Security Engineering: часть 10. Эмиссионная безопасность и Атаки на API

Published: 25.06.2026 10:19 | Updated: 25.06.2026 14:31

Эмиссионная безопасность (Emission Security): Невидимые угрозы и компрометирующие излучения

Эмиссионная безопасность (Emsec) — это область информационной безопасности, посвященная предотвращению атак с использованием компрометирующих излучений, а именно: кондуктивных (передаваемых по проводникам) или радиочастотных электромагнитных сигналов.
Военные организации традиционно уделяют огромное внимание защитным мерам класса **TEMPEST**, которые предотвращают перехват паразитных радиочастотных излучений, генерируемых компьютерами и другим электронным оборудованием, с целью реконструкции обрабатываемых данных. В коммерческом секторе эта угроза стала актуальной в контексте атак по побочным каналам (side-channel attacks), когда вычисления, выполняемые смарт-картой (например, создание цифровой подписи), наблюдаются путем измерения потребляемого устройством тока. Кроме того, исследователи обнаружили атаки, эксплуатирующие паразитные оптические, тепловые и акустические излучения различного оборудования.
Эти угрозы тесно связаны с электромагнитной совместимостью (EMC) и радиочастотными помехами (RFI), которые могут случайно нарушать работу систем. По мере того как все больше устройств подключаются к беспроводным сетям, а тактовые частоты процессоров уходят в диапазон гигагерц, проблемы EMC/RFI, Emsec и электронного warfare (Radio Electronic Warfare) неизбежно пересекаются и усугубляются.

Исторический экскурс: от перекрестных помех до TEMPEST

Перекрестные помехи (crosstalk) в телефонных проводах были хорошо известны пионерам телефонии в XIX веке. Двухпроводные цепи stacking на перекладинах опор часто становились причиной утечек. Решением стало использование скрученных пар. Однако военное применение утечек началось во время британской экспедиции в Нил и Суакин (1884–1885 гг.).

Во время Первой мировой войны полевые телефонные провода, проложенные для связи с войсками, застрявшими в грязи Фландрии, часто проходили параллельно вражеским траншеям на расстоянии всего в несколько сотен ярдов. Использовались одномодовые изолированные кабели с заземлением для уменьшения веса. Вскоре было обнаружено, что утечка в землю вызывает сильные перекрестные помехи, позволяя перехватывать сообщения противника. К 1915 году усилители на электронных лампах позволили увеличить дальность перехвата до 100 ярдов для телефонии и 300 ярдов для азбуки Морзе. К 1916 году одномодовые цепи с заземлением были запрещены в зоне 3000 ярдов от линии фронта.

Вторая мировая война принесла развитие радиолокации, пеленгации и методов связи с низкой вероятностью перехвата. К 1960-м годам паразитные радиочастотные излучения от гетеродинов бытовых телевизоров перехватывались пеленгационным оборудованием в «фургонах TV-детекторов» в Великобритании, где владельцы телевизоров обязаны были платить ежегодный лицензионный сбор.

Переломным моментом в осознании масштабов угрозы стала публикация 1985 года, сделанная голландским исследователем Вимом ван Эком. Он продемонстрировал, как с помощью модифицированного телевизора можно на расстоянии реконструировать изображение, отображаемое на видеомониторе (VDU). Тот факт, что атаки TEMPEST можно было осуществить с помощью самодельного оборудования, вызвал шок в индустрии компьютерной безопасности.

Техническая разведка и контрмеры

Прежде чем анализировать паразитные излучения штатных устройств, необходимо рассмотреть простейшие и наиболее распространенные атаки с использованием внедренных устройств — «жучков».
Арсенал доступных на рынке средств скрытого съема информации огромен:

- **Радиомикрофоны:** Простейшие устройства стоимостью в несколько десятков долларов можно спрятать под столом. Их главный лимитирующий фактор — время работы от батареи (от дней до недель), а радиус действия обычно составляет несколько сотен ярдов.
- **Устройства с внешним питанием:** Более сложные жучки питаются от телефонной линии или электросети, что позволяет им работать бесконечно долго. Некоторые маскируются под электрические адаптеры, совмещая в себе микрофон, радиопередатчик и телекамеру.
- **Лазерные микрофоны:** Работают путем направления лазерного луча на отражающую или полупрозрачную поверхность (например, оконное стекло) в комнате, где ведется разговор. Звуковые волны модулируют отраженный свет, который затем декодируется на расстоянии.
- **Игрушка Furby:** Агентство национальной безопасности США (NSA) запрещало проносить эти игрушки в свои здания. Причина заключалась в том, что Furby «запоминал» и случайно воспроизводил фразы, сказанные в его присутствии, что создавало риск непреднамеренной утечки информации.

Контрмеры требуют высокой квалификации. Для поиска скрытой электроники на близком расстоянии применяются **нелинейные детекторы переходов**. Они излучают слабый радиосигнал и слушают гармоники, генерируемые транзисторами и диодами. Однако если жучок спрятан внутри другого электронного оборудования, детектор оказывается бесполезным. Существуют также дорогие жучки, которые вообще не переизлучают сигнал.
Для защиты помещений применяются экранированные комнаты (клетки Фарадея), а также строгие процедурные меры, такие как запрет на пронос личной электроники и регулярные проверки.

Пассивные атаки: Утечки по кабелям и радиоканалу

Пассивные атаки предполагают, что злоумышленник лишь принимает электромагнитные сигналы, не вмешиваясь в работу системы. Они делятся на кондуктивные (по кабелям) и радиочастотные.

Утечки по силовым и сигнальным кабелям

Инженеры давно знают, что высокочастотные сигналы «утекают» повсюду. Кондуктивные утечки можно подавить тщательным проектированием: использованием фильтров и экранов на силовых и сигнальных линиях.
В военной сфере это реализуется через концепцию разделения **Red/Black** (Красное/Черное). Оборудование, обрабатывающее конфиденциальные данные (Red), должно быть изолировано с помощью фильтров и экранов от оборудования, которое может напрямую передавать сигналы во внешний мир (Black). Особую сложность представляют устройства, имеющие как «красные», так и «черные» соединения (например, шифровальные машины). Стоимость таких мер защиты крайне высока: фильтрация тысяч кабелей в серверной или на авиабазе и строгий контроль конфигурации могут стоить миллионы долларов.

Утечки через радиочастотные сигналы

Любое оборудование излучает радиоволны. Видеомониторы (VDU) генерируют слабый ТВ-сигнал (ВЧ или УВЧ), модулированный искаженной версией отображаемого изображения. Видеосигнал доступен в различных точках оборудования, в частности, в токе луча электронно-лучевой трубки. Эти излучения содержат множество гармоник частоты строчной развертки, некоторые из которых лучше излучаются из-за резонанса кабелей и компонентов на их длине волны. С помощью широкополосного приемника эти эмиссии могут быть приняты и реконструированы в видеоизображение.
Плоские панели (LCD) также уязвимы. В типичном ноутбуке по шарниру проходит последовательная линия от системной платы к дисплею, которая передает видеосигнал и легко может быть перехвачена.
Soft Tempest — это программная технология, разработанная для защиты от перехвата. Она использует программные методы для фильтрации, маскировки или искажения информации, содержащейся в электромагнитных излучениях. Например, исследователи обнаружили, что большая часть энергии, несущей информацию от VDU, сосредоточена в верхней части спектра. Применение

низкочастотного фильтра к стандартному шрифту (свертка с фильтром) делает перехват практически невозможным, при этом визуальное качество текста для пользователя почти не страдает.

Активные атаки: От вирусов до глитчинга

Активные атаки включают не только перехват, но и намеренное вмешательство в работу оборудования или использование внешних воздействий для извлечения данных.

Вирусы TEMPEST

Злоумышленник может написать вредоносный код, который заставит компьютер работать как маломощный радиопередатчик. Изменяя паттерны обращения к памяти или шине данных, вирус модулирует электромагнитное излучение системы, передавая украденные данные на ближайший приемник. Это позволяет обойти даже изолированные от сетей (air-gapped) системы.

Атаки Nonstop

Этот класс атак эксплуатирует электромагнитные излучения, которые случайно индуцируются Nearby (близлежащими) радиопередатчиками. Если оборудование, обрабатывающее конфиденциальные данные, используется рядом с мощным передатчиком (например, на корабле или самолете), сигналы передатчика могут индуцировать токи в цепях компьютера. Из-за нелинейного эффекта переходов эти токи модулируются конфиденциальными данными и переизлучаются.

Глитчинг (Glitching) и дифференциальный анализ неисправностей

В мире смарт-карт и защищенных микроконтроллеров активные атаки часто направлены на внесение сбоев в нормальную работу устройства.
* **Глитчинг:** Злоумышленник вводит переходные процессы (транзиенты) в линии питания или тактирования карты. Например, в ранних банковских приложениях неоправданно высокий тактовый сигнал вызывал сброс только после нескольких циклов. Это позволяло заменить один тактовый импульс двумя более узкими, заставляя процессор выполнить команду NOP (нет операции) вместо запланированной инструкции. Это дает возможность атакующему «перепрыгивать» через проверки доступа.

* **Дифференциальный анализ неисправностей (DFA):** Если в криптографический алгоритм (например, RSA) можно внести случайную ошибку на определенном этапе вычислений, это может привести к катастрофической утечке ключа. Например, если подпись вычисляется с использованием Китайской теоремы об остатках (CRT), ошибка в одной из ветвей вычислений позволяет мгновенно восстановить секретный ключ.

Комбинированные атаки

Наиболее опасные сценарии объединяют пассивные и активные методы. Например, использование лазера для частичной ионизации целевого транзистора в момент, когда проводится анализ потребляемой мощности (Power Analysis). Это позволяет атакующему с высочайшей точностью считывать отдельные биты данных из памяти, даже если логика чипа зашифрована или использует «запутанную» (glue logic) архитектуру.

Атаки по времени (Timing Analysis)

В 1996 году Пол Кохер продемонстрировал, что многие реализации алгоритмов с открытым ключом, такие как RSA и DSA, раскрывают информацию о ключе через время, затрачиваемое на их выполнение. Его идея заключалась в том, что при выполнении возведения в степень программное обеспечение обычно обрабатывает секретную экспоненту бит за битом, и если следующий бит равен единице, выполняется умножение. Это позволяет противнику, который знает первые b бит экспоненты, вычислить (b+1)-й бит, наблюдая за рядом операций возведения в степень. Атака и защита развивались параллельно, и долгое время многие считали свои реализации безопасными, если они использовали Китайскую теорему об остатках. Однако в 2003 году Дэвид Брамлей и Дэн Боне реализовали атаку по времени против Apache с использованием OpenSSL и показали, как извлечь закрытый ключ с удаленного сервера, измерив время около миллиона операций расшифрования. Современные надежные реализации алгоритмов с открытым ключом теперь используют ослепление (blinding) для предотвращения подобных атак.
В 1998 году Джон Келси, Брюс Шнайер, Дэвид Вагнер и Крис Холл указали на то, что блочные шифры, использующие большие S-блоки (такие как AES), могут быть уязвимы к атакам по времени, основанным на промахах кэш-памяти (cache misses). Атакующий может проверять догадки о выходе первого раунда шифра, предсказывая, вызовет ли предполагаемое значение промах кэша при поиске в S-блоке, и сверяя это с наблюдениями. С тех пор ряд исследователей

усовершенствовали эту атаку, и в наши дни наивная реализация AES может быть взломана путем наблюдения всего за несколькими сотнями операций шифрования.

Оптические, акустические и тепловые каналы утечки, оценка угроз Emsec и начало Атак на API

Мы рассмотрели классические радиочастотные и кондуктивные атаки. Однако по мере развития технологий исследователи обнаружили, что утечка информации может происходить через самые неожиданные физические среды: свет, звук и тепло. Кроме того, мы разберем, насколько серьезны эти угрозы для правительств и бизнеса, потом рассмотрим атаки на интерфейсы прикладного программирования (API), где логические ошибки проектирования позволяют обходить даже самую стойкую криптографию.

Оптические, акустические и тепловые каналы утечки

В последние годы поток исследований выявил новые классы атак по побочным каналам, использующие нестандартные физические явления.

Оптические атаки

В 2002 году Маркус Кун продемонстрировал, что содержимое экрана ЭЛТ-монитора (VDU) может быть восстановлено оптически, даже если анализировать лишь диффузный свет, отраженный от стен комнаты или от одежды и лица оператора. Используя высокопроизводительный фотоумножитель и осциллограф, он показал, что люминофоры синего и зеленого цветов в ЭЛТ-трубках затухают через несколько микросекунд. Этот диффузный отраженный свет содержит значительную часть информации с экрана, закодированной во временной области.
Другое оптическое уязвимое место обнаружили Джо Лаффри и Дэвид Амфресс. Они исследовали стробоскопические индикаторы (светодиоды), которые устанавливаются на линиях передачи данных (например, на последовательных портах модемов, маршрутизаторов и другого сетевого оборудования) для

индикации активности. Оказалось, что значительное количество таких индикаторов фактически передают оптические данные, модулируя свет с частотой передаваемых по линии данных.

Акустические атаки

В 2004 году Дмитрий Асонов и Ракеш Агравал показали, что разные клавиши на клавиатуре издают достаточно разные звуки, чтобы их можно было различить. Они обучили нейронную сеть распознавать щелчки клавиш и смогли перехватывать вводимый текст с уровнем ошибок всего в несколько процентов. В 2005 году Ли Чжуан, Фэн Чжоу и Дуг Тайгар развили эту идею, создав атаку, которая позволяла расшифровать запись текста, набранного в течение десяти минут на неизвестной клавиатуре. Они использовали характеристики клавиатуры в сочетании со статистикой естественного языка (английского текста), чтобы определить, какая именно клавиша соответствовала каждому звуку.
Еще более глубокий уровень акустических атак продемонстрировали Эран Тромер и Ади Шамир. Они доказали, что ключи можно перехватить, анализируя акустические эмиссии, генерируемые конденсаторами на материнской плате ПК. Эти звуки находятся на частотах выше 10 кГц и модулируются вычислительными процессами.

Тепловые скрытые каналы

В 2006 году Стивен Мердок обнаружил, что многие компьютеры раскрывают свою загрузку процессора через тепловую утечку. Тактовая частота (clock skew) является функцией температуры окружающей среды и может быть измерена удаленно. Мердок выдвинул гипотезу, что, анализируя эти тепловые паттерны, атакующий может даже определить географическое положение скрытой машины: долготу — по часовому поясу, а широту (медленнее) — по сезонным колебаниям температуры.

Насколько серьезны атаки Emsec?

Угрозы для правительств

Для посольств во враждебных странах угрозы Emsec вполне реальны. Если ваше посольство вынуждено занимать второй этаж офисного здания, где на первом и третьем этажах расположилась местная тайная полиция, защита становится сложнейшей задачей. Экранирование всего электронного оборудования — часть

решения. Однако человеческий фактор остается уязвимостью: например, уборщицы, работающие на противника, могут намеренно ослаблять уплотнители Tempest-защиты на оборудовании.
В то же время в США существует растущий скептицизм относительно того, действительно ли иностранные агенты когда-либо проводили успешные атаки Tempest. Ходят анекдотичные истории, например, о том, что единственное известное использование таких методов перехвата в Северной Америке было осуществлено канадской разведкой, которая просто подслушала переговоры американских дипломатов во время переговоров по продаже зерна.
Реальный скандал, связанный с утечками и нарушением стандартов Emsec, произошел в США с компанией Lockheed-Martin при установке оборудования на суда Береговой охраны в рамках проекта Deepwater. Расследование показало не только дефекты эмиссионной безопасности (неправильные типы кабелей, нарушения правил разделения красных/черных цепей, отсутствующие фильтры), но и целый ряд других проблем: трещины в корпусе, неводонепроницаемые уличные радиостанции и системы видеонаблюдения, не обеспечивающие обзор на 360 градусов.

Угрозы для бизнеса

Для коммерческого сектора атаки Emsec стали серьезной проблемой, особенно в индустрии смарт-карт. Открытие атак дифференциального анализа мощности (DPA) Полем Кохером задержало внедрение банковских смарт-карт на несколько лет. Индустрия была вынуждена разрабатывать ad-hoc механизмы защиты, и хотя текущие устройства защищены лучше, они все еще опираются на сложную смесь аппаратных и программных мер, которые требуют постоянной бдительности.
Кроме того, «несекретные» аспекты управления электромагнитным излучением — электромагнитная совместимость (EMC) и радиочастотные помехи (RFI) — становятся все более критичными. Тактовые частоты растут, беспроводные устройства и сети proliferate (размножаются), а цифровая электроника проникает в устройства, которые ранее были аналоговыми или механическими. Появление программных радиостанций (software radios), которые оцифровывают сигнал на промежуточной частоте и обрабатывают его в ПО, открывает новые возможности как для защиты, так и для атак.

Атаки на API (API Attacks)

Многие якобы защищенные устройства имеют интерфейс прикладного программирования (API), который могут вызывать ненадежные люди и процессы.

- Сервер банка может запросить у подключенного аппаратного модуля безопасности: «Вот номер счета и PIN-код клиента, зашифрованный ключом, которым мы делимся с VISA. Верен ли PIN?».
- Если вы включаете JavaScript в браузере, вы предоставляете API, который владельцы посещаемых вами сайтов могут использовать для выполнения различных действий.
- Защищенная операционная система может ограничивать вызовы, которые может делать приложение, используя монитор ссылок (reference monitor) для предотвращения утечки информации от High к Low.

Естественный вопрос: безопасно ли разделять задачи между доверенным и менее доверенным компонентами? Недавние исследования показали, что ответ очень часто — **нет**. Проектирование безопасных API — невероятно сложная задача.
Безопасность API связана с рядом уже обсуждавшихся проблем. Например, системы многоуровневой безопасности (MLS) накладывают ограничения на потоки статических данных, тогда как API безопасности часто пытается предотвратить утечку информации в условиях плотного и динамичного взаимодействия — что гораздо сложнее. Это также связано с безопасностью протоколов: модули безопасности банков часто уязвимы из-за взаимодействия множества поддерживаемых ими протоколов. И, наконец, это связано с безопасностью ПО: реализация JavaScript в Firefox позволяла вызывающим программам сканировать все переменные, установленные существующими плагинами, что могло скомпрометировать вашу конфиденциальность.
Наиболее распространенный режим отказа API заключается в том, что транзакции, безопасные по отдельности, становятся небезопасными в комбинации — из-за синтаксиса приложения, взаимодействия функций, медленной утечки информации или проблем с параллелизмом.
В контексте встроенных систем мы обсуждали предоплатные счетчики электроэнергии: аппараты продажи токенов используют защищенные от взлома модули для защиты ключей и счетчиков. Атака заключалась в изменении тарифа на электроэнергию на минимально возможное значение и выпуске токенов, дающих право на получение энергии по цене значительно ниже рыночной. Ошибка проектирования заключалась в том, что отчетность по тарифам не была

жестко привязана к сквозному проектированию протокола.
Потенциально критическим примером является инициатива «Trusted Computing». Чип TPM для безопасного хранения криптографических ключей установлен на большинстве материнских плат. Планировалось, что будущие приложения будут иметь «небезопасную» часть, работающую поверх Windows, и «безопасную» часть (NCA), работающую поверх нового ядра безопасности (Nexus). Возникает вопрос: как защитить интерфейс между приложением и NCA? Всякий раз, когда доверенный компьютер общается с менее доверенным, язык их общения имеет критическое значение. Вы должны ожидать, что менее доверенное устройство будет перебирать всевозможные неожиданные комбинации команд, чтобы обмануть более доверенное.

Атаки на API криптографических модулей

Мы узнали много нового об безопасности API на примере сбоев в аппаратных модулях безопасности, используемых банками для защиты PIN-кодов и криптографических ключей для сетей ATM. В 1988 году Лонгли и Ригби выявили важность разделения типов ключей. Однако систематический анализ начался в 2000 году, когда задались вопросом: «Как убедиться, что не существует цепочки из 17 транзакций, которая приведет к утечке открытого ключа?». Изучая руководства, обнаружилась следующая уязвимость.

Атака XOR-To-Null-Key (XOR в нулевой ключ)

Аппаратные модули безопасности управляются транзакциями, отправляемыми хост-компьютерами. Модуль содержит мастер-ключи в защищенной от взлома памяти. Однако, поскольку памяти внутри устройства часто не хватает для всех ключей, рабочие ключи хранятся зашифрованными снаружи устройства. Способ шифрования этих ключей присваивает им своего рода «систему типов».
Например, в модуле безопасности VISA существовала транзакция для генерации терминального мастер-ключа (Terminal Master Key, KMT) для ATM. Поскольку безопасность ATM основана на двойном контроле, нужно было сгенерировать два отдельных компонента ключа, которые могли бы быть введены в ATM двумя разными людьми (например, менеджером и бухгалтером). Модуль генерировал компонент ключа и печатал его открытое значение, а также возвращал его значение, зашифрованное под соответствующим мастер-ключом K_M:
`VSM -> host: {KMTi}KM`
Также существовала транзакция, которая объединяла два таких компонента для

получения терминального ключа:
`Host -> VSM: {KMT1}KM, {KMT2}KM`
`VSM -> host: {KMT1 XOR KMT2}KM`

Идея заключалась в том, что KMT=KMT1⊕KMT2. Однако ничто не мешало программисту взять любой зашифрованный ключ и подать его дважды во вторую транзакцию. В результате получался известный терминальный ключ (ключ, состоящий из всех нулей, так как ключ XOR-ится сам с собой):
`Host -> VSM: {KMT1}KM, {KMT1}KM`
`VSM -> host: {KMT1 XOR KMT1}KM` (результат — нулевой ключ)

Теперь, когда мы внедрили известный ключ в систему, что мы можем с ним сделать? У модуля была транзакция для шифрования любого ключа, зашифрованного под K_M, под любым другим ключом, который сам зашифрован под K_M. Целью этой странной транзакции было позволить банку зашифровать свой ключ верификации PIN под терминальным мастер-ключом, чтобы отправить его в ATM для локальной проверки PIN. Более того, тип ключа «терминальный мастер-ключ» и тип ключа «ключ верификации PIN» были одинаковыми.
Эффект был разрушительным. Программист мог взять мастер-ключ верификации PIN банка (который также хранится вне модуля, зашифрованный с помощью K_M) и заставить модуль зашифровать его под нулевым ключом, который мы только что создали:
`Host -> VSM: {0}KM, {PIN}KM`
`VSM -> host: {PIN}0`

Теперь программист мог расшифровать мастер-ключ верификации PIN — «коронную жемчужину» банка — и вычислить PIN-код для любого клиентского счета. Назначение модуля безопасности было полностью уничтожено, банку проще было бы хранить PIN-коды в открытом виде.

Атака на IBM 4758

Следующая атака была найдена с использованием формальных методов. Майк Бонд построил формальную модель типов ключей и немедленно обнаружил еще одну ошибку. Тип ключа «коммуникационный ключ» использовался для MAC-ключей, которые обладали свойством: вы могли ввести MAC-ключ в открытом виде и получить его зашифрованным под... «любым ключом, зашифрованным под K_M». Это был еще один способ внедрить известный ключ в систему. Вы могли бы ввести номер счета, притворившись, что это MAC-ключ, и получить его зашифрованным с помощью ключа верификации PIN — это дало бы вам PIN-код клиента напрямую.

Вскоре после этого Майк Бонд провел фактическую атаку на IBM 4758. Это потрясло индустрию, так как устройство было сертифицировано по FIPS 140-1 level 4 (фактически правительство США заявило, что оно невзламываемо). В дополнение к атаке «встречи посередине» (meet-in-the-middle), он использовал еще одну скрытую ошибку проектирования — репликацию ключей.
По мере того как DES становился уязвимым к полному перебору, финансовые учреждения перешли на двухключевой тройной DES: блок шифровался левым ключом, расшифровывался правым и снова шифровался левым. IBM усложнила задачу, храня левые и правые ключи отдельно. Они приняли меры, чтобы их нельзя было перепутать (левые и правые ключи шифровались по-разному, имея разные типы), но не смогли жестко связать две половины вместе.
Атака заключалась в следующем:

1. Атакующий генерировал большое количество терминальных мастер-ключей и собирал контрольные значения (check values) каждого из них (контрольное значение вычисляется путем шифрования строки нулей под ключом).
2. Все контрольные значения сохранялись в хеш-таблице.
3. Выполнялся поиск методом грубой силы: угадывался ключ, шифровался фиксированный тестовый шаблон, и результат сравнивался с хеш-таблицей.
4. Используя пространство ключей в 2^56, атакующий, который мог сгенерировать 2^16 целевых ключей (что делалось за обеденный перерыв), мог найти совпадение с усилием около 2^40 (что занимало около недели на ПК).

Имея два одинарных DES-ключа с известными значениями внутри 4758, атакующий мог поменять местами левые и правые половины, чтобы получить известный истинный тройной DES-ключ. Этот известный ключ затем использовался для экспорта других ценных ключей из устройства. Атака была фактически реализована и продемонстрирована в прайм-тайм на телевидении.

Многосторонние вычисления и дифференциальные атаки на протоколы

Следующий набор атак на API модулей безопасности был инициирован Джолионом Клулоу в 2003 году. Они зависели от манипулирования деталями логики приложения для утечки информации.
Его первая атака эксплуатировала сообщения об ошибках. Один из форматов блока PIN, используемых для защиты, объединял PIN и номер счета с помощью

XOR, а затем шифровал их (это делалось для предотвращения атак, когда зашифрованный PIN не был связан с номером счета). Однако XOR оказался плохим способом сделать это. Если с PIN-блоком отправлялся неправильный номер счета, устройство расшифровывало блок, применяло XOR с номером счета, обнаруживало, что результат не является десятичным числом, и возвращало сообщение об ошибке. Подавая серию транзакций с неправильными номерами счетов, можно было быстро вычислить сам PIN.
Еще более простая атака была найдена против метода генерации PIN-кодов IBM. Номер счета клиента шифровался с помощью ключа верификации PIN, давая строку из 16 шестнадцатеричных цифр. Первые четыре преобразовывались в десятичные цифры для использования в качестве PIN с помощью таблицы децимализации (decimalisation table). Это давало «естественный PIN», к которому добавлялся «офсет» (offset), позволяющий клиенту выбрать запоминающийся PIN.
Проблема заключалась в том, что таблицу децимализации можно было манипулировать. Если установить таблицу в нули (0000000000000000), то будет сгенерирован и возвращен в зашифрованном виде PIN «0000». Затем вызов повторялся с таблицей 1000000000000000. Если зашифрованный результат менялся, атакующий знал, что вывод DES содержал «0» в первых четырех цифрах. Используя несколько десятков специально подобранных запросов, можно было определить значение PIN. Поскольку метод сравнивает повторные, но слегка измененные запуски одного и того же протокола, мы назвали эту атаку **дифференциальным анализом протоколов**.
Индустрия изначально отреагировала введением правил для допустимых таблиц децимализации (например, таблица должна иметь не менее восьми различных значений). Но это не помогло (попробуйте 0123456789012345, затем 1123456789012345 и т.д.). Единственным реальным решением было полностью убрать таблицу децимализации.

Атака на EMV

Казалось бы, после публикации атак в 2001 году разработчики модулей безопасности должны были стать осторожнее при добавлении новых транзакций. Но нет! Банковская индустрия продолжала требовать добавления новых функций, которые снова и снова ломали API.
Интересным недавним примером стала транзакция, заказанная консорциумом EMV для поддержки безопасного обмена сообщениями между смарт-картой и банковским модулем безопасности. Цель заключалась в том, чтобы при появлении банковской карты в онлайн-транзакции банк мог приказать ей изменить какой-либо параметр, например, новый ключ. Транзакция «Secure

Messaging For Keys» позволяла серверу приказать модулю безопасности зашифровать текстовое сообщение, за которым следовал ключ, используя ключ типа «для обмена со смарт-картами банка». Шифрование могло быть в режиме CBC или ECB, а текстовое сообщение могло иметь переменную длину.
Это позволяло атакующему выбрать длину сообщения так, чтобы ровно один байт целевого ключа пересекал границу блока шифрования. Этот байт можно было определить, отправляя серию сообщений, которые на один байт длиннее, где лишний байт перебирал все возможные значения, пока не будет найден байт ключа. Затем атакующий мог атаковать каждый из остальных байтов ключа один за другим. Таким образом, из модуля можно было извлечь любой экспортируемый ключ.

Атаки на API операционных систем

Второй класс атак на API связан с параллелизмом и хорошо иллюстрируется уязвимостями, обнаруженными Робертом Уотсоном в оболочках системных вызовов (system call wrappers).
Оболочки системных вызовов используются для усиления безопасности операционных систем; мониторы ссылок (reference monitors) являются примером, как и антивирусное ПО. Оболочки перехватывают вызовы, делаемые приложениями к ОС, анализируют их, проверяют и могут передавать их дальше с некоторыми изменениями: например, процессу Low, пытающемуся получить доступ к данным High, могут быть предоставлены фиктивные данные или сообщение об ошибке.
Когда Джон Андерсон предложил концепцию монитора ссылок в 1972 году, он потребовал, чтобы он был защищенным от взлома, непреодолимым и достаточно маленьким для верификации. На первый взгляд, современные оболочки таковы и есть — пока вы не начнете думать о времени. Оболочки обычно предполагают, что системные вызовы атомарны, но они таковыми не являются; современные ядра ОС очень сильно параллельны. Системные вызовы не атомарны ни по отношению друг к другу, ни по отношению к оболочкам. Существует множество возможностей для того, чтобы два системных вызова raced (соревновались) за доступ к общей памяти, что приводит к атакам типа **TOCTTOU** (Time-of-Check-to-Time-of-Use — «время проверки до времени использования»).
Типичная атака заключается в том, чтобы вызвать гонку (race) по пользовательской памяти, заставив ядро «уснуть» в неудобный момент, например, из-за page fault (ошибки страницы). В атаке на GSWTK Уотсон вызывал путь, имя которого выходило за границу страницы ровно на один байт, заставляя ядро спать, пока страница загружалась; затем он заменял путь в памяти.

Оказалось, что окна для таких гонок велики и надежны.
Процессоры становятся все более параллельными, и ОС оптимизируются для использования этого факта. Этот тип атак может становиться все более серьезной проблемой; более того, по мере того как инструменты анализа кода делают переполнения стека более редкими, вполне вероятно, что именно состояния гонки (race conditions) станут атакой выбора.
Что можно сделать, чтобы ограничить их? Единственным реальным решением является переписывание API. В идеальном мире операционные системы перешли бы на модель передачи сообщений, что исключило бы (или сильно сократило) проблемы параллелизма. Но это hardly практично для вендоров ОС, чьи бизнес-модели зависят от обратной совместимости. Прагматичным компромиссом является встраивание функций в саму ОС для борьбы с атаками параллелизма; Linux Security Modules делают это, как и Mac OS X 10.5 (основанный на TrustedBSD, над которым работал Уотсон).
Вкратце, API, предоставляемые стандартными операционными системами, имеют ряд известных слабостей, но решение в виде оболочки, помещающей еще один API перед уязвимым API, выглядит крайне хрупким в высокопараллельной среде. Оболочке пришлось бы понимать компоновку памяти достаточно полно, чтобы быть полностью эффективной, что сделало бы ее такой же сложной (и уязвимой), как и сама ОС, которую она пыталась защитить.

Методы защиты, коммерческая эксплуатация

Коммерческая эксплуатация (Commercial Exploitation)

Не все атаки Emsec связаны с тайным военным наблюдением или лабораторными взломами защищенных от вмешательства устройств. Упомянутые фургоны для обнаружения неплательщиков лицензий на телевизоры в Британии и шумиху вокруг машин для электронного голосования в Голландии. Существуют и маркетинговые приложения.
Например, американский организатор мероприятий SFX Entertainment отслеживает, какую музыку слушают клиенты в своих автомобильных радиоприемниках, когда они заезжают на парковку площадки. Это делается путем перехвата паразитных радиочастотных излучений от локального

генератора радиоприемника. Хотя это действие законно, оно вызывает раздражение у защитников конфиденциальности. То же самое оборудование продается автосалонам, операторам торговых центров и радиостанциям.

Методы защиты (Defenses)

Техники, которые можно использовать для защиты смарт-карт от активных угроз Emsec, аналогичны тем, что применяются в пассивных случаях, но с некоторыми отличиями. Использование временной случайности (джиттера) по-прежнему полезно, так как наивный противник может больше не знать, когда именно вставлять глитч. Однако опытный противник вполне может анализировать кривую энергопотребления процессора в реальном времени и сравнивать ее с кодом, чтобы выявить критические целевые инструкции. Кроме того, атаки по сбоям (fault attacks) трудно остановить с помощью джиттера, поскольку точное местоположение сбоя в коде обычно не имеет решающего значения.
В некоторых случаях достаточно оборонительного программирования. Например, описанный выше метод перебора PIN-кода предотвращается в более современных картах путем уменьшения счетчика попыток, запроса PIN-кода и последующего увеличения счетчика только в случае правильного ввода. Атаки на протоколы с открытым ключом можно значительно усложнить, если просто проверять результат вычислений.
Другие системы используют специализированное защитное оборудование, например, схему, которая объединяет сброс карты с цепью, обнаруживающей тактовые частоты, которые слишком высоки или слишком низки. Обычные сбросы включают деление тактовой частоты на два на несколько циклов, поэтому злоумышленник, нашедший способ отключить функцию мониторинга, скорее всего, обнаружит, что вообще не может сбросить карту при включении питания.
Оптическое зондировать (optical probing) гораздо сложнее; та атака, с которой сегодня может столкнуться карта, заключается в том, что противник помещает ее в тестовую установку, инициирует криптографическую операцию и направляет лазер на чип, чтобы вызвать единичный сбой устройства в точно измеренный момент времени. С помощью моторизованного столика можно проверять сотни различных целей в час.
В Кембридже разработали и создали прототип технологии защиты, которая может противостоять такому натиску, используя **двухрельсовую самосинхронную логику** (dual-rail self-timed logic). В такой логике, вместо передачи '1' через высокий уровень и '0' через низкий уровень на одной линии, используем две линии для каждого логического бита. Передаем '1' как 'HighLow',

а '0' как 'LowHigh'; конец логической операции — это 'LowLow'. Такая логика известна уже много лет и обладает свойством: если в систему каким-то образом попадает состояние 'HighHigh', оно имеет тенденцию распространяться по всему чипу, блокируя его и делая устройство неработоспособным до сброса. Превратили это в преимущество, определив 'HighHigh' как состояние «тревоги», и перепроектировали логические элементы так, чтобы единичный сбой вызывал распространение тревоги. Также «посыпали» чип множеством датчиков тревоги. И поскольку логика сбалансирована, энергопотребление гораздо меньше зависит от данных; сигналы, которые потенциально могут быть использованы для анализа энергопотребления, были ослаблены примерно на 20 дБ. Ряд других исследователей недавно начали изучать такие экзотические стили проектирования, и некоторые из них начали появляться в коммерческих продуктах. Избыточное проектирование может быть довольно дорогим, стоя как минимум в три раза дороже стандартной КМОП-структуры, но стоимость одного транзистора, возможно, уже снизилась до уровня, когда это имеет смысл.

Резюме по эмиссионной безопасности

Эмиссионная безопасность (Emsec) охватывает целый ряд угроз, при которых безопасность систем может быть скомпрометирована из-за демаскирующих излучений: будь то внедренные «жучки», непреднамеренное радиочастотное или кондуктивное электромагнитное излучение, либо излучения, индуцированные каким-либо внешним воздействием. Хотя изначально это была проблема национальных разведывательных сообществ, сегодня Emsec является реальной проблемой для компаний, производящих такие продукты безопасности, как смарт-карты и банкоматы. Многие из этих продуктов могут быть взломаны путем наблюдения за паразитными радиосигналами или кондуктивными излучениями. Защита от таких угроз не так проста, как может показаться.

Проблемы для исследований

Необходим всеобъемлющий набор стандартов эмиссионной безопасности для коммерческого использования. Военные стандарты (NATO SDIP-27 и USA NSTISSAM) засекречены, хотя и просочились в сеть. Гражданские стандарты RFI/EMC (IEC/CISPR22 и более строгий MIL-STD-461E) просто не были разработаны для защиты информации. Недавняя паника в Голландии по поводу перехвата

Tempest на машинах для голосования показывает, что стандарты необходимы, чтобы покупатели и поставщики оборудования могли решать, нужны ли они в каждом конкретном приложении.

Резюме по API

Интерфейсы со временем становятся богаче, грязнее и неприятнее. Ошибки проектирования интерфейсов широко распространены: от мира криптопроцессоров через различные встраиваемые системы вплоть до антивирусного программного обеспечения и самой операционной системы. Везде, где доверенное программное обеспечение взаимодействует с менее доверенным, интерфейс, скорее всего, «утечет» больше, чем предполагал разработчик доверенной части.
Сбои, как правило, возникают из-за сложности. Мы обсудили две основные истории: криптопроцессоры, которые накопили столько транзакций, что разработчики потеряли способность понимать их взаимодействие, и системные вызовы-оболочки (wrappers), которые пытаются аудировать и фильтровать вызовы к API операционной системы. На уровне чистой компьютерной науки мы могли бы рассматривать первые как примеры проблемы композиции или задачи безопасных многосторонних вычислений, а вторые — как сбои параллелизма. Однако это системы промышленного масштаба, а не «досочные» модели. Модуль безопасности может иметь сотни транзакций, и при каждом обновлении добавляется новая партия. На уровне приложения многие сбои можно рассматривать как взаимодействие функций (feature interactions). Существуют также специфические сбои, такие как медленная утечка информации из плохо спроектированных криптосистем, что не было бы серьезной проблемой в случае одной транзакции, но становится фатальным, когда противник может захватить сервер и внедрить сотни транзакций в секунду.
Безопасность API уже важна и становится еще важнее по мере того, как компьютеры становятся более сложными, разнообразными и параллельными. Если Microsoft выпустит оригинальные механизмы, обещанные для Trusted Computing в будущих версиях Windows, тогда API станут еще более критичными — разработчикам приложений будет предложено писать код, содержащий «более доверенную» часть (NCA) и «менее доверенное» обычное приложение. Даже профессионалы, работающие в компаниях по производству модулей безопасности, выпускают API с серьезными ошибками; какие шансы, что NCA принесут пользу, если все начнут проектировать свои API безопасности самостоятельно?
Что можно сделать? Один из извлеченных уроков заключается в том, что

«безопасный» процессор таковым не является, если его API недостаточно прост для понимания и проверки. Если вы отвечаете за криптографию банка, разумным решением будет заказать модуль безопасности, адаптированный под ваши нужды, чтобы он содержал только те транзакции, которые вам действительно нужны. Если это слишком неудобно или дорого, фильтруйте транзакции и отбрасывайте все, кроме самых необходимых.
Как показывает работа Роберта Уотсона, сложные API для высокопараллельных систем, вероятно, не могут быть исправлены таким образом. Если у вас есть критически важные приложения, зависящие от такой платформы, возможно, вам стоит составить план миграции.
Наконец, вероятно, возникнет целая масса проблем с API при взаимодействии приложений. По мере того как мир переходит к веб-сервисам, которые начинают общаться друг с другом, их API открываются для сторонних разработчиков — как это только что произошло, например, с Facebook. Одной сложности уже достаточно, позже в другой части моего материала обсудим, как сайты социальных сетей, в частности, подталкивают сложность политики безопасности к пределам, пытаясь охватить значительную часть человеческого социального поведения. А управление эволюцией API — одна из самых сложных задач в инженерии безопасности. Если combine это с тем, что экономическое давление толкает веб-приложения к небезопасным настройкам по умолчанию (например, делать всё доступным для поиска), и с высокой вероятностью того, что не все сторонние разработчики будут доброжелательными, становится ясно, что нас ждут проблемы.

Проблемы для исследований (Research Problems)

Подход компьютерной науки к проблеме безопасности API заключается в попытке адаптировать инструменты формальных методов для доказательства безопасности интерфейсов. По этому вопросу растет объем литературы, но методы все еще могут охватывать только довольно простые API. Проверка отдельного протокола достаточно сложна, и исследовательское сообщество потратило большую часть 1990-х годов, учась это делать. Однако протокол может состоять из 2–5 сообщений, тогда как криптопроцессор может иметь сотни транзакций.
Альтернативный подход, как и в мире протоколов, заключается в попытке выработать принципы надежности (robustness principles), которые будут направлять разработчика. Как и в той области, надежность в некоторой степени касается ясности и однозначности. Проверка того, что не существует какой-то непонятной последовательности транзакций, которая нарушит вашу политику

безопасности, достаточно сложна; а когда ваша политика даже не сформулирована точно, это выглядит невозможным.
Однако одной надежности недостаточно. На тактическом уровне история безопасности API преподнесла нам несколько уроков. Атомарность действительно имеет значение (вместе с согласованностью, изоляцией и долговечностью — другими желательными атрибутами систем обработки транзакций). На еще более низком уровне мы увидели пару веских причин, почему использовать исключающее ИЛИ (exclusive-or) для объединения ключей или PIN-кодов — это крайне глупая идея, раньше этого никто не понимал. Какие еще распространенные практики проектирования нам следует пересмотреть?