ЛЕКЦИЯ 15. УЯЗВИМОСТИ ANDROID И IOS

15.1. УЯЗВИМОСТИ ANDROID

15.1.1. Вирусы и вредоносное ПО

Список уверенно возглавляют СМС-троянцы (семейство Android.SmsSend). Целью таких программ является отправка сообщений с повышенной тарификацией на короткие номера. Часть стоимости этих сообщений поступает в карман злоумышленников, обогащая их. Чаще всего они распространяются под видом популярных игр и приложений, таких как Opera Mini, ICQ, Skype, Angry Birds и т. п., при этом используется соответствующая иконка.

Далее по списку следуют более «тяжеловесные» троянцы. К ним относятся, например, Android.Gongfu, Android.Wukong, Android.DreamExploid, Android.Geinimi, Android.Spy и пр. В зависимости от семейства, эти вредоносные программы обладают таким функционалом, как сбор конфиденциальной информации пользователя, добавление закладок в браузер, выполнение команд, поступающих от злоумышленников (функции бэкдора и бота), отправка СМС-сообщений, установка других приложений и т. п.

Существуют также коммерческие программы-шпионы. Эти приложения используются для слежки за пользователями. В их арсенал, в зависимости от класса, стоимости и производителя, входят такие функции, как перехват входящих и исходящих СМС-сообщений и звонков, аудиозапись окружения, отслеживание координат, сбор статистических данных из браузера (например, закладки, история посещений) и т. п.

15.1.2. Уязвимости архитектуры Android и ПО

Одна из главных проблем, с которыми могут столкнуться пользователи – это уязвимости системы, позволяющие получить права root. Существуют специальные приложения, скрипты и программные модули, выполняющие эту задачу. В повседневной жизни подобные вещи пользователям не страшны, так как чаще всего их используют осознанно для получения большего контроля над устройством. Другое дело, что эти же уязвимости (например, CVE-2009-1185 , CVE-2011-1823) взяли на вооружение создатели вредоносных приложений.

Используя эксплойты для повышения своих прав до уровня root, они получают возможность, например, беспрепятственно устанавливать другие программы без разрешения пользователя (как это делают различные модификации Android.Gongfu и Android.DreamExploid). Некоторые вредоносные программы не используют эксплойты сами, а вводят пользователя в заблуждение и побуждают его самого выполнить необходимые действия, тем самым дав вредоносной программе требуемые ей возможности.

Существует возможность создания приложений, которые не будут требовать никаких разрешений для своей работы, что может создать ложное ощущение полной безопасности. Однако на самом деле такие приложения смогут получить доступ к определенной информации (например, файлам, хранящимся на карте памяти в незащищенном виде, списку установленных программ, используемому оператору мобильной связи) и даже отправить эту информацию злоумышленникам через Интернет.

15.1.3. Уязвимости прошивок и ПО

Угрозу представляют также неофициальные или сторонние прошивки. Поводов для беспокойства здесь несколько.

Во-первых, в такие прошивки изначально могут быть встроены вредоносные программы. Во-вторых, когда цифровой подписью образа системы подписывается какое-либо приложение, оно получает те же права, что и сама система, в которой оно работает. Подобный способ заражения применялся, в частности, вредоносной программой Android.SmsHider, которая могла незаметно для пользователей, использующих определенные сторонние прошивки, установить содержащийся в ней троянский apk.

Системные приложения, как стандартные, так и приложения от поставщиков Android-устройств, тоже подвержены уязвимостям. Например, некоторые уязвимости браузера WebKit позволяют потенциальным вредоносным программам выполнить произвольный JavaScript-код и получить доступ к защищенным данным браузера.

Если разработчики прикладного ПО не уделяют достаточное внимание безопасности при работе с данными пользователей, эти данные могут быть скомпрометированы. Атаке могут подвергаться хранящиеся в незащищенном виде регистрационные данные, пароли от банковских карт и прочая конфиденциальная информация.

15.1.4. Репозиторий приложений

Репозиторий приложений Google Play является слабым местом Android. Основная проблема в том, что приложение и его автор не подвергаются полной проверке. Чтобы решить эту проблему, не прибегая к ручной проверке приложений на безопасность, как сделано в Apple App Store, Google ввела сервис Bouncer, представляющий собой виртуальную машину, в которой автоматически запускается любое публикуемое в репозитории приложение. Bouncer выполняет многократный запуск программы, производит множество действий, симулирующих работу пользователя с приложением, и анализирует состояние системы до и после запуска с целью выяснить, не было ли попыток доступа к конфиденциальной информации, отправки SMS на короткие платные номера и так далее. По словам Google, Bouncer позволил сократить количество вредоносов сразу после запуска сервиса на 40%.

15.2. УЯЗВИМОСТИ IOS

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’a:

- 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.

15.2.1. 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.

15.2.2. 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.

15.2.3. 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.

15.2.4. 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.

15.2.5. 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.

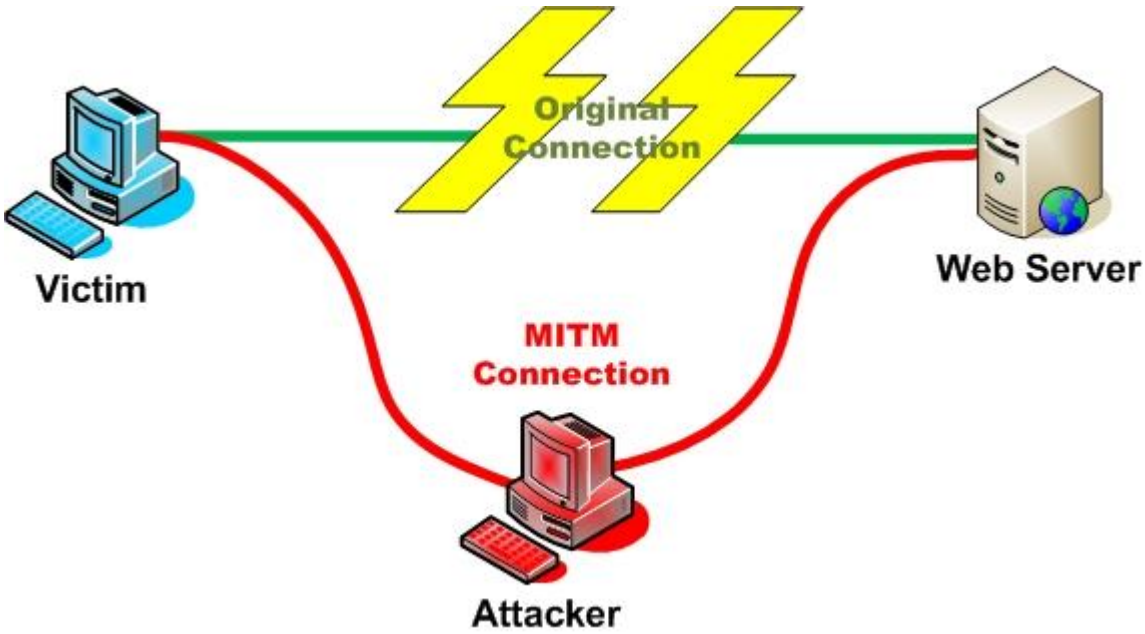
Среди самых крупных из подверженных уязвимости значатся издание Business Insider, агентство Bloomberg, сайты Массачусетского технологического института, «Вести.ру», «Альфа-Банка», «Студии Артемия Лебедева» и «Викимарта».

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.

15.2.6. 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.

15.2.7. 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.

15.3. СРАВНЕНИЕ МЕХАНИЗМОВ БЕЗОПАСНОСТИ В ANDROID И IOS

С точки зрения разработчиков ПО основной риск безопасности - это взлом их приложения и, как следствие, потеря клиентов и бизнеса. Если рассматривать способность мобильных приложений противостоять локальным и веб-атакам, то обе операционные системы находятся примерно в равных условиях. Следуя практикам безопасной разработки, разработчикам вполне под силу создать хорошо защищенное приложение и для Android, и для iOS.

Как правило, приложения для Android представляют собой программы, написанные на языке Java, который невосприимчив к атакам на переполнение буфера, в отличие от программ для iOS, написанных на Objective-C. Приложения для Android можно легко декомпилировать и изменить исходный код на вредоносный, поэтому разработчики должны применять техники обфускации кода.

Приложения, написанные на Objective-C, потенциально уязвимы к переполнению буфера, однако в арсенале iOS-разработчиков присутствуют необходимые механизмы, способные предотвратить успешную эксплуатацию таких уязвимостей. К таким механизмам, прежде всего, относятся параметры компиляции, такие как PIE (Position Independent Executable), SSP (Stack Smashing Protection) и ARC (Automatic Reference Counting). Данные параметры обеспечивают эффективное управление памятью и отсутствие ошибок, которые приводят к переполнениям буфера. В дополнение к этому на презентации восьмой версии iOS был представлен новый язык программирования Swift, призванный заменить собой Objective-C. По заявлениям компании Apple, новый язык является более простым в изучении и более безопасным, чем его предшественник.

Таким образом, Android- и iOS-приложения могут быть одинаково хорошо защищенными, и вероятность взлома этих приложений напрямую зависит от мастерства разработчиков.

Безопасность рядовых пользователей мобильных девайсов зависит от безопасности мобильной ОС, которой они пользуются. Даже если на девайсе установлены только хорошо защищенные приложения, конечные пользователи все равно могут быть успешно атакованы через бреши в самой операционной системе. Если по критерию защищенности мобильных приложений обе ОС находятся на примерно одинаковом уровне, то с точки зрения безопасности самой системы Android и iOS значительно различаются между собой.

Прежде чем говорить о различиях в механизмах защиты двух ОС, стоит отметить наличие в них базовых принципов безопасности, таких как доступность системного раздела только для чтения и разграничение выполняемых процессов на уровне ядра. И в Android, и в iOS системный раздел не доступен для записи, что предотвращает случайное либо целенаправленное изменение файлов системы. Также в обоих ОС реализован принцип «песочницы» (sandbox). Это значит, что каждое приложение работает в изолированном контейнере и не имеет доступа к системным файлам либо ресурсам других приложений.

В системе iOS почти все приложения выполняются под непривилегированным пользователем «mobile». В Android каждому приложению соответствует свой уникальный пользователь, что обеспечивает разграничение прав выполняемых приложений на уровне ядра операционной системы.

Основные различия в механизмах безопасности Android и iOS относятся к принципам разграничения доступа на уровне ядра, к процессу верификации загружаемого в магазины ПО и к принципам контроля прав доступа устанавливаемых приложений.

Перед тем, как появиться в магазине App Store, iOS-приложения тщательно проверяются на наличие уязвимостей и на соответствие стандартам разработки Apple. Также каждое приложение, устанавливаемое на iOS, должно быть подписано уникальным сертификатом программы «iOS Developer Program», который выдается компанией Apple

только после необходимых верификаций разработчика. Описанные меры обеспечивают отсутствие вредоносного ПО в магазине приложений App Store.

Google не проверяет тщательно приложения перед загрузкой их в Google Play, но проводит регулярные сканирования своего магазина на предмет наличия потенциально вредоносного ПО. Такой подход компании Google может показаться достаточно небезопасным. И действительно, в магазине Google Play размещено большое количество потенциально вредоносного ПО (malware). Однако большинство таких программ на самом деле не несут в себе ничего вредоносного и на деле являются обычными рекламными приложениями.

Бывает, что в магазине Google Play присутствуют настоящие вредоносные приложения типа malware, но при должных пользовательских навыках можно полностью оградить себя от воздействия такого рода ПО. Дело в том, что при установке нового приложения на девайс под управлением Android пользователю показывается полный перечень прав доступа, требуемых данному приложению.

По этому перечню пользователь может определить потенциально вредоносное ПО и отменить его установку. Например, если приложение «Фонарь» собирается запрашивать права на доступ к контактным данным либо на доступ к Интернету, то данное приложение с определенной долей вероятности можно отнести к вредоносному ПО.

А что насчет уязвимостей в самой операционной системе? Казалось бы, Android как полностью открытая ОС должна насчитывать огромное количество уязвимостей, найденных специалистами по всему миру. Однако iOS опережает Android по количеству известных уязвимостей (CVE), причем превосходство это довольно значительное. Несмотря на такое большое количество уязвимостей, пользователям iOS не стоит беспокоиться о своей защищенности, так как Apple, как правило, закрывает новые уязвимости достаточно быстро.

Таким образом, в контексте собственной безопасности операционной системы победителем не является ни одна из двух ОС. И Android, и iOS имеют мощные механизмы защиты от хакерских атак, и обе компании, Google и Apple, уделяют безопасности своих систем повышенное внимание.

В последние годы активно растет число пользователей, которые используют свои личные мобильные девайсы для выполнения рабочих задач. Эта тенденция, получившая название BYOD (Bring Your Own Device), несет в себе определенные риски безопасности для корпораций. При помощи уязвимого либо просто утерянного смартфона или планшета злоумышленники могут получить несанкционированный доступ к секретной документации компании либо к ее внутренним ресурсам, например к почте.

В связи с этим возникает потребность в использовании решений типа Mobile Device Management (MDM), позволяющих централизованно управлять политиками безопасности мобильных девайсов, работающих в сетях компании.

С точки зрения безопасности на уровне корпорации ОС от Apple имеет ряд преимуществ перед Android. iOS имеет в своем арсенале мощные средства для централизованного управления девайсами, такие как профили конфигурации, возможность удаленного полного сброса и встроенная поддержка сторонних MDM-решений. Android в чистом виде таких возможностей не имеет. Для интеграции с MDM-системами на Android необходимо предварительно устанавливать специальное ПО.

Стоит отдельно отметить, что компания Samsung ушла далеко вперед в вопросах корпоративной безопасности по сравнению с другими производителями девайсов на Android. Речь идет о программе SAFE (Samsung For Enterprise) и надстройке KNOX, которая представляет собой хорошо защищенный контейнер для всех рабочих активностей пользователей с поддержкой сторонних MDM-систем. Таким образом, все аппараты от Samsung, работающие на Android 4.3 и выше, полностью соответствуют принципам защищенного бизнеса. Тем не менее, Apple имеет гораздо меньшую линейку продуктов, нежели производители Android-девайсов, поэтому ей не составляет труда

обеспечить поддержку систем корпоративной безопасности для всех своих смартфонов, планшетов и актуальных версий ОС. В категории наиболее безопасной для использования на уровне компаний операционной системы победителем выходит iOS.

Резюмируем основные плюсы и недостатки двух операционных систем с точки зрения безопасности:

Android:

Плюсы:

• Открытость для исследователей безопасности;

• Иммунитет приложений к переполнениям буфера;

• Строгий контроль доступа на уровне ядра.

Минусы:

• Большое количество потенциально вредоносного ПО в магазине Google Play;

• Слабые возможности в обеспечении корпоративной безопасности;

• Большое число версий ОС и моделей девайсов от различных вендоров, что усложняет стандартизацию методов защиты.

iOS:

Плюсы:

• Тщательный контроль загружаемых в App Store приложений, и, как следствие, практически полное отсутствие вредоносного ПО в магазине приложений;

• Быстрая реакция Apple на инциденты безопасности;

• Большие возможности по поддержке систем корпоративной безопасности.

Минусы:

• Большое количество известных уязвимостей в самой операционной системе;

• Рост числа возможных векторов для атаки (интеграция в экосистему Apple, система HomeKit).