Привет. Введение Протоколу SSL уже много времени, и ему пора на покой. Тем более – замена ему достаточно давно уже есть. Выглядит она как протокол TLS, который вырос, возмужал, и готов к работе. Я попробую чуть-чуть пробежаться по тому, что и как для этого надо сделать в Windows Server и основных серверных приложениях. У данной […]

atraining.ru/beast-move-from-ssl-to-tls

2013-10-05T07:05:25+08:00

Привет.

Введение

Протоколу SSL уже много времени, и ему пора на покой. Тем более – замена ему достаточно давно уже есть. Выглядит она как протокол TLS, который вырос, возмужал, и готов к работе. Я попробую чуть-чуть пробежаться по тому, что и как для этого надо сделать в Windows Server и основных серверных приложениях.

У данной статьи есть обновлённая и расширенная версия – Бронируем TLS в Windows Server

Оглавление

- Краткая история вопроса – SSL

- Версии и преимущества TLS
    - Про TLS 1.0
    - Про TLS 1.1
    - Про TLS 1.2
    - Про TLS 1.2 и Windows XP SP3
    - Про TLS 1.2 и Windows 2003 SP2
- BEAST: как работает атака на SSL 2.0/3.0 и TLS 1.0
- Включаем TLS на Windows-системе
- Отключаем SSL на Windows-системе
- Закручиваем гайки: Включаем безопасное пересогласование TLS на Windows-системе
- Атака на SSL/TLS – THC-SSL-DOS
- Закручиваем гайки: настройки криптоалгоритмов на хосте
- Управляем настройками согласования SSL/TLS в браузерах
- Проверяем работу TLS 1.1 и 1.2
- Что делать, если у меня нет возможности включить TLS новых версий

Приступим.

Краткая история вопроса – SSL

Протокол SSL был разработан фирмой Netscape, достаточно давно. Я осознанно пропущу исторические подробности, так как они сейчас уже не особо интересны. В его задачи входило следующее:

- Обязательность подтверждения подлинности сервером
- Опциональная проверка подлинности клиента
- Совместная генерация случайного сеансового ключа
- Поддержка различных симметричных алгоритмов для шифрования данных
- Поддержка различных алгоритмов хэширования для реализации проверки целостности через MAC

Первая версия SSL не особо показывалась публике, отсчёт рабочих версий можно начинать с SSL 2.0 (1995й год). Эта версия была первой, которая эксплуатировалась в production, и достаточно оперативно она была доработана до SSL 3.0. Заметим, что хотя это произошло достаточно давно – в 1996 году – стандарт от этого не стал общим и открытым; он оставался стандартом, разработанным конкретной фирмой Netscape, и для его использования в ряде случаев нужна была лицензия. Опубликован IETF он был совсем недавно, в августе; RFC 6101 описывает то, что 15 лет уже стандарт де-факто для защиты сессий множества приложений. Но по сути, с ним уже давно пора прощаться; TLS существует годы (с 1999, если быть точнее), а достаточно безопасная на данный момент версия 1.1 – с 2006 года. Пора, пора.

Версии и преимущества TLS

На данный момент есть три версии протокола TLS: 1.0, 1.1 и 1.2. Они, соответственно, имеют внутренние идентификаторы версии 3.1, 3.2 и 3.3, поэтому иногда называются SSL 3.1, SSL 3.2 и SSL 3.3. Все эти версии поддерживаются как клиентским, так и серверным ПО Microsoft (начиная с IIS 7.5 и IE9), надо только включить. Про клиентское ПО (типа того же Internet Explorer) я уточню ниже, в отдельном пункте – на то оно и клиентское, чтобы включение поддержки данного функционала было бы там несложной операцией. Основное в статье – про серверные вопросы. Давайте чуть разберёмся в функционале версий протокола TLS, а после – включим их поддержку, отключим SSL и закрутим гайки.

Про TLS 1.0

Версия 1.0, по сути, является базовой, и представляет собой доработанный и открытый для всех, а не только для Netscape, вариант SSL 3.0. Хотя, надо отметить, напрямую TLS 1.0 и SSL 3.0 не совместимы; TLS 1.0 “умеет” работать в режиме совместимости с SSL, но именно в режиме, а не “идентично”, как иногда можно прочитать.

Например, у SSL 3.0 есть встроенная в протокол неприятность – половина master key будет создаваться из MD5-хэша достаточно предсказуемых данных, что может привести (учитывая текущее положение MD5) к успешной атаке на коллизию (в результате станет известна половина бит ключа, которым будет шифроваться сессия, а это практически равнозначно компрометации процесса). У TLS 1.0 это поправили, и схема усложнена – берутся и MD5 и SHA-1 хэши, после чего хог’ятся, что сводит возможность вышеуказанной атаки к нулю.

Кстати, из-за этого момента в алгоритме – завязанности ключевой части процесса на MD5, SSL 3.0 не является FIPS140-2 совместимым и при включении FIPS140-2 не согласовывается.

Про TLS 1.1

В версии TLS 1.1 делаются небольшие изменения – в частности, меняется логика задания IV (вектора инициализации), для защиты от атак, имеющих своей целью слабость реализации метода CBC в TLS 1.0, улучшается обработка ошибок и переподключения, а также вносятся другие мелкие изменения, не являющиеся фундаментальными с точки зрения использования протокола. Но по сути, это самая минимальная версия TLS, для которой отсутствуют известные атаки, поэтому её использование – это правильный выбор.

Про TLS 1.2

В TLS 1.2 же список добавлений относительно велик – в нём будут многочисленные улучшения поддержки новых криптографических протоколов, отказ от потенциально уязвимых моментов (например, для процесса генерации псевдослучайного числа –

замена MD5/SHA-1 на SHA-256 либо на явно указанную хэш-функцию), ужесточение многих проверок, обязательность реализации AES и отключение одиночного DES и IDEA, отмена обязательности совместимости реализации с SSLv2 и подобное.

Для нас ключевым является разумное повышение уровня безопасности, поэтому мы будем нацеливаться на TLS 1.1. Не потому что TLS 1.2 плохой, а потому что если “закрутить гайки” только на TLS 1.2, не все клиенты смогут корректно подключаться. Если бы все браузеры с криптографической точки зрения были бы уровня IE9, то можно было бы и TLS 1.2 включить, а остальное объявить старым, но реальность состоит в том, что нам надо хотя бы отсечь гарантированно старые и уязвимые протоколы.

Про TLS 1.2 и Windows XP SP3

Например вот один момент, который надо учитывать при поддержке TLS 1.2. Дело в том, что стандартным методом проверки целостности (MAC – Message Authentication Code), используемым в TLS 1.2, является SHA-2. SHA-2 – это семейство алгоритмов, состоящее из SHA-224, SHA-256, SHA-384, SHA-512, и подобных. Соответственно, “натуральная” поддержка этих алгоритмов приходит только в ядре NT 6.0, с новой версией CryptoAPI. В распространённой же Windows XP данная поддержка (т.е. просто понимаение, что есть SHA-2) появляется только в Service Pack 3, притом поддерживается только 3 варианта SHA-2 – SHA-256, SHA-384, SHA-512. То есть, говоря проще, XP SP2 нормально с TLS 1.2 работать не может никак. Вообще. Потому что будет `NTE_BAD_ALGID`.

Про TLS 1.2 и Windows Server 2003 SP2

К сожалению, ситуация с поддержкой TLS 1.2 на Windows Server 2003 SP2 хуже, чем на XP. В последний Service Pack для 2003го сервера поддержка SHA-2 не вошла. Поэтому попытка проверить целостность в TLS 1.2 при помощи одного из алгоритмов семейства SHA-2, да и, к примеру, посчитать хэш у x.509 сертификата или *.crl приведёт к ошибке. Поддержка SHA-2 аналогичная XP SP3 реализована через патч 938397. Но это ещё не всё. Ведь приключения, связанные с тем, что у NT 5.1 / NT 5.2 есть CAPI2, а у NT 6.0 – CNG, только начинаются. В случае, если в Вашей сети есть Windows XP и/или Windows Server 2003, и есть служба Certification Authority на Windows 2008 и старше, то даже в случае установленного патча 938397 они не смогут корректно запрашивать сертификаты у этого CA. Чтобы они могли это делать, есть патч 968730. Он и добавляет частичную поддержку SHA-2, и учит CAPI2 дружить на уровне запросов сертификата с CNG. Полностью перекрывая собой предыдущий патч. Обязательно установите 968730 на Windows XP SP3 / Windows Server 2003 SP2, если ещё не сделали это.

BEAST: как работает атака на SSL 2.0/3.0 и TLS 1.0

Эта достаточно нашумевшая атака (ну, благодаря прессе, у неё статус “Конец Интернета Почти Вот-Вот”), на самом деле, устроена достаточно просто. Суть в том, что во всех протоколах до TLS 1.1 вместо генерации нового случайного IV (для каждого нового TLS message), использовался последний блок предыдущего TLS message. BEAST пользуется этим, чтобы значительно упростить процесс атаки на ключ – в частности, получив доступ к сессии, он может значительно упростить себе процедуру перебора сессионного ключа, влияя на состав нового вектора инициализации. В TLS 1.1 халтурить перестали, поэтому данная атака там просто не работает. Если попробовать рамочно описать алгоритм атаки, можно это сделать так.

- Пользователь инициирует SSL-сессию к ресурсу (например, к веб-серверу). В этой сессии он постоянно передаёт какой-то элемент (например, жмёт на кнопку Like). Ну т.е. сессия одна, а он ходит по сайтам и делает что-то, что сопровождается хотя бы частично совпадающим обменом данными.
- Например, предположим, что согласовался протокол AES-128 (заметьте – стойкость самого протокола не критична, я его выбрал просто для удобного примера). Его блок будет равняться 128 / 8 = 16 байт.
- Атакующий участвует в обмене трафиком так, что может добавлять свои данные перед данными пользователя. Вариантов много – например, доп.заголовок в HTTP, или модификация исходных данных на уровне источника. Суть в том, чтобы атакующий мог добавить свои данные так, чтобы получился блок “Добавленные_данные_известные_атакующему + нормальные_данные_неизвестные_атакующему”. Т.е. исходное в атаке – это MITM. Если вы думаете, что условий дофига и это всё малореально, просто представьте себе обычного пользователя, который игнорирует антивирус, потому как, к примеру, верит рекламе “В нашей ОС нет вирусов, т.к. у неё красивые скруглённые уголочки да и цена намекает, что качество ну просто обязано быть”, который хочет в онлайне что-нибудь оплатить по карте и видит успокаивающую надпись “Protected by SSL 2.0/3.0 128bit cipher”.
- Атакующий начинает работать. Он вкидывает такие блоки данных в поток, чтобы последним получился блок, у которого 15 байт – добавленные данные, а единственный оставшийся – секретные. После пробует тестово расшифровывать этот блок перебором – перебирать-то интересно, т.к. из 16 байт 15 известны. А подбор 2^8 – это уже очень перспективно, учитывая, что в случае выигрыша можно будет вскрыть всю сессию (ведь никто не мешает впрок отправлять куда-нибудь полный дамп сессии, а после, в случае успеха метода, расшифровать всё целиком).
- Вскрыли 1 байт? Теперь будем подгадывать так, чтобы наших байт в блоке было 14, плюс один известный, плюс один пока что неизвестный

Очевидно, что при реализации этой штуки через ботсеть можно наловить очень много полезного. Обращу внимание, что атака не зависит от используемого симметричного алгоритма, а лишь использует то, что в SSL/TLS используется схема CBC, а не ECB. AES-128, как понятно, я выбрал, чтобы блок был 16 байт. Как

понимаете, в случае AES-256 время вырастет не на 2^128, как в случае линейного криптоанализа, а всего лишь в два раза – надо будет вскрыть не 16, а 32 байта. Ну, а в случае DES с блоком в 64 бита, всё только грустнее. Поэтому надо защищаться.

Включаем TLS на Windows-системе

Сразу обращаю внимание – этот метод включает TLS на уровне операционной системы. Продукты в большинстве своём просто пытаются согласовать варианты протоколов SSL/TLS, начиная “с верхнего доступного”, явных настроек у них обычно нет. Для отключения TLS 1.0 и включения поддержки TLS 1.1 и TLS 1.2 необходимо проделать следующие действия – зайти в реестр, открыть ключ `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\Server` создать в нём значение `DisabledByDefault` и выставить его в единицу, а в ключах `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.1\Server` `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server` выставить то же самое значение в нуль. Это нужно сделать обязательно, т.к. отсутствие этого ключа равняется “отсутствию отмены запрета на поддержку TLS 1.x”. В случае необходимости в явном виде управлять данными протоколами на уровне клиента – всё то же самое, но вместо раздела \Server надо указать \Client.
Примечание: Имеет смысл сделать шаблон групповой политики, и через него в явном виде разрешить на всех машинах в домене поддержку TLS старших версий – что для клиентов, что для серверов. Это уж точно будет лучше, чем дефолтная ситуация с наличием только SSL 3.0 / TLS 1.0.

Теперь пора отключить небезопасные протоколы.

Отключаем SSL на Windows-системе

Согласно RFC 6176 от марта 2011 года, все, кто думают о безопасности, и хотят называться TLS-серверами и клиентами, должны вести себя так:

- TLS-клиенты обязаны никогда не отправлять в сообщениях CLIENT-HELLO версии ниже 3.0
- TLS-сервера обязаны сразу прерывать попытку подключения, если противоположная сторона сообщает о том, что максимально поддерживаемая ей версия протокола – 2.0

Вот так, достаточно четко и строго.

Давайте сделаем всё так же.

Для этого надо будет отключить PCT 1.0 (т.к. он тоже не TLS, если что) и SSL 2.0. SSL 3.0 отключается абсолютно аналогично.

Как отключить SSL 2.0/3.0 на Windows-системе

Нам надо зайти в ключ реестра

`HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0\Client` и поставить там значение `DisabledByDefault` (вида DWORD32) в единицу. Аналогично – для ключа `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0\Server` Если сразу хотите отключить и SSL 3.0, то поступите по аналогии, создав (если его нет, что часто бывает) ключ с именем SSL 3.0 на том же уровне иерархии, где находится раздел с SSL 2.0, и создав соответствующие подключи \Client и \Server.

Как отключить PCT 1.0 на Windows-системе

Тут чуть иначе. Если у Вас Windows Server 2003 – этот протокол уже выключен. Если 2008 – то даже не включится. Но на всякий случай – знайте, как это делается. Надо зайти в ключ `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\PCT 1.0\Server` и добавить в него значение `Enabled` (вида DWORD32), выставив его в нуль. Вроде как всё. PCT 1.0 убили, все SSL тоже, TLS 1.0 тоже. Делайте это всё аккуратно, очень тщательно проверяя, что ПО после этого функционирует. Учтите, что при закручивании гаек в плане “Давайте выключим весь SSL на клиентах” – т.е. не при добавлении старших версий TLS, а при явном запрете младших – с гарантией часть внешних сервисов будет недоступна. Например, большинство социальных сетей, которые декларируют “Доступ к нам теперь по HTTPS, а, значит, безопасен!”, на самом деле крайне экономят силы, включая согласование минимальных версий и самых простых криптоалгоритмов. Их можно понять; 99.9% клиентов глубоко пофиг, DES там или 3DES, а вот что вычислительно задача станет в 3 раза сложнее – очевидно. По моему личному опыту, при “замораживании” на клиенте TLS 1.1 и выше, например, не работает https://twitter.com/ . Т.е. будьте осторожны и тестируйте. Вполне возможно, что вполне даже рабочие ресурсы – типа электронной торговой площадки – будут использовать старые алгоритмы. Ну, а теперь давайте закручивать гайки.

Закручиваем гайки: Включаем безопасное пересогласование TLS на Windows-системе

К стандартам семейства TLS есть интересные дополнительные механизмы, не вошедшие в основные спецификации. Данные механизмы известны гораздо меньше, чем что-то вида *“ну ё, TLS же новый, SSL старый, значит включаешь TLS и безопаснее становится”*, но не в моих правилах подходить к вопросу с такой стороны.

Безопасное пересогласование TLS описывается в стандарте RFC 5746 и решает проблему безопасности, которая возникает в случае работы классического TLS 1.2, который по RFC 5246.

По сути, Windows-хост может работать в плане этого RFC в двух режимах – в режиме совместимости (т.е. допускать и небезопасное, “классическое” пересогласование TLS 1.2), и в “безопасном”, допуская только пересогласование в новом формате. По умолчанию, работа идёт в режиме совместимости. Это не всегда полезно, поэтому надо знать, как включать только безопасное пересогласование TLS 1.2.

Включаем поддержку TLS Renegotiation Indication Extension

Мы будем работать с ключом реестра

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\SCHANNEL`

В нём будет необходимо создать следующие ключи: `AllowInsecureRenegoClients`, `AllowInsecureRenegoServers`, `UseScsvForTls`, `DisableRenegoOnClient`, `DisableRenegoOnServer`. Все – обычные DWORD 32. Теперь подробнее про каждый.

Параметр `AllowInsecureRenegoClients` будет обозначать, можно ли клиентам, подключающимся к данному хосту, использовать небезопасный вариант пересогласования. Обратите внимание – это серверная настройка; т.е. она анализируется, когда какой-либо сервис на этом хосте “слушает” входящие подключения по TLS. Например, IIS. Если хотите, чтобы можно было подключаться всем – поставьте единицу, если только “умным” клиентам, поддерживающим RFC 5746 – то нуль.

Примечание: Патч, который добавляет поддержку этого механизма, вышел в августе 2010 года. То есть, если у Вас XP или выше, и на ней установлены обновления, то поддержка есть.

Параметр `AllowInsecureRenegoServers` будет обозначать, можно ли будет с этого хоста подключаться к серверам, которые не поддерживают механизм безопасного пересогласования. Если параметр равен нулю – клиенты будут прерывать фазу согласования TLS, если сервер ведёт себя некорректно. Если хотите, чтобы можно было подключаться к любым TLS-серверам, вне зависимости от реализации на них поддержки RFC 5746 – поставьте единицу.

Параметр `UseScsvForTls` заслуживает отдельного обсуждения. Дело в том, что в RFC 5746 указывается, что данное расширение необходимо упаковывать в сообщение “ClientHello” протокола TLS. Но это может быть причиной проблем, потому что некоторые сервера – заметьте, именно сервера, т.е. это клиентская настройка – которые не поддерживают данный стандарт, не просто не смогут переключиться на новый механизм, а просто не смогут обработать этот неизвестный им подвид сообщения ClientHello. Это, например, Vista без SP. Чтобы подстраховаться от ситуации, когда “новый” клиент посылает “новый вариант” ClientHello и серверная сторона рвёт связь, надо установить этот параметр в

единицу. В этом случае клиент просто отправит т.н. “Signaling Cipher Suite Value” (как раз SCSV). Ставьте этот параметр только в случае, когда Вам реально нужно подключаться к системе, которая “сбрасывает” сессию после ClientHello.

Можно и масштабнее. Если параметры выше описывали варианты поведения при пересогласовании TLS, то следующие будут включать-выключать поддержку пересогласования как такового.

Если параметр `DisableRenegoOnClient` есть и не равен нулю, то клиент со своей стороны не будет пробовать проводить пересогласование TLS (например, периодически) и не будет отвечать на запросы пересогласования (будет не рвать сессию, а именно игнорировать). Если же параметр равен нулю (или отсутствует), то всё ОК – клиент будет поддерживать пересогласование и пробовать делать его сам.

Если параметр `DisableRenegoOnServer` есть и не равен нулю, то сервер со своей стороны не будет пробовать проводить пересогласование TLS и не будет отвечать на запросы пересогласования (тоже будет не рвать сессию, а именно игнорировать). Если же параметр равен нулю (или отсутствует), сервер будет поддерживать пересогласование и пробовать делать его сам.

Атака на SSL/TLS – THC-SSL-DOS

Достаточно предсказуемая атака, в основе которой лежит идея, что крайне малыми силами со стороны клиента (по сути, просто запросив) можно заставить сервер выполнить математически относительно сложную задачу по перегенерации ключевой информации. В своей практике я сталкивался с таким в 2004 году, в случае с ipsec; один талантливый админ поставил настройки смены ключей так, что они менялись через каждые 100 килобайт (он хотел мегабайты, просто ошибся). В результате сеть работала, но медленно – ipsec в основном занимался IKE/ISAKMP задачами, а не трафиком.

Атака серьёзна – при полутысяче запросов в секунду (выполнимо с домашней машины при наличии обычного широкополосного доступа на 4-8 мегабита) полностью “ложится” процессор уровня Intel Xeon E5645. 100%го способа защиты от атаки сейчас нет. Поэтому надо максимально уменьшить её вероятность.

Для борьбы можно использовать следующие методы:

1. Разгрузку SSL-задач при помощи аппаратных ускорителей
2. Отключение пересогласования TLS
3. Ограничение количества TLS-сессий – как вообще в сумме, так и с одного source ip
4. Упрощение схемы генерации ключей

Защита от THC-SSL-DOS путём покупки SSL-ускорителя

Метод не оптимален, но всё же в ряде случаев изменит ситуацию. Суть в том, чтобы SSL/TLS сессии терминировались не на целевом сервере, а на выделенном устройстве, которое ощутимо быстрее выполняет SSL/TLS – задачи. Проблема в том, что все такие ускорители обычно ускоряют саму работу SSL/TLS – то есть симметричное шифрование трафика и проверку его целостности, а операции вида “генерация и смена ключей”, в силу их относительно малого процента в доле нагрузки при нормальной сессии делаются на таких appliance программно. Специализированного appliance, которое умеет очень быстро именно генерить новые сеансовы ключи для TLS, а не шифровать трафик, видимо, не существует в природе. Увы.

Защита от THC-SSL-DOS путём отключения пересогласования TLS

Это то, что Вы реально можете сделать. Учтите, это может повлиять на совместимость с клиентским ПО, которое в обязательном порядке хочет время от времени менять ключи сессии. Я проверял и не нашёл такого ПО среди обычно используемого в сетях на базе продуктов Microsoft (тестировались Exchange 2010 SP1, Sharepoint 2010 SP1, трафик DC/GC поверх SSL, TMG 2010 SP1). Мной не было найдено аномалий поведения, разрывов TLS-сессий по причине rekeying и прочего. Поэтому Вы можете промотать эту статью чуть выше (до предыдущего пункта) и выставить в единицу параметр `DisableRenegoOnServer`.

Примечание: В принципе, можно использовать любое значение не равное нулю, поэтому единица предлагается для примера

Защита от THC-SSL-DOS путём ограничения количества TLS-сессий – как вообще в сумме, так и с одного source ip

Если Вы отключили пересогласование TLS, то Вы отсекли одну из возможностей данной атаки; впрочем, опубликованный вчера эксплойт как раз её и использовал. Но существует альтернативный вариант – когда Ваш сервер перегружают не частым пересогласованием сессии, а многими фейковыми сессиями. Здесь Вы можете бороться только на сетевом уровне – ограничивая количество сессий с одного адреса, добавляя тайм-аут между установкой новых TLS-сессий, и лимитируя общее количество TLS-сессий у сервера вообще. Лучше применить все 3 этих подхода – например, сделать тайм-аут между TLS-сессиями хотя бы 10 ms (то есть не более 100 новых сессий в секунду до сервера), и разумно ограничить допустимое количество сессий с 1го адреса.

Упрощение схемы генерации ключей

Это – ожидаемое решение со стороны реализации TLS. Навряд ли Вы сделаете это сами. :)

Закручиваем гайки: настройки криптоалгоритмов на хосте

Цель – отключить слабые алгоритмы шифрования. Чтобы они просто не участвовали в процессе согласования. Потому что на данный момент, допустим, наличие в списке поддерживаемых алгоритмов RC2 40bit – это очень плохо, и это необходимо убирать незамедлительно.

Алгоритмы, используемые в SSL/TLS, можно модифицировать двумя основными способами – через групповые политики или через реестр. Первый способ, так как является штатным, предпочтительнее. Выполняется он следующим способом.

Как через групповую политику изменить список поддерживаемых SSL/TLS криптоалгоритмов

Откройте объект групповой политики, через который Вы решите это сделать. Если настраиваете локальный хост – `gpedit.msc`. Выберите там раздел для компьютера, потом административные шаблоны, потом сеть, потом SSL Configuration Settings. Откройте данный параметр и сделайте следующее.

1. Скопируйте в Блокнот все cipher suites, которые там есть.
2. Удалите те, которые включают в себя согласование нестойких алгоритмов и методов (например, RC2, RC4, DES, MD5 и прочее. всякие ужасы вида TLS_RSA_WITH_NULL_MD5 Вам же не нужны, правда?).
3. Превратите список в одну большую строку, которая состоит из cipher suites, разделённых запятыми, и не содержит больше ничего. Пробелов тоже.
4. Вставьте эту строку в окно данной настройки групповой политики.

Учтите, что не всё, что поддерживается системой, присутствует в этом списке. И наоборот – данный список не прикажет системе начать поддерживать неизвестные ей криптоалгоритмы. Например, если у Вас есть системы на Windows XP / Windows Server 2003, убедитесь, что на них установлено обновление KB 948963, которое добавляет поддержку AES-128 и AES-256, необходимых для TLS.

А вот как можно изменить этот список через реестр.

Как через реестр изменить список поддерживаемых SSL/TLS криптоалгоритмов

Откройте редактор реестра. Ваша задача – ключи вида:

`HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\`*`идентификатор_алгоритма размерность_ключа1/размерность_ключа2`*

где идентификатор алгоритма – это, например, RC4, а размерность ключа – например, 128. Понятное дело, что данных комбинаций фиксированное число, поэтому для отключения всех из них в явном виде надо забежать в MSDN и найти эти комбинации. Например, для старых симметричных алгоритмов семейства RCx это будут:

- RC4 128/128

- RC2 128/128
- RC4 64/128
- RC4 56/128
- RC2 56/128
- RC4 40/128
- RC2 40/128

Во всех этих ключах надо будет создать параметр `Enabled` (как обычно, DWORD32) и выставить его в нуль.

Интересным моментом является отключение возможности установить “пустой” SSL – без криптоалгоритма. Ну, типа как в PPP можно вообще без фазы авторизации, так и тут – есть такая тонкость. Чтобы эту тонкость тоже убрать, надо зайти в ключ:

```
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\NULL
```

и опять же создать там `Enabled` со значением 0x0.

Отключаем использование хэш-функции MD5 в SSL/TLS

И даже это можно. Через групповую политику нет (только вручную удалив из полного списка все suites с упоминанием MD5), а через реестр – пожалуйста.

Учтите только, что данная настройка – лидер по количеству приключений после её применения. Классика – умирающий SharePoint 2010; он использует MD5 для генерации внутренних идентификаторов безопасности, поэтому не найдя её очень страдает.

Заходим:

```
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Hashes\MD5
```

и отключаем аналогично – `Enabled` в 0x0. Заодно и MD4 можно, он в принципе может встретиться в IPSec, но если не используется в реальности – имеет смысл его отключить аналогичным способом.

Примечание: Если захотите что-нибудь из этого явно включить обратно, не стирая ключ, есть нюанс: Enabled надо будет ставить не в единицу, а в 0xFFFFFFFF.

Проверяем работу TLS 1.1 и 1.2

Если Вы выяснили, что после отключения поддержки TLS 1.0 вы никуда не можете зайти, и подозреваете, что браузер сломался, есть отличный тестовый сайт:

https://tls.woodgrovebank.com/

Это специальный сайт, который зарегистрирован на одну из вымышленных и использующихся в курсах Microsoft организаций – Woodgrove Bank, а по сути – голый IIS с самоподписанным сертификатом, который поддерживает все виды SSL/TLS и нужен для диагностики. Зайдите на него, установите сертификат (их, кстати, несколько – специально для проверки поддержки различных криптоалгоритмов подписи) и проверите, что и как работает.

Есть второй вариант – https://www.mikestoolbox.net/. Тут возможностей поменьше, но зато быстрее и нагляднее видно – какой протокол поддерживается, что согласовалось.

Управляем настройками согласования SSL/TLS в браузерах

Internet Explorer

Если Вы дочитали до этого места и всё поняли, но не знаете, как включить TLS в IE, то это, по сути, уникальная ситуация. Но тем не менее. Для включения поддержки TLS старших версий необходимо зайти в меню Internet Options -> Advanced, там в списке промотать до раздела Security и сделать соответствующие изменения (как минимум отключить SSL 2.0 и включить TLS 1.1 и TLS 1.2, остальное – на усмотрение).

Google Chrome

У меня под рукой кроме IE только хром, но, как известно из фольклора, оно не браузер. Что хорошо подтверждается тем, что никакого TLS 1.1 и TLS 1.2 в доступной сейчас рабочей версии (14.0.835.137) нет. Т.е. можно сказать проще – на сегодняшний день, 2.10.2011, все поддерживаемые хромом виды безопасных соединений на основе SSL/TLS уязвимы для сентябрьской атаки и данный инструмент не имеет смысла использовать для, допустим, финансовых транзакций, онлайн-платежей и прочих security sensitive штук.

Opera

Поставил оперу 11.51. Пожалуй, тут лучше всех сделано – можно не только выставить нужные протоколы (доступен выбор из SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2), но и нужные комплекты алгоритмов. Делается это зайдя в меню, там – Settings -> Preferences -> Advanced -> Security -> Security Protocols -> Details. Правда, сделано плохо, потому что при включенном варианте “TLS 1.0 + 1.1 + 1.2” согласовывать начинает с TLS 1.0. Специально перепроверил, плюс посмотрел через netmon – удивительно, реально начинает с 1.0, согласовывает и успокаивается. Т.е. если включить все эти 3 протокола в Internet Explorer, то на тестовых сайтах клиент будет определяться как “TLS 1.2 compatible”, а в случае оперы – “TLS 1.0”. Плохо, по сути перечеркивает всё преимущество тонкой настройки – т.е. ну поддерживает она 1.2, что с того-то, если согласовывать пробует

1.0 для начала. То есть или надо вручную выключать TLS 1.0 (тогда большинство публичных https-сайтов типа gmail или facebook отвалятся), или сидеть с уязвимой версией.

После таких “мелочей” люди удивляются, почему IE является корпоративным стандартом.

Что делать, если у меня нет возможности включить TLS новых версий

Если у Вас нет возможности “правильно” обеспечить себе безопасность от атаки BEAST’а, можно использовать тонкий трюк.

Надо зайти в вышеуказанный параметр групповой политики, посвящённый последовательности согласования криптографических комплектов, и выставить единственным TLS_RSA_WITH_RC4_128_MD5. Тонкость в том, что атака BEAST не работает в случае применения данного криптоалгоритма (RC4). Безусловно, в таком случае нельзя отключать RC4 и MD5, а также включать FIPS140-2. Да, конечно, RC4 хуже, чем AES, но если нет возможность включить AES, то чем не вариант? В случае клиентских браузеров это позволяет обезопасить от атаки IE7 и выше.

Краткие рекомендации

- Включите через групповую политику протоколы TLS 1.1 и 1.2 для всех клиентов и серверов в домене.
- Обязательно отключите SSL 2.0 на всех клиентах и серверах.
- Обязательно отключите SSL 3.0 на всех серверах, используемых для внутренних сервисов (DC, порталы, CA, всё подобное).
- Осторожно отключите SSL 3.0 и TLS 1.0 на всех клиентах, проверив, все ли требуемые для работы внешние сервисы будут доступны в этом случае.
- Включите на всех клиентах и серверах пересогласование TLS.
- Включите его (пересогласования) строгое соответствие RFC 5746 – не имеет смысла устанавливать “высокотехнологичный” TLS последней модели, и при этом не брать во внимание well-known security hole.

Более жесткое закручивание гаек – по вкусу.

Заключение

Я надеюсь, что эти несложные действия ощутимо помогут Вам в том, чтобы Вы могли читать про штуки, аналогичные BEAST, только в новостях.