DOI 10.21672/2074-1707.2021.53.1.091-097
УДК 004.85:004.056

АВТОМАТИЗИРОВАННЫЙ ПОИСК УЯЗВИМОСТЕЙ ВЕБ-ПРИЛОЖЕНИЯ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Статья поступила в редакцию 12.12.2020, в окончательном варианте – 15.01.2021.

Выборнова Ольга Николаевна, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, ул. Татищева, 20а,
кандидат технических наук, ORCID: 0001-9458-1093, e-mail: olga.vyb.90@gmail.com
Рыжиков Александр Николаевич, Национальный исследовательский ядерный университет МИФИ, 115409, Российская Федерация, г. Москва, Каширское ш., 31,
магистрант, e-mail: alexander.ryzhikov.work@gmail.com

Проанализирована актуальность задачи создания более эффективного (по сравнению с аналогами) средства автоматизированного поиска уязвимостей на основе современных технологий. Показана схожесть процесса выявлений уязвимостей с марковским процессом принятия решения и обоснована целесообразность применения технологии машинного обучения с подкреплением для решения данной задачи. Поскольку анализ безопасности веб-приложений является в настоящее время наиболее приоритетным и востребованным, в рамках данной работы рассматривается приложение математического аппарата машинного обучения с подкреплением именно к этой предметной области. Представлена математическая модель, описана специфика процесса обучения и тестирования для задачи автоматизированного поиска уязвимостей веб-приложений. На основе анализа руководства по тестированию OWASP определено пространство действий и множество состояний среды. Описаны характеристики программной реализации предложенной модели: Q-обучение реализовано на языке программирования Python, для реализации политики обучения с помощью библиотеки tensorflow была создана нейронная сеть. Продемонстрированы результаты работы агента обучения с подкреплением на реальном веб-приложении, а также их сравнительный анализ с отчетом сканера уязвимостей Acunetix. Полученные данные свидетельствуют о перспективности предложенного решения.

Ключевые слова: уязвимость, автоматизированный поиск уязвимостей, пентестинг, обучение с подкреплением, Q-обучение

AUTOMATED VULNERABILITY SEARCH IN A WEB APPLICATION BASED ON REINFORCEMENT LEARNING

The article was received by the editorial board on 12.12.2020, in the final version – 15.01.2021.

Vybornova Olga N., Astrakhan State University, 20a Tatischev St., Astrakhan, 414056, Russian Federation,
Cand. Sci. (Engineering), ORCID: 0001-9458-1093, e-mail: olga.vyb.90@gmail.com
Ryzhikov Aleksander N., National Research Nuclear University MEPHI, 31 Kashirskoe shosse, Moscow, 115409, Russian Federation,
undergraduate student, e-mail: alexander.ryzhikov.work@gmail.com

We analyzed the urgency of the task of creating a more efficient (compared to analogues) means of automated vulnerability search based on modern technologies. We have shown the similarity of the vulnerabilities identifying process with the Markov decision-making process and justified the feasibility of using reinforcement learning technology for solving this problem. Since the analysis of the web application security is currently the highest priority and in demand, within the framework of this work, the application of the mathematical apparatus of reinforcement learning with to this subject area is considered. The mathematical model is presented, the specifics of the training and testing processes for the problem of automated vulnerability search in web applications are described. Based on an analysis of the OWASP Testing Guide, an action space and a set of environment states are identified. The characteristics of the software implementation of the proposed model are described: Q-learning is implemented in the Python programming language; a neural network was created to implement the learning policy using the tensorflow library. We demonstrated the results of the Reinforcement Learning agent on a real web application, as well as their comparison with the report of the Acunetix Vulnerability Scanner. The findings indicate that the proposed solution is promising.

Keywords: vulnerability, automated vulnerability search, pentesting, reinforcement learning, Q-learning

Graphical annotation (Графическая аннотация)

Введение. Одним из важнейших этапов процесса обеспечения информационной безопасности информационной системы является выявление уязвимостей, или пентестинг (от англ. Penetration Testing). Поиск уязвимостей позволяет получить объективную оценку того, насколько легко осуществить несанкционированный доступ к информационной системе, а также взглянуть на нее с точки зрения злоумышленника, а именно – понять, каким образом можно скомпрометировать данную систему и какие вредоносные действия совершить [6].

Задача обнаружения уязвимостей является трудоемкой, поэтому идея ее автоматизации не нова. Существует большое количество систем автоматизированного поиска уязвимостей. Они применяются в различных средах: обнаруживают уязвимости локальной сети (Nessus, OpenVAS и др.) [1], исследуют безопасность веб-ресурсов (SQL-map, what-web и др.), анализируют бреши в коде программного обеспечения (например, Kaspersky антивирус) и т. д. [2]. При этом в общем случае процесс автоматизированного поиска уязвимостей сводится к полному перебору уязвимостей, содержащихся в базе данных используемого инструмента. Поэтому задача создания более эффективного средства автоматизированного поиска уязвимостей на основе современных технологий является актуальной.

В последнее время наблюдается прорыв в области машинного обучения и искусственного интеллекта. Благодаря приложению глубокого обучения и сложных искусственных нейронных сетей на математический аппарат принятия решений, были достигнуты большие результаты в областях робототехники, эконометрики, компьютерного зрения и т.д. Искусственный интеллект все лучше и лучше справляется с не тривиальными задачами и способен действовать подобно человеку.

В связи с этим целью данной работы является автоматизация процесса поиска уязвимостей на основе применения технологий машинного обучения.

Процесс поиска уязвимостей. Прежде всего необходимо проанализировать классический процесс поиска уязвимостей, производимый человеком вручную. Существует множество методик аудита автоматизированных систем на предмет наличия уязвимостей. Например, общепринятой методикой для тестирования веб-ресурсов на проникновение считается методика OWASP [8]. Данное направление (анализ безопасности веб приложений) является в настоящее время наиболее приоритетным и востребованным [4], поэтому в рамках данной работы целесообразно рассматривать именно эту предметную область.

Анализ методики OWASP показал, что поиск уязвимостей представляет собой итеративный процесс. Целью каждого этапа является сбор дополнительных сведений об исследуемом объекте, которые помогут в дальнейшем анализе. Выбор инструментальных средств для осуществления поиска и эксплуатации уязвимостей при этом зависит от имеющегося набора сведений об исследуемой информационной системе.

Схематически процесс тестирования на проникновение представлен на рисунке 1.

Анализ процесса поиска уязвимостей позволяет сделать вывод о его схожести с Марковским процессом принятия решений (МППР). При этом текущий перечень сведений о системе определяется как состояние системы, а применение той или иной утилиты – как действие [2].

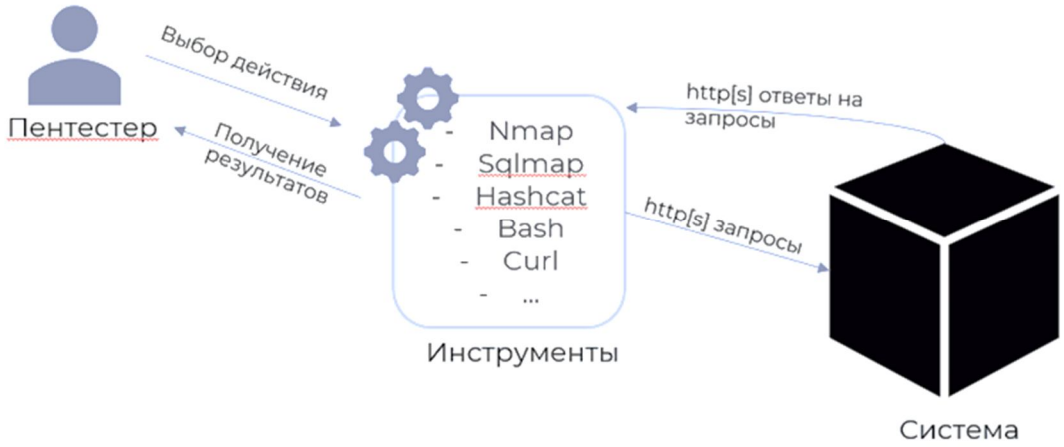

Рисунок 1 – Схема тестирования на проникновение

Построение математической модели. Задаче автоматизированного поиска уязвимостей присущи разнородность входных и выходных данных, отсутствие правильных пар «приложение – уязвимость», взаимодействие модели с объектом анализа, а также схожесть процесса поиска с МППР. В связи с этим при разработке интеллектуальной системы поиска уязвимостей целесообразно применять модель машинного обучения с подкреплением.

Обучение с подкреплением – это обучение машины отображать ситуации в действия, чтобы максимизировать некоторый сигнал поощрения (вознаграждения), принимающий числовые значения. Основная цель обучения с подкреплением – зафиксировать наиболее важные аспекты реальной задачи, имея в виду организацию взаимодействия агента со средой для достижения некоторой цели. Такого рода агент должен иметь возможность в какой-то мере воспринимать состояния окружающей среды, а также – предпринимать действия, которые могут повлиять на состояние среды [7].

В контексте задачи автоматизированного поиска уязвимостей, средой является некий объект исследования: информационная система, сетевой или веб ресурс, программное обеспечение. В произвольный момент времени t агент характеризуется состоянием среды $s_t \in S$ и множеством возможных действий $A(s_t)$. Состоянием является перечень сведений, известных об исследуемом объекте в момент времени t. Выбирая действие $a \in A(s_t)$, агент получает от среды новые сведения, переходит в состояние s_{t+1} и получает выигрыш r_t. При этом если найдены новые уязвимости или хотя бы новые сведения о приложении, выигрыш будет положительным. Отсутствие новых сведений говорит о бесполезности применения действия, следовательно, выигрыш в этом случае будет отрицательным. Основываясь на таком взаимодействии с окружающей средой, агент, обучающийся с подкреплением, должен выработать стратегию P: $S \rightarrow A$, которая максимизирует величину вознаграждения $R = r_0 + r_1 + \ldots + r_n$ в случае МППР, имеющего терминальное состояние, или, в случае МППР без терминальных состояний, величину, рассчитываемую по формуле (1):

$$R = \sum_t \gamma^t r_t \,, \tag{1}$$

где γ – дисконтирующий множитель для «предстоящего выигрыша» ($0 \le \gamma \le 1$) [2].

Действиями в задаче пентестинга являются блоки программного кода, которые взаимодействуют с объектом исследования посредством запросов. В случае веб-приложения используются http и https запросы. Для определения пространства действий целесообразно обратиться к руководству по тестированию OWASP. Согласно данной методике, пространство действий включает в себя:

- сбор первичной информации;
- тестирование конфигурации;
- тестирование идентификации, аутентификации и авторизации;
- тестирование валидации введенных данных;
- тестирование уязвимостей на стороне клиента [8].

Чтобы определить множество состояний среды, было проанализировано, какие данные способны возвращать функции и методы, составляющие пространство действий. Все типы возвращаемых данных можно условно разделить на 3 группы, представленные в таблице 1.

Таблица 1 – Типы возвращаемых значений

Тип	Описание	Примеры
Бинарный	Принимает значений «истина» и «ложь». «Истина» означает, что сетевой узел содержит данную уязвимость или ему присущ данный атрибут	- работает ли данный веб-ресурс по протоколу https; - возвращает ли веб-ресурс, зарегистрированный на данном сетевом узле, cookies
Множественный	Принимают значения из определенного множества, могут быть одинаковыми у нескольких сетевых узлов	- версия веб-сервера Apache может принимать значения из множества {1.0, 1.1, 1.2.1, …}; - адрес панели администратора часто принимает значения из множества {admin, administrator, administrative}
Уникальный	Может принимать как строковые, так и числовые значения. У разных сетевых узлов они не повторяются	- пароль администратора; - уязвимый параметр в get http запросе для слепой sql инъекции

Обозначенные группы данных необходимо привести к формату, пригодному для дальнейшего обучения системы – сделать все параметры бинарными. В случае «уникальных» параметров: если значение параметра существует, то при переводе к бинарному виду он принимает значение «истина», в обратном случае – «ложь».

Обучение и программная реализация модели. Существуют разные методы обучения с подкреплением. Самым известным является Q-обучение, которое целесообразно использовать в поставленной задаче. Суть алгоритма Q-обучения в следующем: агент на основе получаемого от среды вознаграждения формирует функцию полезности Q, что впоследствии дает ему возможность выбирать стратегию поведения не случайно, а учитывать опыт предыдущего взаимодействия со средой. Одно из преимуществ Q-обучения – то, что оно в состоянии сравнить ожидаемую полезность доступных действий, не формируя модели окружающей среды [5, 7].

Таким образом, алгоритм – это функция качества от состояния и действия: $Q{:}S\times A\rightarrow R$.

Перед обучением Q инициализируется случайными значениями. Далее в каждый момент времени t агент выбирает действие a_t, получает награду r_t, переходит в новое состояние s_{t+1}, которое может зависеть от предыдущего состояния s_t и выбранного действия, и обновляет функцию Q. Обновление функции использует взвешенное среднее между старым и новым значениями:

$$Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot \max_a Q(s_{t+1}, a)), \quad (2)$$

где r_t – это награда, полученная при переходе из состояния s_t в состояние s_{t+1}, и α – это скорость обучения ($0 < \alpha \leq 1$). Алгоритм завершает работу, когда агент переходит в терминальное состояние s_{t+1} [5].

Блок-схема алгоритма обучения с подкреплением для задачи автоматизированного поиска уязвимостей, а также пример его работы для одного веб приложения были опубликованы ранее в [2].

Для доказательства эффективности предлагаемого подхода была программно реализована упрощенная модель системы автоматизированного поиска уязвимостей веб-ресурсов. Алгоритм Q-обучения был реализован на языке программирования Python [10]. Для реализации политики обучения с помощью библиотеки tensorflow была создана нейронная сеть [3]. Она имеет следующие характеристики:

1. Входной слой – вектор бинарных значений, описывающих состояние среды. Количество входных нейронов соответствует количеству элементов в векторе состояния среды.
2. Скрытый слой. Количество нейронов скрытого слоя соответствует количеству нейронов на входном слое. Функция активации – функция сигмоиды.
3. Выходной слой – вектор вещественных значений, определяющий предпочтительность применения каждого действия. Количество нейронов выходного слоя соответствует количеству функций в пространстве действий.
4. Оптимизатор – метод обновления весов – «adam» – улучшенный алгоритм стохастического градиентного спуска, обеспечивающий более высокую сходимость.
5. Функция потерь – среднеквадратичная ошибка.

Пространство действий соответствует действиям, представленным в методике OWASP, и содержит в том числе функции таких программных средств анализа уязвимостей, как nmap, SQLmap, WhatWeb, XSScrapy. При реализации пространства действий использовались библиотеки с открытым исходным кодом [9, 11, 13]. Вектор бинарных значений, описывающий текущие сведения о веб ресурсе, состоит из 256 элементов, которые описывают веб сервер, версию веб сервера, субдомены, открытые директории, SQL-инъекции, XSS уязвимости и другие характеристики системы. Были подготовлены тестовый и обучающий набор данных – характеристик различных веб-ресурсов.

Экспериментально были получены значения гиперпараметров модели γ и α. Каждая эпоха обучения представляет собой t итераций, в которых, s_t – состояние на t-й итерации, a_t – действие, которое следует применить на t-й итерации. Награда $r(t)$ равна +1, если найдены новые сведения, и –1 – в противном случае. В качестве оптимального действия на t-м шаге выбирается действие с максимальным значением функции качества. Разработанной модели на вход подавались адреса реальных веб-приложений, на которых модель обучалась в течение более 100 эпох.

Анализ полученных результатов. Классическим подходом для определения качества модели обучения с подкреплением является исследование суммарной награды. Для этого во время обучения для каждого объекта исследования, то есть для каждого веб-приложения, накапливается суммарная награда, затем берется усредненное значение для всех веб-приложений. Таким образом, после нескольких эпох обучения можно построить график зависимости суммарной награды R, полученный за эпоху обучения, от числа эпох e (рис. 2). На графике видно, что на первых эпохах суммарная награда отрицательна. Однако с ростом числа эпох обучения растет и суммарная награда, что говорит об эффективности модели.

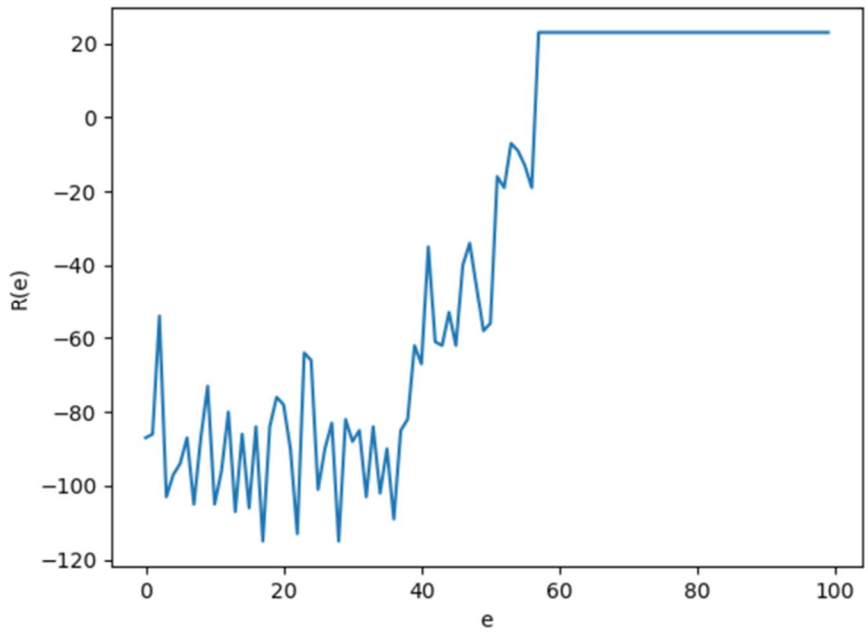

Рисунок 2 – График зависимости суммарного выигрыша от эпохи обучения

Кроме того, с целью проверки работоспособности и эффективности предложенной модели было проведено тестирование реального веб-приложения и сравнение результатов с результатами известного сканера уязвимостей.

Для анализа был выбран веб сайт testphp.vulnweb.com, который предоставляется компанией Acunetix для тестирования работоспособности одноименного сканера уязвимостей [12]. С помощью разработанного приложения был произведен его анализ и получены следующие результаты:

1. Был корректно определен веб-сервер – nginx.
2. Была корректно определена уязвимость integer overflow – у сервера nginx.
3. Была найдена отраженная XSS-уязвимость в POST параметре страницы /guestbook.
4. Была найдена SQL-инъекция в GET-параметрах страницы /search.php.
5. Была корректно определена СУБД – MySQL.
6. SQL-инъекция была эксплуатирована, в результате чего были определены названия всех таблиц и сделан дамп самих таблиц.
7. Был определен пользователь базы данных – acuart.
8. Был проанализирован дамп таблицы users, в котором был найден логин – test и пароль в открытом виде – test.
9. Был успешно произведен вход с использованием обнаруженных логина и пароля.
10. На странице пользователя, доступной после входа, была найдена еще одна XSS-уязвимость – в параметре name пользователя. Уязвимость хранимая, а значит, более опасная.

Это же веб-приложение было проанализировано с помощью сканера уязвимостей Acunetix. Так как данный веб-сайт разрабатывался специально для демонстрации работы этой программы,

очевидно, что сканер тоже показал хорошие результаты. В количественном отношении сканер нашел более 50 уязвимостей, однако большинство из них повторяются или являются ложными срабатываниями. Так, сканер нашел более 20 SQL-инъекций в одних и тех же post-запросах на разных страницах. С другой стороны, он обнаружил уязвимость Directory Traversal, в одном из GET-параметров страницы getimage.php. Действительно, если отправить запрос со значением параметра типа «/../../../../../etc/passwd», сервер вернет ответ со статусом 200. Однако в качестве ответа вернется информация об ошибке, в которой говорится о запрете на доступ в данную директорию. Следовательно, веб-приложение защищено от атак через уязвимость Directory Traversal.

Таким образом, в результате детального сравнительного анализа двух программ была сформирована таблица 2.

Таблица 2 – Сравнение разработанной модели с аналогом

Уязвимость	Acunetix Vulnerability Scanner	Разработанный сканер на основе обучения с подкреплением	Потенциальные угрозы
Integer Overflow	+	+	Отказ в обслуживании веб сервера
SQL-инъекция типа Union	+	+	Утечка данных из базы данных (БД)
SQL-инъекция типа Blind	+	+	Утечка данных из БД
Отраженная XSS-уязвимость	+	+	Перехват атрибутов доступа пользователя путем заманивания его на страницу
Хранимая XSS-уязвимость	-	+	Перехват атрибутов пользователя
Directory Traversal	+	-	Угрозы отсутствуют

Таким образом, агент обучения с подкреплением справился с поставленной задачей лучше. Он не только обнаружил SQL-инъекцию, но и предпринял действия для ее эксплуатации, собрав при этом дополнительные данные, которые пригодились для более глубокого тестирования. Модель автоматически определила страницу аутентификации и с помощью полученных заранее данных смогла войти в систему и обнаружить, благодаря этому, новую, более опасную уязвимость. Следует отметить, что сканер уязвимостей Acunetix нашел гораздо больше уязвимостей ввиду того, что обладает большой базой знаний для проведения автоматических тестов. При этом сканер не способен эксплуатировать уязвимости в реальном времени, что делает его потенциально менее эффективным, чем предложенная модель.

Заключение. Проведенное исследование показывает целесообразность применения математической модели машинного обучения с подкреплением для задачи поиска уязвимостей веб ресурсов. Разработанный сканер уязвимостей является прототипом, однако анализ результатов его работы свидетельствует о перспективности предложенного решения. Расширение базы знаний и пространства действий обеспечит большую эффективность и позволит применять данную модель и разработанное программное средство для реальных задач тестирования веб приложений на проникновение.

Библиографический список

1. Ажмухамедов И. М. Выявление аномалий в вычислительных сетях общего пользования на основе прогнозирования объема сетевого трафика / И. М. Ажмухамедов, А. Н. Марьенков // Проблемы информационной безопасности. Компьютерные системы. – 2012. – № 3. – С. 35–39.

2. Выборнова О. Н. Применение машинного обучения с подкреплением для автоматизированного поиска уязвимостей информационных систем / О. Н. Выборнова, А. Н. Рыжиков // Математические методы в технике и технологиях. – ММТТ. – 2020 – Т. 4. – С. 110–113.

3. Жерон О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow / О. Жерон. – Москва : Вильямс, 2018. – 688 с.

4. Итоги внешних пентестов 2020. – Режим доступа: https://www.ptsecurity.com/ru-ru/research/analytics/external-pentests-2020/, свободный. – Заглавие с экрана. – Яз. рус. (дата обращения: 28.12.2020).

5. Обучение с подкреплением. – Режим доступа: http://neerc.ifmo.ru/wiki/index.php?title=Обучение_с_подкреплением (дата обращения 28.12.2020), свободный. – Заглавие с экрана. – Яз. рус.

6. Плешков А. С. Тестирование на проникновение как анализ защищенности компьютерных систем / А. С. Плешков, Д. Д. Рудер // Известия Алтайского государственного университета. – 2015. – № 1 (85). – С. 174–181.

7. Саттон Р. С. Обучение с подкреплением / Р. С. Саттон, Э. Г. Барто. – Москва : БИНОМ. Лаборатория знаний, 2017. – 380 с.

8. Meucci M. Owasp Testing Guide / M. Meucci, A. Muller. – OWASP Foundation, v4.0, 2014. – 384 p.

9. PwnXss: Vulnerability (XSS) scanner exploit. – Режим доступа: https://github.com/pwn0sec/PwnXSS, свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 28.12.2020).

10. Reinforcement learning tutorial using Python and Keras. – Режим доступа: https://adventuresinmachinelearning.com/reinforcement-learning-tutorial-python-keras/, свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 28.12.2020).

11. Sqlmap. – Режим доступа: https://github.com/sqlmapproject/sqlmap свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 28.12.2020).

12. Test and Demonstration site for Acunetix Web Vulnerability Scanner. – Режим доступа: http://testphp.vulnweb.com/ свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 28.12.2020).

13. WAD – Web Application Detector. – Режим доступа: https://github.com/CERN-CERT/WAD (дата обращения: 28.12.2020), свободный. – Заглавие с экрана. – Яз. англ.

References

1. Azhmukhamedov I. M., Marenkov A. N. Vyyavlenie anomaliy v vychislitelnykh setyakh obshchego polzovaniya na osnove prognozirovaniya obema setevogo trafika [Anomaly detection in public computer networks on the basis of forecasting volume of network traffic]. *Problemy informatsionnoy bezopasnosti. Kompyuternye sistemy* [Information Security Problems. Computer Systems], 2012, no. 3, pp. 35–39.

2. Vybornova O. N., Ryzhikov A. N. Primenenie mashinnogo obucheniya s podkrepleniem dlya avtomatizirovannogo poiska uyazvimostey informatsionnykh sistem [Reinforcement learning for automated vulnerability search]. *Matematicheskie metody v tehnike i tekhnologiyakh – MMTT* [Mathematical Methods in Technics and Technologies – MMTT], 2020, vol. 4, p. 110-113.

3. Geron A. *Prikladnoe mashinnoe obuchenie s pomoshhyu Scikit-Learn i TensorFlow* [Hands-On Machine Learning with Scikit-Learn and TensorFlow]. Moscow, Williams Publ., 2018. 688 p.

4. *Itogi vneshnikh pentestov 2020* [Results of external pentests 2020]. Available at: https://www.ptsecurity.com/ru-ru/research/analytics/external-pentests-2020/ (accessed 28.12.2020).

5. *Obuchenie s podkrepleniem* [Reinforcement learning]. Available at: http://neerc.ifmo.ru/wiki/index.php?title=Обучение_ с_подкреплением (accessed 28.12.2020).

6. Pleshkov A. S., Ruder D. D. Testirovanie na proniknovenie kak analiz zashchishhennosti kompyuternykh sistem [Penetration testing as a security analysis of computer system]. *Izvestiya Altayskogo gosudarstvennogo universiteta* [Izvestiya of Altai State University], 2015, no. 1 (85), pp. 174–181.

7. Sutton R. S., Barto A. G. *Obuchenie s podkrepleniem* [Reinforcement learning]. Moscow, BINOM Publ., 2017. 380 p.

8. Meucci M., Muller A. *Owasp Testing Guide*. OWASP Foundation, v4.0, 2014. 384 p.

9. *PwnXss: Vulnerability (XSS) scanner exploit.* Available at: https://github.com/pwn0sec/PwnXSS (accessed 28.12.2020).

10. *Reinforcement learning tutorial using Python and Keras.* Available at: https://adventuresinmachinelearning.com/reinforcement-learning-tutorial-python-keras/ (accessed 28.12.2020).

11. *Sqlmap.* Available at: https://github.com/sqlmapproject/sqlmap (accessed 28.12.2020).

12. *Test and Demonstration site for Acunetix Web Vulnerability Scanner.* Available at: http://testphp.vulnweb.com/ (accessed 28.12.2020).

13. *WAD – Web Application Detector.* Available at: https://github.com/CERN-CERT/WAD (accessed 28.12.2020).