Certificates for Privilege Escalation | Redfox Security

redfoxsec.com/blog/unleashing-the-potential-of-certificates-for-privilege-escalation

Kunal Kumar April 1, 2024

Unleashing The Potential Of Certificates For Privilege Escalation

- April 1, 2024
- Active Directory
- Kunal Kumar

Maintaining proactive defences against emerging cyber threats is more critical than ever in today's evolving landscape. One area gaining increasing attention is **privilege escalation**—the process of obtaining higher-level access within systems or networks. Among the techniques that exploit certificate-based mechanisms, **CertPotato** stands out for its innovative use of certificates to elevate privileges up to **NT AUTHORITY\SYSTEM**.

Buckle up as we dive deep into how this attack chain works, how certificates and templates play a role, and what defenders can do to stay ahead.

In this blog, we will explore how digital certificates—traditionally used for security—can be exploited for privilege escalation, focusing on the **CertPotato** technique and its use of **Active Directory Certificate Services (ADCS).**

Understanding ADCS: A Key To Privilege Escalation

Exploring Public Key Infrastructure (PKI)

Before delving into CertPotato, let's recap what **Public Key Infrastructure (PKI)** is. PKI manages the creation, distribution, and revocation of digital certificates and their associated public/private key pairs. In Windows environments, this is implemented through **Active Directory Certificate Services (ADCS)**—introduced with Windows Server 2000.

ADCS integrates seamlessly with Microsoft's ecosystem, supporting **TLS encryption**, **binary signing**, **user authentication**, and **file encryption**, making it a critical backbone of secure enterprise operations.

Unveiling the Power of Certificate Templates

To streamline certificate issuance, **certificate templates** define policies, permissions, and parameters for certificate creation. They determine aspects such as:

- Validity period
- Enrollment permissions (who can request the certificate)
- Extended Key Usage (EKU), defining the certificate's purpose

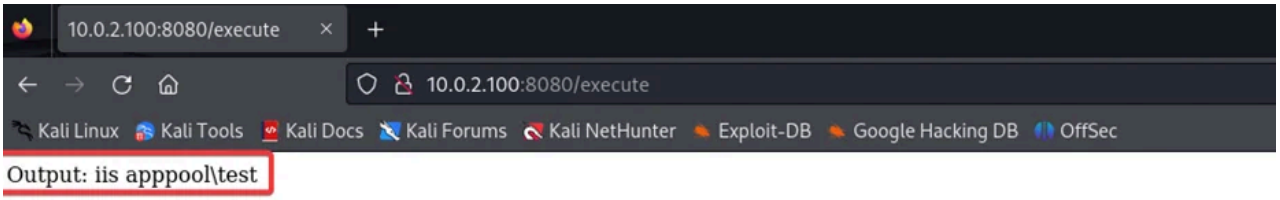
For example, the predefined **Machine** template can be requested by any machine account in the **Domain Computers** group. Misconfigurations in these templates often become the foundation for privilege escalation attacks.

Setting the Stage: The Test Environment

To illustrate the **CertPotato attack chain**, let's build a test setup:

- **DC (10.0.2.10):** Domain Controller hosting the Certificate Authority (CA)
- **CA (10.0.2.100):** Application server running IIS
- **Kali Linux (10.0.2.15):** Attacker machine

After successfully uploading a web shell onto the IIS server, executing `whoami` reveals the service context: `iis apppool\test`—a Microsoft virtual account.



Understanding Service Accounts

A **service account** is a dedicated identity used by Windows services. Some built-in examples include:

1. **LocalSystem (NT AUTHORITY\SYSTEM)**
2. **NetworkService (NT AUTHORITY\Network Service)**
3. **LocalService (NT AUTHORITY\Local Service)**

Only **LocalSystem** and **NetworkService** accounts use the machine account for network authentication.

Windows Server 2008 R2 expanded these options with:

1. **Standalone Managed Service Accounts (sMSA)**
2. **Group Managed Service Accounts (gMSA)**
3. **Virtual Accounts**, used by services like IIS, Exchange, and MSSQL

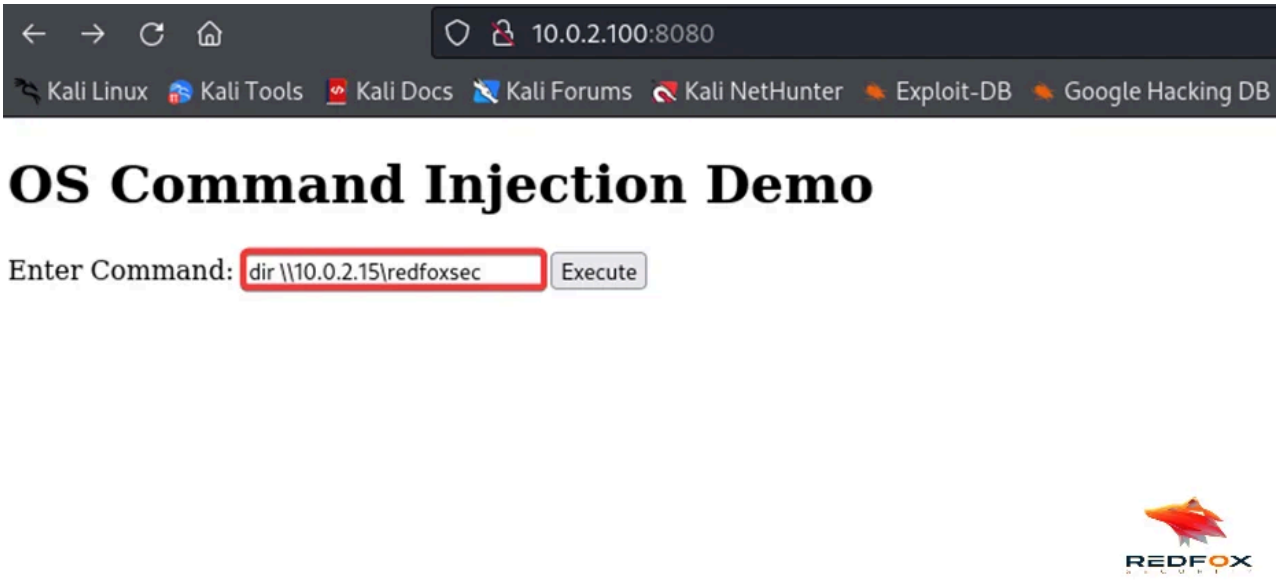
These accounts simplify credential management through Active Directory but also introduce new attack surfaces when misconfigured.

Exploiting The Situation: CertPotato Attack Chain

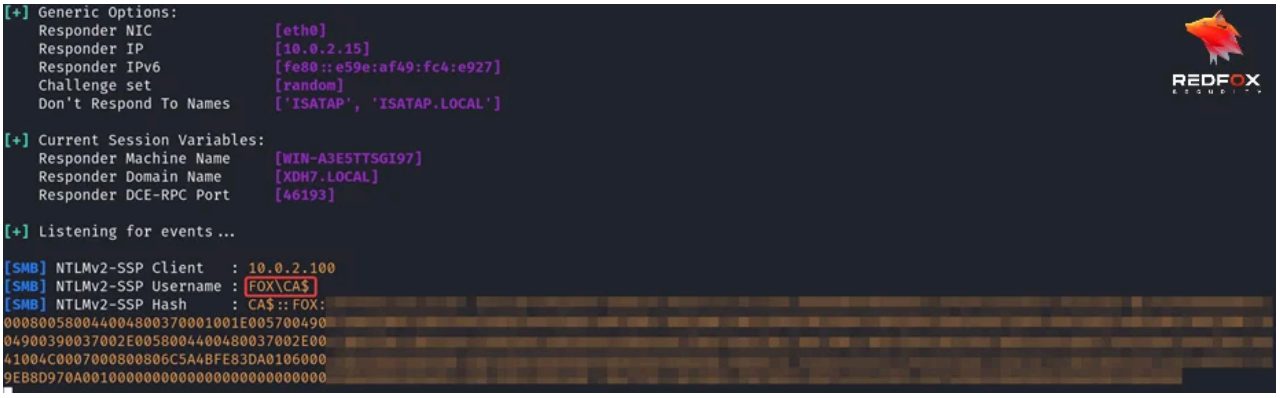
With our web shell operating as `iis apppool\test`, we can now initiate the **CertPotato** privilege escalation process—an alternative to the **Delegate 2 Thyself** RBCD-based method.

Enumerating Remote Shares

To kickstart the attack chain, let's enumerate a remote share from our web shell. Surprisingly, we discover that the default pool account does not attempt to authenticate but the CA$ machine account. This means that when requesting remote information, the operating system falls back to the machine account (CA$) to perform the authentication, which is a valid account within the Active Directory.




```
responder –I eth0
```



From the above response, we can see that the authentication to our responder server is not coming from a service account but from a CA$ machine account.

Coming to the scenario, we can abuse the service account using the tgtdeleg trick to obtain a useful TGT to request a certificate as the machine account.

Before that, we have to understand what TGT delegation is. So, Accounts with Service Principal Name (SPN) records, whether machine or otherwise, that have been granted unconstrained delegation rights can impersonate users to authenticate to any service on any host.

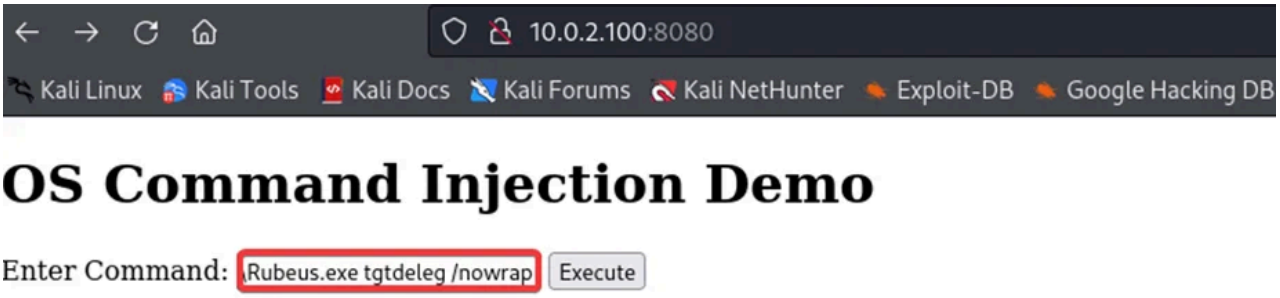
This is possible because when a user with these rights attempts to access a service or server, the AP_REQ packet sent contains an Authenticator encrypted with the session key received in the first Ticket Granting Service Response (TGS-REP) along with the Service Ticket.

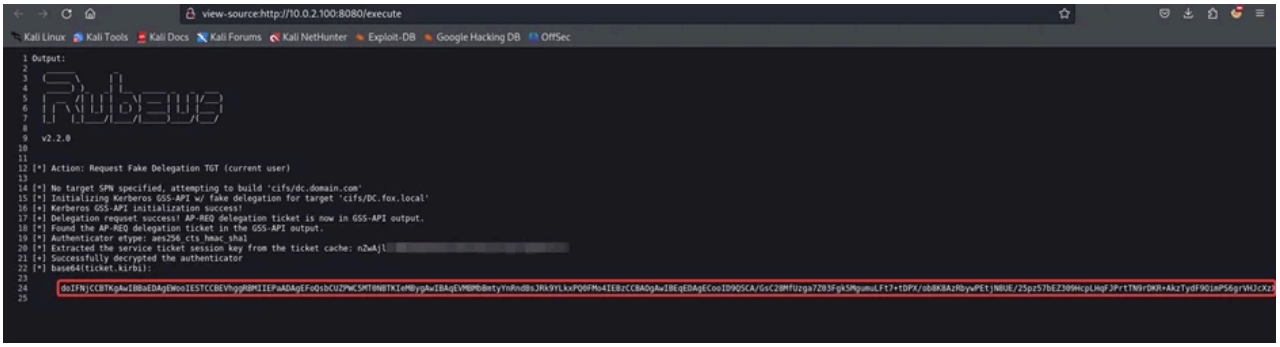
Among the elements within this authenticator is a delegation Ticket Granting Ticket (TGT) with the key linked to the user seeking access with unconstrained delegation.

Therefore, if one manages to obtain the AP_REQ packet and the key used for encrypting the authenticator, one can retrieve the delegation TGT of the user and its associated session key, enabling one to perform unauthorized actions under the guise of that user.

Uploading Rubeus.exe to the target system and obtaining delegation TGT.

```
.\Rubeus.exe tgtdeleg /nowrap
```





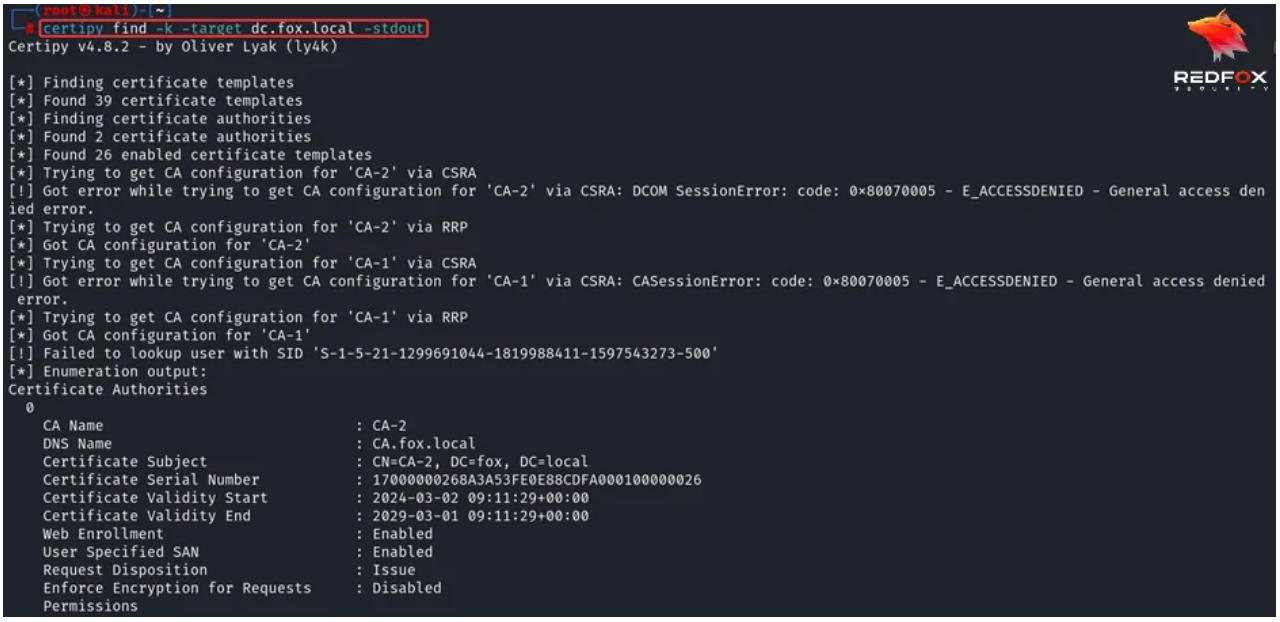
After obtaining the TGT, decoded it using base64 and saved it to a new file ticket.kirbi.

Using Impacket's ticketConverter converted the kirbi file into a ccache file format and exported it into memory to establish the session as CA$ machine account without credentials.

Using Impacket's ticketConverter converted the kirbi file into a ccache file format and exported it into memory to establish the session as CA$ machine account without credentials.

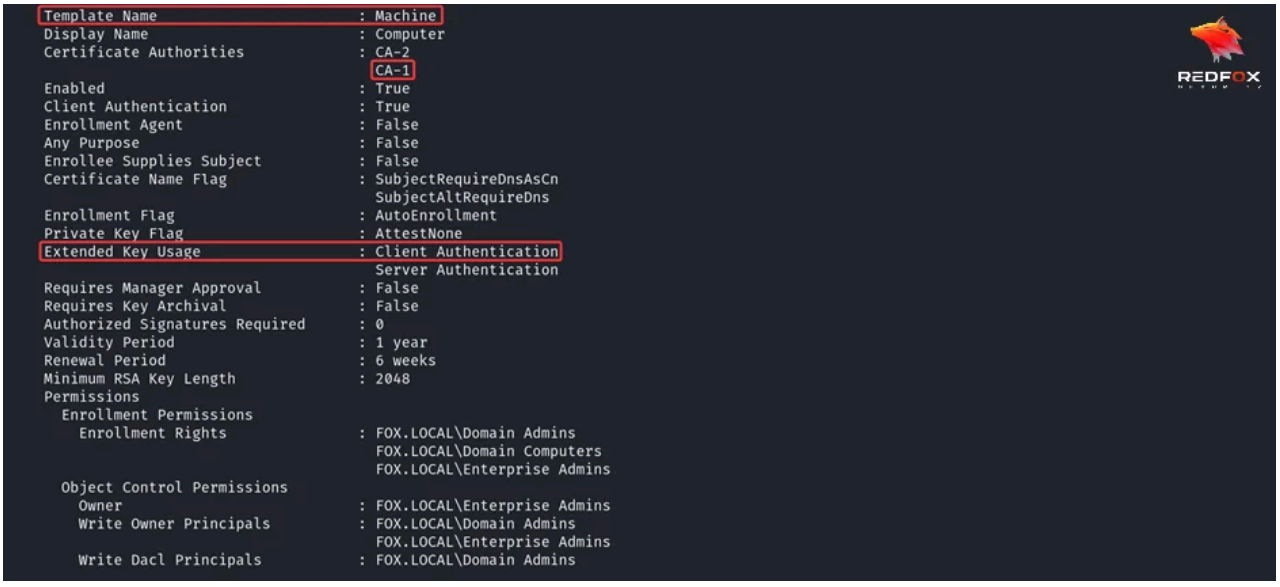
Retrieving Certificate Authority Information

Expanding our survey, we can also retrieve information about the Certificate Authority (CA). This allows us to gather insights into the CA's configuration, including the templates it supports and the certificates it has issued.



Requesting a Machine Certificate

Now, let's use the Certipy tool to request a certificate using our machine account's Ticket-Granting Ticket (TGT). With the TGT loaded, we can enumerate the available certificate templates and choose the "Machine" template. Alternatively, any certificate template with the EKU "Client Authentication" that our machine account can enroll in would suffice

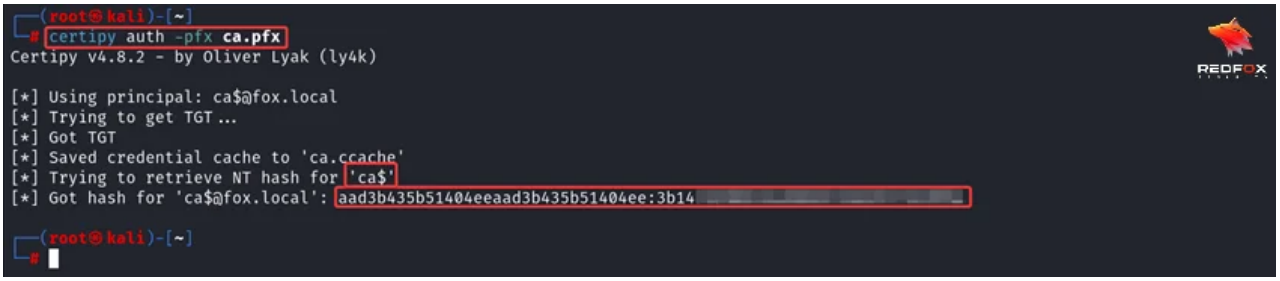


Using certipy for requesting CA$ machine certificate.

Now, using the CA$ certificate, we can use PKINIT authentication technique to obtain the CA$ account hash.
To escalate our privileges further, we can exploit PKINIT authentication, a feature of Kerberos that supports asymmetric authentication. PKINIT allows us to sign the timestamp during pre-authentication with the private key associated with a valid certificate instead of using a password derivative.

However, PKINIT authentication requires the obtained certificate to have specific Extended Key Usages (EKUs), such as “Client Authentication” or “PKINIT Client Authentication. Read more about PKINIT here.

Leveraging PKINIT authentication and the U2U extension, we can obtain the machine account’s hash. This step marks an essential milestone in our quest for privilege escalation.

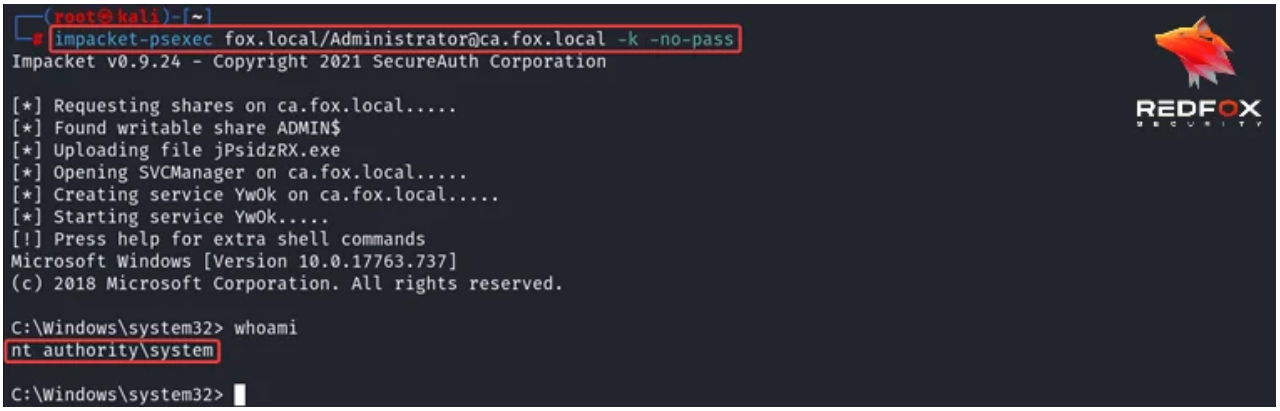


After extracting the Machine account hash, we can forge a silver ticket on the cifs service using Impacket's ticketer, for that we need DomainSID and FQDN of domain.
Using rpcclient and crackmapexec to gather DomainSID and FQDN of domain.

After grabbing the required domain information, we can now forge a silver ticket using Impacket's ticketer, using a machine hash of CA$.

Upgrading to NT AUTHORITY SYSTEM

After getting the silver ticket as Administrator User on CA machine, exported it into memory and now can use psexec to connect to it as NT Authority SYSTEM privileges on CA machine.



TL;DR

The **CertPotato attack chain** highlights how misconfigurations in **ADCS** and **certificate templates** can lead to full system compromise. By chaining **ADCS abuse** with **PKINIT authentication**, attackers can escalate from a simple service account to **NT AUTHORITY\SYSTEM**.

To mitigate this:

- Harden **service accounts** and limit delegation rights.

- Use **gMSA/sMSA** accounts for sensitive services.
- Regularly audit **certificate templates** and **CA permissions**.

Staying vigilant against certificate-based privilege escalation is essential in modern enterprise environments. Understanding these mechanisms empowers defenders to proactively patch weak spots before adversaries exploit them.

At **Redfox Cybersecurity**, our team of offensive security experts helps organizations identify and mitigate such vulnerabilities through **comprehensive penetration testing**.

With years of experience and a research-driven approach, we deliver deep technical insight, hands-on testing, and tailored remediation strategies to protect your digital assets.

Contact us to schedule a consultation or explore our **advanced cybersecurity courses** to upskill your team.

Previous Zero Day In Xbox Privilege Escalation Using Gaming ServiceEoP
Next Master Wi-Fi Connectivity With NodeMCU: Unleashing The Power Of Wi-Fi Wizardry

Recent Blog

September 09, 2025
Is APK Decompilation Legal? What You Need To Know
September 06, 2025
When Hackers Hit the Road: The Jaguar Land Rover Cyberattack
September 05, 2025
This Is the Hacker's Swiss Army Knife. Have You Heard About It?