



GFI Product Manual

GFI LanGuard™

Scripting manual



<http://www.gfi.com>

info@gfi.com

The information and content in this document is provided for informational purposes only and is provided "as is" with no warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. GFI Software is not liable for any damages, including any consequential damages, of any kind that may result from the use of this document. The information is obtained from publicly available sources. Though reasonable effort has been made to ensure the accuracy of the data provided, GFI makes no claim, promise or guarantee about the completeness, accuracy, recency or adequacy of information and is not responsible for misprints, out-of-date information, or errors. GFI makes no warranty, express or implied, and assumes no legal liability or responsibility for the accuracy or completeness of any information contained in this document.

If you believe there are any factual errors in this document, please contact us and we will review your concerns as soon as practical.

All product and company names herein may be trademarks of their respective owners.

GFI LanGuard 2011 is copyright of GFI SOFTWARE Ltd. - 1999-2011 GFI Software Ltd. All rights reserved.

Last updated: 16 August 2011

Version: LANSS-SCP-EN-02.00.02

Table of contents

1	Introduction	1
2	Visual Basic Scripting	3
2.1	Scripting Editor/Debugger	3
2.2	Important standards to follow	4
2.3	Native VBScript Functions supported in GFI LanGuard	4
2.4	Common mistakes and pitfalls	5
2.5	Tips & Tricks.....	6
2.6	Developing a script in the GFI LanGuard debugger	6
2.7	Adding a new scripting based vulnerability check to the scanner tool ...	10
3	Python Scripting	13
3.1	What is the Python programming language?	13
3.2	Creating a new vulnerability check using Python Script Test	14
3.3	Application Programming Interfaces (APIs) available in Python Scripts ..	16
3.4	Debugging Python scripts	18
3.5	Using Libraries and code reusability	18
4	Object Documentation	21
4.1	Socket Object	21
4.2	SNMP Object.....	27
4.3	File Object.....	30
4.4	Registry Object.....	37
4.5	HTTP Object.....	42
4.6	HTTP Headers Object	52
4.7	FTP Object.....	54
4.8	Encode Object.....	67
5	General Functions	69
5.1	About this manual	69
6	Using ActiveX, COM and OLE Automation components	73
6.1	Introduction to using automation objects.....	73
7	Using Libraries and code reusability	75
7.1	Creating libraries	75
7.2	Using libraries	75
	Index	77

List of screenshots

Screenshot 1 Editor Debugger	3
Screenshot 2 Error produced as a result of not declaring an object before it is used	6
Screenshot 3 Breakpoints & line by line execution of the script	7
Screenshot 4 Watches	8
Screenshot 5 Alternative credentials	8
Screenshot 6 Parameters Dialog	9
Screenshot 7 GetParameter function	10
Screenshot 8 Add vulnerability check	11
Screenshot 9 Specify Python script.	12
Screenshot 10 Vulnerability detection.	12
Screenshot 11 Python Script Test.	13
Screenshot 12 Add vulnerability.	15
Screenshot 13 Specify Python script.	16

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 (the scripts are run on the scanned machine):

- » **UNIX shell scripts run through SSH:** The remote machine must be a UNIX machine and allow remote connections via SSH.

Platform independent (the scripts are run on the machine where GFI LanGuard is installed):

- » **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.

2 Visual Basic Scripting

A complete language reference for the Visual Basic programming language used by GFI LanGuard is available here:

<http://www.cypressinc.com/enduser.pdf>

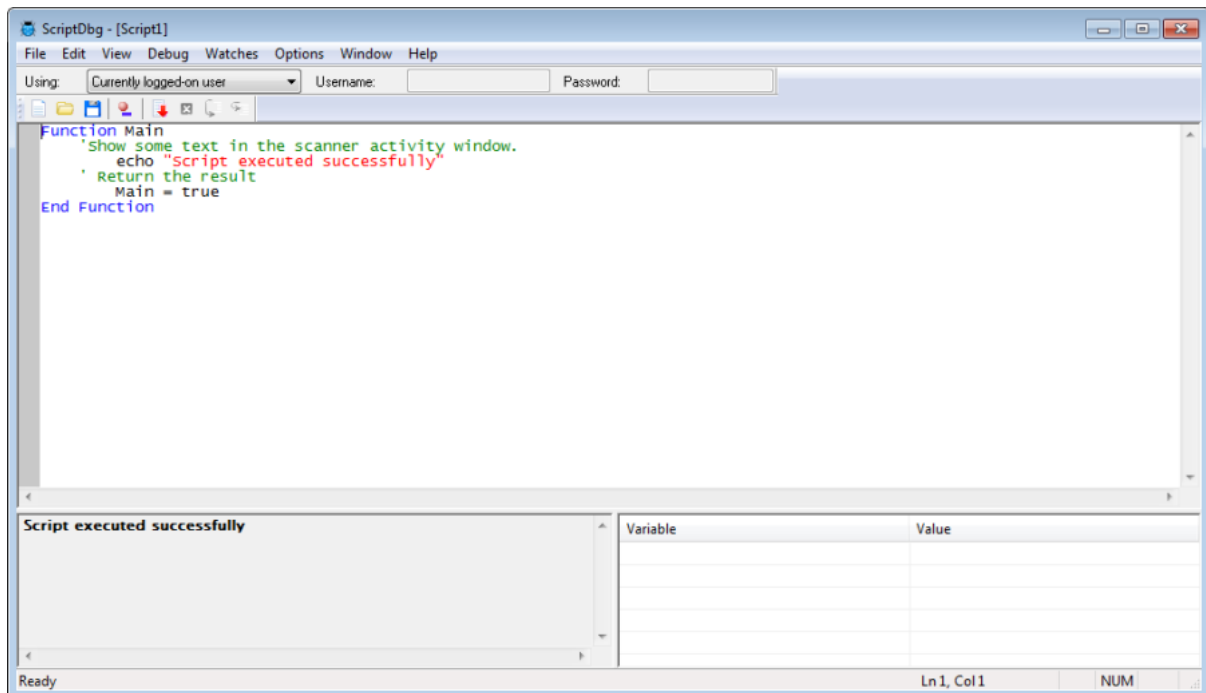
2.1 Scripting Editor/Debugger

The benefit of using an editor is that it is safer and faster to correct and create working scripts. Using an editor will ensure that the script is functional prior to running the script in a working environment. GFI LanGuard editor/debugger enables you to run the script as if in a working environment. The GFI LanGuard editor/debugger has all the common functionality found in an integrated development environment, amongst others it includes:

- » breakpoints
- » step into
- » step over
- » enables you to specify parameters for proper testing.



Running the script in GFI LanGuard without debugging is not recommended. If the script is not functional you will not be able to identify why the script failed. In addition, improper coding can lead to infinite loops which can stall GFI LanGuard scanner.



Screenshot 1 Editor Debugger

2.1.1 Feature list of the GFI LanGuard debugger

- » **Support for variable watches** - Monitor in real time the changing variable values used.
- » **Step into/step over functionality** - Enables you 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.

2.2 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 a Boolean (true or false). The return value is specified by assigning the result value to a variable which has the same name of the function name (Example: 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.

Example

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

2.3 Native VBScript Functions supported in GFI LanGuard



ALL VBScript functions and scripting methods are 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>

2.4 Common mistakes and pitfalls

In VBScript there are two types of variables:

- » **Simple Type** - variables are true types like integer, Boolean, string.
- » **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.



It is recommended you assign a value to all variables/objects a type.

Example

```
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
```

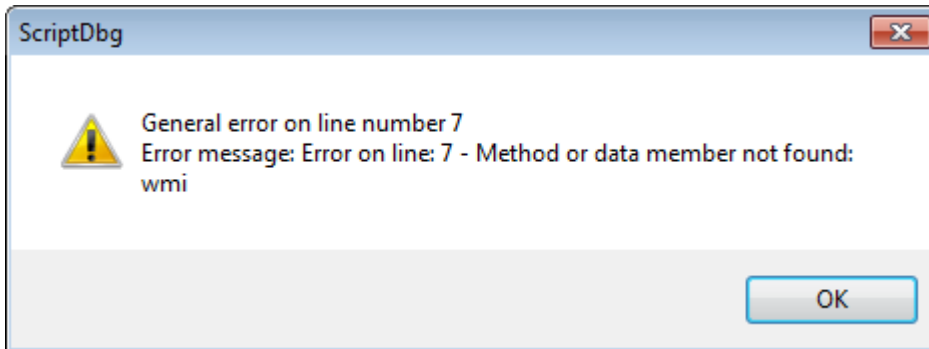


If you try to use an automation object without declaration, the script will fail while compiling . As an example consider the same piece of code but with a missing full declaration of the object variable **wmi**. The instant you try to run the script you will be presented with an error message as well 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

2.5 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.

Example

```

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

```

2.6 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

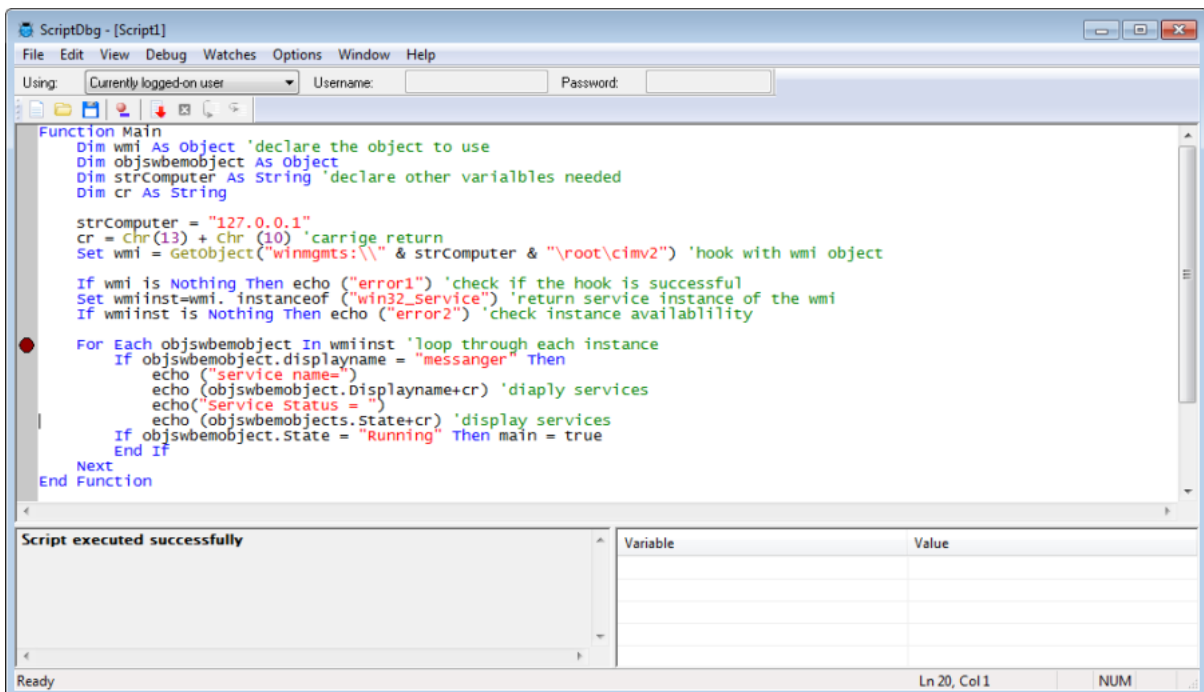
```

2.6.1 Running a script

To run the script, press **F5** or click **Debug ► Go**.

2.6.2 Debugging breakpoints / Step In functionality

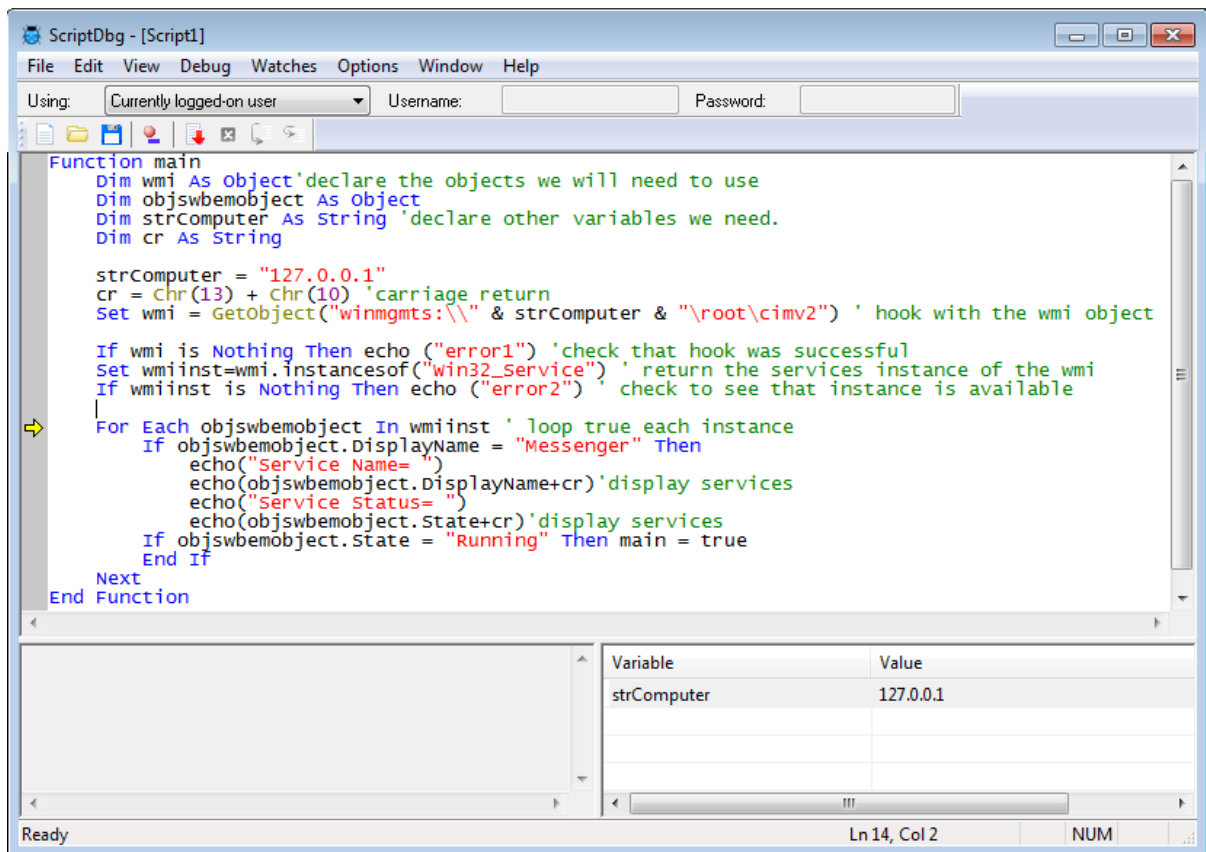
These editor/debugger features enables you to pause the execution at a specific position in the script. For example, you can set a break point to execute when the variable containing the display name of the service is “Messenger”. To do this in the example script, you would go to line 17 (“echo(“Service 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 3 Breakpoints & line by line execution of the script

2.6.3 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.

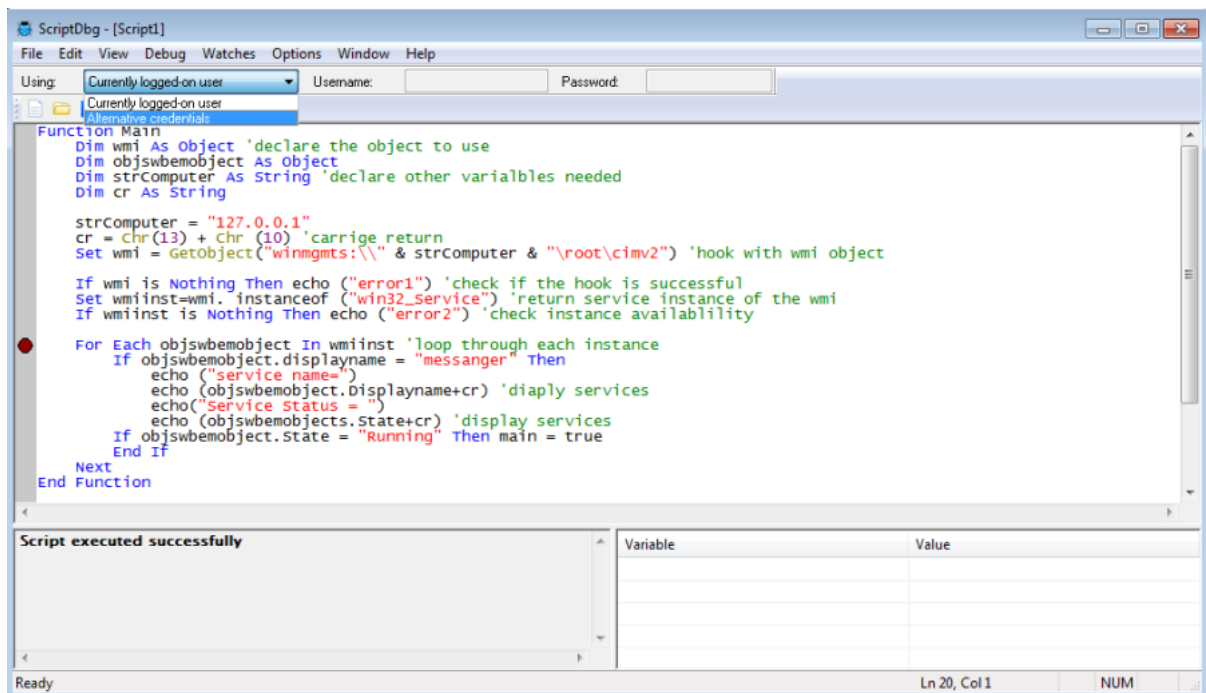


Screenshot 4 Watches

2.6.4 Debugging under alternative user sessions

To test the script under alternative credentials:

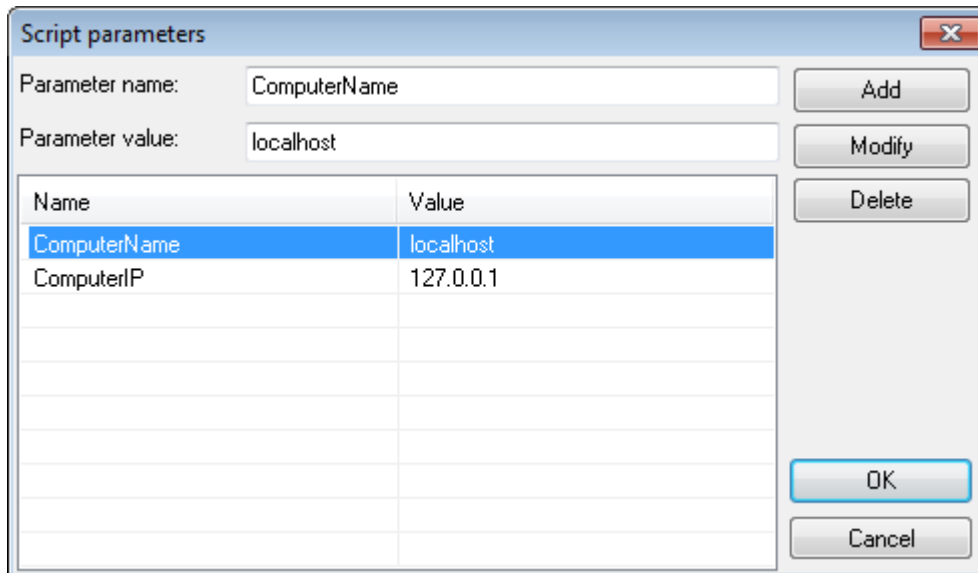
1. Launch GFI LanGuard Script Debugger
2. In the Using drop down list, select Alternative Credentials.
3. Key-in a user name and password in the provided textboxes.



Screenshot 5 Alternative credentials

2.6.5 Sending Parameters to the script

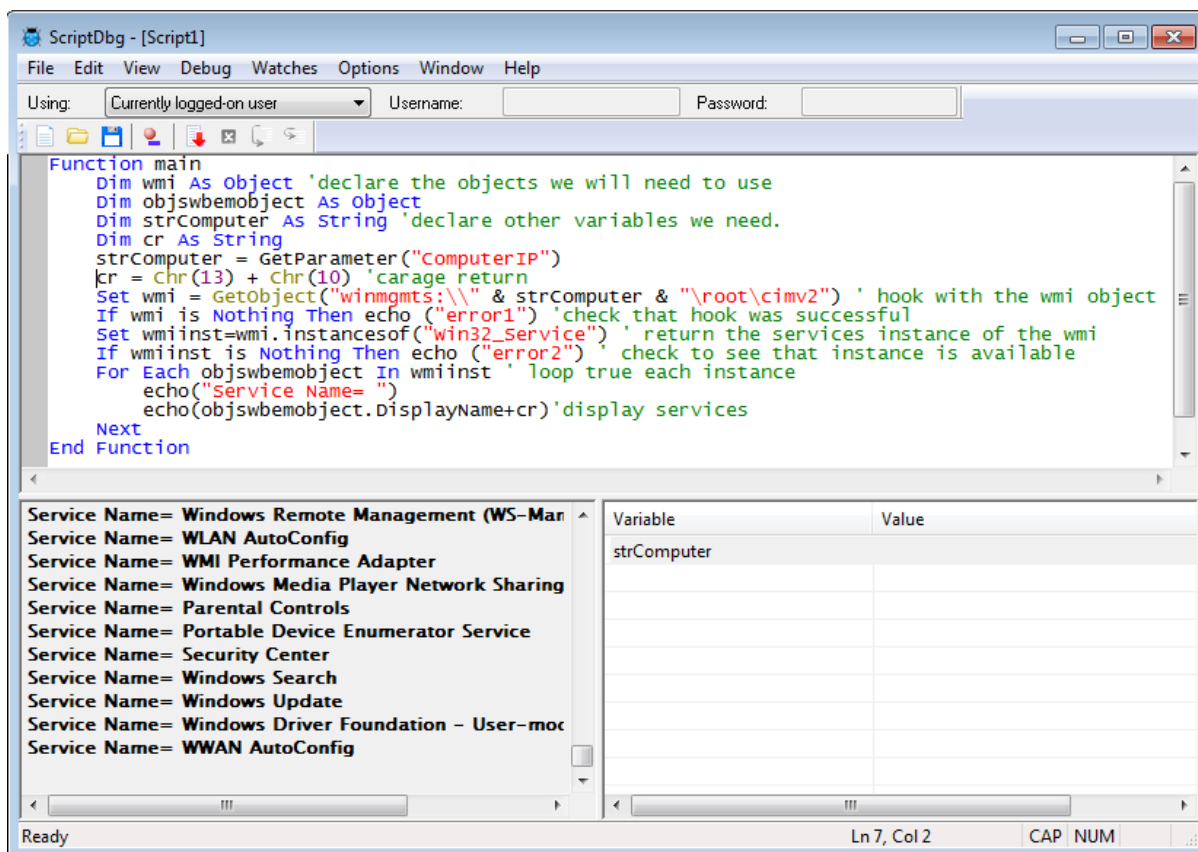
GFI LanGuard scanner tool passes parameters to the scripts when executed. **Example**, the computer name and computer IP of the target machine being scanned for vulnerabilities. To be able to debug your scripts you may want to test with various types of values for these parameters. You can specify alternative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



Screenshot 6 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 example:

```
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 7 GetParameter function

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

The following example describes how to create a new vulnerability check and will run a script to check for the new vulnerability. The script displays **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.

2.7.1 Step 1: Creating the script

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

2. Click **File ► New**

3. Paste the following text in the debugger:

```

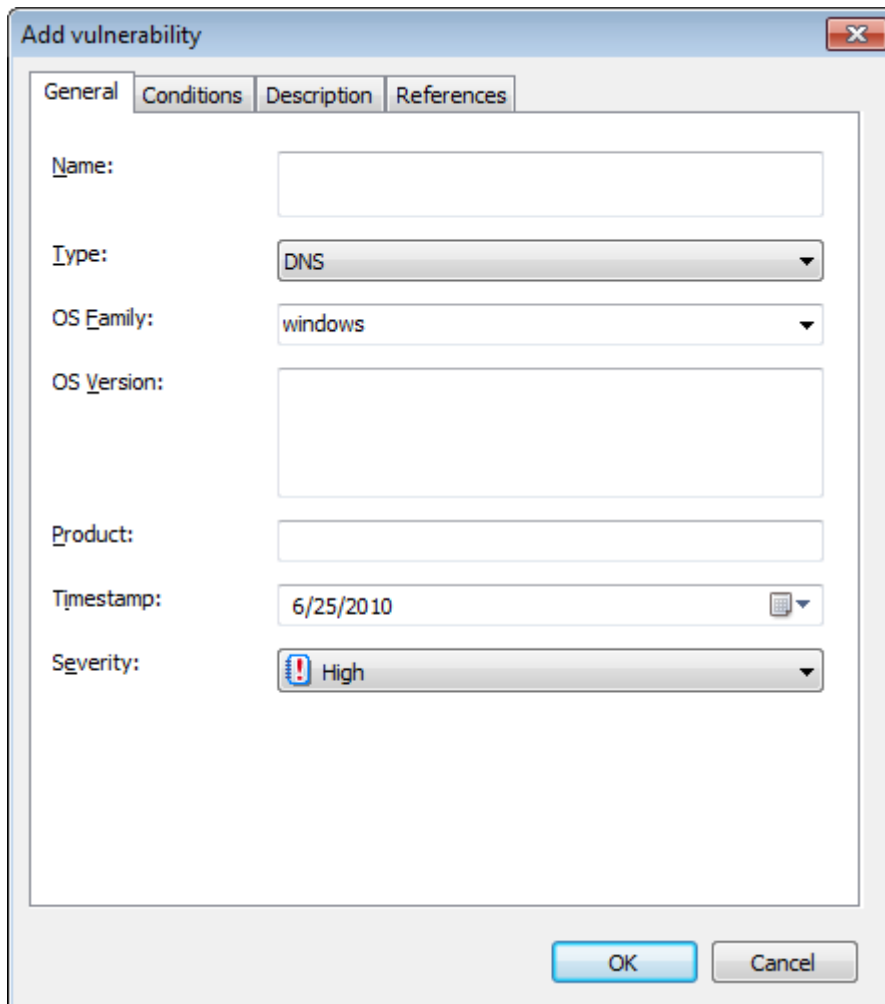
Function Main
    echo "Script has run successfully"
    Main = true
End Function

```

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

2.7.2 Step 2: Creating the new vulnerability check

1. Launch GFI LanGuard management console.
2. From GFI LanGuard button, click **Configure ► Scanning Profiles Editor**
3. Click **New scanning profile** or select the scanning profile to edit
4. Select **Vulnerability Assessment Options** tab ► **Vulnerabilities** ► **Add**.



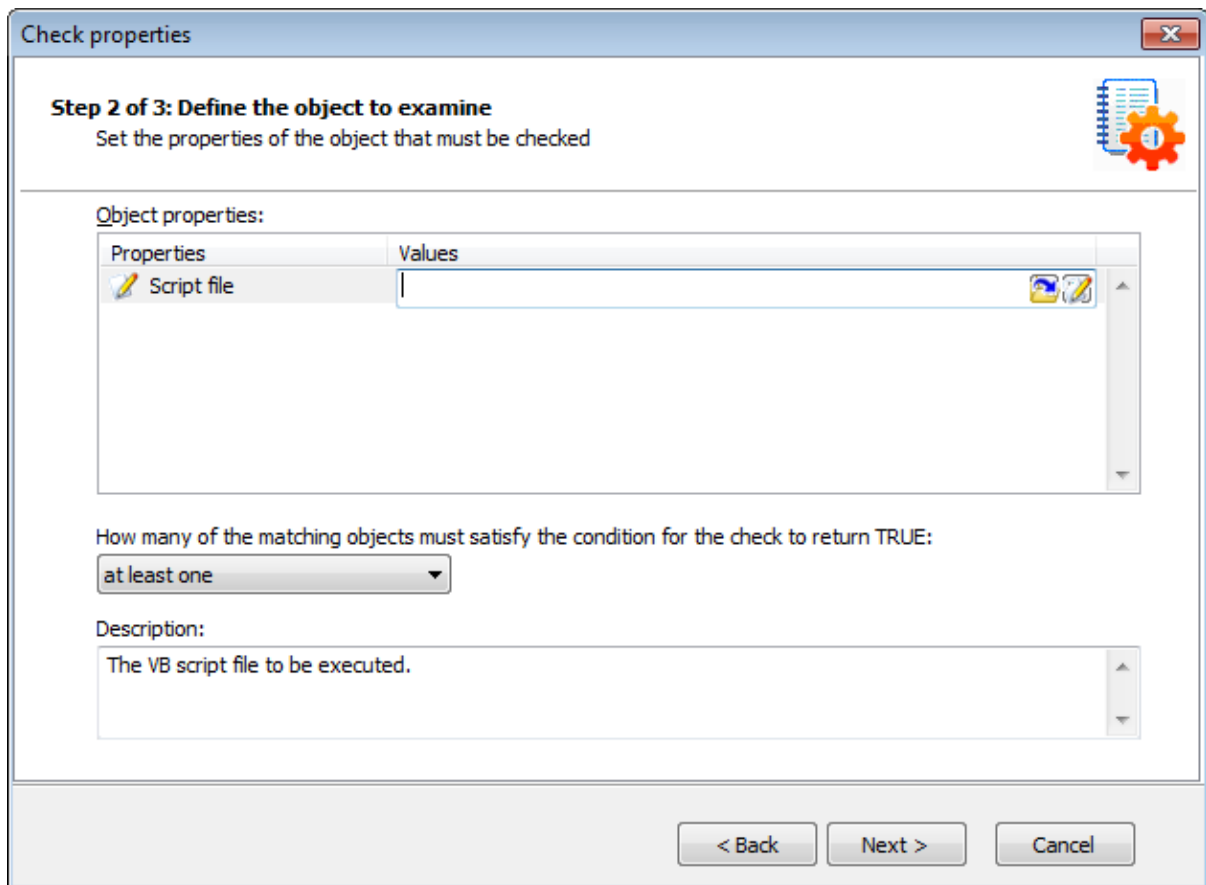
The screenshot shows a dialog box titled "Add vulnerability" with a close button (X) in the top right corner. The dialog has four tabs: "General", "Conditions", "Description", and "References". The "General" tab is selected. The fields are as follows:

- Name:** An empty text input field.
- Type:** A dropdown menu with "DNS" selected.
- OS Family:** A dropdown menu with "windows" selected.
- OS Version:** An empty text input field.
- Product:** An empty text input field.
- Timestamp:** A date picker showing "6/25/2010".
- Severity:** A dropdown menu with "High" selected, indicated by a red exclamation mark icon.

At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

Screenshot 8 Add vulnerability check

5. Enter the basic details including name etc,
6. Select the **Conditions** tab and select the **Add** button.
7. Set the check type to **Independent Checks ► VB Script Test**.
8. Specify the location of the script
(**Example**"<ProductDataDir>\Scripts\newvulnerability.vbs"). Click **Next**.

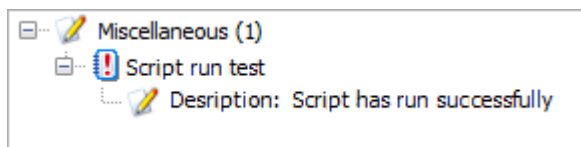


Screenshot 9 Specify Python script.

9. In the **Value** edit box specify the value returned by the VB 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.

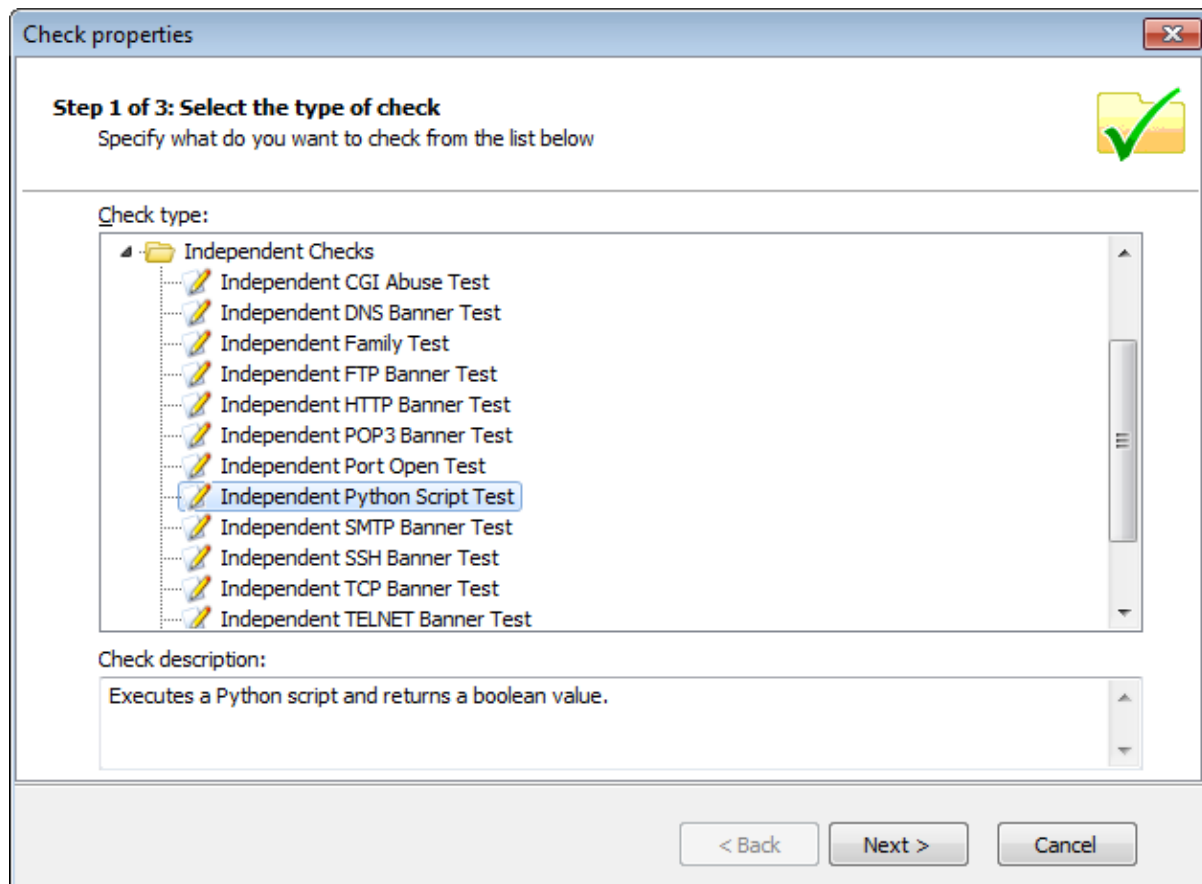
To test it out, simply scan your localhost machine (127.0.0.1) and you should see the vulnerability warning under the miscellaneous section of the vulnerabilities node of the scan results.



Screenshot 10 Vulnerability detection.

3 Python Scripting

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 11 Python Script Test.

3.1 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)
```

3.2 Creating a new vulnerability check using Python Script Test

3.2.1 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)
```

3.2.2 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.

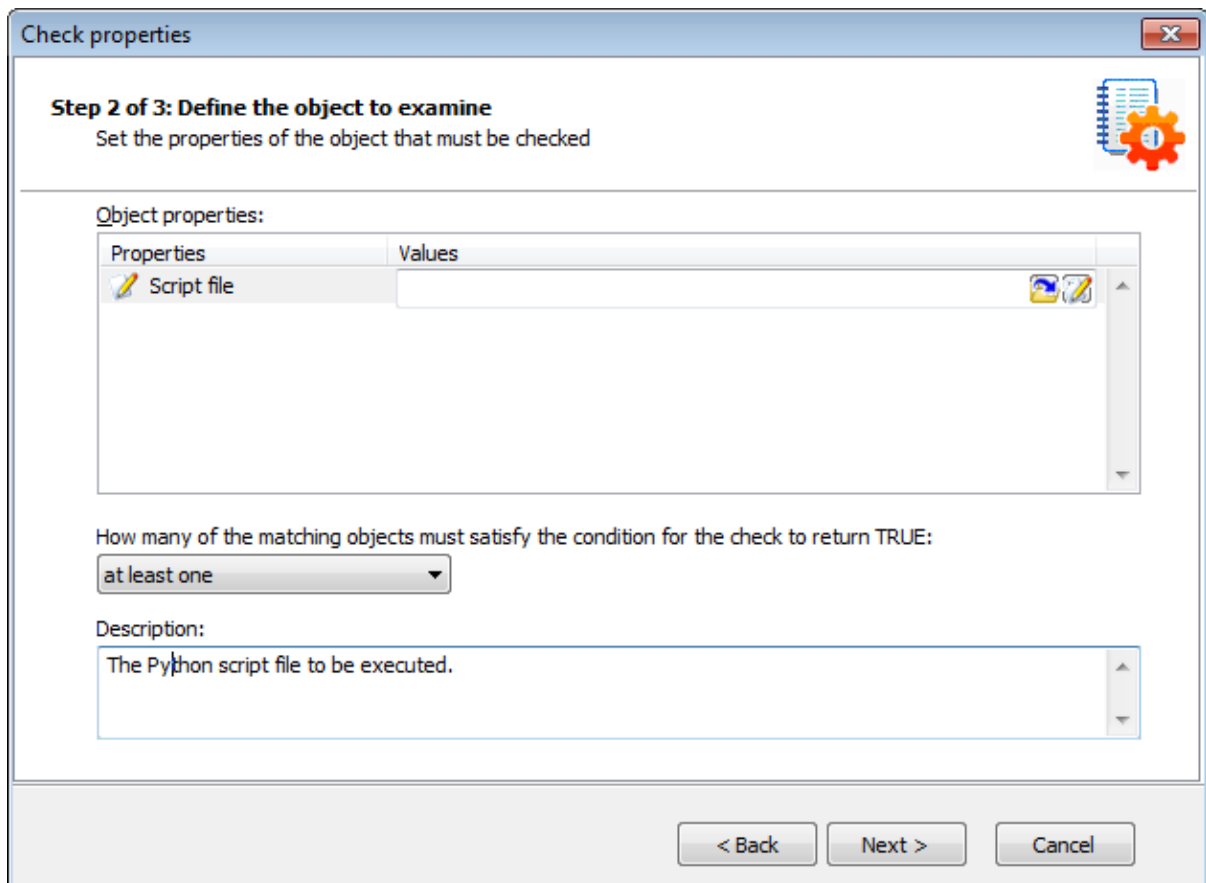
The screenshot shows a window titled "Add vulnerability" with a close button in the top right corner. The window has four tabs: "General", "Conditions", "Description", and "References". The "General" tab is selected. The fields are as follows:

- Name:** An empty text input field.
- Type:** A dropdown menu with "DNS" selected.
- OS Family:** A dropdown menu with "windows" selected.
- OS Version:** An empty text input field.
- Product:** An empty text input field.
- Timestamp:** A text input field containing "6/25/2010" and a calendar icon.
- Severity:** A dropdown menu with "High" selected and a warning icon.

At the bottom of the window are "OK" and "Cancel" buttons.

Screenshot 12 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 (**Example**"<ProductDataDir>\Scripts\newvulnerability.py").
Click **Next**



Screenshot 13 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.

3.3 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.

All of the Python scripts used by security scanner modules (vulnerability checks and security applications information) have access to the following global variables:

- » ComputerIP
- » ComputerName
- » LNSSDir
- » localProgramFilesPath
- » localWindowsPath
- » localSystem32Path
- » User
- » Password

Example

```
ComputerIP = '127.0.0.1'
ComputerName = 'WXPSandbox"
LNSSDir = 'C:\\Program Files\\GFI\\LanGuard 2011\\'
localProgramFilesPath = 'C:\\Program Files'
localWindowsPath = 'C:\\WINDOWS'
localSystem32Path = 'C:\\WINDOWS\\system32'
User = ''
Password = ''.
```

Use these global variables in order to get access to the name or IP address of the scanned machine, to credentials needed in order to connect to the scanned machine etc.

The **GFI LanGuard global functions** are available to Python scripts as methods of the `GlobalFunctions` object. E.g.:

```
# Using global functions.
def main():
    """Return values:
    * 0 - false, failed
    * 1 - true, success"""
    result = 0
    # Display text.
    GlobalFunctions.echo("Hello!")
    # Adds given text to the log file.
    GlobalFunctions.writetolog("Python script started.")
    # Adds given text to the status bar.
    GlobalFunctions.statusbar("Hello StatusBar!")
    # Adds a subnode named like the second parameter to the current
    vulnerability.
    GlobalFunctions.addlistitem("", "Vulnerability description.")
    # Changes the description of the current vulnerability.
    GlobalFunctions.setdescription("List of modems installed")
    GlobalFunctions.writetolog("Python script finished.")
    result = 1
    return(result)
```

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



All of the **LanGuard COM scripting libraries** are available via `win32com.client` to Python scripts.

Example

```
# DNS lookup using LanGuard COM object GlbObj.Socket.
def main():
    """Return values:
    * 0 - false, failed
    * 1 - true, success"""
    result = 0
    import win32com.client
    socket = win32com.client.Dispatch("GlbObj.Socket.1")
    print(socket.DnsLookup('gfi.com'))
    result = 1
    return(result)
```

3.3.1 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>

3.4 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 - Python Editor
- » IDLE and the Python interpreter that ships by default with the Python distribution.

The Python scripts can be run and debugged in an environment similar to the one in GFI LanGuard by running the scripts using the file `lpy.py` available in the Scripts folder of the GFI LanGuard installation.

Example

```
<Python25Folder>\python.exe "<ProductDataDir>\Scripts\lpy.py"
<NameOfPythonScriptFileToTest>
```

3.5 Using Libraries and code reusability

Functionality which is often used in Python scripts can be grouped in Python library scripts which then can be reused from all of the other Python script files.

3.5.1 Creating a library

Create a Python script file which defines Python functions or objects. Copy the file to the <LanGuard main directory>\Library.

Below is an example of a library file.

```
"""
Hello World Python library.
Copy this file to <LanGuard InstallDir>\Library.
"""
def EchoString(text):
    """Prints the given text."""
    GlobalFunctions.echo(text)
    print(text)
```

3.5.2 Using libraries

In order to use a previously created Python library, import that library using the standard Python import mechanism.

Sample of Python script which uses the Python library created above.

```
"""
LibraryTester.py
Requires file EchoLibrary.py to be in directory <LanGuard
InstallDir>\Library.
"""

def main():
    """Return values:
    * 0 - false, failed
    * 1 - true, success"""
    result = 0
    from EchoLibrary import EchoString
    EchoString("Successfully run vulnerability check.")
    result = 1
    return(result)
```


4 Object Documentation

4.1 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 arrays 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.

4.1.1 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
```

4.1.2 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
```

4.1.3 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

```

4.1.4 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

```

4.1.5 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
```

4.1.6 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
```

4.1.7 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
```

4.1.8 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

4.1.9 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

4.2 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.

4.2.1 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")
    Vall = "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 Vall
    echo "Oid: '"+Vall + "'" + cr
    echo "Value: '"+snmp1.Get(Vall)+"'" + cr
    snmp1.Close
End Function
```

4.2.2 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")
    Vall = "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 Vall
    echo "Oid: '"+Vall + "'" + cr
    echo "Value: '"+snmp1.Get(Vall)+"'" + cr
    snmp1.Close
End Function
```

4.2.3 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



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")
    Vall = "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 Vall
    While Vall <> ""
        echo "Oid: '"+Vall + "'" + cr
        echo "Value: '"+snmp1.Get(Vall)+"'" + cr
        Vall = snmp1.GetNext(Vall)
        If InStr(Vall, root) <> 1 Then Vall = ""
    Wend
    snmp1.Close
End Function
```

4.2.4 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")
    Vall = "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(Vall, "Malta") = true Then
        echo("Value Set successfully")
    Else

```

```

        echo("Failed to Set value")
    End If
    snmp1.Close
End Function

```

4.2.5 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")
    Vall = "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 Vall
    echo "Oid: '"+Vall + "'" + cr
    echo "Value: '"+snmp1.Get(Vall)+"'" + cr
    snmp1.Close
End Function

```

4.3 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.

4.3.1 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
```

4.3.2 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
```

```
        End If
    End Function
```

4.3.3 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
```

4.3.4 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\hosts",
GENERIC_READ, Open_Existing) Then
        echo(textfile.read(1024,1))
    End If
End Function
```

```

        textfile.Close
    End If
End Function

```

4.3.5 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

```

4.3.6 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

```

```
End If
End Function
```

4.3.7 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\hosts",
        GENERIC_READ, Open_Existing) Then
        Textfile.Seek 50,0
        echo(textfile.read(1024))
        textfile.Close
    End If
End Function
```

4.3.8 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.



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
```

4.3.9 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\hosts",
        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
```

4.3.10 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
```

4.3.11 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
```

4.3.12 Attributes

Returns the attributes of a file.

Syntax

```
Attributes
```

Returns

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

More Information

Attributes values:

- » Read Only - File is set as read only.
- » Hidden - The file or directory is hidden.
- » 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.



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
```

4.4 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.

4.4.1 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
```

4.4.2 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
```

4.4.3 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)
```



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
```

4.4.4 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\CurrentVersion\Run",
    valueName)
    While Value <> ""
        Echo "ValueName: " & valueName & " = " & value & cr
        Value = ro.GetNextValue(valueName)
    Wend
End Function
```

4.4.5 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

TYPE	REGISTRY VALUE
Long	REG_DWORD
String	REG_SZ
Array of String	REG_MULTI_SZ
Array of Bytes	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 script 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\CurrentVersion\Run",
    valueName)
    While Value <> ""
        Echo "ValueName: " & valueName & " = " & value & cr
        Value = ro.GetNextValue(valueName)
    Wend
End Function
```

4.4.6 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 script 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
    End While
End Function
```

```

        Value = ro.GetNextKey
    Wend
End Function

```

4.4.7 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

```

4.4.8 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

```

4.4.9 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
```

4.5 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.

4.5.1 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

```

4.5.2 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

```

4.5.3 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
```

4.5.4 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
```

4.5.5 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 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

```

4.5.6 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

```

4.5.7 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

```

4.5.8 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
```

4.5.9 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

```

4.5.10 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

```

4.5.11 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

```

4.5.12 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
```

4.5.13 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
```

4.5.14 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
```

4.5.15 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
```

4.5.16 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
```

4.5.17 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

```

4.6 HTTP Headers Object

4.6.1 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 (Example: "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

```

4.6.2 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
```

4.6.3 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
```

Next
End Function

4.7 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.

4.7.1 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
```

4.7.2 GetCurrentDirectory

GetCurrentDirectory retrieves current directory on the ftp server. Any file functions (Example: 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
```

4.7.3 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 an 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

```

4.7.4 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 an 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

```

```

        End If
    End If
End Function

```

4.7.5 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 an 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@lss.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 = "lssDir" & 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

```

4.7.6 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 an 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
```

4.7.7 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 an 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
```

4.7.8 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 an 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
```

4.7.9 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 an 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
```

4.7.10 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
```

4.7.11 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

```

4.7.12 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
    End If
End Function

```

```
Wend
End Function
```

4.7.13 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"
        End If
        FileName = FTPobj.GetFindFileName
        RenameFileName = "renamed_" +
        FTPobj.GetFindFileName
        ret = FTPobj.RenameFile (FileName,
        RenameFileName)
    End If
Wend
End Function
```

4.7.14 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
```

4.7.15 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_ATTRIBUTE_* constants. I.e. FILE_ATTRIBUTE_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

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
```

4.7.16 GetLastError 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
```

4.8 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.

4.8.1 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
```

4.8.2 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
```


5 General Functions

GFI LanGuard provides functions which provide integration between the scripts and GFI LanGuard.

5.1 About this manual

5.1.1 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
```

5.1.2 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
```

5.1.3 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
```

5.1.4 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
```

5.1.5 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
```


6.1 Introduction to using automation objects

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.

6.1.1 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_Service")
'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

```

7 Using Libraries and code reusability

The user has the possibility to create custom libraries which can be used in scripts. This allows you to create libraries with functions which need to be used in from more scripts. Then simply import the library in all the scripts you wish to use the functions in.

7.1 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.

7.2 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

B

Boolean, 4, 33, 55, 57, 59, 60, 62

Breakpoint, 4

Breakpoints, 3, 7

D

DnsLookup, 17, 26, 27

F

FTP, 21, 22, 23, 26, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66

H

HTTP, 2, 24, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53

I

IDLE, 18

L

Libraries, 18, 75

P

parameters, 3, 9, 26

Platform dependent, 1

Platform independent, 1

PythonWin, 18

S

Scanner, 4, 6, 10, 16

SNMP, 27, 28, 29, 30

V

variables, 5, 6, 7, 16, 21

VBScript, 4, 5

W

watches, 4

Winpdb, 18

wmi, 5, 6, 9, 70, 73, 74

USA, CANADA AND CENTRAL AND SOUTH AMERICA

15300 Weston Parkway, Suite 104, Cary, NC 27513, USA

Telephone: +1 (888) 243-4329

Fax: +1 (919) 379-3402

Email: ussales@gfi.com

UK AND REPUBLIC OF IRELAND

Magna House, 18-32 London Road, Staines, Middlesex, TW18 4BP, UK

Telephone: +44 (0) 870 770 5370

Fax: +44 (0) 870 770 5377

Email: sales@gfi.co.uk

EUROPE, MIDDLE EAST AND AFRICA

GFI House, San Andrea Street, San Gwann, SGN 1612, Malta

Telephone: +356 2205 2000

Fax: +356 2138 2419

Email: sales@gfi.com

AUSTRALIA AND NEW ZEALAND

83 King William Road, Unley 5061, South Australia

Telephone: +61 8 8273 3000

Fax: +61 8 8273 3099

Email: sales@gfiap.com



Disclaimer

© 2011. GFI Software. All rights reserved. All product and company names herein may be trademarks of their respective owners.

The information and content in this document is provided for informational purposes only and is provided "as is" with no warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. GFI Software is not liable for any damages, including any consequential damages, of any kind that may result from the use of this document. The information is obtained from publicly available sources. Though reasonable effort has been made to ensure the accuracy of the data provided, GFI makes no claim, promise or guarantee about the completeness, accuracy, recency or adequacy of information and is not responsible for misprints, out-of-date information, or errors. GFI makes no warranty, express or implied, and assumes no legal liability or responsibility for the accuracy or completeness of any information contained in this document.

If you believe there are any factual errors in this document, please contact us and we will review your concerns as soon as practical.