УДК 004.056.5:004.75

ОБ ОДНОМ ИЗ МЕТОДОВ АТАКИ НА ПРОТОКОЛ TLS

Волков В.А.
Харьковский национальный университет радиоэлектроники

Главной угрозой при передаче данных между клиентом и сервером является возможность несанкционированного доступа к конфиденциальной информации. Для противодействия этой угрозе используется протокол TLS. Однако данный протокол не обладает высокой степенью безопасности. В ходе работы был проведен анализ основных атак на протокол TLS и указаны методы противодействия этим атакам. Также был разработан метод перехвата и расшифровки трафика, передаваемого по протоколу HTTPS. Предложенный метод можно применять даже при удаленном прослушивании сети, а также его можно автоматизировать. Дальнейшая автоматизация разработанного метода позволит расшифровывать передаваемые данные в режиме приближенному к реальному времени.
Ключевые слова: протокол TLS, протокол HTTPS, уязвимость, атаки, шифрование, расшифровывание.

Постановка проблемы. В настоящее время существует проблема безопасной передачи пользовательских данных. Каждый человек знает об интернете и о современных инструментах работы с ним, от электронной почты до сервисов обмена сообщениями, но мало кто понимает, как они работают. Когда браузер делает запрос к любимому веб-сайту пользователя, этот запрос должен пройти через множество различных сетей, любая из которых может быть потенциально использована для прослушивания или для вмешательства в установленное соединение. Огромное количество организаций ретранслирует данные с собственного компьютера пользователя на другие компьютеры его локальной сети, через роутеры и свитчи, через провайдера и через множество других промежуточных провайдеров. Если злоумышленник окажется, хотя бы в одной из точек прохождения информации, у него есть возможность посмотреть, какие данные передаются. Как правило, запросы передаются посредством обычного HTTP (Hyper Text Transfer Protocol), в котором и запрос клиента, и ответ сервера передаются в открытом виде. Существует множество причин тому, что HTTP не использует шифрование по умолчанию:

- для шифрования требуется больше вычислительных мощностей;
- передается большее количество данных;
- нельзя использовать кеширование.

Разработчики стандартов либо игнорируют уязвимости в виду их сложной реализации, либо разрабатывают механизм противодействия, который отлично себя показывает, пока не появляется новая уязвимость. В некоторых случаях, когда по каналу связи передается исключительно важная информация, такая как пароли или данные кредитных карт, необходимо обеспечить дополнительные меры, предотвращающие прослушивание таких соединений. Механизмы, поддерживающие данные требования, должны быть защищены от уязвимостей и иметь высокую степень безопасности.

Анализ последних исследований и публикаций. Для передачи конфиденциальных данных протокол HTTP использует шифрование, которое заложено в работу протокола TLS (Transport Layer Security). Однако протокол не обладает высокой степенью безопасности, как это считалось раньше. Уязвимость протокола TLS 1.0, которая считалась теоретической, была продемонстрирована на конференции Ekoparty в сентябре 2011 года. Демонстрация включала в себя дешифрование cookies, использованных для аутентификации пользователя [6]. Ранее Сержем Воденеем

на Eurocrypt 2002 [5] была представлена работа, вкоторой описывался другой метод построения атаки на TLS, также использующий особенности режима CBC. Уязвимость в фазе возобновления соединения, обнаруженная в августе 2009 года, позволяла криптоаналитику, способному взломать https-соединение, добавлять собственные запросы в сообщения, отправленные от клиента к серверу [2]. Также существуют варианты атак, основанные непосредственно на программной реализации протокола, а не на его алгоритме [1].

Выделение не решённых ранее частей общей проблемы. Раньше, проводя анализ TLS-трафика, атакующему приходилось сталкиваться с несколькими проблемами. Суть одной из таких проблем была в том, что для расшифровки более ранних версий протокола, можно было указать программе Wireshark приватные ключи, если они имелись. Эта функциональность утратила свою работоспособность из-за того, что в новых версиях протокола начали продвигать совершенную прямую секретность (Perfect Forward Secrecy) [4], и приватного ключа стало недостаточно, чтобы получить сессионный ключ, который используется для зашифровки и расшифровки данных. Вторая проблема заключается в том, что приватный ключ стали получать с использованием протокола Диффи-Хеллмана, который имеет высокую криптостойкость, к тому же приватный ключ невозможно напрямую выгрузить с клиента, сервера или HSM (Hardware Security Module), в котором тот находится. Из-за таких проблем приходилось прибегать к действиям с сомнительной целесообразностью связанным с расшифровкой трафика через атаку man-in-the-middle (например, через утилиту sslstrip).

Целью данной работы является проведение анализа существующих атак на протокол TLS и разработка метода расшифровки трафика HTTPS, который способен изменить отношение к проведению атак на защищённый трафик, либо помочь работе администраторов корпоративной сети.

1. Описание протокола TLS в HTTPS.

TLS (Transport Layer Security) – наследник SSL (Secure Sockets Layer) – это протокол транспортного уровня, наиболее часто применяемый для обеспечения безопасного соединения HTTPS. TLS расположен ниже протокола HTTP в модели OSI. Это означает, что в процессе выполнения запроса сперва происходят действия, связанные с TLS-соединением, и уже потом, все, что связано с HTTP-соединением.TLS использует несколько криптографических подходов [3]:

1) Асиметричное шифрование (криптосистема с открытым ключом) для генерации общего секретного ключа и аутентификации.

2) Симметричное шифрование, использующее секретный ключ для дальнейшего шифрования запросов и ответов.

На данный момент в протоколе TLS для обмена ключами и проверки их подлинности применяются комбинации алгоритмов: RSA (асимметричный шифр), Diffie-Hellman (безопасный обмен ключами), DSA (алгоритм цифровой подписи), ECDSA. Для симметричного шифрования: RC4, IDEA, Triple DES, SEED, Camellia или AES. Для хеш-функций: MD5, SHA, SHA-256/384.

Алгоритмы могут дополняться в зависимости от версии протокола. До последней версии протокола TLS 1.2 были доступны также следующие алгоритмы симметричного шифрования, но они были убраны как небезопасные: RC2, IDEA, DES.

Кроме этого в актуальной реализации TLS имеется множество мер безопасности:

1) защита от понижения версии протокола к предыдущей версии или менее надёжному алгоритму шифрования;

2) нумерация последовательных записей приложения и использование порядкового номера в коде аутентификации сообщения (MAC);

3) использование ключа в идентификаторе сообщения (только владелец ключа может проверить код аутентификации сообщения). Хеш-код идентификации сообщений (HMAC), используемый в большинстве шифров из набора шифров TLS, был определён в RFC 2104;

4) сообщение, которым заканчивается подтверждение связи («Finished»), содержит в себе хэш всех сообщений, которыми обменялись стороны в процессе подтверждения связи;

5) псевдослучайная функция разбивает входные данные на две части и обрабатывает каждую разной хэш-функцией (MD5 и SHA-1), а затем вычисляет XOR от двух полученных свёрток, чтобы создать код аутентификации сообщения. Это обеспечивает безопасность даже в случае уязвимости одной из хэш-функций.

2. Существующие основные атаки на протокол TLS.

На данный момент основными являются атаки, применимые к протоколу TLS на фазе передачи данных. Все они, так или иначе, основаны на предложениях Барда и Воденея и возможны в моделях нарушителя, предполагающих возможность проведения злоумышленником активных действий, таких, как навязывание блоков открытого текста или шифротекста [2].

Данные типы атак используют следующие три проблемных аспекта:

1) потенциальная возможность чтения при повторном использовании синхропосылки;

2) уязвимость к навязыванию шифротекста;

3) некорректное использование алгоритмов проверки паддинга и алгоритмов проверки имитовставки.

Однако, в настоящее время, к выше упомянутым типам атак уже существуют стойкие сюиты протокола TLS 1.2, использующие шифрование с аутентификацией в режиме GCM. Реализовано это с помощью следующих решений.

Аутентификация сообщений вместе с паддингом. Каждое принимаемое сообщение обязано проходить проверку аутентичности с использованием режима выработки имитовставки блокового шифра, что делает невозможным навязывание сообщений.

Случайный выбор синхропосылки. Благодаря такому решению становятся невозможными любые атаки, использующие уязвимости, связанные с повторным использованием синхропосылок, либо с использованием в качестве синхропосылок ранее присутствующих в канале блоков шифротекста.

Отказ от использования паддинга. Использование режима CNT (гаммирования) не требует использование паддинга, что в свою очередь делает невозможным любые атаки, использующие избыточность вносимую паддингом.

Однако, кроме представленных типов атак существуют и другие, основанные непосредственно на программной реализации протокола, а не на его алгоритме [1]. В качестве примера можно привести уязвимость в фазе возобновления соединения, обнаруженную в августе 2009 года. Она позволяла криптоаналитику, способному взломать https-соединение, добавлять собственные запросы в сообщения, отправленные от клиента к серверу [2]. Так как криптоаналитик не может дешифровать переписку сервера и клиента, этот тип атаки отличается от стандартной атаки, типа человек посередине. В случае, если пользователь не обращает внимания на индикацию браузера о том, что сессия является безопасной (обычно значок замка), уязвимость может быть использована для атаки типа человек посередине [7]. Для устранения этой уязвимости было предложено как на стороне клиента, так и на стороне сервера добавлять информацию о предыдущем соединении и осуществлять проверку при возобновлении соединения. Это было представлено в стандарте RFC 5746, а также реализовано в последних версиях OpenSSL [3] и других библиотеках [3-4].

Описанный же подход в данной статье представляет собой способ расшифровки TLS-трафика, который предполагает у злоумышленника наличие доступа к компьютеру или сети, либо же злоумышленник смог занести на компьютер жертвы закладку, которая сможет собрать данные о сессиях. Такие условия нужны для формирования файла сессионных ключей, который будет использован вместе с соответствующим перехваченным трафиком. Перехватить трафик жертвы можно, находясь в любом участке сети на промежутке между сервером и объектом нападения.

3. Описание реализации.

Ниже приведено описание реализации предложенного метода к расшифровке TLS-трафика. В реализации использовался Wireshark – всем известный анализатор трафика, который помогает провести анализ работы сети, диагностировать проблемы, а также имеет много других полезных возможностей.

Ранее, используя анализаторы трафика для расшифровки TLS-трафика, приходилось сталкиваться с несколькими проблемами. Одной из таких проблем была невозможность легко проанализировать зашифрованный трафик, вроде TLS. Также для расшифровки более ранних версий протокола, можно было указать программе Wireshark приватные ключи, если они имелись, и расшифровывать трафик на лету, но это работало только в том случае, если использовался исключительно алгоритм RSA. Эта функциональность перестала эффективно работать из-за того, что в новых версиях протокола начали продвигать совершенную прямую секретность (Perfect Forward Secrecy) [4], и приватного ключа стало недостаточно, чтобы получить сессионный ключ, который используется для зашифровки и расшифровки данных. Вторая проблема заключается в том, что приватный ключ стали получать с использованием протокола Диффи-Хеллмана, который имеет высокую криптостойкость, к тому же приватный ключ невозможно напрямую выгрузить с клиента, сервера или HSM (Hardware Security Module), в котором тот находится. Из-за таких проблем приходилось прибегать к действиям связанным с расшифровкой трафика через атаку man-in-the-middle (например, через утилиту sslstrip), которая обладает сомнительной целесообразностью.

tlslogfile.log

```
1  # SSL/TLS secrets log file, generated by NSS
2  CLIENT_RANDOM d55d11fbcf5768e47cb3e9cd09c5e6d66f1b4a53c79f9c918bd32ea5f34ffdc6 ab80
3  1c1353097fb0633df2739e37a89fc91570caf3991409062cc91bc750bf3a79e5ff8ac41e451b21b3989
4  df2f17ea9
5  CLIENT_RANDOM d2919907ec3f447e2aab848dd17e073b6409b97aa0ba0cf666b0fecbc2550ee0 632d
6  b0061697d630605cd8a94ec42350615e12f290e3425c7a3786c80596adffc240a6648216fa96212c5ed
7  f32b74c90
```

Рис. 1. Запись сессионных ключей в формате NSS

```
⊞ Frame 55: 1337 bytes on wire (10696 bits), 1337 bytes captured (10696 bits) on interface 0
⊞ Ethernet II, Src: LiteonTe_12:56:f5 (74:e5:43:12:56:f5), Dst: AsustekC_20:13:b8 (08:60:6e:20:13:b8)
⊞ Internet Protocol Version 4, Src: 192.168.2.47 (192.168.2.47), Dst: 77.88.55.66 (77.88.55.66)
⊞ Transmission Control Protocol, Src Port: 62481 (62481), Dst Port: 443 (443), Seq: 346, Ack: 6421, Len: 1283
⊟ Secure Sockets Layer
  ⊟ TLSv1.2 Record Layer: Application Data Protocol: spdy
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 1278
      Encrypted Application Data: 0000000000000001ff18ba0b12dc96d2543bf126f82d00c3...
```

```
0000  08 60 6e 20 13 b8 74 e5  43 12 56 f5 08 00 45 00   .`n ..t. C.V...E.
0010  05 2b 7c 91 40 00 80 06  31 ca c0 a8 02 2f 4d 58   .+|.@... 1..../MX
0020  37 42 f4 11 01 bb b0 cc  a4 6e 8a d9 3e c7 50 18   7B...... .n..>.P.
0030  01 02 56 0e 00 00 17 03  03 04 fe 00 00 00 00 00   ..V..... ........
0040  00 00 01 ff 18 ba 0b 12  dc 96 d2 54 3b f1 26 f8   ........ ...T;.&.
0050  2d 00 c3 58 65 e0 c2 5c  82 8a 1a a9 27 fd 87 59   -..Xe..\ ....'..Y
0060  67 98 2b ad 87 24 f6 6b  7f 58 15 0d 36 c5 e2 74   g.+..$.k .X..6..t
0070  0b 44 85 5f f8 d2 64 b7  74 b8 7a 52 21 bc e1 b8   .D._..d. t.zR!...
0080  18 be cc 2a 29 d7 37 67  25 29 4a 98 b4 eb 4b 41   ...*).7g %)J...KA
0090  40 07 6e 73 65 bb 63 0e  2e 90 ce d8 37 39 29 c9   @.nse.c. ....79).
00a0  9d 09 fc 44 36 dc c4 e9  d3 59 df 8b 28 a1 c7 19   ...D6... .Y..(...
00b0  c8 a8 6f 6c b3 70 b8 02  2c 88 c0 93 cd 02 e7 c3   ..ol.p.. ,.......
00c0  a3 5a 9e 8b 81 dc 8e 1d  5c 9a e9 4f f6 d9 c6 14   .Z...... \..O....
00d0  08 09 b7 68 ca b8 ad 88  c3 b3 8d ca 4d 18 4c 2b   ...h.... ....M.L+
00e0  21 46 0e 5f 31 30 f3 fe  dc 27 ba 47 b6 11 58 97   !F._10.. .'.G..X.
00f0  96 12 f5 65 97 9a c7 0b  93 c3 49 d4 cf 9b 4b 5d   ...e.... ..I...K]
0100  05 65 13 7c bd 04 be 38  18 79 8b 3f f1 98 63 a4   .e.|...8 .y.?..c.
```

Рис. 2. Пример перехваченных зашифрованных пакетов

На помощь для получения ключей приходит логгирование сессионных ключей, именно тех, которые используются для зашифровки и расшифровки трафика. Получение таких логов не является трудоёмким. Их можно получить например из браузеров – с недавних версий браузеры Firefox и Chrome научились выводить в специально задаваемый файл данные, достаточные для деривации (получения) сессионных ключей, которыми шифруется передаваемый/принимаемый ими трафик, поскольку внутри TLS используется симметричное шифрование. Строго говоря, делают это не сами браузеры, а библиотека NSS в их составе; именно она задает формат записываемых файлов. В случае с трафиком от java-приложений создать требуемые лог-файлы можно исследовав отладочные записи JVM (Java Virtual Machines), а именно воспользовавшись опцией javax.net.debug.. Проанализировав полученные от этой опции отладочные записи, можно составить NSS-файл с соответствующим форматом. Этот формат умеет читаться и использоваться Wireshark'ом, чтобы расшифровывать TLS-записи, зашифрованные соответствующими ключами. В данной работе описан общий принцип расшифровки TLS-трафика. Для этого необходимо иметь файл с логгированными записями сессионных ключей в NSS-формате и, собственно, анализатор трафика – в данном случае Wireshark версии 1.6.0 или выше. На рисунке 1 представлен пример записей в лог-файле.

Нельзя не заметить, что Wireshark весьма чувствителен к формату NSS-файла, поэтому лучше тщательно перепроверить сходится ли число байт в каждом элементе строки, и нет ли где-либо лишних пробелов, это может сэкономить время.

Что касается анализатора трафика, то захватывать трафик нужно после того, как начнётся запись ключей в лог-файл, так как в противном случае нам не удастся обладать сессионными ключами, которые соответствуют захваченным TLS-записям. Также необходимо помнить, что ключи «эфемерны». Это значит, что они пригодны лишь для одной TLS-сессии, то есть лог от одного сеанса связи не подойдёт для дешифрации трафика другого сеанса. Также необходимо использовать навыки в работе с Wireshark – отслеживание обмена трафиком с определённым хостом и фильтрация по нужному протоколу, чтобы изначально отбросить ненужные пакеты, проходящие через прослушиваемый интерфейс. На рисунке 2 представлен пример перехваченных TLS-записей в обычном для перехватчика зашифрованном виде.

После того, как удалось сформировать файл с сессионными ключами, нужно привязать файл к Wireshark'у. Для этого требуется выполнить несколько шагов:

1. открыть в Wireshark контекстное меню на любом SSL/TLS пакете;
2. выбрать Protocol Preferences -> Secure Socket Layer Preferences;
3. в открывшемся окне в графе (Pre-)Master Secret logfilename указать путь к сформированному ранее NSS-файлу;
4. нажать ОК и анализировать изменения в перехваченном трафике.

Теперь в содержимом пакета появилась вкладка «Decrypted SSL Data». Теперь, если перейти в эту вкладку можно увидеть текст запроса. Кроме того теперь можно выбрать любой пакет с протоколом SSL или TLS и в его контекстном меню выбрать функцию «Follow SSL Stream» – в результате получается содержимое пакетов в виде, представленном на рисунке 3. Как видно, несмотря на то, что общение проходит по HTTPS, мы видим передаваемый трафик и можем экспортировать его для дальнейшего анализа.

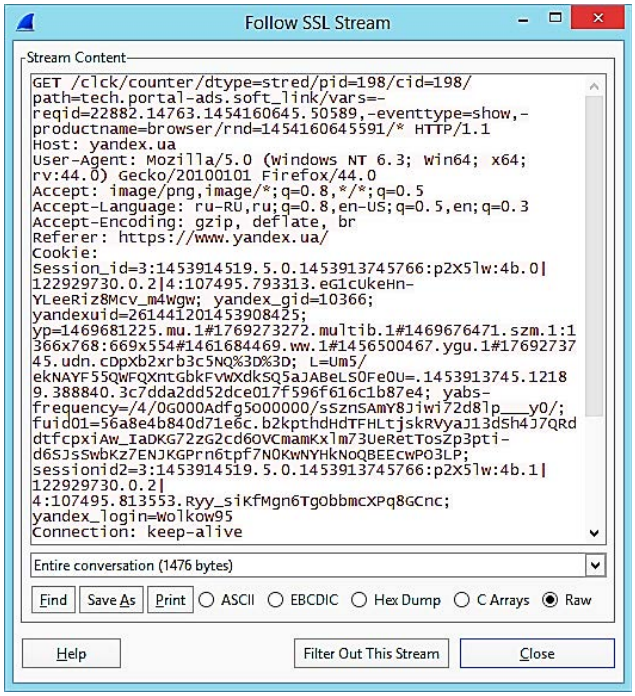

Рис. 3. Содержимое окна функции «FollowSSLStream»

Описанный в данной работе метод обладает основным недостатком: он требует существенных затрат времени на составление файла с сессионными ключами. Однако предложенный метод можно формализовать, а в последующем и автоматизировать, что сократит временные затраты на реализацию данной атаки и, возможно, откроет новые возможности для проведения таких атак. Примечательная особенность описанного метода заключается в том, что не обязательно перехватывать трафик на компьютере, который генерирует TLS-трафик, его можно перехватывать просто сидя в сети и прослушивая её. А добыть файл с сессионными ключами можно, поставив на компьютер жертвы закладку или просто скопировать его, имеядоступ к компьютеру.

Выводы и предложения. В данной работе проведен анализ основных атак на протокол TLS, на основании которого указаны методы противодействия рассмотренным атакам. Также разработан метод расшифровки трафика HTTPS, который можно применять даже при удаленном прослушивании сети, а при его дальнейшей автоматизации он позволит расшифровывать данные практически в режиме онлайн.

Список литературы:

1. BEAST: Surprising crypto attack against HTTPS // ekoparty Security Conference 7 / URL: http://ekoparty.org/2011/thai-duong.php
2. Bleichenbacher D. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1. In Advances in Cryptology CRYPTO'98, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 1462, pp. 1-12, Springer-Verlag, 1998.
3. SSL/TLS in Detail // Microsoft TechNet. – July 31, 2003.
4. SSL: Intercepted today, decrypted tomorrow / URL: http://news.netcraft.com/archives/2013/06/25/ssl-intercepted-today-decrypted-tomorrow.html
5. Vaudenay S. Security Flaws induced by CBC padding – Applications to SSL, IPSEC, WTLS. In Advances in Cryptology – EUROCRYPT '02, volume 2332 of Lecture Notes in Computer Science, pages 534–545. Springer-Verlag, 2002.
6. State of SSL // InfoSec World 2011 / URL: http://blog.ivanristic. com/Qualys_SSL_Labs State_of_SSL_InfoSec_World_April_2011.pdf
7. Hackers break SSL encryption used by millions of sites // The Register. URL: http://www.theregister.co.uk/2011/09/19/beast_exploits_paypal_ssl/
8. Как HTTPS обеспечивает безопасность соединения: что должен знать каждый Web-разработчик / URL: https://habrahabr.ru/post/188042/.

Волков В.А.
Харківський національний університет радіоелектроніки

ПРО ОДИН МЕТОД АТАКИ НА ПРОТОКОЛ TLS

Анотація
Головною уразливістю при передачі даних між клієнтом та сервером є можливість несанкціонованого доступу до конфіденційної інформації. Аби протидіяти цій уразливості використовується протокол TLS. Проте даний протокол не володіє достатньою мірою безпеки. В ході роботи було проведено аналіз основних атак на протокол TLS та вказано методи протидії до них. Також було розроблено метод перехоплення та розшифровування трафіка, який передається з використанням протоколу HTTPS. Запропонований метод можна автоматизувати. Подальша автоматизація розробленого методу дозволить розшифровувати дані, що передаються, майже у режимі реального часу
Ключові слова: протокол TLS, протокол HTTPS, уразливість, атаки, шифрування, розшифровування.

Volkov V.A.
Kharkov National University of Radio Electronics

ABOUT THE ONE OF THE METHODS OF ATTACK ON THE TLS PROTOCOL

Summary
The possibility of unauthorized access to confidential information is the main vulnerability of data transfer between client and server. The TLS protocol is used to counter that threat. However, this protocol does not have a high degree of security. During the work been analyzed the major attacks on the TLS protocol and specified methods to counteract these attacks. Also, a method of intercepting and decrypting traffic transmitted over HTTPS was developed. The proposed method can be used during remote listening to the network, and it is possible to automate. Further automation of the developed method allows decrypting the transmitted data in the approximate real-time mode.
Keywords: TLS protocol, HTTPS protocol, vulnerability, attacks, encryption, decryption.