WEB-ПРИЛОЖЕНИЯ

Построение программы безопасности

Ak Kamal Security

СОДЕРЖАНИЕ

Перевод и подготовка материала проведены компанией Ak Kamal Security. Автором оригинального материала под названием “Building a Web Application Security Program” является компания Securosis (*www.securosis.com*).

ВВЕДЕНИЕ

Современные web-приложения существуют в среде, которая заметно отличается от той, которая была на заре существования сети Интернет. Они стали важными инструментами работы компаний, но, при этом, их функционирование зачастую не очень соответствует тому на что рассчитаны браузеры, движки которых, зачастую, создавались много лет назад. Причем рост и расширение функциональности web-приложений происходили столь быстро, что не всегда удавалось добиться всего, чего хотелось. Это особенно очевидно в том случае, если речь идет об обеспечении безопасности web-приложений. И хотя большинство компании использует определенные механизмы защиты своих web-приложений, только некоторые из них имеют действительно серьезные, комплексные программы обеспечения безопасности web-приложений. Этот факт порождает две проблемы. Во-первых, отсутствие полноценной программы значительно повышает риски и серьезно ухудшает последствия потенциальных атак. А во-вторых, отсутствие единой программы значительно увеличивает стоимость поддержания приемлемого уровня безопасности web-приложения, при котором риски и последствия потенциальных атак будут минимальными.

В данном отчете представлена информация о том, как построить прагматичную программу безопасности web-приложения за разумную стоимость, но, при этом, обеспечив его эффективную защиту. Вместо того, чтобы углубляться в конкретные детали какой-либо одной технологии, мы выделим основные составляющие системы безопасности и то как они должны взаимодействовать между собой. Начнем с того, что расскажем о том, чем отличаются web-приложения от коммерческих и других приложений. Затем мы предложим основные обоснования, которые можно использовать для получения бюджета для данной задачи. А также сосредоточим внимание на специфических потребностях в области безопасности web-приложений, углубимся в детали основных компонентов безопасности и, в результате, объединим их в общую программу безопасности web-приложений, основанную на типичных примерах использования.

ПРОБЛЕМА БЕЗОПАСНОСТИ WEB-ПРИЛОЖЕНИЙ

Коммерческие web-приложения развивались таким образом, что стали головоломкой с точки зрения безопасности. Несмотря на то, что все в ИБ наблюдали за их появлением развитием, на первых этапах их относили к приложениям с не очень высоким уровнем риска. Но в короткое время web-приложения выросли из экспериментальных программ в критически важные бизнес-приложения. Спросите у любого разработчика web-приложений и он расскажет вам историю о том, как их маленький внутренний проект вдруг стал важнейшей частью бизнеса, как только они сделали ошибку, расхвалив его бизнес-подразделению своей компании. В результате, сейчас нет возможности прервать развитие web-приложений и вернуться к основам, чтобы учесть все важные моменты, которые были упущены на заре становления, что в итоге привело к текущей ситуации в плане безопасности:

- До начала активного использования web-приложений, лишь малое число компаний организовали взаимодействие своих внутренних систем с внешним миром. И даже те, кто сделали это, предоставляли доступ лишь бизнес-партнерам, а с широким кругом пользователей работали и вовсе – единицы.
- Web-приложения выросли самостоятельно – ониначинали с информационных сайтов, которые были лишь чуть больше, чем каталоги. А затем через базовые сервисы они были

связаны с критически важными бэк-енд системами.

- Те приложения, что разрабатывались достаточно долго и не менее долго насыщались функционалом, зачастую вызывают к себе излишнее доверие, хотя по факту могут быть также уязвимы, как и те, что делались впопыхах. В данном случае срабатывает классический принцип "лягушка в кастрюле" – если лягушку закинуть в горячую воду, то она сразу выпрыгнет, а если подогревать воду постепенно, то сварится, даже не заметив этого.
- Протоколы web-приложений разработаны с целью обеспечить простоту и гибкость взаимодействия, но при этом, в значительной степени уязвимы.
- Инструменты и методы разработки web-приложений очень быстро развиваются, но по-прежнему полагаются на большое количество унаследованного кода.
- Угрозы для web-приложений развиваются также быстро, как и сами web-приложения, а кроме того, регулярно появляются угрозы в отношении того, что было сделано в прошлом. Web изначально не предназначался для безопасного запуска критических приложений, а это значит, что все разработанные приложения фактически используют неподходящую для этого платформу.
- Только некоторые приложения разработаны с нуля с учетом всех требований безопасности, а также поддерживаются в актуальном состоянии достаточно долго.
- Когда случаются инциденты безопасности, имеются проблемы с их обнаружением и анализом.

В свете всех этих факторов, стоит отметить, что финансирование разработки web-приложений, как правило, является недостаточным. Кроме того, как и все разработчики, создатели web-приложений находятся под постоянным прессингом, жестко ограничены в сроках и, зачастую, вынуждены закрывать глаза на некоторые не совсем правильные с точки зрения безопасности моменты в угоду бизнес-задачам. Мы категорируем проблемы безопасности web-приложений следующим образом:

- Лишь некоторые приложения разработаны с учетом требований безопасности.
- Приложение зависит от используемого web-браузера, который не является ни безопасной, ни доверенной средой.
- Web-приложения развивались с целью обеспечения доступа к самым чувствительным системам бэк-енда из публичной сети без должного уровня безопасности.
- Web не был разработан как безопасная платформа и, соответственно не соответствует тривиальным правилам безопасности – конфиденциальность, целостность, доступность.
- Лишь небольшое количество нарушений безопасности обнаруживаются и позволяют проанализировать потери компании и, соответственно, привести к усилению внимания к проблеме со стороны руководства.
- У нас есть огромный объем старого кода, который постоянно исправляется, но, в тоже время, постоянно дописывается все новый и новый код.
- Web-приложение – это комплекс платформ, инструментов и сервисов от множества провайдеров, каждый со своими вызовами безопасности.
- Многие web-приложения имеют критическое значение, но разработаны без учета этого

факта.

- Если Web-приложение разработано внутри компании, то именно внутри компании должны отслеживаться уязвимости и выпускаться исправления – никто другой эту работу не сделает.
- Требования безопасности зачастую вступают в конфликт с требованиями, выдвигаемыми бизнесом, в частности, если это касается простоты использования и скорости работы.
- Ни один инструмент безопасности web-приложений не обеспечит эффективную защиту в одиночку.

Чтобы подытожить написанное, предлагает такую аналогию: при обеспечении безопасности web-приложений, мы как будто пытаемся защитить истребитель находящийся в центре воздушного боя с постоянно меняющимися функциональными требованиями, деталями, топливом и законами физики. И это притом что в этом процессе истребитель постоянно атакуют миллионы разнообразных вражеских объектов, о которых мы практически ничего не знаем.

ОБОСНОВАНИЯ ДЛЯ БИЗНЕСА

Обеспечение безопасности приложений традиционно страдает от недофинансирования и убедить руководство в необходимости новых затрат на улучшение их защиты достаточно сложно, ведь оно и так отлично работает. Конечно, это касается не только приложений, а фактически является общей проблемой для всей ИБ, когда дополнительные меры по улучшению уровня защищенности не вызывают позитива у бизнеса, так как угрозы и последствия инцидентов плохо поддаются количественной оценке. Построение полной модели обоснования инвестиций в безопасность web-приложений выходит далеко за рамки данного материала, но мы не можем говорить о создании программы безопасности без, по крайней мере, некоторых основных инструментов, которые позволят определить сколько необходимо инвестировать и как убедить руководство поддержать вас. Следующий список не является исчерпывающим обоснованием бизнес-модели, но включает в себя основные драйверы, которые можно использовать для обоснования необходимости инвестиций в безопасность web-приложений:

Соблюдение требований. Нравится нам это или нет, информационная безопасность является объектом контроля со стороны государства, а также регулируется индустриальными стандартами и другими требованиями. Нам хотелось бы разделить обоснование в этом направлении на три отдельные части: собственно сами требования; дополнительные требования, которые могут возникнуть (защита персональных данных и т.д.); снижение затрат на соответствие требованиям и аудит. В среднем организация использует все три фактора для оценки инвестиций в безопасность web-приложения.

Борьба с мошенничеством. Если вы можете достаточно точно оценить потенциальный ущерб от мошенничества, это может быть одним из важных драйверов при обосновании инвестиций в безопасность. А в некоторых случаях вы можете получить конкретные данные о случаях мошенничества и показать каким способом их можно избежать с помощью тех или иных необходимых решений. Также имейте в виду, что вы можете не иметь правильной инфраструктуры для обнаружения и оценки случаев мошенничества, что само по себе может стать достаточным обоснованием. Тесты на проникновение также могут быть хорошим шансом на получение инвестиций для снижения количества случаев мошенничества – тест может показать неизвестные пути для использования эксплойтов, активную атаку или возможности для будущих атак. Вы можете использовать результаты теста на проникновение для оценки потенциальных возможностей для мошенничества и составить план, который позволит снизить риски до приемлемого уровня.

Экономия. Как мы уже упоминали в секции о соблюдении требований, контроль безопасности может снизить расходы на аудит, но это лишь дополнительная возможность для экономии. Использование средств безопасности web-приложений при разработке и в процессе использования позволит снизить затраты на ручной контроль безопасности и устранение выявленных уязвимостей, а также в общем повысить эффективность работы web-приложения. Мы также можем рассчитывать на экономию от сокращения количества инцидентов, а также при восстановлении работоспособности web-приложения после них.

Доступность. При работе с web-приложениями мы оцениваем как время общей доступности сервиса, так и доступность услуг (есть вероятность потери части функционала приложения из-за атаки или устранения ее последствий). Например, крайне редко можно увидеть полное отключение сайта вследствие проблем с безопасностью web-приложений, в то время как отключение части функций, в том числе платежной системы встречается гораздо чаще. Кроме того, часто бывают случаи, когда при атаке приходится самостоятельно отключать некоторые

функции для обеспечения защиты пользователей и их данных, даже в том случае, если атака не задевает возможности данной услуги напрямую.

Защита пользователей. Хотя этот параметр и не поддается прямому подсчету и не может быть выражен какой-либо суммой денег, главным стимулом для инвестиций в безопасность web-приложения является необходимость защиты пользователей и их конфиденциальных данных, так как невнимание к этому моменту может негативно сказаться на их доверии к компании и, соответственно, на ее имидже. Злоумышленники вообще достаточно часто не наносят никакого ущерба скомпрометированному сайту, но используют его для атак на пользователей. То есть даже если вы ничего не теряете в результате атаки на вашу организацию, это остается проблемой, так как ваше web-приложение может стать площадкой для атак на ваших пользователей. Большая часть компаний получает прямой или косвенный доход от данных пользователей и это уже создает обязательство о необходимости защиты этих данных.

Защита репутации. В то время как множество моделей пытаются дать количественную оценку репутации и оценить потенциальные потери от ее ухудшения, реальность такова, что все эти модели фикция – точного метода измерения потенциального ущерба для репутации из-за атак злоумышленников не существует. Несмотря на многочисленные прогнозы, относительно вероятных последствий утечки информации и последствия этого события для репутации, большая их часть не сбывается. А бывает и наоборот. К примеру, после того, как крупный ритейлер TJX допустила одну из крупнейших утечек данных в истории и этот факт был обнародован, продажи пошли вверх. Но только тот факт, что невозможно точно просчитать репутационные риски, не означает, что это не является важным фактором при обосновании необходимости инвестиций в безопасность web-приложений. Просто спросите себя (или менеджмент компании) о том насколько важно приложение для имиджа компании и насколько готовы к рискам связанным с потерей информации клиентов. Доверие пользователей, инвесторов и партнеров вашей компании и сервисам хоть и не зависит напрямую от защищенности web-приложения, но все же этот момент очень важен и может сказаться на общей ценности бизнеса.

Прямые потери от инцидентов. Кроме потерь от мошенничества, так же имеют место быть и прямые финансовые потери. Даже если мы не будем обращать внимания на репутационные потери бизнеса, а также стоимости компании, а сосредоточимся только на потерях, которые выражаются конкретными цифрами, то выясниться, что и они есть. К примеру, на рассылку уведомлений пользователям, восстановление работоспособности и оплату персонала call-центра.

Вы поймете какая комбинация из этих работ лучше, основываясь на потребностях вашей компании и приоритетах менеджмента, но вы должны объединить качественные и количественные оценки для того чтобы верно расставить приоритеты при инвестициях. Если вы работаете с персональными данными (финансы/розница/медицина) то контроль безопасности и снижение расходов будут лучшими доводами. Для web-приложений общего назначения важными будут защита пользователя и репутации, снижение рисков мошенничества и обеспечение доступности. И давайте не будем забывать о том, что все эти доводы также применимы и для внутренних приложений. Независимо от сферы применения, недостатка в обоснования для бизнеса (в отличие от технической стороны) в пользу инвестиций в безопасность web-приложений нет.

WEB-ПРИЛОЖЕНИЯ ДРУГИЕ

Мы много лет потратили на изучение вопросов безопасности сетей и хостов и построили программы безопасности в области информационных технологий сосредоточившись на базовой инфраструктуре. Но, как мы уже отмечали, безопасность web-приложений заметно отличается от всего этого и требует совершенно иного подхода. Безопасность web-приложений отличается и от защиты традиционного ПО, хотя и имеет с ним гораздо больше общего. В предыдущей секции мы сфокусировались на бизнесе, а теперь перейдем к технической части и поговорим о специфических технических и нетехнических отличиях в безопасности web-приложений, перед тем как дадим наши рекомендации по жизненному циклу защиты web-приложений.

ПОЧЕМУ ЗАЩИТЫ WEB-ПРИЛОЖЕНИЙ ОТЛИЧАЕТСЯ ОТ ЗАЩИТЫ СЕТЕЙ И ХОСТОВ

При обеспечении безопасности сети или хоста мы сосредоточены на блокировании пользовательских возможностей в пределах программного обеспечения, устройств и систем. Также при обеспечении безопасности корпоративных приложений мы в основном занимаемся блокированием платформ, защитой коммуникаций, аутентификацией пользователей и осуществляем контроль безопасности посредством средств, предоставляемых платформой приложения. Но в случае с web-приложениями мы сталкиваемся не только со всеми этими вопросами, но и имеет дело со сторонним или собственным кодом. В зависимости от того, является приложение доступным извне или предназначенным только для внутреннего использования, они имеют существенные отличия:

UI браузера: В большинстве приложений мы создаем пользовательский интерфейс. Но в случае с web-приложениями, мы полагаемся на сторонний просмотрщик (браузер), который не можем полностью контролировать и который может контактировать и с другими приложениями в один момент времени.

Собственный код порождает собственные уязвимости: В случае в web-приложениями вы обычно самостоятельно пишите код (используя при этом плагины и фреймворки). Это значит, что большая часть уязвимостей в вашем приложении будет уникальная. Если вы не будете контролировать свое приложение, то никто не сообщит вам о каких-либо имеющихся уязвимостях и никто не предложит патч, позволяющий их закрыть.

Вы разработчик: Когда появится уязвимость, у вас не будет разработчика, который подготовит патч (вы должны, конечно, установить патчи для всех инфраструктурных компонентов, фреймворков и скриптов, которые вы используете). Если вы обслуживаете клиентов, то вам необходимо придерживаться пункта договора об уровне сервиса и обеспечивать необходимую доступность сервиса.

Межсетевые экраны в одиночку не могут защитить web-приложение: Когда мы обнаруживаем уязвимости в нашем корпоративном ПО, операционных системах, приложениях, базах данных и т.д., мы используем инструменты вроде межсетевых экранов или IPS для блокирования возможностей для потенциальной атаки на время ожидания выпуска патча для уязвимого программного обеспечения. Такая “блокировка до патча” имеет лишь ограниченную эффективность. A Web Application Firewall (WAF) не сможет защитить вас от логических ошибок. Хотя WAF может помочь с некоторыми классами атак, но “из коробки” они не знают или не понимают выше приложение, а потому не сможет “прикрыть” пользовательские уязвимости. WAF безусловно является важной частью защиты web-приложений, но лишь в том случае, если он является частью комплексной

программы, о которой мы поговорим позже.

Вечная бета-версия: При разработке традиционного приложения проводится полный цикл проверки: контроль на стадии разработки, внутреннее тестирование, а также, зачастую, дополнительное тестирование с привлечением сторонних пользователей – бета-тестеров. И все это до того, как приложение перейдет в эксплуатацию. Было бы здорово, если бы и web-приложения проходили этот последовательный цикл, но, как уже было сказано, так бывает крайне редко. Большая часть web-приложений, даже рассматриваемых разработчиками как неокончательная, в действительности уже запущена в эксплуатацию и даже стали ключевыми решениями критически важными для бизнеса. Другие приложения и вовсе находятся в режиме постоянного изменения и их разработчики даже не придерживаются формального цикла разработки. Постоянно меняющиеся приложения – это непростая задача для существующих средств контроля безопасности, таких как WAF.

Опора на фреймворки/платформы: Web-приложения редко строятся с нуля, радуя "вылизанным" кодом на C. Чаще всего это смесь разнообразных фреймворков и средств разработки и платформ, которые не всегда разработаны для совместной работы. Задача разработчиков обеспечить взаимодействие всех этих частей, а также собственного кода. Такой подход, зачастую, создает серьезные проблемы безопасности из-за неправильного использования компонентов, их взаимодействия между собой, а также с собственным кодом.

Унаследованный код: Даже если новый код пишется с чистого листа и действительно безопасен, то не стоит забывать, что в большинстве случаев есть еще огромный массив старого кода, который также имеет огромное число уязвимостей, а потому подлежит анализу и исправлению. Если старый код находится в активном использовании, то он должен быть столь же безопасен, как и новый. Зачастую, именно устаревший код становится виновником атак на web-приложение.

Динамический контент: Большая часть web-приложений имеет чрезвычайно динамичный характер, создавая большую часть контента "на лету", нередко с использованием данных (в том числе кода), предоставленных пользователем. А браузер пытается все это обработать, создавая, тем самым, новые классы проблем безопасности.

Новые классы уязвимостей: Как и в случае с классическими приложениями, исследователи и злоумышленники постоянно открывают новые классы уязвимости web-приложений. Таких примеров множество. А потому нужно помнить, что даже идеальный с точки зрения сегодняшнего дня код вовсе не гарантирует то, что он всегда будет безопасен.

Мы перечислили ряд причин по которым следует смотреть на web-приложения иначе, чем на классические. Внедрение программы безопасности web-приложений требует достижения консенсуса в компании. Вносить изменения будет трудно и рискованно. В перспективе это приведет к снижению расходов, но оплатить все изменения придется сразу, не зная в какой мере эти инвестиции оправдаются. То есть для того, чтобы убедить руководство в необходимости инвестиций, вам понадобятся веские причины. И если вы сопоставите стоимость с преимуществами, описанными в этом разделе, вы получите вполне объективный ответ на то как данные расходы на безопасность повлияют на вашу компанию.

ЖИЗНЕННЫЙ ЦИКЛ ЗАЩИТЫ WEB-ПРИЛОЖЕНИЯ

В оставшейся части материала вы найдете рекомендации по действенным шагам, которые способствуют повышению уровня безопасности web-приложений. Учитывая масштабы задачи, мы не сможем дать подробную информацию по каждой составляющей программы безопасности, а рассмотрим их в общих чертах. В то время как web-приложения ставят отличные от привычных задачи, дополнительные шаги для защиты не отличаются от тех, что вы делаете сейчас. Упрощенную схему цикла разработки и использования мы разложили на 3 шага и 7 зон:

БЕЗОПАСНАЯ РАЗРАБОТКА

Процесс и обучение: В этом разделе мы сфокусируемся на построении безопасности в цикле разработки программного обеспечения. Это включает, в том числе, подготовку людей, занимающихся разработкой и улучшение процесса разработки. Обучение разработчиков должно быть ориентировано на обеспечение безопасности функциональных возможностей приложения. Мы также рассмотрим инструменты, которые помогут автоматизировать часть работы: статистический анализ на поиск уязвимостей в коде и динамический анализ для выявления аномального поведения приложения.

- **Безопасная разработка:** Внедрение практик безопасной разработки в процессе создание web-приложения.
- **Статический анализ:** Инструмент для сканирования исходного кода приложения на ошибки безопасности. Также эти инструменты называют “white box”.
- **Динамический анализ:** Инструменты исследующие не исходный код, а само запущенное приложение при попытках атаки. Эти инструменты часто называют “black box”.

БЕЗОПАСНОЕ РАЗВЕРТЫВАНИЕ

На этапе, когда разработка приложения уже закончена или, как минимум, оно доведено до такой стадии, когда возможно использовать его для тестирования, приходит время проверить, что приложение свободно от известных проблем безопасности и настроено нужным образом. В это же время можно начинать оценивать уязвимость приложения и проводить тесты на проникновение, наряду с традиционным анализом конфигурации и проверок согласованности взаимодействия с системами.

- **Оценка уязвимости:** удаленное сканирование web-приложения, как с учетной записью, так и без. Оценка уязвимости web-приложения концентрируется на самом приложении, в то время как стандартная оценка уязвимости сосредотачивается на исследовании хост-платформы.

- **Тестирование на проникновение:** Тестирование на проникновение является фактически попыткой взлома, которая позволяет выявить уязвимости в системе безопасности и те риски, которые они несут. Тестирование на проникновение позволяет оценить недостатки приложения, классифицировать их и правильно расставить приоритеты.

Большая часть людей не уделяет должного внимания тестированию собственного кода, а сосредоточены исключительно на расширении функционала, добавлении новых функции и достижению стабильной работы web-приложения. Это все, безусловно, важно и для этого собирается серьезная команда разработчиков. Таким же серьезным должен быть и подход к безопасности, когда код тщательно проверяется и каждая новая функция, добавляемая в приложение, подвергается тотальному анализу с точки зрения безопасности.

БЕЗОПАСНОЕ ИСПОЛЬЗОВАНИЕ

В этом разделе мы перейдем от инструментов и процессов, которые позволяют сделать приложение защищенным на этапе разработке, к тем, которые обеспечивают возможности обнаружения атак и реагирование на них. Основной акцент здесь делается на возможностях WAF, который должен защищать web-приложение от несанкционированных действий пользователей, а также других инструментах, которые сканируют запросы и выявляют неприемлемую активность в отношении web-приложения и других связанных с ним компонентов. Последние разработки в области средств обнаружения способствуют соблюдению политик, адекватно реагируют на события, а также объединяют несколько сервисов в кооперативную гибридную модель.

- **Web Application Firewall:** Сетевой инструмент, который занимается контролем трафика к web-приложению и сообщаете об известных атаках и/или блокирует их.

- **Мониторинг активности приложения и базы данных:** Инструмент, который контролирует активность приложения и базы данных (с помощью различных методов) для аудита и генерации предупреждений безопасности, основанных на нарушении заданных правил.

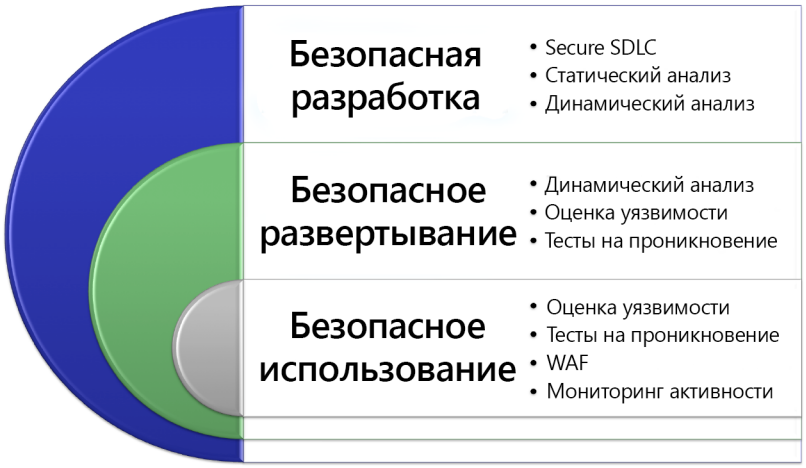
Поскольку многие организации только начинают создавать правила безопасности приложений, мы попытаемся наметить рамки и проиллюстрировать возможности, что позволит рассмотреть общую картину организации web разработки. Стоит подчеркнуть, что мы лишь предоставляем обзор имеющихся возможностей, а также базовую информацию о том как это работает и как это интегрировать. Более подробно, особенно о том “как”, написать в рамках материала невозможно – эта тема заслуживает нескольких полноценных книг. Специфика модификации процессов, применения различных методик и связанных с ними нюансов тоже выходят за рамки этого

документа.

Безопасность web-приложений претерпевает очень быстрые изменения – меняясь также быстро, как злоумышленники находят новые способы атак. Мы обсудим передовые инструменты и будущее рынка, а также конкретные инструменты, используемые для идентификации и защиты от новых видов атак. В то время как инструменты очень важны, они не могут использоваться в одиночку. Вам нужны правильные инструменты в правильных руках как часть правильного процесса. Тем не менее, мы будем уделять значительное количество времени инструментам, по следующим причинам:

- Многие инструменты автоматизируют большую часть передовых подходов к оценке безопасности
- Ваши разработчики и сотрудники службы контроля могут быть не в состоянии идти в ногу с постоянно меняющимися тенденциями информационной безопасности, а также иметь недостаточную компетенцию в данной области, а инструменты помогут в значительной степени нивелировать эти недостатки.
- Объем работ по тестированию на безопасность для обеспечения должного качества требует автоматизации. Соотношение требования/ресурсы для любой рабочей группы без автоматизации будет очень и очень плохим.
- Инструменты обеспечивают платформу для настройки и тестирования политик и являются результатом исследований и огромного опыта, что делает их незаменимыми на всех этапах разработки и использования web-приложения.

В дальнейшем мы разберем каждую из описанных частей и покажем, как они вписываются в общую программу безопасности web-приложения, а также укажем их достоинства и недостатки. Естественно, полного описания мы дать не сможем, так как каждый из разделов достоин отдельной книги, а потому сосредоточимся на основных моментах.



Перекрытие контроля на разных фазах

БЕЗОПАСНАЯ РАЗРАБОТКА

В безопасности web-приложений зачастую уделяется недостаточно внимания процессам модификации, обучения и выбору средств разработки. Безопасностью очень часто начинают заниматься уже после окончания основной разработки, а не во время ее. А ведь внимательность к этим вопросам при разработке позволяет сделать приложение более безопасным при меньших расходах. В этом разделе мы обсудим лучшие варианты для интеграции безопасности в web-приложение на этапе разработки.

БЕЗОПАСНОСТЬ WEB-ПРИЛОЖЕНИЙ: ПОДГОТОВКА И ЖИЗНЕННЫЙ ЦИКЛ РАЗРАБОТКИ ПРИЛОЖЕНИЙ

Большая часть сегодняшних web-приложений были спроектированы, разработаны и развернуты до того как появились нормы и принципы безопасности web-приложений. Безопасные методы разработки в командах web-разработки зачастую входят в сознание только после того, как случаются какие-либо инциденты. Менеджмент же, зачастую задумывается о безопасности только в рамках имеющихся требований, исполнение которых обязательно. Новости часто привлекают внимание к атакам с помощью SQL-инъекций, но большинство разработчиков не представляет как отражаются атаки с использованием межсайтового скриптинга, и еще меньше знает о том, как защититься от них. Практика безопасной разработки приложений находится в зачаточном состоянии и далека до зрелости. Конечно, значительные знания уже накоплены, но они пока еще не всеобъемлющи и доведены далеко не до каждого разработчика.

Независимо от того, что вами движет, образование и модификация процессов являются самыми важными первыми шагами для создания безопасного web-приложения. Если вы разрабатываете новое или модернизируете старое приложение, то менеджеры проекта, разработчики и другие специалисты должны быть осведомлены о потенциальных проблемах безопасности решения, а также о методах безопасного проектирования и написания кода. Учебная программа должна охватывать как общие угрозы, так и известные методы, которые злоумышленники используют для атак на системы. Специализированная подготовка нужна для сотрудников каждой субдисциплины, включая возможности модификации процессов, безопасную модель развертывания платформы, инструменты безопасности и методологию тестирования. Если ваша команда не знает, как обеспечить должный уровень безопасности, то вы будете зависеть в этом вопросе от третьих лиц (поставщики услуг или хакеры), а это обходится гораздо дороже.

Менеджеры проектов должны знать, какие типы угроз могут быть применимы к разрабатываемому web-приложению, и как минимизировать имеющиеся риски, обеспечив при этом требуемый функционал. Разработчики должны понимать принципы работы эксплойтов, то как их протестировать и, конечно же, способы устранения недостатков. Понимание – это ключ к созданию безопасных приложений, поэтому очень хорошо, если ваша компания инвестирует в образование сотрудников, организуя внутренний учебный процесс, проводя образовательные мероприятия, или оплачивает обучение в сторонних учебных центрах. Моделирование угроз, принципы безопасной разработки , снижение поверхности атак, требования функциональной безопасности и тестирование являются ядром принципов безопасной разработки приложений и легко интегрируются во все процессы и этапы разработки.

Процесс также играет важную роль в разработке приложения и влияет на безопасность также как на производительность сотрудников и качество продукта. Если спецификации продукта

не содержит требований к безопасности, вы не можете надеяться на то, что продукт будет безопасным. Продукт, который не подвергается тестированию на безопасность, также как и продукт, который не проходит тестирование функциональности, будет страдать от недостатков и ошибок. Модификация жизненного цикла разработки приложений (Software Development Lifecycle) включающая требования безопасности называется Secure SDLC и включает в себя простые проверки на протяжении всего процесса разработки для того чтобы выявлять проблемы как можно раньше. Secure SDLC - достаточно серьезная тема для углубленного обсуждения, но самое главное, что мы хотели бы донести – это необходимость внедрения требований безопасности в каждый этап разработки.

Инструменты и тестовые случаи, которые мы обсудим позже, могут быть использованы для автоматизации тестирования и контроля, но знания и образование имеют важное значение для их правильного использования. С их помощью можно улучшить процесс разработки и тестирования, а также снизить расходы по сравнению с наложенной безопасностью, одновременно снижая количество уязвимостей в коде. Члены команды, имеющие образование в сфере безопасности способны создавать библиотеки тестов, которые помогают выявить типичные уязвимости в новых версиях кода. Причем, пригодных не только для внутреннего тестирования на этапе разработки, но и полезных при тестировании уже работающего приложения. Экстремальные методики программирования могут помочь подтвердить, что модули и компоненты отвечают требованиям безопасности в рамках тестирования, наряду с проверкой функциональными возможностями, не относящимися к безопасности напрямую. Помните: вы разработчик и ваша команда должна знать код лучше, чем кто-либо еще включая его недостатки.

ИНСТРУМЕНТЫ СТАТИЧЕСКОГО АНАЛИЗА

Есть целый ряд специально разработанных инструментов сторонних разработчиков, которые помогают проверить код на безопасность, Статический анализ анализирует исходный код web-приложения для выявления распространенных уязвимостей, ошибок и упущений в пределах конструкций самого языка программирования. Это своего рода, автоматизированный аналог экспертной оценки. Помимо всего прочего, эти средства в основном используются для сканирования необработанных условий ошибок, наличия объекта и/или определения размера, а также потенциально возможного переполнения буфера.

Эти инструменты используются на стадии разработки для того чтобы выявить недостатки до перехода к более формализованным процедурам тестирования. Ведь чем раньше становится известно о проблеме, тем проще ее исправить. Статический анализ кода выполняется самими разработчиками, что значительно ускоряет поиск ошибок и делает это заметно дешевле, чем если бы анализом занимались отдельные люди. Эти инструменты могут быть интегрированы с управлением исходным кодом для автоматизированного выполнения анализа, опять же. Для того чтобы выявить недостатки в коде как можно раньше.

Статистический анализ эффективен для обнаружения проблем связанных непосредственно с ошибками в коде, которые допускают программисты. Лучшие инструменты хорошо интегрируются с разными средами разработки (предоставление дополнительной информации о выявленных недостатках кода и предоставление вариантов их исправления), расставляют приоритеты обнаруженных уязвимостей на основе заданных критериев, предлагают подробную отчетность и отслеживание тенденций, а также способны взаимодействовать с группой безопасности, участвующей в процессе разработки, не отвлекая других разработчиков.

Но инструменты статического анализа не учитывают все пути в коде, а также не может выявить

некоторые типы уязвимостей, которые проявляются только во время использования приложения. Чтобы восполнить этот пробел, в последние несколько лет стали активно развиваться инструменты для динамического и гибридного анализа.

ИНСТРУМЕНТЫ ДИНАМИЧЕСКОГО АНАЛИЗА

Динамический анализ используется для выявления проблем и уязвимостей, которые не могут быть выявлены при анализе исходного кода или которые намного проще заметить в тот момент, когда приложение запущено. Один из типов анализа называется "fuzzers" которые направляет приложению заведомо вредные или фиктивные материалы для приложения и отслеживает результаты этих действий, а также сбои в работе приложения. На рынке предлагается много разных вариаций, но в целом их можно разделить на три группы:

White Box или Black Box. Тестовое приложение может не иметь никаких начальных знаний о приложении и изучает его в качестве "черного ящика" (Black Box) и с соответствующим подходом. Но в большинстве случаев используется подход "White Box", когда тестирование проводится по поиску подходящих уязвимостей в соответствии с особенностями приложения известных авторизованному пользователю. Тестирование методом "Black Box" является более представительным, так как оно повторяет принципы работы внешнего хакера, который не знаком с внутренней организацией приложения. Но при тестировании web-приложений важны оба подхода. Ведь многие уязвимости могут быть не видны неавторизованному пользователю, но прекрасно видны тому, кто прошел авторизацию.

Отправка запросов. Запросы могут быть случайными, генерироваться на лету на основе ответов, а также быть целевыми. Случайные запросы являются хорошей возможностью для проверки эффективности проверки целостности и поиска общих недостатков в коде, а целевые могут быть использованы для проверки на наличие уже известных уязвимостей.

Выходные данные и действия по окончании тестирования. После тестирования методами White Box и Black Box требуется экспертиза результатов тестов специалистами, которые выделят в них реальные недостатки и ложные срабатывания. Условия появления ошибок найти достаточно легко и большая часть инструментов для динамического анализа обнаруживает и сообщает об их наличии. Некоторые средства мониторинга не только выявляют ошибки, но и указывают на то, в какой именно части кода они находятся. Подобно отладчикам, динамический анализатор может контролировать внутренние ресурсы приложения, когда оно находится в процессе тестирования, наблюдая за памятью, указателями, очередями запросов и переменными. Тесты могут выявить специфические эффекты модели использования системы и проблемные области.

Отчасти из-за этих переменных инструменты для динамического анализа отличаются по скорости, эффективности и автоматизации. Так как они сфокусированы на поведении приложения, они дают конкретные результаты для сценариев "а что если...", чего статические инструменты не могут. Динамический и статический анализ являются взаимодополняющими технологиями, но предназначены для разных аудиторий. Некоторые вендоры предлагают обе возможности в одном инструменте, что позволяет получить более всеобъемлющую оценку.

Хорошо структурированная программа безопасности web-приложений начинается с хорошего образования и интеграции безопасности в жизненный цикл разработки приложений – с моделированием угроз, безопасным дизайном, учетом безопасности функциональных требований, надлежащим использованием статического и динамического анализа, использованием безопасных методов кодирования, непрерывным обучением и формализованным тестированием.

БЕЗОПАСНОЕ РАЗВЕРТЫВАНИЕ

Сохраняя последовательность повествования, мы переводим фокус нашего внимания с разработки на развертывание web-приложения. На стадии развертывания, после проверки качества и безопасности приложения, мы переходим к оценке уязвимости и тестированию на проникновение для того чтобы убедится в соответствии нашего приложения современным требованиям и его устойчивости к атакам. Помните, что мы смотрим на безопасность web-приложений как на непрерывный процесс с перекрывающимися этапами. Хотя мы и делим процесс на отдельные этапы, чтобы сделать повествование более понятным и структурированным, это совсем не значит, что один этап заканчивается, когда начинается другой. К примеру, вы должны продолжать использовать динамический анализ на этапе развертывания, а также продолжать поиск уязвимостей и тесты на проникновение на всех этапах. Также помните, что мы показываем вам общую картину и имеющиеся возможности. Приоритезацию и моменты, на которых стоит особо сфокусироваться, если ваш бюджет ограничен, мы обсудим чуть позже – ведь мы не настолько наивны, чтобы думать, что вы можете позволить себе все решения, которые есть на рынке.

ОЦЕНКА УЯЗВИМОСТИ

При оценке уязвимости, мы сканируем web-приложение для идентификации любых моментов, которые потенциально могут быть использованы против нас атакующими (также мы оцениваем соответствие требования и стандартам, но фокусируемся именно на поиске уязвимостей). Сканирование может быть выполнено посредством инструментов, сервисов или их комбинации.

Оценка уязвимости web-приложений очень отличается от стандартной оценки уязвимости, которая фокусируется на сетях и хостах. В том случае мы фокусируемся на сканировании портов, подключениях к сервисам и используем другие методы для сбора информации и выявления уровней патчей. Как мы уже говорили, даже “стандартные” web-приложения во многом являются самописными, а потому нам необходимо изучать их более внимательно, проверяя функции и логику приложения, а также использовать индивидуальные проверки, чтобы выявить уязвимости web-приложения. При большом количестве самописного кода мы должны меньше полагаться на поиск стандартных уязвимостей и стараться проводить больше тестов путем атаки. Как мы уже говорили, пользовательский код создает пользовательские уязвимости.

При оценке уязвимости web-приложений мы пропускаем традиционные тесты хостов и сетей и фокусируемся на сторонних web-приложениях и фреймворках, а также на технических и бизнес недостатках пользовательских приложений.

Рынок решений для оценки уязвимости web-приложений предлагает как специальные инструменты, так и сервисы. Если вы выберете пусть самостоятельного тестирования, то, кроме подбора правильных инструментов, вы должны быть уверены, что они окажутся в руках опытного оператора, который сможет правильно интерпретировать результаты тестов. Важны использовать обе методики тестирования, когда оператор имеет разные уровни доступа к web-приложению и не имеет таковых вообще.

Инструменты и решения

Имеется целый ряд коммерческих, бесплатных и основанных на открытом коде инструментов

и решений для оценки уязвимости web-приложений с широким функционалом. Некоторые инструменты используют очень ограниченный набор эксплойтов, а потому опытные тестировщики используют не один, а целый набор инструментов, дополняя его ручными методами проверок. Например, есть приложения сфокусированные на поиске и тестировании только уязвимостей связанных с SQL-инъекциями. Инструменты корпоративного класса должны быть более многофункциональными и включать ряд критически важных для web-приложения классов уязвимостей, таких как SQL-инъекции, межсайтовое исполнение скриптов и т.д. OWASP Top 10 является отличным базовым списком основных уязвимостей, но приложения корпоративного класса не должны ограничиваться проверкой только по одному списку или категории уязвимостей. Решение корпоративного класса также должно иметь возможность проверки нескольких приложений, поддерживать отслеживание в течение длительного времени и обеспечивать адекватной отчетностью (особенно в той части, что касается исполнения требований), а также быть настраиваемым для оценки выполнения локальных требований.

Инструменты для оценки уязвимостей, как правило, являются программными, но могут иметь и аппаратную часть. Также они могут работать либо под контролем оператора, либо проводить сканирование в автоматическом режиме по расписанию. Так как web-приложения изменяются достаточно активно, то очень важно сканировать их после внесения любых изменений или новые версии перед развертыванием, а также контролировать действующие приложения на постоянной основе.

Услуги

Не все организации имеют ресурсы (включая экспертизу) для оценки уязвимости собственных приложений. В таком случае есть два варианта – покупка необходимых инструментов или внешняя оценка.

Существует три основные категории услуг по оценке уязвимости web-приложений:

Полностью автоматическое сканирование. Полностью автоматизированное сканирование без участия оператора. Цена такой оценки минимальна, но она дает гораздо больше ошибок. Такие проверки достаточно эффективны для постоянного мониторинга, но их возможности довольно ограничены для проверок перед развертыванием.

Автоматическое сканирование с обработкой результатов оператором. В данном варианте большая часть проверок проводится в автоматическом режиме, после чего результаты расшифровываются человеком, которые, по необходимости проводит дополнительные тесты. По сравнению с полностью автоматической проверкой, данный вариант дает более глубокое исследование и более точные результаты, но стоит, естественно дороже.

Ручное обследование. В данном варианте оценку проводят подготовленные специалисты, которые используют свои собственные инструменты для поиска уязвимостей и их проверки, после чего предоставляют развернутые отчеты. Этот вариант самый эффективный и позволяет выявить больше уязвимостей, чем предыдущие, но он и самый дорогой.

ТЕСТИРОВАНИЕ НА ПРОНИКНОВЕНИЕ

Целью оценки уязвимости является поиск путей, которые потенциальные злоумышленники могут использовать, а тест на проникновение это еще более глубокое исследование, которое позволяет понять – действительно ли найденные уязвимости несут риски для компании в случае атаки. В

тесте на проникновение мы пытаемся атаковать наши приложения аналогично злоумышленнику, чтобы понять какие могут быть последствия у данных действий.

Оценка уязвимостей и тесты на проникновение отлично дополняют друг друга и часто используются вместе, так как первый шаг в любой атаке – это поиск уязвимостей, которые можно было бы эксплуатировать. Целью на этапе оценки уязвимостей является поиск недостатков web-приложения, а при тестировании на проникновение эти недостатки проверяются с точки зрения возможности их использовать, оценить потенциальный ущерб каждой из них и расставить приоритеты по исправлению этих недостатков.

Ключевая ценность теста на проникновение в том, чтобы определить связь между уязвимостью и рисками, которая она несет и принять обоснованное решение о необходимости и срочности ее устранения. Оценка уязвимости дает только информацию о наличии пробелов в безопасности, но не позволяет представить общую картину потенциальных последствий их эксплуатации. Например, при поиске уязвимостей вы выявили, что web-приложение уязвимо для SQL-инъекций, но при попытке эксплуатации это уязвимости может оказаться, что с ее помощью злоумышленник может получить конфиденциальную информацию, изменить функциональность или и вовсе вывести web-приложение из строя. Может оказаться, что незначительные уязвимости могут привести к тому, что злоумышленник получит полный контроль над всем web-приложением.

Некоторые эксперты считают, что тесты на проникновение важны, так как используют те же методы и преследуют те же цели, что и реальный злоумышленник, но мы считаем, что эти тесты очень полезны как инструмент определения приоритетности рисков. Впрочем, обе этих цели тестов являются важными и нельзя отказываться от одной из них, отдавая приоритет другой.

Как и в случае с оценкой уязвимости, тесты на проникновения могу не ограничиваться только web-приложением, но, также включать проверку уязвимостей окружения – операционных систем, сетей, пользовательских приложений и т.д. Инструменты и услуги этого направления доступны на рынке. Но, учитывая, что мы говорим о web-приложениях, мы не будем углубляться в окружение и инфраструктуру, и ограничимся обсуждением аспектов тестирования на проникновение характерных для web-приложений.

Инструменты и решения

Решения для тестов на проникновение дают возможность выявления и проверки возможности эксплуатации уязвимостей web-приложения, а также оценки возможных последствий атак с их использованием. Специалисты по тестам на проникновение используют множество разных инструментов при работе, но большая их часть является узкоспециализированной и не предоставляет широкого функционала. Они запускаются до развертывания web-приложения с использованием тестовых данных, так как эксплуатация некоторых уязвимостей может привести к повреждению хранящихся данных или самого приложения. Решения корпоративного класса имеет дополнительные функции, которые помогают более эффективно управлять рисками и имеют внутреннюю систему оценок уязвимостей, что позволяет снизить затраты при проведении внутренних тестов на проникновение. Также они обычно имеют более широкий охват классов уязвимости, поддерживают “безопасные”для данных и web-приложения методы проверки, что позволяет использовать их в действующих приложениях, а также предоставляют обширную отчетность и автоматизацию текущих оценок. Одним из преимуществ решений для тестирования на проникновения является возможность их использования не только для проверки уже действующего приложения, но и на разных этапах разработки и развертывания web-приложения.

Инструменты и решения всегда поставляются в виде программного обеспечения и должны

использоваться только подготовленными специалистами. Ведь, несмотря на то, что значительная часть процесса автоматизирована, только хорошо подготовленный и опытный специалист может проанализировать результаты автоматического тестирования и эффективно исследовать уязвимость, сделав выводы о последствиях е возможной эксплуатации.

При использовании инструментов для тестирования на проникновение (также как и в случае с инструментами для поиска уязвимостей) у вас есть возможность запускать их в безопасном режиме, что заметно снижает риск возникновения проблем с работающим web-приложением. Но если приложение действительно работает в "боевом" режиме, то будьте осторожны даже при работе в безопасном режиме, так как и в этом случае имеется определенный риск перебоев в работе web-приложения, а также захламления баз данных. Поэтому такое тестирование нужно проводить под максимальным контролем.

Услуги

В отличие от того, что мы видели при изучении поиска уязвимостей, полностью автоматизированных сервисов по тестированию на проникновение нет. Из-за гибкой и постоянно меняющейся природы web-приложений, тесты на проникновение всегда требуют контроля со стороны человека. Услуги по тестированию на проникновения предлагаются либо в качестве разовых услуг либо в качестве подписки, которая подразумевает проведение тестов с определенной периодичностью. Частота тестирования варьируется в зависимости от потребностей бизнеса, критичности и задач web-приложений. Внутренние приложения могут проверяться всего раз в год в рамках регулярного тестирования инфраструктуры.

Перед использованием услуг тестирования на проникновение, необходимо внимательно изучить опыт знания и возможности поставщика услуг. Некоторые компании предлагают немного больше, нежели удаленное сканирование, но не в состоянии предоставить адекватные результаты, которые вы сможете эффективно использовать в риск менеджменте. Другие просто останавливаются после того, как получают доступ, к приложению используя одну уязвимость, не проводя полную проверку. Поэтому при выборе поставщика услуг вы должны предпочтение организациям с отработанным процессам, которые могут предоставить образцы отчетов и будут соответствовать вашим ожиданиям. Большая часть тестов на проникновения структурированы и имеют фиксированное время и глубину исследования. В случае с фиксированным временем (самый распространенный вариант) тестировщики пытаются найти так много уязвимостей как могут в пределах ограниченного по взаимной договоренности времени. Работа также может быть разбита на фазы, к примеру: слепой поиск, поиск под аккаунтом авторизованного пользователя и т.д. Тестирование с фиксированной глубиной подразумевает остановку тестирования только при достижении определенной цели (например, получение прав администратора или доступ к номерам платежных карт) и не зависит от затраченного времени.

Интеграция оценки уязвимости и тестирования на проникновение в безопасное развертывание

Несмотря на то, что большая часть компаний фокусируется на оценке уязвимости и тестировании на проникновении уже действующих внешних web-приложений, на самом деле этот процесс должен начинаться на этапе развертывания и не должны ограничиваться только теми web-приложениями, которые доступны извне. Очень важно провести проверку приложения до того, как вы предоставите доступ к нему внешними или даже внутренним пользователям. Также нет никаких причин, кроме, пожалуй, ресурсов, чтобы не внедрить оценку рисков и тестирование на

проникновение в процесс развития приложения на основных этапах.

Перед развертыванием web-приложения создается тестовая среда, которая аналогична той, в которой будет работать "боевая" система. Все ее элементы, включая версии продуктов и подключения к внешним сервисам должно быть идентичным. Затем проводится оценка уязвимости и тестирование на проникновение. Если вы используете собственные инструменты, то у вас должен быть и обученный персонал, который умеет эти инструменты применять. Если вы пользуетесь услугами сторонней компании, вам необходимо обеспечить удаленный доступ к тестовой среде – изучить и выполнить требования доступа имеющиеся у подрядчиков. Не каждая компания может обеспечить полное и всеобъемлющее тестирование каждого своего приложения, поэтому необходимо приоретизировать список ваших web-приложений, основываясь на критичности приложения для бизнес-операций, использовании приложением конфиденциальных данных и т.д.

Для некоторых крупных и особо важных web-приложений, вы могли бы провести оценку уязвимости и тестирование на проникновение у сторонних компаний еще до начала развертывания. Если же ваше web-приложение критично для бизнеса или рассчитано на публичный доступ, то такая проверка просто необходима.

После того как web-приложение уже развернуто, вашей задачей будет выполнение хотя бы основных проверок при внесении каких-либо изменений в его код, еще до того, как обновление будет установлено на "боевую" систему. Для критически важных приложений периодически пользуйтесь услугами по внешней оценке уязвимости и тестированию на проникновение (по крайней мере, раз в год, или при выпуске крупных обновлений) в дополнение к проведению собственных тестов.

Мы знаем, что у разных компаний имеются разные ресурсы на содержание актуальных инструментов для тестирования и услуги сторонних компаний, а потому вам нужно адаптировать эти рекомендации к реалиям вашей организации. Мы предоставили лишь достаточно общие данные в расчете на достижение лучшего результата и некоторые данные, которые позволят правильно расставить приоритеты.

БЕЗОПАСНОЕ ИСПОЛЬЗОВАНИЕ

В предыдущих частях мы рассказывали о безопасной разработке и безопасном развертывании, а теперь настала пора переходить к безопасному использованию. Этот этап настает, когда разработка и тестирование уже полностью закончены и web-приложение переходит в режим активного использования. Помните, что большая часть того, о чем мы говорили на предыдущих этапах, пригодится вам и на этом, так что не торопитесь выбрасывать использовавшиеся инструменты и забывать полученные навыки. Обновления для web-приложения по-прежнему должны безопасно разрабатываться и проходить всестороннее тестирование перед установкой, оценка уязвимости и тесты на проникновения также должны быть регулярными, а управление конфигурацией и постоянный контроль безопасности становятся даже более важными, чем были до этого.

В фазе безопасного использования мы добавим две новых категории технологий для поддержки двух новых процессов – Web Application Firewall (WAF) в качестве щита отражающего разные типы атак и решения по мониторингу web-приложения и/или базы данных, которое обеспечит контроль и информирование об угрозах безопасности.

WEB APPLICATION FIREWALL (WAF)

Задача WAF состоит в том, чтобы находясь перед web-приложением или в стороне, обеспечивать мониторинг активности приложения и предупреждать об угрозах безопасности или блокировать их. Таким образом WAF выполняет две функции –контроль активности в web-приложении и превентивный контроль безопасности.

WAF – это межсетевой экран (Firewall), который изначально спроектированный с прицелом на контроль HTTP-запросов и блокирование тех из них, которые могут быть потенциально опасными или не соответствуют заданным правилам. Задача WAF в том, что перехватывать SQL-инъекции, межсайтовый скриптинг (XSS), попытки обхода каталогов и другие попытки атак посредством HTTP-запросов, включая их подмену, то есть пресекать любые попытки несанкционированного использования web-приложения. Правила и политики задаваемые WAF позволяют эффективно обеспечить контроль HTTP-трафика в соответствии с функционалом web-приложения. WAF предупреждает или блокирует подозрительную активность на основе имеющихся сигнатур (уже известных атак) или на основе специфических требований web-приложения.

Современные WAF изучают входящие и исходящие запросы, сравнивает их с заданными правилами и выдает оповещения в случае обнаружения какого-либо отклонения. По итогам анализа подозрительных запросов, WAF выбирает один из четырех вариантов действий: 1) пропустить запрос, 2) пропустить, но зафиксировать факт нестандартного поведения, 3) заблокировать передачу запроса, 4) сбросить соединение.

WAF – это, обычно, сетевое устройство. Как правило, его размещают непосредственно перед приложением (режим прокси) или в стороне от приложения, зеркалируя на него трафик, которым обменивается web-приложение. В первом варианте установки все входящие и исходящие запросы перехватываются и проверяются до того как они попадут к web-приложению или пользователю, что позволяет снизить нагрузку на web-приложение. В том случае, если для связи с web-приложением используется SSL-соединение, установленный перед ним WAF должен иметь возможность его расшифровки и анализа. При установке WAF в стороне от приложения, проблема работы с SSL-

трафиком решается посредством выдачи ему копии серверного сертификата, либо установкой после SSL-концентратора. Некоторые разработчики также предлагают WAF в виде плагинов для конкретных платформ, то есть не имеющих своей аппаратной базы.

Эффективность некоторых WAF ограничена качеством политик. Политики являются важными не только для выявления и блокирования известных типов атак, но и для гибкого решения в случае выявления неизвестных типов атак, сохраняя при этом функциональность для нормальных запросов. Сложность web-приложений в комбинации с необходимостью постоянного обновления политик и разными вариантами развертывания, являются непростой задачей для всех разработчиков WAF. При этом просто установка WAF перед web-приложением и активация всех его возможностей – это верный рецепт катастрофы. Нет никаких шансов, что WAF сам поймет все особенности вашего web-приложения, а потому при его развертывании требуется тщательная и внимательная настройка для сохранения возможности пропуска "хороших" запросов и эффективной блокировки "плохих".

Когда WAF разворачивается в режиме мониторинга, то он функционирует по тому же принципу, что и система обнаружения вторжений (IDS). В данном режиме WAF используется только для анализа активности и выдачи предупреждений о нарушениях. В таком режиме обычно разворачивается WAF даже в том случае, если вы планируете использовать возможность блокирования запросов. Это дает возможность настроить WAF и лучше понять особенности функционирования web-приложения до того как вы перейдете к попыткам блокирования запросов. Преимущества этого режима заключается в возможности наблюдать за широким кругом потенциальных атак не беспокоясь о том, что ложные срабатывания приведут к неадекватной блокировке. К недостаткам данного режима можно отнести то, что ваши сотрудники будут тратить больше времени на решение появляющихся инцидентов и ложных срабатываний, а также то, что "плохая" будет блокировать с большей задержкой.

В режиме блокирования WAF будет останавливать соединения, разрывая их (в режиме прокси) или отправляя пакеты сброса TCP (если установлен в стороне). После сброса соединения WAF может заблокировать IP временно или постоянно для того чтобы предотвратить повторные атаки. Режим блокировки является самым эффективным в том случае, если имеется известная уязвимость в вашем приложении, но обновление с исправлением этой уязвимости еще не готово.

Когда уязвимость найдена в вашем приложении, вы можете создать специфическое правило для блокирования атак использующих эту уязвимость в WAF. Это защитит ваше приложение от атак, пока вы будете исправлять код приложения или дожидаться выхода патча от разработчика программного обеспечения. Данная стратегия позволяет достаточно эффективно защищать приложение, сохраняя его функциональность и повышая производительность, но она сработает только в том случае, если у вас есть отлаженные процессы по выявлению и анализу уязвимостей.

Учитывая все вышесказанное, стоит отметить, что отношение к WAF у профессионалов информационной безопасности остается неоднозначным. Хотя WAF очень эффективны против известных угроз, они гораздо менее эффективны в борьбе с новыми угрозами и не всегда адекватно обрабатывают сомнительные запросы. Некоторые продукты класса WAF решают эту проблему за счет более плотной интеграции с инструментами оценки уязвимостей. Независимо от того в каком режиме используется WAF и как эффективно настроены политики, вы должны помнить, что вам потребуются дополнительные инвестиции в эту область.

МОНИТОРИНГ

Мониторинг в основном предназначен для контроля использования web-приложения реальными пользователями, а также для выявления нестандартных запросов, которые могу оказаться "плохими". Принципиальная ценность монбиторинга в возможности узнать те особенности

приложения, с которыми вы не знакомы – это важно не только для правильного использования WAF, но и для общей стратегии безопасности приложения. Хотя WAF и обеспечивает определенный контроль, есть три дополнительных пути мониторинга web-приложений, каждый из которых позволяет по-новому взглянуть на активность приложения:

- Сетевой мониторинг. Мониторинг сетевой активности между пользователями и web-сервером. Эта категория включают WAF, системы обнаружения вторжений, анализаторы пакетов и другие внешние инструменты. Инструменты общей сетевой безопасности и другие инструменты для сниффинга могут перехватывать весь сетевой трафик, но они имеют небольшую совместимость и не могут выделить из общего трафика специфические пакеты web-приложения. Просто просмотра трафика недостаточно для того, чтобы понять, что пользователи пытаются сделать в приложении – требуется интерпретация. Если же решение умеет анализировать специфический трафик web-приложений и способно проводить аудит активности сети, то мы называем такое решение Web Application Monitoring (WAM – Мониторинг web-приложения). Мониторинг сети легко интегрировать, так как он не требует внесения изменений в приложение, и он может только просматривать исходящий и входящий трафик приложения. Это может быть полезно для выявления традиционных атак стека приложений, но гораздо менее полезно, когда трафик коррелируется с операциями более высокого уровня.

- Аудит приложений и сбор логов. Сбор и анализ логов и внутренней активности приложения. Приложения собственной разработки, равно как и сторонней, чаще всего имеют возможности по внутреннему аудиту и сбору логов, но все они отличаются по принципу сбора, принципу описания событий и формату записей. При этом вы все равно можете не получать полной картины того, что происходит в приложении, так как вы ограничены лишь теми данными, которые были добавлены в механизм логирования программистами. Для крупных и распространенных корпоративных приложений, таких как SAP, мы уже видели сторонние инструменты, которые позволяют добавить в них мониторинг или механизмы интерпретации внутренних логов. Общие инструменты логирования и SIEM также могут быть использованы для сбора и интерпретации событий (с определенными ограничениями) активности web-приложений, при условии генерации журналов событий.

- Database Activity Monitoring (DAM – мониторинг активности базы данных). DAM используют разные методы для регистрации событий связанных с обращениями в базу данных и генерации предупреждений при нарушении политик. Благодаря мониторингу активности между web-приложением и базой данных (или внутри нее) DAM может обеспечить более точное изучение и использование данных, а также предоставить дополнительную информацию о взаимодействии в рамках системы при многоступенчатых транзакциях в рамках бизнес-процессов. Некоторые DAM имеют плагины для основных типов приложений (например, SAP и PeopleSoft) для интерпретации событий базы данных в события активности приложения. Учитывая активное взаимодействие подавляющего большинства web-приложений с базами данных, этот подход к мониторингу является достаточно эффективным для отслеживания активности и поиска нарушений политик безопасности.

WAF может использоваться для мониторинга, но специализированные средства предоставляют больше возможностей по контролю активностей и позволяют выявить недостатки и нарушения, которые не являются очевидными только при контроле HTTP-трафика. Фокус WAM сосредоточен на источниках данных, их анализе, регистрации активности и генерации предупреждений о подозрительных событиях, в то время как WAF больше направлен на выявление и блокирование уже известных атак. Имеются значительные отличия между WAF и WAM в таких областях как хранение активностей, агрегирование данных из различных источников и их анализ, а потому

выбор лучшего решения зависит от специфических требований. Также стоит отметить, что решения по мониторингу web-приложений пока еще трудно назвать зрелыми – большая их часть находится на раннем этапе эволюции.

WAF + ОЦЕНКА УЯЗВИМОСТИ

Некоторые вендоры уже начали предлагать гибридные решения, которые объединяют в себе функционал продуктов для оценки уязвимости и WAF. Как уже упоминалось ранее, одной из сложностей при временном блокировании определенных уязвимостей с помощью WAF, является выявление этой уязвимости создание сигнатуры для ее блокирования. Объединив инструменты для оценки с WAF мы можем автоматизировать процесс создания и применения новых политик при выявлении новой уязвимости. Такая модель подразумевает выявление уязвимости и мгновенную передачу данных WAF для того, чтобы тот мог заблокировать определенные возможности.

ПОДВОДИМ ИТОГИ

В этом разделе мы попытаемся объединить все части касающиеся безопасности web-приложений и сделать некую инструкцию, которая позволит вам разработать программу безопасности подходящую для вашей организации. Безопасность web-приложений – это не тот случай, когда есть универсальный рецепт. Риски, размер и сложность приложений у всех разные, равно как и информированность о вопросах безопасности пользователей и сотрудников, а самое главное цели использования web-приложений у каждой организации свои.

Для того чтобы дать какие-то практические рекомендации, мы должны подходить к разработке web-приложения с точки зрения неких типичных задач. Поэтому мы выбрали три сценария для представления общих проблем с безопасностью web-приложений, с которыми сталкиваются компании, и рассмотрим каждый их них через призму программы безопасности. Мы обсудим web-приложения большой корпорации, ориентированные на работу с внешними пользователями, первое приложение средней по размеру компании, а также организации больше среднего размера, задача которой состоит в обеспечении защищенности внутреннего web-приложения. Сначала мы опишем среду для каждого приложения, а затем опишем общую стратегию и конкретные рекомендации.

КРУПНАЯ КОМПАНИЯ С WEB-ПРИЛОЖЕНИЯМИ ОРИЕНТИРОВАННЫМИ НА ВЗАИМОДЕЙСТВИЕ С ВНЕШНИМИ ПОЛЬЗОВАТЕЛЯМИ.

В первом сценарии мы рассмотрим крупную компанию с несколькими ориентированными на внешних пользователей web-приложениями. Они разрабатывались для того чтобы стать единой точкой взаимодействия с клиентами, сотрудниками и бизнес-партнерами. Основными драйверами безопасности в данном случае являются: противодействие мошенничеству, соответствие требованиям регуляторов и обеспечение отказоустойчивости для обеспечения непрерывности бизнес-процессов. Вторичные драйверы: сохранение репутации, готовность к атакам, сохранение активов и данных, а также снижение расходов на содержание. Главный вопрос здесь не в необходимости обеспечения безопасности, а в том какими средствами этого добиваться. Большая часть корпоративных приложений имеют серьезные недостатки в коде и, если быть полностью честными, то основные задачи в данном сценарии в том, чтобы минимизировать риски, которые несут эти недостатки. Ни один продукт “с прилавка”, разработанный сторонними специалистами, не сможет волшебным образом решить все имеющиеся проблемы безопасности, поэтому вам необходимо инвестировать не только, но и в совершенствование собственные процессы разработки, чтобы с каждым новым релизом приложения становились все лучше с точки зрения безопасности.

Предположим, что наша выдуманная крупная компания имеет существующую программу безопасности, команду разработчиков с некоторой степенью зрелости в понимании вопросов безопасности, но вопрос решения проблем даже в такой ситуации не бесспорен. У компании уже может быть специалист по безопасности в штате, но в процессе разработке еще не выстроены процессы оценки безопасности и идентификации проблем. Типичный специалист по безопасности чаще всего исходит из своего опыта защиты сетей и не имеет опыта безопасной разработки и не уделяет особого внимания коду web-приложений. Будем считать, что программа безопасности включает использование инструментов по оценке уязвимости, и они проверяют код на базовые атаки с использованием SQL-инъекций и атак на переполнение буфера. В целом безопасность обеспечивается силами пары сторонних продуктов и усилиями специалиста по безопасности,

указывающего на недостатки в безопасности, которые исправляются в будущих обновлениях.

Рекомендации

Стратегия в данном случае сводится к необходимости включения вопросов безопасности на уровне разработки, сместив акцент с внешних продуктов на внутренние продукты и обучение персонала. Инструменты выбираются и приобретаются для решения конкретных недостатков в навыках команды разработчиков и организационных процессах. WAF используется для того, чтобы обеспечивать безопасность на время разработки обновлении, закрывающих выявленные уязвимости.

Обучение и отладка процессов. Область, которая даст максимальный эффект в виде повышения уровня безопасности – это улучшение знаний и навыков команды разработчиков. Главные недостатки, отмечаемые OWASP, а также другими источниками, указывают на проблемы, которые могут быть решены путем надлежащего написания кода и тестирования... но только в том случае, если команда разработчиков знает что и как нужно делать. Обучение помогает разработчикам выявлять ошибки и проблемы в коде и эффективно снижает количество недостатков в каждом последующем цикле разработки. В развитии персонала можно сосредоточиться на повышении навыков всех разработчиков, а не полагаться на одного или двух выделенных специалистов, которые будут сосредоточены на анализе безопасности.

Безопасный цикл разработки приложений. Должен стать одним из приоритетных требований при разработке приложения, включая в себя определенные требования к коду, проверке и тестированию на разных этапах разработки web-приложения. В противном случае добиться должного уровня будет невозможно, так как все силы будут брошены на доработку возможностей и функционала. Безопасность должны быть частью спецификации продукта и требований к нему, а каждая фаза разработки должны завершаться проверкой на соблюдение этих требований и соответствие спецификации. Это означает, что тесты на безопасности в процессе разработки должно выполняться так же, как и тестирование перед выпуском релиза. Обученные разработчики вполне могут обеспечить анализ кода и разработать тестовые сценарии для проверки, но дополнительные инструменты для автоматизации процессов тестирования и задач по проверке все же необходимы.

Наследуемый код приложения. Есть решение проблемы наследуемого кода. Одной из серьезных проблем – это наследуемый код, который, вполне вероятно, имеет проблемы с безопасностью. Есть несколько вариантов решения этой проблемы, но основными шагами в любом случае будут: 1) Выявление недостатков и проблем в коде (сканирование кода, оценка уязвимости и тестирование на проникновение), 2) Расстановка приоритетов по исправлению выявленных недостатков, 3) Планирование по устранению каждой найденной уязвимости. Общие методы исправления уязвимостей включают: 1) Переписывание сегментов кода, 2) методы инкапсуляции (например, интерфейс), 3) Дополнение существующего кода путем создания процедур проверки путей/методов во время исполнения, 4) временное защита при помощи настройки политик WAF, 5) Перемещение SQL-процессов и проверок в базу данных, и 6) прекращение использования небезопасных функций. Мы рекомендуем использовать статические или динамические инструменты для первоначальной проверки кода. Эти инструменты являются экономически целесообразными, подходят для сканирования больших объемов кода и позволяют достаточно эффективно выявить ошибки. Они обнаруживают и приоретизируют недостатки, позволяя избежать пользовательских ошибок, которые неизбежны при ручном сканировании. Инструменты для анализа также позволяют персоналу получить некоторые дополнительные знания об уязвимостях, в том числе с примерами на разных языках. Полученные в результате

аргументы о 16 тысячах небезопасных вхождений IFRAME – это, конечно, не весело, но придется принять и понять проблему, прежде чем она будет решена.

Внешние проверки. Периодические внешние проверки – оценка уязвимости, тестирование на проникновение и проверка исходного кода – настоятельно рекомендуются. Опытные непредвзятые специалисты, имеющие опыт в анализе угроз, могут выявить те недостатки, которые были пропущены во время внутренних сканирований, а также могут помочь в обучении разработчиков иным векторам атак. Запланируйте внешнее тестирование на проникновение один раз в квартал или полугодие – знания и опыт хороших консультантов выходит далеко за рамки основных угроз, а потому их работа с приложением поможет выявить даже неочевидные уязвимости, которые могут быть использованы злоумышленниками. Мы также рекомендуем использовать инструменты статического анализа для внутренней проверки кода в рамках прохождения тестов контроля качества и внутренним тестированием на проникновение непосредственно перед развертыванием – это позволит провести стресс-тесты приложения без страха вывести из строя "боевое" приложение. Крупные обновления также должны пройти внешний тест на проникновение до того как будут установлены.

Блокирование. Это одна из областей, необходимость в которой напрямую зависит от специфики вашей организации. В случае с корпоративным приложением в использовании WAF является очень важной частью системы безопасности. Он обеспечивает базовую защиту и дает сотрудникам возможность спокойно и без паники разбираться с выявленными уязвимостями в действующей системе. Конечно, возможен вариант, что объем кода вашего приложения не очень велик и особой потребности в WAF нет, но для больших компаний WAF это не опция, а обязательная части системы защиты web-приложения. Циклы разработки и обновления приложения в таких компаниях слишком долгие, чтобы можно было обеспечить быструю реакцию на найденную новую уязвимость. Мы рекомендуем использовать гибридные модели, которые обеспечивают функции WAF и оценки уязвимости, потому что такая комбинация позволяет значительно облегчить разработку и внедрение новых политик для WAF. WAF – это недешево, поэтому мы не можем рекомендовать его использование всем. Но все же, именно этот инструмент обладает гибкостью и позволяет бороться не только с существующими, но и будущими угрозами.

В общем, мы рекомендуем вам принять меры для повышения уровня безопасности в каждой части процесса разработки и развития приложения. Фокусировка на совершенствовании начальных этапов разработки приложения (выработка требований, архитектура, дизайн) принесут наибольшее количество пользы с точки зрения безопасности. Для приложений, которые уже эксплуатируются, мы рекомендуем введение внешних проверок и, если бюджет позволяет, использование WAF. Оценка уязвимости и тестирование на проникновение имеют наиважнейшее значение, независимо от того, новое это приложение или нет, а WAF очень важен для быстрой реакции на выявление новой уязвимости. Учитывая, что риски крупных предприятий выше, а решение любой проблемы сложнее, важно понимать, что и инвестиции в безопасность должны быть серьезными. Требования безопасности, а также их изменения должны быть официально зафиксированы для каждой стадии жизненного цикла приложения – а проверка выполнения этих требований проводится на каждом значимом этапе.

КОМПАНИЯ СРЕДНЕГО РАЗМЕРА И СООТВЕТСТВИЕ ТРЕБОВАНИЯМ PCI

Если мы говорим о безопасности web-приложений и наличии каких-либо требований, то стоит обратиться к Payment Card Industry's Data Security Standard (PCI-DSS). Фактически это единственный стандарт, определяющий специфические требования к безопасности web-приложений. Мы, конечно, можем ворчать о неоднозначности этих требований и способах их улучшить, но не

можем не признать, что PCI в данный момент является главным драйвером для безопасности web-приложений на сегодняшний день. Поэтому в качестве второго сценария мы взяли компанию средних размеров, которой необходимо обеспечить соответствие их web-приложения требованиям данного стандарта.

Профиль этой нашей компании – торговля. Большая часть прибыли генерируется посредством онлайн-продаж. Коммерческий web-сайт относительно новый (менее 3 лет), а команда разработчиков достаточно мала и не имеет достаточно специфических навыков в информационной безопасности. То есть понимание ньюансов хакерских атак не является их сильной стороной. Соответствие PCI является основной задачей, хотя понятно, что кроме выполнения основных требований есть необходимость и в защите от некоторых видов атак. Позитивные моменты: объем кода web-приложения невелик и, так как большая часть прибыли генерируется именно посредством web-приложения, руководство всячески поддерживает инициативы по повышению уровня защищенности.

В двух словах, стандарт PCI-DSS – это стандарт безопасности для компаний, обрабатывающих транзакции платежных карт в рамках интернет-коммерции. С точки зрения правил безопасности – это самые однозначные и понятные требования, включающие специфические требования к средствам обеспечения безопасности и процессам работы с данными платежных карт. При этом имеется у стандарта и определенная гибкость, которая выражается в необходимости соответствия духу требований. То есть, если ваши решения и отличаются от рекомендованных PCI-DSS, но они обеспечивают должный уровень безопасности, то они имеют право на одобрение. Мы сосредоточимся на выполнении требований указанных в пунктах 6.6 и 11.3, но также будет обращаться к разделу 10 и компенсирующим элементам управления.

Рекомендации

Наша стратегия будет сфокусирована на образовании и изменении процессов разработки для того чтобы включить безопасность в цикл разработки. Дополнительно мы предлагаем использовать услугу тестирования на проникновение, что позволит быстрее выявить проблемные области web-приложения. Для начала сфокусируйтесь на выполнении требований, а уже затем можно подумать о вариантах развития защиты на случай роста компании. Не бойтесь пользоваться сторонней помощью для решения разных задач, это касается и выполнения требований PCI, и общих вопросов безопасности приложений.

Тренинги, образование и улучшение процессов. Опять же, максимальный упор мы делаем на образование и тренинги для персонала, в том числе для менеджмента проекта. Несмотря на то, что это требует достаточно солидных временных затрат, он позволяет ощутимо повысить уровень безопасности и является эффективным с экономической точки зрения (за счет улучшения качества кода, что снижает затраты на доработку в перспективе). Правильно написанный код не требует исправлений в дальнейшем. Кроме того, процесс улучшения кода будет постоянным, что позитивно скажется не только на безопасности, но и на качестве приложения в целом. В общем, если кода не много, то именно образование и тренинги являются лучшим путем для относительно небольшой компании и принесут максимум преимуществ.

Внешняя помощь. Подружитесь с аудитором или наймите его в качестве консультанта, который поможет подготовить и провести проверку на соответствие PCI-DSS. Аудитор не просто даст специфические рекомендации по отдельным требованиям PCI, он предоставит экспертную оценку, поможет в толковании некоторых неоднозначных моментов, окажет содействие в разработке стратегии и убережет от необдуманных и неэффективных трат.

Раздел 11.3.2. 11.3 Тестирование на проникновение сети и web-приложения. В данном сценарии мы рекомендуем внешнее тестирование на проникновение – не только потому, что это требование стандарта DSS, но также потому, что независимый эксперт изучит ваше приложение позиции максимально близкой к хакерской. Тем более в дальнейшем ограниченный бюджет наверняка заставит выбирать между WAF и анализом кода в пункте 6.6, и тестирование на проникновение поможет сделать вам правильный выбор. Если же анализ кода уже проводится, то можно утверждать (возможно, безосновательно), что он является некой компенсирующей мерой, позволяющей выполнить требования данного раздела, но мы все равно рекомендуем использовать тестирование на проникновение. Внешнее тестирование дает гораздо больше, чем список специфических уязвимостей, предлагая также идентификацию рисков и выявление сомнительного поведения приложения.

Раздел 6.6. Наши самые большие споры касаются рекомендации использования WAF или анализа кода для удовлетворения требований PCI, указанных в разделе 6.6. Фактически стандарт рекомендует использовать и то и другое, но широко известно, что это весьма дорого. Покупка WAF – это быстрый путь удовлетворить требования раздела 6.6, обеспечив базовый мониторинг и базовую платформу для блокирования атак в будущем. Но контраргументов у этого варианта достаточно – это и уже упомянутая и цена и достаточно сложную и кропотливую работу по настройке политик. С другой стороны, анализ кода квалифицированным специалистом по безопасности поможет выявить слабые места в коде приложения, а также помочь в обучении команды разработчиков указывая на конкретные недостатки. Внешний анализ позволяет быстро оценить ситуацию, в которой вы находитесь, и понять что вам нужно для того, чтобы эту ситуацию улучшить. К недостаткам анализа кода можно отнести стоимость работ, время необходимое для выявления и исправления недостатков, а кроме того, постоянную изменяемость кода, которая требует повторных проверок.

И все же мы рекомендуем выбрать WAF. Привлечение команды профессионалов по безопасности для анализа кода – это, конечно, эффективный путь выявления недостатков, но его ценность заметно снижается, ведь во многом схожие данные дают периодические тесты на проникновение, необходимость проведения которых, как мы помним, отражена в разделе 11.3.2. Кроме того, даже при относительно небольшом или среднем количестве кода, время на коррекцию всех недостатков может оказаться достаточно большим, что не позволит пройти аттестацию на соответствие требованиям PCI в достаточно сжатые сроки. Обратите внимание, что данная рекомендация действительна только для данного сценария. При других исходных данных, к примеру, при наличии более опытного персонала и изначально лучшего качества кода, мы могли пойти совсем по другому пути для достижения необходимого результата.

Мониторинг. Мониторинг активности базы данных (DAM) – это хороший выбор для удовлетворения требований раздела 10, в соответствии с которым необходимо обеспечить полный мониторинг доступа к данным платежных карт. Web-приложения используют реляционную базу данных для хранения номеров платежных карт, транзакции и другие связанные с ними данные. Продукты класса DAM охватывают всю сетевую и консольную активность на платформе базы данных и обеспечивают целенаправленный и экономически эффективный контроль на всех уровня доступа к данным кредитных карт. Рассмотрите именно этот вариант, который будет хорош как для аудиторов, так и для специалистов по безопасности.

РАЗРАБОТКА ВНУТРЕННЕГО WEB-ПРИЛОЖЕНИЯ

В нашем последнем сценарии мы рассмотрим внутреннее web-приложение, предназначенное для партнеров и сотрудников в рамках среднего или крупного бизнеса. Хотя это и не выглядит как

серьезная проблема, но даже для компании со штатом в 100 человек, внутреннее web-приложение может быть весьма критичной точкой с точки зрения бизнеса. Используя данные рабочего процесса, HR, бухгалтерского учета, продаж и других критичных IT-систем, эти приложения обеспечивают эффективную поддержку сотрудников и партнеров компании. Учитывая доступность из web-приложения почти всех данных компании, обеспечение их конфиденциальности и целостности является очень важной задачей. Общее мнение относительно защиты таких систем чаще всего сводятся к тому, что они находятся внутри периметра, а потому не подвержены типичным атакам, но это мнение довольно части приводит к катастрофе. Ведь в том случае, если злоумышленник преодолеет внешний барьер (или окажется инсайдером), то приложение будет практически беззащитным. То есть, не стоит полагаться лишь на сетевую безопасность, необходимо обеспечить хотя бы базовый уровень защиты самого web-приложения и контроль доступа.

Рекомендации

Определить базовый уровень защиты приложения, исправить серьезные уязвимости, максимально использовать преимущества обучения, тренингов и усовершенствования процесса написания кода, для того чтобы повысить его качество.

Оценка уязвимости и тестирование на проникновение. Сканирование web-приложений для оценки уровня защищенности, проверки установки обновлений и недостатков в конфигурации – это первый из рекомендованных шагов, который позволит выявить основные уязвимости. Инструменты оценки защищенности являются самым экономически эффективным путем для определения базовой безопасности и вывода ее на приемлемый уровень. Тесты на проникновение, традиционно могут помочь в оценке потенциальных рисков и расстановке приоритетов при исправлении выявленных уязвимостей.

Образование тренинги и совершенствование процессов. В данном сценарии эти возможности станут, пожалуй, даже более актуальными, чем в других, хотя бы потому, что бизнесу труднее найти обоснования для инвестиций в безопасность приложения, чем в случае с внешними приложениями.

Мониторинг. Контроль и выявление подозрительной активности – это доступный, но эффективный способ выявления проблем. Мы считаем, что WAF в таком случае будет слишком дорогим решением, а потому настоятельно рекомендуем более экономически эффективный способ сбора и анализа активности. Программное обеспечение для мониторинга, которое подключается к web-приложению, зачастую может быть очень эффективным и стоить вполне разумных денег, но бремя анализа всех собираемых данных в данном случае ляжет на вашу команду разработчиков. Мониторинг активности баз данных может быть эффективен для защиты критической информации в бэк-енде и это более разумный выбор, нежели WAM.

ВЫВОДЫ

Наши рекомендации на самом деле весьма специфичны и сильно зависят от особенностей организации и ситуации с web-приложениями. Мы выбрали сценарии для того чтобы продемонстрировать мотивирующие факторы в сочетании с текущим состоянием защищенности web-приложения и руководством по выбору инструментов услуг и изменением внутренних процессов. Но так как приложения поддерживают значительное количество бизнес-функций, процесс по повышению уровню защищенности должен проходить при содействии и максимальном внимании со стороны не только технического персонала, но и ответственных за бизнес-процессов,

а также менеджера проектов. Это, без сомнения осложнит процесс, но движение вперед должно происходить только при понимании и поддержке всех заинтересованных лиц.

В любом случае мы должны помнить, что безопасность это часть общего процесса развития, и наша главная рекомендация для любого сценария – это стремится обеспечить максимальную экономическую эффективность и стараться минимизировать риски сбоя или любого отклонения от нормальной работы. Также важно помнить, что преобразование web-приложения не произойдет в одночасье. Мы не можем себе позволить такую роскошь, как остановка всех обновлений в ожидании момента, когда команда разработчиков исправит все недостатки в безопасности, а потому не нужно забывать, что сторонние продукты и услуги могут оказать неоценимую помощь для решения насущных проблем.

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