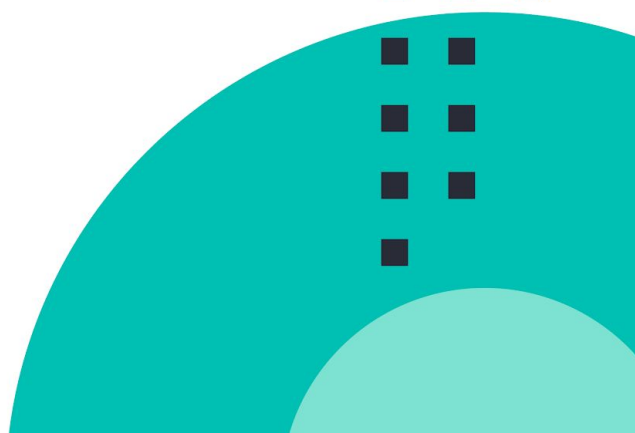




Detecting Access Token Manipulation

William Burgess
Security Research Engineer
[@joehowwolf](https://twitter.com/joehowwolf)



#whoami

#whoami

- Security researcher @ **Elastic** (formerly **ENDGAME**)
- Former security consultant at **MWR**
- Interests include Windows internals and everything undocumented, reverse engineering, and development of endpoint protections
- [@joehowwolf](#)

Objectives

Objectives

Help defence practitioners understand:

- How access tokens **work** in Windows environments
- How attackers **abuse** legitimate Windows functionality to move laterally and compromise entire Active Directory domains
- Their capability to **detect** and **respond** to access token manipulation within their environment

Agenda

1 Windows Security Internals

2 How Attackers Abuse Access Tokens

3 Detecting Access Token Manipulation

Agenda

1 Windows Security Internals

2 How Attackers Abuse Access Tokens

3 Detecting Access Token Manipulation

Agenda

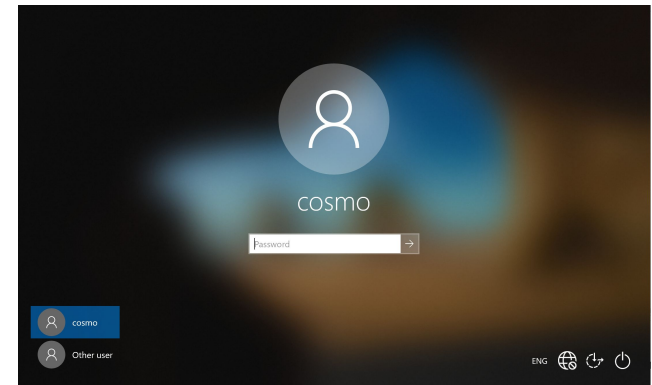
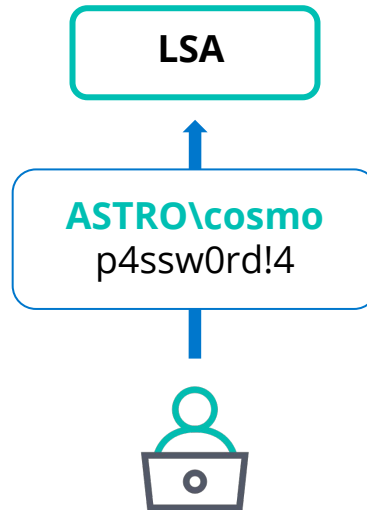
1 Windows Security Internals

2 How Attackers Abuse Access Tokens

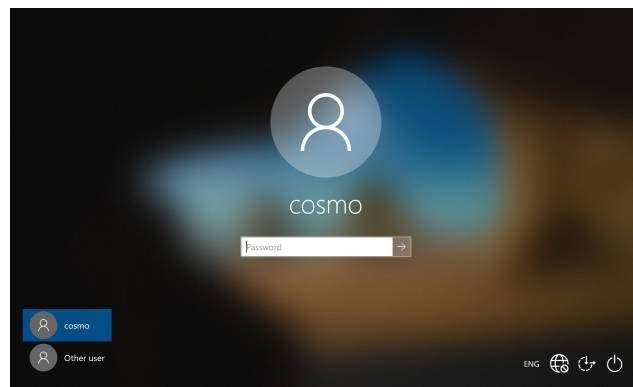
3 Detecting Access Token Manipulation

Windows Security Internals: Logon sessions and Access Tokens

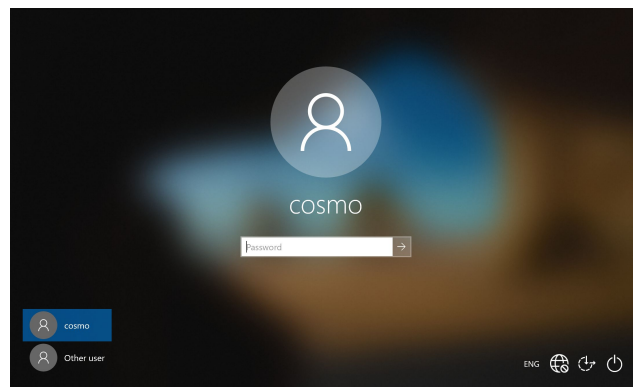
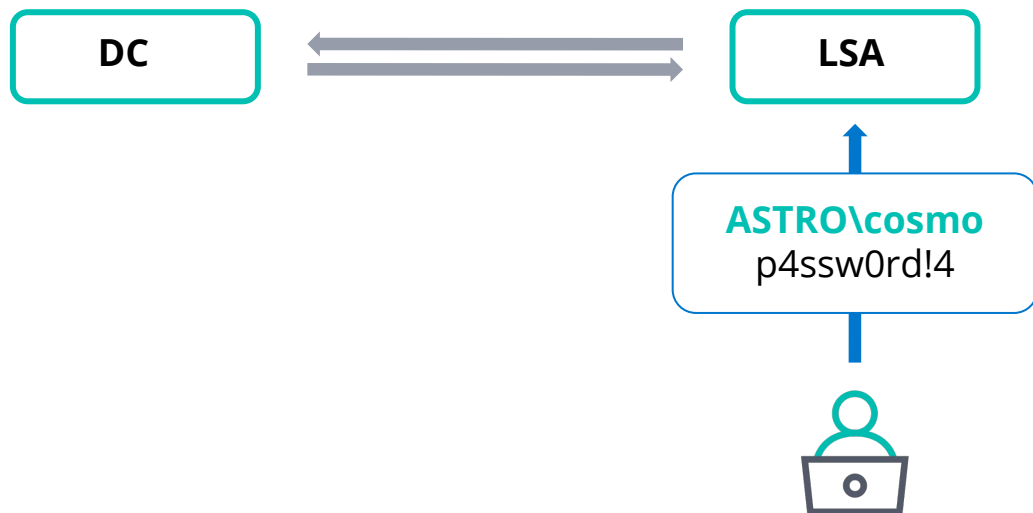
Logon Sessions and Access Tokens



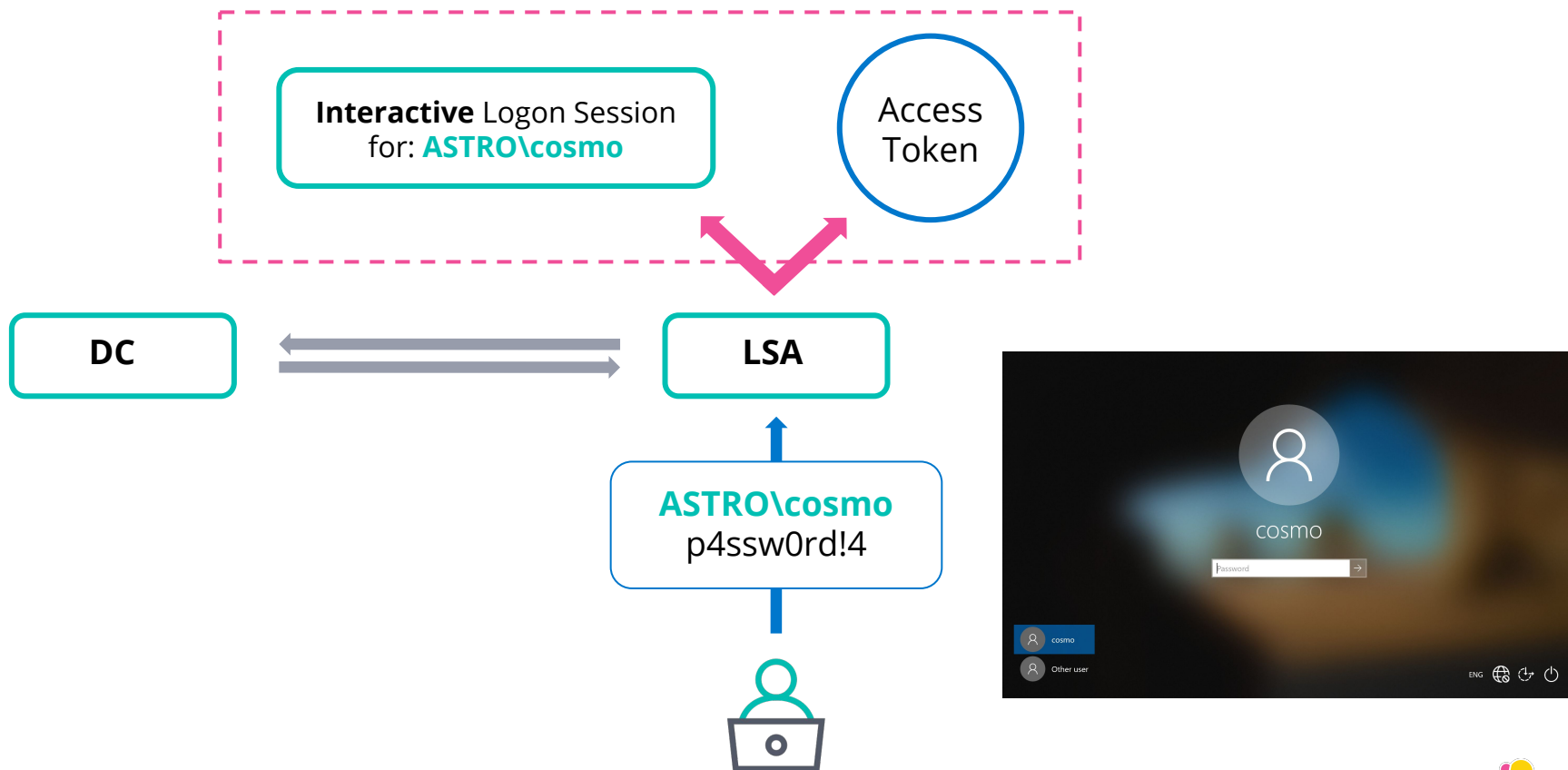
Logon Sessions and Access Tokens



Logon Sessions and Access Tokens



Logon Sessions and Access Tokens



Logon Sessions and Access Tokens

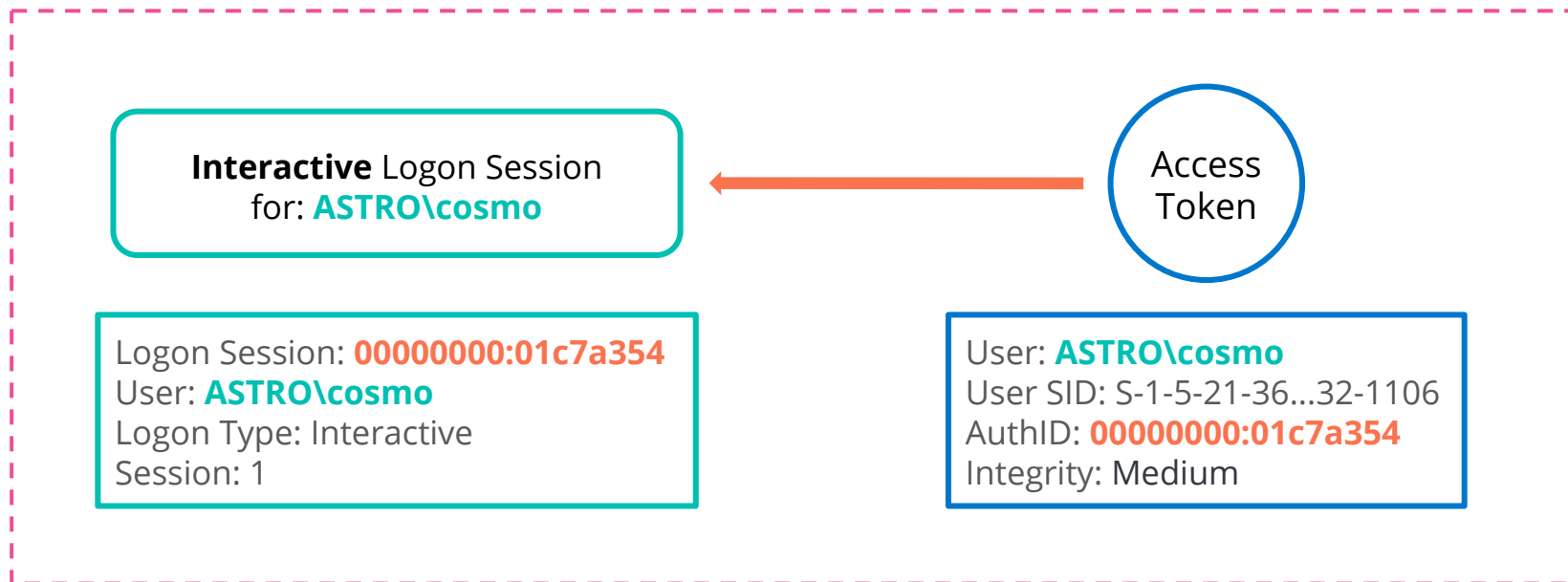
Interactive Logon Session
for: **ASTRO\cosmo**

Logon Session: 00000000:01c7a354
User: **ASTRO\cosmo**
Logon Type: Interactive
Session: 1

Access
Token

User: **ASTRO\cosmo**
User SID: S-1-5-21-36...32-1106
AuthID: 00000000:01c7a354
Integrity: Medium

Logon Sessions and Access Tokens

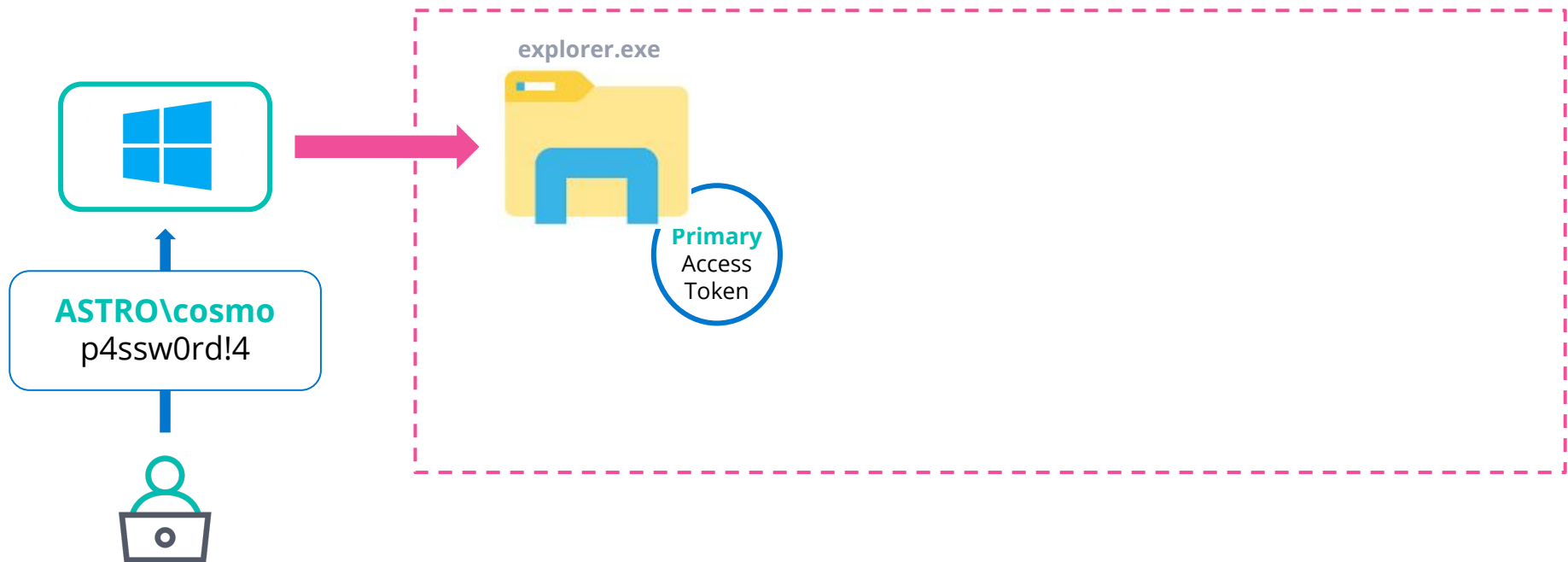


Logon Sessions and Access Tokens



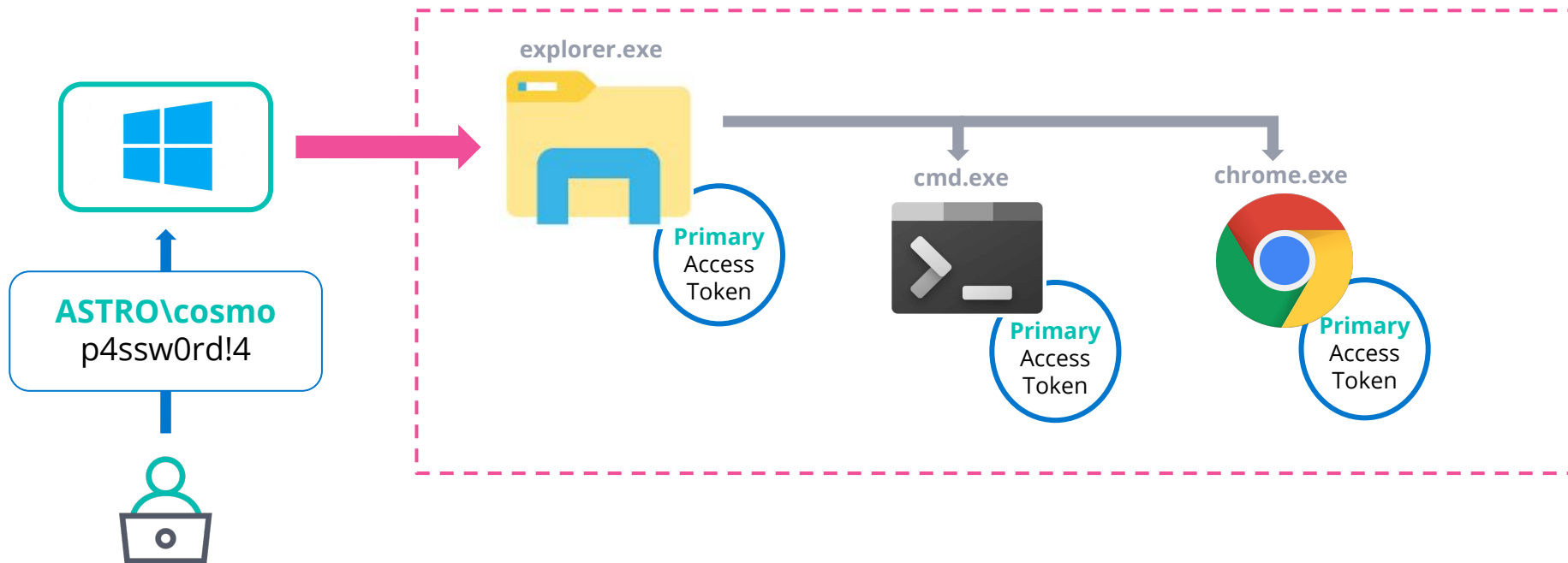
Logon Sessions and Access Tokens

Interactive Session // Winsta0



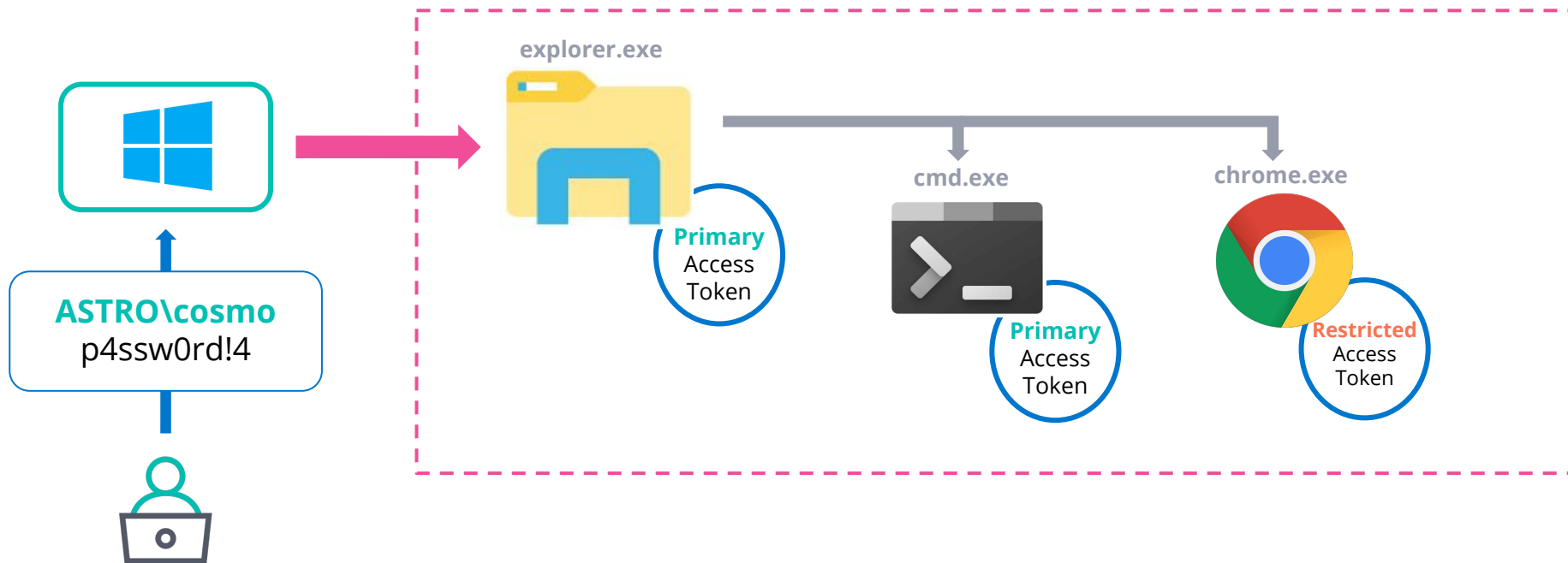
Logon Sessions and Access Tokens

Interactive Session // Winsta0



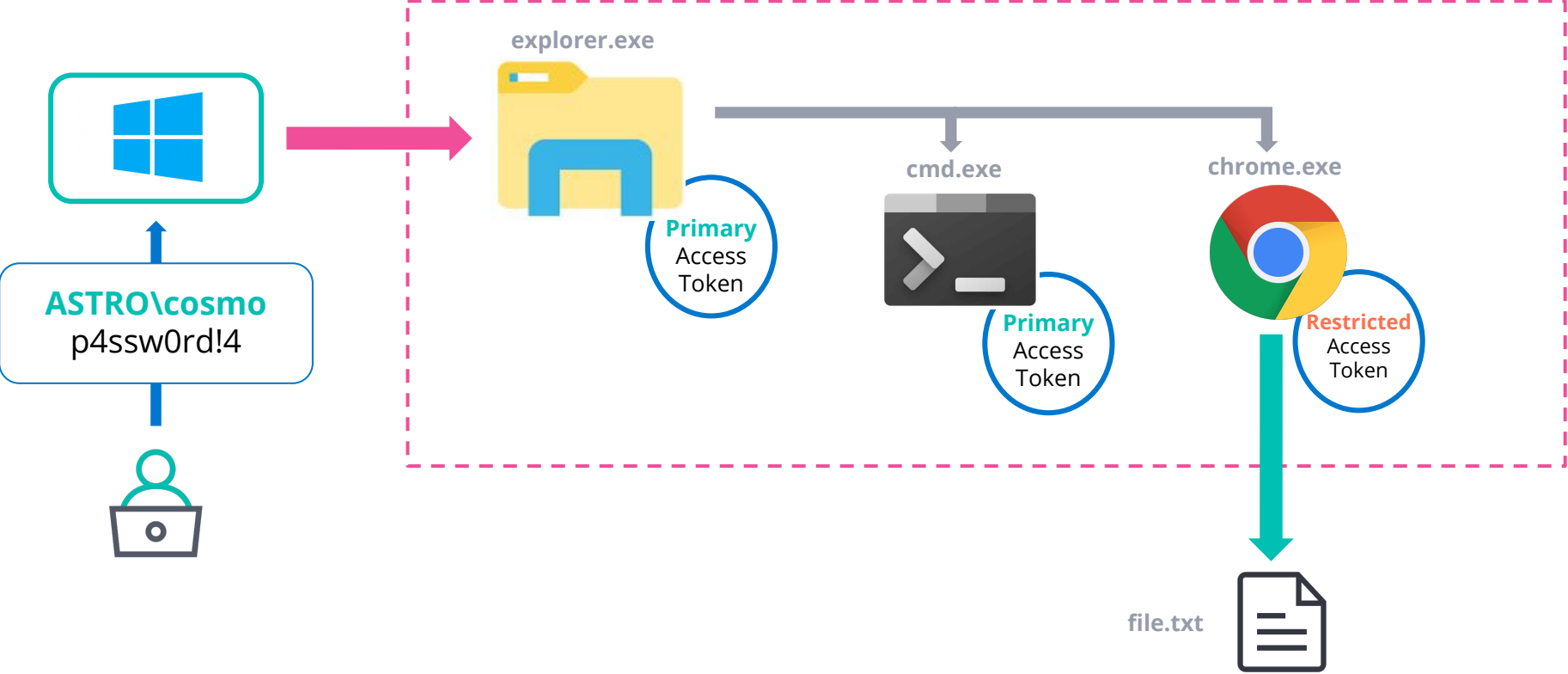
Logon Sessions and Access Tokens

Interactive Session // Winsta0



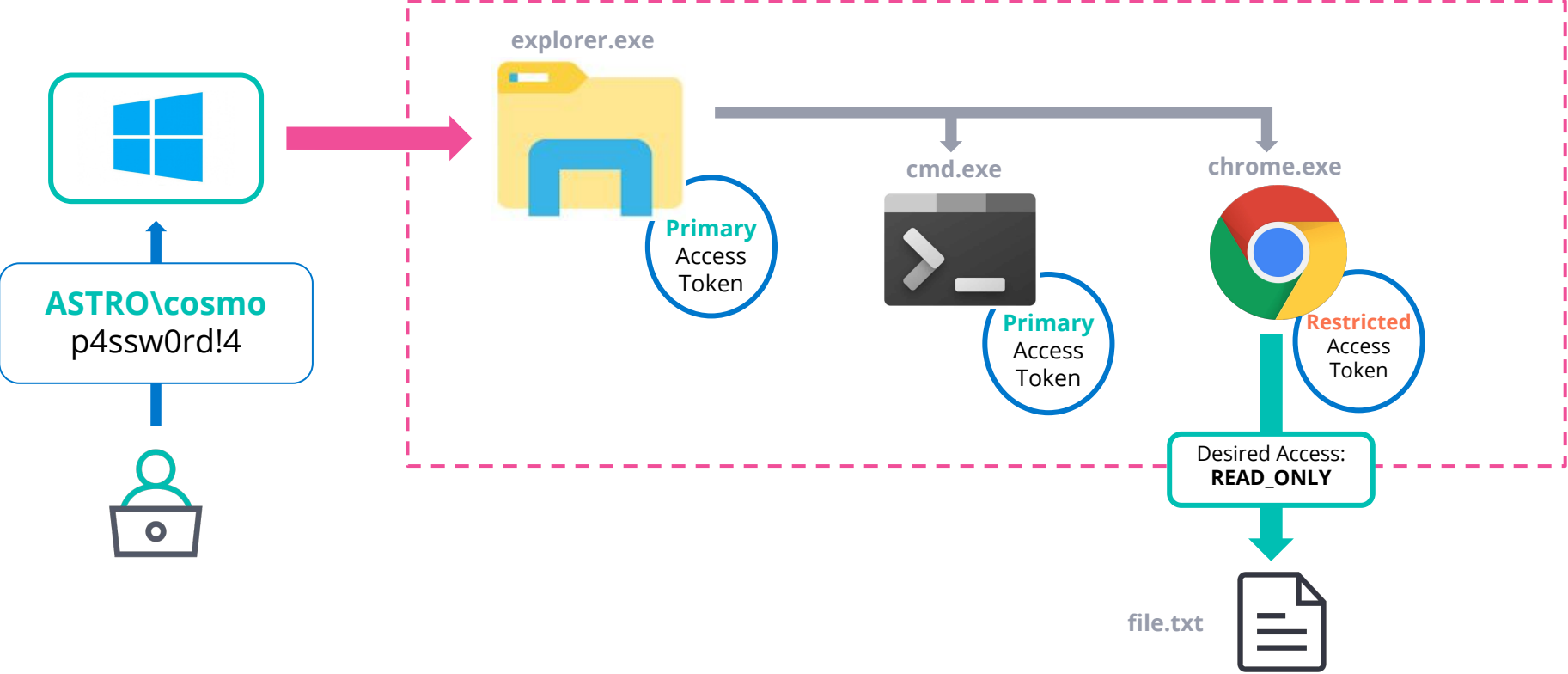
Logon Sessions and Access Tokens

Interactive Session // Winsta0



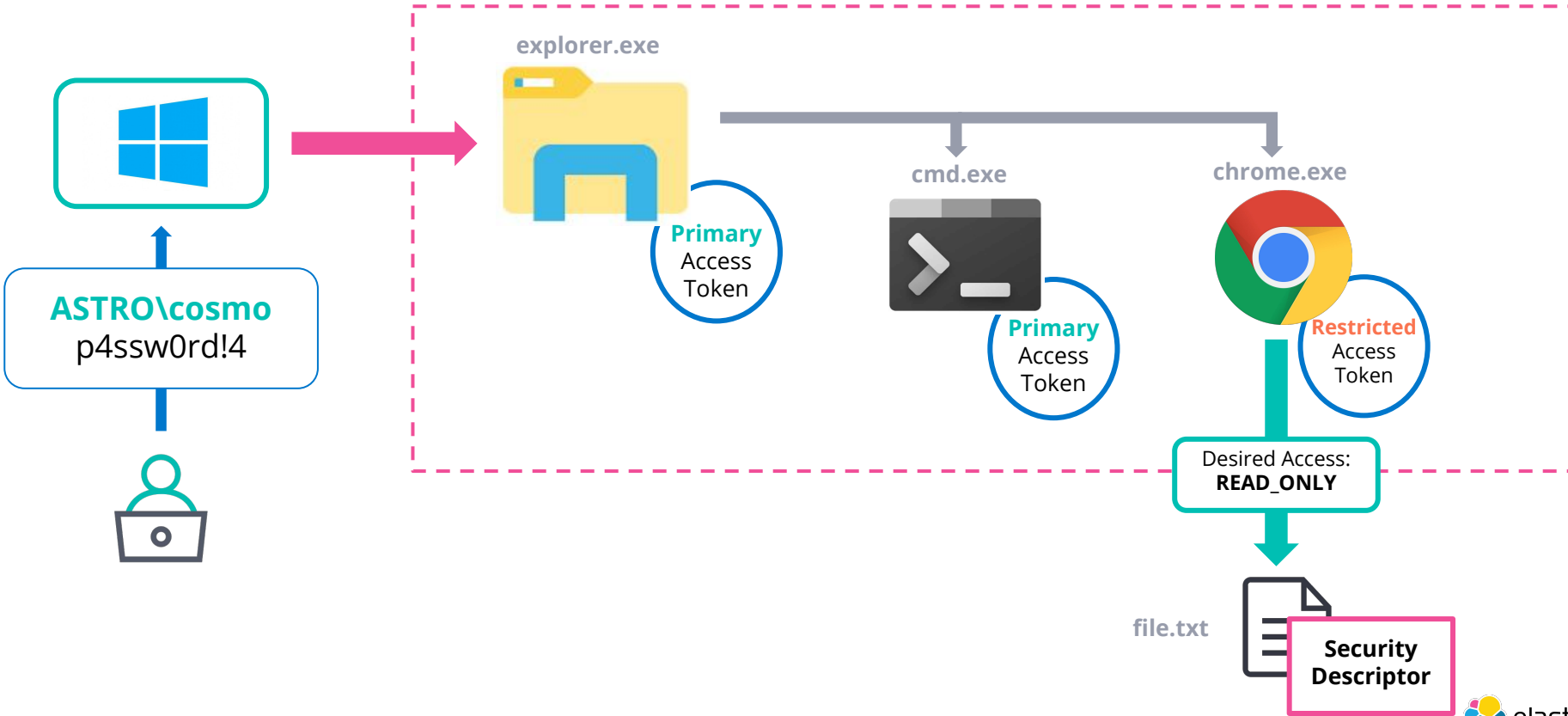
Logon Sessions and Access Tokens

Interactive Session // Winsta0



Logon Sessions and Access Tokens

Interactive Session // Winsta0



Windows Security Internals: Network Authentication

Network Authentication

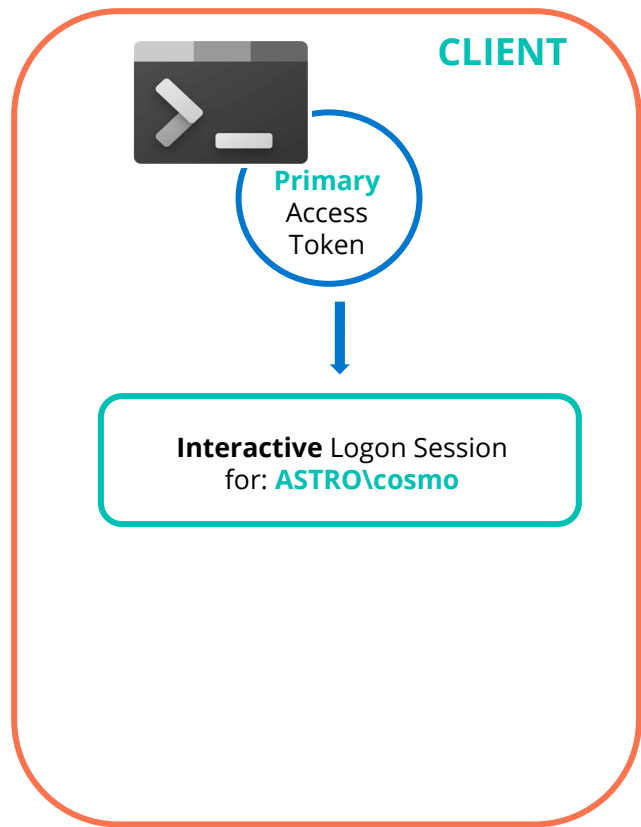
```
Command Prompt
C:\Users\cosmo>net view \\MISTAKENOT-DC-1
Shared resources at \\MISTAKENOT-DC-1

MISTAKENOT-DC-1

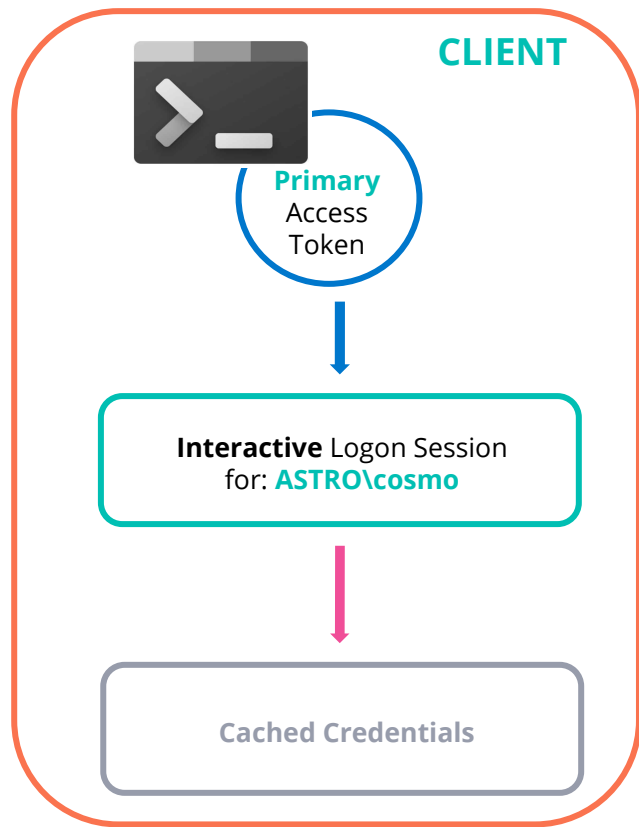
Share name  Type  Used as  Comment
-----
NETLOGON    Disk          Logon server share
SYSVOL      Disk          Logon server share
The command completed successfully.

C:\Users\cosmo>
```

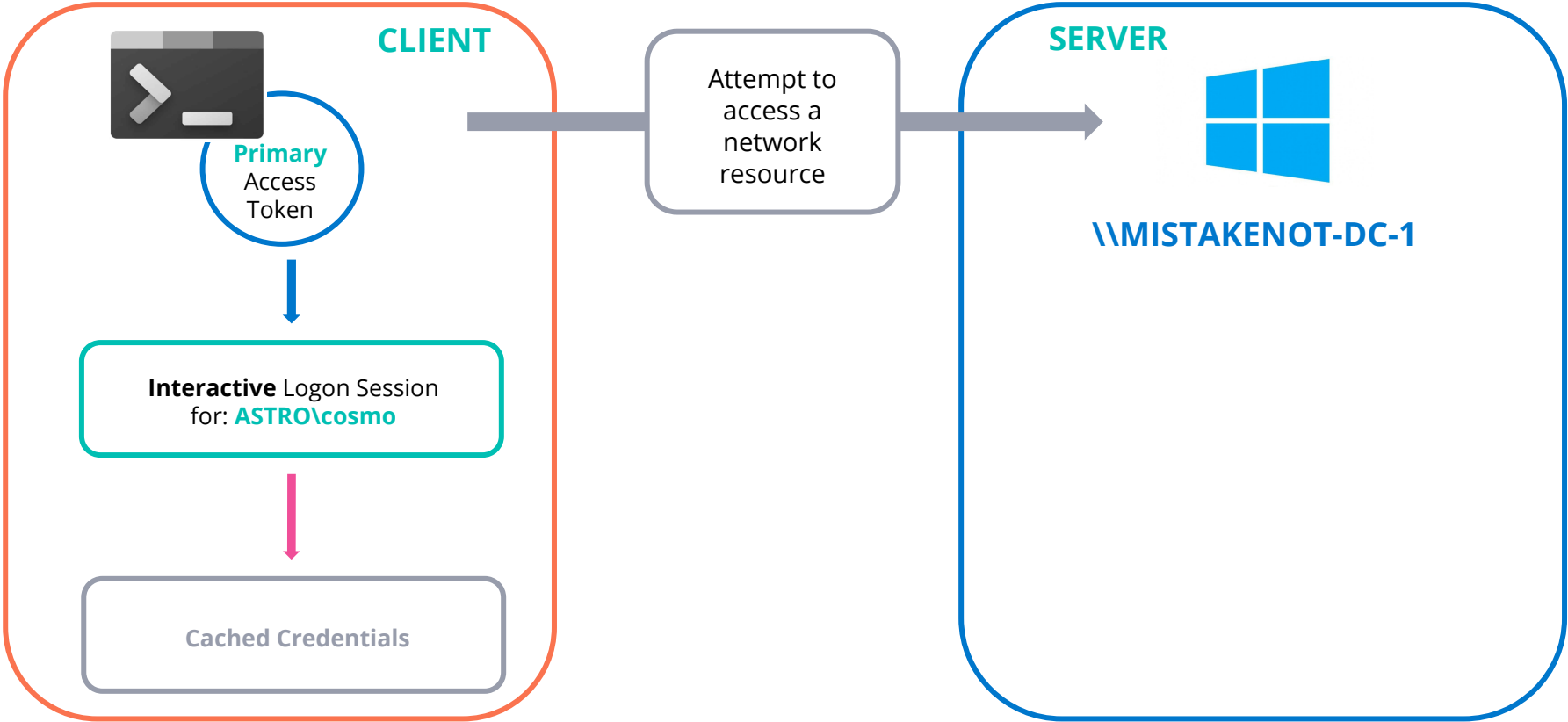

Network Authentication



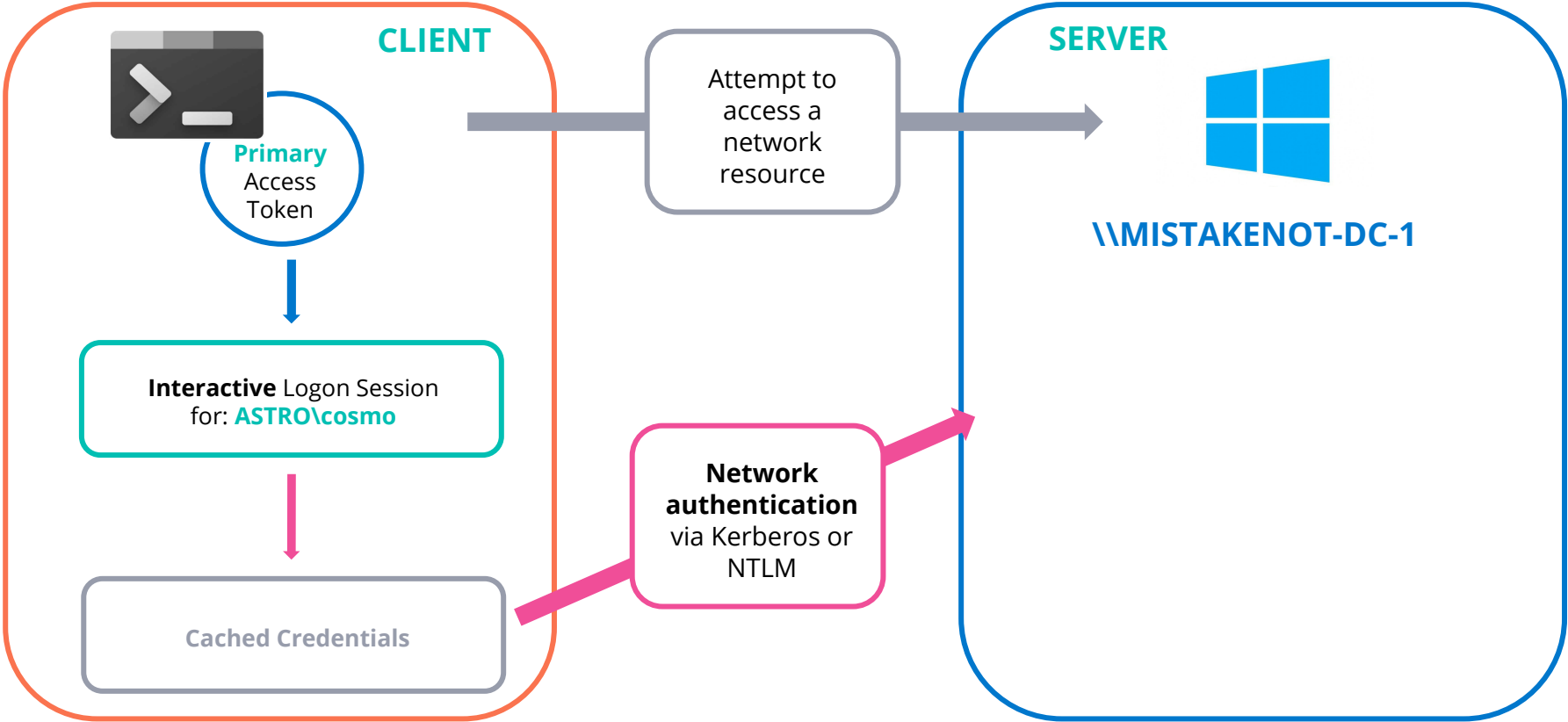
Network Authentication



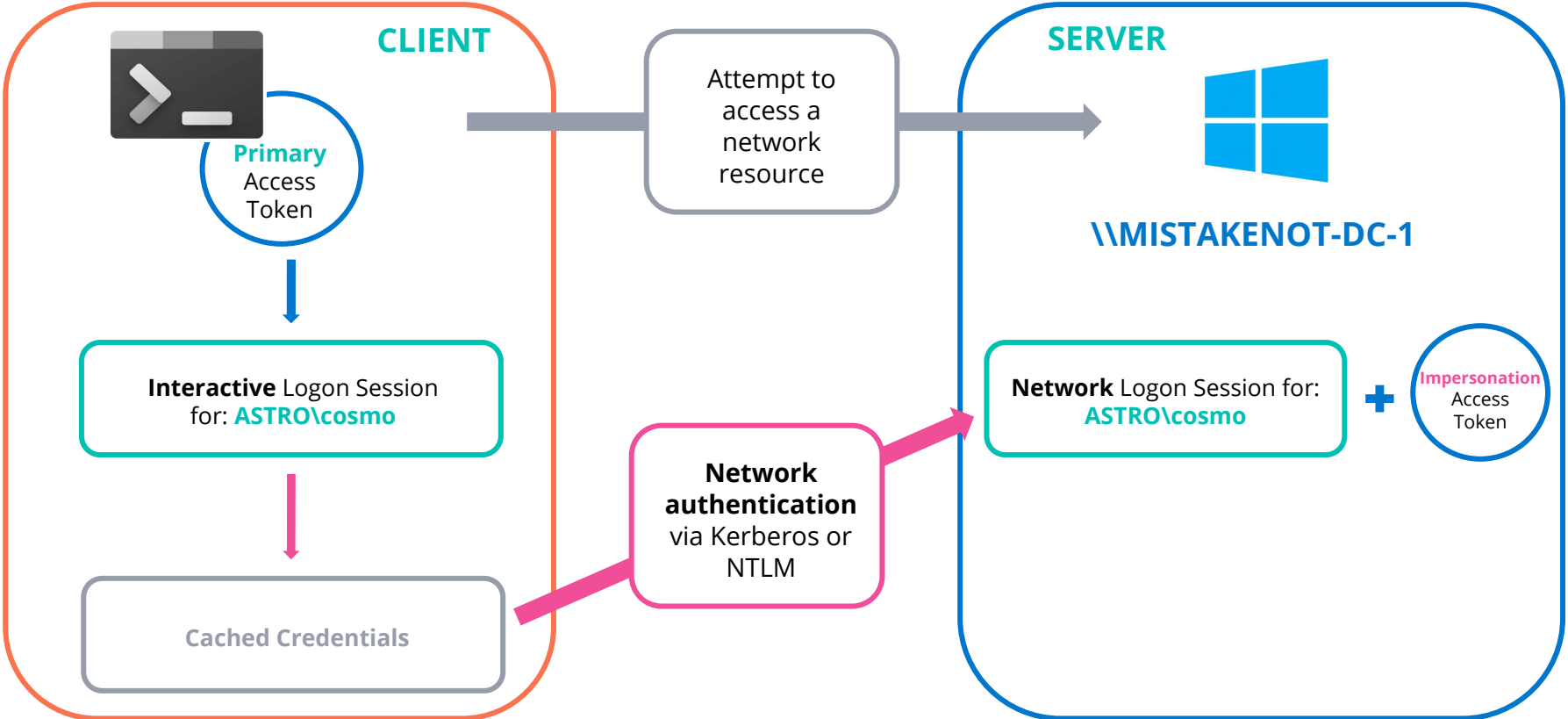
Network Authentication



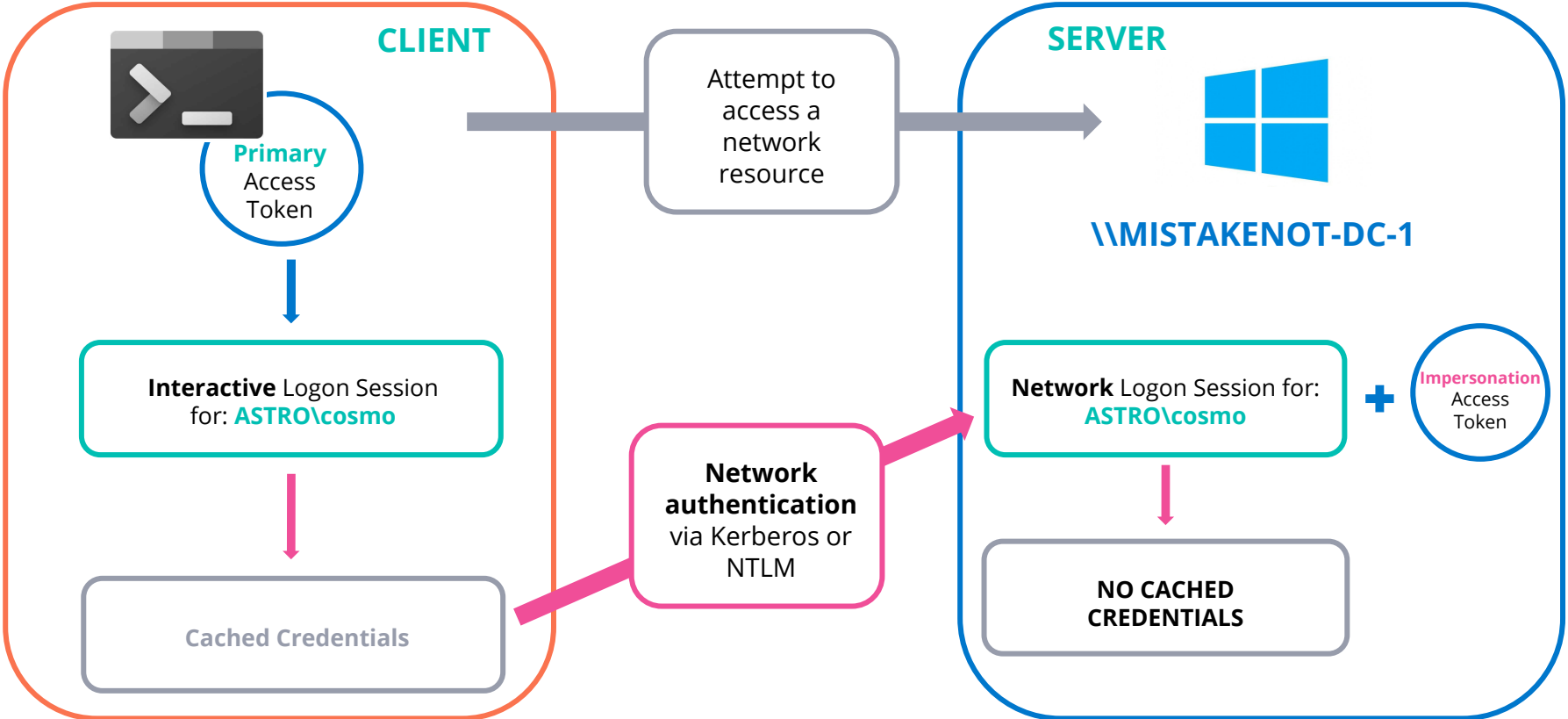
Network Authentication



Network Authentication



Network Authentication



Impersonation

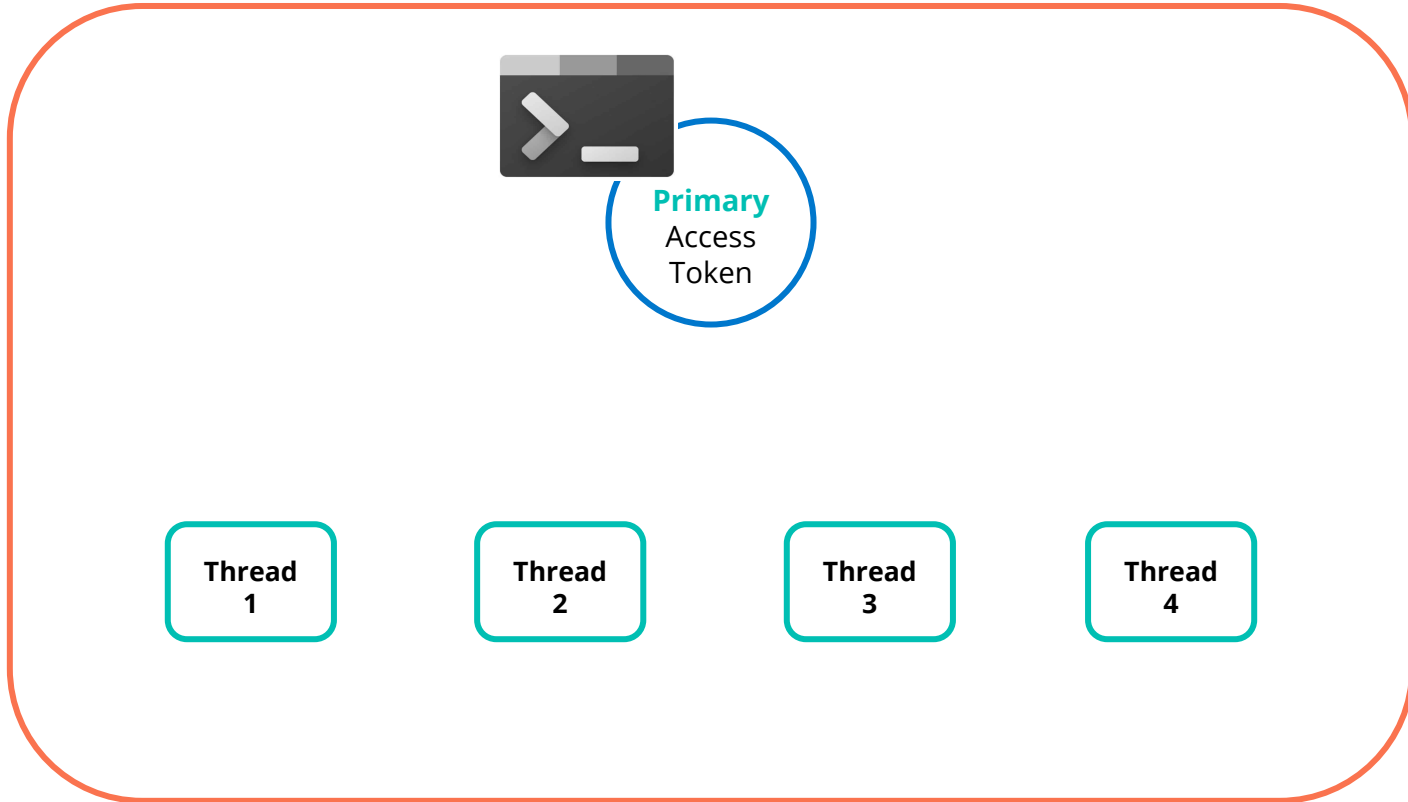


Impersonation

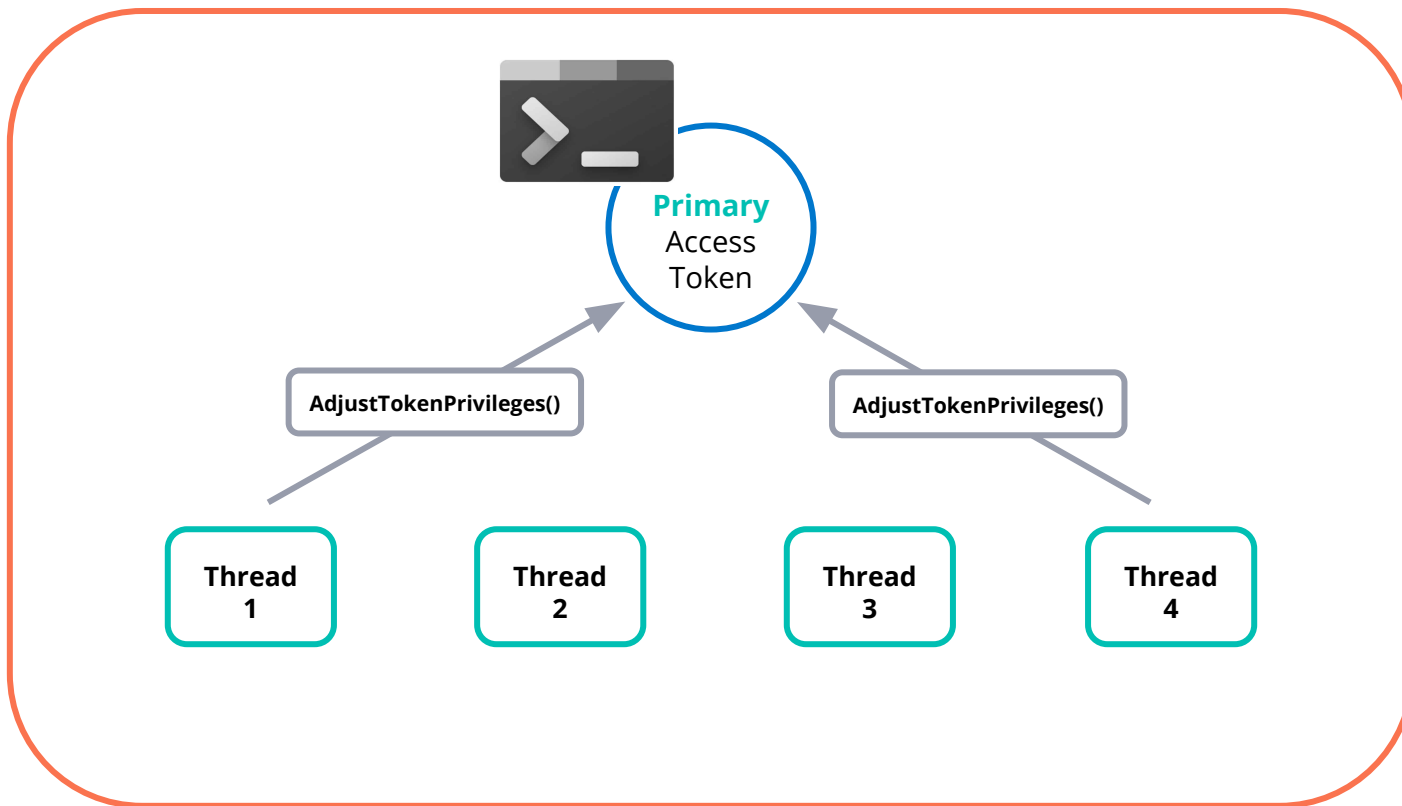
- From the server's perspective, we now have a handle to a token but how do we actually use it?
- Access tokens provide an abstraction for developers to make “**localized changes**” to the security context *without* affecting other processes*
- However, in multi-threaded applications, problems and difficult to debug race conditions may arise if different threads **start enabling or disabling different privileges or modifying DACLS**
- Impersonation allows a thread to **switch to a different security context** (by default threads will inherit the same security context as the primary token of the process)

***Programming Windows Security**, Keith Brown (2000)

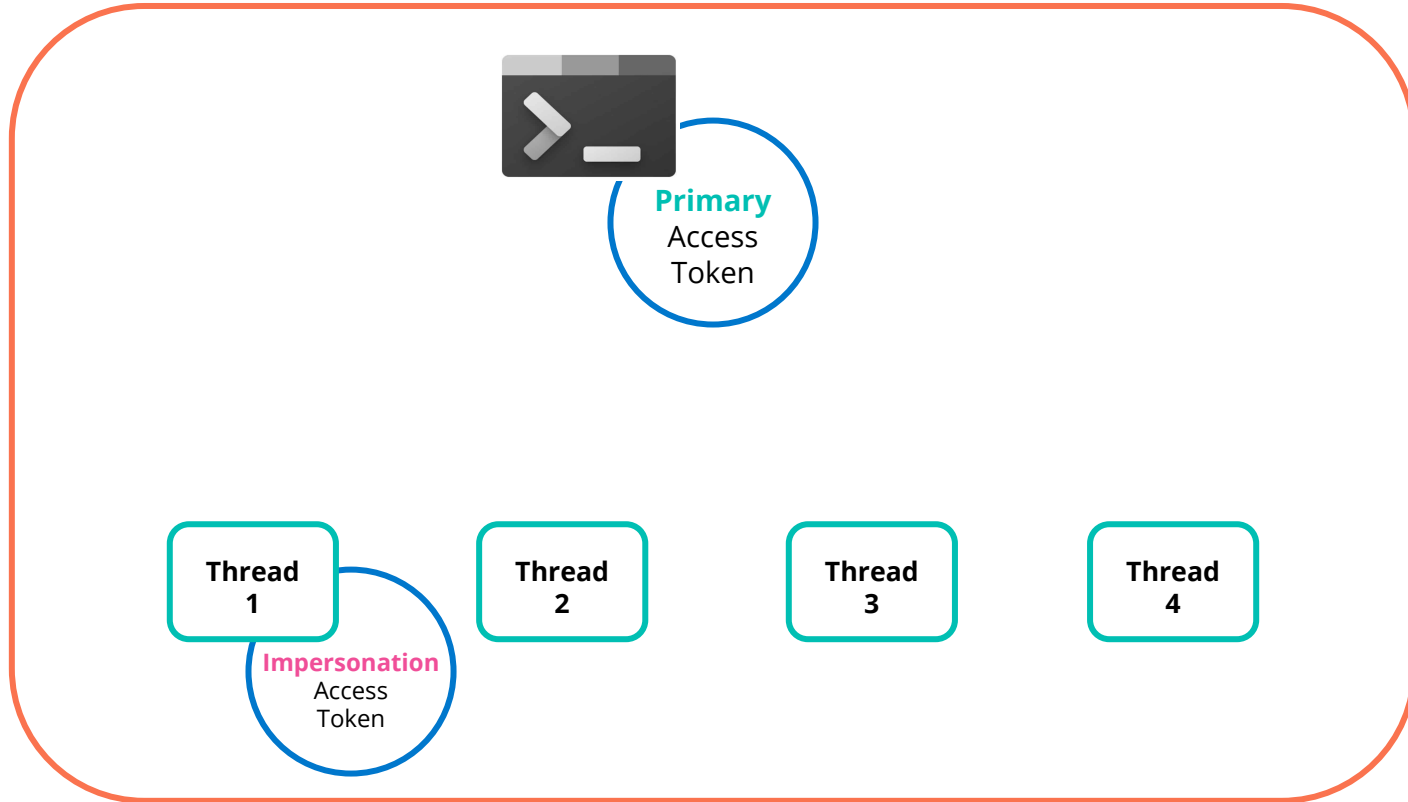
Impersonation



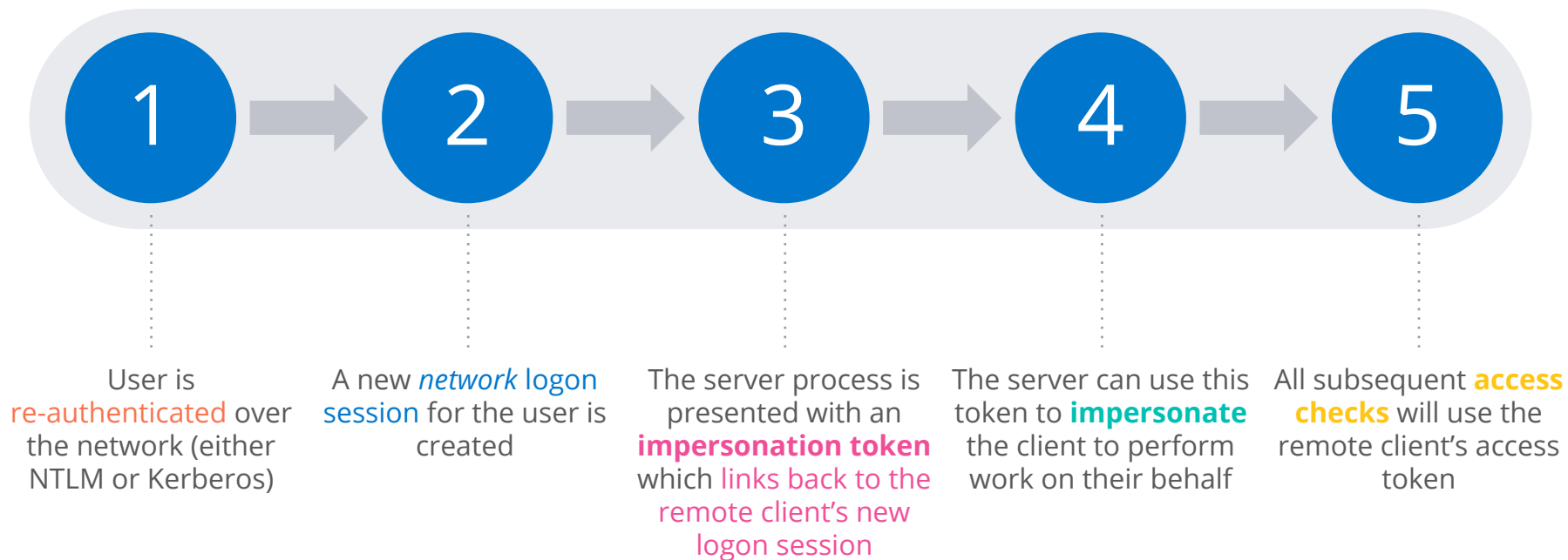
Impersonation



Impersonation



Network Authentication



Network Authentication

- For most of Windows' communication protocols (RPC, DCOM, Named pipes) this process is handled **automatically**
- The server only needs to call the appropriate API **in order to obtain a handle to the remote clients' security context** (e.g. access token) and start impersonating the client via functions such as:

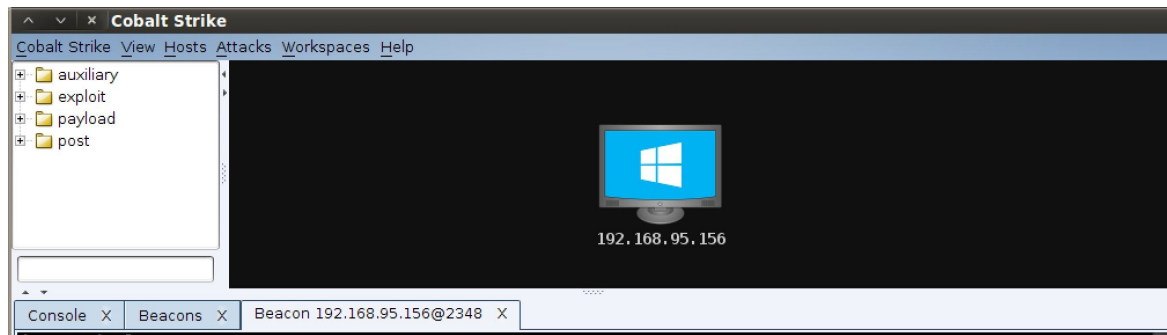
```
ImpersonateNamedPipeClient ()  
RpcImpersonateClient ()  
CoImpersonateClient ()  
DdeImpersonateClient ()
```

How Attackers Abuse Access Tokens

ATT&CK technique: T1134

Initial Compromise

- **Situation:** An attacker has obtained a foothold in a corporate network by phishing a business user
- They have a shell in the context of the compromised user (**who has no privileges across the domain**) and any attempts to authenticate remotely will use the compromised user's credentials (which are no good!)
- The attacker must move laterally but what can they do?



Token Manipulation: The 'Art of the Possible'

1

Steal the token of an already logged-on privileged user (**non-network** logon)

2

Create a new logon session with stolen credentials and impersonate the returned token/spawn a new process with it

3

Change the cached credentials associated with their current access token to **stolen credentials** (e.g. **legitimately** via an API or by **directly modifying** lsass memory)

/NETONLY

The Curious NETONLY Flag...

<https://blog.cobaltstrike.com/2015/12/16/windows-access-tokens-and-alternate-credentials/>

/NETONLY

- The Windows API provides the **LogonUserA/W()** function(s) to create a new logon session for a given user (or principal)
- Both are wrappers around **LogonUserExExW()** in SspiCli.dll

```
BOOL LogonUserA(  
    LPCSTR  lpszUsername,  
    LPCSTR  lpszDomain,  
    LPCSTR  lpszPassword,  
    DWORD   dwLogonType,  
    DWORD   dwLogonProvider,  
    PHANDLE phToken  
);  
  
dwLogonType = LOGON32_LOGON_INTERACTIVE
```

<https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-logonusera>

<https://docs.microsoft.com/en-us/windows/win32/secauthn/logonuserexexw>

/NETONLY

- If you receive a **primary token** and wish to impersonate you must first convert it to an **impersonation token** via **DuplicateTokenEx**:

```
BOOL DuplicateTokenEx(  
    HANDLE                hExistingToken,  
    DWORD                dwDesiredAccess,  
    LPSECURITY_ATTRIBUTES lpTokenAttributes,  
    SECURITY_IMPERSONATION_LEVEL ImpersonationLevel,  
    TOKEN_TYPE            TokenType,  
    PHANDLE              phNewToken  
);
```

```
TokenType = TokenImpersonate
```

/NETONLY

- If you receive an impersonation token you can use either **SetThreadToken** or **ImpersonateLoggedOnUser** to enable the calling thread to impersonate the *security context of the logged-on user*

```
BOOL SetThreadToken(  
    PHANDLE Thread,  
    HANDLE Token  
);  
  
BOOL ImpersonateLoggedOnUser(  
    HANDLE hToken  
);
```

<https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadtoken>
<https://docs.microsoft.com/en-us/windows/win32/api/securitybaseapi/nf-securitybaseapi-impersonateloggedonuser>

/NETONLY

- If you receive an impersonation token you can use either **SetThreadToken** or **ImpersonateLoggedOnUser** to enable the calling thread to impersonate the *security context of the logged-on user*
- Both are wrappers around undocumented **NtSetInformationThread** in Ntdll

```
BOOL SetThreadToken(  
    PHANDLE Thread,  
    HANDLE Token  
);  
  
BOOL ImpersonateLoggedOnUser(  
    HANDLE hToken  
);
```

<http://undocumented.ntinternals.net/UserMode/Undocumented%20Functions/NT%20Objects/Thread/NtSetInformationThread.html>

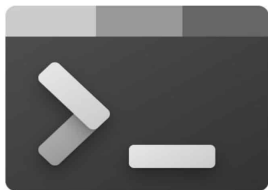
/NETONLY

dwLogonType	Token returned	Cache Credentials?	Is returned token elevated? (if admin)
Interactive (LOGON32_LOGON_INTERACTIVE)	Primary	Yes	No (UAC applies)
Interactive (RID-500/ Local Admin)	Primary	Yes	Yes
Network (LOGON32_LOGON_NETWORK)	Impersonation	No	Yes (+ all privs enabled)
Network (RID-500/ Local Admin)	Impersonation	No	Depends on remote UAC settings*

*See for more info:

<https://blueteamer.blogspot.com/2018/12/disabling-uac-remote-restrictions-to.html>
<https://support.microsoft.com/en-gb/help/951016/description-of-user-account-control-and-remote-restrictions-in-windows>

/NETONLY



```
BOOL LogonUserA(  
    Administrator,  
    ASTRO,  
    x2yts68u!,  
    LOGON32_LOGON_INTERACTIVE,  
    ...  
);
```

/NETONLY

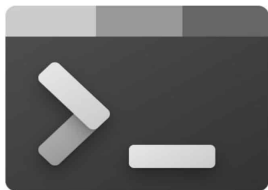


Access is denied.

```
BOOL LogonUserA(  
    Administrator,  
    ASTRO,  
    x2yts68u!,  
    LOGON32_LOGON_INTERACTIVE,  
    ...  
);
```



/NETONLY



```
BOOL LogonUserA(  
    Administrator,  
    ASTRO,  
    x2yts68u!,  
    LOGON32_LOGON_NEW_CREDENTIALS,  
    ...  
);
```

/NETONLY

CLIENT

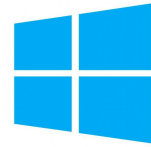


Interactive Logon Session
for: **ASTRO\cosmo**



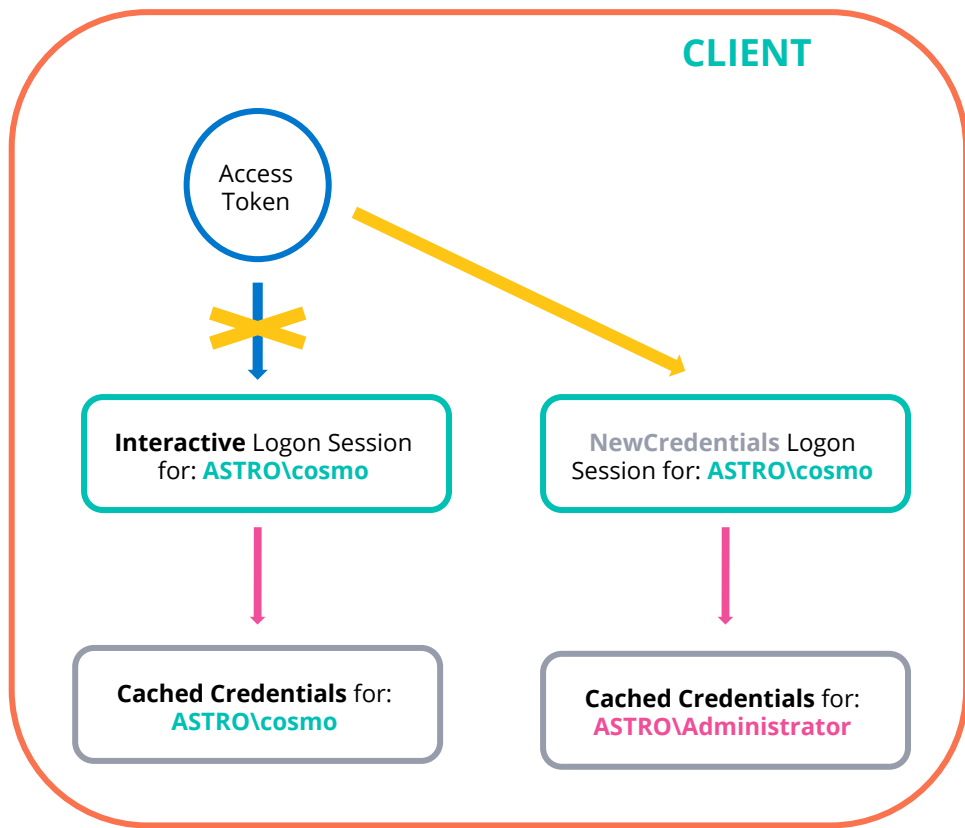
Cached Credentials for:
ASTRO\cosmo

SERVER

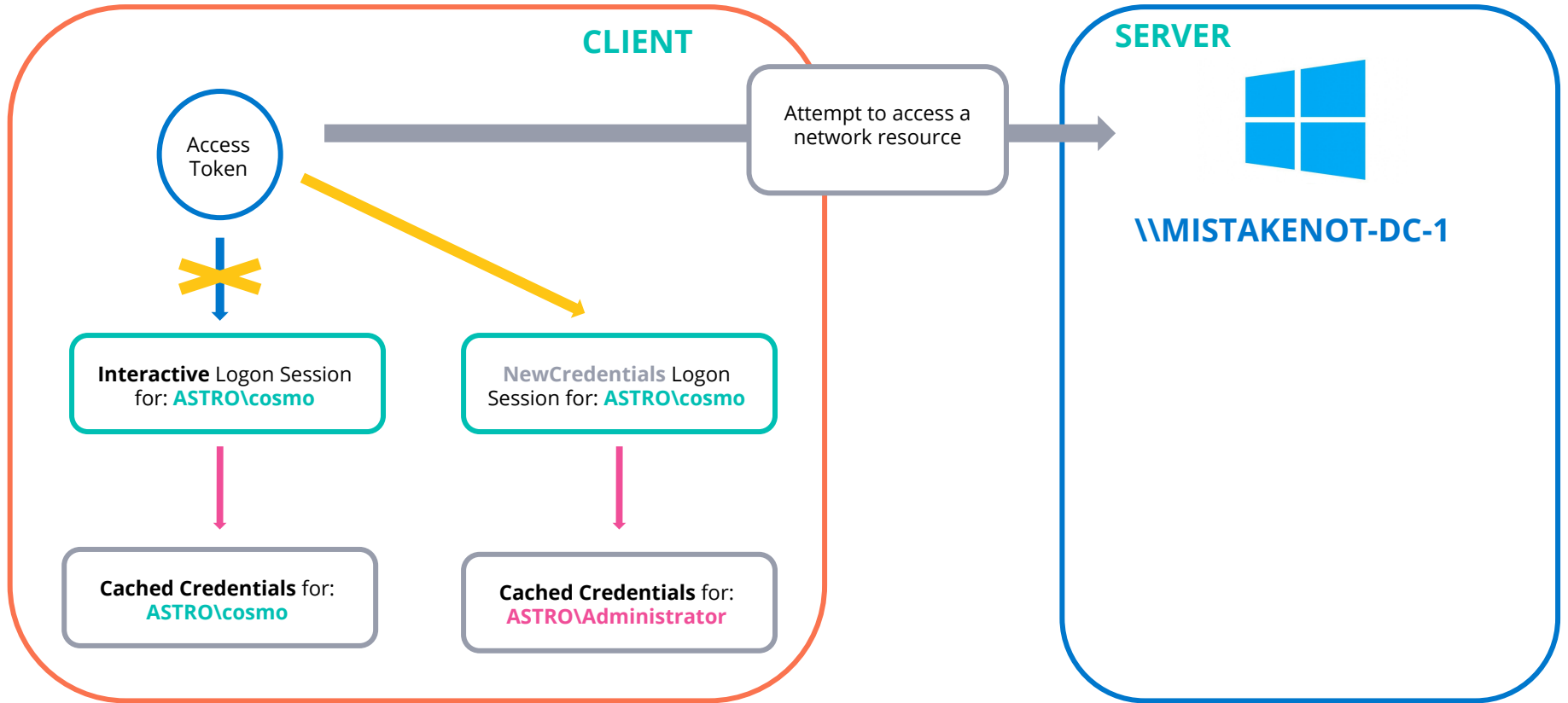


\\MISTAKENOT-DC-1

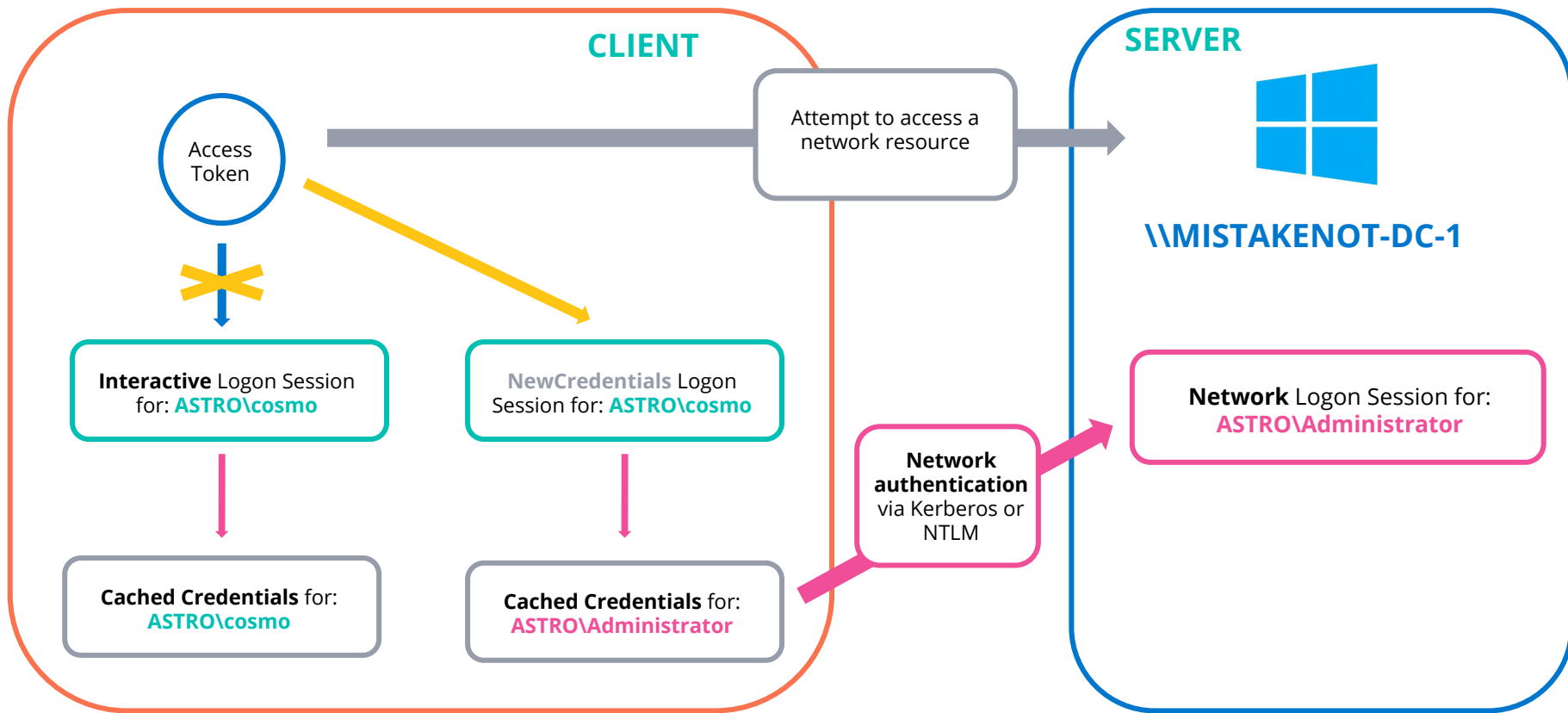
/NETONLY



/NETONLY



/NETONLY



CreateProcessWithLogonW

- Can replicate the same behaviour with **CreateProcessWithLogonW!**

```
BOOL CreateProcessWithLogonW (  
    LPCWSTR          lpUsername,  
    LPCWSTR          lpDomain,  
    LPCWSTR          lpPassword,  
    DWORD            dwLogonFlags ,  
    LPCWSTR          lpApplicationName,  
    LPWSTR           lpCommandLine,  
    DWORD            dwCreationFlags,  
    LPVOID           lpEnvironment,  
    LPCWSTR          lpCurrentDirectory,  
    LPSTARTUPINFO    lpStartupInfo,  
    LPPROCESS_INFORMATION lpProcessInformation  
);
```

```
dwLogonFlags = LOGON_NETCREDENTIALS_ONLY
```

/NETONLY DEMO

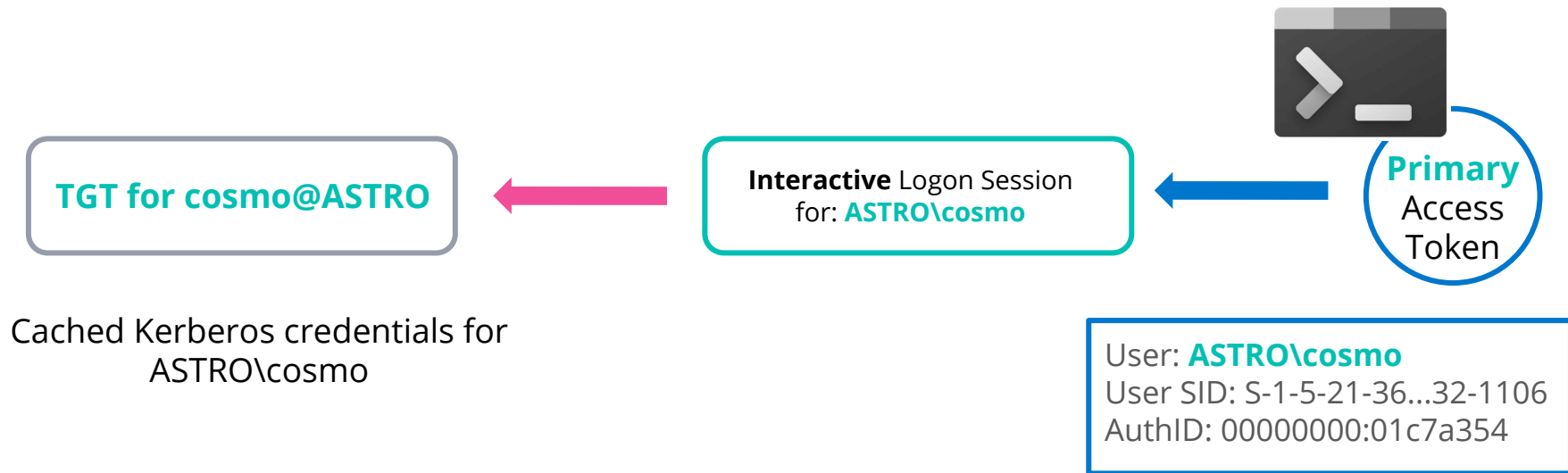
The Curious NETONLY Flag...

<https://blog.cobaltstrike.com/2015/12/16/windows-access-tokens-and-alternate-credentials/>

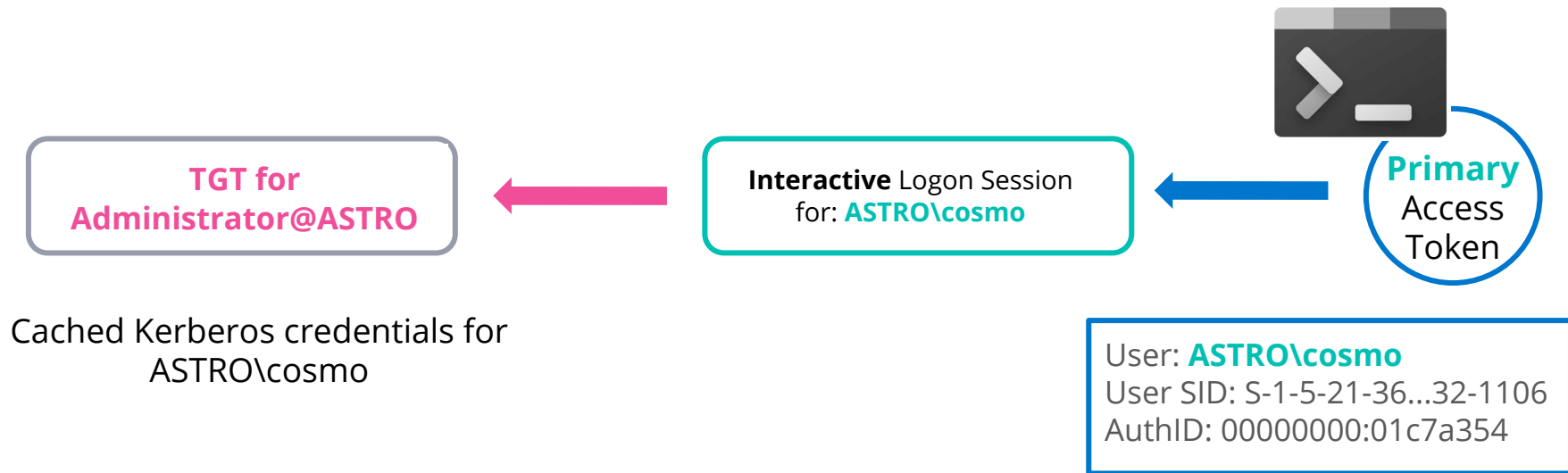
Pass-The-Ticket

Taming the Three-Headed Dog...

Pass-The-Ticket



Pass-The-Ticket



Pass-The-Ticket

```
NTSTATUS LsaConnectUntrusted(  
    PHANDLE LsaHandle  
);
```

```
NTSTATUS LsaLookupAuthenticationPackage(  
    HANDLE LsaHandle,  
    PLSA_STRING PackageName,  
    PULONG AuthenticationPackage  
);
```

```
PackageName = MICROSOFT_KERBEROS_NAME_A
```

Pass-The-Ticket

LsaCallAuthenticationPackage()

```
NTSTATUS LsaCallAuthenticationPackage(  
    HANDLE LsaHandle,  
    ULONG AuthenticationPackage,  
    PVOID ProtocolSubmitBuffer,  
    ULONG SubmitBufferLength,  
    PVOID *ProtocolReturnBuffer,  
    PULONG ReturnBufferLength,  
    PNTSTATUS ProtocolStatus  
);
```

<https://docs.microsoft.com/en-us/windows/win32/api/ntsecapi/nf-ntsecapi-lsacallauthenticationpackage>

<https://github.com/GhostPack/Rubeus/blob/732303e2f182d25e81ae25b4351782b2d0f061d0/Rubeus/lib/LSA.cs#L855>

https://github.com/gentilkiwi/mimikatz/blob/b008188f9fe5668b5dae80c210290c7efa872ffa/mimikatz/modules/kerberos/kuhl_m_kerberos.c#L54

Pass-The-Ticket

KerbSubmitTicketMessage

```
typedef struct _KERB_SUBMIT_TKT_REQUEST {
    KERB_PROTOCOL_MESSAGE_TYPE MessageType;
    LUID LogonId;
    ULONG Flags;
    KERB_CRYPTO_KEY32 Key;
    ULONG KerbCredSize;
    ULONG KerbCredOffset;
} KERB_SUBMIT_TKT_REQUEST, *PKERB_SUBMIT_TKT_REQUEST
```

Pass-The-Ticket DEMO

Taming the Three-Headed Dog...

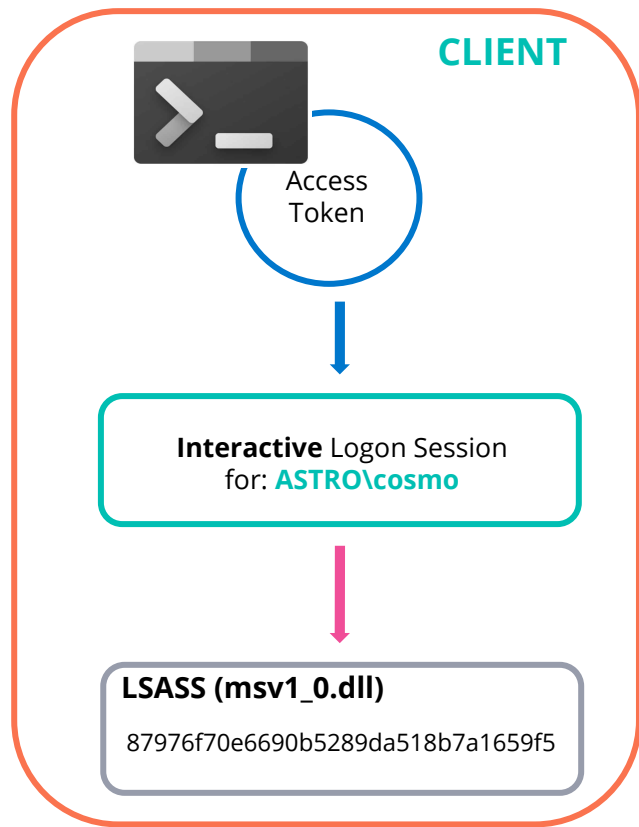
Pass-The-Ticket

- You **do not need privileges** to change the TGT associated with your logon session
- You **do not need to create additional logon sessions** (unless you want to preserve your current TGT!)
- Also note that through **LsaCallAuthenticationPackage()** an attacker (in a high-integrity context*) can also enumerate/dump credentials (e.g. tickets) of other users ***without opening a handle to lsass***
- Hence Kerberos ticket attacks represents a gap in any credential theft logic predicated on a kernel/user-mode object callback (i.e. opening a handle to lsass or reading lsass memory - e.g. **Sysmon process access**)

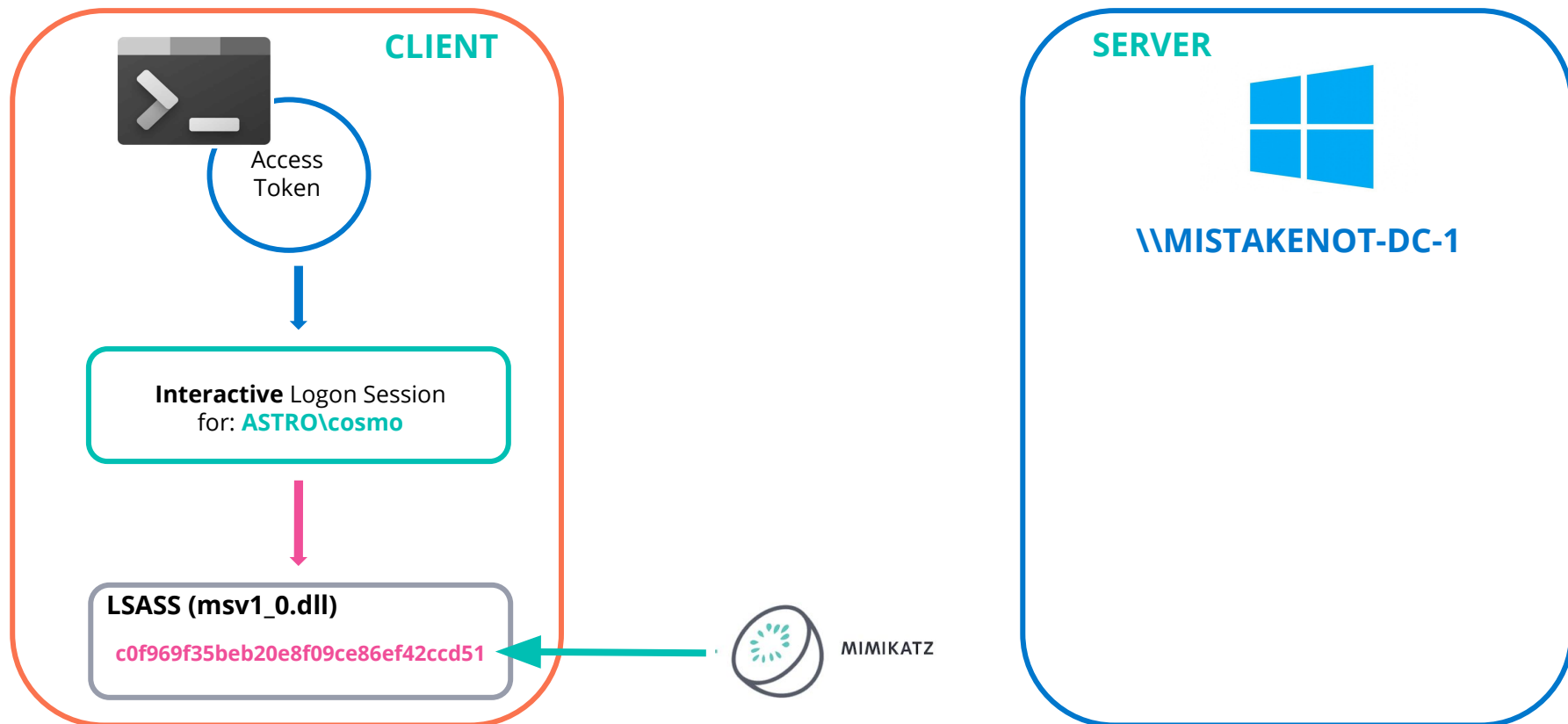
* <https://www.harmj0y.net/blog/redteaming/rubeus-now-with-more-kekeo/>

Overpass-The-Hash

Pass-the-hash



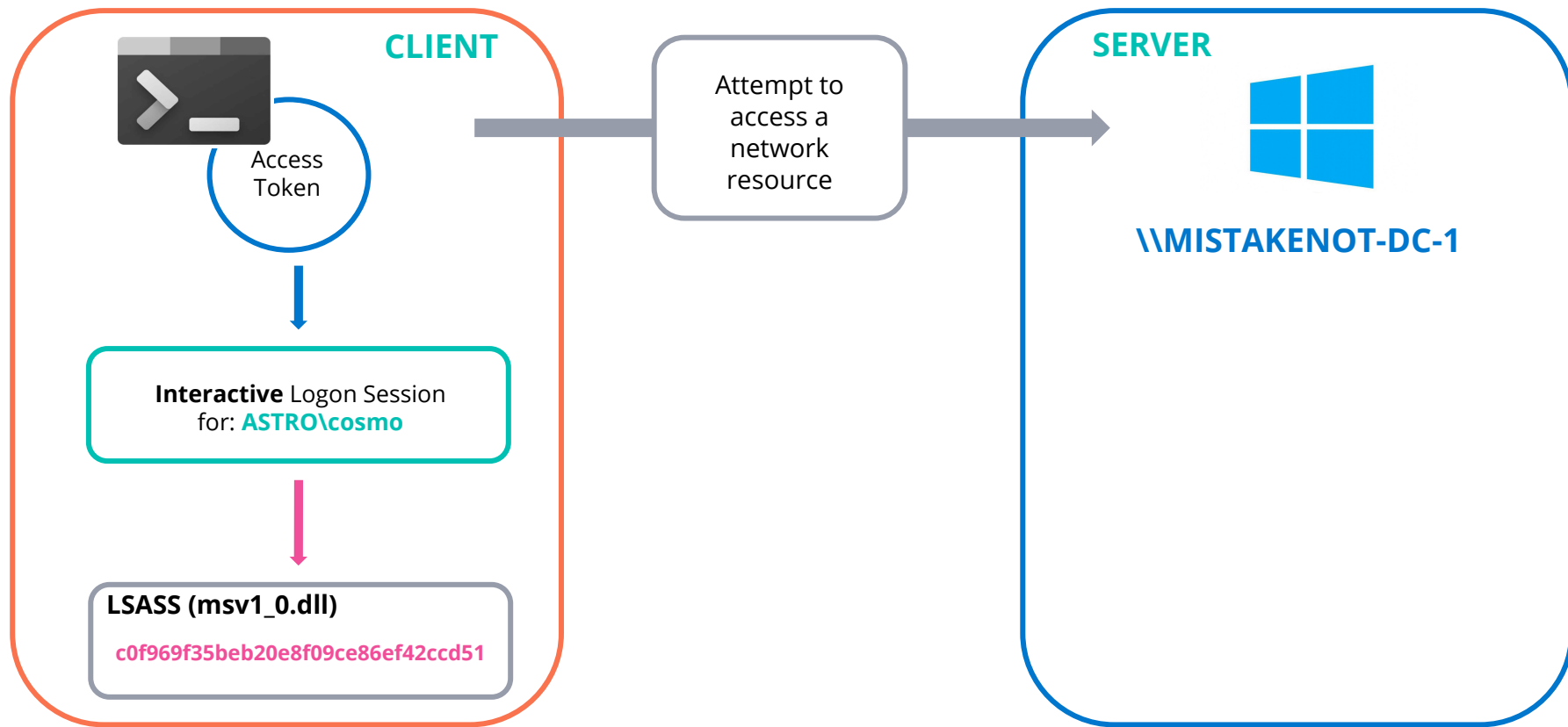
Pass-the-hash



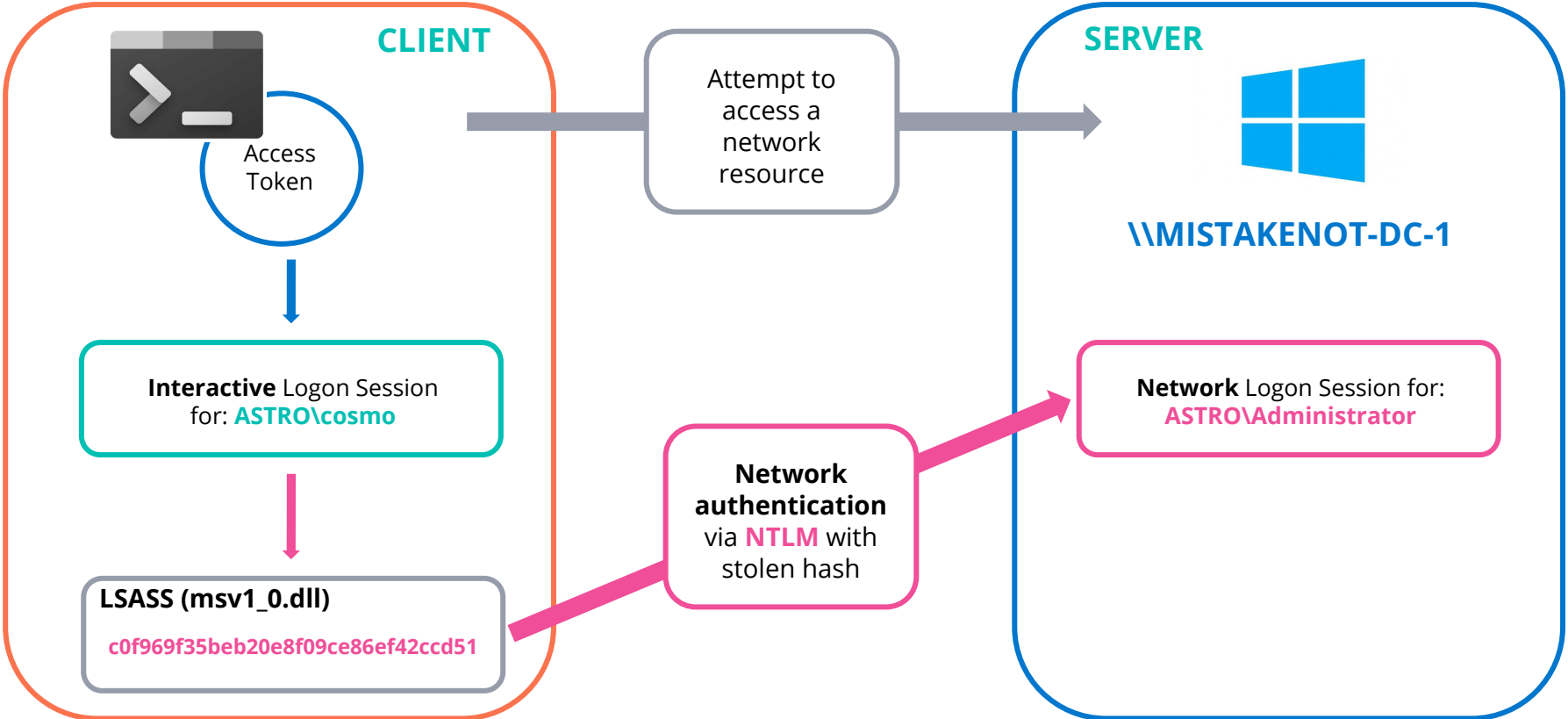
<https://www.slideshare.net/rootedcon/herman-ochoa-wce-internals-rootedcon-2011>

<https://blog.xpsec.com/exploring-mimikatz-part-1/>

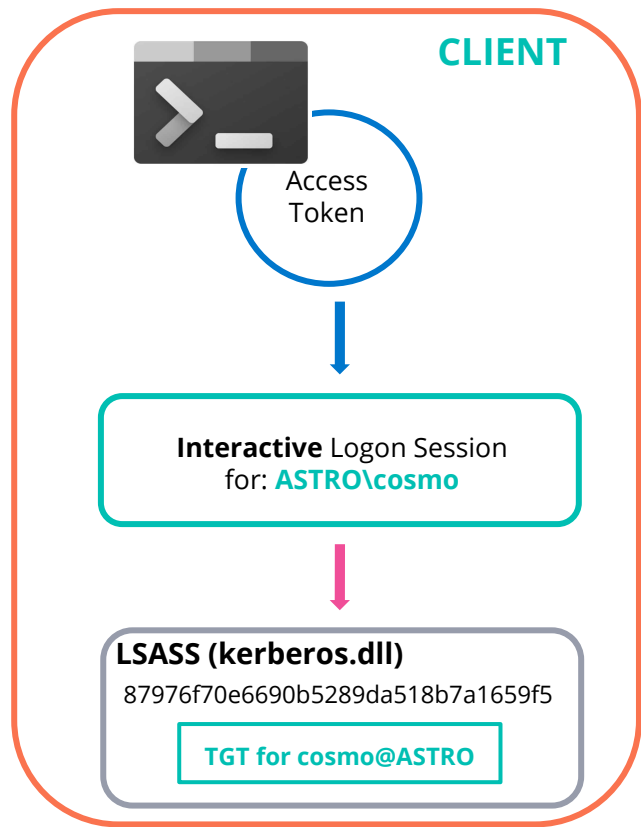
Pass-the-hash



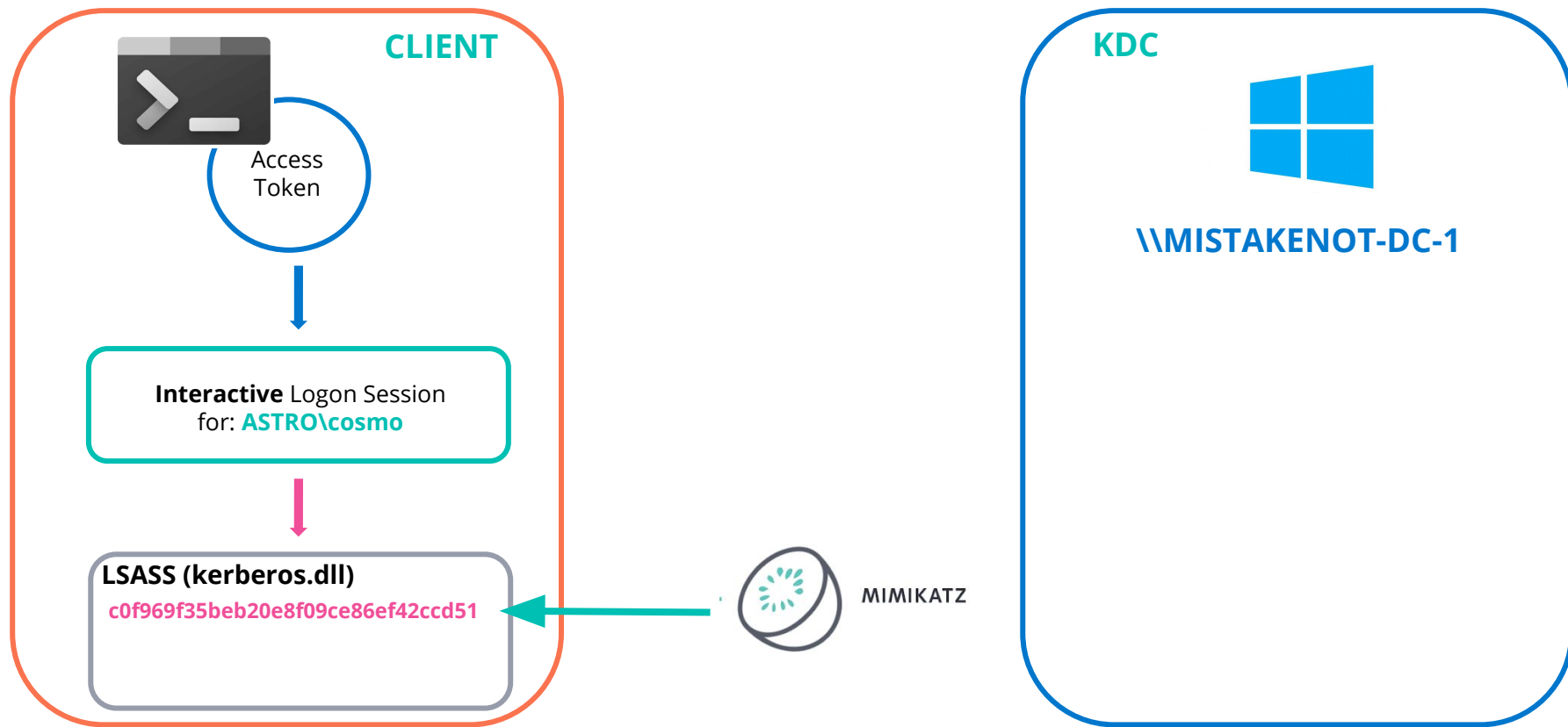
Pass-the-hash



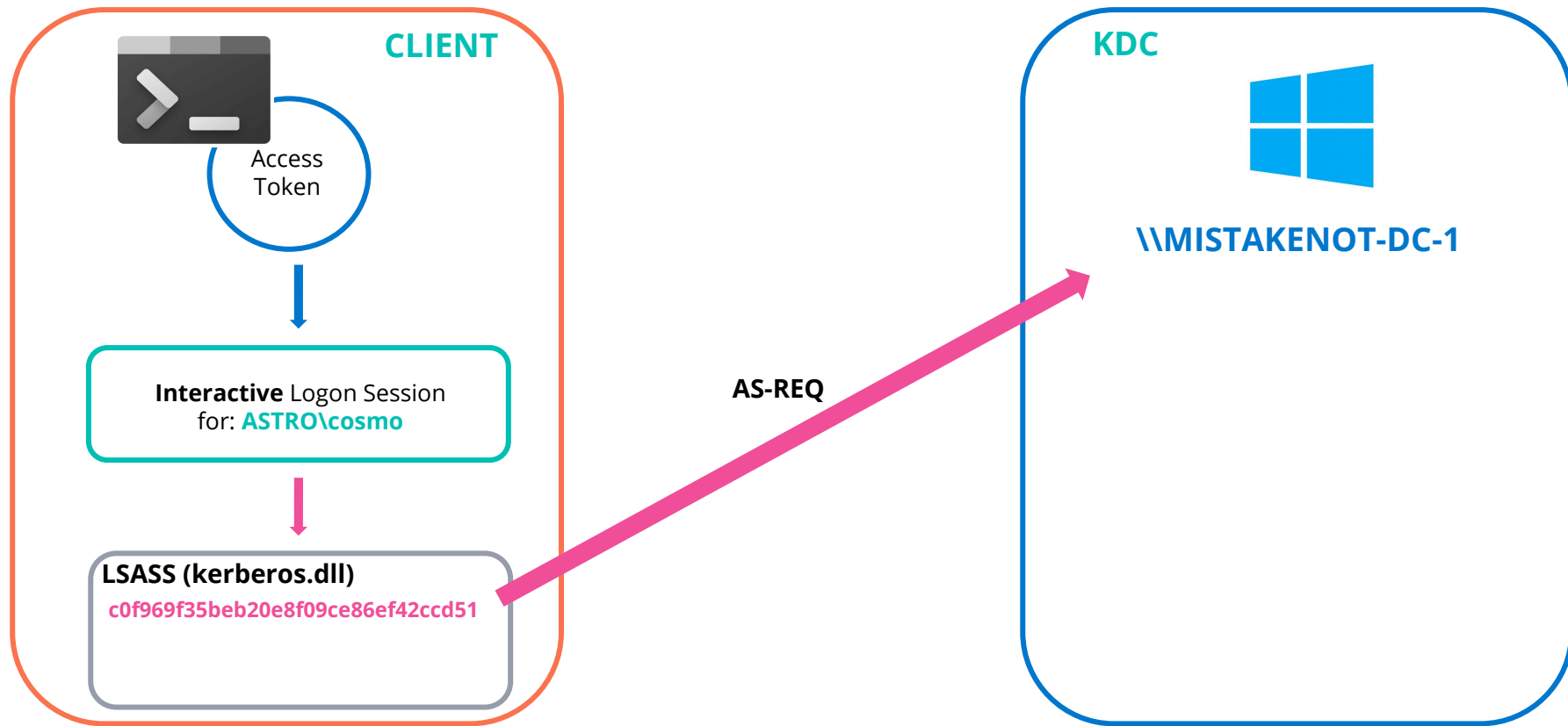
Overpass-the-hash



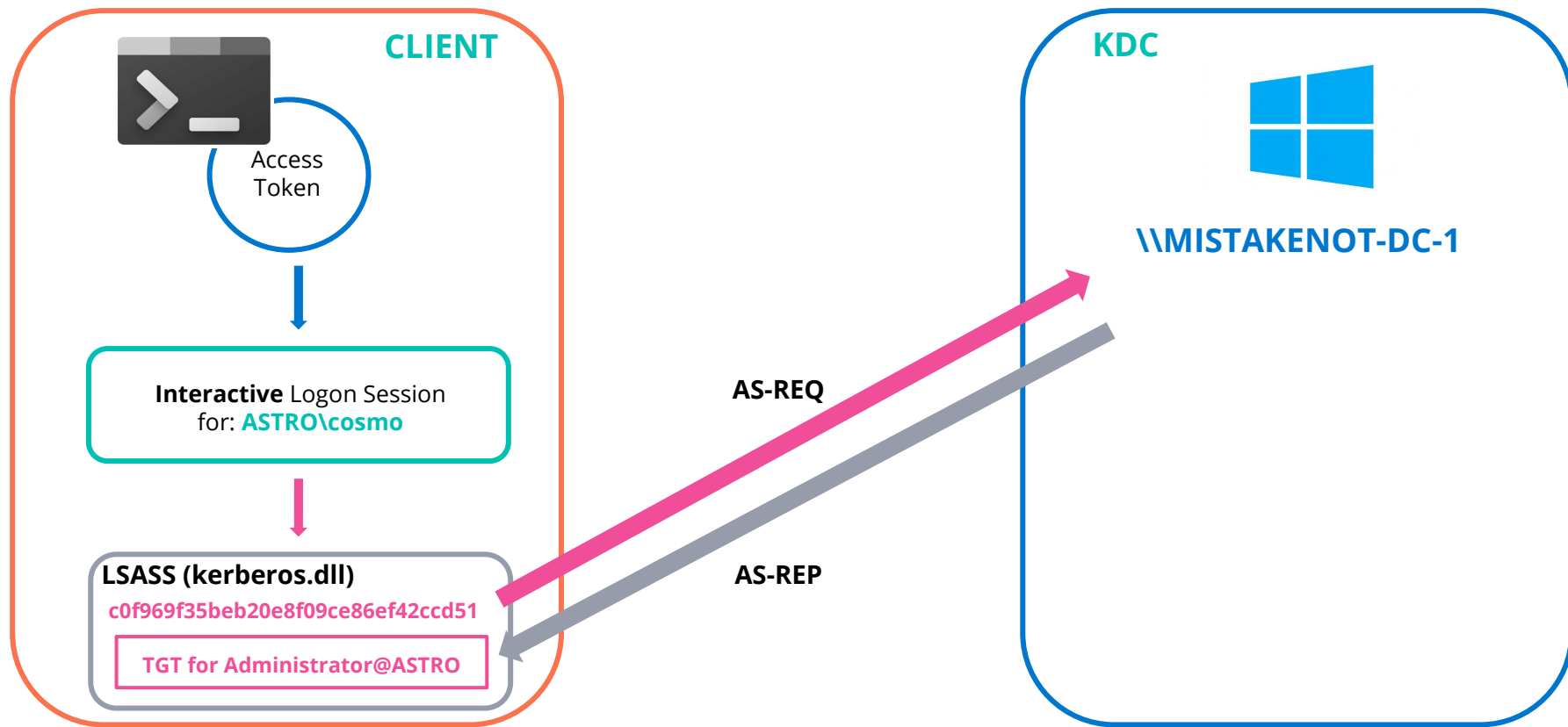
Overpass-the-hash



Overpass-the-hash



Overpass-the-hash



Overpass-the-hash

Mimikatz workflow



Create a new *sacrificial* **NETONLY process** with junk credentials (which *preserves* current TGT and creates a **new logon session**)

Acquire **debug privilege** or **impersonate a SYSTEM token**

Open up a **write handle** to lsass

Patch in **new NTLM hash** associated with **new sacrificial logon session**

The normal Kerberos authentication process starts and returns a **fully fledged TGT**

Detecting Access Token Manipulation

Frida

- Dynamic instrumentation toolkit
- Allows us to hook functions and write **custom and scriptable detection logic** on the fly
- Can capture arguments pre and post function call
- Write detection logic based on **parameters to functions or returned values/output buffers!**

The word 'FRIDA' is written in a large, bold, red, sans-serif font. The letters are slightly spaced out and have a consistent thickness.

<https://frida.re/>

Frida

Basic JS hooking template:

```
// Obtain a Frida NativePointer to function of interest
var functionX = Module.findExportByName("ntdll", "functionX")

// Start hooking function of interest
Interceptor.attach(functionX, {
  onEnter: function (args, state) {
    console.log("[+] Target process called functionX");

    // Read args as required and implement detection logic
  },

  onLeave: function (retval) {
    // Access output buffers or return values as required
  }
});
```

<https://github.com/sensepost/frida-windows-playground>

<https://www.fuzzysecurity.com/tutorials/29.html>

Detecting Access Token Manipulation

Technique

Signals

Other Telemetry Sources

1. NETONLY

- `CreateProcessWithLogonW`
- `LogonUser + Impersonation`

/NETONLY Frida DEMO

The Curious NETONLY Flag...

<https://blog.cobaltstrike.com/2015/12/16/windows-access-tokens-and-alternate-credentials/>

Detecting Access Token Manipulation

Technique	Signals	Other Telemetry Sources
1. NETONLY	<ul style="list-style-type: none">• CreateProcessWithLogonW• LogonUser + Impersonation	<ol style="list-style-type: none">1. Windows Event Logs2. Process events

Detecting Access Token Manipulation

The screenshot displays the Windows Security Event Viewer interface. At the top, the 'Security' window shows 'Number of events: 54'. Below this, a search bar and a table of events are visible. The table has columns for 'Event ID' and 'Task Category', with the value '4624 Logon' highlighted in an orange box. The main pane shows the details for 'Event 4624, Microsoft Windows security auditing.'. The 'General' tab is active, and the 'Friendly View' radio button is selected. The event details are organized into a tree structure under 'system' > 'EventData'. Several fields are highlighted with colored boxes: 'SubjectUserName' (cosmo) and 'SubjectDomainName' (ASTRO) are highlighted in cyan; 'LogonType' (9) and 'LogonProcessName' (seclogon) are highlighted in pink; and 'TargetOutboundUserName' (Administrator) and 'TargetOutboundDomainName' (ASTRO) are highlighted in blue. Other fields include SubjectUserSid, SubjectLogonId, TargetUserSid, TargetDomainName, TargetLogonId, AuthenticationPackageName (Negotiate), WorkstationName, LogonGuid, TransmittedServices, LmPackageName, KeyLength, ProcessId, ProcessName (C:\Windows\System32\svchost.exe), IpAddress (::1), IpPort (0), ImpersonationLevel (%1833), RestrictedAdminMode, VirtualAccount (%1843), TargetLinkedLogonId (0x0), and ElevatedToken (%1843).

Event ID	Task Category
4624	Logon

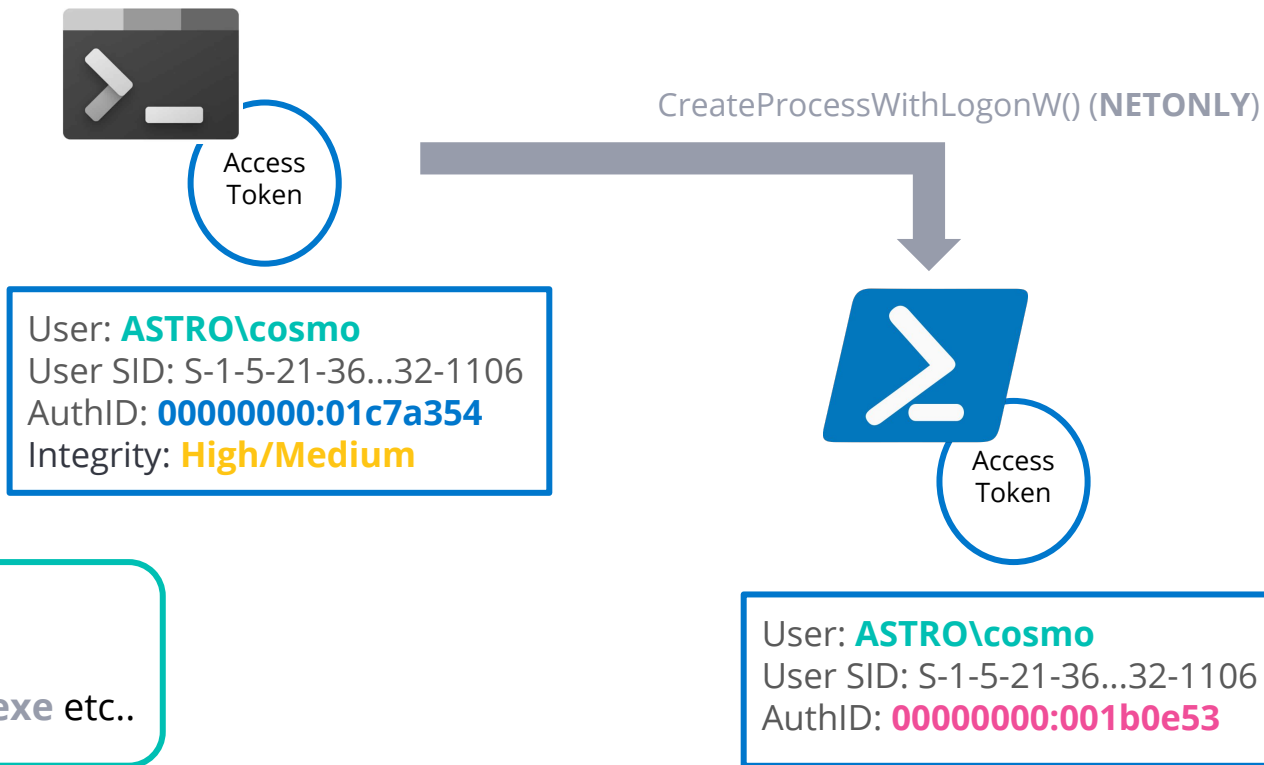
Event 4624, Microsoft Windows security auditing.

General Details

Friendly View XML View

- system
 - EventData
 - SubjectUserSid S-1-5-21-3691787969-2293387988-540293332-1106
 - SubjectUserName cosmo
 - SubjectDomainName ASTRO
 - SubjectLogonId 0x65d25
 - TargetUserSid S-1-5-21-3691787969-2293387988-540293332-1106
 - TargetUserName cosmo
 - TargetDomainName ASTRO
 - TargetLogonId 0x1b0e53
 - LogonType 9
 - LogonProcessName seclogon
 - AuthenticationPackageName Negotiate
 - WorkstationName -
 - LogonGuid {00000000-0000-0000-0000-000000000000}
 - TransmittedServices -
 - LmPackageName -
 - KeyLength 0
 - ProcessId 0x178
 - ProcessName C:\Windows\System32\svchost.exe
 - IpAddress ::1
 - IpPort 0
 - ImpersonationLevel %1833
 - RestrictedAdminMode -
 - TargetOutboundUserName Administrator
 - TargetOutboundDomainName ASTRO
 - VirtualAccount %1843
 - TargetLinkedLogonId 0x0
 - ElevatedToken %1843

Spawning a Process with the NETONLY Flag



Spawning a Process as Another User



Access
Token

CreateProcessWithLogonW()
CreateProcessWithTokenW()
CreateProcessAsUserA() (*Can spoof ppid)



User: **ASTRO\cosmo**
User SID: S-1-5-21-36...32-1106
AuthID: **00000000:01c7a354**
Integrity: **High/Medium**



Access
Token

User: **ASTRO\Administrator**
User SID: **S-1-5-21-36...32-500**
AuthID: 00000000:001aa83d

Ignore:
Runas.exe
ShellRunAs.exe etc..

Spawning a Suspicious/Unsigned Process as Another User

Sample EQL query:

```
sequence
  [process where
    integrity_level in ("low", "medium", "high") and
    not process_name in ("runas.exe", "ShellRunas.exe") ] by unique_pid
  [process where
    (
      process_name in ("cmd.exe", "msbuild.exe", /* etc. */)
      or
      signature_status != "trusted"
    )] by unique_ppid
  | filter events[0].user_sid != events[1].user_sid
```


Detecting Access Token Manipulation

Technique	Signals	Other Telemetry Sources
1. NETONLY	<ul style="list-style-type: none">• CreateProcessWithLogonW• LogonUser + Impersonation	<ol style="list-style-type: none">1. Windows Event Logs2. Process events
2. Pass-The-Ticket	<ul style="list-style-type: none">• LsaCallAuthenticationPackage + KERB_SUBMIT_TKT_REQUEST	


Pass-The-Ticket Frida DEMO

Taming the Three-Headed Dog...

Detecting Access Token Manipulation


Technique	Signals	Other Telemetry Sources
1. NETONLY	<ul style="list-style-type: none">• <code>CreateProcessWithLogonW</code>• <code>LogonUser + Impersonation</code>	<ol style="list-style-type: none">1. Windows Event Logs2. Process events
2. Pass-The-Ticket	<ul style="list-style-type: none">• <code>LsaCallAuthenticationPackage + KERB_SUBMIT_TKT_REQUEST</code>	

Detecting Access Token Manipulation

Technique	Signals	Other Telemetry Sources
1. NETONLY	<ul style="list-style-type: none">• CreateProcessWithLogonW• LogonUser + Impersonation	<ol style="list-style-type: none">1. Windows Event Logs2. Process events
2. Pass-The-Ticket	<ul style="list-style-type: none">• LsaCallAuthenticationPackage + KERB_SUBMIT_TKT_REQUEST	
3. Overpass-The-Hash	<ul style="list-style-type: none">• CreateProcessWithLogonW• Impersonate SYSTEM token• <i>Write</i> handle access to lsass	

Overpass-The-Hash Frida DEMO

Detecting Access Token Manipulation

Technique	Signals	Other Telemetry Sources
1. NETONLY	<ul style="list-style-type: none">• CreateProcessWithLogonW• LogonUser + Impersonation	<ol style="list-style-type: none">1. Windows Event Logs2. Process events
2. Pass-The-Ticket	<ul style="list-style-type: none">• LsaCallAuthenticationPackage + KERB_SUBMIT_TKT_REQUEST	
3. Overpass-The-Hash	<ul style="list-style-type: none">• CreateProcessWithLogonW• Impersonate SYSTEM token• <i>Write</i> handle access to Lsass	<ol style="list-style-type: none">1. Windows Event Logs2. Process events3. Sysmon Event ID 10: Process Access to Lsass

Conclusion

Conclusion

- Windows Security can appear quite intimidating but is **conceptually** simple
- Irrespective of what tools are used or what authentication provider is abused, attackers act under a set of constraints that result in the **same anomalous signals** for access token manipulation
- The techniques shown in this presentation are not intended to be production ready but show the **art of the possible** from both an **offensive** and **defensive** perspective
- This can act as a springboard for defence practitioners to determine their own ability to **detect** and **respond** to these attacks and as ideas for future **threat hunts**



Thank You

—
Search. Observe. Protect.

