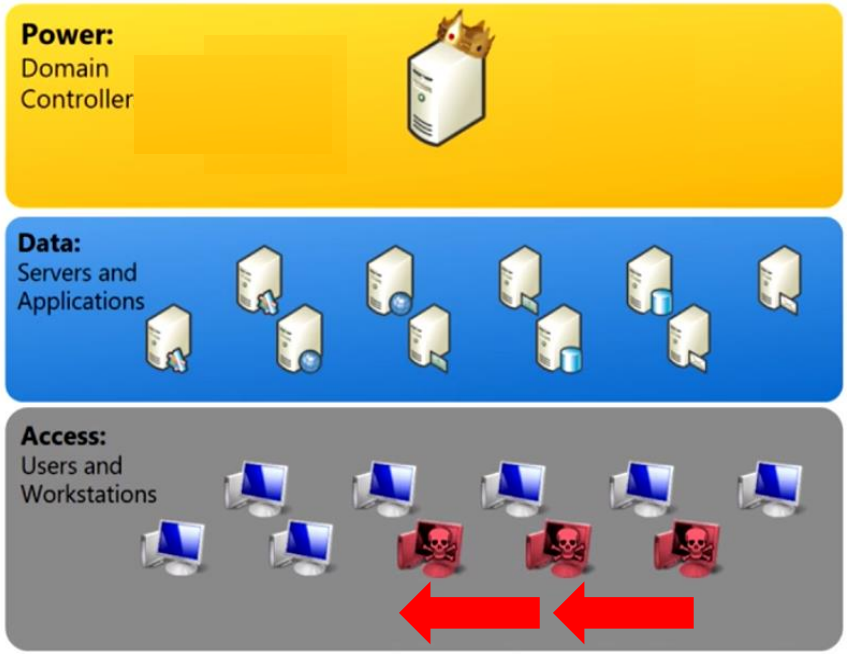
Lateral Movement

Lateral Movement



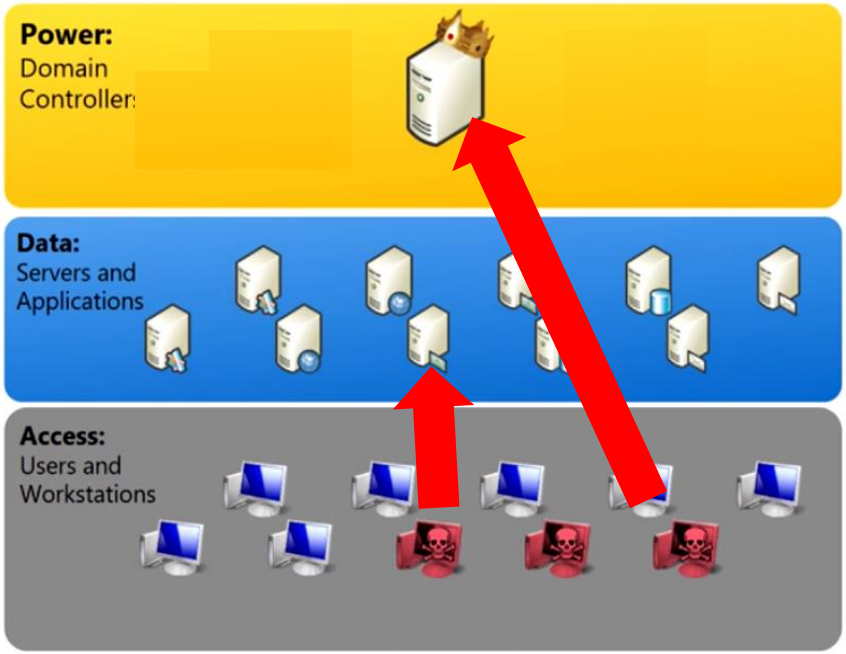
Techniques to **enter** and **control** remote systems

Requires exploring the network to find their target and subsequently gaining access to it.

Reaching their objective often involves pivoting through **multiple systems** and **accounts** to gain.

Adversaries might install **their own** remote access tools, or use legitimate credentials with **native** network and operating system **tools**, which may be stealthier ("living off the land").

Key Objective



Techniques to **enter** and **control** remote systems

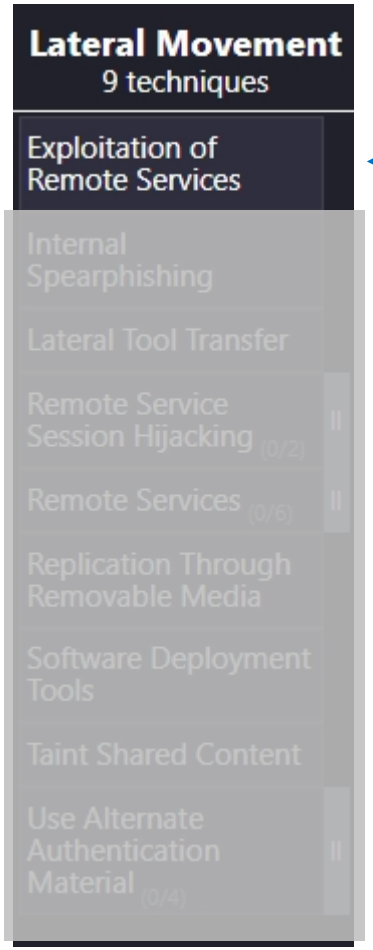
Requires exploring the network to find their target and subsequently gaining access to it.

Reaching their objective often involves pivoting through **multiple systems** and **accounts** to gain.

Key objective: ability to access **critical services**

(for **Impact**)

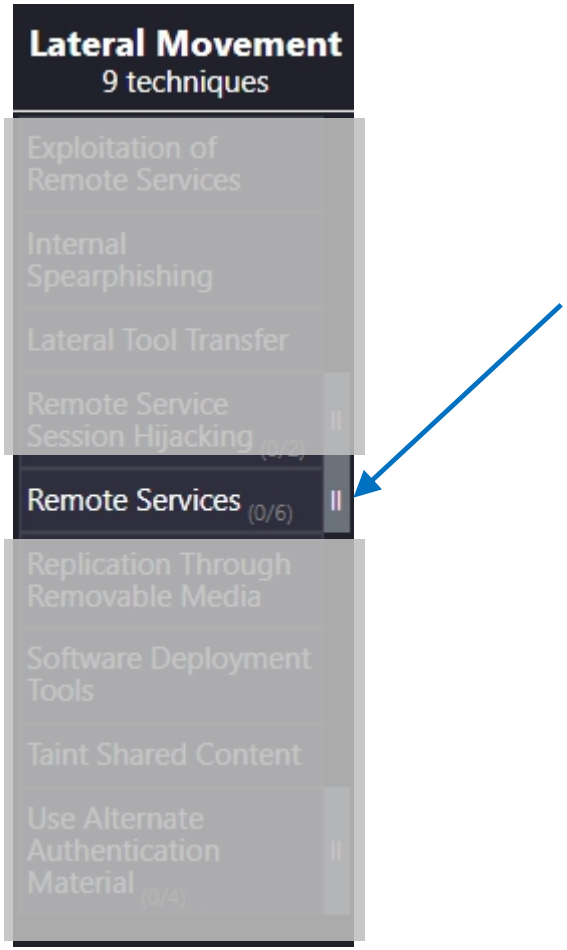
Technique: Vulnerability Exploitation



Adversaries may **exploit** remote services to gain unauthorized access to internal systems once inside of a network.

Exploitation of a **software** vulnerability occurs when an adversary takes advantage of a programming error…to execute adversary-controlled code.

Technique: Remote Services (I)

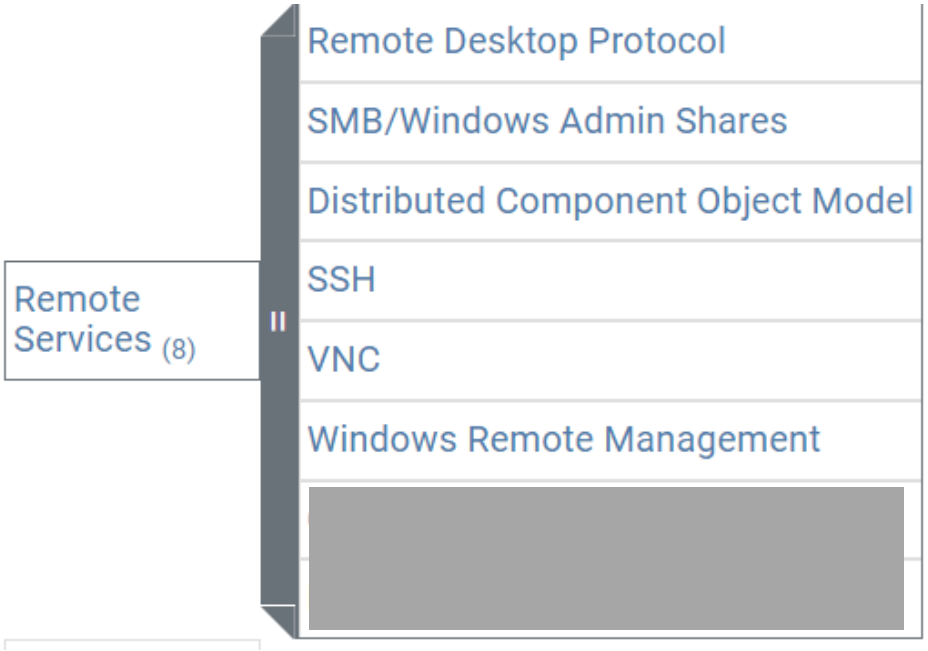


Adversaries may use **Valid Accounts** to log into a service that accepts remote connections

The adversary may then perform actions **as the logged-on user.**

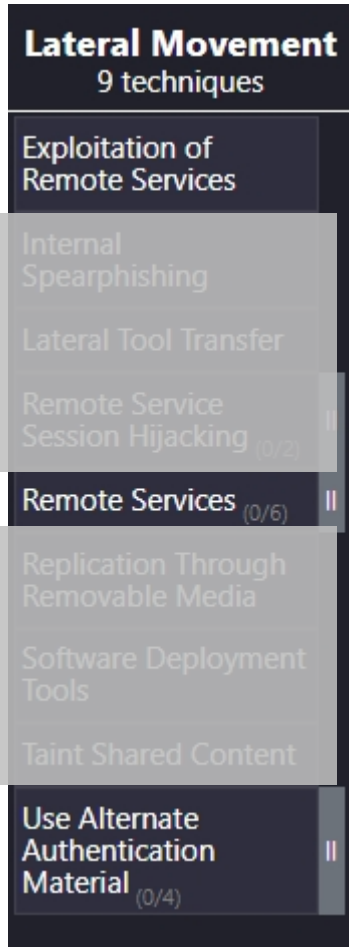
```
psexec \\RemoteNode
                -u Username -p Password
                CmdToExecute
```

Technique: Remote Services (II)



MANY other invocation styles

Technique: Use Alternate Auth Material

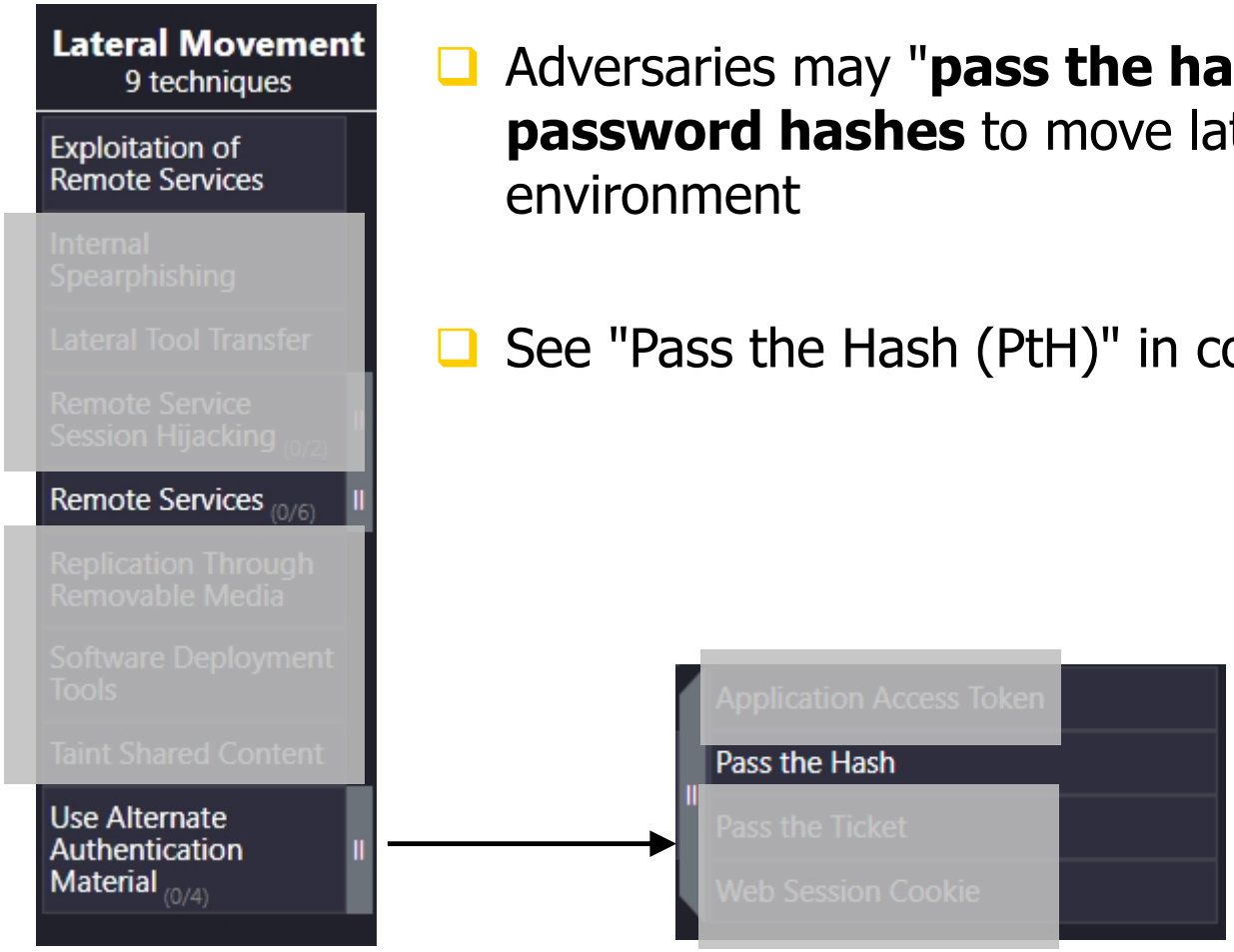


Adversaries may use alternate authentication material, such as **password hashes**, [redacted] in order to move laterally within an environment

Alternate authentication material is **legitimately** generated by systems **after** a user or application successfully authenticates with a Valid Account

By **stealing** alternate authentication material, adversaries are able to **...** authenticate to systems **without knowing the plaintext password** ...

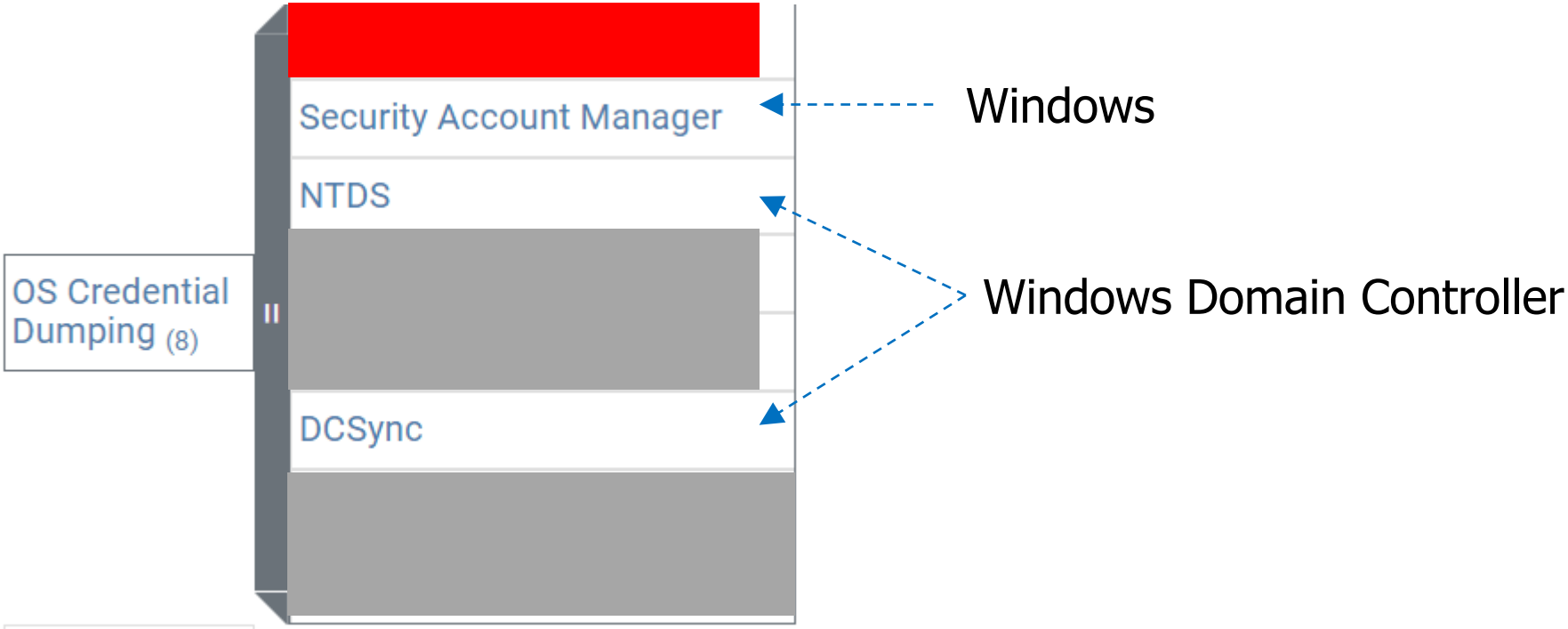
Pass the Hash



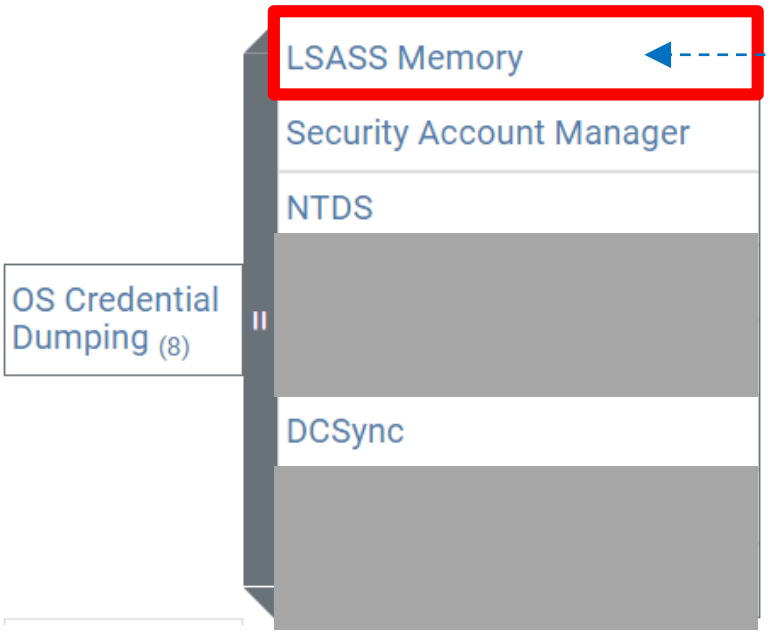
- Adversaries may "**pass the hash**" using **stolen password hashes** to move laterally within an environment
- See "Pass the Hash (PtH)" in companion website

Stealing Password Hashes (REMIND)

Password hashes of
all the accounts



Credentials in Memory

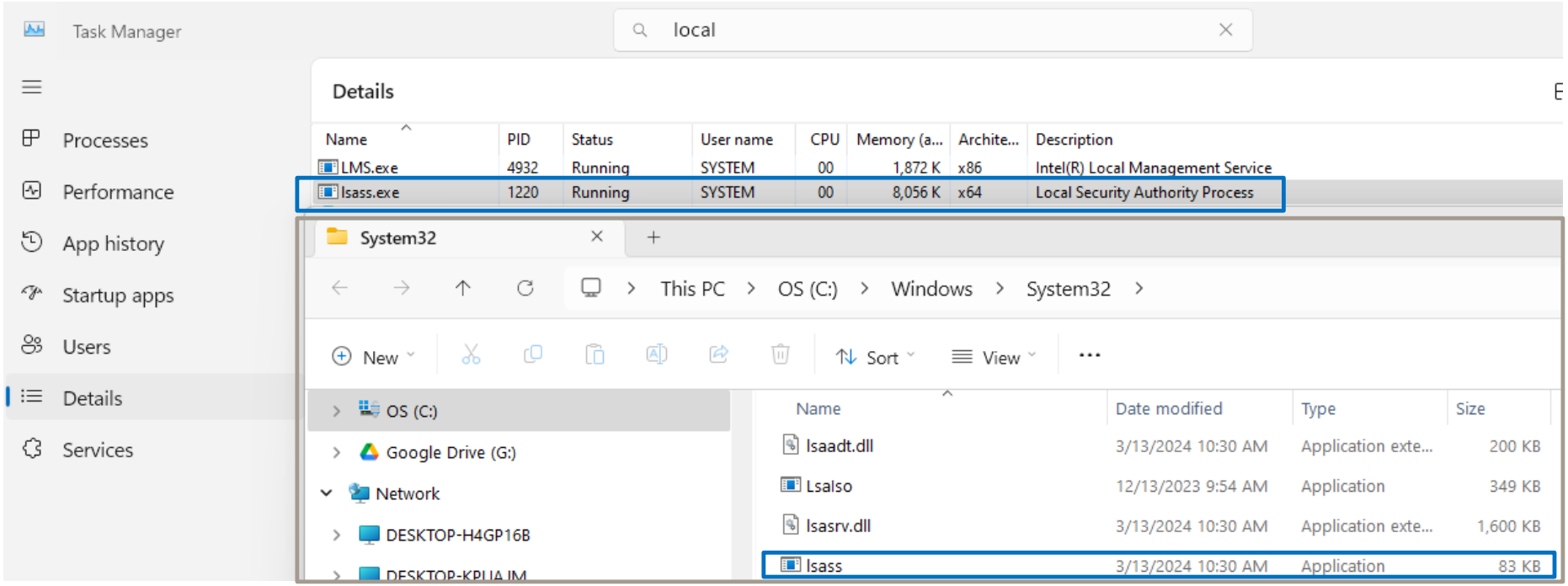


After a user logs on, the system generates and stores a variety of credential materials in LSASS process **memory**.

These credential materials **can be stolen by an administrative user or SYSTEM** and used to conduct **Lateral Movement**

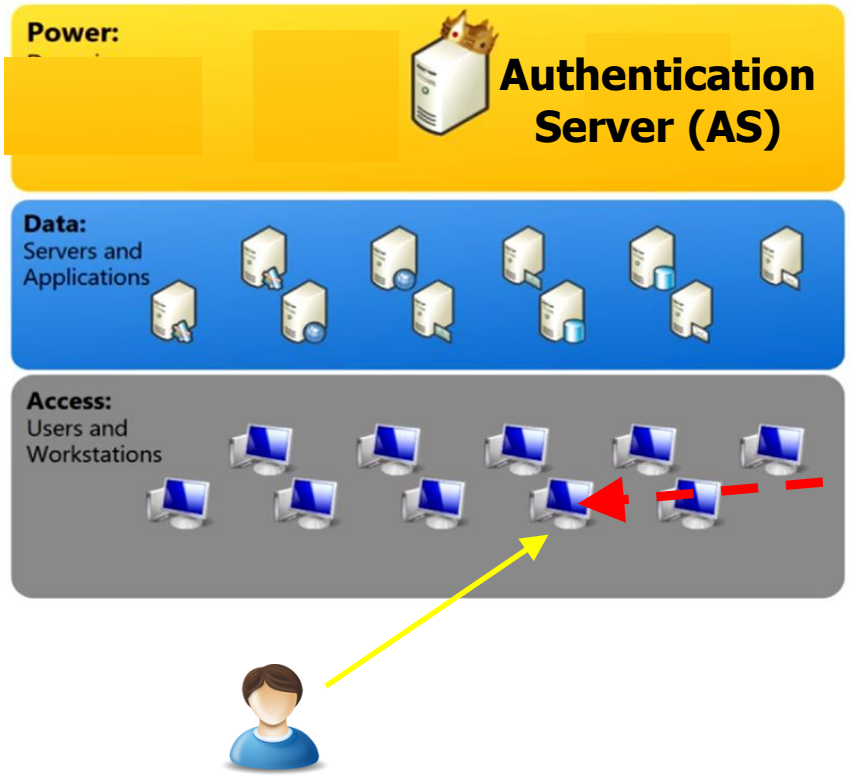
- Which credentials are kept in memory?
- **Extremely important** in practice

LSASS content (in a nutshell)



- For every **logged on account**:
  1. Password hash
  2. Tickets and Session keys

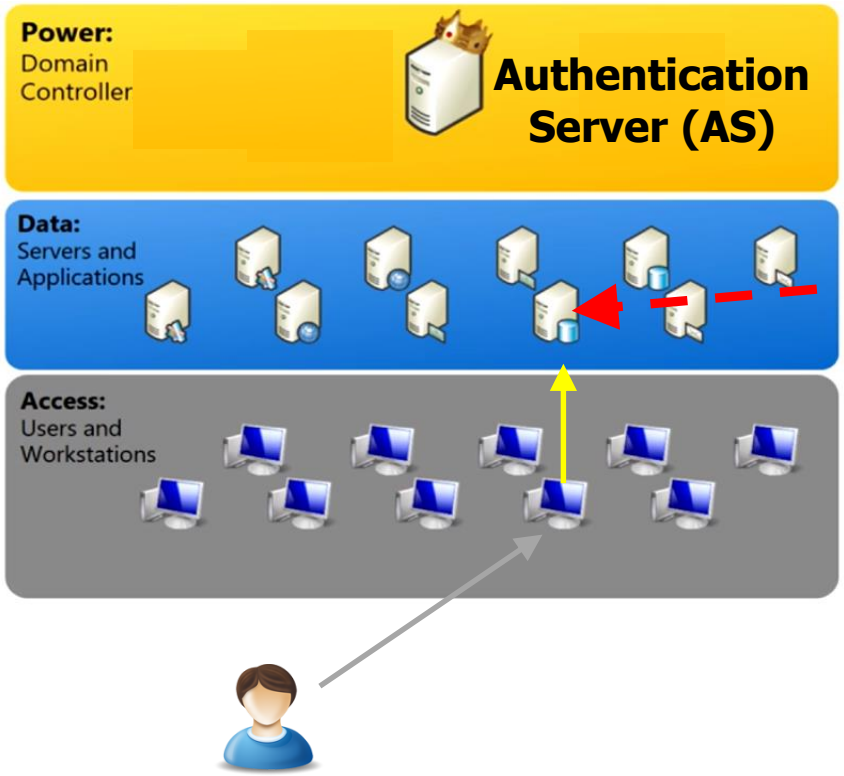
LSASS Content: Interactive Logon



User Account U

- U, H(PWD-U)
- TGT(U) + $K_{U\text{-}TGS}$
- ST(U, wksname) + $K_{U\text{-}WKS}$
- ST(U, S1) + $K_{U\text{-}S1}$
- ST(U, S2) + $K_{U\text{-}S2}$
- …

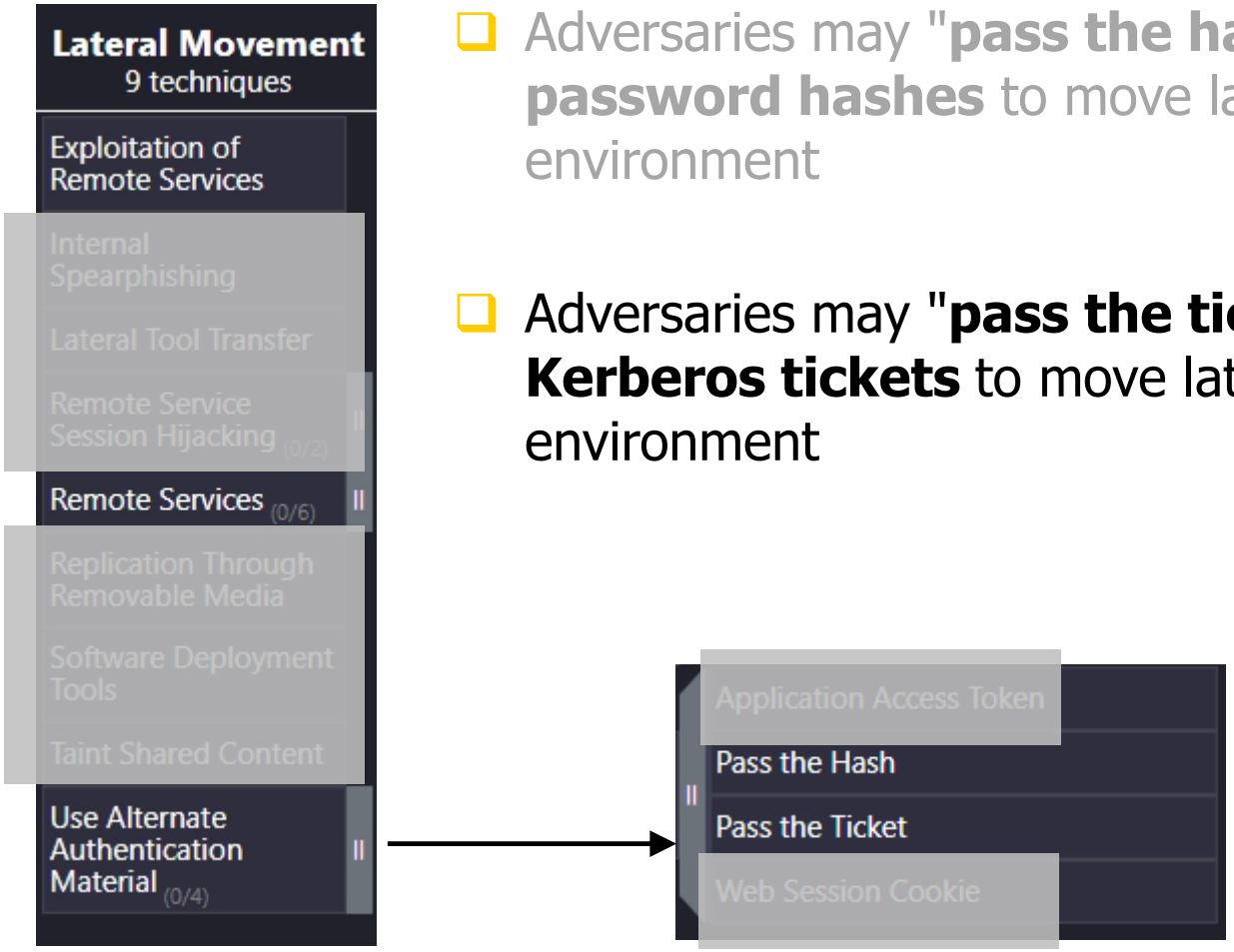
LSASS Content: Network Logon



Service Account US

- **Nothing for supporting U**
- US, H(PWD-US)
- TGT(US) + K_{US-TGS}
- ST(US, hostname) + K_{US-H}

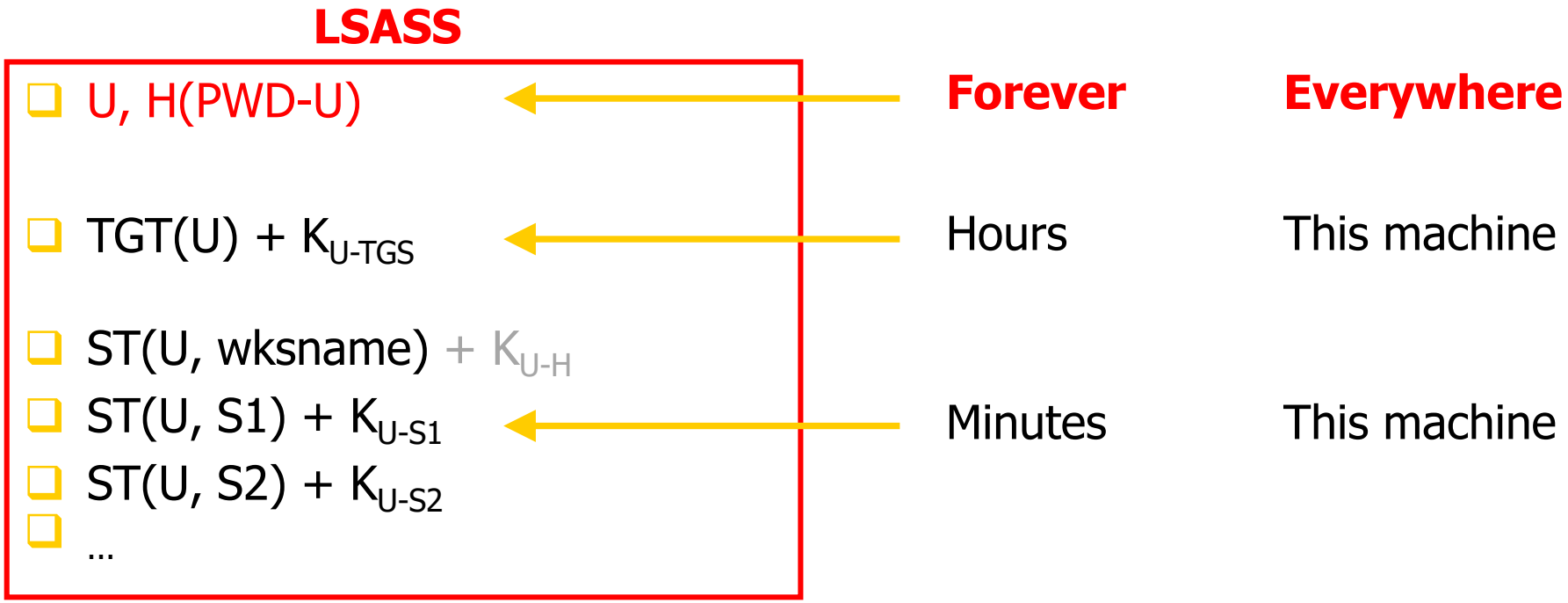
Pass the Ticket



- Adversaries may "**pass the hash**" using **stolen password hashes** to move laterally within an environment
- Adversaries may "**pass the ticket**" using **stolen Kerberos tickets** to move laterally within an environment

For how long? Where?

Dedicated tools for stealing from LSASS
(e.g., Mimikatz, Rubeus, Empire)



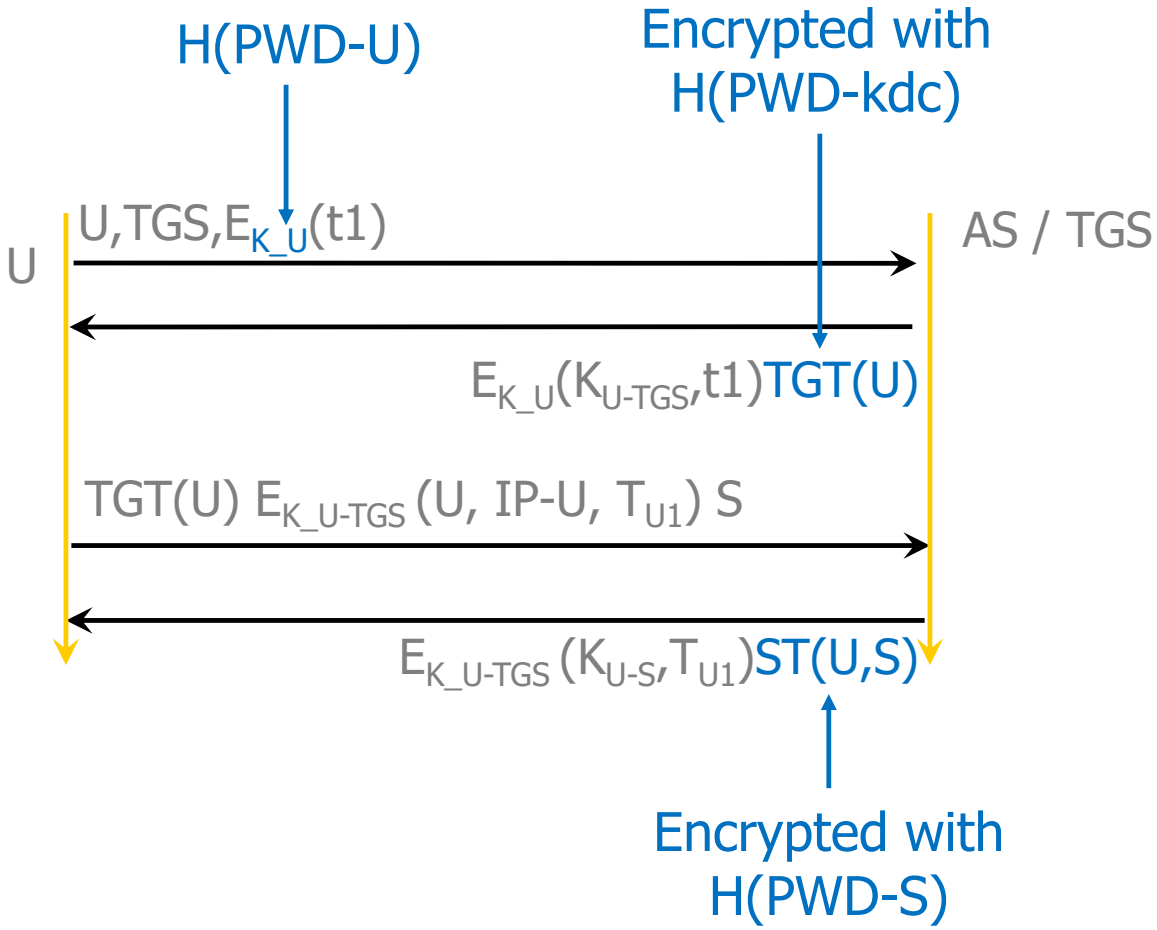
Keep in mind

- SYSTEM process on machine W can steal:
  - Password hashes — all local accounts of W
  - Password hashes — all domain accounts **currently logged on** W
  - TGT — all domain accounts **currently logged on** W
  - ST(-,S) — all domain accounts **currently logged on** W

- Password hashes:
  - **Do not expire**
  - **Can be used anywhere**
- Tickets:
  - Expire in hours / minutes
  - Can be used **only** from where stolen

More offensive use of password hashes

REMIND


H(PWD-U)
Encrypted with
H(PWD-kdc)
U
U,TGS,E_K_U(t1)
AS / TGS
E_K_U(K_U-TGS,t1)TGT(U)
TGT(U) E_K_U-TGS (U, IP-U, T_U1) S
E_K_U-TGS (K_U-S,T_U1)ST(U,S)
Encrypted with
H(PWD-S)

Overpass the Hash

- H(PWD-U) → Obtain TGT(U) from KDC
  - Impersonate U to KDC (thus everywhere)

- Conceptually simple
- Many tools (e.g., Rubeus, Mimikatz, Empire)

Silver Ticket

- H(PWD-S) → Forge ST(U,S) for any U **without contacting KDC**
  - Impersonate U on S
  - Conceptually not particularly useful (if you have H(PWD-S) then probably you can already do what you want on S)
  - Practically useful (simplifies technical steps)

- Conceptually simple
- Many tools (e.g., Rubeus, Mimikatz, Empire)

Golden Ticket

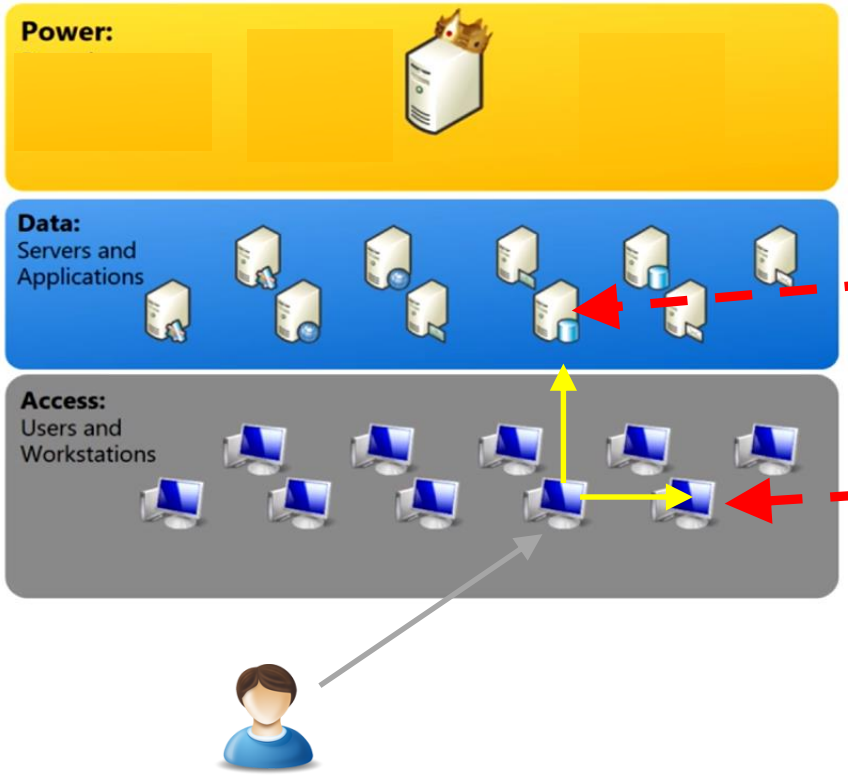
- H(PWD-krbtgt) → Forge TGT(A) for any A **without contacting KDC**
  - Impersonate any account anywhere

- Conceptually simple
- Many tools (e.g., Rubeus, Mimikatz, Empire)

Remote Administration: Important detail

Remote Administration with Network Logon

Network logon does **not**
expose credentials in LSASS



- **Nothing for supporting U**
- **…**

- **Nothing for supporting U**
- **…**

Logon Types (and bad news)

- Logon types:
  - **Interactive** Logon
  - **Network** Logon
    - Only logon type that does not keep credentials in memory
  - **5 more types**
- Choice **not arbitrary**: it depends on what you need to do

- Fact:
  - Many tools for **remote administrative access** are **not** based on network logon
  - Credentials will be in LSASS of the remote node

Common Path to Catastrophe

Common Path to Catastrophe (I-a)

1. **Machine has malware with SYSTEM privilege**
   - User Local Admin and executes malware
   - Malware not associated with local SYSTEM + Privilege escalation exploit
   - Attacker has credentials of Local Admin User

Common Path to Catastrophe (I-b)

1. **Machine has malware with SYSTEM privilege**

- Credentials from SAM (only local accounts)
- More lateral movement steps feasible
  - They will succeed **only** on machines with local accounts that have **the same credentials**

Common Path to Catastrophe (II)

1. Machine has malware with SYSTEM privilege
2. Another (Local **or Domain**) User operates on this machine:
   - Logon type that exposes credentials

- More credentials **from LSASS**
- More lateral movement steps feasible
  - For **domain** accounts,
    they will likely succeed on **many other machines**

Common Path to Catastrophe (III)

1. Machine has malware with SYSTEM privilege
2. **Domain** User with **High Privilege** operates on this machine:
   - Logon type that exposes credentials

- More credentials from LSASS: **game over!**
  - Credentials for Impact on important services
  - Credentials for reading DC
  - ...

Why IT catastrophes occur

- Some of the **key basic reasons:**
  - Local accounts configured as local **administrator**
  - Domain accounts with low privilege configured as local **administrator**
  - Many machines with the **same** local administrator pwd
  - High privilege accounts used for **daily** work
  - High privilege accounts used for **remote** administration

- No rocket science

Tiered administration (A wish…)

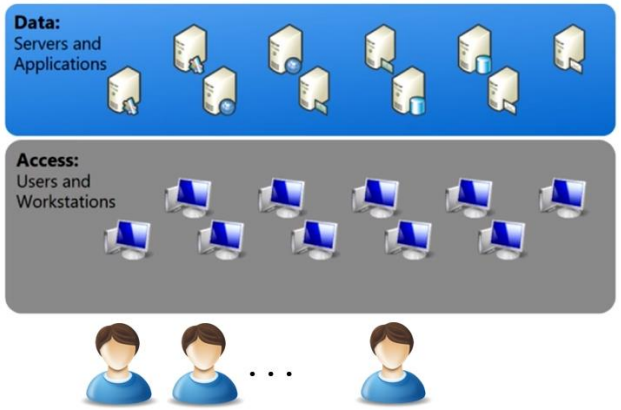
- IT objects are conceptually grouped in **3 tiers**
  1. Domain Controller, Users+Groups+Devices that can manage DC
  2. Critical services, Users+Groups+Devices that can manage them
  3. Everything else

- **Never** access a **resource in a tier** from a **resource in a higher tier**
  - Never logon on a user workstation with credentials that allow managing a critical service
  - Never logon on a user workstation from a workstation that allows managing the domain controller (or a critical service)
  - …

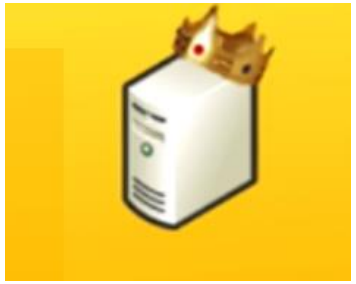
Abuse of Access Rights in AD

AD Access Rights (in a nutshell of a nutshell)

Domain Controller (REMIND)



DOMAIN CONTROLLER



- **Centralized** repository (**Domain Controller**) describes all **IT entities**:
  - All **identities** and their **credentials**
  - All **resources**
  - All **access rights** of identities to resources (ACLs)

AD Objects

- Every entity is an **object**
- Each object has a **name** and a set of **named attributes**
  - Rules specified in LDAP
- **Type** of the object implicitly encoded in certain attributes
- Different objects may have different sets of attributes (even if objects are the same type)

Attributes of a User Account Object

- Object name

```
CN=BARTOLI ALBERTO[5943],OU=042000,OU=personale,DC=ds,DC=units,DC=it
```

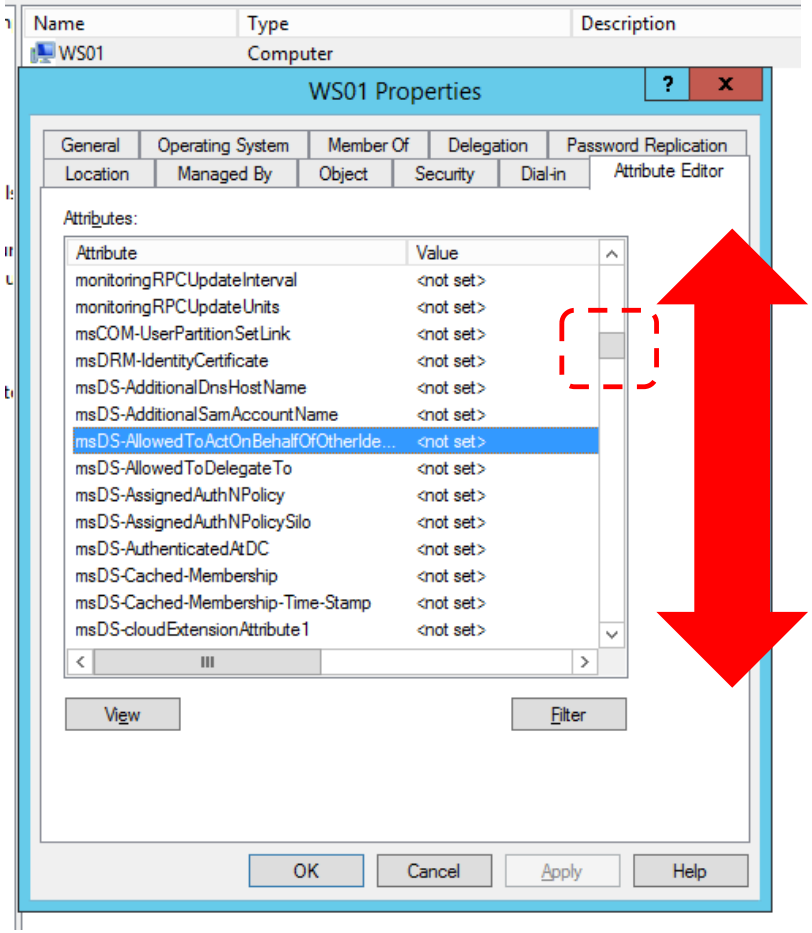
- Object type

```
objectCategory DN      1
        CN=Person,CN=Schema,CN=Configuration,DC=ds,DC=units,DC=it
objectClass    OID     4 top;person;organizationalPerson;user
```

- Some more attributes (**60 total**)

```
accountExpires          Integer8           1 0x0
lastLogonTimestamp      Integer8           1 2/10/2023 13:22
mail                    DirectoryString    1 bartoli.alberto@units.it
mAPIRecipient           Boolean            1 FALSE
name                    DirectoryString    1 BARTOLI ALBERTO [5943]
```

Attributes of a Computer Account Object



Hhmmm…

- Who can **create** / **delete** **AD objects**?
- Who can **write/modify** their **attributes**?
- Who can **read** their **attributes**?

- **Crucial** issue

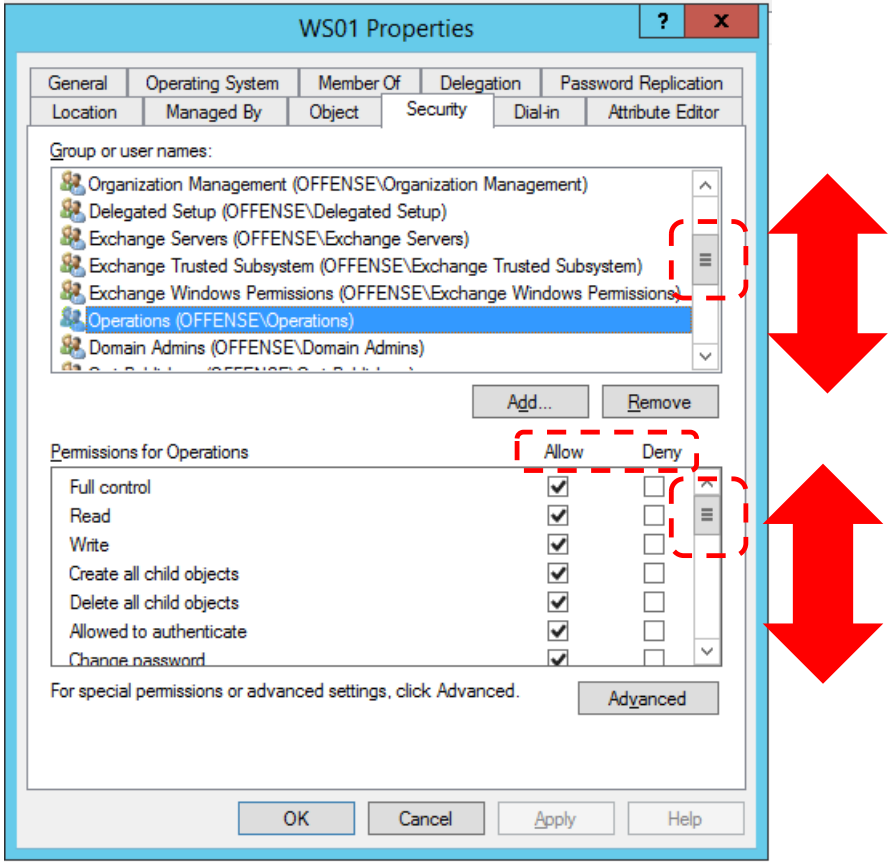
Access Control for AD Objects

- AD Objects have an **ACL** as any other resource
- **Domain Controller** is their **Resource Manager**

- **Write** operations can be specified at the **attribute** level
  - Obj-B, Attr-Y can be modified only by objects having a certain Attr-X

- **Very complex:**
  - ACL structure (Allow / Deny, Inheritance, ...)
  - Resource grouping (multiple, overlapping,...)

ACL of a Computer Account Object



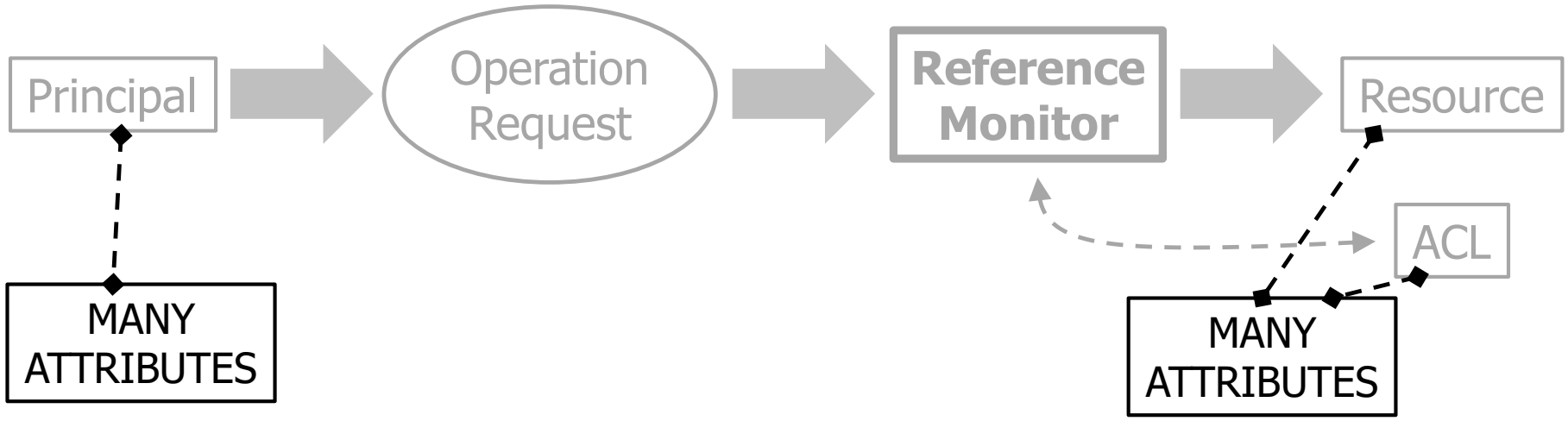
MANY Default Rules

- **Every account can read every attribute of every object**
  - Except for very few special cases (passwords, private keys,...)

- **Predefined Groups** with **predefined Access rights**
  - `Domain Admins` and `Account Operators` can create/delete accounts and reset passwords
  - `Domain Admins` group can modify the composition of the `Account Operators` group
  - `Backup Operators` group can back up and restore files and directories on domain controllers
  - ...

Fact #1

Extremely difficult
to have an accurate understanding of
"**Who can do what on which resources**"



Fact #2

Extremely difficult
to have an accurate understanding of
"**Who can do what on which resources**"

Presence of **hidden** and **unintended** paths to
critical resources
is **very likely**

Fact #3

Every account
can read every attribute
of every object

Attackers can discover many unintended paths that Defenders are unaware of

Huge offensive value

AD Attack Paths

Domain Policy Modification

- ...
- Privilege escalation

Techniques for gaining higher-level permissions on a **system** or a **network**

- **Domain policy modification**
  Adversaries may **modify** the configuration settings of a domain to **escalate privileges** in domain environments...

  Since domain configuration settings control many of the interactions within the Active Directory (AD) environment, there are a **great number of potential attacks** that can stem from this abuse.

Example 1 (hypothetical but realistic)

- U-X, U-Y ∈ Attacker-Credential-Set
- U-X has access right "`administrator of machine group A`"
- U-Y has access right "`add a PC to machine group A`"

- Attacker can:
  - Insert **lots of PCs** in group A (by operating as U-Y)
  - Start monitoring LSASS of **all** those PCs (by operating as U-X)

  - When a server/domain administrator executes a process...bingo!

Example 2 (hypothetical but realistic)

- "`Backup Operators`" is a group that can read data from all servers (by default including Domain Controllers)

- U-X, U-Y ∈ Attacker-Credential-Set
- U-X has access right "`add a user to group Backup Operators`"

- Attacker can:
  - Insert U-Y in "`Backup Operators`"
  - Steal all data that can be read by that group
    - By default even data on Domain Controllers...Game over!

Attacker point of view (More realistic)

- Thousands of users, tens of groups, hundreds of machines
- Extremely complex access rights / user groups patterns

- Tens of credentials in Credential-Set
- **How could the attack proceed?**
  - How many credentials are still missing for reaching a powerful user group?
  - Where could I try to obtain them?
  - Which sequence of configuration changes / lateral movement should I execute?

  - Which workstations / services / accounts / groups do I fully or partly control ?
  - ...perhaps in a complex **transitive** way (group inheritance)?

Defender point of view (More realistic)

- Thousands of users, tens of groups, hundreds of machines
- Extremely complex access rights / user groups patterns

- **Which unintended paths to critical resources?**
  - How many credentials do users in group X need for reaching a powerful user group?
  - ...for becoming local admin of an interesting workstation?
  - Is there any unintended (set of) user(s) that belong to a powerful user group?
  - Is there any user that could become very close to a powerful user group by abusing his access rights?
  - ...

Bloodhound (I)

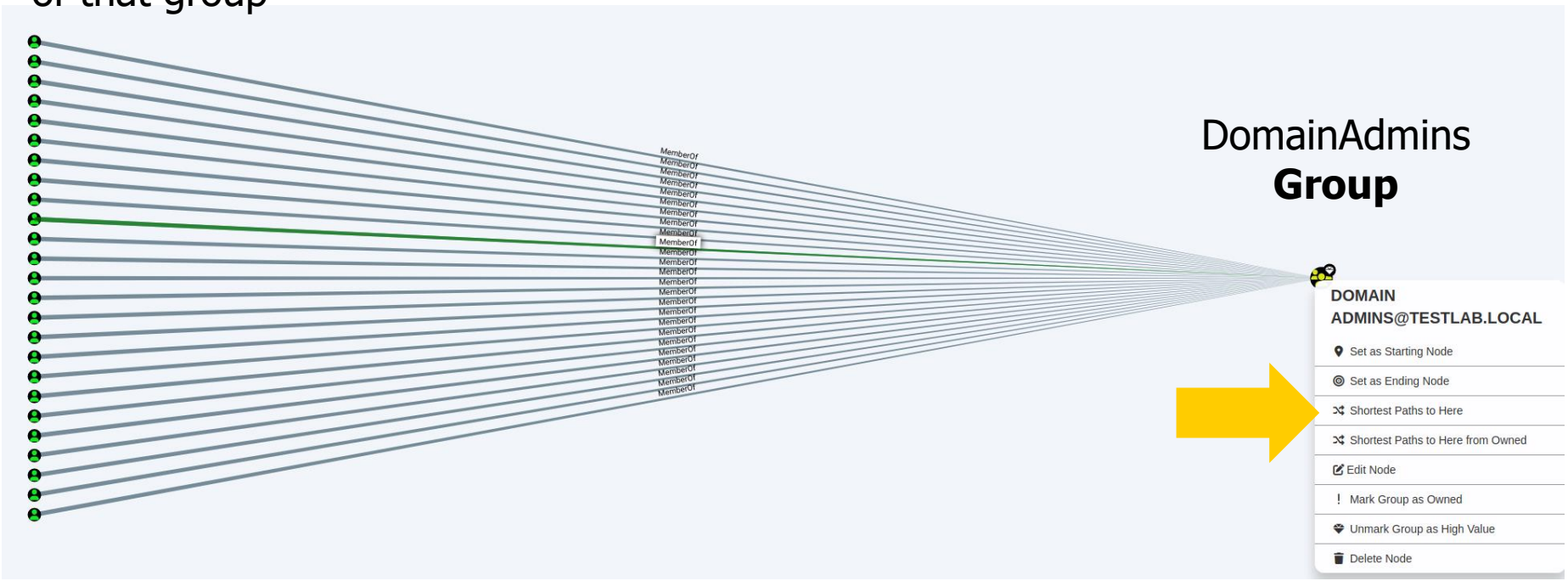
- Reveal the **hidden** and **often unintended** relationships within an Active Directory environment

- **Attackers** can easily **identify** highly complex attack paths that would otherwise be impossible to quickly identify.
- **Defenders** can identify and **eliminate** those same attack paths.

- **LDAP** queries to Domain Controllers
  + **GUI**
  + **Graph** DB

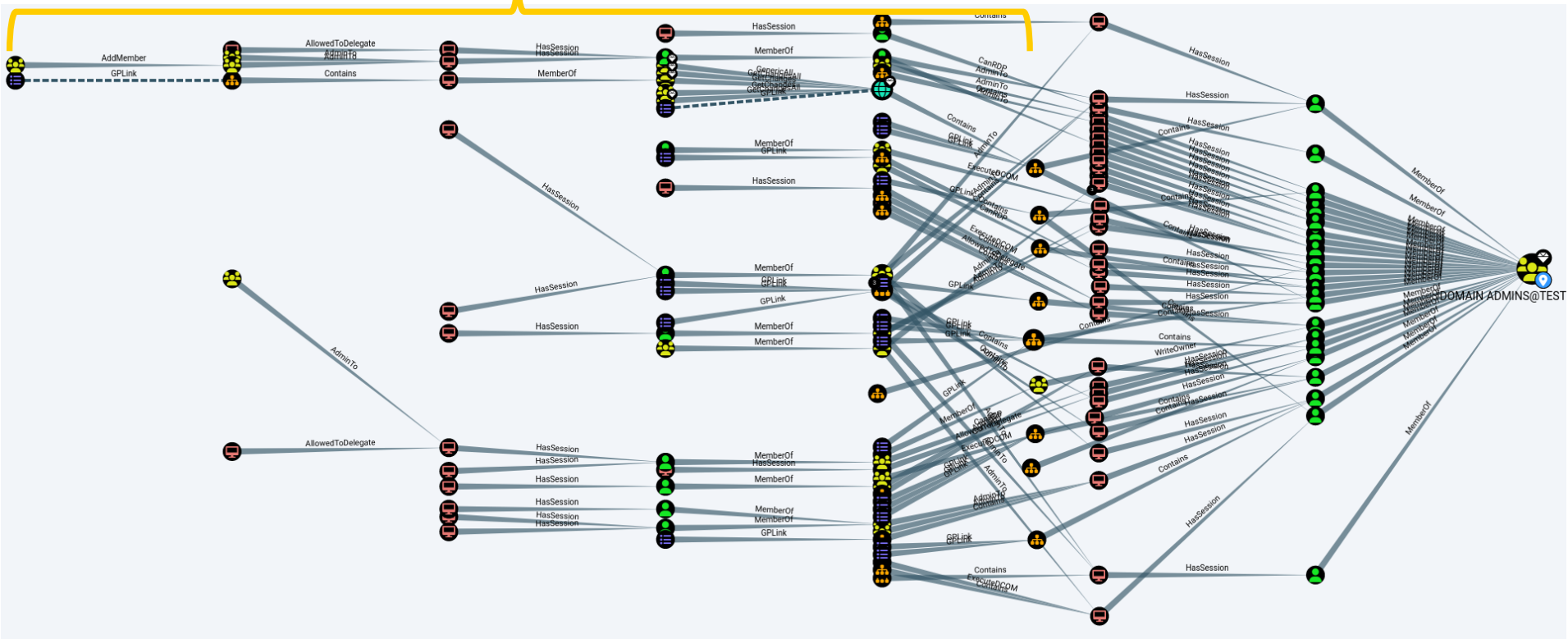
Example: DomainAdmins Group

Users members of that group



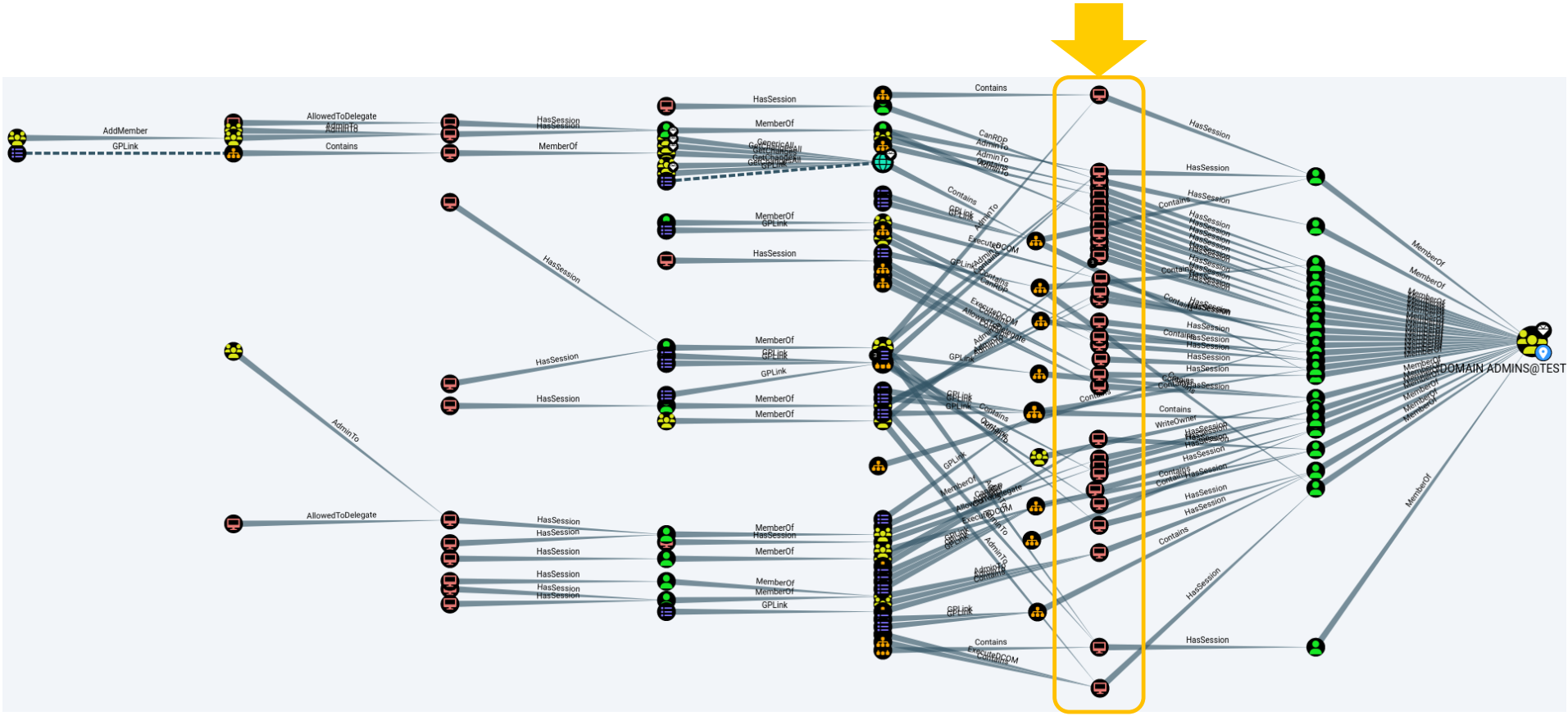
How can I arrive there? (I)

- Complex graph of **IT objects** and of **relations** between objects
  - Account memberOf Group
  - Account WriteAll Account



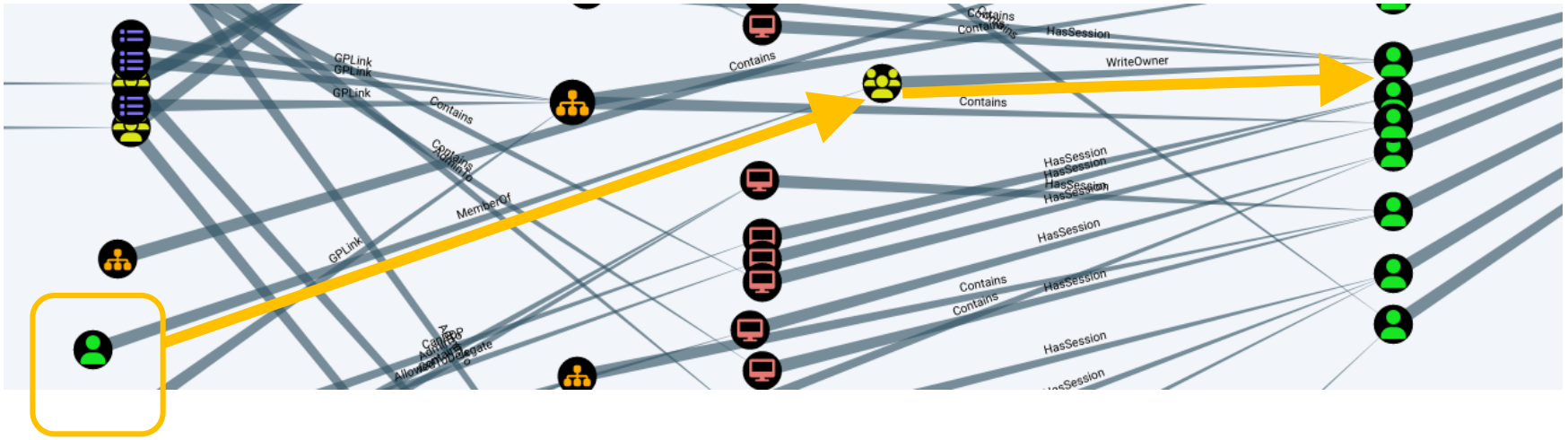
How can I arrive there? (II)

- Complex graph of **IT objects** and of **relations** between objects
  - Account memberOf DomainAdmin hasSession Computer



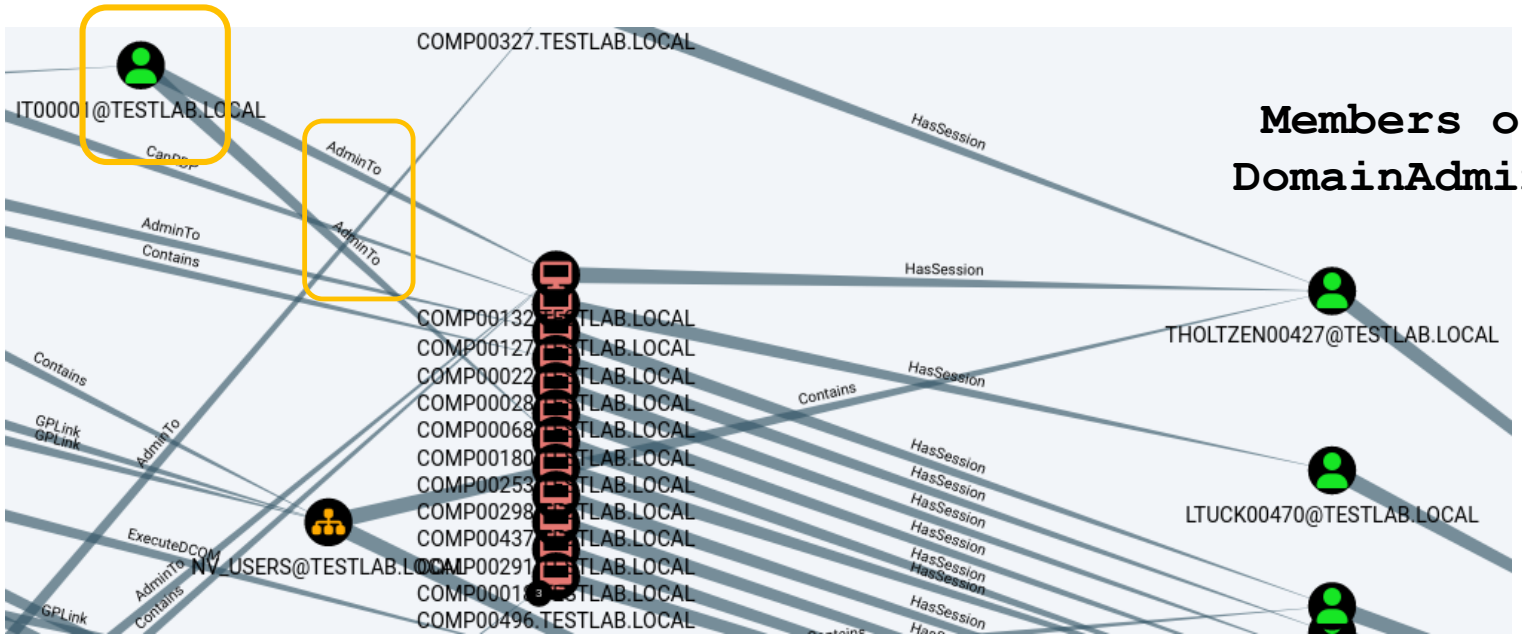
(Probably) Unintended Relationship

- This user is member of a group that can change the password of members of DomainAdmins group
- Obtaining the credentials of this user means GAME OVER

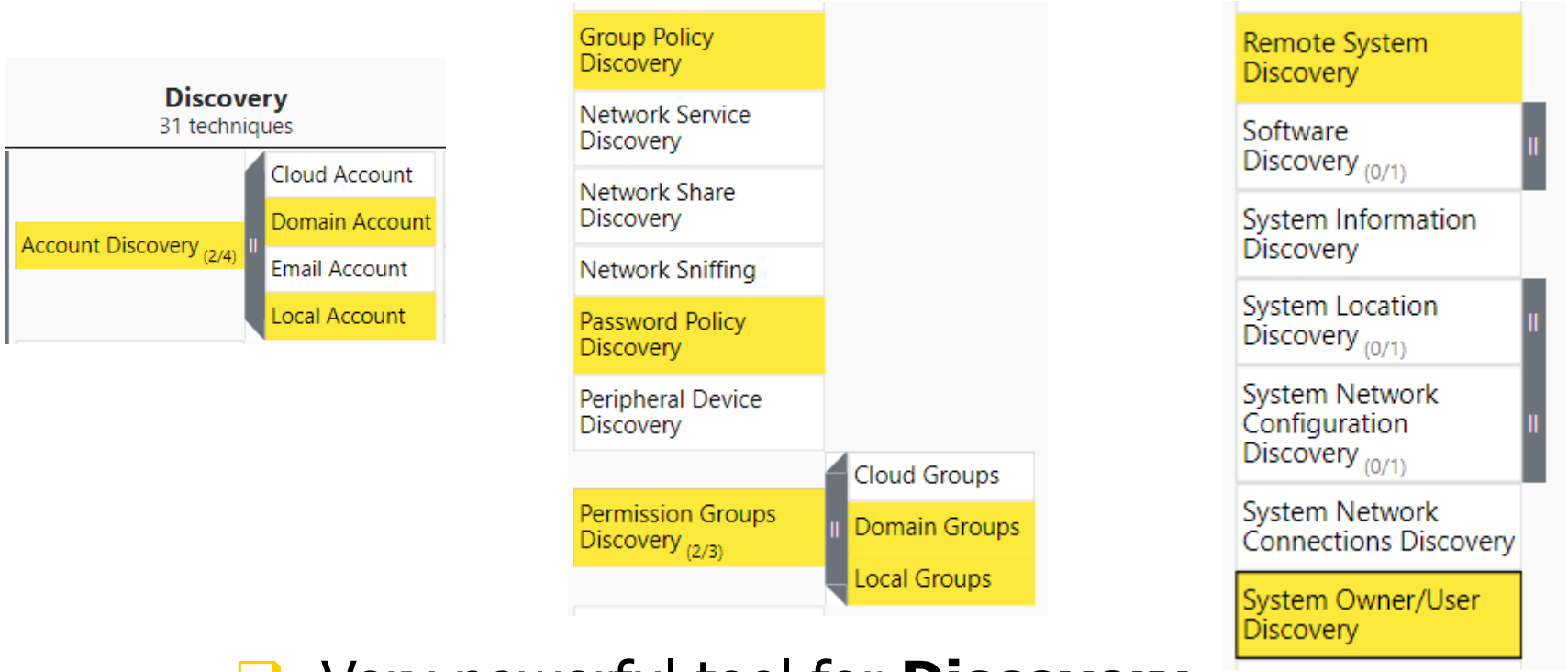


Bad Practice

- This user is local admin (= can steal password hashes) of computers were some members of DomainAdmins group are currently logged on
- Obtaining the credentials of this user could mean GAME OVER



Bloodhound (II)



- Very powerful tool for **Discovery**
- Can be executed by **any** domain user (no special privilege needed) !

Bloodhound (III)

- Very powerful tool for Discovery
- Can be executed by **any** domain user (no special privilege needed) !
- ...and **next steps** for Lateral Movement / Privilege Escalation
  - Identify promising **targets**
  - Suggest **how** to take control of them

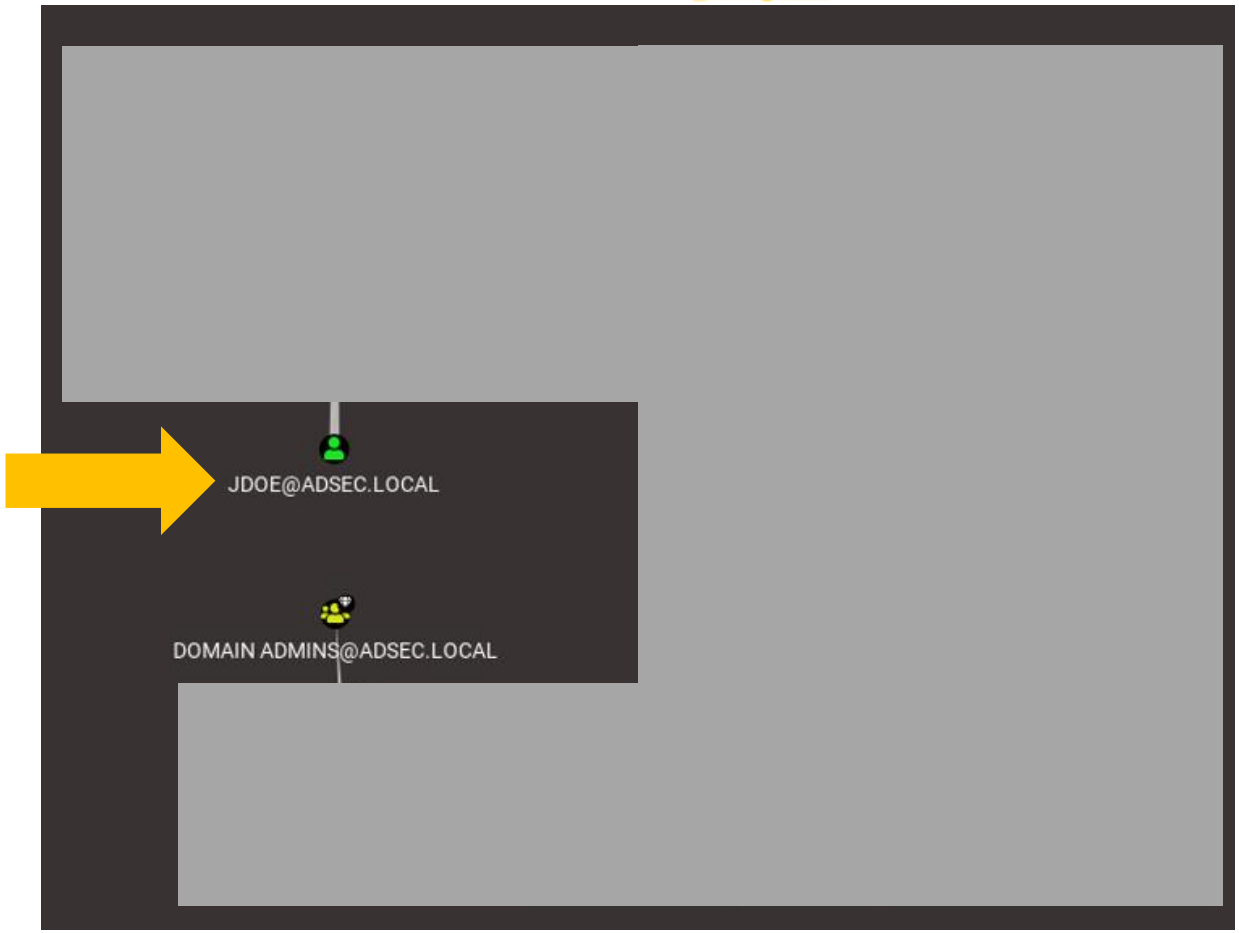
More Complex Example (Attacker point of view)

- I have the credentials of user `JDOE`
- I want to join the `Domain Admins` group

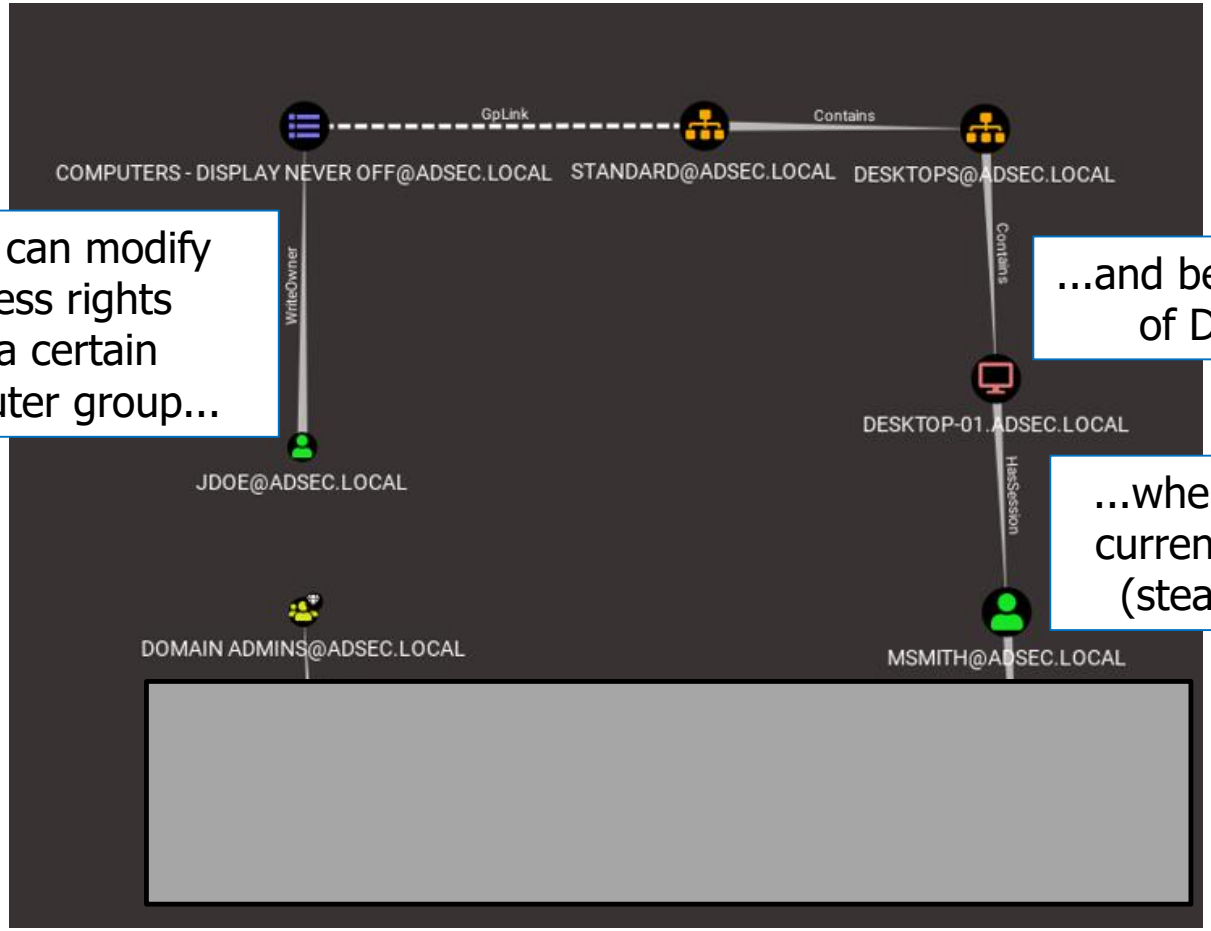
- Typical Attacker questions:
  - Can I join that group with access rights **I already have**? How?
  - If not, which users/machines should I attack? How?

- Example in next slides assumes further attacks are needed

Example:
JDOE User→Domain Admin group (I)



Example:
JDOE User→Domain Admin group (II)

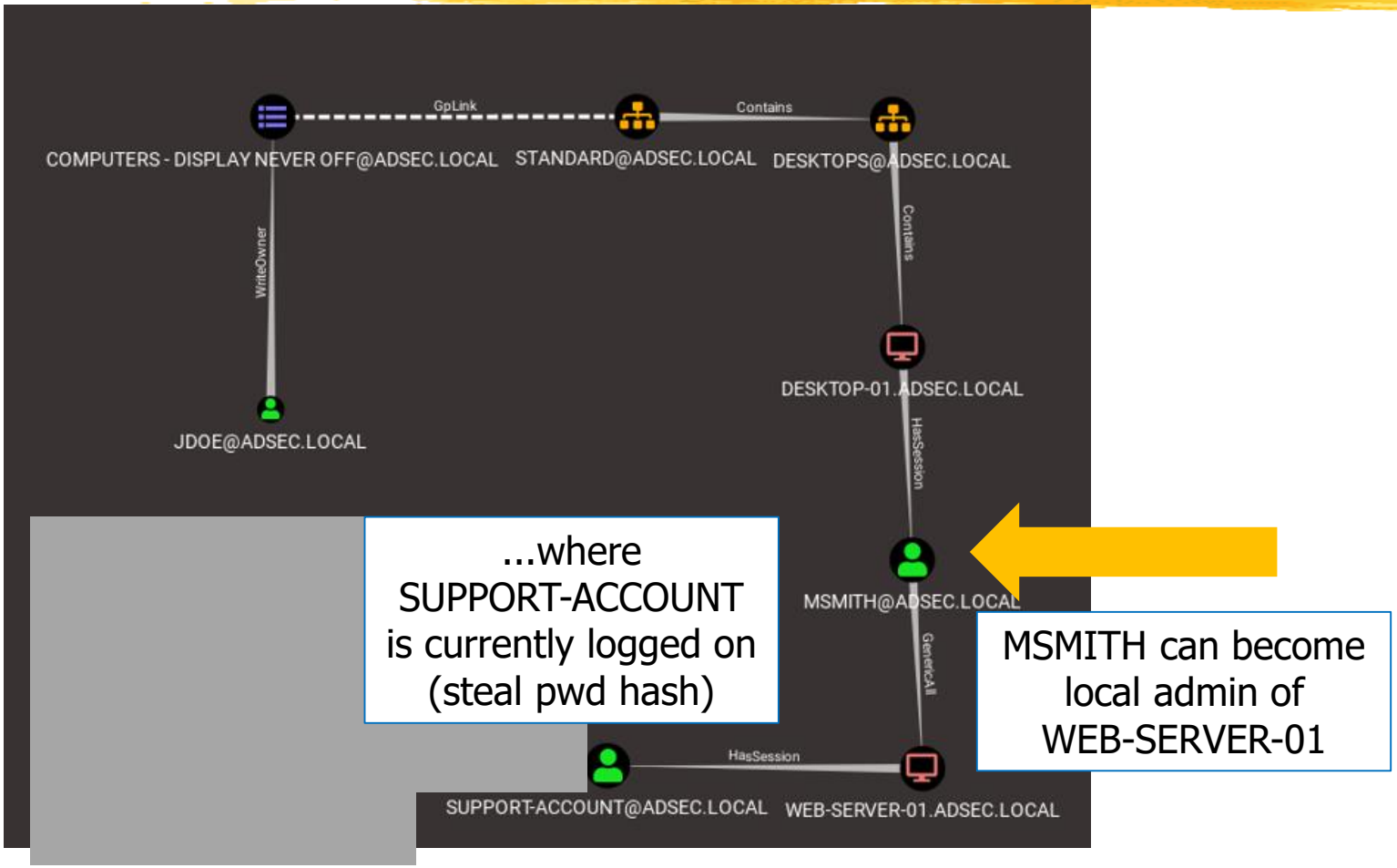


JDOE can modify access rights of a certain computer group...

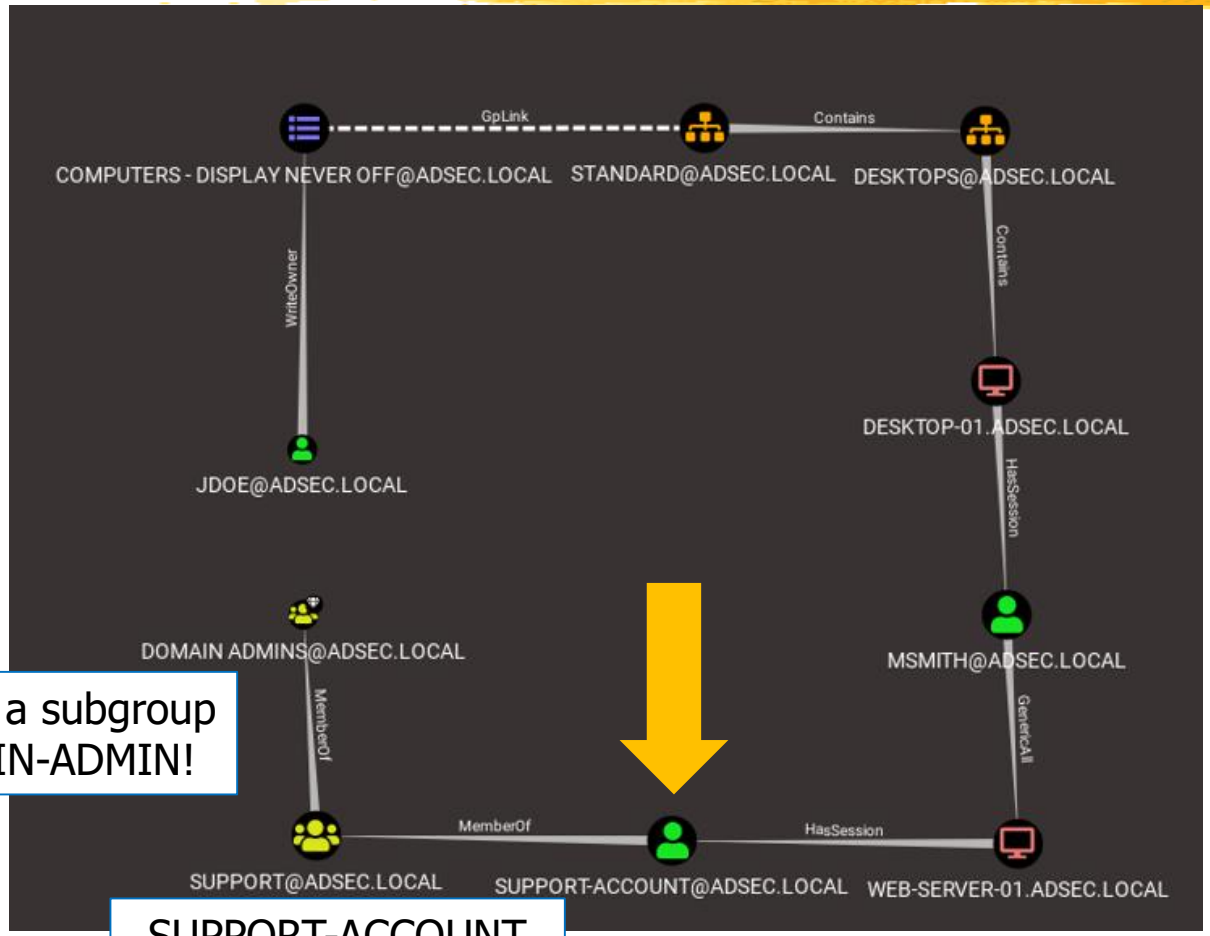
...and become local admin of DESKTOP-01...

...where MSMITH is currently logged on (steal pwd hash)

Example:
JDOE User→Domain Admin group (III)



Example:
JDOE User→Domain Admin group (IV)



...which is a subgroup of DOMAIN-ADMIN!

SUPPORT-ACCOUNT belongs to SUPPORT group

Pre-built Queries

Principle of Least Privilege

- Principle of **Least Privilege** (Saltzer and Schroeder **1975**)
- **Every** program and every user of the system should operate using the **least** set of privileges **necessary** to complete the job.

- Please take a moment to reflect and admire its depth
- Examples in the next slides

THINK ABOUT IT
NOW THAT YOU KNOW (A LITTLE OF) AD!

Example (I)

- Stealing credentials from LSASS:
  - Why normal user U1 is local admin?
  - Why a user administers a PC with credentials that allow administering a server?

- AD Attack path:
  - Why user UX can add any other user to `Backup Operators` group?

Example (II)

- **Much deeper than it seems**
- Does workstation A **really** need the right to open a TCP connection to workstation B?
- If this right is not necessary, then it should be removed

- Lateral movement from A to B would become **impossible**

Helpful Analogy

- Doorkeeper is given guns and bombs as soon as he enters in charge
  - Guns and bombs =
    Rights to execute **more** operations than **necessary** for the job
- Modifying procedures so that he is not given guns and bombs is **costly**
- It does not matter: He will never use guns and bombs

- Doorkeeper might:
  - Decide to use them anyway
  - Be forced to use them
  - **Get crazy**
  - **Act while being hypnotized**

Recent Alert

Alert (AA21-265A)

Conti Ransomware

Original release date: September 22, 2021 | Last revised: March 09, 2022

- Reported Conti ransomware attacks against U.S. and international organizations have risen to **more than 1,000**.
- CISA, FBI, and NSA recommend that network defenders apply the following **mitigations** to reduce the risk of compromise by Conti ransomware attacks.
- **…**
- Implement and ensure **robust network segmentation** between networks and functions **to reduce the spread** of the ransomware.
- Regularly audit administrative user accounts and **configure access controls under the principle of least privilege**

AD Attack Examples

AD Attack Examples

- Allow executing **steps** of **longer** attack chains
- Each one requires certain **pre-conditions**
  1. Attacker has access rights for **certain** operations
  2. One or more of:
     - Misconfigurations
     - **Overprivilege**
     - Presence of **optional** modules
     - **Insecure default**

- Execution with specialized software tools and **not trivial technical skills**

Important

- They are all **abuses** of **legitimate functionalities**
- **Unintended** combinations of access rights
- Many, many, many more examples than shown here
- Still discovered more or less routinely

Saltzer and Schroeder strike again!

Economy of mechanism:
Keep the design as simple and small as possible.

- This well-known principle applies to any aspect of a system, but it deserves **emphasis for protection mechanisms** for this reason:
- Design and implementation **errors that result in unwanted access paths** will **not** be noticed during **normal** use (since normal use usually does **not** include attempts to exercise **improper** access paths)

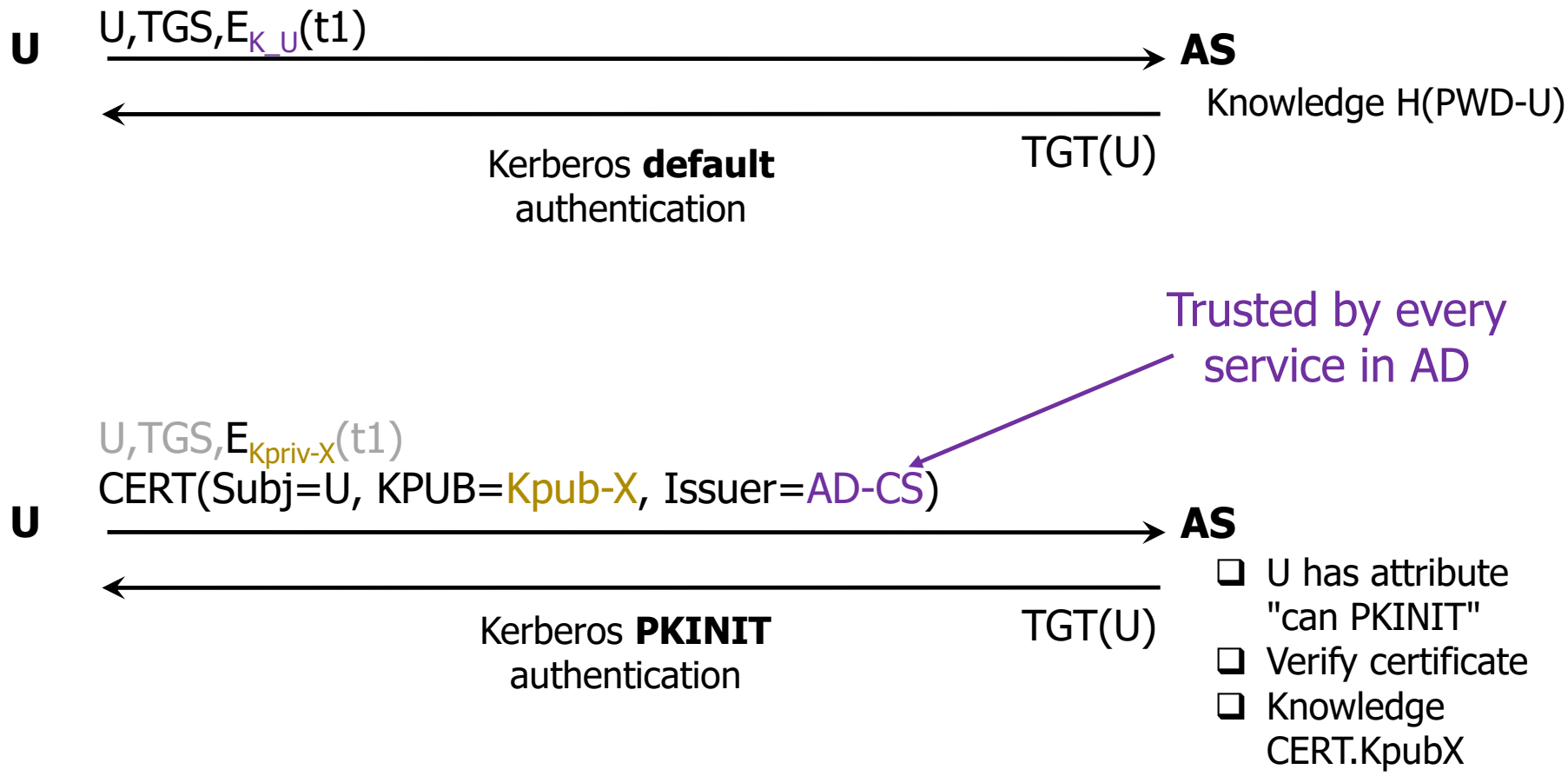
Certified pre-owned

Kerberos
PKINIT Authentication (I)

- Use case: user U authenticates to workstation with **smartcard** without providing any password
- Smartcard contains:
  - Kpriv-X, Kpub-X
  - **Certificate** <S=U, K=Kpub-X, Issuer=TrustedByKDC>

- Workstation cannot obtain TGT(U) by proving knowledge of H(pwd-U)
- Workstation obtains TGT(U) by:
  - Sending certificate
  - Proving knowledge of Kpriv-X

Kerberos
PKINIT Authentication (II)



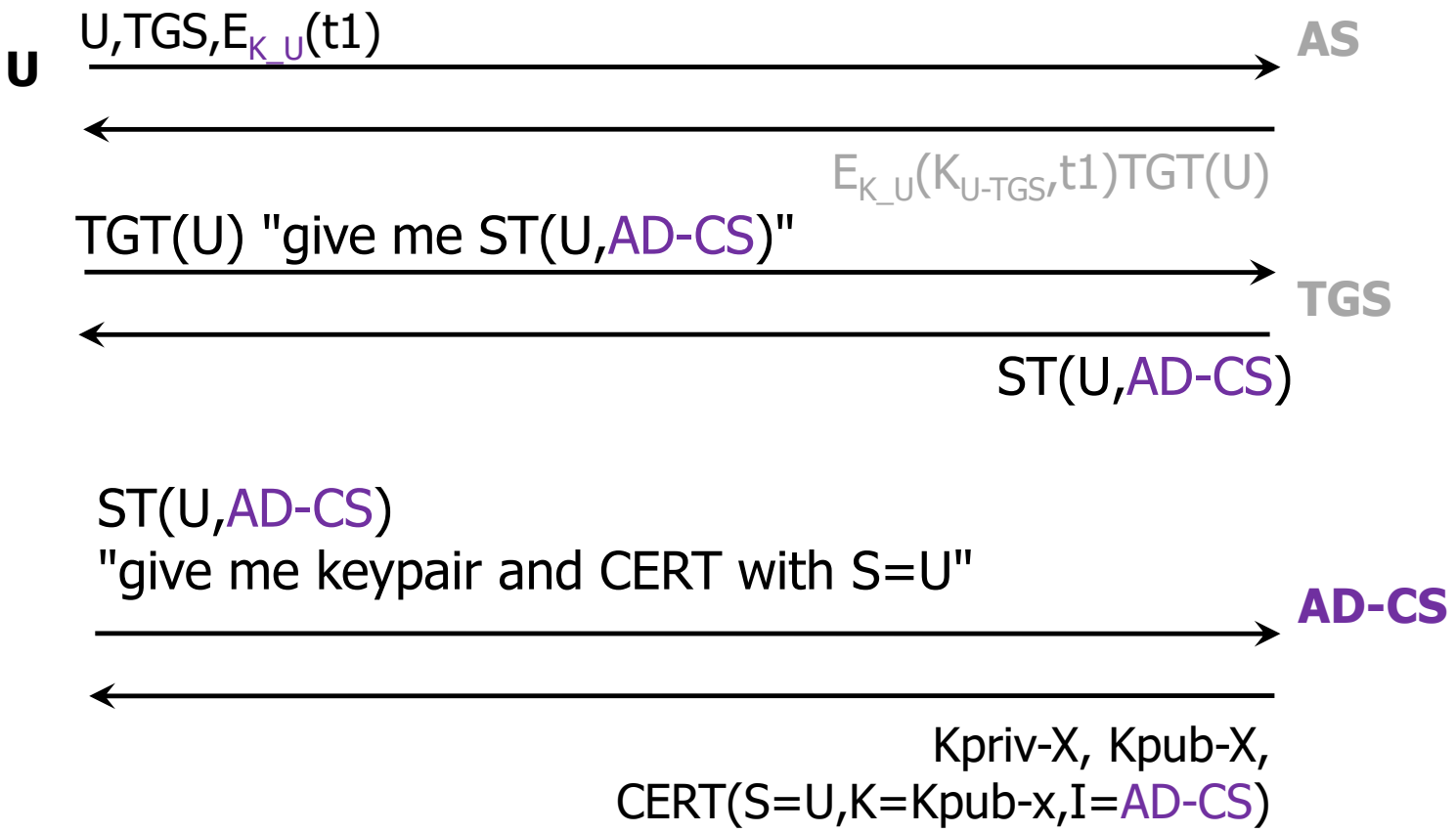
AD Certificate Service

- **Certificate Service** (**AD-CS**) issues **keypairs** and **certificates** to authenticated users
- Certificates accepted by every other service
  - AD-CS in KeySet and TrustSet

- Usage outline:
  1. U authenticates to AD-CS with ST(U,AD-CS)
  2. U requests to AD-CS a keypair and certificate with Subject=U

  Later, U can authenticate to AS with Kerberos PKINIT
  (see next slide)

Using AD-CS


U
U,TGS,E_K_U(t1)
AS
E_K_U(K_U-TGS,t1)TGT(U)
TGT(U) "give me ST(U,AD-CS)"
TGS
ST(U,AD-CS)
ST(U,AD-CS)
"give me keypair and CERT with S=U"
AD-CS
Kpriv-X, Kpub-X,
CERT(S=U,K=Kpub-x,I=AD-CS)

AD-CS Fact #1

- **Access rights** to **AD-CS** **very complex**
  - **Many** operations
  - Decision depends on **many** attributes
    - Who can request a certificate
    - Who can set the Subject in a certificate
    - …
- Access rights to **AD-CS** configured in a set of **predefined** "certificate **templates**"

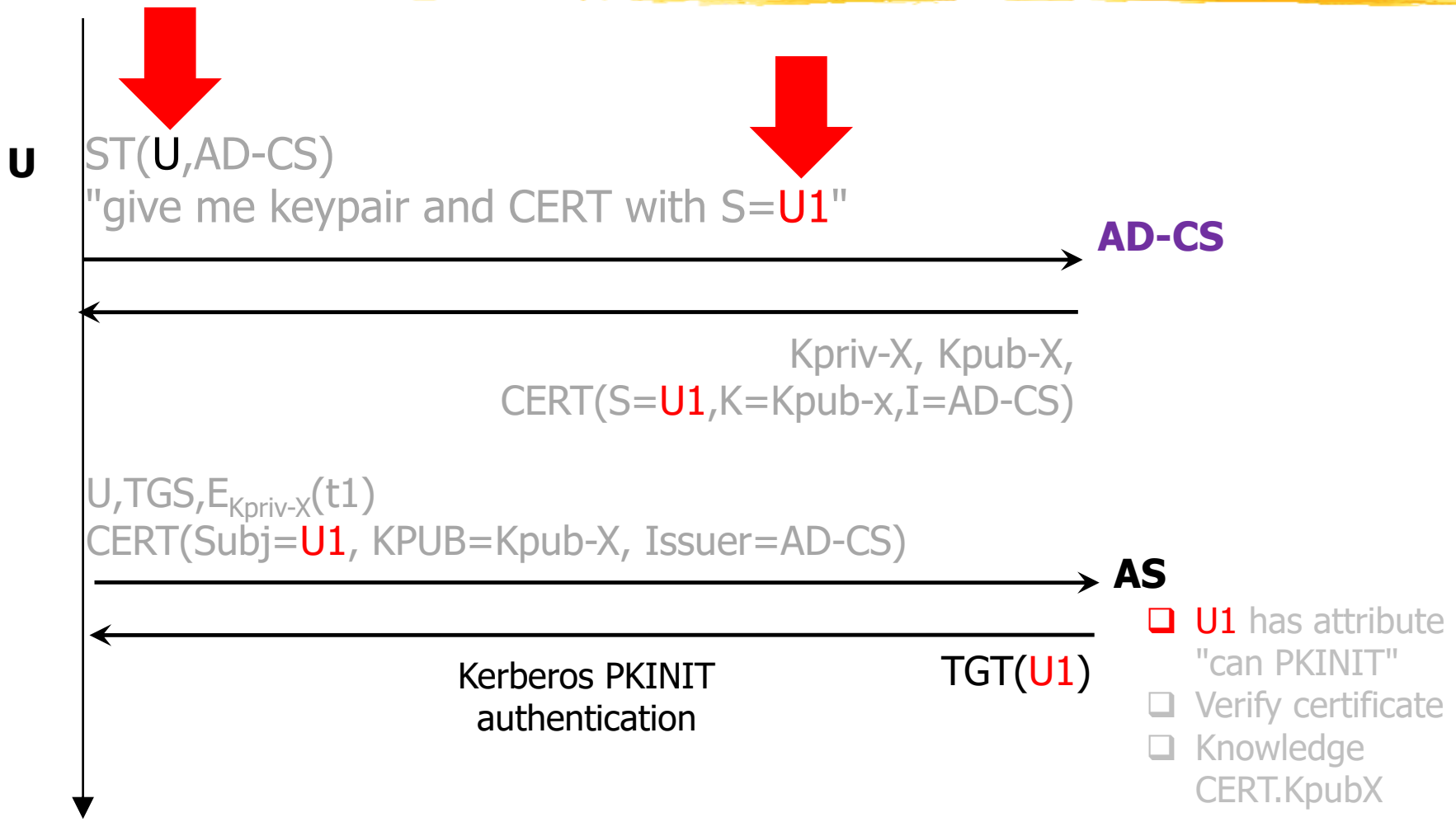
AD-CS Fact #2

- Access rights to **AD-CS** configured in a set of **predefined** "certificate **templates**"

- One of the **predefined templates** has these access rights:
  - Who can request a certificate — Any domain user
  - Who can set the Subject in a certificate — The requester of the cert!
  - …

- **Insecure default**
- **Nobody had realized until last year**

Exploitation



Exploitation (April 2022) (I)

1. **Initial Access**
   - Impersonate some U-X
2. **Discovery**
   - Determine that AD-CS is installed
   - Determine presence of insecure certificate template
   - Identify username of domain admin, say U-DA
   - Verify that U-DA can auth with PKINIT
3. **Credential Access**
   - Obtain ST(U-X, AD-CS)
   - Obtain keypair and certificate with S=U-DA

(**Privilege Escalation** with "Valid Account")

Exploitation (April 2022) (II)

Trello From the Other Side: Tracking APT29 Phishing Campaigns MANDIANT

search
"mandiant apt29 phishing"

- APT29 is a Russian espionage group...likely sponsored by the Foreign Intelligence Service (SVR).
- To gain access to a victim environment, APT29 sent spear-phishing emails disguised as embassy administrative updates
- In multiple cases, APT29 was able to gain Domain Admin in less than 12 hours from the initial phishing payload's execution.
- APT29 was also observed exploiting **misconfigured certificate templates** to allow them to impersonate admin users.

Exploitation (April 2022) (III)

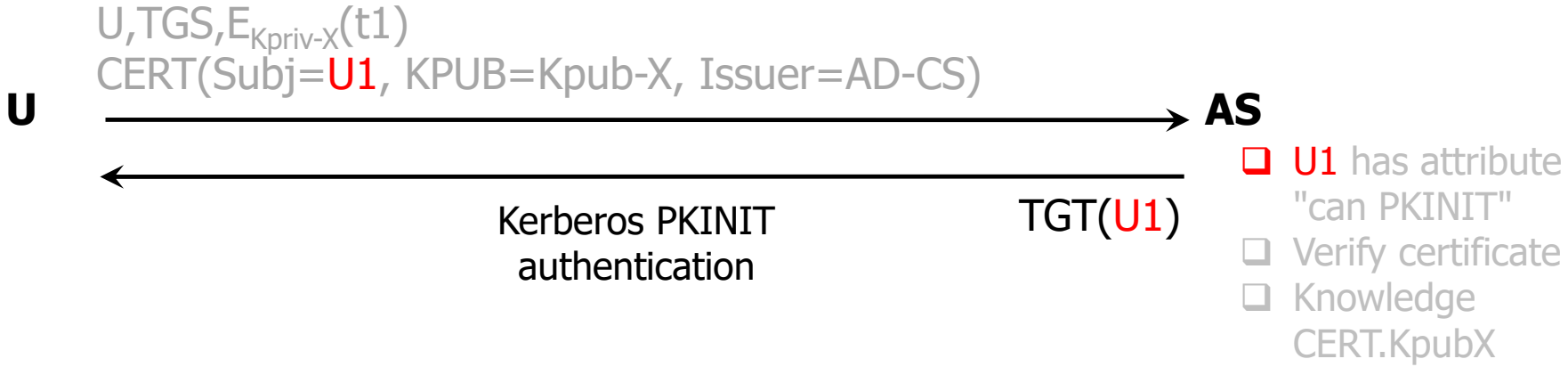
KNOWN EXPLOITED VULNERABILITIES CATALOG

CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

CVE-2022-26923	Microsoft	Active Directory	Microsoft Active Directory Domain Services Privilege Escalation Vulnerability	2022-08-18	An authenticated user could manipulate attributes on computer accounts they own or manage, and acquire a certificate from Active Directory Certificate Services that would allow for privilege escalation to SYSTEM.	Apply updates per vendor instructions.	2022-09-08

Notes https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2022-26923

Nice property



- U1 changes password
- Keypair and certificate remains valid!
  (U may still impersonate U1)

Shadow Credentials

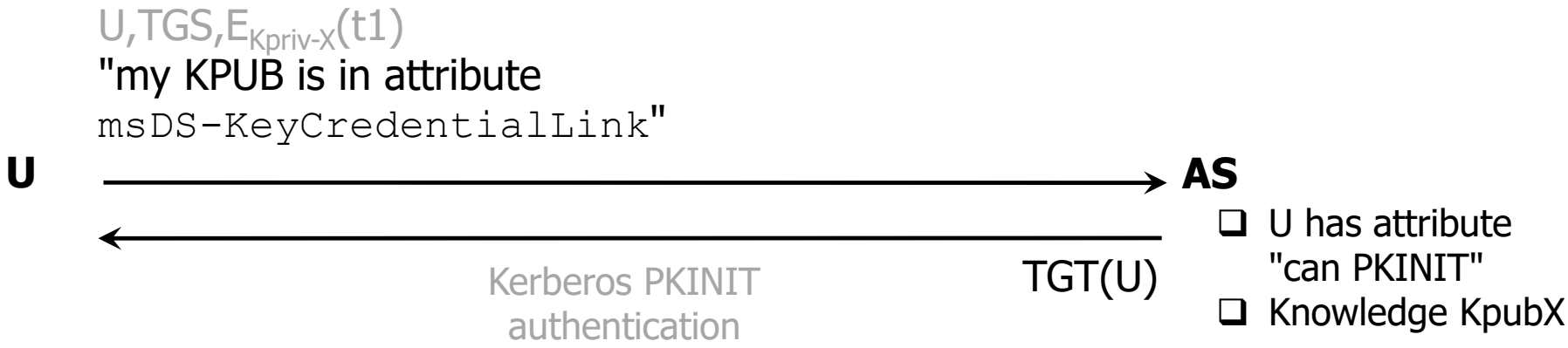
Trusted Devices

- Use case:
  - Users can enroll a **trusted device** for **passwordless** MFA (e.g., a smartphone or PC with biometric authentication)
  - Trusted device generates Kpriv-x, Kpub-x
  - Kpub-x must be securely associated with User that owns the device (attribute `msDS-KeyCredentialLink` of User)
- Implementation:
  - Client software with TGT(U1) stores Kpub-x in attribute `msDS-KeyCredentialLink` of U1

AD-Fact

Users can authenticate with PKINIT Kerberos Authentication:

- Even **without any certificate** issued by AD-CS
- Even if AD-CS is **not** installed (!)



Consequence

- Hidden / Mysterious (undesired) path that may exist:
  User UX has WRITE access on `msDS-KeyCredentialLink` of U

- UX creates Kpriv_X, Kpub_x and writes Kpub_x in attribute `msDS-KeyCredentialLink` of U

- UX impersonates U with PKINIT
- ...even if U changes password

- UX has "**shadow credentials**" almost certainly unknown to U

PKINIT: Password Hash?

U2U (I-a)

- Use Case: User U authenticates with PKINIT (Smartcard)
  - Obtains TGT(U) and $K_{U\text{-}TGS}$
  - Does **not** have H(PWD-U) (= K_U)

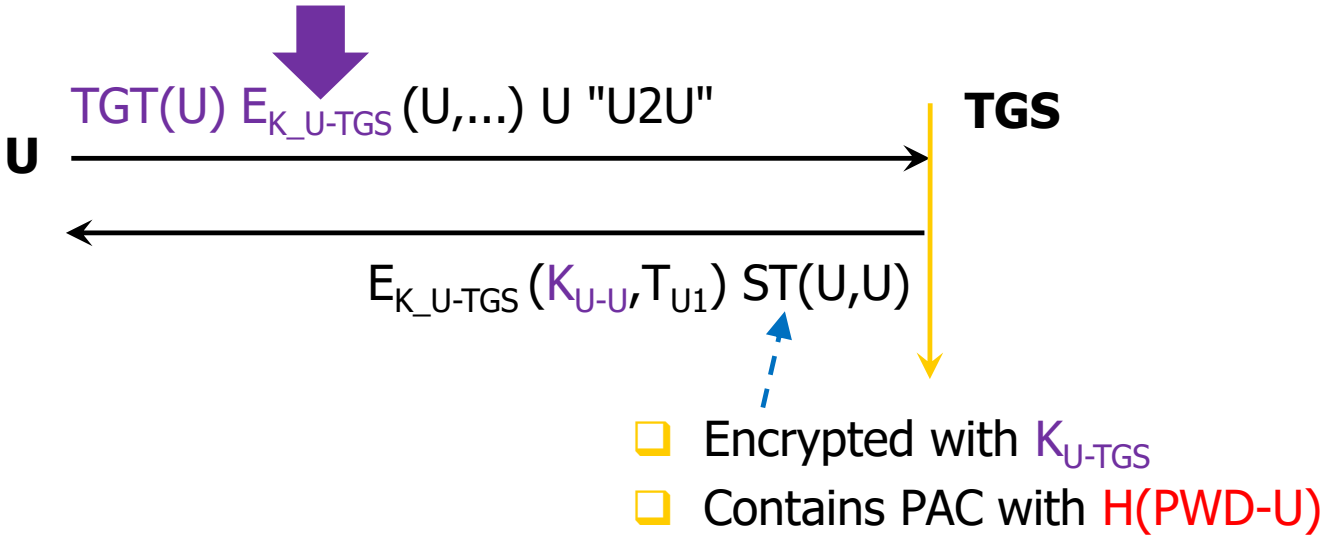
- Can obtain ST(U,*)
- **Cannot** access services that support **only NTLM**

U2U (I-b)

- Use Case: User U authenticates with PKINIT (Smartcard)
  - Obtains TGT(U) and $K_{U\text{-}TGS}$
  - Does **not** have H(PWD-U) (= K_U)

- Solution: Obtain a "special" **ST(U,U)** that:
  - Contains H(PWD-U)
  - Is encrypted in $K_{U\text{-}TGS}$ (rather than in K_U)

U2U (II)


U
TGS
TGT(U) $E_{K\_U\text{-}TGS}$ (U,...) U "U2U"
$E_{K\_U\text{-}TGS}$ ($K_{U\text{-}U}$,T_{U1}) ST(U,U)
Encrypted with $K_{U\text{-}TGS}$
Contains PAC with H(PWD-U)

PKINIT Abuse: Password Hash?

- Attacker UX impersonates U by abusing PKINIT
  - Obtains TGT(U) and $K_{U\text{-}TGS}$
  - Does **not** have H(PWD-U) (= K_U)

- Can obtain ST(U,*)
- **Cannot** impersonate U on services that support **only NTLM**
- **Cannot** brute force U **password** (if it was needed)

- Obtain ST(U, U)!