ТОКЕНИЗАЦИЯ

Принципы функционирования и выбор решения

Ak Kamal Security

СОДЕРЖАНИЕ

Перевод и подготовка материала проведены компанией Ak Kamal Security. Автором оригинального материала под названием “Understanding and Selecting a Tokenization Solution” является компания Securosis (*www.securosis.com*).

ВВЕДЕНИЕ

Одной из самых важных задач в информационной безопасности является защита конфиденциальных данных, которая является комплексной и довольно сложной. Даже самые опытные профессионалы подходят к этому делу с изрядной долей волнения. Организация эффективной защиты конфиденциальных данных в рамках приложений, баз данных, устройств хранения и групп серверов сама по себе является непростой и еще больше усложняется при использовании разнородных систем и противоречивых требованиях бизнеса.

Как правило, в случае с конфиденциальными данными мы рассматриваем разнообразные решения с использованием шифрования, но в последние годы заметен рост интереса к другой технологии - токенизации.

Должным образом реализованное шифрование является одним из наиболее эффективных средств обеспечения безопасности конфиденциальных данных. Оно позволяет обеспечить доступ к данным только авторизованным пользователям и защищает данные, как при хранении, так и при передаче. Но шифрование не единственный вариант защиты – есть и альтернативные методы. Причем иногда самым правильным решением будет не попытка защитить конфиденциальные данные шифрованием, а вообще отказаться от их передачи.

Токенизация как раз является технологией, которая предоставляет эту возможность – ее принцип заключается в подмене реальных конфиденциальных данных некими значениями – токенами. Токенизация в чем-то схожа с шифрованием – обе технологии занимаются маскированием реальных данных, но подход к этому процессу в корне отличается. В случае шифрования, процесс сокрытия данных обратим при наличии правильного ключа. Любой человек, получив ключ шифрования, сможет восстановить исходные данные.

В случае с токенизацией, процесс не является обратимым, ведь вместо данных при токенизации используется токен, не несущий в себе никакой конфиденциальной информации, и лишь логически связанный с реальными данными, которые хранятся в хорошо защищенной базе данных. При этом токен может иметь тот же формат (размер и структура), что и оригинальные данные, что позволяет минимизировать необходимость внесения изменений в работающие с ним приложения. Но при этом похищать токен бессмысленно, так как он не поможет получить никакие реальные конфиденциальные данные.

Главное преимущество токенизации в том, что токены могут полноценно использоваться внутри среды их применения, но совершенно бесполезны вне ее. Токенизация идеально подходит для защиты конфиденциальных данных, таких как номера кредитных карт, номера социального страхования, а также любых других данных, которые злоумышленники регулярно похищают с целью использования или продажи на “черном рынке”. Если злоумышленникам не удастся взломать сам сервер токенизации, чтобы получить связанные с токенами реальные данные, то похищенные токены не дадут им ровным счетом никаких возможностей их использовать.

Повышение интереса к токенизации связано, прежде всего, с тем, что она позволяет обеспечить защиту данных при потенциально низкой стоимости. Добавление шифрования в системы – особенно в уже действующие – значительно увеличивает нагрузку на них. Внесение изменений в приложения для внедрения шифрования может значительно повысить расходы, снизить производительность и увеличить требования к программистам и администраторам систем. Кроме того, в случае использования разнородных систем, возникает необходимость шифровании, расшифровании и повторном шифровании данных в различных местах, что создает

дополнительные уязвимости, которые могут быть использованы злоумышленниками. Причем, чем больше ключей используется в системе, тем больше возможностей для ее атаки. Работа с ключами особенно опасна, учитывая распространенность вредоносного ПО для перехвата данных непосредственно в оперативной памяти, которое позволяет получить доступ к ключам даже не имея административных привилегий на зараженной машине.

Помимо минимизации требований к внесению изменений в используемые приложения, токенизация снижает риски для конфиденциальных данных. При правильной реализации, приложения смогут использовать токены во всей системе и обращаться к защищенным конфиденциальным данным только в случае крайней необходимости. Приложения могут хранить, использовать и совершать транзакции, оперируя только токеном и не подвергая реальные данные риску. Хотя некоторые операции, конечно, требуют обращения к реальным данным, токенизация позволяет минимизировать их использование.

Например, одним из самых распространенных примеров использования токенизации является ее применение для платеже с использованием платежных карт. Более подробно этот сценарий применения будет рассмотрен позже, но использование токена вместо значения номера платежной карты позволяет провести отслеживание транзакции и ее выполнение без риска компрометации реальных данных карты. Доступ к реальному номеру понадобится только в процессинговом центре. Ну а в том случае, если процессинговый центр также использует токенизацию, то возможно проведение транзакции вообще без использования реальных данных.

Впрочем, это не означает, что токенизация всегда будет лучшим выбором, нежели шифрование. Два этих решения тесно связаны между собой и фокус в том, чтобы определить, какое из решений будет лучше в данных конкретных обстоятельствах. Например, каждая система токенизации полагается на шифрование для защиты хранящихся конфиденциальных данных. Но иногда, лучший способ обеспечения безопасности конфиденциальных данных – это отказ от их хранения в большей части мест системы. Токенизация позволяет добиться этого, сохранив при этом возможность работать с конфиденциальными данными, используя токены.

В данном материале мы максимально углубимся в токенизацию и постараемся понять, как работает технология, исследовать разные варианты использования и сценарии развертывания, а также обратим внимание на критерии выбора, которые позволят подобрать правильное решение. Кроме того мы рассмотрим использование сторонних сервисов, позволяющих выполнить требования PCI, и возможность разработки собственных решений для внутренних приложений. Прежде чем мы углубимся в специфику, важно уточнить значение некоторых терминов, которые мы будем использовать в данном материале.

Термины и определения

- **Токен.** Случайные данные, используемые в качестве некоего суррогата для других данных. В токене нет математических зависимостей между случайными и оригинальными данными, поэтому оригинальные данные не могут быть определены по значению токена. Ассоциация между оригинальными и случайными данными ведется только в базе данных и сопоставление оригинальных данных и суррогата может быть проведено только в ней.
- **Токенизация.** Процесс создания токенов.
- **Шифрование.** Обратимое преобразование данных с целью сделать их нечитаемыми для пользователей, у которых отсутствует ключ, который позволяет провести обратное преобразование и вернуть данным исходный вид.

- **Хеш.** Необратимое преобразование по определенному алгоритму массива данных в битовую строку фиксированной длины с неслучайным значением.
- **Зашифрованный токен.** Данные, которые были зашифрованы, используя специальные методы шифрования с сохранением формата и типа данных, а не заменены случайным значением. Обычно такие зашифрованные данные используются в виде токенов, но могут быть расшифрованы с целью получить оригинальные данные. Технически это решение относится к шифрованию, а не токенизации.

ОБОСНОВАНИЕ ДЛЯ БИЗНЕСА

Оправдание инвестиций в токенизацию состоит из двух отдельных шагов – во-первых, определение необходимости в защите данных; во-вторых, выбор токенизации как решения наилучшим образом подходящего для вашей системы.

Охват всех возможных обоснований выходит за рамки данного материала, но ключевыми мотивационными моментами являются:

- Упрощение соответствия требованиям стандартов безопасности
- Снижение затрат на соответствие требованиям безопасности
- Защита от угроз
- Снижение рисков

После того как вы решили защитить данные, возникает вопрос о выборе наилучшего метода их защиты. Токенизация – решение достаточно узкоспециализированное, но призвано решить достаточно распространенную и серьезную проблему: защитить отдельные части высокоценных конфиденциальных данных в приложениях, базах данных, системах хранения и сетях.

Наиболее активно токенизация используется для защиты разнообразных идентификаторов, таких как номера платежных карт, номера социального страхования и прочие персональные данные. Чуть реже можно наблюдать применение токенизации для защиты всех данных клиентов/сотрудников. Разница между этими двумя подходами (мы их рассмотрим подробно при изучении архитектуры решения) в том, что в первом случае сервер токенизации использует только оригинальные данные и токен, которых их заменяет, в то время как во втором случае токен может нести дополнительные данные, такие как имена и адреса.

Доводы в пользу выбора токенизации включают:

- **Упрощение и снижение затрат на соответствие требованиям стандартов безопасности.** По причине того, что токенизация полностью заменяет конфиденциальные данные случайными, системы ее использующие зачастую освобождаются от проверок, которые обязательны для систем работающих с конфиденциальными/персональными данными. Например, в случае замены номеров платежных карт в вашей системе токенами, оценка степени защиты этих данных будет исключена из аудита (например, PCI), что уменьшает его объем, продолжительность и, естественно, стоимость проведения.
- **Минимальная необходимость внесения изменений в приложения.** Токенизация часто внедряется в уже действующую систему, где теоретически возможно и использование шифрования. Учитывая, что токенизация позволяет подменить реальные данные токеном, имеющим тот же формат и тип, что и реальные данные, внесение значительных изменений в приложения использующие данные в большинстве случаев не требуется, что снижает затраты на развертывание решения. При шифровании данных все значительно сложнее. Учитывая, что зашифрованные данные, как, например, номера кредитных карт в корне меняют свою структуру и формат, использовать их в базах данных и приложениях без проведения доработок просто невозможно.
- **Одно место хранения конфиденциальных данных.** Ключевым преимуществом токенизации является возможность хранить конфиденциальные данные лишь в одном месте

– на сервере токенизации, где они надежно хранятся в зашифрованном виде. Это снижает риски по сравнению с шифрованием, при использовании которого конфиденциальные данные доступны в разных местах.

- **Маскировка по умолчанию.** Значение токена случайно, благодаря чему он может эффективно использоваться для маскирования данных. Вам не нужно беспокоиться о добавлении маскировки для приложений, так как реальные значения никогда не раскрываются, за исключением тех случаев, когда токен некорректно используется в вашей среде. Решения по токенизации предлагают множество вариантов форматирования для сохранения значений для отчетности и аналитики, как классическая маскировка, но полностью токенизированные решения обеспечивают большую безопасность и снижают вероятность утечки данных и применения обратного инжиниринга.

Наиболее распространенной причиной, по которой организации выбирают токенизацию, является снижение затрат, которое достигается уменьшением объема и сложности аудита. Также мы видим, что токенизацию выбирают компании, нуждающиеся в повышении уровня безопасности в крупных корпоративных системах, которые, благодаря ей, заметно снижают риск утечки данных и сводят к минимуму применение шифрования.

ОБЗОР ПРОЦЕССА ТОКЕНИЗАЦИИ

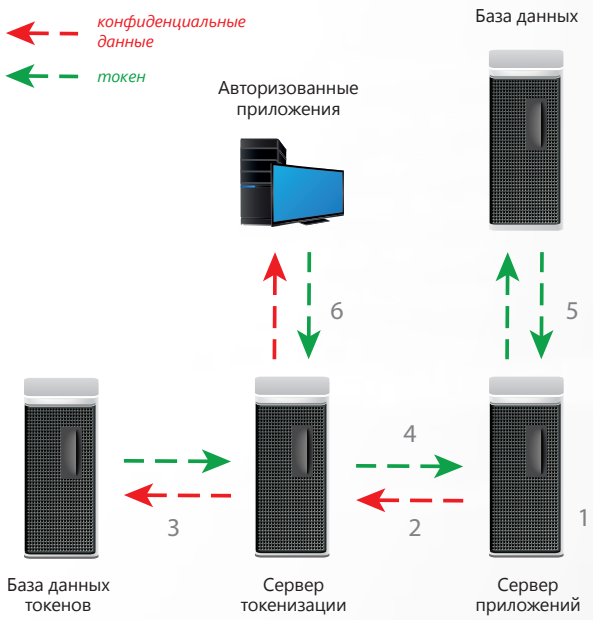
Концептуально, токенизация – это очень простой процесс. Фактически это просто замена одного значения (конфиденциальных данных) другим значением (токеном) – которой не несет в себе никакой важной информации. Но токен небесполезен – он имеет большой значение внутри среды в которой он должен работать(или даже во множестве сред), но вне ее теряет всякий смысл.

Представьте себе жетон для оплаты проезда в метро или подарочный сертификат. Вы используете деньги для их покупки, а затем используете эти вещи для оплаты проезда или покупок в магазине. Это и есть токены, которые являются аналогом денег, но используются в ограниченной системе. Также как и токен, они могут использоваться только в определенной среде, а вне ее совершенно бесполезны.

Токенизация в приложениях и базах данных делает то же самое. Мы берем часть данных, как например номера платежных карт, и преобразовываем их в локальные токены, бесполезные вне системы, которая может принять его и использовать. Теоретически кто-то, может быть, сможет использовать токен в вашей среде, если они получат доступ к приложению, но они не смогут использовать этот же токен где-то еще. Это значительно снижает риски, а также упрощает выполнение некоторых требований стандартов безопасности.

Базовая архитектурная модель

1. *Приложение собирает или генерирует части конфиденциальных данных.*
2. *Данные немедленно передаются на сервер токенизации – они не хранятся локально.*
3. *Сервер токенизации генерирует для данных случайные или полуслучайные токены. Конфиденциальные данные и токены хранятся в защищенной базе данных (обычно в зашифрованном виде).*
4. *Сервер токенизации возвращает токен приложению.*
5. *Приложение сохраняет токен, а не конфиденциальные данные. Токен используется для большинства операций приложения.*



6. *При необходимости получить реальные конфиденциальные данные, авторизованное приложение или пользователь могут запросить их. Конфиденциальные данные не хранятся в локальных базах данных и в большинстве случаев доступ к ним значительно ограничен, что значительно снижает риск утечки.*

Для того чтобы эта система работала как положено, вы должны убедиться в нескольких вещах:

1. В том, что нет никакого способа получить оригинальные данные в обход сервера токенизации. В этом главное отличие от шифрования, где вы можете использовать ключ для получения доступа к данным, независимо от места их получения.
2. Все коммуникации зашифрованы.
3. Приложение никогда не сохраняет конфиденциальные данные и хранит только токен.
4. В идеале приложение вообще не должно знать о самих конфиденциальных данных.Как это сделать мы обсудим позже – есть варианты развертывания с разделением обязанностей. Например, использовать систему транзакций недоступную для пользователей, но с доступом к конфиденциальным данным. Вы также можете иметь одну систему для сбора данных и отправки их серверу токенизации, вторую для работы с клиентами, а третью для операций, где требуется доступ к реальным данным.
5. Сервер токенизации и база данных, где хранятся токены и конфиденциальные данные, должны быть хорошо защищены.

Но, в конце концов, все сводится к одной общей идее токенизации – взять что-то широко применимое и заменить это на что-то с ограниченной применимостью.

ТОКЕНЫ

Создание токенов и соответствующей инфраструктуры напрямую влияют на безопасность конфиденциальных данных, позволяя использовать токены в других приложениях и повышая производительность системы. Для того чтобы иметь возможность сравнивать разные решения, важно понимать основы создания токенов.

Давайте вкратце рассмотрим этот процесс. Каждый раз, когда конфиденциальные данные будут попадать на сервер токенизации, будет выполняться три основных шага. В первую очередь создается токен. Затем токен и соответствующие ему реальные данные сохраняются вместе в базе данных токенов. И уже после этого токен возвращается к инициировавшему процесс приложению. Целью процесса является защита конфиденциальных данных без ущерба для функциональности приложений, поэтому мы не можем просто создать случайные наборы данных. Формат токена должен соответствовать формату исходных данных, благодаря чему он может быть использован так, как будто это оригинальное значение. Например, токен номера социального страхования должен иметь, как минимум, такой же размер (и, желательно, тип данных), что и оригинальный номер. Использующие эти номера приложения и базы данных будут принимать эти значения, так как они соответствуют ограничениям, распространяемым на исходные данные.

Создание токенов

Существует три способа создания токенов, но только один из них мы обычно рекомендуем.

- **Генерация случайных чисел.** Этот метод подразумевает замену исходных данных случайными числовыми и/или буквенными значениями и это как раз тот метод, который мы можем рекомендовать. Токены, полностью состоящие из случайных данных, предлагают высочайший уровень безопасности, так как не подвержены извлечению исходных данных с использованием метода обратного инжиниринга. Некоторые производители используют генераторы для создания токенов, которые используют следующее значение в серии для создания токенов – это не так безопасно, как полностью состоящие из случайных данных, но они генерируются очень быстро и вполне подходят для многих задач (не PCI). Основным преимуществом случайных чисел является возможность легко адаптироваться к каким-либо ограничениям формата данных (подробнее – ниже) и могут быть созданы заранее, чтобы повысить производительность.

- **Шифрование.** Этот метод генерирует токен путем шифрования данных. Защищаемая информация дополняется случайными данными, чтобы предотвратить использование обратного инжиниринга, а затем шифруется с помощью секретного ключа сервера токенизации. Преимуществом данного метода является то, что такой токен достаточно защищен от использования метода обратного инжиниринга, но при этом исходное значение токена может быть восстановлено в случае необходимости. Недостатки, однако, значительны – производительность очень низкая, необходимо использовать алгоритмы шифрования с сохранением формата данных (Format Preserving Encryption (FPE)), а конфиденциальные данные могут быть раскрыты, если будет скомпрометирован ключ. Несмотря на это, многие крупные и географически распределенные компании, в которых требуется доступ к исходным данным, выбирают именно зашифрованные токены, несмотря на то, что, по сути, данный вариант нельзя в полной мере называть токенизацией.

- **Односторонняя хеш-функция.** При сооздании токенов с использованием функции хеширования выполняется преобразование исходных данных с выполнением необратимой математической операции. Этот подход дает достаточно высокую производительность, а элементы могут быть отформатированы так, чтобы соответствовать любому типу данных. Как и в случае с зашифрованными токенами, хеш создается с добавлением случайных данных, чтобы избежать атак "по словарю". Но, в отличие от шифрования, токены, созданные с помощью функции хеширования необратимы – получить из них исходные данные невозможно в принципе. Безопасность данных токенов не так сильна, как у токенов полностью состоящих из случайных данных, но все же, их безопасность, а также производительность и гибкость заметно лучше, чем у зашифрованных токенов.

Помните, что некоторые серверы токенизации, как коммерческие, так и с открытым исходным кодом используют неэффективные или небезопасные методы генерации токенов. В частности, некоторые из них используют обратимую маскировку данных или алгоритмы шифрования без добавления случайных данных, в результате чего токены могут быть легко скомпрометированы.

Свойства токенов

Решения по токенизации должны быть достаточно гибкими, чтобы работать с разными форматами конфиденциальных данных – такими как персональные идентификационные данные, номера социального страхования и номера кредитных карт. В некоторых случаях, возможно введение дополнительных ограничений в форматы токенов. К примеру, при замене номеров социального страхования придется оставлять реальными последние четыре цифры, чтобы сотрудник, которые обслуживает клиентов, мог идентифицировать клиента предъявляющего номер карты.

При токенизации платежных карт, токены должны иметь тот же размер, что и оригинальный номер карты – такая реализация гарантирует, что токены пройдут даже проверку LUHN. Благодаря использованию токенов аналогичных реальным номерам карт, внесение изменений в системы, которые работают с этими номерами, не требуется – токены используются точно так же, как и стандартные номера. Но в отличие от реальной карты или номера социального страхования, токен нельзя использовать как финансовый инструмент – он не имеет никакой ценности, являясь лишь ссылкой на оригинальную транзакцию или реальный счет. Соответствие токена и номера платежной карты является уникальным для каждой платежной системы, так что если даже одна из баз данных токенов будет скомпрометирована, что позволит злоумышленникам совершать транзакции в этой системе (теоретически такое возможно), полученные ими токены не смогут использоваться в других системах. И самое главное, что из токенов в таком случае никак не удастся получить реальные данные кредитных карт.

Каждый тип данных требует индивидуального подхода и разработчики систем токенизации предлагают разные варианты.

Хранение токенов

Токены, вместе с данными, которые они представляют, хранятся в максимально защищенных базах данных с ограниченным доступом. Данные обычно шифруются для того чтобы защитить их в случае компрометации базы данных или ее хищения. Сервер токенизации является единственной точкой контакта для любой системы транзакций, платежной системы или точки сбора данных для уменьшения рисков. Доступ к базе данных строго ограничивается.

Токены используются для представления определенных данных для разных событий, возможно и в нескольких системах, так что серверы токенизации могут выдавать разные токены для одних и тех же данных. Номер платежной карты, к примеру, может получать уникальный токен для каждой новой транзакции. Сервер токенизации должен не только выпускать всех эти токены, но и обеспечивать их хранение с привязкой множество токенов к одному идентификатору пользователя. Многие сценарии использования требуют, чтобы сервер токенизации поддерживал множественные токены для каждого набора оригинальных данных, обеспечивая привязку этого множества к одному идентификатору. Это обеспечивает лучшую конфиденциальность и изоляцию, если приложению не требуется соотносить транзакции с номером карт. Приложения, которые используют конфиденциальные данные (как, например, номера платежных карт) для сопоставления аккаунта или выполнения транзакций, потребуют модификации, чтобы получить возможность использовать только те данные, которые остаются доступными при токенизации (например, условный номер клиента).

Серверы токенизации могут быть развернуты внутри компании или использоваться “как сервис”. Варианты развертывания мы обсудим позже.

Хранение токенов в приложениях

Когда сервер токенизации возвращает токен приложению, приложение должно обеспечить его безопасное хранение, а также эффективное удаление оригинальных конфиденциальных данных. Это очень важно не только для обеспечения безопасности данных, но и для поддержания согласованности транзакций. В идеале приложение вообще не должно хранить конфиденциальные данные у себя, а лишь получать их и сразу отправлять серверу токенизации. Сервер токенизации в свою очередь является единственным местом в системе, в котором по значению токена могут быть найдены сами конфиденциальные данные. Поддерживаемые приложения же привязывают значение к некоему идентификатору, вроде имени пользователя, идентификатора транзакции, клиента и т.д. Это делается для того, чтобы приложения, которые используют токены, были устойчивы к отказам каналов связи, а сервер токенизации мог использоваться для синхронизации и восстановления данных в приложении.

У этого решения есть одна потенциальная слабость: всякий раз, когда конфиденциальные данные собраны или получены по запросу от сервера токенизации, они могут сохраниться в памяти, логах или виртуальной памяти. Это самая уязвимая часть архитектуры. Чуть позже мы обсудим, некоторые из многих способов разделить функции для того, чтобы минимизировать данные риски.

СЕРВЕРЫ ТОКЕНИЗАЦИИ

Серверы токенизации делают гораздо больше, нежели просто создание и хранение токенов. Среди других выполняемых ими функций выделяются: шифрование, проверка пользователей и возращение конфиденциальных данных авторизованным приложениям, когда это необходимо. Многие из действий производимых сервером токенизации незаметны для конечных пользователей систем использующих токенизацию, и они не должны беспокоиться о них. Но внутренний дизайн сервера токенизации имеет значительное влияние на производительность, масштабируемость и безопасность решения. Вы должны оценить эти функции выбирая решение, чтобы избежать проблем в будущем.

Для упрощения понимания в этом разделе мы будем использовать в качестве основного примера номера платежных карт, но вы должны помнить, что вместо них токенизированы могут быть практически любые другие данные. Чтобы лучше понять функции выполняемые сервером токенизации, давайте запомним, что у его есть две основные функции. Первая – это получение конфиденциальных данных от авторизованных приложений или пользователей, ответы приложениям и пользователям с выдачей токенов или уже используемых токенов, выпуск новых токенов с привязкой к нему данных и шифрование последних. Это основная часть всех запросов и действий сервера токенизации. Вторая – сервер токенизации возвращает расшифрованную информацию одобренным приложениям, которые предоставили токен с принятой авторизацией.

Аутентификация

Аутентификация – это основа безопасности сервера токенизации, которая необходима для проверки подлинности подключенных приложений и отдельных пользователей. Для защиты от потенциальных атак, серверы токенизации должны использовать двухстороннюю аутентификацию всех подключаемых приложений до обслуживания запросов. Первым шагом в этом процессе является установка связи с взаимной проверкой подлинности по протоколу SSL/TLS и проверка подлинности сертификатов подключаемого приложения. Строгой аутентификации должно быть вполне достаточно, хотя некоторые реализации требуют дополнительной защиты.

Вторым этапом проверки подлинности является проверка пользователя инициировавшего запрос. Это может быть администратор системы/приложения, использующий специфические административные права, или один из множества сервисов имеющий права на запрос токена или привязанных к токену конфиденциальных данных. Сервер токенизации различает роли пользователей и обслуживает запросы только от одобренных пользователей через авторизованные приложения в авторизованных места. Сервер токенизации также может ограничивать проведение транзакций, позволяя выполнять лишь строго ограниченное подмножество запросов к базе данных.

Шифрование данных

Хотя технически конфиденциальные данные могут и не шифроваться сервером токенизации в базе токенов, на практике в каждой реализации контент шифруется. Это означает, что до записи в базу данных, данные должны быть зашифрованы с помощью общепринятого надежного алгоритма шифрования. После создания токена, сервер токенизации шифрует номер платежной карты с определенным ключом шифрования, используемым только для этого сервера. Затем данные, вместе с соответствующим токеном, хранятся в базе данных.

Каждый действующий сервер основан на реляционной базе данных. Серверы логически группируют токены, номера платежных карт и дополнительную информацию в строках базы данных, обеспечивая логическую связь зависимых друг от друга данных. На данный момент используется два метода шифрования: либо шифрование на уровне полей, либо прозрачное шифрование. При шифровании на уровне полей шифруются отдельные строки (или конкретные поля в ней). Это позволяет серверу токенизации хранить данные от разных приложений (например, платежные карты от определенных продавцов) используя разные ключи шифрования. Некоторые системы токенизации используют прозрачное шифрование, когда шифруется вся база данных с использованием одного ключа. Оба метода шифрования защищают данные в том случае, если кто-то получает доступ непосредственно к жесткому диску или резервной копии, но шифрование на уровне полей допускают большую детализацию и используется чаще.

Наконец некоторые реализации используют ассиметричное шифрование для защиты данных, когда они собираются приложением или PoS-терминалом и отправляются на сервер. Сессия соединения в таком случае, как правило, все равно шифруется (SSL/TLS), но не для обеспечения дополнительной безопасности, а для проверки подлинности. Сервер токенизации в таком случае становится точкой расшифровки данных с помощью закрытого ключа, возвращая данным исходный вид для выпуска токена.

Управление ключами

Если вы используете зашифрованные токены, шифруете хранящие данные или ваш сервер токенизации использует шифрование, да и в любом другом случае использования шифрования вам нужно управлять ключами шифрования. Эта возможность может быть предоставлена разработчиком системы токенизации в виде отдельного приложения или в качестве аппаратного модуля безопасности (HSM – Hardware Security Module), если он поддерживается. В любом случае ключи хранятся отдельно от зашифрованных данных и алгоритмов, что обеспечивает безопасность в том случае, если сервер будет скомпрометирован. А также помогает реализовать разделение обязанностей между разными системами и администраторами. Каждый сервер токенизации имеет один или несколько уникальных ключей для шифрования номеров платежных карт и других конфиденциальных данных. Связь между сервером токенизации и сервером ключей также осуществляется по зашифрованному каналу с взаимной проверкой подлинности.

Системы токенизации также нуждаются в управлении ассиметричными ключами, используемыми для связи с приложениями и устройствами. Как и с любым шифрованием, управление ключами сервера/устройства/функции должно поддерживать защищенное хранение ключей, их перевыпуск, резервное копирование/восстановление и фиксирование действий.

Хранение токенов

Хранение токенов является одним из самых сложных аспектов серверов токенизации. Токены используются как некие ссылки на конфиденциальные данные и предыдущие транзакции, что является серьезной проблемой с точки зрения производительности. Некоторые приложения требуют дополнительных мер по обеспечению безопасности при генерации и хранении токена, так как токен сохраняется в нестандартном формате. Варианты использования, как, например, в финансовых операциях с одноразовыми и многоразовыми токенами, требуют замысловатые стратегии хранения для обеспечения баланса между безопасностью и производительностью.

Давайте рассмотрим некоторые из этих проблем подробнее:

- **Многоразовые токены.** Некоторые системы используют единственный токен в качестве маркера каждого набора конфиденциальных данных. То есть, платежная карта, используемая в конкретном торговом месте, будет всегда представлена одним токеном, используемым для всех транзакций. Это называется "многоразовый токен". Такой тип токена, является наиболее удобным для хранения и использования – всего один токен для сотен сделок. Но многоразовые токены – это не очень хорошее решение с точки зрения безопасности, кроме того, при их использовании нет возможности предоставлять по запросу только ту часть конфиденциальных данных, которая требуется для конкретной операции – выдаются сразу все данные. При этом потребность в получении лишь части данных имеется. Например, в медицинских учреждениях необходима возможность получения анонимных данных пациента, а также данные платежных карт, которые используются в разных торговых точках. Большинство серверов токенизации поддерживают модель с выдачей множества токенов для одного набора данных, благодаря чему можно использовать схему с запросами только необходимой части данных.

- **Поиск данных по токену.** Получение данных платежной карты по токену реализуется достаточно просто, потому что токен используется в качестве ссылки (обычно в зашифрованном виде) на конфиденциальные данные. Но как определить, был ли выдан токен для конкретной платежной карты при использовании многоразовых токенов? Большинство шифровальщиков используют специальные режимы работы для защиты от утечек шаблонов, так что, шифруя один и тот же номер карты, вы всегда получаете разные результаты. Это означает, что вы не можете использовать номер карты для того, чтобы определить – был ли выдан соответствующий ей токен или нет. Есть три возможных пути решения этой задачи. Один из вариантов подразумевает выработку хеша номера карты и хранение его вместе с зашифрованными данными карты и токеном. В данном случае хеш будет являться индексом используемым для поиска. Второй вариант решения заключается в отключении функций шифрования влияющих на генерацию случайных данных, что обеспечит одинаковый результат при шифровании одних и тех же данных. Благодаря этому будет достаточно просто сравнить новое значение с уже имеющимися в базе. Но бесплатного сыра не бывает – столь простой способ делает данные уязвимыми к атакам "по словарю". Ну и наконец, вы можете использовать одноразовые токены для каждой отдельной транзакции. Не все схемы хорошо подходят для решения любых задач – нужно будет убедиться, что производительность и безопасность решения того или иного производителя будут достаточными для конкретной системы.

- **Конфликт токенов.** Серверы токенизации, использующиеся для работы с платежными картами, имеют несколько ограничений: они должны сохранить тот же формат данных, что и оригинальные карты и оставить открытыми последние 4 значения номера. Некоторые продавцы также требуют, чтобы токены проходили проверку LUHN. Это создает проблему ограниченного количества номеров токенов. Количество допустимых LUHN 12-значных чисел создает вероятность генерации двух аналогичных токенов, особенно в том случае, если используются одноразовые токены. Обязательно изучите, какие меры предпринял разработчик системы токенизации для того чтобы снизить вероятность (а лучше, вообще, избежать) конфликта токенов.

АРХИТЕКТУРА И ИНТЕГРАЦИЯ

При выборе решения по токенизации, важно понимать основные архитектурные модели, а также принципы их развертывания, взаимодействия с приложениями и особенности интеграции в поддерживаемые системы.

Архитектура

Есть три пути создания сервера токенизации:

1. Установка готового сервера токенизации с поддержкой базы данных.
2. Встроенные/интегрированные решения со сторонним программным обеспечением.
3. Полностью реализованные в базе данных.

Большинство коммерческих решений по токенизации выпускаются в виде готовых приложений, подключаемых к выделенным базам данных, и как минимум один производитель предлагает решение в виде программно-аппаратного комплекса. Все криптографические операции проводятся в пределах приложений (вне базы данных), а база данных лишь обеспечивает хранение и обеспечение безопасности этого процесса. Серверы токенизации используют стандартные системы управления базами данных, например Oracle или SQL Server, но максимально защищенные. Они могут находиться на одной и той же физической или виртуальной системе, в отдельных системах или могут быть интегрированы в кластер балансировки нагрузки. В этой модели (сервер токенизации в виде готового приложения или устройства с вынесенной базой данных) сервер токенизации управляет всеми запросами в базы данных и обеспечивает коммуникацию с внешними приложениями. Прямые подключения к базе данных запрещены, а все криптографические операции выполняются в пределах сервера токенизации, а не базы данных.

В интегрированных решениях программное решение для токенизации встроено в приложение, совмещенное с поддерживаемой базой данных. Вместо введения шлюза токенизации в процесс операций с платежными картами, существующие прикладные функции изменяются с целью использования токенов. Для пользователей системы нет практически никакой разницы в том, используется ли интегрированный сервис токенизации или отдельное приложение, но бэкенд в этих двух подходах существенно отличается. Во-первых, эта модель развертывания, которая, как правило, включает в себя внесение некоторых изменений в хост-приложение для обеспечения хранения и использования токенов. Кроме того, каждый токен может использоваться только в одном экземпляре приложения. Управление ключами, хранение конфиденциальных данных и токенов – все это также исполняется прямо в приложении. Глубокая интеграция всех функций в данном подходе, делает его очень эффективным для небольших компаний. Но невозможность совместного использования токенов несколькими системами, а также невысокая производительность делают его безынтересным для крупных и географически распределенных компаний.

Наконец, можно управлять токенизацией полностью в базе данных – без использования дополнительного программного обеспечения. Эта возможность зависит от процедуры хранения, поддержки шифрования, и тщательно проработанного контроля доступа и авторизации в базе данных. Данный метод реализации схож с большинством решений по маскировке данных. База

данных автоматически анализирует входящие запросы с целью идентификации и шифрования конфиденциальных данных. Процедура хранения создает связанный с этими данными токен, (обычно для этого используется генератор последовательностей самой базы данных), а после этого возвращает токен, как результат операции, инициатору запроса. Все связанные с токеном данные, хранятся в базе данных. Отдельные функции хранилища позволяют получить доступ к зашифрованным данным. Эта модель была распространена до появления сторонних коммерческих решений по токенизации, но сейчас практически не используются в силу отсутствия поддержки передовых функций безопасности и невозможности использовать внешние криптографические библиотеки и системы управления ключами.

Есть еще несколько архитектурных особенностей:

- Внешнее управление ключами и криптографическими операциями, как правило, поддерживаются во всех приведенных архитектурных моделях. Это позволяет использовать аппаратные модули безопасности (HSM) для повышения уровня безопасности, если это необходимо.
- Развертывание действительно больших систем токенизации может потребовать синхронизации нескольких серверов токенизации в расположенных в разных местах центрах обработки данных. Эта возможность должна поддерживаться серверами токенизации, но она есть не во всех продуктах. Этот момент будет рассмотрен ниже, когда перейдем к моделям использования и развертыванию.
- •аже если вы используете автономный сервер токенизации, вы можете использовать плагины для подключения к нему и управления дополнительными базами данных. Это не превратит базу данных в сервер токенизации, как мы описали в одной из моделей, но обеспечит поддержку безопасных коммуникаций с распределенными системами, которые нуждаются в доступе к любому токену или конфиденциальным данным.

Интеграция

Токенизация должна быть интегрирована с различными базами данных и приложениями. Есть три способа коммуникаций с сервером токенизации:

1. **Вызовы API приложений:** приложения могут выполнять прямые запросы к серверу токенизации. Но в данный момент решения предлагающие доступ приложениям к функциям токенизации крайне редки. Из-за сложности криптографических процессов и необходимости правильного использования сервера токенизации, вендоры предоставляют программные агенты, модули или библиотеки для поддержки интеграции решений по токенизации. Они используются на тех же устройствах, что и вызывающее приложение, что снимает с приложения необходимость использовать API – модули работают с сервером и поддерживают все необходимые методы коммуникации и форматы данных. Это снижает необходимость вносить изменения в программный код приложений и обеспечивает лучшую безопасность, что особенно важно, учитывая, что разработчики приложений далеко не всегда имеют необходимую компетенцию в сфере безопасности. Эти модули форматируют данные в соответствии с требованиями сервиса токенизации и устанавливают безопасное соединение с ним. Как правило, использование таких модулей – это самый безопасный вариант, так как они включают в себя все необходимые криптографические функции, используемые для шифрования новых данных с использованием публичного ключа сервера.

2. **Прокси-агенты:** программные агенты, которые перехватывают обращения к базе данных (например, путем замены компонентов ODRC или JDBC). В этой модели использования, процесс или приложение, отправляющее конфиденциальные данные могут, в принципе, не знать о процессе токенизации. Приложение просто отсылает данные, как делает это обычно, а прокси-агент перехватывает запрос. Агент заменяет конфиденциальные данные токеном, а далее пересылает их серверу токенизации или поддерживаемому серверу приложений. Эта модель минимизирует необходимость во внесении изменений в приложения, так как вам нужно заменить соединение между приложением и базой данных, а новое программное обеспечение будет самостоятельно управлять токенизацией. Но в этой модели есть потенциальные узкие места и проблема отказоустойчивости, так как все решения работают последовательно с используемыми системами обработки транзакций.
3. **Стандартные запросы к базе данных:** сервер токенизации перехватывает и интерпретирует запросы к базе данных. SQL анализируется на предмет конфиденциальных данных, которые заменяются токенами при стандартных запросах. Это потенциально менее безопасный вариант, особенно при получении контента для токенизации, но он очень прост в реализации.

Это все выглядит достаточно сложно, но на самом деле есть всего две основные функции, которые нужны:

1. Отправление новых данных, которые должны быть токенизированы и получение токена.
2. В случае авторизации должна быть возможность получить конфиденциальные данные по токену.

Остальные функции сервер должен выполнять самостоятельно.

РАЗВЕРТЫВАНИЕ И УПРАВЛЕНИЕ

Мы уже рассмотрели внутреннюю структуру серверов токенизации, а также поговорили об архитектуре и интеграции сервиса токенизации. Теперь пора рассмотреть некоторые модели развертывания и то, как они соответствуют различным бизнес-процессам. Ведь, к примеру, задача по защите медицинских данных, использующихся одновременно множеством компаний, сильно отличается от задачи обработки платежных карт тысячами торговых точек.

Центральный сервер токенизации

Самой распространенной моделью развертывания на сегодняшний день является использование одного сервера токенизации, который устанавливается между серверами приложений и серверами транзакций. Сервер токенизации выпускает один или несколько токенов для каждого экземпляра полученной конфиденциальной информации. Для приложений он становится некой библиотекой ссылок, хранящей конфиденциальную информацию, которую может отыскать и выдать приложению при необходимости. Сервер токенизации встраивается в существующую систему транзакций, создавая новый шаг по замене данных, между бизнес-приложениями и бэкендом, занимающимся обработкой данных.

Как упоминалось ранее, эта модель очень хороша с точки зрения безопасности, так как обеспечивает централизованное хранение всех конфиденциальных данных в одном высокозащищенном сервере. Кроме того, он очень прост в развертывании, так как все чувствительные сервисы размещаются в одном месте. Ограничение числа мест, которые хранят и обеспечивают доступ к конфиденциальным данным, повышает безопасность и снижает объем аудита, так как используется меньшее количество систем оперирующих конфиденциальными данными.

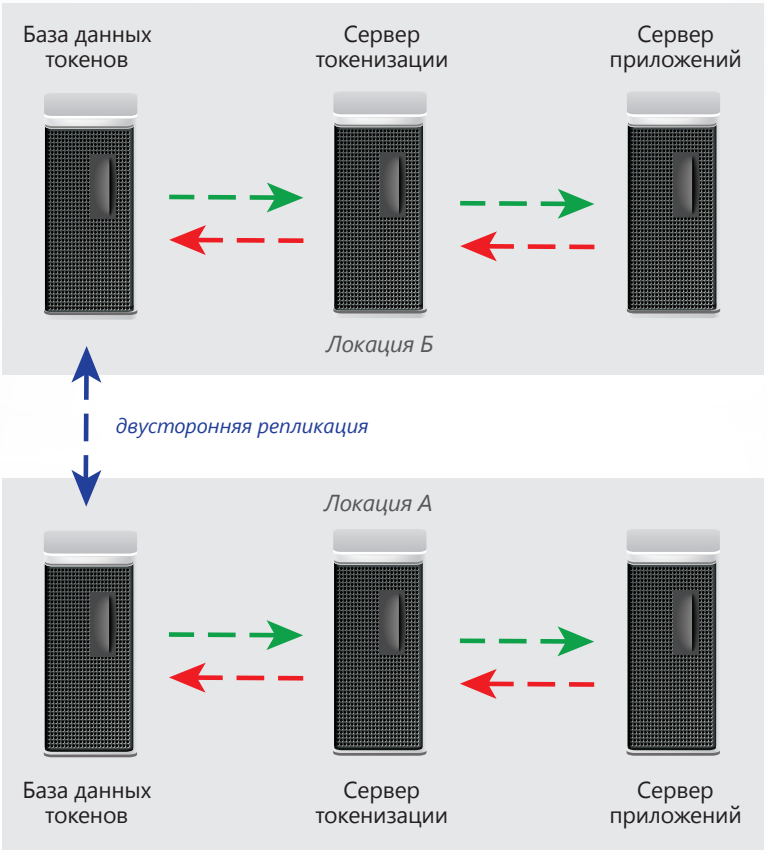
Центральный сервер токенизации отлично работает в среде малого бизнеса за счет консолидации операций, но не очень хорош с точки зрения масштабирования, а потому плохо подходит для больших, географически распределенных компаний. Кроме того, он не гарантирует безостановочной работы системы, а потому не подходит и для интернет-бизнеса. Недостатками данной схемы развертывания также являются:

- **Задержка.** Создание нового токена, поиск существующих клиентов и проверка целостности данных требуют серьезных вычислительных ресурсов. Разработчики многое делают для того, чтобы решить эту проблему, но проблемы с задержками есть до сих пор, что делает это решение непригодным для определенных операций или поддержки транзакций.
- **Отказоустойчивость.** Если центральный сервер токенизации отключен или недоступен из-за сбоя в работе сети, вся обработка конфиденциальных данных (например, заказов) останавливается. Все процессы в бэкенде по конвертации в/из токены, также прекращается.
- **География.** Удаленные офисы, особенно если они расположены в отдаленных районах, страдают от задержек сети, проблем маршрутизации и отключений интернета. Удаленный поиск токенов производится достаточно медленно, а бизнес-приложения и процессинг страдают в случае длительных отключений от сети.

Для решения проблем с производительностью, отказоустойчивостью и сетевой недоступностью, можно использовать несколько других вариантов развертывания, предлагаемых разработчиками систем токенизации.

Распределенные серверы токенизации

В случае с распределенными серверами базы данных токенов становятся доступны в нескольких локациях. Каждый из серверов токенизации в этой системе имеет свою копию токенов и зашифрованных данных и поддерживает весь необходимый функционал.



Эта модель решает проблемы с задержками при поиске токена, а также проблему отказоустойчивости. Благодаря тому, что каждый сервер токенизации является зеркалом других серверов, если какой-то один из них выходит из строя, его функции распределяются между остальными. Кроме того, благодаря использованию множества серверов, имеется возможность балансировки нагрузки на каждый из них. Это решение является достаточно дорогим, но при этом подходит для обработки финансовых транзакций.

Хотя эта модель предлагает лучший вариант с точки зрения производительности и безотказности работы, процесс синхронизации токенов и конфиденциальных данных требует более тщательного изучения. Множественные копии, хранящиеся на разных серверах, требуют внимания к синхронизации для обеспечения правильного использования многоразовых токенов и обнаружения мошенничества/неправомерного использования. Организация процесса синхронизации и управления ключами на нескольких серверах токенизации, равно как и расписание синхронизации, должны быть предусмотрены еще на этапе планирования развертывания системы. Кроме того, важно помнить, что использование нескольких мест хранения конфиденциальных данных, которые должны быть должным образом защищены, поддерживаться и подвергаться аудиту – все это неизбежно повышает стоимость системы.

Реплицируемые серверы токенизации

В реплицируемой модели развертывания, один сервер токенизации обозначен как “активный”, а другой (или другие) как “пассивный”. В этой модели, если активный сервер, вдруг становится недоступен, то пассивный сервер становится активным до тех пор, пока не будет восстановлена работоспособность основного. Эта модель, которую можно назвать “зеркалированием”, улучшает отказоустойчивость системы при минимизации проблем с согласованием данных. Обычно, резервный сервер устанавливается там же, где и основной, но в принципе, он может быть установлен и в другом месте. Эта модель отличается от модели с распределенными серверами токенизации только в том, что здесь одновременно активен только один сервер, а остальные просто хранят ту же информацию, но в работу вступают только по необходимости.

Модель с реплицируемыми серверами токенизации предназначена для обеспечения согласованности операций, что критически важно при развертывании токенизации в банковских или медицинских системах. Главное здесь, что при отключении основного сервера его тут же подменяет резервный, и операции проводятся в нормальном режиме. Синхронизация в этой модели также очень проста, так как пассивный сервер должен быть простой копией активного, а двунаправленная или множественная синхронизация не требуется. Для обеспечения этой возможности обычно используются встроенные в реляционные СУБД возможности зеркалирования.

Система управления

Как и в любом серьезном приложении, сервер токенизации имеет стандартные функции управления, но они имеют дополнительные требования по безопасности:

- Управление пользователями, включая аутентификацию, контроль доступа и авторизацию – для пользователей, приложений и подключений к базе данных. Дополнительно большинство решений по токенизации требуют распределения обязанностей, чтобы ограничить административный доступ к защищенным данным.
- Резервное копирование и восстановление данных, конфигурации системы и ключей шифрования, если шифрование производится на сервере токенизации. Защищаемые данные при резервном копировании должны всегда храниться в зашифрованном виде.
- Мониторинг и протоколирование – журналы аудита, особенно системных изменений, административного доступа и операций с ключом шифрования, должны быть частью любого решения, которое вы выберете. Эти журналы необходимы для того, чтобы система удовлетворяла нормативным требованиям.

Руководство «Visa Tokenization Best Practices» рекомендует мониторинг системы для своевременного обнаружения неисправностей или аномальных и подозрительных действий с запросами по сопоставлению токена к данным. Это не является частью нормативных требований, но является очень хорошей практикой в безопасности. Это относится как ко всем запросам генерации токенов, так и запросам конфиденциальных данных. Мониторинг должен фиксировать и анализировать активность в отношении политик безопасности и операций. Некоторые системы блокируют неаутентифицированные запросы или даже запросы не имеющие соответствующих сетевых атрибутов. Кроме того, система должна подавлять запросы, если в них обнаруживаются признаки атак “по словарю” или атак “по радужной таблице”.

Комментарии к рекомендациям Visa

В 2010 году Visa выпустила рекомендации по токенизации в платежных системах. Учитывая близость данной рекомендации к теме материала, мы думаем нелишним будет прокомментировать некоторые из имеющихся в нем вопросов.

Если вы обременены необходимостью соответствовать требованиям PCI-DSS, то наверняка не будете писать свое собственное решение по токенизации. И даже если вы выберете коммерческое решение, и оно будет вполне соответствовать требованиям PCI-DSS и рекомендациям Visa, далеко не факт, что оно будет действительно обеспечивать должный уровень безопасности. Если вы заинтересованы лишь в получении системы в принципе, то любая их них будет неплоха. Но если вы не хотите проблем с реализацией безопасности сервера токенизации, то лучше выбрать то решение, которое предлагает лучший уровень безопасности.

В отношении рекомендаций Visa есть две проблемы: Visa не пропагандирует использование генераторов случайных чисел для токенов и не дает никаких рекомендаций относительно безопасности хранения данных.

Необходимость в генераторе случайных чисел

Принципом токенизации является замена токеном реальных (конфиденциальных) данных, которые не могут быть восстановлены, используя методы обратного инжиниринга. Но при выборе стратегии создания токенов, вы должны решить, хотите ли вы иметь возможность получить данные вне базы данных сервера токенизации. Если вы хотите, чтобы осталась возможность преобразовать токен в данные вне системы, то используйте шифрование. Но если это не требуется, то есть лучшие способы защитить номера платежных карт.

Первое предложение Visa: просто использовать случайные числа. Если для генерации случайного числа не используются математические функции, то и восстановить исходные данные (в данном случае номер платежной карты) невозможно. В таком случае единственным способом получить номер карты – это поиск в базе данных сервера токенизации. Случайные токены легко генерировать, а также достаточно просто управлять их размером и типом данных. Это должно быть основным решением, так как у большинства компаний нет никакой необходимости в извлечении из токена реального номера платежной карты.

Что касается шифрования, то почему вместо “стойких криптографических алгоритмов” не использовать в этом шаге одноразовый блокнот (one-time pad). Это идеальное решение для данного вида подмены значений и более безопасное, нежели другие. Мы считаем, что Visa не предлагает их просто потому, что использует огромное число больших платежных процессоров для распределенных операций и хочет чтобы значения номеров платежных карт извлекались в разных местах. Это означает, что им необходимо шифровать токены и распространять ключи.

Зашифрованное значение – это не токен

Если значение зашифровано – это шифрование, а не токенизация. Шифрование искажает, а токенизация удаляет оригинальные данные.

Концептуально основными преимуществами токенизации являются:

1. Из токена невозможно получить оригинальное значение.
2. Токен сохраняет ту же структуру и тип данных, что и оригинальное значение.

При шифровании, конечно, возможно сохранить структуру и тип данных, но в данном случае все равно остается возможность восстановить оригинальное значение, если у вас есть ключ и алгоритм.

При оценке решений будьте осторожны и убедитесь, что потенциальный поставщик использует правильную терминологию. Токенизация и шифрование являются разными процессами и технологиями, хоть и тесно связаны между собой. Если токен представляет собой зашифрованное значение – это шифрование, а не токенизация.

Хеширование

Если токены базируются на алгоритме хеширования, то убедитесь, что токены достаточно защищены от атак "по словарю" и атак "по радужной таблице". Также убедитесь, что используемый алгоритм хеширования утвержден NIST или другими органами стандартизации и удостоверьтесь, чтобы версия вендора была проверена сторонним экспертом. Даже очень маленькие ошибки в реализации функции хеширования может сделать ее использование небезопасным.

Различимость токенов

Как вы узнаете, что токен это токен? Когда токен использует форму и тип данных оригинального значения, как вы сможете отличить их? Некоторые токены, сгенерированные в соответствии с требованиями PCI, сохраняют четыре последних значения оригинального номера платежной карты и проходят проверку LUHN. В данном случае отличить токены от реальных номеров почти невозможно. Если вам необходима "различимость токенов", узнайте о ней у вашего вендора. Visa рекомендует эту возможность как опциональную для всех серверов токенизации, но в зависимости в используемых вами типов токенов, эта возможность может быть недоступна.

Шифрование хранилища данных

Солидные поставщики для работы с платежными картами и серверами токенизации будут использовать реляционную базу данных. Для защиты конфиденциальных данных в реляционном хранилище вам необходимо шифровать их – вопрос только в том, как? Есть много способов развертывания шифрования для защиты данных при хранении, но мы будем рассматривать только три из них – те, что соответствуют стандарту PCI-DSS: шифрование на уровне приложения, шифрование на уровне полей и прозрачное шифрование базы данных. Как следует из названия, шифрование на уровне приложения, шифрует данные еще до внесения их в базу данных. Шифрование на уровне полей шифрует значение полей в базе данных. Ну а прозрачное шифрование базы данных подразумевает шифрование непосредственно в базе данных. Все варианты должны использовать сервис управления ключами для обеспечения надлежащего распределения обязанностей. Если вы используете только одну базу данных, то она должна быть серьезно защищена и в данном случае идеально подойдет прозрачное шифрование базы данных. К тому же оно невероятно просто в развертывании. Если же у вас более одного приложения, которое подключается к базе данных сервера токенизации или вы используете реплицируемые/избыточные базы данных, то шифрование на уровне приложений обеспечит лучшую безопасность.

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Мы рассмотрели большую часть технологий необходимых для создания и развертывания сервера токенизации. Вооружившись этими знаниями, мы можем рассмотреть самую важную часть – примеры использования токенизации. Какая модель лучше подходит для конкретной среды? Какие факторы будут самыми весомыми в решении? Три приведенных ниже примера использования покрывают большинство возможных ситуаций. Учитывая, что соблюдение требований PCI являются основным мотивом для использования токенизации, первые два примера будут демонстрировать фокусирование на разных возможностях развертывания систем соответствующих PCI.

Средний бизнес. Розничная торговля

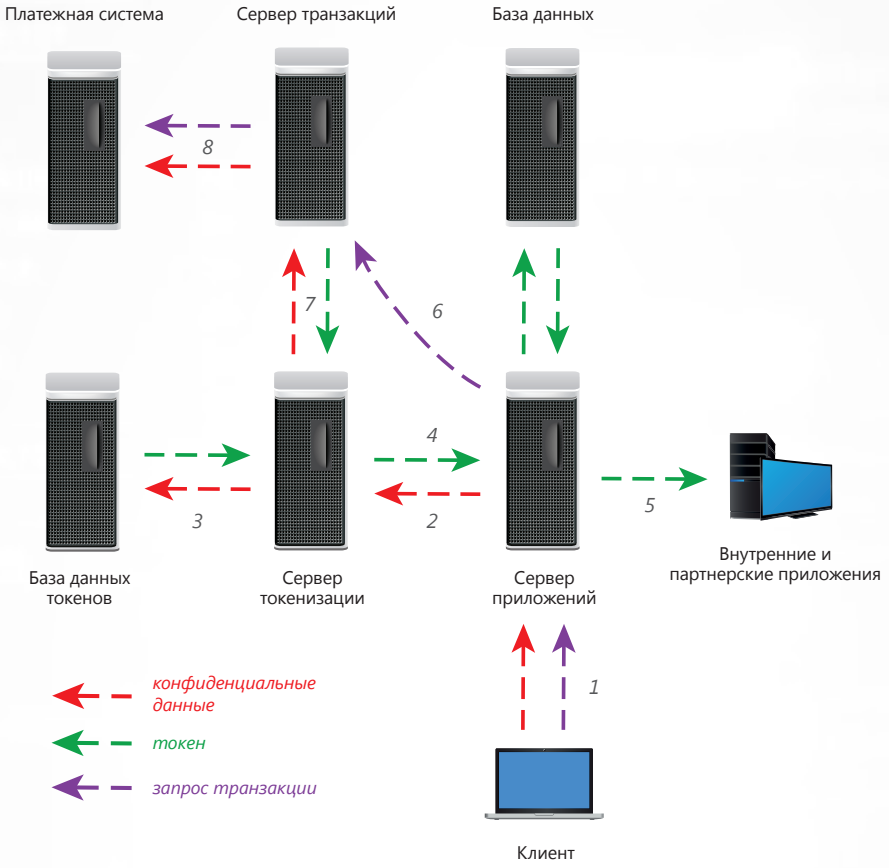
Наш первый пример – это ритейлер среднего размера, которому необходимо соответствовать требованиям PCI. Компания принимает платежные карты, но работает исключительно через интернет, а потому не нуждается в поддержке платежных терминалов. Их задача – соответствие требованиям PCI, и главные вопрос в том, как добиться этого за разумные деньги. Как и в большинстве других подобных решений, большая часть работающих в компании систем разработаны до того как появились правила хранения данных платежных карт и используют номер карты для идентификации клиентов и заказов. Это значит, что все действия (заказы, оплата, выставление счетов, обслуживание пользователей и BI системы) все завязано на этот номер, в дополнение к авторизации на web-сайте и системе оплаты.

Информация о кредитных картах разбросана по многим системам, а потому контроля доступа и строгой аутентификации недостаточно для того чтобы решить проблему. В системах есть слишком много точек доступа, что дает злоумышленникам много шансов скомпрометировать одну или несколько систем. Кроме того, некоторые системы компании доступны партнерам для продвижения продаж и выполнения заказов. В результате, решения по безопасности должны охватывать все системы и влиять на работу почти каждого сотрудника. При этом большая часть внутренних систем транзакций не нуждаются в реальных номерах платежных карт – они предназначены лишь для хранения и используют номера в качестве референсных значений. Эти системы используют прозрачное шифрование, что означает, что они автоматически возвращают расшифрованные данные, защищая их по факту только в пределах накопителя. Контроль доступа и шифрование данных в данном случае не являются достаточными мерами обеспечения безопасности в данном сценарии и не позволяют достичь соответствия требованиям PCI.

Основной целью этого проекта является обеспечение соответствия PCI, но есть достаточно серьезные вторичные цели: минимизация затрат, простая интеграция и необходимость непрерывно контролировать соблюдение требований. Учитывая имеющиеся требование, которое запрещает хранить данные карт в открытом виде, у компании имеется выбор между двумя решениями – шифрованием или токенизацией. Они могут либо реализовать шифрование на каждой из имеющихся платформ приложений, либо использовать центральный сервер токенизации для замены номеров платежных карт токенами.

Для нашего теоретического клиента мы рекомендуем внутреннюю (домашнюю) токенизацию. Домашний сервер токенизации будет обеспечивать использование токенов вместо реальных номеров платежных карт. Это позволит полностью удалить реальные данные карт и внести минимальные изменения в платформы, которые используют платежные карты – принимают их от клиентов, организуют расчеты и т.д. – а остальные будут работать с токенами сохраняющими формат реальных номеров карт. Мы рекомендуем использовать сервер над одним из приложений, если продавцу

понадобится использовать токены в нескольких приложениях. Это делает достаточно простым сегментирование, то есть отделение пользователей и сервисов, которые используют токены от тех, которые действительно нуждаются в доступе к реальным номерам платежных карт.



1. *Клиент делает запрос на покупку. Если это новый клиент, то данные кредитной карты отправляются через SSL-соединение. В будущих покупках будет отправляться только запрос на проведение транзакции.*
2. *Сервер приложений обрабатывает запрос. Если кредитная карта еще не использовалась в системе, то данные отправляются на сервер токенизации, используя его API, и запрашивает новый токен.*
3. *Сервер токенизации создает токен и сохраняет его, вместе с зашифрованным номером кредитной карты.*
4. *Сервер токенизации возвращает токен, который сохраняется в базе данных приложения, вместе с другими данными клиента.*
5. *Токен используется во всей среде продавца вместо номера платежной карты.*
6. *По завершению платежной операции, сервер приложений отправляет запрос на сервер транзакций.*
7. *Сервер транзакций посылает запрос серверу токенизации и получает в ответ реальный номер платежной карты.*
8. *Информация о транзакции, включая реальный номер карты, отсылается в платежный процессор для завершения транзакции.*

В то время как шифрование может защитить данные платежных карт без токенизации и потребует внесения минимальных изменений в пользовательский интерфейс приложений и в базы данных, это потребует серьезных модификаций каждой системы, которая занимается обработкой платежных карт. Кроме того, решения по шифрованию потребуют использования системы управлению ключами для защиты криптографических ключей. Но решающим фактором против шифрования будет стоимость модернизации систем при внедрении шифрования на уровне приложений – особенно если они были разработаны третьей стороной. В результате на доработку приложений, внедрение системы управления ключами, и другие дополнительные действия потребуется немало средств и времени. Фактически эти дополнительные действия обойдутся значительно дороже, нежели сама стоимость решений по шифрованию или токенизации.

Целью данного сценария, как вы помните, является соответствие требованиям и защита данных. Но в реальности, решение о покупке в большинстве случаев зависит от стоимости – суммарных расходов на приобретение, обслуживание и управление. При этом важно помнить, что для торговых точек, которые работают с платежными картами, в соответствии с ростом бизнеса растет и стоимость соответствия требованиям. А потому, лучшим выборов будет тот, который обойдется дешевле в долгосрочной перспективе. С точки зрения относительной безопасности, шифрование и токенизация примерно эквивалентны. Также нет особой разницы между этими двумя решениями по расходам на приобретение и эксплуатацию. Но есть существенная разница в расходах на внедрение и проведение аудита.

Токенизация как сервис

Риски безопасности и требования PCI, стали толчком к тому, чтобы некоторые процессинговые центры стали предлагать токенизацию как сервис. Розничные компании могут подписаться на него и тем самым избавить себя от необходимости хранить у себя данные кредитных карт – вместо них для проведения транзакций будут использоваться токены. Это интересный подход, который позволяет почти полностью удалить номера платежных карт из своих процессов и систем.

Компромисс при выборе такого решения состоит в том, что вы оказываетесь полностью привязанными к выбранному сервису с требованием использовать только их оборудование и программное обеспечение (обычно и то и другое поставляется в комплекте). В результате вы снижаете риски, избавляясь от взаимодействия с реальными номерами кредитных карт в вашей организации по приемлемой цене, но возможно с потенциально большой стоимостью смены поставщика сервиса.

Многие крупные поставщики предлагают законченные решения, использующие токенизацию, шифрование или комбинацию этих технологий. В нашем примере мы рассмотрим токенизацию на примере стандартного терминала оплаты (PoS-терминал), такого же как используются в огромном количестве торговых точек.

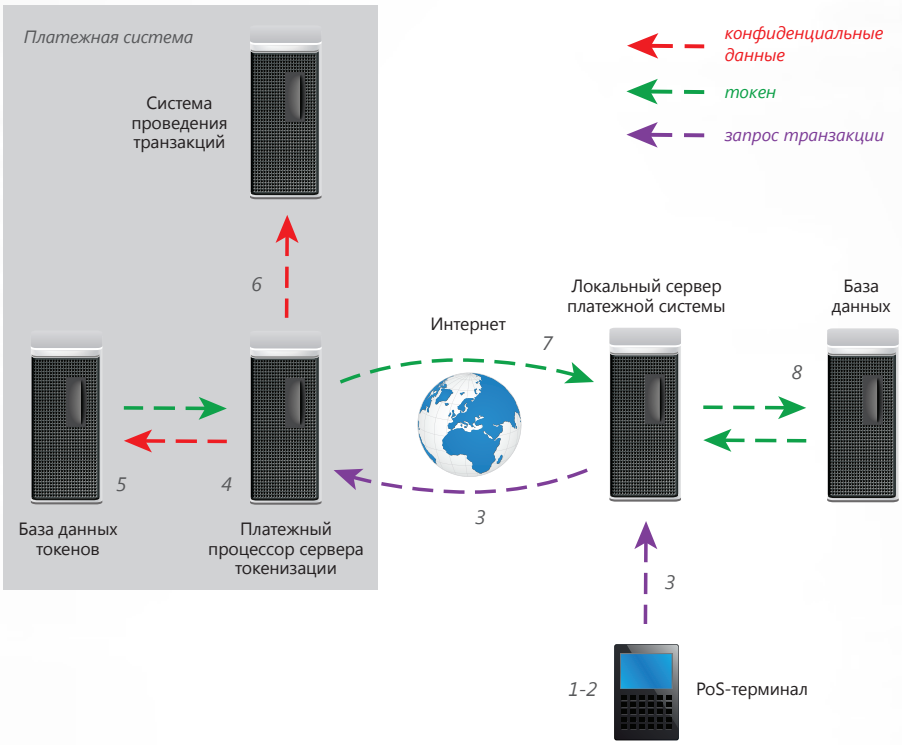
Сначала немного об архитектуре торговых точек, которая включает в себя три компонента:

1. PoS-терминалы в точках продаж для считывания платежных карт.
2. Приложение для обработки и управления транзакциями.
3. База данных для хранения информации о транзакции.

Как обычно клиент использует свою платежную карту в PoS-терминале, который затем связывается с локальным сервером, который в свою очередь соединяет его либо с центральным сервером процессинга в системе продавца для авторизации платежа или пакетного клиринга, или напрямую к

платежному процессору. Информация о транзакции, включая номер платежной карты, сохраняется на локальном и/или центральном сервере. Соответствующие PCI конфигурации шифруют данные платежной карты в локальной и центральной базе данных, а также при их передаче.

Когда в процессоре оплаты применяется токенизация, процесс меняется следующим образом:



1. *Розничный покупатель использует платежную карту в PoS-терминале.*
2. *PoS-терминал шифрует номер платежной карты публичным ключом сервера токенизации платежного процессора.*
3. *Информация о транзакции (включая реальный номер карты, дополнительные данные, данные транзакции и идентификатор продавца) в зашифрованном виде отправляются платежному процессору.*
4. *Сервер токенизации платежного процессора расшифровывает номер платежной карты и генерирует для него токен. Если же данный номер платежной карты уже есть в базе данных токенов, он использует уже имеющийся токен (при использовании многоразовых токенов) или генерирует новый токен специально для данной транзакции (если используются одноразовые токены). Многоразовые токены могут использоваться различными продавцами.*
5. *Токен, данные платежной карты и, возможно, идентификатор пользователя сохраняются в базе данных сервера токенизации.*
6. *Номер платежной карты используется платежным процессором для авторизации и подачи в банк-эмитент.*
7. *Токен возвращается в локальную или центральную платежные системы продавца как одобрение/отказ в проведении сделки, который передает его на PoS-терминал.*
8. *Продавец сохраняет токен вместе с информацией о транзакции в своих системах/базах данных. Все запросы продавца по поводу урегулированию каких-либо вопросов по транзакции будут ссылаться на токен.*

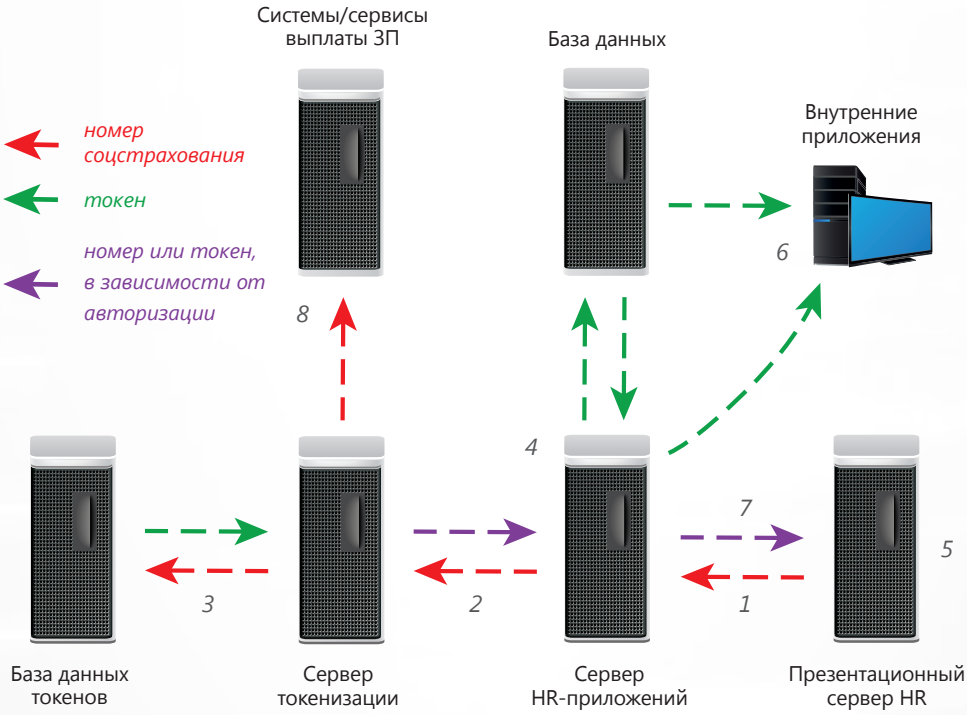
Ключевым моментом в этой схеме является то, что реальный номер карты шифруется в точке сбора и больше никогда не используется внутри системы продавца. Продавец использует токен во всех случаях, где ранее использовался реальный номер карты, например, для обработки возврата средств покупателю.

Это достаточно новое решение и различные поставщики используют различные варианты, но в основном архитектура остается примерно такой же.

Защита персональных данных сотрудников

Не каждый случай применения токенизации подразумевает выполнение PCI-DSS. Есть не менее эффективные варианты применения для защиты персональных данных, которые демонстрируют иной подход к использованию сервисов токенизации. Сейчас мы рассмотрим вариант использования токенов для замены номеров социального страхования в приложениях HR. Это решение должно защитить данные номера при нормальном использовании сотрудниками и предоставляющей услуги третьей стороной, в то же время, предоставляя авторизованный доступ сотрудникам отдела кадров, включая доступ к информации о заработной плате и премиях.

В нашем примере сотрудники используют специальное приложение для отдела кадров для просмотра информации о премиях, а также внесения изменений в свою учетную запись. Во время первого использования сотрудник вносит в систему все свои данные, возможно включая номер социального страхования. Чтобы понять, как токены будут работать в этом сценарии, давайте рассмотрим схему процесса:



1. *Процесс создания учетной записи сотрудника начинается с ввода учетных данных и добавления персональной информации, включая номер социального страхования. Этот шаг, как правило, выполняется сотрудником отдела кадров при опросе сотрудника.*

2. *Через безопасное соединение презентационный сервер передает данные HR-приложению. HR-приложение анализирует запрос и, если в нем имеется номер социального страхования, выделяет его и отправляет на сервер токенизации.*

3. *Сервер токенизации проверяет соединение с сервером HR-приложения и запрос, после чего создает токен для номера социального страхования и сохраняет пару токен/номер в базе данных. Затем он отправляет токен на сервер HR-приложения.*

4. *Сервер HR-приложения сохраняет данные о сотруднике вместе с токеном и передает токен презентационному серверу. Временно хранящаяся копия оригинального номера перезаписывается так, чтобы ее не осталось в памяти.*

5. *Презентационный сервер отображает пользователю страницу об успешном создании учетной записи, в том числе токенизированное значение. Оригинальный номер, как и в HR-приложении, полностью стирается.*

6. *Токен используется во всех внутренних приложениях, где ранее использовался реальный номер социального страхования.*

7. *Когда сотрудникам отдела кадров необходимо получить реальный номер социального страхования сотрудника (как правило, для выплаты заработной платы и премии), они могут получить его после авторизации в приложении (реальное значение доступно по запросу через API сервера токенизации). Хотя реальный номер социального страхования сотрудника и будет виден пользователю, он не будет храниться ни в приложении, ни на презентационном сервере. Обычно не сложно уберечь конфиденциальные данные от внесения в журналы событий, но теоретически их попадание туда возможно. Это, безусловно, реальный риск, но заметно меньший, нежели работа с реальными номерами социального страхования.*

8. *Реальный номер социального страхования используется по необходимости при соединении с системами или сервисами, используемыми для выплаты заработной платы и премий. Его использование, конечно, лучше минимизировать, но в реальности многие (большинство?) приложения и сервисы по-прежнему требуют предоставления реального номера – особенно при выплате заработной платы и уплате налогов.*

Приложения, уже содержащие номера социального страхования, проходят процесс токенизации автоматически, без участия пользователей. Многие старые приложения используют номер социального страхования как основной ключ для доступа к записи пользователя. В таком случае токенизация будет достаточно сложной и может привести к простоям и необходимости внесения структурных изменений.

Отметим, что токены заменяющие номера социального страхования могут сохранять не только формат, но и четыре последних значения оригинального номера. Отображение последних четырех цифр позволяет отделу кадров и клиентским сервисам использовать токены для идентификации сотрудников, но сохранять конфиденциальность за счет маскировки первых пяти значений. Нет никаких причин для сотрудника указывать свой номер социального страхования после первичного создания учетной записи, но он может обратиться за помощью к отделу кадров в том случае, если вдруг возникнет необходимость в проверке оригинального номера, внесенного в базу данных. Другие сотрудники компании, не относящиеся к отделу кадров, не должны видеть номера социального страхования вообще. Сервер HR-приложений, также как и презентационный сервер могут демонстрировать только токенизированные значения во внутренних web-приложениях для сотрудников и никогда не показывать оригинальные номера.

Что действительно отличает данный пример использования, так это необходимость в регулярном доступе HR-приложений к оригинальным номерам социального страхования. В отличие от PCI-токенизации, где запросы на получение реального номера платежной карты приходят достаточно редко, бухгалтерский учет и другие системы обращаются за оригинальными значениями достаточно регулярно. В нашем примере, уполномоченные сотрудники отдела кадров получают оригинальные номера через HR-приложение. Это делается автоматически через HR-приложение уполномоченными и авторизованными сотрудниками отдела кадров, а представление номера ограничено только специфическим HR-интерфейсом. После того как сервер HR-приложений извлек данные сотрудника из базы данных, приложение выдает запрос серверу токенизации на получение реального номера социального страхования, и после его получения отправляет на презентационный сервер.

Точно также возможна и автоматизация при начислении заработной платы, при которой HR-приложения инструктируют сервер токенизации о необходимости отправки реального значения номера социального страхования определенного сотрудника в соответствующую подсистему. Номера социального страхования в таком случае извлекаются сервером токенизации и отправляются поддерживающему приложению по защищенному каналу. В данном случае номер социального страхования становится доступен для сторонних авторизованных провайдеров.

КРИТЕРИИ ВЫБОРА

Если вы рассматриваете решения по токенизации, то мы можем предположить, что вы хотите сократить риски раскрытия конфиденциальных данных, а также снизить расходы необходимые для соответствия требованиям. Мы не хотим занижать сложность токенизации, но процесс выбора на самом деле очень прост. В конечном счете, есть лишь несколько вопросов, на которые нужно ответить. Соответствует ли это требованиям моего бизнеса? Будет лучше использовать внутренне решение или воспользоваться сторонники услугами? Какие приложения нуждаются в использовании токенизации и насколько сложно будет ее внедрить?

Для некоторых организаций процесс выбора предельно прост. Если у вас небольшая компания и есть необходимость в соответствии требований PCI, то стоит отдать токенизацию для вашей платежной системы на аутсорсинг. Аутсорсинговый сервис токенизации должен легко интегрироваться со внешними сервисами, особенно если они предоставляются тем же поставщиком – по крайней мере должно быть что-то совместимое и одобренное для использования в их инфраструктуре. Почему аутсорсинг? Потому что большинство небольших компаний не обладают необходимыми ресурсами и опытом для того чтобы создать достаточно защищенный сервер токенизации и управлять им. Даже если опыта достаточно, то выбор готового решения от вендора обойдется дешевле и снимет с вас большую часть ответственности.

Использование токенизации как сервиса для платежной системы на самом деле отличный вариант для любой компании, особенно если и платежная система отдана на аутсорсинг, но такой подход, как правило, редко встречается в крупных компаниях.

Итак, нужно делать выбор. Вот наши рекомендации по этому процессу:

1. **Определите бизнес-требования.** Самым главным фактором является проблема бизнеса, которую нужно решить. Целесообразность решения основывается на его способности решать проблемы безопасности или способствовать соблюдению требований. Из основных требований, конечно, PCI, но, к счастью, большинство серверов токенизации разработано с их учетом. Также у вендоров есть решения подходящие для работы с другими данными, такими как медицинская информация, номера социального страхования и т.д.

2. **Изучите схему используемых вами систем.** Определите системы, которые хранят конфиденциальные данные, включая платформы, базы данных и приложения. И оцените, в каких из них возможна замена конфиденциальных данных на токены.

3. **Определите требования систем и приложений.** Теперь вы знаете, какие платформы вам необходимо поддерживать и пора определить конкретные требования интеграции. Это в основном касается баз данных – какие языки используют приложения для записи в них, как вы аутентифицируете пользователей и насколько распределенными являются ваши приложения и дата центры.

4. **Определите требования к токенам.** Изучите, какие данные используются вашими приложениями, и определите, какие токены вас нужны или являются предпочтительными – одноразовые или многоразовые. Могут ли токены быть отформатированы так, чтобы они были приняты бизнес-средой? Если требуется доступ к оригинальным данным в распределенной среде, то может быть подойдут зашифрованные “токены” с сохранением исходного формата?

5. **Оцените варианты.** В этот момент вы уже должны понимать требования бизнеса, особенности вашей системы, требования к интеграции приложений и требования к токенам. Этого достаточно для того, чтобы приступить к оценке решений предлагаемых на рынке, а также сравнить сервисы с решениями, разворачивающимися на вашей стороне.

Это все довольно просто, но очень важно заранее определится с вашими бизнес-требованиями, чтобы не дать поставщику направить вас по ложному пути. Кроме того вы должны хорошо понимать требования к интеграции еще до того, как первый продавец войдет в вашу дверь.

Есть еще ряд второстепенных факторов при выборе сервера токенизации.

- **Аутентификация.** Как сервер токенизации будет интегрироваться с вашими системами идентификации и контроля доступа? Это имеет значение для внешних сервисов токенизации, но особенно важно для внутренних баз данных токенов, которые хранят конфиденциальные данные. Вы должны тщательно контролировать, какие пользователи могут делать запросы токенов и какие могут запрашивать реальные данные платежных карт и другую информацию. Убедитесь, что ваши системы контроля буду т интегрированы с выбранным решением.

- **Безопасность сервера токенизации.** Какие возможности и функции может предложить сервер токенизации для шифрования базы данных, мониторинга транзакций, защиты коммуникаций и проверки запросов. С другой стороны, какую часть функций будет обеспечивать разработчик, а какую вы?

- **Масштабируемость.** Как можно увеличить производительность сервера токенизации при необходимости?

- **Управление ключами.** Будут ли шифрование и управление ключами интегрированы в сервер токенизации или потребуется внешний сервис? Для токенов, полученных с помощью шифрования, проверьте, как ключи используются и управляются.

- **Производительность.** Скорость обработки платежа имеет прямое влияние на клиента и его отношения к продавцу. Поэтому важным является вопрос – имеет ли сервер токенизации достаточную производительность для реагирования на новые запросы маркеров и готов ли он к пиковым нагрузкам?

- **Отказоустойчивость.** Приложения обработки платежей нетерпимы к отключениям сервера токенизации. Должна быть возможность обеспечения безотказности, а сервис должен обеспечивать определенный уровень обслуживания. Если ваша системы должна быть доступна в любое время и не терпит простоев, убедитесь, что выбранное решение удовлетворяет этому требованию.

О НАС

Компания Ak Kamal Security, основанная в 2006 году, специализируется на разработке, поставках, внедрении и сопровождении средств криптографической защиты информации. Продукция компании, а также наших партнеров, среди которых крупные игроки на рынке информационной безопасности, покрывает практически весь спектр задач по обеспечению безопасности, стоящих перед бизнесом и другими структурами, для которых критично сохранение конфиденциальности данных и обеспечение безопасности бизнес-процессов.

Одним из важнейших преимуществ компании является географическая близость к нашим клиентам, знание специфики казахстанского рынка и особенностей законодательства в сфере информационной безопасности. Кроме того, это позволяет нам быстро реагировать на все запросы клиентов касающихся наших разработок – будь то обеспечение технической поддержки, или пожелания по улучшению и расширению функционала в соответствии со спецификой их пользования в каждом конкретном случае.

Обратившись в Ak Kamal Security, Вы найдете в нашем лице надежного и компетентного партнера и консультанта по вопросам информационной безопасности. Накопленный опыт и прочные партнерские отношения с ведущими производителями средств защиты информации, профессиональный подход к решению любых вопросов в этой сфере, а также внимательность к запросам клиента – гарантия высокого качества нашей работы.

Ak Kamal Security

КОНТАКТЫ

ТОО «Ak Kamal Security», г. Алматы, ул. Каблукова, 257
Тел: +7 (727) 381-05-26, +7 (727) 222-00-92
Факс: +7 (727) 381-00-39
Mail: info@akkamal.kz
Web: www.akkamal.kz, www.e-security.kz, www.mysign.kz