1 Remote Access

We wanted to give the attacker remote access to the victim's machine. The simplest way and probably the best one is to use SSH.

1.1 What is SSH

SSH stands for Secure Shell. It enables the the user to log into a remote machine with his username on that remote machine. The user can then transfer files or execute commands on that remote machine as if they were using it directly.

SSH can authenticate users either by passwords or SSH keys. SSH keys are public and private key pairs and can be generated by a number of cryptographic algorithms like RSA, DSA, or ECDSA. If a user wants to connect to a remote machine using ssh keys, their public key must be saved in some file on that remote machine. The user in that case will not be prompted to enter any password.

1.2 Design

As soon as the rootkit runs on the victim's machine, it saves the attacker's public key, downloads and runs the ssh daemon and sends the username and the IP of the victim to the attacker. The attacker can then use ssh victim_username@victim_ip and they are now logged in with the victim's username. if the victim is root then the attacker will also be root.

1.3 Implementation

Python was used as it has pretty good native libraries that were needed and it is much easier to grasp and implement with than C

SSH was used with RSA keys since it is the one of the most common remote access methods to Unix servers and it is the most secure way.

All the libraries used are native python libraries

Two scripts of course need to be implemented; one will be running on the victim's machine and the other on the attacker's. The one on the victim's is a standalone program and actually can be run without sudo privilege unlike the rootkit module (though it would need some luck to work as we will explain).

The attacker's script is very simple. It just waits for the rootkit to send it the username of the victim and then it displays the username and the IP of the victim (which it got when the connection was established) to the attacker. This is easily implemented using socket API. So the attacker script is a server waiting to get a message from the rootkit and when it gets the message it displays it and terminates

Most of the work is done on the victim's machine. The script adds the public key of the attacker to ~/.ssh/authorized_keys. If the victim is root then ~ is /root. If not then it is /home/victim_username.

The script then downloads and runs the ssh daemon. This is done using the subprocess Python module which creates a Unix process given its command. After that, the rootkit gets the username and sends it to the attacker using sockets.

Note that if the victim does not have root access, there will be no way to download and run the ssh daemon. The script will not fail in that case and will still send the username to the attacker. If the ssh daemon were running on the victim machine anyway then the attacker will be able to ssh into it. That is why it is possible for this program to work without sudo privilege. If ssh daemon was not running then it is impossible for the attacker to log into the victim's machine

1.4 References

The only thing that needed research was SSH and this is a very good article about it. https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys