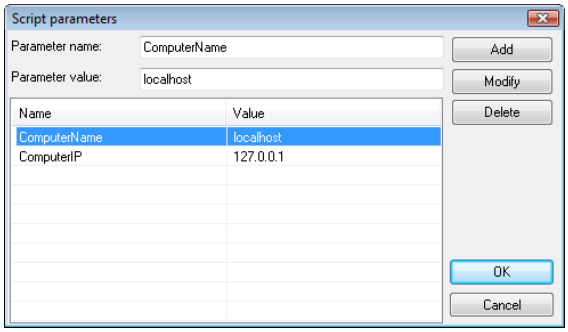
Lanuch **GFI LANguard Script Debugger** ► select the **Using** drop down list ► Select **Alternative Credentials**.



Screenshot 6 Alternative credentials

1.6.5 Sending Parameters to the script

GFI LANguard scanner tool passes parameters to the scripts when executed for e.g. the computer name and computer IP of the target machine being scanned for vulnerabilities. To be able to debug your scripts you may want to test with various types of values for these parameters. You can specify alternative values for these parameters from **Options ► Parameters.**

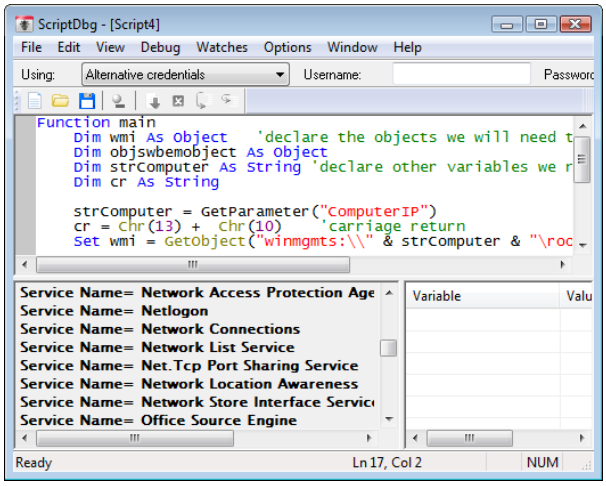


Screenshot 7 Parameters Dialog

In order to gain access to these parameters in scripts, one has to use a special GFI LANguard function called **GetParameter** and pass it the name of the parameter you want, for e.g.:

```
Function main
      Dim wmi As Object 'declare the objects we will need
      to use
      Dim objswbemobject As Object
      Dim strComputer As String 'declare other variables
      we need.
      Dim cr As String
      strComputer = GetParameter("ComputerIP")
      cr = Chr(13) + Chr(10) 'carage return
      Set wmi = GetObject("winmgmts:\\" & strComputer &
      "\root\cimv2") ' hook with the wmi object
      If wmi is Nothing Then echo ("error1") 'check that
      hook was successful
```

```
        Set    wmiinst=wmi.instancesof("Win32_Service")    '
        return the services instance of the wmi
        If wmiinst is Nothing Then echo ("error2") ' check
        to see that instance is available
        For Each objswbemobject In wmiinst ' loop true each
        instance
              echo("Service Name= ")
              echo(objswbemobject.DisplayName+cr)'display
              services
        Next
End Function
```



Screenshot 8 GetParameter function.

1.7 Adding a new scripting based vulnerability check to the scanner tool

In this example we will demonstrate how to create a new vulnerability check which will run a script to check for that vulnerability. The script that we will use simply displays the text **Script ran successfully** in the Scanner Activity Window, and will indicate to GFI LANguard that vulnerability has been detected and should be reported to the administrator.

To achieve this you have to:

Step 1: Create a script which checks for the vulnerability (as described in the previous section)

Step 2: Create a new vulnerability check in the UI which will run the above script.

1.7.1 Step 1: Creating the script

1. Launch the **GFI LANguard Script Debugger** from **Start ► Programs ► GFI LANguard ► Script Debugger**

2. **File ► New**

3. Paste the following text in the debugger:

```
Function Main
      echo "Script has run successfully"
      Main = true
```

```
End Function
```

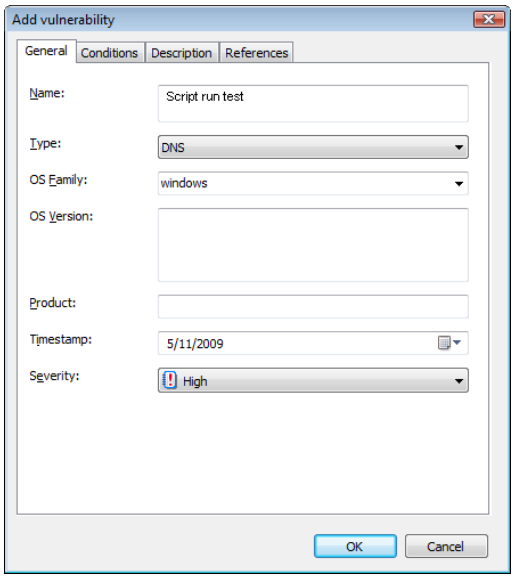
4. Save the file to a directory of your choice "c:\myscript.vbs"

1.7.2 Step 2: Creating the new vulnerability check

1. Launch the GFI LANguard Main Menu ►Goto **Configure ► Scanning Profiles Editor ► New scanning profile** or select the scanning profile to edit

2. Select **Vulnerability Assessment Options tab ►Vulnerabilities ► Vulnerabilities type**

3. Click **Miscellaneous type** and select **Add** button which will bring up the new vulnerability check dialog.



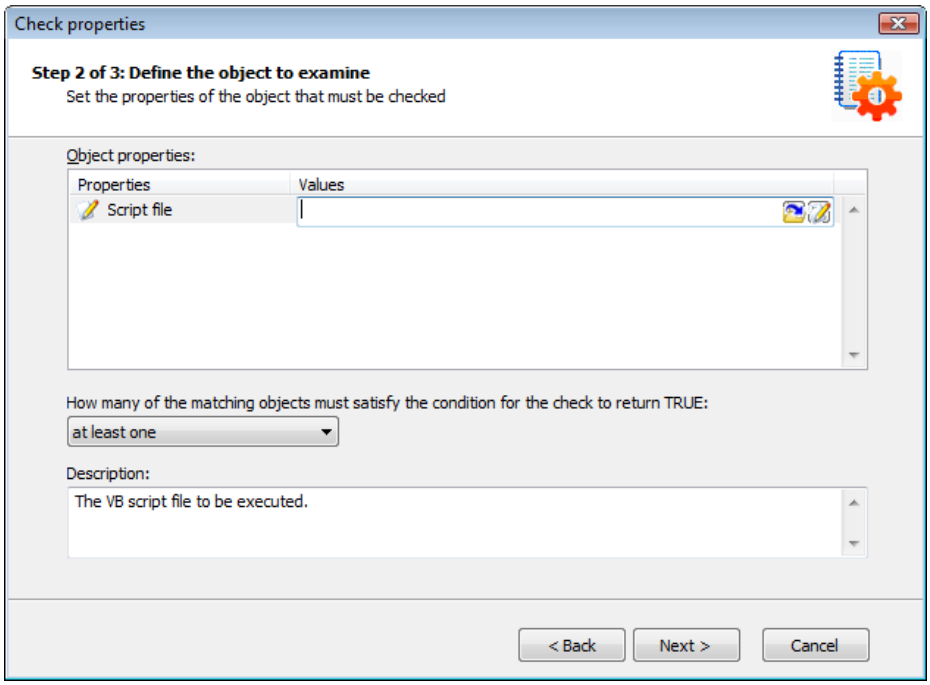
Screenshot 9 Add vulnerability check.

4. Enter the basic details including name etc,

5. Select the **Conditions** tab and select the **Add** button.

6. Set the check type to **Independent Checks ► VB Script Test.**

7. Specify the location of the script (e.g. "`<ProductDataDir>\Scripts\`newvulnerability.vbs"). Click **Next**.

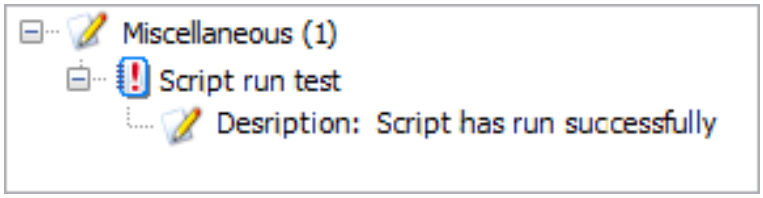


Screenshot 10 Specify Python script.

8. In the **Value** edit box specify the value returned by the Python script when the vulnerability is discovered. Click **Finish** button.

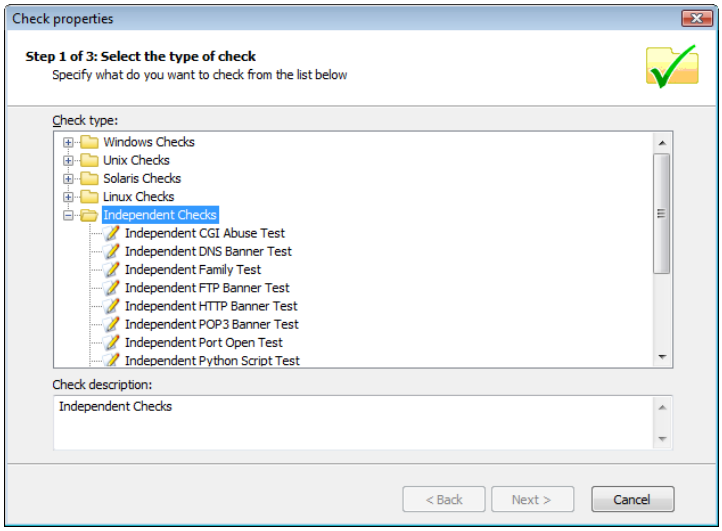
The vulnerability check is added and will be included in the list of vulnerabilities checked for on the next scan of a computer.

To test it out, simply scan your localhost machine (127.0.0.1) and you should see the vulnerability warning under the miscellaneous section of the vulnerabilities node of the scan results.



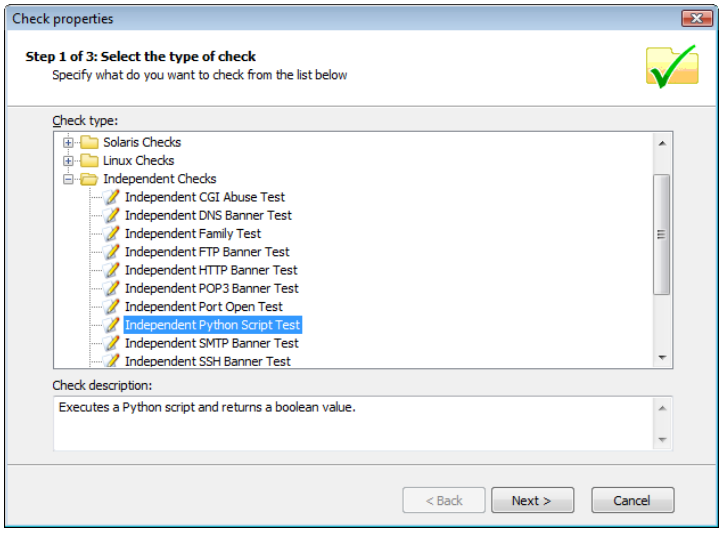
Screenshot 11 Vulnerability detection.

2. Python Scripting



Screenshot 12 Independent checks.

Starting with version 9.0, GFI LANguard supports a new type of vulnerability checks: **Python Script Test**. This type of check is available under the Independent Checks type.



Screenshot 13 Phyton Script Test.

2.1 What is the Python programming language?

Python is an interpreted programming language created by Guido van Rossum in 1990. Python is entirely dynamically typed and uses automatic memory management.

One important thing to remember is that instead of punctuation or keywords, Python source code uses indentation itself to indicate the run of a block. Example of a factorial function in Python:

```
def factorial(x):
 if x == 0:
 return 1
 else:
 return x * factorial(x-1)
```

2.2 Creating a new vulnerability check of type Python Script Test

2.2.1 Step 1: Creating the script

Create the following valid Python script test.

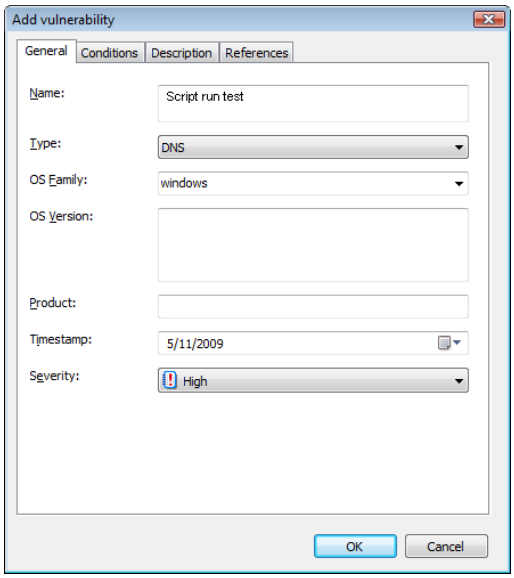
```
#PythonSuccessfullCheck.py
"""
See the file <ProductDataDir>\Scripts\lpy.py for details.
"""
def main():
"""Return values:
* 0 - false, failed
* 1 - true, success"""
result = 0
#Your code here...
result = 1
return(result)
```

2.2.2 Step 2: Creating the new vulnerability check

1. Open **GFI LANguard Main Menu ► Configure ► Scanning Profiles Editor ► New scanning profile** or select the scanning profile to edit

2. Open **Vulnerability Assessment Options** tab► **Vulnerabilities ► Vulnerabilities type**

3. Click **Miscellaneous** type and click **Add**, which will bring up the new vulnerability check dialog.



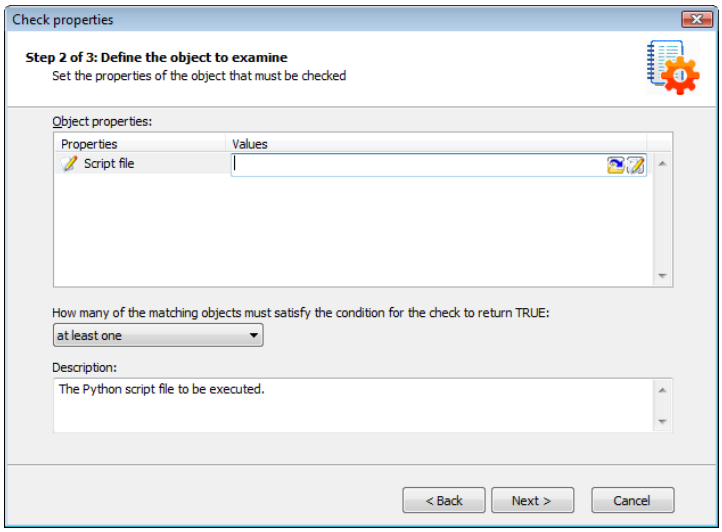
Screenshot 14 Add vulnerability.

4. Enter the basic details, name etc,

5. Select the **Conditions** tab and click **Add** button.

6. Set the check type to **Independent Checks ► Python Script Test**

7. Specify the location of the script (e.g. "`<ProductDataDir>\Scripts\`newvulnerability.py"). Click **Next**



Screenshot 15 Specify Python script.

8. In the **Value** edit box specify the value returned by the Python script when the vulnerability is discovered. Click **Finish** button.

9.The vulnerability check is added and will be included in the list of vulnerabilities checked for on the next scan of a computer.

2.3 Application Programming Interfaces (APIs) available in Python Scripts

GFI LANguard embeds most of the default Python implementation (also known as CPython, available from www.python.org). Currently we use Python version 2.5. Therefore, most existing Python scripts work with minor

modifications in GFI LANguard. The most important modification is that scripts must have a `main()` function which returns `1` or `0`.

GFI LANguard also provides a COM client API for Python scripts in the form of pywin32.

NOTE: All of the LANguard COM scripting libraries are available via `win32com.client` to Python scripts. For details, see below.

2.3.1 Documentation for Pywin32

```
#Hello world for pywin32
def main():
"""Return values:
* 0 - false, failed
* 1 - true, success"""
result = 0
import win32com.client
strComputer = "."
objWMIService = win32com.client.Dispatch("WbemScripting.SWbemLocator")
objSWbemServices = objWMIService.ConnectServer(strComputer,"root\cimv2")
colItems = objSWbemServices.ExecQuery("Select * from Win32_PhysicalMemory")
for objItem in colItems:
print("Capacity: %s\n" % objItem.Capacity)
result = 1
return(result)
```

Additional documentation is available from:

http://sourceforge.net/projects/pywin32/

http://www.boddie.org.uk/python/COM.html

2.4 Debugging Python scripts

Currently GFI LANguard does not provide a script debugger for Python scripts.

Any Python IDE, debugger or command line interpreter that runs Python scripts using the currently supported Python 2.5 and pywin32 distributions for Windows can help writing correct Python scripts for GFI LANguard.

Some Python IDEs or interpreters useful for editing and debugging Python scripts are: PythonWin, Winpdb, SPE IDE - Stani's Python Editor, IDLE and of course the Python interpreter that ships by default with the Python distribution.

3. Functions List

List of functions which were available in version 3.X of GFI LANguard and their equivalents in this version.

NOTE: Some functions are still standalone functions (marked with <Global>) and others are accessed via dedicated objects.

Function	Function Category in GFI LANguard 3.X	Object container in GFI LANguard 9.X
Open_tcp	Networking	Socket
Open_udb	Networking	Socket
Close	Networking	Socket
Recv	Networking	Socket
Send	Networking	Socket
RecvFrom *1	Networking	<obsolete>
SendTo	Networking	Socket
DnsLookup	Networking	Socket
ReverseDnsLookup	Networking	Socket
SetTimeout	<not available>	Socket
Whois	Networking	<obsolete>
SnmpGet*2	SNMP	SNMP
SnmpGetNext *2	SNMP	SNMP
SnmpSet *2	SNMP	SNMP
Connect	<not available>	SNMP
Close	<not available>	SNMP
Length *3	String	<VB equivalent>
Pos *3	String	<VB equivalent>
Left *3	String	<VB equivalent>
Right *3	String	<VB equivalent>
Delete *3	String	<VB equivalent>
Uppercase *3	String	<VB equivalent>
Lowercase *3	String	<VB equivalent>
Ord *3	String	<VB equivalent>
Dup *3	String	<VB equivalent>
Chr *3	String	<VB equivalent>
Mid *3	String	<VB equivalent>
Trim *3	String	<VB equivalent>
Strtoint *3	Conversion	<VB equivalent>
Inttostr *3	Conversion	<VB equivalent>
Inttohex *3	Conversion	<VB equivalent>
Base64Encode	Conversion	Encoding

Base64Decode	Conversion	Encoding
RegistryRead[*2]	Registry	Registry
Write	<not available>	Registry
GetFirstValue	<not available>	Registry
GetNextValue	<not available>	Registry
GetFirstKey	<not available>	Registry
GetNextKey	<not available>	Registry
DeleteValue	<not available>	Registry
DeleteKey	<not available>	Registry
RegExp [*3]	Miscellaneous	<VB equivalent>
Sleep[*4]	Miscellaneous	<Global>
Echo[*4]	Miscellaneous	<Global>
StatusBar[*4]	Miscellaneous	<Global>
WriteToLog[*4]	Miscellaneous	<Global>
AddListItem	<not available>	<Global>
SetDescription	<not available>	<Global>
Connect	<not available>	File
Open	<not available>	File
Close	<not available>	File
Read	<not available>	File
Write	<not available>	File
Writeline	<not available>	File
Seek	<not available>	File
Delete	<not available>	File
Size	<not available>	File
FileVersion	<not available>	File
ProductVersion	<not available>	File
Attribute	<not available>	File
Connect	<not available>	HTTP
GetURL	<not available>	HTTP
PostURL	<not available>	HTTP
SendRequest	<not available>	HTTP
AddHeader	<not available>	HTTP
ClearRequestHeaders	<not available>	HTTP
HeaderValue	<not available>	HTTPHeaders
HeaderName	<not available>	HTTPHeaders
Connect	<not available>	FTP
GetCurrentDirectory	<not available>	FTP
SetCurrentDirectory	<not available>	FTP
CreateDirectory	<not available>	FTP
RemoveDirectory	<not available>	FTP
DeleteFile	<not available>	FTP
GetFile	<not available>	FTP
PutFile	<not available>	FTP
RenameFile	<not available>	FTP
FindFirstFile	<not available>	FTP
FindNextFile	<not available>	FTP

FindFileClose	<not available>	FTP
GetFindFileName	<not available>	FTP
GetFindFileSize	<not available>	FTP
GetFindFileAttributes	<not available>	FTP
Base64Encode	Conversion	Encode
Base64Decode	Conversion	Encode

*1	Function is no longer applicable in GFI LANguard 8.X since it was replaced by another function or equivalent member of the parent Object. For e.g., (RecvFrom functionality is available by the Recv function of the Socket Object)
*2	Function is no longer standalone but part of a parent Object. For e.g. SnmpGet, snmpGetNext, and snmpSet are now named Get, GetNext, and set respectively in GFI LANguard 8.X . This is because they are no longer separate standalone functions but part of the SNMP Object.
*3	These functions are now available via their VB native functions equivalents
*4	Global functions are not part of an Object. Use these functions directly as normal native VB functions in scripts

4. Old -> New Function Mapping

With the extended VBScript language based scripting engine some of the functions available in previous versions of GFI LANguard are now available under different function names / object locations.

This section describes the equivalent function in GFI LANguard 9.X of a particular function available in GFI LANguard 3.X

Function name in GFI LANguard 3.x	Parent Object / VB Equivalent in GFI LANguard 9.X	Function name in GFI LANguard 9.X
RecvFrom	Socket	Recv
SnmpGet	SNMP	Get
SnmpGetNext	SNMP	GetNext
SnmpSet	SNMP	Set
Length	<Native VB language>	Len
Pos	<Native VB language>	Instr
Left	<Native VB language>	Left
Right	<Native VB language>	Right
Delete *1	<Native VB language>	<not available>
Uppercase	<Native VB language>	UCase
LowerCase	<Native VB language>	LCase
Ord	<Native VB language>	Asc
Dup *2	<Native VB language>	<not available>
Chr	<Native VB language>	Chr
Mid	<Native VB language>	Mid
Trim	<Native VB language>	Trim
Strtoint	<Native VB language>	Cint
Inttostr	<Native VB language>	Cstr
Inttohex	<Native VB language>	Hex
RegistryRead	Registry	Read
RegExp *3	<ActiveX>	RegExp

*1	Delete is not implemented but it can be achieved by using left and right in conjunction. **Example:** In order to convert x = Delete(x, 5, 4) to the new system use the following system: x = Left(x,4)+Right(x, 5)
*2	In GFI LANguard 3.X, Dup was used to create multiple copies of a string. This functionality is now achieved using a loop: **Example:** LongString=”Create Multiple Copies Of This String” For Copies = 1 to 5

	LongString = LongString + LongString Next Copies
*3	These functions are now available via their VB native functions equivalents

*4	Regular expression functionality can be used using the Microsoft regular expression activeX component.
	Example:
	`Dim regexp As Object`
	`Set regexp = CreateObject("VBScript.regexp")`
	`regexp.pattern = "[A-Za-z]:(\\[A-Za-z0-9_.-])*"`
	`If regexp.test(string_to_Check) Then`
	`' Function to execute if regular expression returns true go here`
	`End If`

5. Object Documentation

5.1 Socket Object:

The Socket Object is used to allow communication with TCP servers and UDP sockets. It supports various functionalities such as configurable timeouts, dns lookups and also reverse dns lookups. The Socket object handles also various data types in its sending and receiving operation. It can handle both strings and also raw binary data. Strings are handled by using normal string variables and binary data is handled by using array of bytes. The receive function (called recv) also accepts a parameter that instructs it as to which data type to return. This allows the user to specify if he wants his data to be received as string or as binary.

5.1.1 OpenTcp

OpenTcp is used to establish a connection with a remote server using TCP.

Syntax

OpenTcp(ip, port)

Returns

Socket Object

Example

'This Script displays the banner of an ftp server that is running locally

'It can be made to work with any ftp server simply by changing the value of the variable IP

```
Function Main
      Dim SocketObject As Object
      Dim ip As String
      Dim port As String
      Dim strResponse As String
      Ip = "127.0.0.1"
      Port = "21"
      Socket.SetTimeout 5000,5000
      Set SocketObject = Socket.OpenTcp(Ip,Port)
      If Not SocketObject is Nothing Then
      'check to see that the Object returned successfully
      strResponse = SocketObject.Recv(1024)
      echo(strResponse)
      SocketObject.Close
      End If
End Function
```

5.1.2 OpenUdp

OpenUdp is used to establish a connection with a remote server using UDP.

Syntax

OpenUdp()

Returns

Socket Object

Example

```
'This script connects with a DNS server, specified by
the IP variable and runs a query for www.gfi.com and
‘then displays the result
Function Main
      Dim SocketObject As Object
      Dim ip As String
      Dim port As String
      Dim rawdata As Variant
      Dim Response As Variant
      Ip = "172.16.130.40"
      Port = "53"
      strRequest="www.gfi.com"
      rawdata   =   Array(0,3,1,0,0,1,0,0,0,0,0,0,3,&H77,
      &H77,  &H77,  &H03,  &H67,  &H66,  &H69,  &H03,  &H63,
      &H6F,&H6D, 0,0,1,0,1)
      '^^^^^^^^^^
      'This part is the packet header of our request, it
      includes informations such as flags
      '^^^^^^^^^^
      'This is the request itself, www.gfi.com, note that
      '.' are 'represented as &H03 instead of &H2E
      '^^^^^^^^^^
      'This is the End header of our packet
      Set SocketObject = Socket.OpenUdp()
      If Not SocketObject is Nothing Then
            'check  to  see  that  the  Object  returned
            successfully
            SocketObject.SendTo IP,port,rawdata
            Response = SocketObject.Recv(1024)
            For     a     =     UBound(response)-3     To
            UBound(response)
                  echo(Response(a))
                  If a <> UBound(response) Then
                  echo(".")
                  End If
            Next a
            SocketObject.Close
```

```
    End If
End Function
```

5.1.3 Close

Close is used to free a previously assigned socket object.

Syntax

Close

Returns

No data returned.

Example

'This Script Displays the banner of an ftp server that is running locally

'it can be made to work with any ftp server simply by changing the value of the variable IP

```
Function Main
    Dim SocketObject As Object
    Dim ip As String
    Dim port As String
    Dim strResponse As String
    Ip = "127.0.0.1"
    Port = "21"
    Socket.SetTimeout 5000,5000
    Set SocketObject = Socket.OpenTcp(Ip,Port)
    If Not SocketObject is Nothing Then
        'check to see that the Object returned successfully
        strResponse = SocketObject.Recv(1024)
        echo(strResponse)
        SocketObject.Close
    End If
End Function
```

5.1.4 Recv

Recv is used to retrieve data from a socket (used for both TCP and UDP transmissions).

Syntax

Recv(SizeInBytes, [DataType])

More Information

The SizeInBytes parameter specifies how much of the buffer will be returned. The optional parameter “DataType” can be used to specify in what format the buffer should be returned. If nothing is specified the buffer is analyzed, and the appropriate DataType will be set accordingly.

Possible options for the DataType parameter are as follow:

0 – Return buffer as an array of bytes (ideal for raw data).

1 – Return Buffer as a string (ideal if you know that the buffer consists of raw text)

2 – Return buffer as string, convert non printable characters into "." Ideal when you know that the buffer is

mixed between plain text and special characters but when you're just interested in the plain text part.

Returns

String or an array of bytes.

Example

'This Script displays the banner of an ftp server that is running locally

'it can be made to work with any ftp server simply by changing the value of the variable IP

```
Function Main
      Dim SocketObject As Object
      Dim ip As String
      Dim port As String
      Dim strResponse As String
      Ip = "127.0.0.1"
      Port = "21"
      Socket.SetTimeout 5000,5000
      Set SocketObject = Socket.OpenTcp(Ip,Port)
      If Not SocketObject is Nothing Then
            'check to see that the Object returned
            successfully
            strResponse = SocketObject.Recv(1024,1)
            echo(strResponse)
            SocketObject.Close
      End If
End Function
```

5.1.5 Send

Send is used to send data to the current open socket over a TCP connection.

Syntax

Send (data, [SizeInBytes])

Returns

The actual amount of sent bytes.

More Information

The Send function can only be used with an open Socket Object that was opened on a TCP connect. To Send data over a UDP Connection see the SendTo function further on in the document.

The Send function accepts an optional parameter (SizeInBytes) which specifies how much of the buffer which was passed to the data field will actually be sent. If this optional parameter is omitted, then the size is automatically calculated.

Example

'This Script displays the default page in raw html of a web server running locally

'the script can be made to work with any web server simply by changing the value of the variable IP

```
Function Main
      Dim SocketObject As Object
      Dim ip As String
      Dim port As String
      Dim req As String
      Dim strResponse As String
      Ip = "172.16.130.112"
      Port = "80"
      req = "GET / HTTP/1.0"
      cr = Chr(13) + Chr(10)'carriage return and line feed
      req = CStr(req +cr +cr)
      Socket.SetTimeout 5000,5000
      Set SocketObject = Socket.OpenTcp(Ip,Port)
      If Not SocketObject is Nothing Then 'check to see that the Object returned successfully
            SocketObject.Send(CStr(req))
            strResponse = SocketObject.Recv(1024)
            While Len(CStr(strResponse)) <> 0
                  echo(strResponse)
                  StrResponse = SocketObject.Recv(1024)
            Wend
            echo(strResponse)
      End If
End Function
```

5.1.6 SendTo

SendTo is used to send data to the current open socket over a UDP Connection.

Syntax

SendTo (ip, port, data, [SizeInBytes])

Returns

The actual amount of bytes sent.

More Information

The SendTo function can only be used with an open Socket object that was opened on a UDP connect, in order to send data over a TCP Connection please check the Send function described earlier on in the document.

The SendTo function accepts an optional parameter (SizeInBytes) which specifies how much of the buffer which was passed to data field will actually

be sent. If this optional parameter is omitted, then the size is automatically calculated.

Example

```
'This script connects with a DNS server, specified by
the IP variable and runs a query for www.gfi.com and
‘than displays the result
Function Main
      Dim SocketObject As Object
      Dim ip As String
      Dim port As String
      Dim rawdata As Variant
      Dim Response As Variant
      Ip = "172.16.130.40"
      Port = "53"
      strRequest="www.gfi.com"
      rawdata  =  Array(0,3,1,0,0,1,0,0,0,0,0,0,3,  &H77,
      &H77,  &H77,  &H03,  &H67,  &H66,  &H69,  &H03,  &H63,
      &H6F,&H6D, 0,0,1,0,1)
      Set SocketObject = Socket.OpenUdp()
      If Not SocketObject is Nothing Then
       'check   to   see   that   the   Object   returned
      successfully
            SocketObject.SendTo IP,port,rawdata
            Response = SocketObject.Recv(1024)
                  For    a    =    UBound(response)-3    To
                  UBound(response)
                  echo(Response(a))
                  If a <> UBound(response) Then
                  echo(".")
                  End If
                  Next a
            SocketObject.Close
      End If
End Function
```

5.1.7 SetTimeout

The default timeout for sending / receiving data is 2 seconds. SetTimeout is used to set a different timeout both for sending and receiving data.

Syntax

SetTimeout(SendTimeout, RecieveTimeout)

Returns

No data returned.

More Information

SetTimeout needs to be set before setting the object which will be used for sending and receiving. Passed parameters for timeouts are in milliseconds. If -1 is passed as one of the value, the currently set value will be used.

Example

'This Script displays the banner of an ftp server that is running locally

'it can be made to work with any ftp server simply by changing the value of the variable IP

```
Function Main
      Dim SocketObject As Object
      Dim ip As String
      Dim port As String
      Dim strResponse As String
      Ip = "127.0.0.1"
      Port = "21"
      Socket.SetTimeout -1,5000
      Set SocketObject = Socket.OpenTcp(Ip,Port)
      If Not SocketObject is Nothing Then
      'check to see that the Object returned successfully
            strResponse = SocketObject.Recv(1024)
            echo(strResponse)
            SocketObject.Close
      End If
End Function
```

5.1.8 DnsLookup

DnsLookup is used to resolve host names into IP addresses. This function is mostly used when you wish to connect to servers and you do not know their IP.

Syntax

DnsLookup(hostname)

Returns

String (IP Address)

Example

‘very simple dns lookup and reverse lookup

```
Function Main
      Dim SocketObject As Object
      Dim ServerName As String
      Dim IP As String
      Dim ResolvedName As String
      cr = Chr(13) + Chr(10)'Carriage return and line
      feed
      ServerName = "whois.networksolutions.com"
      echo("Resolving"+cr)
      Socket.SetTimeout 5000,5000
```

```
    ip = socket.DnsLookup(ServerName)
    echo(ServerName + " resolves to the IP Address:"+cr
    )
    echo(ip +cr)
    ResolvedName = Socket.ReverseDnsLookup(ip)
    echo(cr)
    echo("IP Address "+ip+ " resolves to "+cr)
    echo(ResolvedName+cr)
End Function
```

5.1.9 ReverseDnsLookup

ReverseDnsLookup is used to resolve IP addresses into host names.

Syntax

ReverseDnsLookup(IP)

Returns

String : Containing the returned hostname.

Example

‘Very simple dns lookup and reverse lookup

```
Function Main
    Dim SocketObject As Object
    Dim ServerName As String
    Dim IP As String
    Dim ResolvedName As String
    cr = Chr(13) + Chr(10)
    'Carriage return and line feed
    ServerName = "whois.networksolutions.com"
    echo("Resolving"+cr)
    Socket.SetTimeout 5000,5000
    ip = socket.DnsLookup(ServerName)
    echo(ServerName + " resolves to the IP Address:"+cr
    )
    echo(ip +cr)
    ResolvedName = Socket.ReverseDnsLookup(ip)
    echo(cr)
    echo("IP Address "+ip+ " resolves to "+cr)
    echo(ResolvedName+cr)
End Function
```

5.2 SNMP Object:

The SNMP Object allows users to connect to SNMP for querying, and setting of values. The object also allows for the enumeration of Object Identifiers (OID). SNMP is generally used to retrieve system information on a service or device. Various devices also feature SNMP servers and thus using this object one can query various properties of these devices and

deduce possible security issues/weaknesses/incorrect configurations for those devices.

5.2.1 Connect

Connect is used to establish a connection with a remote server and return an SNMP object to it.

Syntax

Connect(ip, community_string)

Returns

SNMP Object

Example

'Very Simple SNMP Client that retrieves the SysName from a computer which has an SNMP server installed

```
Function Main
      Dim snmp1 As Object
      cr = Chr(13) + Chr(10)'Carriage return and line
      feed
      Set snmp1 = SNMP.Connect("127.0.0.1", "public")
      Val1 = "1.3.6.1.2.1.1.5.0"'OID of the sysName
      root = "1.3.6.1.2.1.1."'OID of the systems Object
      snmp1.Get Val1
      echo "Oid: '"+Val1 + "'"+cr
      echo "Value: '"+snmp1.Get(Val1)+"'"+cr
      snmp1.Close
End Function
```

5.2.2 Get

Get is used to retrieve the corresponding string to the specified OID.

Syntax

Get (oid)

Returns

String

Example

```
'Very Simple SNMP Client that retrieves the SysName from
a computer which has an SNMP server installed
Function Main
      Dim snmp1 As Object
      cr = Chr(13) + Chr(10)'Carriage return and line
      feed
      Set snmp1 = SNMP.Connect("127.0.0.1", "public")
      Val1 = "1.3.6.1.2.1.1.5.0"'OID of the sysName
      root = "1.3.6.1.2.1.1."'OID of the systems Object
      snmp1.Get Val1
      echo "Oid: '"+Val1 + "'"+cr
```

```
      echo "Value: '"+snmp1.Get(Val1)+"'"+cr
      snmp1.Close
End Function
```

5.2.3 GetNext

GetNext is used to retrieve the next corresponding string to the specified OID.

Syntax

GetNext (oid)

Returns

String

Example

'Very Simple SNMP Client that retrieves all the strings pertaining to the system Object from a computer which has an SNMP server installed

NOTE: that this is raw information, for example the uptime (OID 1.3.5.1.2.1.1.3.0) is displayed as hundreds of a second.

```
Function Main
      Dim snmp1 As Object
      cr = Chr(13) + Chr(10)'Carriage return and line
      feed
      Set snmp1 = SNMP.Connect("127.0.0.1", "public")
      Val1 = "1.3.6.1.2.1.1.1.0"'OID of the sysName
      root = "1.3.6.1.2.1.1."'OID of the systems Object
      'snmp1.Get Val1
      While Val1 <> ""'
            echo "Oid: '"+Val1 + "'"+cr
            echo "Value: '"+snmp1.Get(Val1)+"'"+cr
            Val1 = snmp1.GetNext(Val1)
            If InStr(Val1, root) <> 1 Then Val1 =""
      Wend
      snmp1.Close
End Function
```

5.2.4 Set

Set is used to set a value to a specified OID.

Syntax

Set (oid, String)

Returns

True if successful, false otherwise.

Example

```
'Very Simple SNMP Client that sets the sysLocation of a
computer which has an SNMP server installed on it to
"Malta"
```

```
'Please note that by default this script will always
fail because generally, the public community would be
set to read only
Function Main
      Dim snmp1 As Object
      cr = Chr(13) + Chr(10)'Carriage return and line
      feed
      Set snmp1 = SNMP.Connect("127.0.0.1", "public")
      Val1 = "1.3.6.1.2.1.1.6.0"'OID of the sysName
      root = "1.3.6.1.2.1.1."'OID of the systems Object
      If snmp1.Set(Val1, "Malta") = true Then
            echo("Value Set successfully")
      Else
            echo("Failed to Set value")
      End If
      snmp1.Close
End Function
```

5.2.5 Close

Close is used to close an open SNMP session.

Syntax

Close

Returns

No data returned.

Example

```
'Very Simple SNMP Client that retrieves the sysName from
a computer which has an SNMP server installed
Function Main
      Dim snmp1 As Object
      cr = Chr(13) + Chr(10)'Carriage return and line
      feed
      Set snmp1 = SNMP.Connect("127.0.0.1", "public")
      Val1 = "1.3.6.1.2.1.1.5.0"'OID of the sysName
      root = "1.3.6.1.2.1.1."'OID of the systems Object
      snmp1.Get Val1
      echo "Oid: '"+Val1 + "'"+cr
      echo "Value: '"+snmp1.Get(Val1)+"'"+cr
      snmp1.Close
End Function
```

5.3 File Object:

The File Object allows the user to open files both remotely and locally and perform read and write operations on them. Files can be opened in various modes ranging from creating a new file to opening an existent file to opening

a file and deleting its contents. Files can also be opened for reading, writing and also query mode only (where one can only check files size and attributes without being able to write or read from the file). Apart from writing and reading operations the File Object also supports common file operations such as checking file size, seeking and attributes.

5.3.1 Connect

Connect is used to connect to a machine (either local or remote) on which you want to open files.

Syntax

Connect (IP or NetBIOS name)

Returns

File Object

Example

'This script opens a file (test.txt) on the local C drive and writes 2 lines to it

```
Function Main
      Dim textfile As Object
      Set textfile = File.Connect("127.0.0.1")
      If    textfile.Open("c:\test.txt",    GENERIC_WRITE,
      CREATE_ALWAYS) Then
            textfile.WriteLine("Hi, This is a test file")
            textfile.WriteLine("It was created using GFI
            LANguard scripting")
            textfile.Close
      End If
End Function
```

5.3.2 Open

Opens a file for read or write.

Syntax

Open (Filename, mode, disposition)

Returns

True if the open operation succeeds, False otherwise.

More Information

Mode:

0 - Open file in query access mode, attributes maybe queried but the file may not be accessed

GENERIC_READ- Opens file for reading

GENERIC_WRITE- Open File for writing

Disposition:

CREATE_NEW- Creates a new file. The function fails if the specified file already exists.

CREATE_ALWAYS - Creates a new file. The function overwrites the file if it exists.

OPEN_EXISTING - Opens the file. The function fails if the file does not exist.

OPEN_ALWAYS - Opens the file, if it exists. If the file does not exist, the function creates the file.

TRUNCATE_EXISTING - Opens the file. Once opened, the file is truncated so that its size is zero bytes.

Example

'This script opens a file (test.txt) on the local C drive and writes 2 lines to it

```
Function Main
      Dim textfile As Object
      Set textfile = File.Connect("127.0.0.1")
      If    textfile.Open("c:\test.txt",    GENERIC_WRITE,
      CREATE_ALWAYS) Then
            textfile.WriteLine("Hi, This is a test file")
            textfile.WriteLine("It was created using GFI
            LANguard scripting")
            textfile.Close
      End If
End Function
```

5.3.3 Close

Close is used to close an instance of an open file.

Syntax

Close

Returns

No data returned.

Example

'This script opens a file (test.txt) on the local C drive and writes 2 lines to it

```
Function Main
      Dim textfile As Object
      Set textfile = File.Connect("127.0.0.1")
      If    textfile.Open("c:\test.txt",    GENERIC_WRITE,
      CREATE_ALWAYS) Then
            textfile.WriteLine("Hi, This is a test file")
            textfile.WriteLine("It was created using GFI
            LANguard scripting")
            textfile.Close
      End If
End Function
```

5.3.4 Read

Read is used to read a string of (x) length from a text file.

Syntax

Read(number_of_bytes, [DataType])

Returns

String

More Information

DataType is an optional parameter. If omitted the correct data type will be auto detected by the system.

Possible options for the DataType parameter are as follow:

0 – Return buffer as an array of bytes (ideal for raw data).

1 – Return Buffer as a string (ideal if you know that the buffer consists of raw text)

2 – Return buffer as string, non printable characters are ignored. This is Ideal when you know that the buffer is mixed between plain text and special characters but when you're just interested in the plain text part.

Example

```
'This script displays the contents of the hosts file
Function Main
      Dim textfile As Object
      Set textfile = File.Connect("127.0.0.1")
      If
      textfile.Open("c:\windows\system32\drivers\etc\host
      s", GENERIC_READ, Open_Existing) Then
            echo(textfile.read(1024,1))
            textfile.Close
      End If
End Function
```

5.3.5 Write

Write is used to write a string to a file without appending a CRLF (Carriage Return Line Feed) at the end of the provided string.

Syntax

Write(string, [number_of_bytes])

Returns

No data returned.

More Information

Number_of_bytes is an optional parameter, if omitted its value will be automatically calculated according to the size of the string passed.

Example

'This script opens a file (test.txt) on the local C drive and writes a couple of lines to it.

'The first line is composed by a write and WriteLine function

```
Function Main
      Dim textfile As Object
      Set textfile = File.Connect("127.0.0.1")
      If    textfile.Open("c:\test.txt",    GENERIC_WRITE,
      CREATE_ALWAYS) Then
            textfile.Write("Hi,")
            textfile.WriteLine(" This is a test file")
```

```
            textfile.WriteLine("It was created using GFI
            LANguard scripting engine")
            textfile.Close
      End If
End Function
```

5.3.6 WriteLine

WriteLine is used to write a string to a file and append a CRLF (Carriage Return Line Feed) at the end of the provided string

Syntax

WriteLine(string)

Returns

Boolean: True (non-zero) if write succeeded and False (zero) otherwise

Example

'This script opens a file (test.txt) on the local C drive and writes 2 lines to it

```
Function Main
      Dim textfile As Object
      Set textfile = File.Connect("127.0.0.1")
      If    textfile.Open("c:\test.txt",    GENERIC_WRITE,
      CREATE_ALWAYS) Then
            textfile.WriteLine("Hi, This is a test file")
            textfile.WriteLine("It was created using GFI
            LANguard scripting")
            textfile.Close
      End If
End Function
```

5.3.7 Seek

Seek is used to move to an alternative position in the file (from where to read or write)

Syntax

Seek(Distance, Method)

Returns

Current position in the file

More Information

Distance is a measurement of how many bytes to move the cursor.

Method can be one of the following:

0 -Move cursor specified distance starting from the beginning of the file

1 -Move cursor specified distance starting from current position in the file

2 -Move cursor specified distance starting from the end of the file

Example

'This script displays the contents of the hosts file after moving the cursor 50 characters into the file

```
Function Main
```

```
      Dim textfile As Object
      Set textfile = File.Connect("127.0.0.1")
      If
      textfile.Open("c:\windows\system32\drivers\etc\host
      s", GENERIC_READ, Open_Existing) Then
            Textfile.Seek 50,0
            echo(textfile.read(1024))
            textfile.Close
      End If
End Function
```

5.3.8 Delete

Delete is used to delete files on the hard disk

Syntax

Delete (path to file)

More Information

You must be connected to the machine before you can delete the file.

NOTE: Do not open the file you are currently running delete on, or else the file would be locked and the delete operation will fail.

Returns

True if the delete operation succeeds, False otherwise.

Example

'This script deletes the file (test.txt) on the local C drive if it exists.

```
Function Main
      Dim textfile As Object
      Set textfile = File.Connect("127.0.0.1")
      If textfile.Delete("c:\test.txt") = true Then
            echo("File Deleted Successfully")
      else
            echo("Delete Failed")
      End If
End Function
```

5.3.9 Size

Size returns the size of a file.

Syntax

Size ([highpart])

Returns

Size of the file (the lowpart property of the size of the file)

More Information

A file size has two parts. A highpart and a low part. What is returned by the function is the lowpart. The highpart is an optional parameter in which you can Get the highpart size property of a file should you need it.

Example

'Displays the size of the hosts file

```
Function Main
      Dim textfile As Object
      Dim size As Integer
      Set textfile = File.Connect("127.0.0.1")
      If
      textfile.Open("c:\windows\system32\drivers\etc\host
      s", GENERIC_READ, Open_Existing) Then
            size = Textfile.Size
            echo("your host file has a file size of : " &
            size & " bytes")
            textfile.Close
      End If
End Function
```

5.3.10 FileVersion

FileVersion returns the version of a file if it has the necessary properties assigned.

Syntax

FileVersion (String)

Returns

Version of the file if file version information exists (returns a string).

Example

'Displays the file version of the calc.exe

```
Function Main
      Dim textfile As Object
      Dim ver As String
      Set textfile = File.Connect("127.0.0.1")
      If    textfile.Open("c:\windows\system32\calc.exe",
      GENERIC_READ, Open_Existing) Then
            ver = Textfile.FileVersion
            echo("Your Calc.exe file version is : " &
            ver)
            textfile.Close
      End If
End Function
```

5.3.11 ProductVersion

ProductVersion returns the product version property of a file if this information exists.

Syntax

ProductVersion

Returns

String

Example

'Displays the product version of the calc.exe

```
Function Main
      Dim textfile As Object
      Dim ver As String
      Set textfile = File.Connect("127.0.0.1")
      If    textfile.Open("c:\windows\system32\calc.exe",
      GENERIC_READ, Open_Existing) Then
            ver = Textfile.ProductVersion
            echo("Your Calc.exe Product version is : " &
            ver)
            textfile.Close
      End If
End Function
```

5.3.12 Attributes

Returns the attributes of a file.

Syntax

Attributes

Returns

Integer : containing encoded in it the attributes of the file.

More Information

Attributes values:

1- Read Only – File is set as read only.

2- Hidden – The file or directory is hidden.

4- System – The file or directory is an operation system file or directory.

16- Directory – This item is a directory.

32- Archive – The file or directory is an archive file or directory.

64- Device – Reserved, not to be used.

128- Normal – The File has no attributes.

256- Temporary File – This file is marked as being temporary.

512- Sparse File – This file has the sparse attribute assigned to it.

1024- Reparse point – the file or directory has an associated reparse point.

2048- Compressed - The file or directory is compressed.

4096- Offline – The file has been moved into offline storage and data is not currently available.

8192 - No Index – This file will not be indexed.

16384- Encrypted – This file is encrypted.

NOTE: If the file has a mixture of these attributes, the value will add to each other. Example an archive which is also read only and hidden, would return a value of 35 (32 for archive, 1 for read only and 2 for hidden)

Example

'Displays the attributes of the calc.exe

```
Function Main
```

```
    Dim textfile As Object
    Dim att As Integer
    Set textfile = File.Connect("127.0.0.1")
    If    textfile.Open("c:\windows\system32\calc.exe",
    GENERIC_READ, Open_Existing) Then
    att = Textfile.Attributes
    echo("Your Calc.exe attribute value is : " & att)
    textfile.Close
    End If
End Function
```

5.4 Registry Object:

The Registry Object contains functions designed to enable users to retrieve and set data in the registry both locally and remotely. The object supports all the different data types found in the registry: (reg_dword, reg_sz,,reg, multi_sz, reg_binary).

The object also provides functions for enumeration and deletion of keys and values.

5.4.1 Connect

Used to create a Connection to the registry of the specified machine.

Syntax

```
Connect (IP or Netbios name)
```

Returns

Registry Object

Example

```
'This script Gets the version of Internet Explorer by
reading it directly from the registry.
Function Main
    Dim Ro As Object
    Dim ie_version as string
    Set Ro = Registry.Connect("127.0.0.1")
    ie_version  =  ro.Read("SOFTWARE\Microsoft\Internet
    Explorer\Version Vector", "IE")
    echo "IE Version is " + ie_version
End Function
```

5.4.2 Read

Read is a function used to read values of registry keys

Syntax

Read(Path, ValueName)

Returns

Long - if registry value is REG_DWORD

String - if registry value is REG_SZ

Array of Strings- If registry value is REG_MULTI_SZ

Array of bytes - if registry value is REG_BINARY

Example

```
'This script gets the version of Internet Explorer by
reading it directly from the registry.
Function Main
      Dim Ro As Object
      Dim ie_version as string
      Set Ro = Registry.Connect("127.0.0.1")
      ie_version  =  ro.Read("SOFTWARE\Microsoft\Internet
      Explorer\Version Vector", "IE")
      echo "IE Version is " + ie_version
End Function
```

5.4.3 Write

Write is a function used to write values to registry keys

Syntax

Write(Path, ValueName, Value)

Returns

No data returned.

More Information

Use the following declaration to achieve the correct value type

Long -if registry value is REG_DWORD

String -if registry value is REG_SZ

Array of Strings- if registry value is REG_MULTI_SZ

(arrays need to be declared as variants and then value assigned to them using the array() function)

Example: Dim test as variant

Test = array(10,2,10)

NOTE : If the key does not exist, it will be created.

Example

```
'This script writes the value "test" to a particular Key
'SOFTWARE\Microsoft\testkey\testsubkey.
Function Main
      Dim Ro As Object
      Dim test As String
      test = "testvalue"
      Set Ro = Registry.Connect("127.0.0.1")
      ro.write              "SOFTWARE\Microsoft\testkey",
      "testsubkey",test
End Function
```

5.4.4 GetFirstValue

GetFirstValue is a function whose purpose is to initiate the enumeration of a registry path.

Syntax

```
GetFirstValue(Path, ValueName)
```

Returns

Long - if registry value is REG_DWORD

String - if registry value is REG_SZ

Array of Strings- If registry value is REG_MULTI_SZ

Array of bytes - if registry value is REG_BINARY

More Information

ValueName must be a variable of type variant. GetFirstValue will return the name of the attribute which contains the value returned inside the variable ValueName.

Example

'This scripts lists all of the programs that run on startup

```
Function Main
      Dim Ro As Object
      Dim valueName as variant
      cr = Chr(13) + Chr(10)
      Set Ro = Registry.Connect("127.0.0.1")
      Value                                                =
      ro.GetFirstValue("SOFTWARE\Microsoft\Windows\Curren
      tVersion\Run", valueName)
      While Value <> ""
            Echo "ValueName: " & valueName & " = " &
            value & cr
            Value = ro.GetNextValue(valueName)
      Wend
End Function
```

5.4.5 GetNextValue

GetNextValue is a function used in the enumeration process of registry paths. It will return subsequent values, on the sequence started by GetFirstValue.

Syntax

```
GetNextValue(ValueName)
```

Returns

Long - if registry value is REG_DWORD

String - if registry value is REG_SZ

Array of Strings- If registry value is REG_MULTI_SZ

Array of bytes - if registry value is REG_BINARY

More Information

ValueName must be a variable of type variant. GetNextValue will return the name of the attribute which contained the value returned inside the variable ValueName.

Example

'This scripts lists all of the programs that run on startup

```
Function Main
      Dim Ro As Object
      Dim valueName as variant
      cr = Chr(13) + Chr(10)
      Set Ro = Registry.Connect("127.0.0.1")
      Value                                                    =
      ro.GetFirstValue("SOFTWARE\Microsoft\Windows\Curren
      tVersion\Run", valueName)
      While Value <> ""
            Echo "ValueName: " & valueName & " = " &
            value & cr
            Value = ro.GetNextValue(valueName)
      Wend
End Function
```

5.4.6 GetFirstKey

Used to start the enumeration of keys residing in a registry path.

Syntax

```
GetFirstKey(Path)
```

Returns

```
String – name of the first key
```

Example

'This scripts lists all of keys under Microsoft

```
Function Main
      Dim Ro As Object
      cr = Chr(13) + Chr(10)
      Set Ro = Registry.Connect("127.0.0.1")
      Value = ro.GetFirstKey("SOFTWARE\Microsoft")
      While Value <> ""
            Echo "Keyname = " & value & cr
            Value = ro.GetNextKey
      Wend
End Function
```

5.4.7 GetNextKey

GetNextKey is used to continue the enumeration of keys which was started by the GetFirstKey function.

Syntax

GetNextKey

Returns

String : containing name of key

Example

'This scripts lists all of keys under Microsoft

```
Function Main
      Dim Ro As Object
      cr = Chr(13) + Chr(10)
      Set Ro = Registry.Connect("127.0.0.1")
      Value = ro.GetFirstKey("SOFTWARE\Microsoft")
      While Value <> ""
            Echo "Keyname = " & value & cr
            Value = ro.GetNextKey
      Wend
End Function
```

5.4.8 DeleteValue

DeleteValue is a function used to delete values from the registry keys.

Syntax

DeleteValue(Path, ValueName)

Returns

0 – on deletion success, error number on failure.

Example

```
'This script deletes the registry key created in the
write example above
Function Main
      Dim Ro As Object
      Dim result As Integer
      Set Ro = Registry.Connect("127.0.0.1")
      result                                          =
      ro.DeleteValue("SOFTWARE\Microsoft\testkey",
      "testsubkey")
      If result = 0 Then
            Echo "Value Deleted Successfully"
      Else
            Echo "Failed to Delete Value, Error code: " &
            result
      End If
End Function
```

5.4.9 DeleteKey

DeleteKey is a function used to delete registry keys.

Syntax

DeleteKey(Path)

Returns

0 – on deletion success, error number on failure.

Example

```
'This script deletes the registry key created in the
write example above
```

```
Function Main
      Dim Ro As Object
      Dim result As Integer
      Set Ro = Registry.Connect("127.0.0.1")
      result = ro.DeleteKey("SOFTWARE\Microsoft\testkey")
      If result = 0 Then
            Echo "Value Deleted Successfully"
      Else
            Echo "Failed to Delete Value, Error code: " &
            result
      End If
End Function
```

5.5 HTTP Object

This object contains a number of functions which make it easier for a user to perform web requests. This object has support for a wide variety of scenarios including authentication, proxies, proxy authentication and header manipulation. Both get and post retrieval methods are supported. The object also supports setting of custom headers and verbs. Each request not only returns the headers and the body of that particular request but also the result code of the operation. This means that if the script is aimed at verifying if a page exists or not, the user will not have to parse the reply but just check the returned code, for example, if the code returned is 404, this means that the page requested doesn't exist.

5.5.1 Connect

Connect is used to set the hostname or IP address and the port of the HTTP server in the Object.

Syntax

HTTP.Connect (STRING "hostname", LONG port)

Hostname can be the IP address or the hostname (eg. www.gfi.com)

Port is the port number – an Integer between 1 and 65535

Returns

HTTP Object

Example

```
' This script will do a GET request and print out the
return code
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' set up the request type
      HTTPobj.GetURL("/")
      ' to pass through the proxy with automatic
      authentication
```

```
      ' Authentication needs to be set to 1
      HTTPobj.Authentication = 1
      ' Send the GET request
      HTTPResponse = HTTPobj.SendRequest ()
      echo "Result: " + cstr(HTTPResponse)
End Function
```

5.5.2 GetURL

GetUrl is used to initiate a GET request to an HTTP server. GET requests are used to retrieve documents on the HTTP server.

Syntax

```
GetUrl (STRING document)
Document is a string (eg. "/index.html")
```

Returns

No data returned.

Example

```
' This script will do a GET request and print out the
return code
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      '  to pass through the proxy with automatic
      authentication
      ' Authentication needs to be Set to 1
      HTTPobj.Authentication = 1
      ' Send the GET request
      HTTPResponse = HTTPobj.SendRequest ()
      echo "Result: " + cstr(HTTPResponse)
End Function
```

5.5.3 PostURL

PostUrl is used to initiate a POST request to an HTTP server. POST requests are used to send data to an HTTP server.

Syntax

```
PostUrl (STRING document, STRING data)
Document is a string (eg. "/index.html")
Data is a string (eg. "value1=data1")
```

Returns

No data returned.

Example

```
' This script will do a POST request and print out the
return code
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.PostURL "/", "test"
      '  to  pass  through  the  proxy  with  automatic
      authentication
      ' Authentication needs to be set to 1
      HTTPobj.Authentication = 1
      ' Send the POST request
      HTTPResponse = HTTPobj.SendRequest ()
      echo "Result: " + cstr(HTTPResponse)
End Function
```

5.5.4 SendRequest

SendRequest is used to Send the initiated HTTP request. For example, if previously the GetURL method was used a GET request will be sent.

Syntax

SendRequest ()

Return value

HTTP Reponse code.

Example

```
' This script will do a GET request and print out the
return code
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      '  to  pass  through  the  proxy  with  automatic
      authentication
      ' Authentication needs to be set to 1
      HTTPobj.Authentication = 1
      ' Send the GET request
      HTTPResponse = HTTPobj.SendRequest ()
      echo "Result: " + cstr(HTTPResponse)
End Function
```

5.5.5 AddHeader

AddHeader modifies an initiated request to add, delete or modify an existing header.

Syntax

AddHeader (STRING name, STRING value)

Name is a string (eg. "Content-Type"). If the name already exists, the value of that name will be overwritten with the value specified.

Value is a string (eg. "text/html"). If the value is empty the header will be deleted if it exists.

Return Value

No data returned.

Example

```
' This script will modify some headers in an attempt to
launch
' a Cross Site Scripting attack on log file parsers
Function Main
      Dim HTTPObj As Object
      Dim headers As Variant
      ip = "www.gfi.com"
      port = 80
      cr = Chr(13) + Chr(10)
      XSSTest = "<script>alert('The new GFI LANguard
      features detection of Cross Site Scripting
      Detection')</script>"
      Set HTTPobj = HTTP.Connect (ip,port)
      ' headers to try
      headers = Array ( "Host", "User-Agent", "Accept",
      "X-Header1" , "X-Proxy", "Cookie" )
      HTTPobj.GetURL("/")
      HTTPobj.Authentication = 1
      ' a loop for each header which might be used to
      ' inject XSS signature. Send a request every time
      For a = LBound(headers) To UBound(headers)
            HTTPobj.ClearRequestHeaders
            HTTPobj.AddHeader headers(a), XSSTest
            ' Send the GET request with our custom
            headers
            HTTPResponse = HTTPobj.SendRequest ()
            echo CStr(a) + " result: " +
            CStr(HTTPResponse)+cr
      Next
End Function
```

ClearRequestHeaders

Clears all headers which were previously set with the AddHeader method.

Syntax

ClearRequestHeaders

Return Value

No data returned.

Example

```
' This script will modify some headers in an attempt to
launch
' a Cross Site Scripting attack on log file parsers
Function Main
      Dim HTTPObj As Object
      Dim headers As Variant
      ip = "www.gfi.com"
      port = 80
      cr = Chr(13) + Chr(10)
      XSSTest = "<script>alert('The new GFI LANGUARD
      features detection of Cross Site Scripting
      Detection')</script>"
      Set HTTPobj = HTTP.Connect (ip,port)
      ' headers to try
      headers = Array ( "Host", "User-Agent", "Accept",
      "X-Header1" , "X-Proxy", "Cookie" )
      HTTPobj.GetURL("/")
      HTTPobj.Authentication = 1
      ' a loop for each header which might be used to
      ' inject XSS signiture. Send a request every time
      For a = LBound(headers) To UBound(headers)
            HTTPobj.ClearRequestHeaders
            HTTPobj.AddHeader headers(a), XSSTest
            ' Send the GET request with our custom
            headers
            HTTPResponse = HTTPobj.SendRequest ()
            echo CStr(a) + " result: " +
            CStr(HTTPResponse)+cr
      Next
End Function
```

5.5.6 Verb Property

Determines the HTTP request method. This property is set implicitly when using GetURL and PostURL methods.

Syntax

HTTPObject.Verb

Verb: String (read/write)

Example

```
' This script will Send an OPTIONS http request
```

```
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.Authentication = 1
      HTTPobj.Verb = "OPTIONS"
      ' Send the OPTIONS request with our custom headers
      HTTPResponse = HTTPobj.SendRequest ()
      echo HTTPobj.RawResponseHeaders
End Function
```

5.5.7 HTTPVersion Property

Determines the HTTP version. If not set, the HTTPVersion will be "HTTP/1.1"

Syntax

```
HTTPObject.HTTPVersion
HTTPVersion: String (read/write)
```

Example

```
' This script will Send an HTTP/1.0 request
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.Authentication = 1
      HTTPobj.HTTPVersion = "HTTP/1.0"
      ' Send the GET request with our custom headers
      HTTPResponse = HTTPobj.SendRequest ()
      echo HTTPobj.RawResponseHeaders
End Function
```

5.5.8 IP Property

Used to set or retrieve the IP address or host name.

Syntax

```
HTTPObject.IP
IP: String (read/write)
```

Example

```
' This script will re-use the same Object to connect to
a different host and Send the same request
Function Main
      Dim HTTPObj As Object
      ip1 = "www.gfi.com"
      ip2 = "127.0.0.1"
      port = 80
      cr = Chr(13) + Chr(10)
      Set HTTPobj = HTTP.Connect (ip1,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      ' to pass through the proxy with automatic
      authentication
      ' Authentication needs to be set to 1
      HTTPobj.Authentication = 1
      ' Send the GET request
      HTTPResponse1 = HTTPobj.SendRequest ()
      HTTPobj.IP = ip2
      HTTPResponse2 = HTTPobj.SendRequest ()
      echo "Result: " + CStr(HTTPResponse1)+cr
      echo "Result: " + CStr(HTTPResponse2)+cr
End Function
```

5.5.9 Port Property

Sets or retrieves the port of the HTTP server to connect to.

Syntax

```
HTTPObject.Port
Port: String (read/write)
```

Example

```
' This script will re-use the same Object to connect to
a different port and Send the same request
Function Main
      Dim HTTPObj as Object
      ip = "127.0.0.1"
      port1 = 80
      port2 = 81
      cr = Chr(13) + Chr(10)
      Set HTTPobj = HTTP.Connect (ip,port1)
      ' Set up the request type
      HTTPobj.GetURL("/")
      ' to pass through the proxy with automatic
      authentication
      ' Authentication needs to be set to 1
      HTTPobj.Authentication = 1
```

```
      ' Send the GET request
      HTTPResponse1 = HTTPobj.SendRequest ()
      HTTPobj.PORT = port2
      HTTPResponse2 = HTTPobj.SendRequest ()
      echo "Result: " + cstr(HTTPResponse1)+cr
      echo "Result: " + cstr(HTTPResponse2)+cr
End Function
```

5.5.10 RawResponseHeaders Property

Contains all headers in the HTTP response. Each header is separated by a CR/LF pair.

Syntax

```
HTTPObject.RawResponseHeaders
RawResponseHeaders: String (read)
```

Example

```
' This script will Send an OPTIONS http request
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.Authentication = 1
      HTTPobj.Verb = "OPTIONS"
      ' Send the OPTIONS request with our custom headers
      HTTPResponse = HTTPobj.SendRequest ()
      echo HTTPobj.RawResponseHeaders
End Function
```

5.5.11 Body Property

Contains the response body.

Syntax

```
HTTPObject.Body
Body: String (read)
```

Example

```
' This script will do a GET request and print out the
body
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
```

```
    ' Set up the request type
    HTTPobj.GetURL("/")
    ' to pass through the proxy with automatic
    authentication
    ' Authentication needs to be set to 1
    HTTPobj.Authentication = 1
    ' Send the GET request
    HTTPResponse = HTTPobj.SendRequest ()
    echo HTTPobj.Body
End Function
```

5.5.12 Authentication Property

Enables or disables HTTP and Proxy authentication. Authentication is implicitly set to TRUE if ProxyUser, ProxyPassword, HttpUser and HttpPassword are set.

Syntax

```
HTTPObject.Authentication
Authentication: BOOLEAN (read/write)
```

Example

```
' This script will do a GET request and print out the
return code
Function Main
    Dim HTTPObj as Object
    ip = "www.gfi.com"
    port = 80
    Set HTTPobj = HTTP.Connect (ip,port)
    ' Set up the request type
    HTTPobj.GetURL("/")
    ' to pass through the proxy with automatic
    authentication
    ' Authentication needs to be set to 1
    HTTPobj.Authentication = 1
    ' Send the GET request
    HTTPResponse = HTTPobj.SendRequest ()
    echo HTTPResponse
End Function
```

5.5.13 ProxyUser Property

Username for the Proxy Authentication

Syntax

```
HTTPObject.ProxyUser
ProxyUser: String (read/write)
```

Example

```
' This script will do a GET request and print out the
return code
' Sets the username and password as "LANguard_test"
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.ProxyUser = "LANguard_test"
      HTTPobj.Proxypassword = "LANguard_test"
      ' Send the GET request
      HTTPResponse = HTTPobj.SendRequest ()
      echo HTTPobj.Body
End Function
```

5.5.14 ProxyPassword Property

Password for the Proxy Authentication

Syntax

HTTPObject.ProxyPassword

ProxyPassword: String (read/write)

Example

```
' This script will do a GET request and print out the
return code
' Sets the username and password as "LANguard_test"
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.ProxyUser = "LANguard_test"
      HTTPobj.Proxypassword = "LANguard_test"
      ' Send the GET request
      HTTPResponse = HTTPobj.SendRequest ()
      echo HTTPobj.Body
End Function
```

5.5.15 HttpUser Property

Username for the HTTP Authentication

Syntax

```
HTTPObject.HttpUser
HttpUser: String (read/write)
```

Example

```
' This script will do a GET request and print out the
return code
' Sets the username and password as "LANguard_test"
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.HTTPUser = "LANguard_test"
      HTTPobj.HTTPpassword = "LANguard_test"
      ' Send the GET request
      HTTPResponse = HTTPobj.SendRequest ()
      echo HTTPobj.Body
End Function
```

5.5.16 HttpPassword Property

Password for the HTTP Authentication

Syntax

HTTPObject.HttpPassword

HttpPassword: String (read/write)

Example

```
' This script will do a GET request and print out the
return code
' Sets the username and password as "LANguard_test"
Function Main
      Dim HTTPObj as Object
      ip = "www.gfi.com"
      port = 80
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.HttpUser = "LANguard_test"
      HTTPobj.Httppassword = "LANguard_test"
      ' Send the GET request
      HTTPResponse = HTTPobj.SendRequest ()
      echo HTTPobj.Body
End Function
```

5.5.17 ResponseHeaders Property

Header Object which gives access to individual response headers.

Syntax

```
HTTPObject.ReponseHeaders
ResponseHeaders: Object (read)
```

Example

```
' This script will print out the name of the HTTP server
Function Main
      Dim HTTPObj as Object
      Dim headers as Object
      ip = "www.apache.org"
      port = 80
      cr = Chr(13) + Chr(10)
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.verb = "HEAD"
      ' to pass through the proxy with automatic
      authentication
      ' Authentication needs to be set to 1
      HTTPobj.Authentication = 1
      ' Send the HEAD request
      HTTPResponse = HTTPobj.SendRequest ()
      ' Set new Object called headers
      Set headers = HTTPobj.ResponseHeaders
      ' HTTPResponse contains the return code
      echo "Result: " + cstr(HTTPResponse) + cr
      ' the http result will look something like :
      ' HTTP/1.1 200 OK
      ' Server: Microsoft-IIS/5.0
      ' Date: Tue, 28 Oct 2003 10:23:19 GMT
      ' Content-Length: 1270
      ' Content-Type: text/html
      echo "Server running on " + ip + " is " +
      headers.HeaderValue("server") + cr
End Function
```

5.6 HTTP Headers Object

5.6.1 HeaderValue

HeaderValue retrieves the value of the Header from the HTTPHeaders Object.

Syntax

HeaderValue (VARIANT index)

Index can be a string or long value. String value will be used if you want to retrieve a value when given the header name (eg. “Server”). HeaderValue can also be retrieved given an index. The valid range for this index is between 0 and the number of headers.

Returns

String : The value of the header.

Example

```
' This script will print out the name of the HTTP server
Function Main
      Dim HTTPObj as Object
      Dim headers as Object
      ip = "www.gfi.org"
      port = 80
      cr = Chr(13) + Chr(10)
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.verb = "HEAD"
      ' Send the HEAD request
      HTTPResponse = HTTPobj.SendRequest ()
      ' Set new Object called headers
      Set headers = HTTPobj.ResponseHeaders
      ' HTTPResponse contains the return code
      echo "Result: " + cstr(HTTPResponse) + cr
      echo "Server running on " + ip + " is " +
      headers.HeaderValue("server") + cr
End Function
```

5.6.2 HeaderName

HeaderName retrieves the name of the header from the HTTPHeader Object.

Syntax

```
HeaderName (LONG index)
```

The valid range for index between 0 and the number of headers.

Returns

String : The name of the header.

Example

```
' This script will print the headers
Function Main
      Dim HTTPObj as Object
      Dim headers as Object
      ip = "www.gfi.com"
      port = 80
```

```
      cr = Chr(13) + Chr(10)
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.verb = "HEAD"
      ' Send the HEAD request
      HTTPResponse = HTTPobj.SendRequest ()
      ' Set new Object called headers
      Set headers = HTTPobj.ResponseHeaders
      ' headers.count contains the number of headers
      (long)
      echo "header count: " & CStr(headers.Count) & cr
      upbound = headers.Count - 1
      ' for each header, echo back the HeaderName and
      Header value
      For hn=0 To upbound
            echo headers.HeaderName(hn) & vbTab & "-->" &
            vbtab & headers.HeaderValue(hn) & cr
      Next
End Function
```

5.6.3 Count Property

Returns the number of header entries in the HTTPHeaders Object.

Syntax

```
HTTPHeadersObject.Count
Count: Long (read)
```

Example

```
' This script will print the headers
Function Main
      Dim HTTPObj as Object
      Dim headers as Object
      ip = "www.gfi.com"
      port = 80
      cr = Chr(13) + Chr(10)
      Set HTTPobj = HTTP.Connect (ip,port)
      ' Set up the request type
      HTTPobj.GetURL("/")
      HTTPobj.verb = "HEAD"
      ' Send the HEAD request
      HTTPResponse = HTTPobj.SendRequest ()
      ' Set new Object called headers
      Set headers = HTTPobj.ResponseHeaders
      ' headers.count contains the number of headers
      (long)
```

```
        echo "header count: " & CStr(headers.Count) & cr
        upbound = headers.Count - 1
        ' for each header, echo back the HeaderName and
        Header value
        For hn=0 To upbound
              echo headers.HeaderName(hn) & vbTab & "-->" &
              vbtab & headers.HeaderValue(hn) & cr
        Next
End Function
```

5.7 FTP Object

The Ftp Object is a collection of functions which make ftp upload / download very simple. The object has the functionality to connect to remote ftp servers, put and retrieve files, rename or delete files. It's also possible to enumerate all the files on the remote server if one so wishes. Another functionality of this object is the ability to retrieve information regarding the files on the server (attributes / size). Apart from all of this, the ftp object can also create / delete folders on the remote server as well as changing of directories.

5.7.1 Connect

Connect is used to determine the hostname or IP address and the port of the FTP server.

Syntax

FTPObject connect (STRING hostname, LONG port, BOOL PassiveMode STRING user, STRING password)

Hostname can be the IP address or the hostname (eg. www.gfi.com)

Port is the port number – an Integer between 1 and 65535

PassiveMode is either TRUE or FALSE. False Sets the mode to Active.

User is the ftp username. For anonymous logon specify username as "anonymous".

Password is the ftp password. For anonymous logon use an e-mail address such as (lnss@gfi.com) as password.

Returns

FTP Object

Example

```
' an example which echoes the current ftp working
directory
Function Main
      Dim FTPobj as Object
      ip = "127.0.0.1"
      port = 21
      mode = FALSE
      username = "anonymous"
      password = "test@lnss.com"
```

```
      Set                                FTPobj=FTP.Connect
      (ip,21,mode,username,password)
      cdir = FTPobj.GetCurrentDirectory
      echo cdir
End Function
```

5.7.2 GetCurrentDirectory

GetCurrentDirectory retrieves current directory on the ftp server. Any file functions (eg. Upload or download) are relative to this directory.

Syntax

```
STRING GetCurrentDirectory()
```

Returns

The current working directory on the ftp server as a string.

Example

```
' an example which echoes the current ftp working
directory
Function Main
      Dim FTPobj as Object
      ip = "127.0.0.1"
      port = 21
      mode = FALSE
      username = "anonymous"
            password = "test@lnss.com"
      ' create a new ftp Connection
      Set                                FTPobj=FTP.Connect
      (ip,21,mode,username,password)
      cdir = FTPobj.GetCurrentDirectory
      echo cdir
End Function
```

5.7.3 SetCurrentDirectory

SetCurrentDirectory sets the directory location on the remote ftp server. Any file functions (eg. Upload or download) are relative to this directory.

Syntax

```
SetCurrentDirectory(STRING directory)
Directory is a string.
```

Returns

Boolean. If it returns TRUE, the function has succeeded, otherwise it means that and error was returned. When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
' an example which Sets the current working directory
Function Main
      Dim FTPobj as Object
      ' configure as needed
```

```
      ip = "127.0.0.1"
      port = 21
      mode = FALSE
      username = "anonymous"
      password = "test@lnss.com"
      directory = "/pub/"
      ' create a new ftp Connection
      Set                              FTPobj=FTP.Connect
      (ip,21,mode,username,password)
      ' Set the current working directory to /pub/
      RET = FTPobj.SetCurrentDirectory (directory)
      if RET Then
            echo "Set current directory to " + directory
            + " succeeded"
            else
            echo "failed to Set current dir: " +
            CStr(FTPobj.LastError)
      End If
End Function
```

5.7.4 CreateDirectory

CreateDirectory creates a new directory on the remote ftp server.

Syntax

CreateDirectory(STRING directory)

Directory is a string.

Returns

Boolean. If it returns TRUE, the function has succeeded, otherwise it means that and error was returned. When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
'an example which echoes the current ftp working
directory
Function Random(N)
      Random = Int(N*Rnd)
End Function
Function Main
      Dim FTPobj as Object
      ' configure as needed
      ip = "127.0.0.1"
      port = 21
      mode = FALSE
      username = "anonymous"
      password = "test@lnss.com"
      cr = Chr(13) + Chr(10)
```

```
' initialise randomization
Randomize
 ‘ now generate a random number to be added to the filenames
for K = 1 to 10
     randomnumber = randomnumber + cstr(Random(10))
next
tempDir = "lnssDir" & randomnumber
' create a new ftp Connection
Set FTPobj=FTP.Connect (ip,21,mode,username,password)
' attempt to create a new directory after an anonymous ftp Connection
if FTPobj.CreateDirectory ( tempDir ) = TRUE then
     echo "Directory create access is available to anonymous ftp at " + ip & cr
     ' now attempt to Delete the directory
     if FTPobj.RemoveDirectory ( tempDir ) = TRUE then
          echo "Directory Delete access is available to anonymous ftp at " + ip & cr
     else
          echo "Directory Delete access is not available. You might need to Delete directories created by GFI LANguard" & cr
     End If
End If
End Function
```

5.7.5 RemoveDirectory

RemoveDirectory creates a new directory on the remote ftp server.

Syntax

```
RemoveDirectory(STRING directory)
```

```
Directory is a string.
```

Returns

Boolean. If it returns TRUE, the function has succeeded, otherwise it means that and error was returned. When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
' an example which echoes the current ftp working directory
Function Random(N)
     Random = Int(N*Rnd)
```

```
End Function
Function Main
      Dim FTPobj as Object
      ' configure as needed
      ip = "127.0.0.1"
      port = 21
      mode = FALSE
      username = "anonymous"
      password = "test@lnss.com"
      cr = Chr(13) + Chr(10)
      ' initialise randomization
      Randomize
      ' now generate a random number to be added to the
      filenames
      for K = 1 to 10
      randomnumber = randomnumber + cstr(Random(10))
      next
      tempDir = "lnssDir" & randomnumber
      ' create a new ftp Connection
      Set                              FTPobj=FTP.Connect
      (ip,21,mode,username,password)
      '  attempt  to  create  a  new  directory  after  an
      anonymous ftp Connection
      if FTPobj.CreateDirectory ( tempDir ) = TRUE then
            echo "Directory create access is available to
            anonymous ftp at " + ip & cr
            ' now attempt to Delete the directory
                  if FTPobj.RemoveDirectory ( tempDir ) =
                  TRUE then
                  echo   "Directory   Delete   access   is
                  available to anonymous ftp at " + ip &
                  cr
            else
                  echo "Directory Delete access is not
                  available. You might need to Delete
                  directories created by GFI LANguard" &
                  cr
            End If
      End If
End Function
```

5.7.6 DeleteFile

Delete file on the remote ftp server.

Syntax

```
DeleteFile(STRING file)
```

File is a string (eg. “readme.txt”)

Returns

Boolean. If it returns TRUE, the function has succeeded, otherwise it means that and error was returned. When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
' an example which uploads a file and deletes it on a
remote ftp server
Function Random(N)
     Random = Int(N*Rnd)
End Function
Function Main
     Dim FTPobj As Object
     Dim fl As Object
     ' configure as needed
     ip = "127.0.0.1"
     port = 21
     mode = FALSE
     username = "anonymous"
     password = "test@lnss.com"
     cr = Chr(13) + Chr(10)
     Set fl = file.Connect("127.0.0.1")
     fl.Open       "testfile.txt",       GENERIC_WRITE,
     CREATE_ALWAYS
     fl.writeline("This is a testfile")
     ' initialise randomization
     Randomize
     fl.Close
     ' now generate a random number to be added to the
     filenames
     For K = 1 To 10
          randomnumber      =      randomnumber      &
          CStr(Random(10))
     Next
     tempFile = "lnssFile" + randomnumber
     ' create a new ftp Connection
     Set                              FTPobj=FTP.Connect
     (ip,21,mode,username,password)
          If FTPobj.PutFile ( "testfile.txt", tempFile
          ) = TRUE Then
               echo "File write access is available to
               anonymous ftp at " + ip & cr
               If FTPobj.DeleteFile ( tempFile ) = TRUE
               Then
```

```
                echo "File Delete access is available to
                anonymous ftp at " + ip& cr
                Else
                echo "File Delete access is not
                available. You might need to Delete
                files created by GFI LANguard" & cr
            End If
        End If
        fl.Delete("testfile.txt")
End Function
```

5.7.7 GetFile

GetFile retrieves a file from the remote machine. The file is then stored locally.

Syntax

```
GetFile(STRING remotefile, String localfile)
RemoteFile is a string (eg. "readme.txt")
LocalFile is a string (eg. "readmecopy.txt")
```

Returns

Boolean. If it returns TRUE, the function has succeeded, otherwise it means that and error was returned. When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
' an example of GetFile function in the FTP Object
' retrieves all files found in the root of the ftp
server.
Function Main
      Dim FTPobj as Object
      Const DIRECTORYMASK=&H10
      ip = "127.0.0.1"
      port = 21
      cr = Chr(13) + Chr(10)
      Set          FTPobj          =          FTP.Connect
      (ip,port,TRUE,"anonymous","lnss@gfi.com")
      Found=FTPobj.FindFirstFile("*")
      While Found
            If      (FTPobj.GetFindFileAttributes      And
            DIRECTORYMASK) = DIRECTORYMASK Then
                  FileType="directory"
            Else
                  FileType="file"
                  ret           =           FTPobj.GetFile
                  (FTPobj.GetFindFileName,
                  FTPobj.GetFindFileName)
            End If
```

```
            echo "File: " + FTPobj.GetFindFileName + "
            size: " + CStr(FTPobj.GetFindFileSize) + "
            bytes type: " + FileType & cr
            Found=FTPobj.FindNextFile
      Wend
End Function
```

5.7.8 PutFile

PutFile uploads a file from the local disk to the remote ftp server.

Syntax

PutFile(STRING localfile, STRING remotefile)

Localfile is a string (eg. “readme.txt”)

Remotefile is a string (eg. “readme.txt”)

Returns

Boolean. If it returns TRUE, the function has succeeded, otherwise it means that and error was returned. When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
' an example which uploads a file and deletes it on a
remote ftp server
Function Random(N)
      Random = Int(N*Rnd)
End Function
Function Main
      Dim FTPobj As Object
      Dim fl As Object
      ' configure as needed
      ip = "127.0.0.1"
      port = 21
      mode = FALSE
      username = "anonymous"
      password = "test@lnss.com"
      cr = Chr(13) + Chr(10)
      Set fl = file.Connect("127.0.0.1")
      fl.Open       "testfile.txt",       GENERIC_WRITE,
      CREATE_ALWAYS
      fl.writeline("This is a testfile")
      ' initialise randomization
      Randomize
      fl.Close
      ' now generate a random number to be added to the
      filenames
      For K = 1 To 10
      randomnumber = randomnumber & CStr(Random(10))
```

```
      Next
      tempFile = "lnssFile" + randomnumber
      ' create a new ftp Connection
      Set                              FTPobj=FTP.Connect
      (ip,21,mode,username,password)
      If FTPobj.PutFile ( "testfile.txt", tempFile ) =
      TRUE Then
            echo "File write access is available to
            anonymous ftp at " + ip & cr
            If FTPobj.DeleteFile ( tempFile ) = TRUE Then
                  echo "File Delete access is available to
                  anonymous ftp at " + ip& cr
            Else
                  echo "File Delete access is not
                  available. You might need to Delete
                  files created by GFI LANguard" & cr
            End If
      End If
      fl.Delete("testfile.txt")
End Function
```

5.7.9 RenameFile

RenameFile renames files on the remote ftp server.

Syntax

RenameFile(STRING originalFileName, STRING renamedFileName)

originalFileName is a string.

renamedFileName is a string.

Returns

Boolean. If it returns TRUE, the function has succeeded, otherwise it means that and error was returned. When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
' an example of RenameFile function in the FTP Object
' renames all files found in the root of the ftp server.
Function Main
      Dim FTPobj as Object
      Const DIRECTORYMASK=&H10
      ip = "127.0.0.1"
      port = 21
      cr = Chr(13) + Chr(10)
      Set          FTPobj          =          FTP.Connect
      (ip,port,TRUE,"anonymous","lnss@gfi.com")
      Found=FTPobj.FindFirstFile("*")
      While Found
```

```
            If      (FTPobj.GetFindFileAttributes      And
            DIRECTORYMASK) = DIRECTORYMASK Then
                  FileType="directory"
            Else
                  FileType="file"
                  FileName = FTPobj.GetFindFileName
                  RenameFileName     =     "renamed_"     +
                  FTPobj.GetFindFileName
                  ret   =   FTPobj.RenameFile   (FileName,
                  RenameFileName)
            End If
            echo  "File:  "  +  FTPobj.GetFindFileName  +  "
            size:  "  +  CStr(FTPobj.GetFindFileSize)  +  "
            bytes type: " + FileType & cr
            Found=FTPobj.FindNextFile
      Wend
End Function
```

5.7.10 FindFirstFile

FindFirstFile initiates and enumeration of files and directories in the current directory on the remote ftp server.

Syntax

FindFirstFile(STRING filemask)

Filemask is a string. Usually this would be “*” to enumerate all files.

Returns

Boolean. If it returns TRUE, that means that at least one file on the remote ftp server matched.

File name and file size for the first matching file can be retrieved using GetFindFileName()/GetFindFileSize() methods. FindNextFile() method is used to move to next matching file.

FindFirstFile will returns FALSE in case no matching files were found.

FindFirstFile will also returns FALSE on subsequent calls to FindFirstFile() if current search operation has not been Closed with FindFileClose() method.

When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
' an example of RenameFile function in the FTP Object
' renames all files found in the root of the ftp server.
Function Main
      Dim FTPobj as Object
      Const DIRECTORYMASK=&H10
      ip = "127.0.0.1"
      port = 21
      cr = Chr(13) + Chr(10)
```

```
      Set          FTPobj          =          FTP.Connect
      (ip,port,TRUE,"anonymous","lnss@gfi.com")
      Found=FTPobj.FindFirstFile("*")
      While Found
            If      (FTPobj.GetFindFileAttributes      And
            DIRECTORYMASK) = DIRECTORYMASK Then
                  FileType="directory"
            Else
                  FileType="file"
                  FileName = FTPobj.GetFindFileName
                  RenameFileName     =     "renamed_"     +
                  FTPobj.GetFindFileName
                  ret    =    FTPobj.RenameFile    (FileName,
                  RenameFileName)
            End If
            echo  "File:  "  +  FTPobj.GetFindFileName  +  "
            size:  "  +  CStr(FTPobj.GetFindFileSize)  +  "
            bytes type: " + FileType & cr
            Found=FTPobj.FindNextFile
      Wend
End Function
```

5.7.11 FindNextFile

Searches for the next file matching the filemask specified by the FindFirstFile method.

Syntax

FindNextFile

Returns

Boolean. If it returns TRUE, that means that more files were found which match the filemask specified by the FindFirstFile method. File name and file size for the first matching file can be retrieved using GetFindFileName()/GetFindFileSize() methods. FindNextFile will returns FALSE in case no matching files were found.FindNextFile must be called in-between a successful call to FindFirstFile() and a call to FindFileClose(). The method will return FALSE if called outside this scope.

When FALSE is returned, FTPObject.LastError will return the WIN32 error code.

Example

```
' an example of RenameFile function in the FTP Object
' renames all files found in the root of the ftp server.
Function Main
      Dim FTPobj as Object
      Const DIRECTORYMASK=&H10
      ip = "127.0.0.1"
      port = 21
      cr = Chr(13) + Chr(10)
```

```
	Set FTPobj = FTP.Connect
	(ip,port,TRUE,"anonymous","lnss@gfi.com")
	Found=FTPobj.FindFirstFile("*")
	While Found
		If (FTPobj.GetFindFileAttributes And
		DIRECTORYMASK) = DIRECTORYMASK Then
			FileType="directory"
		Else
			FileType="file"
			FileName = FTPobj.GetFindFileName
			RenameFileName = "renamed_" +
			FTPobj.GetFindFileName
			ret = FTPobj.RenameFile (FileName,
			RenameFileName)
		End If
		echo "File: " + FTPobj.GetFindFileName + "
		size: " + CStr(FTPobj.GetFindFileSize) + "
		bytes type: " + FileType & cr
		Found=FTPobj.FindNextFile
	Wend
End Function
```

5.7.12 FindFileClose

Searches for the next file matching the filemask specified by the FindFirstFile method. There is no need to call this if call to FindFirstFile() failed.

Syntax

FindFileClose

Returns

No data returned.

Example

```
' an example of FindFileClose function in the FTP Object
' searches for a certain file until found in the root.
Function Main
	Dim FTPobj as Object
	Const DIRECTORYMASK=&H10
	ip = "127.0.0.1"
	port = 21
	cr = Chr(13) + Chr(10)
	Set FTPobj = FTP.Connect
	(ip,port,TRUE,"anonymous","lnss@gfi.com")
	Found=FTPobj.FindFirstFile("*")
	While Found
		If (FTPobj.GetFindFileAttributes And
		DIRECTORYMASK) = DIRECTORYMASK Then
```

```
                FileType="directory"
                Found=FTPobj.FindNextFile
                Else
                FileType="file"
                if  FTPobj.GetFindFileName  =  "test.zip"
                then
                echo "test.zip exists" & cr
                FTPobj.FindFileClose
                Found = false
                else
                echo "test.zip does not exist" & cr
                Found=FTPobj.FindNextFile
                End If
           End If
     Wend
End Function
```

5.7.13 *GetFindFileName*

GetFindFileName retrieves the filename of the currently matched file after a successful call to either FindFirstFile or FindNextFile methods.

When FindFileClose is called, GetFindFileName, GetFindFileSize and GetFindFileAttributes should not be used since this will make the Scripting engine fail.

Syntax

GetFindFileName

Returns

The name of the file. This is a string.

Example

```
' an example of RenameFile function in the FTP Object
' renames all files found in the root of the ftp server.
Function Main
     Dim FTPobj as Object
     Const DIRECTORYMASK=&H10
     ip = "127.0.0.1"
     port = 21
     Set          FTPobj          =          FTP.Connect
     (ip,port,TRUE,"anonymous","lnss@gfi.com")
     Found=FTPobj.FindFirstFile("*")
     While Found
          If      (FTPobj.GetFindFileAttributes      And
          DIRECTORYMASK) = DIRECTORYMASK Then
               FileType="directory"
               Else
               FileType="file"
```

```
            FileName = FTPobj.GetFindFileName
            RenameFileName = "renamed_" + FTPobj.GetFindFileName
            ret = FTPobj.RenameFile (FileName, RenameFileName)
        End If
    Wend
End Function
```

5.7.14 GetFindFileSize

GetFindFileSize retrieves the file size of the currently matched file after a successful call to either FindFirstFile or FindNextFile methods.

When FindFileClose is called, GetFindFileName, GetFindFileSize and GetFindFileAttributes should not be used since this will make the Scripting engine fail.

Syntax

GetFileSize

Returns

File size of the currently matched file. Long Integer.

Example

```
' an example of RenameFile function in the FTP Object
' renames all files found in the root of the ftp server.
Function Main
    Dim FTPobj as Object
    Const DIRECTORYMASK=&H10
    ip = "127.0.0.1"
    port = 21
    cr = Chr(13) + Chr(10)
    Set FTPobj = FTP.Connect (ip,port,TRUE,"anonymous","lnss@gfi.com")
    Found=FTPobj.FindFirstFile("*")
    While Found
        If (FTPobj.GetFindFileAttributes And DIRECTORYMASK) = DIRECTORYMASK Then
            FileType="directory"
        Else
            FileType="file"
        End If
        echo "File: " + FTPobj.GetFindFileName + " size: " + CStr(FTPobj.GetFindFileSize) + " bytes type: " + FileType & cr
        Found=FTPobj.FindNextFile
    Wend
End Function
```

5.7.15 GetFindFileAttributes

GetFindFileAttributes retrieves the file Attributes of the currently matched file after a successful call to either FindFirstFile or FindNextFile methods.

When FindFileClose is called, GetFindFileName, GetFindFileSize and GetFindFileAttributes should not be used since this will make the Scripting engine fail.

Syntax

GetFindFileAttributes

Returns

File attributes of currently matched file. These are the attributes from dwFileAttributes member in WIN32 defined structure WIN32_FIND_DATA. Bit masks are defined as FILE_ATTRUTE_* constants. I.e. FILE_ATTRUTE_DIRECTORY is defined as 0x10.

FILE_ATTRIBUTE_READONLY	&H1
FILE_ATTRIBUTE_HIDDEN	&H2
FILE_ATTRIBUTE_SYSTEM	&H4
FILE_ATTRIBUTE_DIRECTORY	&H10
FILE_ATTRIBUTE_ARCHIVE	&H20
FILE_ATTRIBUTE_DEVICE	&H40
FILE_ATTRIBUTE_NORMAL	&H80
FILE_ATTRIBUTE_TEMPORARY	&H100
FILE_ATTRIBUTE_SPARSE_FILE	&H200
FILE_ATTRIBUTE_REPARSE_POINT	&H400
FILE_ATTRIBUTE_COMPRESSED	&H800
FILE_ATTRIBUTE_OFFLINE	&H1000
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	&H2000
FILE_ATTRIBUTE_ENCRYPTED	&H4000

Example

```
' an example of RenameFile function in the FTP Object
' renames all files found in the root of the ftp server.
Function Main
      Dim FTPobj as Object
      Const DIRECTORYMASK=&H10
      ip = "127.0.0.1"
      port = 21
      Set          FTPobj          =          FTP.Connect
      (ip,port,TRUE,"anonymous","lnss@gfi.com")
      Found=FTPobj.FindFirstFile("*")
      While Found
            If      (FTPobj.GetFindFileAttributes      And
            DIRECTORYMASK) = DIRECTORYMASK Then
                  FileType="directory"
            Else
```

```
            FileType="file"
        End If
        echo "File: " + FTPobj.GetFindFileName + "
        size: " + CStr(FTPobj.GetFindFileSize) + "
        bytes type: " + FileType
        Found=FTPobj.FindNextFile
    Wend
End Function
```

5.7.16 LastError Property

LastError contains the WIN32 error code set by various methods when these return a FALSE and fail. This value should be checked if necessary before calling another method that can set this property in case of error.

Syntax

FTPobj.LastError: STRING (read)

Example

```
' an example which Sets the current working directory
Function Main
    Dim FTPobj as Object
    ' configure as needed
    ip = "127.0.0.1"
    port = 21
    mode = FALSE
    username = "anonymous"
    password = "test@lnss.com"
    directory = "/pub/"
    ' create a new ftp Connection
    Set                              FTPobj=FTP.Connect
    (ip,21,mode,username,password)
    ' Set the current working directory to /pub/
    RET = FTPobj.SetCurrentDirectory (directory)
    if RET Then
        echo "Set current directory to " + directory
        + " succeeded"
    else
        echo "failed to Set current dir: " +
        CStr(FTPobj.LastError)
    End If
End Function
```

5.8 Encode Object:

The encode object offers the user the ability to encode or decode base 64 strings. This can come in handy in various applications. Base64 encoding is heavily used in emails as well as various authentication schemes including http.

5.8.1 Base64Encode

Base64Encode is used to encode a string into Base64 representation

Syntax

Base64Encode(String)

Returns

String

Example

```
Function Main
      Dim message As String
      Dim encoded As String
      Dim decoded As String
      cr = Chr(13)+Chr(10)'Carriage return and line feed
      message = "String to be encoded"
      encoded = Encode.Base64Encode(message)
      echo "Encoded Text : "
      echo encoded
      echo cr
      decoded = Encode.Base64Decode(encoded)
      echo "Decoded Text :"+decoded+cr
End Function
```

5.8.2 Base64Decode

Base64Decode is used to decode a Base64 Representation string into its original format

Syntax

Base64Decode(String)

Returns

String

Example

```
Function Main
      Dim message As String
      Dim encoded As String
      Dim decoded As String
      cr = Chr(13) + Chr(10)'Carriage return and line
      feed
      message = "String to be encoded"
      encoded = Encode.Base64Encode(message)
      echo "Encoded Text : "
      echo encoded
      echo cr
      decoded = Encode.Base64Decode(encoded)
      echo "Decoded Text :"+decoded+cr
End Function
```

6. General Functions

6.1 List of functions

6.1.1 Echo

Echo is a simple function that displays output

Syntax

Echo (String)

Returns

No data returned.

Example

```
'This example displays the word Test
Function Main
      echo "test"
End Function
```

6.1.2 WriteToLog

Writetolog will write any string passed to it, in the scripting engine log file

Syntax

WriteToLog(String)

Returns

No data returned.

Example

```
Function Main
      WritetoLog "test"
End Function
```

6.1.3 StatusBar

StatusBar is used to display a string in the status bar of the current active component

Syntax

StatusBar(String)

Returns

No data returned.

Example:

```
Function Main
      StatusBar "test"
End Function
```

6.1.4 AddListItem

AddListItem is a function which allows scripts to return feedback to the user. This function will add any string passed to it as a sub node of the triggered vulnerability. The AddListItem function takes 2 different parameters. The first parameter specifies the parent node and the second parameter, the string to be added to the tree. If the parent node is left empty, the function will add the specified string to the top available node (the vulnerability parent node). The tree can only have 1 level though even though it can have as many siblings as required.

Syntax

```
AddListItem(String,String)
```

Returns

```
N/A
```

Example

```
Function MAIN
      Dim wmi As Object
      Dim objset As Object
      Dim obj As Object
      Dim monitor As Object
      Dim prop As Object
      Set                        wmi                        =
      GetObject("winmgmts:\\127.0.0.1\root\cimv2")
      Set objset = wmi.instancesof("Win32_service")
      For Each obj In objset
            Set monitor = obj
            For Each prop In monitor.properties_
                  If VarType(prop.value) = 8 Then
                  If Not (IsNull(prop.value)) Then
                  If prop.name = "Name" Then
                  If left(prop.value,1) = "a" then
                  AddListItem("A",prop.value)
                  End If
                  If left(prop.value,1) = "b" then
                  AddListItem("B",prop.value)
                  End If
                  If left(prop.value,1) = "c" Then
                  AddListItem("C",prop.value)
                  End if
                  End If
                  End If
                  End If
            Next
      Next
      main = true
```

```
End Function
```

6.1.5 SetDescription

SetDescription is used to return simple feedback to the user by means of programmatically changing the vulnerability description to indicate a more detailed reason for the vulnerability trigger. SetDescription takes only one parameter. The string passed to the function will be set as the new description for the vulnerability once it is triggered.

Syntax

```
SetDescription(String)
```

Returns

```
N/A
```

Example

```
Function Main
      SetDescription ("This New description will be set
      in place of the one specified in the
      vulnerability")
      Main=true
End Function
```

7. Using ActiveX, COM and OLE Automation components

7.1 Introduction to using automation objects

One of the major inherited advantages of the new scripting engine is that now the user has the power of a programming language at his disposal to achieve his goals. Programming possibilities are endless.

In order to bind with these automations we use the function CreateObject. This function returns an Object that links to the automation we want to use. For Example if I want to use Microsoft's © Regular Expression Object I would do the following

Example

```
Function Main
      Dim regexp As Object
      Dim test As String
      Set regexp = CreateObject("VBScript.RegExp")
      regexp.pattern = "[A-Za-z]:(\\[A-Za-z0-9_.-])*"
      test = "c:\windows\"
      If regexp.test(test) Then
            echo("This is a path")
      Else
            echo("This is not a path")
      End If
End Function
```

The above example uses regular expression to check if the variable test holds a path or not.

The flexibility of this system not only allows using these objects to enhance GFI LANguard scripts, but also allows the scripting engine to be used for any needed purpose. Example, It is now possible to create a script to scan for signs of a **Worm X**, Clean it if found and also generate a report in Excel as reference of the changes made, while displaying in GFI LANguard that a worm was found and cleaned successfully.

Below is an example of such automation, the script lists the services running on the local machine and their status into an excel worksheet. This particular example requires Excel to be installed and also the availability of WMI which should be pre-installed on Windows 2000 machine upwards.

7.1.1 Sample automation object usage script

'This Script Connects with excel automation Object, creates a new sheet and exports to it the list of services and their respective status'This script requires Excel and also WMI.

NOTE: Windows Management Instrumentation, come pre-installed on Windows 2000 upwards. It must be installed on Windows 9x and Windows NT. download link: http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=AFE41F46-E213-4CBF-9C5B-FBF236E0E875

```
Function Main
      Dim excel As Object 'Creation of objects needed by
      the script
      Dim book As Object
      Dim sheet As Object
      Dim range As Object
      Dim columns As Object
      Dim wmiobj As Object
      Dim objswbemobject As Object
      strComputer = "127.0.0.1"
      Set wmiobj = GetObject("winmgmts:\\" & strComputer
      & "\root\cimv2")
      'Getting a wmi Object which allows retrieval of
      various information
      If wmiobj is Nothing Then
            echo ("Error1 Unable to create WMI Object")
            Else
            Set excel = CreateObject("Excel.application")
            'Connecting to the Excel Automation Object
            If excel is Nothing Then
                  Echo("Unable to create Excel Object")
                  Else
                  echo ("initalised session with Excel
                  Version "&excel.version) 'display
                  excel's Version
                  Set book = excel.workbooks.add 'Add
                  workbook
                  Set sheet = Book.Worksheets.add 'Add
                  worksheet to workbook
                  sheet.cells(1,1) = "This Sheet has been
                  generated from with a GFI LANguard
                  Script"
                  sheet.cells(3,1) = "Service Name" 'Setup
                  Column names
                  sheet.cells(3,2) = "State"
                  sheet.cells(3,3) = "Started"
                  Set
                  wmiinst=wmiobj.instancesof("Win32_Servic
                  e") 'Retrieve Services info
                  If wmiinst is Nothing Then
```

```
                    echo ("error2: Unable to retrieve
                    services information")
                    Else
                    lnpos = 4
                    For Each objswbemobject In wmiinst
                    'Loop through all services objects
                    lnpos = lnpos + 1
                    sheet.cells(lnpos,1)                    =
                    objswbemobject.DisplayName
                    'Enter services info into the excel
                    sheet
                    sheet.cells(lnpos,2)                    =
                    objswbemobject.State
                    sheet.cells(lnpos,3)                    =
                    objswbemobject.Started
                    Next
                    sheet.columns.AutoFit 'Auto fit Columns
                    sheet.application.visible     =     true
                    'Display the excel sheet
                    End If
               End If
          End If
End Function
```

8. Using Libraries and code reusability

Another advantage of the new scripting language is its ability to use libraries in scripts. This allows you to create libraries with their most used function and than simply import the library in all the scripts you wish to use the functions in.

8.1 Creating libraries

Libraries are simple scripts themselves which usually contain a number of functions. Libraries as opposed to normal scripts should not have a main function. If a main function is defined in a library, a duplicate declaration error will occur if the script that uses the function has a main function as well, or if the script does not have a main function, than the main function of the library will be called first.

Libraries should be put in the configured Library directory. This generally is <LANguard main directory>\library

It is also possible to place the library in a sub directory, but only under the configured library directory.

8.2 Using libraries

In order to call functions in libraries, first the library needs to be included in the script you are doing. This is done by using the include directive. The include directive is used in the following manner. First you have to put a 2 character combination " ' **#** ". Than just write the word include after the # and the library name between two double quotes.

Example

```
' #include "mylibrary"
```

This will sort of virtually paste the code in mylibrary at the start of the script and thus all function in mylibrary will become available the script that is currently being developed.

Example

'This is the library (saved in a filename called "library")

```
Function Ver
      Ver = "1.0"
End Function
```

Above is a library that contains a single function called Ver that returns a string "1.0"

```
' This is the script that uses the library we declared above
' #include "library"
Function Main
      Dim libraryver As String
      libraryver = Ver()
      echo libraryver
End Function
```

This script simply uses the function stored in the library to retrieve a string that it than displays.

Index