Повышение устойчивости к состязательным атакам моделей машинного обучения для обнаружения межсайтового выполнения сценариев

М.А. Хамзаева, О.Р. Лапонина

***Аннотация* — В данной статье предложен подход для увеличения устойчивости алгоритмов машинного и глубокого обучения к состязательным атакам.**

В статье рассматриваются различные способы проведения атак межсайтового выполнения сценариев. Анализируется ряд исследований применения алгоритмов машинного и глубокого обучения (ML/DL) для обнаружения атак межсайтового выполнения сценариев. Описывается общий алгоритм проведения состязательных атак на ML/DL алгоритмы.

Рассматривается сценарий атаки на модель-детектор, функционирующую в режиме «черного ящика». Методом генерации состязательных примеров является обучение с подкреплением. В качестве атакуемых моделей были выбраны MLP, CNN и LSTM.

Для создания состязательных атак используется алгоритм Proximal Policy Optimization (PPO), который дает более стабильное обучение атакующей модели и имеет меньшую чувствительность к гиперпараметрам, сохраняя при этом достаточную производительность.

Решение на основе алгоритма PPO использует отобранные и проверенные на синтаксическую корректность мутации, которые затем используются для пополнения набора данных при повторном обучении модели-детектора. Результаты экспериментов демонстрируют эффективность и применимость предложенного решения. Использовались стандартные метрики качества. Несмотря на изначально высокое значение F1-меры, все модели пропустили 98% и более состязательных примеров, продемонстрировав полную неустойчивость к состязательной атаке. Авторами реализовано двенадцать мутаций XSS-атак, четыре из которых показали наибольшую эффективность.

В дополнение к генерации состязательных примеров, реализовано логирование полученных примеров для формирования датасета для дообучения, а также сбор статистики по успешным или неудачным мутациям.

***Ключевые слова* — атака межсайтового выполнения сценариев, XSS, состязательная атака, Proximal Policy Optimization, PPO, MLP, CNN, LSTM.**

I. Введение

Веб-приложения используются практически всеми людьми, так как они обеспечивают доступ к всевозможным ресурсам и услугам. Но как следствие большинству веб-приложений приходится хранить и обрабатывать те или иные конфиденциальные данные, что делает их привлекательной целью для злоумышленников. Современные веб-приложения подвержены множеству атак, среди которых межсайтовый скриптинг (Cross-Site Scripting, XSS) выделяется как особенно распространённая и опасная уязвимость, которая регулярно вносится OWASP в 10 наиболее часто встречающихся уязвимостей в веб-приложениях [1], и на настоящий момент определяется как вариант атаки инъекцией. XSS-атака заключается во внедрении злонамеренного скрипта в веб-страницу таким образом, что он впоследствии выполняется в браузере жертвы. Способами внедрения могут быть как социальная инженерия – попытка заставить пользователя перейти по подготовленной заранее ссылке, так и взаимодействие с базой данных веб-приложения для внедрения кода, который воспроизведется у пользователя при посещении конкретной страницы. Это позволяет злоумышленнику похищать конфиденциальные данные (например, cookies сессии), выполнять произвольные действия от имени пользователя, распространять вредоносное ПО и так далее.

II. Цели работы

Для обнаружения атак межсайтового выполнения сценариев все чаще используются модели машинного и глубокого обучения, но все они сталкиваются с проблемой уязвимости к состязательным атакам. В связи с этим основными целями статьи являются:

1. Анализ основных способов создания состязательных атак на модели машинного и глубокого обучения.
2. Разработка способа увеличения устойчивости к состязательным атакам.

И для достижения данных целей выделяются следующие подзадачи:

1. Проанализировать особенность атак межсайтового выполнения сценариев.
2. Проанализировать возможные мутации,

Статья получена 17 мая 2025.
М.А. Хамзаева – МГУ имени М.В. Ломоносова (email: khamzaeva.m.a@gmail.com).

О.Р. Лапонина – МГУ имени М.В. Ломоносова (email: laponina@oit.cmc.msu.ru).

сохраняющие синтаксическую целостность атак.
3. Выбрать ML/DL модели для экспериментов.
4. Разработать алгоритм для реализации состязательных атак.
5. Провести эксперимент по увеличению устойчивости рассмотренных моделей к состязательным атакам.

III. Атака межсайтового выполнения сценариев

Атака межсайтового выполнения сценариев (XSS) имеет обширную область применения и большое разнообразие реализаций, что делает ее достаточно сложной для обнаружения и требует постоянного обновления детектирующих систем. XSS можно провести практически в любом HTML-контексте, где отображается неэкранированный пользовательский ввод – в теле тега, значении атрибута, комментарии и так далее [2]. По типам XSS обычно делятся на отраженный XSS, когда результат атаки возвращается жертве сразу, хранимый XSS, когда вредоносный код внедряется на сервер и затем пересылается жертве в ответе на запрос, и XSS на основе DOM [3].

Гибкость языков HTML и JavaScript позволяет злоумышленникам использовать различные мутации, которые позволяют обходить определение XSS на основе сигнатур. Основными способами таких мутаций являются:

1. Использование разных регистров для символов в том или ином теге или атрибуте. Существует вероятность, что в детекторе игнорируется чувствительность к регистру. Например, вместо **<script>** можно использовать **<SCRiPt>** или **<SCRIPT>**.

2. Использование HTML-кодирования. Специальные символы, такие как **<**, **>**, **"** и **'**, можно закодировать в их HTML-эквиваленты (например, **<**, **>**, **"**, **'**). Детектор может не распознать закодированные символы как XSS-атаку.

3. Использование URL-кодирования. Символы также можно кодировать в URL-формате (например, **%3C**, **%3E**). Это эффективно, если XSS-атак внедрена в URL-параметры.

4. Использование Unicode-кодирования. Можно использовать Unicode-представление символов, например **<** для символа **<**.

Если детектор распознает определенные ключевые слова, то возможны следующие мутации:

1. Разделение строк. Можно разбить строку JavaScript на несколько частей и объединить их с помощью оператора конкатенации.

2. Конкатенации строк. Аналогично разделению строк, можно использовать методы конкатенации:

```
<img src=x
onerror=alert(String.fromCharCode (88,83,83))
```

3. Использование нулевых символов. Вставка нулевого символа (**%00**) может запутать детектор, сохранив при этом исполняемость кода атаки.

4. Использование пробельных символов. Табуляция или перевод строки "**&NewLine**;" сработают аналогично предыдущему способу.

5. Использование комментариев. Комментарии могут быть использованы в середине ключевых слов или тегов, что затруднит понимание общей картины детектором:

```
<scri<!--comment-->pt>alert('XSS')</script>
```

6. Использование обратных слэшей. В некоторых случаях можно использовать обратные слэши для экранирования символов или для разделения ключевых слов.

Так же возможно добавление в качестве шума безопасных данных, что опять же увеличит вероятность ошибки при классификации вредоносного и обычного текста.

Перечисленные изменения не меняют семантику изначального кода XSS-атаки, но запутывают его, усложняя обнаружение.

Если детектор корректно выявляет определенные HTML-теги (например, **<script>**), то можно использовать другие мутации, например, альтернативные теги (**img**, **body**, **iframe** и т.д.) или атрибуты событий (**onclick**, **onmouseover**, **onfocus** и т.д.), которые не нарушают семантику XSS:

```
<iframe src="javascript:alert('XSS')">
```

Многообразие возможных приемов модификации и их комбинаций осложняет задачу обнаружения и приводит к успешному проведению атак межсайтового выполнения сценариев.

IV. Методы обнаружения

A. *Классические модели машинного обучения*

Исследование [4] от 2018 года показывает, что SVM, k-NN и Random Forest могут быть успешно использованы для создания классификаторов для обнаружения XSS, что открывает перспективы для дальнейших экспериментов с алгоритмами и данными.

Так, в статье [5] 2022 года вместе с моделью SVM предлагается использовать следующий способ подготовки данных: проводить векторизацию на уровне полезной нагрузки, а не на уровне слов, так как метки в данных относятся именно к совокупности слов. Поэтому после токенизации получается векторное представление каждого токена, рассчитывается средний вектор каждого представления и затем они объединяются в один вектор для всего экземпляра данных, который в итоге и подается на вход модели для обучения.

В работе [6] используют алгоритм на основе XGBoost, а для выделения подмножества полезных признаков предлагают комбинацию критерия прироста информации и алгоритма SBS (Sequential Backward Selection), что позволило получить точность 99,59% на тестовом наборе.

B. *Глубокое обучение*

Достаточно много решений предложено в области глубокого обучения. Это направление хорошо подходит для анализа сложных текстовых структур и предоставляет большие возможности для обнаружения XSS.

В работе [7] используют сверточную нейронную сеть (CNN) и предлагают на этапе предобработки данных кодировать символы, связанные с XSS или SQL, в пары тип-значение, используя уже имеющиеся предметно-ориентированные знания.

В [8] авторы проводят токенизацию на уровне слов в соответствии с разработанными ими регулярными выражениями, учитывающими особенности скриптового языка, и используют word2vec для извлечения признаков. Для обучения используется модель LSTM, представляющая собой разновидность рекуррентной нейронной сети (RNN).

Использование LSTM также рассматривается в работе [9]. Для подготовки данных используется подход, аналогичный представленному в [8], но для повышения эффективности выделения признаков в модель LSTM добавляется механизм внимания, возвращающий значения распределения вероятности внимания.

Более сложная структура использования LSTM в комбинации с CNN приводится в [10]. Извлечение признаков ведется по двум каналам. Первый основан на CNN и представляет собой три параллельных одномерных сверточных слоя, извлекающих локальные признаки с последующим объединением. Второй извлекает семантические признаки и состоит из двунаправленного LSTM (Bi-LSTM), который обеспечивает лучшую работу за счет дополнительного слоя LSTM с противоположным направлением и следующего за ним блока Self-Attention, который призван компенсировать проблемы Bi-LSTM. Затем результаты обоих каналов объединяются, что и дает улучшенный результат. Предобработка при этом остается аналогичной предыдущим работам.

Ещё одним направлением исследований является представление XSS в виде графов. Поскольку полезная нагрузка XSS имеет внутреннюю структуру (HTML-теги, атрибуты, скрипты), авторы [11] предлагают конвертировать входные строки в графовое представление и применять графовую сверточную сеть (GCN) для классификации.

Как мы видим, существует большое количество работ на тему обнаружения XSS с использованием моделей машинного обучения, однако вопрос устойчивости разработанных моделей к состязательным атакам авторами обычно не рассматривается.

V. Состязательные атаки

Одной из проблем моделей машинного обучения является уязвимость моделей к состязательным атакам. Воздействие злоумышленником может производиться на любой этап цикла жизни системы, от аппаратного проектирования до итоговой эксплуатации [12]. Состязательная атака в узком смысле представляет собой добавление небольшого намеренного возмущения (шума) к входному примеру данных, которое в целом не меняет семантики атаки, но серьезно влияет на решение модели, приводя к ошибочному результату.

Классифицировать состязательные атаки можно по различным критериям. Так, в зависимости от доступности модели злоумышленнику, может быть:

- Атака “белого ящика” (white-box attack). Атакующий имеет полный доступ к архитектуре модели, ее параметрам и градиентам.
- Атака “черного ящика” (black-box attack). Атакующий не имеет информации о модели, может только отправлять запросы и получать ответ.
- Атака “серого ящика” (gray-box attack). Атакующий имеет частичную информацию о модели.

Или же по методу генерации состязательных примеров:

- Атаки на основе градиентов (gradient-based attacks). Используется градиент функции потерь модели для нахождения оптимального возмущения.
- Атаки на основе оптимизации (optimization-based attacks). Задачу генерации состязательного примера формулируется как задача оптимизации.
- Атаки на основе генеративных моделей (generative model-based attacks). Для создания состязательных примеров используют генеративные модели.

В случае, когда мы не знаем ничего о внутреннем устройстве атакуемой модели и можем ориентироваться только на ее конечный ответ, то есть в условиях «черного ящика», создание состязательных примеров возможно на основе различных методов оптимизации: дискретной, байесовской, оптимизации нулевого порядка. Возможна генерация примеров в условиях «белого ящика» на модели известной архитектуры с последующим переносом полученных примеров в целевую модель [13]. Так же находят применение мутационный фаззинг – когда шум итерационно добавляется к входным данным, и обучение с подкреплением (RL), когда обучается атакующая модель для подбора оптимальных мутаций и стратегий для атаки.

В работе рассматривается сценарий атаки на модель-детектор, функционирующую в режиме «черного ящика». Методом генерации состязательных примеров является обучение с подкреплением.

Если говорить про состязательные атаки конкретно в контексте обнаружения XSS, то возникает дополнительная проблема в добавлении шума, так как нельзя добавить в XSS совсем случайные изменения – требуется при всех изменениях сохранить работоспособность кода. Это условие ограничивает пространство действий и приводит к необходимости отбора корректных модификаций.

Общая схема состязательной атаки на модель обнаружения выглядит следующим образом:

1. Выбирается известный XSS и подается на вход детектору.
2. Если детектор классифицировал его верно, пример передается атакующей модели, которая выбирает некоторую из доступных модификаций, модификация применяется и уже новый вариант отправляется на вход детектору.
3. Если на этот раз детектор определил наш пример как безопасный, то есть ошибся – мы получили состязательный пример. Иначе повторяется шаг 2 до тех пор, пока не будет получен неверный ответ или до какого-то разумного предела, когда стоит перейти к следующему примеру.

Обучение с подкреплением предлагает большое количество разнообразных алгоритмов для обучения атакующей модели. Так, в [14] предлагают модель «чёрного ящика» на основе алгоритма Soft Q-learning. Агент RL, по сути, проводит фаззинг входной строки, последовательно применяя различные стратегии видоизменения (например, разные схемы экранирования и кодирования) и получая вознаграждение, если итоговый пример не обнаружен моделью. В экспериментах этот подход показал более 85% успешных ускользаний от нескольких различных детекторов XSS, то есть подавляющее большинство атакующих примеров модель ошибочно классифицировала как безопасные.

Похожие идеи были реализованы с использованием алгоритма Soft Actor-Critic (SAC) в работе [15]. Примеры, сгенерированные их SAC-агентом, так же снижали точность детектирования по сравнению с исходными данными, получая до 90% обхода.

Создавать примеры XSS, разбив задачу на две модели, предлагают в [16]. Одна модель отвечает за действия, необходимые для выхода из HTML-контекста, а вторая за обход механизмов фильтрации и обработки входных данных, что позволяет создавать более эффективные вредоносные примеры.

В работе [17] идея опять схожая с [14], [15], но используется алгоритм Deep Double Q-Network (DDQN) и помимо атаки, производится успешная попытка повышения устойчивости к полученным примерам – за десять итераций количество пропущенных состязательных примеров у модели снижается практически до 1%.

В [18] реализуют генеративно-состязательную сеть на основе Monte Carlo Tree Search (MCTS). Хотя авторы и используют очень ограниченный набор модификаций, атака успешно проходит, а с добавлением полученных данных в обучающую выборку устойчивость к состязательным примерам возрастает на 8%.

VI. Оценка качества

Так как стоит задача бинарной классификации, для оценки качества детектирующих моделей используются базовые метрики, такие как:

Accuracy: количество верно классифицированных данных;

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

Recall: количество верно классифицированных XSS по отношению к общему количеству XSS, проходивших классификацию;

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

Precision: количество верно классифицированных XSS по отношению к общему количеству данных, классифицированных как XSS;

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

FPR: частота ложных срабатываний, когда безопасные данные классифицируются как XSS

$$FPR = \frac{FP}{TN+FP} \quad (4)$$

F1-metrics: среднее гармоническое Recall и Precision;

$$2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (5)$$

Error Rate: отношение пропущенных детектором состязательных примеров к общему число поданных состязательных примеров. Чем выше значение ER, тем лучше атака и хуже детектор и наоборот.

$$ER = \frac{\text{Число пропущенных вредоносных примеров}}{\text{Общее число состязательных примеров}} \quad (6)$$

True Positive (TP) - количество (процент) правильно классифицированных XSS.

True Negative (TN) - количество (процент) правильно классифицированных безопасных данных.

False Positive (FP) - количество (процент) неправильно классифицированных безопасных данных (данные приняты за опасные).

False Negative (FN) - количество (процент) неправильно классифицированных XSS (данные приняты за безопасные).

VII. Состязательное обучение

Одним из перспективных подходов повышения устойчивости моделей является состязательное обучение – повторное обучение детектора на специальных атакующих примерах. Идея состоит в том, чтобы включить в обучающую выборку сгенерированные злоумышленником варианты XSS, обошедшие изначально детектор. Модель, «увидев» их во время обучения, в дальнейшем не даст себя обмануть аналогичным приёмом. Такой подход впервые был опробован в области компьютерного зрения (например, для защиты нейросетей от малозаметных изменений пикселей) и доказал свою эффективность.

VIII. Предложенное решение

A. *Выбор мутаций*

Как уже было сказано выше, для создания состязательных XSS примеров требуется ограниченное пространство модификаций, которые называются мутациями. Опираясь на материалы OWASP [19] и другие примеры манипуляций [20], проводимых злоумышленниками для маскировки своей XSS-атаки, мы отобрали набор часто используемых мутаций. Они были проверены на корректность в контексте сохранения выполнимости XSS, и в итоге сформирован набор распространенных и эффективных мутаций для дальнейшего использования (Таблица 1).

Таблица 1. Реализованные мутации

1. Кодирование символов в HTML
2. Кодирование символов в Unicode
3. Hex-кодирование
4. Смена регистра символов
5. Дублирование HTML тегов
6. Замена пробела на "%0A" или "%0D"
7. Замена "**`alert`**" с использованием функции "**`top`**"
8. Замена "(" и ")" на "`"
9. Добавление пробельных символов в "**`javascript`**"

10. Добавление пустого байта в теги
11. Добавление комментариев
12. Добавление случайной строки перед тегами

B. Детали реализации

В качестве моделей детекции, аналогично авторам рассмотренных статей, мы выбрали MLP, CNN и LSTM.

• Многослойный перцептрон (Multilayer perceptron, MLP) – класс искусственных нейронных сетей, состоящий из входного, одного или нескольких скрытых и выходного слоёв. Каждый нейрон в слое выполняет линейное преобразование входных данных с последующим применением нелинейной активационной функции.

• Свёрточная нейронная сеть (Convolutional neural network, CNN) – вид нейронной сети, который хорошо подходит для работы с данными, имеющими структуру, например последовательности или решетки и использующий свертки для обнаружения шаблонов в данных.

• Long short-term memory (долгая краткосрочная память, LSTM) – вид рекуррентной нейронной сети (RNN), предназначенный для работы с последовательными данными, который эффективно запоминает долгосрочную информацию и справляется с проблемами затухающего градиента.

Для создания состязательных примеров предлагается использовать алгоритм Proximal Policy Optimization (PPO) [21]. По сравнению с алгоритмами, применяемыми в предыдущих работах, PPO дает более стабильное обучение атакующей модели и имеет меньшую чувствительность к гиперпараметрам, чем те же SAC или DDQN, сохраняя при этом достаточную производительность. Так же, в отличии от SAC, не требуется дополнительная настройка для работы с дискретным пространством действий.

В процессе обучения атакующая модель получает штраф -1 за каждый шаг, который не привел к обходу детектора и награду +10, когда эпизод заканчивается успехом, что мотивирует ее подбирать кратчайший сценарий для обхода. Возможен вариант, когда дается награда за любое приближение уверенности детектора к тому, что подаваемый пример является безопасным, но мы рассматриваем более строгий случай, когда нам доступен только бинарный ответ модели: 1 – данные являются XSS, 0 – данные безопасные.

В работе использовались библиотеки: Gym – для задания RL-среды, re – для работы с регулярными выражениями в ходе обработки данных и добавления мутаций.

Общая схема решения представлена на Рисунке 1. Помимо основного цикла генерации состязательных

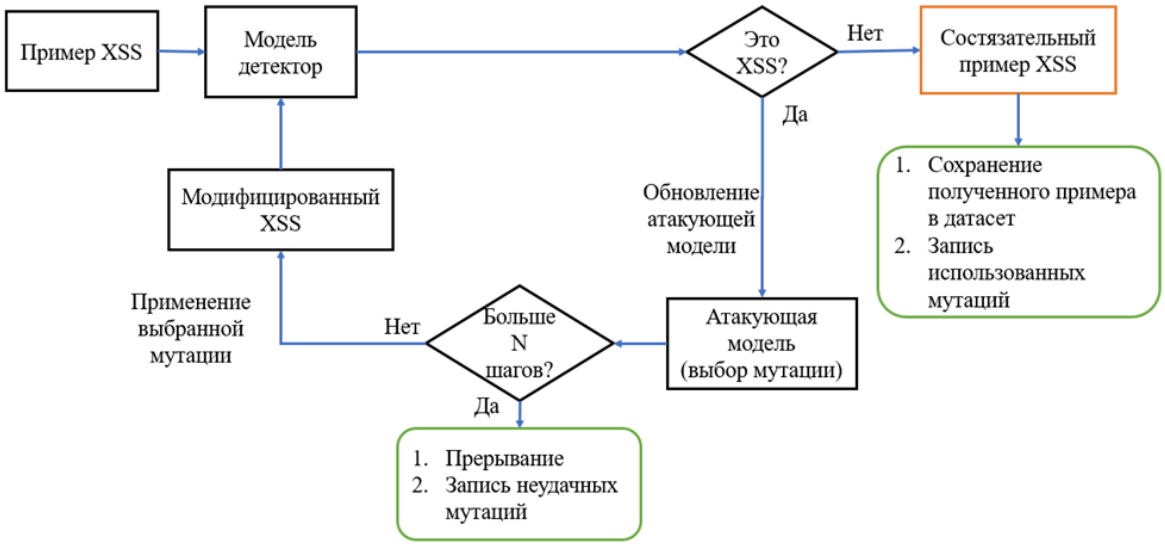

Рис. 1. Схема предложенного решения

примеров, реализовано логирование полученных примеров для формирования датасета для дообучения, а также сбор статистики по успешным или неудачным мутациям.

C. Результаты проведенных экспериментов

Для экспериментов рассматривались различные наборы данных ([22], [23]), но выбор был остановлен на XSSDatasets [24], содержащем большое количество примеров отраженного XSS.

Результаты обучения выбранных моделей и атаки на них представлены в Таблице 2. Несмотря на высокую F1-меру, все модели пропустили 98% состязательных примеров и больше, продемонстрировав полную неустойчивость к состязательной атаке.

Таблица 2. Базовые модели

Модель	LSTM	CNN	MLP
Accuracy	91.05 %	95.90 %	96.27 %
Precision	92.24 %	95.97 %	95.10 %
Recall	89.63 %	95.84 %	97.56 %
F1	90.93 %	95.90 %	96.31 %
ER	99.21 %	98.81 %	98.02 %

Следующим этапом модели-детекторы были обучены заново, уже с добавлением в набор данных состязательных примеров, полученных после атаки (Таблица 3). Можно заметить незначительное снижение метрики F1, но при этом значение ER опускается в некоторых случаях на 29%. Из чего следует вывод, что

добавление в обучающую выборку примеров с мутациями, которые применялись в ходе атаки, увеличило устойчивость моделей обнаружения.

Таблица 3. Модели, обученные с добавлением состязательных примеров

Модель	LSTM	CNN	MLP
Accuracy	86.76 %	95.54 %	95.59 %
Precision	82.67 %	94.99 %	94.27 %
Recall	93.01 %	96.14 %	97.08 %
F1	87.54 %	95.56 %	95.65 %
ER	85.87 %	69.91%	77.56 %

Самыми успешными из отобранных нами мутации оказались 7, 11, 8 и 5 (Рис.2), то есть набор данных, выбранный нами для экспериментов, стоит дополнить примерами с замененной функцией **`alert`** и различным шумом из безопасных данных.

В начале процесса обучения атакующая модель перебирала все доступные варианты, но обнаружив наиболее оптимальные сосредоточилась на их применении как при обучении, так и при последующей атаке.

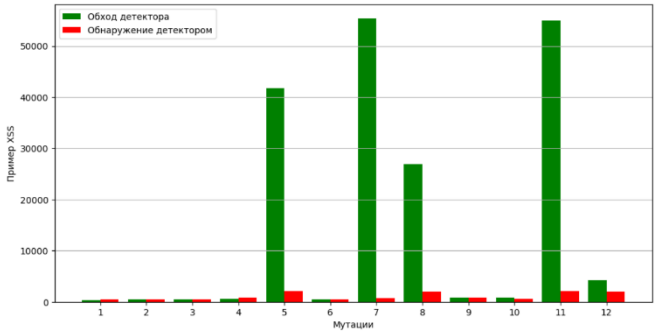

Рис. 2. Распределение применения мутаций при обучении атакующей модели

Однако можно заметить, что остальные мутации так же в некоторых случаях обходили детектор и о полной устойчивости к ним говорить нельзя. Они могут использоваться при создании более сложных примеров после того, как уязвимость к «очевидным», выявленным первыми, модификациям будет уменьшена.

Для обработки более редких мутаций можно обучать атакующую модель, исключив из пространства действий уже выделившиеся, или же обучить с новыми данными детектор чтобы атакующая модель пришла к новым стратегиям самостоятельно.

Повторив несколько раз атаку и обучение с пополненными данными, можно получить последовательное снижение ER, повысив устойчивость ко всему выбранному набору мутаций.

Полученные результаты показывают, что отобранный нами набор модификаций в комбинации с использованием более простого в настройке алгоритма PPO показывает свою эффективность и применимость на практике для повышения устойчивости моделей обнаружения XSS к состязательным атакам и в целом для их тестирования.

В дальнейшем возможно расширение используемого набора мутаций, а также гибкая адаптация для тестирования других моделей-детекторов, так как в связи с работой в условиях черного ящика нашему решению не требуются знания об их внутреннем устройстве.

IX. Заключение

Атака межсайтового выполнения сценариев остается серьезной угрозой в мире веб-безопасности и требует постоянного внимания и обновления данных. Предложено уже большое количество решений для обнаружения на основе алгоритмов машинного обучения, но требуется большое внимание к повышению их устойчивости к состязательным атакам. В данной статье мы проанализировали основные мутации, применяемые для проведения атаки XSS, отобрали и проверили некоторый их набор, и реализовали и успешно протестировали генерацию состязательных примеров на основе алгоритма PPO. Данный подход можно в дальнейшем применять и к другим моделям и датасетам, пополняя наборы данных для достижения желаемого уровня устойчивости.

Благодарности

Тема статьи, написанной по результатам магистерской диссертации, соответствует направлению “Кибербезопасность” на факультете ВМК МГУ [25]. Как примеры похожих работ см. публикации [26,27].

Как обычно, отмечаем работы В.П. Куприяновского и его многочисленных соавторов, положивших начало цифровой повестке в журнале [28, 29].

Библиография

[1] The Open Web Application Security Project (OWASP) Top 10, https://owasp.org/www-project-top-ten/, 2021.
[2] Cross Site Scripting (XSS). OWASP, https://owasp.org/www-community/attacks/xss/.
[3] Weamie, S. Cross-Site Scripting Attacks and Defensive Techniques: A Comprehensive Survey* // International Journal of Communications, Network and System Sciences, 15, 126-148, 2022
[4] Mereani F. A., Howe J. M. Detecting Cross-Site Scripting Attacks Using Machine Learning // The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018). — Cham, 2018. — С. 200—210.
[5] Fawaz Mahiuob Mohammed Mokbal D. W., Wang X. Detect Cross-Site Scripting Attacks Using Average Word Embedding and Support Vector Machine // International Journal of Network Security. — 2022.
[6] F. M. M. Mokbal [и др.] XGBXSS: An Extreme Gradient Boosting Detection Framework for Cross-Site Scripting Attacks Based on Hybrid Feature Selection Approach and Parameters Optimization // Journal of Information Security and Applications. — 2021. — Т. 58. — С. 102813.
[7] Abaimov S., Bianchi G. CODDLE: Code-Injection Detection With Deep Learning // IEEE Access. — 2019. — Т. 7. — С. 128617—128627.
[8] Y. Fang [и др.] DeepXSS: Cross Site Scripting Detection Based on Deep Learning // Proceedings of the 2018 International Conference on Computing and Artificial Intelligence. — Chengdu, China : Association for Computing Machinery, 2018. — С. 47—51. — (ICCAI ’18).
[9] L. Lei [и др.] XSS Detection Technology Based on LSTM-Attention // 2020 5th International Conference on Control, Robotics and Cybernetics (CRC). — 2020. — С. 175—180.
[10] T. Hu [и др.] Cross-site scripting detection with two-channel feature fusion embedded in self-attention mechanism // Computers Security. — 2023. — Т. 124. — С. 102990.
[11] Z. Liu [и др.] GraphXSS: An efficient XSS payload detection approach based on graph convolutional network // Computers Security. — 2022. —Т. 114. — С. 102597.

[12] 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.
[13] Apostol Vassilev, Alina Oprea, Alie Fordyce, and Hyrum Anderson. Adversarial machine learning: A taxonomy and terminology of attacks and mitigations. Technical Report. // National Institute of Standards and Technology. 2024
[14] Q. Wang [и др.] Black-box adversarial attacks on XSS attack detection model // Computers Security. — 2022. — Т. 113. — С. 102554.
[15] L. Chen [и др.] XSS adversarial example attacks based on deep reinforcement learning // Computers Security. — 2022. — Т. 120. — С. 102831.
[16] Foley M., Maffeis S. Haxss: Hierarchical Reinforcement Learning for XSS Payload Generation // 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). — 2022. — С. 147—158.
[17] Y. Fang [и др.] RLXSS: Optimizing XSS Detection Model to Defend Against Adversarial Attacks Based on Reinforcement Learning // Future Internet. — 2019. — Т. 11, № 8.
[18] X. Zhang [и др.] Adversarial Examples Detection for XSS Attacks Based on Generative Adversarial Networks // IEEE Access. — 2020. — Т. 8. — С. 10989—10996.
[19] OWASP. Cross Site Scripting Prevention Cheat Sheet. — https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html.
[20] PortSwigger. Cross-site scripting (XSS) cheat sheet. https://portswigger.net/web-security/crosssite-scripting/cheat-sheet.
[21] J. Schulman [и др.] Proximal Policy Optimization Algorithms // arXiv preprint arXiv:1707.06347v2 — 2017.
[22] Cross site scripting XSS dataset for Deep learning. https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning.
[23] XSS dataset. https://github.com/fawaz2015/XSS-dataset.
[24] XSSDataSets, https://github.com/fmereani/Cross-Site-Scripting-XSS/tree/master/XSSDataSets.
[25] Сухомлин, Владимир Александрович, et al. "Модель цифровых навыков кибербезопасности 2020." Современные информационные технологии и ИТ-образование 16.3 (2020): 695-710.
[26] Yudova, E. A., and Olga R. Laponina. "Analysis of the possibilities of using machine learning technologies to detect attacks on web applications." International Journal of Open Information Technologies 10.1 (2021): 61-68.
[27] Merkulov, Artem S., and Olga R. Laponina. "Testing Cross-Site Scripting (XSS) Vulnerabilities in an Online Payment Web Application." International Journal of Open Information Technologies 7.10 (2019): 59-70.
[28] Умная инфраструктура, физические и информационные активы, Smart Cities, BIM, GIS и IoT / В. П. Куприяновский, В. В. Аленьков, И. А. Соколов [и др.] // International Journal of Open Information Technologies. – 2017. – Т. 5, № 10. – С. 55-86. – EDN ZISODV.
[29] Развитие транспортно-логистических отраслей Европейского Союза: открытый BIM, Интернет Вещей и кибер-физические системы / В. П. Куприяновский, В. В. Аленьков, А. В. Степаненко [и др.] // International Journal of Open Information Technologies. – 2018. – Т. 6, № 2. – С. 54-100. – EDN YNIRFG.

Improving the Resilience of Machine Learning Models to Adversarial Attacks for Cross-Site Scripting Detection

M.A. Khamzaeva, O.R. Laponina

***Abstract* — This paper proposes an approach to increase the resilience of machine and deep learning algorithms to adversarial attacks. The paper considers various ways to conduct cross-site scripting attacks. A number of studies on the use of machine and deep learning (ML/DL) algorithms for detecting cross-site scripting attacks are analyzed. A general algorithm for conducting adversarial attacks on ML/DL algorithms is described. A scenario for an attack on a detector model operating in the "black box" mode is considered. The method for generating adversarial examples is reinforcement learning. MLP, CNN, and LSTM were selected as the models to be attacked. To generate adversarial attacks, the Proximal Policy Optimization (PPO) algorithm is used, which provides more stable training of the attacking model and has less sensitivity to hyperparameters, while maintaining sufficient performance. The solution based on the PPO algorithm uses selected mutations checked for syntactic correctness, which are then used to replenish the dataset when retraining the detector model. The experimental results demonstrate the efficiency and applicability of the proposed solution. Standard quality metrics were used. Despite the initially high F1-measure value, all models missed 98% or more of the adversarial examples, demonstrating complete instability to the adversarial attack. The authors implemented twelve mutations of XSS attacks, four of which showed the greatest efficiency. In addition to generating adversarial examples, logging of the obtained examples was implemented to form a dataset for additional training, as well as collecting statistics on successful and unsuccessful mutations.**

***Keywords* — cross-site scripting attack, XSS, adversarial attack, Proximal Policy Optimization, PPO, MLP, CNN, LSTM.**

REFERENCES

[1] The Open Web Application Security Project (OWASP) Top 10, https://owasp.org/www-project-top-ten/, 2021.
[2] Cross Site Scripting (XSS). OWASP, https://owasp.org/www-community/attacks/xss/.
[3] Weamie, S. Cross-Site Scripting Attacks and Defensive Techniques: A Comprehensive Survey* // International Journal of Communications, Network and System Sciences, 15, 126-148, 2022
[4] Mereani F. A., Howe J. M. Detecting Cross-Site Scripting Attacks Using Machine Learning // The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018). — Cham, 2018. — S. 200—210.
[5] Fawaz Mahiuob Mohammed Mokbal D. W., Wang X. Detect Cross-Site Scripting Attacks Using Average Word Embedding and Support Vector Machine // International Journal of Network Security. — 2022.
[6] F. M. M. Mokbal [i dr.] XGBXSS: An Extreme Gradient Boosting Detection Framework for Cross-Site Scripting Attacks Based on Hybrid Feature Selection Approach and Parameters Optimization // Journal of Information Security and Applications. — 2021. — T. 58. — S. 102813.
[7] Abaimov S., Bianchi G. CODDLE: Code-Injection Detection With Deep Learning // IEEE Access. — 2019. — T. 7. — S. 128617—128627.
[8] Y. Fang [i dr.] DeepXSS: Cross Site Scripting Detection Based on Deep Learning // Proceedings of the 2018 International Conference on Computing and Artificial Intelligence. — Chengdu, China : Association for Computing Machinery, 2018. — S. 47—51. — (ICCAI '18).
[9] L. Lei [i dr.] XSS Detection Technology Based on LSTM-Attention // 2020 5th International Conference on Control, Robotics and Cybernetics (CRC). — 2020. — S. 175—180.
[10] T. Hu [i dr.] Cross-site scripting detection with two-channel feature fusion embedded in self-attention mechanism // Computers Security. — 2023. — T. 124. — S. 102990.
[11] Z. Liu [i dr.] GraphXSS: An efficient XSS payload detection approach based on graph convolutional network // Computers Security. — 2022. —T. 114. — S. 102597.
[12] 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.
[13] Apostol Vassilev, Alina Oprea, Alie Fordyce, and Hyrum Anderson. Adversarial machine learning: A taxonomy and terminology of attacks and mitigations. Technical Report. // National Institute of Standards and Technology. 2024
[14] Q. Wang [i dr.] Black-box adversarial attacks on XSS attack detection model // Computers Security. — 2022. — T. 113. — S. 102554.
[15] L. Chen [i dr.] XSS adversarial example attacks based on deep reinforcement learning // Computers Security. — 2022. — T. 120. — S. 102831.
[16] Foley M., Maffeis S. Haxss: Hierarchical Reinforcement Learning for XSS Payload Generation // 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). — 2022. — S. 147—158.
[17] Y. Fang [i dr.] RLXSS: Optimizing XSS Detection Model to Defend Against Adversarial Attacks Based on Reinforcement Learning // Future Internet. — 2019. — T. 11, # 8.
[18] X. Zhang [i dr.] Adversarial Examples Detection for XSS Attacks Based on Generative Adversarial Networks // IEEE Access. — 2020. — T. 8. — S. 10989—10996.
[19] OWASP. Cross Site Scripting Prevention Cheat Sheet. — https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html.
[20] PortSwigger. Cross-site scripting (XSS) cheat sheet. https://portswigger.net/web-security/crosssite-scripting/cheat-sheet.
[21] J. Schulman [i dr.] Proximal Policy Optimization Algorithms // arXiv preprint arXiv:1707.06347v2 — 2017.
[22] Cross site scripting XSS dataset for Deep learning. https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning.
[23] XSS dataset. https://github.com/fawaz2015/XSS-dataset.
[24] XSSDataSets, https://github.com/fmereani/Cross-Site-Scripting-XSS/tree/master/XSSDataSets.

[25] Suhomlin, Vladimir Aleksandrovich, et al. "Model' cifrovyh navykov kiberbezopasnosti 2020." Sovremennye informacionnye tehnologii i IT-obrazovanie 16.3 (2020): 695-710.
[26] Yudova, E. A., and Olga R. Laponina. "Analysis of the possibilities of using machine learning technologies to detect attacks on web applications." International Journal of Open Information Technologies 10.1 (2021): 61-68.
[27] Merkulov, Artem S., and Olga R. Laponina. "Testing Cross-Site Scripting (XSS) Vulnerabilities in an Online Payment Web Application." International Journal of Open Information Technologies 7.10 (2019): 59-70.
[28] Umnaja infrastruktura, fizicheskie i informacionnye aktivy, Smart Cities, BIM, GIS i IoT / V. P. Kuprijanovskij, V. V. Alen'kov, I. A. Sokolov [i dr.] // International Journal of Open Information Technologies. – 2017. – T. 5, # 10. – S. 55-86. – EDN ZISODV.
[29] Razvitie transportno-logisticheskih otraslej Evropejskogo Sojuza: otkrytyj BIM, Internet Veshhej i kiber-fizicheskie sistemy / V. P. Kuprijanovskij, V. V. Alen'kov, A. V. Stepanenko [i dr.] // International Journal of Open Information Technologies. – 2018. – T. 6, # 2. – S. 54-100. – EDN YNIRFG.ferences