

注意看，這些Windows的Potatoes太狠了！ 解析5種基於 MS-RPCE 的攻擊手法

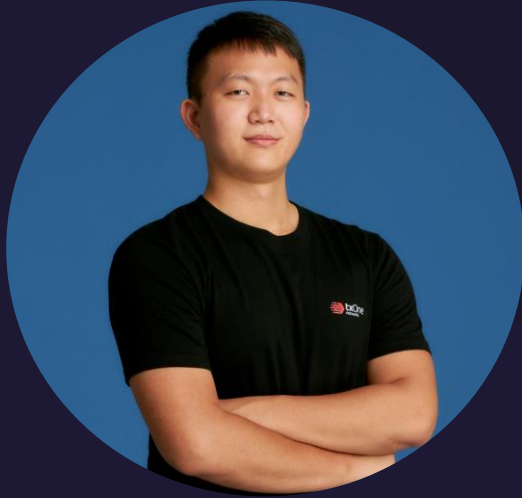
Hank Chen and Sheng-Hao Ma

PSIRT and Threat Research at TXOne Networks

May 11, 2023

| CYBERSEC2023

Who Are We?



Hank Chen

Threat Researcher
PSIRT and Threat Research

- Spoke at BlackHat USA, FIRST, HITCON, VXCON, and ThreatCon
- Instructor of Ministry of National Defense
- Teaching assistant of Cryptography and Information Security Course in Taiwan NTHU and CCoE Taiwan
- Member of CTF team 10sec and TSJ



Sheng-Hao Ma

Threat Researcher
PSIRT and Threat Research

- Spoke at Black Hat, DEFCON, HITB, VXCON, HITCON, ROOTCON, and CYBERSEC
- Instructor of CCoE Taiwan, Ministry of National Defense, Ministry of Education, and etc.
- The author of the popular security book "Windows API Warfare: The Definitive Guide for Malware Researchers"

Outline

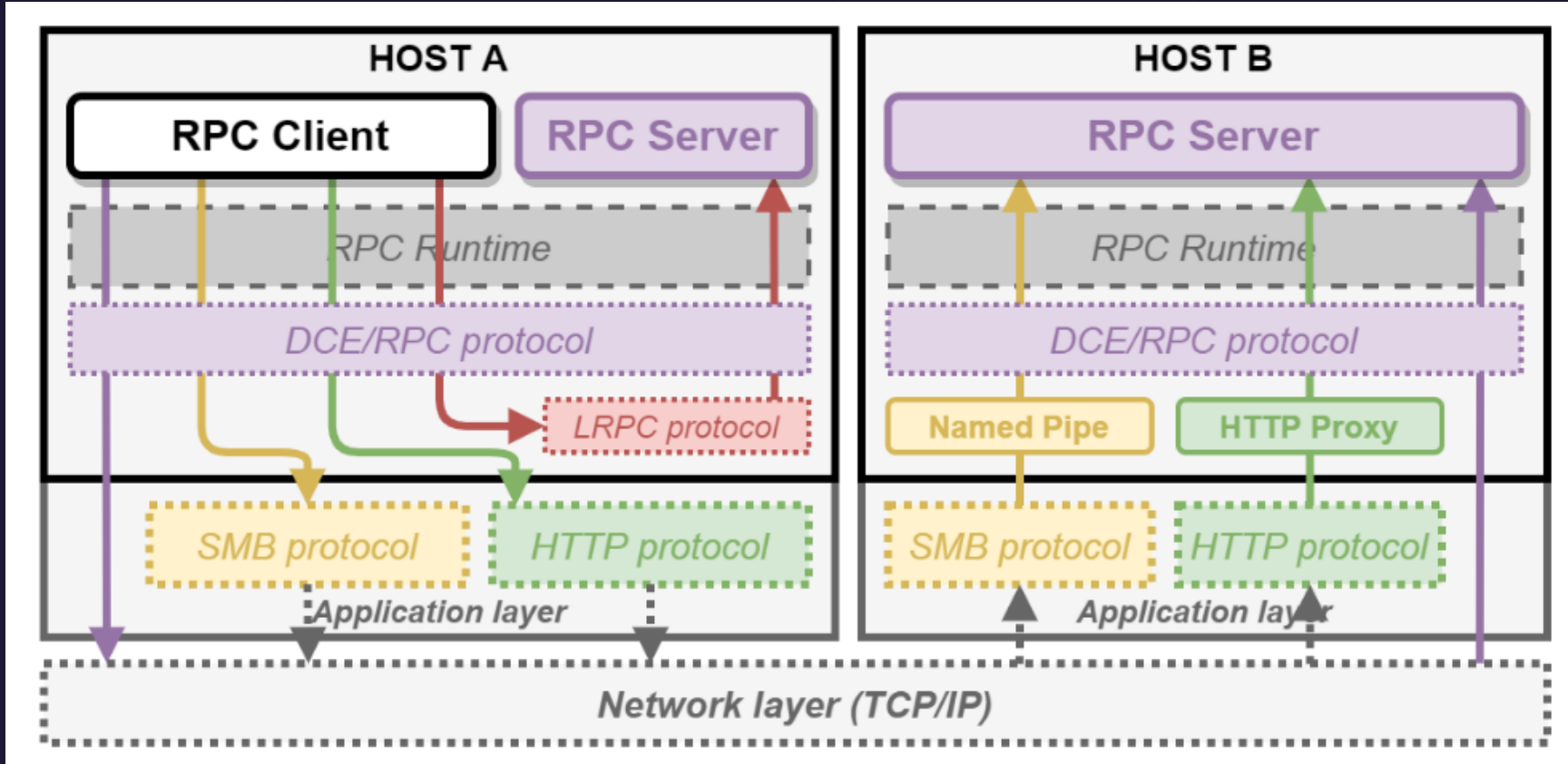
- The Knowledge about Potatoes
- How These Potatoes Work?
- How to Mitigate These Potatoes Attack?

The Knowledge about Potatoes

The Knowledge about Potatoes

- RPC
- DCOM
- Access Token
- Service Account

How RPC works?



<https://itm4n.github.io/from-rpcview-to-petitpotam/>

Register RPC Server by Windows APIs

RPC Client

RPC Server

RpcServerUseProtseqEp

- RPC connection method

RpcServerRegisterIf

- If allow local connection or not
- **Security callbacks**
- MIDL

RpcServerRegisterIf2

- If allow local connection or not
- **Security callbacks**
- MIDL

RpcServerRegisterIf3

- If allow local connection or not
- **Security callbacks**
- MIDL
- **Security Descriptor**

RpcServerRegisterAuthInfo

- **Authentication Service Provider**

Authentication Binding

RpcServerInqBindings

Dynamic Port

RpcEpRegister

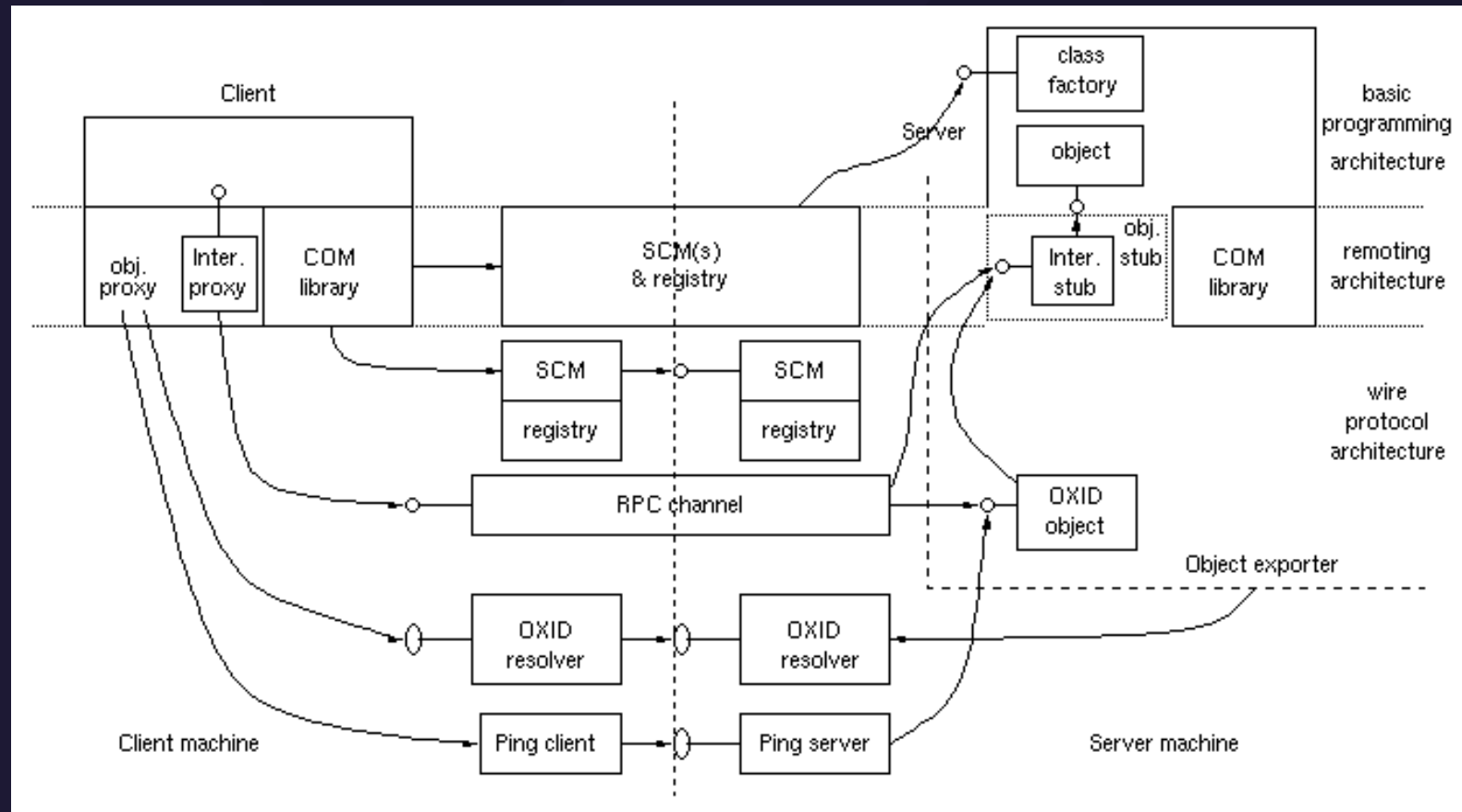
RpcServerListen

RpcBindingSetAuthInfoEx

- **Authentication level**
- **Authentication Service Provider**

[MS-DCOM]: Distributed Component Object Model (DCOM) Remote Protocol

- Built on top of MS-RPCE, DCOM extends the Component Object Model (COM) over a network by providing facilities for creating and activating objects, and for managing object references, object lifetimes, and object interface queries.



Abuse DCOM NTLM Authentication with OBJREF

DCOM Client

Implement IStorage IMarshal Interface
and **forge fake OBJREF** in IMarshal::MarshalInterface()



CoGetInstanceFromIStorage() will activate the target COM server and trigger the serialization of the **fake OBJREF**



DCOM Server

IMarshal::MarshalInterface() write marshalling data to IStream object



Unmarshal **fake OBJREF** & Extracts the Object Exporter ID (OXID)

OBJREF

signature	flags	iid
iid		u_objref
u_objref		

- signature (4 bytes): This MUST be set to the value 0x574f454d (**MEOW**).
- flags (4 bytes): This MUST be set to ONE of the following values.

Value	Meaning
OBJREF_STANDARD 0x00000001	u_objref MUST contain an OBJREF_STANDARD.
OBJREF_HANDLER 0x00000002	u_objref MUST contain an OBJREF_HANDLER.
OBJREF_CUSTOM 0x00000004	u_objref MUST contain an OBJREF_CUSTOM.
OBJREF_EXTENDED 0x00000008	u_objref MUST contain an OBJREF_EXTENDED.

OBJREF_STANDARD

signature	flags	iid
iid		std
std		
saResAddr		

- std (40 bytes): This MUST be an **STDOBJREF**.
- saResAddr (variable): A **DUALSTRINGARRAY** that MUST contain the network and security bindings for the object resolver service on the server.

STDOBJREF

signature	flags	iid	
iid		flags	cPublicRefs
oxid		oid	
ipid			
saResAddr			

- flags (4 bytes): This can be one of the following values. Any other value **MUST** be ignored by the client.
- cPublicRefs (4 bytes): The number of public references on the server object, which **MUST** be released later.
- oxid (8 bytes): This **MUST** be an OXID identifying the object exporter that contains the object.
- oid (8 bytes): This **MUST** be an OID identifying the object.
- ipid (16 bytes): This **MUST** be an IPID identifying a specific interface on the object.

OBJREF_STANDARD

DUALSTRINGARRAY

wNumEntries	wSecurityOffset	StringBinding
		\x00 \x00
SecBinding		
		\x00 \x00

- wNumEntries (2 bytes): The number of unsigned shorts from the first entry in the StringBinding array to the end of the buffer.
- wSecurityOffset (2 bytes): The number of unsigned shorts from the first entry in the StringBinding array to the first entry in the SecBinding array.
- **StringBinding** (variable): An array of one or more STRINGBINDING structures that SHOULD be ordered in decreasing order of preference by the object server.
- **SecBinding** (variable): An array of one or more SECURITYBINDING structures that SHOULD be ordered in decreasing order of preference by the object server.

Abuse DCOM NTLM Authentication with OBJREF

DCOM Client

Implement IStorage IMarshal Interface
and **forge fake OBJREF** in IMarshal::MarshalInterface()

CoGetInstanceFromIStorage() will activate the target
COM server and trigger the serialization of the **fake
OBJREF**

RPC Server (IObjectExporter)

IObjectExporter::ResolveOxid2() return an arbitrary
RPC binding string for a fake COM object

RPC Server (IRemUnknown2)

Force DCOM Server call IRemUnknown2::RemRelease()
and Impersonate privileged client token

DCOM Server

IMarshal::MarshalInterface() write marshalling data to
IStream object

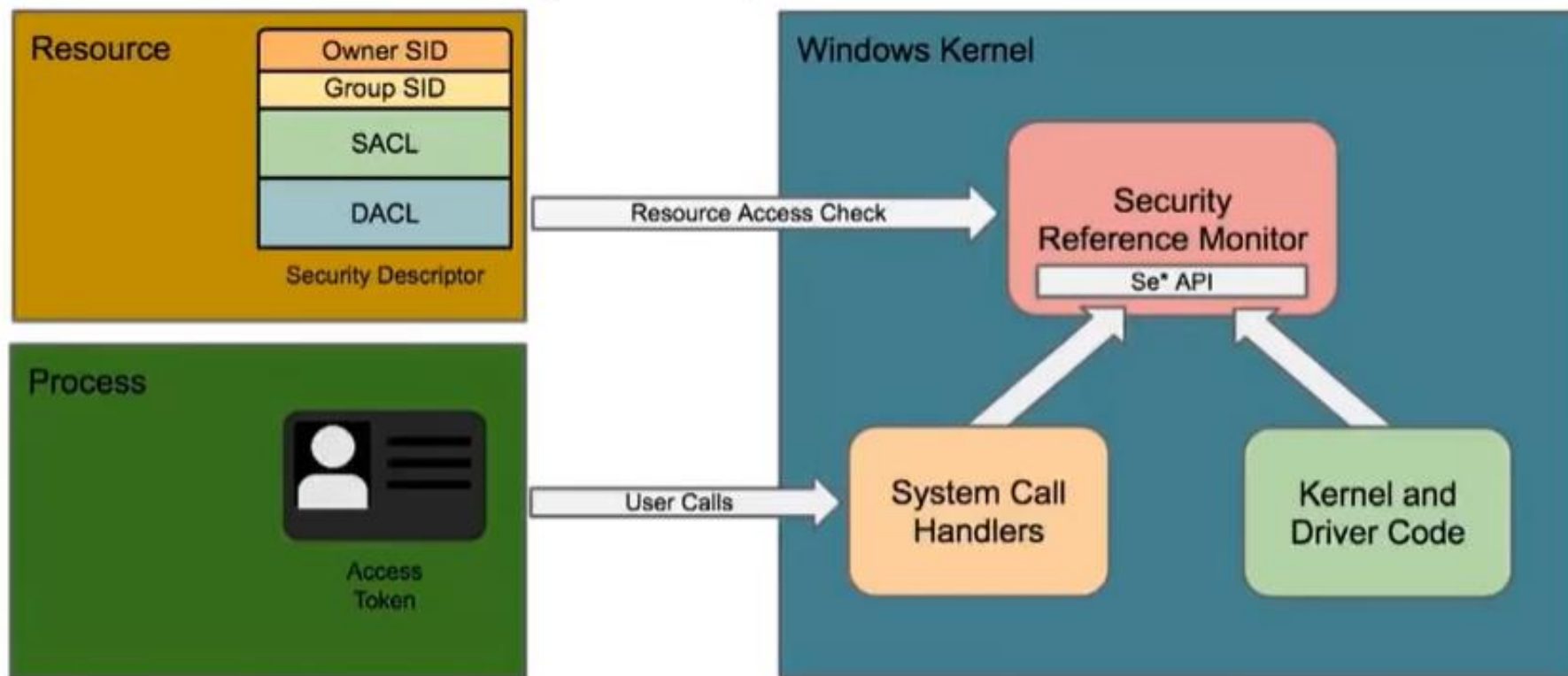
Unmarshal **fake OBJREF** & Extracts the Object Exporter ID (O
XID)

Contacts the **OXID resolver** service specified by the RPC bind
ing information of the COM server which hosts the object in
the OBJREF

Establishes a connection to the RPC endpoint to access the
object's interfaces

James Forshaw - Social Engineering The Windows Kernel: Finding And Exploiting Token Handling Vulnerabilities

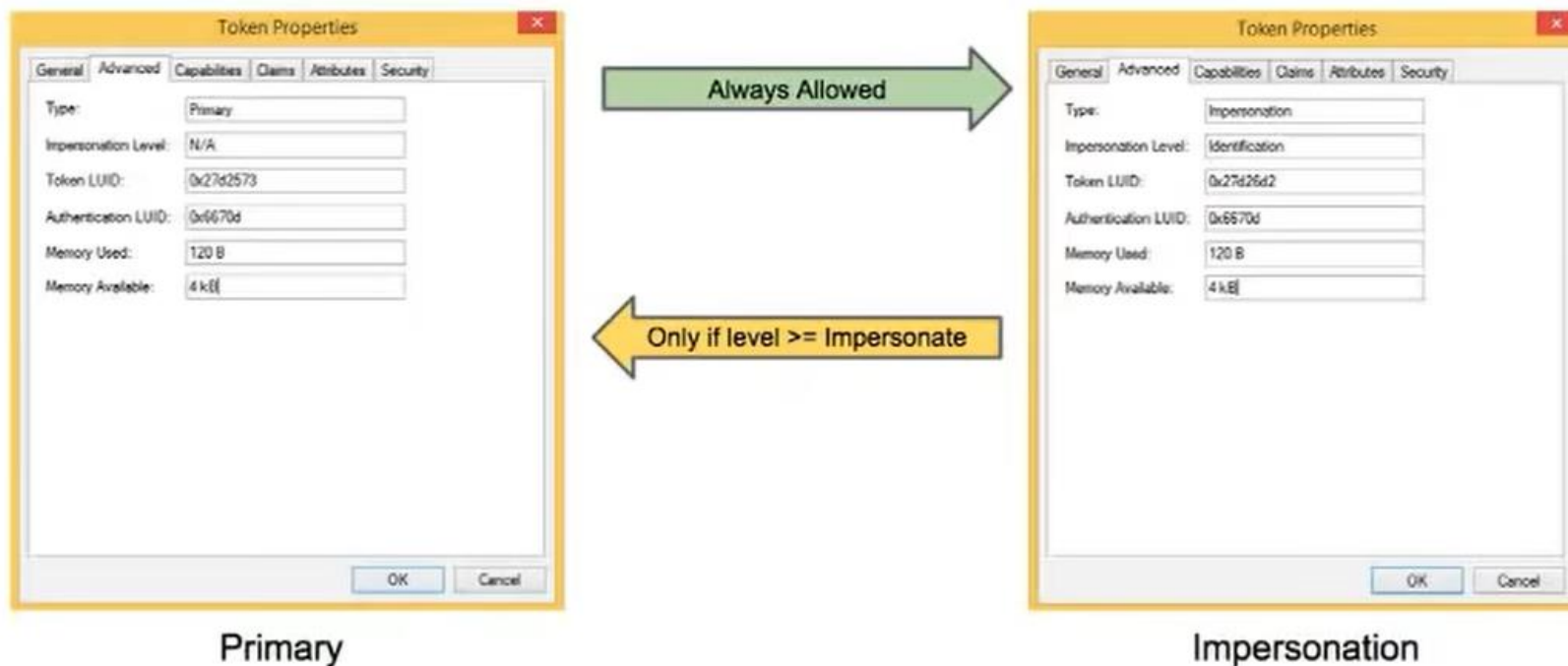
Windows Security Components



James Forshaw - Social Engineering The Windows Kernel: Finding And Exploiting Token Handling Vulnerabilities

Token Duplication

Can duplicate token's between types if have TOKEN_DUPLICATE access on handle
Duplicated Token can be referenced with any set of access rights



James Forshaw @tiranid0

15

SSPI Function from Lsass

Function	Description
AcceptSecurityContext (General)	Used by a server to create a <i>security context</i> based on an opaque message received from a client.
ApplyControlToken	Applies a supplemental security message to an existing security context.
CompleteAuthToken	Completes an authentication token. This function is used by protocols, such as DCE, that need to revise the security information after the transport application has updated some message parameters.
DeleteSecurityContext	Frees a security context and associated resources.
FreeContextBuffer	Frees a memory buffer allocated by a security package.
ImpersonateSecurityContext	Impersonates the security context to appear as the client to the system.
InitializeSecurityContext (General)	Used by a client to initiate a security context by generating an opaque message to be passed to a server.
QueryContextAttributes (General)	Enables a transport application to query a <i>security package</i> for certain <i>attributes</i> of a security <i>context</i> .
QuerySecurityContextToken	Obtains the <i>access token</i> for a client <i>security context</i> and uses it directly.
SetContextAttributes	Enables a transport application to set <i>attributes</i> of a security <i>context</i> for a <i>security package</i> . This function is supported only by the Schannel security package.
RevertSecurityContext	Allows a <i>security package</i> to discontinue the impersonation of the caller and restore its own <i>security context</i> .

James Forshaw - Social Engineering The Windows Kernel: Finding And Exploiting Token Handling Vulnerabilities

NTLM Negotiation

- LSASS exposes APIs to do network authentication
- Can get localhost NTLM authentication using redirected WebDAV
- Creates an impersonation level token even for a normal user

```
// Init a AV scan to \\localhost\\fake\\fake
BYTE* type1 = GetType1();
BYTE* challenge;

AcceptSecurityContext(hContext type1, &challenge);
BYTE* result = SetChallengeAndGetResult(challenge);
AcceptSecurityContext(hContext, result, ...);

QuerySecurityContextToken(hContext, &hToken);
```

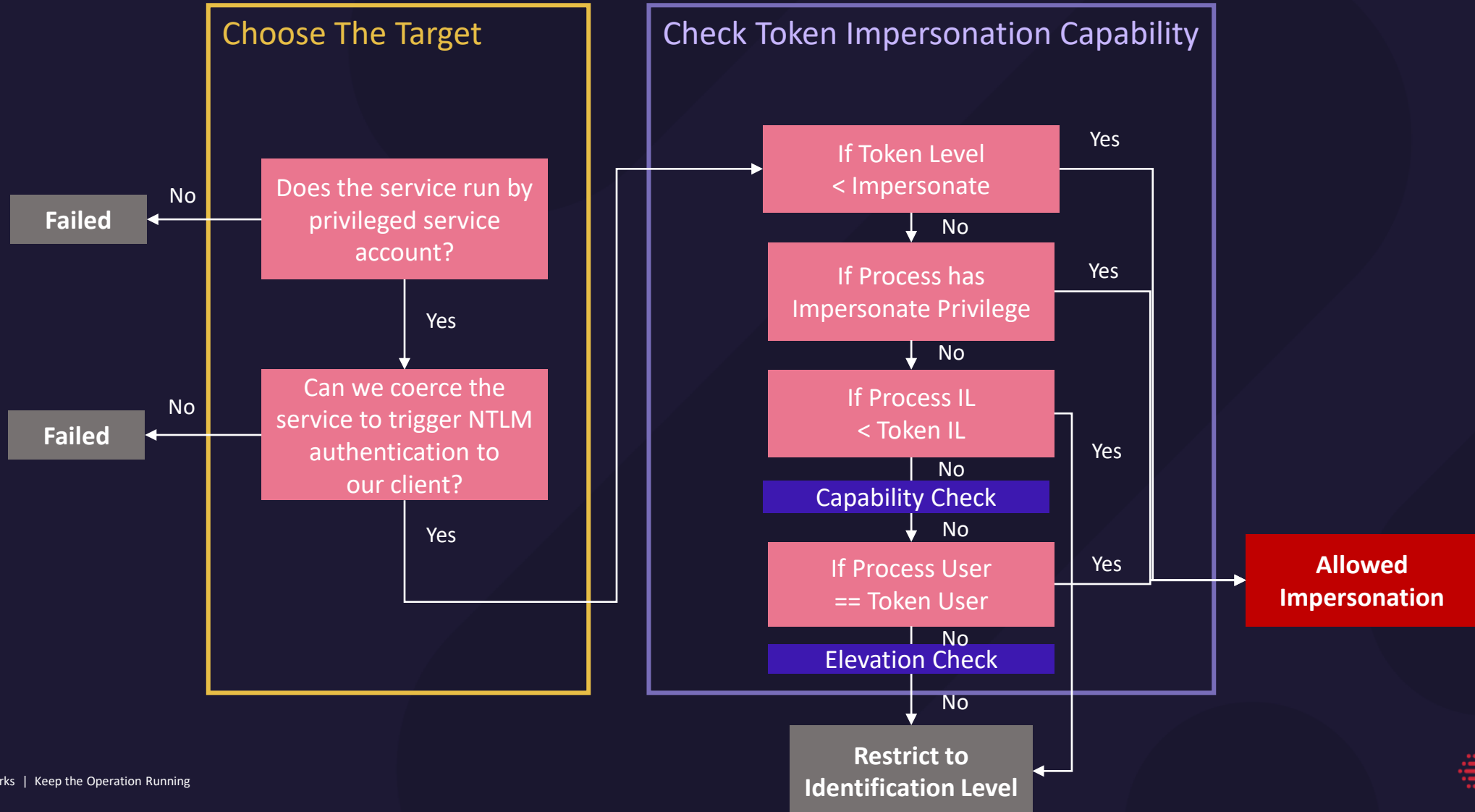
James Forshaw @tiraniddo

26

Windows Service

- Windows Service Accounts
 - Windows Service must be run with a Service Account User
 - Service Account types
 - Local System
 - Local Service / Network Service Accounts
 - Managed Service & Virtual Accounts
- Windows Service Hardening (WSH)
 - Until Windows Server 2003/XP every service was run as SYSTEM
 - Compromised service == Compromised machine
 - Limited Service Accounts, Reduced Privileges, Write-Restricted Token, ...

How Attackers Think ...



Potatoes

From a Windows Service Accounts to NT AUTHORITY\SYSTEM

A Brief History of Potatoes

Issue 325: Windows: DCOM DCE/RPC Local NTLM Reflection Elevation of Privilege

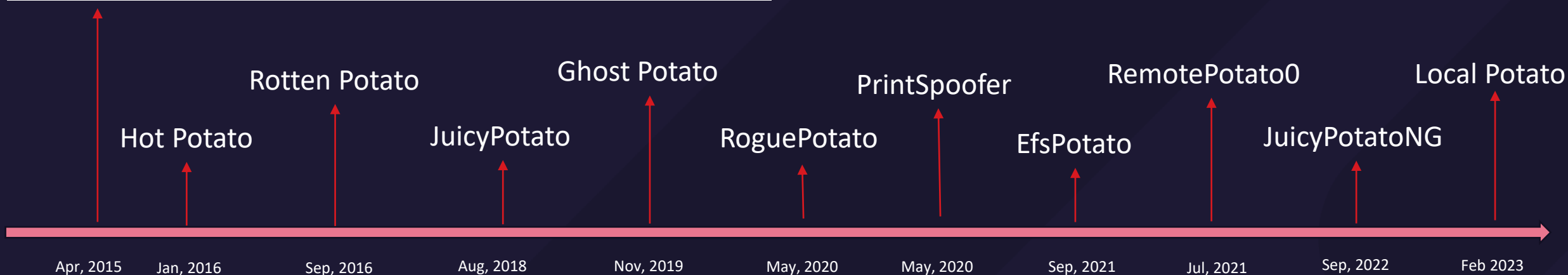
Reported by forshaw@google.com on Fri, Apr 10, 2015, 2:42 AM GMT+8

Project Member

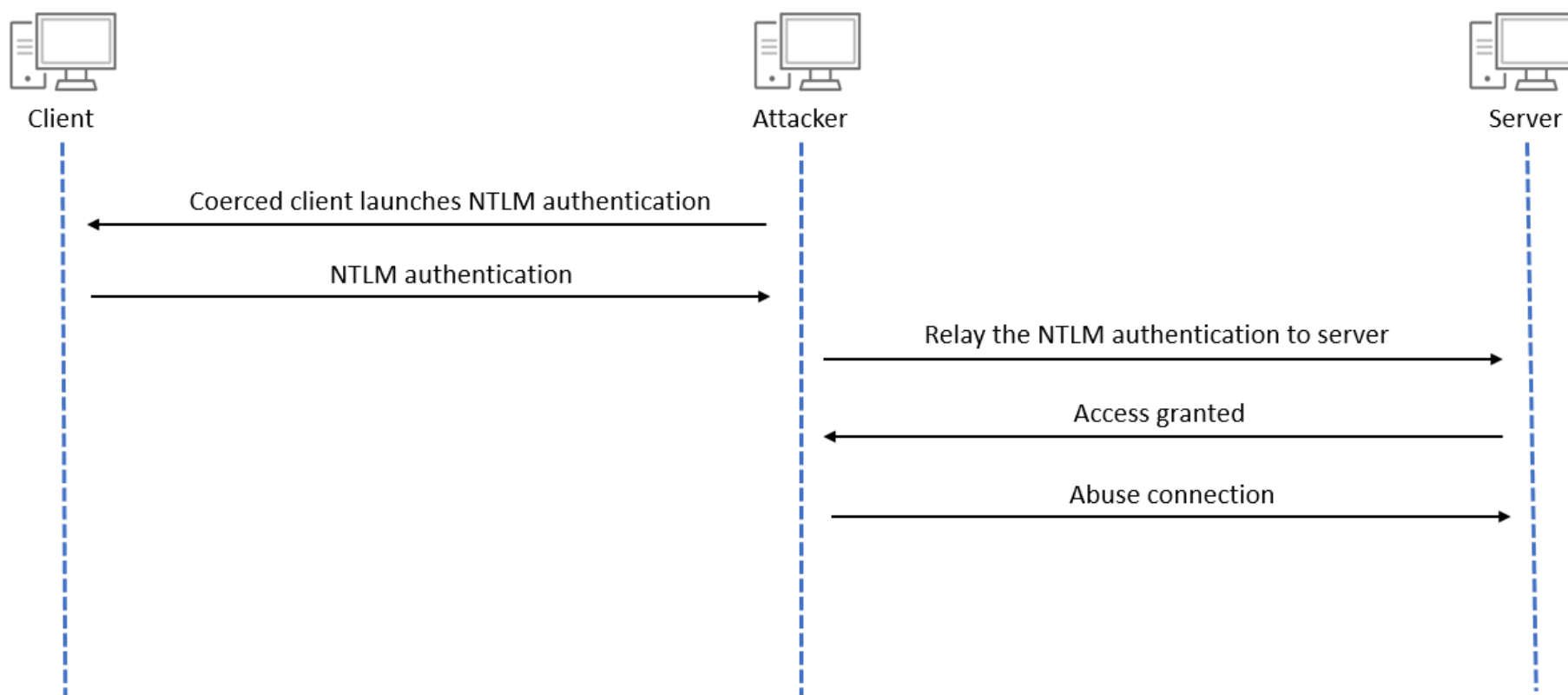
Windows: DCOM DCE/RPC Local NTLM Reflection Elevation of Privilege

Platform: Windows 8.1 Update (not tested on Windows 7, 10)

Class: Elevation of Privilege



NTLM Relay 101



The Techniques Adopted by Potatoes

- Ghost Potato
 - NTLM Challenge Cache Timeout (CVE-2019-1384), ntlmrelayx
- Hot Potato
 - Windows Update Service <-> SMB
 - NBNS Spoofing, WPAD Poisoning, NTLM Relay
- Juicy Potato
 - DCOM <-> RPC
 - NTLM Relay

- Efs Potato Impersonate
 - EFSRPC -> Named Pipe
 - NamedPipe Path Resolution, PetitPotam

- Rogue Potato Impersonate
 - DCOM -> Named Pipe
 - External OXID Resolving, Fake Epmapper, NamedPipe Path Resolution

- RemotePotato0
 - DCOM -> HTTP <-> LDAP
 - External OXID Resolving, IRemUnknown2::RemRelease, ntlmrelayx

- JuicyPotatoNG Impersonate
 - DCOM -> RPC
 - OBJREF Moniker Binding, SSPI Hook

- Local Potato
 - DCOM -> RPC <-> SMB
 - OBJREF Moniker Binding, SSPI Hook, Swap NTLM Context Handle (CVE-2023-21746)

Hot Potato

Victim's Computer

SMB
Server

6. **Relay** NTLM Negotiate

7. NTLM Challenge

10. NTLM Auth

11. PsExec

HotPotato.exe

Fake WPAD Proxy Server

1. NBNS spoofer exhaust all UDP port to deny DNS query
2. Masquerade as WPAD and Response WPAD host has IP address 127.0.0.1

3. HTTP request

4. Response 401 for NTLM authentication

5. NTLM Negotiate

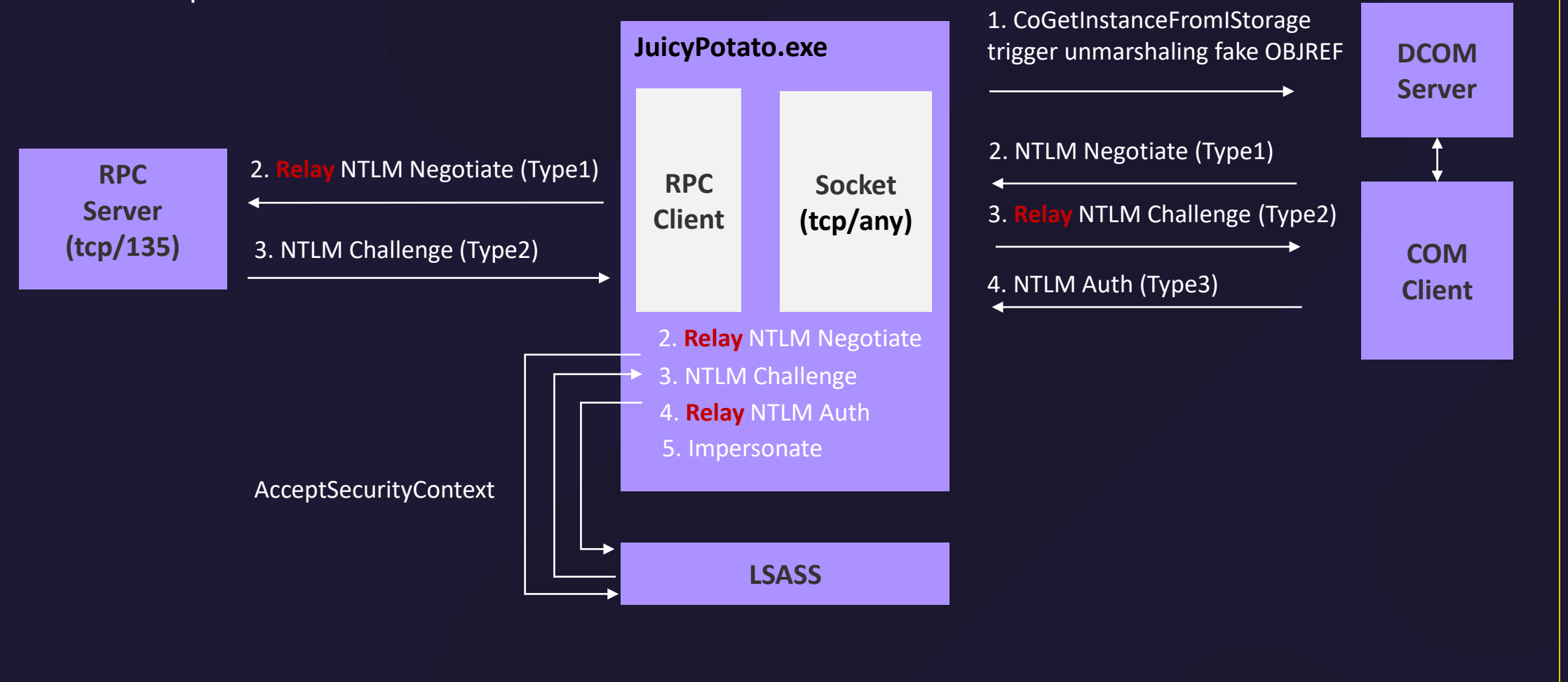
8. **Relay** NTLM Challenge

9. NTLM Auth

Windows
Update
Service

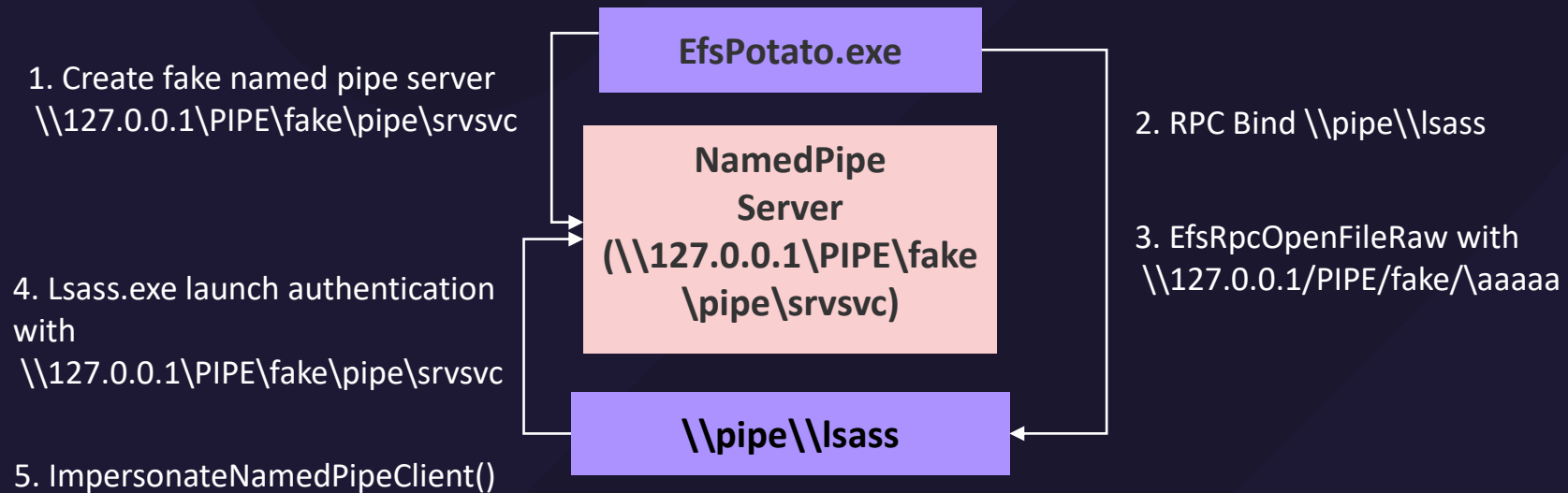
Juicy Potato

Victim's Computer

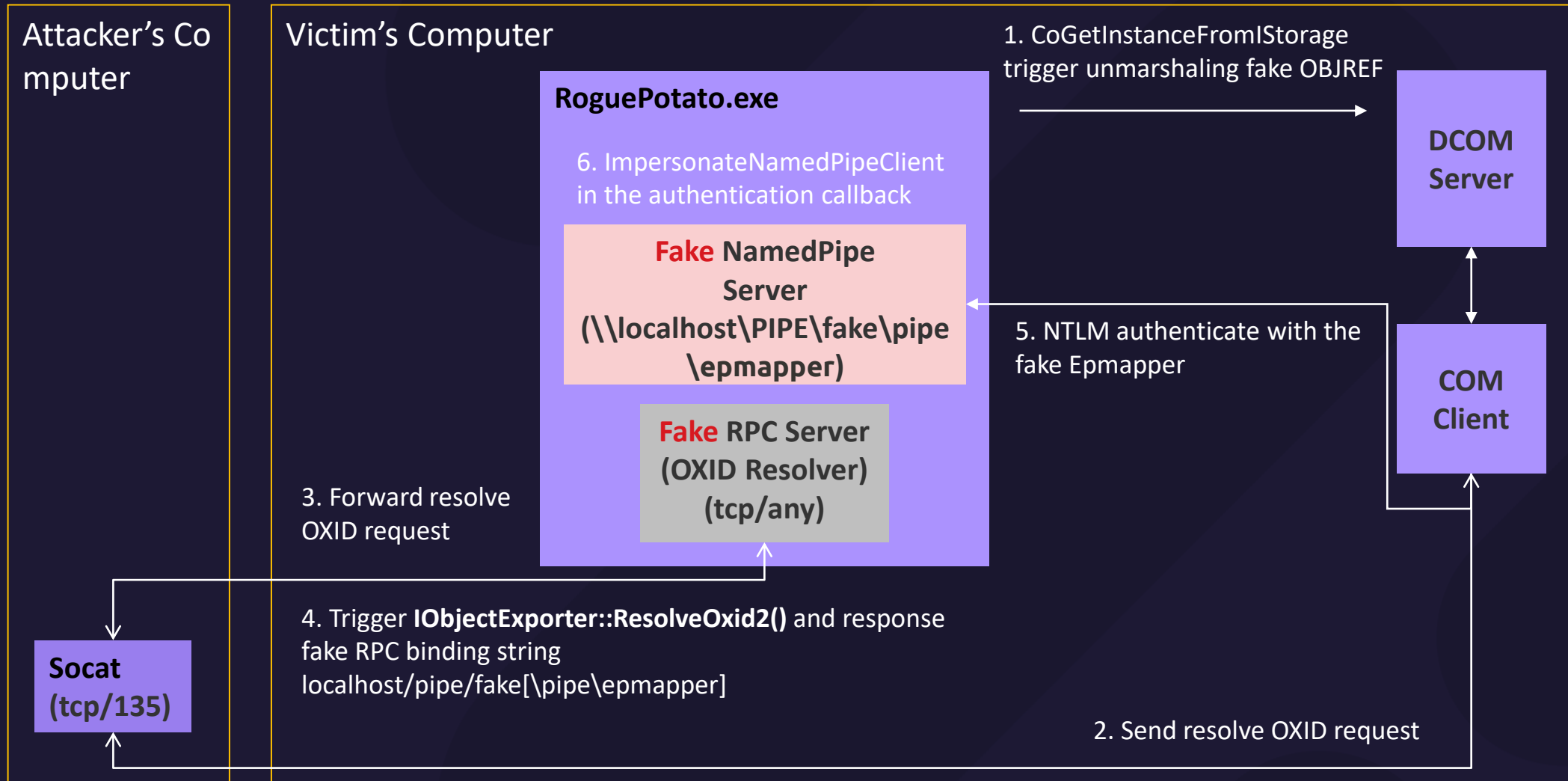


Efs Potato

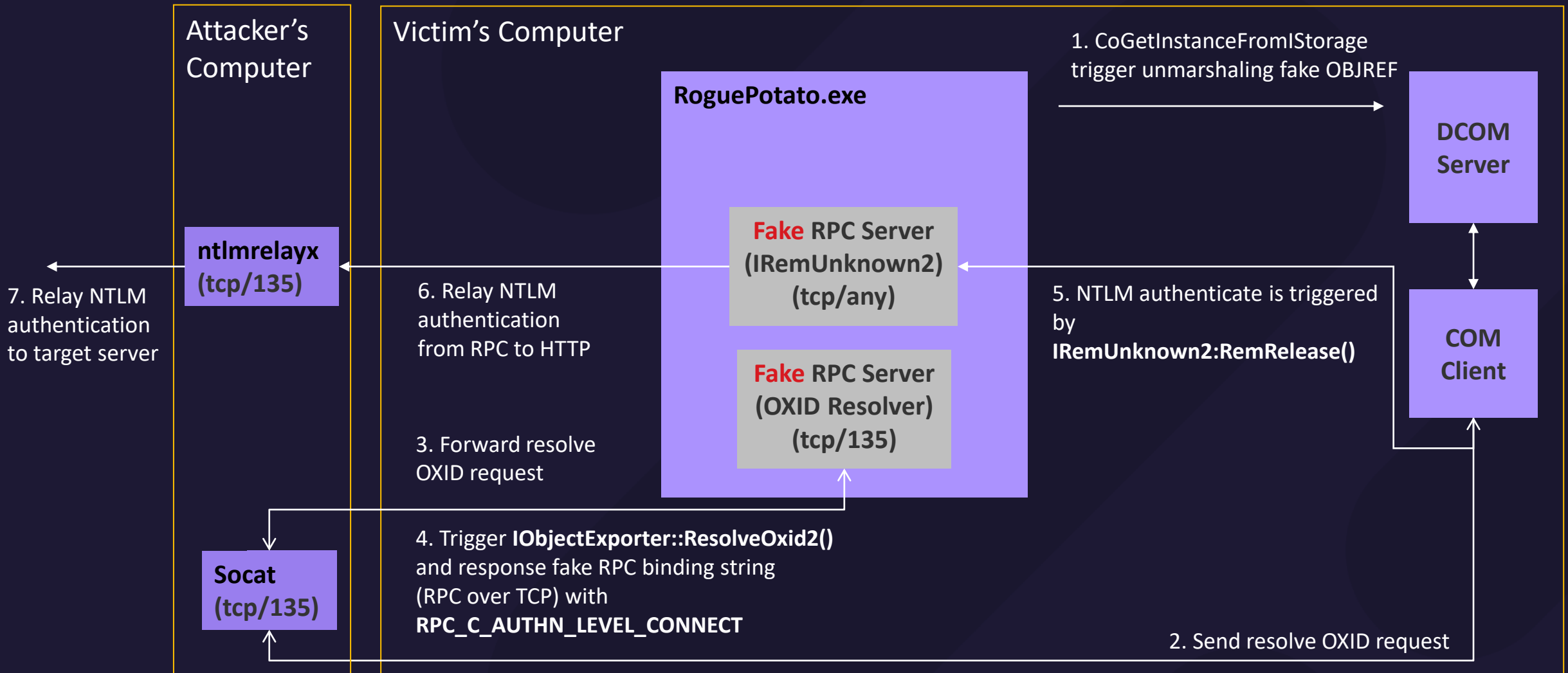
Victim's Computer



Rogue Potato

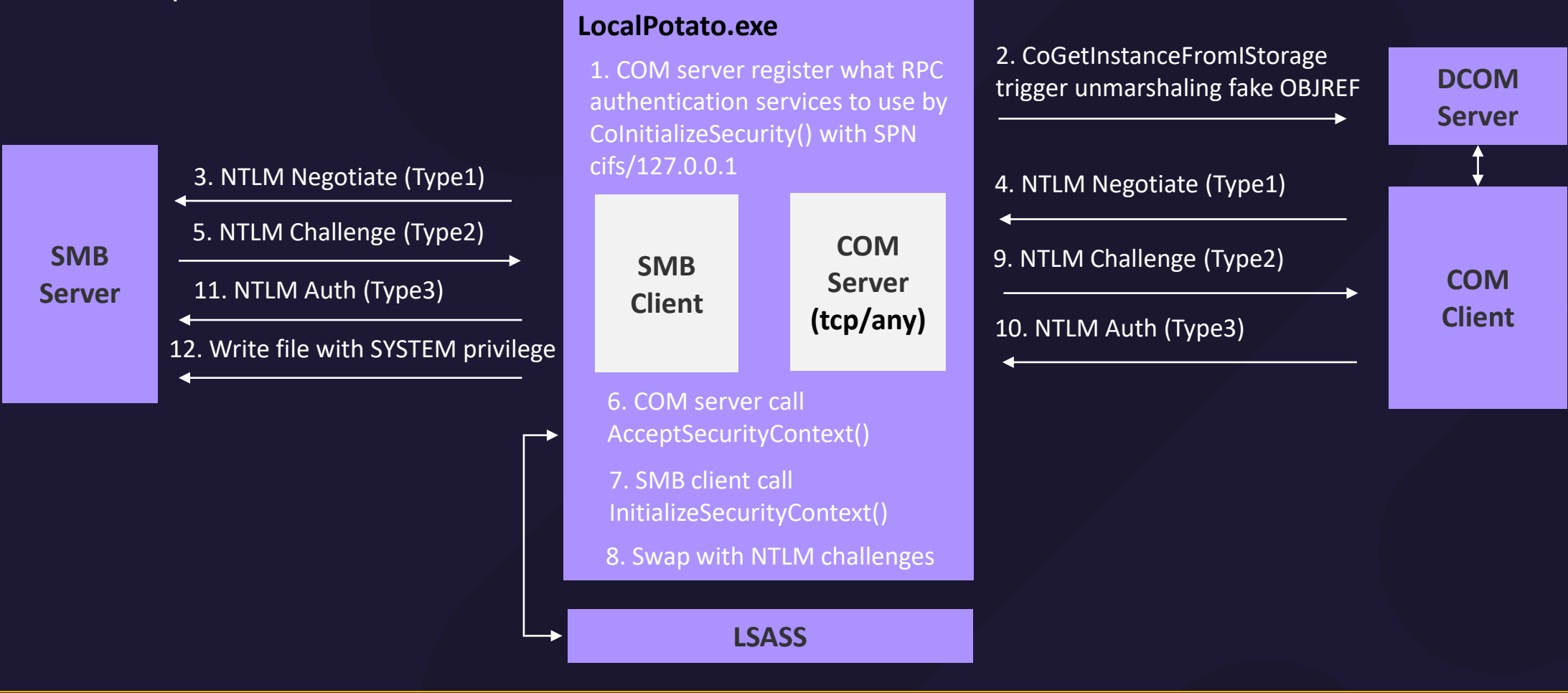


RemotePotato0



Local Potato

Victim's Computer



How to Mitigate These Potatoes Attack?

For Example: Yara Rule for RemotePotato0

```
rule SentinelOne_RemotePotato0_privesc {
  meta:
    author = "SentinelOne"
    description = "Detects RemotePotato0 binary"
    reference = "https://www.sentinelone.com/wp-content/uploads/relaying-potatoes-dce-rpc-ntlm-relay-eop"
  strings:
    $import1 = "CoGetInstanceFromIStorage"
    $storage_clsids = "{00000306-0000-0000-c000-000000000046}" nocase wide ascii
    $meow_header = { 4d 45 4f 57 }
    $clsid1 = "{11111111-2222-3333-4444-555555555555}" nocase wide ascii
    $clsid2 = "{5167B42F-C111-47A1-ACC4-8EABE61B0B54}" nocase wide ascii
  condition:
    (uint16(0) == 0x5A4D) and $import1 and $storage_clsids and $meow_header and 1 of ($clsid*)
}
```

<https://www.sentinelone.com/labs/relaying-potatoes-another-unexpected-privilege-escalation-vulnerability-in-windows-rpc-protocol/>

What does James Forshaw say?

Experiment Results

We now need to do our basic analysis of the results. Let's start with calculating the percentage of writable resources for each token type relative to the total number of resources. From my single experiment run I got the following table:

Token	Writable	Writable (WR)	Total
Control	99.83%	N/A	13171
Network Service	65.00%	0.00%	300
Local Service	62.89%	0.70%	574

As we expected the control token had almost 100% of the owned resources writable by the user. However for the two service accounts both had over 60% of their owned resources writable when using an unrestricted token. That level is almost completely eliminated when using a WR token, there were no writable resources for NS and only 4 resources writable from LS, which was less than 1%. Those 4 resources were just Events, from a service perspective not very exciting though there were ACL'ed to everyone which is unusual.

<https://www.tiraniddo.dev/2020/01/empirically-assessing-windows-service.html>

Mitigations from Antonio Cocomazzi

1. Change the sid type of the service to “WRITE RESTRICTED”
 - `sc.exe sidtype SampleService restricted`
2. Use virtual service accounts or create your own
 - `sc.exe config SampleService obj= "NT SERVICE\SampleService"`
3. Remove the impersonation privileges by specifying the only required privileges for the service (Least-Privilege)
 - `sc.exe privs SampleService SeChangeNotifyPrivilege/SeCreateGlobalPrivilege`

Review the Interfaces Security - RpcView

RpcView

File Options View Filter Help

Endpoints

Pid	Protocol	Name
1064	ncalrpc	OLE9AFE8DBE37BF47CBDE773A604E93
1064	ncalrpc	LRPC-6bd028caaa91eccfb6
1100	ncalrpc	OLE6C173C09D9CF6E4BEA9CAD98E847
1636	ncalrpc	OLE3CF16BF98DCC5DE18E2674EA39E6
3956	ncalrpc	webcache_{031b98cf-4a69-4c31-ab42-fd9b3c199...
3956	ncalrpc	PlaySoundKRpc1
4320	ncalrpc	OLE4E5AE7B49FB814187E3D1298CDF9
1996	ncalrpc	OLE1EC4AF3D99B9B2B094E947C7E80D
8040	ncalrpc	OLED58740D58AEFA2871BF219D5AB84
5608	ncalrpc	OLEBF76C364756DACEF7FFA776042B3
7304	ncalrpc	OLEE5E74B3075455453019A108BE2C9
7512	ncalrpc	OLED91F1C0EABE9E965182A94C4E9C8
7708	ncalrpc	OLE67337B24304F8955BB952B9AF257
8430	ncalrpc	OLE70E3C0BFC707003F5E5EDDC30F5D00

Processes

Name	Pid	Pt
svchost.exe	7716	
svchost.exe	9528	
lsass.exe	688	
fontdrvhost.exe	832	
csrss.exe	532	
winlogon.exe	628	
dwm.exe	536	
fontdrvhost.exe	784	
MusNotifylcon.exe	1780	C:
GoogleCrashHandler64.exe	2004	
explorer.exe	4320	C:
msedge.exe	3680	C:

Interfaces

Stub	Callback	Name	Base	Locatic
interpreted	0x00007ffe0063...		0x00007ffe0062...	C:\Win
interpreted	0x00007ffe0063...		0x00007ffe0062...	C:\Win
interpreted	0x00007ffe0063...		0x00007ffe0062...	C:\Win
interpreted	0x00007ffe0063...		0x00007ffe0062...	C:\Win
interpreted	0x00007ffe0063...		0x00007ffe0062...	C:\Win
interpreted	0x00007ffe0063...		0x00007ffe0062...	C:\Win
interpreted	0x00007ffe01ff0...		0x00007ffe01f20...	C:\Win
interpreted	0x00007ffe1809...		0x00007ffe17fc0...	C:\Win
interpreted	0x00007ffe1809...		0x00007ffe17fc0...	C:\Win
interpreted	0x00007ffe1809...		0x00007ffe17fc0...	C:\Win
interpreted	0x00007ffe1809...		0x00007ffe17fc0...	C:\Win
interpreted	0x00007ffe1809...		0x00007ffe17fc0...	C:\Win

Procedures

Index	Name	Address	Format
-------	------	---------	--------

Endpoints: 54/54 Interfaces: 66/66 Processes: 176/176

Review The Security of Interfaces - OleViewDoNet

The screenshot shows the OleView.NET v1.11 - 64bit application window. The 'CLSIDs' tab is active, displaying a list of CLSIDs. The selected CLSID is 00000301-a8f2-4877-ba0a-fd2b6645fb94, which corresponds to the PSFactoryBuffer interface. The properties of this interface are displayed in the right pane, including the executable path, process ID, application ID, access permissions, LRPC permissions, user, security flags, and STA HWND.

Property	Value
Executable Path:	C:\Windows\System32\RemoteAppLifetimeManager.exe
Process ID:	22924
AppID:	CD57F3A9-8247-496F-B51F-00EAE15128BE
Access Permissions:	O:PSG:BUD:(A;;CCDCWP;;;WD)(A;;CCDCWP;;;AN)
LRPC Permissions:	D:(A;;0xefff3ffff;;;WD)(A;;0xefff3ffff;;;AN)
User	NT AUTHORITY\NETWORK SERVICE
Security Flags:	Capabilities: APPID, Authn Level: CONNECT, Imp Level: IDENTIFY, Unmarshal Policy: HYBRID
STA HWND:	0x0

Review the Interfaces Security – Token Viewer

cmd.exe:6264 - User DESKTOP-G0BDQ1E\wac - TokenId 0...

Main Details Groups Privileges Default Dacl Misc Operations Security

Token

User: DESKTOP-G0BDQ1E\wac

User SID: S-1-5-21-72394402-3803442102-2434996194-1001

Token Type: Primary

Impersonation Level: N/A

Token ID: 00000000-01844A7A

Authentication ID: 00000000-00030454

Origin Login ID: 00000000-000003E7

Modified ID: 00000000-01844A6D

Integrity Level: High Set Integrity Level

Session ID: 1

Elevation Type: Full Linked Token

Is Elevated: True

Source

Name: User32

Id: 00000000-000303DE

Token Viewer

Processes Threads Handles Logon User Services

Current Process Name Filter: Hide Unrestricted Apply Filter

Process ID	Name	User	Integrity Level	Restricted	App Container
228	ctfmon.exe	DESKTOP-G0BDQ1E\wac	High	False	False
1460	cmd.exe	DESKTOP-G0BDQ1E\wac	High	False	False
2460	proccp64.exe	DESKTOP-G0BDQ1E\wac	High	False	False
3020	cmd.exe	DESKTOP-G0BDQ1E\wac	High	False	False
3192	conhost.exe	DESKTOP-G0BDQ1E\wac	High	False	False
6264	cmd.exe	DESKTOP-G0BDQ1E\wac	High	False	False
6848	cmd.exe	DESKTOP-G0BDQ1E\wac	High	False	False
6968	conhost.exe	DESKTOP-G0BDQ1E\wac	High	False	False
3396	SearchUI.exe	DESKTOP-G0BDQ1E\wac	Low	False	True
4012	ShellExperienceHost.exe	DESKTOP-G0BDQ1E\wac	Low	False	True
4692	SkypeHost.exe	DESKTOP-G0BDQ1E\wac	Low	False	True
4888	MicrosoftEdge.exe	DESKTOP-G0BDQ1E\wac	Low	False	True
4896	LockApp.exe	DESKTOP-G0BDQ1E\wac	Low	False	True
5484	MicrosoftEdgeCP.exe	DESKTOP-G0BDQ1E\wac	Low	False	True
5492	MicrosoftEdgeCP.exe	DESKTOP-G0BDQ1E\wac	Low	False	True
6788	Microsoft.Photos.exe	DESKTOP-G0BDQ1E\wac	Low	False	True
712	dllhost.exe	DESKTOP-G0BDQ1E\wac	Medium	False	False
1128	TokenViewer.exe	DESKTOP-G0BDQ1E\wac	Medium	False	False

References

- Itm4n
 - Website: <https://itm4n.github.io/>
 - Keywords: RpcView, PrintSpoofer, PetimPotam, Efs Potato
- Foxglovesecurity
 - Website: <https://foxglovesecurity.com/>
 - Keywords: Hot Potato, Rotten Potato
- **Andrea Pierini & Antonio Cocomazzi**
 - Website: <https://decoder.cloud/>, <https://www.localpotato.com/>
 - Keywords: Rogue Potato, RemotePotato0, JuicyPotatoNG, Local Potato
- ophe
 - Website: <https://ohpe.it/>
 - Keywords: JuicyPotato
- shenaniganslabs
 - Website: <https://shenaniganslabs.io/>
 - Keywords: Ghost Potato
- **James Forshaw**
 - Website: <https://googleprojectzero.blogspot.com/>
 - Token Impersonation, SSPI Hook, OXID Resolver, DCOM Relay

Thanks for your listening!

OT Cybersecurity. **Simplified.**



Keep the Operation
Running



掃描QR Code到TXOne攤位#C240玩扭蛋換好禮