Федеральное государственное автономное образовательное учреждение высшего образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ»

Эссе по защите информации «Атаки протокольного туннелирования»

Выполнил: Дурнов Алексей Николаевич
студент Б01-009

Долгопрудный, 2023

1 Введение

2 Классификация систем обхода блокировок

В целях содействия свободе слова и беспрепятственному обмену информацией в Интернете, ученые предлагали различные решения для обхода блокировок и борьбы с цензурой на протяжении многих лет. Попытаем классифицировать их и перечислить их преимущества и недостатки.

1. **Системы на основе прокси-серверов**: К таким системам относятся прокси-сервера, VPN, Tor и т.д. В них клиенты передают свой трафик через промежуточные прокси-сервера, которые от имени клиентов подключаются к заблокированным сайтам. Подобный подход позволяет легко разворачивать такие системы, однако их также легко детектировать по публичным IP-адрессам прокси-серверов. Таким образом, противнику не составляет труда блокировать их сразу же после обнаружения.

2. **Системы ложной маршрутизации**: Достаточно перспективный подход, при котором клиент, находящийся под цензурой, должен подключаться к незаблокированным веб-сайтам. Эти запросы содержат скрытую информацию, которая позволяет специальным промежуточным маршрутизаторам (маршрутизаторам-обманкам) на пути следования перехватывать их и расшифровывать истинное цензурируемое направление, к которому клиент хочет получить доступ. Затем эти маршрутизаторы-обманки передают запросы и ответы между клиентами и заблокированными сайтами, при этом делая вид, что клиент подключен к незаблокируемому веб-сайту. Примерами таких систем являются Telex, Cirripede, Slitheen, Tapdance, Waterfall of Liberty, SiegeBreaker и др. Чтобы заблокировать такие системы, может потребоваться принятие сложных мер: изменение политики маршрутизации на уровне всей страны. Такие изменения маршрутизации на практике оказываются непомерно дорогими. Таким образом, противникам становится очень сложно их блокировать. Однако для функционирования таких систем требуется сотрудничество с провайдерами и, следовательно, они представляют собой препятствие для развертывания.

3. **Системы, основанные на мимикрии**: Такие системы пытаются замаскировать и передать цензурируемое содержимое под сообщением обычного протокола приложений. Например, SkypeMorph помогает получить доступ к заблокированным сайтам, имитируя протокол общения Skype. Однако такие протоколы очень легко блокируются, так как тяжело передать все особенности базового протокола. Кроме того, их эффективность зависит от скорости протоколов прикрытия. Для Skype выделяется низкие скорости передачи данных, что приводит к низкому уровню QoS для просмотра веб-страниц.

4. **Системы, основанные на туннелировании**: Они полагаются на инкапсуляцию скрытого трафика в сообщениях стандартных прикладных протоколов – например, электронную почту, VoIP, видеопотоки, онлайн-игры и т.д. В качестве примеров можно привести SWEET, Covertcast, Delta Shaper, Freewave, CloudTransport, Rook и др. Такие системы являются улучшением по сравнению с системами, основанные на мимикрии, поскольку они не имитируют протокольные сообщения, а непосредственно используют их в качестве скрытых каналов.

Они используют все особенности прикрывающего протокола, в то время как цензурируемое содержимое инкапсулируется в полезную нагрузку. Это очень сильно затрудняет цензору к их дифференциации от обычных (базовых) протокольных сообщений. Таким образом, может потребоваться блокировка всех базовых приложений (таких как электронная почта, облачные сервисы и т.д.), что может привести к масштабным убыткам. Однако такие системы всё также плохо обеспечивают QoS.

5. **Прочие системы**: Существуют и другие системы для борьбы с блокировками, не относящихся ни к одному из выше перечисленных типов. Domain fronting используют различные популярные облачные сервисы, например, Google app engine, для доступа к цензурируемому контенту. Запрос к прокси-серверу (скрытому за облачным сервером), скрывается в протоколе HTTPS, который направляется на доменное имя безобидного внешнего интерфейса облачного сервера. Этот модуль расшифровывает HTTPS-запрос и направляет его на прокси-сервер. Для блокирования таких сервисов противнику может потребоваться заблокировать весь фасадный сервис (например, Google app engine), тем самым тем самым блокируя другие сторонние приложения, использующие эту платформу. Разворачивать такие системы не является экономически эффективным.

3 Tor

С тех пор как в 2002 году Tor стал общедоступным и начал завоевывать популярность среди пользователей во всем мире как система для анонимного общения и обхода цензуры, многие страны пытались заблокировать подключение к нему своих граждан. Эти попытки начинались с простых методов, таких как внесение сайта Tor в черный список, чтобы пользователи не могли попасть на него и скачать клиентское ПО Tor, и со временем становились все более изощренными: активно загружался список узлов Tor (также называемых ретрансляторами) с Tor Directory Servers и вносился в черный список, устанавливались DPI-системы для поиска характеристик связи Tor (например, набор шифров рукопожатия TLS в Tor), а также активное зондирование (выдача себя за Tor-клиента и подключение к подозрительным серверам для проверки наличия у них Tor-релея). Tor сообщество не стояло на месте и разрабатывали методы обхода попыток блокировок, главным образом внедряя Tor Bridges и Pluggable Transports (PTs).

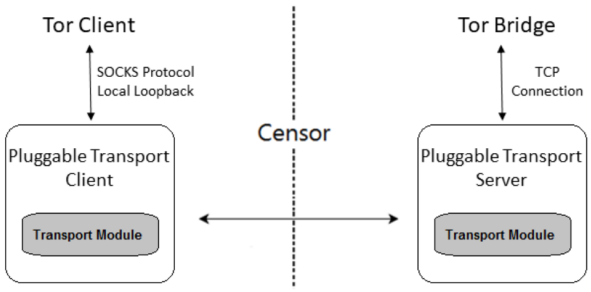

Рис. 1: Схема Pluggable Transports

PT представляют собой общую основу для разработки и развертывания технологий обхода блокировок. Их основная задача - обфусцировать соединение между клиентом Tor и мостом, служащим защитой входа в Tor, так, чтобы оно выглядело доброкачественным. PT состоит из двух частей, одна из которых устанавливается на стороне Tor-клиента, а другая - на стороне моста. PT открывает SOCKS-прокси для клиентского приложения Tor, обфусцирует или иным образом преобразует трафик, прежде чем направить его на мост. На стороне моста сервер PT выставляет обратный прокси, который принимает соединения от клиентов PT и декодирует обфускацию/трансформацию, примененную к трафику, перед передачей его реальному приложению моста. Данные, передаваемые между PT-клиентом и PT-сервером, могут быть зашифрованы, разрезаны на части общей длины или иным образом замаскированы, что делает их труднодоступными для цензора, который может обнаружить данные Tor и заблокировать их. Преобразование/обфускация данных и обратные операции выполняются транспортными модулями/протоколами обфускации, используемыми PT. По состоянию на сентябрь 2017 года, доступными и развернутыми протоколами обфускации в браузере Tor являются: obfs3, obfs4, ScrambleSuit, FTE и meek.

1. **Obfs3**: создает дополнительный уровень шифрования поверх TLS-соединения Tor, чтобы скрыть его уникальные характеристики. Для обмена ключами шифрования используется неаутентифицированный кастомизированное рукопожатие Диффи-Хеллмана. В результате этот протокол подвержен активным атакам на зондирование.

2. **ScrambleSuit**: защищает от активных зондирующих атак, используя для аутентификации внеканальный обмен секретами и номерами сеансов. ScrambleSuit также способен изменять свой сетевой отпечаток (распределение длины пакетов, время между приходами и т.д.). Этот протокол является предшественником Obfs4 и подвержен цензуре на основе белых списков.

3. **Obfs4**: имеет те же возможности, что и ScrambleSuit, но использует технику Elligator для обфускации открытых ключей и протокол ntor для односторонней аутентификации. В результате протокол работает быстрее, чем ScrambleSuit, и добавляет возможность мостовой аутентификации. Этот протокол также может быть заблокирован с помощью цензуры, основанной на белых списках.

4. **Meek**: использует технику, называемую Domain Fronting, для ретрансляции Tor-трафика на Tor-мост через сторонние серверы (т.е. CDN, такие как Amazon CloudFront и Microsoft Azure).

5. **Format-Transforming Encryption (FTE)**: преобразовывает трафик Tor в произвольные форматы стандартных протоколов, используя их языковые описания.

Эти протоколы обфускации можно разделить на две группы:

1. **Протоколы случайных потоков (obfs3, obfs4 и ScrambleSuit)**. Обмен данными в этих протоколах осуществляется в виде потоков случайных байтов, которые невозможно соотнести ни с одним известным протоколом. Эти протоколы состоят из двух фаз: фазы рукопожатия, на которой обе участвующие стороны надежно обмениваются ключами и/или номерами, и фазы коммуникации, состоящей в обмене зашифрованными сообщениями с использованием установленных ключей.

2. **Структурированные потоковые протоколы (FTE и meek)**, которые пытаются имитировать известные протоколы из "белого списка", такие как HTTP.

Цензурирующие страны, обладающие возможностями активного зондирования, могут выявить мосты, взаимодействующие с использованием obfs3. Кроме того, протоколы случайных потоков относятся к категории "похожих на ничто", что означает, что они могут быть идентифицированы и заблокированы цензором с помощью стратегии "белого списка", поскольку их отпечаток (включая рукопожатие) не соответствует ни одному известному протоколу. С другой стороны, структурированные потоковые протоколы, имитирующие широко распространенные протоколы, устойчивы к блокированию на основе "белых списков". Однако они не защищают от активного зондирования.

4 FTE

Часть DPI систем используют явно или неявно регулярные выражения для классификации протоколов на прикладном уровне, поэтому рассмотрим механизмы, позволяющие злоумышленнику принудительно идентифицировать протокол, против любого DPI, основанного на проверке регулярных выражений. Основная идея заключается в том, чтобы встроить защиту от DPI в схемы шифрования. Добавим к обычному интерфейсу шифрования возможность принимать на вход регулярное выражение. Задача этого регулярного выражения определять формат шифротекстов: это означает, что шифротексты, взятые в виде строк соответствующего алфавита, гарантированно будут соответствовать заданному регулярному выражению. Такой криптографический примитив называется форматно-преобразующим шифрованием. Благодаря правильному выбору регулярного выражения, шифротексты, которые на самом деле несут исходный протокол, будут классифицироваться DPI как сообщения от другого протокола, по нашему выбору.

Схема FTE состоит из трех алгоритмов: генерация ключа, шифрования и дешифрования. Генерация ключа работает, как и в обычном шифровании, с выдачей случайно ывбранного симметричного ключа K. Шифрование Enc принимает ключ K, формат F и сообщение M. Оно может быть случайным, полным или детерминированным и всегда выдаёт шифротекст C или символ ошибки $\perp$. Расшифрование Dec принимает ключ K, формат F и шифротекст C. Его выходом является сообщение или символ ошибки $\perp$. Формат F задаёт множество $L(F)$, называемое языком F. Требование заключается в том, что любой шифротекст , выводимый Enc, должен быть элементом $L(F)$.

FTE является схожим с шифрованием сохраняющий формат (FPE), впервые формализированного Bellare, Ristenpart, Rogaway и Stegers (BRRS). В нём также используются форматы, но при этом требуется, чтобы и открытый текст, и шифротекст были элементами одного и того языка, определенного формата.

Хочется, чтобы FTE могло поддерживать форматы, описываемые регулярными выражениями. Это легко позволит "программировать" форматы и наделит FTE теми выразительными возможностями, что и DPI системы, основанные на регулярных выражениях.

Реализация $Enc(K, F, M)$ для регулярного выражения F:

1. Шифруем сообщение M с помощью стандартной схемы аутентифицированного шифрования, получая промежуточный шифротекст Y.

2. Рассматриваем Y как число в $\mathbb{Z}_{|L(F)|}$ (в кольце класса вычетов по модулю размера языка).

3. Применяем кодирующию функцию unrank: $\mathbb{Z}_{|L(F)|} \to L(F)$.

Для возможности расшифровки требуем, чтобы unrank была биективной функцией с эффективно вычислимой обратной функцией rank: $L(F) \to \mathbb{Z}_{|L(F)|}$. Ключевой алгоритмической проблемой является реализация функции rank и unrank эффективно. Они связывают с каждой строкой языка её ранг, т.е. позицию в общем упорядченном языке. Goldberg и Sipser предложили эффективный способ ранжирования регулярного языка, когда этот язык представлен детерминированным конечным автоматом (DFA). BRRS использовали его для нереализованной схемы FPE для произвольных регулярных языков, закодированных как DFA, но они также подчеркивают, что, асимптотически говоря, доказательно не существует способа дать эффективные функции rank и unrank, начиная с регулярных выражений. Существует стандартные средства для преобразования регулярного выражения в недетерминированный конечный автомат (NFA), а затем в DFA, но второй шаг приводит к экспоненциальному раздуванию размера состояния. Такое поведение в худшем случае не является проблемой для FTE, отчасти потому, что типы регулярных выражений, используемых в DPI, сами по себе явно разработаны для того, чтобы избежать раздувания в худшем случае. Предложенная реализация использует компромисс между временем и памятью, предложенный Goldberg и Sipser, для поддержки более эффективной работы во время выполнения программы путем предварительного вычисления таблиц, позволяющих (не) ранжировать все строки $x \in L$, где $|x| \leq n$. Сложность таких предварительной обработки составляет $O(n \cdot |\Sigma| \cdot |Q|)$, где Σ - базовый алфавит, а Q - множество состояний DFA, реализующего регулярное выражение FTE. С учетом эти таблиц, сложность функций $rank_L$ и $unrank_L$ состовляет $\Omega(n)$ и $O(n \cdot |\Sigma|)$, где n - длина выхода $rank_L$ и входа $unrank_L$ соответственно. Подход "Encrypt-then-Unrank" сохраняет конфиденциальность сообщений и аутентичность базовой схемы аутентифицированного шифрования.

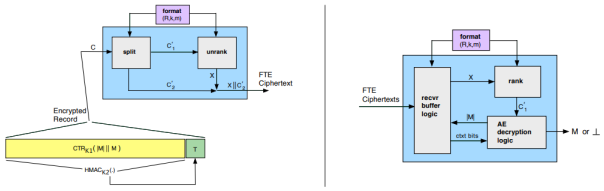

Рис. 2: Схема шифрования и дешифрования

Уровень записи FTE Для преобразования произвольных TCP-потоков необходимы дополнительные механизмы "уровень записи позволяющие буферизировать, кодировать, безошибочно разбирать и декодировать потоки сообщений FTE и наоборот.

Предполагается, что отправитель и получатель используют заранее установленный набор ключей, возможно, полученных из одного общего мастер-ключа. Алгоритмы приведены в конце оригинальной статьи. Реализация форматов FTE Будет полезно для формата указывать не только регулярное выражение, поэтому формат F для уровня записи представляет собой набор (R, k, m): регулярное выражение R, число k $>$ 0, определяющее длину используемых строк из языка L(R), целое число m 0, определяющее количество бит неформатированного текста, добавляемых в

конец FTE-кодированного сообщения. Причина использования строк определенной длины заключается в том, что она является удобным способом представления лего разбираемых шифротекстов FTE, а значение ь используется для того, чтобы дешево обеспечить большую емкость в случае, когда искомым языком является фактически является любая строка с префиксом в L(R). Таким образом, для F = (R, k, m) язык имеет вид L(F) = (L(R) k) || m, где - алфавит регулярного выражения. Для простоты будем полагать, что = 0, 1, но в реализациях обычно используются более мощные алфавиты. По умолчанию используются значения m = 218 и k = 210. Отправитель FTE Рассмотрим формат F = (R, k, M) Пусть Lk = L(R) 0, 1k - k-битный фрагмент L(R), который будем использовать, а t = floor(log2 (|Lk(R)|)) - емкость фрагмента, т.е. количество бит, которое можно закодировать с помощью этого фрагмента. Первое действие это подготовка зашифрованной записи с помощью ключа K: Из буфера открытых сообщений берется открытое сообщение M длиной не более |M| < m. Затем формируется открытая текстовая запись, содержащая кодировку |M|, за которой следует M. Эта запись шифруется с помощью стандартной схемы аутентинфицированного шифрования (AE) с ключом K и получается шифротекст C. Предполагается, что |C| определяется только |M|, что справедливо для AE, использующихся на практике. Дополняем M, если требуется, чтобы гарантировать |C| t (условие криптостойкости по Шенону). В предложенной реализации используется режим CTR для AES и аутентифицирует полученнный шифротекст с помощью HMAC-SHA256. Зашифрованная запись C передаётся модулю split и добавляется в его внутренний буфер. Задача split - создать две строки: одну передать в unrank для форматирования, а другую передать как есть. В частности, модуль split забирает до t + m бит (и не менее t бит) с передней части буффера. Эту часть обозначим как C'. Заметим, что C' может быть полным шифротекстом AE, частью одного или включить биты из нескольких шифротекстов. Затем разделяет C' на части C'1 и C'2, где |C'1| = t и |C'2| m. Первая часть C'1 отдаётся функции unrank, которая выдаёт форматированную строку X из Lk(R). И наконец, конкатенация X || C'2 становится шифротекстом FTE отправителя, который может быть передан. Получатель FTE Одной из главных проблем здесь является отсутствие явных маркеров для разграничения границ шифротекстов FTE. Для решения этой проблемы получатель использует тот факт, что отправитель использовал фиксированный фрагмент Lk(R). Как только в его буфере появляется первые k бит шифротекста FTE, он рассматривает их как строку X из Lk(R) и применяет функцию rank(X) для восстановления C'1. (Или выдаётся ошибка, если X не из Lk(R)). Затем C'1 передаётся в алгоритм расшифровки AE для получения l = |M| и, возможно, некоторых начальных бит сообщения M. (Эти последние биты сообщения ещё не должны быть переданы на более высокие уровни). Отметим, что схема AE должна быть способна выполнять инкрементное дешифрование и она не должна быть уязвима для атак, которые злоупотребляют использованием поля длины до обеспечения целостности. Режим CTR и HMAC обеспечивают эти свойства. Учитывая значение l, приемник теперь знает, сколько ещё бит шифротекста AE ожидается. Отсюда он может удалить из входного буфера до m бит шифротекста, а затем вернуться в состояние в котором он обрабатывает последующие k-бит в буфере в виде строки Lk(R), применяя ранг и т.д. При получении полного шифротекста AE завешает дешифрование, проверяет целостность шифротекста и только теперь освобождает буферизированные биты сообщения вплоть до приложения. Согласование регулярных выражений Уникальной особенностью FTE является возможность быстрого изменения регулярных выражений "на лету". Хотелось бы иметь возможность согласовывать формат, но по-

скольку в сети все данные, передаваемые по сети, должны быть отформатированы для прохождения DPI-проверки, согласование регулярных выражений в открытом виде невозможно. Предлагается решать эту проблема на момент установления tcp соединения, считая, что сервер и клиент поддерживают начальный большой набор возможных регулярных выражений. Т.е. считается, что у клиента сервера есть общий упорядоченный список FTE форматов (F1, F2, ..., Fn), а также что они согласовали свои криптографические ключи вне канала связи. Для каждого TCP-соединения клиент определяет FTE формат Fi, который он хочет использовать для клиент-сервер сообщений, и FTE формат Fj, который он хочет использовать для сервер-клиент сообщений. Затем клиент собирает сообщение M $\leftarrow$ i || j, шифрует его, как особый тип сообщений для согласования, используя значение первого байта открытого текста (зарезервирован), кодирует его с помощью Fi, и отправляет на сервер. Когда сервер принимает сообщение от клиента, он пытается перебрать список форматов, пытаясь расшифровать сообщение. После успешной расшифровки он вычисляет необходимое форматы и использует Fi для клиент-сервер сообщений и Fj для сервер-клиент сообщений. Сервер завершает сеанс, отвечая клиенту сообщением с типом завершения согласования. Мы можем усовершенствовать обычную реализацию на стороне приемника за счёт наличия процедуры rank, которая содержит специальные проверки на быстрое завершение разбора, когда строка была отранжирована, но не принята конечным автоматом. Это позволяет автомату быстро исключить определенные автоматы, и тем самым обеспечить поддержку десятков автоматов.

5 DNS-Morph

6 Заключение

Список литературы

[1] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, Till Stegers. Format-Preserving Encryption.

[2] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, Thomas Shrimpton. Protocol Misidentification Made Easy.

[3] Rami Ailabouni, Orr Dunkelman, Sara Bitan. DNS-Morph UDP-Based Bootstrapping Protocol For Tor.

[4] Piyush Sharma, Devashish Gosain. Camoufler Accessing The Censored Web By Utilizing Instant Messaging.

[5] Zhonghang Sui, Hui Shu, Fei Kang, Yuyao Huang, Guoyu Huo. A Comprehensive Review of Tunnel Detection on Multilayer Protocols.