Shenggui Jin

110984504

Project: Covert C&C and Exfiltration

Introduction:

This program demonstrates the attack of Covert Command & Control and Data Exfiltration attack. The program is written entirely in python with scapy module. The idea of the covert channel is hiding encrypted data inside packet's payload and send to infected client device. The client-side program will sniff the traffic and perform actions if target packets arrive. It also asynchronously sends notification to server if it detects client-side user activity. The program also supports transferring of large file in both directions, making it possible for data leak and injection of malicious files.

Terminology:

--**server**: In this documentation server means the attacker. (Attacker pretending to be a server)

--**client**: In this documentation client means the victim.

Setup:

--**Environment**: 32bit Kali linux OS for both attacker and victim. (Windows 10 also works but requires specifying exact program path to run command)

--**Modify IP**: open both 'client.py' and 'server.py' and modify the ip's accordingly. (I've tried to use netifaces module to get ip automatically but kali linux doesn't have this or pip)

--**Modify Ports (Optional)**: The hard-coded ports are used for filtering packets. If you have any other program communicating between victim and attacker that uses these ports, the program might not work properly. If so, then modify the ports as well.

How To Use

--Both program supports "-v" flag and "-d" flag. With "-v", programs will print message if a packet is sent. With "-d", server program will print the answer packet of the command packet, and client program will print debug messages such as the status of file transfer. Both flags are for debugging purposes.

--Run both programs using "python3". "client.py" will hang and listen for server packet while server.py will prompt user to input. User can run any bash command or program, including ">" and ">>" redirection to files, except "|", "$" and other special symbols for bash.

--To trigger asynchronous notification, visit any websites without https (just http), and notification will come up on the server side.

--To transfer large files, there are 2 commands:

```
“copy      remote_src   my_dest”

“copy2   my_src    remote_dest”.
```

First one copies remote file to current directory. Second one copies local file to remote directory.

These 2 commands behave exactly like “scp” except that you don’t need to provide hostname. Provide arguments like normal “cp” command.

--A trivial thing about cat command or any other command that prints to the terminal: Due to the packet size limit, I truncate any outputs that are greater than 1000 bytes in length to 1000 bytes. To view the entire contents, you can use “head, tail” or simply transfer the entire file to local directory and read it.

Notice:

Because the program sets timeout for each send and receive operation. A slow in performance in either side might cause race condition, leading to unexpected result. Also make sure that there are no other programs that frequently communicate between server and client. Even though the chance is very small, it is possible that the port and id might collide, causing the sniffer to capture wrong packets.