# Smuggling via Windows services display name - Lateral movement



Generated By 0xsp.com

# Introduction

During red teaming operation, Lateral movement or know as an east-west movement refers to the technique to move deeper into a network. Such an attack allows a threat actor to avoid detection and retain persistent access. This newly published research explains how to take advantage of windows services, it details how to mimic windows services display names to deploy malicious beacon or even Meterpreter session.

# Windows API Analysis

Before we start explaining deeply how I was able to achieve that, let's have a look into the current windows CreateserviceA function.

7					
SC_HANDLE C	SC_HANDLE CreateServiceA(				
SC_HANDLE hSCManager,					
LPCSTR	lpServiceName,				
LPCSTR	lpDisplayName,				
DWORD	dwDesiredAccess,				
DWORD	dwServiceType,				
DWORD	dwStartType,				
DWORD	dwErrorControl,				
LPCSTR	lpBinaryPathName,				
LPCSTR	lpLoadOrderGroup,				
LPDWORD	lpdwTagId,				
LPCSTR	lpDependencies,				
LPCSTR	lpServiceStartName,				
LPCSTR	lpPassword				
);					

we can notice that IpserviceName has enough character space to place a payload but MSDN mentioned that's service name comparisons are always case insensitive Forward-slash (/) and backslash () are not valid service name characters. While API parameter Ipdisplayname allows 256 character string length and there is no bad character filtration. So that means we can at least insert some base64 formatted strings into parameter values.

# Manual reproduce

by using a tool such as a process hacker we can test if we were able to create a service and fill lpdisplayname with base64 value.

🍃 Refresh   👳 Options	Find handles or DL	Ls 🎊 System info	ormation 🛛 🗖 🗔	X	Search Services (Ctrl	
Processes Services Net	vork Disk					
Name	Display name		Туре	Status	Start type	
2 1394ohci	1394 OHCI Compliant Host Controller		Driver	Stopped	Demand start	
💮 3ware	3ware		Driver	Stopped	Demand start	
Tcpip6	@todo.dll,-100;Micros	oft IPv6 Protoco	Driver	Stopped	Demand start	
Segment1	AAAAAAAAAAAAAAA		Own process	Stopped	Demand start	
AcpiDev	ACPI Devices driver		Driver	Stopped	Demand start	
AcpiPmi	ACPI Powel	Demand start				
acpipagr	ACPI Proce	Demand start				
acpitime	ACPI Wake Name:	TEST			Demand start	
AxInstSV	ActiveX Ins				Demand start	
Acx01000	Acx01000 Display na	x00xyUmJ8EiJ2kiJ+UG6AtnIX//	Demand start			
💮 arcsas	Adaptec S/ Type:	Own process	$\sim$		Demand start	
AdobeARMservice	Adobe Acr				Auto start	
ADP80XX	ADP80XX Start type:	Demand start	~		Demand start	
💐 afunix	afunix Error cont	rol: Ignore	~		System start	
AarSvc	Agent Acti				Demand start	
AarSvc_10e440	Agent Activ	Demand start				
AJRouter	AllJoyn Rou	Demand start (trigg				
💭 amdgpio2	AMD GPIO			Cancer	Demand start	
💭 amdi2c	AMD I2C Controller Se	ervice	Driver	Stopped	Demand start	
AmdK8	AMD K8 Processor Dr	iver	Driver	Stopped	Demand start	
AmdPPM	AMD Processor Driver		Driver	Stopped	Demand start	
💮 amdsata	amdsata		Driver	Stopped	Demand start	
💮 amdsbs	amdsbs		Driver	Stopped	Demand start	
💮 amdxata	amdxata		Driver	Stopped	Demand start	
AFD	Ancillary Function Driv	ver for Winsock	Driver	Running	System start	
Processes Services Ne	twork Disk		^	· ·	••••	
e Name	Display name		^		Туре	
TEST	g8QgXon2akBBWWgAEAAAQVhlifJIMclBulikU+X/1UiJw0mJx00xyUmJ8EiJ2kiJ+UG6AtnIX//					
				K		
e						
ch						
51			8			
lc						

as you see in the previous screenshots it is possible to add base64 value into Displayname.

#### Heart of the problem

That's due to insufficient filtration of the bad characters submitted the service display name string and there is not actual rate limitation for creating services.

# Type of exploitation

By using this attack you can accomplish it inside the joined domain, Windows Vista and the above

default policy doesn't allow remote users with administration privileges to connect until you turn off this policy

## Creating the technique

Now the idea is clear for me to start developing a working PoC that will leverage this weakness to deploy malicious binary and execute it at the end.

first, we need to build an automation module to interact with OpenSCManager API and create a service with base64 values segmented into 150 – 250 lengths. after creating the service, the program will deploy a piece of code to start the service, pop out the service display name content into the file, and then delete it to avoid duplicates. For more understanding, you can have a look at the following diagram.



By using a simple Powershell script we get the service display name value only and pip it out into a file.



### Putting the pieces together

our technique is almost ready, now it is time to use my development skills to build a complete tool to load the binary file and encode it into base64 format then divide each line into 150-250 character length. after that, it will communicate with OpenSCmanager to create a service and inject each line into lpDisplayname.

simultaneously, it will modify the value of each create service executable path as the following

C:\windows\system32\cmd.exe /c powershell -command "get-service SERVICENAME | select-string -expand |out-file tmp\_payload.txt

so let's consider you are going to deliver a 7 kb Meterpreter payload or cobalt strike, then the program will create/delete around 65 services including the final payload. or you can shorten the path of attack and way of compromise by switching to another attack mode using modify service only (-m option)

And finally, the tool will create a final service titled "final\_stage" which will stand for decoding the content of tmp\_payload.txt into a valid executable and execute it successfully.

#### PoC Usage

the usage of the tool is straightforward, it requires a target machine name and valid access credentials.

Poc.exe -s -t Machine -u USERNAME -p PASSWORD -d DOMAIN -f PATHOFLOCALBINARY

Recently, I have pushed another update for the tool by performing the attack with modify service name instead of creating and deleting multiple services, to test the alternative attack by

chopper.exe -m -t machine -u username -p password -d domain -f
pathoflocalbinary

#### Tool

https://github.com/lawrenceamer/TChopper

