Python Bot

Table of Contents

Introduction

This document explores the utilization of a Bot.net to simulate a Bot attack where the user downloads a file from a compromised website and when is interacted with, the communication between the controller and the bot is initiated. A random message is sent over the network to the controller every time the Bot is executed. How these messages are encoded is explored later.

Additionally, a few techniques are implemented to hide the files that the user has to execute. The victim in this exercise is intended to be a Windows workstation.

Furthermore, the bot must be present in the victim workstation and while this document does not explore how to compromise the victim machine, previous exercises within the project make this exploit available. In this case, the vulnerable website www.iFruit.com has a container where an administrator can post messages of the day, this feature when used by an attacker with access to the administrator account can make the bot downloadable with a click from any user that enters the web site.

Bot exploration

Python 3 has been used to develop the bot and the controller respectively to create a listening socket on the attacker's machine waiting for connection on port 5005. The controller will wait until there is no more data written into the TCP buffer from the bot to close the connection.

Given the implementation and characteristics of the environment the bot and controller have been hard coded to use specific IP addresses and ports, although, it can be configured with a customized public or private IPs for different uses and purposes.

The controller named 'Server.py' has to be executed with the following command – ***python3 server.py*** in the directory where it's located (assuming that it is being executed in a Linux based operating system). It will then wait for an incoming connection.

Once the user executes the bot, with a double click in a Windows operating system, it will send the messages that were coded previously, however, one of the messages lists the '*C:\Users*' folder achieving more dynamic results and a potential vulnerability to expose other systems.

The bot has been 'randomized' to send a message each time it is executed, this means that not all the prepared messages are sent, every time the bot is executed there is one out of six chances for a specific message to be sent, the idea behind this feature is to allow further analysis of the bot.

Also, note that the messages are not printed out to the console once they are received, instead, they can be captured with **Wireshark** as they are transmitted in plain text over the network. At last, and when a message has been received, the server sends back an acknowledgement just before closing the connection which is not displayed to the console either.

Concealment

A user that recognizes an *'.exe'* might not use the bot as it could be potentially dangerous, it is good practice not to execute files of which the precedence it's unknown. To conceal the file the **Python** *client_start.py* has been modified to *notes.jpeg* changing also how Windows recognizes the file, in this case using a image icon for it, although it cannot be executed since it is missing the *.exe* extension used to start programs in Windows. In this case a shortcut of the image file has been created, however, it has also been modified to display a *.zip* icon and to execute the previously mentioned image file from the **CMD** without the user knowledge. For the attack to work it is necessary that both files the image and shortcut to be in the same folder.

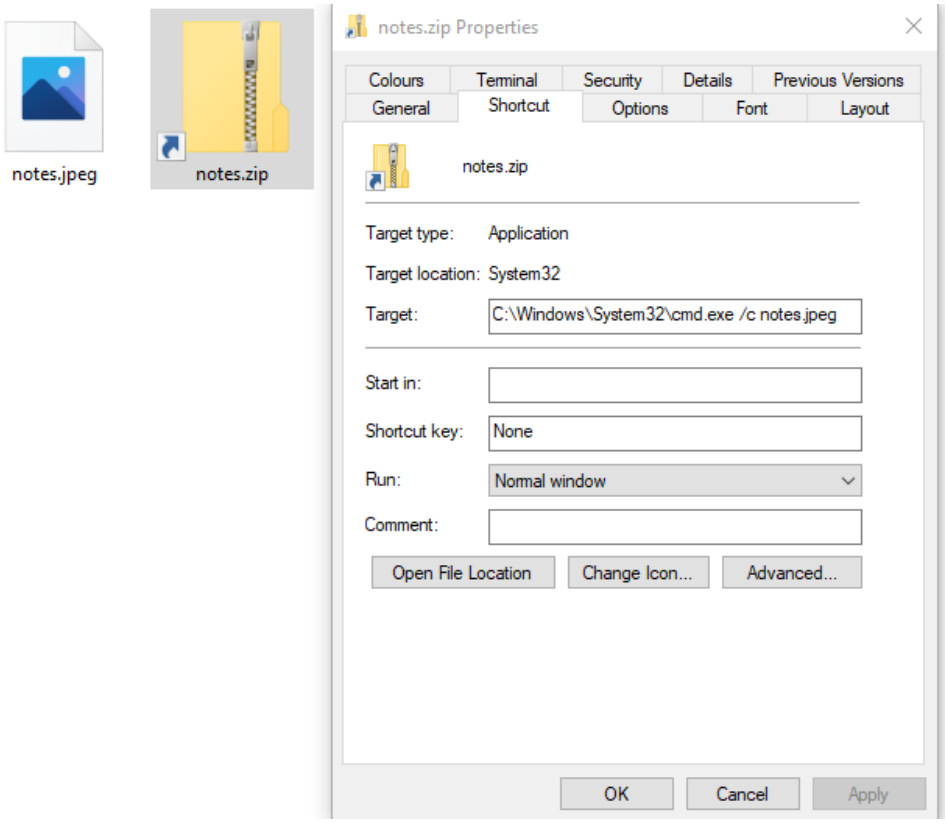

Figure 1

Figure 1 demonstrates how the shortcut is able to execute the file even if the format has been changed to *.jpeg* from them or make them take an action of which they are not aware.

```
C:\Windows\System32\cmd.exe /c notes.jpeg
```

The above command is inserted in the target of the shortcut, opening the CMD in background (option /c) and executing *notes.jpeg*.

Messages

The bot contains six messages in total of which one is sent every time it is executed, and the controller is listening.

The first message contains a string outputting *'Sent.. there should be 5'* which gives a clue of how many more messages there are, note that the order in which these are sent is random.

The second piece of code executes *'cmd /c "dir"'* after changing the directory to *'C:\\Users'*, then it is encoded and sent. Note that all messages must be encoded before they are sent over the TCP tunnel. This message also displays an error with '*try me differently, not my environment*' if the Users folder is not found, the intention is to give a hint in case the bot is used in a Linux based operating system.

The third message is further encoded with a cipher, Vigenere with key 5. By knowing this information, it is possible to convert back the string into readable text which would output '*THEY HAVE WHAT WE NEED AND SHOULD NOT LET PASS*'

The fourth message is encoded with a Caesar alphabet with 5 shifts, if decoded it should read '*IT WILL BE AN SCANDAL, WE MUST HIDE*'

The fifth message clarifies in plain text that the bot should also be used in a Windows computer so the Users folders can be listed and sent over to the controller.

At last, the sixth message sends an encoded image in **base64** that has a hidden message inside. By using stenography capabilities, a text string is added to the image that can only be disclosed if opened with a text editor. This can be accomplished by issuing the following command on the CMD in Windows.

```
copy /b name.jpg + name.txt hidden.jpg
```

A new file is created that has joined both binary strings the one from the image and that of the text, figure 2, then it is encoded with an online to into **base64** of which an string is created, that string is then sent over the TCP tunnel making it available through a network sniffer, however there are no clues as if text is hidden on the image or what system has been used to encode it.

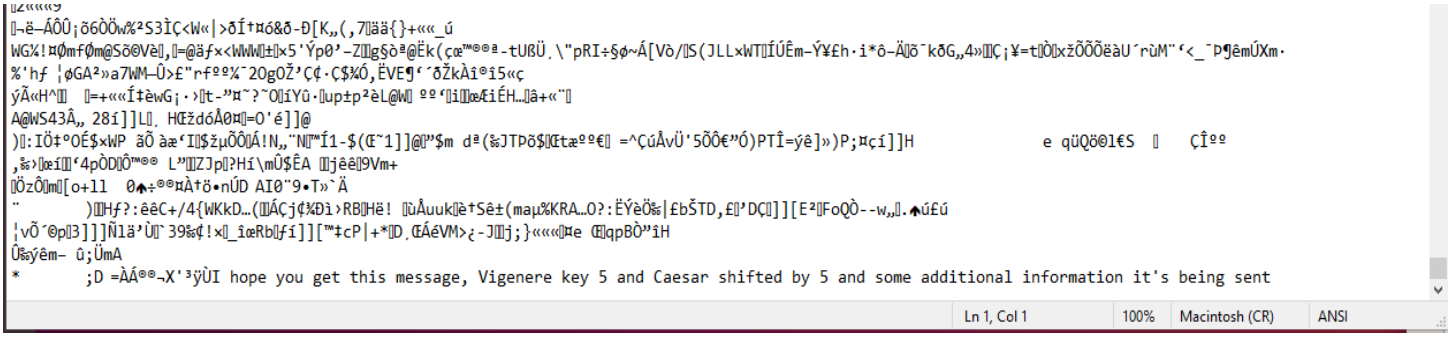

Figure 2

Implementation

To serve the bot where other users can download it an Apache 2 server has been setup. Also used in other exploits within this project. The server has the two files that have to be downloaded, the bot and the shortcut acting as executable. Once access to the administrator account of iFruit.com has been obtained, either with a brute force attack or by retrieving usernames and passwords from the web site, the posting feature is used to set a link where the bot files can be downloaded. A representation can be seen in figure 3.

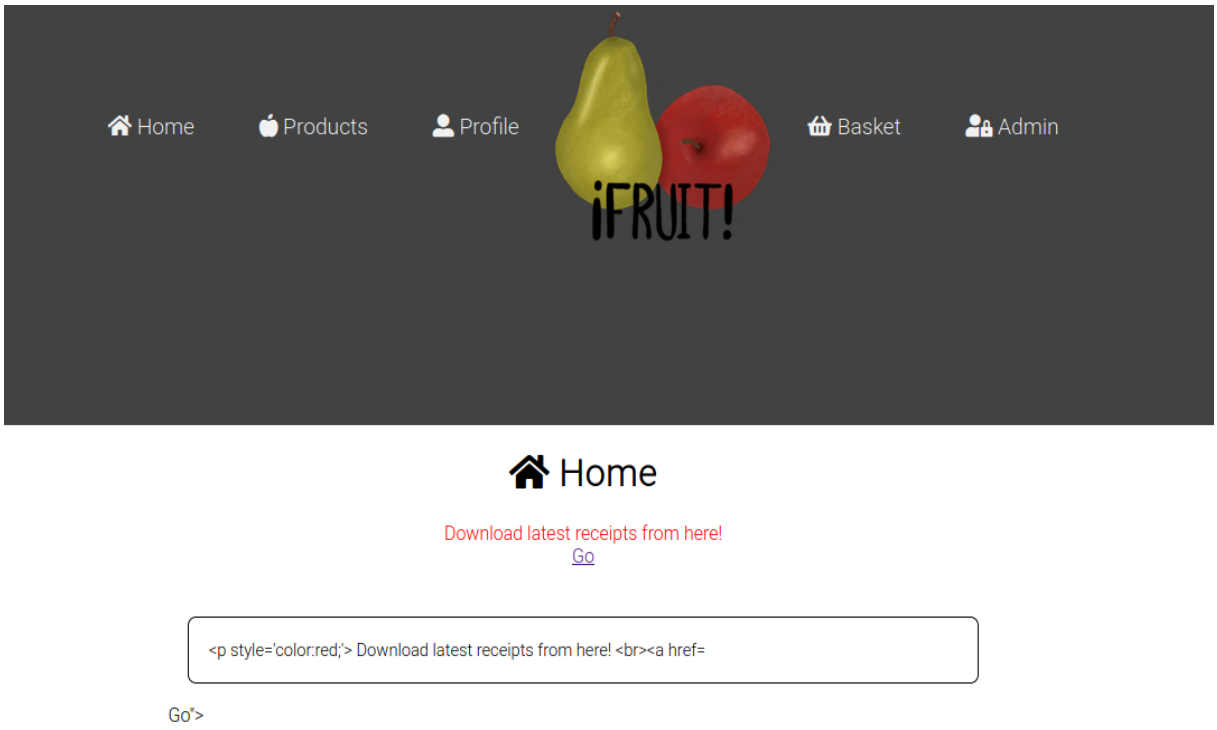

Figure 3

It is important to note that the box to submit the message is not available in accounts that are not administrator and therefore other users would not see it. Also, a paragraph indicating where to get receipts has been set to add credibility. Once clicked on a redirection takes place to the compromised web site that contains the bot files.

After the download and execution of the *'notes.zip'* file, figure 4 shows how the controller receives the message in which the image with text hidden in it has been encoded and sent over. A Wireshark capture shows the text being sent but not being outputted to the console. Additionally, in the latest version, the controller only output one line of '+' and '-' to indicate that the connection has been successful.

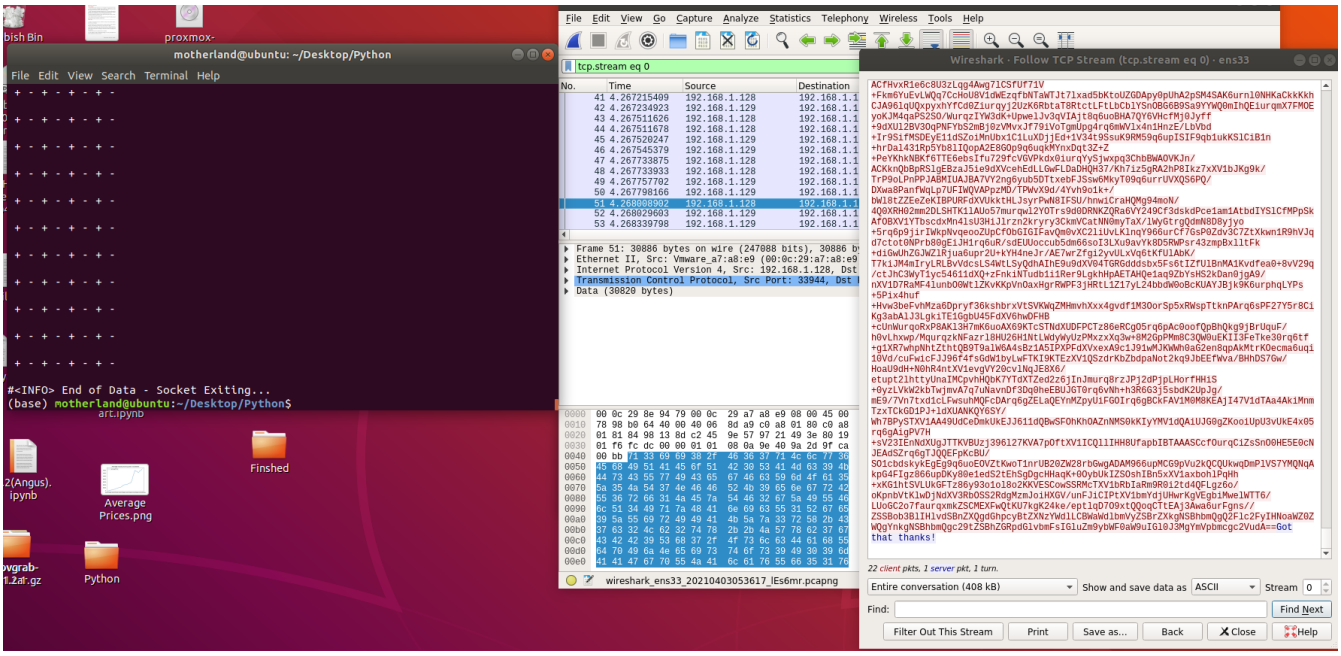

Figure 4

At last, the bot has been compiled with **Pyinstaller** so it can be executed in a foreign environment without Python3 installed, meaning that the bot can be used in any Windows workstation regardless of its configuration installation. Also, by modifying the controller to ask for an input of the user for the variable *'tcp_ip'* it would be possible to enter where the server listens for connection manually. The same can be set in the bot, although it would lose the concealment as the user would have to interact with it.

Sources

Accessed on 2021, *What PyInstaller Does And How It does It*. At: https://security.stackexchange.com/questions/189198/how-can-i-fine-tune-hydra-to-find-the-correct-password/189241#189241

W3Schools., (2021). *Python Try Except*. At: https://www.w3schools.com/python/python_try_except.asp

Walter Glenn., (2019), *How to Customize Your Icons in Windows.* At: https://www.howtogeek.com/howto/13631/customize-your-icons-in-windows-7-and-vista/#:~:text=Change%20the%20Icon%20of%20Any%20Shortcut&text=Right%2Dclick%20the%20shortcut%20and,ICO%20file%20that%20contains%20icons.

Walter Glenn., (2016), *How to Run Command Prompt Commands from a Windows Shortcut*. At: https://www.howtogeek.com/277403/how-to-run-command-prompt-commands-from-a-windows-shortcut/

Data to Fish., (2020), *Create Executable from Python Script Using Pyinstaller*. At: https://datatofish.com/executable-pyinstaller/

Bogdan Bele., (2019), *How To Hide Text Inside Image Files*. At: https://www.groovypost.com/howto/hide-text-inside-image-files/

Code Beauty., (2020), *Convert Your Base64 to Image*. At: https://codebeautify.org/base64-to-image-converter