Тестирование уязвимостей межсайтового выполнения сценариев (XSS) в веб-приложении онлайн-платежей

А.С. Меркулов, О.Р. Лапонина

***Аннотация* – объектом исследования является веб-приложение онлайн-платежей компании «Payture», которая сотрудничает с крупными компаниями и банками. Компании «Payture» выступает в качестве платежного шлюза между ТСП, банками и платежными системами, предлагая гибкое интеграционное API.**

В работе анализируются способы тестирования веб-приложения онлайн-платежей «Payture» на наличие известных векторов атак межсайтового скриптинга. Определяется рекомендуемый список мер защиты как для данного приложения, так и для аналогичных приложений.

Рассмотрены типы XSS-атак, что позволяет спроектировать проведение атаки и проанализировать элементы веб-приложения, в которых могут быть скрыты XSS-уязвимости.

Описываются бесплатные и коммерческие программные продукты, которые позволяет производить комплексный или частичный анализ веб-приложений на предмет наличия XSS-уязвимостей.

Рассматриваются основные возможности API компании «Payture», определяется последовательность действий для поиска XSS-уязвимостей.

Сделаны предложения по совместному использованию нескольких инструментов анализа XSS-уязвимостей в веб-приложениях.

***Ключевые слова* – XSS-уязвимости, веб-приложение онлайн-платежей «Payture», NetSparker, OWASP Xenotix XSS, CSP Evaluator, Wapiti3.**

I. Введение

В настоящее время все больше людей покупают товары в интернет-магазинах, а также используют различные электронные платежные инструменты.

Уязвимости веб-приложений постоянно изучаются злоумышленниками, целью которых может быть, например, кража важной личной информации держателей пластиковых карт, пользователей электронных кошельков и финансовых транзакций коммерческих и государственных компаний. Утечка данных о клиентах, пластиковых картах, финансовых транзакциях и личных данных обычно приводит к множеству негативных отзывов о компании, за которыми следуют судебные санкции и санкции властей, что может привести к финансовым потерям и даже к закрытию компании.

XSS-атака (Cross Site Scripting), которую также называют «межсайтовое выполнение сценариев», это вид уязвимости компьютерной безопасности, часто встречающийся в веб-приложениях. В последние годы пятерка самых распространенных атак включала атаку с использованием межсайтовых сценариев, на которую приходится почти треть общего числа атак. Как следствие, анализу XSS-атак в веб-приложениях для онлайн-платежей уделяется большое внимание исследовательскими фондами и компаниями, которые стремятся обеспечить защиту, безопасность, целостность и простоту услуг онлайн-платежей. В одном из самых авторитетных рейтингов, «Топ-10 OWASP», который был создан Фондом OWASP, также большое внимание уделяется XSS-угрозам.

Имеется достаточно большое количество инструментальных средств проверки наличия данной уязвимости в веб-приложении. Наиболее обсуждаемыми и используемыми как разработчиками приложений, так и тестировщиками являются NetSparker, OWASP Xenotix XSS, CSP Evaluator, Wapiti3.

Объектом данного исследования является веб-приложение онлайн-платежей одного из лидеров на рынке платежных шлюзов в секторе электронной коммерции на возможность выполнения XSS-атак, также в работе анализируются инструментальные средства, позволяющие находить XSS-уязвимости. Так как на данный момент большинство приложений подобного рода имеет достаточно сложную архитектуру, сложную программную реализацию и гибкий инструментарий для пользователей, это увеличивает количество различных вариантов настроек и конфигураций веб-приложения. Такое разнообразие возможностей имеет и обратную сторону, когда при определенного рода комбинациях злоумышленник может выявить слабые места в системе.

II. Постановка задачи

Цель данной работы состояла в исследовании принципов межсайтового скриптинга, изучении методик проведения векторов данных атак, выполнение тестирования на возможность проведения XSS-атак на веб-приложение онлайн-

Статья получена 03 сентября 2019.
А.С. Меркулов – МГУ имени М.В. Ломоносова (email: merkulov.artem.s@gmail.com).
О.Р. Лапонина – МГУ имени М.В. Ломоносова (email: laponina@oit.cmc.msu.ru).

платежей компании «Payture», а также определение рекомендуемых инструментов тестирования межсайтового скриптинга в веб-приложениях онлайн-платежей.

III. АТАКИ МЕЖСАЙТОВОГО СКРИПТИНГА (XSS)

Данная уязвимость позволяет злоумышленникам внедрять сценарии в веб-страницы, которые в дальнейшем будут получены другими пользователями, в контексте которых будут выполнены внедренные сценарии. Данная уязвимость может использоваться злоумышленниками, например, для обхода средств контроля доступа. Эффективность атак варьируется в диапазоне от мелких неприятностей до значительного риска безопасности, в зависимости от чувствительности данных, обрабатываемых уязвимым сайтом. В частности, на странице или в HTTP-cookie могут быть весьма уязвимые данные (например, идентификатор сессии или номера платежных документов), а там, где нет защиты от CSRF-атаки, атакующий может выполнить любые действия, доступные пользователю.

IV. ТИПЫ XSS

Типы XSS-атак можно классифицировать по векторам атак, по каналам внедрения скрипта и по способу воздействия. Нас в первую очередь будут интересовать XSS-атаки, различающиеся по вектору атак.

A. *Постоянный (хранимый) XSS*

Постоянный или хранимый XSS является наиболее опасной атакой. Данная атака возможна, когда злоумышленнику удается внедрить на сервер вредоносный код, который затем будет выполняться в браузере при обращении жертвы к уязвимой странице.

Рисунок 1. Схема атаки при хранимом XSS

B. *Непостоянный (отраженный) XSS*

Отражённая XSS-атака возникает, когда пользователь переходит по специально подготовленной ссылке. В этом случае данные, предоставленные жертвой чаще всего в параметрах HTTP-запроса или в HTML-форме, исполняются на стороне сервера, что может привести пересылке данных пользователя злоумышленнику, если не была выполнена необходимая проверка.

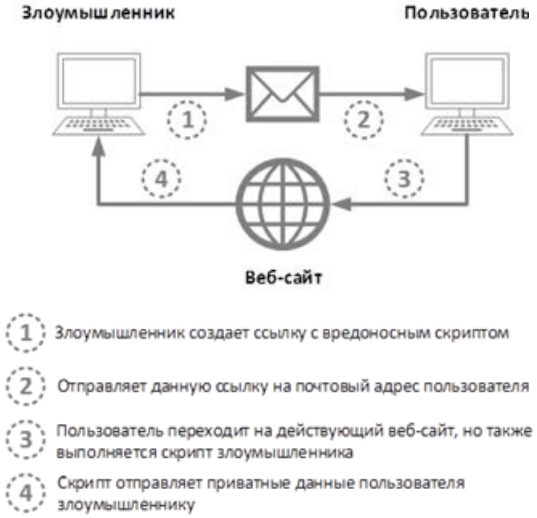

Рисунок 2. Схема атаки при отраженном XSS

C. *XSS с модификацией DOM*

Данная XSS-атака возникает на стороне клиента во время выполнения вредоносного JavaScript-сценария, который изменяет DOM HTML-страницы. При отсутствии достаточной фильтрации можно посредством модификации DOM добиться получения доступа JavaScript-кода к данным, которые находятся в контексте жертвы.

V. ОБЗОР СУЩЕСТВУЮЩИХ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПОИСКА XSS-УЯЗВИМОСТЕЙ

Мы использовали наиболее известные инструменты анализа веб-приложений на наличие XSS-уязвимостей.

1) NetSparker

Один из самых мощных на сегодняшний день инструментов. На рынке является практически стандартом и входит в список рекомендованных инструментов по обеспечению безопасности открытого проекта обеспечения безопасности веб-приложений OWASP. Поддерживает параллельный поиск уязвимостей разного типа. Данный продукт имеет достаточно понятный интерфейс.

В качестве основного недостатка следует отметить, что данный продукт не дает возможности предварительно авторизоваться в веб-приложении. Таким образом, потенциально опасные формы, доступные авторизованным пользователям, не могут быть проверены. Немаловажным является и тот факт,

что бесплатная версия дает весьма ограниченный функционал, кроме того, доступна она всего лишь в течение двух недель. А цена в 3500$ в год может быть по карману далеко не каждой компании.

2) OWASP Xenotix XSS

OWASP Xenotix XSS Exploit Framework включает большое количество скриптов, а также дополнительных утилит для полноценного анализа сайта на наличие XSS-уязвимостей и проведения атак. Особенностью данного ПО является наличие трех окон для различных браузеров (IE, FireFox, Chrome), чтобы выявить атаки, которые по той или иной причине могут сработать только в определенном браузере, так как эти браузеры используют разные движки (IE – Trident, Firefox – Webkit, Firefox - Gecko).

Данное программное обеспечение распространяется бесплатно и дает возможность проанализировать приложение на большое количество атак.

3) CSP Evaluator

CSP Evaluator позволяет проверять, является ли политика безопасности контента (CSP) достаточной для защиты от XSS-атак. Данное инструментальное средство помогает в процессе анализа политик CSP, что обычно выполняется вручную, в частности, помогает определить возможные способы обхода CSP. Проверки CSP Evaluator направлены на то, чтобы помочь разработчикам усилить свои CSP, тем самым повысив безопасность своих приложений. Данное приложение разрабатывалось специалистами компании Google и раньше использовалось только внутри компании. На данный момент оно доступно для всех пользователей.

4) Wapiti3

Wapiti3 позволяет проводить аудит безопасности веб-приложений. Сканирование выполняется по принципу «черного ящика» (не изучает исходный код веб-приложения), просматриваются веб-страницы развернутого веб-приложения, ищутся сценарии и формы, в которые можно вводить данные. Как только Wapiti3 получает список URL, форм и их входных значений, он проверяет допустимые диапазоны значений каждого входного параметра, вставляя данные, тем самым определяя, уязвимо ли приложение.

Для выполнения атак Wapiti3 использует HTTP-методы GET и POST. Wapiti3 также может изменять данные в именах файлов, проверяя тем самым корректность загрузок на сервер. Предупреждения выдаются при обнаружении какой-либо аномалии. Wapiti3 способен отличить постоянную и отраженную XSS-уязвимости.

VI. Описание API сервиса онлайн-платежей компании «Payture»

Программное решение компании «Payture» предоставляет различные способы интеграции и использования платежных шлюзов.

Различные схемы интеграции имеют свои отличительные особенности. Мы рассмотрим один из наиболее часто используемых способов интеграции между покупателем, ТСП (Торгово-сервисным предприятием) и Payture, который является платежным шлюзом. Данный способ интеграции чаще всего используют небольшие компании.

Программный интерфейс Payture InPay предназначен для ТСП, заинтересованных в доступе к удобному и гибкому решению для автоматизации электронных платежей. По своей функциональности он аналогичен Payture API. Основное его отличие от Payture API – данные банковской карты покупателя вводятся не на веб-странице продавца, а на странице защищенного шлюза Payture. Таким образом, при использовании Payture InPay продавец избавлен от необходимости самостоятельно обеспечивать безопасность платежной информации покупателя – данную защиту обеспечивает Payture.

Интерфейс Payture InPay позволяет выполнять стандартный набор операций электронной коммерции с вводом данных банковских карт на странице шлюза Payture.

Протокол обмена информацией с сервером происходит по схеме запрос-ответ.

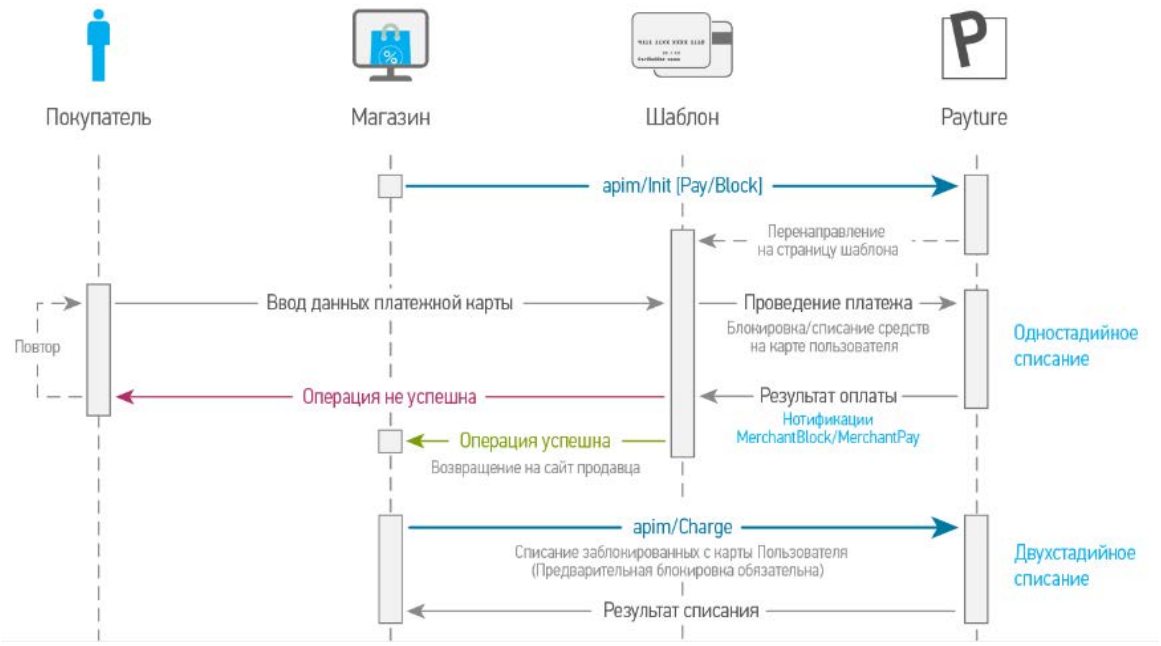

Рисунок 3. Схема обработки запросов Payture

1) Инициализация запроса на оплату (Init)

Выполняется обязательно перед операцией списания средств или блокировки средств на карте покупателя при двухстадийном платеже. Результатом исполнения запроса является подготовка к перенаправлению пользователя на страницу платежного шлюза Payture для ввода данных банковской карты.

2) Блокировка или списание средств (Block/Pay)

Запрос списания/блокировки средств с перенаправлением покупателя на страницу платежного шлюза. Выполняется после успешной команды Init. Результатом обработки запроса является списание денежных средств (при одностадийной схеме проведения платежа), либо блокировка средств на карте Покупателя (при двухстадийной схеме проведения платежа). Списанные средства могут быть (полностью или частично) возвращены на карту Покупателя при помощи команды Refund.

После успешной или неуспешной оплаты выполняется перенаправление (redirect) на стандартный шаблон статусы оплаты. Также возможны опции настройки переадресации на страницу магазина.

3) Списание после блокировки (Charge)

Запрос используется для списания денежных средств с карты покупателя, предварительно заблокированных командой Pay. Выполняется в рамках двухстадийной схемы проведения платежа, после успешной команды Init с ключом «SessionType=Block». Результатом обработки запроса является списание с карты покупателя суммы, не превышающей заблокированной (равной или меньшей).

4) Разблокировка (Unblock)

Запрос позволяет полностью снять блокирование денежных средств, ранее заблокированных на карте покупателя командой Pay. Выполняется в рамках двухстадийной схемы проведения платежа. Результатом обработки запроса является полная разблокировка суммы, заблокированной на карте покупателя.

5) Полный или частичный возврат (Refund)

Этот запрос используется для возврата денежных средств, списанных командой Pay (при одностадийной схеме проведения платежа) или Charge (при двухстадийной схеме).

Результатом обработки запроса является полный или частичный возврат списанных денежных средств на карту покупателя. Для успешного возврата необходимо, чтобы на момент исполнения запроса платеж имел статус Charged.

6) Статус операции (PayStatus)

Запрос используется для получения информации о текущем состоянии платежа. Рекомендуется также использовать его в случае неполучения ответа от шлюза Payture, при проведении других запросов по платежу. Выполняется как при одностадийной, так и при двухстадийной схемы проведения платежа. Результатом обработки запроса является получение актуального статуса платежа.

Важно отметить, что передача любых параметров происходит со стороны ТСП, что делает невозможным для непосредственного плательщика каким-либо образом исказить или подменить передаваемые данные в запросе. Атака возможна

только в случае наличия уязвимостей на стороне ТСП, который предоставляет своим клиентам сервисы оплаты. Злоумышленнику требуется выявить уязвимость на стороне ТСП, с использованием которой он сможет повлиять на запрос и передать вредоносные данные на сторону платежного шлюза.

Также важно отметить, что большинство используемых полей в запросах, а именно Key – ключ терминала, SessionType – режим стадийной оплаты, Amount – сумма оплаты, имеют четко установленные требования. Например, Key выдается сотрудником платежного шлюза и является уникальным значением, SessionType является четко фиксированной меткой стадийности оплаты. Но кроме этого каждая транзакция имеет один дополнительный параметр OrderId – уникальный код операции, который фактически является ключом операции и не имеет ограничений по своей спецификации и требованиям. Плюс к этому негласным правилом платежных сервисов является включать по желанию ТСП дополнительный блок параметров в запросе – CustomFields, в котором могут передавать любые требуемые параметры в формате Key=Value для дальнейшего использования на стороне ТСП.

VII. Поиск уязвимостей

Мы будем искать XSS-уязвимости, взаимодействую с сервером с помощью стандартных команд Payture. Мы будем использовать несколько различных инструментов поиска уязвимостей, которые в совокупности должны нам предоставить целостную картину наличия уязвимостей.

A. Инициализация запроса на оплату

Мы выполним стандартную последовательность запросов для проведения оплаты. Результат выполнения каждой команды напрямую влияет на последовательность и логику выполнения последующего запроса. Также при этом важной особенностью таких цепочек запросов является то, что параметры, передаваемые при инициализации (Init), используются в последующих командах. В рамках первой команды (Init) создается сессия, к которой привязываются все создаваемые параметры.

Мы будем выстраивать наши векторы атак по двум параметрам OrderId и CustomFields с учетом данных, полученных по результатам анализа документации, которая доступна в открытом виде и также может быть использована злоумышленниками.

Первым шагом мы выполним автоматическое тестирование с помощью программы Wapiti3, которая предлагает автоматические настройки для тестирования.

По результатам полностью автоматического анализа уязвимостей не найдено, что не удивительно, так как платежные сервисы имеют стойкую защиту и готовы к подобного рода атакам.

Мы постараемся зайти с другой стороны и выполним анализ в ручном режиме с использованием Xenotix XSS. Мы более гибко настроим передачу параметров, чтобы пройти первые этапы проверки запроса.

Выполним ручное тестирование с подменой выделенных параметров с помощь Xenotix XSS.

```
http://localhost:7080/apim/Init?Key=Artem_test&Data=SessionType=Pay;OrderId=%3Cscript%3Ealert(%27XSS%27);%3C/script%3E;Amount=100;CustomFields=Key1==%3Cscript%3Ealert(%27XSS%27);%3C/script%3E
```

Здесь мы получаем более интересный результат, а именно в полях OrderId и CustomFields можно вставить вредоносные команды, которые будут рассматриваться как ключ транзакции OrderId и дополнительные параметры CustomFields. Данные параметры проинициализированы на стороне платежного шлюза и сохраняются в БД.

Теперь для более многофакторного анализа мы выполним сканирование с помощью NetSparker, данная программа позволяет выполнять анализ в комбинированном режиме, т.е. у нас есть возможность не выполнять ручную подстановку, а позволить системе выделить ключевые уязвимости, при этом нам требуется установить фокусировку на ранее определенные параметры для атаки OrderId и CustomFields.

По результатам комбинированного анализа мы смогли выделить также два уязвимых параметр OrderId и CustomFields, что является хорошим результатом.

В результате инициализации платежный шлюз вернул нам параметр SessionId, с которым мы можем продолжить цепочку команд, и сохранил передаваемые нами данные в БД.

B. Ввод данных пластиковой карты и оплата

В рамках данного шага используется уже проинициализированная сессия цепочки транзакций. Здесь мы будем использовать уже полученный и активный идентификатор SessionId.

Так как ввод данных пластиковой карты подразумевает наличие веб-формы, мы выполним проверку с помощью инструмента CSP Evaluator, который позволяет проанализировать страницу на возможность взлома кликов и других атак внедрения кода, возникающих в результате выполнения вредоносного контента в контексте доверенной веб-страницы и корректности источника получения контента, из которого браузерам разрешено выполнять загрузку следующих элементов: JavaScript, CSS, фреймы HTML, шрифты, изображения, встраиваемые объекты, такие как Java-

апплеты, ActiveX, аудио- и видеофайлы, и другие элементы HTML5. Данный метод в большей мере ориентирован на разработчиков, так как он позволяет отслеживать возможные ошибки и уязвимости, которые разработчик может устранить на раннем этапе. Данное приложение является расширением для браузера, мы запустим его на странице ввода данных пластиковой карты.

По результатам данной проверки мы не обнаружили уязвимостей и слабостей системы, что является хорошим результатом для разработчиков, но не дает нам дополнительной информации для выполнения атаки.

Мы продолжим анализ в ручном режиме с использованием Xenotix XSS, при этом набор доступных параметров для запроса у нас ограничен. Любая подмена или дополнение параметра SessionId приводит к неуспешным результатам, что является правильным поведением системы в данной ситуации. Мы продолжим выполнение команды с использованием ранее полученного SessionId и постараемся перейти на страницу оплаты.

У нас существует возможность также проанализировать DOM-модель. По результатам данной проверки уязвимости не найдены.

Теперь выполним проверку защищенности полей ввода карточных данных и можем ли мы их использовать для внедрения вредоносных команд. Мы получаем, что поля PAN, CVV, Date, Card Holder не могут допустить ввод и передачу каких-либо невалидных или запрещенных данных.

Также выполняется проверка PAN по алгоритму Luna, что не позволяет нам указать хаотичный набор цифр. Это означает достаточно защищенную форму, что лишний раз подтверждает невозможность непосредственного плательщика повлиять или исказить значения передаваемых данных.

Теперь мы выполним анализ с помощью NetSparker.

По результатам мы видим, что уязвимостей в данной форме мы обнаружить не смогли.

Но в рамках данного шага мы смогли продвинуться вперед по цепочке команд и выполнить оплату. Мы помним, что на первом этапе мы смогли передать вредоносный код в полях OrderId и CustomFields, и теперь мы успешно выполнили команду Pay, в рамках которой были записаны вредоносные значения в БД.

При этом также хотелось бы отметить, что связка команд Pay и Block с последующим Charge являются равносильными по смыслу, и в рамках данного анализа мы рассмотрели более наглядный шаг оплаты через команду Pay напрямую, чтобы у нас была возможность сразу получить желаемый результат операции.

Теперь рассмотрим, каким образом мы сможем передать и использовать параметры операции с вредоносным кодом, который мы сохранили в БД платежного шлюза на этапе инициализации и подтвердили на этапе оплаты, на стороны ТСП и непосредственного плательщика для выполнения в вредоносного кода браузере.

C. Передача данных на стороны ТСП и клиента

В рамках данной команды мы можем запросить статус выполненной операции. Она используется для промежуточного уточнения статуса или для финального статуса транзакции. Также мы можем отслеживать промежуточные статусы на пути всей цепочки транзакций.

```
http://localhost:7080/apim/PayStatus?Key=Artem_test&OrderId=%3Cscript%3Ealert(%27XSS%27);%3C/script%3E
```

Самым интересным является то, что данная команда имеет достаточно широкий спектр возможностей, и мы можем либо просто запросить статус операции, либо с ранее настроенными параметрами выполнить редирект на другую страницу или на страницу магазина, в котором пользователь совершал оплату. В большинстве случаев выполняется именно такая переадресацию. К данной переадресации можно добавить ранее полученные параметры транзакции, чаще всего это как раз OrderId и CustomFields. Будем выполнять последовательный анализ с помощью инструментов сканирования.

В первую очередь выполним полностью автоматическое тестирования с помощью Wapiti3.

По результатам данного тестирования уязвимостей в системе не выявлено, также как и на предыдущих этапах проверки.

Теперь выполним анализ в ручном режиме с использованием Xenotix XSS. Будем выполнять поочередную подмену параметров и проверять какой результат мы получим. Здесь мы не можем использовать дополнительных параметров. Проверим единственную и множественную передачу параметров.

Мы видим, что система срабатывает корректно и отклоняет некорректные параметры, либо дополненные параметры в запросе.

Теперь выполним тестирование с помощью NetSpaker, который позволяет выполнять анализ в комбинированном режиме, при условии, что нам известны принципы работы API.

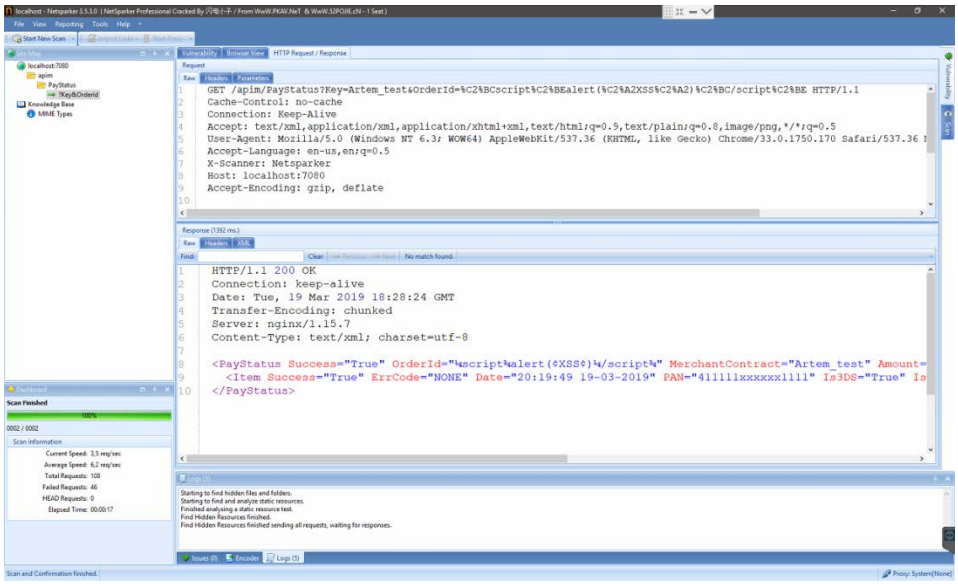

Рисунок 4. Комбинированное сканирование | команда PayStatus | NetSparker

В результате мы смогли вернуть ранее инициализированный параметр OrderId, в котором мы передавали вредоносный код на первом этапе инициализации. Данный параметр может быть передан на сторонний сайт или отображен в браузере пользователя. Данный вредоносный скрипт хранится на сервере платежного шлюза и может быть выполнен всякий раз при вызове метода статуса транзакции. Но в действительности данная возможность открывается злоумышленнику только одноразово, так как многие клиенты ТСП практически никогда не возвращаются к статусам своих оплат и не проверяют возвращаемую информацию после оплаты. В большинстве случаев клиенты проверяют чек на оплату, который отправляется им дополнительно на адрес электронной почту или в виде смс-сообщения на телефон.

VIII. Методы борьбы с XSS уязвимостями

В результате у нас получился достаточно интересный результат. Мы смогли передать вредоносный код с помощью двух параметров OrderId и CustomFields в рамках команды инициализации запроса на оплату, после мы подтвердили это оплатой через шаблон и данные сохранились в БД. После мы смогли выполнить проверку статуса операции с вызовом вредоносного кода, которую в дальнейшем передается на сторону ТСП или клиента, в зависимости от настроенной интеграции и параметров терминала.

По результатам выполнено тестирования мы можем составить таблицу в разрезе областей тестирования и используемых инструментов.

Таблица 1. Сводная таблица инструментов сканирования и их результативности

Тестируемая область	**Инструмент тестирования**			
	Wapiti3	NetSparker	CSP Evaluator	Xenotix XSS
Инициализация запроса на оплату	▼	▲	▬	▲
Блокировка или списание средств с пластиковой карты	▼	▼	▼	▼
Получение статуса транзакции и передача информации на сторону клиента или ТСП	▼	▲	▬	▲

▲ - уязвимости найдены
▼ - уязвимостей не найдено
▬ - нет требуемого функционала для анализа

NetSparker и Xenotix XSS показали лучший результат при анализе достаточно надежного платежного шлюза, определили наиболее опасные сценарии, которые могут быть использованы злоумышленниками.

Благодаря выполненному анализу мы можем определить перечень рекомендаций по защите на стороне сервере, который позволит минимизировать риски и обеспечить наиболее надежную защиту платежного шлюза.

Список рекомендаций:

1) Кодирование управляющих HTML-символов, JavaScript, CSS и URL перед отображением в браузере с использованием открытого проекта OWASP AntiSamy или кастомизированных реализаций и расширений.

2) Кодирование входных данных с помощью библиотек OWASP Encoding Project или кастомизированной реализации.

3) Указание кодировки на каждой веб-странице.

4) Обеспечение безопасности cookies, которая может быть реализована путём ограничения домена и пути для принимаемых cookies.

5) Использование TLS, обеспечивающего криптографически защищенную поддержку сессии на стороне сервера.

6) Использование заголовка CSP, позволяющего задавать список допустимых источников, с которых можно подгружать различные данные, такие как JS, CSS, изображения и пр. Политика описывается с помощью специальных директив, каждая из которых отвечает за отдельный вид ресурсов или policy area. Политика обязательно должна содержать директиву default-src, которая будет использоваться для тех ресурсов, для которых не будет описано отдельных правил (полный список можно найти в описании директивы default-src). Для того чтобы предотвратить исполнение встраиваемых скриптов и заблокировать использование eval(), необходимо определить директиву default-src или script-src. Также использование директивы default-src или style-src позволит ограничить использование встраиваемых стилей как в style-атрибутах, так и в тегах <style>.

7) Регулярный анализ всех компонентов приложения и взаимодействующих сервисов с помощью программных инструментов (NetSparker, OWASP Xenotix XSS, CSP Evaluator, Wapiti3 и другие).

IX. Прототип инструмента по анализу XSS уязвимостей

Предлагается прототип приложения, который позволяет аккумулировать различные сторонние библиотеки, так как различные реализации библиотек имеют индивидуальные механизмы анализа XSS уязвимостей.

При создании приложения использовался паттерн проектирования MVVM.

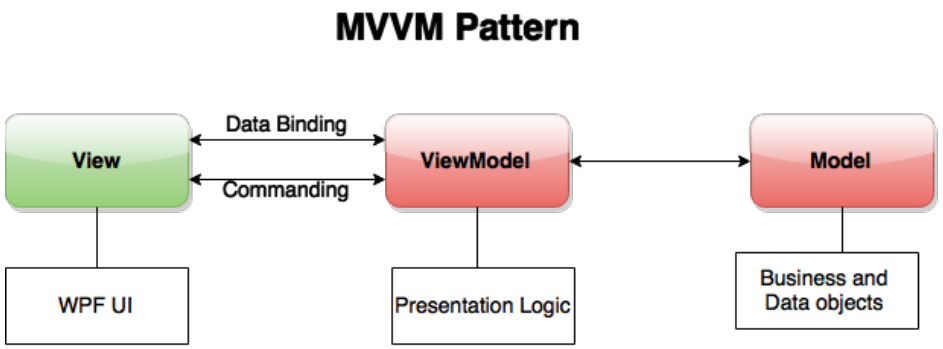

Рисунок 5. Паттерн MVVM

Данное приложение можно разделить на абстрактные модули, которые позволять нам создать универсальный каркас, основанный на технологии WPF.

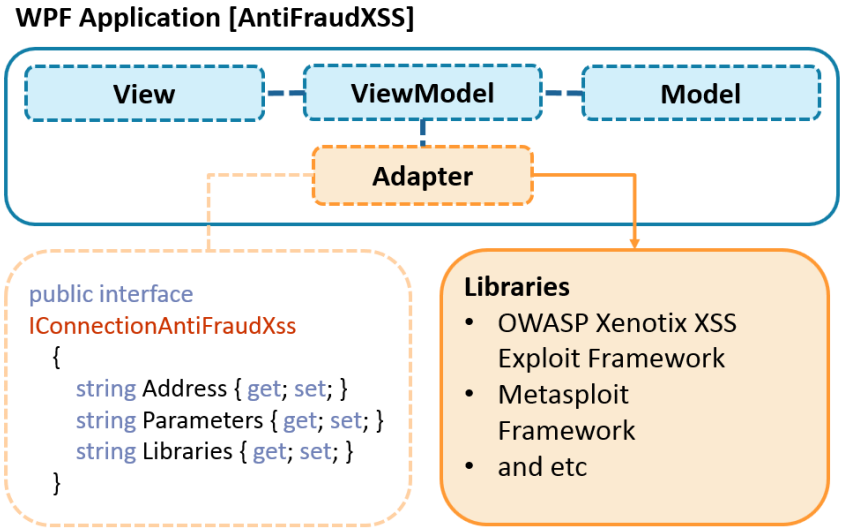

Рисунок 6. Прототип модели WPF Application [AntiFraudXSS]

Давайте подробнее рассмотрим компоненты приложения:

1) View – графическое представление, с которым взаимодействуют пользователь. Реализовано с помощью декларативного языка разметки XAML.

2) Model – описывает основную модель данных и сущностей приложения.

3) ViewModel – реализует основную логику работы приложения и позволяет взаимодействовать View и Model,

4) Adapter – компонент, который выступает в качестве связывающего механизма между основной частью приложения и интегрируемыми сторонними библиотеками.

При этом все сторонние библиотеки интегрируются через единый интерфейс, что позволяет нам единообразно работать с различными библиотеками, предоставляя пользователю простой и понятный интерфейс.

Графически интерфейс пользователя представляет простой и доступный набор настроек.

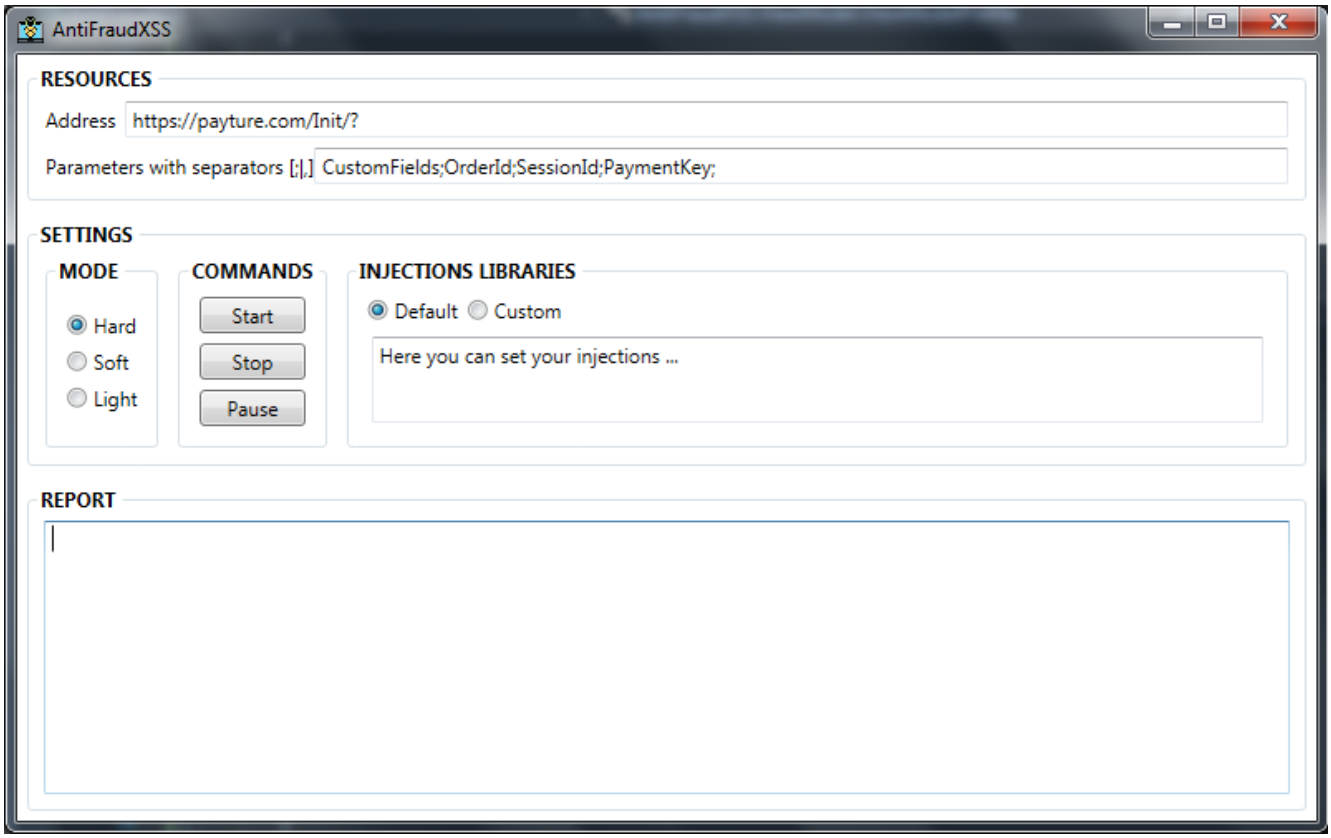

Рисунок 7. Графический интерфейс прототипа приложения

В результате мы сможем:

1) Создать гибкий и простой в использовании инструмент, который сможет анализировать приложения на предмет XSS уязвимостей.

2) Снимает с пользователя затраты по настройке нескольких инструментов.

3) Увеличивает вариативность атак в рамках анализа приложения, с применением различных механизмов и реализаций в рамках каждой библиотеки.

4) Позволяет пользователю запускать сценарий проверки по наиболее продуктивному пути, который мы смогли выявить и резюмировать в рамках текущей работы.

Программное решение размещено на внешнем ресурсе и доступно по ссылке:

https://drive.google.com/file/d/14kjesiN_m7glXTZlGhi16jyNoqPO06UX/view?usp=sharing_eil&invite=CKSE58QJ&ts=5cfbc4c9

X. Заключение

В рамках данной работы была исследована атака на веб-приложение и платежные шлюзы «Payture» типа Cross Site Scripting. Были изучены основные принципы ее работы и приведена классификация. Из различных источников был собран список XSS-инъекций, посредством которого было решено проверять веб-приложение на наличие XSS-уязвимостей. В рамках работы, были изучены существующие бесплатные и платные программные решения (NetSparker, OWASP Xenotix XSS, CSP Evaluator, Wapiti3), позволяющие производить автоматизированное, ручное и комбинированное тестирование безопасности приложения в контексте XSS-уязвимости. Выявлены основные достоинства и недостатки этих систем. С использованием тестовой среды приложения были выполнены наиболее вероятные сценарии атак и также комплексного анализа на наличие XSS-уязвимостей у веб-приложения, мы смогли передать вредоносный код на этапе инициализации запроса на оплату, на последующем шаге подтвердить транзакции с помощью оплаты пластиковой картой и выполнить сценарии передачи вредоносного параметра на сторону ТСП или клиента, при этом важно отметить что данная последовательность действий была осуществлена при первоначальном анализе принципа работы платежного шлюза и особенностей работы ТСП и платежного терминала. Проанализированы полученные результаты и сформирован список рекомендаций, который сможет обеспечить наиболее высокий уровень защиты электронно-финансовых операций.

Также был предложен прототип приложения, которое позволяет аккумулировать сторонние инструменты и библиотеки по анализу приложений на предмет XSS уязвимостей.

Библиография

[1] Cross-site Scripting (XSS). OWASP™ Foundation. The free and open software security community. URL: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[2] OWASP Top 10 -2017. The Ten Most Critical Web Application Security Risks. OWASP™ Foundation. The free and open software security community. URL: https://www.owasp.org/index.php/Top_10-2017_Top_10

[3] Seth Fogie, Robert Hansen, Jeremiah Grossman, Petko Petkov, Anton Rager, «XSS Attacks: Cross Site Scripting Exploits and Defense», Elsevier, Inc., 482p., USA, 2007.

[4] Статистика атак на веб-приложения. URL: https://www.ptsecurity.com/ru-ru/research/analytics/web-application-attacks-2018/

[5] Payture [Software]. Тестовая среда коммерческого продукта и документация API. URL: https://payture.com/api

[6] The web-application vulnerability scanner Wapiti3 [Software]. URL: http://wapiti.sourceforge.net/

[7] Web Applicaytion Security Solution NetSparker [Software]. URL: https://www.netsparker.com/

[8] OWASP Xenotix XSS Exploit Framework [Software]. URL: https://xenotix.in/

[9] CSP Evaluator [Software]. URL: https://csp-evaluator.withgoogle.com/

[10] OWASP AntiSamy Project [Software]. OWASP™ Foundation. The free and open software security community. URL: https://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project

[11] Прототип реализованного инструмента AntiFraudXSS [Ресурс]: https://drive.google.com/file/d/14kjesiN_m7glXTZlGhi16jyNoqPO06UX/view?usp=sharing_eil&invite=CKSE58QJ&ts=5cfbc4c9

Testing Cross-Site Scripting (XSS) Vulnerabilities in an Online Payment Web Application

Artem S. Merkulov, Olga R. Laponina

***Abstract* – the object of the study is a web-based online payment company Payture, which cooperates with large companies and banks. Payture acts as a payment gateway between merchants, banks and payment systems, offering a flexible integration API. This paper analyzes how to test the Payture online payment web application for the presence of known cross-site scripting attack vectors. A recommended list of protection measures is determined for this application as well as for similar applications. The types of XSS attacks are considered, which allows you to design an attack and analyze the elements of a web application that can contain XSS vulnerabilities. Free and commercial software products are described that allow for a comprehensive or partial analysis of web applications for XSS vulnerabilities. The main features of the API company "Payture" are considered, the sequence of actions for finding XSS vulnerabilities is determined. Suggestions have been made for sharing several tools for analyzing XSS vulnerabilities in web applications.**

***Keywords* - XSS vulnerabilities, Payture online payment web application, NetSparker, OWASP Xenotix XSS, CSP Evaluator, Wapiti3.**

REFERENCES

[1] Cross-site Scripting (XSS). OWASP™ Foundation. The free and open software security community. URL: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[2] OWASP Top 10 -2017. The Ten Most Critical Web Application Security Risks. OWASP™ Foundation. The free and open software security community. URL: https://www.owasp.org/index.php/Top_10-2017_Top_10

[3] Seth Fogie, Robert Hansen, Jeremiah Grossman, Petko Petkov, Anton Rager, «XSS Attacks: Cross Site Scripting Exploits and Defense», Elsevier, Inc., 482p., USA, 2007.

[4] Statistics of attacks on web applications. URL: https://www.ptsecurity.com/ru-ru/research/analytics/web-application-attacks-2018/

[5] Payture [Software]. Commercial product test environment and API documentation. URL: https://payture.com/api

[6] The web-application vulnerability scanner Wapiti3 [Software]. URL: http://wapiti.sourceforge.net/

[7] Web Applicaytion Security Solution NetSparker [Software]. URL: https://www.netsparker.com/

[8] OWASP Xenotix XSS Exploit Framework [Software]. URL: https://xenotix.in/

[9] CSP Evaluator [Software]. URL: https://csp-evaluator.withgoogle.com/

[10] OWASP AntiSamy Project [Software]. OWASP™ Foundation. The free and open software security community. URL: https://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project

[11] Prototype implemented tool AntiFraudXSS [Resource]: https://drive.google.com/file/d/14kjesiN_m7glXTZlGhi16jyNoqPO06UX/view?usp=sharing_eil&invite=CKSE58QJ&ts=5cfbc4c9