

Яндекс



Безопасность аутентификации в Windows & AD

Василенко Анатолий, инженер по информационной безопасности

План

- › Аутентификация в Active Directory
 - › NTLM
 - › Kerberos
- › Смарт-карты
- › ADACS
- › Pass-the-hash (и VBS)
- › SSO в масштабах компании

Active Directory

- | **Directory** – иерархическая структура, содержащая информацию о различных объектах в вашей сети (включая пользовательские и компьютерные аккаунты, группы, политики безопасности, информацию о сервисах, dns доменах, принтерах, ...).
- | **Directory Service** – набор сервисов предоставляющих доступ к информации вместе с самой информацией.
Active Directory¹ – реализация directory service предоставляемая Microsoft.

Интерфейсы взаимодействия с Active Directory:

- > LDAP
- > AD Web Services
- > Протоколы аутентификации (Kerberos, NTLM)
- > ...

¹ https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/d2435927-0999-4c62-8c6d-13ba31a52e1a

NTLM

NTLM – жаргон

В терминологии ужасная путаница.

Хеши на основе пользовательского пароля:

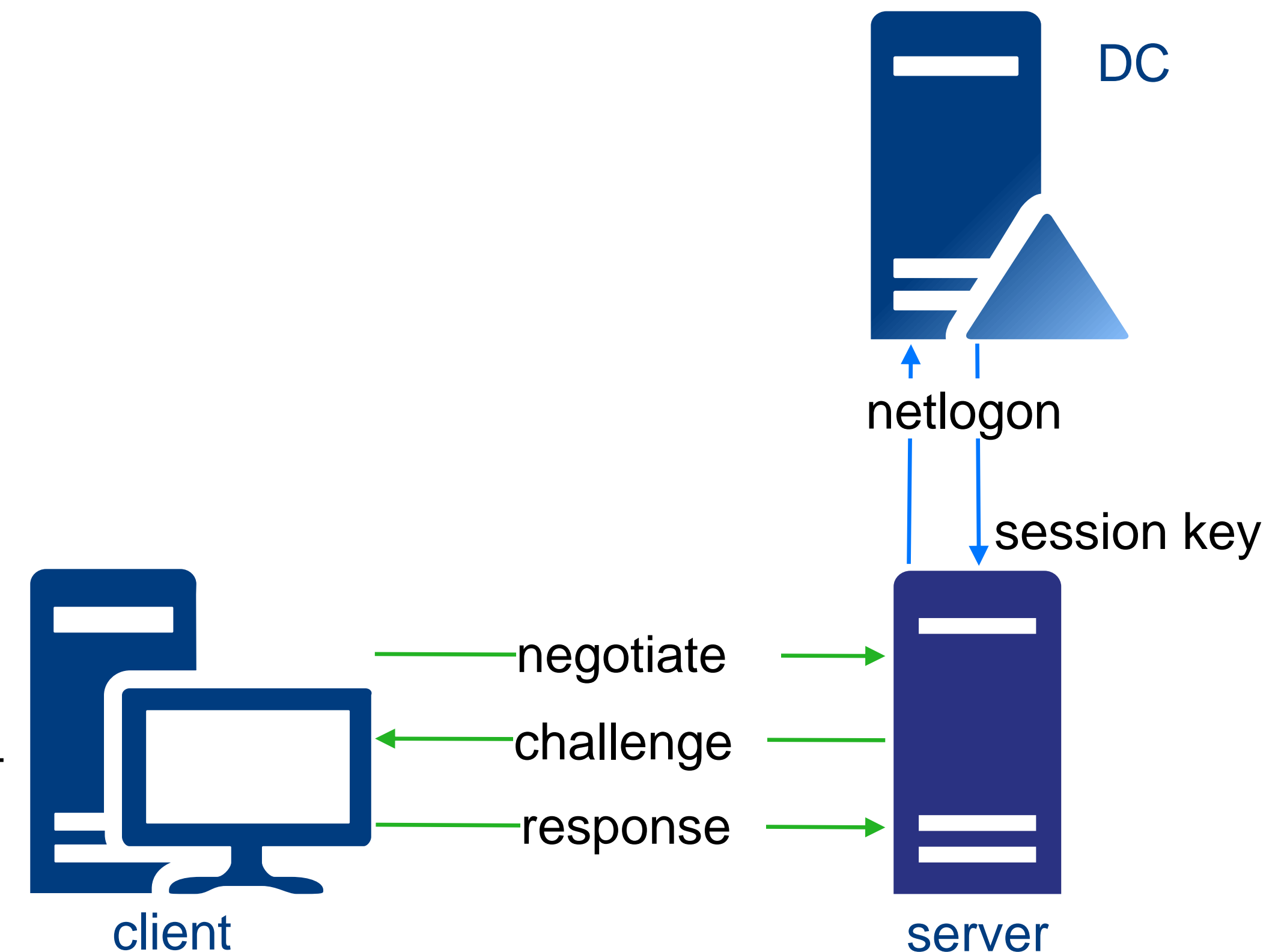
- › LM/LMOWF (des) (“aad3b435b51404ee aad3b435b51404ee”)
- › NT/NTOWF (md4) (иногда называют NTLM)

Протоколы аутентификации:

- › NTLMv1/NetNTLMv1 (des)
- › NTLMv2/NetNTLMv2 (hmac-md5)

Модификации протокола аутентификации:

- › NTLM2 – hmac-md5 криптография в v1 структуре данных;
 - › поддерживало v2 криптографию при аутентификации на старых серверах;
- › NetNTLMv1/2 with ESS (Extended Session Security) – когда клиент также подмешивает собственный challenge;
 - › защищает v1 от rainbow атак;



<https://medium.com/@petergombos/lm-ntlm-net-ntlmv2-oh-my-a9b235c58ed4>

<https://davenport.sourceforge.net/ntlm.html>

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/b38c36ed-2804-4868-a9ff-8dd3182128e4

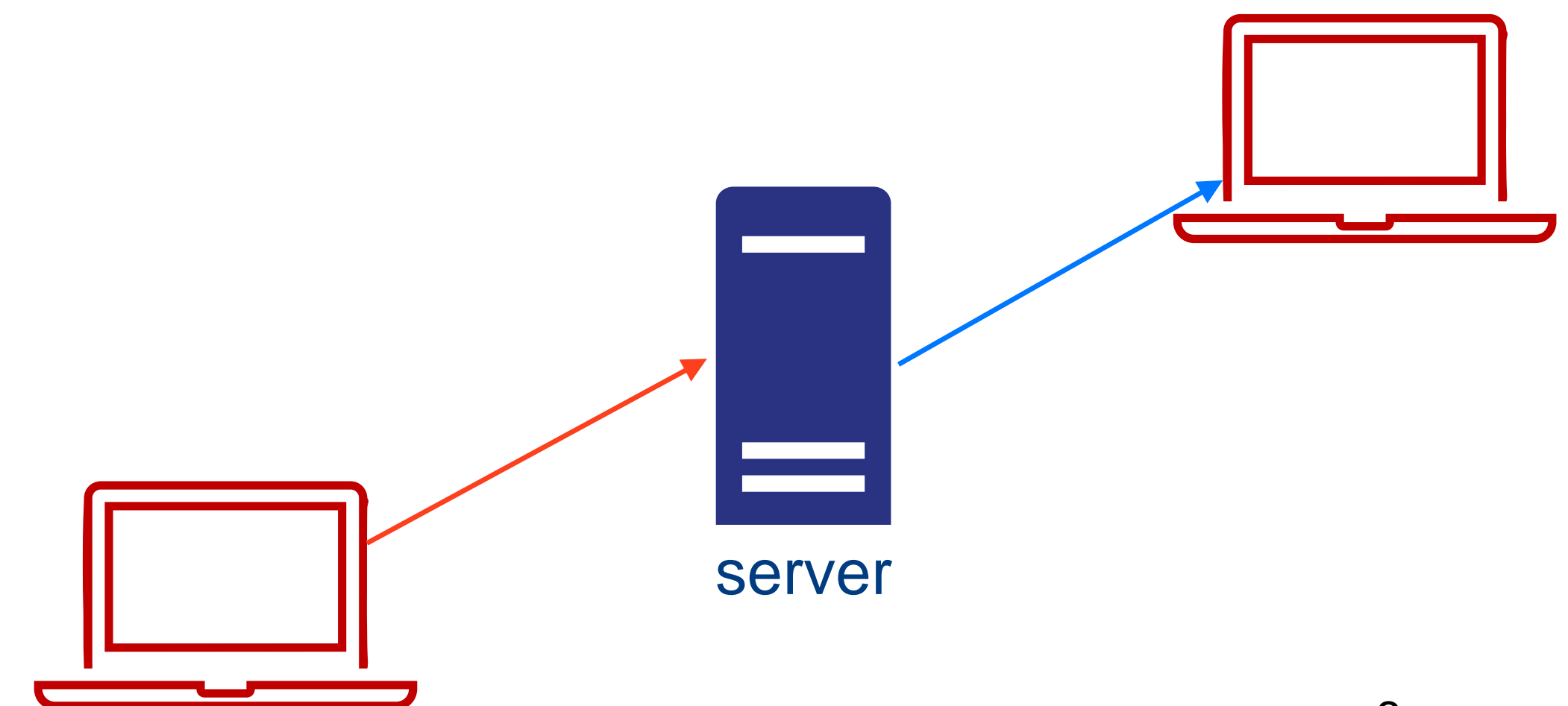
SSRF-like примитив

Функциональные возможности, позволяющие пользователю вызвать со стороны сервера/сервиса подключение по заданному адресу, с аутентификацией от имени сервера/сервиса.

- › printer bug, EFS (“PetitPotam”), DFSCoerce, ...¹ - **очень опасный примитив**;
- › вредоносные ссылки, вложения, файлы (например, `file.scf`);
- › эксплуатация из не-привилегированного контекста (например, `mssql: dir_tree`)

Очень опасный примитив, который атакующие используют при выстраивании цепочек эксплуатации.

- › Microsoft не считает это уязвимостью².



¹ <https://github.com/p0dalirius/windows-coerced-authentication-methods>

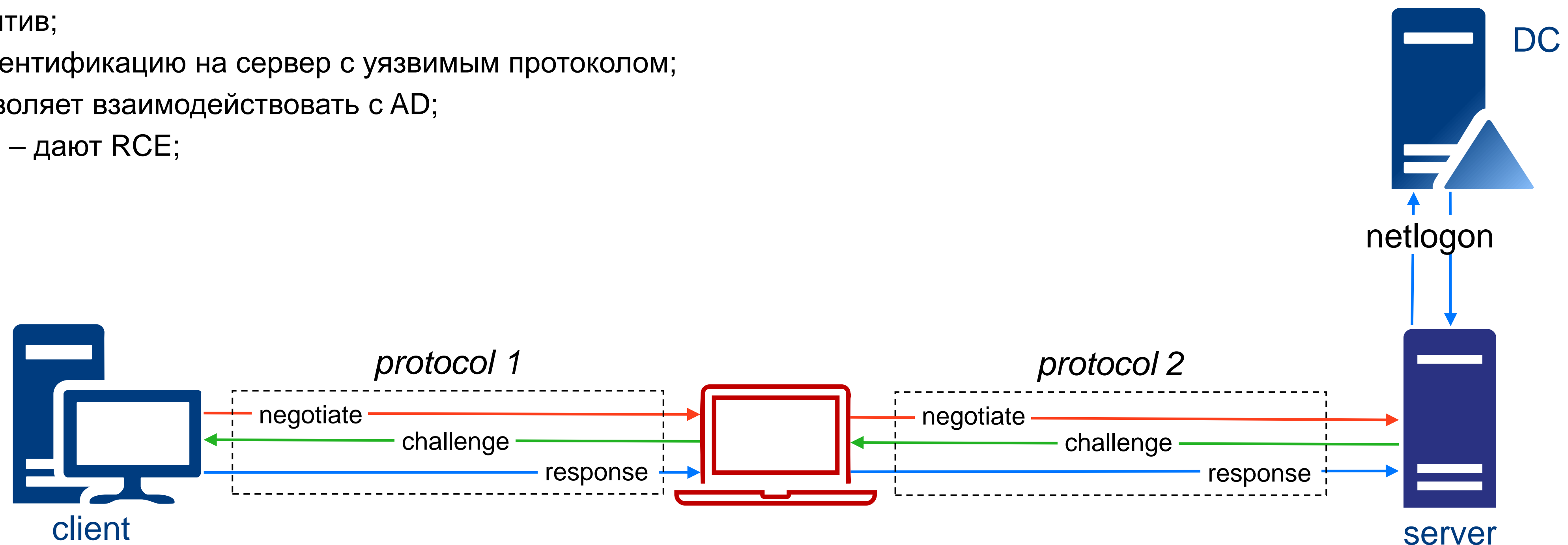
² “не баг, а фича”

NTLM-relay атаки

- › ntlm-relay – разновидность MITM атаки, при которой атакующий аутентифицируется от имени пользователя на сервере;
- › Протоколы приложений могут быть разными: smb, dcom, ldap/ldaps, http/https, mssql, winrm, ...;
- › К сожалению, для данной атаки существует очень богатый и развитым тулкит;

Поведение атакующего:

1. получить подключение от жертвы на контролируемый узел, например, используя:
 - › атаки на сеть (например, *-spoofing, ...);
 - › SSRF-like примитив;
2. произвести relay аутентификацию на сервер с уязвимым протоколом;
 - › ldap/ldaps – позволяет взаимодействовать с AD;
 - › smb/dcom/winrm – дают RCE;
 - › ...



Kerberos-relay атаки

Kerberos – тоже уязвим к relay атакам.

- › 2021.10: <https://googleprojectzero.blogspot.com/2021/10/using-kerberos-for-authentication-relay.html>
- › 2022.02: <https://dirkjanm.io/relaying-kerberos-over-dns-with-krbrelayx-and-mitm6/>

Принципы схожи: сетевая атака + relay аутентификации на другой сервер.

Ранее было принято считать, что отключение NTLM аутентификации в пользу Kerberos на сервисе – достаточно для защиты от relay атак.

Сейчас (“пока что”) правильнее говорить: Kerberos-relay атакующим сложнее эксплуатировать, чем NTLM-relay.

- › “Как будто бы”, ssrf-like и прочие методы не подходят для начала атаки, необходимо выполнять сетевые атаки.

Защита от *-relay атак (#1)

Защита со стороны сервисов:

- › “signing” – выработанный **сессионный ключ используется для подписи** сообщений в передаваемом протоколе; (signing следует включать в принудительном режиме на стороне сервера)
 - › ldap-signing, smb-signing, dcom-integrity¹, ...
- › “channel binding” – добавление CBT² (Channel Binding Token) токена в протокол аутентификации, также называют EPA³ (Extended Protection for Authentication), Microsoft поддерживает для взаимодействий по TLS:
 - › ldaps-binding, winrm channel binding, mssql channel binding⁴;
 - › https-binding (Microsoft поддерживает не во всех своих продуктах, бывают сложности с WCF⁵ приложениями);
- › “service validation” (“service binding”) – проверяет «SPN» из аутентификации:
 - › smb target name validation⁶;
 - › mssql service binding⁴;
 - › http service binding⁵;

Однако:

- › EPA – хоть и срабатывает, но не защищает, если клиент не валидирует сертификат сервера;
- › “service validation” – имеет сложности с поддержкой SPN в актуальном состоянии;

Задача для владельцев сервисов (администраторов): внедрить и адаптировать эти меры защиты.

¹ <https://learn.microsoft.com/en-us/windows/win32/rpc/authentication-level-constants>

² <https://www.rfc-editor.org/rfc/rfc5929>

³ [https://learn.microsoft.com/en-us/previous-versions/dotnet/netframework-3.5/dd767318\(v=vs.90\)](https://learn.microsoft.com/en-us/previous-versions/dotnet/netframework-3.5/dd767318(v=vs.90))

⁴ <https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/connect-to-the-database-engine-using-extended-protection?view=sql-server-ver16>

⁵ <https://learn.microsoft.com/en-us/iis/configuration/system.webserver/security/authentication/windowsauthentication/extendedprotection/>

⁶ [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/jj852272\(v=ws.11\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/jj852272(v=ws.11))

NTLM – ntlmv1 vs ntlmv2

- › недостатки NTLMv1, отсутствующие в v2:
 - › v1 всегда уязвима к -relay атакам, т.к. в формировании MIC (Message Integrity Code) не участвуют важные поля. Поэтому можно:
 1. удалить MIC или;
 2. подменить netbios имя компьютера и запросить сессионный ключ у домен контроллера;
 - › v1 не содержит поля с CBT токеном (Channel Binding Token), а значит EPA (Enhanced Protection for Authentication) не применима;
 - › v1 уязвима к брутфорс атаке по радужным таблицам (т.к. это DES от NT хеша), т.е. по перехваченному challenge-response можно в короткие сроки восстановить NT хеш;
- › оба протокола:
 - › уязвимы к -relay атакам, до сих пор;
 - › уязвимы к брутфорс атаке по перехваченным challenge-response;
 - › применима pass-the-hash атака, т.е. атакующий имеющий NT хеш может аутентифицироваться от имени пользователя;
 - › бывают уязвимости:
 - › CVE-2015-005 , CVE-2019-1019, CVE-2019-1040 (“drop-the-mic”), CVE-2019-1166, ...;

Hardening NTLM аутентификации (#2)

Противодействие возможным атакам:

- › Отключайте NetNTLMv1;
- › Отключайте хранение LM-хешей;
- › Повышайте уровень безопасности сетей (защита от сетевых атак);
- › Отключайте ssrf-like примитивы (например, на DC не нужен сервис принтерной печати);
- › Фильтруйте трафик;
- › Отключайте webclient сервис¹ (потому что для HTTP signing со стороны клиента всегда отключён);
- › Своевременно обновляйте сервера (CVE-2020-1113 – исполнение кода через MS-TSCH (не требовал подписи пакетов));

Программа максимум: отключите NTLM аутентификацию в домене;

¹ <https://pentestlab.blog/2021/10/20/lateral-movement-webclient/>

Kerberos

Kerberos аутентификация

Клиент: «хочу подключиться к сервису SMB на узле files.domain.local»

Аутентификация пользователя

1. AS REQ – Authentication Server request – запрос TGT тикета
2. AS REP – получение Ticket Granting Ticket (TGT) тикета

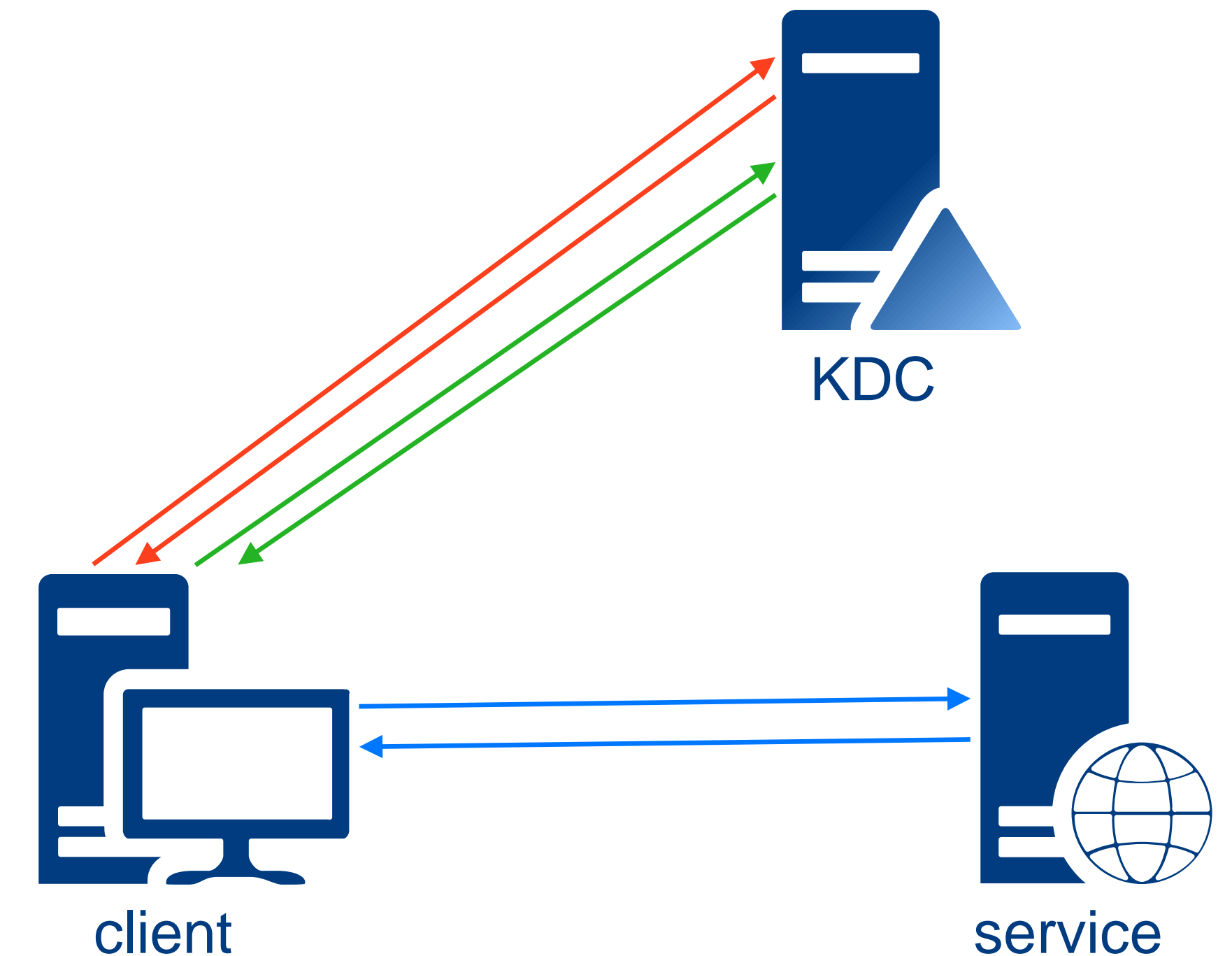
Запрос тикета для доступа к сервису cifs/files.domain.local

3. TGS REQ – Ticket Granting Service request (TGT+authenticator)
4. TGS REP – получение TGS тикета

Установление соединения с сервисом:

Передача TGS тикета по прикладному протоколу

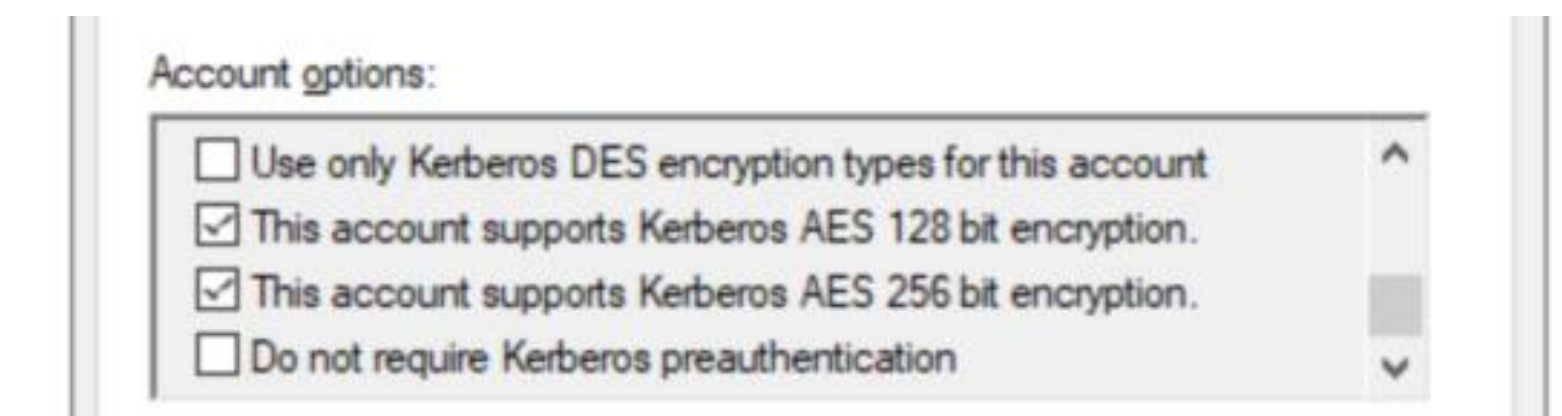
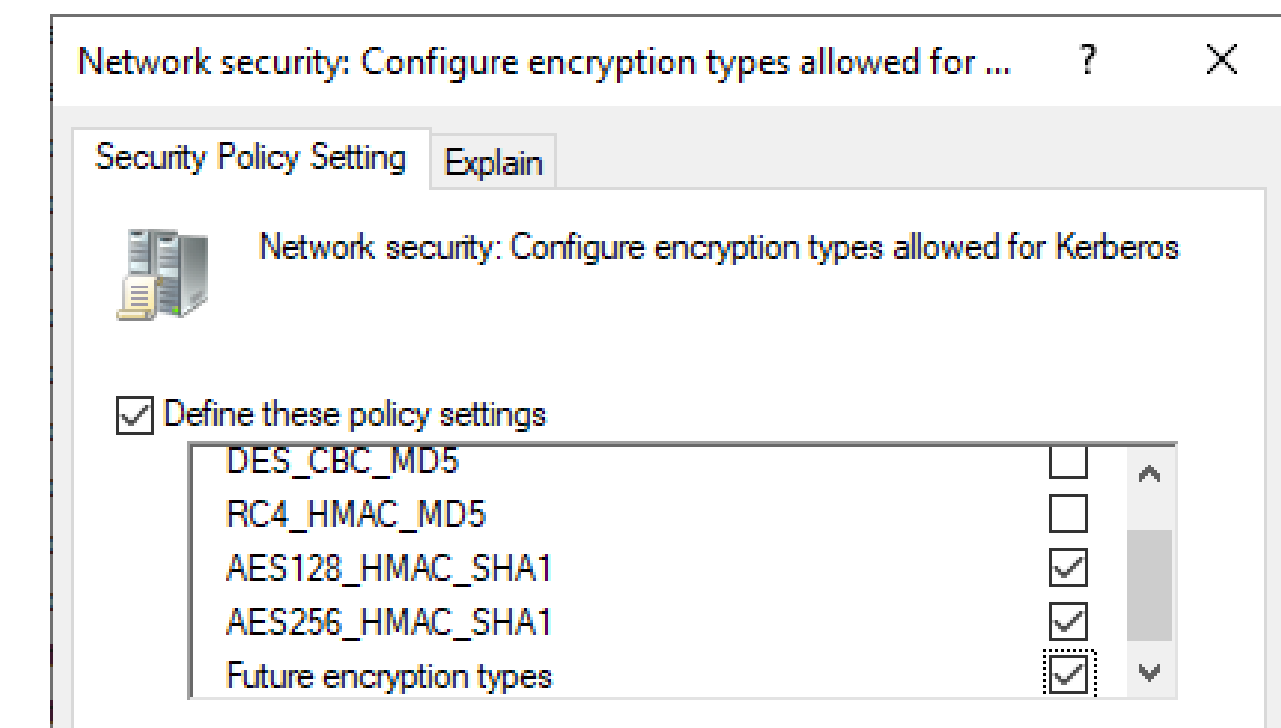
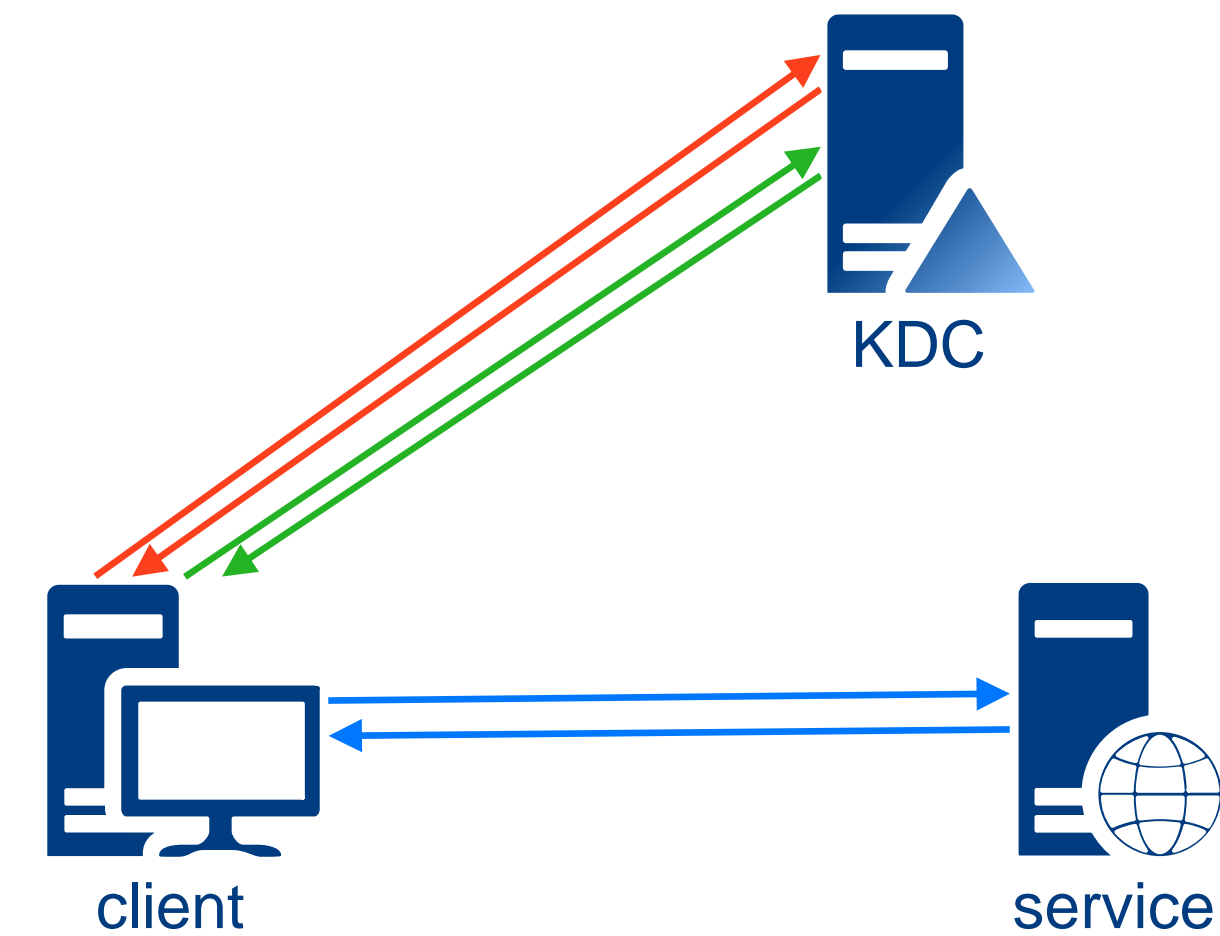
5. AP REQ – APplication request – аутентификация TGS тикетом (TGS + authenticator)
6. AP REP – ответ сервера (OK)



Kerberos тикеты и криптография

Клиент: «хочу подключиться к сервису SMB на узле *files.domain.local*»

- › говорят “ticket” – помнят о связке “*ticket + session key*”;
- › говорят “сервис” – имеют в виду “сервис с определённым SPN и работающим из под определённой учётной записи”;
- › секреты бывают:
 - › des – устарел;
 - › rc4 (== NT хеш) – не солёный;
 - › aes128/aes256 – соль у пользователя: “DOMAIN.FQDNusername”;
- › однако реализовано 16+ криптографических алгоритмов¹;
 - › бывает слабая криптография: CVE-2022-33647/CVE-2022-33679¹;
- › rc4 ~~использовался~~ используется по умолчанию для TGS тикетов:
 - › только у компьютеров по умолчанию включены aes128/256;
 - › KB5021131 (CVE-2022-37966) – предоставили управляемость логикой выбора протокола по умолчанию;



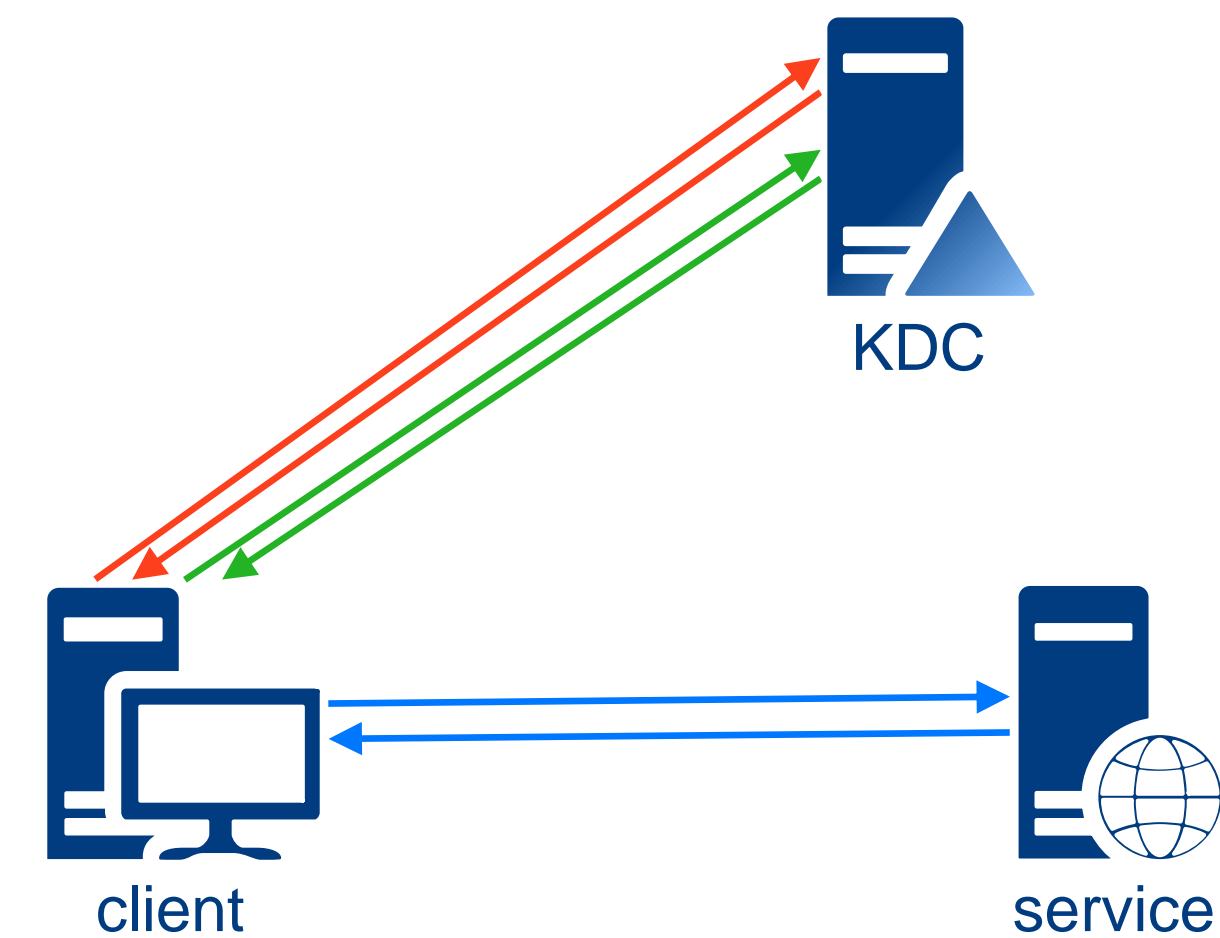
¹ <https://googleprojectzero.blogspot.com/2022/10/rc4-is-still-considered-harmful.html>

Kerberos – содержимое тикетов

Основные компоненты Kerberos Response:

- › метаданные (зашифрованные для пользователя):
 - › пользователь, FQDN;
 - › время жизни тикета, флаги (**forwardable**, ...), ...;
 - › **сессионный ключ**;
- › Kerberos тикет:
 - › не зашифрованные данные:
 - › **имя сервиса**
 - › зашифрованные данные
 - › метаданные, флаги;
 - › **сессионный ключ**;
 - › PAC (Privileged Attribute Certificate) :
 - › пользователь (имя, статус, ...), его RID, членство в группах (RID групп), ...;
 - › **подпись сервиса** (на секрете сервиса);
 - › **3 подписи¹ на секрете KDC** (krbtgt);

Доступа у сервиса к KDC может не быть



¹ две дополнительных подписи появились вследствие CVE-2020-17049 (“bronze bit”) и CVE-2022-37967

Kerberos – keylist

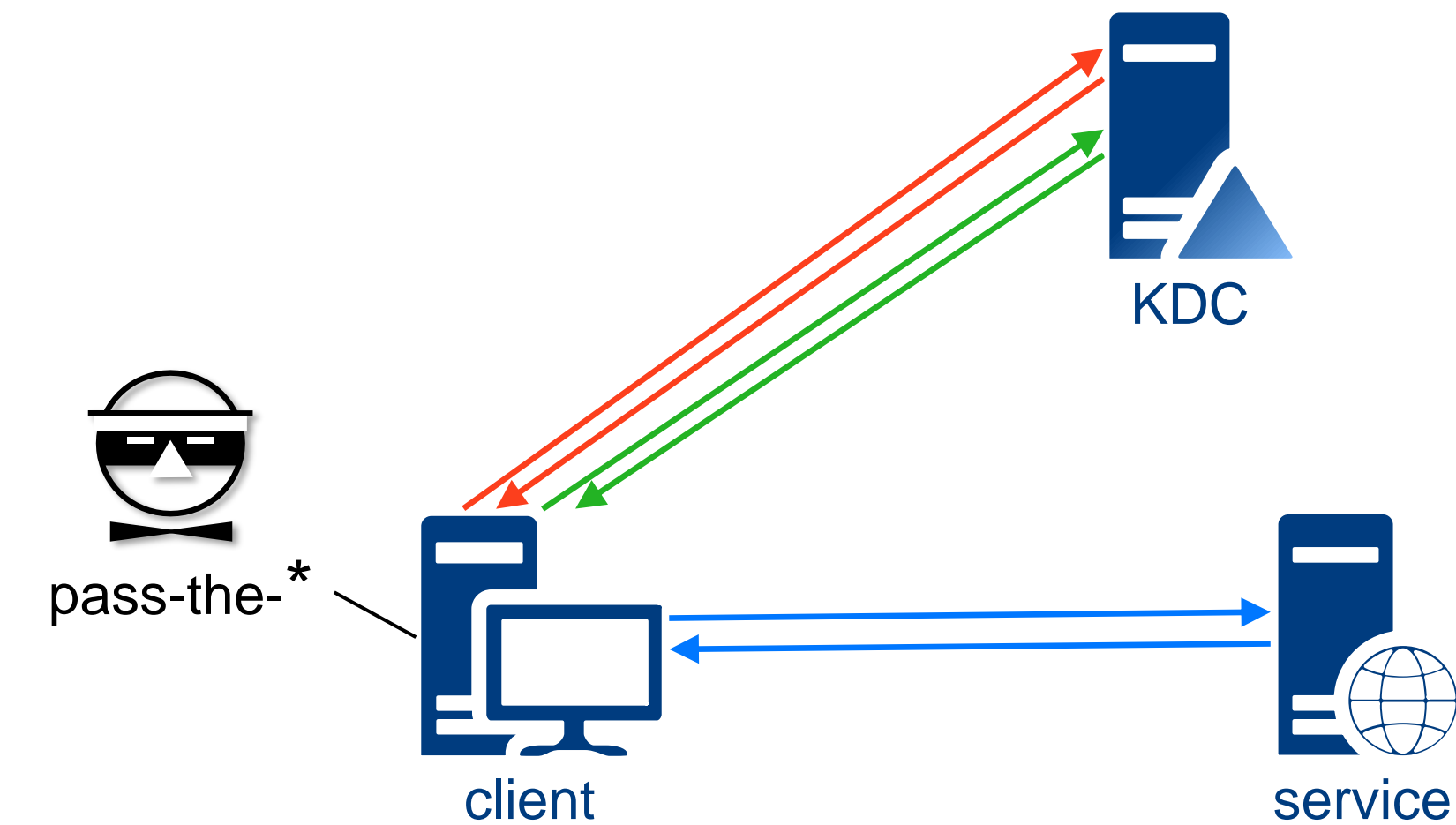
Как хранят пароли сервисы:

- › сервис на windows:
 - › при запуске сервиса из под local service/network service/... – используется УЗ компьютера;
 - › при запуске сервиса из под отдельной учётной записи – windows прикопает пароль УЗ;
 - › при запуске сервиса из под gMSA – windows заберёт пароль при старте сервиса с DC (*best practice*);
- › сервис не под windows:
 - › в *.keytab файле будут храниться хеши, помеченные некоторой текстовой строкой (e.g. UPN или SPN или ...);

Как хранятся тикеты:

- › windows – в памяти;
- › не windows – в файлах *.ccache;
- › пентестеры – *.ccache, *.kirbi, ...;

Следствие: атаки pass-the-key/pass-the-ticket/overpass-the-hash.



Kerberos: *-roasting атаки

› *-roasting атаки – это брутфорс атаки на перебор пароля учётной записи.

AsREQroasting

› AS REQ содержит структуру (включая timestamp) зашифрованную на секрете пользователя;

AsREProasting

› Для учётных записей с флагом `DONT_REQ_PREAUTH`, KDC предоставляет AS REP без пре-аутентификации. AS REP содержит информацию зашифрованную на секрете пользователя;

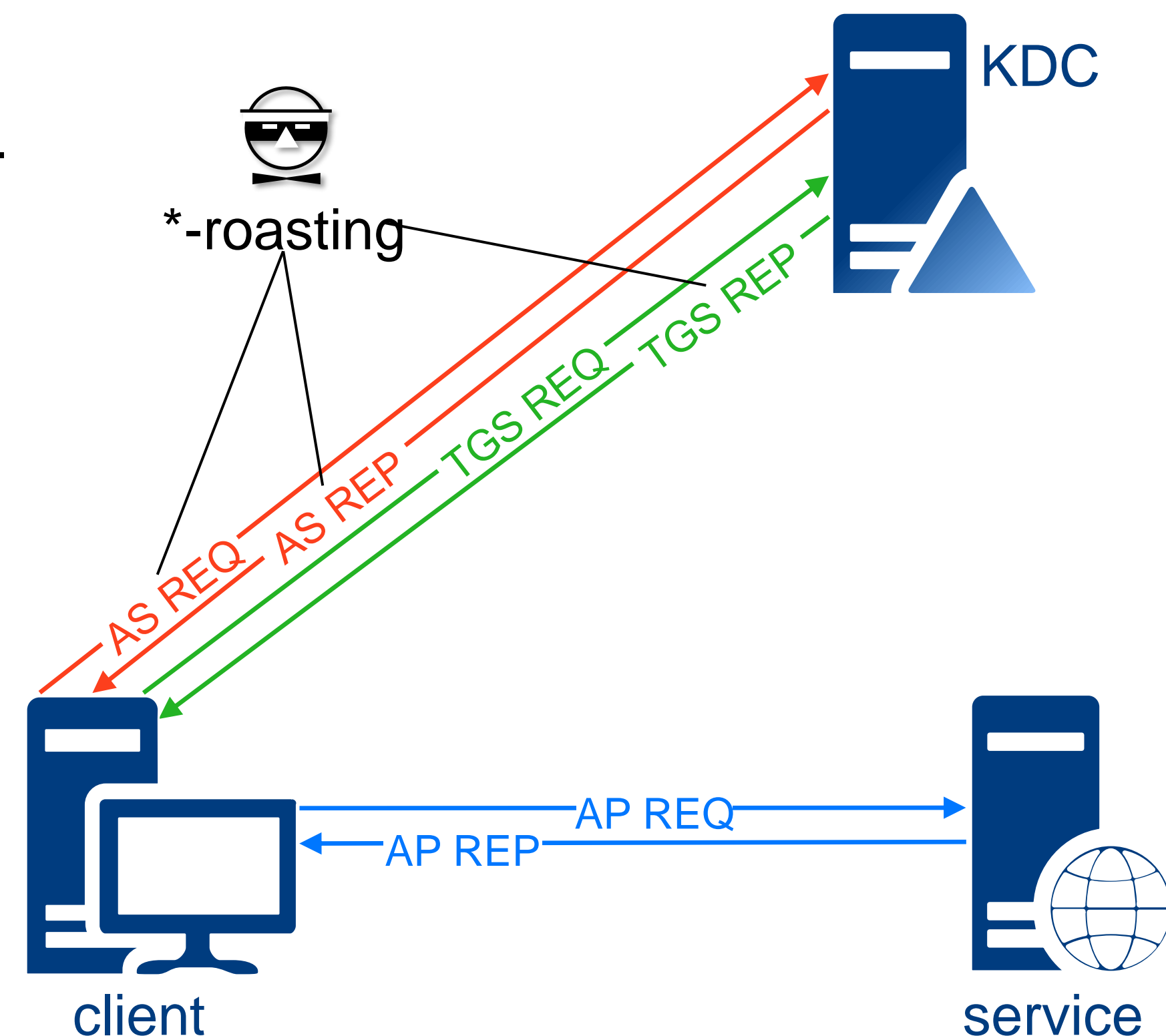
Kerberoasting

› Для каждого сервиса можно запросить TGS тикет, который будет зашифрован на секрете сервисной учётной записи;

› Задать сервис можно используя различные Principal Name Types¹;

› Kerberoasting без аутентификации:

› В AS REQ можно указать SPN сервиса (вместо `krbtgt/DOMAIN.COM`). `DONT_REQ_PREAUTH` учётная запись может без аутентификации запрашивать тикеты к сервисам²;



Account options:

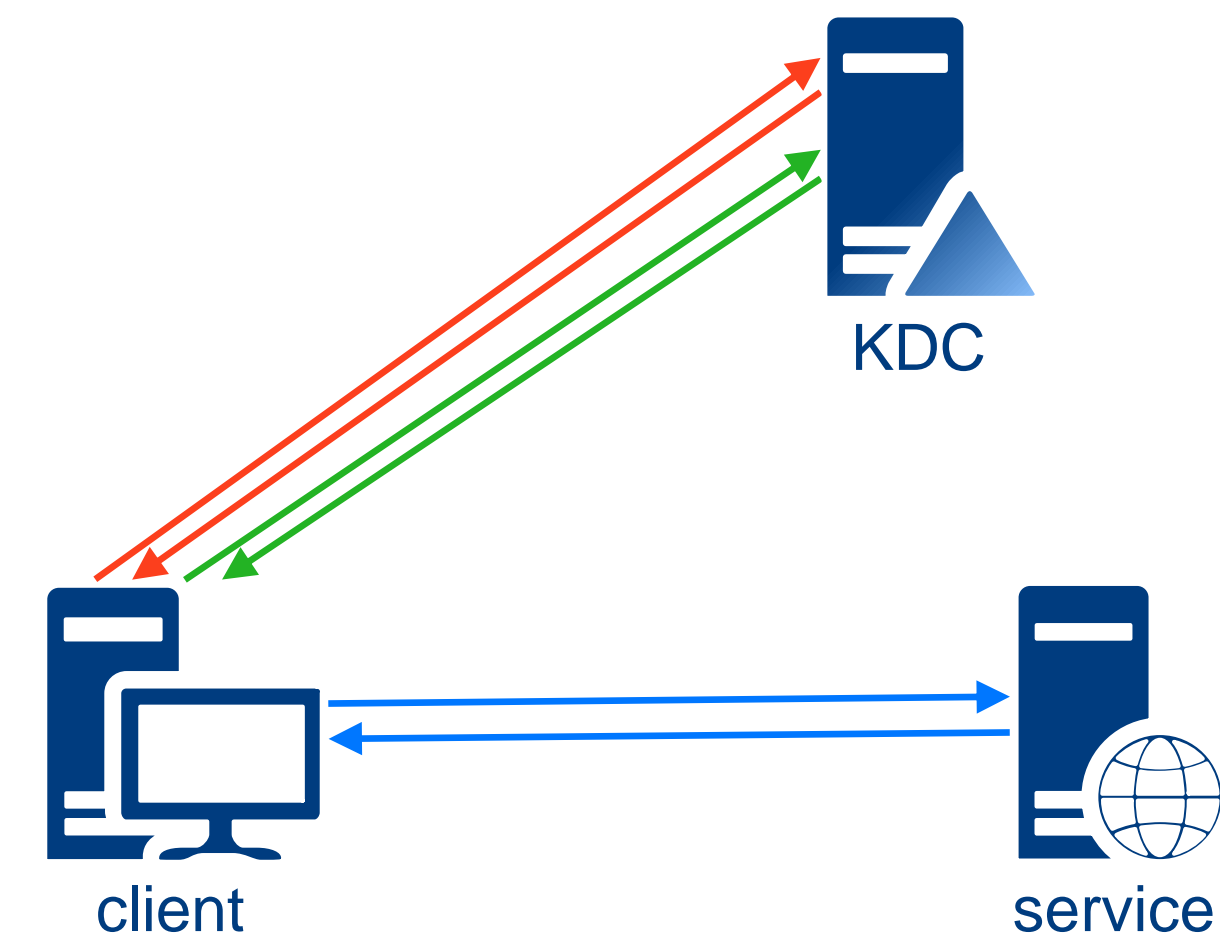
- ☐ Use only Kerberos DES encryption types for this account
- ☐ This account supports Kerberos AES 128 bit encryption.
- ☐ This account supports Kerberos AES 256 bit encryption.
- ☒ Do not require Kerberos preauthentication

¹ <https://swarm.ptsecurity.com/kerberoasting-without-spns/>

² <https://www.semperis.com/blog/new-attack-paths-as-requested-sts/>

Kerberos – golden, silver, ... tickets

- › TGT тикет – зашифрован на секрете `krbtgt` аккаунта;
 - › PAC содержит подпись, на секрете `krbtgt` аккаунта;
- › TGS тикет – зашифрован на секрете аккаунта из под которого работает сервис;
 - › PAC содержит подпись, на секрете аккаунта из под которого работает сервис;
 - › PAC содержит 3 подписи, на секрете `krbtgt` аккаунта;



Следовательно:

- › **(golden ticket)** зная секрет `krbtgt` аккаунта можно **подделать TGT тикет** любого пользователя;
 - › первые 20 минут DC не проверяет golden ticket¹;
 - › начиная с KB5008380 (CVE-2021-42287) – TGT частично проверяется, так что аккаунт должен “существовать”;
- › **(silver ticket)** зная секрет сервисного аккаунта можно “почти” **подделать любой TGS тикет**;
 - › в некоторых случаях¹ тикет может быть верифицирован на DC (“`ValidateKdcPacSignature`”²);
 - › такой тикет не подойдёт для использования в constrained delegation;
 - › используя S4U2Self можно получить полностью корректный TGS тикет к сервису;

¹ <https://passing-the-hash.blogspot.com/2014/09/pac-validation-20-minute-rule-and.html>

² <https://blog.netwrix.com/2023/01/10/what-is-the-kerberos-pac/>

³ diamond / sapphire ticket – тикеты, сформированные по образцу настоящих тикетов, с добавлением необходимых изменений (членств в группах).

Kerberos тикеты – в роли токенов аутентификации

TGT/TGS – это токены аутентификации с подписью

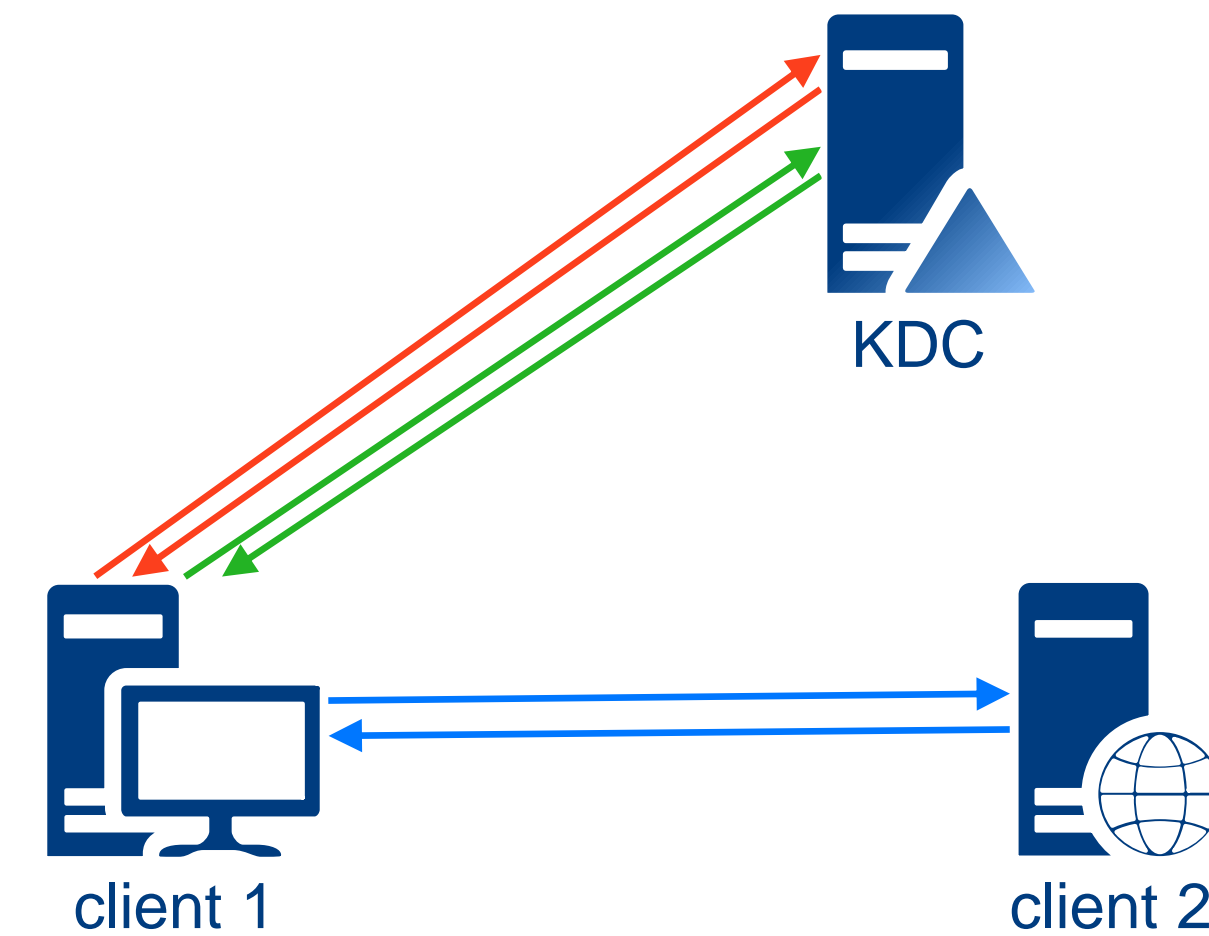
- › Тикеты TGT/TGS – нельзя отозвать (только ротировать общие секреты (`krbtgt` – дважды));
 - › по умолчанию: время жизни TGT - 10 часов, TGS – 8 часов;
- › TGS тикеты работают даже после блокировки учётной записи (TGT работает свои первые 20 минут);
- › TGT/TGS тикеты могут содержать в себе “старые” членства в группах;
- › TGT тикеты работают даже после смены пароля пользователем и разблокировки;

- › У сервиса может не быть сетевого доступа к KDC вообще;

- › При инвентаризации, не используемая УЗ – это такая УЗ которая, и не логинится (`lastLogonTimestamp`¹) и к ней не запрашивают TGS тикеты (или отсутствуют SPN записи);

¹ `lastLogonTimestamp` атрибут реплицируется между DC, но обновляется при логоне пользователя, только если устарел больше, чем на `msDS-LogonTimeSyncInterval`. (в отличие от `lastLogon` – обновляется всегда при логоне, но не реплицируется между DC)

Kerberos – U2U¹ режим



- › Основное отличие от U2S:
 - › SPN не нужен. Вместо этого клиент получает от сервиса TGT тикет сервиса (без сессионного ключа);
- › Запрос TGS для U2U:
 - › В TGS REQ передаётся “additional ticket”, содержащий TGT тикет сервиса;
 - › **TGS тикет к сервису шифруется** не на основе секрета сервиса, а **на сессионном ключе** из TGT тикета сервиса.

¹ <https://www.rfc-editor.org/rfc/rfc4120#section-3.7>

Kerberos – задача делегации

Постановка задачи:

сервис: «мне нужно ходить к другому сервису от имени пользователя»

› однако, мы не хотим запрашивать и передавать пароль (или хеш) пользователя;

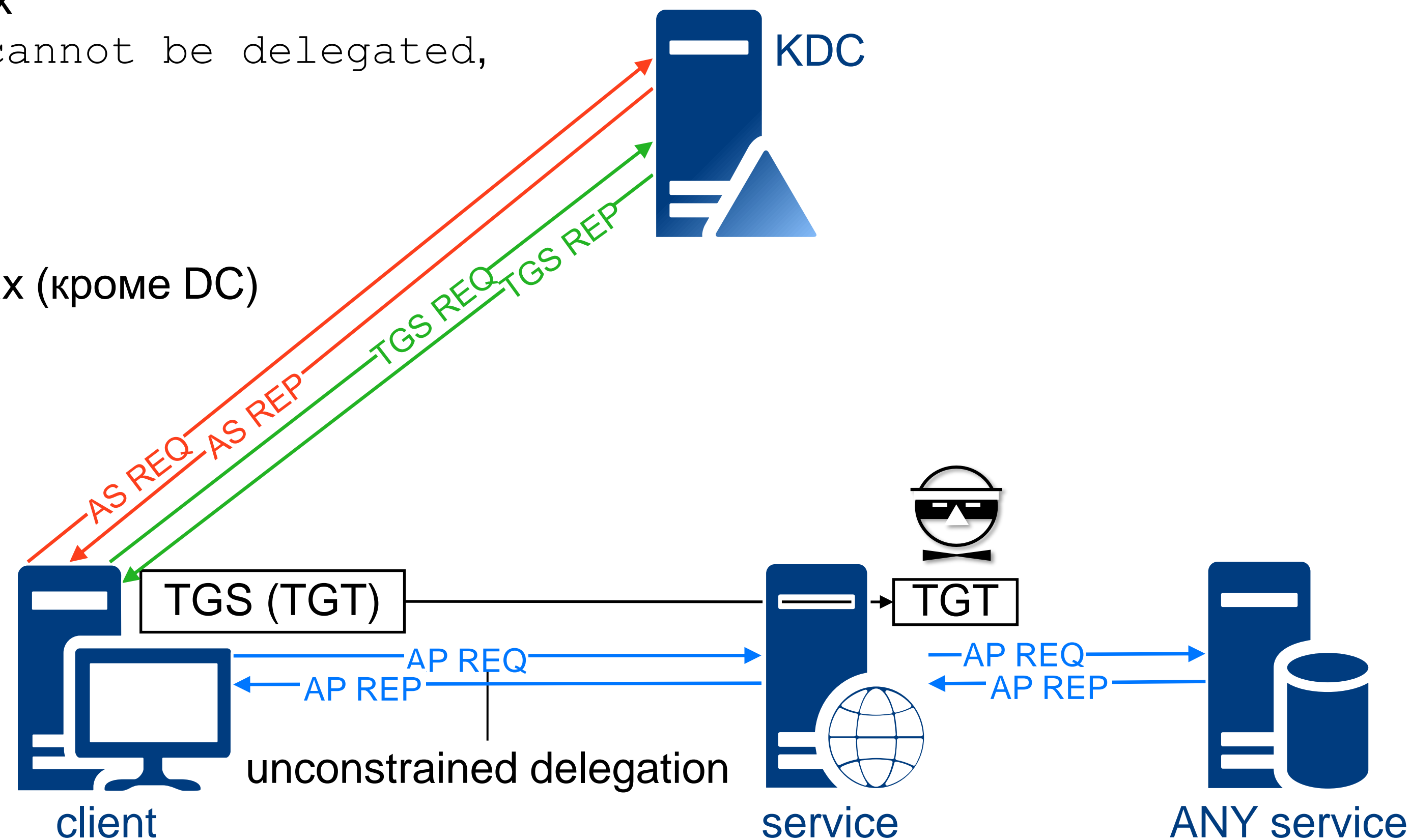
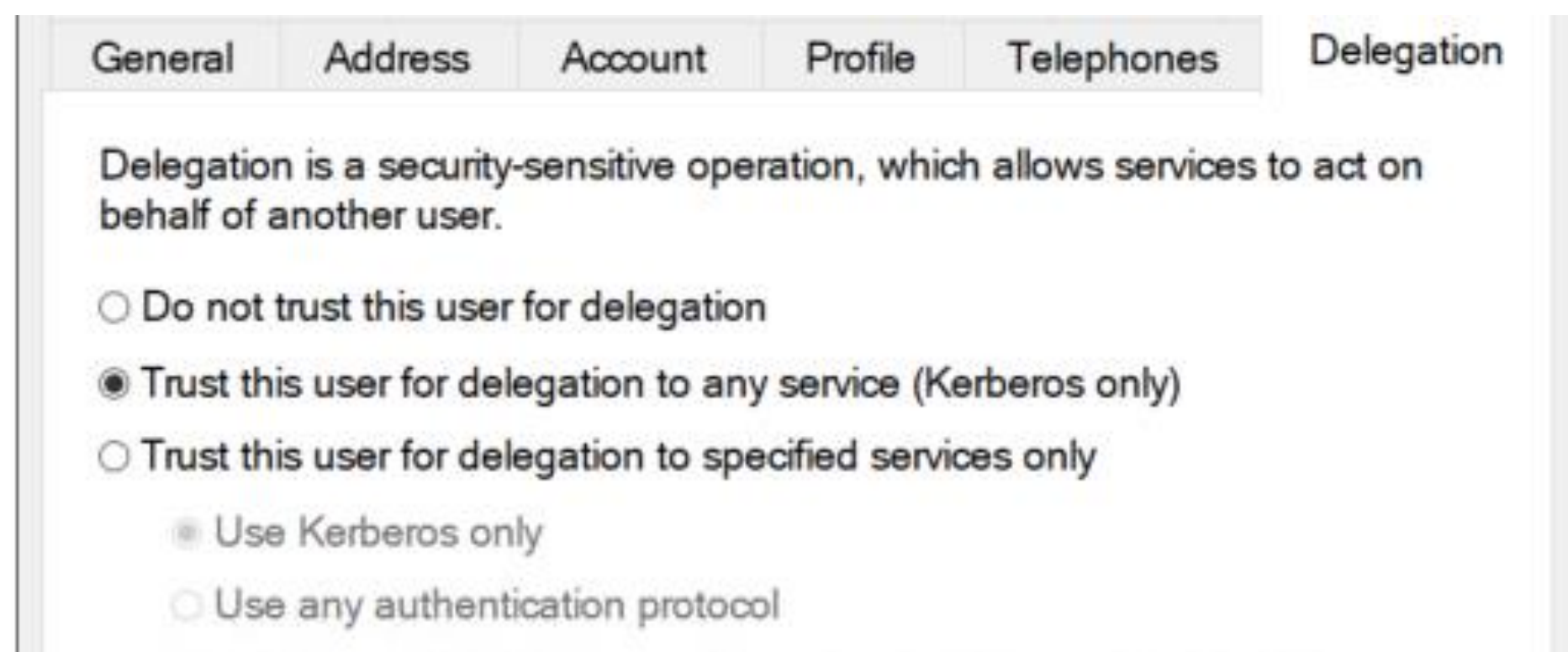
Kerberos – unconstrained delegation

очень не безопасно:

- › позволяет имперсонировать пришедшего пользователя БЕЗ ограничений;
- › пользователя (например, домен контроллер) можно “позвать” используя ssrf-like трюки;
- › включается: флагом `TRUSTED_FOR_DELEGATION` в атрибуте `userAccountControl`;
- › *ограничение:*
 - › Не работает для пользователей, защищённых или флагом `Account is sensitive and cannot be delegated`, или членством в группе `Protected Users`;

рекомендация:

- › не использовать ни при каких обстоятельствах (кроме DC)
- альтернатива: `constrained delegation`
- › следить, что отключено на domain trusts (с 2019.07 отключено по умолчанию)



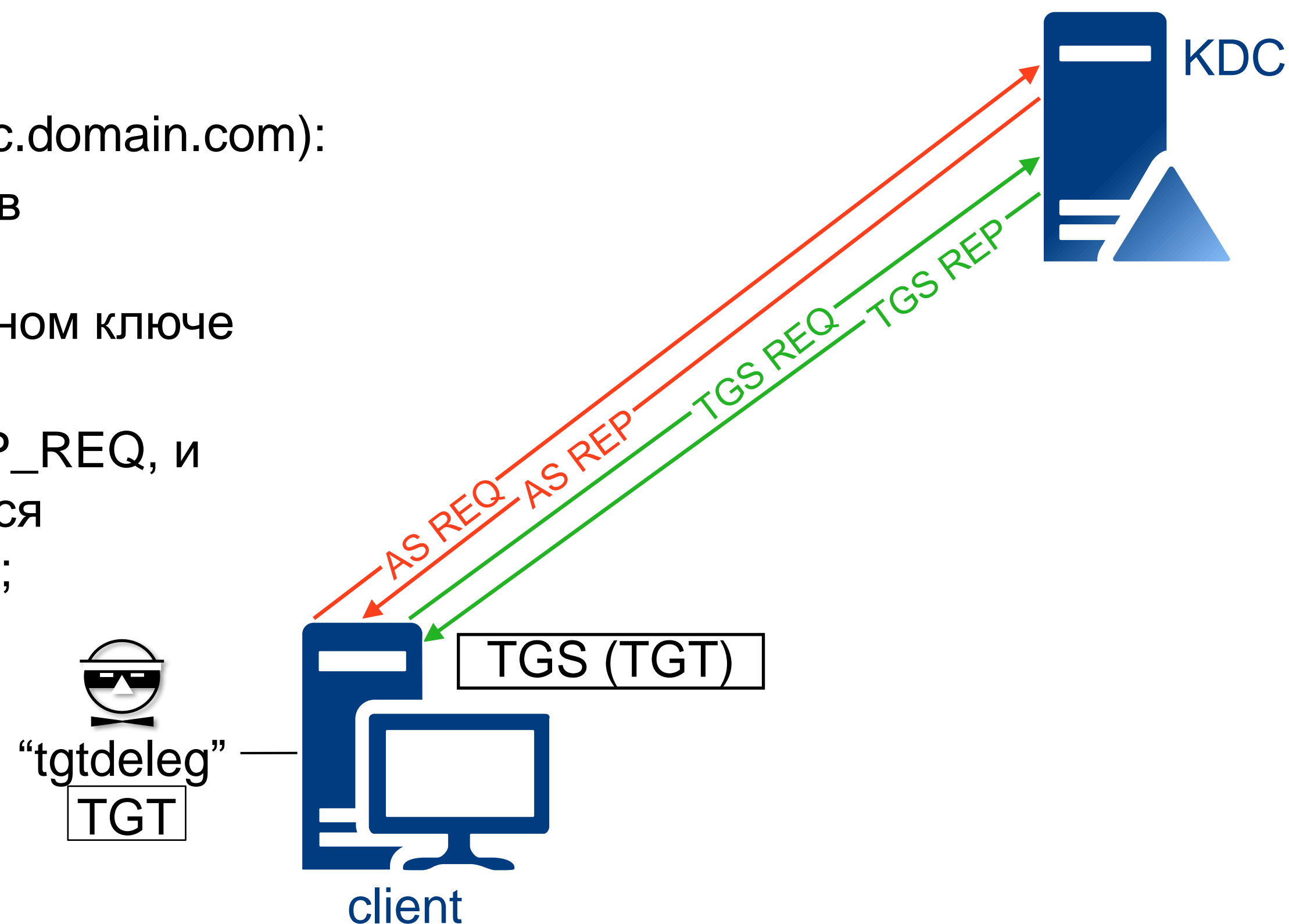
Kerberos – tgtdeleg трюк

последствия unconstrained delegation:

При получении RCE (не привилегированной) из под учётной записи, злоумышленник может получить TGT^{1, 2} (@gentilkiwi):

При запросе TGS на сервис с unconstrained delegation (cifs/dc.domain.com):

- › TGT тикет (+ сессионный ключ от него) инкапсулируется в “authenticator”, передающийся в AP_REQ.
И “authenticator”, и тикет в нём зашифрованы на сессионном ключе TGS;
- › winapi вынуждено отдавать пользователю в userland и AP_REQ, и сессионный ключ. (к примеру, чтобы аутентифицироваться пользовательским клиентом по собственному протоколу);

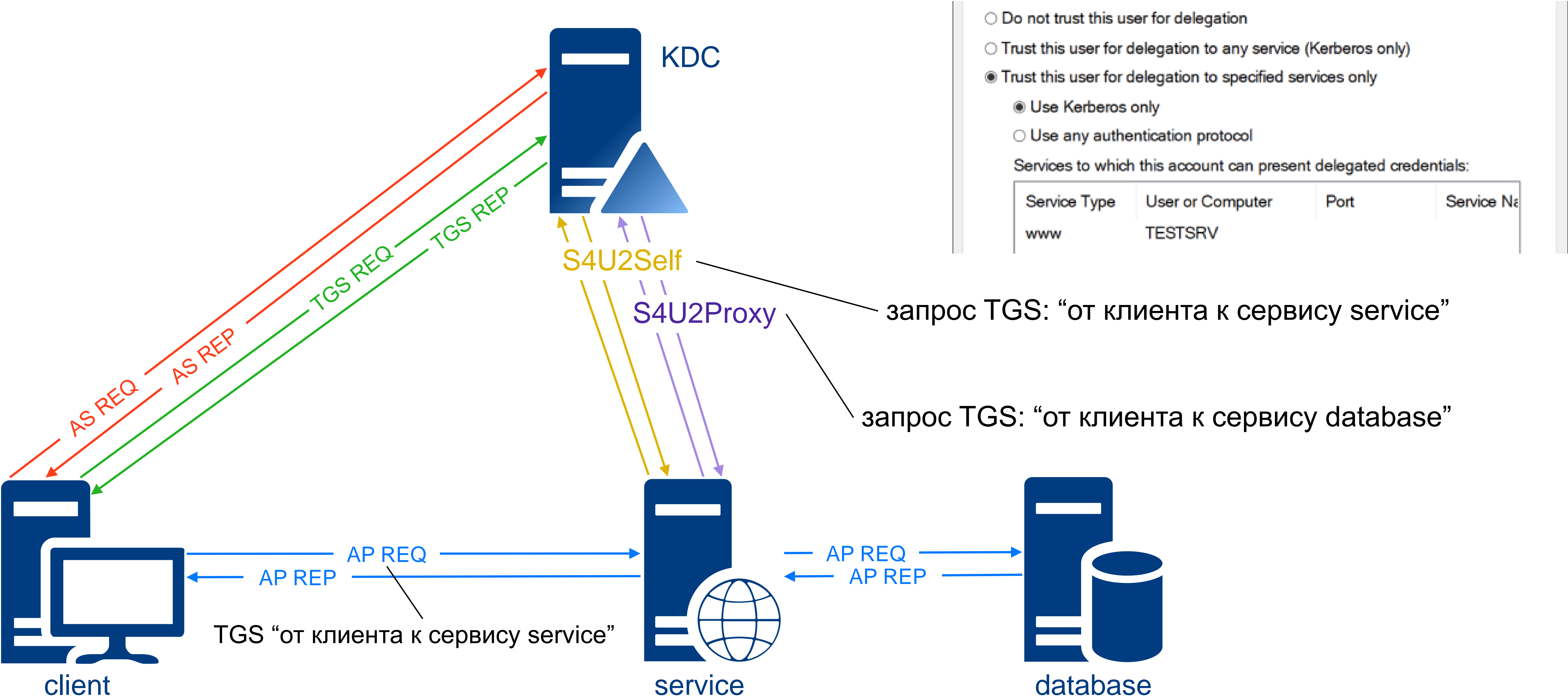


Не работает для учётных записей, защищённых от делегации.

¹ kekeo (2018): https://github.com/gentilkiwi/kekeo/blob/4fbb44ec54ff093ae0fbe4471de19681a8e71a86/kekeo/modules/kuhl_m_tgt.c#L189-L327

² rubeus: <https://github.com/GhostPack/Rubeus/blob/f6685f4b714295395f816e2c8839b750080c2c2b/Rubeus/lib/LSA.cs#L1318-L1565>

Kerberos – constrained delegation



Kerberos – constrained delegation

Account options:

- ☒ Account is sensitive and cannot be delegated
- ☐ Use only Kerberos DES encryption types for this account
- ☐ This account supports Kerberos AES 128 bit encryption.
- ☐ This account supports Kerberos AES 256 bit encryption.

S4U2Self

- › Сервис А может попросить у KDC тикет TGS “от клиента к самому себе”.
- › *ограничение*: **forwardable** флага **не** будет¹, если:
 - › Клиент защищён группой Protected Users или флагом Account is sensitive and cannot be delegated;
 - › У сервиса не включена constrained delegation “use any authentication protocol” (TRUSTED_TO_AUTH_FOR_DELEGATION);

S4U2Proxy

- › Сервис А может попросить у KDC тикет TGS “от клиента к другому сервису В”;
- › *ограничения*:
 1. только при предъявлении TGS тикета “от клиента к самому себе”;
 2. constrained delegation (CD):
 - › предъявленный тикет должен иметь флаг **forwardable**;
 - › whitelist “других сервисов” в атрибуте msDS-AllowedToDelegateTo учётной записи сервиса А;
 2. resource-based constrained delegation (RBCD):
 - › “другой сервис” указал доверие сервису А в собственном атрибуте msDS-AllowedToActOnBehalfOfOtherIdentity;
 - › клиент не защищён группой Protected Users или cannot be delegated флагом;
 - › **forwardable** флаг не проверяется;
- › полученный тикет – всегда содержит **forwardable** флаг;

¹ “bronze-bit” KB4598347 (CVE-2020-17049) – forwardable flag указывался только в не подписанной части тикета, поэтому его можно было подделать.
<https://www.netspi.com/blog/technical/network-penetration-testing/cve-2020-17049-kerberos-bronze-bit-overview/>

Kerberos – привилегии для настройки delegation

Unconstrained Delegation

- › требует привилегии `SeEnableDelegationPrivilege` на Домен Контроллере;

Constrained Delegation

- › требует привилегии `SeEnableDelegationPrivilege` на Домен Контроллере и право на изменение содержимого атрибута `msDS-AllowedToDelegateTo` учётной записи;

Resource-Based Constrained Delegation

- › все учётные записи пользователей/компьютеров имеют привилегию на изменение содержимого собственного атрибута `msDS-AllowedToActOnBehalfOfOtherIdentity`;

Kerberos – constrained delegation риски

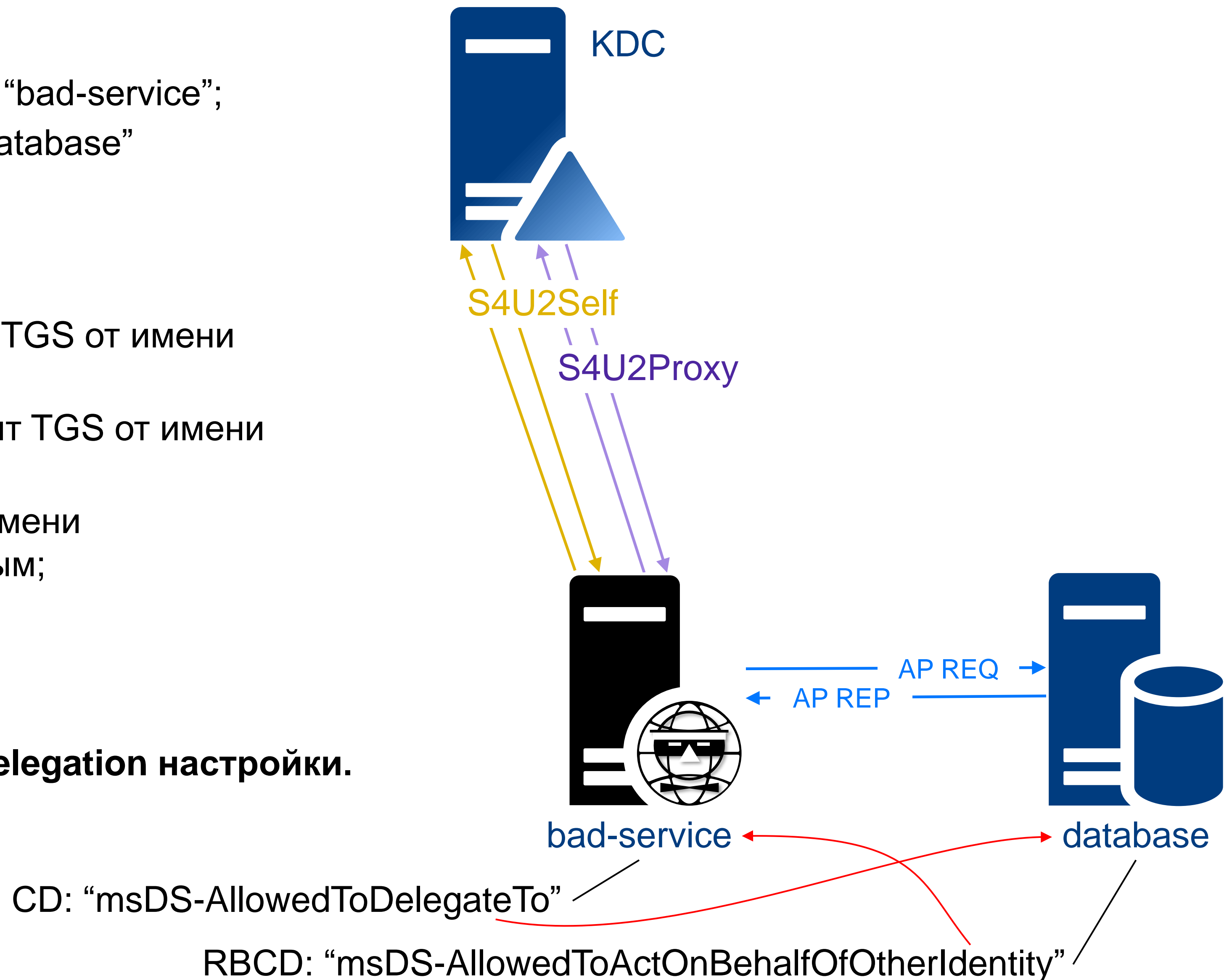
Ситуация:

- › Атакующий контролирует учётную запись “bad-service”;
- › Настроена делегация от “bad-service” к “database” (CD или RBCD);

Риск:

1. Используя S4U2Self атакующий запросит TGS от имени администратора к “bad-service”;
2. Используя S4U2Proxy атакующий запросит TGS от имени администратора к “database”;
3. Атакующий подключится к “database” от имени администратора и получит доступ к данным;

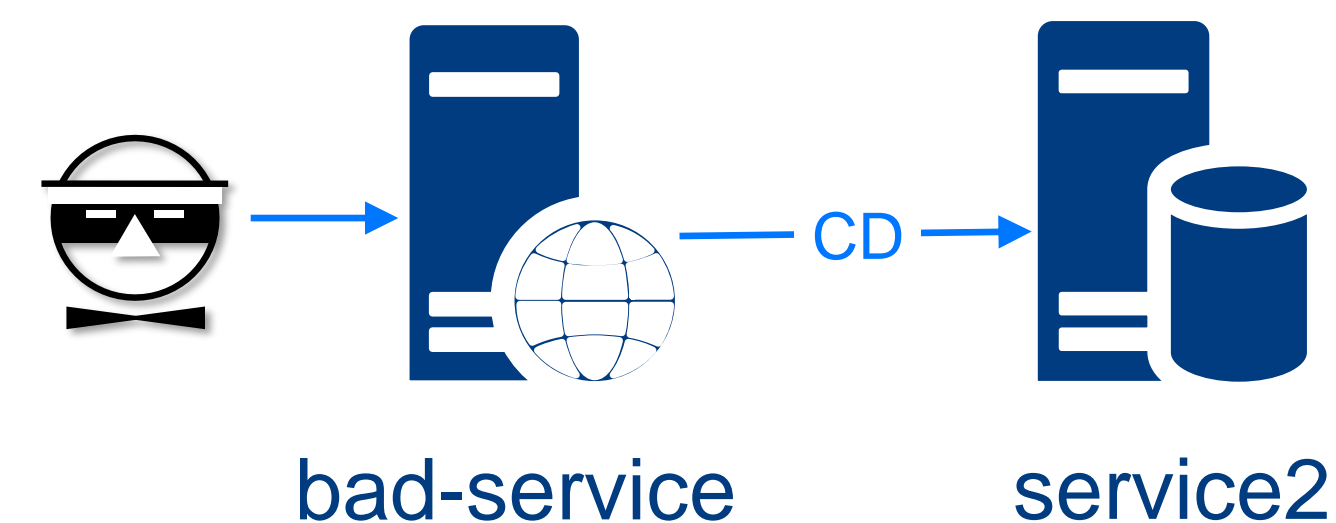
Необходимо мониторить constrained delegation настройки.



Kerberos – поведение злоумышленников

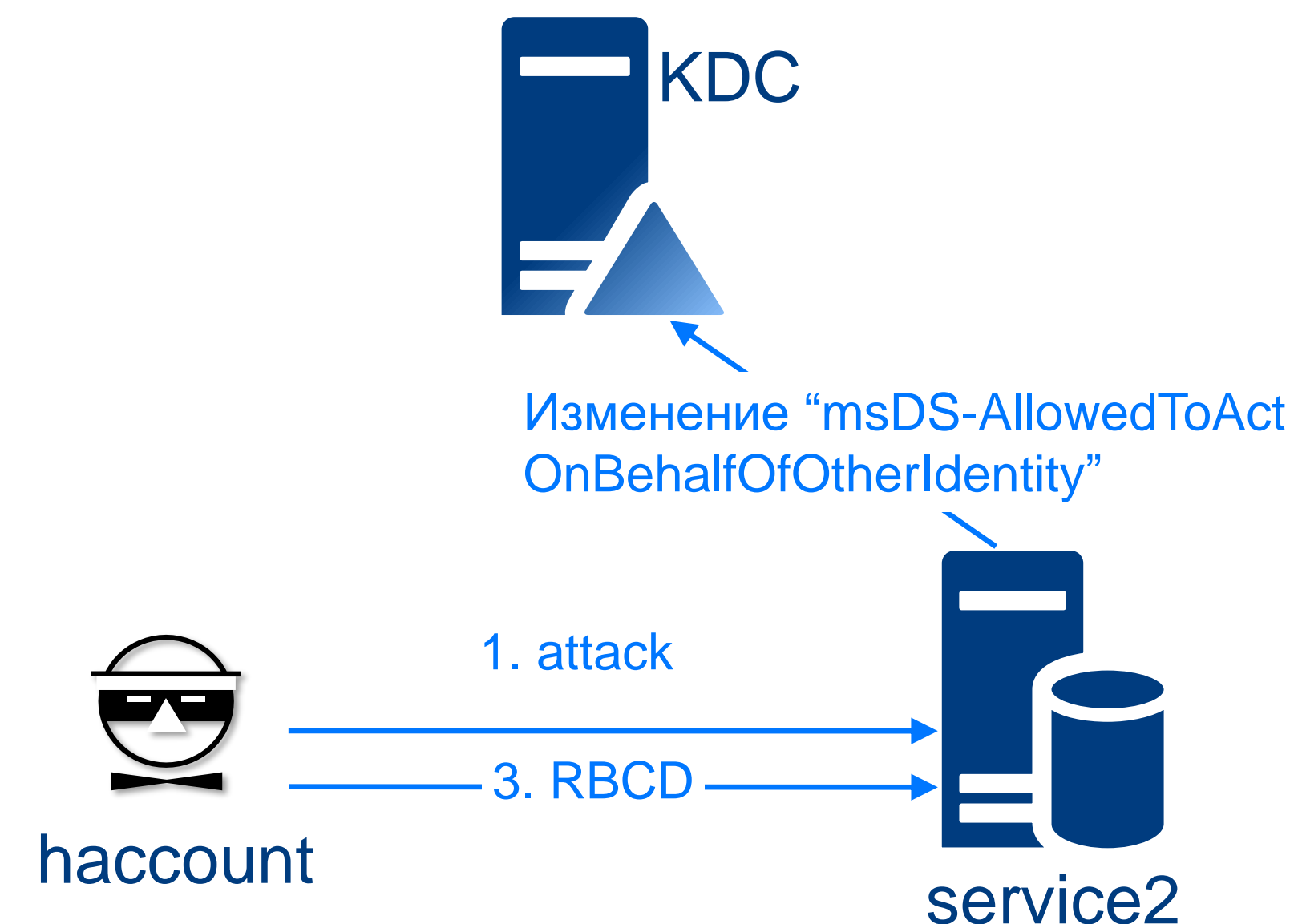
Constrained Delegation

- Атакующий после компрометации “bad-service”, компрометируют все сервисы, к которым разрешена делегация.



Resource-Based Constrained Delegation

- Атакующий строит цепочку эксплуатации:
 - Подбирает подходящую учётную запись “haccount”;
 - 1. выполняет атаку, позволяющую настроить RBCD от “haccount” к “service2”, например:
 - *-relay атака на сервис (с использованием ранее описанных ssrf-like техник);
 - выполнение команд из непривилегированного контекста (NETWORK SERVICE)
 - 2. Компрометирует целевой сервис через RBCD;



Kerberos – delegation hardening (оговорка #1)

servicePrincipalName в Kerberos тикете – не подписан, и может быть подменён.

- › Для сервиса тикет будет корректен, если он сможет его расшифровать на своём секрете.
(валидация SPN - опциональна)

Следствие:

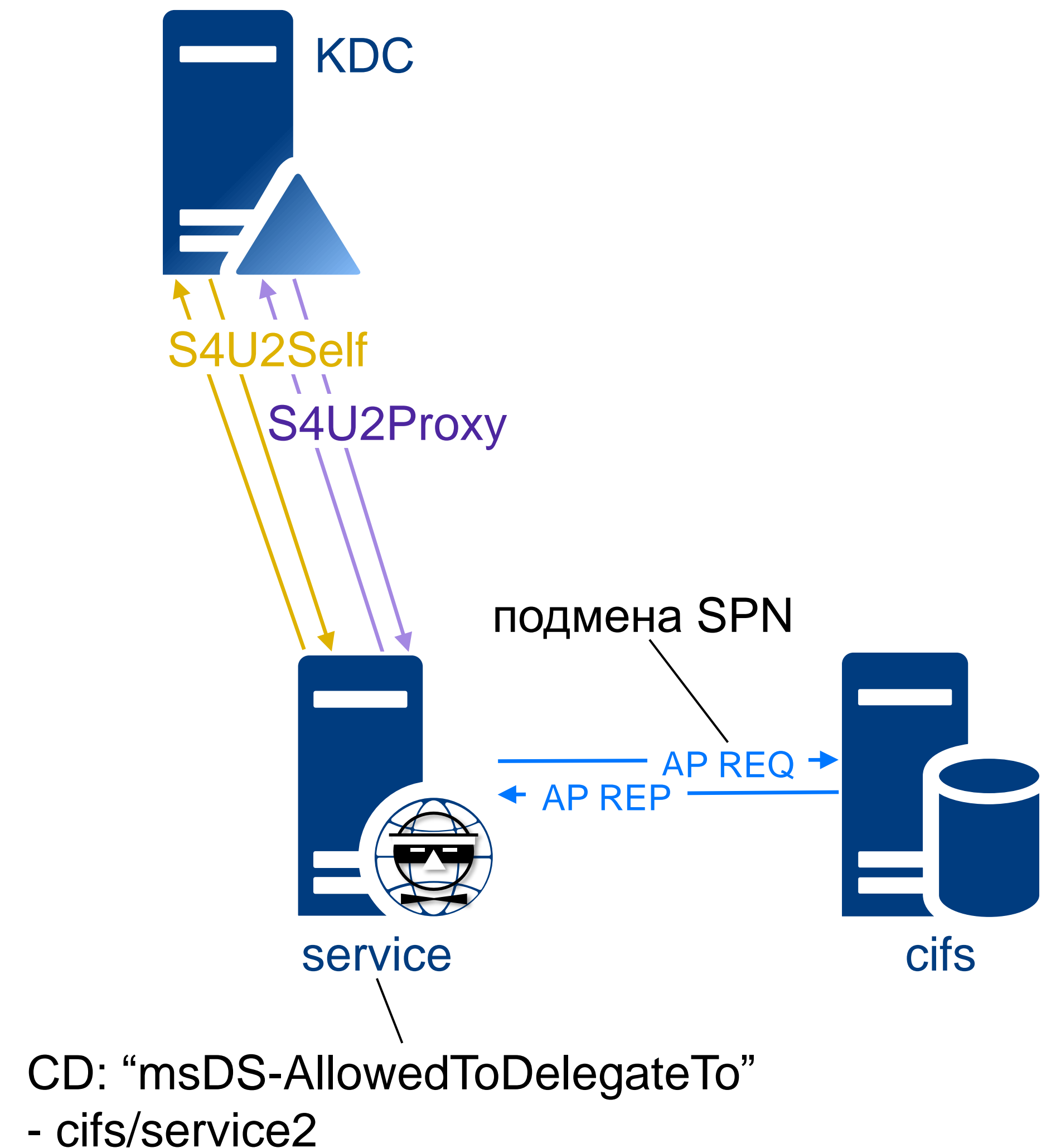
- › Если вы разрешили делегацию к сервису А, то вы разрешили делегацию **ко всем сервисам**, работающим **из под того же самого аккаунта**.

- › актуально для многих служб, работающих из под учётной записи компьютера с SPN “HOST/pc.domain.com”;

“HOST” – раскрывается в 50+ сервисов (задаётся в атрибуте “sPNMappings” объекта “CN=Directory Service”):

“host=dcom,cifs,http,dhcp,rpc,rpcss,spooler,...”;

Внимательно выбирайте SPN при делегации



Kerberos – delegation hardening (оговорка #2)

- › Самый простой способ получить контролируемую учётную запись с SPN:
| **зарегистрировать новый компьютер за счёт** `machineAccountQuota`

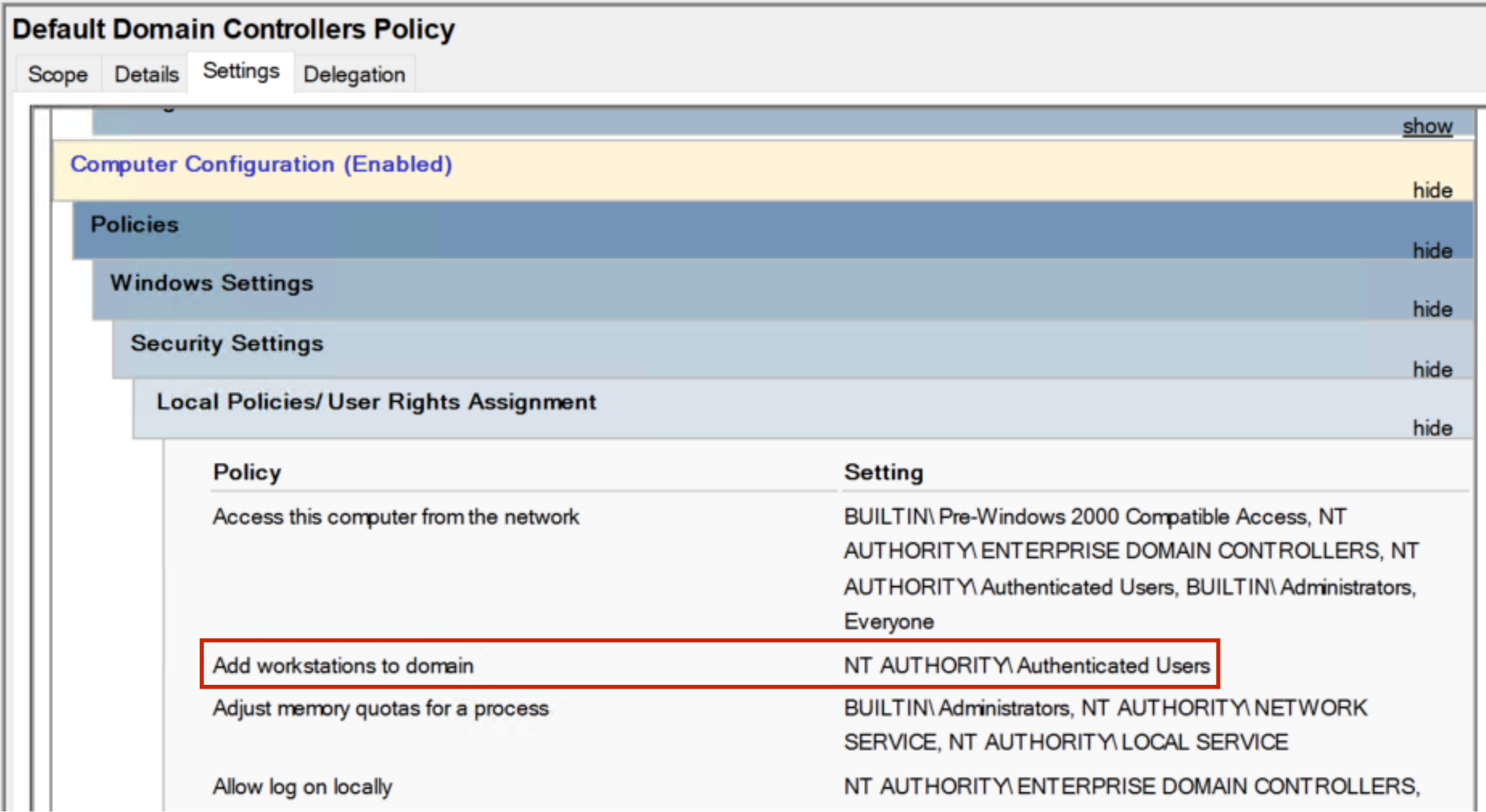
machineAccountQuota (MAC)

- › Всем держателям привилегии SeMachineAccountPrivilege на Домен Контроллере разрешено ввести в домен компьютеры в кол-ве указанном в атрибуте msds-MachineAccountQuota;
 - › расположение по умолчанию: CN=Computers;
 - › владелец нового объекта по умолчанию: Domain Admins;
 - › Привилегии на компьютер по умолчанию¹:

Субъект	Привилегия	Свойство объекта
-----	-----	-----
CONTOSO\myrobot	ExtendedRight,	GenericRead
CONTOSO\myrobot	WriteProperty	Display-Name
CONTOSO\myrobot	WriteProperty	SAM-Account-Name
CONTOSO\myrobot	WriteProperty	User-Logon
CONTOSO\myrobot	WriteProperty	Description
CONTOSO\myrobot	WriteProperty	User-Account-Restrictions
CONTOSO\myrobot	Self	DNS-Host-Name-Attributes
CONTOSO\myrobot	Self	Validated-SPN

- › Компьютер может устанавливать себе SPN;

Субъект	Привилегия	Свойство объекта
-----	-----	-----
NT AUTHORITY\SELF	Self	Validated-SPN



Ограничивайте machineAccountQuota

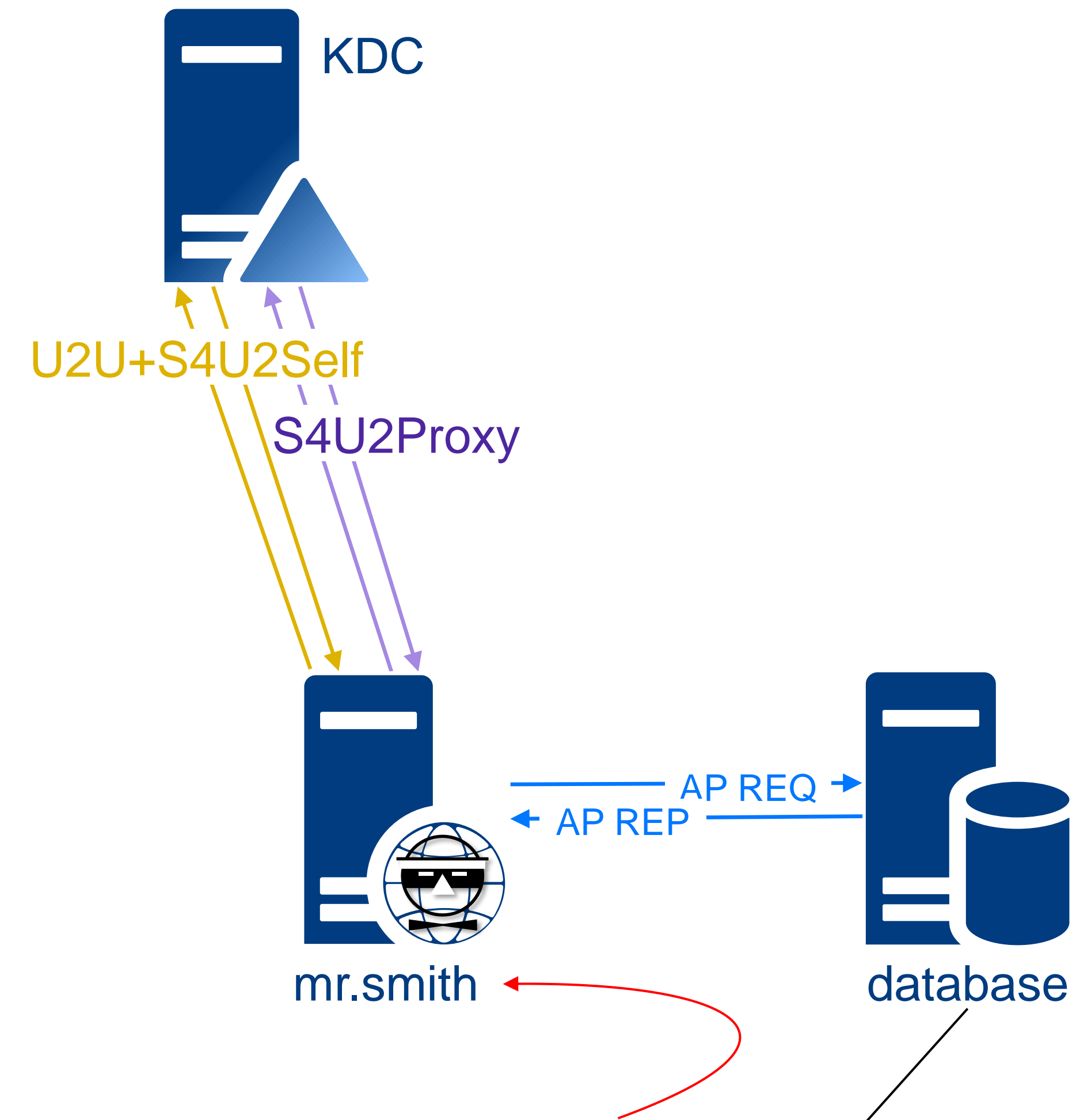
¹ <https://learn.microsoft.com/en-us/windows/win32/adschema/control-access-rights> (extended rights, property sets, validated writes)

Kerberos – delegation hardening (оговорка #3)

Наличие SPN – не обязательно, может быть использована обычная учётная запись¹ (@James Forshaw).

Цепочка действий злоумышленника:

1. (вместо традиционного S4U2Self)
 - › атакующий выполняет U2U+S4U2Self (“mr.smith” запрашивает rc4 TGS от другого пользователя на себя);
 - › “mr.smith” меняет свой NT хеш на DC на сессионный ключ из TGT тикета²;
(теперь TGS тикет зашифрованный на сессионном ключе, также зашифрован и на секрете пользователя);
2. Используя S4U2Proху атакующий запросит TGS от имени администратора к “database”;
3. Атакующий подключится к “database” от имени администратора и получит доступ к данным;



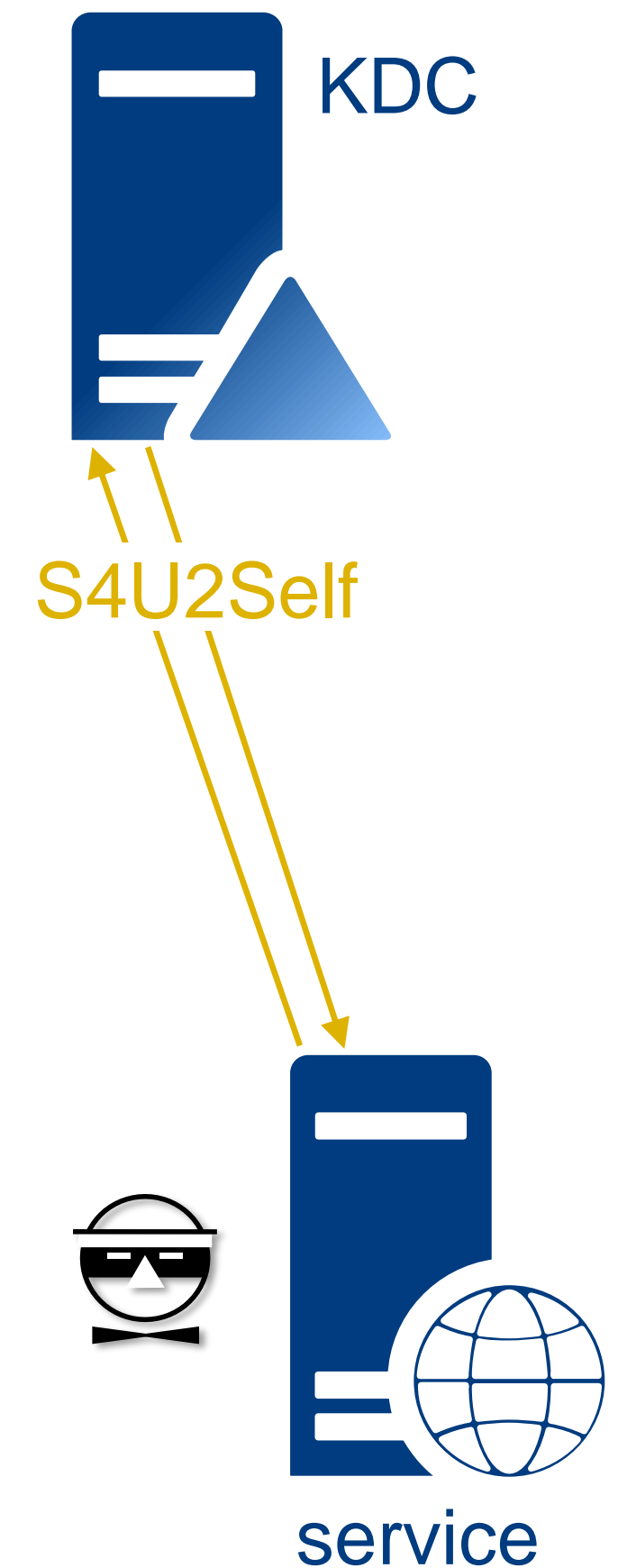
RBCD: “msDS-AllowedToActOnBehalfOfOtherIdentity”

¹ <https://www.tiraniddo.dev/2022/05/exploiting-rbcd-using-normal-user.html> , <https://github.com/GhostPack/Rubeus/pull/137>

² SamrChangePasswordUser - https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-samr/9699d8ca-e1a4-433c-a8c3-d7bebeb01476

Kerberos – “silver” S4U2Self (оговорка #?)

- › S4U2Self может быть использована для получения “silver” тикета от имени любого пользователя;
(даже для пользователя, защищённого “Protected Users” или “not delegated”)



К сожалению, эту «фичу» не отключить, просто в TGS не будет forwardable флага.

Kerberos – delegation hardening (оговорка #?)

Безопасность “Kerberos only” сводится к безопасности “Protocol Transition” с использованием RBCD¹ (@_nwodtuhs).

Исходная ситуация:

› Атакующий скомпрометировал “service” с настроенной CD;

Цепочка действий злоумышленника:

1. Атакующий настраивает для “service” RBCD к самому себе;
2. Используя S4U2Self атакующий запрашивает TGS от имени администратора к “service” (TGS без **forwardable** флага);
3. Используя S4U2Proху (благодаря RBCD) атакующий запрашивает TGS от имени администратора к “service” (TGS содержит **forwardable** флаг);
4. Выполнив S4U2Proху (благодаря “Kerberos only” CD) атакующий запрашивает TGS от имени администратора к “adminka-db01”.

“Kerberos only” – не является мерой безопасности.

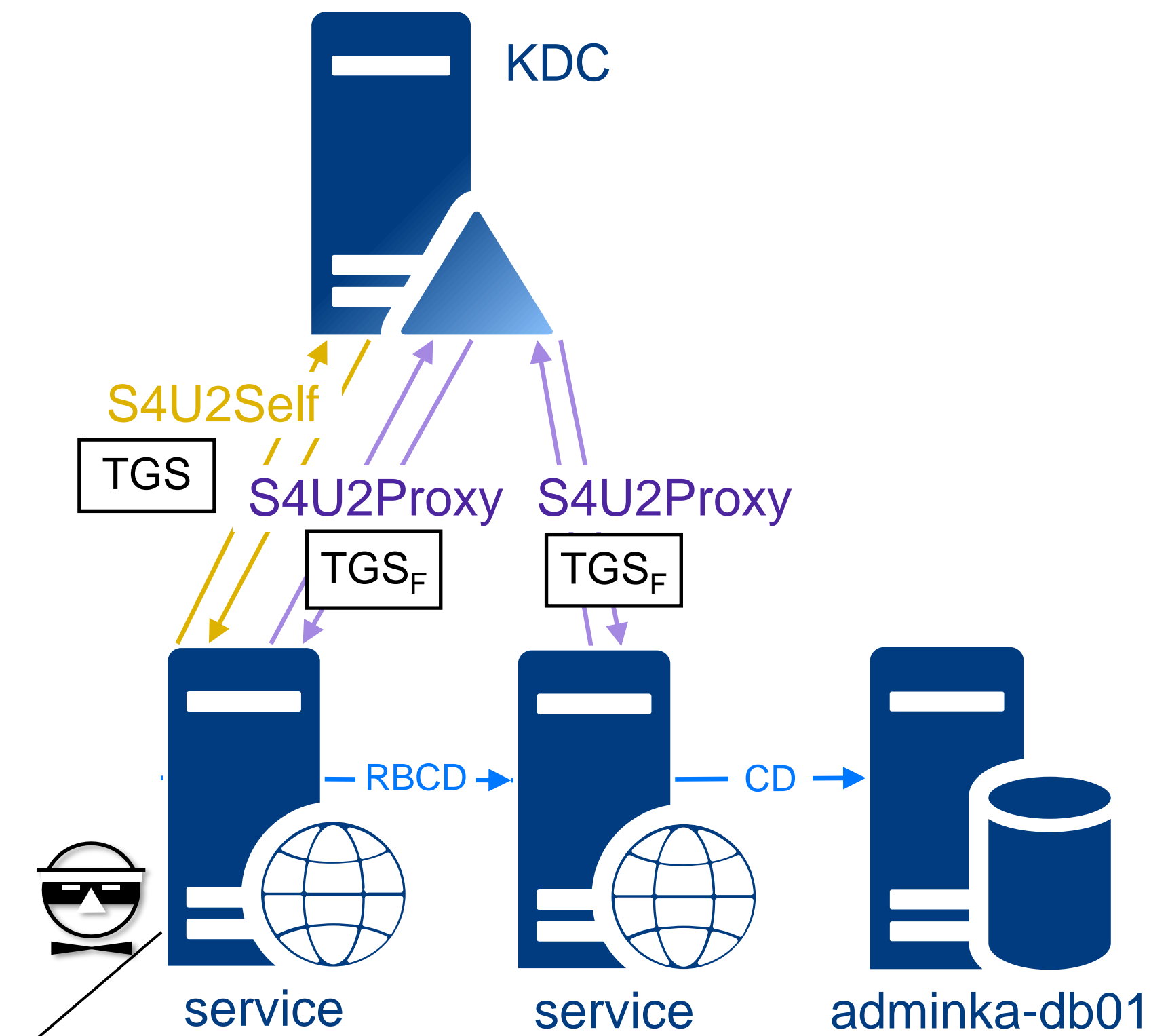
☒ Trust this user for delegation to specified services only

☒ Use Kerberos only

☐ Use any authentication protocol

Services to which this account can present delegated credentials:

Service Type	User or Computer	Port	Se
MSSQLSvc	adminka-db01.constoso.com	1433	



¹ <https://insomnihack.ch/talks-2022/#TQMXCN> (@_nwodtuhs (@shutdown))
“Delegating Kerberos to bypass Kerberos delegation limitation”

CD: “msDS-AllowedToDelegateTo”

RBCD: “msDS-AllowedToAct...”

Kerberos delegation – последствия

Повышение привилегий из под Network Service / Local Service.

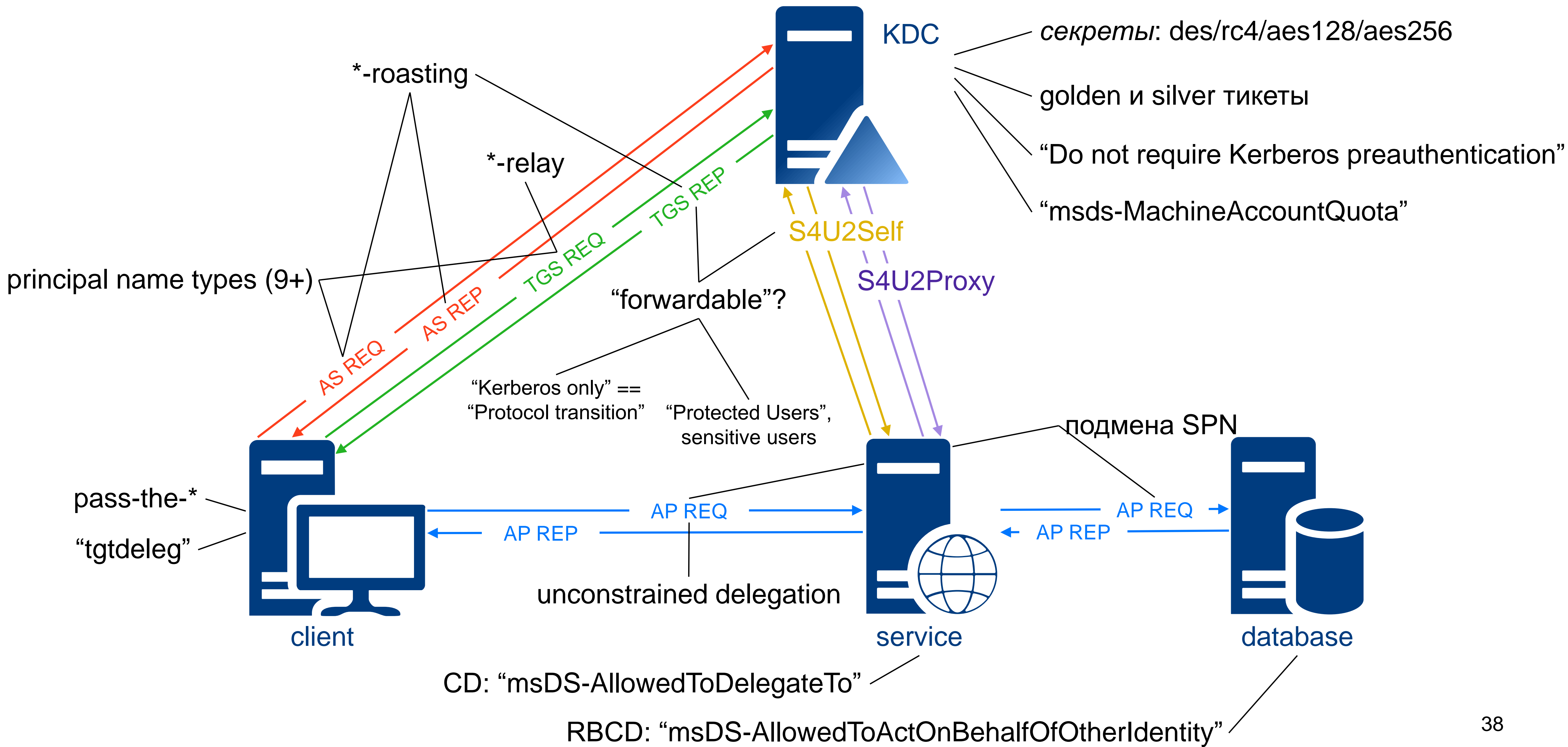
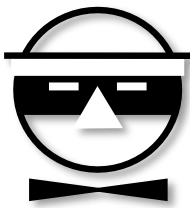
Сетевые взаимодействия из под network/local service выполняются от имени учётной записи компьютера.

- › Атакующий может используя tgtdeleg (для получения TGT тикета) и S4U2Self (для получения TGS тикета от админа к компьютеру) получить TGS тикет для административного доступа к компьютеру.¹
- › Атакующий может из контекста network/local service настроить компрометирующую RBCD для компьютерной УЗ. Далее используя УЗ атакующего имперсонировать администратора и аутентифицироваться на компьютере.

Вывод: у атакующего есть возможности угнать учётную запись из под которой он выполняется (#1).

¹ <https://cyberstoph.org/posts/2021/06/abusing-kerberos-s4u2self-for-local-privilege-escalation/>

Kerberos – Всё вместе



Kerberos – defensive

- › Своевременно обновляйте домен контроллеры (CVE-2022-37967, CVE-2021-42287¹, CVE-2020-17049, ...);
- › Ограничивайте machineAccountQuota;
- › Уменьшайте время жизни TGT/TGS тикетов (*это не отзываемые токены*);
- › Отказывайтесь от rc4 в пользу aes256;
- › Регулярно ротируйте пароль `krbtgt`. Ротируйте пароль сервисных аккаунтов и делайте их сложными, используйте gMSA² всегда, когда это возможно;
 - › gMSA – не панацея, потому что компьютер компрометирует gMSA на использование которой у него есть права;
- › Включайте поддержку “Armoring”³ (“FAST”)⁴;
 - › защищает от AsREQRoasting атаки и ломает атакующим часть их инструментария;
- › Защищайте максимальное число учётных записей (“Protected Users” или “NOT_DELEGATED” флаг);
- › Никогда не используйте “DONT_REQ_PREAUTH” флаг (“Do not require Kerberos preauthentication”);
- › Мониторьте атрибуты в AD отвечающие за делегацию (контролируйте действия администраторов, ловите атакующих);
 - › учитывайте возможные атаки: подмена SPN, несостоятельность “Kerberos only”, ...;
 - › никогда не используйте unconstrained delegation;
 - › по возможности: запрещайте RBCD привилегию на уровне ACL;
- › Мониторьте аномалии;

¹ <https://exploit.ph/cve-2021-42287-cve-2021-42278-weaponisation.html> - как изменить sAMAccountName, прикинувшись домен контроллером.

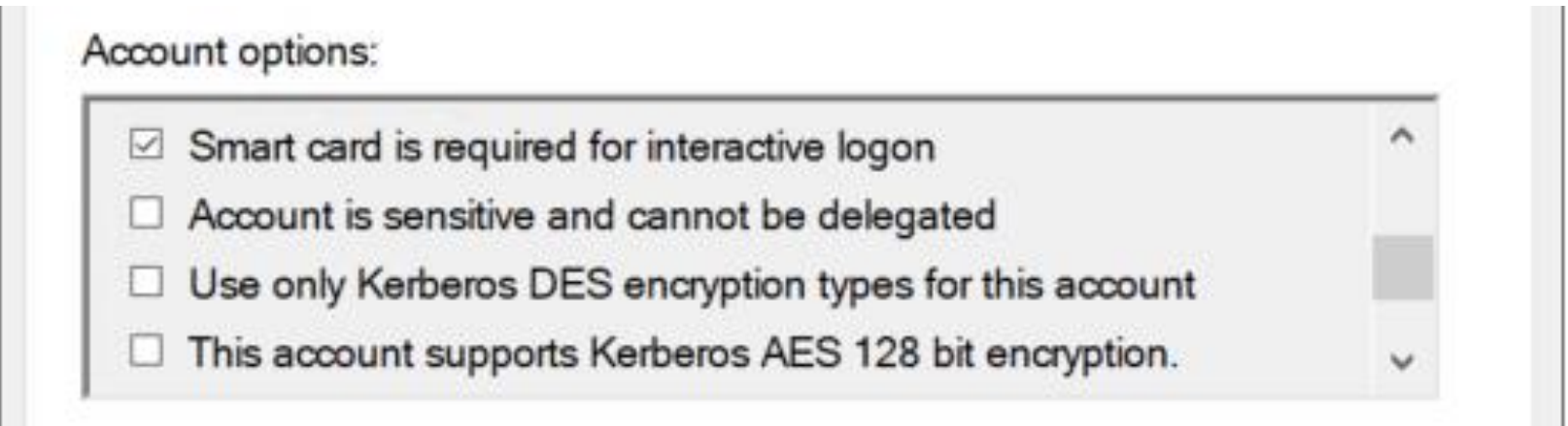
² gMSA (group Managed Service Account) – учётная запись с автоматически управляемым (и ротируемым) паролём.

³ https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-kile/2a32282e-dd48-4ad9-a542-609804b02cc9

⁴ <https://datatracker.ietf.org/doc/html/rfc6113>

Смарт-карты

Аутентификация по смарткартам – the great

- › Аутентификация по смарткартам – это аутентификация по сертификату;
 - › “PKINIT is a preauthentication mechanism for Kerberos 5 which uses X.509 certificates to authenticate the KDC to clients and vice versa.”¹
- › Для хранения приватного ключа могут использоваться различные крипто-провайдеры; (смарткарты (физические или виртуальные) (winscard.dll), TPM, программное хранилище, ...)
- › “Smart card is required for interactive logon”The image shows a Windows 'Account options' dialog box. It has a title bar 'Account options:' and a list of four checkboxes. The first checkbox, 'Smart card is required for interactive logon', is checked. The other three checkboxes are unchecked: 'Account is sensitive and cannot be delegated', 'Use only Kerberos DES encryption types for this account', and 'This account supports Kerberos AES 128 bit encryption'. There are up and down arrow buttons on the right side of the list box.
- › приводит к удалению AES секретов и ротации NT секрета для учётки – **это конечная цель безопасности, факт отказа от пароля**;
- › не позволяет запросить TGT тикет по паролю, только по сертификату (через PKINIT);
- › однако, NTLM аутентификация продолжает работать – **pass-the-hash продолжает работать**;
- › Включайте авторотацию пароля при логоне² и жёсткую парольную политику (FGPP);
 - › NT хеш ротируется для учётки с устаревшим секретом в момент аутентификации (PKINIT);
 - › если администратор отключит себе “smart-card only”, то вскоре у него протухнет пароль;

¹ <https://web.mit.edu/kerberos/krb5-1.12/doc/admin/pkinit.html>, <https://datatracker.ietf.org/doc/rfc4556/>

² Enable rolling of expiring NTLM secrets during sign on, for users who are required to use Microsoft Passport or smart card for interactive sign on (msDS-ExpirePasswordsOnSmartCardOnlyAccounts)

Аутентификация по смарткартам – the ugly

- › Проблема безопасной процедуры выдачи и обновления сертификатов;
- › При аутентификации по смарткарте в windows, PIN-код хранится в lsass.exe в открытом виде;
- › Особенности RDP:
 - › пробрасываются «все сертификаты со всех смарткарт»;
(потому что в RDP сессии пользователя WinSCard.dll все запросы пробрасывает в RDP channel)
 - › PIN-код передаётся серверу и хранится в lsass.exe;
- › Запросы на крипто-операции можно перехватывать и пивотить на удалённую систему;
- › PKINIT уязвима к пред-вычислению подписей для будущего времени¹;
 - › Включайте на DC “PKInit Freshness Extension”² в принудительном режиме.
- › Проблема названия: «PIN» - люди воспринимают, как «только цифры» и могут не использовать буквы и спец.символы;

¹ @gentilkiwi “You (Dis)Liked Mimikatz? Wait For Kekeo” (BlueHat 2019)

² <https://www.rfc-editor.org/rfc/rfc8070#section-1>

Windows Hello (for Business ?)

Windows Hello != Windows Hello for Business

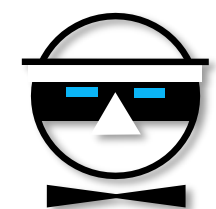
Windows Hello¹:

- › *при использовании пин-кода*: пароль пользователя будет зашифрован на пин коде и с использованием drapi-ng (из под учётной записи компьютера).
- › при наличии TPM, он также используется в шифровании;
- › *при использовании биометрии*: пароль пользователя будет храниться с использованием drapi-ng (из под учётной записи компьютера);

при повышении привилегий, пароль пользователя можно извлечь в открытом виде

Windows Hello for Business² – это аутентификация на сертификатах

- › Организация выдачи сертификатов – это конструктор³ (key trust vs certificate trust, AzureAD vs ADFS, ...)
- › 2FA – обязательный пререквизит;
- › “msds-KeyCredentialLink”⁴ (Key Trust) – атрибут, содержащий публичный ключ/сертификат, на основе которых, субъект может аутентифицироваться;
- › пользователи не могут записывать в этот атрибут (получение сертификата только через WHfB процедуру)
- › компьютерные учётные записи по умолчанию имеют право на запись в этот атрибут;



¹ <https://www.insecurity.be/blog/2020/12/24/dpapi-in-depth-with-tooling-standalone-dpapi/>

² “автобус видел? – совсем не похож”

³ <https://learn.microsoft.com/en-us/windows/security/identity-protection/hello-for-business/hello-how-it-works-provisioning>

⁴ https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/d4b9b239-dbe8-4475-b6f9-745612c64ed0

Kerberos PKINIT – “unpack-the-hash” (оговорка #?)

Используя PKINIT можно получить NT хеш учётной записи в открытом виде.

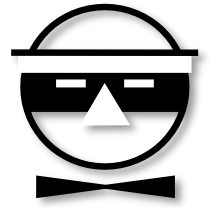
Kerberos цепочка:

1. Запросить TGT по сертификату (через PKINIT):
 - › TGT PAC будет содержать PAC_CREDENTIAL_INFO (NTLM_SUPPLEMENTAL_CREDENTIAL), содержащую NT хеш пользователя;
2. Используя Kerberos U2U запрашиваем TGS “от себя к себе”:
 - › TGS (включая NT хеш) будет зашифрован на сессионном ключе из additional TGT;
 - › Таким образом, TGS можно расшифровать и извлечь NT хеш;

Операционная система использует этот функционал, чтобы получить NT хеш для пользователя, залогинившегося по сертификату.

- › Атакующие ведут себя также ...
- › В случае компрометации Enterprise CA злоумышленник может выпустить сертификаты на имя компьютерных или пользовательских учётных записей и извлечь их NT хеши.

WHfB – shadow credentials (оговорка #?)



Механизм записи сертификата в `msds-KeyCredentialLink` назвали «Shadow Credentials».

- › Атакующие добавляют в `msds-KeyCredentialLink` свой публичный ключ, чтобы закрепиться на скомпрометированном компьютере, или повысить привилегии и развить атаку;
- › Примерами атак могут быть:
 - › повышение из непривилегированного контекста (например, `network service, ...`);
 - › закрепление по результатам успешной *-relay атаки на Idap домен контроллера;

Помимо закрепления, используя shadow credentials, можно извлечь NT хеш учётной записи компьютера (используя `unpac-the-hash`).

У атакующего есть возможности угнать учётную запись из под которой он выполняются (#2).

Смарткарты – defensive

- › До внедрения смарткарт – вы должны иметь планы по отключению парольной аутентификации;
- › Включайте частую ротацию хешей;
- › Заранее продумайте безопасный механизм выдачи сертификатов;
- › Принудительно включайте “PKInit Freshness Extension” расширение;
- › Мониторьте изменения в атрибуте “msds-KeyCredentialLink”;
- › Отключайте или ограничивайте NTLM аутентификацию по возможности;

ADCS

Безопасность ADCS

ADCS – это минное поле

(уязвимости в конфигурации СА, шаблонов сертификатов, уязвимости на DC, *-relay атаки, ...)

- › ESC1-ESC8 – “Certified Pre-Owned”¹ (2021)
- › ESC9, ESC10² (@ly4k) (2022)
- › ESC11³
- › популяризация привела к взрывному росту хактулов и прочих тулов ...

Любые изменения связанные ADCS должны тщательно анализироваться службой ИБ.

- › Если вам не нужна интеграция с AD, то есть альтернативы:
 - › <https://github.com/cloudflare/cfssl>
 - › <https://github.com/smallstep/certificates>
 - › ...

Используйте HSM, держите root СА в offline, защищайте СА, ...

¹ <https://posts.specterops.io/certified-pre-owned-d95910965cd2>

² <https://research.ifcr.dk/certipy-4-0-esc9-esc10-bloodhound-gui-new-authentication-and-request-methods-and-more-7237d88061f7>

³ <https://blog.compass-security.com/2022/11/relaying-to-ad-certificate-services-over-rpc/>

Безопасность сертификатов СА

Сертификаты не предоставляют встроенной возможности разграничивать свою применимость между приложениями.

Если в EKU сертификата указан “Client Authentication”, то:

- › можно аутентифицироваться на веб-сервисе;
 - › можно аутентифицироваться по Kerberos с использованием PKINIT;
 - › можно аутентифицироваться по vpn;
 - › ...
- › Чтобы ограничить применимость сертификата, нужно:
- › или костылить индивидуальные проверки на стороне сервиса (если такая возможность имеется) + костылить внесение дополнительных полей в (безопасную?) процедуру выдачи сертификатов;
 - › **или заводить отдельные IntermediateCA и на стороне сервиса ограничивать доверенную цепочку сертификатов;**
 - › или вы “гарцующий пони”: настраиваете ваши сервисы определять пользователя по разным полям из сертификата (например, смарткарты по полю UPN, веб-сервис по CN, ...)¹;

При внедрении смарткарт или WHfB рекомендуется:

- › завести отдельный intermediate AuthCA;
- ~~И на стороне домена в контейнере NTAUTH ограничить доверие только для этого СА;~~
- › **прямо сейчас** удалите лишние сертификаты из NTAUTH контейнера вашего домена (NTAUTH содержит сертификаты, которым доверяет DC при аутентификации по PKINIT).

¹ вы проиграете в этой игре

DC – маппинг сертификата в пользователя

- › PKINIT¹
 - › msDS-KeyCredentialLink¹ (KeyTrust) – цепочка сертификата не проверяется;
 - › szOID_NTDS_CA_SECURITY_EXT² – содержит SID учётной записи (с мая 2022);
 - › Subject Alternative Name¹ (“implicit mapping”) содержимое сертификата:
 - › DNSName будет использовано для поиска среди учётных записей с битом WORKSTATION_TRUST_ACCOUNT или SERVER_TRUST_ACCOUNT;
 - › UPN будет использовано:
 - › для поиска УЗ по атрибуту userPrincipalName, потом по sAMAccountName;
 - › для добавления `\$` и
для поиска УЗ по атрибуту userPrincipalName, потом по sAMAccountName;
 - › altSecurityIdentities² (“explicit mapping”) – 6+ разных способов выбора соответствия по serial number, CN, email, sha1, ...;
- › SChannel³ – аутентификация по TLS – в частности используется при аутентификации в LDAPS по клиентскому сертификату;
 - › До KB5014754², использовалась отдельная процедура аутентификации (несколько типов маппинга), после мая 2022 по умолчанию оставили только “strong” аутентификацию (на основе S4U2Self⁴);

¹ https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-pkca/01c4acb8-c366-4d31-93a5-fbf2d59c8b27

² KB5014754 (CVE-2022-26923, ...) <https://support.microsoft.com/en-us/topic/kb5014754-certificate-based-authentication-changes-on-windows-domain-controllers-ad2c23b0-15d8-4340-a468-4d4f3b188f16>

³ <https://offsec.almond.consulting/authenticating-with-certificates-when-pkinit-is-not-supported.html>

⁴ https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu/cd9d5ca7-ce20-4693-872b-2f5dd41cbff6

ADCS – defensive

- › Используйте отдельные intermediate CA;
- › Прямо сейчас удалите лишние сертификаты в NTAUTH контейнере;
- › Следите за безопасностью ваших CA (уязвимости ESC1, ...) (hsm, offline root CA, ...);
- › Следуйте рекомендациям Microsoft и применяйте изменения в механизмах аутентификации;
- › Отключайте SChannel маппинг, оставляя только S4U2Self-mapping;
- › Маппинг через поле `szOID_NTDS_CA_SECURITY_EXT` предпочтительнее других видов маппинга (explicit или implicit);

pass-the-hash

pass-the-hash

Ответ microsoft¹: “*Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft, Version 1 and 2*” (2012)

вкратце:

- › глава “Why can’t Microsoft release an update to address this issue?”;
- › рекомендации по минимизации утечки секретов (NT/AES хешей);

С тех пор Microsoft вложил в множество технологий:

- › ...
- › VBS (Virtualization Based Security);
- › ...

¹ <https://www.microsoft.com/en-us/download/details.aspx?id=36036>

VBS (Virtualization Based Security)

Попробуем разобраться в терминах:

VBS (“core isolation”):

- › **Windows Hypervisor**

- › **VTL0** - традиционное windows окружение (ОС, которую мы привыкли видеть, `ntoskrnl.exe`)
- › **VTL1** - изолированное окружение (Virtual Secure Mode (VSM)¹) (“secure kernel” (`securekernel.exe`) + IUM (Isolated User Mode))

Trustlets (aka “secure process”, “trusted process”, “IUM process”):

- › проверки целостности исполняемых файлов (HVCI, KMCI, UMCI, ...);
- › проверка целостности загрузки (CCI (configurable code integrity));
- › credential guard (содержит в себе учётные данные и предоставляет интерфейс взаимодействия)²;
- › ...

Маркетинговые названия для разного набора “фич”: “Device Guard”, “secured-core PC”, ...

- › Дополнительные механизмы изоляции через виртуализацию:
 - › Application Guard (виртуализация для Edge, MS Office)
 - › Windows Sandbox

¹ <https://learn.microsoft.com/en-us/windows/win32/procthread/isolated-user-mode--ium--processes>

² в одной из видео-лекций от Microsoft упоминалось, что credential guard даже участвует в защите vTPM (в случае его использования).

Обеспечение безопасности lsass.exe

*Не существует серебряной пули, способной **гарантированно** защитить секреты на скомпрометированной ОС, в наших силах сделать жизнь атакующего много-много труднее:*

- › Отключение Digest SSP провайдера;
- › Credential Guard
 - › компрометация через NetNTLMv1¹ (v1 всё ещё стреляет!) + свободное использование для netNTLMv2 (“pass-the-challenge”);
 - › патчинг Credential Guard для включения wdigest²;
- › RunAsPPL³
 - › Вредоносный драйвер⁴, или легитимно-уязвимые вредоносные драйверы;
 - › Dll hijacking через \KnownDlls⁵;

Прочие утечки пароля:

- › При логоне пароль пользователя проходит через множество подсистем, где его можно перехватить:
 - › SSP provider⁶;
 - › Network provider⁷;
- › Извлечь пароль из других мест, например:
 - › TermService или mstsc.exe⁸;
 - › сохранённые в браузере пароли, credentials manager, секреты из конфигурационных файлов, ...

¹ <https://research.ifcr.dk/pass-the-challenge-defeating-windows-defender-credential-guard-31a892eee22> (2022)

² <https://itm4n.github.io/credential-guard-bypass/>

³ <https://itm4n.github.io/lsass-runasppl/>

⁴ <https://posts.specterops.io/mimidrv-in-depth-4d273d19e148>

⁵ <https://blog.scrt.ch/2021/04/22/bypassing-lsa-protection-in-userland/>

⁶ <https://github.com/gentilkiwi/mimikatz/tree/master/mimidrv>

⁷ <https://www.scip.ch/en/?labs.20220217>

⁸ https://github.com/gentilkiwi/mimikatz/blob/master/mimikatz/modules/kuhl_m_ts.c

SSO в масштабах компании

Время SSO 20-летней давности

Задача UX:

- › Нужна поддержка double-hop. Залогинившись на ОС, нужно, чтобы с неё, прозрачно для пользователя, работал доступ к другим ресурсам (smb, http, ...);

Однако в те годы:

- › ещё не существует passwordless, не умеют в смарткарты;
- › нет “идеи” lateral movement, да и в целом безопасность не очень развита;
- › пробрасывать сокет как сегодня у ssh не догадались;

Поэтому:

- › Будем передавать пароль и хранить его в памяти для прозрачной аутентификации;
 - › позже: хранить хеши (или пин-код) в памяти
 - › позже: пытаться защищать хеши и тикеты через Credential Guard, пин-код всё также в памяти

Сегодня, концепция почти не изменилась:

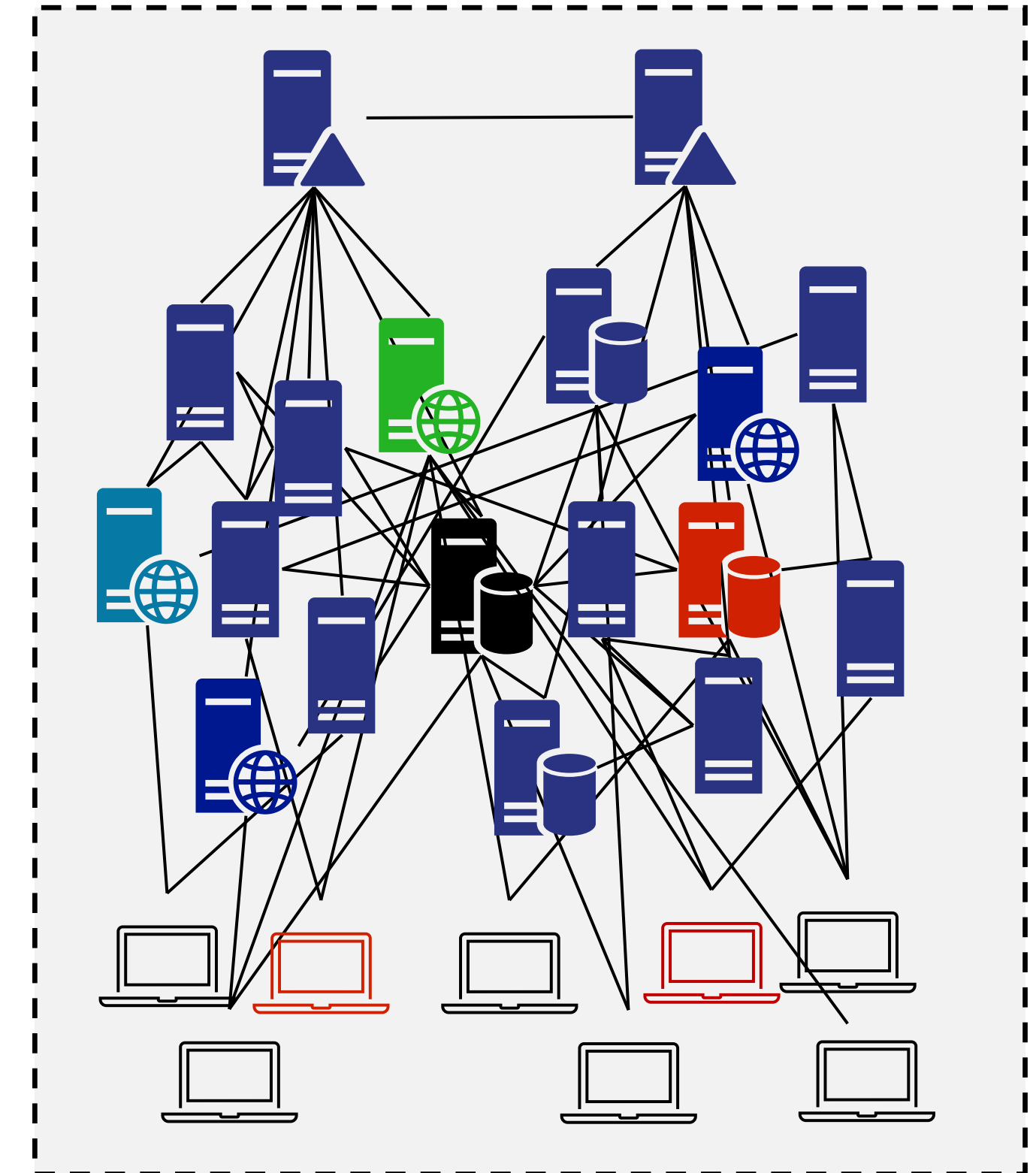
- › RDP – передаёт пароль (а при аутентификации по смарткарте – передаёт пин-код);
- › WinRM – поддерживает “CredSSP”², которая передаёт пароль (и до сих пор есть продукты это использующие);
- › HTTP-basic аутентификация всё ещё встречается;
- › ...

¹ RDP в режиме remote Credential Guard умеет “пробрасывать аутентификацию на хост” (но работает не всегда стабильно)

² <https://learn.microsoft.com/en-us/windows/win32/secauthn/credential-security-support-provider>

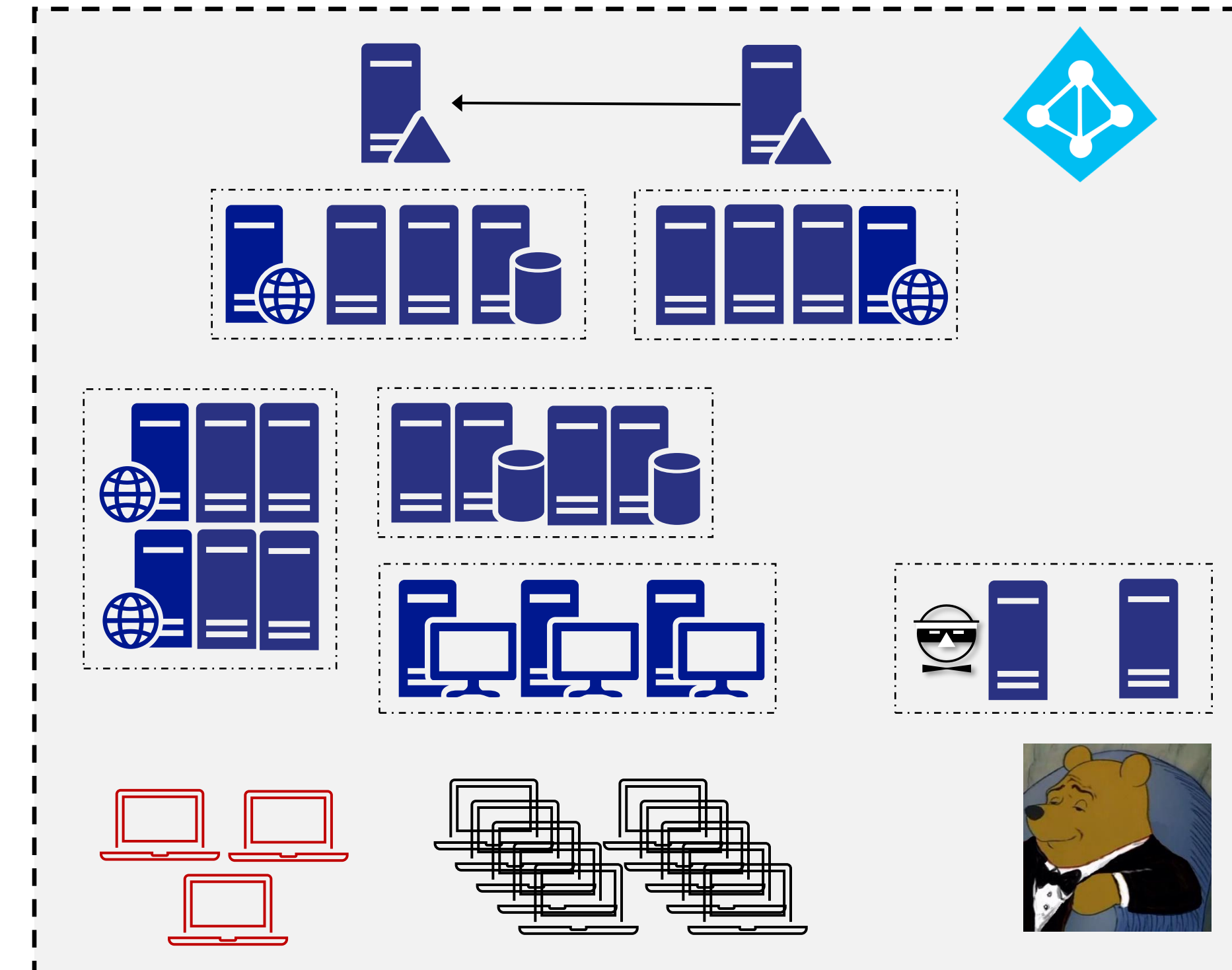
Проблема проброса учётных данных

- › Как выглядит “типичный” enterprise:
 - › Сколько учётных записей у администраторов?
 - | **ЦЕЛЫХ ДВЕ:** от всех серверов и домен админская
- › Где логинятся администраторы?
 - | **езде**
- › Где можно найти кешированные хеши (NT/AES) или тикеты?
 - | **езде**
- › Сколько серверов нужно взломать, чтобы скомпрометировать примерно всё?
(клиентские/приватные данные, бизнес приложения, бухгалтерия, терминальные сервера, бекапы, exchange, sccm/wsus, ADCS, ...)
 - | **ОДИН**



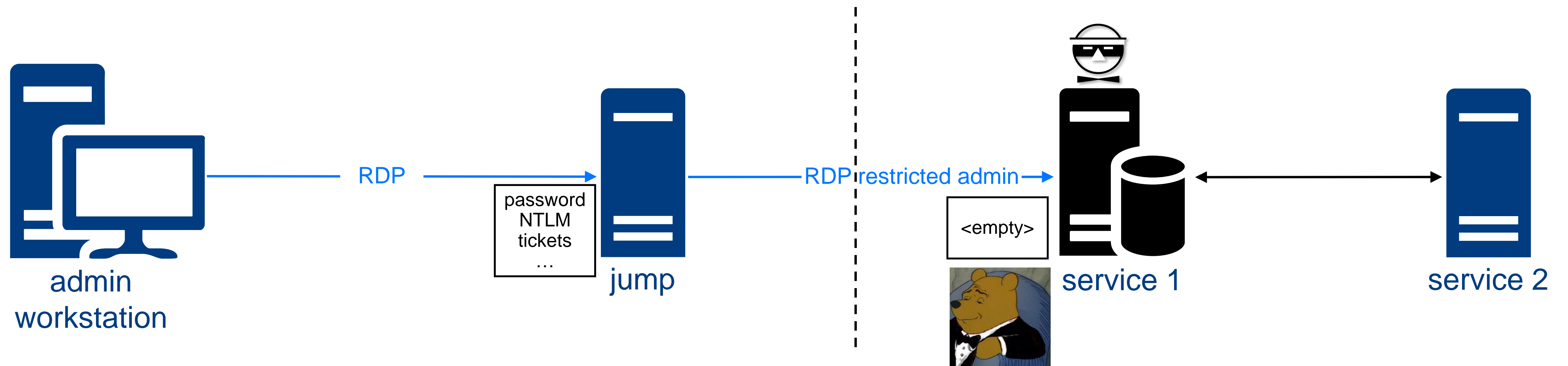
“Разграниченный” enterprise

- › сервисы разделены на **непересекающиеся** группы - *тиры*;
- › привилегии и доступы:
 - › “principle of least privilege”;
 - › заданы декларативно и мониторятся;
- › Сколько учётных записей у администраторов?
 - › **“несколько” – для разных тиров**
- › Где логинятся администраторы?
 - › **езде, но учётки не совпадают**
- › Где можно найти кешированные хеши (NT/AES) или тикеты?
 - › **езде, но учётки не совпадают**
- › Что даёт злоумышленнику взлом одного сервера?
 - › компрометацию **ОГРАНИЧЕННОЙ** группы серверов (одного тира)
- › слабые места тирной модели:
 - › **может быть не так просто разбить всё на тиры из-за различных межсервисных связей;**
 - › **контролируйте, чтобы пароли не переиспользовались;**



“Сломать SSO”

- › В качестве альтернативы тирной схеме можно “не пробрасывать” учётные данные администратора¹;
 - › “restrictedadmin” режим подключения по RDP²;

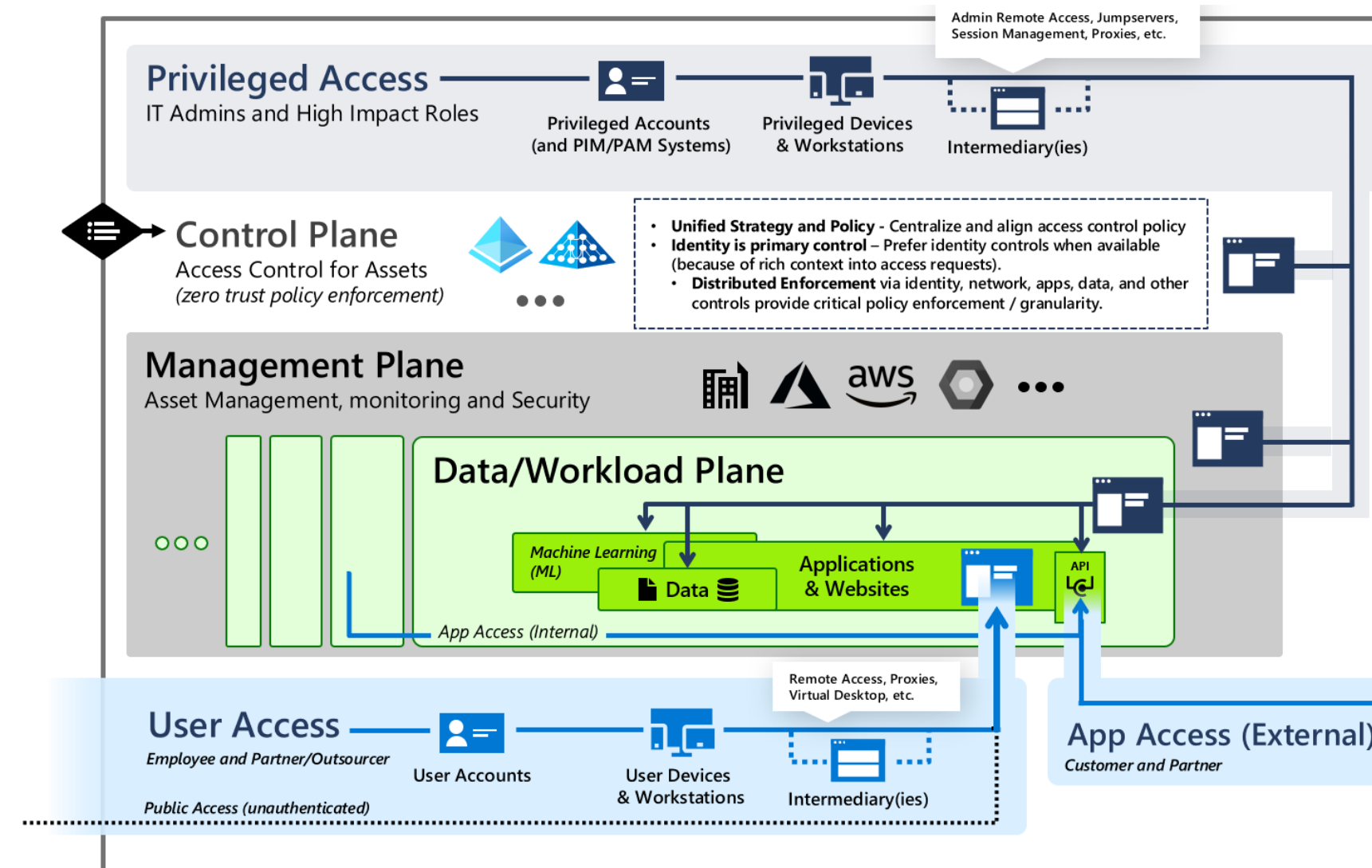
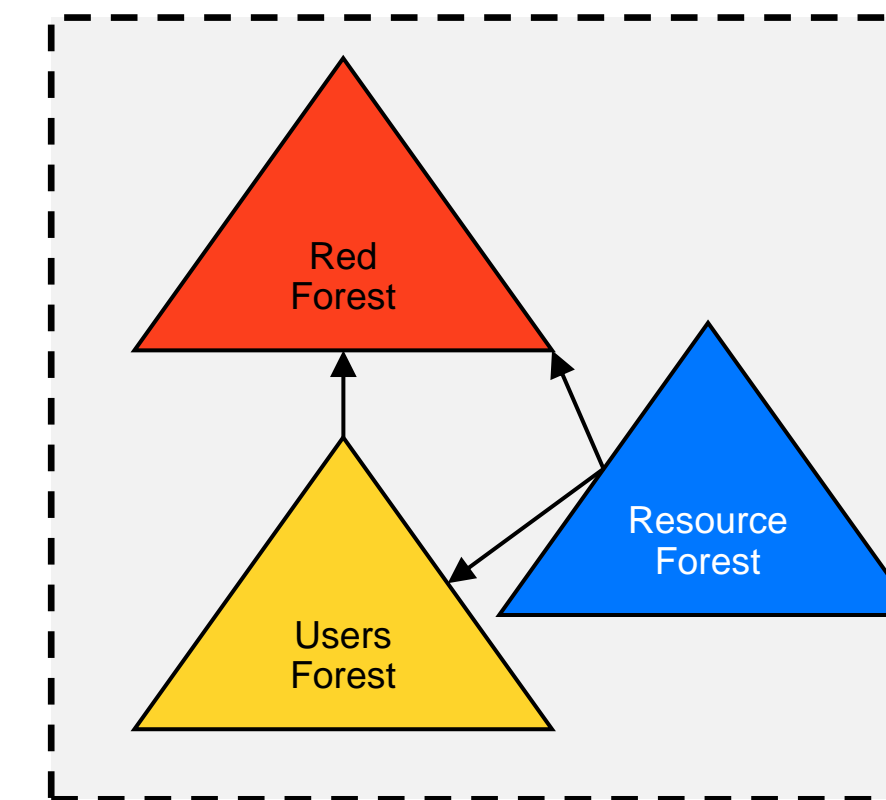
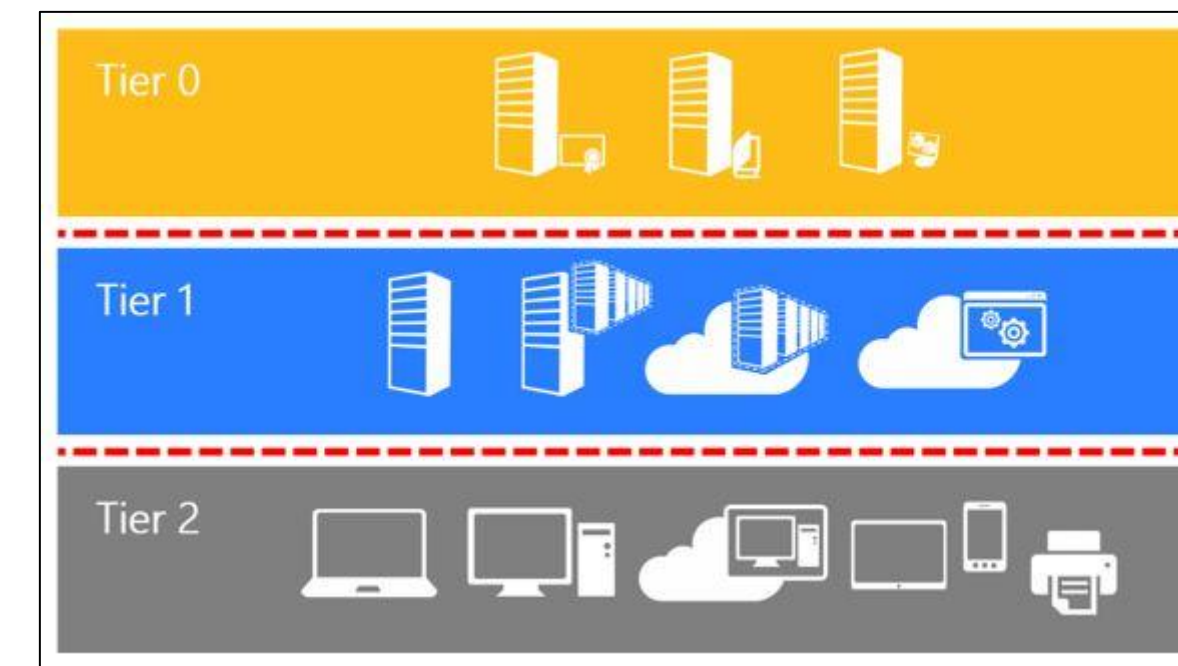


¹ <https://labs.withsecure.com/publications/undisable>

² <https://learn.microsoft.com/en-us/windows/security/identity-protection/remote-credential-guard>

Best practices

- › “legacy tier model”¹;
- › ESAE (“Red Forest”)² + resource forest;
 - › но, защита строится только вокруг администраторов;
- › Enterprise access model¹, privileged access strategy, ...;
 - › “shift-to-cloud”;
- › Хорошая модель безопасности также включает в себя:
 - › понимание поверхности атаки и путей её развития;
 - › наличие описанной ролевой модели,
 - › учитывающей и пользователей и роботов;
 - › управление привилегированным доступом;
 - › just-in-time, double-approve, temporary, ...;
 - › привилегированные рабочие станции (PAW);
 - › MFA;
 - › покрытие мониторингами;
 - › наличие бекапов (на случай инцидента);
 - › ...



Privileged Access

Enables IT administrators and other high impact roles to access to sensitive systems and data.
Stronger security for higher impact accounts

Control and Management Planes

Provide unified access and management for workloads and assets (*and provide attackers shortcut for illicit objectives*)

Data/Workloads

Create and store business value in

- Business processes (in apps/workloads)
- Intellectual property (in data and apps)

User and App Access

How employees, partners, and customers access these resources

¹ <https://learn.microsoft.com/en-us/security/compass/privileged-access-access-model>

² <https://learn.microsoft.com/en-us/security/compass/esae-retirement>

Образ мышления: ничему не доверяй

- › Стройте безопасность исходя из того, что:
 - | **Любой доступный сервис может быть взломан;**
 - › (а недоступный сервис – стать доступным)
 - | **Атакующий всегда может повысить привилегии;**
 - | **Скомпрометировав ОС, атакующий скомпрометировал на ней всё;**
 - › сервисы;
 - › пользователей;
 - › секреты;
 - › админов;

Итоги

Защищайте свою Active Directory, особенно аутентификацию

- › это первое, что атакующий попытается взломать после проникновения во внутреннюю сеть;
- › будьте в тренде - механизмы аутентификации до сих пор меняются (вследствие уязвимостей);
- › некоторые проблемы аутентификации ложатся на вас, вручную внедряйте контрмеры, архитектура AD не слишком *“secure by default”*;
- › проблемы, которые не решаются превентивно – закрывайте мониторингами; (*“prevention is ideal, detection is a must”*)



Спасибо за внимание

дополнительные материалы

История, документация

Kerberos – это протокол аутентификации по сети

- › Kerberos v5 – RFC 4120 (<https://datatracker.ietf.org/doc/html/rfc4120>) (2005) (1993)
Реализация от MIT: <https://web.mit.edu/kerberos/>
- › @microsoft Technical Documents:
 - [MS-KILE]: Kerberos Protocol Extensions
 - [MS-PAC]: Privilege Attribute Certificate Data Structure
 - [MS-SFU]: Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol
 - [MS-KKDCP]: Kerberos Key Distribution Center (KDC) Proxy Protocol
 - [MS-PKCA]: Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol
 - ...

Kerberos GSS-API

- › GSS-API (Generic Security Service Application Program Interface) Version 2, Update 1 – [RFC2743](#) (2000)
– предоставляет общий интерфейс для сервисов безопасности (механизм аутентификации, обеспечения целостности, конфиденциальности)
- › SPNEGO (Simple and Protected GSS-API Negotiation Mechanism) – [RFC4178](#) (2005)
– механизм позволяющий субъектам определить общий набор поддерживаемых GSS-API механизмов
- › The Kerberos Network Authentication Service (V5) – [RFC4120](#) (2005)
– протокол Kerberos
- › The Kerberos Version 5 GSS-API Mechanism: Version 2 – [RFC4121](#) (2005)
– GSS-API интерфейс протокола Kerberos
- › SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows – [RFC4559](#) (2006)
– как браузер MSIE использует GSS-API интерфейс для аутентификации (`WWW-Authenticate: Negotiate`)
 - › нарушает условие stateless протокола HTTP/1.1 ([RFC7230](#))
- › [\[MS-NTHT\]: NTLM Over HTTP Protocol](#) (2007)
– (`WWW-Authenticate: NTLM ...`)

Technical Documents – публичная и обновляемая документация @microsoft для используемых протоколов