Механизмы шифрования в современных вымогателях

Вымогатели представляют из себя основную угрозу для пользователей ПК, активность которой растет с течением времени. Согласно недавнему полугодовому отчету Cisco, вымогатели доминируют на рынке вредоносных программ и являются самым прибыльным типом вредоносного ПО для злоумышленников за всю историю. О вымогателях уже было написано достаточно много и большинство людей понимают, что они из себя представляют.

Вымогатели используют различные приемы для блокирования доступа к компьютеру жертвы или файлам пользователя, после чего требуют выкуп за восстановление к ним доступа. В последнее время мы наблюдаем существенное увеличение активности вымогателей-шифровальщиков, которые специализируются на шифровании файлов пользователя. В этом материале мы рассмотрим методы шифрования, используемые шифровальщиками, а также проанализируем ошибки, которые допускают авторы при их разработке.

Общие сведения

Шифрование является важнейшим элементом шифровальщиков, так как вся успешность такого «бизнеса» зависит от того, насколько успешно оно используется для блокирования доступа к файлам пользователя. Само шифрование не является каким-либо вредоносным методом или операцией. Оно представляет из себя мощный и легитимный инструмент, используемый простыми пользователями, корпоративными, а также правительственными для защиты данных от несанкционированного доступа.

Шифровальщики предназначены для «кражи» данных пользователя. Они привлекают для этой цели шифрование, которое предоставляет им возможность предотвратить доступ к данным их законному владельцу, так как он не знает ключа шифрования. К данным могут получить доступ только злоумышленники, которые знают этот ключ.

После своего запуска в системе, шифровальщик начинает модифицировать файлы или критические служебные данные файловой системы таким образом, что они могут быть прочитаны только путем их восстановления в оригинальное состояние. В свою очередь, эта операция требует использования ключа расшифровки файлов, который известен только злоумышленникам. Очевидно, что в таком случае, шифрование и обратная операция дешифрования используются для вредоносных целей.

Кроме этого, шифрование используется авторами вредоносного ПО для безопасного взаимодействия вредоносного ПО с его управляющим C&C-сервером, который, в свою очередь, хранит необходимый для расшифровки данных ключ. С использованием этого ключа данные файловой системы или сами файлы могут быть восстановлены в первоначальное состояние.

Авторы шифровальщиков используют преимущества обоих типов шифрования, как симметричного, так и ассиметричного. Это дает им оптимальную производительность и удобство при шифровании данных. В случае с симметричной схемой шифрования, один и тот же секретный ключ используется как для шифрования данных, так и для их расшифровки. Для асимметричного шифрования используется два ключа: закрытый и открытый. Первый известен только злоумышленникам и применяется для расшифровки данных, а второй является публичным и используется для шифрования данных.

Симметричное шифрование является полезным для шифрования с точки зрения удобства и производительности, позволяя вредоносной программе зашифровать данные в разумные сроки. С другой стороны, асимметричное шифрование используется для шифрования симметричного ключа, который может изменяться. Таким образом, злоумышленники могут поддерживать один ключ расшифровки для всех своих жертв, вместо хранения ключа в каждом конкретном случае заражения.

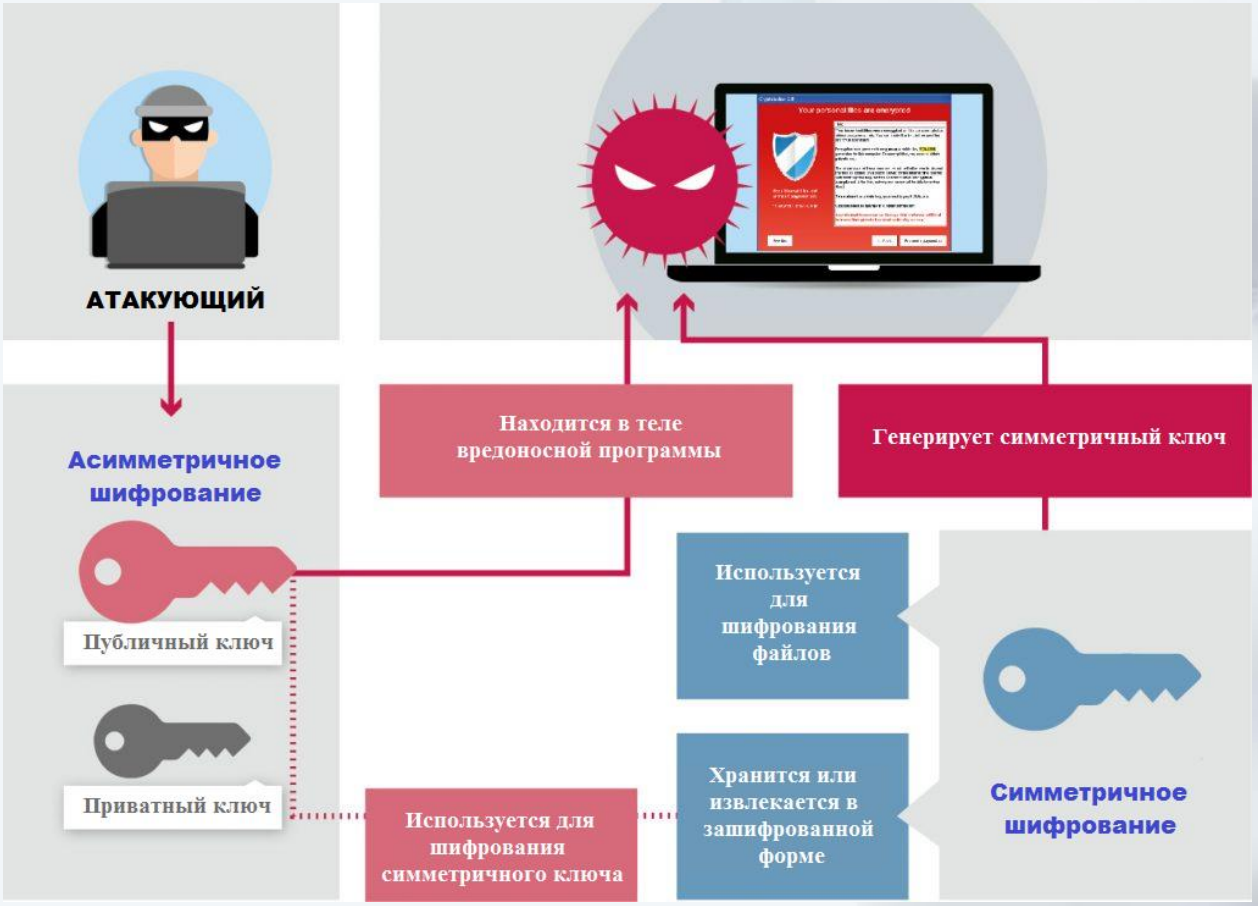

Рис. 1. Схема работы симметричного и асимметричного шифрования, которое используется в шифровальщиках.

Механизм защиты симметричного ключа может изменяться, но, как правило, для этого используется публичный ключ, который находится в теле вредоносной программы или извлекается с управляющего C&C-сервера, а затем привлекается к его шифрованию. После окончания процесса шифрования, симметричный ключ часто передается на сервер или хранится в системе жертвы.

Взаимодействие с управляющим C&C-сервером

Шифровальщики используют криптографию при работе со своим C&C-сервером для сохранения конфиденциальности передаваемых данных, а также сокрытия вредоносной активности. Более ранние семейства и версии шифровальщиков уделяли особое внимание конфиденциальности при работе с C&C. Более новые семейства и модификации используют такие проверенные стандарты шифрования для своих протоколов как TLS.

Шифрование трафика между скомпрометированной системой и сервером делает более сложной задачу обнаружения вредоносной активности, поскольку ее сложно отличить от того протокола шифрования, который используется в легитимных операциях, например, онлайн-банкинга.

Для инспектирования зашифрованного трафика, сетевые средства безопасности требуют от сети предполагаемой жертвы присутствия дополнительных возможностей. Например, для проверки зашифрованного посредством TLS трафика, может быть использовано решение IDS/IPS, которое располагается на пути передачи данных между компьютерами внутренней сети и удаленными серверами. Проверка трафика TLS также возможна на уровне самого компьютера, в зависимости от возможностей используемого антивирусного средства.

В том случае, когда внутренняя сеть не имеет возможности для отслеживания трафика, вредоносная программа использует шифрование для подключения к C&C, которое будет оставаться необнаруженным пока адреса C&C не будут найдены и добавлены в черный список вредоносных IP-адресов и доменов. Таким образом, данный способ защиты является более реактивным.

Ошибки в реализации шифрования

Специалисты ESET уже публиковали отчет об эволюции шифровальщиков, в котором явно указывалась их растущая распространенность. Число семейств и модификаций шифровальщиков значительно возросло начиная с 2011 г. Кроме этого, увеличилось число платформ, на работу в которых рассчитаны вымогатели.

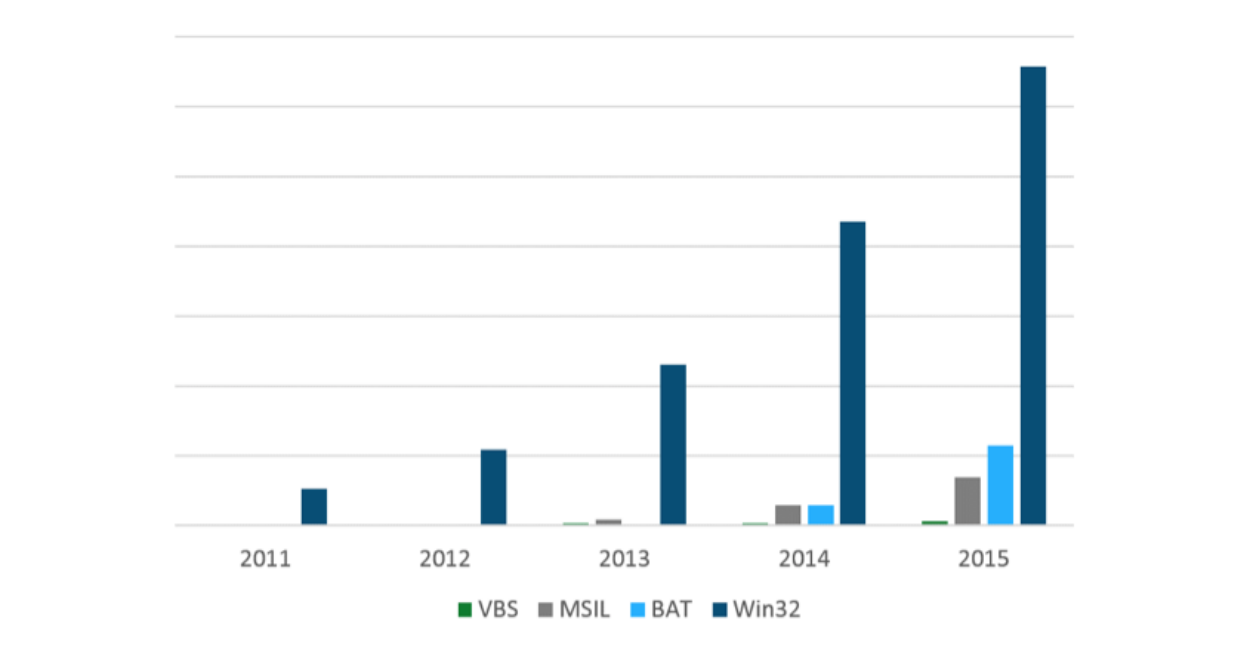

Рис. 2. Уровень распространенности шифровальщиков за период с 2011 по 2015 гг.

Оставшаяся часть нашего анализа будет посвящена тому, каким образом реализовано шифрование в четырех семействах шифровальщиков на разных этапах их эволюции: CryptoDefense (2014), TorrentLocker (2014), а также ставшие популярными TeslaCrypt (2015) и Petya (2016 г.).

Тип шифровальщика	Использует оба типа шифрования
CryptoDefense	-
TorrentLocker	+

TeslaCrypt	+
Petya	-

Цель нашего исследования заключается в том, чтобы показать, каким образом шифровальщики развивались с течением времени и проанализировать ошибки их авторов от глупейших до более сложных.

Когда вымогатель CryptoDefense впервые был замечен в использовании злоумышленниками, его исследование показало, что он был очень похож на CryptoLocker. Последний был одним из первых в своем роде шифровальщиком. Примечателен тот факт, что CryptoDefense имел серьезные недоработки в реализации механизма своего взаимодействия с управляющим C&C-сервером, а также в процессе шифрования файлов.

Протокол для работы с C&C основан на использовании POST-запросов HTTP-протокола, отправляемых вредоносной программой с зараженного хоста. Для обеспечения безопасности при работе с C&C, CryptoDefense использовал обфускацию URL-адресов в POST-запросах HTTP-протокола. Таким образом осуществлялось сокрытие ключа шифрования, который использовался для шифрования сообщения. Тело POST-запроса содержало сообщение C&C-протокола, зашифрованное с помощью алгоритма RC4 и скрытого ключа из URL-адреса POST.

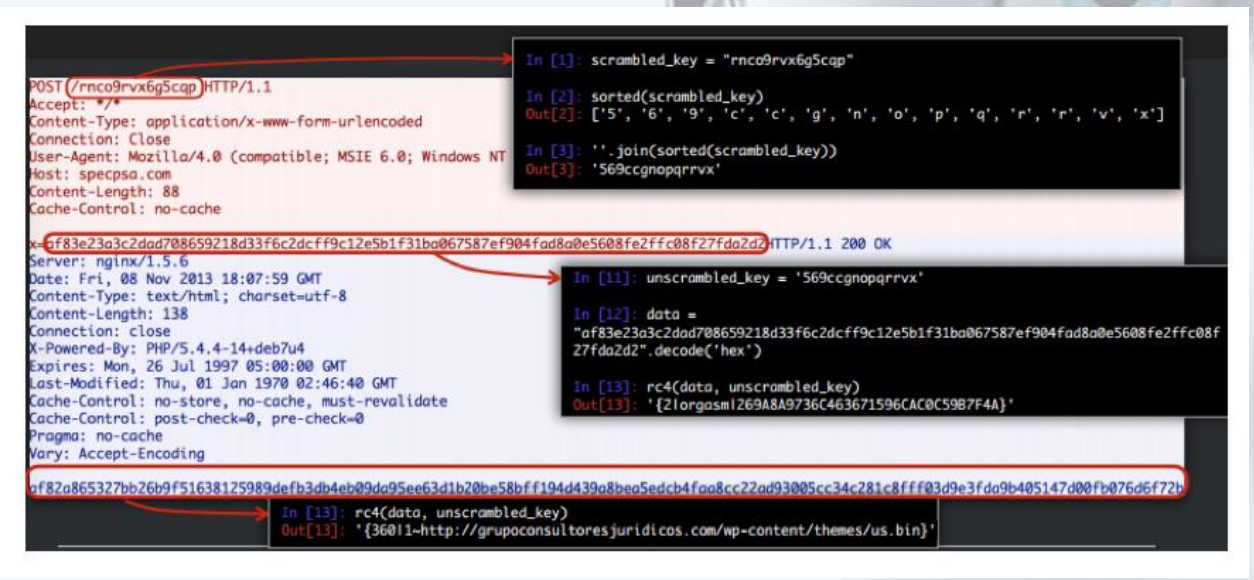

Рис. 3. Расшифровка C&C протокола шифровальщика CryptoDefense.

В случае перехвата зашифрованного CryptoDefense сообщения, оно может быть легко расшифровано. Сам POST-запрос содержит в себе всю информацию, необходимую для восстановления ключа шифрования и раскрытия механизмов работы протокола взаимодействия с C&C. Это, в свою очередь, позволяет создавать сетевую сигнатуру для обнаружения активности вымогателя и блокирования его вредоносных действий, в частности, правильной работы полезной нагрузки.

В случае с шифрованием файлов, ошибка в реализации алгоритма удивительно похожа на предыдущую. Для того, чтобы понять ее, рассмотрим алгоритм выполнения действий вымогателя CryptoLocker.

1. После компрометации жертвы, вымогатель уведомляет об этом свой C&C-сервер и указывает при этом идентификатор (ID) кампании, а также уникальный ID системы.
2. Управляющий C&C-сервер отвечает сообщением OK для подтверждения.
3. Клиент обращается к серверу еще раз, указывая идентификатор кампании и уникальный

идентификатор системы.

4. Сервер предоставляет вымогателю сообщение с требованием выкупа и публичный ключ RSA-2048 из пары ключей, которая была сгенерирована для данной комбинации ID кампании/ID системы жертвы. Закрытый ключ из этой пары никогда не покидает C&C-сервер.

5. Вымогатель подтверждает C&C-серверу успешное принятие сообщения с требованием выкупа и публичного ключа. Он также подтверждает окончание процесса шифрования файлов.

Уязвимое место в механизме работы CryptoLocker заключается в присутствии шагов 3 и 4, так как в таком случае шифрование файлов у пользователя происходит только в случае успешного общения вымогателя с C&C-сервером. Данные шаги были удалены в алгоритме работы CryptoDefense, который генерирует пару ключей на самой системе жертвы.

Тем не менее, авторы CryptoDefense упустили из вида небольшую деталь, которая заключается в том, что вымогатель забывает удалить из системы приватный ключ RSA. Этот ключ может быть использован для расшифровки зашифрованных файлов и может быть найден в системе. После этого достаточно использовать одну из API функций Windows для расшифровки файлов.

Таким образом, авторы CryptoDefense исправили одну ошибку в работе вымогателя, но сразу же допустили вторую.

TorrentLocker

Ранние версии TorrentLocker также содержали ошибки в реализации шифрования, но опирались на очень безопасный алгоритм шифрования файлов AES в режиме CTR. Тем не менее, тот способ, которым был реализован этот алгоритм, привел к присутствию слабых мест в общей схеме шифрования.

Алгоритм AES представляет из себя блочный шифр, т. е. он подразумевает под собой шифрование данных блоками фиксированного размера (16 байт). AES в режиме CTR принимает на вход начальное значение в качестве параметра, известного как вектор инициализации или Initialization Vector (IV) для расширения значения ключа (который может иметь разрядность 128, 192 или 256 бит) до размера самих подлежащих для шифрования данных, что называется ключевым потоком (keystream). После этого, к ключевому потоку применяется операция XOR на данных исходного сообщения, имитируя концепцию потокового шифра.

Эксплуатация уязвимого места в реализации шифрования TorrentLocker была довольно простая: на скомпрометированной системе, вымогатель генерирует ключевой поток размером 2МБ и подвергает шифрованию первые 2МБ каждого файла. В том случае, когда размер файла меньше 2МБ, он полностью шифруется этим методом.

Тем не менее, использование AES-CTR было совершенно аналогично методу повторного использования вектора инициализации и ключа потокового шифра, что является распространенной ошибкой среди новичков в криптографии. Имея в наличии хотя бы один зашифрованный файл и его оригинальное исходное содержимое, довольно просто восстановить ключевой поток и расшифровать остальные файлы. В более поздних версиях TorrentLocker авторы исправили данную ошибку путем замены режима работы AES CTR на CBC, который будет описан нами в следующем разделе посвященном TeslaCrypt.

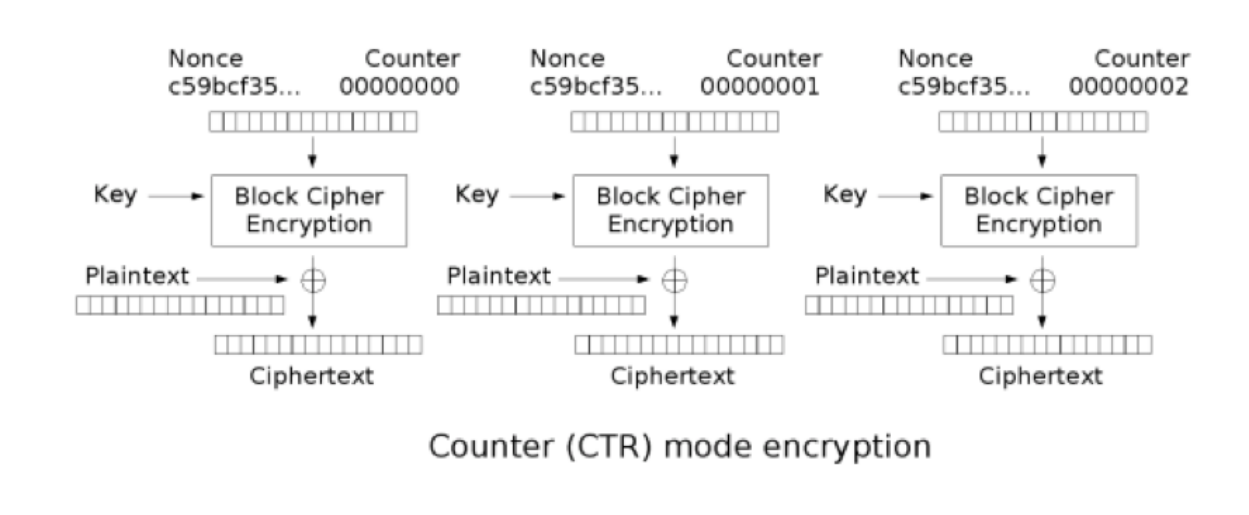

Рис. 4. Схема режима шифрования CTR.

Petya

Шифровальщик Petya использует отличный от других вымогателей подход для выполнения своих вредоносных функций. Вместо того, чтобы шифровать каждый файл отдельно, он направлен на компрометацию структур данных диска или файловой системы. Целью Petya является сектор главной загрузочной записи (MBR) системы жертвы, который используется для загрузки ОС.

Авторы Petya выбрали алгоритм шифрования Salsa20, который принадлежит проекту выявления новых поточных шифров eSTREAM. Данная инициатива является проектом по содействию разработки новых потоковых шрифтов для замены уже устаревших типа RC4. Использование авторами Petya шифра Salsa20 означает эволюцию шифровальщиков, в которых теперь используются новые и более надежные шифры.

Когда Petya запускается на исполнение, он начинает процесс заражения системы, который состоит из двух этапов. Процесс шифрования MBR довольно хорошо реализован. Он начинается с модификации MBR, а затем вызывает BSOD для того, чтобы заставить пользователя перезагрузить систему. После перезагрузки пользователю отображается фальшивый экран известного инструмента CHKDSK, пока Petya выполняет дополнительные операции по шифрованию MBR. После этого он перезагружает систему еще раз и показывает запугивающее изображение пользователю с текстом требования выкупа.

Рис. 5. Пугающая пользователя картинка шифровальщика Petya с требованием выкупа.

Несмотря на устрашающее сообщение с требованием выкупа, существует способ расшифровки MBR, так как авторы допустили досадные ошибки в коде вымогателя. Самый важный недостаток, который позволяет расшифровать MBR без обращения к злоумышленникам, заключается в способе, который шифровальщик использует для получения ключа восстановления (recovery key).

Для того, чтобы понять суть данного недостатка, рассмотрим ситуацию шифрования с другой стороны. Начнем с исследования того процесса, каким образом ключ восстановления, который предоставляется злоумышленниками жертве, используется для расшифровки файлов. После этого мы объясним, каким образом этот баг в реализации может быть использован расшифровки MBR без знания всего ключа целиком.

Поле для набора символов ключа расшифровки ограничено 54 символами, а его длина фиксирована и составляет 16 символов. Как только пользователь вводит ключ в поле сообщения с требованием выкупа, выполняется его проверка. В процессе проверки, введенный ключ восстановления расширяется до 32-х байт ключа шифрования с помощью специального алгоритма и сравнивается с проверочным буфером в последовательности. Если закодированный ключ соответствует буферу проверки, он поступает на вход функции реализации алгоритма шифрования Salsa20 (на самом деле Salsa10), после чего происходит расшифровка данных секторов диска.

Хотя метод расширения ключа выполняется именно на ключе восстановления, что удваивает его размер, он не добавляет никакой дополнительной энтропии ключа, поскольку алгоритм расширения является детерминированным. Таким образом, его безопасность, в данному случае, ограничена количеством 54^16 различных ключей, что соответствует уровню безопасности 92-х бит, (т. е. log2(54^16) = 92.07) и находится за пределами возможного успешного использования метода грубой силы для поиска ключа, например, NSA имеет возможность перебора ключей длиной до 80 бит.

Авторы Petya допустили ошибку в реализации основной функции Salsa20, которая создает массив из 16 слов, содержащих константы и ключ из так называемой главной таблицы, которая используется в процессе шифрования. Авторы шифровальщика исправили существующую реализацию Salsa20 для поддержки ее 16-битной архитектуры, но они пропустили ключевую часть. Причиной, по которой мы так внимательно относимся к реализации Salsa20, является режим исполнения кода вымогателя. Так как Petya предназначен для работы в качестве загрузчика, который исполняется в 16-битном реальном режиме работы x86 микропроцессоров.

Для создания вышеуказанного массива, две копии основной таблицы создаются в формате прямого порядка байт (little-endian). Авторы шифровальщика изменили переменные типа uint32_t на uint16_t, для адаптирования кода под 16-разрядную архитектуру. Тем не менее, они пренебрегли адаптацией новой разрядности для выполнения цикла при работе с основной таблицей в основной функции Salsa20. Из-за этой ошибки, после чтения двух байт смещение для следующей операции чтения данных учеличивается не на два, а на четыре байта.

```
static void s20_hash(uint8_t seq[64])
{
  int i;

  uint16_t x[16]; // 16-bit vectors
  uint16_t z[16]; // 16-bit vectors

  for (i = 0; i < 16; ++i)
    x[i] = z[i] = s20_littleendian16(seq + (4 * i)); // reads short (2 bytes),
but increments by 4 bytes (sizeof(uint32))

  for (i = 0; i < 10; ++i)
    s20_doubleround(z);

  for (i = 0; i < 16; ++i) {
    z[i] += x[i];
    s20_rev_littleendian(seq + (4 * i), z[i]);
  }
}
```

Рис. 6. Ошибка в реализации Salsa20 для 16-битной архитектуры.

По этой причине, в операции шифрования Salsa20, фактически, участвует только половина ключа. Это снижает уровень безопасности всей схемы шифрования Petya с 92-х бит до 46-ти. В таком случае, ключ можно подобрать в течение секунд используя метод грубой силы даже на обычных компьютерах.

TeslaCrypt

Не является случайным тот факт, что шифровальщик TeslaCrypt считается одним из самых успешных семейств вымогателей. Механизмы шифрования в TeslaCrypt реализованы весьма аккуратно, а выбор алгоритмов для этой цели указывает на то, что авторы имеют некоторую подготовку в области криптографии.

TeslaCrypt использует алгоритм AES-256 для шифрования файлов; тем не менее, в отличие от уязвимой версии TorrentLocker, он использует CBC в качестве режима работы шифра. В таком режиме каждый блок зашифрованных данных (часть зашифрованного сообщения фиксированного размера) выступает в качестве вектора инициализации IV при шифровании следующего блока, при этом все блоки шифруются одним и тем же ключом. Данный факт показывает, что это не поточный или другой аналогичный ему шифр, что делает используемую против TorrentLocker процедуру поиска ключа бесполезной в данном случае.

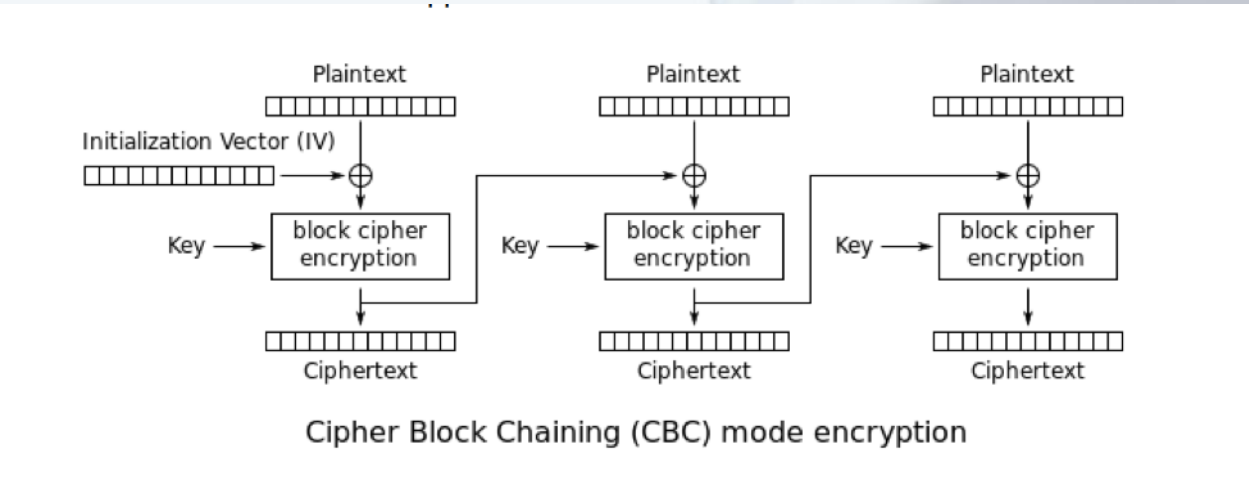

Рис. 7. Схема режима шифрования CBC.

Сам ключ AES, в случае с TeslaCrypt, шифруется с использованием алгоритма, похожего на схему Эль-Гамаля, а затем отправляется на C&C-сервер. Выбор алгоритма эллиптической криптографии Elliptic Curve Cryptography (ECC) авторами вместо RSA объясняется удобством для уклонения от обнаружения, так как ECC приводит к сокращению объема зашифрованного текста и занимает меньше времени для своего выполнения.

Тем не менее, авторы TeslaCrypt v2.2.0 (a.k.a. TeslaCrypt v8) совершили одну ошибку, которая является менее тривиальной в отличие от предыдущих. Как было указано на сайте BleepingComputer и в исследовании Talos, ключ восстановления данных (recovery key, RK), предназначенный для защиты ключа шифрования, генерируется как результат умножения общего секретного ключа (C2K), указанного управляющим C&C-сервером, и секретного ключа, используемого для шифрования файлов (FK).

RK = C2K * FK

Ключ восстановления хранится на скомпрометированной системе и позволяет осуществить процесс восстановления FK в том случае, когда C2K известен. Однако, авторы TeslaCrypt v2.2.0 ошибочно предположили, что они могут воспользоваться механизмом факторизации, лежащим в основе RSA.

Даже спустя десятилетия, исследователи не смогли обнаружить эффективный алгоритм вычисления простых множителей заданного числа (таких больших, какие используются в RSA). Тем не менее, это не означает, что задача факторизации не может быть решена в будущем.

Длина ключа восстановления данных не такая большая для того, чтобы сделать задачу факторизации невыполнимой. Следовательно, атака методом грубой силы является тем способом, который может использоваться для поиска ключа шифрования с последующей расшифровкой файлов.

Может ли шифрование использоваться против шифровальщиков?

Возникает вопрос, действительно ли шифровальщики не специализируются на шифровании уже зашифрованных файлов? Ответом является и «да» и «нет».

Очевидно, что можно еще раз зашифровать любые зашифрованные данные. По сути, данный факт лежит в основе идеи onion маршрутизации. Вкратце, onion маршрутизация предназначена для обеспечения конфиденциальности данных и их передачи через узлы или ноды маршрутизации. Представим себе схему из трех узлов, так что, когда пользователь захочет получить доступ к веб-сайту, запрос будет зашифрован трижды. В первый раз узлом 3, далее узлом 2, после чего узлом 1. При проходе запроса через каждый узел, он будет зашифрован ключом текущего узла, таким образом только узел 3 будет иметь на входе оригинальные данные еще до их прохождения по маршруту. Узел 3 выполняет запрос от имени пользователя и отправляет ответ обратно с использованием аналогичной схемы шифрования, которая применяется к запросу.

Для обеспечения эффективности и точности шифрования файлов, шифровальщики обычно фильтруют файлы по их расширениям; например, список файлов TorrentLocker в 2014 выглядел следующим образом. В случае попадания расширения файла в список, он подвергается шифрованию, в противном случае, отбрасывается.

С другой стороны, инструменты шифрования часто добавляют некоторое заранее определенное расширение к названиям зашифрованных файлов. Это интересный факт, поскольку мы не наблюдали примеры шифровальщиков, которые были бы нацелены на инструменты шифрования файлов. Из этого следует, что в случае использования одного из таких инструментов на компьютере, который подвергся компрометации со стороны шифровальщика, данные на нем не будут зашифрованы вымогателем.

Превентивные меры

Превентивные меры для защиты от вымогателей уже были опубликованы в разных источниках. В общем случае, следующие рекомендации будут полезны для пользователей.

- Следует использовать хороший антивирусный продукт с функцией проактивной защиты HIPS и защитой от фишинга.
- Обращайте внимание на скрытые расширения файлов и не переходите по ссылкам в сообщениях, полученных из неизвестных источников.
- Используйте резервное копирование файлов и выполняйте его как можно чаще.
- Избегайте монтирования директорий с файлами резервного копирования в свой компьютер. Вымогатель может подвергнуть шифрованию файлы, размещенные на съемном носителе, в облаке и другим местам, которые могут быть досягаемы из файловой системы ОС.

В случае уже произошедшей компрометации системы шифровальщиком, можно попытаться расшифровать файлы с помощью инструментов расшифровки, которые были разработаны исследователями безопасности. Компания ESET имеет в наличие несколько таких инструментов для расшифровки некоторых семейств вымогателей и их модификаций, что позволяет пользователю получить доступ к файлам без оплаты выкупа.

Кроме этого, отделение National High Tech Crime Unit голландской полиции и Europol European Cybercrime Centre, в сотрудничестве с компаниями в области безопасности, также предоставляют инструменты для оказания помощи жертвам шифровальщиков.