# ЛЕКЦИЯ 5 - МЕТОДЫ ЗАЩИТЫ IOS

## ВВЕДЕНИЕ

Многие пользователи доверяют официальным приложениям для iPhone, iPad от компании Apple, но с недоверием относятся к тем программам, которые скачивают из App Store. Данные опасения не беспочвенны, сегодня мы с помощью телефона не только отправляем SMS и электронные письма, но и оплачиваем счета и пользуемся банковскими услугами и корпоративной информацией. Именно поэтому в Apple уделяют много внимания безопасности iOS-приложений.

## 5.1. БАЗОВЫЕ ПОНЯТИЯ О БЕЗОПАСНОСТИ В IOS

Основным документом, содержащим информацию о системе безопасности iOS, является “iOS Security”. Основываясь на этом документе Apple, систему безопасности iOS можно разделить на 4 большие части:

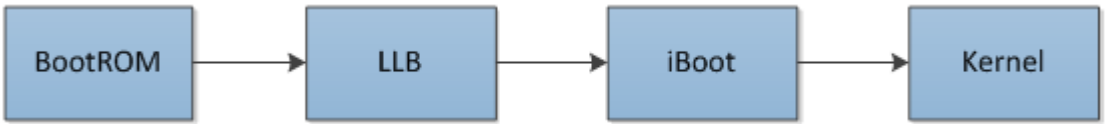
- базовые механизмы безопасности;
- шифрование и защита данных;
- сетевая безопасность;
- доступ к устройству.

### 5.1.1. Базовые механизмы безопасности

Принцип безопасности заложен в саму основу iOS. В iPhone, iPad и iPod touch действует несколько уровней безопасности. Низкоуровневые аппаратные функции защищают устройство от вредоносного ПО, когда как высокоуровневые функции iOS обеспечивают безопасный доступ к пользовательским данным и корпоративной информации, а также предотвращают несанкционированное использование устройства.

Благодаря тесной взаимосвязи программного и аппаратного обеспечения в устройствах под управлением iOS, каждый шаг, начиная от нагрузки системы и заканчивая установкой приложений, анализируется с точки зрения безопасности и эффективности использования ресурсов. Целостность системы безопасности напрямую зависит от целостности и надежности ядра iOS – XNU.

Каждый шаг загрузки содержит компоненты, которые имеют цифровую подпись Apple. На изображена цепочка доверенной загрузки:

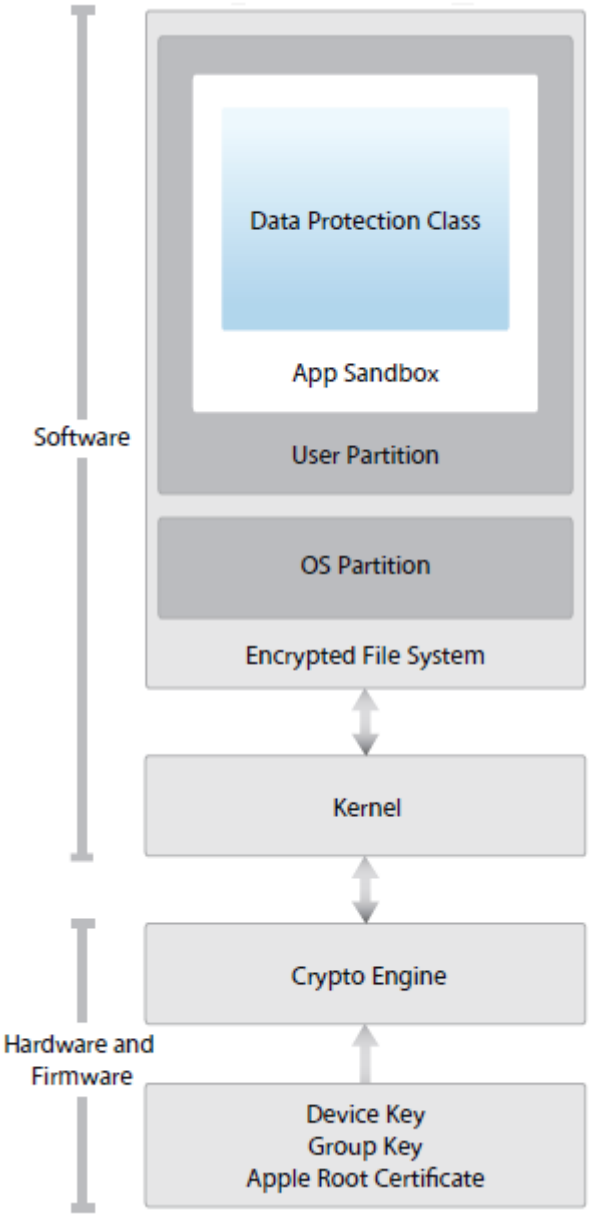



При включении устройства процессор немедленно начинает исполнять код из read-only памяти, так называемой BootROM. Код зашивается в чип еще при производстве, поэтому к коду имеется безоговорочное доверие. BootROM также содержит сертификат корневого центра сертификации Apple. С помощью сертификата BootROM проверяет цифровую подпись низкоуровнего загрузчика (Low-Level Bootloader - LLB). LLB затем проверяет подпись и запускает следующий загрузчик – iBoot. И, наконец, iBoot проверяет и запускает ядро.

Цепочка доверенной загрузки выполняет две функции. Во-первых, она гарантирует, что низкоуровневое ПО не подверглось никаким несанкционированным изменениям; и во-вторых, позволяет iOS загружаться только на устройствах Apple.

Если какому-либо шагу в процессе не удается проверить или запустить следующий шаг, то загрузка прекращается, устройство входит в режим восстановления (recovery mode) и на экране отображается надпись “Connect to iTunes”. Если же сбой происходит уже на первом шаге, то есть если BootROM не удалось проверить и загрузить LLB, то устройство входит в режим DFU (Device Firmware Upgrade – обновление прошивки устройства). В обоих случаях необходимо подключить устройство к iTunes и вернуться к заводским настройкам.

В целом архитектура безопасности iOS выглядит так:



### 5.1.2. Персонализация системного ПО

Apple периодически выпускает обновления безопасности, о которых оповещает пользователя на самом устройстве или через iTunes. Apple делает все возможное, чтобы обновления безопасности были установлены на устройствах пользователя как можно быстрее.

Теперь представим себе такую ситуацию: нарушитель получил доступ к устройству и хочет понизить версию системы, чтобы использовать уязвимости, которые исправлены в новой версии. Для предотвращения подобных ситуаций, в iOS существует механизм персонализации системного ПО (System Software Personalization).

Обновления iOS можно установить либо с помощью iTunes, либо через Wi-Fi. В первом случае скачивается и устанавливается полная копия iOS. При обновлении по Wi-Fi скачиваются только необходимые “дельта-файлы”.

Во время инсталляции или обновления iOS iTunes или само устройство (в случае обновления по Wi-Fi) подключается к серверу обновления Apple (gs.apple.com) и посылает следующую информацию:

1. список криптографических параметров каждого обновления;
2. случайное число;
3. уникальный идентификатор устройства (ECID).

Получив данные от клиента, сервер проверяет, возможна ли установка нового обновления на устройство пользователя. Если установка возможна, то сервер отправляет специальное разрешение пользователю. Только после успешной проверки пользователь может продолжить их установку.

Как уже упоминалось, BootROM содержит сертификат корневого центра сертификации Apple, поэтому цепочка доверенной загрузки сможет проверить электронную подпись сервера обновлений. Механизм персонализации также гарантирует, что не удастся скопировать старую версию iOS с одного устройства на другое. Благодаря случайному числу, нарушитель не сможет сохранить ответ сервера, чтобы в будущем использовать его для понижения версии системы (атака повтором).

### 5.1.3. Подписание кода приложений

После загрузки iOS следит за тем, какое пользовательское приложение может быть запущено, а какое нет. Для гарантии того, что все приложения поступили из проверенного источника, и что в них не было внесено никаких изменений, iOS требует, чтобы каждый исполняемый код был подписан. Приложения, поставляемые вместе с устройством (например, Mail или Safari), подписываются самой Apple. Обязательное подписание кода расширяет рамки действия принципа доверия с уровня операционной системы на уровень приложений и препятствует выполнению вредоносного или самомодифицирующегося кода.

Для публикации своих приложений сторонние разработчики должны зарегистрироваться в Apple и присоединиться к программе iOS Developer Program. Перед выпуском сертификата Apple проверяет личность каждого разработчика. Сертификат дает разработчикам возможность подписывать приложения и размещать их в Apple Store. Приложение также проверяется Apple на предмет нормального функционирования, возможных багов и.т.п.

Но и тут не обходится без прецедентов. Уследить за всеми приложениями не представляется возможным, и иногда за звучным названием (например, Microsoft Word 2012) и немаленькой ценой скрывается совершенно бесполезное приложение или приложение, не выполняющее заявленных функций. Нерадивые разработчики также

могут создавать искусственные отзывы, чтобы искусственно поднять рейтинг своим приложениям.

Очень часто перед организациями встает необходимость писать программы для внутреннего использования и распространять их среди своих рабочих. Для этого организации могут присоединиться к программе iOS Developer Enterprise Program (iDEP). После проверки организация может получить специальный профайл, который позволяет разрабатывать приложения для внутреннего пользования. Чтобы запускать корпоративное приложение, пользователь должен на свое устройство установить профайл. Это также гарантирует, что доступ к приложению будут иметь только работники компании.

В отличие от других мобильных платформ, iOS не позволяет пользователям установить потенциально вредоносное ПО или исполнять код из сомнительного источника. Каждый раз перед загрузкой исполняемых страниц в память операционная система производит проверку подписи кода. Проверка позволяет выявить нежелательные изменения в коде с момента установки или обновления приложения.

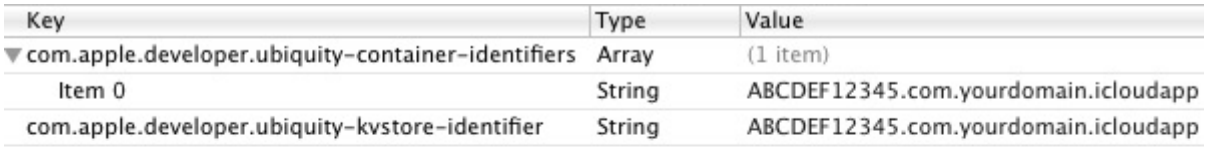
### 5.1.4. Безопасность времени выполнения процессов

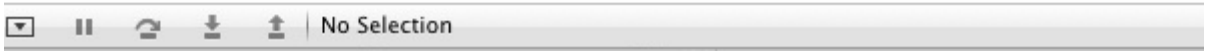
Итак, допустим, что приложение поступило из доверенного источника, и его код надлежащим образом подписан. Но и на этом меры безопасности не заканчиваются: iOS также следит за тем, чтобы приложение никак не повлияло на работу других приложений или системы. Именно поэтому все сторонние приложения работают в “песочнице” (sandbox). У каждого приложения есть домашняя папка, в которой хранятся его файлы, домашняя папка создается во время установки приложения на устройство. При необходимости доступ к системной информации приложение может получить посредством API или системных служб.

Системные файлы и ресурсы также защищены от пользовательских приложений. Большинство системных процессов, как и все процессы сторонних приложений, выполняются с правами непривилегированного пользователя “mobile”. Раздел, на котором установлена iOS, доступен только для чтения.

Необязательные системные утилиты (например, удаленное подключение или шелл) исключены из системного ПО, а API не позволяют пользовательским процессам повысить свои привилегии, чтобы внести изменения в другие приложения или саму iOS.

Доступ сторонних приложений к пользовательской информации или специальным сервисам (таким как iCloud) контролируется с помощью “прав доступа” (entitlements). Права доступа встроены в приложение и представляют собой пару “ключ-значение”. Поскольку права доступа имеют цифровую подпись, то их нельзя изменить. Права доступа широко используются системными приложениями и демонами для осуществления специальных привилегированных операций, которые в противном случае потребовали бы прав root’а. Благодаря использованию прав доступа в значительной степени снижается риск несанкционированного повышения привилегий.



Следующий механизм – рандомизация адресного пространства (Address space layout randomization - ASLR) – защищает систему от эксплуатации багов в памяти. ASLR – это технология, благодаря которой случайном образом изменяется расположение важных структур в адресном пространстве процесса. Встроенные приложения используют ASLR для гарантии того, что размещение всех регионов памяти во время запуска рандомизируется. Кроме того, Xcode – среда разработки приложений для iOS – автоматически компилирует сторонние приложения с поддержкой ASLR.

И, наконец, последний механизм защиты - это ARM's Execute Never (XN). Функция ARM's XN помечает страницы в памяти как недоступные для выполнения кода. iOS позволяет приложениям использовать страницы, помеченные одновременно, как доступные для записи и выполнения кода, только при выполнении строгих условий: у приложения должно присутствовать право доступа "dynamic-codesigning", а такое право доступа есть только у приложений, выпущенных самой Apple. И даже при выполнении этого условия, приложение сможет запросить страницу, доступную для записи и выполнения, только один раз. В частности, этим пользуется Safari для своего JIT-компилятора JavaScript.

## 5.2. НАРУШЕНИЕ БЕЗОПАСНОСТИ

Jailbreak (англ. "побег из тюрьмы") – это процедура, позволяющая получить полные права доступа ко всем разделам на устройстве.

Главная цель при jailbreak'е – модифицировать файл /private/etc/fstab, чтобы смонтировать системный раздел, как доступный для чтения/записи. Jailbreak также подразумевает модификацию службы AFC (служба используется iTunes для доступа к файловой системе устройства); модифицированная служба называется AFC2, и она позволяет получить доступ к файловой системе уже с правами root. Современные jailbreak'и также делают патч ядра для обхода механизма подписания кода и других ограничений.

Существует три вида jailbreak'а:

- привязанный (tethered) – такой jailbreak держится до перезагрузки устройства. После перезагрузки потребуется вновь сделать jailbreak, причем устройство будет неработоспособных до тех пор, пока не будет сделан повторный jailbreak;
- полупривязанный (semi-tethered) – jailbreak также работает только до перезагрузки, но после перезагрузки устройство все же можно использовать;
- непривязанный (untethered) – jailbreak в истинном смысле слова: не слетает при перезагрузке.

Jailbreak возможен именно благодаря уязвимостям в коде. Существует множество эксплойтов, использующих уязвимости на различных уровнях системы, например:

- на уровне BootROM (0x24000 Segment Overflow, usb_control_msg(0xA1, 1) Exploit). Эксплойты этого уровня могут нарушать принцип цепочки доверия с самого начала, например, обходя проверку подписи LLB.
- на уровне iBoot (iBoot Environment Variable Overflow, usb_control_msg(0x21, 2) Exploit).
- на уровне ядра (IOSurface Kernel Exploit, ndrv_setspec() Integer Overflow, HFS Heap Overflow).
- и даже на уровне пользовательских процессов (MobileBackup Copy Exploit, T1 Font Integer Overflow, Racoon String Format Overflow Exploit).

В каждой новой версии операционной системы Apple старается исправлять уязвимости предыдущей системы.

## 5.3. XARA-УЯЗВИМОСТИ В IOS

XARA, от Cross-App Resource Access - атаки, основанные на способах коммуникации между собой различных приложений на OS X и iOS.

### 5.3.1. Перехват данных, передаваемых через URL-схемы

В iOS (как, и в OS X) приложения могут общаться между собой посредством URL-схем. Как это выглядит: для примера, возьмем ситуацию, когда в мобильный почтовый клиент Почта приходит e-mail, содержащий в себе какой-нибудь адрес. Почтовик по нажатию на этот адрес перебрасывает пользователя в навигационное приложение, скажем Карты. Единственный способ реализовать такой переход в iOS — это вызвать в первом приложении ссылку вида: maps://55,678+32,432.

Система отлавливает вызов такого URL и проверяет, какое из установленных приложений может его обработать. Если такое приложение найдено, то происходит переход, и открывшаяся программа уже сама решает, каким образом поступить с данными (в нашем случае — координатами). Но что если не одно, а два приложения заявляют, что работают с URL такого вида? Существуют следующие правила поведения операционных систем:

- OSX перебросит пользователя в приложение, которое первым заявило, что оно работает с этой схемой;
- iOS перебросит пользователя в приложение, которое последним заявило, что оно работает с этой схемой.

Стоит отметить, что в отличие от Android, пользователю не предоставляется выбор, какое из приложений должно открыться. Нет никаких дополнительных механизмов аутентификации, ничего.

Таким образом, уязвимость эксплуатируется следующим образом: мы создаем свое приложение, Hijack, в свойствах которого указываем возможность обрабатывать URL-схему maps, и отдаем его пользователю. В отличие от Maps, наша вредоносная программа не откроет карту, а просто перешлет эти координаты на наш сервер.

Та же схема работает с любыми данными, передаваемыми между приложениями — к примеру, токенами, получаемыми при авторизации через Facebook или Twitter.

Слабое звено таких схем обычно заключается в способе передачи атакуемому пользователю вредоносной программы. Что примечательно — Apple не ведет никакого общего каталога URL-схем, поэтому такое приложение спокойно пройдет модерацию и будет доступно для загрузки.

### 5.3.2. Модификация прав доступа keychain на OS X

В отличии от iOS, каждому из свойств, хранимых в Keychain на OS X можно установить дополнительные права доступа (Access Control List), в которых будет перечислено, каким приложениям предоставляется доступ к нему.

Рассмотрим пример: пользователь скачивает из Mac App Store приложение, например Book — социальную сеть. Как и любое другое приложение, при вводе авторизационных данных оно пытается сохранить их в Keychain. Перед тем, как создать новую запись, система проверит, нет ли уже в связке ключей записи с такими параметрами. Это сделано для того, чтобы при обновлении программ пользовательские данные не слетали. Если такая запись найдена, то ее содержимое перезапишется и ACL не изменится, если нет — то создастся новая запись и ACL для приложения Book.

За день до установки Book пользователь скачал себе другое приложение, уже упомянутый Hijacker. Это вредоносное ПО первым добавило в Keychain запись с такими же параметрами, которые требуются оригинальному приложению, и в ACL добавило себе все права. Таким образом, когда пользователь попытается сохранить свой пароль, доступ к нему будет уже у двух приложений — и Book, и Hijacker. Что интересно, вредоносное приложение не обязательно должно быть установлено первым — у любого стороннего приложения есть возможность удалить определенную запись в Keychain и перезаписать ее своей.

### 5.3.3. Доступ к песочнице

Общеизвестно, что все приложения в OS X работают в рамках своей песочницы и не имеют доступ к другим программам. Эти песочницы представлены в файловой системе папками, названием которых выступает уникальный идентификатор приложения. Mac App Store при проверке всех публикуемых программ проверяет этот ID на уникальность, поэтому, казалось бы, повторений быть не должно.

Сложности вносит возможность включать в оригинальное приложение дополнительные подпрограммы — хелперы, фреймворки, все что угодно, у чего есть свой Bundle ID. Каждая из таких подпрограмм работает в своей песочнице. Уязвимость заключается в том, что Apple не проверяет на уникальность идентификаторы этих хелперов — в следствие чего два разных приложения могут работать в одной и той же песочнице и получить полный доступ к данным друг друга.

### 5.3.4. Уязвимости общения через WebSocket

WebSocket — это протокол, с помощью которого сервер может общаться с клиентом. Он интегрирован как часть стандарта HTML5, и позволяет содержимому WebView в браузере обращаться к любому другому приложению в системе, прокидывая данные через определенный TCP-порт. Какая-либо аутентификация отсутствует и в этом случае — этот порт может слушать кто угодно. Расширение в браузере никак не может проверить, с кем конкретно оно общается.

Пример эксплуатации уязвимости —1Password. Его браузерное расширение собирало введенные пользователем в различных формах пароли и номера кредитных карт, и через порт 6263 отправляло их приложению. Программу, которая слушает этот же порт, логирует все полученные данные, и собирает базу. И все это спокойно проходит проверку в App Store.

### 5.3.5. Уязвимость FREAK

Уязвимость под названием FREAK впервые обнаружили криптографы из исследовательских компаний INRIA, Microsoft Research и IMDEA, а широкую известность ей принесла публикация специалиста по безопасности Мэтью Грина (Matthew Green) из Университета Джона Хопкинса.

Название уязвимости «атака FREAK» происходит от фразы «Factoring attack on RSA-EXPORT Keys», означающей способ подбора открытых ключей к «экспортному» шифрованию RSA. Суть уязвимости заключается в том, что злоумышленники могут заставить браузеры использовать более слабое шифрование, чем принято обычно. Тогда они смогут взломать его за считанные часы, получив не только доступ к чужим личным данным, но и возможность управлять содержимым страниц в браузере вплоть до кнопки лайка Фейсбука.

Причина появления FREAK лежит в старом требовании властей США, принятом ещё в 1990-е годы. После внедрения шифрования в браузер от Netscape государство требовало от технологических компаний оставлять в своих алгоритмах шифрования «лазейку» для спецслужб при экспортировании своих продуктов за рубеж.

Агентства вроде АНБ и ФБР опасались, что не смогут расшифровать слишком стойкий шифр, если понадобится вести слежку за пользователями в других странах. Несмотря на то, что требование не применять сильную криптозащиту в «экспортных» продуктах было снято в конце 1990-х, более слабое шифрование было интегрировано в множество программ и оставалось незамеченным публикой до недавнего времени.

«Экспортное» шифрование использовало ключи безопасности длиной в 512 бит, а не 1024, как было принято обычно. По данным исследователей, такой ключ можно было подобрать в течение семи часов при помощи мощности 75 обычных компьютеров или аренды аналогичной мощности за 100 долларов в «облачном» сервисе вроде Amazon Web Services. Демонстрацию подбора 512-битного ключа впервые публично провели в 1999 году.

До сих пор никто публично не сообщал об удачной попытке взлома шифрования с длиной ключа в 1024 бита. По оценке Мэтью Грина, для этого требуется мощность порядка нескольких миллионов обычных компьютеров и работа команды хакеров в течение года. Кроме того, многие сервисы сегодня используют ключи длиной в 2048 бита, что делает шифрование ещё более надёжным.

В результате ошибки в браузерах, использующих протоколы OpenSSL (Android) или SecureTransport (Apple), злоумышленники могли заставить соединение проходить через «экспортное», более слабое шифрование, даже если пользователь (то есть его браузер) не требовал такого подключения. Доказательство функциональности атаки FREAK подтвердили специалисты из INRIA на примере сайта АНБ.

Исследователи запустили специальный сайт, на котором можно проверить, уязвим ли используемый браузер перед атакой FREAK, и список сайтов, соединение с которыми в данный момент может быть подвержено слежке. Всего таких серверов около 36,7% от общего количество сайтов с сертификатом безопасности, однако среди первого миллиона рейтинга Alexa их количество снизилось с 12,2% до 9,7% после того, как о FREAK стало широко известно.

Среди самых крупных из подверженных уязвимости значатся издание Business Insider, агентство Bloomberg, сайты Массачусетского технологического института, «Вести.ру», «Альфа-Банка», «Студии Артемия Лебедева» и «Викимарта».

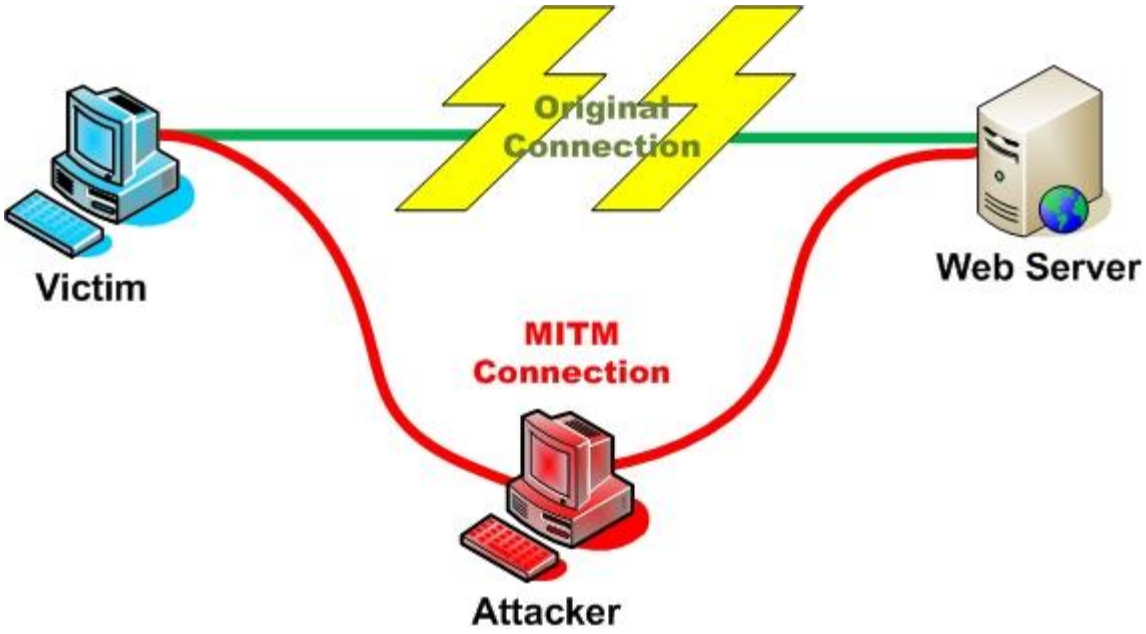
Как сообщает газета Washington Post, некоторое время оставались уязвимыми даже крупные государственные сайты, например, FBI.gov, Whitehouse.gov и NSA.gov, однако позднее их защитили от FREAK. В Apple отреагировали на новость об обнаруженной уязвимости и пообещали выпустить патч на следующей неделе как для Mac OS X, так и для iOS. В Facebook внесли изменения в алгоритмы своего сайта connect.facebook.com, так что он перестал быть уязвимым.

Мэтью Грин признаёт, что атака, совершённая на сайт АНБ, на самом деле является атакой на её хостера Akamai, и не является «взломом АНБ». Тем не менее, существование уязвимости FREAK доказывает, что желание властей оставлять так называемые «бэкдоры» — лазейки в алгоритмах шифрования — приводит к печальным последствиям. Системы сегодня настолько сложны, что даже обычная нагрузка на них может привести к поломке. Для новых бэкдоров места нет.

### 5.3.6. Уязвимость к атаке MITM

Цель атаки man-in-the-middle (MITM) — перехватить сообщения, передающиеся между двумя системами. Например, в стандартной HTTP-транзакции клиент и сервер

общаются с помощью TCP-соединения. Используя различные методы, злоумышленник может разбить оригинальное TCP-соединение на два новых, одно между собой и клиентом, другое между собой и сервером.




После перехвата TCP-соединения, злоумышленник действует как прокси, при этом он может читать данные и даже изменять их.

Атака MITM является довольно эффективной из-за природы HTTP-протокола и передаваемых данных, основанных на ASCII. Например, применяя man-in-the-middle attack, можно перехватить куки сессии пользователя, а также изменить структуру HTTP-заголовка.

### 5.3.7. Уязвимости при https

HTTPS - это обычный протокол HTTP, который поддерживает шифрование. Он может защитить передачу данных в виде обычного текста. HTTPS использует SSL или TLS для шифрования запросов и ответов веб сервера, делая их непроницаемыми для сниферов и атак "человек посередине".

Однако, даже HTTPS-соединение не панацея. Атака "человек посередине" может быть осуществлена при использовании HTTPS-соединения. С единственной разницей в том, что нужно будет создать две независимых SSL-сессии, по одной на каждое TCP-соединение.

Не смотря на то, что метод атаки не прост, хакер, вооруженный необходимым оборудованием и программным обеспечением, вполне может провернуть подобную атаку даже при зашифрованном соединении.

Man-in-the-middle это не только техника нападения на TCP-соединения. Этим также пользуются в ходе разработки веб-приложений, оценивая уязвимости скриптов.

## ЗАКЛЮЧЕНИЕ

Несмотря на кажущуюся неуязвимость системы iOS, всегда найдутся способы обойти функции безопасности. Рассмотренные уязвимости еще раз подтверждают простую истину: полностью безопасных систем нет; и один из способов держать уровень безопасности на приемлемом уровне – это своевременные обновления.