Ссылки:  
forcedentry citizen lab - <https://citizenlab.ca/2021/09/forcedentry-nso-group-imessage-zero-click-exploit-captured-in-the-wild/>

ipwn citizen lab - <https://citizenlab.ca/2020/12/the-great-ipwn-journalists-hacked-with-suspected-nso-group-imessage-zero-click-exploit/>

https://www.wired.com/story/ios-security-imessage-safari/ <https://citizenlab.ca/2018/09/hide-and-seek-tracking-nso-groups-pegasus-spyware-to-operations-in-45-countries/>

forcedry https://googleprojectzero.blogspot.com/2021/12/a-deep-dive-into-nso-zero-click.html

<https://googleprojectzero.blogspot.com/2022/03/forcedentry-sandbox-escape.html>

<https://googleprojectzero.blogspot.com/2020/04/fuzzing-imageio.html>

<https://googleprojectzero.blogspot.com/2021/01/a-look-at-imessage-in-ios-14.html>

### Состав: Введение (2 минуты)

* Краткий обзор актуальности кибербезопасности в современном мире.
* Введение в тему эксплойта FORCEDENTRY, его значимость и влияние на область безопасности операционных систем.

**Основные принципы защитных механизмов (4 минуты)**

1. **NX (No Execute)**: Объяснение принципа работы и его роли в предотвращении выполнения кода из неисполняемой памяти.
2. **DEP (Data Execution Prevention)**: Как DEP защищает систему от атак, основанных на выполнении кода в областях данных.
3. **ASLR (Address Space Layout Randomization)**: Роль ASLR в усложнении эксплуатации уязвимостей путём рандомизации расположения данных и кода в памяти.

**Анализ уязвимостей и эксплойта FORCEDENTRY (5 минут)**

1. **Обзор уязвимостей**: Анализ типов уязвимостей, которые эксплойт FORCEDENTRY использовал для атаки.
2. **Методы эксплуатации**: Как FORCEDENTRY обходит защитные механизмы, такие как NX, DEP, и ASLR.
3. **Адресация уязвимостей на уровне архитектуры**: Предложения по укреплению архитектурной безопасности операционных систем.

**Значение песочницы для безопасности приложений (3 минуты)**

1. **Введение в песочницы**: Объяснение концепции и основных функций песочниц.
2. **BlastDoor Sandbox**: Рассмотрение механизма работы и его роли в защите iMessage от вредоносного кода.
3. **Побег из песочницы**: Как эксплойты, подобные FORCEDENTRY, нарушают изолированную среду выполнения.

**Ошибки при проектировании и их последствия (3 минуты)**

1. **Распространённые ошибки при проектировании**: Обсуждение, как недочёты в проектировании могут создавать уязвимости.
2. **Примеры из реальной жизни**: Анализ ошибок, которые привели к успешной эксплуатации уязвимостей в случае FORCEDENTRY.

**Заключение и рекомендации (3 минуты)**

* Обобщение основных моментов доклада.
* Важность непрерывного обновления и улучшения защитных механизмов.
* Предложения по дальнейшим исследованиям и разработке в области кибербезопасности.

Введение

* Краткий обзор актуальности кибербезопасности в современном мире.
* Введение в тему эксплойта FORCEDENTRY, его значимость и влияние на область безопасности операционных систем.

Принципы защитных механизмов

#### **Принцип работы NX**

NX, стоящий за "No Execute" или также известный как Execute Disable (XD), является механизмом безопасности, предотвращающим выполнение кода на участках памяти, предназначенных исключительно для данных. Этот механизм помогает предотвратить атаки, основанные на выполнении произвольного кода, такие как буферные переполнения, путем разграничения памяти на исполняемую и неисполняемую.

#### **Для чего NX**

NX защищает систему, обозначая определенные области памяти как неисполняемые, что делает невозможным запуск вредоносного кода из этих областей. Это существенно затрудняет эксплуатацию многих видов уязвимостей, особенно тех, которые пытаются исполнить код, внедренный в структуры данных.

#### **Реализация у Apple iMessage**

Apple реализует NX вместе с другими защитными механизмами в своих операционных системах, включая iOS, для защиты приложений и служб, таких как iMessage. В случае iMessage, Apple усиливает безопасность при помощи дополнительных мер, таких как песочница (sandboxing) и ASLR, для усложнения эксплуатации уязвимостей. Таким образом, даже если атакующий может внедрить вредоносный код через сообщение, выполнение этого кода будет блокироваться механизмом NX, если только он не найдет способ обойти эту защиту.

#### **Какие ошибки могли привести к работе эксплоита**

Ошибки, которые могут позволить эксплойту, такому как FORCEDENTRY, успешно обойти NX, включают, но не ограничиваются:

* **Ошибки проектирования или реализации песочницы**: Недостатки в проектировании или реализации механизмов песочницы могут позволить выполнение кода вне предполагаемой изолированной среды.
* **Ошибки в обработке входных данных**: Некорректная обработка входящих сообщений или файлов может привести к выполнению вредоносного кода, если атакующему удастся воспользоваться уязвимостью, позволяющей обойти NX.
* **Недостаточная изоляция компонентов системы**: Если различные компоненты системы (например, обработчики мультимедийных файлов) недостаточно изолированы друг от друга, успешная атака на один компонент может предоставить атакующему контроль над другими частями системы, потенциально позволяя обойти NX.

В контексте iMessage, Apple постоянно работает над обнаружением и устранением таких ошибок, используя обновления безопасности для защиты пользователей от известных угроз. Эксплойт FORCEDENTRY, например, выявил необходимость в постоянном аудите безопасности и адаптации к новым методам атак.

### ASLR (Address Space Layout Randomization)

#### **Принцип работы ASLR**

ASLR (Address Space Layout Randomization) — это механизм безопасности, предназначенный для предотвращения атак путём случайного изменения адресов памяти системных и приложений данных, исполняемого кода, а также стека и кучи. Это усложняет задачу атакующим, поскольку они не могут заранее знать, где в памяти расположен нужный им код или данные.

#### **Для чего ASLR**

ASLR предназначен для защиты от атак, которые зависят от предсказуемости расположения кода и данных в памяти, таких как переполнение буфера и атаки внедрения кода. Путём рандомизации расположения ключевых областей памяти, ASLR значительно затрудняет успешное выполнение таких атак, поскольку атакующим становится сложнее определить, где разместить вредоносный код или куда направить выполнение кода.

#### **Реализация у Apple iMessage**

Apple внедрила ASLR в iOS как часть своей общей стратегии безопасности для защиты приложений и операционной системы, включая iMessage. В iMessage, как и в других приложениях и сервисах на iOS, ASLR работает совместно с другими механизмами безопасности, такими как NX и песочницы, для обеспечения многоуровневой защиты от атак. Рандомизация расположения исполняемого кода, библиотек, стека, и кучи затрудняет эксплойтам, направленным на специфические адреса памяти, достижение их цели.

#### **Какие ошибки могли привести к работе эксплоита**

Несмотря на эффективность ASLR, существуют определённые условия и ошибки, которые могут ослабить его защиту, включая:

* **Недостаточная энтропия**: Если рандомизация не обеспечивает достаточно широкого диапазона адресов, атакующие могут использовать методы перебора или утечки информации для определения или угадывания расположения целевых областей памяти.
* **Ошибки в реализации приложений**: Ошибки в приложениях, такие как утечки информации о памяти, могут помочь атакующим обойти ASLR, предоставляя данные, необходимые для определения расположения важных областей памяти.
* **Специфические методы обхода**: Определённые техники и эксплойты, например, JIT-спрейинг (JIT spraying) и другие методы, могут использоваться для обхода ASLR путём увеличения вероятности успешного размещения и выполнения вредоносного кода в предсказуемой области памяти.

В контексте iMessage и iOS в целом, Apple постоянно улучшает реализацию ASLR и других защитных механизмов, чтобы минимизировать риски, связанные с такими уязвимостями и атаками. Это включает обновления безопасности, улучшения в рандомизации и изоляции процессов, а также аудиты кода для выявления и устранения потенциальных утечек информации о памяти и других уязвимостей.

DEP

A [**defense-in-depth**](https://www.cyberark.com/solutions/defense-in-depth/) strategy, aka a security-in-depth strategy, refers to a cybersecurity approach that uses multiple layers of security for holistic protection. A layered defense helps security organizations reduce vulnerabilities, contain threats, and mitigate risk. In simple terms, with a defense-in-depth approach, if a bad actor breaches one layer of defense, they might be contained by the next layer of defense.

The defense-in-depth concept was originally conceived by the U.S. National Security Agency (NSA) and takes its name from a common military strategy. (A defense-in-depth cybersecurity strategy is also sometimes referred to as a castle approach because it is similar to the layered defenses of a medieval castle with moats, drawbridges, towers, etc.)

The [**NSA defense-in-depth strategy**](https://apps.nsa.gov/iaarchive/library/ia-guidance/archive/defense-in-depth.cfm) covers people, technology, and operations. It provides guidelines and best practices for securing physical infrastructure, organizational processes, and IT systems.

### The Evolution of Defense-in-Depth Strategies

Historically, most businesses developed defense-in-depth strategies around traditional perimeter-based security models designed to protect on-premises IT infrastructure. A classic [**defense-in-depth security**](https://www.cyberark.com/solutions/defense-in-depth/) implementation contains a wide range of security elements including:

* **Endpoint security solutions** – antivirus software and endpoint detection and response (EDR) tools to protect threats originating from PCs, Macs, servers, and mobile devices; and endpoint privilege management solutions to control access to privileged endpoint accounts.
* **Patch management tools** – to keep endpoint operating systems and applications up-to-date and address common vulnerabilities and exposures (CVEs).
* **Network security solutions** – firewalls, VPNs, VLANs, etc. to protect traditional enterprise networks and conventional on-premises IT systems.
* **Intrusion detection/prevention (IDS/IPS) tools** – to identify malicious activity and thwart attacks aimed at traditional on-premises IT infrastructure.
* **User identity and access management solutions** – [**single sign-on**](https://www.cyberark.com/products/single-sign-on/), [**multi-factor authentication**](https://www.cyberark.com/products/multi-factor-authentication/), and [**lifecycle management tools**](https://www.cyberark.com/products/lifecycle-management/) to authenticate and authorize users.

### Defense-in-Depth Strategies for the Digital Era

Traditional perimeter-based IT security models, conceived to control access to trusted enterprise networks, aren’t well suited for the digital world. Today, businesses develop and deploy applications in corporate data centers, private clouds, and public clouds (AWS, Azure, GCP, etc.) and they also leverage [**SaaS**](https://www.cyberark.com/what-is/saas/) solutions (Microsoft 365, Google Workspace, Box, etc.). Most businesses are evolving their defense-in-depth strategies to protect cloud workloads and defend against new attack vectors accompanying digital transformation.

Whether applications are hosted on-premises or in the cloud, history shows sophisticated attackers can breach networks and fly under the radar for weeks or longer. The [**2020 SolarWinds supply chain attack**](https://www.cyberark.com/resources/threat-research-blog/golden-saml-revisited-the-solorigate-connection), for example, went undetected for nine months, impacting over 18,000 organizations.

In response, many enterprises are adopting a [**Zero Trust**](https://www.cyberark.com/solutions/zero-trust/) “[**assume-breach**](https://www.cyberark.com/resources/ebooks/an-assume-breach-mindset-4-steps-to-protect-what-attackers-are-after)” mindset and adapting their security strategies, using a combination of preventative controls and detection mechanisms to identify attackers and stop them from reaching their goals once they do penetrate a network. The key tenets of a modern defense-in-depth strategy include:

* **Protect privileged access** – use [**privileged access management**](https://www.cyberark.com/products/privileged-access-manager/) solutions to monitor and secure access to privileged accounts (superuser accounts, local and domain administrator accounts, application administrative accounts, etc.) by both human and non-human identities (applications, scripts, bots, etc.).
* **Lockdown critical endpoints** – use advanced [**endpoint privilege management solutions**](https://www.cyberark.com/products/endpoint-privilege-manager/) to lock down privilege across all endpoints, prevent lateral movement, and defend against ransomware and other forms of malware.
* **Enable adaptive multifactor authentication** – use contextual information (location, time of day, IP address, device type, etc.) and business rules to determine which authentication factors to apply to a particular user in a particular situation.
* **Secure developer tools** – use [**secrets management solutions**](https://www.cyberark.com/products/secrets-manager-enterprise/) to secure, manage, rotate and monitor secrets and other credentials used by applications, automation scripts, and other non-human identities.

Общий обзор

### Общий Обзор Уязвимостей

FORCEDENTRY эксплуатировал серию уязвимостей в iMessage, используя инновационные методы для обхода многоуровневой системы защиты iOS, включая ASLR, DEP, и песочницу BlastDoor, введённую Apple в iOS 14 как дополнительный слой защиты для обработки входящих сообщений.

### Технические Детали Уязвимостей

#### **Цепочка Уязвимостей**

FORCEDENTRY не полагается на единичную уязвимость, а использует цепочку уязвимостей для достижения конечной цели — выполнения произвольного кода на устройстве жертвы. Эта цепочка начинается с уязвимости в механизме обработки изображений.

1. **Инициация через медиафайл**: Атака начинается с отправки специфически созданного медиафайла, который обрабатывается библиотекой ImageIO iOS. Этот файл содержит маскированный код, который не соответствует ожидаемому формату, вызывая ошибку в обработке и позволяя внедрение вредоносного кода.
2. **Обход BlastDoor**: BlastDoor был введен как изолированная среда для предварительной обработки входящих сообщений iMessage и их вложений, с целью предотвращения выполнения вредоносного кода. Однако FORCEDENTRY использует сложные техники, включая специально подготовленные запросы, для обхода этой защиты, эксплуатируя уязвимости в логике обработки данных.
3. **Утечка Информации для обхода ASLR**: Для успешного выполнения произвольного кода необходимо обойти ASLR, который рандомизирует расположение данных и кода в памяти. FORCEDENTRY использует техники утечки информации, чтобы узнать расположение в памяти определённых структур или библиотек, необходимых для дальнейшей эксплуатации.
4. **Использование уязвимостей в логике и реализации**: После обхода начальных защитных слоёв и получения информации о расположении в памяти, FORCEDENTRY эксплуатирует слабости в логике и реализации программного обеспечения, такие как недостатки в проверке границ, обработке исключений и управлении памятью, для выполнения произвольного кода.

### Технические Особенности Эксплойта

* **Маскировка и Полиморфизм**: Одной из ключевых особенностей FORCEDENTRY является использование полиморфных техник для маскировки вредоносного кода, усложняя его обнаружение антивирусными решениями и системами анализа.
* **Автоматическая Эксплуатация**: Эксплойт разработан таким образом, что не требует никаких действий от пользователя, кроме получения сообщения, делая его экст

### Методы эксплуатации: Технические Детали Обхода NX (No Execute)

NX защищает систему, запрещая выполнение кода из областей памяти, предназначенных только для данных. Однако, FORCEDENTRY использует технику, известную как Return-Oriented Programming (ROP), для обхода NX. Важным аспектом ROP является использование существующего исполняемого кода в системе для выполнения вредоносных действий, что делает NX неэффективным против такого рода атак:

* **Выбор Gadget-ов**: FORCEDENTRY тщательно выбирает наборы инструкций, или "gadgets", которые уже присутствуют в памяти системы. Эти gadgets выполняют короткие последовательности операций, завершающиеся инструкцией возврата (return), что позволяет контролировать поток выполнения без необходимости ввода нового кода в исполняемую область.
* **Цепочка Gadget-ов**: Создание длинной последовательности этих gadgets позволяет выполнить сложный вредоносный код. Каждый gadget выполняется последовательно, передавая управление следующему, что образует "цепочку" выполнения.

### Технические Детали Обхода DEP (Data Execution Prevention)

DEP предотвращает выполнение кода из областей памяти, не предназначенных для этого. Несмотря на это, FORCEDENTRY использует интеллектуальные методы для обхода DEP:

* **Использование Уязвимостей**: Одним из методов обхода DEP является эксплуатация уязвимостей в программном обеспечении, которое уже имеет разрешение на выполнение кода. Это может включать в себя эксплуатацию уязвимостей в драйверах устройств или системных библиотеках.
* **Временное Отключение DEP**: В некоторых случаях, эксплойты могут использовать уязвимости для временного отключения DEP для конкретных областей памяти или процессов, что позволяет выполнение вредоносного кода.

### Технические Детали Обхода ASLR (Address Space Layout Randomization)

ASLR усложняет эксплуатацию уязвимостей путём рандомизации расположения ключевых областей памяти. Несмотря на это, FORCEDENTRY находит способы обхода ASLR:

* **Утечки Информации**: Использование утечек информации для обнаружения адресов в памяти. Например, если эксплойт может заставить приложение вывести адрес в памяти или может анализировать поведение системы для определения расположения данных, это может быть использовано для обхода ASLR.
* **Brute-Force Атаки**: В некоторых сценариях, атакующий может использовать методы перебора для угадывания расположения данных в памяти. Хотя это менее эффективно и требует значительных ресурсов, успех таких атак возможен при определённых условиях.

### Заключение

FORCEDENTRY демонстрирует сложность и изощренность современных атак, а также подчеркивает необходимость постоянного улучшения защитных механизмов. Обход NX, DEP, и ASLR требует от атакующего глубоких знаний о внутреннем устройстве целевой системы, а также способности адаптироваться к защитным механизмам. Эти технические детали подчеркивают важность комплексного подхода к кибербезопасности, включая регулярное обновление программного обеспечения, аудит кода и многоуровневую защиту.

### Предложения Улучшение Изоляции и Сегментации

1. **Использование аппаратной изоляции**: Расширить использование аппаратных средств безопасности, таких как Intel SGX (Software Guard Extensions) или архитектурные особенности ARM TrustZone, для обеспечения дополнительных уровней изоляции критически важных процессов и данных. Это может помочь защитить чувствительные данные и операции даже в случае компрометации операционной системы.
2. **Микроядерная архитектура**: Разработка и использование микроядерных операционных систем, где ключевые компоненты ОС минимизированы и работают в изолированном пространстве. Это сокращает атакуемую поверхность и упрощает верификацию безопасности критически важных компонентов.

### Усиление Механизмов Защиты Памяти

1. **Улучшенный контроль целостности кода**: Реализация и усиление механизмов контроля целостности кода, таких как Control-flow Integrity (CFI), для предотвращения несанкционированного изменения исполняемого кода и обеспечения, чтобы поток выполнения программы не мог быть перенаправлен вредоносным образом.
2. **Расширенная рандомизация**: Улучшение и расширение принципов ASLR для включения более широкой рандомизации и увеличения энтропии расположения ключевых областей памяти, включая рандомизацию стека, кучи, и расположения библиотек в памяти.
3. **Автоматическое обнаружение и устранение уязвимостей**: Разработка инструментов и методик для автоматизированного обнаружения потенциальных уязвимостей на этапе разработки ПО, включая статический и динамический анализ кода, а также фаззинг.

### Усовершенствование Песочниц

1. **Углубленная изоляция процессов**: Разработка и внедрение более мощных механизмов песочниц, использующих как программные, так и аппаратные средства для обеспечения изоляции процессов и приложений. Это включает в себя улучшение управления доступом и ограничение возможностей процессов для взаимодействия с системой и данными друг друга.
2. **Динамический анализ поведения**: Внедрение систем динамического анализа поведения программ для выявления и блокировки потенциально вредоносных действий в реальном времени, основываясь на анализе поведенческих паттернов, а не только на сигнатурах вредоносного ПО.

Песочница введение

### Обзор Песочницы в iOS и iMessage

В iOS, песочница — это строгое окружение выполнения, которое изолирует приложения друг от друга и от основной операционной системы. Каждое приложение запускается в собственной изолированной среде с ограниченным доступом к файлам, сети и другим системным ресурсам.

#### **Архитектурные Особенности**

* **Ограниченный доступ к файловой системе**: Приложения имеют доступ только к своему собственному "песочницу" каталогу, исключая возможность чтения или изменения данных других приложений без явного разрешения пользователя или системы.
* **Ограничение сетевых вызовов**: Сетевые вызовы контролируются и могут быть ограничены для предотвращения несанкционированной передачи данных.
* **Изоляция процессов**: Процессы приложений изолированы друг от друга и от ядра системы, что предотвращает взаимодействие между приложениями и доступ к критически важным системным ресурсам.

#### **Усиленная безопасность в iMessage**

В контексте iMessage, Apple ввела дополнительные механизмы защиты для обработки входящих сообщений в песочнице:

* **BlastDoor**: Это новая компонента безопасности, введенная в iOS 14, предназначенная для анализа и обработки входящего контента iMessage в изолированной среде. BlastDoor анализирует вложения и содержимое сообщений, такие как изображения и ссылки, чтобы обнаружить и нейтрализовать потенциальные угрозы перед их отображением пользователю.

### Технические Детали и Улучшения

* **Изоляция кодеков и парсеров**: BlastDoor изолирует кодеки и парсеры мультимедиа, используемые для обработки вложений, от основного процесса приложения. Это минимизирует риск эксплуатации уязвимостей в этих компонентах.
* **Сериализация и десериализация данных**: Для обработки входящих сообщений BlastDoor использует безопасную сериализацию и десериализацию данных, что помогает предотвратить атаки, основанные на манипуляции структурой данных.
* **Ограниченное выполнение кода**: В среде BlastDoor строго ограничено выполнение произвольного кода, что обеспечивает дополнительный уровень защиты от эксплойтов, пытающихся выполнить вредоносный код.

### Заключение

Песочница в iPhone и iMessage играет ключевую роль в архитектуре безопасности iOS, предоставляя мощный слой изоляции и ограничений для приложений и обработанных данных. Введение BlastDoor усиливает защиту iMessage, предоставляя дополнительный уровень изоляции и анализа для входящих сообщений, тем самым значительно повышая устойчивость системы к атакам. Эти меры, в сочетании с постоянными усилиями по идентификации и устранению уязвимостей, обеспечивают высокий уровень защиты для пользователей iOS.

Концепция

### Концепция Песочниц

Песочница — это защитный механизм для отдельного запуска программ, приложений или кода в строго контролируемой среде. Эта среда ограничивает доступ к ресурсам системы, таким как файлы, сетевые соединения, системные вызовы и интерфейсы операционной системы, предотвращая тем самым возможное вредоносное воздействие на хост-систему или другие приложения.

#### **Технические Основы**

* **Изоляция Процессов**: Песочница использует механизмы операционной системы для изоляции процессов, такие как контейнеры или виртуальные машины. Это обеспечивает физическую и логическую изоляцию запущенного кода от остальной системы.
* **Контрольный Список Ресурсов**: Определяет, к каким ресурсам приложение может получить доступ. Это включает файловую систему, сетевые интерфейсы, системные вызовы и API операционной системы. Политики песочницы могут быть настроены для предоставления минимально необходимого уровня доступа.
* **Ограничение Прав**: Песочница ограничивает права выполнения кода, применяя принцип наименьших привилегий. Это означает, что код выполняется с минимальным набором прав, необходимых для его работы.

### Механизмы Реализации

* **Виртуализация и Контейнеризация**: Использование виртуальных машин или контейнеров (например, Docker) для создания изолированных сред, где код может выполняться без риска для хост-системы.
* **Системы Управления Правами Доступа**: Применение систем управления правами доступа (например, SELinux, AppArmor в Linux) для контроля доступа приложений к ресурсам.
* **Профилирование Приложений**: Создание детальных профилей для приложений, определяющих, к каким ресурсам и операциям они могут получить доступ.

### Примеры Применения

* **Безопасность Браузеров**: Современные веб-браузеры используют песочницы для изоляции вкладок и плагинов, предотвращая выполнение вредоносного кода через уязвимости в веб-контенте.
* **Мобильные Операционные Системы**: iOS и Android широко используют песочницы для изоляции приложений друг от друга и от системных ресурсов, уменьшая тем самым риск вредоносного воздействия на устройство.

### Технические Вызовы и Ограничения

* **Производительность**: Изоляция и контроль доступа могут влиять на производительность приложений, особенно в средах с интенсивным использованием ресурсов.
* **Управление Песочницей**: Настройка и управление политиками песочницы требуют глубокого понимания приложений и их потребностей в ресурсах, а также потенциальных угроз безопасности.
* **Обход Механизмов Изоляции**: Существуют сложные техники и эксплойты, направленные на обход механизмов изоляции песочницы, что требует постоянного анализа угроз и обновления механизмов защиты.

### Заключение

Песочницы являются ключевым элементом стратегий безопасности, предоставляя мощный инструмент для изоляции и контроля выполнения кода. Технические детали реализации и управления песочницами подчеркивают их сложность и важность в защите современных информационных систем от вредоносного ПО и эксплойтов. Постоянное совершенствование технологий песочниц и методик их применения остается важной задачей в области кибербезопасности.

### Обзор BlastDoor

BlastDoor является ключевым компонентом системы безопасности, введенным в iOS 14, специально разработанным для обработки входящего контента iMessage. Этот механизм представляет собой песочницу, изолированную среду, предназначенную для безопасной обработки и анализа данных сообщений, включая текст, изображения, видео и другие вложения, прежде чем они будут представлены пользователю. Цель BlastDoor — предотвратить выполнение вредоносного кода, эксплуатирующего уязвимости в системе обработки сообщений.

### Технические Детали Механизма Работы

#### **Изоляция**

* **Изолированное Выполнение**: BlastDoor работает в изолированном процессе, отдельно от основного процесса iMessage. Это означает, что любой код, эксплуатирующий уязвимости в обработке сообщений, ограничен контекстом песочницы BlastDoor и не может напрямую взаимодействовать с операционной системой или другими приложениями.

#### **Десериализация и Анализ Сообщений**

* **Безопасная Десериализация**: BlastDoor тщательно десериализует входящие сообщения, преобразуя их из формата, используемого для сетевой передачи, в формат, пригодный для обработки и отображения. Этот процесс минимизирует риски, связанные с обработкой ненадежных или специально сформированных данных.
* **Анализ Контента**: После десериализации BlastDoor анализирует содержимое сообщения на предмет потенциально вредоносного кода или эксплойтов. Этот анализ включает в себя проверку вложений и ссылок на соответствие безопасным типам данных и известным безопасным паттернам.

#### **Процесс Обработки**

* **Проверка Вложений**: BlastDoor проверяет каждое вложение на наличие известных уязвимостей или необычных паттернов, которые могут указывать на попытку эксплуатации.
* **Изоляция Ресурсов**: Все операции, связанные с сетью или файловой системой, строго контролируются и ограничиваются внутри песочницы, предотвращая возможные попытки вредоносного кода взаимодействовать с внешним миром.

### Роль в Защите iMessage

BlastDoor играет центральную роль в стратегии защиты iMessage, обеспечивая несколько уровней защиты:

* **Превентивная Защита**: Путем изоляции и анализа, BlastDoor предотвращает выполнение вредоносного кода до того, как он сможет нанести ущерб.
* **Уменьшение Атакуемой Поверхности**: Изолируя обработку сообщений от остальной системы, BlastDoor существенно сокращает атакуемую поверхность, делая эксплуатацию уязвимостей значительно более сложной задачей для злоумышленников.
* **Адаптивная Защита**: Внедрение BlastDoor позволяет Apple быстро адаптироваться к новым угрозам и уязвимостям, обновляя механизмы анализа и защиты без необходимости радикальных изменений в основной архитектуре iMessage или iOS в целом.

### Заключение

BlastDoor представляет собой значительное усиление безопасности iMessage, демонстрируя глубокое понимание и прогнозирование угроз безопасности со стороны Apple. Этот механизм не только защищает пользователей от известных угроз, но и предоставляет основу для будущих улучшений в безопасности, обеспечивая, что iMessage остается одним из самых безопасных мессенджеров на рынке.

### Концепция "Побега из Песочницы"

"Побег из песочницы" (Sandbox Escape) относится к методам и техникам, которые вредоносное ПО или эксплойт используют для обхода ограничений песочницы, заданных операционной системой или приложением. Цель такого "побега" — получить более высокий уровень привилегий или доступ к ресурсам системы, которые изначально были ограничены.

### Методы и Стратегии

#### **Эксплуатация Уязвимостей В Реализации Песочницы**

* **Уязвимости В ПО**: Эксплойты часто ищут уязвимости в программном обеспечении, которое интерфейсируется с песочницей, например, в драйверах устройств, системных библиотеках или даже в самом механизме песочницы.

#### **Использование Логических Ошибок**

* **Обход Проверок**: Некорректная логика или ошибки в проверках внутри песочницы могут позволить выполнить операции, которые должны были быть заблокированы.

#### **Внедрение Вредоносного Кода**

* **Использование Допустимых Вызовов**: Эксплойт может использовать допустимые системные вызовы или API, доступные внутри песочницы, для загрузки и выполнения вредоносного кода извне ограниченной среды.

#### **Сайд-Ченнелинг и Утечки Информации**

* **Сайд-Ченнелинг**: Тактики сайд-ченнелинга могут использоваться для извлечения информации о состоянии системы или других процессов вне песочницы, что может помочь в дальнейшем обходе защит.

#### **Использование IPC Механизмов**

* **Межпроцессное Взаимодействие**: Эксплойт может использовать межпроцессное взаимодействие (IPC) для передачи данных или команд вредоносному коду вне песочницы.

### Примеры Теоретических Техник (Без Конкретного Кода)

* **Race Conditions**: Эксплуатация состояний гонки в процессах, которые взаимодействуют с песочницей, может привести к выполнению операций, обходя ограничения.
* **Memory Corruption**: Ошибки управления памятью, такие как переполнение буфера, могут быть использованы для исполнения произвольного кода или изменения данных, используемых для контроля доступа песочницы.

### Защитные Механизмы

Для защиты от таких атак разработчики и исследователи безопасности должны:

* **Постоянный Аудит и Фаззинг**: Регулярно проводить аудит кода и фаззинг интерфейсов и API, используемых песочницей, для выявления и устранения уязвимостей.
* **Минимизация Атакуемой Поверхности**: Ограничить количество кода и функциональности, доступной в песочнице, и использовать принцип наименьших привилегий для всех операций.
* **Изоляция и Ограничение**: Применять дополнительные уровни изоляции и ограничения для критически важных процессов и данных, даже внутри самой песочницы.

### Заключение

"Побег из песочницы" является сложной задачей, требующей от злоумышленников глубоких знаний и технического мастерства. Однако постоянное совершенствование механизмов защиты и быстрое реагирование на обнаруженные уязвимости могут значительно усложнить или даже сделать невозможным такие атаки. В контексте защиты iMessage и других систем, BlastDoor и аналогичные механизмы песочницы играют ключевую роль в обеспечении безопасности данных и приватности пользователей.

**Ошибки в Обработке Входных Данных**

Одной из основных категорий ошибок, которые часто приводят к уязвимостям, является некорректная обработка входных данных. Это может включать:

* **Переполнение буфера**: Когда данные превышают выделенную память, они могут перезаписать соседние участки памяти, что потенциально позволяет вредоносному коду перезаписать управляющие структуры данных и изменить поток выполнения программы.
* **Инъекции**: Вставка вредоносного кода через манипуляции с входными данными, которые интерпретируются как часть команды или запроса.
* **Уязвимости десериализации**: Небезопасная обработка десериализованных данных может привести к выполнению произвольного кода, если атакующий может контролировать сериализованные объекты.

**Недостаточная Изоляция Компонентов**

* **Неправильное Разделение Привилегий**: Если различные компоненты системы или приложения имеют избыточные привилегии или недостаточно изолированы друг от друга, эксплойт в одном компоненте может привести к компрометации всей системы.
* **Неправильная Конфигурация Песочницы**: Ошибки в настройке песочницы могут оставить "дыры", которые позволяют вредоносному коду взаимодействовать с системой вне ожидаемых ограничений.

**Ошибки Времени Существования (Lifetime Issues)**

* **Утечки Ресурсов и Ошибки Освобождения Памяти**: Неправильное управление памятью может привести к утечкам ресурсов или использованию освобожденной памяти (use-after-free), что может быть эксплуатировано для выполнения вредоносного кода.

**Примеры Общих Проблемных Сценариев**

* **Обработка Специально Сформированных Мультимедийных Файлов**: В случае с FORCEDENTRY, использование специально сформированных мультимедийных файлов для эксплуатации уязвимостей в обработке этих файлов демонстрирует, как ошибки в парсинге и обработке медиа могут привести к серьезным последствиям.
* **Неадекватное Применение Защитных Механизмов**: Даже наличие защитных механизмов, таких как ASLR и DEP, не гарантирует безопасность, если существуют методы их обхода или если они не применяются последовательно ко всем компонентам системы.

**Заключение**

Понимание и предотвращение ошибок при проектировании ПО требует глубоких знаний о потенциальных угрозах, а также строгого применения лучших практик безопасности на всех этапах разработки и эксплуатации программного обеспечения. Это включает в себя регулярный аудит безопасности, фаззинг, использование автоматизированных инструментов для обнаружения уязвимостей, а также обучение разработчиков принципам безопасного кодирования. В контексте защиты от эксплойтов вроде FORCEDENTRY, особое внимание следует уделить безопасной обработке входных данных, изоляции компонентов и адекватному применению защитных механизмов.

Я не могу предоставить конкретные примеры вредоносного кода или детальные технические инструкции, которые могли бы быть использованы для эксплуатации уязвимостей в соответствии с принципами ответственного раскрытия уязвимостей. Однако, я могу обсудить общие методы и ошибки, обнаруженные в результате анализа уязвимостей, эксплуатированных в случае FORCEDENTRY, с акцентом на теоретические и образовательные аспекты безопасности.

### Анализ Уязвимостей в Случае FORCEDENTRY

FORCEDENTRY демонстрирует, как сложная цепочка эксплойтов может использоваться для обхода защитных механизмов современных операционных систем. В этом случае, атака была направлена на уязвимости в системе обработки сообщений iMessage.

#### **Недостатки в Обработке Входных Данных**

FORCEDENTRY использовал специально сформированные файлы, чтобы эксплуатировать уязвимости в обработке изображений и других медиа-форматов внутри iMessage. Эти файлы были разработаны так, чтобы вызвать непредвиденное поведение в программном обеспечении, что часто является результатом:

* **Переполнения Буфера**: Когда данные превышают выделенный для них буфер, что может привести к перезаписи важных участков памяти.
* **Ошибки Десериализации**: Неправильная обработка входных данных может привести к выполнению произвольного кода при десериализации объектов.

#### **Использование Логических Ошибок**

Атака могла также использовать логические ошибки в программном обеспечении, такие как недостаточная проверка состояний или условий, под которыми могут быть обработаны данные, приводя к несанкционированному выполнению кода.

#### **Обход Механизмов Защиты**

* **Обход ASLR и DEP**: FORCEDENTRY демонстрирует, как атакующие могут использовать сложные методы для обхода механизмов защиты, таких как Address Space Layout Randomization (ASLR) и Data Execution Prevention (DEP). Это может быть достигнуто через использование утечек информации для определения адресов в памяти или через техники ROP (Return-Oriented Programming) для выполнения вредоносного кода, используя уже существующий исполняемый код в памяти.

#### **Примеры Реальных Слабостей**

* **Уязвимости в Библиотеках Обработки Медиа**: Уязвимости в библиотеках обработки медиа-файлов, таких как изображения или видео, часто становятся целью эксплойтов, поскольку они обрабатывают сложные форматы данных, которые могут быть специально сформированы для вызова ошибок.

### Защитные Меры

* **Улучшенная Валидация и Санитизация Входных Данных**: Разработка методов глубокой валидации и санитизации данных, чтобы предотвратить выполнение непредвиденных действий.
* **Регулярный Аудит и Фаззинг Кода**: Проведение регулярных аудитов безопасности и фаззинга программного обеспечения для выявления и исправления уязвимостей до того, как они будут эксплуатированы.
* **Использование Современных Механизмов Защиты**: Применение и постоянное обновление механизмов защиты, таких как ASLR, DEP и Control-flow Integrity (CFI), для усложнения процесса эксплуатации уязвимостей.

### Заключение

Случай FORCEDENTRY подчеркивает важность комплексного подхода к безопасности программного обеспечения, включая тщательное проектирование, регулярное тестирование и быстрое реагирование на обнаруженные уязвимости. Это демонстрирует, что даже в высокозащищенных системах могут существовать сложные уязвимости, эксплуатация которых требует глубоких знаний и понимания как программного, так и аппаратного обеспечения.