Модель применения состязательных атак на системы обнаружения межсайтового выполнения сценариев

А.А. Гусаров

***Аннотация* – В данной статье обсуждается тема атак использующих уязвимости межсайтового выполнения сценариев, являющихся одной из основных угроз веб-безопасности. В статье представлена классификация данной атаки, а также описаны различные варианты вектора атаки. Приводится пример выполнения атаки. Также анализируется повышение уровня использования методов машинного/глубокого обучения в области обнаружения атак межсайтового выполнения сценариев и уязвимость данного методов к состязательным атакам. Статья является полезным ресурсом для разработчиков, которые интересуются вопросами безопасности систем обнаружения межсайтового выполнения сценариев на основе машинного/глубокого обучения. Она предоставляет описание модели применения атаки уклонения на такие системы, основываясь на парадигме глубокого обучения с подкреплением. В статье предлагаются различные варианты для подбора параметров, описываемой модели, такие как алгоритм выбора модификации или сами модификации изначального кода атаки. Благодаря использованию реализации такой модели можно тестировать существующие системы обнаружения межсайтового выполнения сценариев, а также получать дополнительную информацию для более качественного их обучения.**

***Ключевые слова* – атака межсайтового скриптинга, обнаружение атак, состязательные атаки, глубокое обучение.**

I. Введение

В условиях быстрого развития Интернета доступ к нему появляется все у большего количества людей. Такой спрос привел к появлению множества различных веб-приложений, получивших широкое распространение, и способных повысить качество жизни пользователя. Но, в свою очередь, эти сервисы становятся объектом кибератак, поскольку они являются возможной точкой доступа к конфиденциальной информации и базам данных, а из-за обширного количества различных уязвимостей, существует множество атак, приводящих к различным убыткам. Одной из самых распространенных наряду с инъекцией внешних сущностей XML (XXE), SQL-инъекцией и другими является атака межсайтового скриптинга (XSS).

XSS-атаки позволяют злоумышленнику внедрять вредоносные скрипты в браузер жертвы, что приводит к различным побочным эффектам, таким как компрометация данных, кража cookies, паролей, номеров кредитных карт, даже распространению вредоносного кода. Хоть на сегодняшней день было предложено уже много методов защиты от данной уязвимости и ее обнаружения, а исследования по этой теме ведутся постоянно, в реальности очень сложно справиться со всеми разновидностями XSS-атак. В 2022 году эта атака остается в списке десяти самых главных угроз веб-безопасности по версии Open Web Applications Security Project (OWASP).

Целью данной статьи является анализ состояния в области исследования атак межсайтового скриптинга, а также описание модели применения атаки уклонения на системы обнаружения этих атак и ее параметров, которые могут быть полезными при реализации тестирования систем обнаружения, основанных на методах машинного/глубокого обучения.

II. Уязвимость Межсайтового скриптинга (XSS)

Атаки межсайтового скриптинга (XSS) - это тип инъекций, при которых вредоносный код внедряется в уязвимые веб-сайты. XSS-атаки возникают, когда злоумышленник использует веб-приложение для отправки вредоносного кода, обычно в виде сценария на стороне браузера, другому конечному пользователю. Недостатки, которые позволяют этим атакам быть успешными, довольно широко распространены и проявляются везде, где веб-приложение использует ввод данных от пользователя в генерируемых им результатах без их проверки или кодирования.

Злоумышленник может использовать XSS для отправки вредоносного скрипта ничего не подозревающему пользователю. Браузер конечного пользователя не имеет возможности узнать, что скрипт не заслуживает доверия, и выполнит его. Поскольку предполагается, что сценарий пришел из доверенного источника, вредоносный сценарий может получить доступ к любым файлам cookie, маркерам сеанса или другой конфиденциальной информации, сохраненной браузером и используемой на этом сайте. Эти сценарии могут даже переписать содержимое HTML-страницы.

A. Типы XSS

Обычно XSS-атаки классифицируются на три основные категории [1]:

1. **Reflected (непостоянные).** Эта XSS-атака также известна как отраженная XSS-атака, когда внедренный скрипт возвращается обратно на сервер веб-приложения жертвы в виде сообщения об ошибке, результатов работы поисковых систем в веб-приложениях или любой дополнительной ответной веб-страницы

или сообщения, которое включает в себя несколько или всю часть входных данных, переданных на стороне сервера. Злоумышленник обычно побуждает жертву посетить URL, содержащий вредоносный код, с помощью специальных средств (например, электронной почты)

2. **Stored (постоянные).** Этот тип XSS-атаки также известен как хранимая XSS-атака, которая обычно встречается в тех областях веб-приложений, где пользователям Интернета разрешено вводить HTML/JavaScript-код (например, в текстовом поле поиска). По этой причине хранимая XSS-атака хранит вредоносный скрипт в области хранения веб-приложения. Stored XSS намного более опасная угроза, чем Reflected, с той точки зрения, что в этом случае она не требует дополнительных действий от пользователя, таких как перейти по чужой ссылке
3. **DOM-Based XSS атаки.** Его отличие от двух предыдущих типов заключается в том, что он не требует участия сервера. Злоумышленник изменяет DOM-структуру веб-страницы в браузере жертвы, посредством ввода данных, которые при обработке клиентской стороной, приводят к тому, что код страницы выполнится неожиданным образом.

Ниже (Рисунок 1) приведена цепочка шагов эксплуатации уязвимостей атаки Reflected XSS:

1. Злоумышленник передаёт url с вредоносным содержимым *http://..search=<script>alert('xss')</script>*
2. Атакуемый не замечает подвоха и ожидает увидеть что-то о чем говорится в письме с ссылкой
3. Жертва переходит по ссылке и отправляет запрос на уязвимый сервер
4. Сервер возвращает ответ, внутри которого внедрен вредоносный код
5. Код исполняется и происходит непредвиденное событие

.

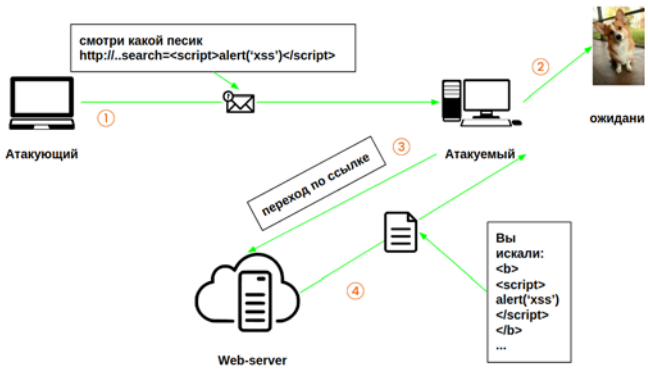

Рисунок 1. Пример выполнения атаки Reflected XSS

B. Контекст XSS

Выше были рассмотрены векторы атаки, но также для XSS важным понятием является контекст внедрения, а именно ситуации или местоположения, в которых пользовательский ввод может отразиться в DOM структуре и, если он не санирован или не закодирован должным образом, может привести к XSS. Веб-страница хоть и является всего лишь текстом, но браузером, различные её разделы могут интерпретироваться не одинаково, например, как HTML, CSS или JavaScript.

Как и в случае обычного языка, слово может иметь различное значению, в зависимости от используемого контекста, так и то каким образом пользовательский ввод повлияет на атаку, связано с выбором браузера того, как он будет интерпретироваться пользовательский ввод. Ниже показаны различные контексты, в которых пользовательский ввод может возникнуть на странице [2].

Простой HTML контекст

В теле существующего HTML тэга или в начале и в конце страницы вне тэга <html>. <html_тэг>пользовательский_ввод</html_тэг>. В этом контексте можно в пользовательский ввод вписать валидный HTML любого рода и он немедленно будет воспроизведён браузером.

Например для тега textarea подойдет такой ввод: *</textarea><script>alert(1)</script>*

Контекст значения HTML атрибута

Внутри открытого HTML тэга, после имени атрибута. <html_тэг имя_атрибута="пользовательский_ввод" />

Выполнение кода в этом контексте будет зависеть от типа атрибута, в котором появляется ввода. Например:

- Атрибуты события

  Это такие атрибуты как onclick, onload и т.д. и значения этих атрибутов выполняются как JavaScript. Поэтому всё здесь – это то же самое, что и JavaScript контекст.

- Специальные URL атрибуты

  Это URL атрибуты, где ввод обычных URL может привести к проблемам безопасности. Несколько примеров:

  <iframe src="пользовательский_ввод"

  <link href="пользовательский_ввод"

  <button formaction="пользовательский_ввод"

  Ввод просто абсолютного http или https URL в этих случаях может оказать эффект на безопасность веб-сайта.

Контекст HTML комментариев

<!--комментарий пользовательский_ввод комментарий-->

Это не исполняемый контекст и требуется выйти из контекста для выполнения кода. Ввод --> завершит этот контекст и переключит весь последующий текст в HTML контекст.

Например: *--><img src=x onerror=alert(1)>*

Контекст JavaScript

Внутри раздела страницы JavaScript кода.

<script> пользовательский_ввод </script>

Это относится к разделу, заключённому в тэги SCRIPT, в значения атрибутов обработчиков событий и в URL, обрабатывающихся с JavaScript.

Внутри JavaScript пользовательский ввод может появляться в различных контекстах. Если пользовательский ввод между тэгами SCRIPT, то не имеет значения, в каком из контекстов он появился, можно переключиться на контекст HTML просто включив закрывающий тэг SCRIPT, а затем вставить любой HTML.

Например:

</script><img src=x onerror=alert(1)>

Как можно заметить, различных вариаций атаки большое множество, что способствует осложнению обнаружения и защиты от XSS. На данный момент существуют различные сайты(шпаргалки), на которых можно найти различные варианты полезной нагрузки и даже способы обхода тех или иных способов защиты и предотвращения атак [3], [4], [5].

III. Состояние исследований

Подавляющее большинство современных исследования XSS атак можно классифицировать по следующим направлениям: исследование XSS-уязвимостей и обнаружение XSS-атак.

Количество работ, посвященных уязвимости межсайтового скриптинга растет с каждым годом. Со временем меняются тренды и подходы. На рисунках ниже представлены результаты обзора нескольких десятков статей за 2022-2023 год, найденных по ключевому слову: “xss” на сервисе Google Scholar.

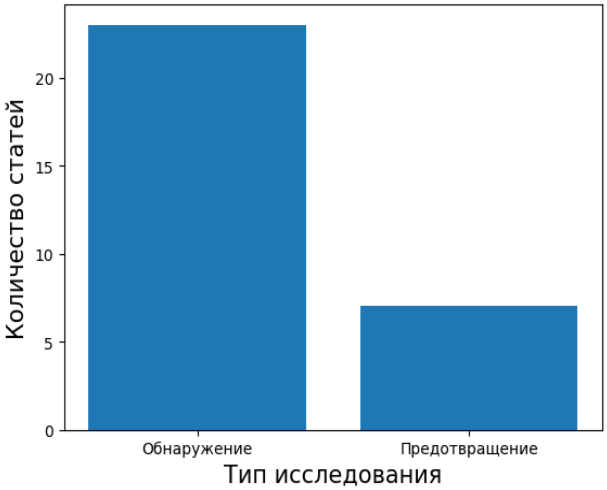

Рисунок 2. Распределение области исследования статей по их типу

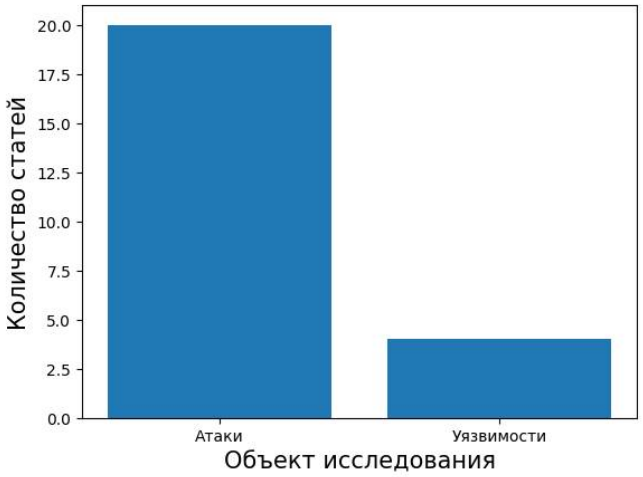

Рисунок 3 . Распределение области исследования статей по объекту исследования

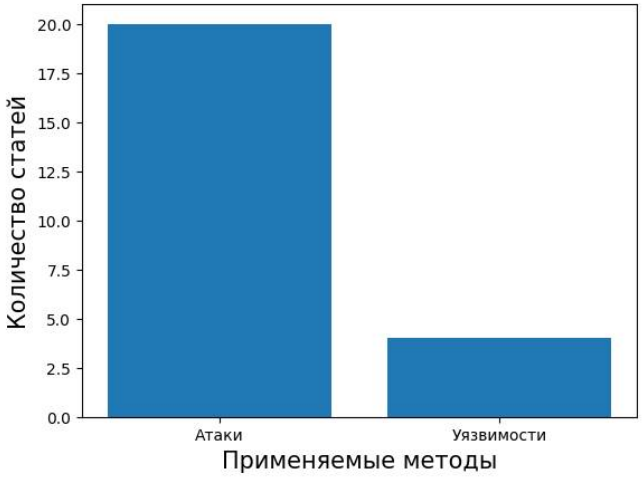

Рисунок 4. Методы, применяемые в исследованиях

На рисунках 2 и 3 видно, что область обнаружения атак исследуется в последнее время чаще остальных. Такой же результат был получен в работе [6] при рассмотрении статей за 2004-2014 годы. А рисунок 4 демонстрирует интерес исследователей к использованию различных методов машинного/глубокого обучения в своих работах.

*A. Обнаруже*ние XSS-атак

Машинное и глубокое обучения за последние годы стали широко распространены и сейчас применяются повсеместно. Область безопасности веб-приложений не стала исключением.

Фанг и др. [7] предложили модель для обнаружения XSS-атак на основе глубокого обучения. В их модели полезная нагрузка декодировалась с помощью обычных методов, после чего признаки полезной нагрузки были извлекались с помощью word2vec и обучались с помощью рекуррентных нейронных сетей с долговременной памятью (LSTM). Результаты показали, хорошую точность обнаружения, тем самым показав, что модель может эффективно идентифицировать XSS-атаки.

Мокбал и др. [8] предложили метод динамического извлечения признаков и модель обнаружения XSS-атак на основе многослойного перцептрона (MLP). В MLPXSS используется набор данных реальной среды, после чего динамически извлекаются признаки полезных нагрузок XSS. Точность, вероятность обнаружения, коэффициент ложных срабатываний и оценка AUC-ROC показали следующие результаты 99,32%, 98,35%, 0,3% и 99,02% соответственно. Однако они выполняли разбор JavaScript и разбор HTML путем обхода, что было неэффективно для обнаружения атак в реальной среде. Данная модель потенциально может быть применима для обнаружения атак на основе XSS как на стороне клиента, так и на стороне сервера, но не является эффективной для обнаружения атак в реальной среде.

Текерек, 2021 [9] предложил архитектуру обнаружения веб-атак на основе аномалий в веб-приложении с использованием методов глубокого обучения. Общая архитектура разделена на предварительную обработку данных и сверточную нейронную сеть (CNN). Это позволило обнаруживать различные атаки на веб-приложения. Результаты показали неплохую точность обнаружения на наборе данных CSIC2010v2, но злоумышленники имели возможности обойти этап предварительной обработки, используя необычные пакеты http-запросов.

В 2022 была опубликована статья [10], в которой была предложена модель обнаружения полезной нагрузки межсайтового скриптинга на основе графовых сверточных сетей, которая может идентифицировать полезную нагрузку межсайтового скриптинга (GraphXSS). Для обучения обработанные данные выборки строились в виде графовой структуры, а потом использовались графовая сверточная сеть и остаточная сеть. В ходе экспериментов модель, основанная на графовой сверточной сети (GCN), смогла достичь значения AUC 0,997 в условиях небольшой выборки. Также показано, что с добавлением структуры остаточной сети к GCN модель могла сходиться и стабилизироваться в условиях многослойности с приблизительно такой же точностью.

Хотя такие методы обнаружения эффективны и исследователи добиваются все лучших результатов, проблема всех этих подходов заключается в том, что они уязвимы к состязательным атакам. При исследованиях, обучение зачастую происходит только на известных наборах данных, не учитывая влияние злокачественных примеров. Так, на практике, модель может быть дискредитирована, если злоумышленник воспользуется, например, методом уклонения, и злокачественные данные не участвовали в процессе обучения.

Ниже представлены примеры применения данных атак на модели обнаружения XSS-атак на основе машинного/глубокого обучения.

Авторы [11] разработали модель XSS-атак на основе алгоритма Dueling Deep Q Networks. Поскольку выбранная ими стратегия обхода является относительно простой, скорость обхода невелика - всего менее 10%.

Ванг и др. [12] предложили метод атаки XSS с использованием алгоритма Soft Q-learning, они разделили весь процесс обхода на 2 этапа: обход HTML и обход JavaScript, что позволило достичь хорошей производительности и степени уклонения в 85%.

В статье [13] авторы попытались преодолеть недостатки, связанные с низкой скоростью обхода и плохим использованием кодировки, предложив правила мутации на основе всевозможных способов, как например, кодировка кода или добавление комментариев в теги. Также в отличие от [12], в этой статье за основу берется другой алгоритм глубокого обучения с подкреплением – SAC (Soft Actor-Critic), который за счет возможности энтропийной регуляризации и максимизации, позволяет сделать выбор стратегии более случайным.

IV. СОСТЯЗАТЕЛЬНЫЕ АТАКИ

Состязательные атаки на системы машинного/глубокого обучения заключаются в том, что злоумышленник пытается сознательно модифицировать данные, используемые на разных этапах работы машинного обучения. Такие действия в основном направлены на то, чтобы понизить качество работы системы или добиться желаемого результата работы системы при определенных условиях. Это возможно, так как системы машинного/глубокого обучения сильно зависимы от данных, поэтому любые изменения в них приводят к изменению работы модели [14].

Данные на этапе тренировки модели непосредственно эту модель и определяют. Соответственно, модификации тренировочных данных изменят (могут изменить) модель по сравнению с оригинальными данными.

Состязательные атаки можно различать в зависимости от разных факторов [15]:

- этап машинного обучения, на который нацелена атака
- эффекта, которого добивается злоумышленник
- информация об атакуемой системе

Два наиболее распространенных типа состязательных атак это атака уклонением и атака отравлением. Согласно факторам описанным выше:

- Атака уклонением - затрагивает этап тестирования данных, нацелена на формирования новых вариантов нагрузки, способных обойти классификатор
- Атака отравлением - затрагивает этап обучения, и нацелена на изменение работы самой модели

Для модели описанной в данной работе подойдет тип атаки уклонением, так как модели, на которые будет совершаться атака используются уже обученные и тренировочные данные нам не доступны. Но с помощью подобранных тестовых данных, можно получить новые виды злонамеренное нагрузки, которая будет более эффективно обходить различные алгоритмы обнаружения XSS. Полученные результаты также можно в будущем добавить в тренировочные данные, что позволит улучшить результаты обнаружения и понизить шанс обойти их, как например в [16].

V. Модель атаки уклонением с использованием глубокого обучения с подкреплением

Так как основная цель атаки уклонения в том, чтобы получить измененный вариант изначального примера из тестовой выборки путем модификаций, то суть обучения будет заключаться в выборе наиболее эффективной последовательности допустимых изменений на основании результатов систем обнаружения. Такой цели в полной мере удовлетворяет идея глубокого обучения с подкреплением, суть которой в обучении некого агента при взаимодействии с выбранной средой посредством применения различных действий.

На рисунке 5 показана модель применения атаки уклонением на способы обнаружения межсайтового выполнения сценариев, основанная на глубоком обучении с подкреплением.

Согласно принципам глубокого обучения с подкреплением у данной модели есть следующие составляющие:

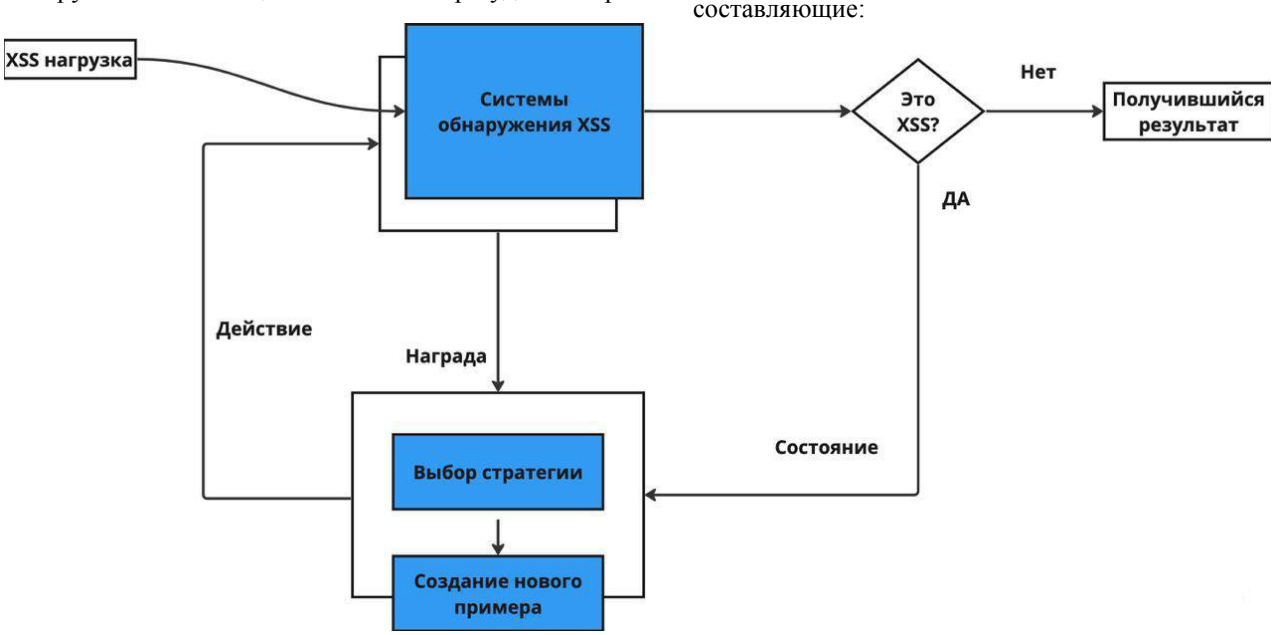

Рисунок 5. Модель атаки уклонения

A. Среда

В качестве среды, с которой будет взаимодействовать агент, для последующего обучения в данной модели выступают рабочие системы обнаружения межсайтового выполнения сценариев на основе методов машинного/глубокого обучения.

Таких методов в нынешней практике появляется все больше, но из них можно выделить те, что являются наиболее популярными и при этом показывают неплохие результаты:

- **MLP**
  Многослойный персептрон (Multilayer Perceptron) — это класс искусственных нейронных сетей прямого распространения, обычно состоящих как минимум из трех слоёв: входного, скрытого и выходного. За исключением входных, все нейроны использует нелинейную функцию активации (рисунок 6).
- **CNN**
  Сверточная нейронная сеть (convolutional neural network) — специальная архитектура нейронных сетей, состоящая из различного рода слоев, и выходы промежуточных из них образуют матрицу или набор матриц
- **LSTM**
  LSTM (long short-term memory, долгая краткосрочная память) — специальный тип рекуррентной нейронной сети, который разработан с целью устранения проблемы долгосрочной зависимости.

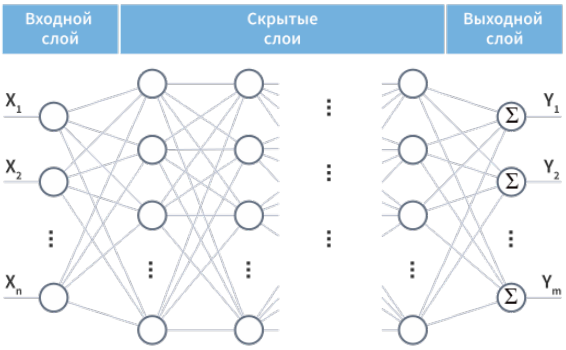

Рисунок 6. Многослойный персептрон [https://wiki.loginom.ru/articles/multilayer-neural-net.html]

Представленные выше методы часто используются для сравнения результатов с разрабатываемыми новыми методами. Для таких целей или тестирования работоспособности модели атаки уклонения можно использовать готовые системы обнаружения, например как в [17].

B. Агент

Агент отвечает за выбор того, какое действие применить на основании состояния, полученного от среды, а также за применение выбранного действия.

Существует множество различных алгоритмов используемых в глубоком обучении с подкреплением, и выбор их должен зависеть от параметров рассматриваемой задачи, поставленных целей и возможностей. Для сравнения алгоритмов используются следующие характеристики:

- Эффективность выборки - сколько шагов (взаимодействий со средой) необходимо для обучения хорошей политики
- Стабильность и сходимость - показатель скорости и простоты сходимости модели.
- Допущения - отображает различные ограничения, как например, применение на дискретном или непрерывном пространстве действий или предположение о том, что состояние среды при обучении может быть сброшено для начала нового эпизода
- Исследование - показатель глубины исследования пространства действий

Далее описывается сравнение нескольких алгоритмов на основе характеристик, представленных выше [18], [19]:

- **Proximal Policy Optimization (PPO)** - популярный и легкий в применении on-policy алгоритм, основанный на Policy Gradient. Данный алгоритм применим как для дискретных, так и для непрерывных пространств действий. Его преимущество в том, что он достигает правильного баланса между производительностью и понятностью.
- **Deep Deterministic Policy Gradient (DDPG)** - off-policy алгоритм, который совмещает в себе походы Policy Gradient и Q-learning. Он был разработан специально для непрерывного пространства действий. В стандартном случае он требует меньшего числа взаимодействий со средой, чем PPO, но обладает не лучшей сходимостью и редко производит поиск действий. Для хорошей работы нужен тщательный выбор гиперпараметров.
- **Soft Actor-Critic (SAC)** - off-policy алгоритм, использующий идею Actor-Critic, чьей ключевой особенностью является стремление максимизации не только вознаграждения, но и энтропии политики. Это позволяет лучше исследовать пространство действий, что также может привести к лучшей сходимости. Применим для непрерывного пространства действий. Также требует настройки гиперпараметра, отвечающего за энтропию.

Как будет описано ниже, для модели, рассматриваемой в данной работе, пространство действий выбрано дискретным, а потому из представленных выше алгоритмов в стандартном представлении подойдет лишь алгоритм PPO. SAC также можно применить, но это требует дополнительных настроек.

Так, если требуется первоначальная проверка уязвимости систем обнаружения к атаке уклонения, можно воспользоваться алгоритмом PPO, если целью является нахождения наилучшего покрытия для вариантов обхода системы, то лучше использовать SAC и другие алгоритмы, которые можно хорошо настроить .

C. Действие

При взаимодействии со средой агент, на основе полученного состояния применяет определенное действие, которое приводит к изменениям в среде и получению нового состояния. Для модели описанной на рисунке , действием будет служить модификация XSS-нагрузки таким образом, что после некоторого количества таких модификаций в конечном итоге получиться обойти системы обнаружения, но при этом оставив сохранив функционал, изменяемой нагрузки. Такие модификации возможны так как атаки межсайтового выполнения сценариев имеют большую вариативность, что связано с различными контекстами, которые были рассмотрены в прошлой разделе 2 данной работы. Пространством действий описанной модели является дискретное множество правил модификаций, пример которых представлен в Таблице 1.

Таблица 1 Пространство действий

Правила модификаций	
1. Использование смешанного регистра символов 2. Добавление валидных слов в HTML-код 3. Замена пробелов на "%0D" 4. Замена "(", ")" на "`" 5. Замена "(", ")" на "(" и “)”	6. Вставка "
" в "javascript:" 7. Замена ":"на ":" 8. Замена "http://" на "//" 9. Замена "alert" на "top['al'+'ert']" 10. Замена "alert" на "top[/al/.source+/ert/.source]"

Чтобы составить подходящую таблицу как вариант можно использовать следующий алгоритм:

1. найти различные сайты “шпаргалки” [3], [4], [5], где приведено множество различных XSS-нагрузок, для обхода тех или иных способов защиты и обнаружения
2. выделить различные паттерны изменения изначальной нагрузки (если применено несколько модификаций, лучше всего выделить именно однозначные простые действия). Например, в таком HTML-тексте:

   <iMG src/onerror=prompt(8)>

   можно выделить применение смешанного регистра для тегов, а также замену пробельных символов на “/”
3. для каждого простого действия можно посмотреть его эффективность применив его к различным исходным текстам и сравнить результаты на выходе системы обнаружения, так составив список возможных модификаций, который можно изначально

отранжировать по полученным коэффициентам эффективности

4. из полученного списка можно составить различные выборки и при обучении тестировать различные сочетания

D. Состояние

Состояние относится к текущему положению, возвращаемой средой. Чтобы состояние отражало реальное положение среды в рассматриваемой модели, оно должно предоставлять структурную информацию о ныне полученной XSS-нагрузке.

Такой информацией может являться знание о предыдущих модификациях. Так, в [13] предлагается в качестве состояния завести вектор, где номер модификации из таблицы 1, является значением вектора на позиции, соответствующей шагу, когда действие было применено. Поэтому изначально заводится вектор S_0 = [0, 0, 0, - - - , 0], заполненный нулями, так как пока никаких модификаций еще не происходило. Потом, каждый раз когда изначальный вектор атаки проходит шаг алгоритма и определяется как вредоносный, значение вектора изменяется в зависимости от действия, выбранного агентом. Например, если на 1-ом шаге, агент выбрал действие 2, то состояние изменится на S_1 = [2, 0, 0, - - - , 0].

Также как отражение состояния среды может быть использована информация о представлении XSS-нагрузке в виде вектора (выход после применения word2vec).

E. Награда

При обучении с подкреплением агент всегда старается максимизировать вознаграждение, поэтому правильный подход к выбору награды (r) очень важен. Среди подходов можно рассмотреть следующие:

- высчитывание награды для каждого шага. Пусть S_i - состояние на i-ом проходе алгоритма обучения, тогда вознаграждение можно выбрать следующим способом:

$$r_i = S_i - S_{i-1} \quad \textbf{(1)}$$

- фиксированные значения, зависящее от состояний. Пусть S_i - состояние на i-ом проходе алгоритма обучения, тогда вознаграждение можно выбрать следующим способом:

$$r = 15, \textit{ если } S_i - S_{i-1} > 0, \quad \textbf{(2)}$$

$$\textit{иначе } r = -1$$

- фиксированные значения, не меняющиеся в течение всей работы алгоритма. Пусть D_i - ответ среды (результат классификации алгоритмами обнаружения) на i-ом проходе алгоритма обучения, тогда вознаграждение можно выбрать следующим способом:

$$r = 15, \textit{ если } S_i\ 0, \quad \textbf{(3)}$$

$$r = -1, \textit{ если } S_i\ 1$$

Агент должен не только получить максимальное вознаграждение, но и достичь свою цель. В рассматриваемой в данной статье модели, целью является получение новой XSS-нагрузки, успешно избегающей системы обнаружения. Тогда применимо к такой цели можно выбрать один из следующих вариантов:

- используя один из первых двух подходов по выбору награды, описанных выше, максимизировать значение уверенности среды в том, что полученная XSS-нагрузка на самом деле является безопасной
- использовать третий способ, который будет сигнализировать о том, достигнута ли цель обучения.

В первом варианте, максимального суммарного вознаграждения можно добиться применением большого количества действий, так как значение вознаграждения изменяется в зависимости от шага. Во втором варианте положительное значение вознаграждения возможно получить лишь достигнув цели, в остальных случаях, применяя очередное действие, суммарное вознаграждение лишь уменьшается. Данный способ может быть более действенным в случае, когда нужно добиться минимального количества модификаций, прежде чем получить нужный результат.

VI. Заключение

Данная статья показывает, что в настоящем, тенденции в области исследования атак межсайтового выполнения сценариев лежат в сторону методов обнаружения данного типа атак с применением методов машинного/глубокого обучения. Эти методы показывают себя как очень эффективные, но все равно имеют недостатки. Была описана модель проведения атаки уклонения на такие системы обнаружения и показаны варианты как можно подобрать параметры для такой модели. Это может быть полезным для тестирования и улучшения работы существующих систем обнаружения.

В заключении можно отметить, что способы борьбы с межсайтовым выполнением сценариев для проведения атак играют ключевую роль в обеспечении надежности работы веб-приложений. Разработчики должны учитывать требования к безопасности при проектировании своих сервисов, следить выполнением стандартных правил построения защиты, не использовать заведомо уязвимый код и обращать внимание на различные технологии, помогающие протестировать приложение или обеспечить дополнительную защиту.

Библиография

[1] Gupta S., Gupta B. B. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art //International Journal of System Assurance Engineering and Management. – 2017. – Т. 8. – №. 1. – С. 512-530.

[2] Уроки по XSS: Урок 3. Контексты внедрения XSS. URL: https://hackware.ru/?p=1234

[3] XSS-payload-list. URL: https://github.com/payloadbox/xss-payload-list

[4] XSS Filter Evasion Cheat Sheet [HTML] (https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html)
[5] Cross-site scripting (XSS) cheat sheet. URL: https://portswigger.net/web-security/cross-site-scripting/cheat-sheet
[6] Khan N., Johari A., Adnan S. A Taxonomy Study of XSS Vulnerabilities //Asian J. Inf. Technol. – 2017. – Т. 16. – С. 169-177.
[7] Fang Y. et al. DeepXSS: Cross site scripting detection based on deep learning //Proceedings of the 2018 international conference on computing and artificial intelligence. – 2018. – С. 47-51.
[8] Mokbal F. M. M. et al. MLPXSS: an integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique //IEEE Access. – 2019. – Т. 7. – С. 100567-100580.
[9] Tekerek A. A novel architecture for web-based attack detection using convolutional neural network //Computers & Security. – 2021. – Т. 100. – С. 102096.
[10] Liu Z. et al. GraphXSS: an efficient XSS payload detection approach based on graph convolutional network //Computers & Security. – 2022. – Т. 114. – С. 102597.
[11] Fang Y. et al. RLXSS: Optimizing XSS detection model to defend against adversarial attacks based on reinforcement learning //Future Internet. – 2019. – Т. 11. – №. 8. – С. 177.
[12] Wang Q. et al. Black-box adversarial attacks on XSS attack detection model //Computers & Security. – 2022. – Т. 113. – С. 102554.
[13] Chen L. et al. XSS adversarial example attacks based on deep reinforcement learning //Computers & Security. – 2022. – Т. 120. – С. 102831.
[14] Намиот Д. Е., Ильюшин Е. А., Чижов И. В. АТАКИ НА СИСТЕМЫ МАШИННОГО ОБУЧЕНИЯ-ОБЩИЕ ПРОБЛЕМЫ И МЕТОДЫ //International Journal of Open Information Technologies. – 2022. – Т. 10. – №. 3. – С. 17-22.
[15] Chakraborty A. et al. Adversarial attacks and defences: A survey //arXiv preprint arXiv:1810.00069. – 2018.
[16] Mondal B., Banerjee A., Gupta S. XSS Filter detection using Trust Region Policy Optimization //2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC). – IEEE, 2023. – С. 1-4.
[17] DL_for_xss, 2017. URL: https://github.com/SparkSharly/DL_for_xss
[18] RL — Reinforcement Learning Algorithms Comparison. URL: https://jonathan-hui.medium.com/rl-reinforcement-learning-algorithms-comparison-76df90f180cf
[19] Ivanov S. Reinforcement Learning Textbook //arXiv preprint arXiv:2201.09746. – 2022.

Алексей Гусаров – МГУ имени М.В. Ломоносова (email: remy1781@gmail.com)

A model for adversarial attacks on cross-site script execution detection systems

A.A. Gusarov

***Abstract* – This article discusses the topic of attacks that exploit cross-site scripting vulnerabilities, which are one of the main threats to web security. The article presents a classification of this attack and describes the different variants of the attack vector. An example of attack execution is given. It also analyzes the increasing use of machine/deep learning techniques to detect cross-site scripting attacks and the vulnerability of this technique to adversarial attacks. The paper is a useful resource for developers who are interested in the security of cross-site scripting detection systems based on machine/deep learning. It provides a description of a model for applying an evasion attack to such systems, based on the reinforcement learning paradigm. The paper proposes various options for fitting the parameters described by the model, such as a modification selection algorithm or the modifications themselves of the original attack code. By using the implementation of such a model, it is possible to test existing cross-site scripting detection systems as well as gain additional information for better training them.**

***Keywords* – XSS, attack detection, adversarial attacks, deep learning.**

REFERENCES

[1] Gupta S., Gupta B. B. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art //International Journal of System Assurance Engineering and Management. – 2017. – T. 8. – #. 1. – S. 512-530.

[2] Uroki po XSS: Urok 3. Konteksty vnedrenija XSS. URL: https://hackware.ru/?p=1234

[3] XSS-payload-list. URL: https://github.com/payloadbox/xss-payload-list

[4] XSS Filter Evasion Cheat Sheet [HTML] (https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html)

[5] Cross-site scripting (XSS) cheat sheet. URL: https://portswigger.net/web-security/cross-site-scripting/cheat-sheet

[6] Khan N., Johari A., Adnan S. A Taxonomy Study of XSS Vulnerabilities //Asian J. Inf. Technol. – 2017. – T. 16. – S. 169-177.

[7] Fang Y. et al. DeepXSS: Cross site scripting detection based on deep learning //Proceedings of the 2018 international conference on computing and artificial intelligence. – 2018. – S. 47-51.

[8] Mokbal F. M. M. et al. MLPXSS: an integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique //IEEE Access. – 2019. – T. 7. – S. 100567-100580.

[9] Tekerek A. A novel architecture for web-based attack detection using convolutional neural network //Computers & Security. – 2021. – T. 100. – S. 102096.

[10] Liu Z. et al. GraphXSS: an efficient XSS payload detection approach based on graph convolutional network //Computers & Security. – 2022. – T. 114. – S. 102597.

[11] Fang Y. et al. RLXSS: Optimizing XSS detection model to defend against adversarial attacks based on reinforcement learning //Future Internet. – 2019. – T. 11. – #. 8. – S. 177.

[12] Wang Q. et al. Black-box adversarial attacks on XSS attack detection model //Computers & Security. – 2022. – T. 113. – S. 102554.

[13] Chen L. et al. XSS adversarial example attacks based on deep reinforcement learning //Computers & Security. – 2022. – T. 120. – S. 102831.

[14] Namiot D. E., Il'jushin E. A., Chizhov I. V. ATAKI NA SISTEMY MAShINNOGO OBUChENIJa-OBShhIE PROBLEMY I METODY //International Journal of Open Information Technologies. – 2022. – T. 10. – #. 3. – S. 17-22.

[15] Chakraborty A. et al. Adversarial attacks and defences: A survey //arXiv preprint arXiv:1810.00069. – 2018.

[16] Mondal B., Banerjee A., Gupta S. XSS Filter detection using Trust Region Policy Optimization //2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC). – IEEE, 2023. – S. 1-4.

[17] DL_for_xss, 2017. URL: https://github.com/SparkSharly/DL_for_xss

[18] RL — Reinforcement Learning Algorithms Comparison. URL: https://jonathan-hui.medium.com/rl-reinforcement-learning-algorithms-comparison-76df90f180cf

[19] Ivanov S. Reinforcement Learning Textbook //arXiv preprint arXiv:2201.09746. – 2022.