SEGC\_EXP01

Matteo Di Fabio: 264339

Luca Genovese: 264364

Main goal

The goal of the experience is to simulate the behavior of a worm, that from outside the VM, will enumerate the VM ports; find the backdoor; connect to it; send and execute an executable file on the victim’s PC.

*note: all the following files are available in the zip and they contain all the comment with the explanation of the main commands and instructions:*

*backdoor.sh*

*backdoor.service*

*Attack\_1.sh*

*Attack\_2\_U.sh*

*Attack\_2\_T.sh*

*Attack\_3.sh*

*nmap\_out\_TCP.txt*

*nmap\_out\_UDP.txt*

*port\_scanning\_TCP.txt*

*port\_scanning\_UDP.txt*

Introduction

To virtualize the machine we use the software VirtualBox.

To simulate this experience we identify 3 possible set to configure and prepare the VM: bridge connection, NAT service with port forwarding and host-only.

**Bridge connection:** In the bridge connection it is necessary to connect the host PC (the PC on which the VM is running) with both: ethernet and WiFi interfaces; in this way the host use one interface to have the access to internet (on the host) and the other to create a bridge and give internet access also to VM. With this configuration it is theoretically possible to access to VM even from another PC connected to the same network.

**NAT:** Using a NAT, VirtualBox create a private network with internet access on the host and VM,but the VM is not normally reachable from outside. But using the port forwarding, if a user is connected to the same network of the host’s machine, also in this case it is possible to access to the VM machine (from outside). The disadvantage is that you need to configure manually the port forwarding.

**HostOnly:** With HostOnly VirtualBox create a private network, without internet access, in this way the host and the VM can communicate without any possibility of external access.

As HostOnly allows to have a private and close environment to do the simulation, we decided to use this method. Moreover as the host machine is a Windows, we choose to configure and prepare another VM for the attacker to have a more suitable OS to carry out the attack in a proper manner.

**The attacker** uses a VM with Kali-Linux 2020.1b: an open source project, derived from a Debian Linux distribution, designed for digital forensics and penetration testing. In fact kali is maintained and funded by Offensive Security, a provider of world-class information security training and penetration testing services. So, the reason to use it, is that has many tool which help the attacker to do his work efficiently and quietly. Some features that make Kali such a powerful tool are:

1. Full Customisation of Kali ISOs:

You can create your own Kali with the tool you need.

2. Kali Linux Live USB with LUKS Encrypted Persistence:

Kali has extensive support for USB live installs, allowing for features such as file persistence or full (USB) disk encryption.

3. Kali Linux Full Disk Encryption:

Having the ability to perform a full disk encryption of your sensitive penetration testing computer drive is an essential feature needed in our industry. Just the thought of unencrypted client data getting lost or mishandled is horrific.

Pre-Installed tools

1.Metasploit - Metasploit is a framework for developing exploits, shellcodes, fuzzing tool, payloads etc. And it has a very vast collection of exploits and exploitation tools bundled into this single framework.

2. NMap - Nmap is used to scan whole networks for open ports and for mapping networks and a lot more things. It is mainly used for scanning networks and discover the online PC’s and for security auditing. Most of the network admins use Nmap to discover online computer’s, open ports and manage services running. It uses raw IP packets in such a creative way to know what hosts are available on the network and what ports are open which services (applications name and version) are running on those systems.

3. Wireshark - Wireshark is an open source tool for network analysis and profiling network traffic and packets and this kind of tools are referred as Network Sniffers.

**The victim** instead is a VM that runs common Ubuntu 18.04.

Configuration of Victim’s PC and Backdoor installation

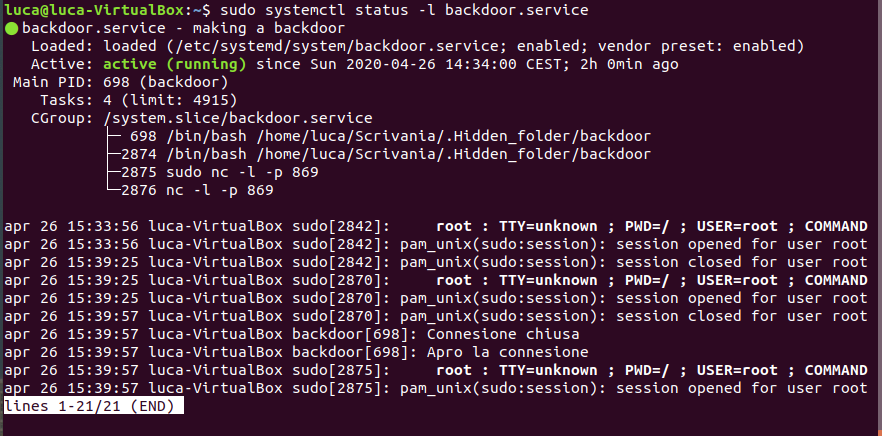
To configure the victim VM we have to set the network interface as “HostOnly” on VirtualBox settings (the same things has to be done also in the attacker VM); then to allow internet access on victim’s PC it is enabled also a second network interface set as “NAT”, in this way it is possible to simulate the real behavior of the machines.  
At this point we proceed with backdoor installation.

To do that we suppose that the attacker has had physical access to the victim’s PC just for one time and had privileged rights (root) to be able to execute command and instruction like an admin. During this session the attacker installed some tools like netcat and net-tools, that are not pre-installed on Ubuntu 18.04: netcat is the responsible to the opening of the backdoor while net-tools is useful to get network informations about victim’s PC like his IP address and so on.

*note: from now on the attacker’s PC will be called ‘A’ and the victim’s PC is ‘B’*

To install the backdoor the attacker has to perform the following steps:

* Copy the backdoorscript in B
  + In our case it is called *backdoor.sh* and is copied in an hidden folder in the home directory. Normally this kind of file are copied in systems directory like init.d or ppp or rc.local and so on, that contains already a lot of system startup files and it is very easy to mimetize it with a similar name.
* Automatic backdoor enabling, at startup
  + To do that we activate the service *backdoor.service* on the systemd software suite. This service was created by us to launch *backdoor.sh* every time the user login. The screenshot shows the status of the service:



Conduct of the Attack

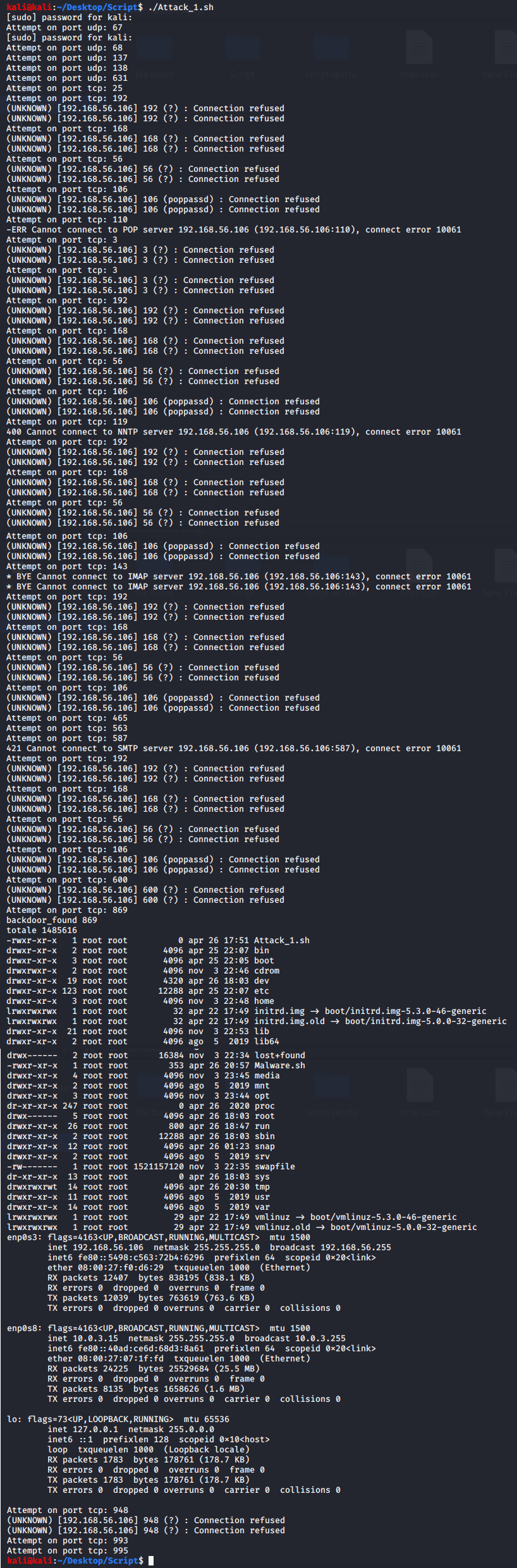
1. On B, we opened some random port to simulate a real case, in which more than one port is open and the worm should be able to recognize the backdoor.
2. On A the attacker has 3 script *Attack\_1.sh, Attack\_2\_U.sh, Attack\_2\_T.sh and Attack\_3.sh.*

* *Attack\_1.sh* scans all the port of B to finds the opens ones: both UDP and TCP and launch *Attack\_2\_U.sh and Attack\_2\_T.sh* respectively for UDP and TCP (passing the port to attack as argument).
* *Attack\_2\_U.sh* and *Attack\_2\_T.sh,* respectively for UDP and TCP connection, connect A to B using the port passed as argument, allowing the attacker to send a file (*Attack\_3.sh)*  and obtain the shell control of B to execute it (in case of success). If the connection fails, the script just doesn’t care and continue with the following port.

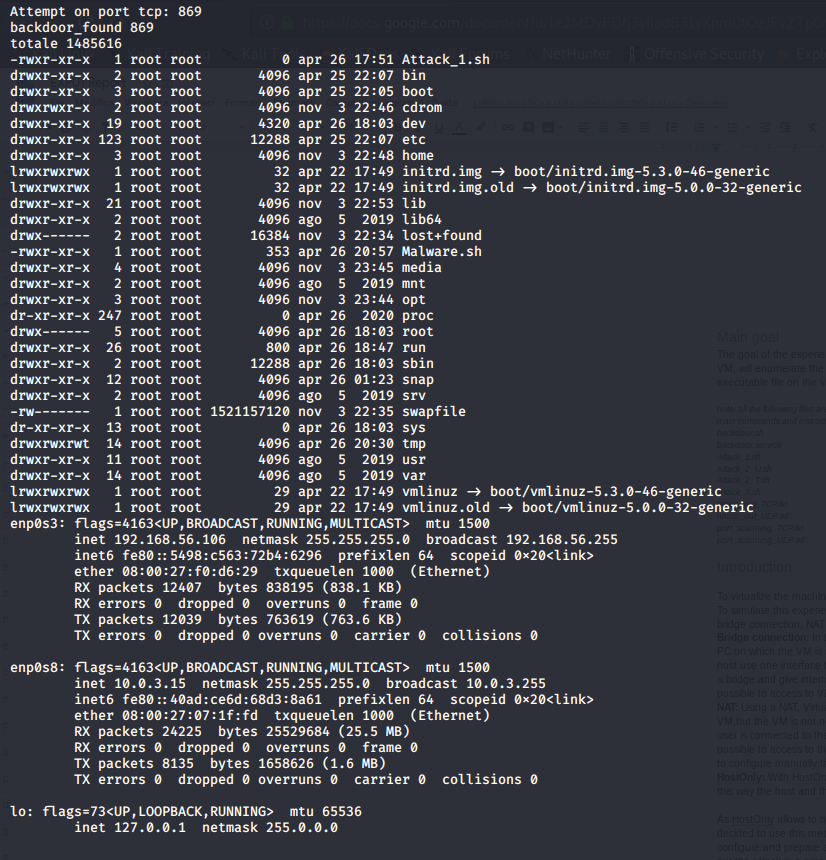
The attacker, reading the output of the shell, can understand which is the correct backdoor, because if it is he will be able to read the result of the execution of *Attack\_3.sh.*

* *Attack\_3.sh* this is the file to send into B, in our case it only retrieve some information about B as the IP and the list of file of the current directory of the backdoor otherwise you can do all what you want, as execute installed application.

Here it is possible to observe an example of the output that the attacker could read:



As it is possible to observe we can see udp doesn’t give a feedback in case of failure during the connection while tcp give an *“(UNKNOWN) [IP address] Port\_number : Connection refused”* response, instead for port 869 we can see the connection was established because it is possible to see on the command line the output of the “malware”, in fact the file (Attack\_1.sh) is sent to B and is executed returning the output into the network and received by A.

Here a detail of backdoor:

tools used:

* nmap ver 7.80
* netcat-traditional v 1.10-41.1+b1