OFFZONE
2025

Windows Harvest: Extracting Windows Secrets Under the Radar

Haidar Kabibo

Middle application security specialist, Assume Birch

@purpleshift, @assume_birch

Whoami_

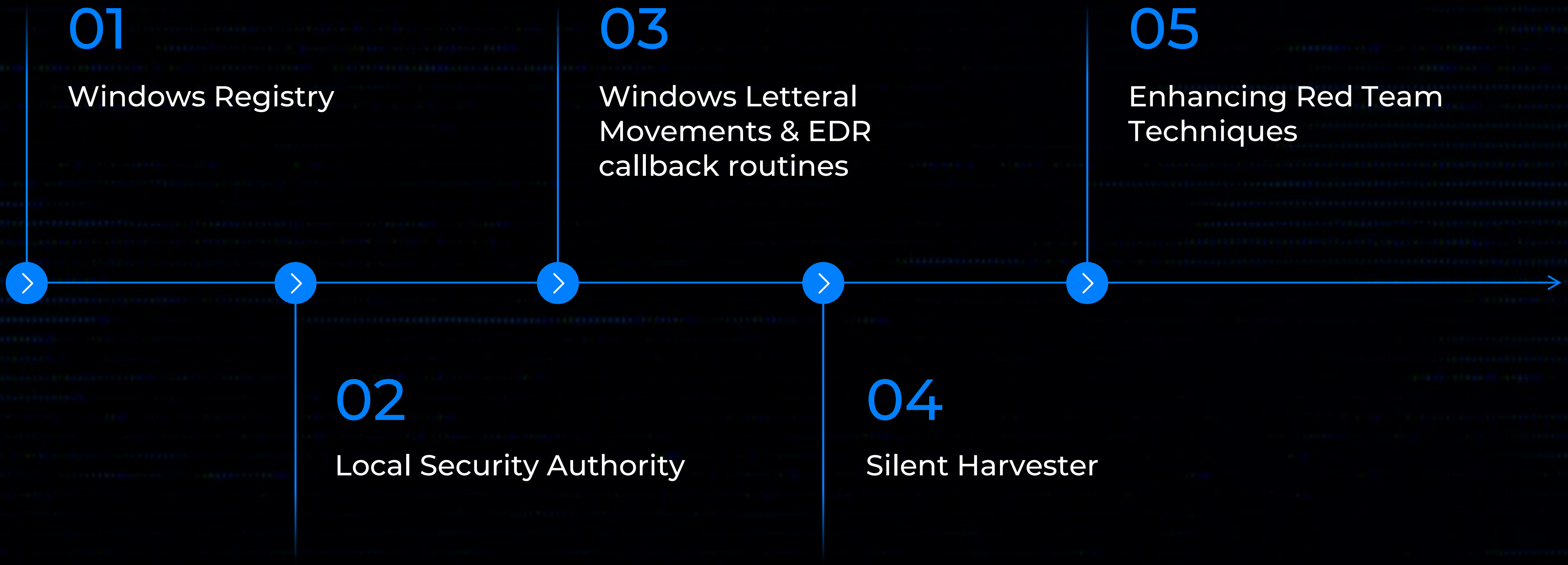
- Member of Industrial Security Services team
- Do Mainly Windows Researches
- Publisher of NauthNRPC for windows users enumeration
- Masters of None:
  - Pentest
  - RE
  - AppSec
  - ICS
  - Network
  - Radio

```
$ echo d2hvYW1pCg== |  base64 –d  |  bash
Sud0Ru
```

NauthNRPC

What this talk about?

RoadMap



01 Windows Registry

02 Local Security Authority

03 Windows Letteral Movements & EDR callback routines

04 Silent Harvester

05 Enhancing Red Team Techniques

Windows Registry

Windows Registry

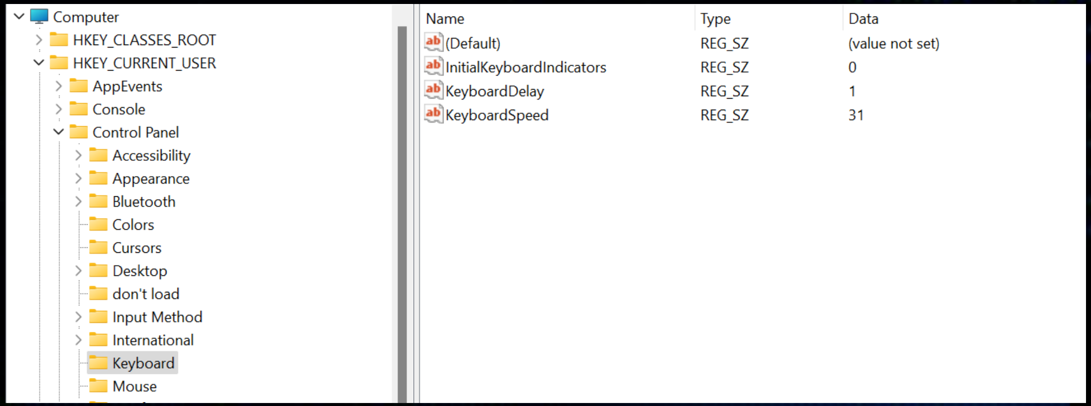
The Registry_

- Central database for OS & application configuration
- Replaced old INI files for unified settings management
- Stores: User profile, System & hardware info, App settings & file associations, Device driver configurations
- Enables: Fast access & updates, Persistent state across reboots
- Used by: Kernel, drivers, services, SAM, applications

Windows Registry

Registry Architectural Structure_

- Organized as a tree of keys and values
- Key = folder; can contain subkeys & values
- Value = file; holds named data of a defined type
- Paths use backslashes (e.g., HKEY_LOCAL_MACHINE\Software\...)
- Top-level keys are root keys (predefined keys)
- Root keys start with HKEY and have standard abbreviations



- Computer
  - HKEY_CLASSES_ROOT
  - HKEY_CURRENT_USER
    - AppEvents
    - Console
    - Control Panel
      - Accessibility
      - Appearance
      - Bluetooth
      - Colors
      - Cursors
      - Desktop
      - don't load
      - Input Method
      - International
      - Keyboard
      - Mouse

Name	Type	Data
(Default)	REG_SZ	(value not set)
InitialKeyboardIndicators	REG_SZ	0
KeyboardDelay	REG_SZ	1
KeyboardSpeed	REG_SZ	31

Windows Registry

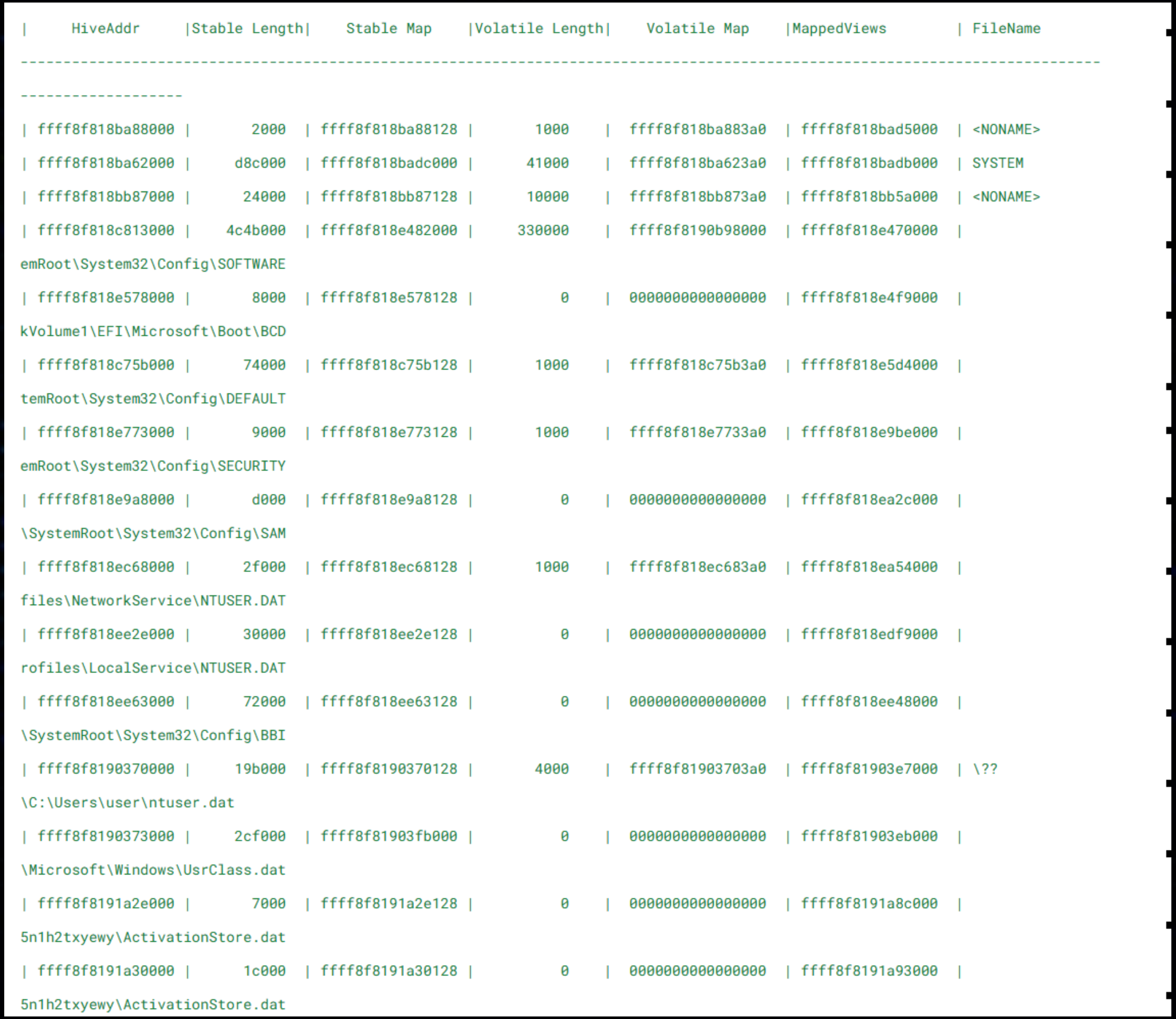
Root Key	Short description	Data on the disk
HKEY_LOCAL_MACHINE (HKLM)	Computer-wide configuration: hardware info, OS settings, installed software. Includes the critical SAM, SYSTEM, SECURITY and SOFTWARE sub-hives that load at boot	Separate files in %SystemRoot%\System32\Config\* • SOFTWARE• SYSTEM• SAM• SECURITY• DEFAULT
HKEY_USERS (HKU)	Contains every loaded user profile. Each account has a subkey named by its security identifier (SID); the active user's subkey is mirrored in HKCU.	One NTUSER.DAT (and USRCLASS.DAT) file per user, stored under each user's profile directory (e.g., **C:\Users\Alice**).
HKEY_CURRENT_USER (HKCU)	Settings for the user who is currently signed in—desktop, environment variables, app prefs. It's just a view into that user's SID key under HKU.	%UserProfile%\NTUSER.DAT (main profile data)+ %LocalAppData%\Microsoft\Windows\USRCLASS.DAT (per-user class registrations)

Windows Registry

Registry hives_

- A hive is a logical group of registry keys, subkeys, and values
- Acts as a standalone database in regf format
- Loaded into memory at OS startup or user login (represented by CMHIVE structure in kernel space)
- Each hive serves a specific system purpose
- For example: SYSTEM, SECURTY, SAM

Windows Registry



```
|     HiveAddr      |Stable Length|    Stable Map     |Volatile Length|    Volatile Map    |MappedViews         | FileName
--------------------------------------------------------------------------------------------------------------------------------
-------------------
| ffff8f818ba88000 |        2000   | ffff8f818ba88128 |       1000      |  ffff8f818ba883a0  | ffff8f818bad5000  | <NONAME>
| ffff8f818ba62000 |      d8c000   | ffff8f818badc000 |      41000      |  ffff8f818ba623a0  | ffff8f818badb000  | SYSTEM
| ffff8f818bb87000 |       24000   | ffff8f818bb87128 |      10000      |  ffff8f818bb873a0  | ffff8f818bb5a000  | <NONAME>
| ffff8f818c813000 |     4c4b000   | ffff8f818e482000 |     330000      |  ffff8f8190b98000  | ffff8f818e470000  |
emRoot\System32\Config\SOFTWARE
| ffff8f818e578000 |        8000   | ffff8f818e578128 |          0      |  0000000000000000  | ffff8f818e4f9000  |
kVolume1\EFI\Microsoft\Boot\BCD
| ffff8f818c75b000 |       74000   | ffff8f818c75b128 |       1000      |  ffff8f818c75b3a0  | ffff8f818e5d4000  |
temRoot\System32\Config\DEFAULT
| ffff8f818e773000 |        9000   | ffff8f818e773128 |       1000      |  ffff8f818e7733a0  | ffff8f818e9be000  |
emRoot\System32\Config\SECURITY
| ffff8f818e9a8000 |        d000   | ffff8f818e9a8128 |          0      |  0000000000000000  | ffff8f818ea2c000  |
\SystemRoot\System32\Config\SAM
| ffff8f818ec68000 |       2f000   | ffff8f818ec68128 |       1000      |  ffff8f818ec683a0  | ffff8f818ea54000  |
files\NetworkService\NTUSER.DAT
| ffff8f818ee2e000 |       30000   | ffff8f818ee2e128 |          0      |  0000000000000000  | ffff8f818edf9000  |
rofiles\LocalService\NTUSER.DAT
| ffff8f818ee63000 |       72000   | ffff8f818ee63128 |          0      |  0000000000000000  | ffff8f818ee48000  |
\SystemRoot\System32\Config\BBI
| ffff8f8190370000 |      19b000   | ffff8f8190370128 |       4000      |  ffff8f81903703a0  | ffff8f81903e7000  | \??
\C:\Users\user\ntuser.dat
| ffff8f8190373000 |      2cf000   | ffff8f81903fb000 |          0      |  0000000000000000  | ffff8f81903eb000  |
\Microsoft\Windows\UsrClass.dat
| ffff8f8191a2e000 |        7000   | ffff8f8191a2e128 |          0      |  0000000000000000  | ffff8f8191a8c000  |
5n1h2txyewy\ActivationStore.dat
| ffff8f8191a30000 |       1c000   | ffff8f8191a30128 |          0      |  0000000000000000  | ffff8f8191a93000  |
5n1h2txyewy\ActivationStore.dat
```

Windows Registry

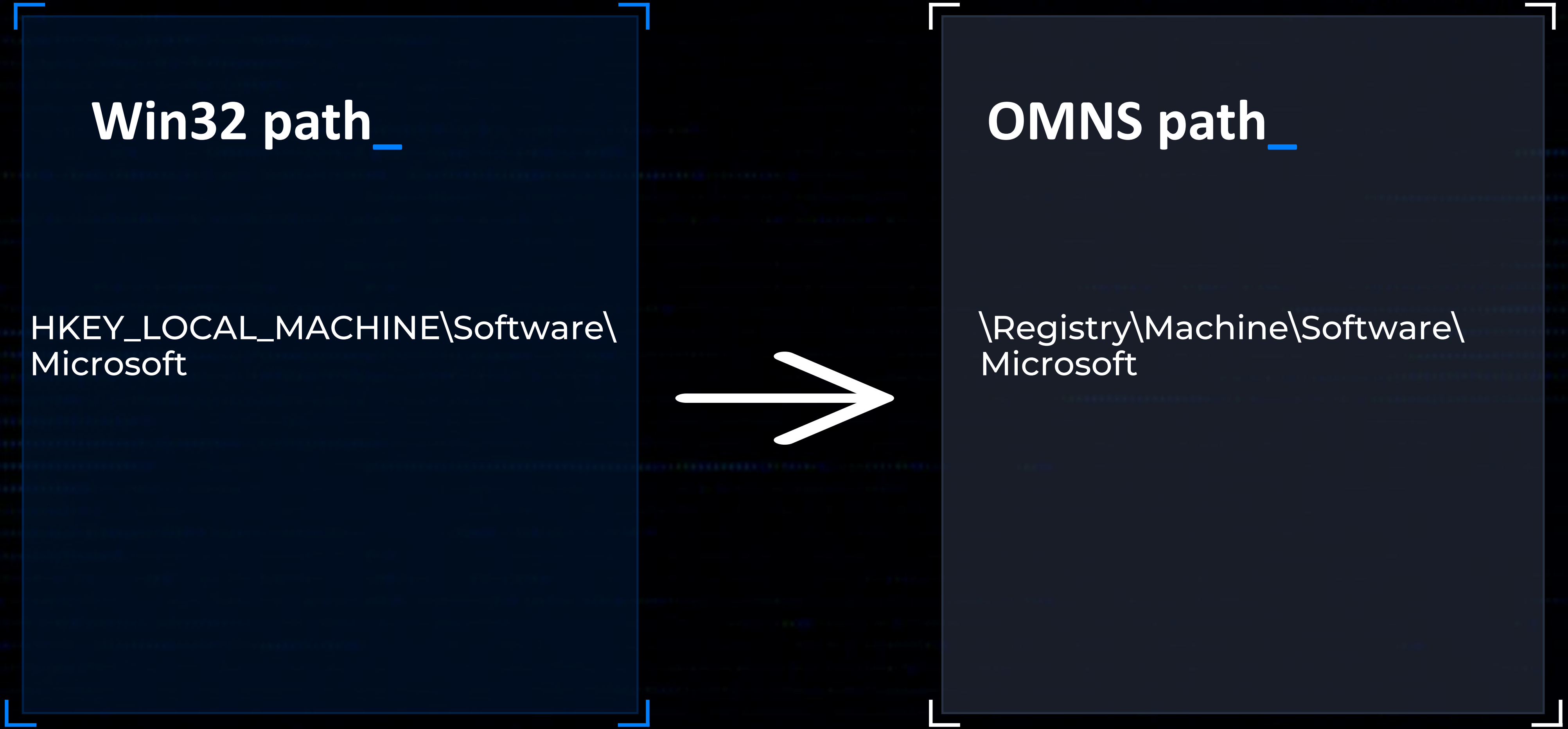
Registry as Kernel Objects_

- Windows kernel uses a unified object model (e.g., processes, files, mutexes)
- All objects are reference-counted and live in system space
- User-mode apps interact via handles, preserving system integrity
- The registry is implemented by the Configuration Manager (CM)
- Registry keys = kernel objects, managed by the Object Manager
- Root of registry in kernel namespace: \Registry

Windows Registry
Sub Keys under \Registry

Sub-Key	Kernel-level view
A	Application-hive root. Every time a process calls RegLoadAppKey, Windows mounts that hive under \REGISTRY\A\<GUID>\....
MACHINE	The well-known system hive that becomes HKEY_LOCAL_MACHINE.
USER	Root of all user hives—what we see as HKEY_USERS (and, via linking, HKEY_CURRENT_USER).
WC	Windows-Container root.

Windows Registry



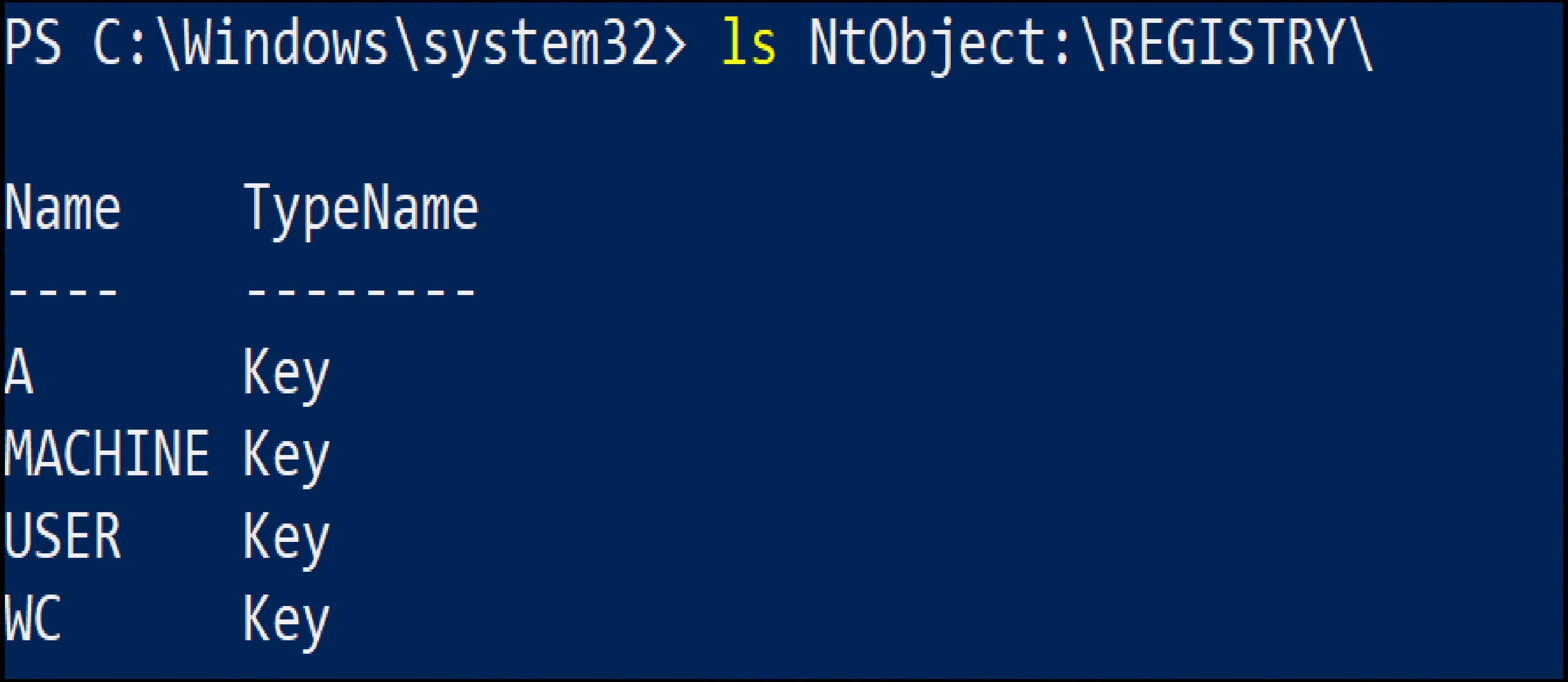
Win32 path_

HKEY_LOCAL_MACHINE\Software\Microsoft

OMNS path_

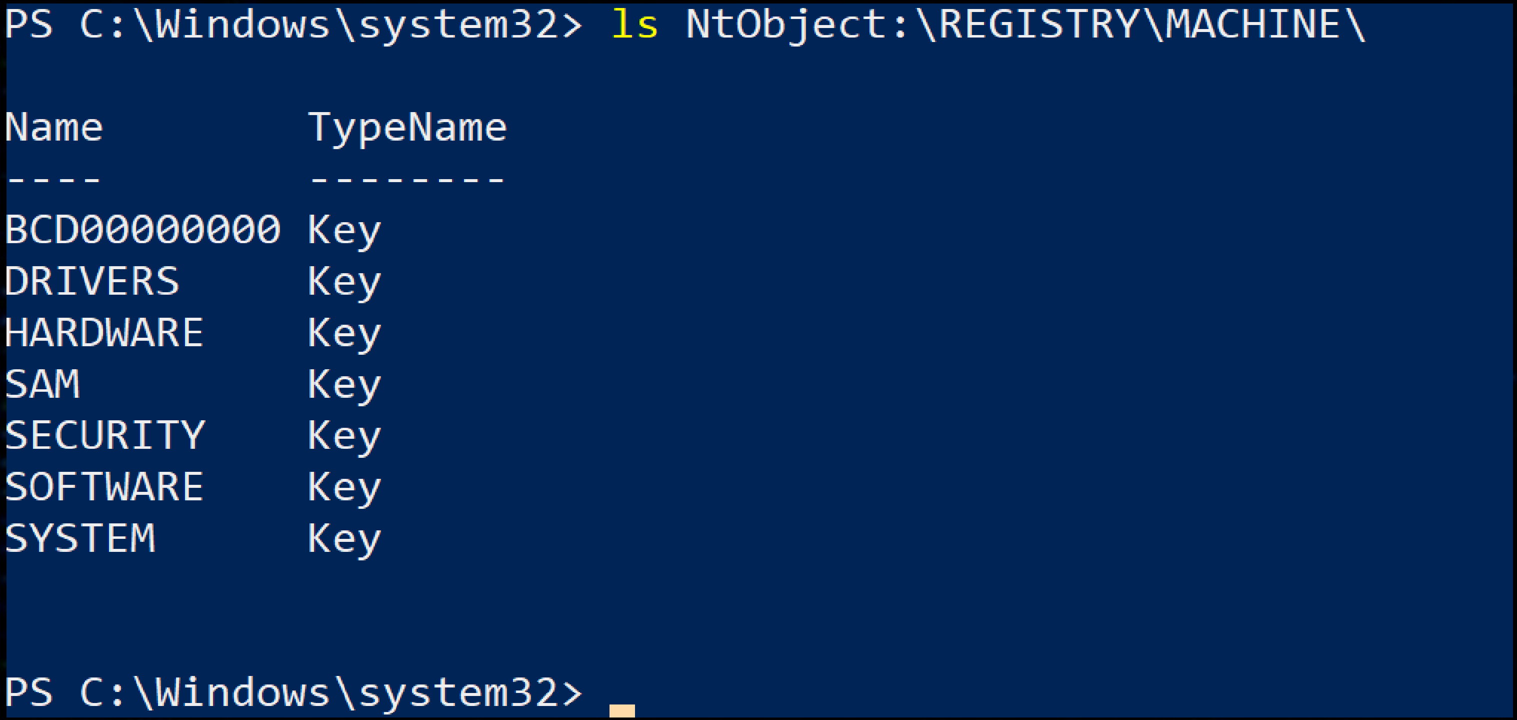
\Registry\Machine\Software\Microsoft

Windows Registry
NtObjectManager

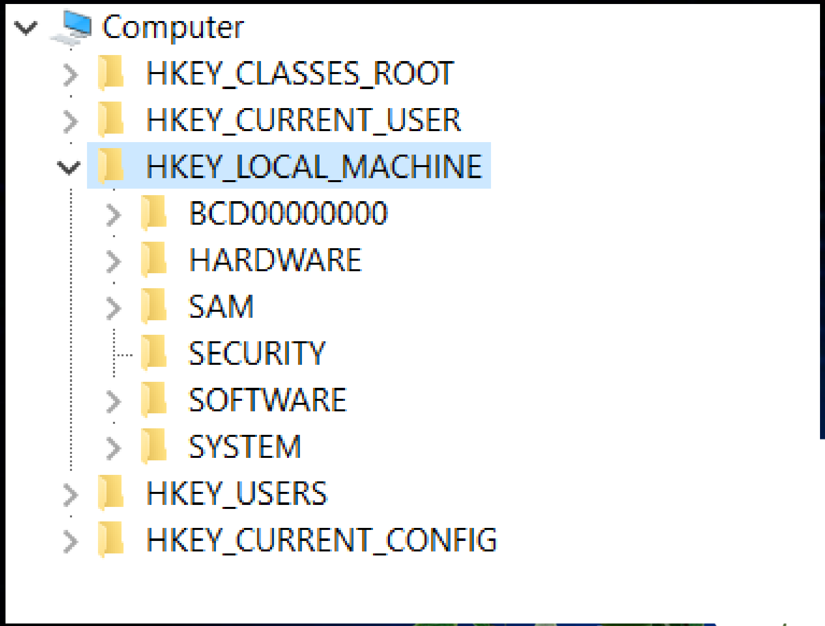


```
PS C:\Windows\system32> ls NtObject:\REGISTRY\

Name    TypeName
----    --------
A       Key
MACHINE Key
USER    Key
WC      Key
```

Windows Registry
NtObjectManager

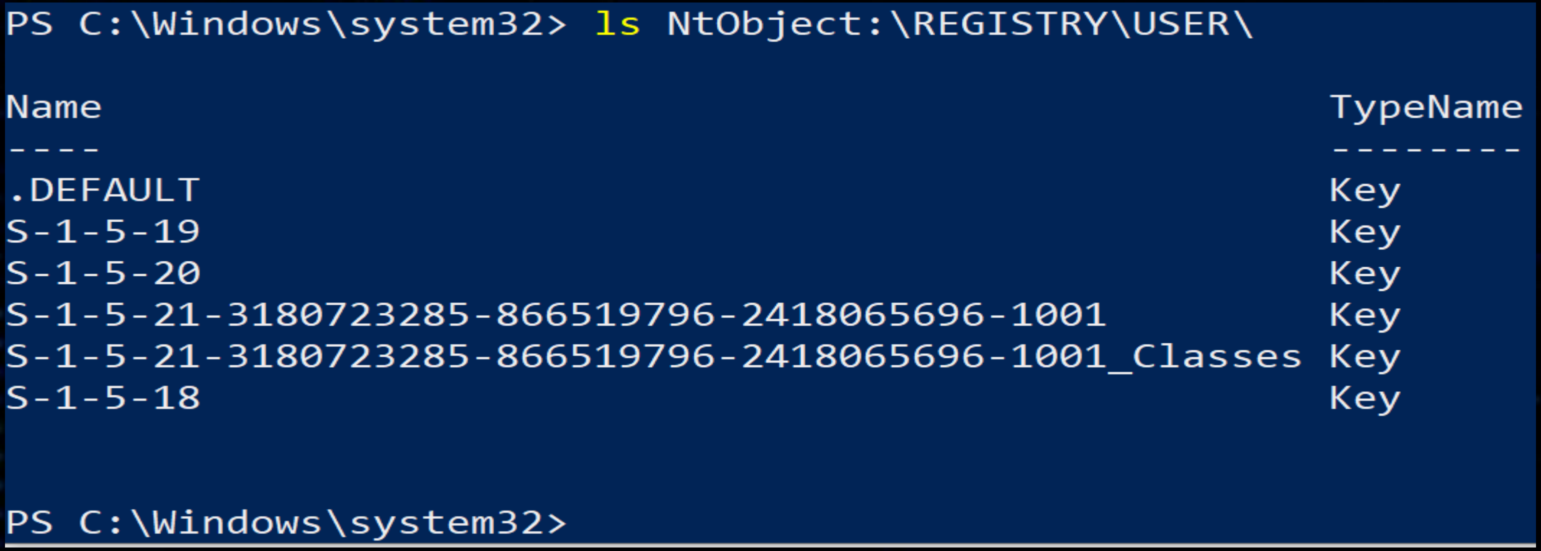


```
PS C:\Windows\system32> ls NtObject:\REGISTRY\MACHINE\

Name         TypeName
----         --------
BCD00000000  Key
DRIVERS      Key
HARDWARE     Key
SAM          Key
SECURITY     Key
SOFTWARE     Key
SYSTEM       Key


PS C:\Windows\system32> _
```

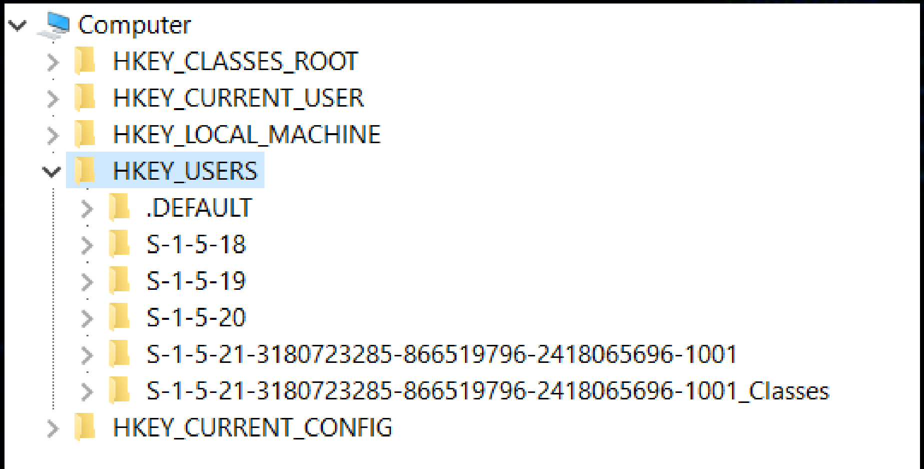


```
Computer
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
    BCD00000000
    HARDWARE
    SAM
    SECURITY
    SOFTWARE
    SYSTEM
  HKEY_USERS
  HKEY_CURRENT_CONFIG
```

Windows Registry
NtObjectManager



```
PS C:\Windows\system32> ls NtObject:\REGISTRY\USER\

Name                                                  TypeName
----                                                  --------
.DEFAULT                                              Key
S-1-5-19                                              Key
S-1-5-20                                              Key
S-1-5-21-3180723285-866519796-2418065696-1001         Key
S-1-5-21-3180723285-866519796-2418065696-1001_Classes Key
S-1-5-18                                              Key


PS C:\Windows\system32>
```



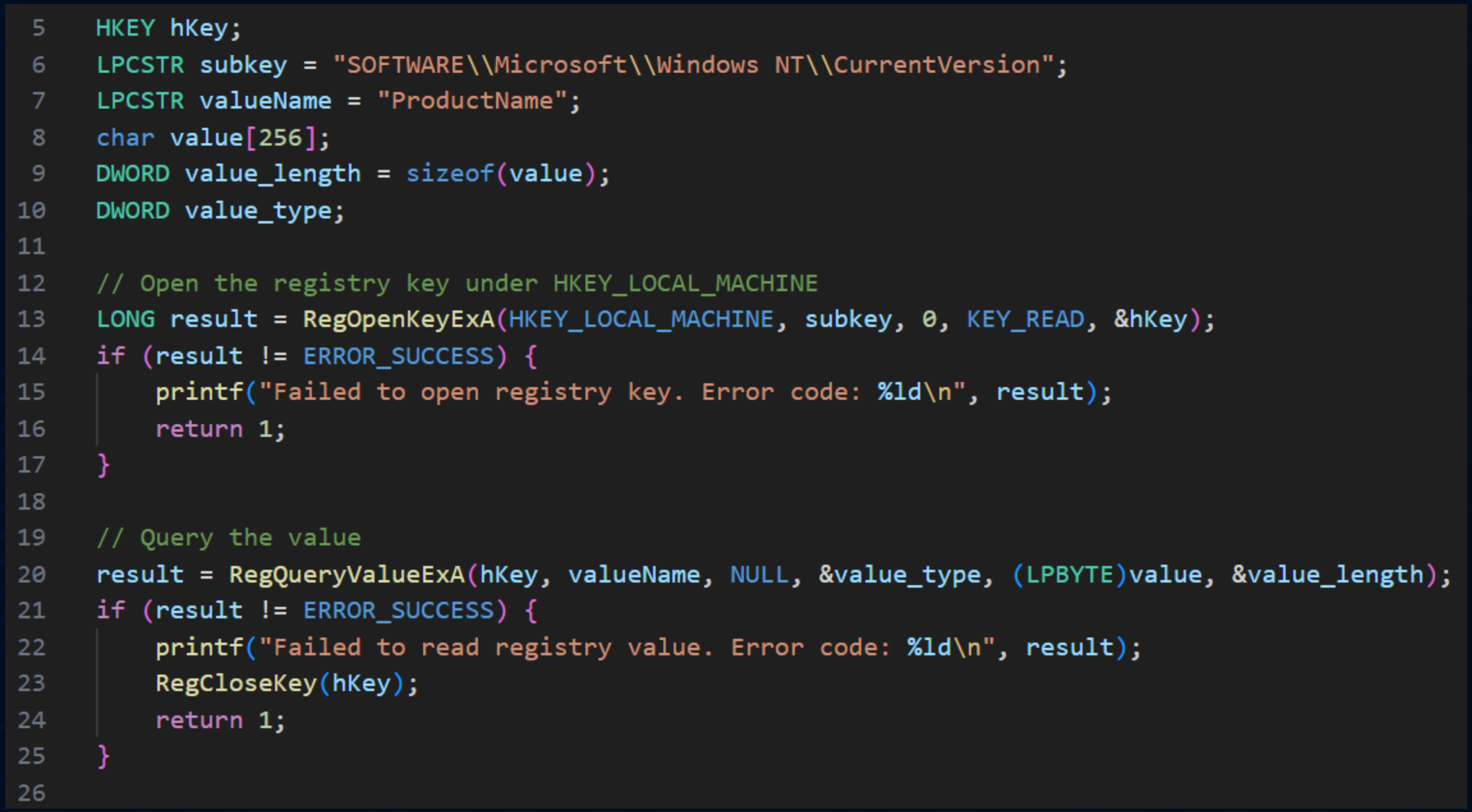
```
Computer
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
  HKEY_USERS
    .DEFAULT
    S-1-5-18
    S-1-5-19
    S-1-5-20
    S-1-5-21-3180723285-866519796-2418065696-1001
    S-1-5-21-3180723285-866519796-2418065696-1001_Classes
  HKEY_CURRENT_CONFIG
```

Windows Registry

Deep Down To NTDLL_

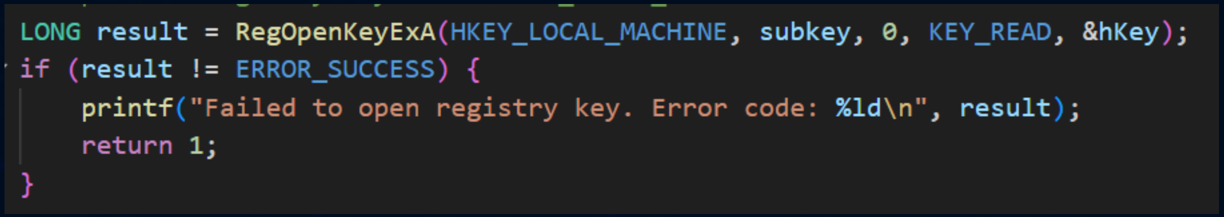
- let’s look at how the actual opening of a registry key happens, from user-mode down to ntdll
- Read a value inside a specific key
- “ProductName” inside “SOFTWARE\Microsoft\Windows NT\CurrentVersion”
- This subkey exists in most Windows versions.
- we will focus on how to open the key, rather than how to read the value itself

Windows Registry Deep Down To NTDLL

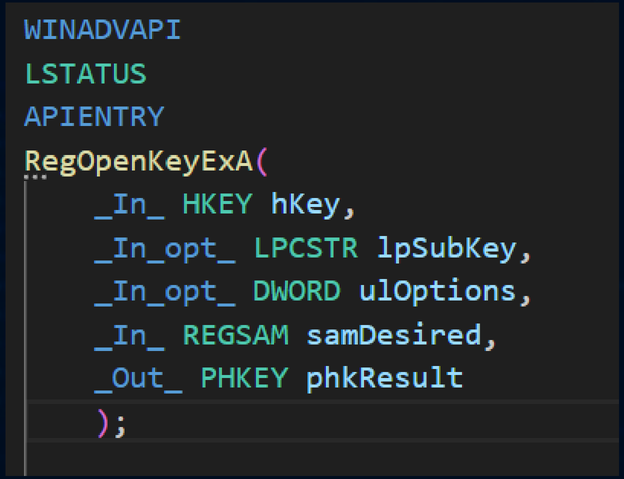


```c
HKEY hKey;
LPCSTR subkey = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion";
LPCSTR valueName = "ProductName";
char value[256];
DWORD value_length = sizeof(value);
DWORD value_type;

// Open the registry key under HKEY_LOCAL_MACHINE
LONG result = RegOpenKeyExA(HKEY_LOCAL_MACHINE, subkey, 0, KEY_READ, &hKey);
if (result != ERROR_SUCCESS) {
    printf("Failed to open registry key. Error code: %ld\n", result);
    return 1;
}

// Query the value
result = RegQueryValueExA(hKey, valueName, NULL, &value_type, (LPBYTE)value, &value_length);
if (result != ERROR_SUCCESS) {
    printf("Failed to read registry value. Error code: %ld\n", result);
    RegCloseKey(hKey);
    return 1;
}

```

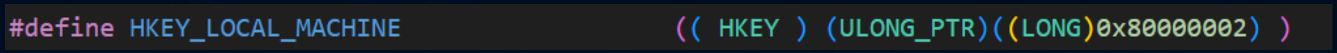
Windows Registry
Deep Down To NTDLL



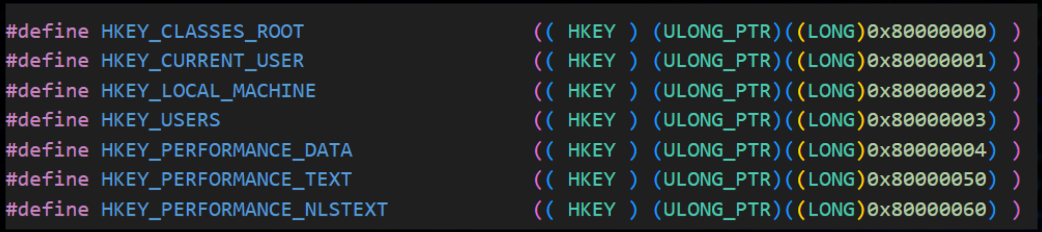
```c
LONG result = RegOpenKeyExA(HKEY_LOCAL_MACHINE, subkey, 0, KEY_READ, &hKey);
if (result != ERROR_SUCCESS) {
    printf("Failed to open registry key. Error code: %ld\n", result);
    return 1;
}
```



```c
WINADVAPI
LSTATUS
APIENTRY
RegOpenKeyExA(
    _In_ HKEY hKey,
    _In_opt_ LPCSTR lpSubKey,
    _In_opt_ DWORD ulOptions,
    _In_ REGSAM samDesired,
    _Out_ PHKEY phkResult
    );
```

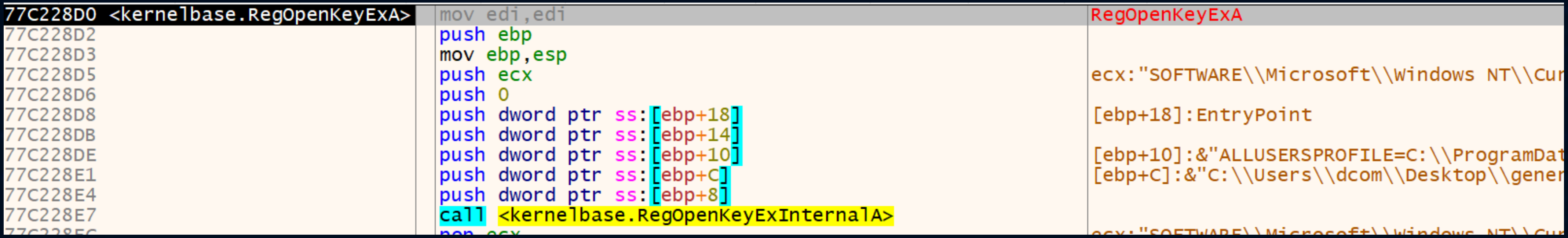


```c
#define HKEY_LOCAL_MACHINE                  (( HKEY ) (ULONG_PTR)((LONG)0x80000002) )
```

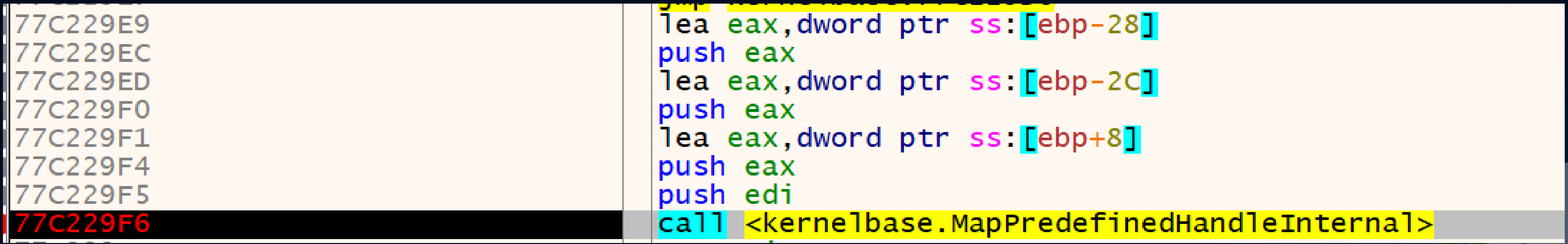


```c
#define HKEY_CLASSES_ROOT                   (( HKEY ) (ULONG_PTR)((LONG)0x80000000) )
#define HKEY_CURRENT_USER                   (( HKEY ) (ULONG_PTR)((LONG)0x80000001) )
#define HKEY_LOCAL_MACHINE                  (( HKEY ) (ULONG_PTR)((LONG)0x80000002) )
#define HKEY_USERS                          (( HKEY ) (ULONG_PTR)((LONG)0x80000003) )
#define HKEY_PERFORMANCE_DATA               (( HKEY ) (ULONG_PTR)((LONG)0x80000004) )
#define HKEY_PERFORMANCE_TEXT               (( HKEY ) (ULONG_PTR)((LONG)0x80000050) )
#define HKEY_PERFORMANCE_NLSTEXT            (( HKEY ) (ULONG_PTR)((LONG)0x80000060) )
```

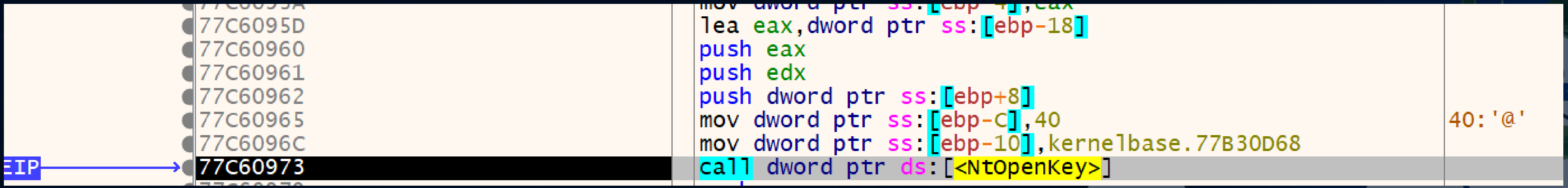
Windows Registry
Deep Down To NTDLL



```
77C228D0 <kernelbase.RegOpenKeyExA>  mov edi,edi                               RegOpenKeyExA
77C228D2                             push ebp
77C228D3                             mov ebp,esp
77C228D5                             push ecx                                  ecx:"SOFTWARE\\Microsoft\\Windows NT\\Cur
77C228D6                             push 0
77C228D8                             push dword ptr ss:[ebp+18]                [ebp+18]:EntryPoint
77C228DB                             push dword ptr ss:[ebp+14]
77C228DE                             push dword ptr ss:[ebp+10]                [ebp+10]:&"ALLUSERSPROFILE=C:\\ProgramDat
77C228E1                             push dword ptr ss:[ebp+C]                 [ebp+C]:&"C:\\Users\\dcom\\Desktop\\gener
77C228E4                             push dword ptr ss:[ebp+8]
77C228E7                             call <kernelbase.RegOpenKeyExInternalA>
```

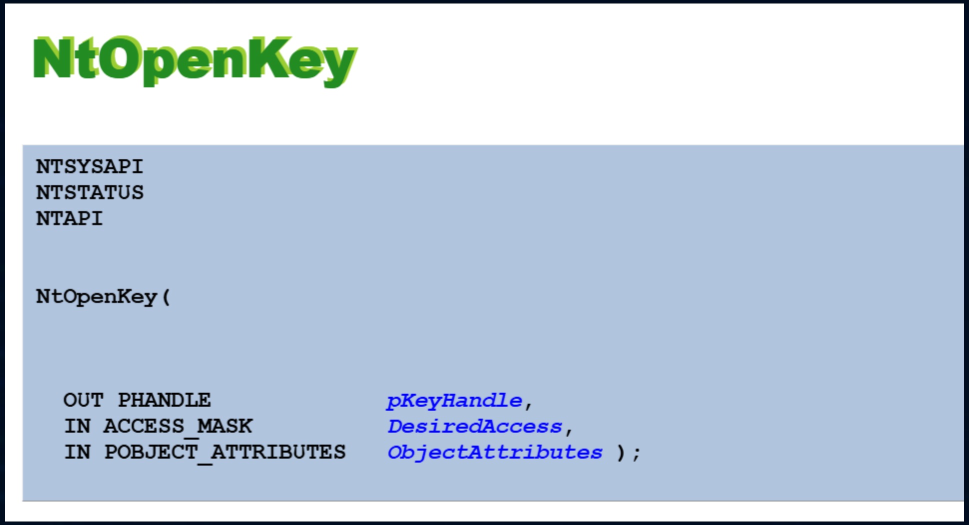


```
77C229E9     lea eax,dword ptr ss:[ebp-28]
77C229EC     push eax
77C229ED     lea eax,dword ptr ss:[ebp-2C]
77C229F0     push eax
77C229F1     lea eax,dword ptr ss:[ebp+8]
77C229F4     push eax
77C229F5     push edi
77C229F6     call <kernelbase.MapPredefinedHandleInternal>
```



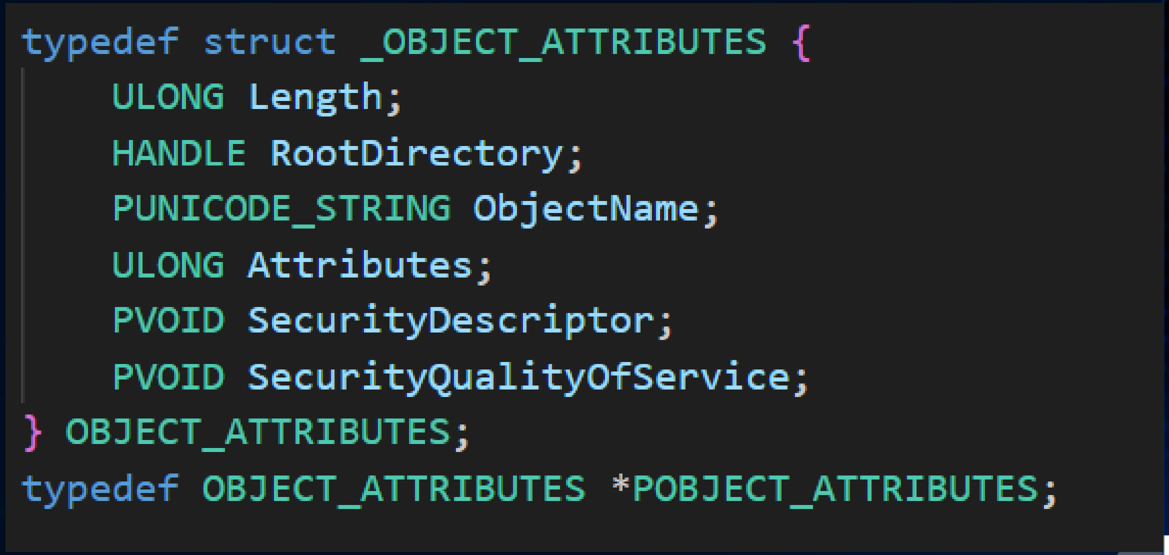
```
          77C6095D   lea eax,dword ptr ss:[ebp-18]
          77C60960   push eax
          77C60961   push edx
          77C60962   push dword ptr ss:[ebp+8]
          77C60965   mov dword ptr ss:[ebp-C],40                    40:'@'
          77C6096C   mov dword ptr ss:[ebp-10],kernelbase.77B30D68
EIP ----> 77C60973   call dword ptr ds:[<NtOpenKey>]
```

Windows Registry
Deep Down To NTDLL



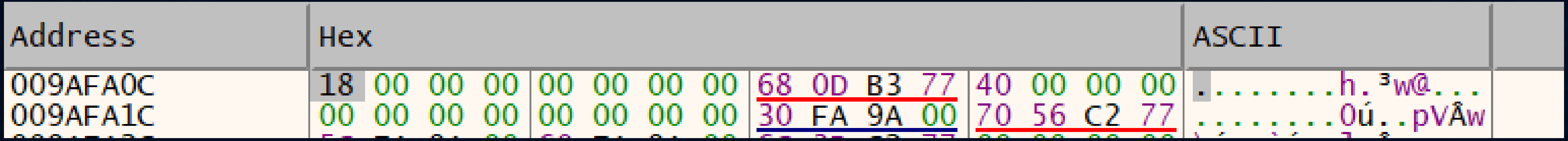
NtOpenKey

```
NTSYSAPI
NTSTATUS
NTAPI


NtOpenKey(



  OUT PHANDLE               pKeyHandle,
  IN ACCESS_MASK            DesiredAccess,
  IN POBJECT_ATTRIBUTES     ObjectAttributes );
```

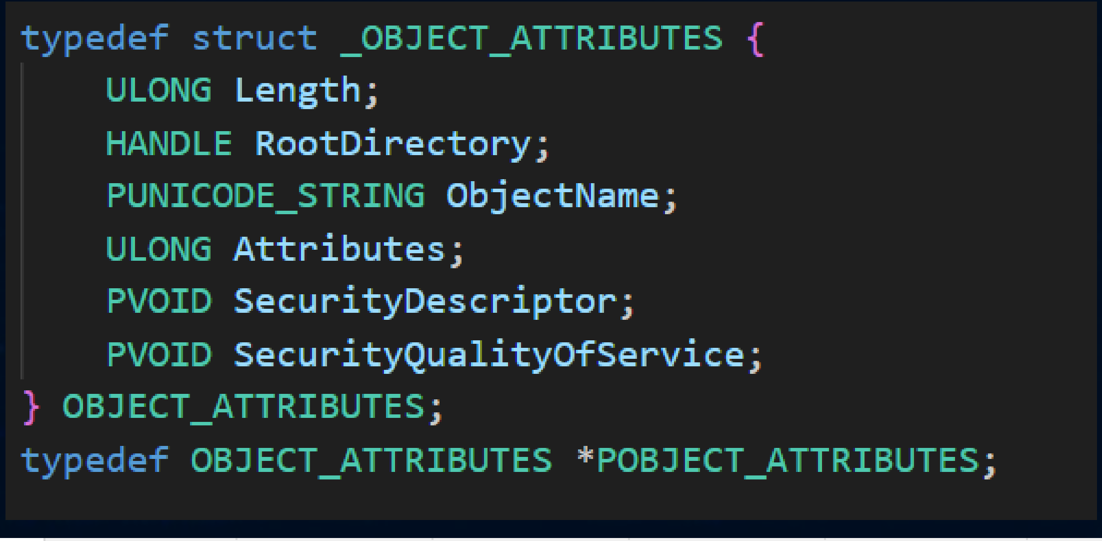


```
typedef struct _OBJECT_ATTRIBUTES {
    ULONG Length;
    HANDLE RootDirectory;
    PUNICODE_STRING ObjectName;
    ULONG Attributes;
    PVOID SecurityDescriptor;
    PVOID SecurityQualityOfService;
} OBJECT_ATTRIBUTES;
typedef OBJECT_ATTRIBUTES *POBJECT_ATTRIBUTES;
```

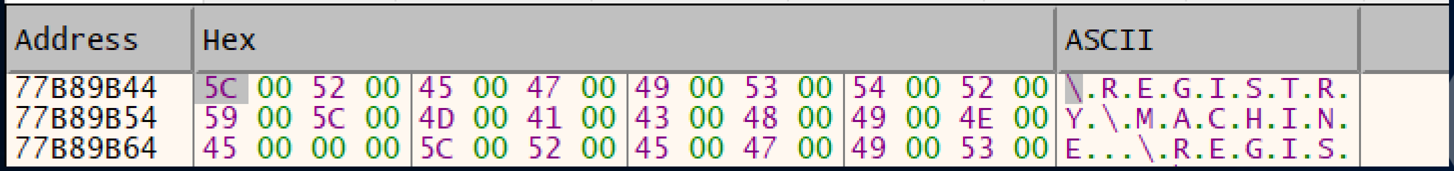
Windows Registry
Deep Down To NTDLL



```
Address   Hex                                              ASCII
009AFA0C  18 00 00 00 00 00 00 00 68 0D B3 77 40 00 00 00  ........h.³w@...
009AFA1C  00 00 00 00 00 00 00 00 30 FA 9A 00 70 56 C2 77  ........0ú..pVÂw
```

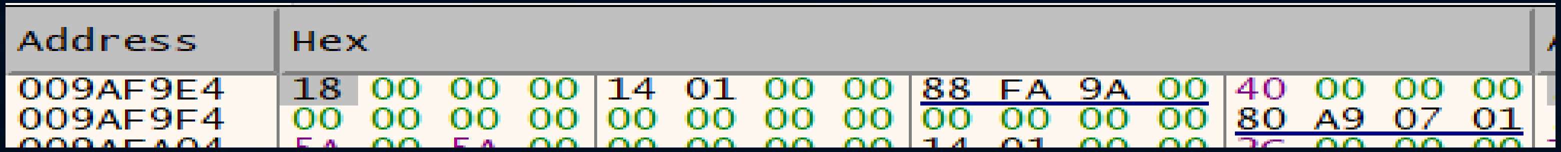


```
typedef struct _OBJECT_ATTRIBUTES {
    ULONG Length;
    HANDLE RootDirectory;
    PUNICODE_STRING ObjectName;
    ULONG Attributes;
    PVOID SecurityDescriptor;
    PVOID SecurityQualityOfService;
} OBJECT_ATTRIBUTES;
typedef OBJECT_ATTRIBUTES *POBJECT_ATTRIBUTES;
```

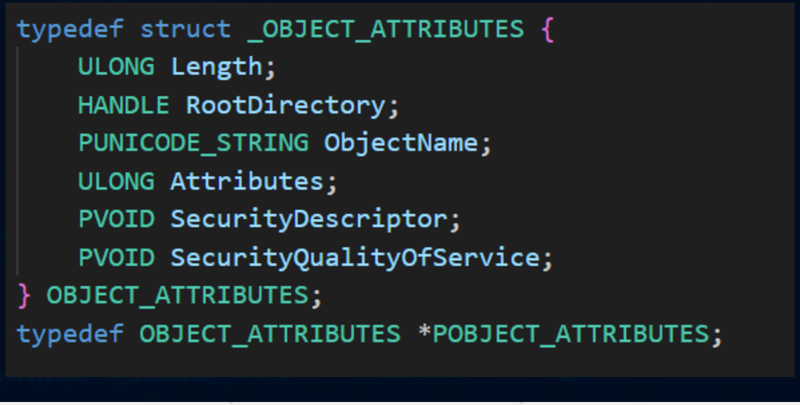


```
Address   Hex                                              ASCII
77B89B44  5C 00 52 00 45 00 47 00 49 00 53 00 54 00 52 00  \.R.E.G.I.S.T.R.
77B89B54  59 00 5C 00 4D 00 41 00 43 00 48 00 49 00 4E 00  Y.\.M.A.C.H.I.N.
77B89B64  45 00 00 00 5C 00 52 00 45 00 47 00 49 00 53 00  E...\.R.E.G.I.S.
```

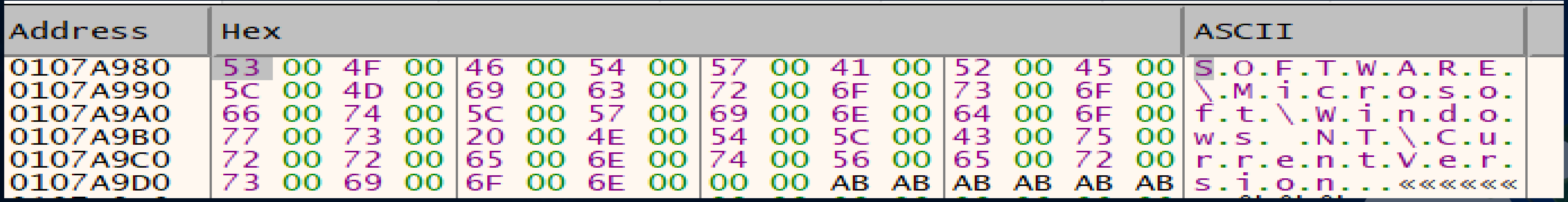
Windows Registry
Deep Down To NTDLL



```
Address   Hex
009AF9E4  18 00 00 00 14 01 00 00 88 FA 9A 00 40 00 00 00
009AF9F4  00 00 00 00 00 00 00 00 00 00 00 00 80 A9 07 01
```

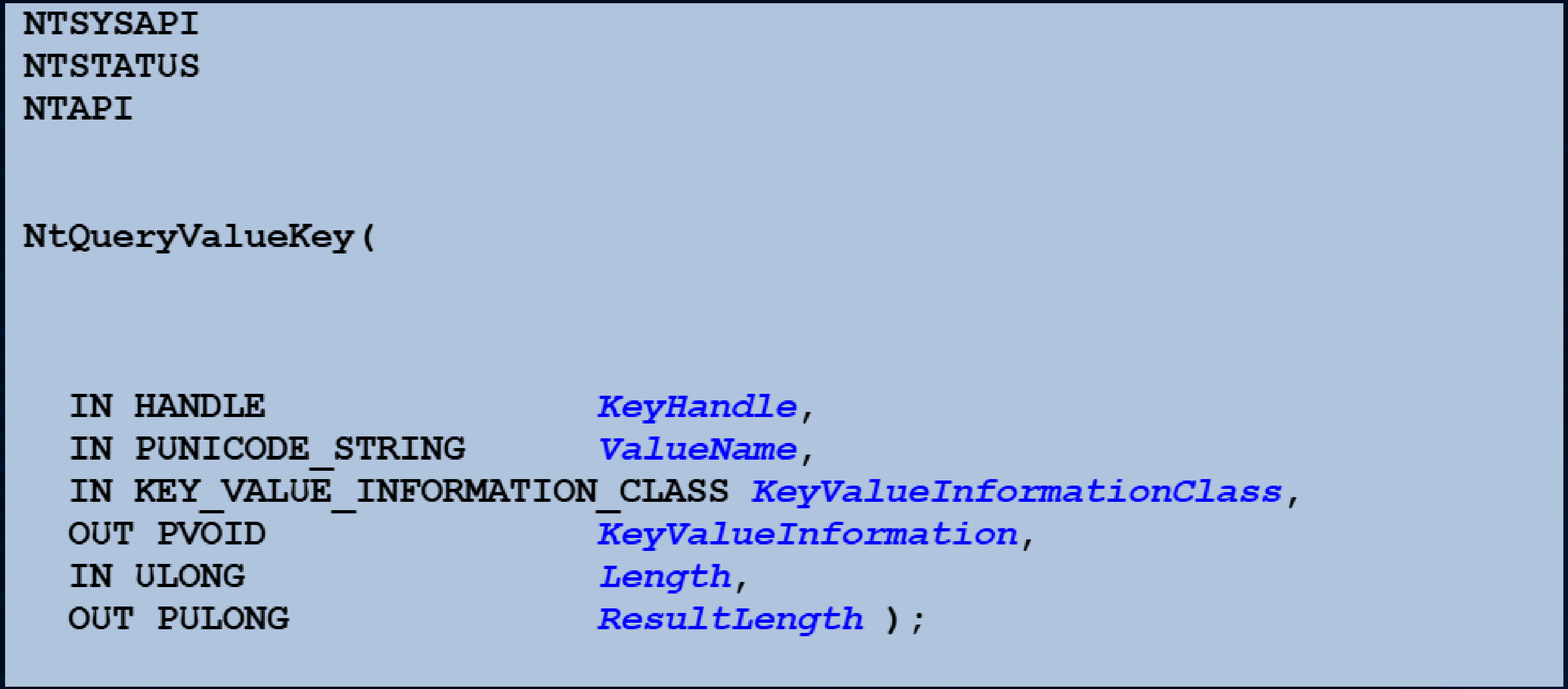


```
typedef struct _OBJECT_ATTRIBUTES {
    ULONG Length;
    HANDLE RootDirectory;
    PUNICODE_STRING ObjectName;
    ULONG Attributes;
    PVOID SecurityDescriptor;
    PVOID SecurityQualityOfService;
} OBJECT_ATTRIBUTES;
typedef OBJECT_ATTRIBUTES *POBJECT_ATTRIBUTES;
```



```
Address   Hex                                              ASCII
0107A980  53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00  S.O.F.T.W.A.R.E.
0107A990  5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00  \.M.i.c.r.o.s.o.
0107A9A0  66 00 74 00 5C 00 57 00 69 00 6E 00 64 00 6F 00  f.t.\.W.i.n.d.o.
0107A9B0  77 00 73 00 20 00 4E 00 54 00 5C 00 43 00 75 00  w.s. .N.T.\.C.u.
0107A9C0  72 00 72 00 65 00 6E 00 74 00 56 00 65 00 72 00  r.r.e.n.t.V.e.r.
0107A9D0  73 00 69 00 6F 00 6E 00 00 00 AB AB AB AB AB AB  s.i.o.n...««««««
```

Windows Registry
Deep Down To NTDLL



```
NTSYSAPI
NTSTATUS
NTAPI


NtQueryValueKey(



   IN HANDLE                    KeyHandle,
   IN PUNICODE_STRING           ValueName,
   IN KEY_VALUE_INFORMATION_CLASS KeyValueInformationClass,
   OUT PVOID                    KeyValueInformation,
   IN ULONG                     Length,
   OUT PULONG                   ResultLength );
```

Local Security Authority (LSA)

Local Security Authority

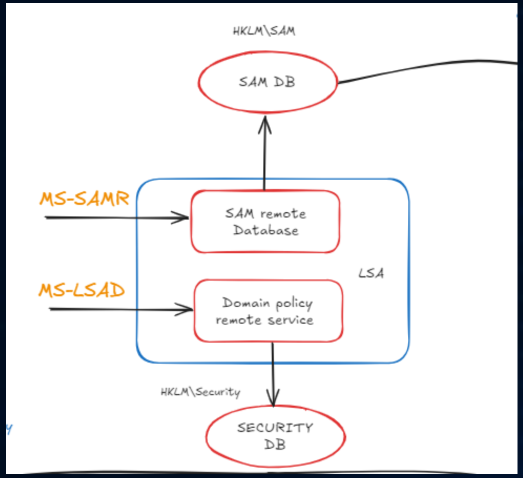
Definition_

- **subsystem that manages all aspects of local security on a computer**
- Logon process starts with identity proof (e.g., username + password)
- Credentials are validated by logon package
- LSA run under LSASS process
- LSA creates access token after authentication
- LSA maintains the user credentials
- These credentials are encrypted & protected by LSA
- Stored in memory and in SAM / SECURITY hives
- Never exposed in plain text

Local Security Authority

On Disk Databases_

- LSA maintains two databases: SAM, Security (Policy)
- SAM database – maps to the SAM registry hive and holds users and groups for the built-in and local domains
- Security (Policy) database – maps to the SECURITY registry hive and stores user privileges, trusted-domain information, and Windows secrets (often called *LSA secrets*).
- Both databases can be accessed remotely through specific RPC protocols that expose well-defined interfaces



Local Security Authority

SAM Database_

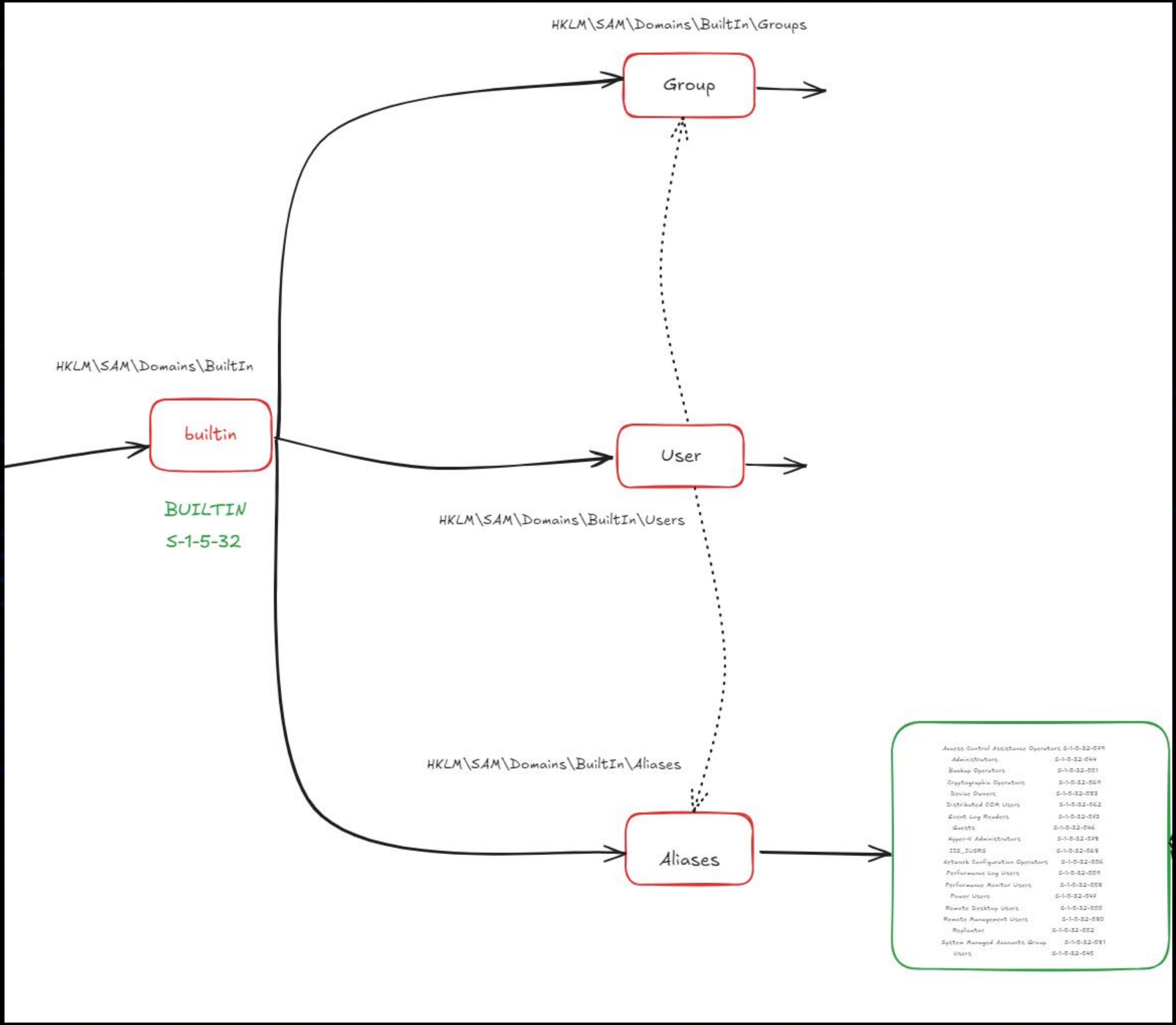
- SAM exposes different type of objects
- Server object - The entire SAM database context on a machine
- Domain object (Local or Built-in)
- Each domain includes 3 object types:

A. User – Windows user account

B. Alias – Local group (e.g., Administrators, Users)

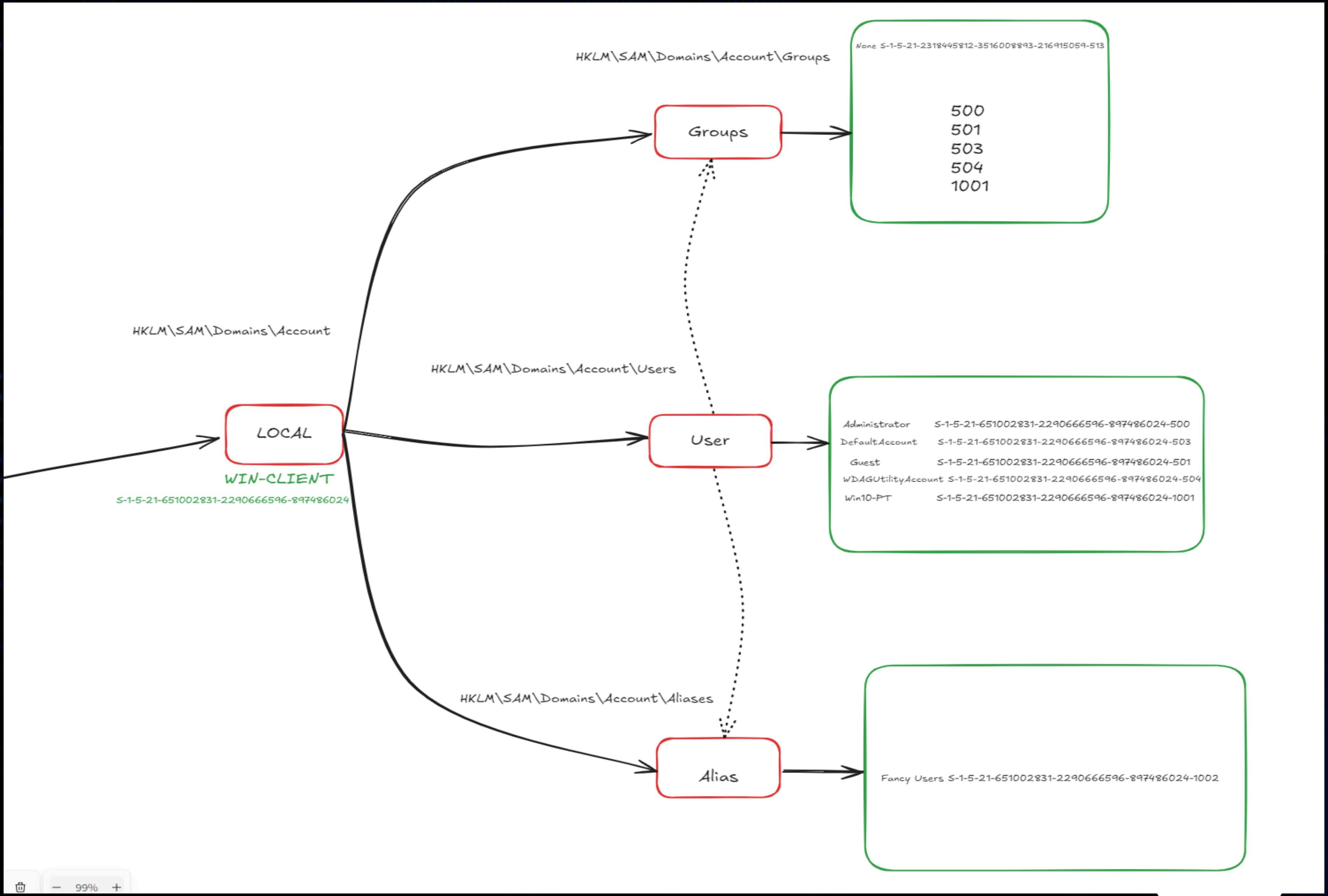
C. Group – Used mainly in Active Directory environments

Local Security Authority
SAM Database


HKLM\SAM\Domains\BuiltIn\Groups
Group
HKLM\SAM\Domains\BuiltIn
builtin
BUILTIN
S-1-5-32
User
HKLM\SAM\Domains\BuiltIn\Users
HKLM\SAM\Domains\BuiltIn\Aliases
Aliases

Access Control Assistance Operators	S-1-5-32-579
Administrators	S-1-5-32-544
Backup Operators	S-1-5-32-551
Cryptographic Operators	S-1-5-32-569
Device Owners	S-1-5-32-583
Distributed COM Users	S-1-5-32-562
Event Log Readers	S-1-5-32-573
Guests	S-1-5-32-546
Hyper-V Administrators	S-1-5-32-578
IIS_IUSRS	S-1-5-32-568
Network Configuration Operators	S-1-5-32-556
Performance Log Users	S-1-5-32-559
Performance Monitor Users	S-1-5-32-558
Power Users	S-1-5-32-547
Remote Desktop Users	S-1-5-32-555
Remote Management Users	S-1-5-32-580
Replicator	S-1-5-32-552
System Managed Accounts Group	S-1-5-32-581
Users	S-1-5-32-545

Local Security Authority
SAM Database



HKLM\SAM\Domains\Account

LOCAL

WIN-CLIENT

S-1-5-21-651002831-2290666596-897486024

HKLM\SAM\Domains\Account\Groups

Groups

None S-1-5-21-2318445812-3516008893-216915059-513

500
501
503
504
1001

HKLM\SAM\Domains\Account\Users

User

Administrator	S-1-5-21-651002831-2290666596-897486024-500
DefaultAccount	S-1-5-21-651002831-2290666596-897486024-503
Guest	S-1-5-21-651002831-2290666596-897486024-501
WDAGUtilityAccount	S-1-5-21-651002831-2290666596-897486024-504
Win10-PT	S-1-5-21-651002831-2290666596-897486024-1001

HKLM\SAM\Domains\Account\Aliases

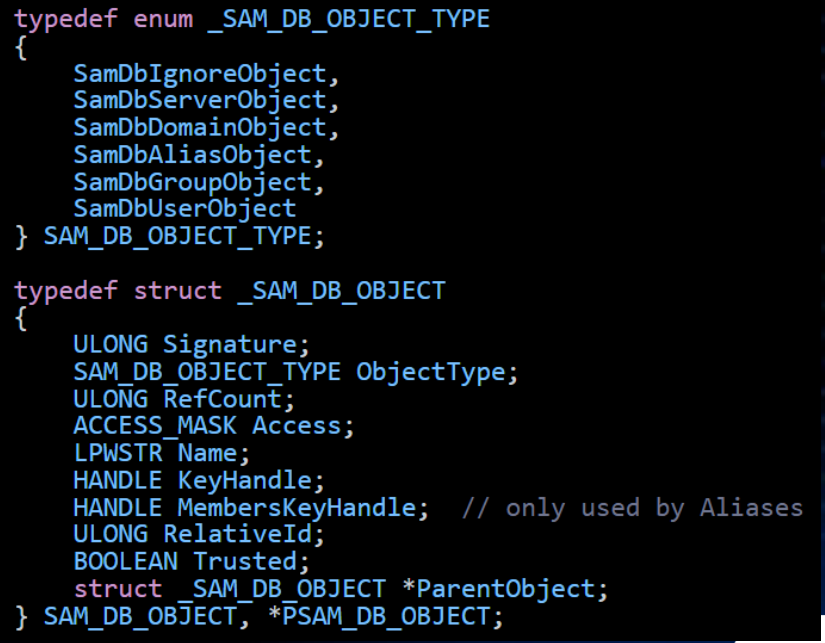
Alias

Fancy Users S-1-5-21-651002831-2290666596-897486024-1002

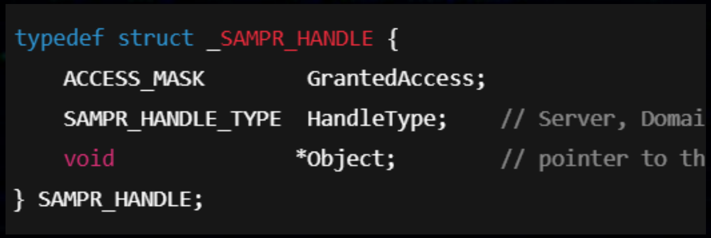
Local Security Authority

SAM Database objects_

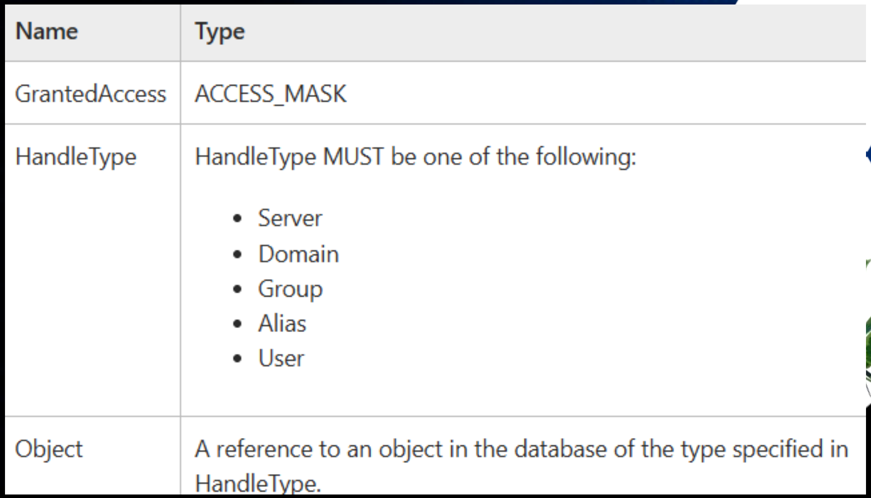
- It’s not kernel objects
- It’s data structure in LSASS memory (_SAM_DB_OBJECT)
- The returned handle to this object is located inside RPC context handles
- It’s used to maintain session state between the client and the server



```
typedef enum _SAM_DB_OBJECT_TYPE
{
    SamDbIgnoreObject,
    SamDbServerObject,
    SamDbDomainObject,
    SamDbAliasObject,
    SamDbGroupObject,
    SamDbUserObject
} SAM_DB_OBJECT_TYPE;

typedef struct _SAM_DB_OBJECT
{
    ULONG Signature;
    SAM_DB_OBJECT_TYPE ObjectType;
    ULONG RefCount;
    ACCESS_MASK Access;
    LPWSTR Name;
    HANDLE KeyHandle;
    HANDLE MembersKeyHandle;  // only used by Aliases
    ULONG RelativeId;
    BOOLEAN Trusted;
    struct _SAM_DB_OBJECT *ParentObject;
} SAM_DB_OBJECT, *PSAM_DB_OBJECT;
```



```
typedef struct _SAMPR_HANDLE {
    ACCESS_MASK        GrantedAccess;
    SAMPR_HANDLE_TYPE  HandleType;    // Server, Domai
    void              *Object;        // pointer to th
} SAMPR_HANDLE;
```



Name	Type
GrantedAccess	ACCESS_MASK
HandleType	HandleType MUST be one of the following: • Server • Domain • Group • Alias • User
Object	A reference to an object in the database of the type specified in HandleType.

Local Security Authority

Security (Policy) Database_

- It exposes four objects:

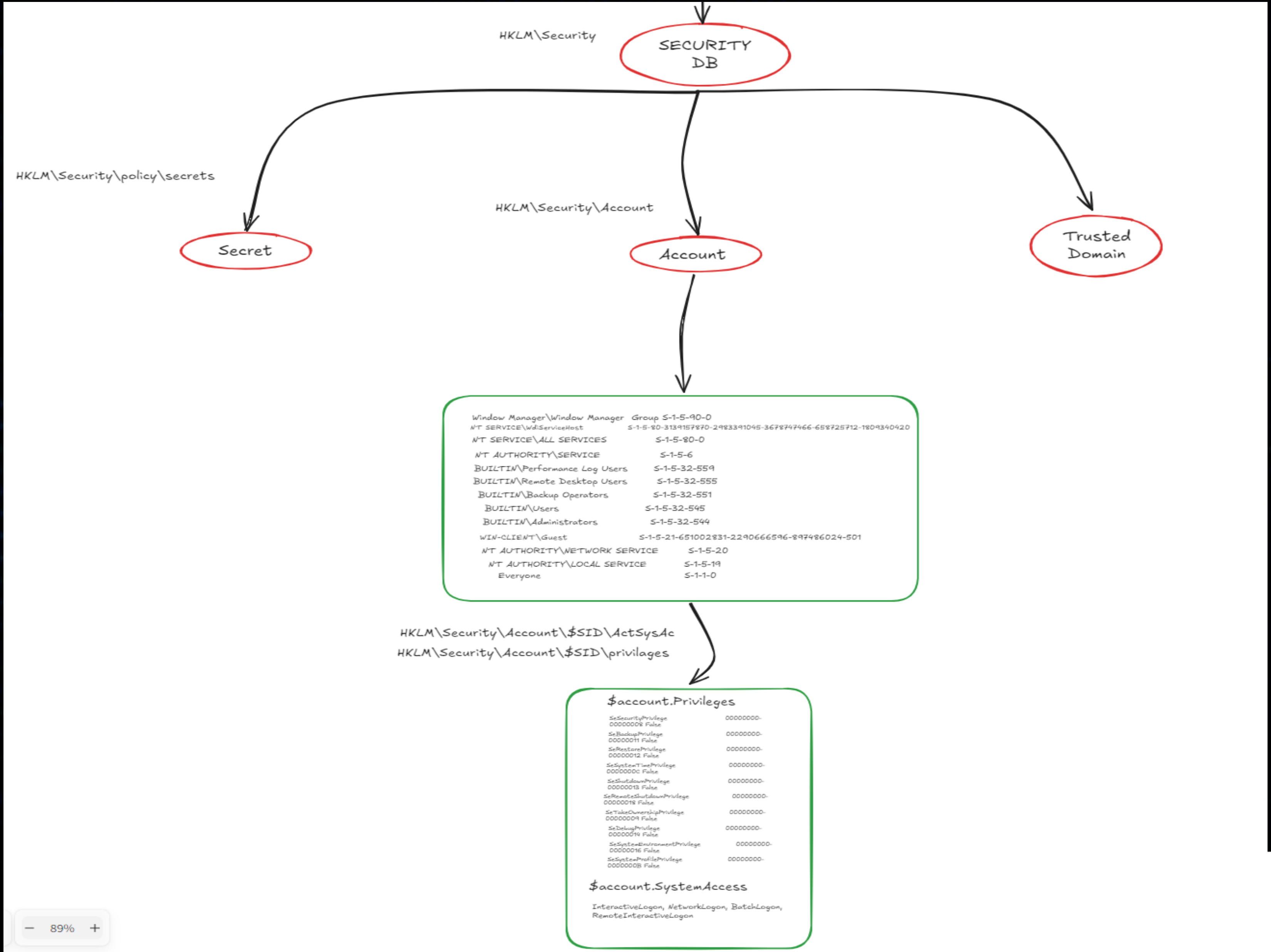
  A. Policy- Root object for system security policy

  B. Trusted-Domain – describes trust relationships between domains in a forest.

  C. Account – stores the user-rights assignments (privileges)

  D. Secret – holds encrypted LSA secrets such as machine-account passwords, cached credentials, and DPAPI keys.

Local Security Authority
SECURITY Database


HKLM\Security
SECURITY DB
HKLM\Security\policy\secrets
Secret
HKLM\Security\Account
Account
Trusted Domain
HKLM\Security\Account\$SID\ActSysAc
HKLM\Security\Account\$SID\privilages
89%

Window Manager\Window Manager Group	S-1-5-90-0
NT SERVICE\WdiServiceHost	S-1-5-80-3139157870-2983391045-3678747466-658725712-1809340420
NT SERVICE\ALL SERVICES	S-1-5-80-0
NT AUTHORITY\SERVICE	S-1-5-6
BUILTIN\Performance Log Users	S-1-5-32-559
BUILTIN\Remote Desktop Users	S-1-5-32-555
BUILTIN\Backup Operators	S-1-5-32-551
BUILTIN\Users	S-1-5-32-545
BUILTIN\Administrators	S-1-5-32-544
WIN-CLIENT\Guest	S-1-5-21-651002831-2290666596-897486024-501
NT AUTHORITY\NETWORK SERVICE	S-1-5-20
NT AUTHORITY\LOCAL SERVICE	S-1-5-19
Everyone	S-1-1-0

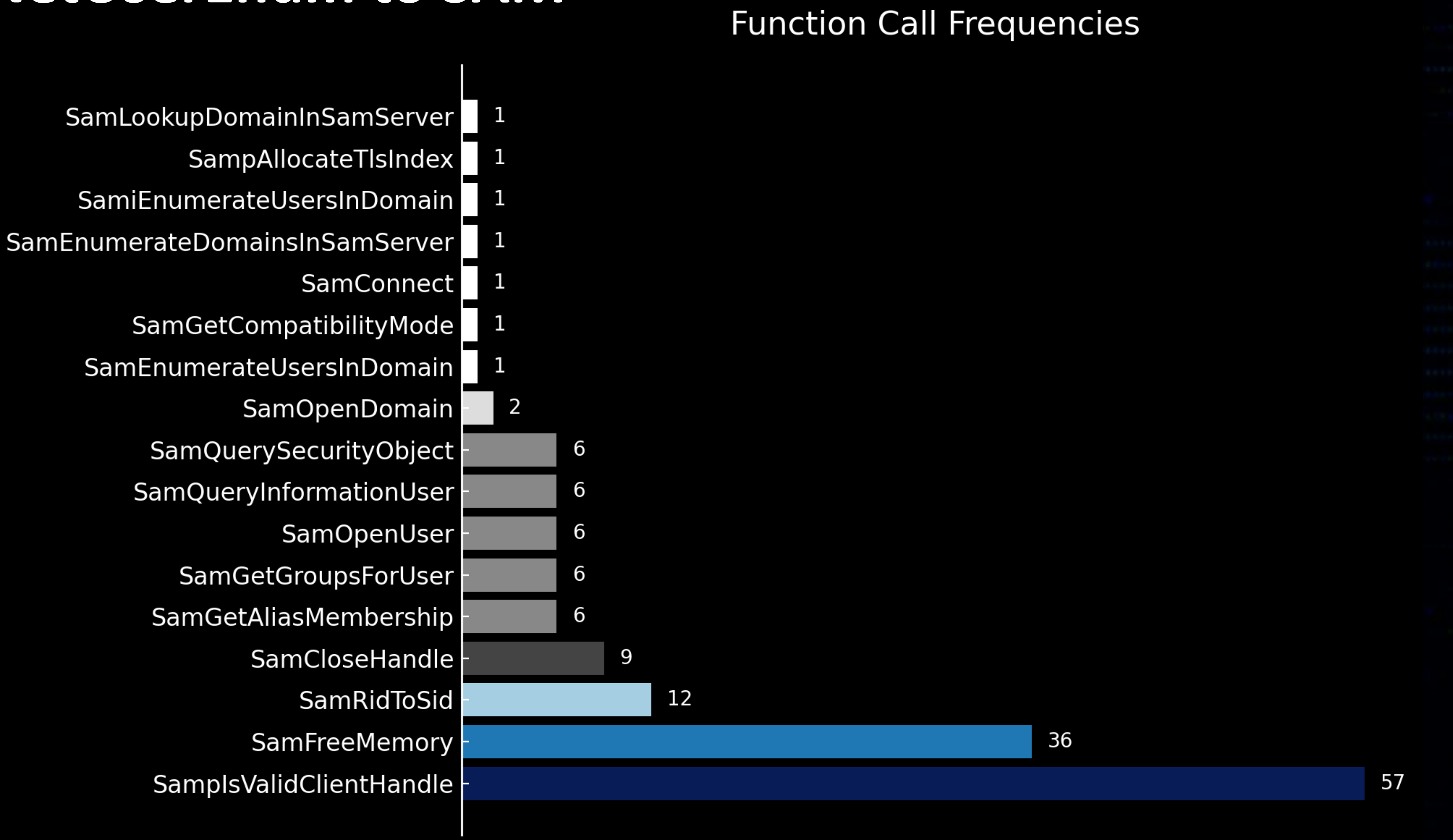
$account.Privileges

```
SeSecurityPrivilege                 00000000-
00000008 False
SeBackupPrivilege                   00000000-
00000011 False
SeRestorePrivilege                  00000000-
00000012 False
SeSystemTimePrivilege               00000000-
0000000C False
SeShutdownPrivilege                 00000000-
00000013 False
SeRemoteShutdownPrivilege           00000000-
00000018 False
SeTakeOwnershipPrivilege            00000000-
00000009 False
SeDebugPrivilege                    00000000-
00000014 False
SeSystemEnvironmentPrivilege        00000000-
00000016 False
SeSystemProfilePrivilege            00000000-
0000000B False
```

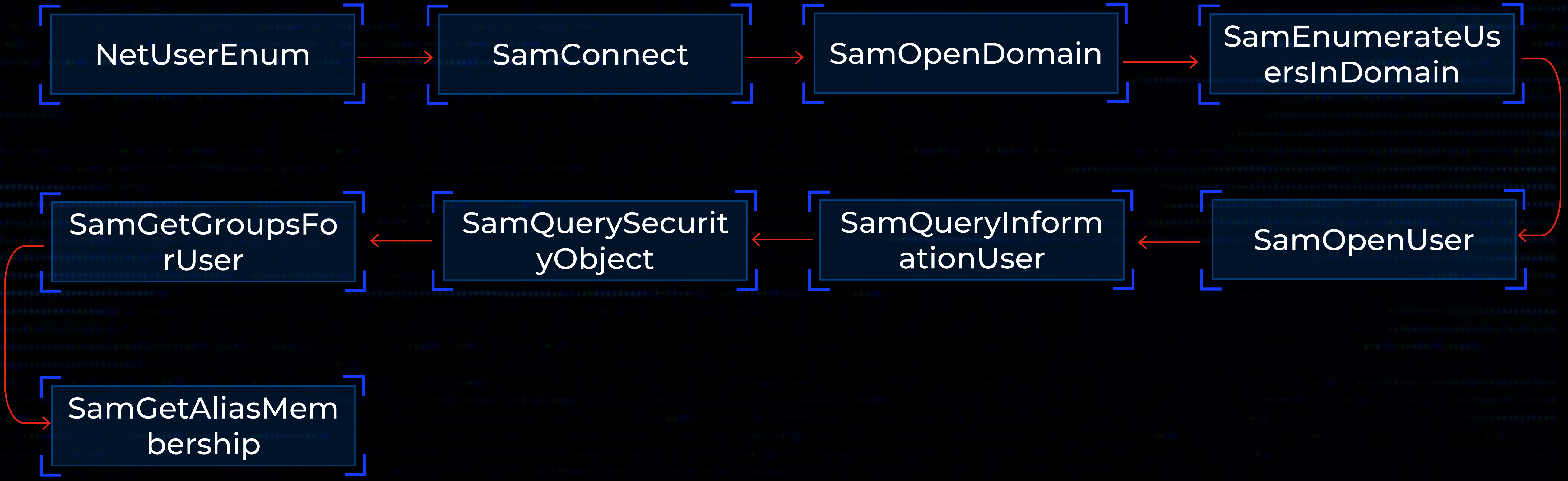
$account.SystemAccess

InteractiveLogon, NetworkLogon, BatchLogon, RemoteInteractiveLogon

Local Security Authority
From NetUserEnum to SAM



Local Security Authority
From NetUserEnum to SAM



NetUserEnum

SamConnect

SamOpenDomain

SamEnumerateUsersInDomain

SamOpenUser

SamQueryInformationUser

SamQuerySecurityObject

SamGetGroupsForUser

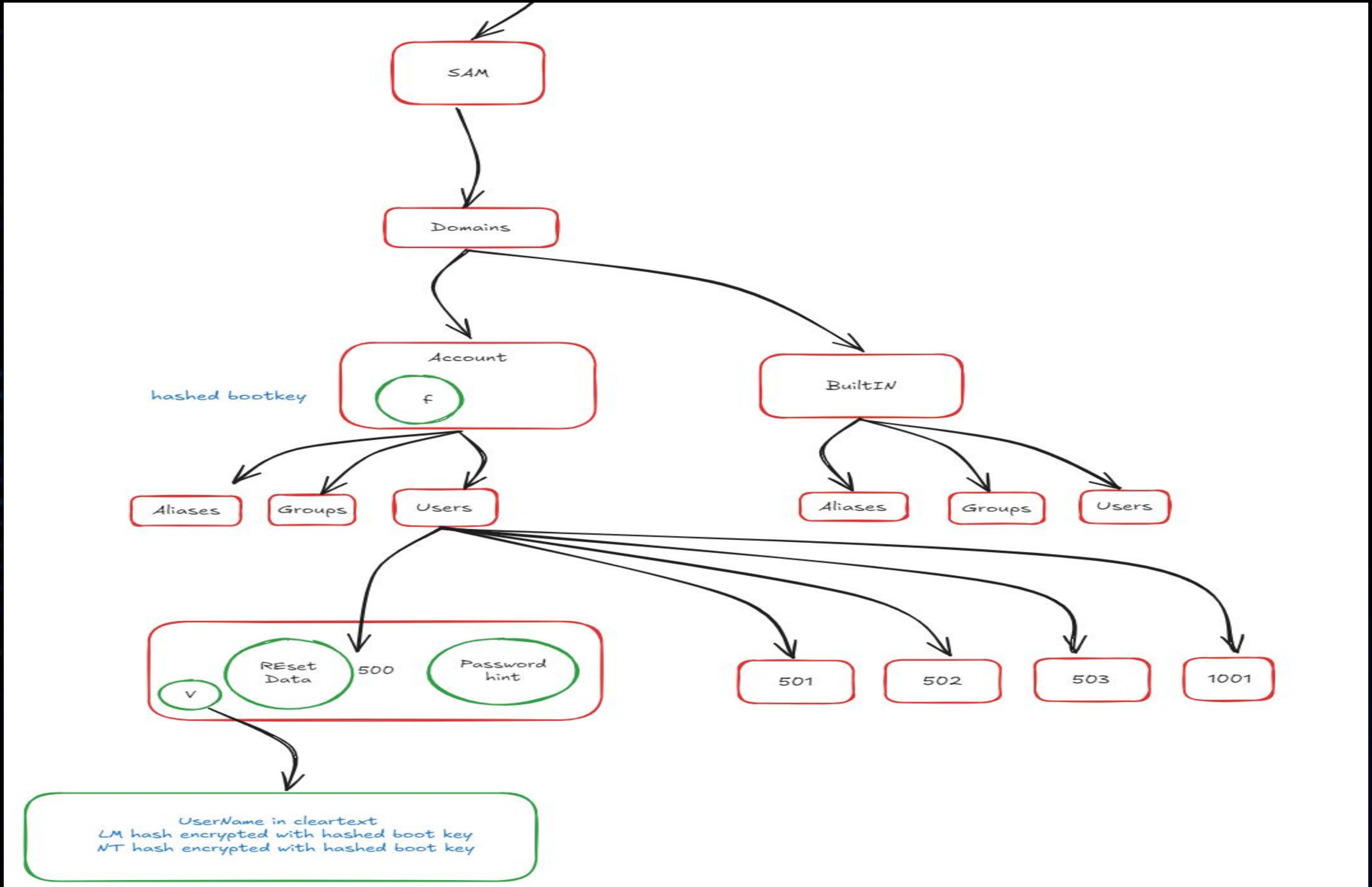
SamGetAliasMembership

Local Security Authority

Windows Secrets Storage_

- Password hashes are stored in the SAM hive
- It’s stored in encrypted form
- Not accessible via SAMR APIs
- To access these values, direct interaction with the registry is required (RegOpenKeyEx).
- Access requires SYSTEM-level privileges
- Even local admins cannot access raw data without elevation

Local Security Authority
SAM Database, Windows Creds



Windows Lateral Movement

Windows Lateral Movement

Credential Collection Methods_

- After initial access, attackers aim to move to other systems (cleartext or PTH)
- Goal: Harvest credentials/secrets for reuse
- Remote Collection: Requires elevated (non-UAC-filtered) token
- Local Collection (on-host): Requires SYSTEM or (non-UAC-filtered) token
- Memory collection: Accessing LSASS collection.

Windows Lateral Movement
Remote Collection

Approach	Technique	Down Sides
Impacket-secretsdump	• RPC to enable Remote Registry • Use functions under RRP to: A. Read bootkey from SYSTEM B. Backup the hives (SAM, Security) RegSaveKey • Download it through SMB • Extract the secrets offline	• Enabling remote registry • Saving hives on the disk
Impacket-secretsdump Inline mode (recommended)	• RPC to enable Remote Registry • Use functions under RRP to: A. Read bootkey from SYSTEM B. Read values inside (SAM, Security) BaseRegOpenKey with SeBackupPrivilege • Extract the secrets	• Enabling remote registry
NetExec ntds-dump-raw	• Execute Powershell script that: A. Read raw data from the physical disk B. Getting handle to device kernel object \Device\Harddisk0\DR0	• Executing Powershell script • Writing raw data on the disk is also not a good idea

Windows Lateral Movement
Local Collection

Approach	Technique	Down Sides
reg save command line	• Backup SYSTEM, SECURITY, SAM to on disk files using RegSaveKey API • Use it for offline extraction	• Writing data on the disk • RegSaveKey api is monitored by EDRs
Native executable with Win32 API <<winreg.h>>	• Using functions like RegOpenKey, RegQueryValue to obtain registry values on the fly • Extract the sercrets	• It needs SYSTEM privileges • Triggering EDRs untrusted process tries to access sensitive values
regedit.exe Export (recommended)	• Run regedit.exe GUI application as administrator • Export the SYSTEM, SECURITY, SAM with text format • Import them in VM • Extract secrets by usual ways	• Need Interactive session

Windows Lateral Movement

EDR callback routines_

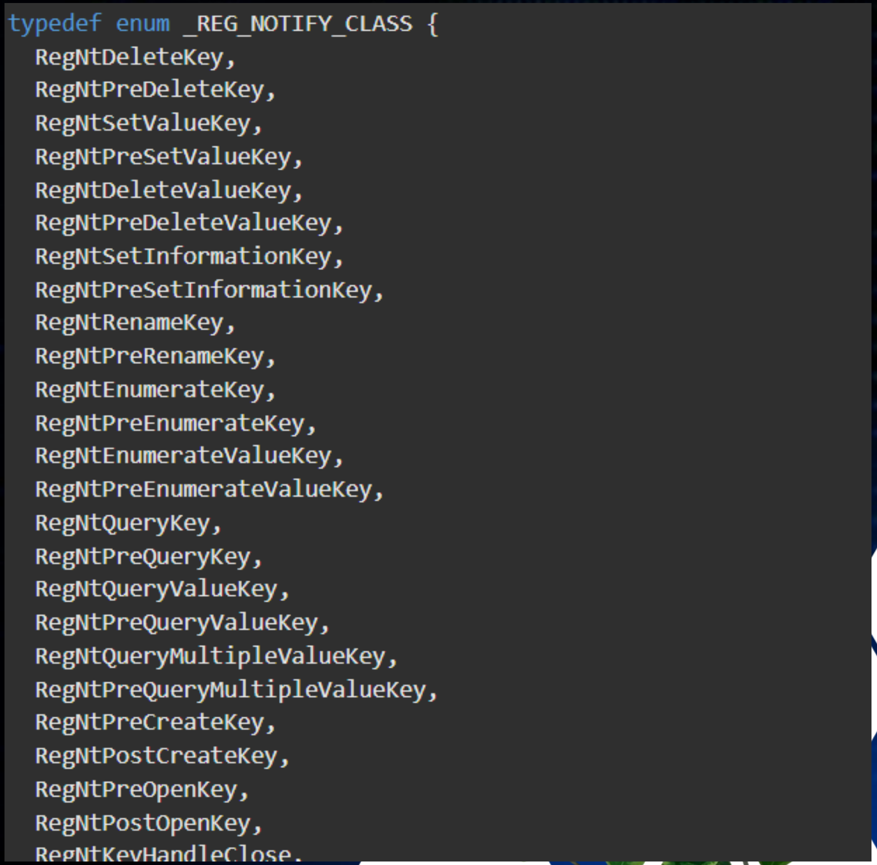
- Modern EDRs use kernel-mode callbacks to track system events
- Events include: Process creation/termination, Image loading, Registry activity
- For registry monitoring, EDR driver uses:

  A. CmRegisterCallbackEx* to register a callback function

  B. Kernel calls the function on registry access
- Callback receives:

  A. REG_NOTIFY_CLASS (event type i.e. RegNtPreEnumerateValueKey)

  B. Event-specific data (key path, access mask, etc.)



```
typedef enum _REG_NOTIFY_CLASS {
  RegNtDeleteKey,
  RegNtPreDeleteKey,
  RegNtSetValueKey,
  RegNtPreSetValueKey,
  RegNtDeleteValueKey,
  RegNtPreDeleteValueKey,
  RegNtSetInformationKey,
  RegNtPreSetInformationKey,
  RegNtRenameKey,
  RegNtPreRenameKey,
  RegNtEnumerateKey,
  RegNtPreEnumerateKey,
  RegNtEnumerateValueKey,
  RegNtPreEnumerateValueKey,
  RegNtQueryKey,
  RegNtPreQueryKey,
  RegNtQueryValueKey,
  RegNtPreQueryValueKey,
  RegNtQueryMultipleValueKey,
  RegNtPreQueryMultipleValueKey,
  RegNtPreCreateKey,
  RegNtPostCreateKey,
  RegNtPreOpenKey,
  RegNtPostOpenKey,
  RegNtKeyHandleClose,
```

Windows Lateral Movement

EDR callback routines_

- Registry generates thousands of ops per minute
- Full monitoring would hurt performance
- EDRs optimize by:

    A. Filtering for certain operations

    B. Monitor only for sensitive keys (e.g., HKLM\SAM, HKLM\SECURITY)

Windows Lateral Movement

EDR Registry Operations Bypass_

- Exploit blind spots:

  A. For example if they are monitoring events under HKLM only, use HKCU or HKU\
  B. Use uncommon APIs for Read/Write data

- Attack the callback routines:

  A. Stop EDR from getting those notifications.
  B. Identify a target driver's callback
  C. Patch it by RETN

Silent Harvester

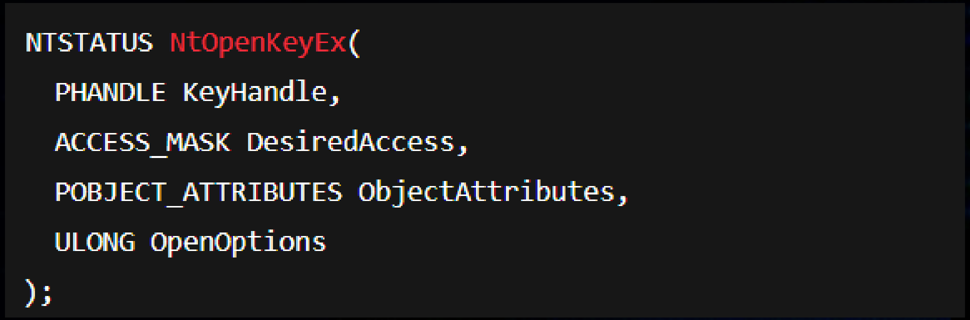
Silent Harvester

Reading secrets on the fly_

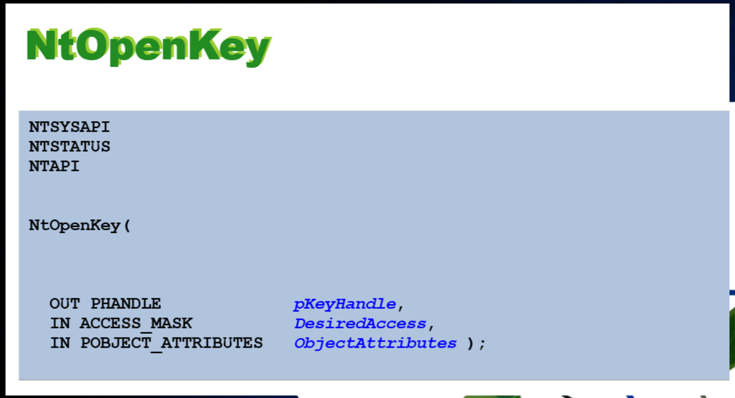
- Goal: Avoid disk writes and Remote Registry
- Tool runs as local administrator, not SYSTEM
- Traditional tools require SYSTEM; no public method existed
- Inspiration: James Forshaw's NtObjectManager:

  A. Mounts registry as PowerShell drive

  B. Reads protected keys using native APIs



```
NTSTATUS NtOpenKeyEx(
  PHANDLE KeyHandle,
  ACCESS_MASK DesiredAccess,
  POBJECT_ATTRIBUTES ObjectAttributes,
  ULONG OpenOptions
);
```



NtOpenKey

```
NTSYSAPI
NTSTATUS
NTAPI

NtOpenKey(


  OUT PHANDLE              pKeyHandle,
  IN ACCESS_MASK           DesiredAccess,
  IN POBJECT_ATTRIBUTES    ObjectAttributes );
```

Silent Harvester

Reading secrets on the fly_

- Uses undocumented API: NtOpenKeyEx
- Critical flags in OpenOptions: REG_OPTION_OPEN_LINK (0x08), REG_OPTION_BACKUP_RESTORE (0x04)
- With SeBackupPrivilege, access bypasses ACL checks
- Users inside administrator group can read SAM & SECURITY directly from memory
- Use NtQueryValueKey / RegQueryValueExW
- No disk backup created

Silent Harvester

Be under the Radar_

- Admin with SeBackupPrivilege can access protected keys on the fly
- Problem: EDRs detect access via common APIs
- Example: RegQueryValueExW + high-risk paths (e.g., SAM, SECURITY)
- EDRs monitor a limited set of high-volume API calls
- Goal: Use a less common API that avoids detection

Silent Harvester

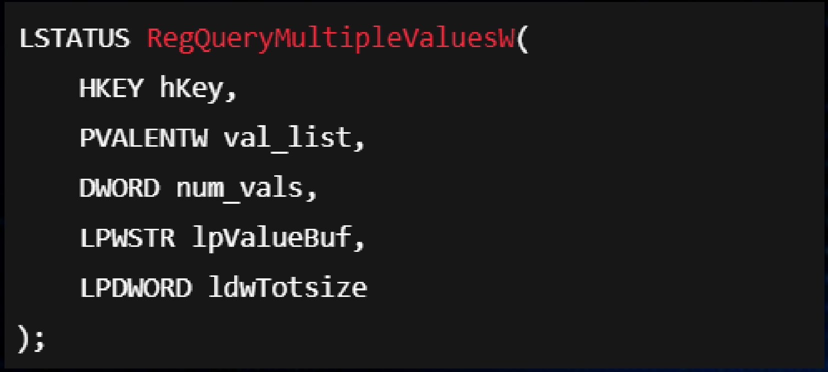
Be under the Radar_

- Rarely used API: RegQueryMultipleValuesW
- Retrieves multiple values from an open registry key
- Key points:

  A. Uses handle from NtOpenKeyEx

  B. Reads values into a caller-supplied buffer

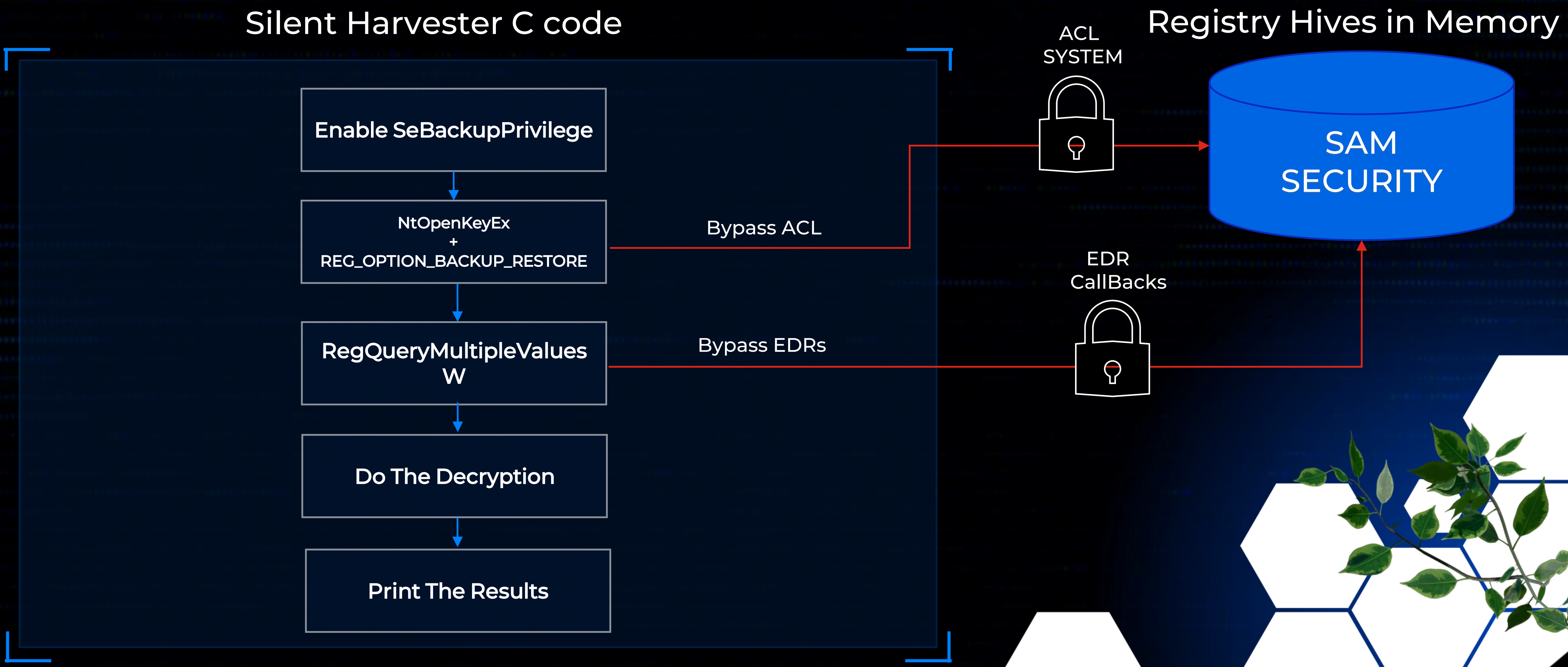
  C. Triggers no alerts across tested EDR platforms



```
LSTATUS RegQueryMultipleValuesW(
    HKEY hKey,
    PVALENTW val_list,
    DWORD num_vals,
    LPWSTR lpValueBuf,
    LPDWORD ldwTotsize
);
```

Silent Harvester

Full Stealth Strategy_



Enhancing Red Team Techniques

Enhancing Red Team Techniques

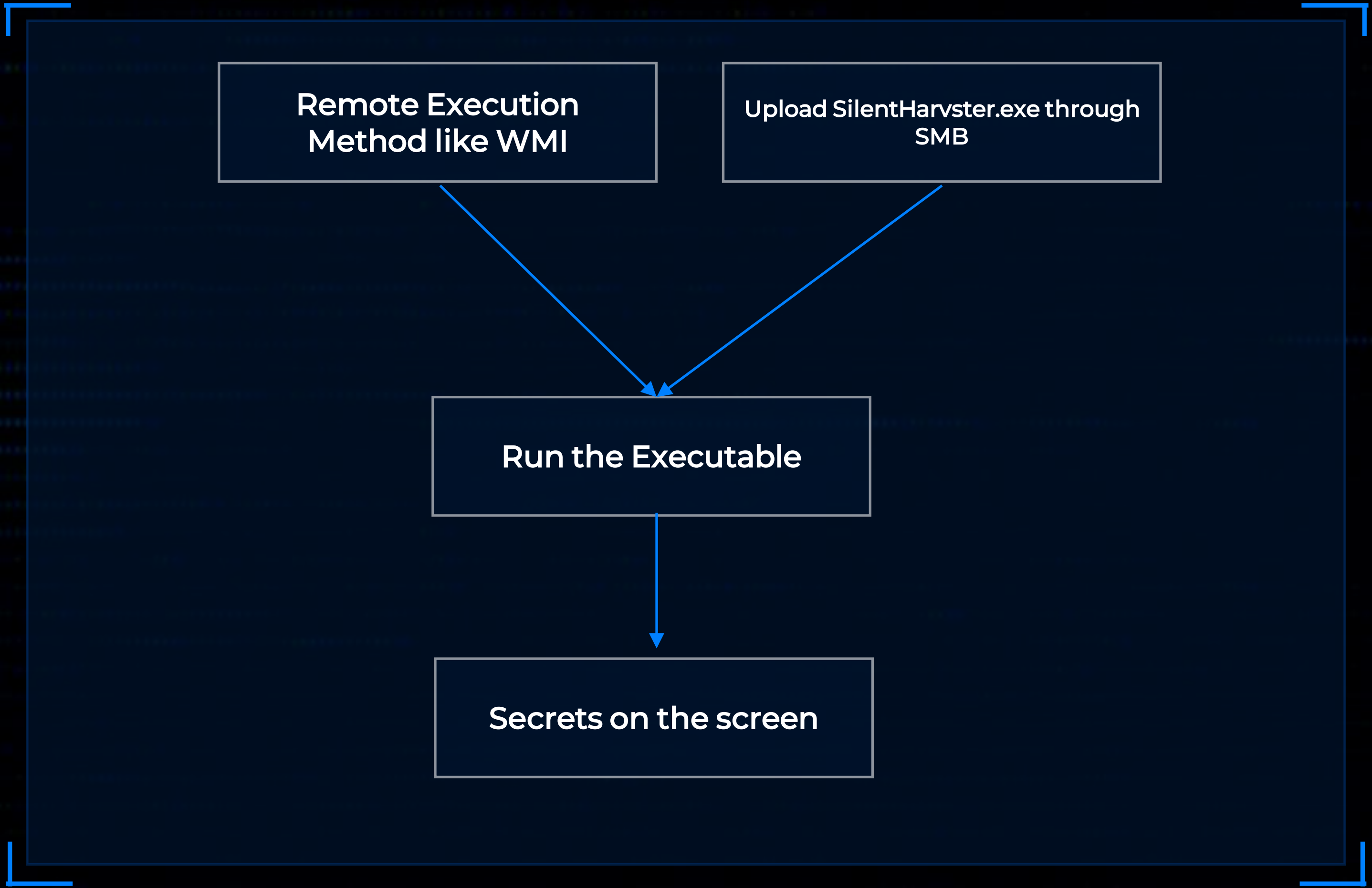
Current Approaches Downsides _

- Enables Remote Registry
- Saves data to disk
- Accessing sensitive data by untrusted processes under monitored APIs
- SYSTEM-level privilege

Enhancing Red Team Techniques

Example of Full Vector No 1_

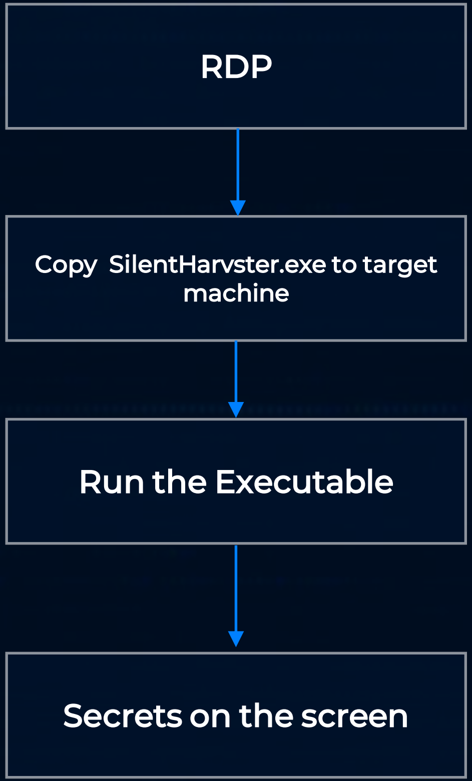
User with non filtered token


Remote Execution Method like WMI
Upload SilentHarvster.exe through SMB
Run the Executable
Secrets on the screen

Enhancing Red Team Techniques

Example of Full Vector No 2_

User with RDP access with administrative privileges



RDP

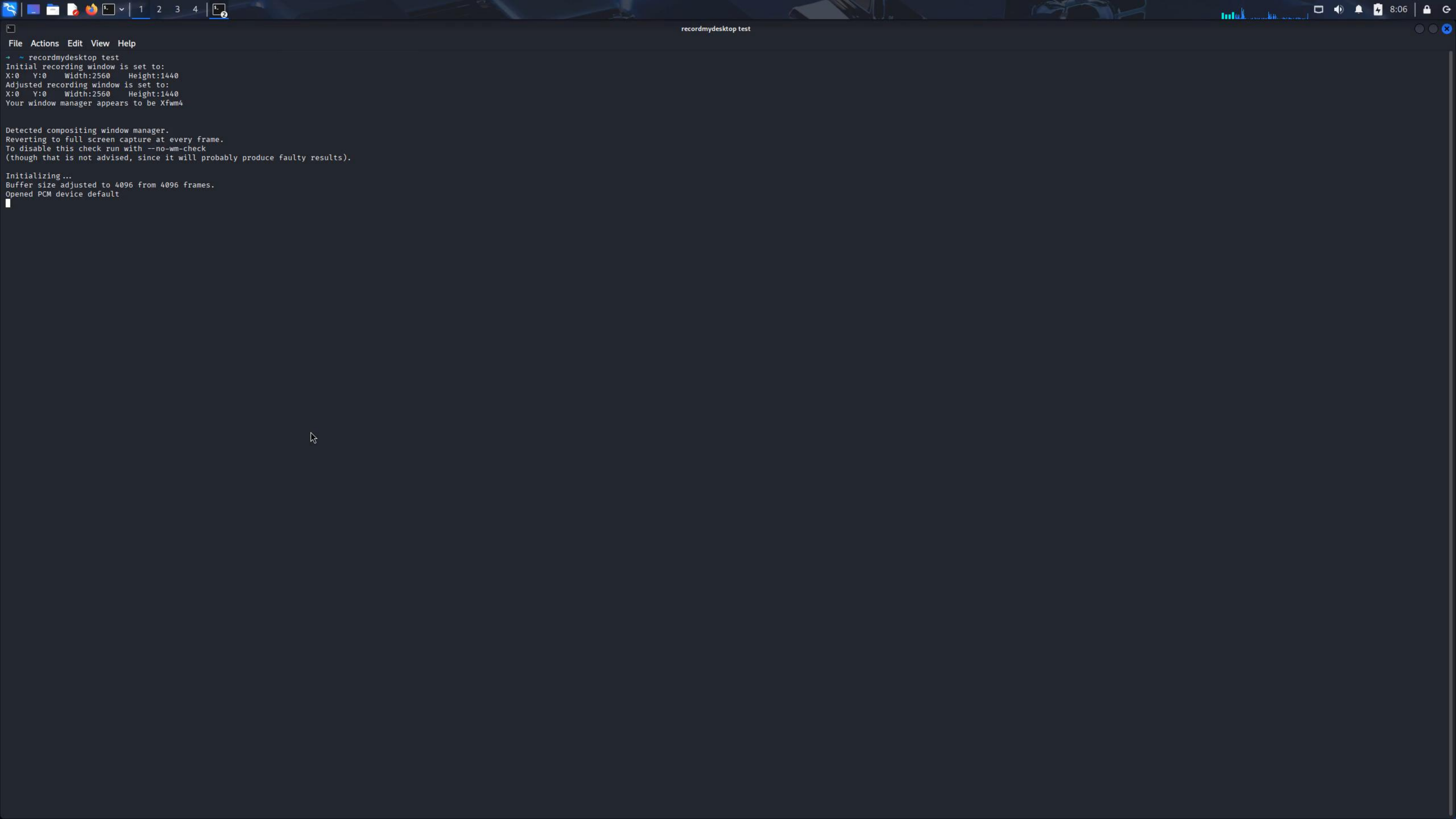
Copy SilentHarvster.exe to target machine

Run the Executable

Secrets on the screen

Enhancing Red Team Techniques

Demo_



```
→ ~ recordmydesktop test
Initial recording window is set to:
X:0   Y:0    Width:2560    Height:1440
Adjusted recording window is set to:
X:0   Y:0    Width:2560    Height:1440
Your window manager appears to be Xfwm4


Detected compositing window manager.
Reverting to full screen capture at every frame.
To disable this check run with --no-wm-check
(though that is not advised, since it will probably produce faulty results).

Initializing...
Buffer size adjusted to 4096 from 4096 frames.
Opened PCM device default
```

Conclusion

- Start with simple methods
- If detected, experiment with less common APIs to read sensitive values
- As a last resort, consider running tools via trusted processes (e.g., python.exe)
- Always consider monitoring less common APIs

References

- Google project zero Registry Adventure Series:
  https://googleprojectzero.blogspot.com/2024/04/the-windows-registry-adventure-1.html
- Windows Security internals, James Forshaw:
  https://www.oreilly.com/library/view/windows-security-internals/9781098168834/
- NtObjectManager:
  https://www.powershellgallery.com/packages/NtObjectManager/1.1.32
- https://undocumented.ntinternals.net/
- impacket
  https://github.com/fortra/impacket
- netexec
  https://github.com/Pennyw0rth/NetExec
- Evading EDR, Matt Hand
  https://www.oreilly.com/library/view/evading-edr/9781098168742/

OFFZONE
2025

X: @haider_kabibo

Q&A