BOVSTT

Buffer Overflow Vulnerability Services Tester Tool

Author: Ivan Ricart Borges
Platform: Microsoft Windows
IDE: DEV-C ver-4.9.9.2
Compiler: MinGW
Dependences: Libwsock32.a
Version: 2.0
Project: https://github.com/iricartb/buffer-overflow-vulnerability-services-tester-tool
Mail: iricartb@gmail.com
Linkedin: https://www.linkedin.com/in/ivan-ricart-borges

1. DETAILED OVERVIEW

Program to detect the existence of remote / local stack-based buffer-overflow vulnerabilities using the standard communication protocol for each service.

The application allows to customize the testing mechanism of each service through templates, these templates are simply plain text files, which accept some kind of special words (see STF section), these files are stored in the <services> folder with a direct association between the protocol and the template and with the extension STF (Service Tester File).

Currently the application version 2.0 supports the FTP, POP3 and SMTP protocol.

To carry out this task the application allows to specify different types of parameters.

1.1 PARAMETERS

1.1.1 Aplication Layer Protocol

Description: Specifies the type of protocol to be tested.
Required: Yes
Options: -ap --application-layer-protocol <protocol>
Accepted values: FTP, POP3 or SMTP

1.1.2 Target Hostname IP

Description: Specifies host / ip address to be tested
Required: Yes
Options: -th --target-hostname-ip <hostname>
Accepted values: Any valid host / ip address.

1.1.3 Target Port

Description: Specifies the destination port of the service.
Required: No
Options: -tp --target-port <port>
Accepted values: 1 - 65535

If the user does not enter this parameter the application will automatically try to connect to the default destination port according to the service and the type of encryption.
For example for POP3 service and SSL encryption the default port would be 995.

1.1.4 Cryptographic Security Protocol

Description: Specifies the type of service encryption.
Required: No
Options: -cp --cryptographic-security-protocol <crypt protocol>
Accepted values: SSL, TLS
Note: No support yet.

1.1.5 Login Username

Description: Specifies the user of the credentials.
Required: No
Options: -lu --login-username <username>
Accepted values: Alphanumeric value.

This parameter allows to customize the authentication mechanism of the protocol.

The application will initiate the authentication protocol through user / password as soon as it reads the #AUTH macro within the STF file associated with the protocol. If the authentication by user / password fails, the program will cancel its execution.

Every time the application reads the keyword <login-username> inside the STF file, it will be replaced by the value of this parameter.

1.1.6 Login Password

Description: Specifies the password of the credentials.
Required: No
Options: -lu --login-password <password>
Accepted values: Alphanumeric value.

This parameter allows to customize the authentication mechanism of the protocol.

Every time the application reads the keyword <login-password> inside the STF file, it will be replaced by the value of this parameter.

1.1.7 Buffer Size Length

Description: Specifies the buffer size.
Required: No
Options: -bs --buffer-size-length <size>
Accepted values: Numeric value greater than 0.
Default value: 4096

This parameter allows to customize the size of the buffer to send.

Every time the application reads the keyword <buffer> inside the STF file, it will be replaced by the sentence { --buffer-character } * { --buffer- size-length }, in this case for example **A*4096**.

1.1.8 Buffer Character

Description: Specifies the buffer character.
Required: No
Options: -bc --buffer-character <character>
Accepted values: Alphanumeric value.
Default value: 'A'

1.1.9 Output Verbose

Description: Specifies whether the user wants to obtain more information during the negotiation process with the remote host.
Required: No
Options: -ov --output-verbose
Accepted values: none

1.1.10 Credits

Description: View the author of the program.
Required: No
Options: -c --credits
Alone: Yes, cannot be combined with another parameter.

1.1.11 Version

Description: View the version of the program.
Required: No
Options: -v --version
Alone: Yes, cannot be combined with another parameter.

1.2 STF FILES

1.2.1 Description

The STF files could be considered as a template, these are simply plain text files, which accept some kind of special words, these files are stored in the <services> folder with a direct association between the protocol and the template and with the extension STF (Service Tester File).

For example for the FTP protocol there is an STF file in the <services> folder called FTP.stf, for SMTP there is an STF file called SMTP.stf and so on.

Once the connection to the remote host is established, the application begins to read the corresponding STF file, later it'll read line by line until finalizing the file or until it finds an error.

Each line of the file **represents a command to send** to the remote host, with the particularity that it accepts a series of keywords that will be translated at runtime, these keywords are as follows:

<login-username>: Each time the application finds this tag inside the file STF, this will be replaced by the value of the parameter -lu --login-username entered by the user.

<login-password>: Each time the application finds this tag inside the file STF, this will be replaced by the value of the parameter -lp --login-password entered by the user.

<buffer >: Each time the application finds this tag inside the file STF, this will be replaced by the values of the parameters { --buffer-character } * { --buffer- size-length } entered by the user.

<remote-domain>: Each time the application finds this tag inside the file STF, this will be replaced by the domain value of the parameter -th --target-hostname-ip entered by the user.

These files also accept a series of macros that allow to change the behavior of the testing mechanism, These macros are as follows:

> #AUTH: Must be entered without further information, implies that **all the sentences that follow will be executed only if the process of authentication has been satisfactory**. The authentication process is automatic, for this it is important that the user has entered the user and password as parameters in the application.
>
> Its use is not obligatory, but in case of applying it we could send commands to the remote server where only the authenticated users can have access.
>
> #RETURN <VALUE> : <COMMAND>: The command <COMMAND> will be sent only if a return value <VALUE> has been returned in the last send process, **otherwise the test program will stop**, could be considered as a conditional command, in case the remote host has answered in its last command a certain value, the system continues with the test.

1.2.2 Example

Let's imagine the following file FTP.stf:

```
USER <buffer>
PASS <buffer>
#AUTH
MKD <buffer>
RMD <buffer>
```

To simplify the example, let's imagine that the user has entered the buffer size of **10** as the parameter.

Once the connection to the remote host is established, the application will start reading line by line.

First will find the sentence USER <buffer>, which will be translated by USER AAAAAAAAAA, in case the information is sent correctly and does not cause a buffer overflow, the application will continue with its execution.

Later will find the sentence PASS <buffer>, which will be translated by PASS AAAAAAAAAA, in case the information is sent correctly and does not cause a buffer overflow, the application will continue with its execution.

Later will find the sentence #AUTH, then the system will start the authentication mechanism of user / password, if the process is successful the application will continue to read the file.

Finally will find the sentences MKD <buffer> and RMD <buffer>, which will be translated by MKD AAAAAAAAAA and RMD AAAAAAAAAA respectively. These commands will only be sent to the remote host if the authentication process has been successful.

1.3 OUTPUTS

In each of the commands sent to the remote host, the program can display 3 output types.

> [OK]: The information has been sent correctly and no buffer overflow has been found in the sending of the information.

[FAIL]: Error during the sending of the information, the conditional value of the #RETURN macro has not been met or the authentication mechanism has failed by the #AUTH macro.

[WARNING]: **Possible buffer overflow found** or connection closed by remote host.

2. COMPATIBILITY / REQUIREMENTS

Currently the system supports the **Microsoft Windows** platform and to generate the corresponding binary file only the Dev-C ++ IDE should be downloaded

Platform: Microsoft Windows
IDE: DEV-C ver-4.9.9.2
Compiler: MinGW
Dependences: Libwsock32.a (included in Dev-C++ IDE)

The Dev-C++ IDE can be downloaded from the following link:

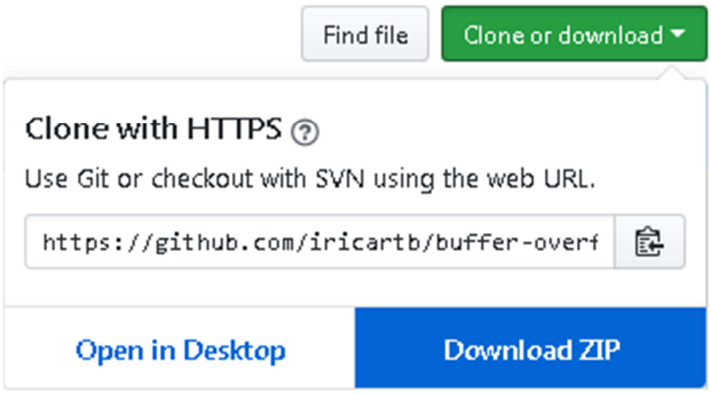
https://sourceforge.net/projects/dev-cpp/files/Binaries/Dev-C%2B%2B%204.9.9.2/devcpp-4.9.9.2_nomingw_setup.exe/download?use_mirror=netix&r=&use_mirror=netix

3. INSTALLATION

To install the application the following steps must be taken:

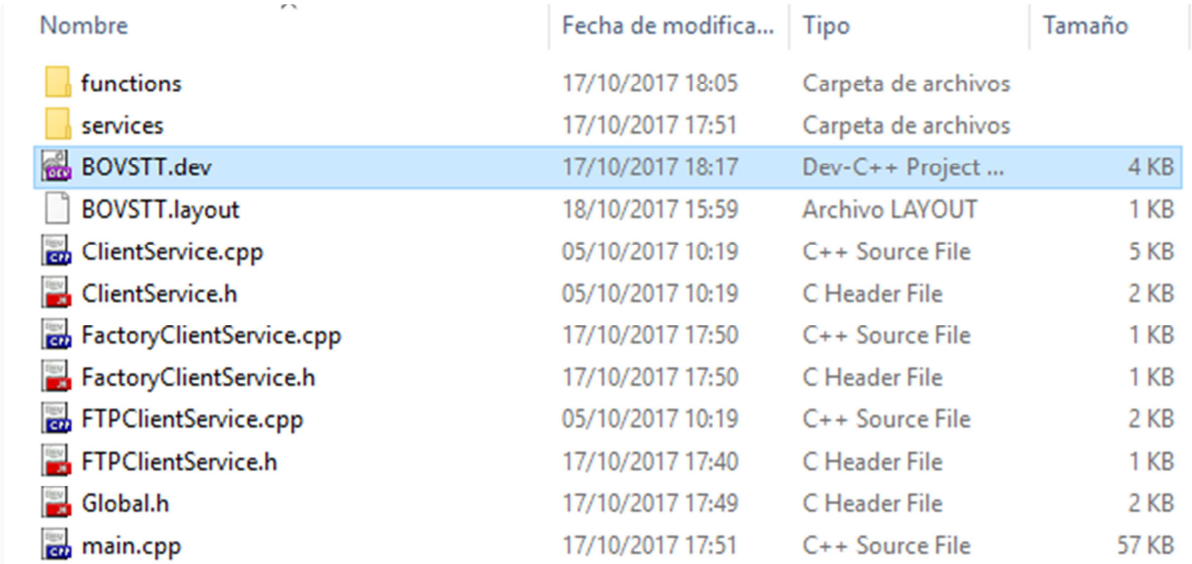
1. Installing the Dev-C ++ IDE: Go to the next link and run the setup. https://sourceforge.net/projects/dev-cpp/files/Binaries/Dev-C%2B%2B%204.9.9.2/devcpp-4.9.9.2_nomingw_setup.exe/download?use_mirror=netix&r=&use_mirror=netix

2. Download the GitHub project: Go to https://github.com/iricartb/buffer-overflow-vulnerability-services-tester-tool and press the download button in zip.



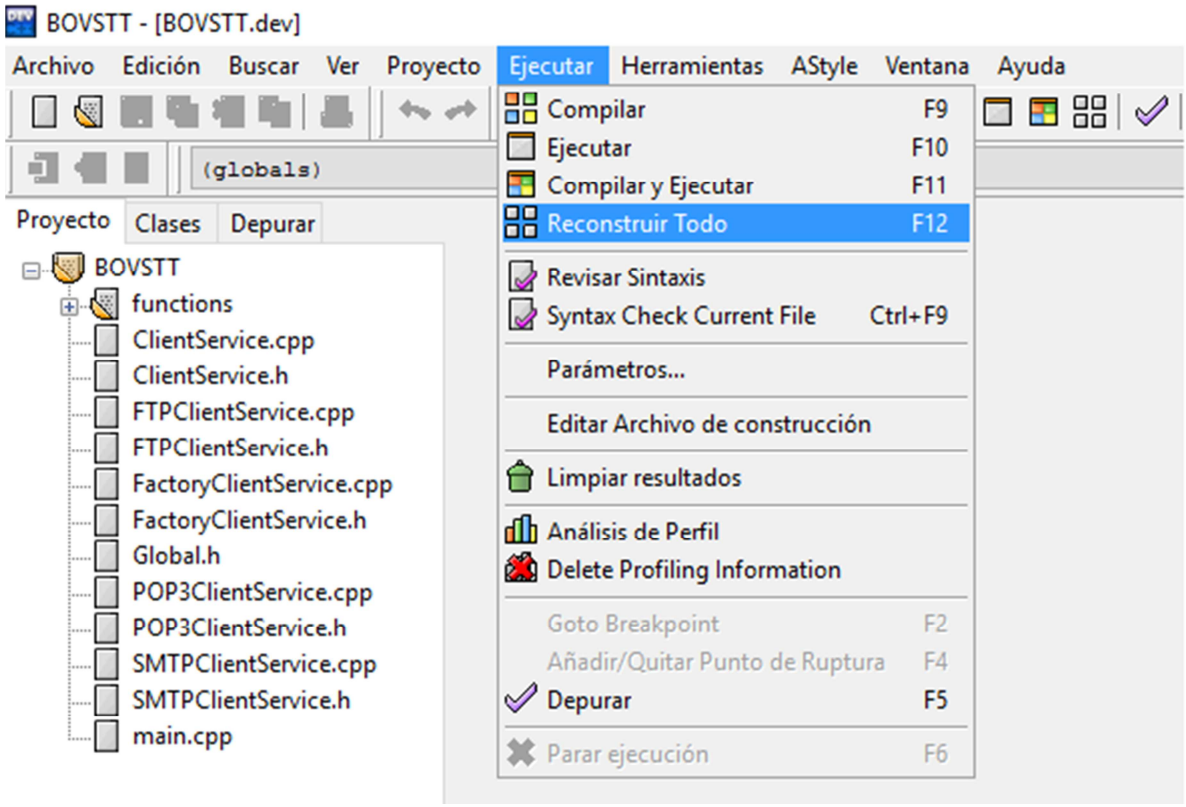
3. Unzip the zip project using a decompression program.

4. Double click on the file BOVSTT.dev to load the Project.

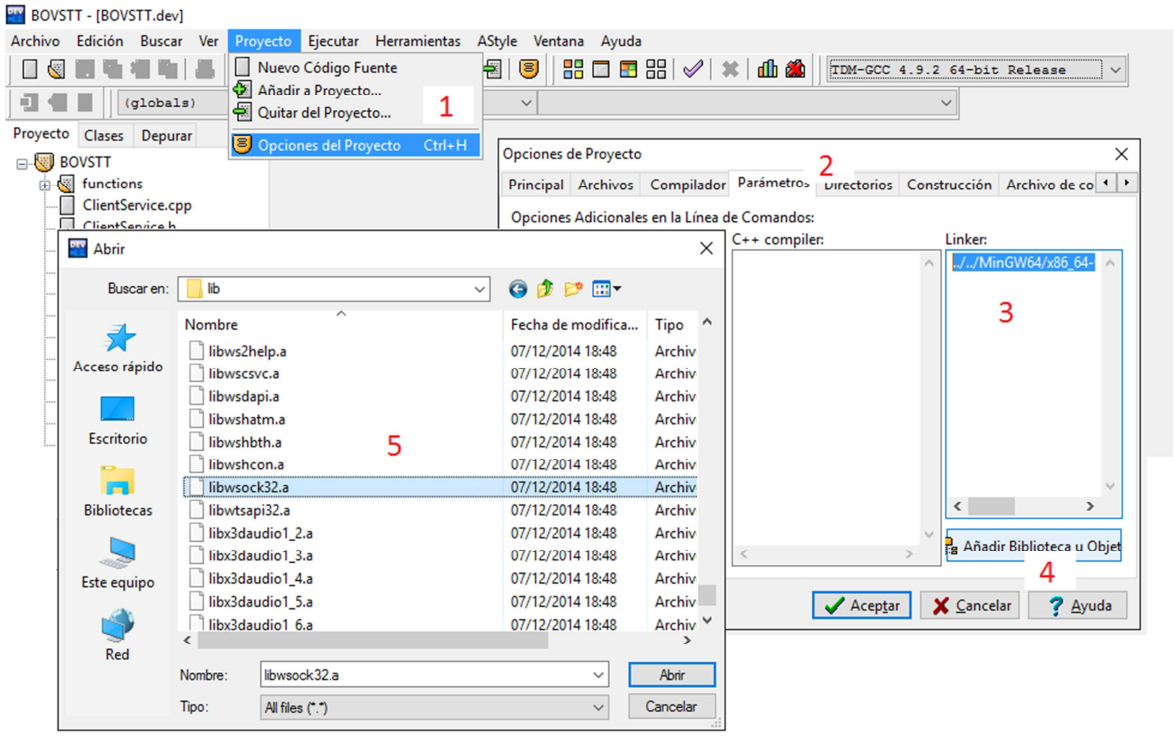


Nombre	Fecha de modifica...	Tipo	Tamaño
functions	17/10/2017 18:05	Carpeta de archivos	
services	17/10/2017 17:51	Carpeta de archivos	
BOVSTT.dev	17/10/2017 18:17	Dev-C++ Project ...	4 KB
BOVSTT.layout	18/10/2017 15:59	Archivo LAYOUT	1 KB
ClientService.cpp	05/10/2017 10:19	C++ Source File	5 KB
ClientService.h	05/10/2017 10:19	C Header File	2 KB
FactoryClientService.cpp	17/10/2017 17:50	C++ Source File	1 KB
FactoryClientService.h	17/10/2017 17:50	C Header File	1 KB
FTPClientService.cpp	05/10/2017 10:19	C++ Source File	2 KB
FTPClientService.h	17/10/2017 17:40	C Header File	1 KB
Global.h	17/10/2017 17:49	C Header File	2 KB
main.cpp	17/10/2017 17:51	C++ Source File	57 KB

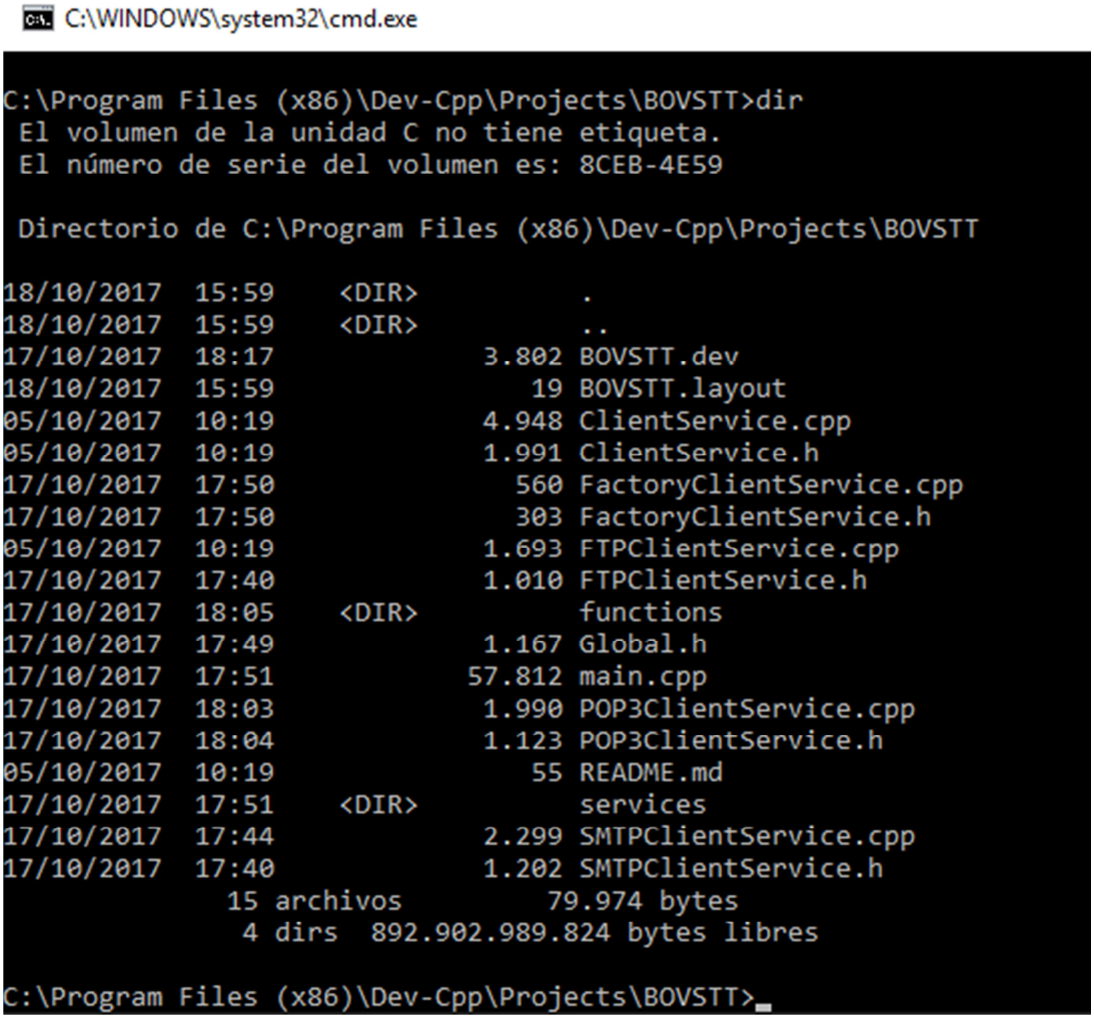
5. In the Dev-C ++ IDE go to the **Execute** menu and click on the option to **rebuild all** (F12). If a dependency error occurs go to point 6, otherwise go to point 7.



6. In the Dev-C ++ IDE go to the **Project** menu and click on the option **Project options** (1), later go to parameters tab (2) and delete the line that appears in the Linker section (3), then click on the **add library** button and finally find the libwsock32.a library in the lib folder of the Dev-C ++ IDE (5), select it and return to point 5.



7. At this point the BOVSTT.exe executable file should exist. Run the windows cmd.exe console and browse the filesystem until you find the project path.



8. Finally run the BOVSTT.exe file with its parameters to start the test process.