black hat USA 2022

Taking Kerberos To The Next Level

James Forshaw | @tiraniddo

- Researcher @ Google Project Zero
- Specialize in Windows
  - Local Privilege Escalation
  - RPC/COM Internals
  - Token manipulation
- NtApiDotNet | D2J | OleViewDotNet

“Never met a logical vulnerability I didn’t like.”

Nick Landers | @monoxgas

- Adversarial R&D @ NetSPI
- Also specialize in Windows
  - Offensive tooling suites
  - Payload architectures
  - Vulnerability research
- sRDI | Dark Side Ops

“Your Prod is our Dev.”

Assumptions

You understand the basics of Kerberos

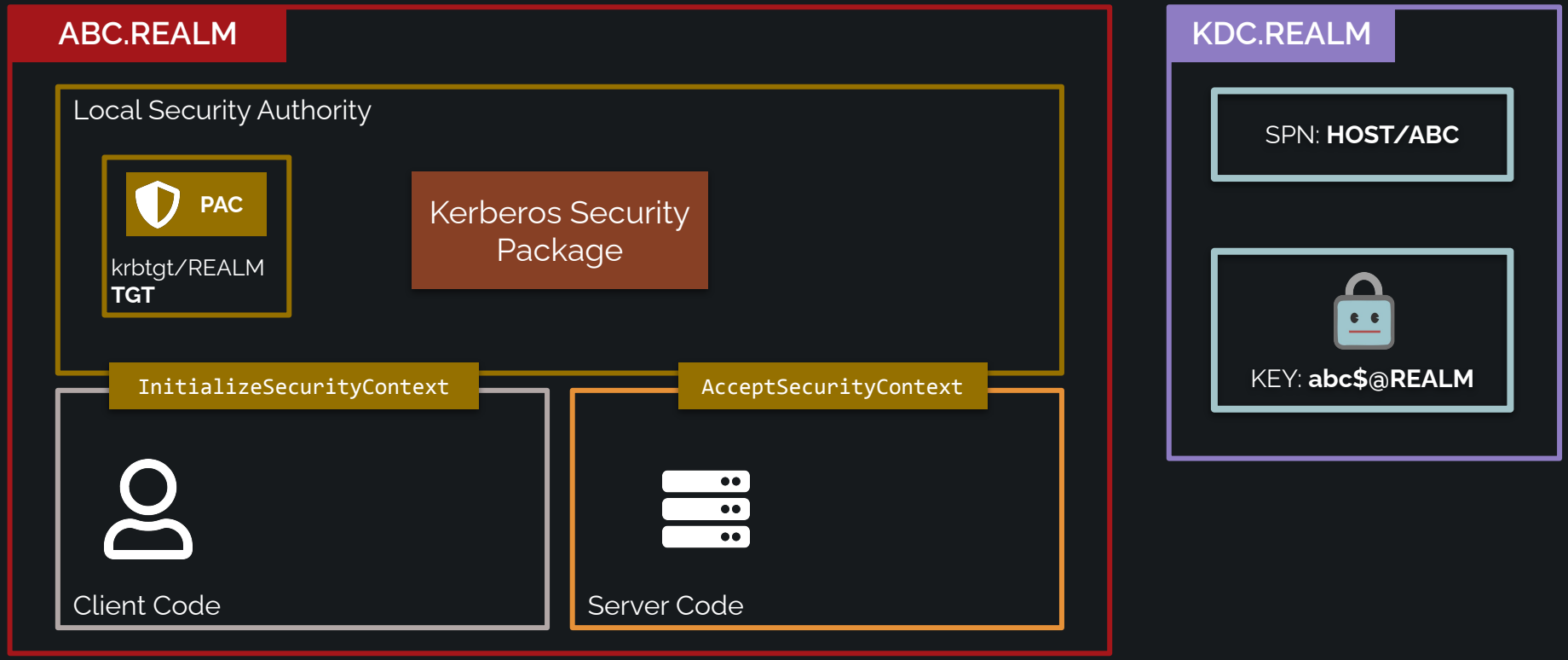
You're (somewhat) familiar with existing remote attacks

You want to see some local privilege escalation (LPE)

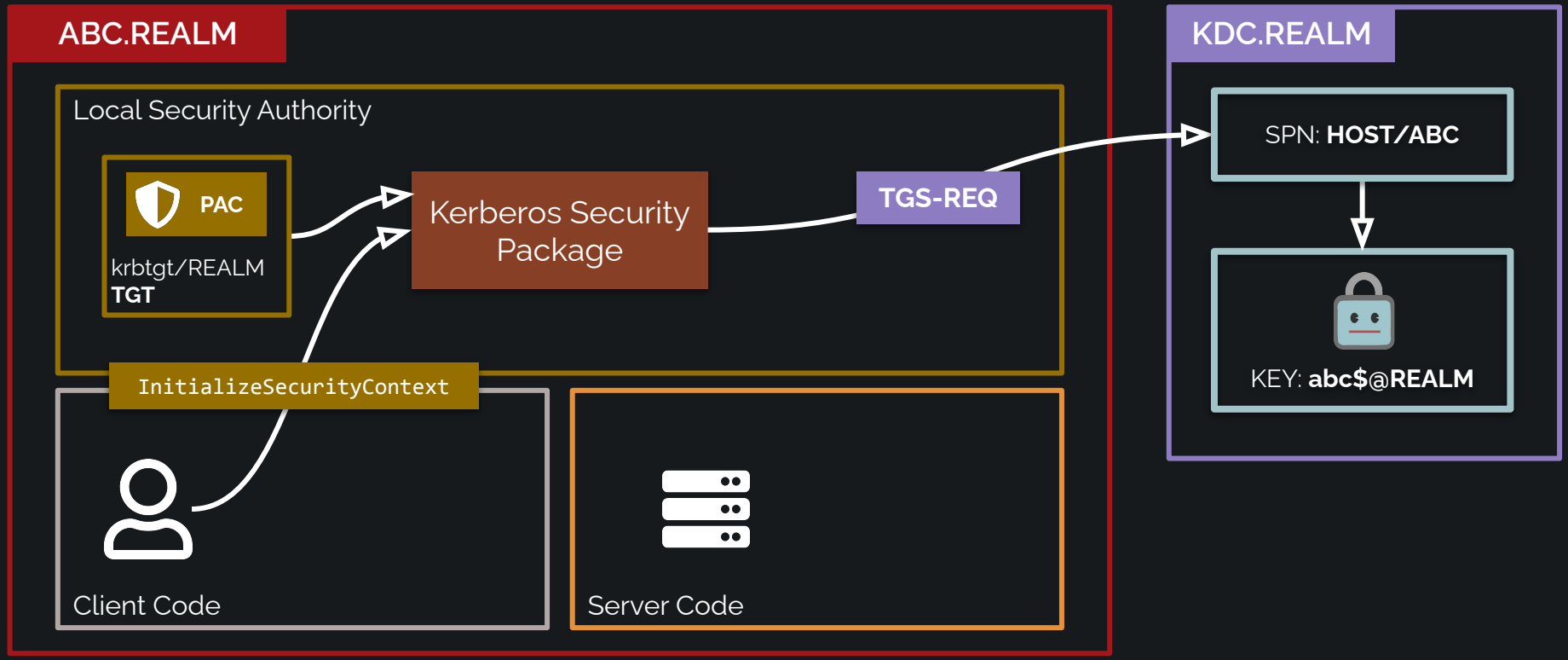
Talking to Yourself

can be good for you

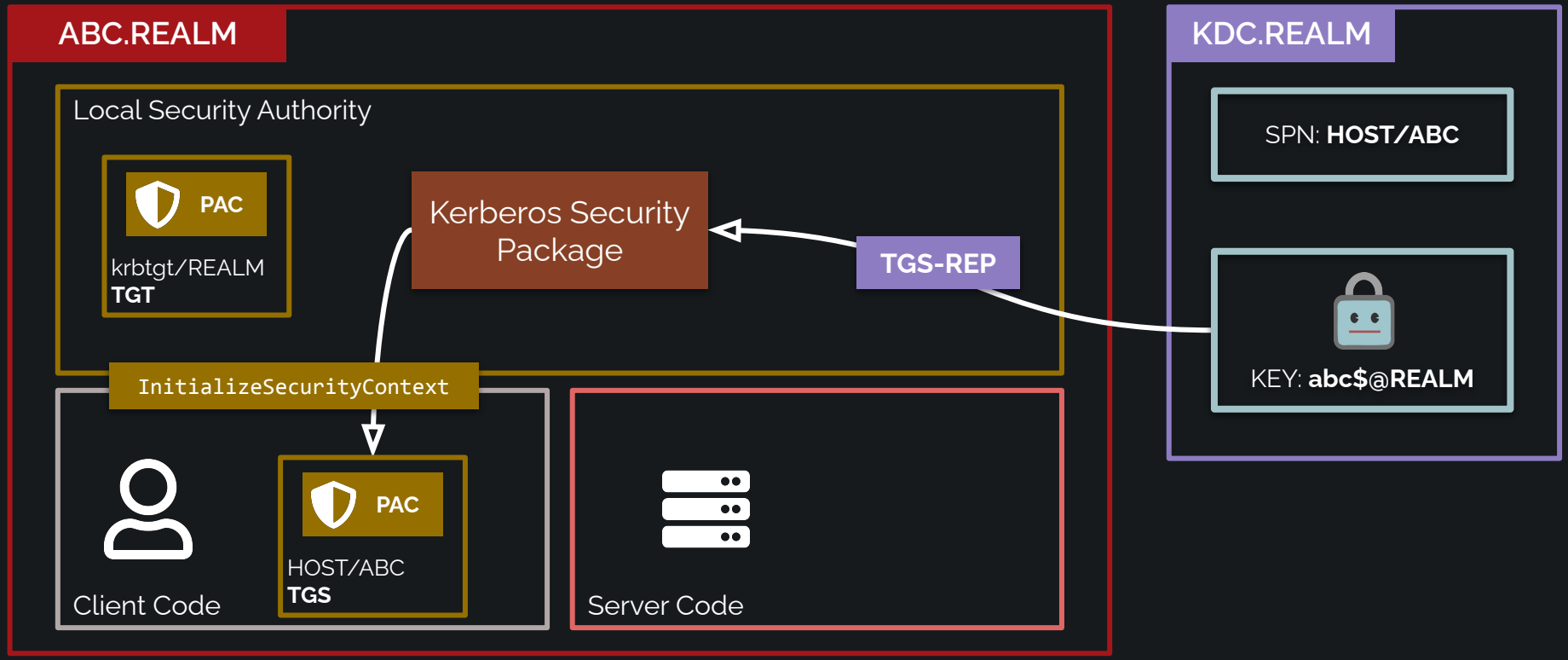
Local Kerberos Authentication



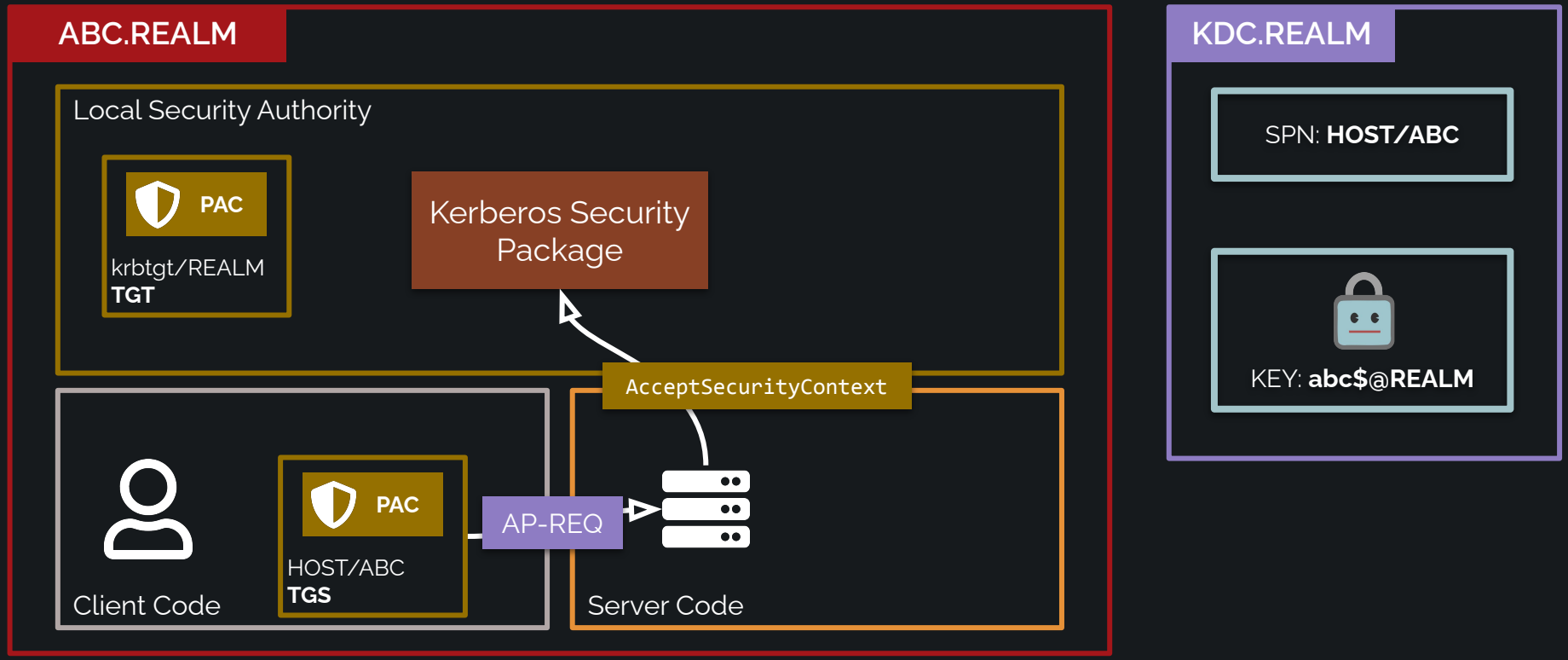
Local Kerberos Authentication



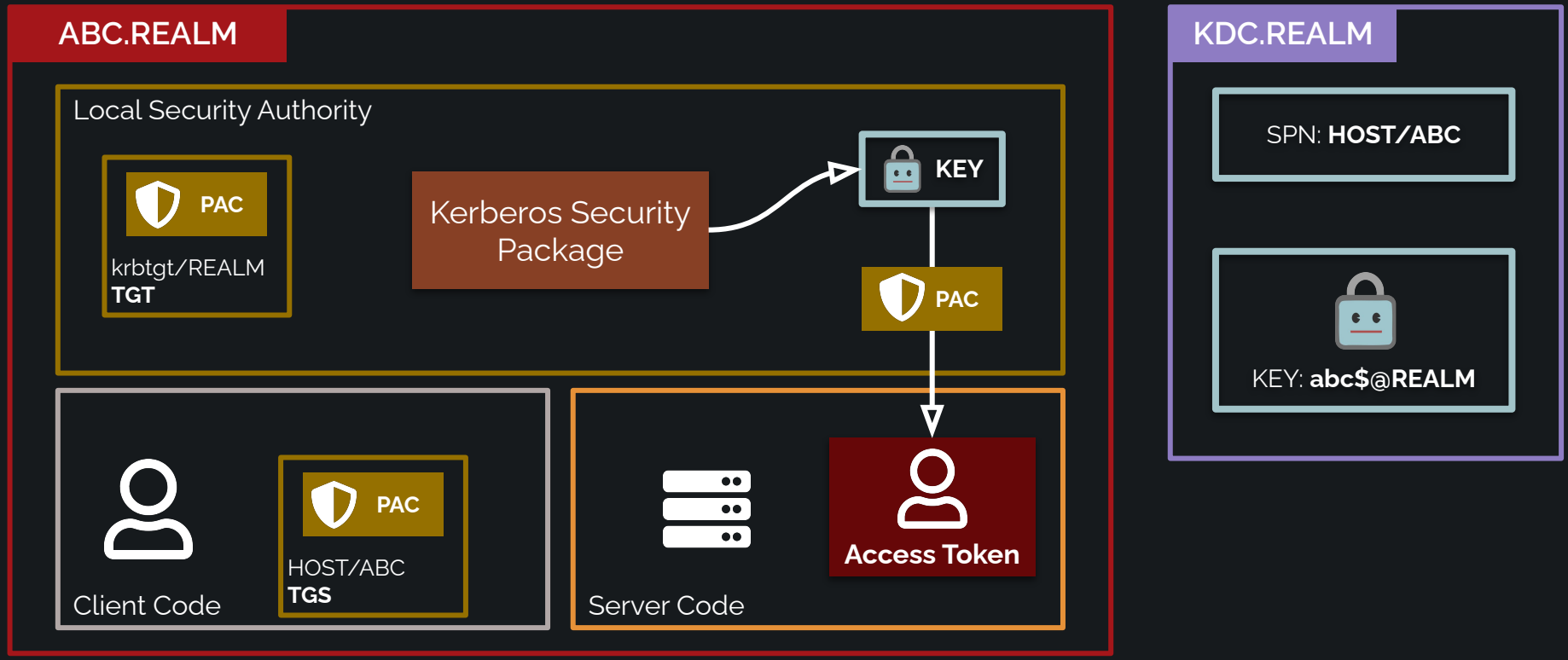
Local Kerberos Authentication



Local Kerberos Authentication

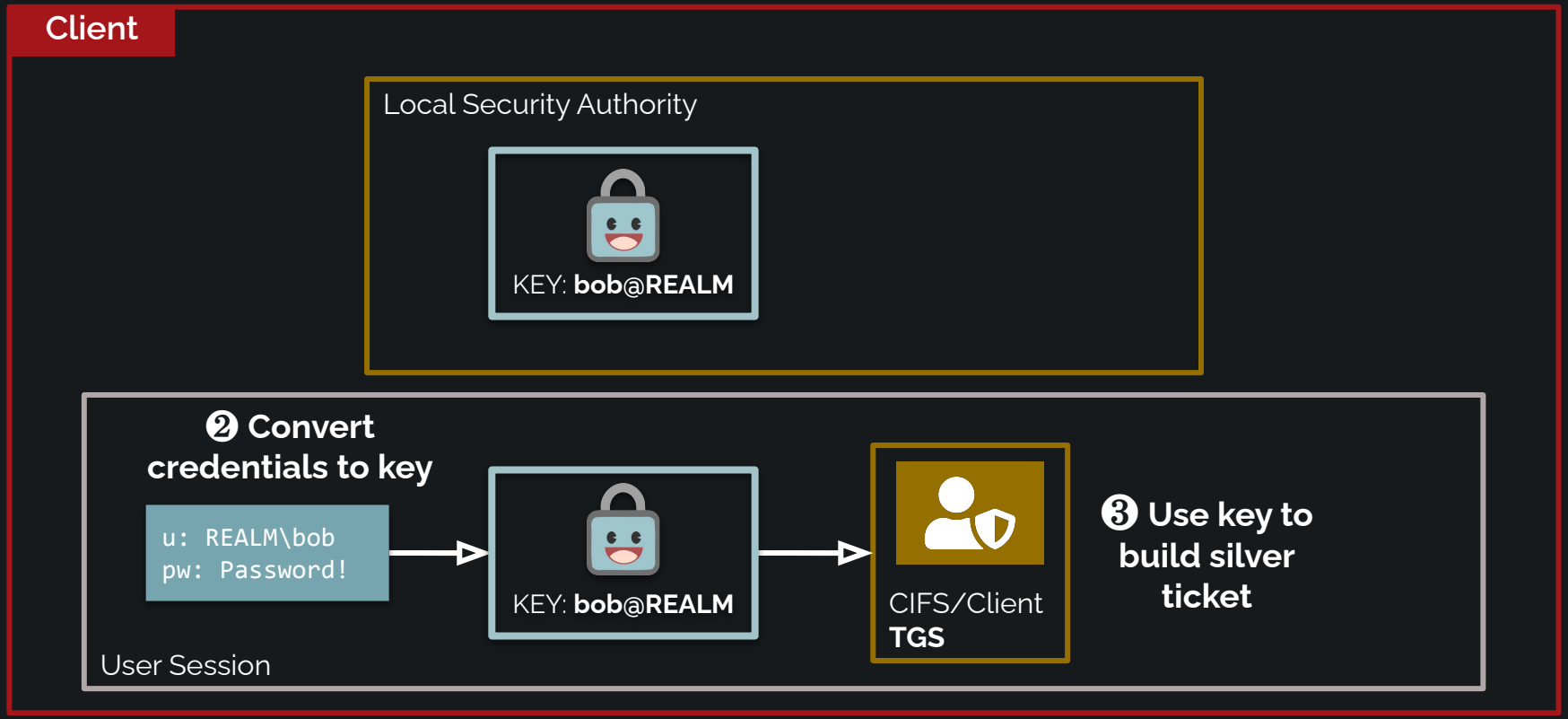


Local Kerberos Authentication

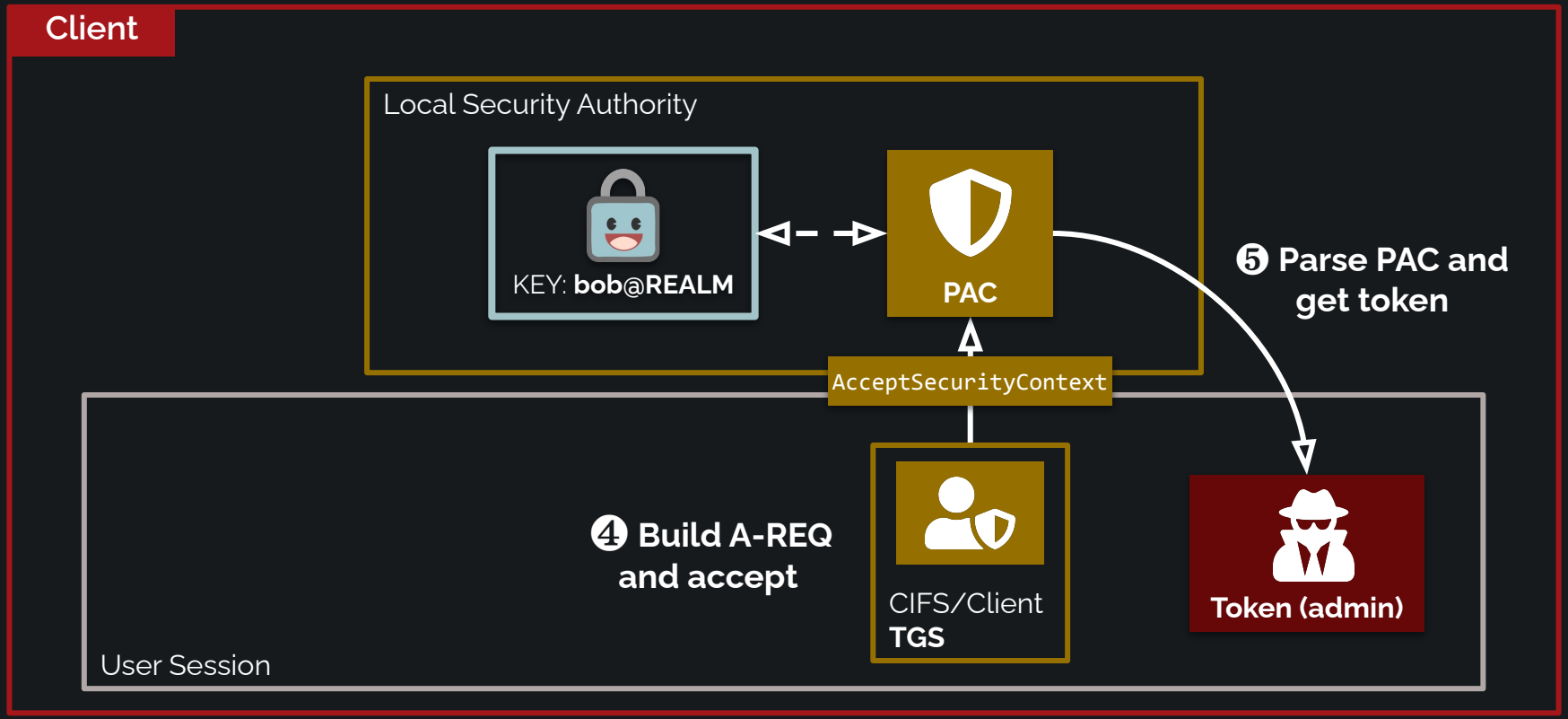


Local Kerberos Silver Ticket

Local Kerberos Silver Ticket



Local Kerberos Silver Ticket

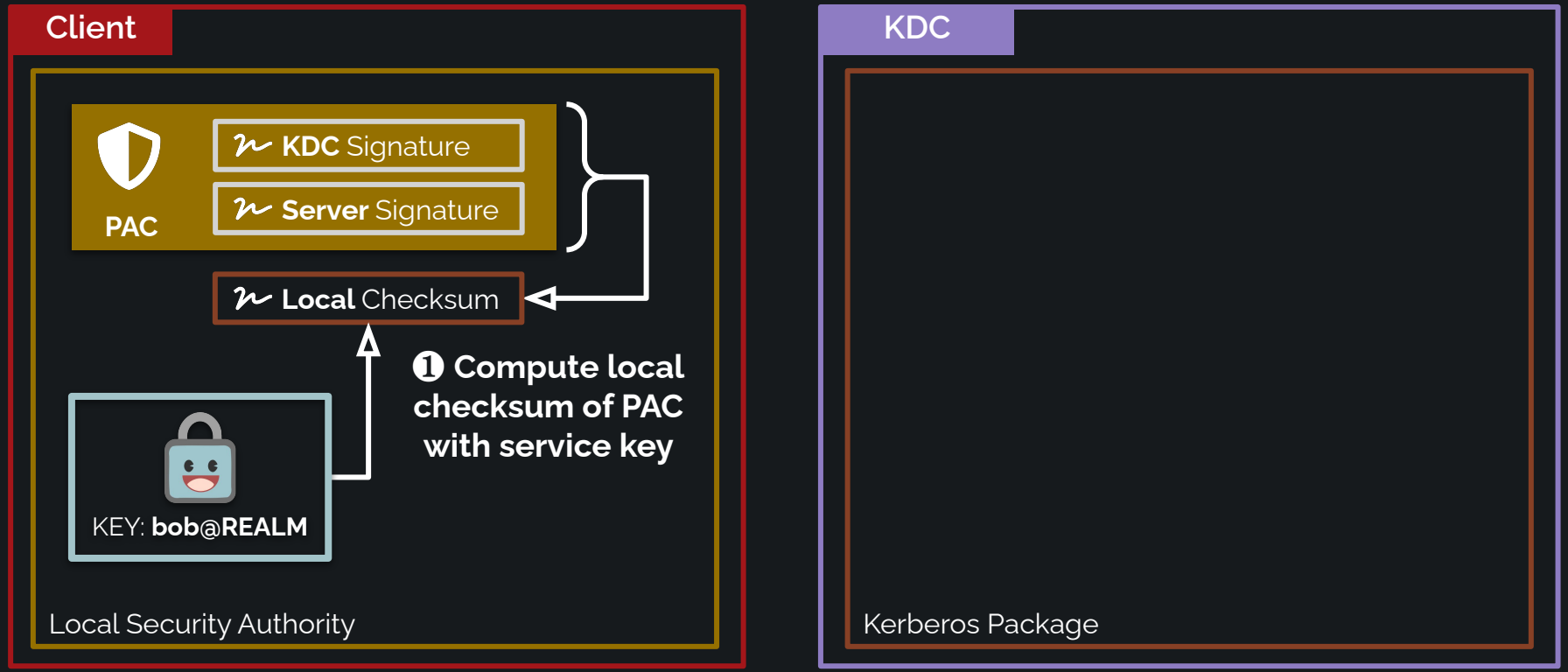


Demo Time

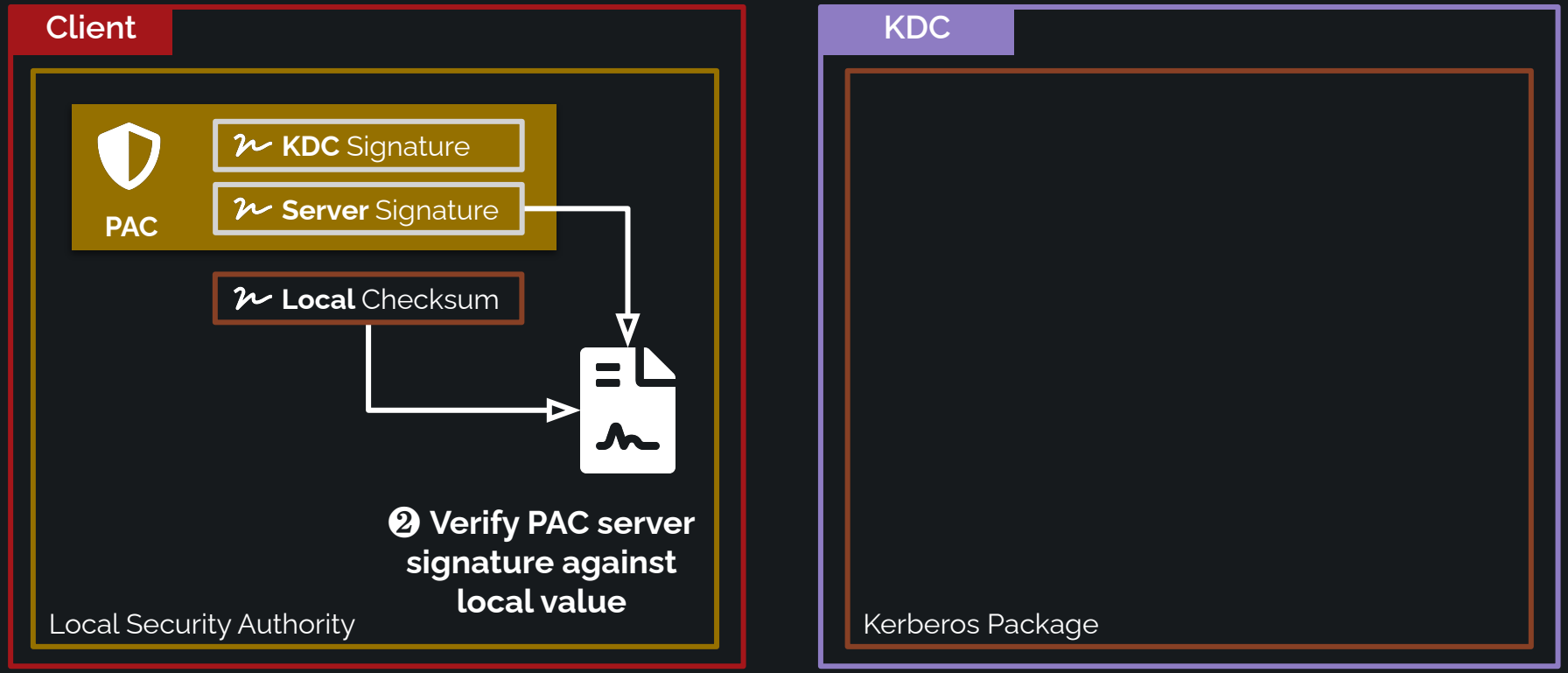
LSA Internals

and how to break them

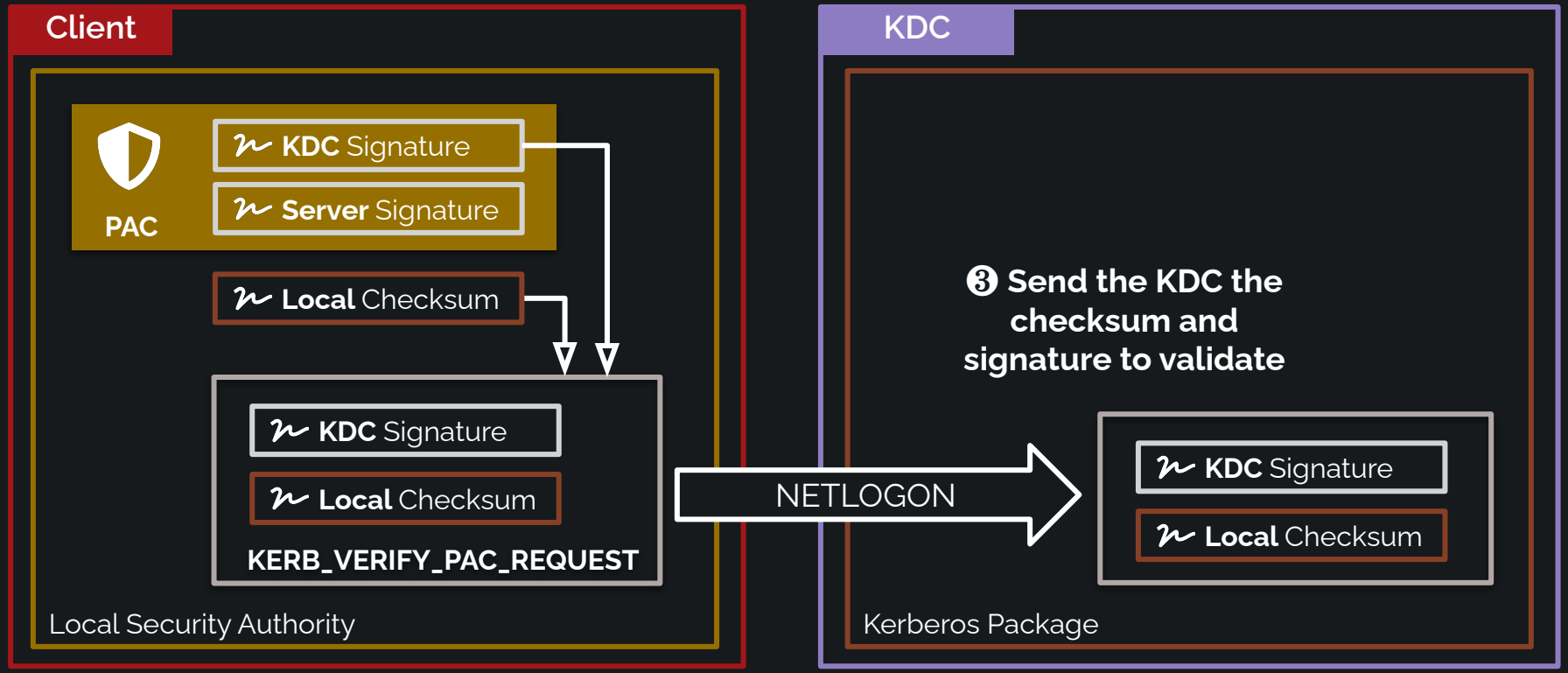
PAC Signature Validation



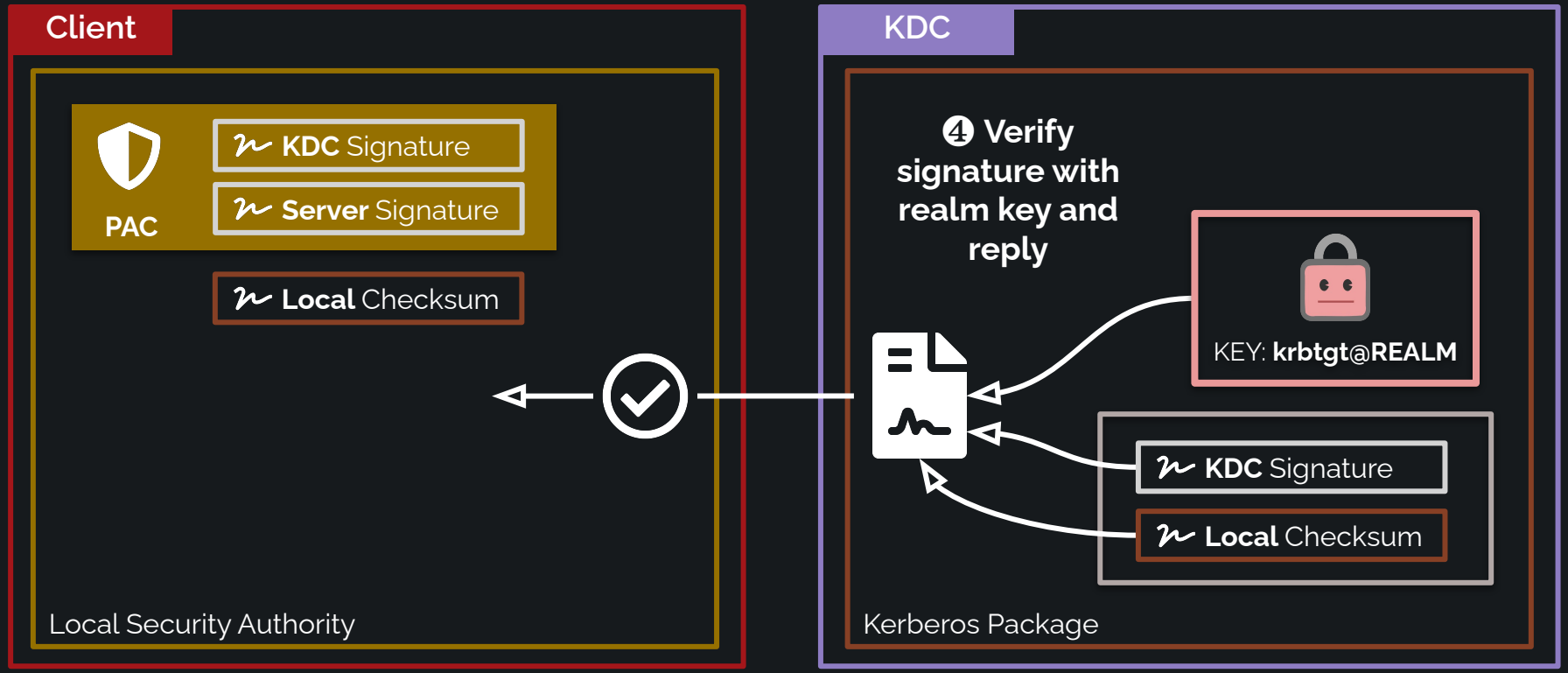
PAC Signature Validation



PAC Signature Validation



PAC Signature Validation



So how do Silver Tickets ever work?

(PAC validation isn't always enabled)

Logon Session:

"SYSTEM" Equivalent

`SeTcbPrivilege || SYSTEM || LOCAL/NETWORK SERVICE`

Credentials Handle:

cred->Flags & SECPKG_CRED_ATTR_PAC_BYPASS

(auto) **AcquireCredentialsHandle** w/
SECPKG_CRED_INBOUND && NT AUTHORITY\SERVICE &&
!KerbGlobalValidateKDCPACSignature

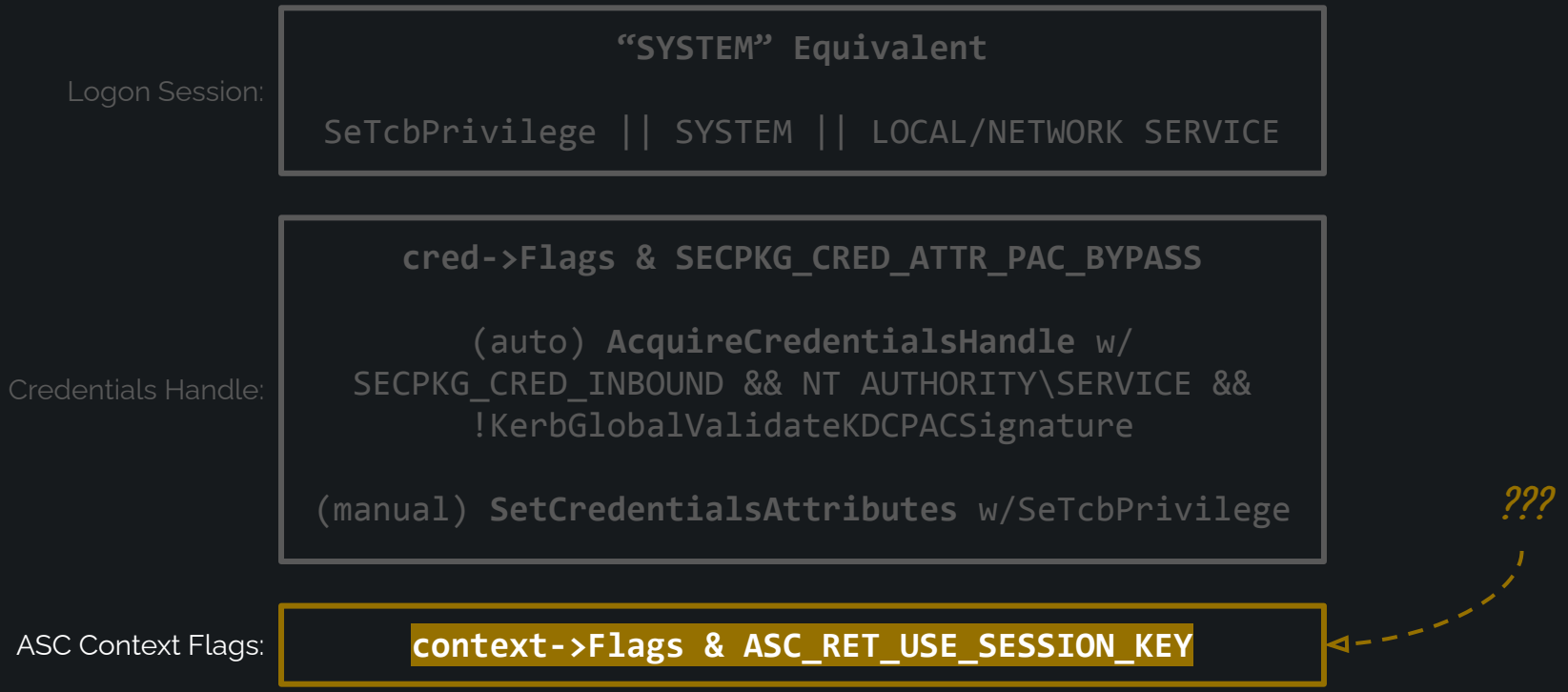
(manual) **SetCredentialsAttributes** w/SeTcbPrivilege

ASC Context Flags:

context->Flags & ASC_RET_USE_SESSION_KEY

So how do Silver Tickets ever work?

(PAC validation isn't always enabled)



ASC_RET_USE_SESSION_KEY ?

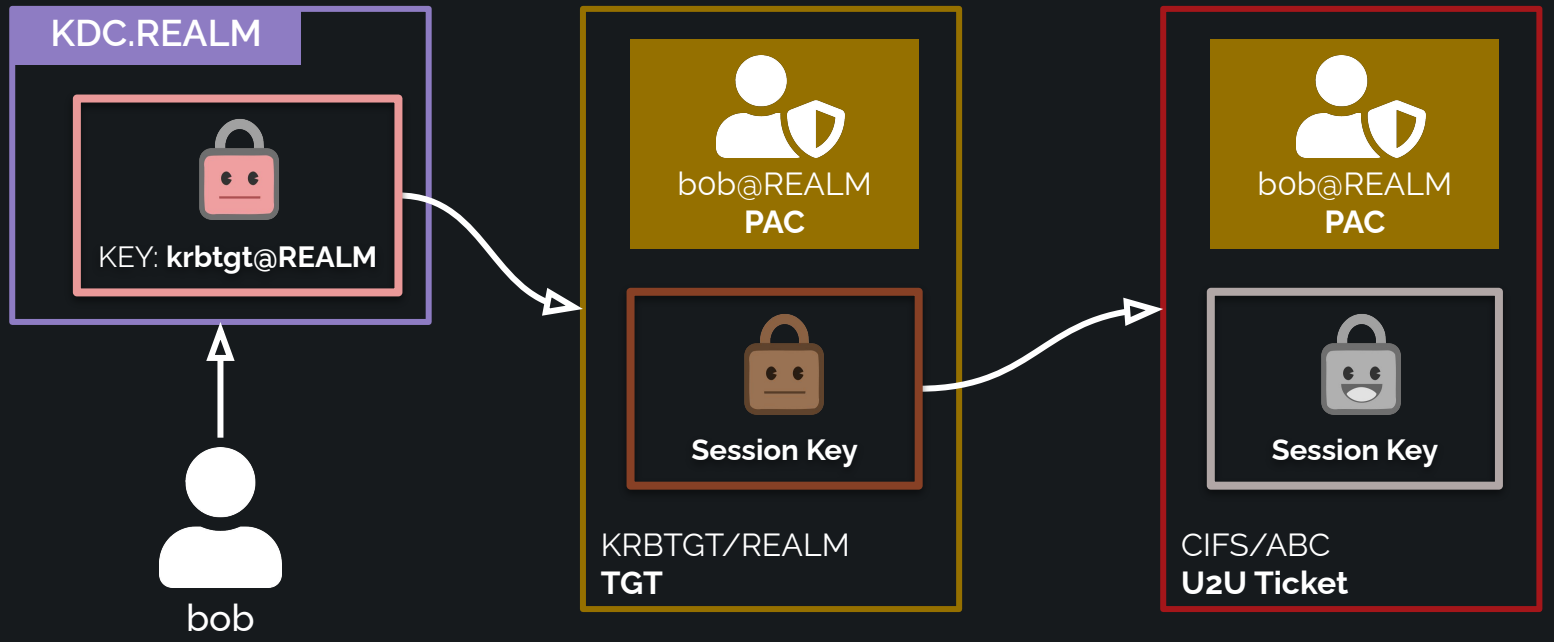
3.2.3. Receipt of KRB_AP_REQ Message

[...] If the **USE-SESSION-KEY** flag is set in the ap-options field, it indicates to the server that user-to-user authentication is in use, and that the **ticket is encrypted in the session key from the server's TGT rather than in the server's secret key.**

See Section 3.7 for a more complete description of the effect of **user-to-user authentication** on all messages in the Kerberos protocol.

RFC 4120 - Kerberos V5

User to User Authentication



We now need the session key from our TGT to build the Silver Ticket

Trying to get a TGT + Session Key

```
PS C:\> $ticket = Get-KerberosTicket krbtgt
PS C:\> $ticket.SessionKey

KeyEncryption  : AES256_CTS_HMAC_SHA1_96
Principal      : krbtgt/DOM.LOC@DOM.LOC
NameType       : SRV_INST

PS C:\> $ticket.SessionKey.Key | Format-HexDump
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

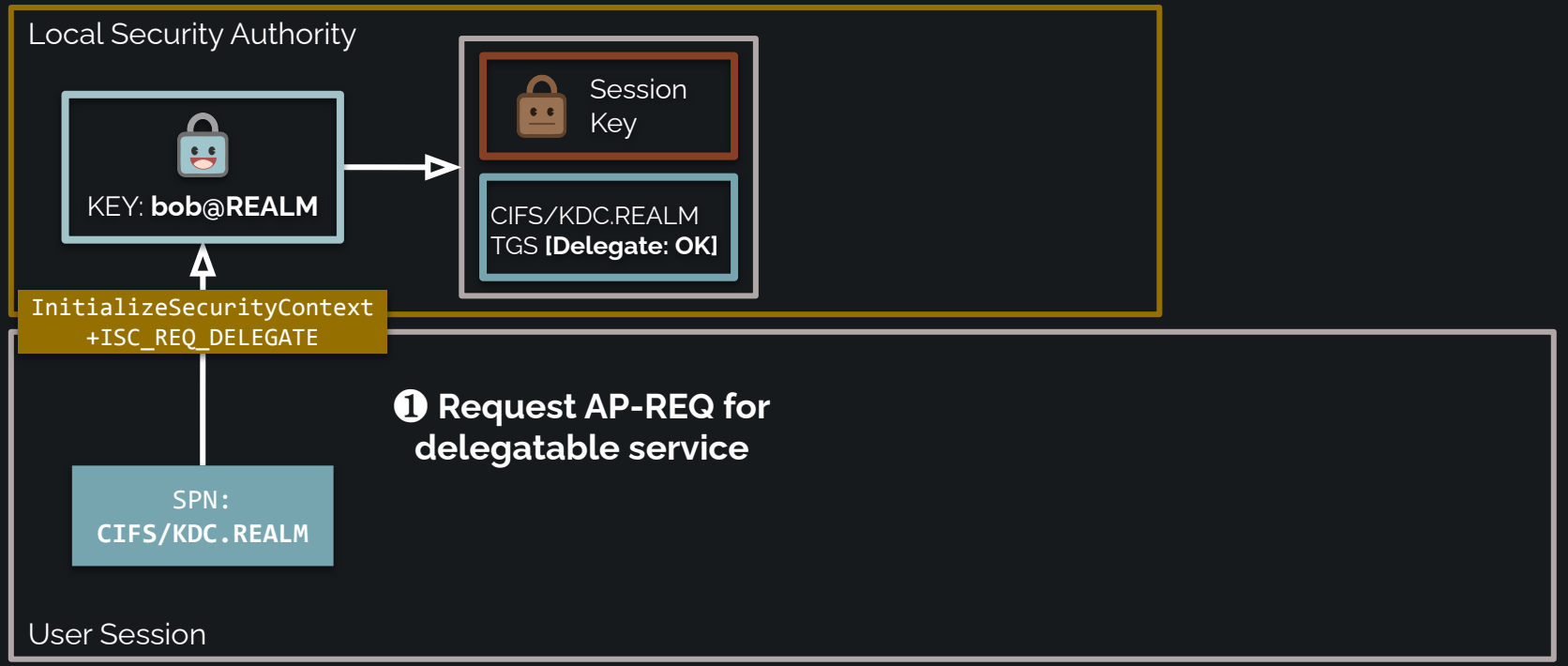
No session key ?

Benjamin Delpy
@gentilkiwi

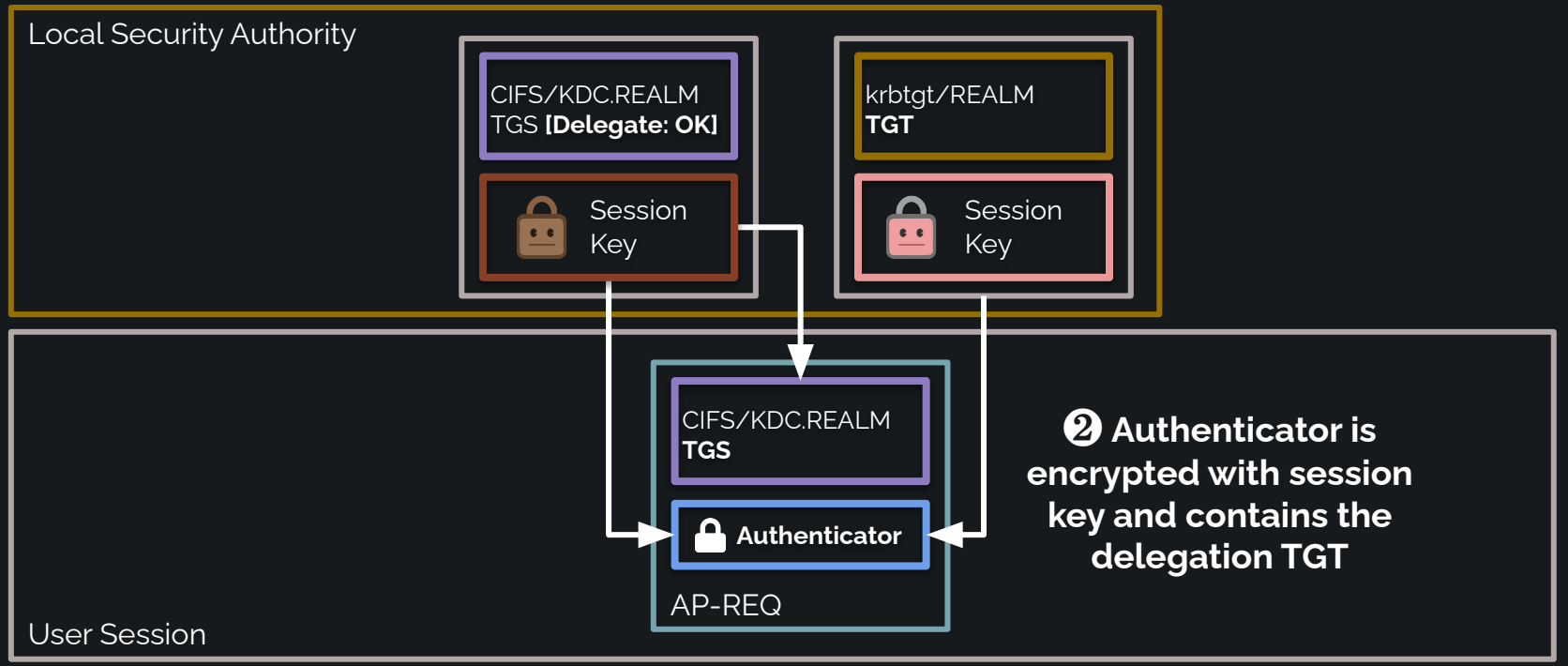
Want to get a usable Kerberos TGT without admin rights/allowtgtsessionkey?
It's easy with a delegation ticket! (enabled by default...)
No special requirement, just some love
> github.com/gentilkiwi/kek...
Thank you @elad_shamir (and @TheColonial) for evil ideas!

https://twitter.com/gentilkiwi/status/998219775485661184

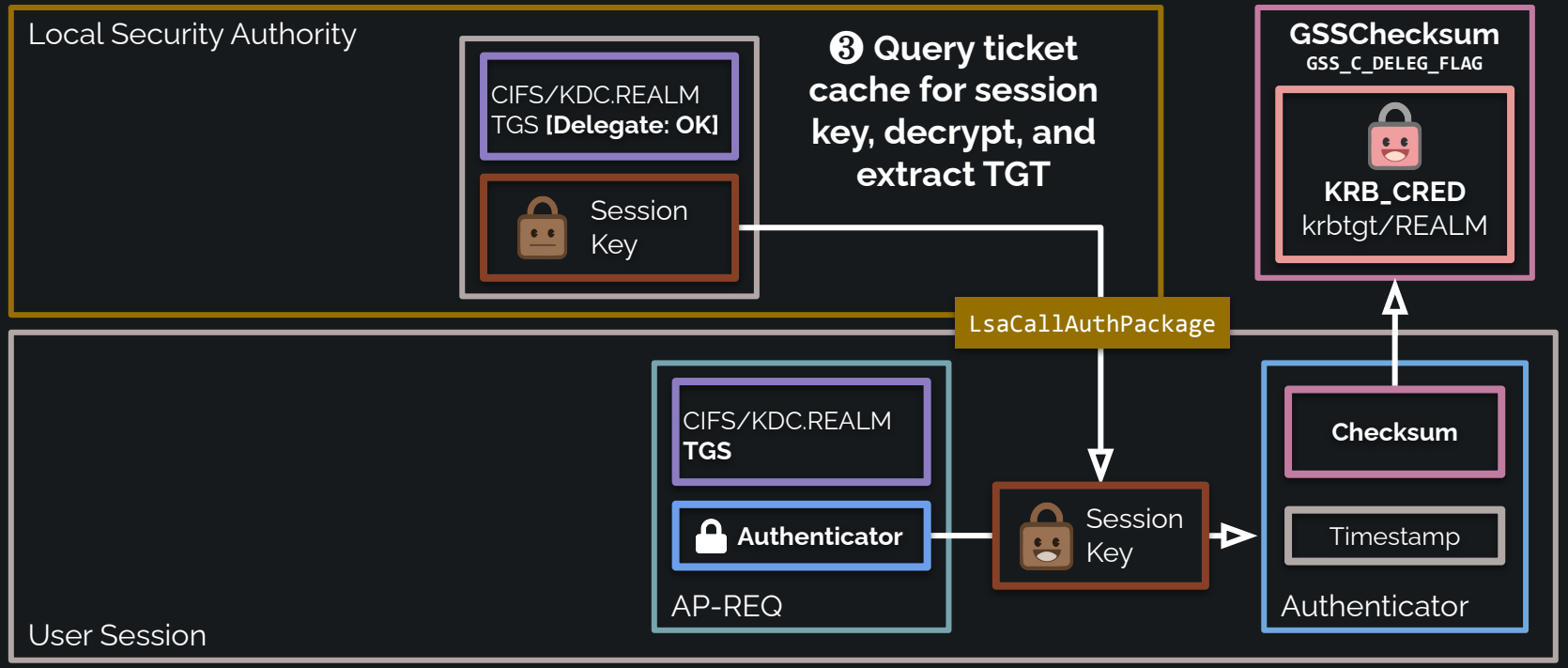
Unconstrained Delegation TGT Extraction



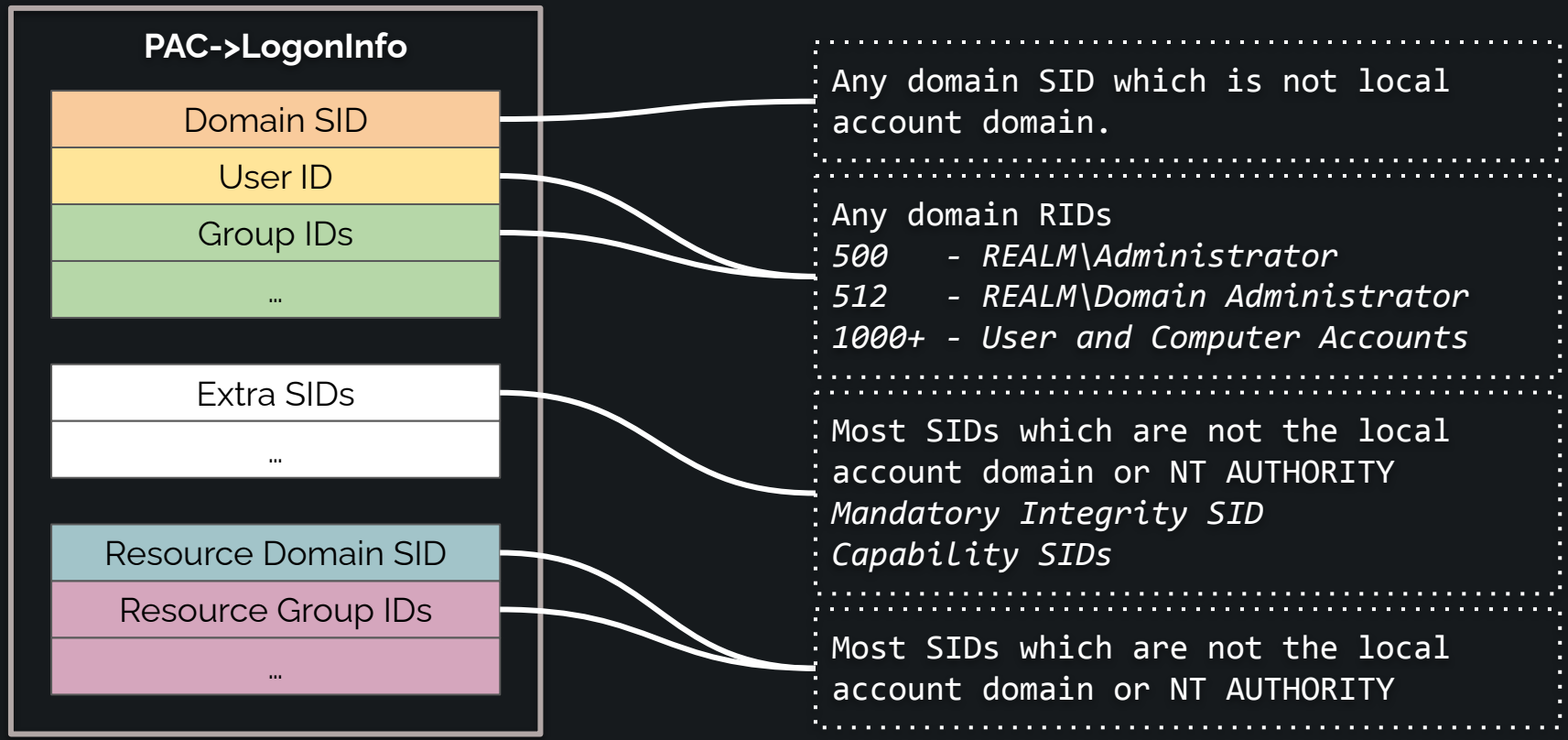
Unconstrained Delegation TGT Extraction



Unconstrained Delegation TGT Extraction

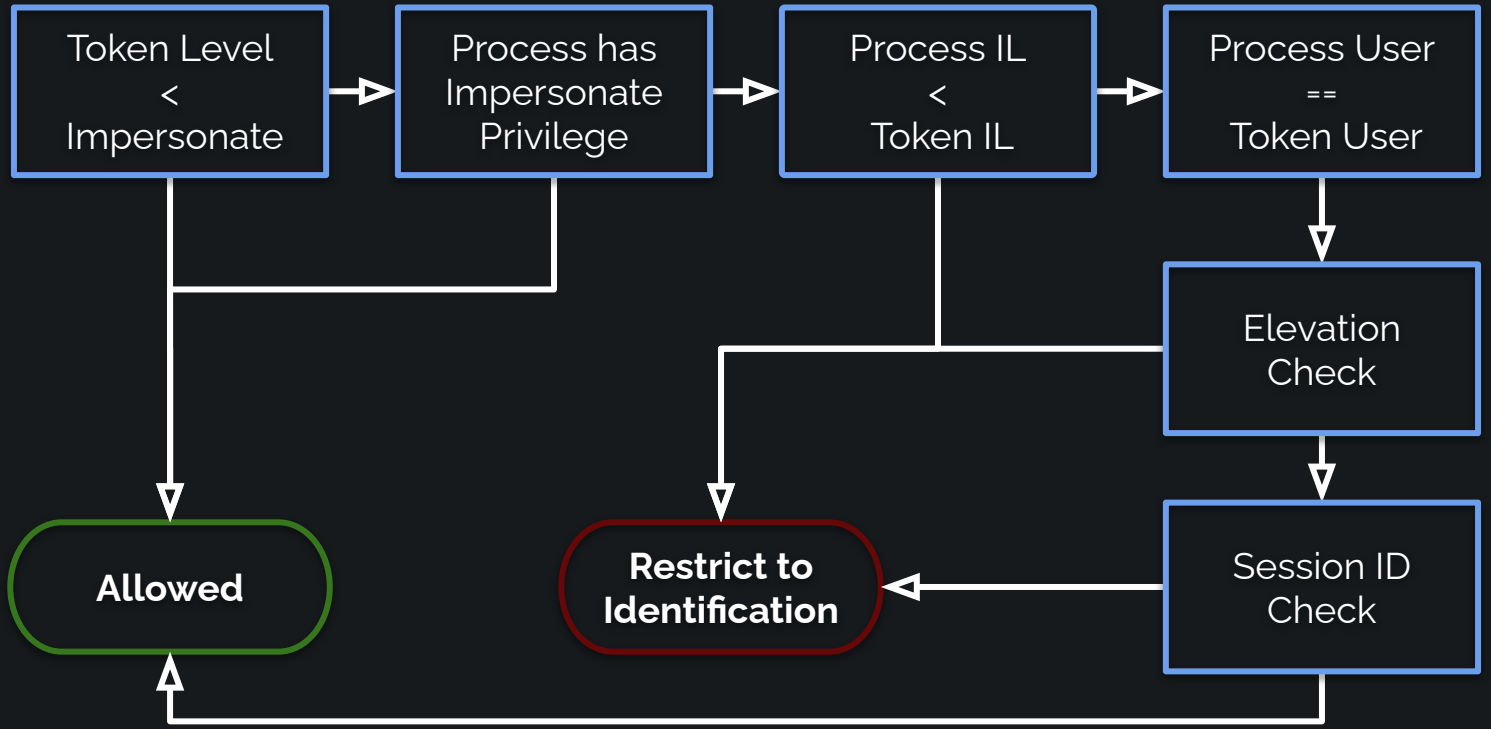


What can you add to the PAC ?

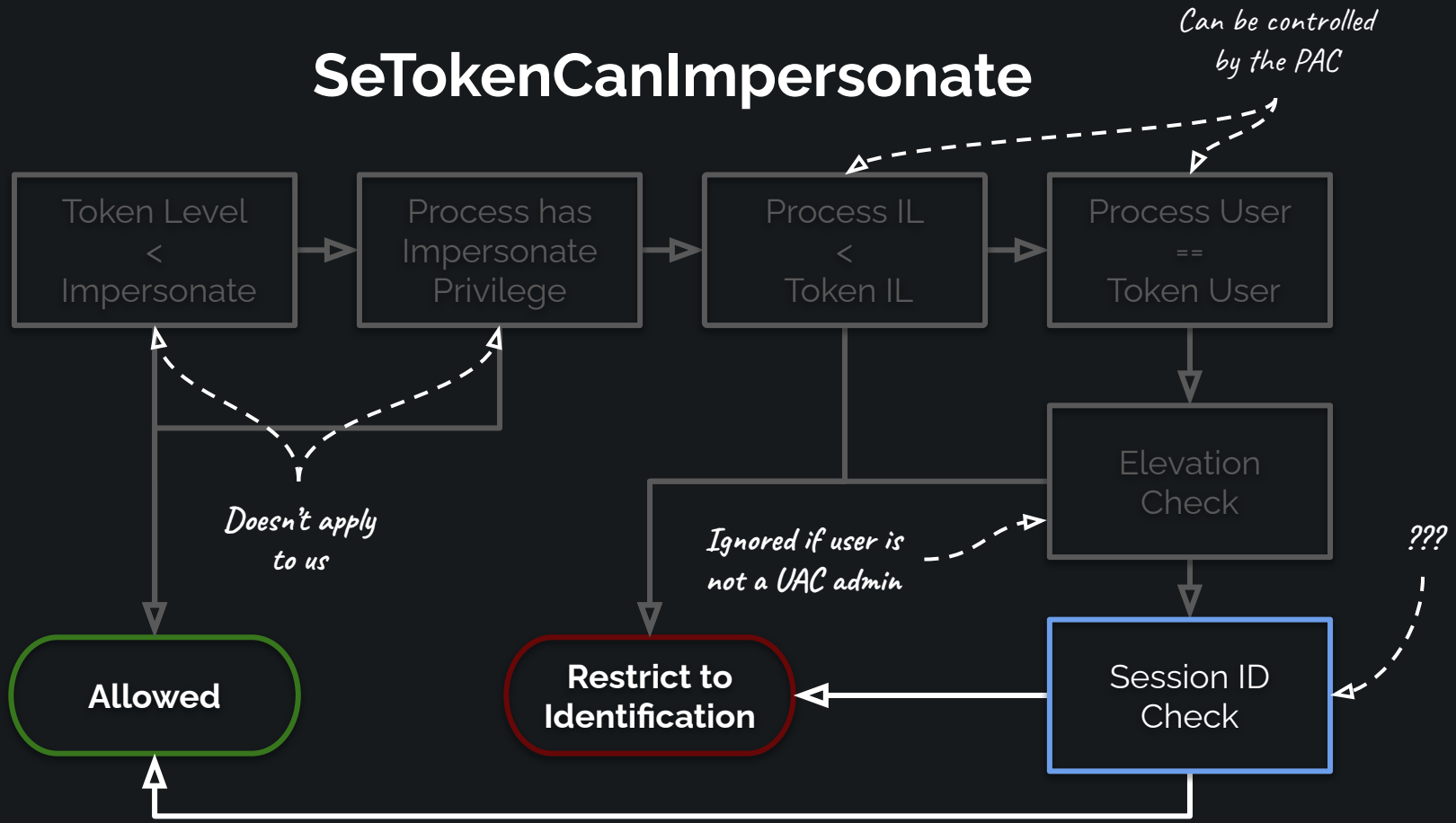


Demo Time

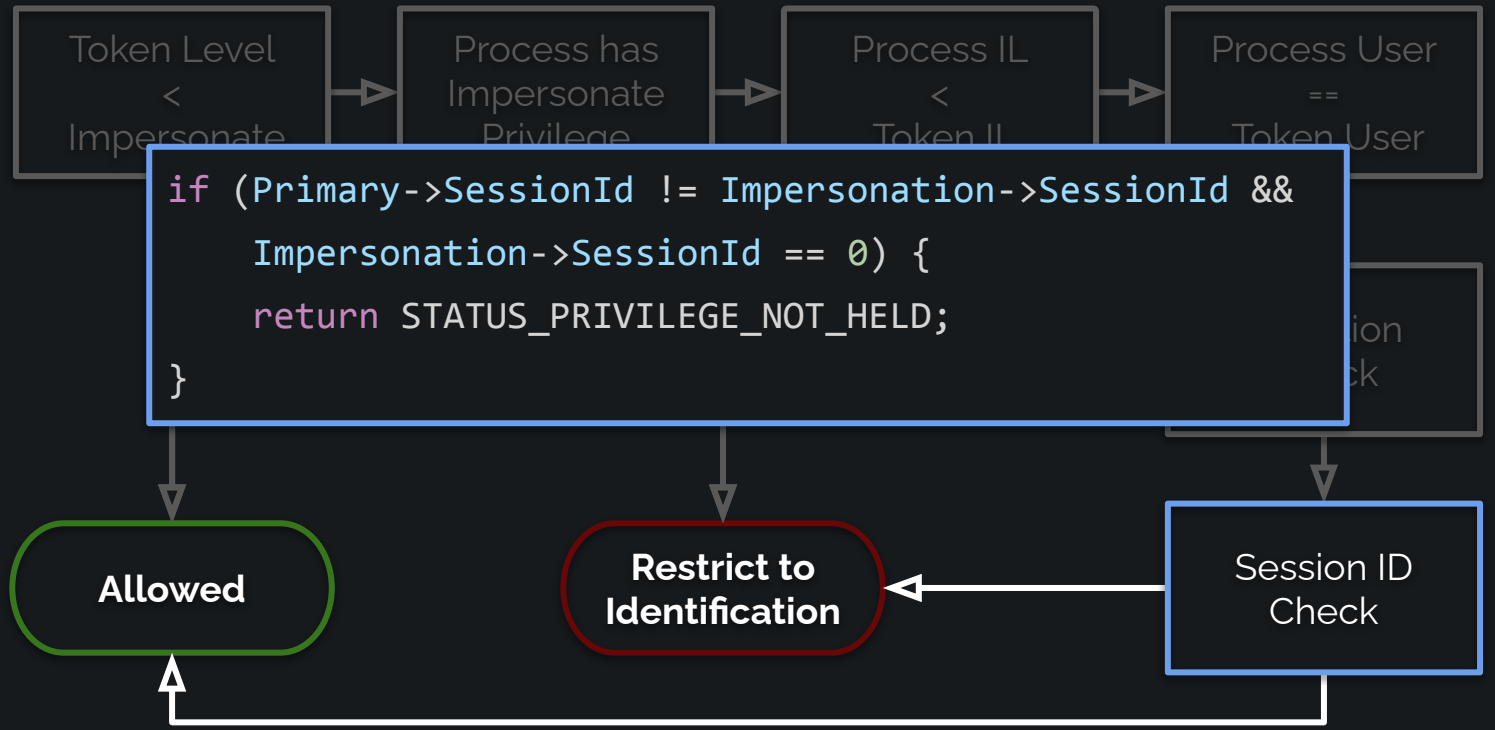
SeTokenCanImpersonate



SeTokenCanImpersonate

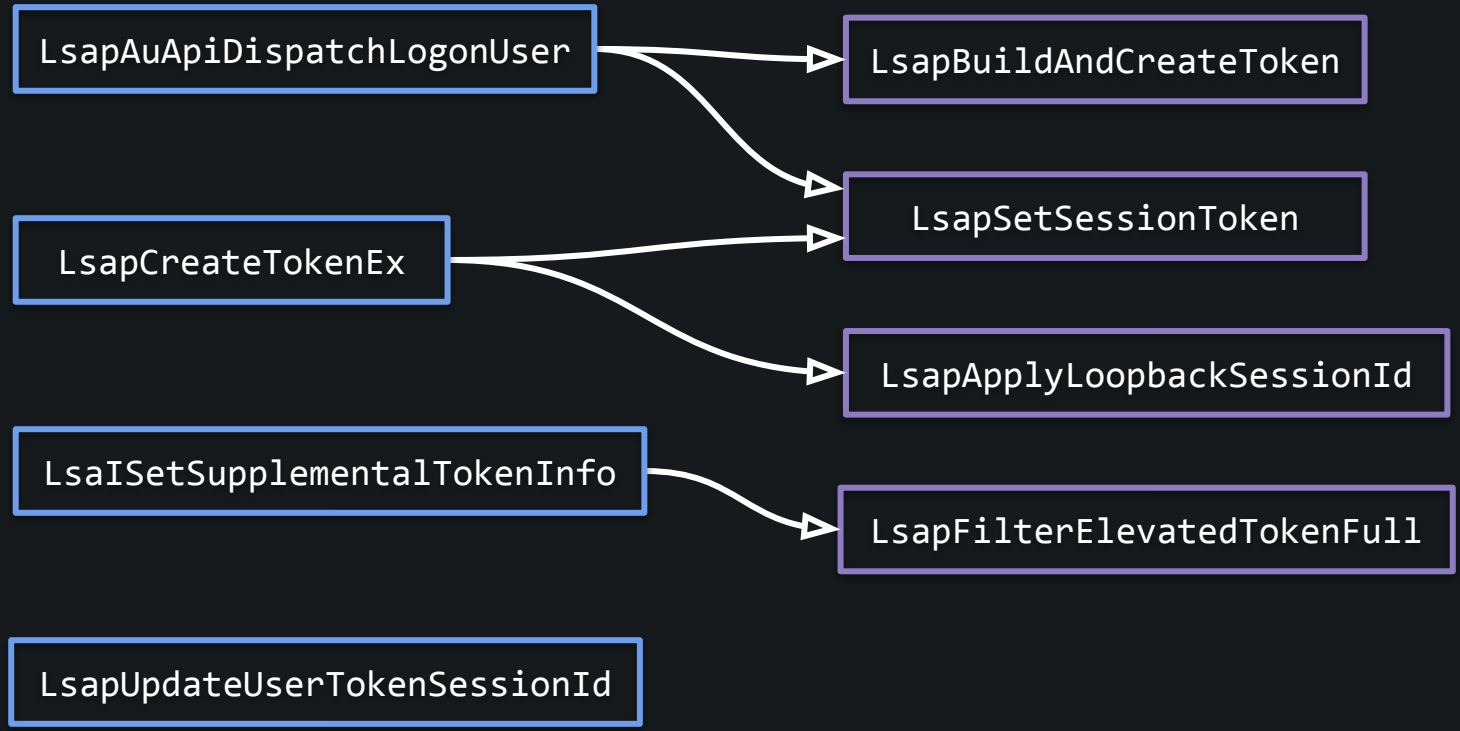


SeTokenCanImpersonate



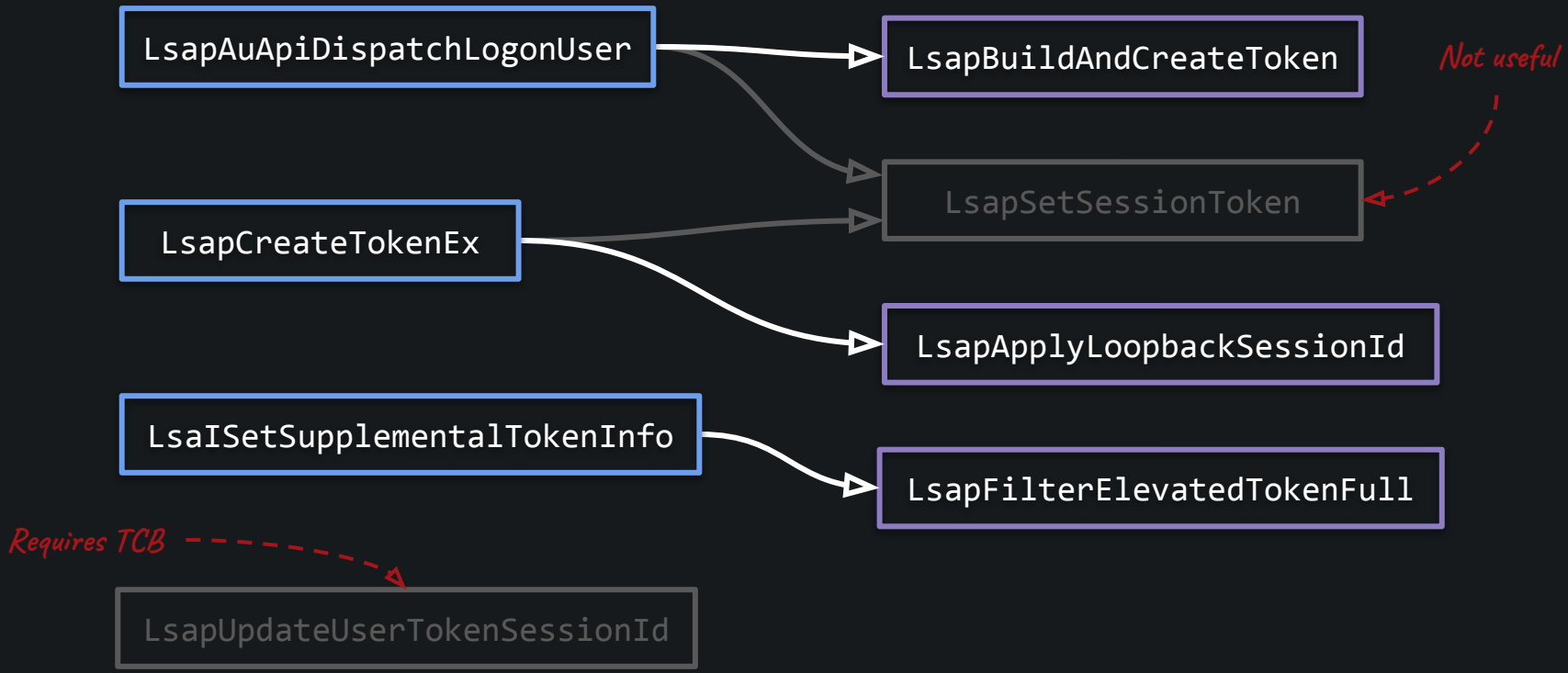
Hunting for Session Update Primitives

`NtSetInformationToken(..., TokenSessionId, ...)`



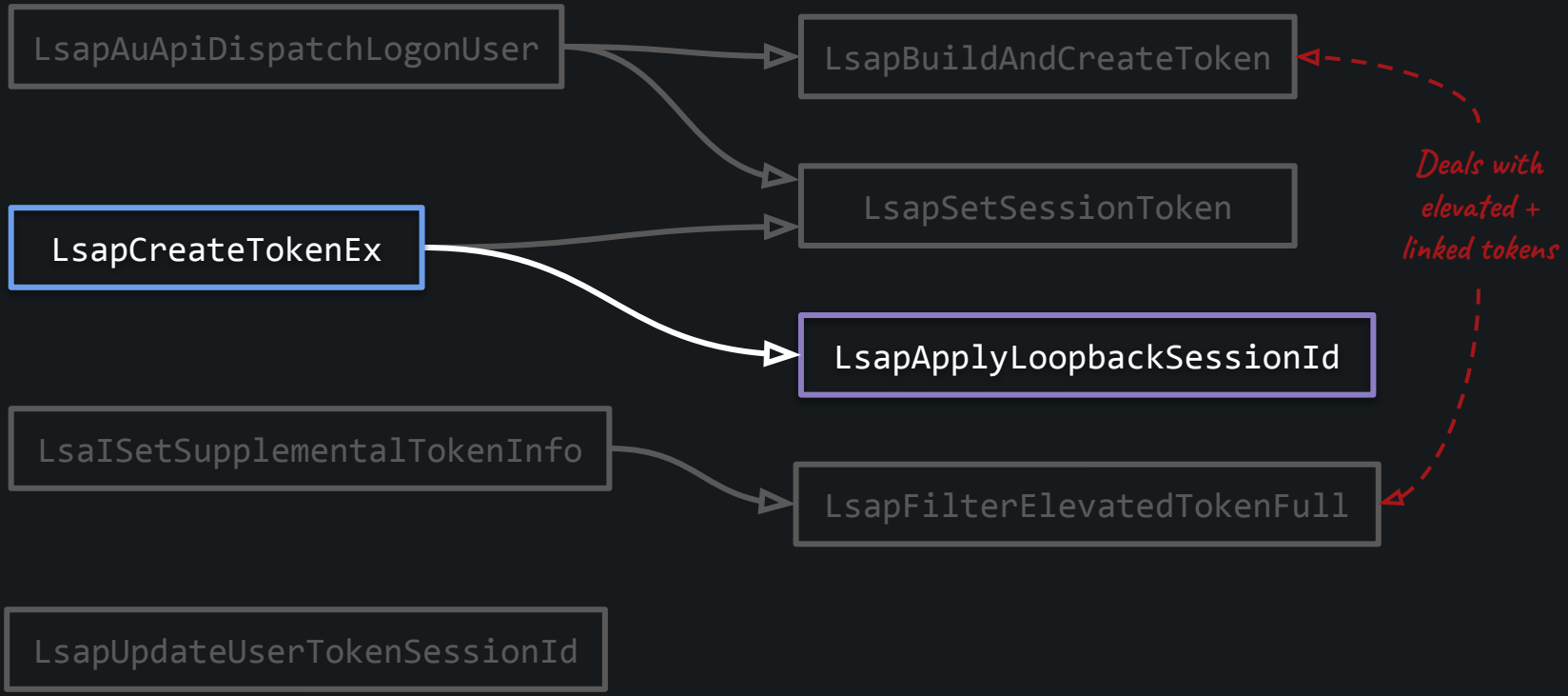
Hunting for Session Update Primitives

`NtSetInformationToken(..., TokenSessionId, ...)`



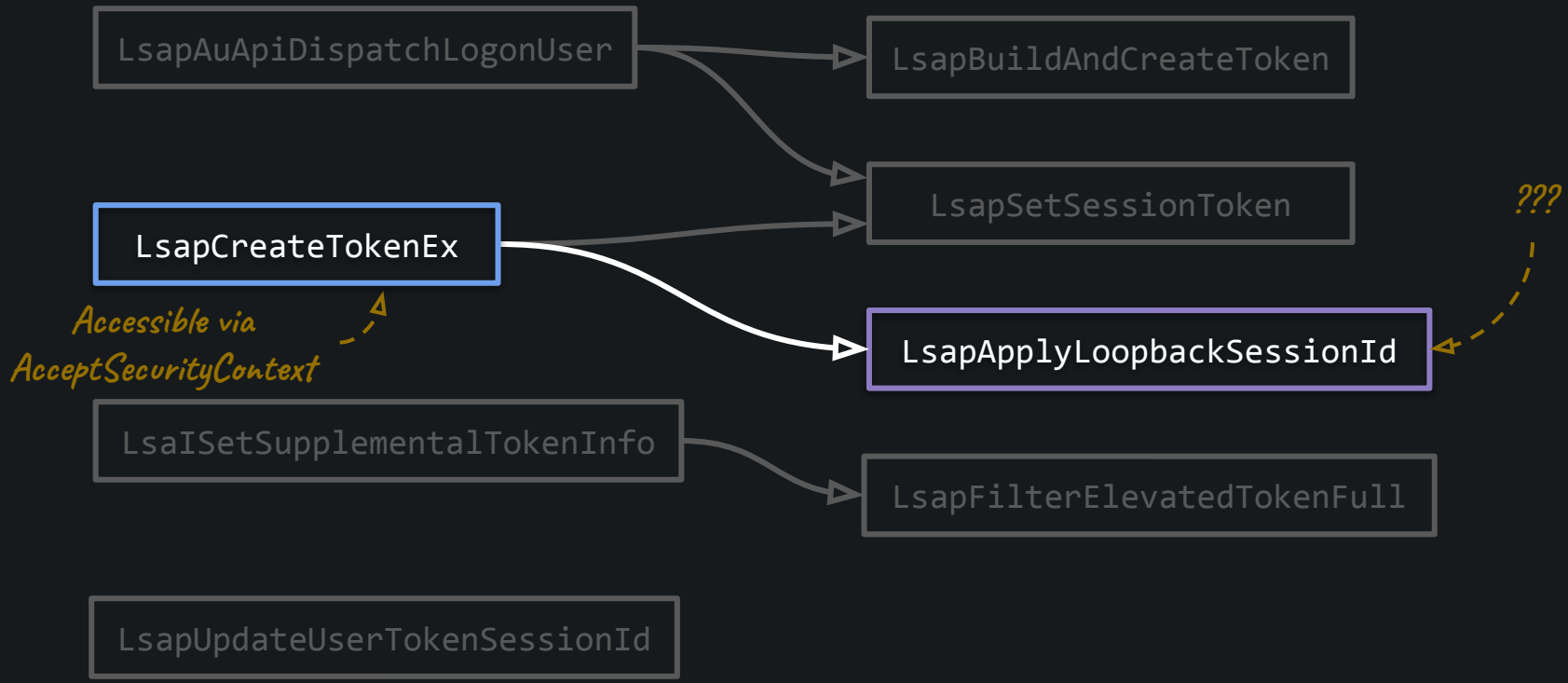
Hunting for Session Update Primitives

`NtSetInformationToken(..., TokenSessionId, ...)`

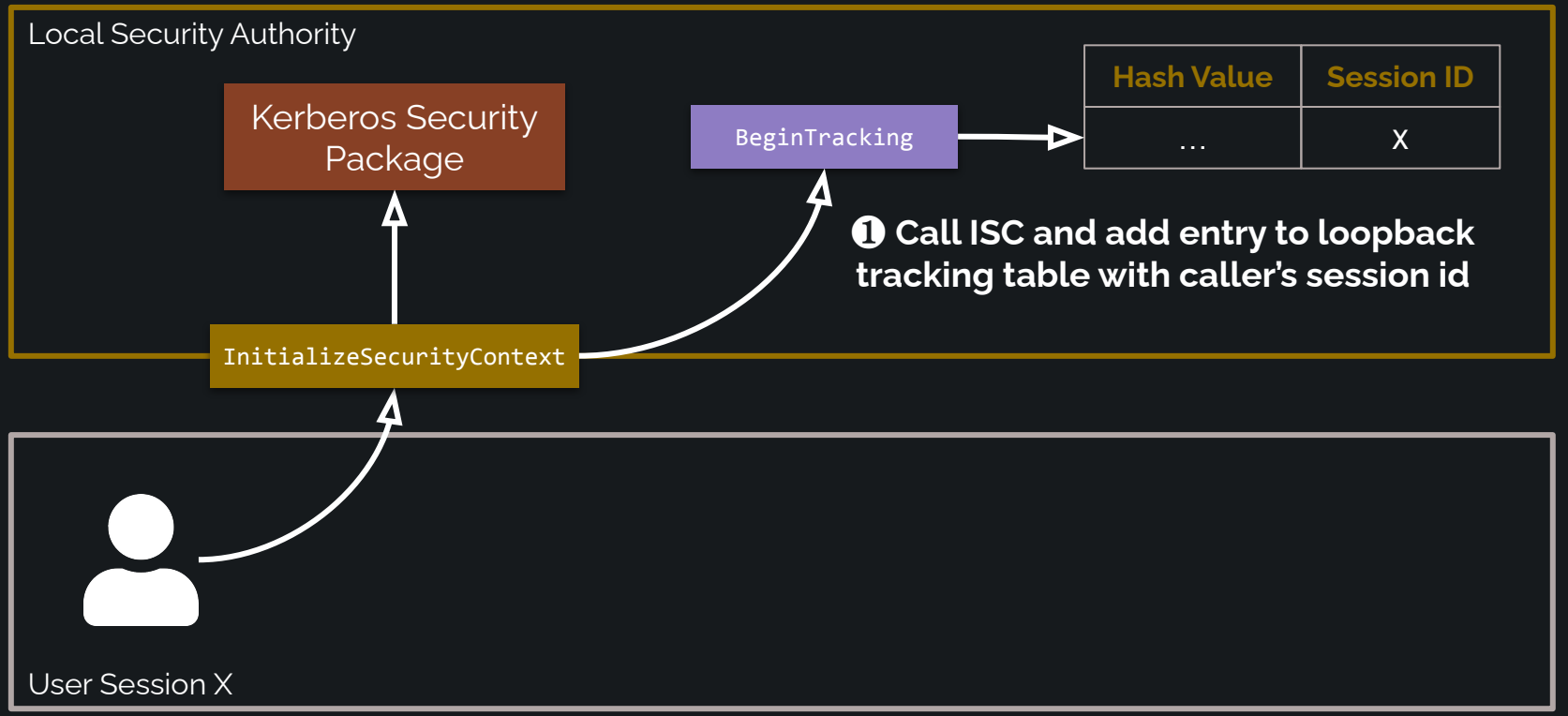


Hunting for Session Update Primitives

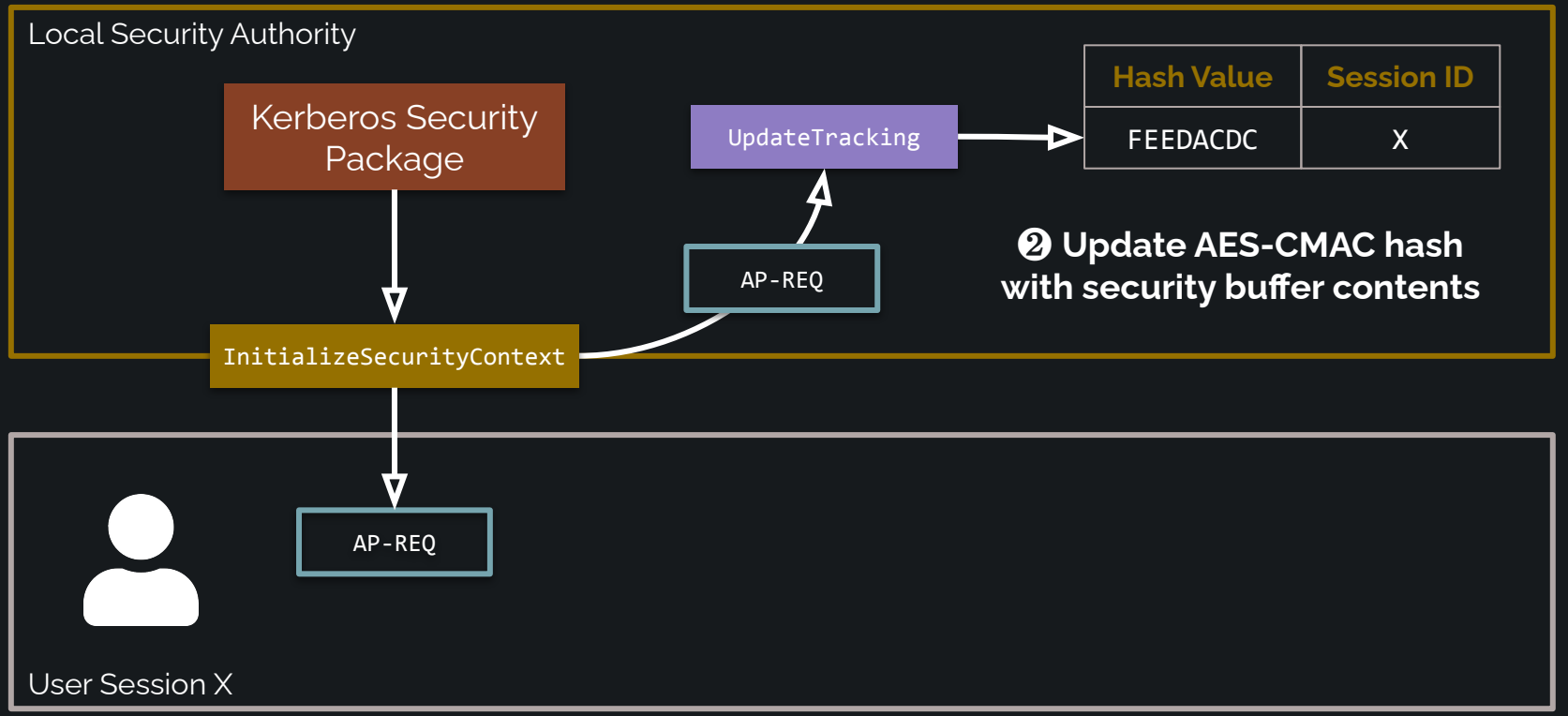
`NtSetInformationToken(..., TokenSessionId, ...)`



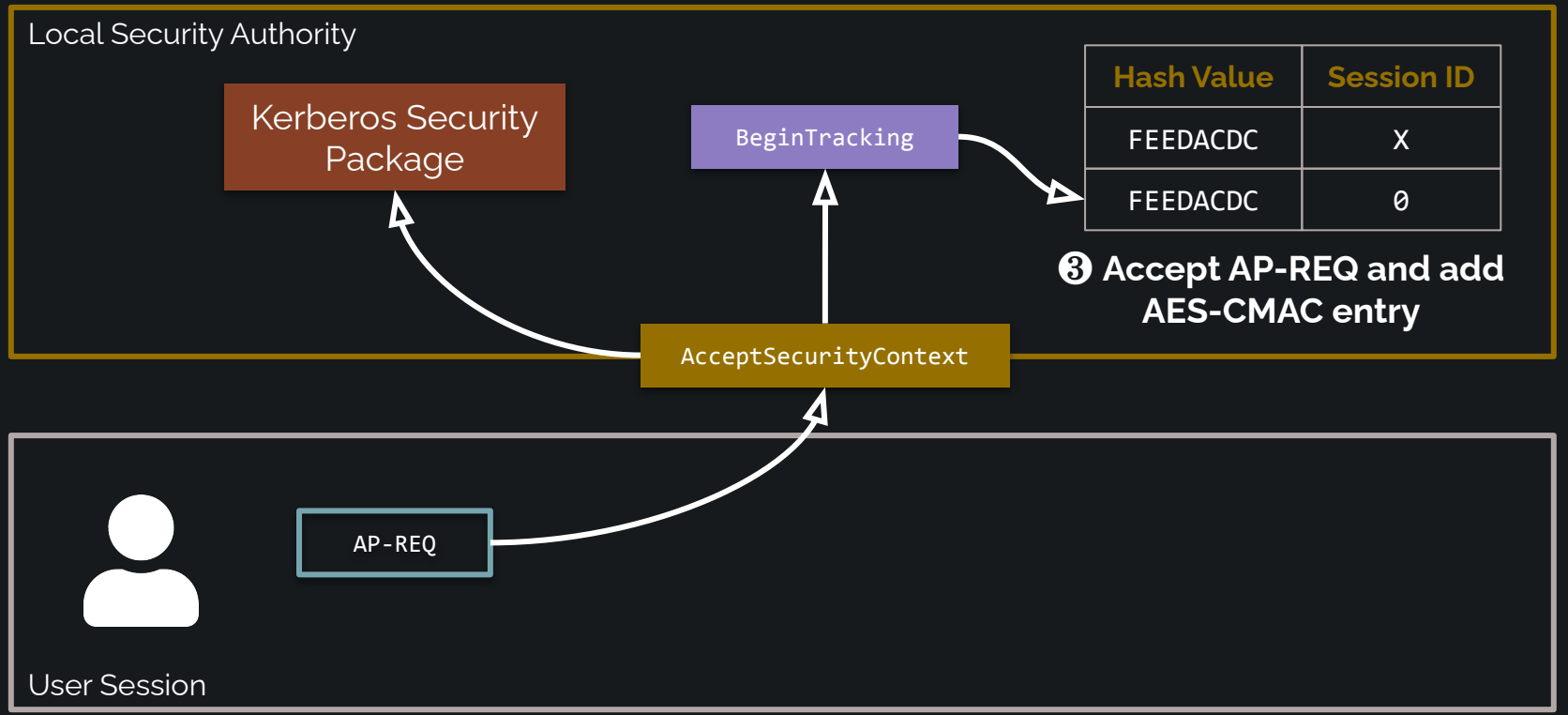
LSA Loopback Library



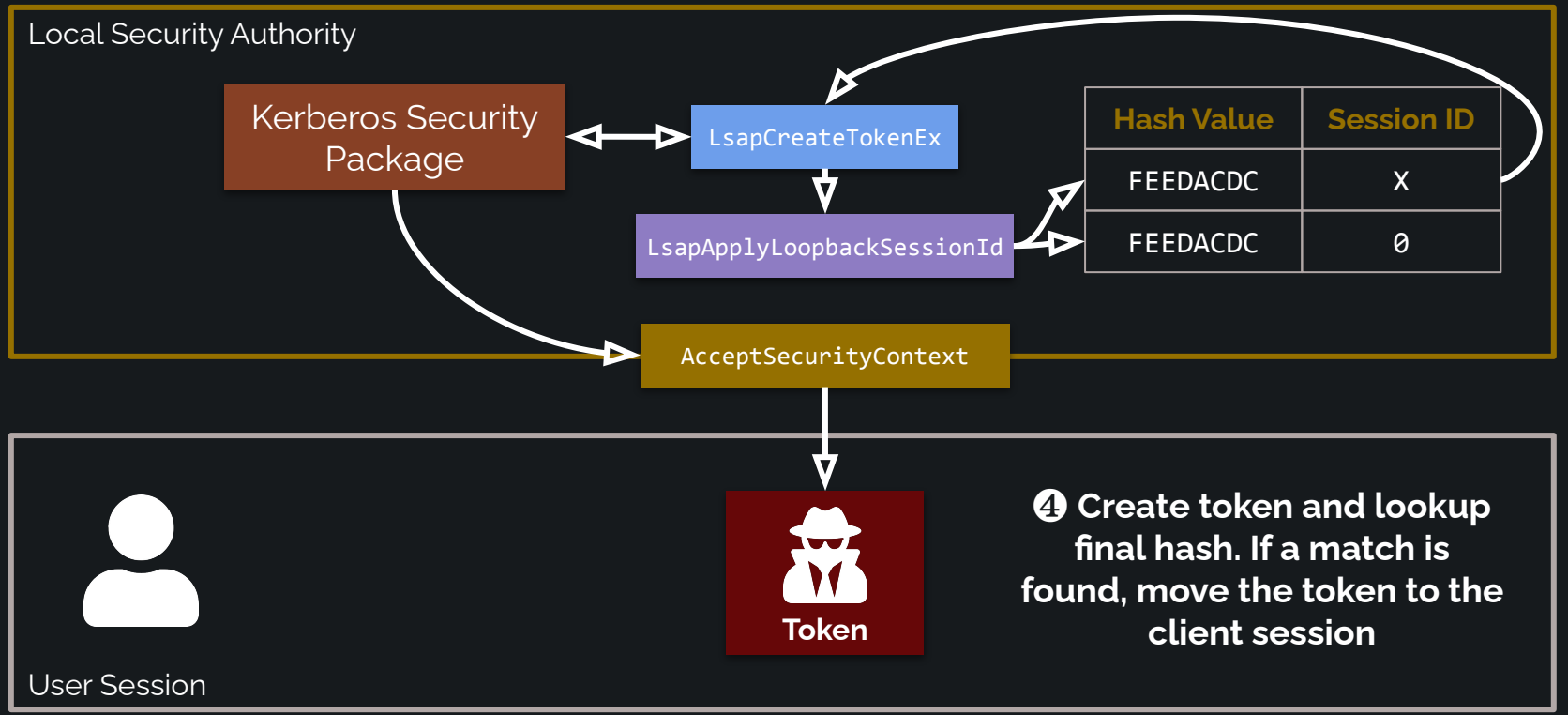
LSA Loopback Library



LSA Loopback Library



LSA Loopback Library

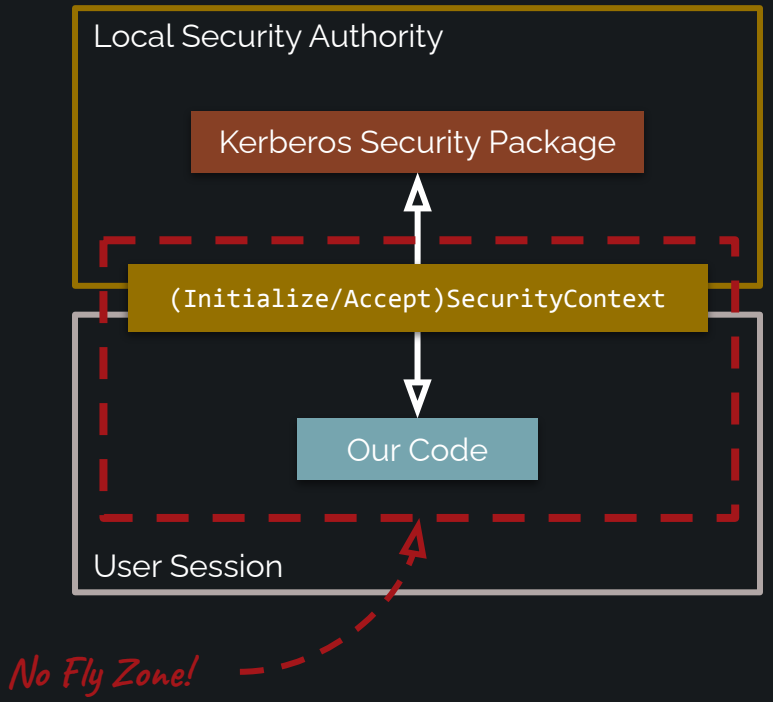


LSA Loopback Library
ELI5

1. Loopback Library will hash all security buffers between LSA and clients. **If hashes match when a token is being built, the token will be moved to the client session.**

2. We need to start using **InitializeSecurityContext** with our silver tickets to get the hash entry initialized.

3. We need to modify the PAC inside the AP-REQ, but **if we touch the buffers the hash lookup will break.** (or will it?)



Loopback Security Buffer Hashing Bug

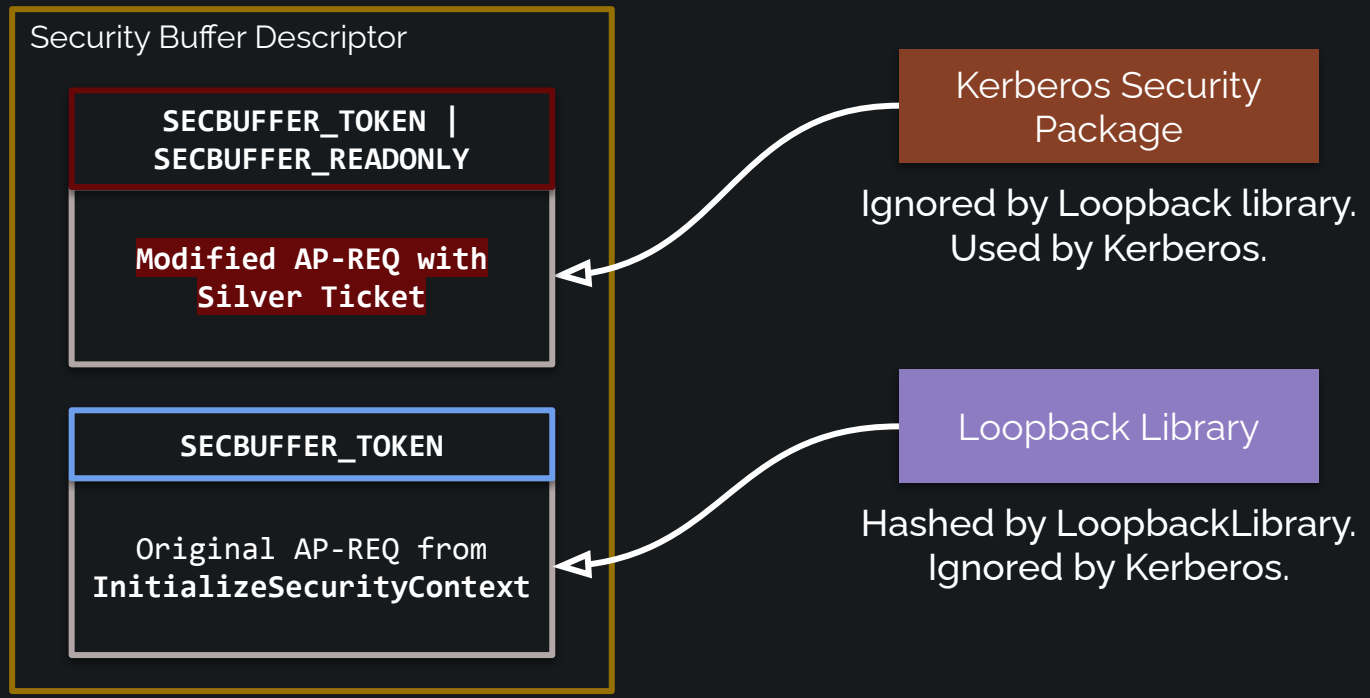
```
PSecBufferDesc pInput = ...;
for(ULONG i = 0; i < pInput->cBuffers; ++i) {
  PSecBuffer pBuffer = &pInput->pBuffers[i];

  if (pBuffer.BufferType == SECBUFFER_TOKEN) {
    BCryptHashData(hHash, pDirectionGuid, cbDirectionGuid);
    BCryptHashData(hHash, pBuffer->pvBuffer, pBuffer->cbBuffer);
  }
}
```

Security Buffer Types

Buffer Type	Meaning	Value
SECBUFFER_EMPTY	Undefined, replaced by the security package function	0x00000000
SECBUFFER_TOKEN	**Security token**	**0x00000002**
...		
SECBUFFER_READONLY	**Buffer is read-only, no checksum**	**0x80000000**
SECBUFFER_READONLY_WITH_CHECKSUM	**Buffer is read-only, and checksummed**	**0x10000000**

The buffer types can be combined using a **bitwise-OR** operation with the READONLY buffer types.

Type Confusion in AcceptSecurityContext



Demo Time

Fixed in Windows 11 ?

Windows 10:

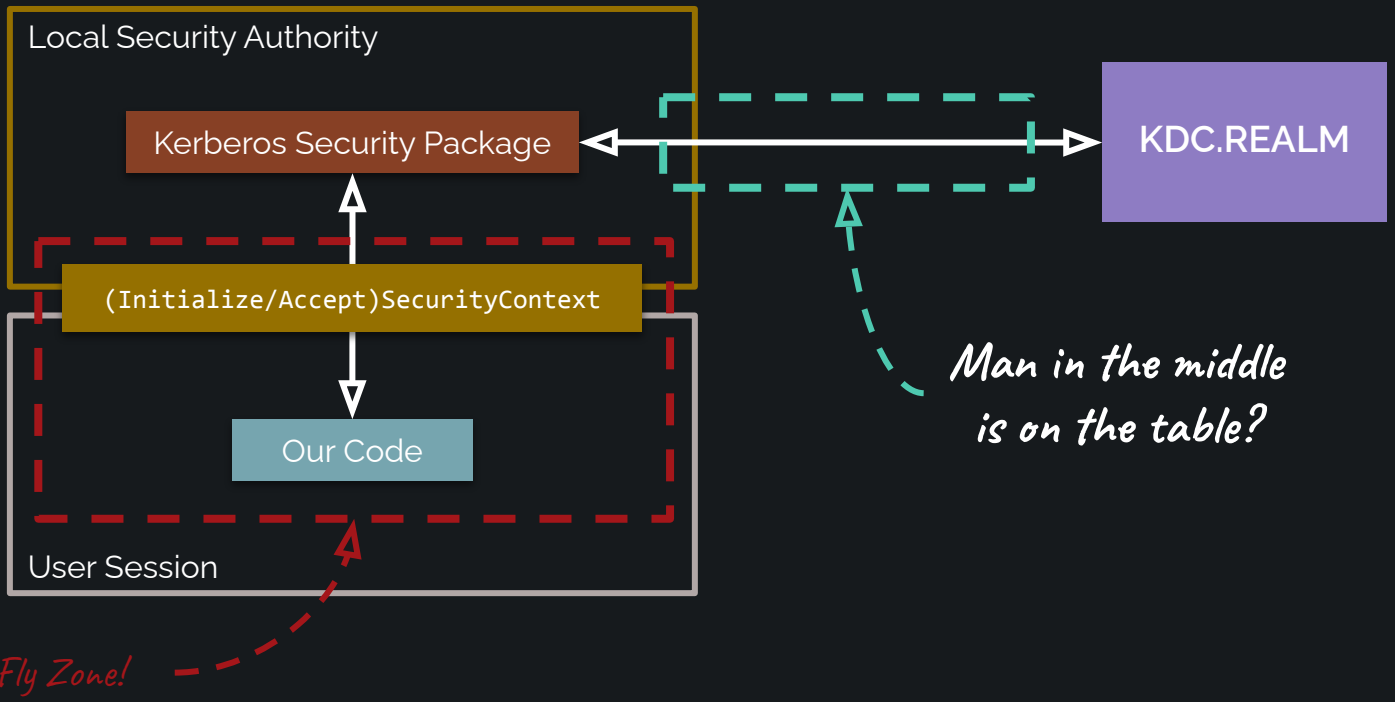
```
if (pBuffer.BufferType == SECBUFFER_TOKEN) {
    BCryptHashData(hHash, pDirectionGuid, cbDirectionGuid);
    BCryptHashData(hHash, pBuffer->pvBuffer, pBuffer->cbBuffer);
}
```

Windows 11:

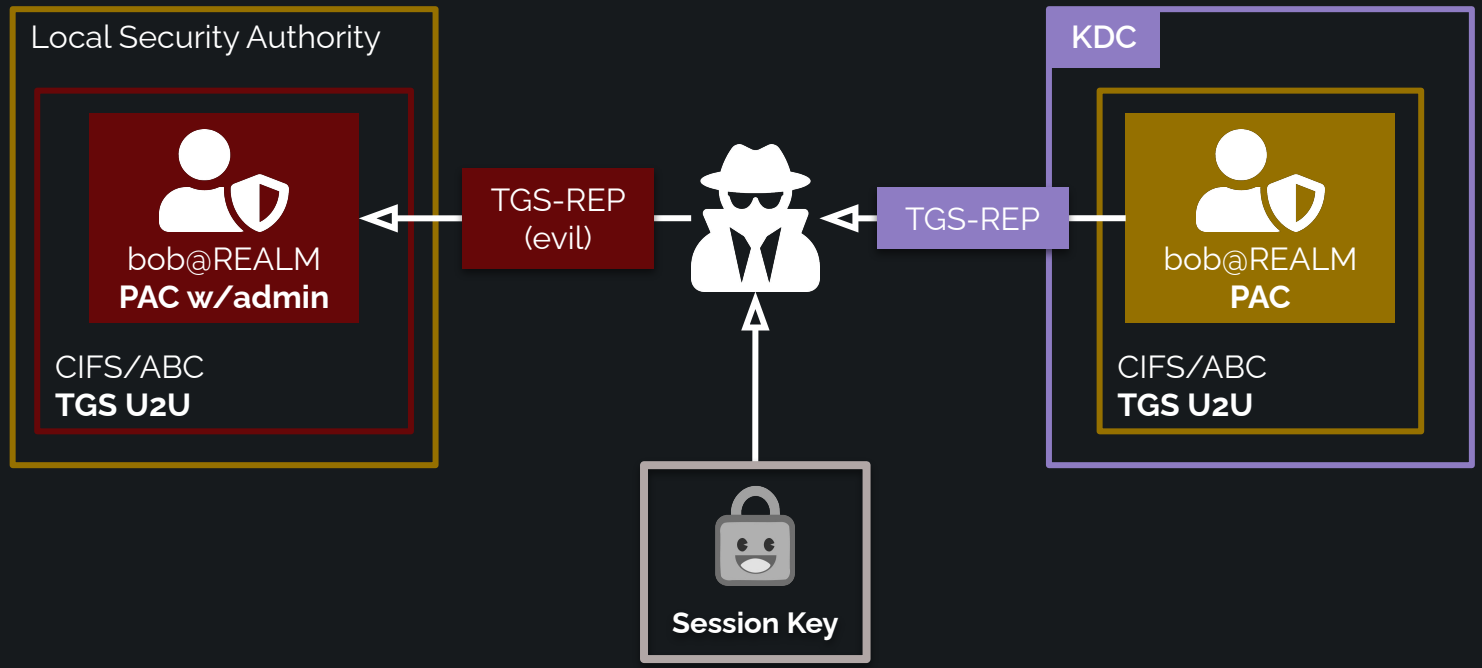
Masking the upper byte out

```
if ((pBuffer.BufferType & ~SECBUFFER_ATTRMASK) == SECBUFFER_TOKEN) {
    BCryptHashData(hHash, pDirectionGuid, cbDirectionGuid);
    BCryptHashData(hHash, pBuffer->pvBuffer, pBuffer->cbBuffer);
}
```

LSA Loopback Library
Danger Zone



Modifying on the Wire



What about Credential Guard ?

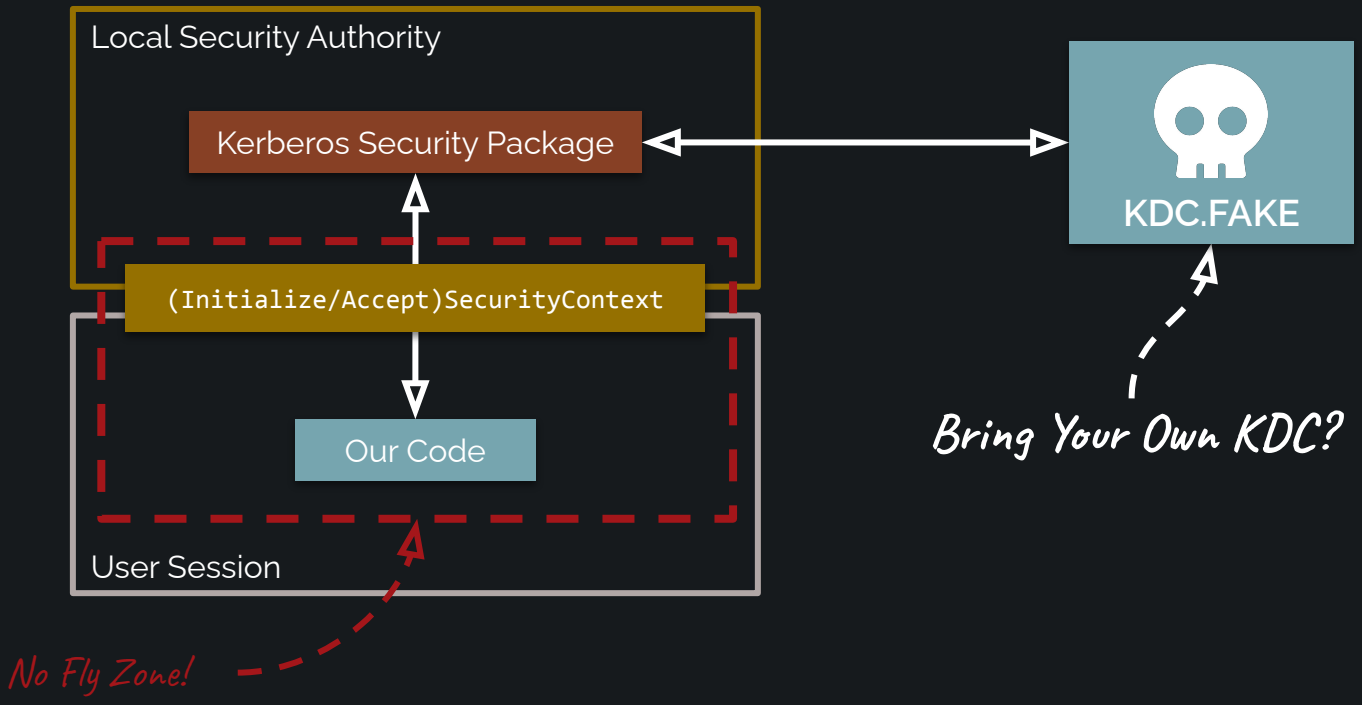
Kerberos Considerations

When you enable Windows Defender Credential Guard, you can no longer use Kerberos unconstrained delegation or DES encryption. Unconstrained delegation could allow attackers to extract Kerberos keys from the isolated LSA process. Use constrained or resource-based Kerberos delegation instead.

https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-considerations

LSA Loopback Library

BYOKDC

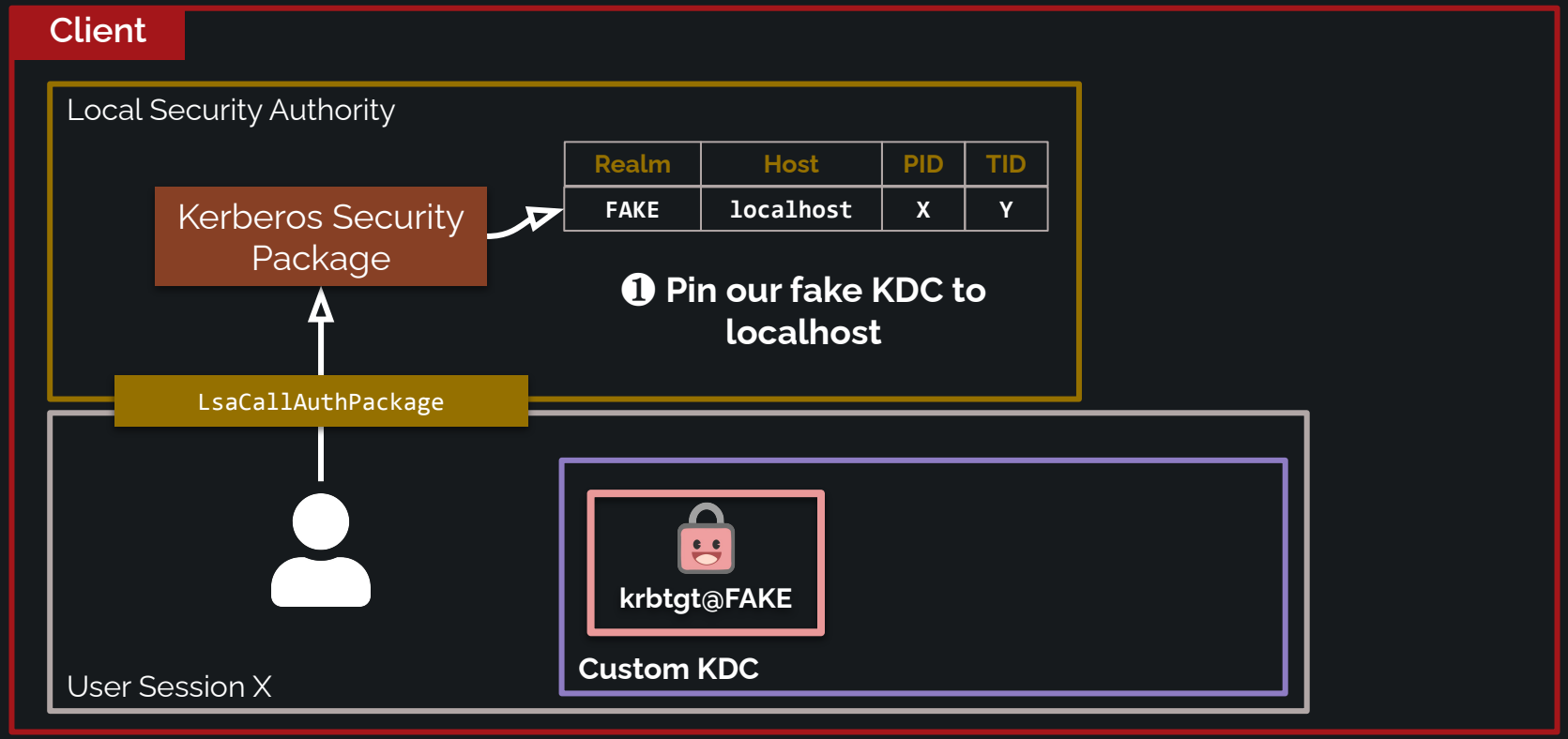


KDC Pinning

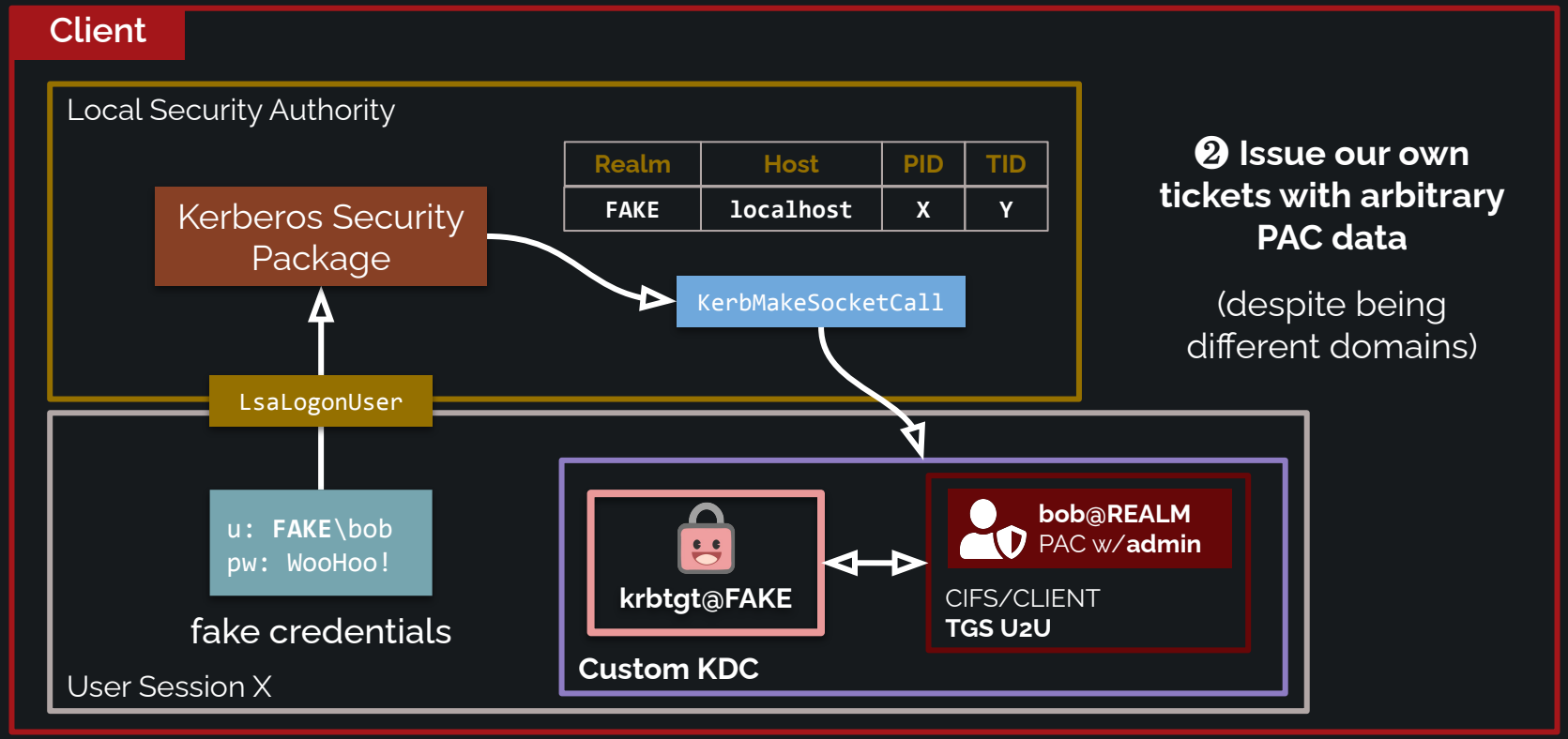
```
struct SECPKG_CALL_PACKAGE_PIN_DC_REQUEST {
    ULONG          MessageType;
    ULONG          Flags;
    UNICODE_STRING DomainName;
    UNICODE_STRING DcName;
    ULONG          DcFlags;
};
```

MessageType can be ***SecPkgCallPackagePinDcMessage*** or ***KerbPinKdcMessage***

TGS-REP Local KDC



TGS-REP Local KDC

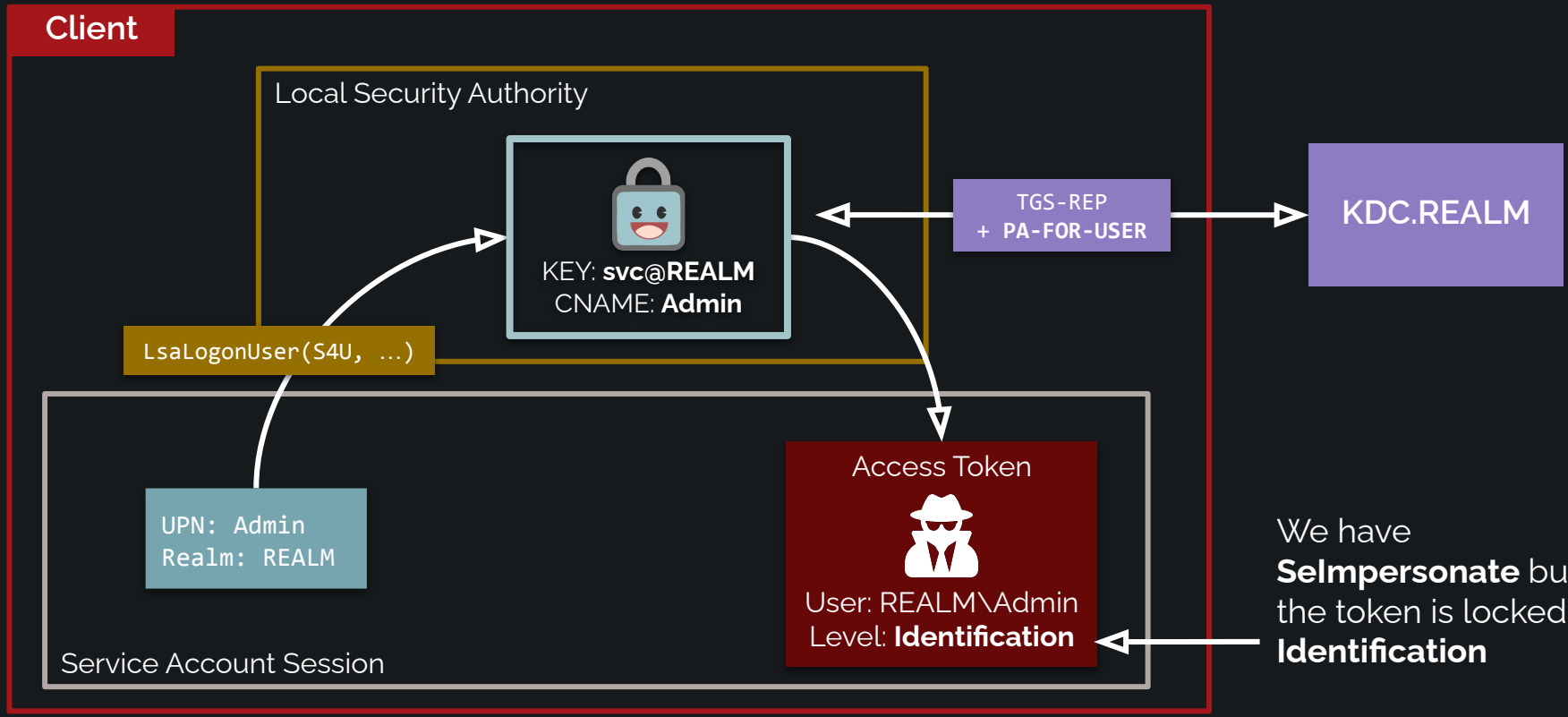


Demo Time

“Security Boundaries”

and where they aren’t

Service Account S4U2Self

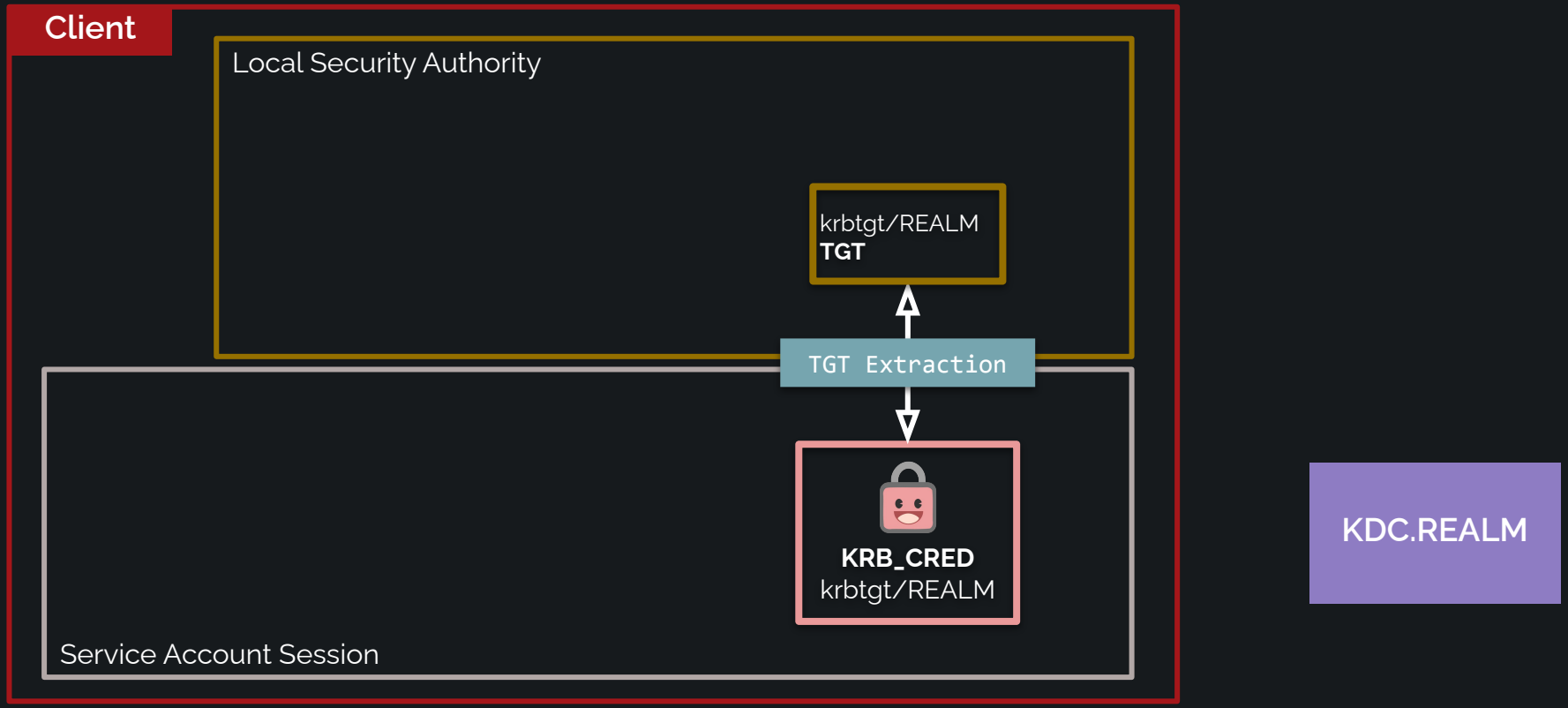


S4U TCB Privilege Check

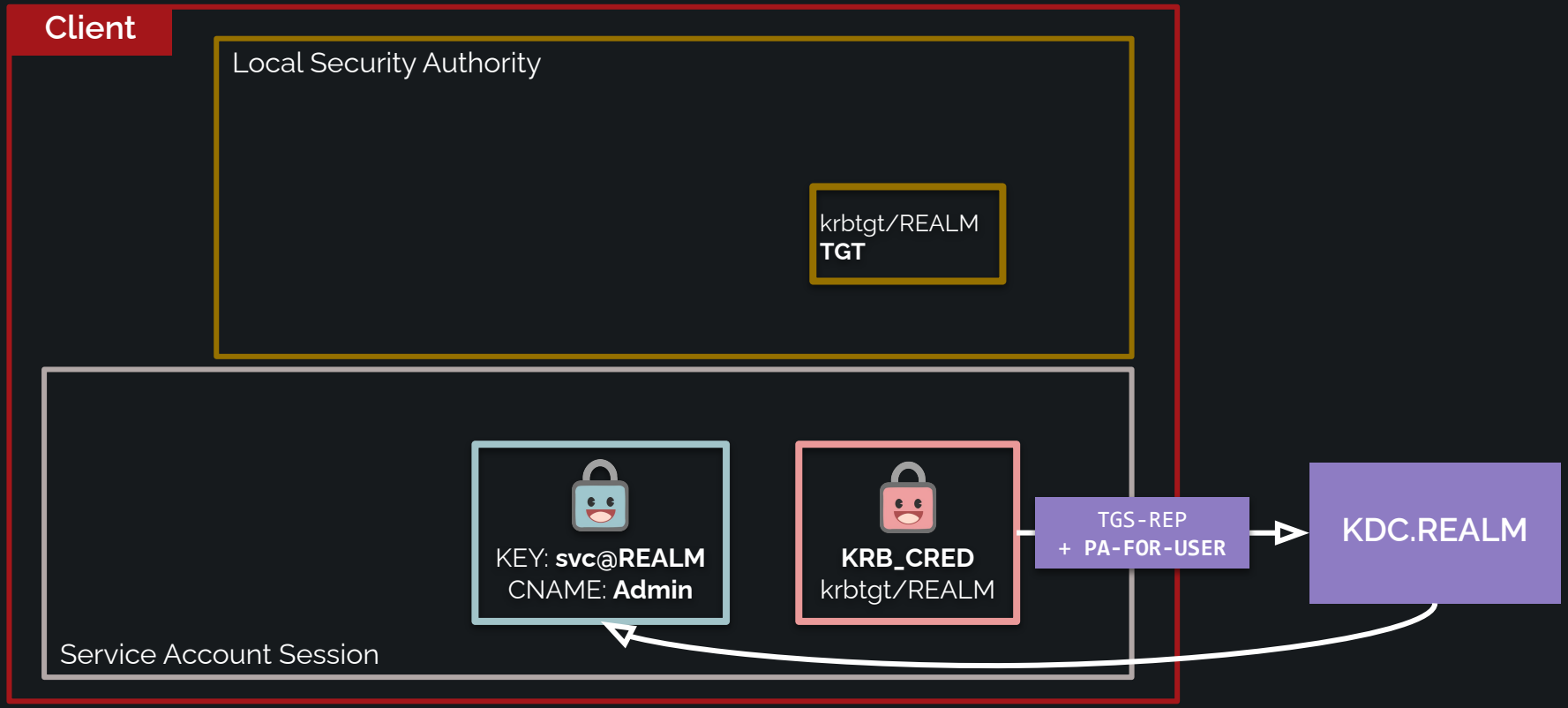
```
KerbCreateTokenFromLogonTicket(...) {
    if (MessageType == KerbTicketLogon || MessageType == KerbTicketUnlockLogon ||
        MessageType == KerbS4ULogon || ...
    ){
        if (!ClientInfo.HasTcbPrivilege)
            PrimaryCredentials->Flags |= PRIMARY_CRED_LOGON_NO_TCB;
    }
}
```

```
LsapAuApiDispatchLogonUser(...) {
    BOOL UseIdentify = PrimaryCredentials.Flags & PRIMARY_CRED_LOGON_NO_TCB;
    LsapCreateV3Token(...
        (UseIdentify ? TokenImpersonation : TokenPrimary),
        (UseIdentify ? SecurityIdentification : SecurityImpersonation),
        &Token
    );
}
```

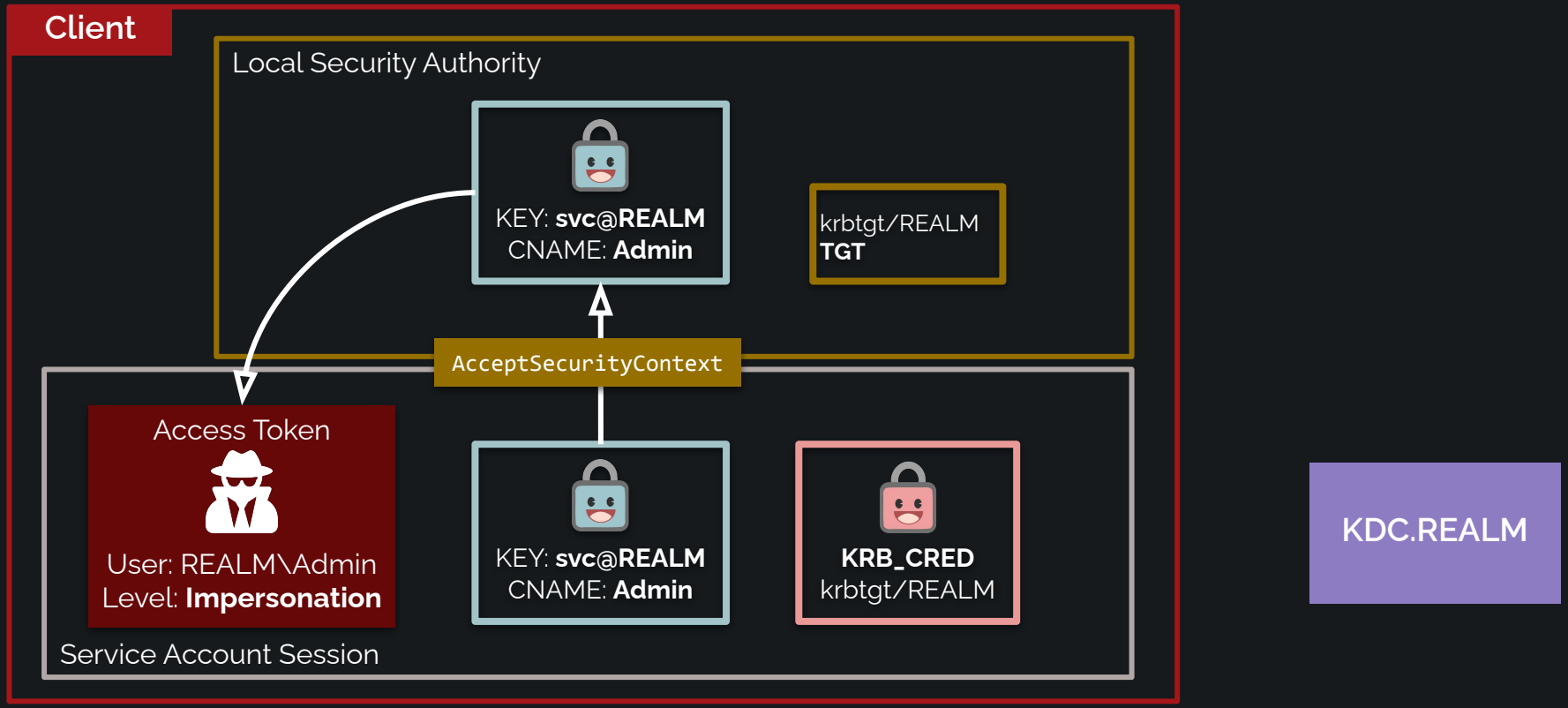
Service Account S4U2Self



Service Account S4U2Self



Service Account S4U2Self



What about UAC ?

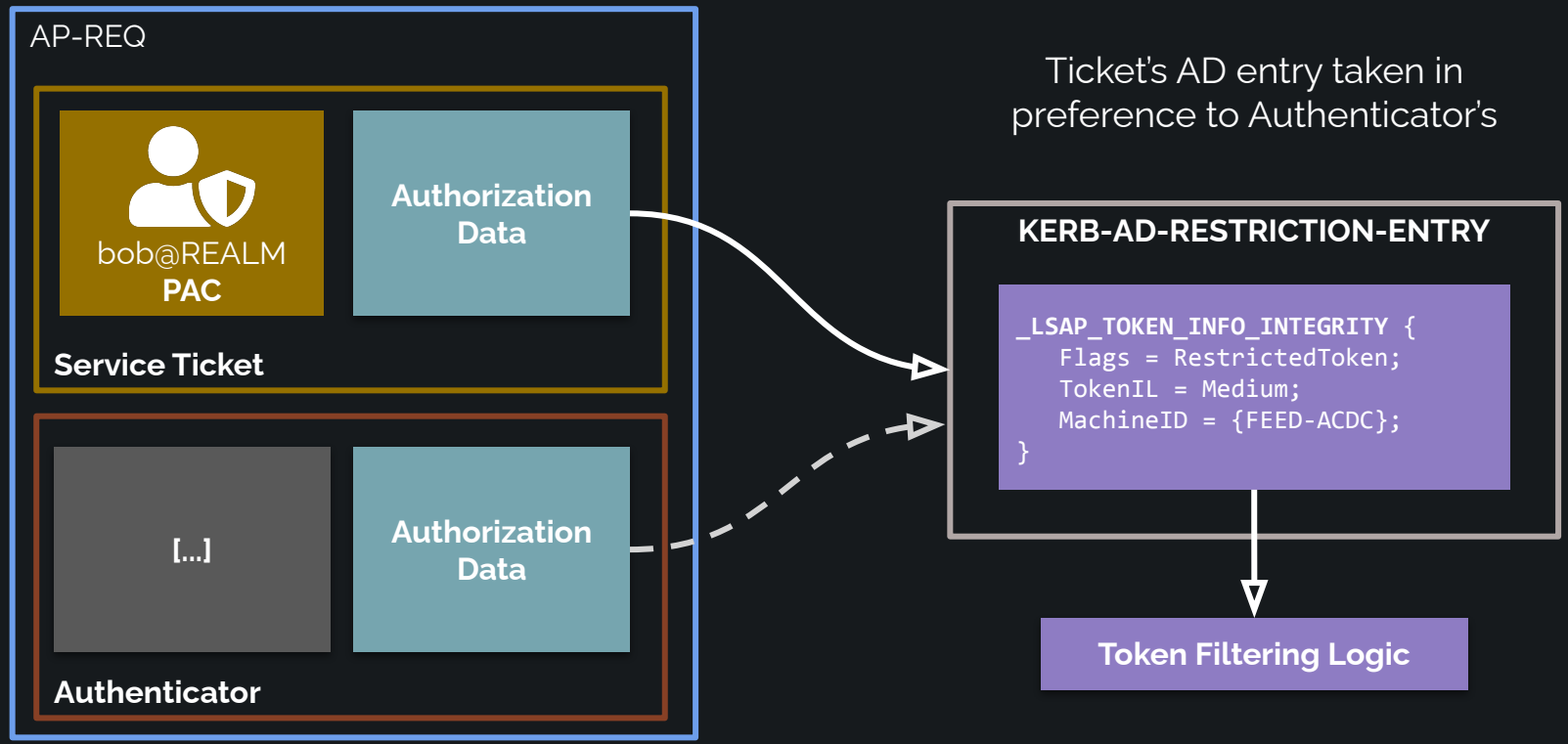
2.2.5. LSAP_TOKEN_INFO_INTEGRITY

The LSAP_TOKEN_INFO_INTEGRITY structure specifies the integrity level information for the client.<7>

```
typedef struct _LSAP_TOKEN_INFO_INTEGRITY {
    unsigned long Flags;
    unsigned long TokenIL;
    unsigned char MachineID[32];
}
```

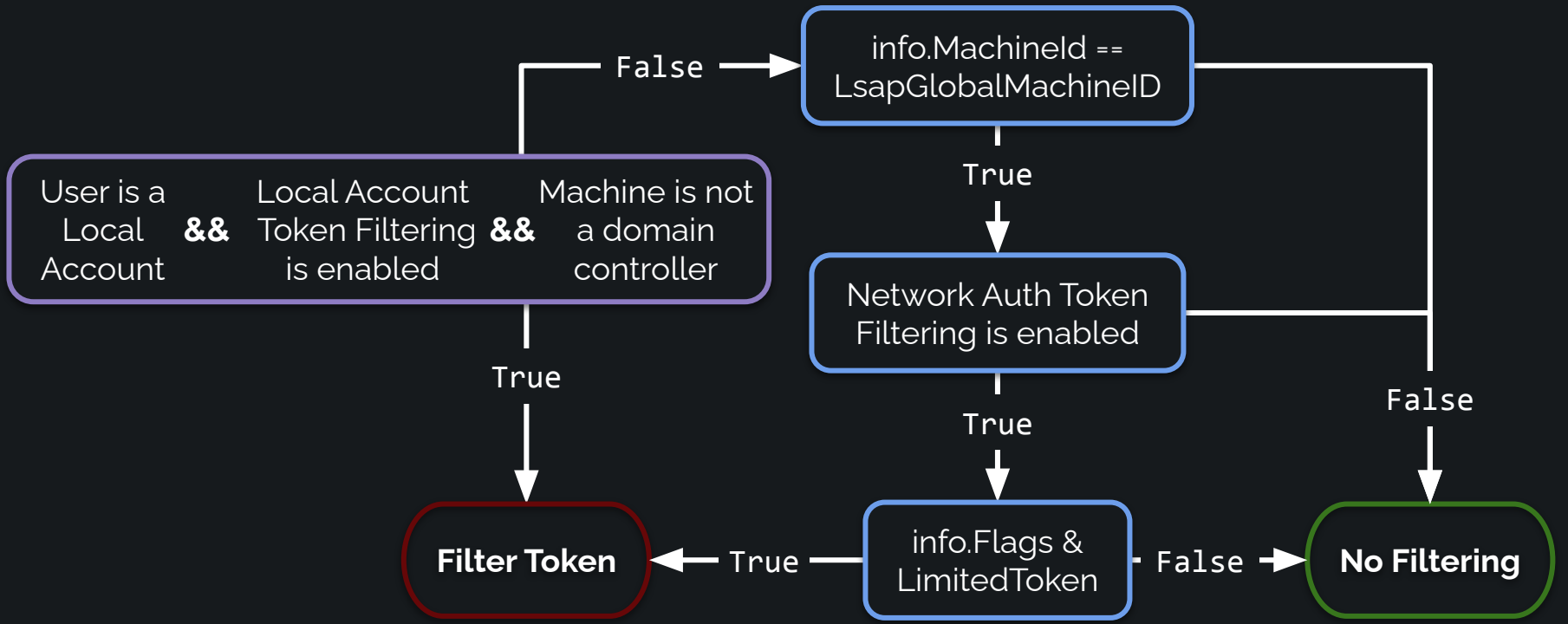
[MS-KILE]: Kerberos Protocol Extensions

Authoriziation Data Entries

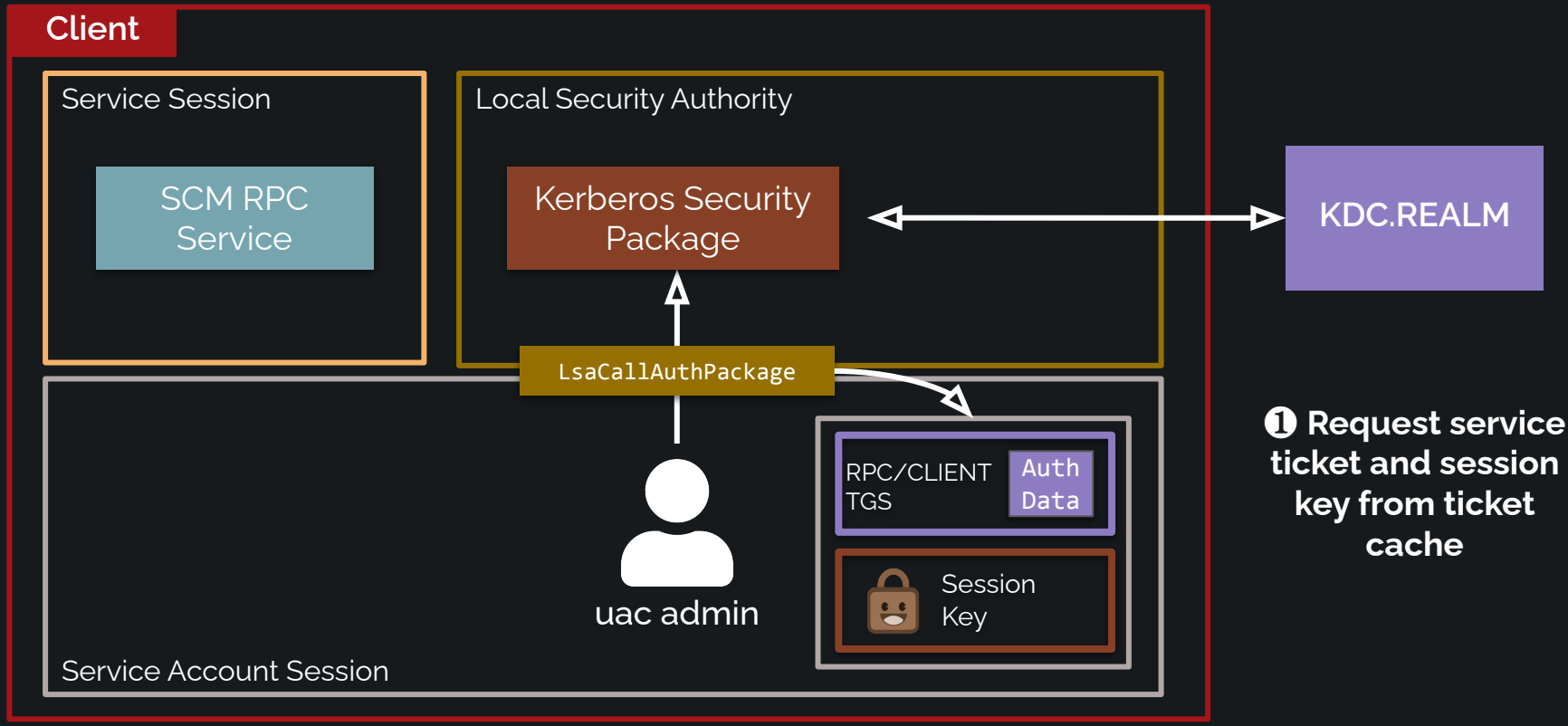


LSA Token Filtering

via `LsaISetSupplementalTokenInfo()`



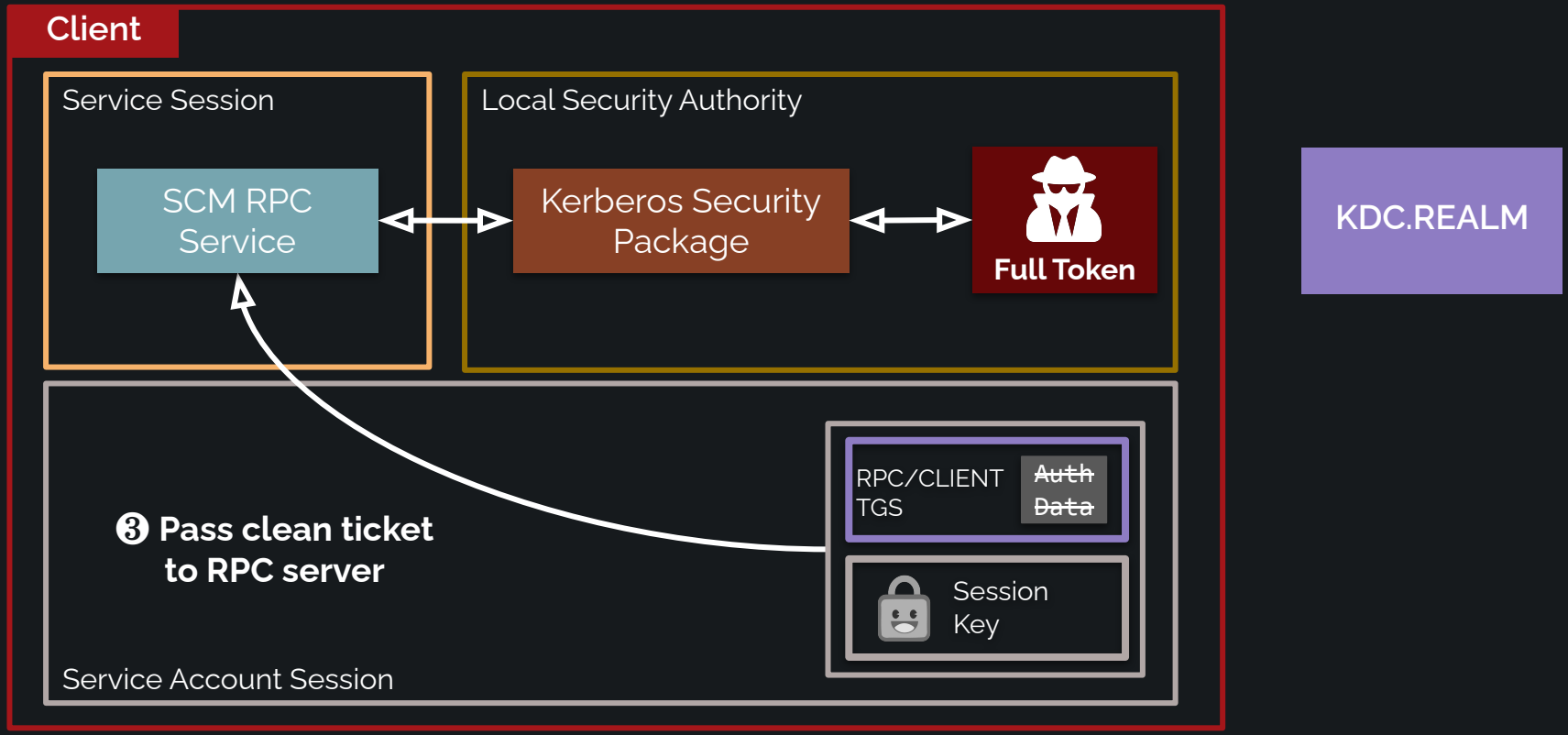
Kerberos UAC Bypass



❶ Request service ticket and session key from ticket cache

Kerberos UAC Bypass

Kerberos UAC Bypass



Demo Time

Wrap Up Time

CVE-2022-35756

Logon Session:

“SYSTEM” Equivalent

`SeTcbPrivilege || SYSTEM || LOCAL/NETWORK SERVICE`

Credentials Handle:

`cred->Flags & SECPKG_CRED_ATTR_PAC_BYPASS`

(auto) **AcquireCredentialsHandle** w/
`SECPKG_CRED_INBOUND && NT AUTHORITY\SERVICE && !KerbGlobalValidateKDCPACSignature`

(manual) **SetCredentialsAttributes** w/SeTcbPrivilege

ASC Context Flags:

~~`context->Flags & ASC_RET_USE_SESSION_KEY`~~

Mitigation Thoughts

- **Enable *KerbGlobalValidateKDCPACSignature***
  - Prevent “NT AUTHORITY\SERVICE” SID from bypassing PAC verification
  - Doesn’t prevent “LOCAL/NETWORK SERVICE” or “SYSTEM” though
- **Force Kerberos Armoring / FAST**
  - Makes it harder to tamper with network traffic
- **Enable Credential Guard**
  - Block trivial access to TGT session keys
- **Build Kerberos firewall rules**
  - Block access to KDCs outside an approved list

Detection Thoughts

Security -> Logon/Logoff -> Special Logon -> **Event 4672**

```
Special privileges assigned to new logon.

Subject:

Security ID:  REALM\bob
Account Name:  bob
Account Domain:  REALM
Logon ID:  0x4b842

Privileges:

SeSecurityPrivilege
SeTakeOwnershipPrivilege
SeLoadDriverPrivilege
...
```

Limitations of Time

NtApiDotNet (tooling used in presentation)
https://github.com/googleprojectzero/sandbox-attacksurface-analysis-tools

UAC Bypass Trickery
https://www.tiraniddo.dev/2022/03/bypassing-uac-in-most-complex-way.html

Remote Credential Guard Code Execution
https://bugs.chromium.org/p/project-zero/issues/detail?id=2271

AppContainer Escapes
https://bugs.chromium.org/p/project-zero/issues/detail?id=2273

LSASS Impersonation Check Failures
https://bugs.chromium.org/p/project-zero/issues/detail?id=2278

Service Account S4U Elevation
https://cyberstoph.org/posts/2021/06/abusing-kerberos-s4u2self-for-local-privilege-escalation/

Acknowledgements

Elad Shamir | @**elad_shamir**

Benjamin Delpy | @**gentilkiwi**

Will Schroeder | @**harmj0y**

Charlie Clark | @**exploitph**

Christoph Falta | @**cfalta**

One Last Thing !

Questions ?