Настройка VLESS + TCP + REALITY + VISION + uTLS #3518

iambabyninja started this conversation in **Show and tell**

iambabyninja on Jul 9, 2024

В данном руководстве мы с Вами подробно рассмотрим настройку VLESS TCP REALITY + FLOW + uTLS, а так же рассмотрим детально механизмы их работы.

Необходимо внимательно изучить это руководство полностью. На каждом этапе, от изучения основ до рассмотрения реализации различных механизмов и тонкостей настройки, мы будем углубляться в детали каждого предыдущего пункта. Такой подход позволит вам не только понять отдельные аспекты продукта, но и сформировать комплексное и целостное представление о его работе и возможностях.

В конечном итоге, последовательное и подробное изучение всех разделов обеспечит всестороннее понимание и позволит эффективно использовать продукт в вашей работе.

VLESS

Определение

VLESS — это легкий транспортный протокол, не сохраняющий состояние, который разделен на inbound и outbound части и может использоваться в качестве моста между клиентом и сервером Xray. Метод аутентификации протокола основан на UUID (Universally Unique Identifier).

Запрос и ответ

VLESS имеет следующую структуру запроса:

1 байт	16 байт	1 байт	M байт	1 байт	2 байт
Версия протокола	Соответствующий UUID	Длина дополнительной информации M	Дополнительная информация в ProtoBuf	Инструкции	Пор

VLESS имеет следующую структуру ответа:

1 байт	1 байт	N байт	Y байт
Версия протокола, соответствующая запросу	Длина дополнительной информации N	Дополнительная информация в ProtoBuf	Ответ

Режим передачи

VLESS поддерживает множество режимов передачи: TCP, gRPC, H2, QUIC, WebSocket , mKCP.

Однако, из-за совокупности факторов, о которых будет сказано немногим позже, для REALITY рекомендуется использовать TCP как единственный надежный режим передачи данных.

VLESS не предусматривает шифрования, поэтому обязательным условием для его использования является наличие надежного канала, такого как TLS или REALITY.

REALITY

Введение

REALITY представляет собой усовершенствование существующих технологий TLS, реализуя полный TLS 1.3 с определенным SNI веб-сайта для маскировки, при этом сохраняя внешний вид трафика, аналогичный обычному TLS 1.3.
Для упрощения дальнейшего восприятия важно запомнить, что REALITY — это форк пакета TLS из GOLANG.

Задачи

Ключевые задачи, решенные при разработке REALITY:

1. **Устойчивость к атакам на цепочку сертификатов**
   REALITY улучшает устойчивость к атакам, связанным с цепочкой сертификатов. В традиционном TLS безопасность соединения зависит от доверия к центрам сертификации (CA). Атаки на цепочку сертификатов возможны, если злоумышленник скомпрометирует или подделает сертификат, выданный CA. REALITY уменьшает эту уязвимость, предоставляя дополнительные механизмы безопасности.

2. **Сокрытие отпечатков TLS**
   Отпечатки TLS — это уникальные характеристики, которые могут использоваться для идентификации и блокировки определенных типов зашифрованного трафика. В странах с жесткой интернет-цензурой правительства могут использовать отпечатки TLS для обнаружения и блокировки трафика. REALITY минимизирует эти отпечатки, делая трафик менее узнаваемым и более устойчивым к цензуре и блокировкам.

3. **Прямая секретность**
   Этот принцип безопасности гарантирует, что даже в случае компрометации ключей шифрования в будущем злоумышленники не смогут расшифровать прошлые сессии. REALITY поддерживает прямую секретность, обеспечивая, что каждая сессия зашифрована уникальным сеансовым ключом.

4. **Отсутствие зависимости от домена**
   REALITY позволяет указывать чужие сайты без необходимости покупки собственного домена или настройки TLS-сервера, что делает его более удобным для использования и позволяет представлять полностью реальный TLS с определенным SNI посреднику.

5. **Совместимость с существующими протоколами**
   REALITY разработан для совместимости с существующими протоколами, что позволяет интегрировать его в существующие системы без кардинальных изменений. Однако это не рекомендуется из-за существующих и легко распознаваемых особенностей TLS.

6. **Прозрачность для пользователей и серверов**
   Протокол предназначен для работы таким образом, чтобы быть прозрачным как для клиентов, так и для серверов, что облегчает его внедрение и использование.

7. **Поддержка современных стандартов**
   REALITY поддерживает современные стандарты безопасности и шифрования, обеспечивая высокий уровень защиты данных.

8. **Гибкость и настройка**
   Протокол предлагает различные опции настройки для удовлетворения специфических требований безопасности и производительности.

Философия дизайна REALITY

При разработке REALITY, придерживались трех основных принципов:

1. **Максимальный уровень безопасности с ограничением факторов, контролируемых человеком, и минимизацией их влияния**
   Все процессы протокола автоматизированы для минимизации человеческого вмешательства, что снижает риск ошибок.
2. **Доверие к серверу, недоверие к клиенту, даже если информация о нодах, хранящаяся на клиенте, может быть полностью скомпрометирована**
   Серверные данные и ключи тщательно защищены, контроль доступа строго осуществляется сервером, а критически важные данные хранятся только на сервере, исключая их компрометацию.
3. **Выборочность сервера в отношении клиента**
   Например, сервер может отказать в подключении устаревших версий Xray-core для предотвращения уязвимостей и ошибок.

Минимальные требования

Минимальные требования к камуфляжному веб-сайту:

1. Зарубежный веб-сайт.

2. Поддержка TLS 1.3 и H2.
3. Адрес, который не перенаправляется куда-либо еще (домен может быть перенаправлен на www).

Дополнительные критерии:

1. **IP-адрес камуфляжного веб-сайта близок к IP-адресу вашего сервера.**
   Наличие IP-адреса, близкого к вашему серверу, означает, что камуфляжный веб-сайт имеет IP-адрес, который близок к IP-адресу вашего сервера с точки зрения сетевого расстояния или географического местоположения.
   Это может сделать соединение более эффективным и менее подозрительным.

2. **Сообщения о рукопожатии после «Server Hello» шифруются вместе.**
   Некоторые веб-сайты могут не шифровать сообщения рукопожатия после «Server Hello» вместе, а вместо этого отправлять их отдельно в виде открытого текста. Это может предоставить информацию о сервере и клиенте, например, поддерживаемые ими наборы шифров, расширения и сертификаты. Злоумышленники могут использовать это для идентификации сервера и клиента.
   Поэтому сервер и клиент должны обменяться зашифрованными данными после первоначального подтверждения TLS, предотвращая подслушивание или вмешательство третьих лиц.

3. **Сервер поддерживает OCSP.**
   Сервер поддерживает сшивание OCSP, что означает, что сервер может предоставить подтверждение действительности своего сертификата, не требуя от клиента обращения в центр сертификации.
   Это может улучшить производительность и конфиденциальность соединения.

Принцип работы

Существует ошибочное мнение, что REALITY использует метод "воровства сертификатов". REALITY реализует полный TLS 1.3, который не требует и не осуществляет подмены сертификатов, так как в TLS 1.3 сертификаты зашифрованы и недоступны для просмотра посредником.

После Server Hello все сообщения зашифрованы, поэтому сервер REALITY перехватывает только сообщения рукопожатия сервера (Server Hello) от камуфляжного сайта включая их длину и временные характеристики, заменяя key_share

Из чего следует, что клиент REALITY получает временный доверенный сертификат, выданный временным аутентификационным ключом, и в нормальных условиях никогда не получает настоящий сертификат камуфляжного сайта.

Общая схема взаимодействия выглядит следующим образом:

1. Прокси клиент -> Прокси сервер: "Client Hello"
2. Прокси сервер -> Камуфляжный сайт: "Client Hello"
3. Камуфляжный сайт -> Прокси сервер: "Server Hello"
4. Прокси сервер -> Прокси клиент: "Server Hello"

Клиент отправляет запрос TLS Client-Hello на сервер Reality. Сервер Reality перенаправляет этот запрос на адрес камуфляжного сайта. Камуфляжный сайт отвечает с TLS Server-Hello, сервер получает его и характеристики длины последующих сообщений рукопожатия, а затем пересылает обратно клиенту. Если специальные поля в запросе TLS Client-Hello клиента были корректными, сервер Reality переключается в режим прокси. Далее начинается передача данных через VLESS, включая авторизацию по UUID и другие связанные процессы.

Поток данных

Вообще, основная наша логика заключается в использовании TLS для маскировки прокси-трафика среди самого обычного интернет-трафика.
Как говорят: «Спрятать дерево в лесу».
Для обычного TLS прокси-протокола (например, Trojan), клиент прокси пользователя устанавливает настоящее TLS-соединение с прокси-сервером, и через этот зашифрованный туннель приложение (например, браузер) устанавливает еще одно TLS-соединение с целевым сервером (например, Google). В этот момент браузер и сервер Google обеспечивают end-to-end шифрование, и никто (включая наш прокси) не может расшифровать или подделать отправляемую информацию.

Вот краткая схема:

```
Браузер ---- Прокси-клиент ==== Прокси-сервер ---- Google
```

И тут мы понимаем, что когда прокси-данные проходят через цензора, они фактически шифруются дважды. Однако, в этом процессе, 99% трафика на самом деле не требует дополнительного шифрования. Поскольку данные, зашифрованные с помощью TLS 1.3 (а мы с вами помним, что REALITY = TLS 1.3), внешне выглядят абсолютно одинаково, цензор не может их различить.

Таким образом, логика идеального прокси должна быть следующей:

1. Устанавливаем TLS-соединение.
2. Внутри зашифрованного туннеля отслеживаем коммуникацию между браузером и сайтом, определяя, является ли текущее соединение TLS 1.3.
3. Если нет, то используем обычный метод TLS-прокси, продолжая использовать зашифрованный туннель.
4. Если да, то дожидаемся начала передачи зашифрованных данных обеими сторонами, а клиент прокси вставляет UUID в первый внутренний зашифрованный пакет, сигнализируя серверу прокси: мы готовы к "голому" соединению!
5. Начиная со второго внутреннего зашифрованного пакета, мы больше не выполняем никаких операций с данными, а просто копируем их.

Поздравляю! Мы изобрели xtls-rprx-vision.

Мы можем заметить, что:

1. Внешний TLS, инициированный прокси, является настоящим, поэтому цензор не может его взломать.
2. "Голый" трафик на стороне прокси просто копируется без изменений. Если цензор попытается вмешаться, он получит стандартный ответ от браузера и сервера Google, который не будет отличаться от обычного доступа.

3. Сигналом для "голого" соединения служит UUID, который является приватной информацией прокси, чтобы избежать уязвимостей, подобных тем, что были в Go, связанных с использованием кодовых характеристик.

Одним из важных недостатков обычного TLS-прокси является "шифрование в шифровании". Как обсуждалось выше, хотя внешний вид зашифрованных пакетов неотличим для цензора, неизбежным является увеличение заголовка каждого пакета с каждым уровнем шифрования. Это увеличение может быть незначительным, но для маленьких данных (например, пакетов ответов) оно может быть более заметным, и некоторые пакеты могут превышать ограничения MTU сетевого уровня. Самое важное - каждый пакет увеличивается на одинаковую длину, что создает определенные статистические характеристики.

Можно сказать, что когда передаются данные TLS 1.3, 99% наших пакетов имеют почти идеальные характеристики трафика, потому что это оригинальные данные, не обработанные прокси.

Наверное, у вас возник вопрос: почему мы говорим, что 99% пакетов являются оригинальными данными? Что представляют собой оставшиеся 1%? Нам нужно исследовать, что происходит в типичном прокси при встрече с трафиком TLS 1.3 в первых нескольких пакетах.

Наглядно, после установления зашифрованного канала:

1. Первый пакет: Прокси-клиент -> Прокси-сервер: "Привет, цель этого прокси-сессии - Google, вот мой UUID."
2. Второй пакет: Прокси-клиент <- Прокси-сервер: "Привет, получил твой запрос на прокси, начинай отправлять данные."
3. Третий пакет: Браузер -> Прокси-клиент -> Прокси-сервер -> Google: "Привет, Google, я собираюсь начать с тобой зашифрованное общение, вот мои поддерживаемые методы шифрования..." (Этот пакет также называется TLS Client Hello).
4. Четвертый пакет: Браузер <- Прокси-клиент <- Прокси-сервер <- Google: "Привет, пользователь, вот сертификат Google, мы будем использовать шифрование TLS_AES_128_GCM_SHA256, давай начнем зашифрованное общение!" (Этот пакет также называется TLS Server Hello).
5. Пятый пакет: Браузер -> Прокси-клиент -> Прокси-сервер -> Google: "Понял, давай начнем зашифрованное общение!"

Как вы знаете, протоколы прокси разнообразны, но основа одна и та же: если пользователь использует любое TLS-соединение, необходимо выполнить вышеупомянутый процесс рукопожатия. Как упоминалось ранее, внешнее TLS-шифрование можно считать абсолютно безопасным, но для цензора, помимо взлома информации, есть возможность использовать некоторые дополнительные данные для идентификации этих пяти пакетов.

Самой очевидной характеристикой этих пяти пакетов является их длина. В частности:

1. Первый пакет: очень короткий, единственная переменная - целевой адрес.
2. Второй пакет: очень короткий, почти фиксированный.
3. Третий пакет: короткий, с небольшими изменениями, почти единственная переменная - целевой SNI.
4. Четвертый пакет: длинный, с большими изменениями.
5. Пятый пакет: очень короткий, с небольшими изменениями.

Можно интуитивно почувствовать, что характеристики длины этих пакетов довольно очевидны. В Vision подход к решению этой проблемы также прост: мы увеличиваем длину каждого короткого пакета до диапазона от 900 до 1400. Обратите внимание, что этот метод отличается от традиционного случайного заполнения; мы не просто добавляем заполнение ко всем пакетам, а основываясь на нашем анализе внутреннего трафика, целенаправленно заполняем характерные пакеты TLS-рукопожатия.

Важное момент состоит в том, что при использовании vision, использовать mux не представляется возможным.

Как мы обсуждали выше, использование vision означает, что прокси не выполняет никаких операций, кроме копирования 1-в-1 от клиентского соединения к целевому соединению.

Mux означает, что прокси использует одно соединение для передачи нескольких клиентских соединений.

В пределах одного mux-соединения нельзя использовать vision, так как нет способа определить, куда отправлять данные.

Параметры REALITY

После того, как мы поняли базовые механизмы нашего прокси, мы можем перейти к его настройке. Ниже приведено полное описание всех параметров Reality, которыми Вы можете оперировать при настройке.

```
{
  "dest": "example.com:443",
  "serverNames": ["example.com", "www.example.com"],
  "privateKey": "MMX7m0Mj3faUstoEm5NBdegeXkHG6ZB78xzBv2n3ZUA",
  "publicKey": "7xUz-xONEFhoGAb7XdmzhIUYybAlCnRvd1L0Ax_7yk4",
  "shortIds": ["", "0123456789abcdef"],
  "spiderX": "/blog",
  "maxTimeDiff": 0,
  "show": false,
  "minClientVer": "1.8.6",
  "maxClientVer": "1.8.15",
  "fingerprint": "chrome",
  "xver": 0
}
```

DEST/SNI

Выше по тексту, мы постоянно говорили о камуфляжном веб-сайте.
Настройка данных полей, будет играть ключевую роль в вашем прокси сервисе, поэтому выбору DEST/SNI требует уделить особенно много времени.

> `dest` : строка

Обязательное поле.

Заполняется на сервере.

Пример: `example.com:443`

Пример: `228.008.114.881:443`

Пример: `/var/lib/marzban.socket`

Пример: `8443`

`"dest"` здесь означает "цель", которую REALITY имитирует в реальном времени, и она (цель) взаимодействует с REALITY.

`"dest"` должен соответствовать минимальным, выдвигаемым требованиям.

Как мы с Вами обсуждали выше, по принципу своей работы, по крайней мере на данный момент, REALITY каждый раз должен получать пакет рукопожатия, поэтому важно учитывать, насколько близок и стабилен целевой сайт, так как выбор этого самого целевого сайта/домена существенно влияет на задержку, скорость и его стабильность.

Выше, мы с Вами ввели термин, что REALITY имитирует цель, в связи с чем мы можем раскрыть несколько несколько нюансов, которые люди записывают в минусы, хотя это и не так:

1. Если dest «лежит», то и наше прокси соединение не будет установлено
   Считаю большим плюсом данный нюанс, ведь как мы рассмотрели выше, для режима прокси клиент должен получить временный доверенный сертификат, подписанный временным ключом аутентификации, во всех иных случаях это будет просто port forward на dest, соответственно, было бы странно, при «лежащем» dest отправлять цензора в пустоту и тем самым раскрывая свой прокси.
2. По той же причине не сделан кэш или сборка предварительных данных, ведь на время кэша, характеристики dest могли измениться

> `serverNames` : массив строк

Обязательное поле.

Заполняется на сервере и на клиенте.

Пример: `["example.com", "www.example.com"]`

Пример: `[" "]`

Список разрешенных SNI.

На данный момент, wildcard не поддерживается.

При настройке этих параметров, важно помнить следующее:

На самом деле, если у dest есть действительный сертификат по умолчанию, клиент может использовать другие SNI для подключения, так как сервер будет отвечать с характеристиками, соответствующими ответу dest с сертификатом по умолчанию.

Поэтому сервер обязан ограничивать диапазон допустимых значений для serverNames, чтобы предотвратить случайное использование SNI клиентами, если только это не сделано нарочно, и хотя REALITY поддерживает такую возможность, но если указанный dest фактически не имеет того serverName, это может стать явным признаком, поэтому не следует злоупотреблять этим (так же как и пустым serverName).

В совокупности:

1. Если имена сервера (serverName) в клиентской и серверной конфигурациях совпадают, но отличаются от имен, указанных в SSL-сертификате, соединение может быть установлено.
2. Если имена сервера (serverName) в клиентской и серверной конфигурациях отличаются и не совпадают с именами, указанными в SSL-сертификате, соединение не устанавливается.

Обычно параметр servername заполняется, при необходимости, соответствующими SANs, перечисленным в SSL-сертификате для веб-сайта в dest.

Как было сказано выше, есть нюанс в том, что для обеспечения эффекта маскировки, Xray будет напрямую перенаправлять трафик с недействительной аутентификацией (незаконные запросы reality) на указанный dest. Если IP-адрес сайта, указанного в dest, особенный (например, сайт использует CDN CloudFlare), это может привести к тому, что ваш сервер будет выступать в роли port forward CloudFlare, что может вызвать утечку трафика после сканирования.
Чтобы предотвратить такую ситуацию, можно рассмотреть возможность использования Nginx или других методов для фильтрации SNI, которые не соответствуют требованиям.

Очень важно, не использовать крупные сайты в качестве DEST/SNI*

Множество крупных сайтов имеют CDN в стране, поэтому их использование не рекомендуется (это например касается Google и других крупных игроков, хотя Microsoft в 22 году отключила CDN в РФ). Ведь если у сайта есть сервера в стране, то с точки зрения наблюдателя (цензора) в нормальных условиях, вы должны обращаться к локальным адресам внутри страны, а не выходить за ее пределы.

Так же, например, в случае с официальным репозиторием REALITY, dl.google.com приводится только для иллюстрации.

Никто и никогда, в здравом уме, не станет использовать Google в качестве dest.

Для очень немногих сайтов, например, для того же dl.google.com, Chrome добавляет дополнительную информацию в Client Finished, но из-за недостатков библиотеки uTLS мы этого не делаем.

Дак что же делать?

Если у Вас нет своего домена, и если Вы не находитесь в регионе, где необходимо использовать домены из белого списка, то рекомендуется использовать отличный инструмент https://github.com/XTLS/RealiTLScanner для сканирования соседних доменов

Если у вас есть собственный домен - используйте его.

Приватные/публичные ключи

На этом пункте хотелось бы остановиться подробнее, для лучшего понимания внутренних механизмов.

Как мы обсуждали ранее, REALITY это tls 1.3, соответственно, все механизмы для него идентичны. В данном случае, алгоритмом обмена ключами по умолчанию будет являться ECDHE с кривой X25519.

Быть может Вы задаетесь вопросом, почему в REALITY для аутентификации не используется напрямую UUID, а добавлен механизм публичных и приватных ключей?

На самом деле, если использовать симметричные ключи, то при получении конфигурации клиента, цензор сможет провести атаку MITM, но если использовать публичные и приватные ключи X25519 в сочетании с key_share TLSv1.3, то даже при получении публичного ключа клиента цензор не сможет использовать его для проверки соединения как REALITY, не говоря уже о проведении эффективной атаки.

Как мы обсуждали выше, REALITY разработан с учетом того, что конфигурация клиента может быть раскрыта, поэтому безопасность здесь на высшем уровне: защитите приватный ключ сервера REALITY, и ваш трафик будет в безопасности.

Конечно, регулярная смена публичных и приватных ключей еще лучше.

Почему это очень важно и почему мы уделили с Вами столько времени этому разделу?

Простой пример: если вы случайно раскроете пароль SS или UUID VMess, цензор сможет расшифровать весь ваш прошлый и будущий зашифрованный трафик, а также провести идеальную атаку MITM.

Вы можете спросить, как же может произойти утечка? Облачное резервное копирование телефона, загрузка через клавиатуру, буфер обмена, отечественное программное обеспечение и другие неизвестные нам способы.

> `privateKey` : строка

Обязательное значение

Задается ~~только~~ на сервере

Пример: `MMX7m0Mj3faUstoEm5NBdegeXkHG6ZB78xzBv2n3ZUA`

Сгенерировать ключи можно путем выполнения команды в консоли
`./xray x25519`

Для Marzban команда будет выглядеть `docker exec marzban-marzban-1 xray x25519`

Сервер использует privateKey из конфигурации и key_share из Client Hello для вычисления общего секретного ключа, затем с помощью HKDF генерирует "временный ключ аутентификации", используемый для дешифрования и проверки запроса клиента, после чего генерирует временный доверенный сертификат Ed25519, подпись которого является HMAC "временного ключа аутентификации" для его публичного ключа.

> `publicKey` : строка

Заполняется только на клиенте.

Пример : `7xUz-xONEFhoGAb7XdmzhIUYybAlCnRvd1L0Ax_7yk4`

клиент использует приватный ключ, соответствующий key_share в Client Hello, и publicKey из конфигурации для вычисления общего секретного ключа, затем с помощью HKDF генерирует "временный ключ аутентификации", использующийся для AEAD аутентификации и шифрования номера версии, временной метки и Short ID, с приложенными данными всего процесса рукопожатия, результат заполняется в session ID для проверки сервером запроса.

Таким образом, publicKey и privateKey в конфигурации используются для установки общего секрета между клиентом и сервером, и для генерации временных аутентификационных ключей и временных доверенных сертификатов, необходимых для безопасного и аутентифицированного обмена данными в рамках REALITY.

Многих ошибочно считают, что ключи, состоящие из 256 бит хуже, чем ключи 2048 бит, не беря в расчет того, что RSA и DHE требуют больших ключей для обеспечения аналогичной безопасности по сравнению с алгоритмами, основанными на эллиптических кривых.
Для простоты понимания, например в RSA, идет разложении числа на простые множители.
Раньше это была сложная задача, но она становится все легче с ростом мощности компьютеров и для того что бы защититься от них(взлома с помощью их), нужно использовать большие ключи, 2048 или 3072 бит или еще больше.
В тот же момент x25519, основан на более сложной математической задаче, которая труднее решается даже при наличии супер мощных компьютеров, и из за сложности, разложение логарифма на эллиптических кривых, обеспечивает высокую безопасность при меньших размерах ключей.

Как мы обсуждали выше, ключ длиной 256 бит для X25519 дает уровень безопасности, эквивалентный ключу длиной 3072 бит для RSA.

ShortID

> `shortId` : массив строк

Обязательное поле.

Список shortId, доступных клиенту используется для различения разных клиентов.

Диапазон от 0 до f, длина должна быть кратна 2 и не превышать 16 символов.

Можно сгенерировать путем выполнения в консоли команды `openssl rand -hex 8`

Если включены пустые значения, shortId клиента может быть пустым.

Пример: `[" ", "e13c3f07bcffd6e9"]`

Пример: `[" "]`

Пример: `["c28e3p08uiqqh6e5", "e13c3f07bcffd6e9"]`

Разделом выше, мы с Вами рассмотрели где и когда принимает участие данные параметр.

SpiderX

> `spiderX` : строка

Необязательное поле.

Заполняется на клиенте.

Пример: `/`

Пример: `/blog`

Рекомендуется, чтобы начальный путь и параметры сканера были разными для каждого клиента.

Для того, что бы понять, что такое «паук», или режим сканирования, нам нужно еще немного окунуться в REALITY.

Ранее мы все время рассматривали только один вариант, когда REALITY получал временный доверенный сертификат и устанавливал прокси соединения, но на самом деле, REALITY, может различать несколько типов сертификатов:

1. временный доверенный
2. настоящий
3. недействительный

при получении различных типов сертификатов, клиент reality выполняет соотвествующие действия:

1. При получении временного доверенного сертификата устанавливается прокси-соединение, и всё идет по обычной схеме.
2. При получении настоящего сертификата переходит в режим сканирования (crawler).
3. При получении недействительного сертификата отправляется TLS-предупреждение и соединение разрывается.

Соотвественно, для каждого из случаев, когда клиент reality получает настоящий сертификат, а именно:

1. Сервер REALITY отклоняет Client Hello клиента, и трафик перенаправляется на целевой веб-сайт.
2. Client Hello клиента перенаправляется на целевой веб-сайт посредником.
3. Атаки MITM, возможно с участием целевого веб-сайта или атака цепочки сертификатов.

То он (клиент), переходит в режим сканирования (crawler):
Запускаем

1. Запускаем одно соединение HTTP/2 с dest , и отправляются параллельно по одному и тому же соединению, GET-запросы к различным URL-адресам на целевом сервере (target/dest), по схеме Целевой сайт + path, где изначальный path определяется параметром SpiderX.

2. по факту получения ответов на наши запросы они анализируются, и все найденные ссылки добавляются в список путей (path) и это продолжается.

На самом деле механизм немного сложнее (с установкой UA, Referrer, рандомизацией запросов, времени ожидания и так далее, но мы это опустим)

Соотвественно, этим механизмом, мы (клиент reality) имитируем поведение реального пользователя в выше перечисленных, экстренных случаях.

MaxTimeDiff

> `MaxTimeDiff` : ЧИСЛО

Необязательное поле.

Заполняется на сервере.

Максимальная разрешенная разница во времени в миллисекундах.

Пример: `0`

Пример: `60000`

MaxTimeDiff является страховкой для того, что бы предотвратить возможные, недоброжелательные манипуляции посредника.

В Client Hello протокола REALITY все элементы максимально задействованы, а дополнительные данные AEAD включают весь Client Hello.

Посредник не может изменить ни один бит, и все что ему остаётся, это только повторная отправка и хоть она и бесполезна (поскольку у посредника нет временного приватного ключа, соответствующего key_share, он не сможет расшифровать сообщение сервера), тем не менее, это все еще может вызвать ненужные ответы сервера.

А вдруг, если данных много, и у него случайно есть соответствующий приватный ключ?

Поэтому в REALITY есть зашифрованный таймстамп.

На самом деле, таймстамп, отправляемый клиентом, точен только до секунды. Единица измерения maxTimeDiff – миллисекунды,

Если нужно установить maxTimeDiff, то рекомендуется начать со значения 60000+, если не хотите синхронизироваться, можно даже с одного дня.

SHOW

> `show` : булево значение

Вывод отладочной информации

Заполняется на сервере.

Пример: `true`

minClientVer

> `minClientVer` : строка

Необязательное значение.

Заполняется на сервере.

Минимально разрешенная версия клиента, от которого будет принято подключение

Пример: `1.8.6`

maxClientVer

> `maxClientVer` : строка

Необязательное значение.

Заполняется на сервере.

Максимально разрешенная версия клиента, от которого будет принято подключение

Пример: `1.8.6`

fingerprint

`fingerprint` : строка

Обязательное поле.

Заполняется на клиенте.

Пример: `iOS`

Указывает отпечаток сообщения TLS Client Hello.

Если значение не задано, то имитация отпечатка не будет включена.

При включении Xray будет **имитировать** отпечаток TLS через библиотеку uTLS или сгенерирует его случайным образом.

Поддерживаются три типа опций:

1. Имитировать отпечатки TLS последних версий популярных браузеров, включая:

- `"chrome"`
- `"firefox"`
- `"safari"`
- `"ios"`
- `"android"`
- `"edge"`
- `"360"`
- `"qq"`

2. Сгенерировать отпечаток автоматически при запуске xray

- `"random"` : случайным образом выбрать один из актуальных браузеров
- `"randomized"` : сгенерировать полностью случайный и уникальный отпечаток (100% совместим с TLS 1.3, используя X25519)

3. Использовать собственные переменные отпечатков uTLS, такие как `"HelloRandomizedNoALPN"` `"HelloChrome_106_Shuffle"` .

Эта функция только **имитирует** отпечаток сообщения TLS Client Hello, оставляя другие поведения такими же, как в оригинальном Go TLS.

xver

> `xver` : строка

Необязательное значение

Заполняется на сервере.

Пример: `0`

Протокол PROXY используется для передачи реального исходного IP-адреса и порта запроса.

Версия может быть установлена на 1 или 2, со значением по умолчанию 0, что означает отсутствие использования протокола PROXY. Рекомендуется использовать версию 1, если это необходимо.

На данный момент версии 1 и 2 имеют одинаковую функциональность, но различную структуру: версия 1 представляет собой текстовый формат, а версия 2 — бинарный.

Настройка сервера

Выше, мы с Вами разобрали основные параметры доступные для конфигурирования.
Теперь же давайте настроим сервер xray-core для работы с REALITY.

Минимальная конфигурация Вашего сервера будет выглядеть так.
При желании, Вы можете настроить его исходя из ранее полученных знаний.

```
{
  "log": {
    "loglevel": "warning"
  },
  "inbounds": [
    {
      "listen": "0.0.0.0",
      "port": 443,
      "protocol": "vless",
      "settings": {
        "clients": [
          {
            "id": "dfccaec7-6bed-499b-af77-0275d55d573c",
            "flow": "xtls-rprx-vision"
          }
```

```
        ],
        "decryption": "none"
      },
      "streamSettings": {
        "network": "tcp",
        "security": "reality",
        "realitySettings": {
          "dest": "pupalupa.com",
          "serverNames": [
            "www.pupalupa.com",
            "pupalupa.com"
          ],
          "privateKey": "2KZ4uouMKgI8nR-LDJNP1_MHisCJOmKGj9jUjZLncVU",
          "shortIds": [
            "6ba85179e30d4fc2"
          ]
        }
      },
      "sniffing": {
        "enabled": true,
        "destOverride": [
          "http",
          "tls",
          "quic"
        ]
      }
    }
  ],
  "outbounds": [
    {
      "protocol": "freedom",
      "tag": "direct"
    },
    {
      "protocol": "blackhole",
      "tag": "block"
    }
  ]
}
```

Настройка клиента

Минимальная конфигурация Вашего клиента будет выглядеть так.
При желании, Вы можете настроить его исходя из ранее полученных знаний.

```
{
  "log": {
    "loglevel": "warning"
  },
  "routing": {
    "rules": [
      {
        "ip": [
          "geoip:private"
        ],
        "outboundTag": "direct"
      }
```

```
      ]
    },
    "inbounds": [
      {
        "listen": "127.0.0.1",
        "port": 10808,
        "protocol": "socks"
      },
      {
        "listen": "127.0.0.1",
        "port": 10809,
        "protocol": "http"
      }
    ],
    "outbounds": [
      {
        "protocol": "vless",
        "settings": {
          "vnext": [
            {
              "address": "",
              "port": 443,
              "users": [
                {
                  "id": "dfccaec7-6bed-499b-af77-0275d55d573c",
                  "encryption": "none",
                  "flow": "xtls-rprx-vision"
                }
              ]
            }
          ]
        },
        "streamSettings": {
          "network": "tcp",
          "security": "reality",
          "realitySettings": {
            "fingerprint": "chrome",
            "serverName": "www.pupalupa.com",
            "publicKey": "Z84J2IelR9ch3k8VtlVhhs5ycBUlXA7wHBWcBrjqnAw",
            "shortId": "6ba85179e30d4fc2"
          }
        },
        "tag": "proxy"
      },
      {
        "protocol": "freedom",
        "tag": "direct"
      }
    ]
  }
```

49 39 8 11 7

11 comments · 8 replies

Oldest | Newest | Top

egasvegas1109 on Jul 29, 2024

Спасибо за статью. У меня есть вопрос по поводу shortIds. Вы сообщили, что он используется для различения клиентов, но почему xray не использует uuid для этого? Также если имеется несколько клиентов в конфиге, то для каждого клиента нужно делать свой shortId или разрешается один на всех?

↑ 20 👍 7 0 replies

RPRX on Jul 29, 2024 Maintainer

希望俄罗斯有大牛给 REALITY 写个分析~~并翻译出中文版，我懒得写完小作文了~~

↑ 2 ❤️ 1 2 replies

Jacfger on Aug 14, 2024 edited

I have few questions regarding what domain should be used. So in the article the author said no large company site should be used. But the reality tutorial in the x-ray official GitHub is recommending the use of www.microsoft.com; and since I've tried using it in mainland and didn't encounter any problems, does that mean I'm already good to go, or should I still search for another domain and replace it?

Another related question is the author says if we have our own domain, just use that. But wouldn't it look more suspicious if someone just suddenly start connecting to a new small site randomly? (Even assuming it's just for personal use? Like for example me watching YouTube which uses quite some large bandwidth and sustainably?)

👍 2

sudoloveme on Sep 27, 2024

For the censor, it's not hard to see that your IP address is not from Microsoft of Google or Cloudflare and others IP range, so it's not the best way to configure yours Reality. Better to use selfsteal or noname website that doesn't have own IP range.

ilya-varivchenko on Aug 16, 2024

Отличная статья - большое спасибо! У меня возник вопрос : правильно ли я понимаю, что при заходе на IP моего VPN шлюза при включенном протоколе VLESS + VISION прохожий должен увидеть сайт, под который я маскируюсь? Что бы я ни делал - вижу отлуп браузера так как мой IP не совпадает с сертификатом который прилетает от того сервера, под который я маскируюсь. Что я не так делаю ?

↑ 3 👍 2 3 replies

iambabyninja on Aug 16, 2024 **Author** edited

> Отличная статья - большое спасибо! У меня возник вопрос : правильно ли я понимаю, что при заходе на IP моего VPN шлюза при включенном протоколе VLESS + VISION прохожий должен увидеть сайт, под который я маскируюсь? Что бы я ни делал - вижу отлуп браузера так как мой IP не совпадает с сертификатом который прилетает от того сервера, под который я маскируюсь. Что я не так делаю ?

Как было сказано выше, для режима прокси, клиент должен получить временный доверенный сертификат, подписанный временным ключом аутентификации, во всех иных случаях это будет просто port forward на dest.

Если Вы обращаетесь к айпи напрямую, сообщение TLS Client Hello, отправленное браузером, не содержит SNI, а хост в заголовке HTTP неверен.
Ответ зависит от dest, и с большой вероятностью Вы получите что то невразумительное.
Можете проверить через curl

```
curl -v --resolve www.microsoft.com:443:151.101.65.69 https://www.microsoft.com
```

вместо 151.101.xx.xx должен быть IP вашего сервера
вместо Microsoft.com - Ваш маскировочный сайт

👍 1

Ruhkukah on Aug 18, 2024 edited

Добрый день,

Большое спасибо за статью! У меня аналогичный вопрос. Попытался использовать несколько разных маскировочных вебсайтов, однако когда пытаюсь зайти на свой сервер из браузера, то получаю разные ошибки, обычно связанные с невалидным сертификатом - Сафари выдает, что не может установить безопасное соединение в одном случае, в другом - пишет что проблема с сертификатом. Хром пишет, что сайт uses an unsupported protocol и выдает ошибку ERR_SSL_VERSION_OR_CIPHER_MISMATCH). Однако проверка через curl во всех случаях выдает, что все хорошо.

Так как все-таки сделать так, что заходя из любого браузера на мой VPS загружался маскировочный сайт?

👍 1

zumm on Oct 10, 2024

> Добрый день,

> Большое спасибо за статью! У меня аналогичный вопрос. Попытался использовать несколько разных маскировочных вебсайтов, однако когда пытаюсь зайти на свой сервер из браузера, то получаю разные ошибки, обычно связанные с невалидным сертификатом - Сафари выдает, что не может установить безопасное соединение в одном случае, в другом - пишет что проблема с сертификатом. Хром пишет, что сайт uses an unsupported protocol и выдает ошибку ERR_SSL_VERSION_OR_CIPHER_MISMATCH). Однако проверка через curl во всех случаях выдает, что все хорошо.
>
> Так как все-таки сделать так, что заходя из любого браузера на мой VPS загружался маскировочный сайт?

Добавить соответствующую запись в hosts. :)

Jacfger on Aug 16, 2024

edited

Спасибо за отличный урок по использованию протокола Reality. Но у меня есть несколько вопросов относительно того, какой домен следует использовать. Итак, в статье вы сказали, что не следует использовать сайт крупной компании. Но в практическом руководстве в официальном репозитории Xray на GitHub рекомендуется использовать www.microsoft.com; и поскольку я пробовал использовать его в материковом Китае и не столкнулся с какими-либо проблемами (например, я могу пройти через gfw), означает ли это, что я уже готов к работе, или мне все равно следует искать другой домен и заменять его из-за из соображений безопасности?

Еще один связанный с этим вопрос: вы сказали, что если у нас есть собственный домен, просто используйте его. Но не будет ли это выглядеть более подозрительно, если кто-то вдруг начнет случайным образом подключаться к новому небольшому сайту? (Даже если предположить, что это только для личного использования? Как, например, я смотрю YouTube, который выглядит как устойчивое соединение с высокой пропускной способностью? Разве это не выглядит подозрительно?)

извините за мой русский, если мои слова не имеют смысла. поскольку я действительно хочу узнать об этом больше, но я не говорю по-русски.

↑ 27 👍 18

0 replies

air900 on Aug 29, 2024

Подробно и доступно

↑ 1

0 replies

Jolymmiles on Sep 23, 2024

up

1

0 replies

nichind on Oct 8, 2024

спасибо за труд!

1

0 replies

AndreyRz on Oct 12, 2024

Приветствую, а нет ли у Вас гайда по настройке vless с mkcp? Пробовал настраивать по гайдам из интернета, не подключается нормально ни к впс, ни даже к домашнему серверу в пределах локальной сети.

1

0 replies

gigiretu on Nov 5, 2024

у меня не создаётся ключ приватный на сервере marzban docker exec marzban-marzban-1 xray x25519 ввожу и ничего Error response from daemon: No such container: marzban-marzban-1

1

1 reply

MaxConsolas on Nov 17, 2024

```
docker ps
```

И смотри имя контейнера

MoskalenkoM on Nov 9, 2024

Спасибо за статью, в ней многие вопросы для конфигурации соединения и клиента были раскрыты. Можете подсказать, как ограничить один конфиг одним одновременным соединением, т.е, чтобы 2 пользователя с одним конфигом не могли подключиться одновременно.
Благодарю!

1

2 replies

egasvegas1109 on Nov 10, 2024

никак, это не реализовано в xray сейчас, но вроде как коммьюнити хотят это сделать. пока можно поставить fali2ban + limit ip

MoskalenkoM on Nov 11, 2024

Лимит по IP не вариант, на одном IP могут быть 2 разных конфига, или один конфиг может быть на разных IP.

devopg on Nov 17, 2024

Если какая известная проблема в REALITY что он путает сертификаты при большом потоке данных? #4022

↑ 1

0 replies

Category

Show and tell

Labels

None yet

16 participants