Противодействие атакам на протокол TLS

Леонтьев Сергей Ефимович, ООО «КРИПТО-ПРО», lse@cryptopro.ru
Попов Владимир Олегович, ООО «КРИПТО-ПРО», vpopov@cryptopro.ru
Смышляев Станислав Витальевич, ООО «КРИПТО-ПРО», svs@cryptopro.ru

1. Введение

В сентябре 2011 года на конференции Ekoparty в Аргентине была представлена работа Дуонга и Риззо (см. [15,16]), посвященная практической реализации известной теоретической атаки на протокол SSL/TLS. Данная атака была предложена Грегори Бардом (citeBard) за 7 лет до работы Дуонга и Риззо и основывалась на особенностях режима CBC работы блоковых шифров в случае возможности проведения нарушителем атаки с выбором открытого текста при заранее известном значении синхропосылки. Стоит заметить, что уже в этой работе в дополнение к самому описанию метода криптоанализа Бард описывал возможности применения данного метода к протоколу TLS. Идея предложенного метода криптоанализа возникла еще раньше в 2002 году у Дэя ([9]).

Ранее Сержем Воденеем на Eurocrypt 2002 ([13]) была представлена работа, в которой описывался другой метод построения атаки на TLS, также использующий особенности режима CBC. В этой работе, в дополнение к самому описанию метода криптоанализа, Воденей описывал возможности применения данного метода к реально используемым протоколам — как SSL/TLS, так и IPSEC, WTLS и SSH2. При этом сама идея построения аналогичных атак с выбором шифртекста на использующие определенные режимы шифрования и виды паддинга протоколы возникла еще раньше — на CRYPTO 1998 ([3]).

В настоящей работе приводится краткий обзор основных атак, применимых к протоколу TLS на фазе передачи данных. Все они, так или иначе, основаны на идеях атак Барда и Воденея и возможны в моделях нарушителя, предполагающих возможность проведения злоумышленником активных действий, таких, как навязывание блоков открытого текста или шифртекста. Предлагается новая использующая временные характеристики атака на фазу передачи данных протокола TLS, возможная несмотря на принятые в версиях 1.1–1.2 контрмеры.

В заключение введения отметим, что определенный класс временных атак построен и на фазу handshake протокола TLS (см. [4,5,6]), предваряющую передачу данных.

2. Атака Барда

Напомним, что при работе блокового шифра в режиме CBC фиксируется синхропосылка $C_0 = IV$, имеющая длину, равную длине блока шифра. Затем для получения каждого блока шифртекста C_i, $i = 1,2, \ldots$, вычисляется значение $C_i = E_K(M_i \oplus C_{i-1})$, где E_K — функция зашифрования блокового шифра на ключе K. Расшифрование блока C_i сообщения производится в соответствии с выражением $M_i = D_K(C_i) \oplus C_{i-1}$, где $D_K = E_K^{-1}$, M_i — блок открытого текста.

Рассмотрим модель нарушителя, сходную с классической моделью IND-CPA, в которой угрозой является различение гипотез о равенстве либо неравенстве некоторого блока M_i открытого текста $M = (M_1|M_2|\ldots|M_m)$ (соответствующего известному шифртексту $C = (C_1|C_2|\ldots|C_m)$) фиксированному блоку M^*. Атака осуществляется с выбором открытого текста при известном IV. Для осуществления атаки нарушитель навязывает отправителю открытый текст для зашифрования, такой, что его начальный блок равен $M' = C_{i-1} \oplus IV \oplus M^*$. В этом случае первый блок шифртекста будет равен $C' = E_K(M' \oplus IV) = E_K(C_{i-1} \oplus IV \oplus M^* \oplus IV) = E_K(C_{i-1} \oplus M^*)$. Таким образом, гипотеза о

равенстве M_i и M^* верна тогда и только тогда, когда блокшифртекста C' в точности равен C_i.

3. Атака Барда в случае TLS

Как ясно из описания, для осуществления атаки Барда на практике нарушителю требуется иметь возможность читать пакеты в канале связи, навязывать обладателю ключа блоки данных для зашифрования, предварительно получая информацию о значении синхропосылки IV, которую обладатель ключа будет использовать для зашифрования навязываемых данных. Последняя проблема в случае TLS версии 1.0 легко разрешима — в качестве IV для очередного пакета используется последний блок шифртекста предыдущего пакета. Как следует из раннего варианта работы Риззо и Дуонга ([10]), схематичное описание решения оставшихся двух проблем состоит в следующем:

- запускается модуль (плагин на стороне клиента, сниффер в промежуточном шлюзе и т.п.), осуществляющий просмотр всего исходящего от клиента трафика — в частности, пакетов атакуемого TLS-соединения;
- после установления клиентом TLS-соединения, являющегося целью атаки, в браузере клиента (например, с использованием технологии межсайтового скриптинга) запускается JavaScript-код, осуществляющий навязывание блоков данных для зашифрования в рамках TLS-сессии.

Такая особенность работы протокола TLS, как использование в качестве синхропосылки для каждого пакета заранее известного нарушителю блока, была устранена в версии 1.1 TLS уже в 2006 году. Тем не менее, на апрель 2011 года (см. [17]) число серверов, поддерживающих версии TLS 1.1 или 1.2, было пренебрежимо мало по сравнению с числом тех, что поддерживают исключительно TLS 1.0, а среди популярных браузеров (см. [18]) TLS 1.2 поддерживает только Opera 10 (при этом по умолчанию TLS 1.2 выключен).

4. Атака Воденея: уязвимость CBC при определенных видах паддинга

Приведем краткое описание предложенного Воденеем метода криптоанализа блоковых шифров, работающих в режиме CBC, при использовании паддинга PKCS#5. Напомним, что для дополнения сообщения M паддингом PKCS#5 при использовании с блоковым шифром с длиной блока b байтов ($b \leq 255$) сообщение M дополняется $s = b - (len(M) \mod b)$ байтами, значение каждого из которых равно s.

Предположим, что у нарушителя есть доступ к некоторому оракулу O_K, на вход которому подается пара из синхропосылки $\tilde{C}_0$ и шифртекста $\tilde{C}$ и который возвращает TRUE в том и только в том случае, когда в получаемом из $widetildeC$ расшифрованием в режиме CBC с синхропосылкой $\tilde{C}_0$ тексте содержится корректный PKCS#5 паддинг.

Для частичного дешифрования блока C_i шифртекста $C = (C_1|C_2|\dots|C_m) = E_K(M_1|M_2|\dots|M_m)$ проводятся следующие действия:

1. Случайным образом выбирается порядка 2^8 блоков $\tilde{C}_{i-1}^j$, $j = 1,2\dots$
2. Для каждого из $\tilde{C}_{i-1}^j$, $j = 1,2\dots$, на вход оракулу подается строка $\tilde{C}^j = (C_0|C_1|\dots|C_{i-2}|\tilde{C}_{i-1}^j|C_i)$. Если при некотором j для $\tilde{C}_{i-1}^j$ оракул возвращает TRUE, C_{i-1}^* полагается равным $\tilde{C}_{i-1}^j$.
3. С использованием информации о том, что в блоке $P_i = C_{i-1}^* \oplus D_K(C_i)$ содержится корректный PKCS#5 паддинг, ищется длина паддинга: перебираются сообщения $C_{i-1}^*(1) = C_{i-1}^* \oplus (00|00|\dots|00|01)$,

$C^*_{i-1}(2) = C^*_{i-1} \oplus (00|\ldots|00|01|00)$, …, $C^*_{i-1}(b) = C^*_{i-1} \oplus (01|00|\ldots|00|00)$ и на вход оракулу O_K подаются строки $(C_0|C_1|\ldots|C_{i-2}|C(j)|C_i)$, $j = b, b-1, \ldots, 1$. В случае, если ответ FALSE впервые появится от оракула при $j = j^*$, становится известно, что в сообщении, получаемом расшифрованием на ключе K из шифртекста $(C_0|C_1|\ldots|C_{i-2}|\tilde{C}^*_{i-1}|C_i)$ в конце стоит j^* байтов со значением j^*.

4. Для нахождения последних j^* байтов M_i вычисляется значение выражения $\tilde{C}^*_{i-1} \oplus C_{i-1} \oplus (00|00|\ldots|\underbrace{j^*|j^*|\ldots|j^*}_{j^*})$ и берутся его последние j^* байтов.

Заметим, что методы криптоанализа, аналогичные описанному, применимы и при других видах паддинга. Рассмотрению уязвимостей при использовании семи других видов паддинга, включая используемый в ESP протокола IPSec, посвящены работы [2,11].

5. Атака Воденея в случае TLS

Пусть два узла A и B связаны каналом связи и взаимодействуют по протоколу TLS версии 1.0, и есть активный нарушитель C, который находится посередине и имеет возможность как читать пакеты в канале, так и посылать произвольные пакеты в канал в сторону узла B. Для практической реализации вышеописанной атаки нарушителю C после отправки каждого пакета с модифицированным шифртекстом требуется получать информацию о том, корректен ли открытый текст, соответствующий модифицированному шифртексту, с точки зрения используемого алгоритма паддинга. Заметим, что для реализации оракула, описанного в работе Воденея, достаточно, чтобы код ошибки при неправильном паддинге был отличен от кода ошибки в случае верного паддинга, но неверного значения имитовставки. В случае использования TLS v.1.0 для аутентификации сообщения используется имитозащита, получаемая по открытому тексту и проверяемая уже после проверки корректности паддинга. Перехватывая идущие от B пакеты, нарушитель C в ответ на каждый пакет с модифицированным шифртекстом получит один из двух кодов возврата:

1. **decryption_failed** — на стороне B расшифрование пакета прервалось на шаге проверки паддинга — паддинг некорректен;
2. **bad_record_mac** — на стороне B расшифрование пакета прервалось на шаге проверки имитовставки — следовательно, паддинг корректен.

В случае, если для аутентификации сообщения используется имитозащита, получаемая также по открытому тексту и проверяемая после корректности паддинга, но без различия между кодами ошибок в двух возможных случаях, оракул O_K может быть реализован на практике с помощью отслеживания времени δt, проходящего от начала процесса аутентификации сообщения до отказа принимать сообщение из-за его неаутентичности (см. [7,8]). Заметим, что в этом случае (по причине необходимости наличия существенной разницы между значением δt в случае сообщения с подходящим паддингом, но неправильной имитовставкой, и значением δt в случае неправильной имитовставки) требуется большая длина сообщений и нет возможности получать информацию о начальных блоках открытого текста.

6. Результаты Кравчика

В случае использования просто устроенного паддинга методам криптоанализа, аналогичным методу Воденея, подвержен не только режим CBC работы блоковых шифров, но и все режимы, при которых расшифрование каждого блока производится с помощью наложения гаммы: CFB, OFB. При этом даже использование паддинга, вносящего существенную дополнительную энтропию, в некоторых случаях не является препятствием для построения аналогичных атак (см. [11]).

Данные результаты хорошо согласуются с полученными ранее (и изложенными на CRYPTO 2001) Хьюго Кравчиком ([12]) результатами об отсутствии, в общем случае, IND-CCA стойкости (то есть, стойкости против угрозы получения частичной информации о шифртексте при атаке с выбором шифртекста) криптосистем с секретным ключом, работающих по принципу <<аутентификация, затем шифрование>> (а, вообще говоря, криптосистем с секретным ключом, в которых шифрованию предшествует дополнение открытого текста имеющим малуюэнтропию отрезком текста), даже в случае IND-CPA стойкости (стойкости против угрозы получения частичной информации о шифртексте при атаке с выбором открытого текста) используемой непосредственно для шифрования криптосистемы.

7. Предлагаемая атака

Для ликвидации уязвимости к атакам, аналогичным атаке Воденея, принят ряд контрмер, связанных с порядком вычисления имитовставки при некорректном паддинге. В пункте 6.2.3.2 RFC 5246 (описание протокола TLS версии 1.2, [19]) присутствует следующий аспект: указано, что лучшим методом защиты от временных атак, аналогичных атаке Воденея, является вычисление значения имитовставки даже в случае некорректного паддинга, например, вычисление значения имитовставки по всему полученному после расшифрования фрагменту, полагая отсутствие паддинга.

В конце пункта 6.2.3.2 также указано, что предлагаемые меры оставляют некоторый временной канал утечки информации, связанный с наличием разницы по времени вычисления имитовставки в случае сообщения с корректным паддингом (время вычисления самого сообщения) и модифицированного сообщения с некорректным паддингом (время вычисления сообщения и соответствующих паддингу байтов), однако данный канал незначителен по причине большого размера блока современных алгоритмов выработки имитовставки.

Однако заметим, что в том же пункте 6.2.3.2 указано, что паддинг может иметь любую длину, не превосходящую 255 байтов, обеспечивающую должное выравнивание. Таким образом, при размере блока алгоритма выработки имитовставки в 16 байтов сообщение длиной, например, 16 байтов, дополненное 240 байтами вида $0xf0$, будет считаться дополненным корректным паддингом и при проверке корректности сообщения будет вычисляться имитовставка на 1 блок. В случае же контролируемой модификации конечных блоков данного открытого текста (производимой аналогично описанному выше методу Воденея) паддинг будет приниматься как некорректный и имитовставка будет вычисляться на 16 блоков, обеспечивая, таким образом, существенную разницу во времени вычисления, создающую потенциальный канал утечки информации.

Опишем построение данного канала утечки и использования этого канала для дешифрования блока C_i передаваемого по каналу связи шифртекста $C = (C_1|C_2|\dots|C_m) = E_K(M_1|M_2|\dots|M_m)$. Для сокращения объема обозначений будем полагать, что длины блоков используемых алгоритмов шифрования и имитовставки совпадают и равны b, $1 \le b \le 128$, b делит 256.

Атака проводится в предположении о возможности зашифрования модифицированного справа от блока M_i текста M на текущем ключе и наличием доступа к оракулу, вычисляющему предикат о корректности паддинга открытого текста, соответствующего поданному на вход шифртексту.

Для построения оракула надо иметь возможность различать следующие два события:

- Вычисляется имитовставка на сообщение длины не большей $2b$ байтов;
- Вычисляется имитовставка на сообщение длины 255 байтов.

Предлагаемый метод состоит в выполнении следующей процедуры.

1. Положить $t = 1$.
2. Положить $s_t = 256 - 2b + t$.
3. Навязать отправителю зашифрование на текущем ключе любого сообщения вида $(\dots|M_i)|(\underbrace{s_t|s_t|\dots|s_t}_{256-2b})$.
4. Положить $(\dots)|(w^t_{b-(t-1)}|w^t_{b-(t-2)}|\dots|w^t_b)|C_t|Q_t$ равным полученному шифртексту, где Q_t имеет длину $256 - 2b$ байтов, C_t — длину b байтов, w^i_k — отдельные байты.
5. Положить для всякого j, $j = b - (t-2), b - (t-3), \dots, b$ байт блока C'_t с номером j равным $r^t_j \oplus w^t_j \oplus w^{t-1}_j \oplus s_t \oplus s_{t-1}$.
6. Случайным образом выбрать значение байта с номером $\mathrm{b} - (\mathrm{t} - 1)$ блока $\mathrm{C'_t}$. Если оракул на входе $(\mathrm{C'_t}|\mathrm{C_t}|\mathrm{Q_t})$ выдаст FALSE, перейти к пункту 3.
7. Положить $(\mathrm{r^{t+1}_{b-(t-1)}}, \mathrm{r^{t+1}_{b-(t-2)}}, \dots, \mathrm{r^{t+1}_b})$ равными последним t байтам блока $\mathrm{C'_t}$.
8. Увеличить t на 1. Если $\mathrm{t} \le b$, перейти к пункту 2.
9. Вычислить блок $\mathrm{M_i}$: $\mathrm{M_i} = \mathrm{C'_b} \oplus (\mathrm{w^b_1}|\mathrm{w^b_2}|\dots|\mathrm{w^b_b}) \oplus (\mathrm{s_b}|\mathrm{s_b}|\dots|\mathrm{s_b})$.

Таким образом, блок M_i восстанавливается побайтово, начиная с последнего байта, причем для восстановления каждого байта требуется в среднем 128 опробований. Заметим, что после каждой ошибки расшифрования при проверке имитовставки или паддинга соединение разрывается и производится смена ключа, поэтому перед каждым выполнением пункта 7 требуется выполнять пункт 3, для соответствия текущему значению ключа.

8. Заключение

Вышеописанные типы атак используют следующие три проблемных аспекта:

1. потенциальная возможность бесключевого чтения при повторном использовании синхропосылки;
2. уязвимость к навязыванию шифртекста;
3. некорректное использование алгоритмов проверки паддинга и алгоритмов проверки имитовставки.

В настоящее время стойкими к описанным типам атак являются (см., например, [20]) сюиты протокола TLS 1.2, использующие шифрование с аутентификацией в режиме GCM (см. [14]):

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.

Также не подвержены этим видам атак сюиты потокового шифрования:

- TLS_RSA_WITH_RC4_128_SHA,
- TLS_RSA_WITH_RC4_128_MD5.

Исключить применимость данного класса атак возможно с помощью следующих решений.

- **Аутентификация сообщений вместе с паддингом.** Каждое принимаемое сообщение обязано проходить проверку аутентичности с использованием режима выработки имитовставки блокового шифра ГОСТ 28147-89, что делает невозможным навязывание сообщений. Сквозная имита вместе с защитой имитой номера пакета и заголовка позволяет предотвратить перестановку пакетов. Выработка имитовставки на сообщение вместе с паддингом позволит предотвратить атаки, связанные с различием в реакции на навязанное сообщение.

Заметим, что в стандарте SSL v2 аутентификация сообщений производится вместе с паддингом. Это позже было изменено, что привело к появлению описанных выше атак.

- **Случайный выбор синхропосылки.** Обработка каждого пакета должна предваряться отдельным запуском генератора случайных чисел для порождения новой, никак не зависящей от предыдущих данных в канале, синхропосылки. Благодаря этому невозможны любые атаки, использующие уязвимости, связанные с повторным использованием синхропосылок либо с использованием в качестве синхропосылок ранее присутствовавших в канале блоков шифртекста.
- **Отказ от использования паддинга.** Использование режима CNT (гаммирования) работы блокового шифра, не требующего использования паддинга, делает невозможными любые атаки, использующие вносимую алгоритмами паддинга избыточность.

Данные решения заложены в реализацию TLS на базе российских стандартов криптографической защиты данных. Таким образом, для ликвидации уязвимости ко всем описанным типам атак на TLS предлагается использовать следующие сюиты:

- TLS_GOSTR341094_WITH_28147_CNT_IMIT,
- TLS_GOSTR341001_WITH_28147_CNT_IMIT.

References

[1] Gregory V. Bard. Vulnerability of SSL to Chosen-Plaintext Attack URL: http://eprint.iacr.org/2004/111.pdf.

[2] John Black, Hector Urtubia. Side-Channel Attacks on Symmetric Encryption Schemes: The Case for Authenticated Encryption. URL: http://www.usenix.org/event/sec02/full_papers/black/black.pdf.

[3] D. Bleichenbacher. 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.

[4] D. Brumley and D. Boneh. Remote timing attacks are practical. In Proceedings of the 12th USENIX Security Symposium, 2003.

[5] D. Brumley and D. Boneh. Remote timing attacks are practical. Computer Networks, 48(5):701–716, 2005.

[6] B.B. Brumley and N. Tuveri. Remote Timing Attacks are Still Practical. URL: http://eprint.iacr.org/2011/232.pdf.

[7] B. Canvel, A. Hiltgen, S. Vaudenay and M. Vuagnoux. Password Interception in a SSL/TLS Channel. Advances in Cryptology — CRYPTO 2003, LNCS vol. 2729, 2003.

[8] B. Canvel. Password Interception in a SSL/TLS Channel. URL: http://lasecwww.epfl.ch/memo/memo_ssl.shtml.

[9] W. Dai. An Attack Against SSH2 Protocol, Feb. 2002. Email to the ietf-ssh@netbsd.org email list, 2002.

[10] T. Duong, J. Rizzo. Here-Come-the-XOR-Ninjas. URL: http://blog.cfrtechnologies.biz/wp-content/uploads/2011/09/Here-Come-the-XOR-Ninjas.pdf.

[11] Vlastimil Klima and Tomas Rosa. Side Channel Attacks on CBC Encrypted Messages in the PKCS#7 Format. URL: http://eprint.iacr.org/2003/098.pdf.

[12] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In Advances in Cryptology CRYPTO 2001, Lecture Notes in Computer Science 1462, 2001 - Springer. Pp. 310-331.

[13] S. Vaudenay. Security Flaws induced by CBC padding — Applications toSSL, IPSEC, WTLS. In Advances in Cryptology —– EUROCRYPT '02, volume 2332 of Lecture Notes in

Computer Science, pages 534–545. Springer-Verlag, 2002
[14] NIST Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication, SP 800-38D, November 2007.
[15] Hackers break SSL encryption used by millions of sites // The Register. URL: http://www.theregister.co.uk/2011/09/19/beast_exploits_paypal_ssl/
[16] BEAST: Surprising crypto attack against HTTPS // ekoparty Security Conference 7. URL: http://ekoparty.org/2011/thai-duong.php
[17] State of SSL //InfoSec World 2011. URL: http://blog.ivanristic.com/Qualys_SSL_Labs-State_of_SSL_InfoSec_World_April_2011.pdf
[18] Отчет о степени поддержки в браузерах и http-серверах версий SSL/TLS и шифров // OpenNET. URL: http://www.opennet.ru/opennews/art.shtml?num=31839
[19] RFC 5246, The Transport Layer Security (TLS) Protocol Version 1.2 // URL: http://tools.ietf.org/html/rfc5246
[20] Cipher Suite Mitigation For Beast // PhoneFactor. URL: http://www.phonefactor.com/resources/CipherSuiteMitigationForBeast.pdf