Формализованная модель сетевой атаки SSL renegotiation

Садырин Д. С., студент кафедры БИТ, Университет ИТМО
dsadyrin@ptsecurity.com

Аннотация

Сетевые атаки являются актуальными и часто используемыми в наши дни. В данной работе производится сравнение существующих способов описания сетевых атак и производится рассмотрение нового способа описания на примере атаки SSL Renegotiation.

Введение

При рассмотрении проблем уязвимостей к атакам в компьютерных сетях важно определить сам термин «сетевая атака». На основании терминологии ГОСТ под сетевой атакой следует понимать компьютерную атаку с использованием протоколов межсетевого взаимодействия [1].

"Медленные" атаки являются разновидностью DOS атак, вызывающих нагрузку на сервере цели за счет одновременно открытых легитимных соединений или сложных процессорных вычислений, что в итоге может привести к отказу в обслуживании. Детектирование такого рода сложно само по себе за счет неоднородности нагрузки во времени. Хотя на данный момент существуют различные методы обнаружения и защиты от таких атак на уровне сервера-жертвы или сервера-посредника, на уровне сетевых магистралей их обнаружить очень сложно. Растянутый во времени вредоносный трафик теряется в общем потоке проходящего трафика большого объема.

Исследование возможностей детектирования медленных сетевых атак на уровне таких магистралей является актуальной и мало раскрытой в открытых источниках темой.

Формальное описание сетевых атак

В данной работе рассматривается способ формального описания сетевых атак. В качестве примера взята атака SSL renegotiation, она позволяет осуществить атаку класса «отказ в обслуживании». Суть атаки

заключается в том, что процесс установки защищенного соединения расходует во много раз больше ресурсов на серверной части, чем на клиентской. Это позволяет злоумышленнику, отправив множество запросов, потребить ресурсы сервера.

Существуют различные подходы к моделированию атак с учетом различных классов атак, такие как: основанные на графах, байесовские сети, сети Петри.

Рассмотрим модели атак, основанные на графах. Под графом атак понимается граф, содержащий все известные траектории реализации нарушителем угроз. Ключевой проблемой построения графа атак для больших сетей является масштабируемость, связанная с формированием графа атак для сетей с большим числом хостов и уязвимостей.

Примером моделирования атак с помощью байесовских сетей являются байесовские графы атак, которые представляют собой направленные ациклические графы, где вершины ассоциируются с инцидентами, рассматриваемыми также как элементарные условия, а ребра моделируют конъюнкцию или дизъюнкцию элементарных условий. Преимущества и недостатки байесовских графов атак такие же, как у деревьев атак. В отличие от деревьев атак байесовские графы атак имеют дополнительные преимущества, так как они представляют собой вероятностные модели, которые позволяют учитывать случаи неопределенности исходных данных о моделируемых атаках. Также среди разновидностей байесовских сетей следует выделить скрытые марковские модели, которые используются при моделировании атак из-за удобства исследования путей в пространстве состояний, каждое из которых характеризуется заданной вероятностью.

Сети Петри являются одним из широко используемых способов формального описания атак. Наиболее часто используются раскрашенные сети Петри, стохастические сети Петри и нечеткие сети Петри.

Одним из распространенных подходов к моделированию атак с помощью графов является подход с использованием деревьев атак. Дерево атак – некоторое представление сценария атаки в виде направленного дерева, его корень - успешная атака, атака идет от листьев (исходные события) снизу вверх по узлам (события). В узлах могут находиться логические функции, по исходу которых атака может продолжиться, перейдя выше по дереву. Модели, основанные на деревьях атак, имеют

следующие преимущества: наглядность, масштабируемость, универсальность. К недостаткам моделей, основанных на деревьях атак, следует отнести трудности моделирования циклических атак и отсутствие возможностей динамического моделирования [2].

Множество перечисленных подходов затрудняют процесс перевода формального описания в эксперимент.

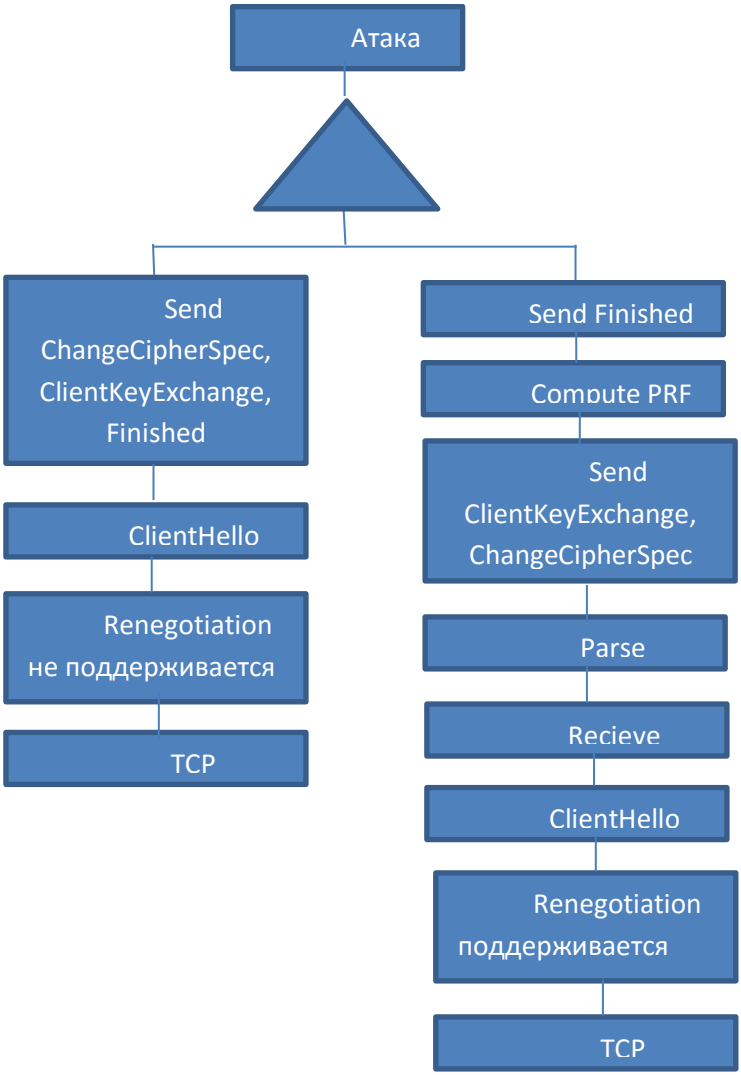
Цель работы

Целью работы является создание эффективного формального описания атаки, которое возможно легко автоматически воспроизвести на практике в ходе эксперимента в рамках текущей сетевой обстановки. В данном исследовании формальное описание состоит из нескольких составляющих:

- описание на естественном языке
- математическая модель
- псевдокод
- описание на языке NASL
- дерево сценария атаки.

Конечное описание задается в виде конфигурационного файла, содержащем в себе логику атаки, её параметры и возможный диапазон значений. Далее описание подается на вход симулятору атак, результатом которого является массив данных и их графическое представление. Используя полученные данные, производится исследование атаки и определение признаков для её детектирования.

Дерево атаки



Client Hello – обмен параметрами SSL-соединения со стороны клиента

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites;
    CompressionMethod compression_methods;
    Extension extensions;
} ClientHello;
```

Server Hello – обмен параметрами SSL-соединения со стороны сервера

```
struct {
    ProtocolVersion server_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
} ServerHello;
```

Key Exchange – процедура обмена ключами

Как видно из дерева атаки, возможно два сценария в зависимости от наличия опции renegotiation у SSL-соединения на сервере. Если опция включена, атакующий может отправлять специально сформированные данные и в одном TCP-соединении, тем самым уменьшается время на повторное соединение. При выключенной опции renegotiation, атакующему необходимо каждый раз заново подключаться к серверу и отправлять случайные данные, не дожидаясь его ответа.

Математическая модель атаки

Реализация алгоритма RSA в библиотеке SSL использует алгоритм быстрого возведения в степень. Алгоритм основан на том, что для возведения числа *x* в степень *n* не обязательно перемножать число *x* на само себя *n* раз, а можно перемножать уже вычисленные степени. Так, например, если $n = 2^k$ степень двойки, то для возведения в степень *n*

достаточно число возвести в квадрат k раз, затратив при этом k умножений вместо 2^k.

В этом случае в среднем требуется $\frac{1}{2} \cdot \ln n$ операций умножения.[3]

Также для ускорения дешифрования используется китайская теорема об остатках. Её применение сокращает количество операций. Она основана на том факте, что числа p и q в разложении $N = pq$ известны владельцу закрытого ключа, и можно вычислить[4]:

$$m_p = C^d \mod p = C^{d \mod p-1} \mod p,$$
$$m_q = C^d \mod q = C^{d \mod q-1} \mod q.$$

Выведем зависимость суммарного времени атаки от длины ключа при использовании алгоритма RSA.

$$p \approx q \approx \sqrt{N}$$
$$T = \left(\frac{1}{2}\log_2 p + \frac{1}{2}\log_2 q\right) * C = \left(\frac{n}{4} + \frac{n}{4}\right) * C = \frac{n * C}{2}$$

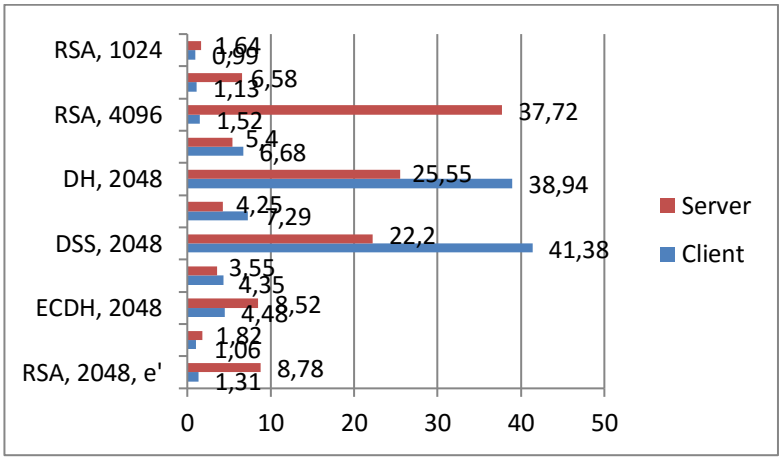
T – суммарное время атаки

n – число бит в ключе

C – число запросов к серверу

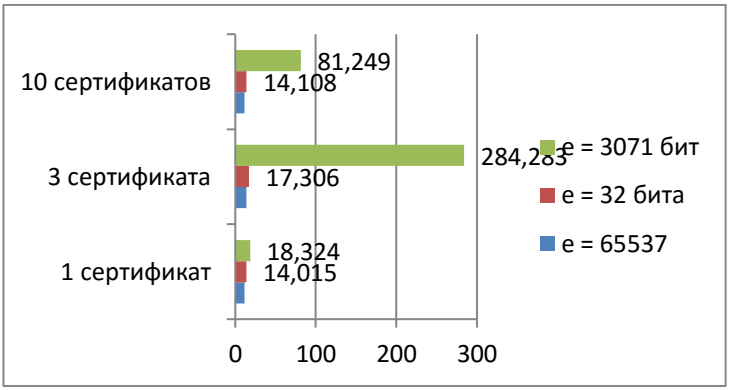
Параметры атаки

Также, среди параметров атаки нужно выделить тип алгоритма шифрования и длину ключа, используемые в соединении. Была исследована зависимость процессорного времени, потраченного клиентской и серверной частью в зависимости от алгоритмов. В ходе эксперимента измерялось число секунд для установления защищенного соединения при выполнении 1000 итераций атаки. Результаты отражены на диаграмме.



Также была исследована зависимость процессорного времени сервера при проведении атаки от длины цепочки сертификатов и размера публичной экспоненты в алгоритме RSA. В каждом сертификате из данной цепочки использовался ключ заданной длины. Измерения проводились для 1000 итераций атаки.

Длина цепочки сертификата	e (65537)	e (32 бита)	e (3071 бит)
1 сертификат	11,784	14,108	81,249
3 сертификата	14,015	17,306	284,283
10 сертификатов	18,324	25,374	693,216



Приведем пример описания атаки конфигурационным файлом для симулятора:

```
{
  "name": "ssl renegotiation attack",
  "protocol": "tcp",
  "port": "443",
  "threads":"10",
  "delay": "2000",
  "bitrate": 256,
  "cipher" : "RSA",
  "keylen" : 2048,
  "cert_cain_length": 1
}
```

В нем содержится основная необходимая информация для проведения симуляции атаки.

- name – название атаки
- protocol – тип протокола
- port – номер порта
- threads – число атак в единицу времени
- delay – задержка после запроса
- bitrate – скорость передачи данных

- cipher – используемый алгоритм шифрования
- keylen – длина ключа
- cert_cain_length – число сертификатов в цепочке

Заключение

Формализованное описание и конфигурационный файл атаки может быть использован как входные данные симулятору для проведение атаки. Результатом симулятора является массив данных и их графическое представление. Используя полученные данные, производится исследование атаки и определение признаков для её детектирования.

Литература

1. ГОСТ Р 51275-2006 [Электронный ресурс] РОССТАНДАРТ Федеральное агентство по техническому регулированию и метрологи URL: http://protect.gost.ru/v.aspx?control=8&baseC=-1&page=0&month=-1&year=-1&search=&RegNum=1&DocOnPageCount=15&id=121176&pageK=CF3CD2AE-2D8F-4721-8EAB-4C278DF101C5 (дата обращения 10.03.2016)
2. *Д.И. Котенко, И.В. Котенко, И.Б. Саенко* Методы и средства моделирования атак в больших компьютерных сетях: состояние проблемы // Труды СПИИРАН. 2012. Вып. 3(22). ISSN 2078-9181 (печ.)
3. Алгоритмы быстрого возведения в степень [Электронный ресурс] Wikipedia, the free encyclopedia – URL: https://en.wikipedia.org/wiki/Exponentiation_by_squaring (дата обращения 10.03.2016)
4. Использование китайской теоремы об остатках для ускорения расшифрования [Электронный ресурс] Wikipedia, the free encyclopedia – URL: https://ru.wikipedia.org/wiki/RSA#.D0.98.D1.81.D0.BF.D0.BE.D0.BB.D1.8C.D0.B7.D0.BE.D0.B2.D0.B0.D0.BD.D0.B8.D0.B5_.D0.BA.D0.B8.D1.82.D0.B0.D0.B9.D1.81.D0.BA.D0.BE.D0.B9_.D1.82.D0.B5.D0.BE.D1.80.D0.B5.D0.BC.D1.8B_.D0.BE.D0.B1_.D0.BE.D1.81.D1.82.D0.B0.D1.82.D0.BA.D0.B0.D1.85_.D0.B4.D0.BB.D1.8F_.D1.83.D1.81.D0.BA.D0.BE.D1.80.D0.B5.D0.BD.D0.B8.D1.8F_.D1.80.D0.B0.D1.81.D1.88.D0.B8.D1.84.D1.80.D0.BE.D0.B2.D0.B0.D0.BD.D0.B8.D1.8F (дата обращения 10.03.2016)