Kerberos для специалиста по тестированию на проникновение. Часть 6. PKINIT

ardent101.github.io/posts/shadow_creds

Ardent101 December 12, 2022

В статье рассматривается устройство одного из расширений протокола Kerberos - PKINIT. После теоретической вводной будут разобраны техники проведения атак, связанные с указанным расширением, в частности: Shadow Credentials и UnPAC the hash.

Если вкратце, то первая техника позволяет атакующему закрепиться в домене, не смотря на смену паролей, а вторая получить NT-хеш пароля учетной записи при наличии права изменения атрибута msDS-KeyCredentialLink указанной учетной записи.

Вводная теория

PKINIT (Public Key Cryptography for Initial Authentication in Kerberos) - расширение протокола Kerberos, позволяющее использовать криптографию с открытом ключом на этапе предварительной аутентификации.

Зачем понадобилось это расширение? Какими преимуществами оно обладает?

> По-хорошему для грамотного ответа необходимо обладать специализированными знаниями в криптографии, а размышлять “на пальцах” о подобных вопросах неблагодарное занятие. Те кто ничего не знал вряд ли поймут, те кто знал сочтут за невежду. Тем не менее этот материал не является строгой научной статьей, поэтому попробую предоставить небольшую вводную для общего развития и лучшего понимания о чем идет речь.

При оценке защищенности информационных ресурсов организации одним из наиболее типовых и классических недостатков является использование словарных или нестойких к атакам методом подбора паролей. Сам по себе протокол Kerberos никак не позволяет защититься от подобных атак (см. ранее про распыление). Безусловно в Active Directory есть возможность реализовать строгую парольную политику, кроме того можно осуществлять мониторинг журнала событий с целью выявления и пресечения указанных атак. В ряде случаев это работает, но как решить проблему “на корню”?

Начнем очень издалека. Пароль - что это вообще такое? В ГОСТ Р 58833-2020 “*Защита информации. Идентификация и аутентификация. Общие положения*” есть следующее определение:

Пароль - конфиденциальная аутентификационная информация, обычно состоящая из строки знаков.

Ключевое слово - “обычно”. Действительно в качестве пароля также могут выступать: аппаратные токены, отпечатки пальцев, сетчатка глаза, голос, да мало ли что еще. Более того пароль может включать в себя сразу несколько факторов, например смарт карта с пин кодом.

> Оффтоп: следует различать двухфакторною и двухэтапную аутентификацию, чуть более подробно простым языком об этом можно прочитать здесь.

Пароли, “состоящие из строки знаков”, обладают рядом недостатков, а именно:

- пароли могут быть словарными
- пароли надо помнить
- пользователи могут непреднамеренно раскрыть свои пароли в ходе фишинговых атак, в почтовой переписке или в файле на общедоступном сетевом хранилище
- при захвате сервера атакующий зачастую может извлечь пароль пользователя и использовать его для атаки методом повторного воспроизведения

Расширение PKINIT помогает минимизировать приведенные выше проблемы в рамках Kerberos. PKINIT реализует поддержку протоколов с открытым ключом и в частности позволяет внедрить аутентификацию по сертификатам или двухфакторную аутентификацию. Таким образом PKINIT может гарантировать использование несловарных, стойких к атакам метода перебора паролей.

Далее попробуем разобраться, что такое “открытый ключ”.

Криптография с открытым ключом

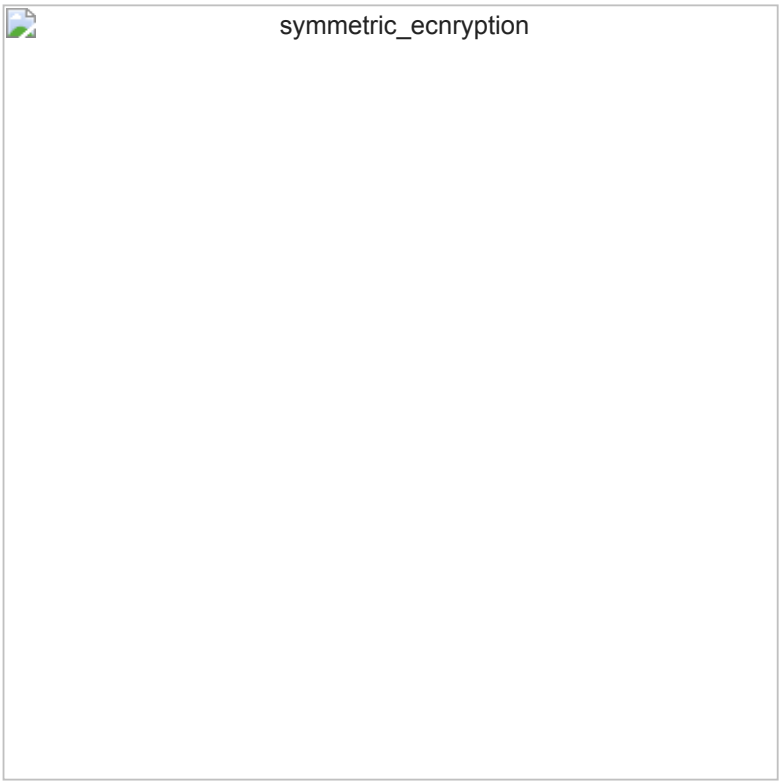
Для начала рассмотрим, в чем разница между симметричным и ассиметричныи шифрованием.

При симметричном шифровании для шифрования и расшифрования данных используется один и тот же ключ, то есть от клиента и сервера требуется наличие общего одинакового секрета. В частности, именно этот вид шифрования по умолчанию используется в Kerberos в ходе предварительной аутентификации.

При асимметричном шифровании используется пара математически связанных между собой ключей: открытый и закрытый. Закрытый хранится в тайне, в то время, как открытый является публичным. То, что было зашифровано на одном ключе, расшифровывается с использованием второго.

Для лучшего понимания рассмотрим следующую бытовую аналогию:

> При симметричном шифровании сообщение передается в ящике, закрытом на обычный замок. Ключ от замка хранится как у отправителя, так и у получателя. Для шифрования отправителю требуется положить сообщение в ящик и запереть его на замок с использованием ключа. Получатель открывает замок с помощью того же самого ключа и извлекает сообщение.



Общий принцип симметричного шифрования

При асимметричном шифровании сообщение передается в ящике, закрытом на *необычный замок*. Особенность замка заключается в том, что для его закрытия ничего не требуется, замок достаточно только защелкнуть, а вот для открытия уже необходим ключ. Получатель создает много копий замка и раздает их всем желающим, а ключ хранит у себя в секрете. Каждый, кто хочет отправить сообщение, кладет его в ящик и закрывает его на защелкивающийся замок, открыть который может только получатель.

Схема передачи сессионного ключа с использованием ассиметричного шифрования

Симметричное шифрование работает быстро, но его узким местом является управление ключами, в частности возникают трудности при передаче общего секретного ключа по открытым каналам связи. Это важно при установлении защищенного соединения, например в сети Интернет.

Криптография с открытым (ассиметричным) ключом позволяет эффективно решить указанную проблему, но при этом работает медленнее. В итоге на практике используется сочетание двух видов шифрования.

Для лучшего понимания рассмотрим искусственный, но отчасти приближенный к реальности пример. Представим себе сеть, где для каждого пользователя имеется свой сервер, на котором развернута почта и личный блог с контактной информацией.

Рассмотрим возможный процесс обмена сообщениями в указанной сети:

Общий принцип асимметричного шифрования

1. Пользователь Алиса находит сайт пользователя Боб и в разделе с контактной информацией узнает открытый ключ Боба
2. Алиса генерирует секретный сессионный ключ, зашифровывает его с использованием открытого ключа Боба и отправляет получившиеся зашифрованное сообщение Бобу
3. Боб, получив сообщение, расшифровывает его с помощью своего закрытого ключа и получает сессионный ключ
4. Боб отправляет Алисе сообщение, зашифрованное с использованием полученного сессионного ключа. Дальнейшее общение шифруется на установленном сессионном ключе.

Важно отметить, что указанный пример подвержен атаке “человек по середине”. Атакующий может навязать отправителю свой сервер, вместо сервера Боба, например с помощью ARP-spoofing. Подложный сервер может содержать копию сайта Боба, но уже с другим значением открытого ключа, секретный ключ к которому имеется у атакующего. В результате отправитель направит атакующему сообщение зашифрованное на подставном открытом ключе.

Возникает вопрос, как Алиса может убедиться в том, что открытый ключ Боба действительно принадлежит Бобу? То есть требуется аутентификация открытых ключей. Таким образом мы подошли к следующей теме.

Электронная цифровая подпись

Криптография с открытым ключом работает и в другую сторону. То, что было зашифровано с использованием закрытого ключа, может быть расшифровано с помощью соответствующего открытого ключа.

Какой в этом смысл? Допустим, что мы обладаем подлинным открытым ключом некого Боба и этот самый Боб хочет выступить с публичным сообщением от своего имени. Тогда:

1. Боб пишет текст сообщения и зашифровывает его с использованием своего закрытого ключа
2. Боб публикует у себя на сайте зашифрованное сообщение
3. Любой желающий может расшифровать сообщение и прочитать его содержимое

Тот факт, что сообщение было успешно расшифровано, свидетельствует о том, что оно было написано владельцем закрытого ключа, то есть Бобом. Тем самым способом возможно “подписывать” информацию.

Иными словами цифровая подпись позволяет получателю сообщения убедиться в аутентичности источника информации, а также проверить, была ли информация изменена (искажена), пока находилась в пути. Таким образом, цифровая подпись является средством аутентификации и контроля целостности данных.

На практике шифровать само сообщение ресурсозатратно, поэтому зашифровывается хеш сообщения, полученный с использованием специальной криптографической хеш-функции. Всюду далее к исходному сообщению прикладывается указанный шифрованный хеш, который и называют подписью. Любой желающий убедиться в подлинности должен самостоятельно вычислить хеш от сообщения и сравнить его с результатом расшифрования подписи.

Центр сертификации

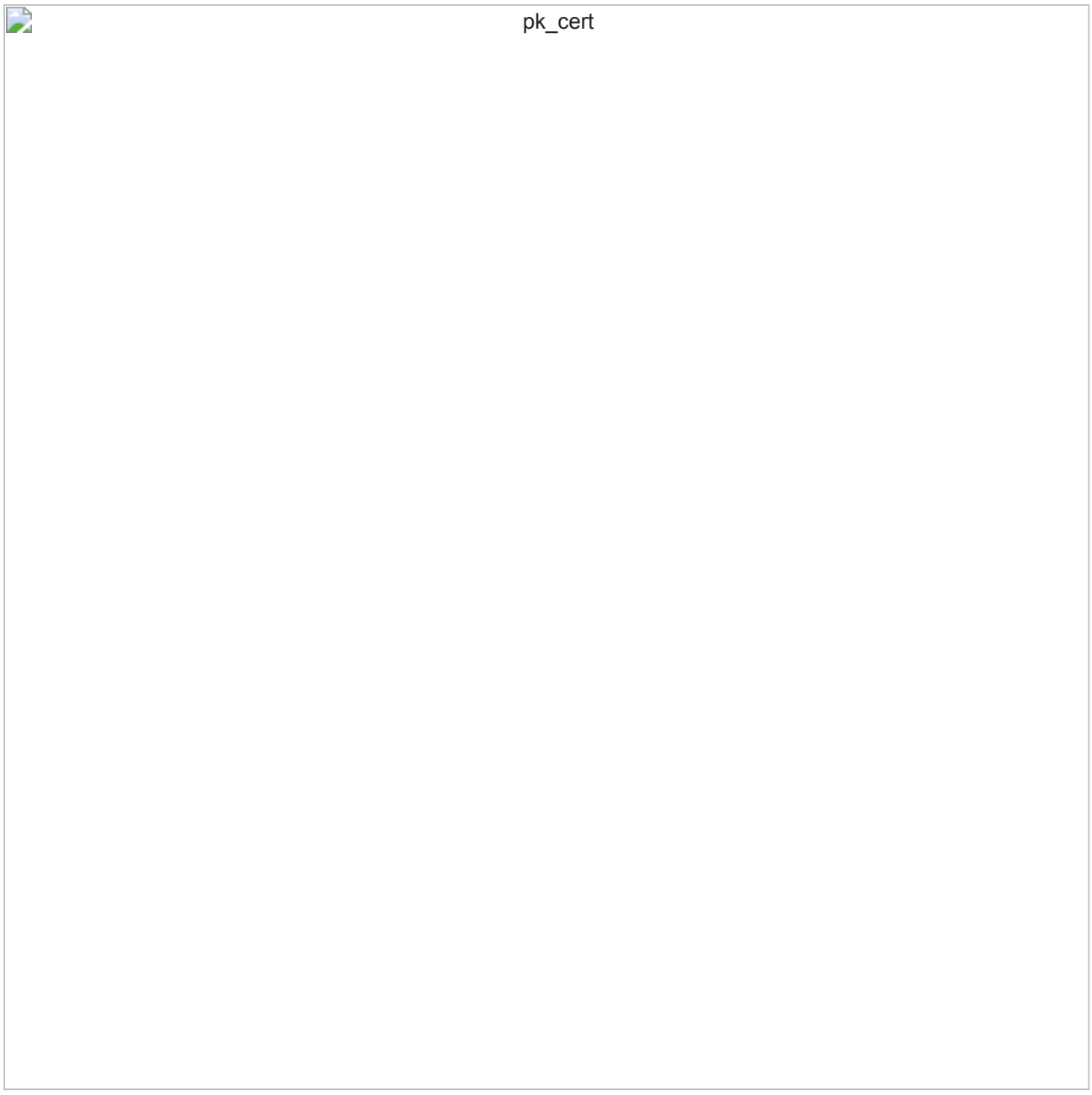
Тем не менее проблема аутентификации открытых ключей пока остается нерешенной. Одним из решений является введение сертификации открытых ключей с помощью доверенной третьей стороны - центра сертификации. В этом случае перед публикацией своего открытого ключа пользователь должен обратиться в центр сертификации чтобы получить сертификат.

Сертификат открытого ключа может содержать множество полей, например:

- Имя пользователя
- Открытый ключ пользователя
- Срок действия сертификата
- Наименование центра сертификации

Но самым главным полем является *цифровая подпись*, представляющая собой хеш от всех полей сертификата, зашифрованный на закрытом ключе центра сертификации.

Рассмотрим пример процедуры обмена сообщениями при наличии центра сертификации:



Пример обмена сообщений с использованием сертифицированного открытого ключа

1. Алиса генерирует пару открытый / закрытый ключ.
2. Алиса аутентифицируется в центре сертификации и передает туда свои данные, в частности открытый ключ.
3. Центр сертификации проверяет полученные данные и подписывает их с использованием своего закрытого ключа. В результате получается сертификат открытого ключа, который передается Алисе.

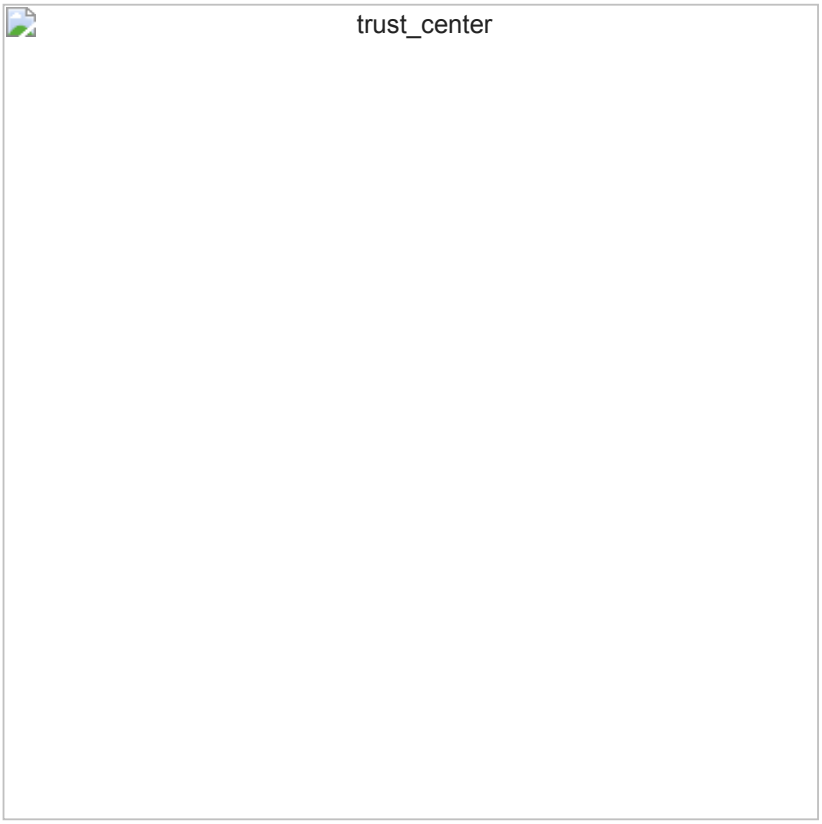
> Шаги 1-3 считаются стандартными для всех пользователей сети. Можно считать, что при регистрации каждый пользователь получает сертификат своего открытого ключа и открытый ключ Центра сертификации. Таким образом у Боба также есть сертификат для своего открытого ключа и открытый ключ Центра сертификации.

4. Желая отправить сообщение, Боб с сайта Алисы извлекает открытый ключ и соответствующий ему сертификат. С помощью открытого ключа Центра сертификации Боб проверяет подлинность полученного сертификата. Боб доверяет Центру сертификации, а успешная проверка цифровой подписи сертификата означает, что Центр сертификации в свою очередь проверил принадлежность открытого ключа Алисе. Поэтому Боб может быть уверен, что извлеченный с сайта открытый ключ действительно принадлежит Алисе.
5. Боб шифрует сообщение с использованием открытого ключа Алисы и подписывает его с использованием своего закрытого ключа. Получившиеся данные передаются Алисе.
6. Получив сообщение, Алиса расшифровывает его с помощью своего закрытого ключа. Кроме того, благодаря электронной подписи Алиса может убедиться, что сообщение было отправлено именно Бобом. Таким образом обеспечивается взаимная аутентификация пользователей.

> Любопытно отметить, что в рассмотренной схеме Центр сертификации необходим только при регистрации пользователя. Все остальное время Центр сертификации может быть вообще отключен.

Приведенный пример иллюстрирует общую идею, что за счет введения дополнительной доверенной сущности возможно решить проблему управления открытыми ключами. На практике разумеется все устроено несколько сложнее.

Важно заметить, что с внедрением процедуры сертификации в домене появляется еще один доверительный центр.



Центры доверия с сертификацией и без

PKINIT в Kerberos Active Directory

Расширение PKINIT позволяет использовать криптографию с открытым ключом на этапе предварительной аутентификации Kerberos. В первой части уже был подробно рассмотрен этап предварительной аутентификации. Вкратце напомню, что в “традиционном” случае от клиента и сервера требовалось знание общего симметричного ключа.

Теперь рассмотрим в первом приближении обмен сообщениями с использованием PKINIT:

AS_REQ запрос с PKINIT

0. Клиент отправляет серверу аутентификации текущую метку времени, подписанную своим закрытым ключом, а также сертификат предназначенный для проверки подлинности передаваемого открытого ключа.
1. Сервер аутентификации проверяет при помощи открытого ключа центра сертификации проверяет подлинность полученного сертификата открытого ключа клиента.
2. С использованием проверенного открытого ключа клиента сервер аутентификации проверяет подпись метки времени и сравнивает её значение с текущим.

AS_REP запрос с PKINIT

0. Сервер аутентификации отправляет ответ Клиенту.
1. Клиент с использованием своего закрытого ключа расшифровывает одну из частей принятого ответа.
2. В расшифрованной части клиент проверяет сертификат открытого ключа сервера аутентификации.
3. При помощи полученного открытого ключа сервера аутентификации клиент проверяет подпись AS-REP ключа.
4. С использованием AS-REP ключа Клиент расшифровывает сессионный ключ для общения с контроллером домена.
5. Клиент получает TGT и сессионный ключ для KDC.

После передачи TGT и сессионного ключа остальные этапы Kerberos аутентификации производятся точно также, как в “традиционном” случае.

Дотошному читателю может быть не понятно, зачем использовать промежуточный симметричный ключ (см. шаг №3). Почему вместо указанного ключа сразу не передавать сессионный ключ? Насколько понимаю, дело в том, что симметричный ключ может вырабатываться разными методами, имеющие свои преимущества и недостатки. Так глубоко копать нет смысла, поэтому для примера мы рассмотрели один, наиболее распространенный метод. Кроме того необходимо понимать, что рассмотренный пример является упрощением и в реальности передаваемые сообщения содержат больше полей, в частности содержащие случайные метки и идентифицирующую информацию.

> Более подробно ознакомиться с устройством PKINIT можно в RFC 4556

Физическое хранение закрытых ключей

Остался незатронутым вопрос хранения пользователями своих закрытых ключей. Для этого могут использоваться различные специализированные внешние устройства (смарт-карты, токены и др.) или внутренние аппаратные модули (TPM). Подобные устройства обладают встроенными устойчивыми к взлому хранилищами данных, а также микросхемами позволяющими выполнять необходимые криптографические операции.

Закрытый ключ может быть импортирован на устройство или сгенерирован самим устройством самостоятельно. Для активации устройства с целью выполнения операций требующих знания закрытого ключа, как правило необходимо пройти дополнительную проверку, например ввести пин-код.

Таким образом обеспечивается двухфакторная аутентификация. Первый фактор - (чем обладаю?) физическое устройство, второй фактор (что знаю?) - пин-код.

Двухфакторная аутентификация в Kerberos обладает множеством преимуществ:

- Гарантируется использование псевдослучайных длинных паролей.
- Исключается возможность хранения пароля в незащищенном месте, например в записке на рабочем столе или в файле на общедоступной сетевой папке.
- Пароль нельзя случайно разболтать или отправить третьему лицу, таким образом минимизируются фишинговые атаки.

Тем не менее существует ряд мифов, которые стоит развенчать.

Миф 1: Использование внешнего устройства при двухфакторной аутентификации позволяет безопасно осуществлять вход в домен с незащищенных рабочих станций, так как вся ключевая информация хранится на изолированном носителе.

> Это утверждение некорректно. Как минимум на указанном рабочем месте будет присутствовать активный в течении нескольких часов TGT, скорее всего позволяющий продлить срок действия до 7 суток.

Миф 2: При использование двухфакторной аутентификации по Kerberos исключается использование протокола NTLM.

> Это неправда. Двухфакторная аутентификация в Kerberos может осуществляться с применением протокола NTLM. Подробнее почему так и как это можно эксплуатировать рассмотрим дальше.

NTLM в PKINIT

Представим, что организация внедрила в Active Directory аутентификацию с использованием смарт-карт по протоколу Kerberos, но как быть с ресурсами, которые поддерживают вход только по NTLM?

Для того, чтобы лучше разобраться в этом вопросе рассмотрим процесс интерактивного входа пользователя на рабочую станцию по смарт-карте с использованием расширения PKINIT.

Пользователь вставляет смарт-карту в устройство для чтения и вводит пароль. Далее PA_PK_AS_REQ-сообщение отправляется также, как было рассмотрено ранее. Отличия начинаются на втором шаге. Сервер аутентификации возвращает TGT, содержащий в PAC NT-хеш пароля клиента, зашифрованный на AS-REP ключе. После получения PA_PK_AS_REP-сообщения клиент не может получить доступ к своему NT-хешу, так как не знает секрет KDC, с использованием которого зашифрован TGT.

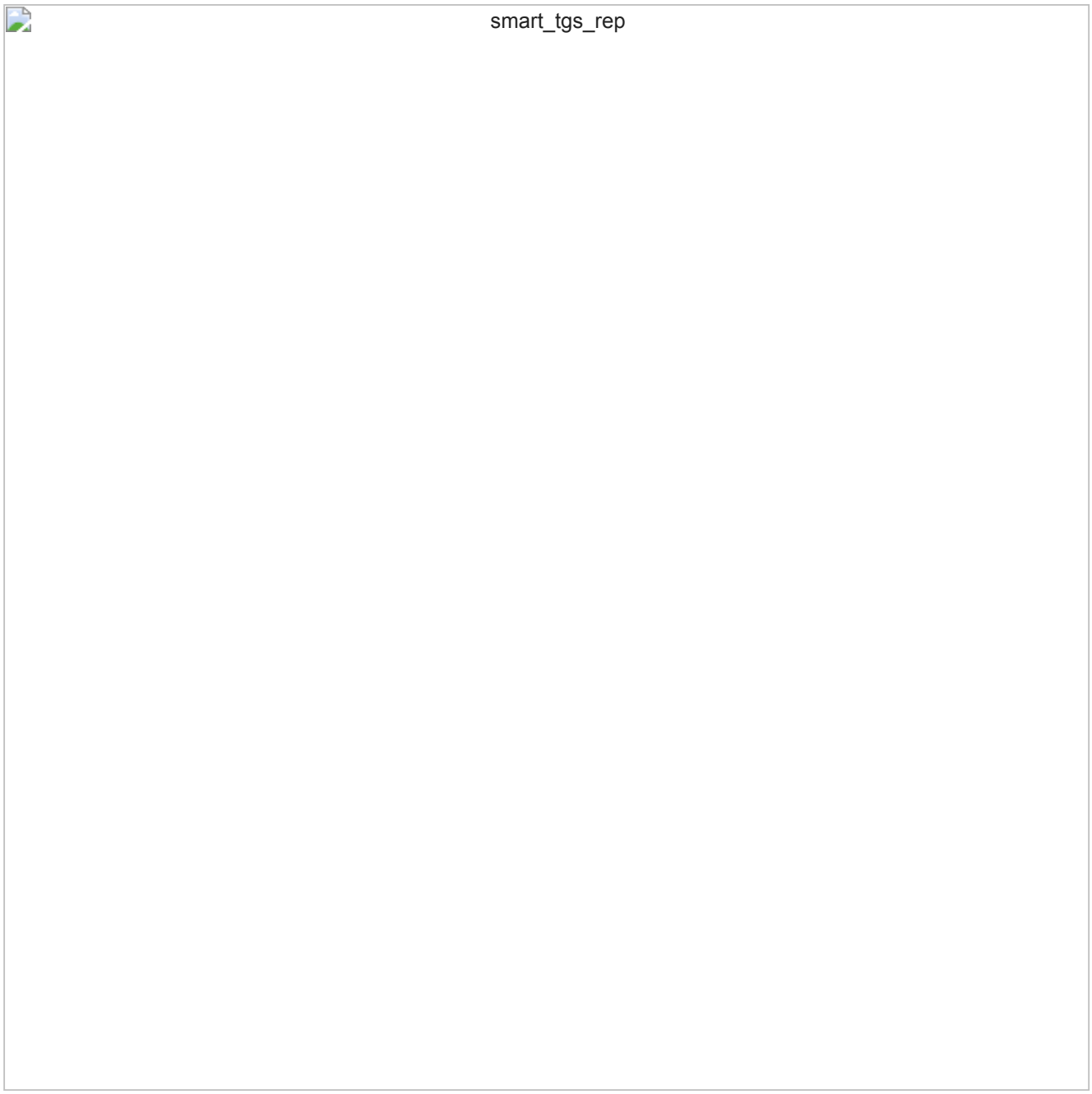
PA_PK_AS_REP при использовании смарт-карты

Далее клиент отправляет контроллеру домена запрос на получение TGS билета для доступа к рабочей станции. Обратите внимание, что в качестве принципала сервиса указывается *host/hostname.domain*.

TGS_REQ при использовании смарт-карты

Контроллер домена копирует PAC, извлеченный из TGT, в TGS билет и отправляет получившийся TGS билет клиенту. Теперь вспоминаем, что речь идет об интерактивном входе в систему, то есть “секрет хоста” считается известным значением в рамках рабочей станции.

Система предоставит доступ для клиента, но гораздо важнее, что система расшифрует NT-хеш клиента с помощью AS-REP ключа и поместит указанный NT-хеш в кэш LSA. В дальнейшем в случае если какой-либо сервис не сможет работать по протоколу Kerberos, система использует для аутентификации хранящийся в LSA NT-хеш.



TGS_REP при использовании смарт-карты

Таким образом внедрение смарт-карт не исключает применения протокола NTLM.

Дополнительно почитать об аутентификации по смарт-картам можно в следующих источниках:

- MS-PKCA - Раздел 4
- Leave the Door 0pen – Abusing 5martcard Authentication with Anderson PAC - Eran Nachshon.
- “Weaknesses and Best Practices of Public Key Kerberos with Smart Cards” Brad Hill.

Key Trust

Рассмотренный ранее пример аутентификации с помощью центров сертификации и смарт-карт представляет модель доверия на основе сертификатов. Исторически это одна из первых моделей, реализованных в Active Directory, но со временем были добавлены и

другие модели. В частности Microsoft в ходе разработки Windows Hello внедрила в Windows Server 2016 новую модель доверия под названием Key Trust.

> Рассказ про Windows Hello заслуживает отдельного материала. Вкратце и грубо Windows Hello - технология позволяющая осуществлять двухфакторную аутентификацию, в том числе с использованием биометрии (отпечаток пальца, радужная оболочка глаза, распознавание лица), при помощи встроенных функций операционных систем семейства Windows. Работает указанная технология как для персональных компьютеров, так и в домене (Windows Hello for Business). Более подробно ознакомиться с Windows Hello можно по следующим ссылкам:
> "Exploiting Windows Hello for Business", BlackHat, 2019 год.
> Официальная документация Windows Hello

В Key Trust контроллер домена для проверки подлинности ключа клиента использует не сертификат, а специальный атрибут *msDS-KeyCredentialLink* содержащий открытый ключ учетной записи указанного клиента. Если быть точнее, то атрибут *msDS-KeyCredentialLink* является многозначным, так как у учетной записи может быть несколько устройств с которых она осуществляет вход в систему. Такое возможно, например, если у пользователя две рабочих станции, и соответственно для хранения закрытых ключей используются разные модули TPM.

При рассмотрении Key Trust может возникнуть ошибочное ощущение, что необходимость в центре сертификации теперь отсутствует. На самом деле это не так. Дело в том, что клиенты должны быть способны проверить подлинность открытого ключа контроллера домена, а для этого все же требуется сертификат, а значит и центр сертификации.

Таким образом в Key Trust клиенты проверяют подлинность контроллера домена по сертификату открытого ключа, а контроллер в свою очередь проверяет подлинность получаемых сообщений, пытаясь расшифровать их открытым ключом из атрибута учетной записи.

Практическая часть

Как представленная выше информация может пригодиться при тестировании на проникновение?

Во вступлении уже были анонсированы атаки Shadow Credentials и UnPAC the Hash, но есть и другие атаки. Теперь перейдем к детальному разбору возможных атак и тонкостей при их реализации.

PassTheCertificate

Условия для проведения атаки: наличие ключевой пары и соответствующего сертификата от учетной записи пользователя.

Результат успешной атаки: доступ к ресурсам домена с правами атакуемой учетной записи.

Начнем с наиболее простой атаки PassTheCertificate (PtC). По сути PassTheCertificate не атака, а полезный кирпичик будущих атак, представляющий собой легитимное действие.

Для начала рассмотрим в каких форматах могут храниться ключи с сертификатами. Как правило, сталкиваться придется с файлами в формате PEM или PFX.

PEM - включает файлы со следующими расширениями:

- .pem - предназначен для хранения открытого ключа учетной записи и сертификата к нему
- .key - содержит закрытый ключ учетной записи

PFX - содержит открытый, закрытый ключи учетной записи и сертификат в одном файле, который зашифрован с использованием пароля.

Чуть более подробно почитать о форматах файлов криптографических ключей можно здесь.

При наличие указанных файлов возможно осуществить аутентификацию и получить TGT к учетной записи для которой предназначались ключи.

Команды для выполнения атаки в Linux:

Для работы с PKINIT в Linux существует набор скриптов PKINITtools за авторством Dirk-jan.

Для аутентификации по сертификату воспользуемся утилитой gettgtpkinit.py

Аутентификация по сертификату в формате PFX:

```
gettgtpkinit.py -cert-pfx "PATH_TO_PFX_CERT" -pfx-pass "CERT_PASSWORD"
"FQDN_DOMAIN/TARGET_SAMNAME" "TGT_CCACHE_FILE"
```

Аутентификация по сертификату в формате PFX, кодированном в Base64:

```
gettgtpkinit.py -pfx-base64 $(cat "PATH_TO_B64_PFX_CERT") "FQDN_DOMAIN/TARGET_SAMNAME"
"TGT_CCACHE_FILE"
```

Аутентификация по сертификату в формате PEM:

```
gettgtpkinit.py -cert-pem "PATH_TO_PEM_CERT" -key-pem "PATH_TO_PEM_KEY"
"FQDN_DOMAIN/TARGET_SAMNAME" "TGT_CCACHE_FILE"
```

Команды для выполнения атаки в Windows:

Rubeus поддерживает работу с PFX-сертификатами, кодированными в Base64:

```
Rubeus.exe asktgt /user:"TARGET_SAMNAME" /certificate:"BASE64_CERTIFICATE"
/password:"CERTIFICATE_PASSWORD" /domain:"FQDN_DOMAIN" /dc:"DOMAIN_CONTROLLER" /show
```

В Windows сконвертировать PEM в PFX возможно с помощью openssl:

```
openssl pkcs12 -in "cert.pem" -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0"
-export -out "cert.pfx"
```

В продолжение PassTheCertificate логично использовать PassTheTicket (PassTheTicket), но есть и другие атаки которые будут рассмотрены далее.

Источник информации - thehacker.recipes

UnPAC the Hash

Условия для проведения атаки: наличие ключевой пары и соответствующего сертификата к учетной записи.

Результат успешной атаки: NT-хеш пароля учетной записи.

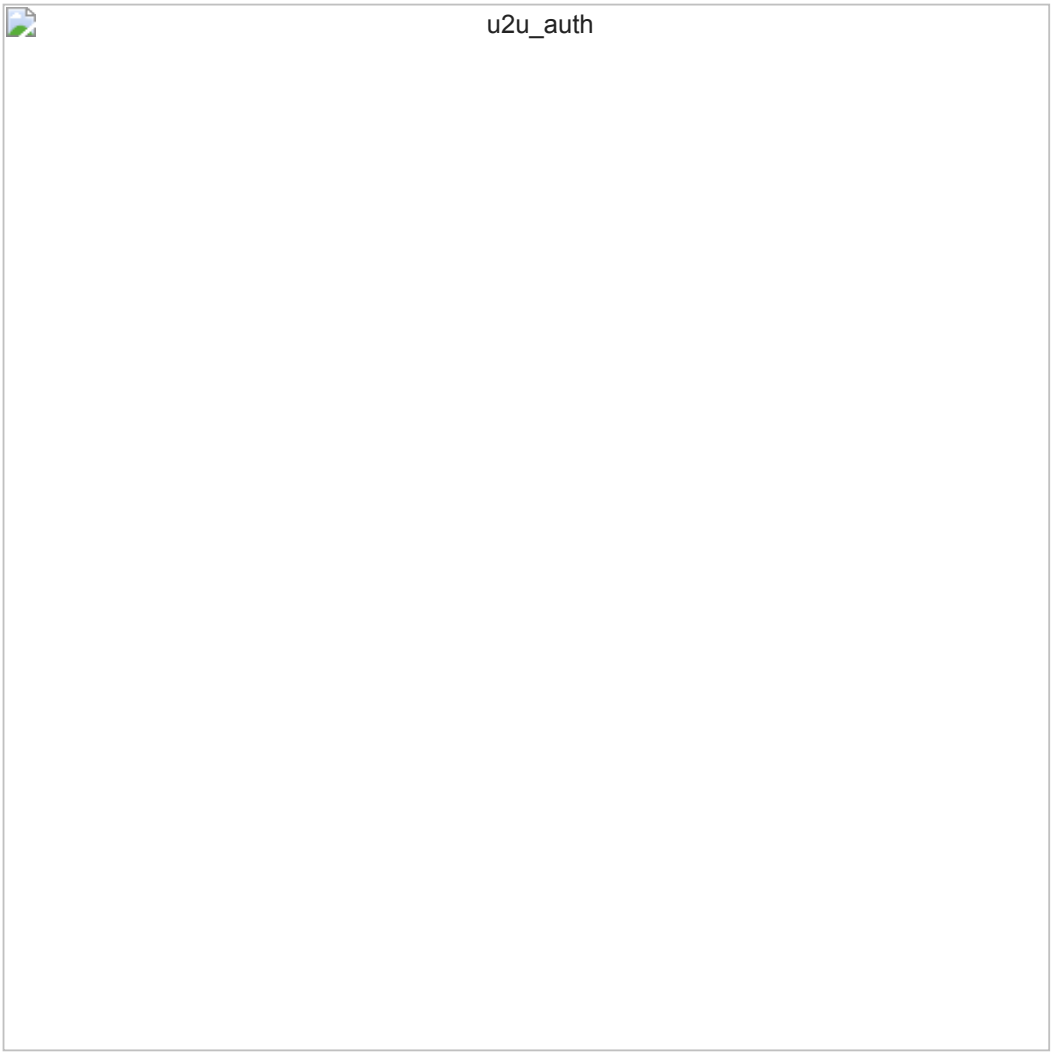
TGT это хорошо, но NT-хеш еще лучше. Для того, чтобы разобраться в атаке UnPAC the Hash рассмотрим, как устроена аутентификация User-to-User (U2U) по протоколу Kerberos.

User-to-User аутентификация

U2U-аутентификация изначально задумывалась, чтобы обычные пользователи на своих рабочих станциях могли предоставлять другим пользователям некоторые сервисы, например NFS или FTP.

Рабочая станция пользователя считается незащищенным объектом, поэтому использовать её для хранения долговременных секретов нецелесообразно. В этом случае в TGS-билете в качестве секрета сервиса применяется кратковременный сессионный “ключ пользователя для KDC”. Но ведь KDC не хранит сессионные ключи, а извлекает их из TGT. Тогда как же KDC зашифровывает TGS-билет?

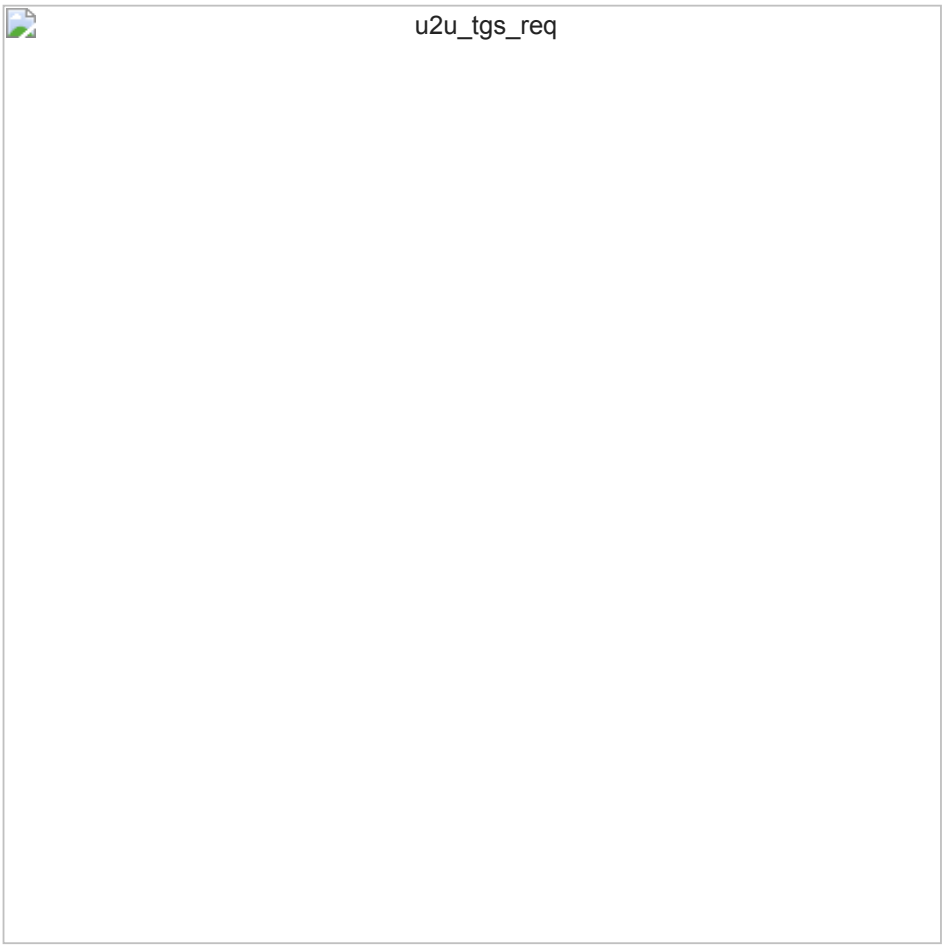
Для лучшего понимания происходящего пошагово разберем этапы U2U-аутентификация.



1. Начало U2U аутентификации

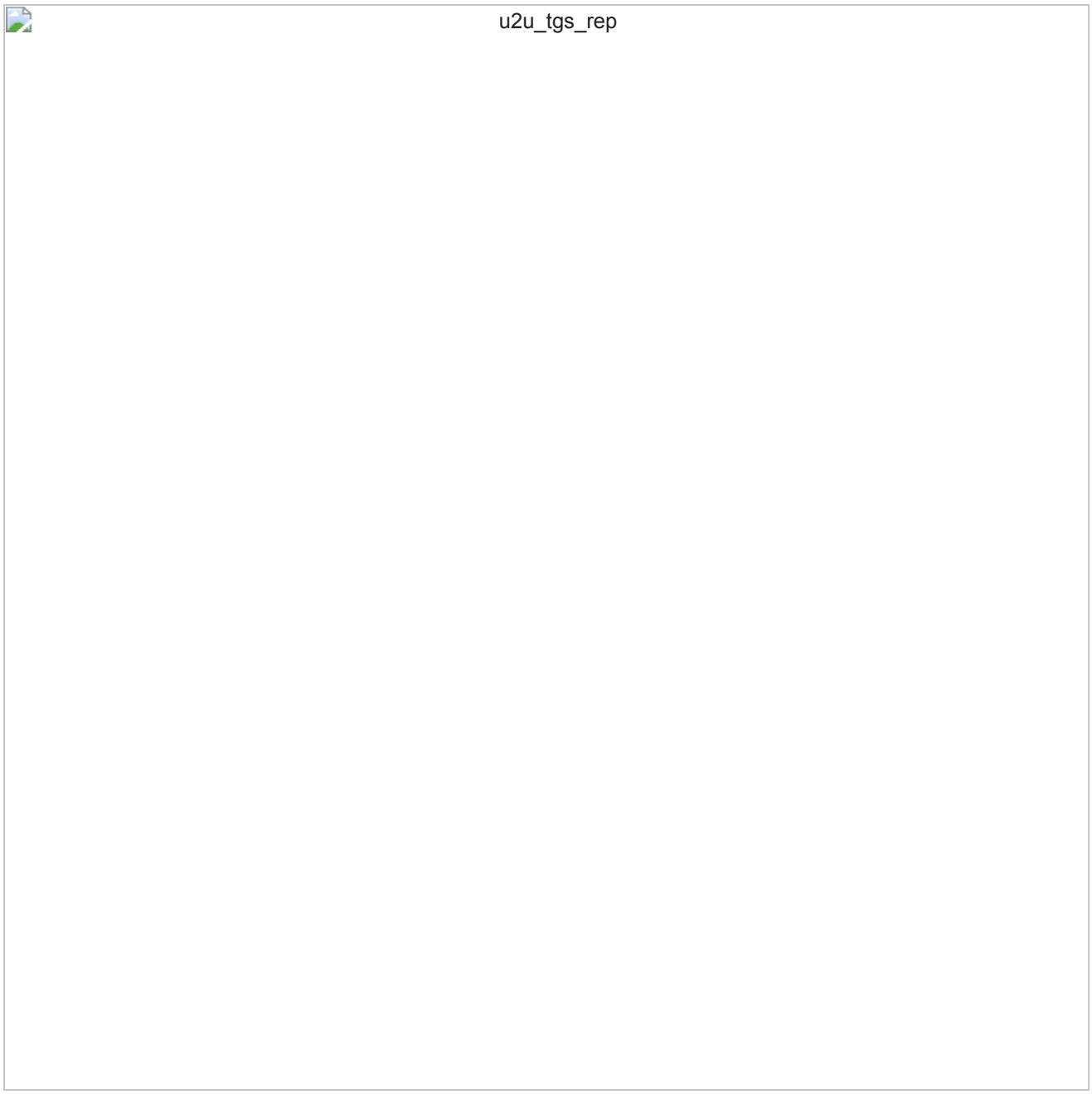
1. Алиса проходит аутентификацию к Сервису Боба и предъявляет TGS-билет, зашифрованный с использованием долговременного секрета Боба (например NT-хеша). Сервис Боба не способен расшифровать предъявляемый TGS-билет, так как указанный сервис работает на рабочей станции и не имеет доступа к долговременному ключу Боба. Поэтому в ответ Сервис Боба отправляет сообщение о том, что должен использоваться механизм U2U и прилагает TGT Боба.

> Передавать TGT в открытом виде нормально, несмотря на то что весь сетевой трафик считается известным третье стороне. Более того именно так и делается в KRB_AS_REP. Сам по себе TGT Боба без знания сессионного ключа для KDC не представляет никакой ценности и не позволяет проводить атаку PassTheTicket.



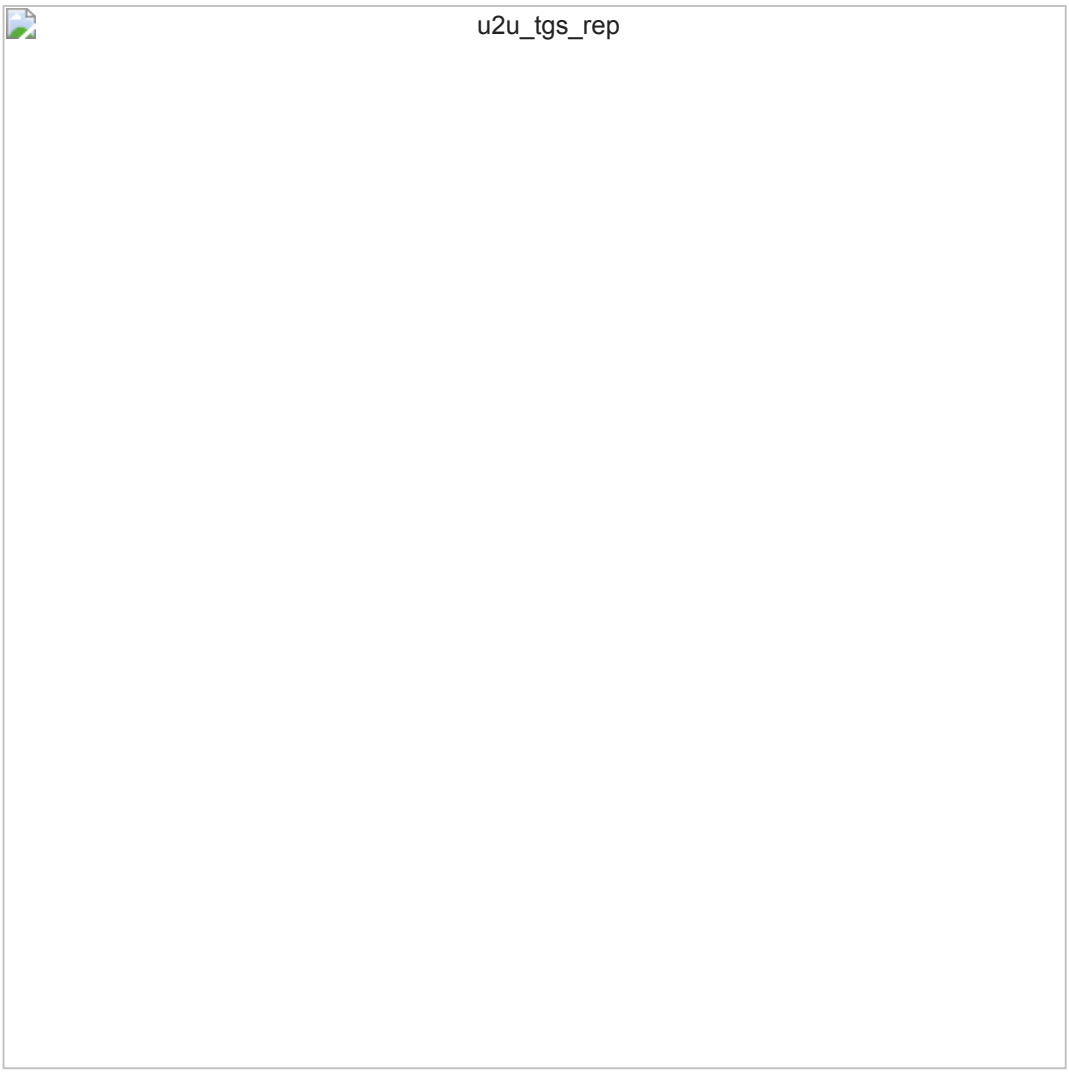
2. TGS_REQ при U2U аутентификации

2. Алиса запрашивает у контроллера домена TGS-билет с использованием U2U-аутентификации, о чем свидетельствует специальный флаг. Кроме того, в качестве дополнительного билета Алиса прилагает TGT Боба. Контроллер домена в свою очередь извлекает из полученных TGT сессионные ключи Алисы и Боба.



3. TGS_REP при U2U аутентификации

3. Контроллер домена отправляет Алисе TGS-билет зашифрованный с использованием извлеченного на предыдущем шаге сессионного ключа Боба.



4. AP_REQ при U2U аутентификации

4. Теперь Сервис Боба может расшифровать полученный TGS-билет и предоставить доступ для Алисы.

> Использование кратковременного ключа в качестве секрета также не позволяет провести Kerberoasting в отношении простых пользователей.

В итоге зачем были нужны непростые объяснения всех этих механизмов?

Объяснение атаки UnPAC the Hash

Дело в том, что пользователь может осуществить PKINIT + U2U запрос на аутентификацию к самому себе и таким образом получить свой NT-хеш.

Для начала представим, что Алиса уже получила TGT с использованием PKINIT, как это было рассмотрено ранее.

U2U_TGS_REQ при аутентификации к "самому себе"

Алиса обращается к KDC с запросом U2U-аутентификации к своему же сервису и предоставляет в качестве дополнительного билета свой TGT, полученный в результате PKINIT и при этом содержащий PAC со своим NT-хешем.

U2U_TGS_REP при аутентификации к "самому себе"

Контроллер домена возвращает Алисе TGS-билет, зашифрованный с использованием сессионного ключа Алисы для KDC. Алиса расшифровывает TGS-билет при помощи своего сессионного ключа, а затем расшифровывает свой NT-хеш при помощи AS-REP ключа.

Таким образом обладая ключевой парой и соответствующим сертификатом от учетной записи, возможно узнать NT-хеш пароля указанной учетной записи. На практике атака PassTheCert в связке с UnPAC the Hash активно используются для эксплуатации результатов других атак.

Команды для выполнения атаки в Linux:

Сначала запрашиваем TGT при помощи PKINIT:

```
gettgtpkinit.py -cert-pfx "PATH_TO_CERTIFICATE" -pfx-pass "CERTIFICATE_PASSWORD"
"FQDN_DOMAIN/TARGET_SAMNAME" "TGT_CCACHE_FILE"
```

С использованием полученного TGT и AS-REP ключа (отобразится в консоли в результате выполнения gettgtpkinit) получаем NT-хеш:

```
export KRB5CCNAME="TGT_CCACHE_FILE"
getnthash.py -key 'AS-REP key' 'FQDN_DOMAIN'/'TARGET_SAMNAME'
```

Команды для выполнения атаки в Windows:

```
Rubeus.exe asktgt /getcredentials /user:"TARGET_SAMNAME" /certificate:"BASE64_CERTIFICATE"
/password:"CERTIFICATE_PASSWORD" /domain:"FQDN_DOMAIN" /dc:"DOMAIN_CONTROLLER" /show
```

Источники информации:

Shadow Credentials

Условия для проведения атаки:

- Хотя бы один контроллер домена функционирует под управлением ОС Windows Server 2016 или выше
- Функциональный уровень домена Active Directory - Windows Server 2016
- Контроллер домена обладает сертификатом для своего открытого ключа
- Возможность осуществить запись в атрибут msDS-KeyCredentialLink целевой учетной записи

Результат успешной атаки: доступ к ресурсам домена с правами атакуемой учетной записи (NT-хеш).

Если атакующий обладает правом на запись в атрибут *msDS-KeyCredentialLink* некоторой учетной записи, то тогда он может поступить следующим образом:

1. Самостоятельно сгенерировать пару открытый-закрытый ключ
2. Записать открытый ключ в атрибут *msDS-KeyCredentialLink* атакуемой учетной записи (см. ранее)
3. Осуществить атаку Pass the Certificate
4. Осуществить атаку UnPAC the Hash и получить NT-хеш пароля атакуемой учетной записи

> Наличие права на запись в атрибут *msDS-KeyCredentialLink* является частным случаем наличия следующих прав: GenericAll, GenericWrite, WriteOwner, WriteDACL.

Реализация атаки

В 2022 году на конференции Phdays Шлюндин Павел представил доклад “Другой взгляд атакующего на ACL в AD” в котором среди прочего мимоходом рассматривается эксплуатация права записи в атрибут *msDS-KeyCredentialLink*.

Более того, автор выложил в открытый доступ тренировочный образ виртуальной машины со специально установленными уязвимыми настройками, а также инструмент ldap_shell, предназначенный для автоматизации эксплуатации атак на ACL.

Для демонстрации атаки Shadow Credentials на практике воспользуемся предоставленной виртуальной машиной (redteam.bro/user:P@ssw0rd).

Выявить право на запись в атрибут *msDS-KeyCredentialLink* или равноценное право возможно с помощью BloodHound. Отдельно возможность записи в атрибут *msDS-KeyCredentialLink* отображается с помощью ребра *AddKeyCredentialLink*, но мы рассмотрим более общий случай.

Часть цепочки атаки, построенной с использованием BH

Из анализа видно, что пользователь User опосредованно обладает правом *GenericWrite* (позволяет осуществлять запись в атрибут *msDS-KeyCredentialLink*) в отношении User1. Для реализации атаки воспользуемся утилитой ldap_shell.

Для начала выполним шаги, избавляющие от опосредованности и подводящие к интересующей нас атаке Shadow Credentials:

Необязательный этап

Теперь непосредственно проведем атаку Shadow Credentials:

Эксплуатация Shadow Credentials с помощью ldap_shell

Существуют и другие инструменты для реализации указанной атаки. Посмотреть больше примеров эксплуатации атаки Shadow Credentials можно в прохождениях машины Outdated на HackTheBox (пример).

Команды для выполнения атаки в Windows:

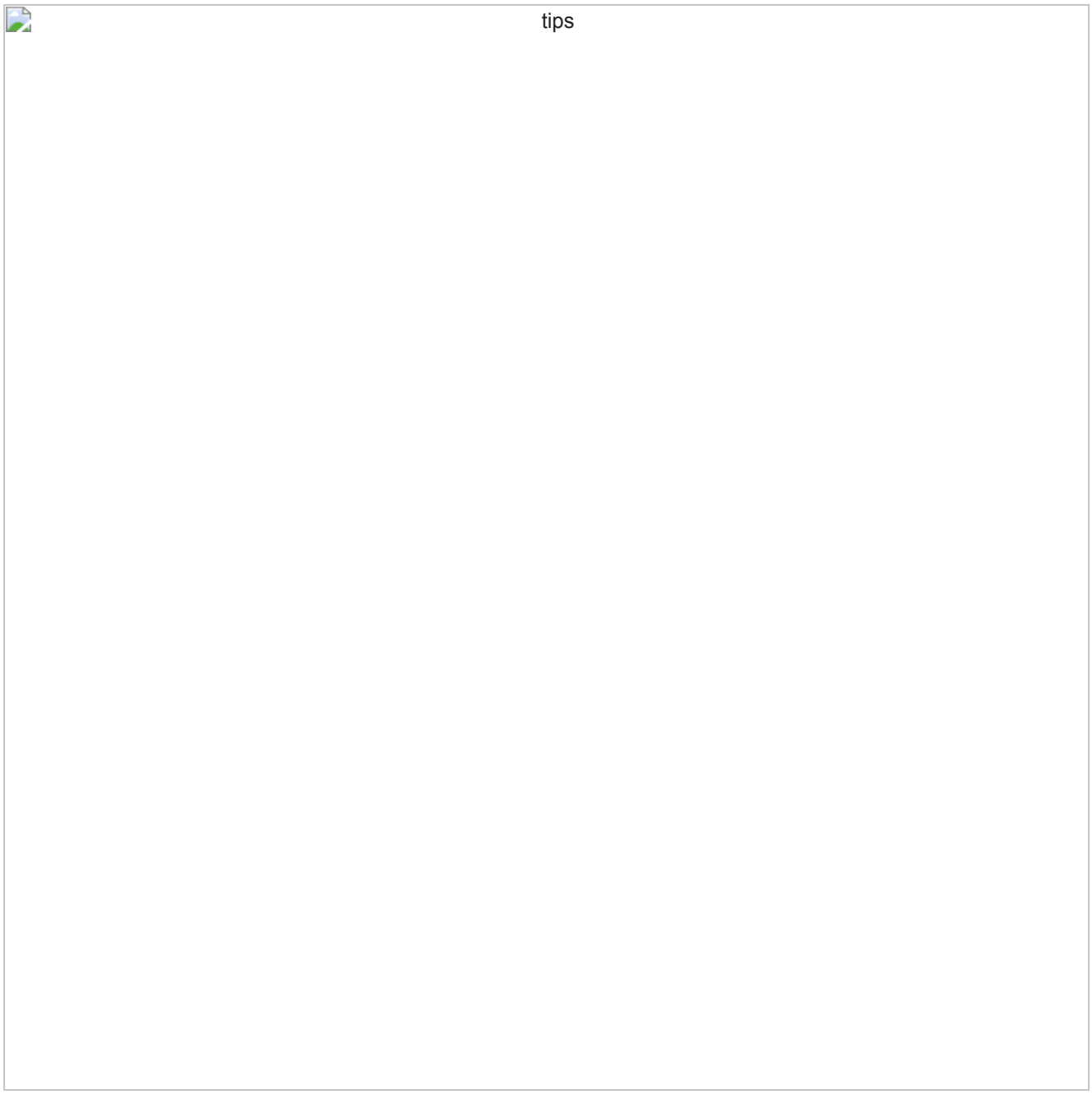
Одним из первых инструментов, эксплуатирующих Shadow Credentials был Whisker:

```
Whisker.exe add /target:"TARGET_SAMNAME" /domain:"FQDN_DOMAIN" /dc:"DOMAIN_CONTROLLER"
/path:"cert.pfx" /password:"pfx-password"
```

Команды для выполнения атаки в Linux:

Со временем был создан клон Whisker на python - PyWhisker:

```
pywhisker.py -d "FQDN_DOMAIN" -u "user1" -p "CERTIFICATE_PASSWORD" --target
"TARGET_SAMNAME" --action "list"
```



Пример реализации атаки Shadow Credentials от PTSwarm

Преимущества Shadow Credentials

Существуют другие способы эксплуатации Generic-прав. Проведем небольшое сравнение:

- в отличие от RBCD эксплуатация Shadow Credentials не требует создания новой машинной учетной записи, удалить которую возможно только при наличии прав уровня администратора домена
- в отличие от Целевого Kerberoasting эксплуатация Shadow Credentials занимает непродолжительное время и не зависит от сложности текущего пароля пользователя

Кроме того, эксплуатация Shadow Credentials имеет другие преимущества:

- Не требует перезаписи пароля к учетной записи, что иногда расценивается как отказ в обслуживании и не допускается при проведении тестирования на проникновение

- Позволяет закрепиться в домене, так как даже при смене пароля к скомпрометированной учетной записи атакующий сохраняет возможность аутентифицироваться по сгенерированной ключевой паре с помощью PKINIT.
- Атакующему не требуется наделять дополнительными правами доступа какой-либо из легитимных объектов домена, компрометация которого впоследствии может быть выявлена.
- Закрепление особенно актуально применительно к машинным учетным записям, потому что пароли машинных учетных записей обновляются каждый 30 дней. Кроме того машинные учетные записи могут самостоятельно осуществлять запись в свой атрибут *msDS-KeyCredentialLink* в случае если указанный атрибут пустой.

> Пользовательские учетные записи не могут самостоятельно изменить свой атрибут *msDS-KeyCredentialLink*.

Relay атака

Эксплуатировать Shadow Credentials возможно в том числе в ходе Relay-атаки.

Что такое Relay атака и как её осуществлять, заслуживает отдельного материала, но для тех кто хоть немного в теме приведу пример с необходимыми аргументами командной строки:

```
ntlmrelayx.py -t ldap://dc-ip --shadow-credentials
```

Источники:

[I’m bringing relaying back: A comprehensive guide on relaying anno 2022]

Enterprises Key Admin

Во многих инструментах (BloodHound, Pingcastle), выполняющих анализ небезопасных настроек Active Directory, особое внимание уделяется членству пользователей домена в определенных группах, например: Backup Operators, Account Operators, Server Operators и др.

Чуть подробнее об этом можно прочитать в следующих материалах:

Дело в том, что члены указанных групп неявно обладают правами уровня администратора домена. По умолчанию приведенные группы не содержат никаких членов за некоторым исключением, но в реальности в них можно встретить нестандартные учетные записи.

Применительно к теме статьи в Active Directory существуют две встроенные группы “Key Admins” и “Enterprise Key Admins”. Члены указанных групп обладают правом на изменение атрибута ms-DS-Key-Credential-Link для объектов компьютер, что позволяет осуществлять атаку Shadow Credentials.

В BloodHound наличие пользователей в группе “Key Admins” можно проверить следующим запросом:

```
match (u:User)-[:MemberOf]->(g:Group {name:"KEY ADMINS@DOMAIN.LOCAL"}) return u.name
```

Заключение

В настоящем материале были рассмотрены некоторые основные принципы работы расширения PKINIT и сертификатов в Active Directory. В дальнейшем были разобраны типовые техники атак с использованием рассмотренных принципов, а также приведены практические примеры и команды для реализации указанных атак.

Используемые источники

- “Shadow Credentials: Abusing Key Trust Account Mapping for Account Takeover”
- Книга “Kerberos: The Definitive Guide” за авторством Jason Garman
- Hacker Recipes
- Telegram канал “CyberSecrets”