ОБНАРУЖЕНИЕ АТАК НА ВЕБ-ПРИЛОЖЕНИЕ С ПОМОЩЬЮ САМООРГАНИЗУЮЩИХСЯ КАРТ КОХОНЕНА

Долгачев М. В.[1], Москвичев А. Д.[2], Москвичева К. С.[3]

DOI: 10.21681/2311-3456-2024-1-38-44

Цель статьи: *увеличение эффективности обнаружения атак на веб-приложения.*

Метод: *использование самоорганизующейся карты Кохонена для выявления атак на веб-приложения в режиме реального времени.*

Полученный результат: *рассмотрена самоорганизующаяся карта Кохонена как средство обнаружения аномальных данных. Проанализирована возможность интеграции самоорганизующейся карты Кохонена со средством защиты веб-приложений от атак и уязвимостей, то есть с системой класса Web Application Firewall. Реализовано программное средство, позволяющее выявлять аномалии в HTTP запросах и ответах средствами самоорганизующейся карты Кохонена, подобраны параметры для нейронной сети. Выбраны метрики, извлекающиеся из HTTP запросов для анализа нейронной сетью. Произведена интеграция реализованного программного средства с веб-сервером NGINX средствами модуля NGX JavaScript. Проведено функциональное и нагрузочное тестирование полученного комплекса средствами сканера безопасности OWASP ZAP. Полученные результаты позволили сделать вывод о том, что самоорганизующаяся карта Кохонена эффективно выявляет аномалии, однако ее необходимо использовать с шаблонными методами анализа.*

Практическая ценность: *в рамках исследования разработана методология тестирования средств защиты веб-приложений. Описаны программные средства, из которых состоит стенд для тестирования. Перечислены метрики, позволяющие объективно оценить эффективность средства защиты веб-приложения.*

Ключевые слова: *компьютерная атака, Web Application Firewall, защита информации, нейронные сети, анализ трафика, система обнаружения вторжений, Mutillidae.*

DETECTION OF ATTACKS ON WEB APPLICATION USING SELF-ORGANIZING KOHONEN MAPS

Dolgachev M. V.[4], Moskvichev A. D.[5], Moskvicheva K. S.[6]

Purpose of the article: *improving the effectiveness of web application attack detection.*

Method: *using self-organizing maps for real-time detection of attacks on web applications.*

The result: *The self-organizing map was considered as a means of detecting anomalous data. The possibility of integrating the self-organizing map with a web application firewall system for protection against attacks and vulnerabilities was analyzed. A software tool was implemented to detect anomalies in HTTP requests and responses using the self-organizing map. Parameters for the neural network were selected, and metrics were chosen for analysis by the neural network from the HTTP requests. The implemented software tool was integrated with the NGINX web server using the NGX JavaScript module. Functional and load testing of the integrated system was conducted using the OWASP ZAP security scanner. The results obtained led to the conclusion that the self-organizing map effectively detects anomalies, but it should be used in conjunction with template-based analysis methods.*

Practical value: *within the framework of the study, a methodology for testing web application security tools has been developed. The software that makes up the test bench is described. The metrics that allow you to objectively assess the effectiveness of the web application protection tool are listed.*

Keywords: *computer attack, Web Application Firewall, information security, OWASP, NGINX, Self-Organizing Maps, HTTP, neural networks.*

1 Долгачев Михаил Владимирович, кандидат технических наук, доцент, ФГБОУ ВО «Тихоокеанский государственный университет», г. Хабаровск, Россия. E-mail: 007428@pnu.edu.ru ORCID:0000-0003-1520-800X

2 Москвичев Антон Дмитриевич, аспирант, ФГБОУ ВО «Тихоокеанский государственный университет», г. Хабаровск, Россия. E-mail: anton.moskvichev.1996@yandex.ru ORCID: 0000-0001-6532-2463

3 Москвичева Ксения Сергеевна, студент, ФГБОУ ВО «Тихоокеанский государственный университет», г. Хабаровск, Россия. E-mail: 2016104073@pnu.edu.ru

4 Mihail V. Dolgachev, Ph.D. (in Tech.), Pacific National University, Khabarovsk, Russia. E mail: 007428@pnu.edu.ru. ORCID: 0000-0003-1520-800X

5 Anton D. Moskvichev, postgraduate, Pacific National University, Khabarovsk, Russia. E mail: anton.moskvichev.1996@yandex.ru ORCID: 0000-0001-6532-2463

6 Ksenia S. Moskvicheva, student, Pacific National University, Khabarovsk, Russia. E mail: 2016104073@pnu.edu.ru

Введение

Web Application Firewall (WAF) – это межсетевой экран, специально разработанный для защиты веб-приложений от различных атак и уязвимостей. Он размещается между веб-сервером и клиентами веб-приложения и контролирует входящий и исходящий трафик. Может представлять из себя как программный, так и программно-аппаратный комплекс [1].

Чаще всего WAF используют шаблонные правила, заранее предопределенные аналитиками. Основной их недостаток – отсутствие адаптивности. Правила предопределены заранее и не могут изменяться автоматически для учета новых уязвимостей или атак.

Например, если WAF использует шаблонные правила для обнаружения атак типа SQL-инъекции, они могут быть эффективными против известных шаблонов атак. Однако могут существовать новые уязвимости или варианты атак, которые не учитываются в этих шаблонах. Без обновления шаблонов правил, WAF не сможет корректно обнаруживать и, как следствие, защищать веб-приложение от новых угроз [2].

Для достижения высокого уровня безопасности рекомендуется использовать комбинацию шаблонных правил и алгоритмов машинного обучения, чтобы обеспечить более адаптивную и эффективную защиту веб-приложений.

Самоорганизующаяся карта Кохонена (или Self-Organizing Map) – это нейронная сеть без учителя, используемая для визуализации и анализа сложных данных. Самоорганизующиеся карты Кохонена применяются во многих областях и задачах анализа данных. Одна из них – обнаружение аномалий. Карта Кохонена способна выявлять аномальные или необычные образцы данных, которые не соответствуют ожидаемым моделям. Эта способность делает ее полезной для обнаружения и предотвращения аномальных событий или атак в различных областях. Интеграция карты Кохонена в качестве модуля WAF может позволить более эффективно выявлять угрозы информационной безопасности, тем самым обеспечивая более высокий уровень защиты от неизвестных ранее атак [3].

1. Самоорганизующаяся карта Кохонена, общие сведения

Самоорганизующаяся карта Кохонена представляет собой двумерную сетку, состоящую из нейронов, которые упорядочены в рамках определенной топологии. В качестве топологии могут выступать прямоугольная или шестиугольная сетки. Каждый нейрон имеет веса, которые инициализируются случайным образом и представляют собой векторы той же размерности, что и входные данные [4].

Прямоугольная сетка – это простой вариант топологии, где нейроны размещены в виде прямоугольной матрицы. В этой топологии каждый нейрон имеет четырех соседей внутри сетки. Преимущество прямоугольной сетки состоит в ее простоте и прямолинейности, что делает ее хорошим выбором для задач, где важна пространственная организация данных. Или если карта Кохонена должна быть сгенерирована с определенными ограничениями.

В шестиугольной сетке нейрон находится рядом с шестью соседями. Шестиугольная сетка обеспечивает большую симметрию и позволяет более гибко адаптироваться к структуре данных. Это особенно полезно, если данные не очень хорошо выравнены или имеют сложные взаимосвязи.

Процесс обучения нейронной сети состоит из нескольких итераций искать наилучшие соответствия между образцами данных и нейронами карты. Он проходит через следующие шаги:

1. ***Инициализация весов***. Это процесс, при котором веса нейронов инициализируются случайным образом.

При инициализации весов в самоорганизующейся карте Кохонена, обычно используется случайная инициализация. Каждый нейрон имеет вектор весов, который определяет его положение на карте. Вектор весов инициализируется случайными значениями, чтобы обеспечить случайный разброс нейронов на карте.

Типичным способом инициализации является выбор случайного значения для каждой компоненты вектора весов из некоторого диапазона. Часто используется равномерное распределение случайных чисел или гауссовское распределение. Например, значения весов могут быть случайно выбраны из интервала [0,1] или из стандартного нормального распределения.

Случайная инициализация весов позволяет нейронам начать адаптироваться к различным образцам данных в процессе обучения и постепенно организовывать их на карте. В дальнейшем веса будут изменяться и обновляться в соответствии с входными данными и принципами алгоритма обучения самоорганизующейся карты Кохонена.

2. ***Выбор образца данных***. Главная цель этого шага – выбрать случайный образец данных из обучающей выборки для дальнейшего сопоставления с нейронами на карте.

Обычно образец данных выбирается случайным образом из доступного набора данных. Это может быть случайный элемент из обучающего набора данных, выбранный равномерно или случайным образом с заданными вероятностями.

Например, если имеется набор данных в виде матрицы, то выбор образца может быть выполнен как выбор случайной строки из матрицы для текущей

итерации. Другим примером может быть выбор случайного изображения из набора изображений.

Выбор образца данных происходит для каждой итерации обучения, повторяя этот процесс, пока не пройдет заданное число итераций или не будет достигнуто условие остановки. Каждый выбранный образец данных затем используется для обновления весов нейронов на карте и позволяет картам Кохонена соответствовать структуре данных.

3. ***Выбор победителя***. Расстояние между входным образцом и весами каждого нейрона вычисляется, и выбирается нейрон с наименьшим расстоянием (наиболее близкий к входному образцу). Этот нейрон называется «победителем».

Для вычисления расстояния между входным образцом и весами каждого нейрона на самоорганизующейся карте Кохонена обычно используется евклидово расстояние. Однако в зависимости от задачи и требований могут использоваться и другие метрики расстояния. Формула для вычисления евклидового расстояния между векторами u и v длиной n выглядит следующим образом:

$$d(u,v) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}, \quad (1)$$

где $u[i]$, $v[i]$ – i-е элементы векторов u и v.

Другие часто используемые метрики расстояния включают в себя:

- ✓ Манхэттенское расстояние (или L1-норма), которое вычисляется как сумма абсолютных разностей между координатами векторов:

$$d(u,v) = \sum_{i=1}^{n} |u_i - v_i|, \quad (2)$$

- ✓ Косинусное расстояние:

$$d(u,v) = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}} \quad (3)$$

Выбор метрики расстояния зависит от характеристик данных и целей задачи. Обычно для расчета расстояний используются готовые функции или методы, предоставляемые программными библиотеками для работы с векторными операциями или машинным обучением.

4. **Обновление весов**. Веса победителя и его соседей обновляются таким образом, чтобы они становились более похожими на входной образец. Это позволяет картам Кохонена адаптироваться к структуре данных и организовывать схожие образцы рядом друг с другом на карте.

Обновление весов в самоорганизующейся карте Кохонена осуществляется после выбора победителя и его ближайших соседей. Этот процесс позволяет нейронам адаптироваться к входным данным и организовывать себя на карте более эффективно.

Процесс обновления весов нейронов может быть описан следующим образом:

1) Определение радиуса окрестности: на основе текущей итерации обучения и параметров, таких как скорость обучения, определяется радиус окрестности вокруг победителя, в которой будет происходить обновление весов. Радиус окрестности обычно начинается с некоторого начального значения и уменьшается по мере продвижения обучения.
2) Обновление весов победителя и его соседей. Веса победителя и его ближайших соседей обновляются на основе формулы, которая обычно учитывает расстояние между нейронами и входным образцом данных. Обновление весов происходит таким образом, чтобы более похожие нейроны были более привлекательными для входных данных и постепенно приближались к ним.
3) Обновление весов остальных нейронов. Веса остальных нейронов на карте также обновляются в соответствии с их расстоянием от победителя. Чем ближе нейрон к победителю, тем больше он будет обновлять свои веса.

Обновление весов осуществляется в каждой итерации обучения и повторяется для каждого входного образца данных. Постепенно, на основе локальной конкуренции и сотрудничества между нейронами, самоорганизующаяся карта Кохонена эффективно организует и группирует данные на карте.

5. **Уменьшение шага обучения**. По мере продвижения обучения, шаг обновления весов уменьшается, позволяя сети сходиться к устойчивым состояниям.

Уменьшение шага обучения в самоорганизующейся карте Кохонена является важным процессом, который происходит по мере продвижения обучения. Это позволяет более медленно обновлять веса нейронов по мере приближения к стабильным состояниям карты.

Обычно уменьшение шага обучения осуществляется постепенно и зависит от номера текущей итерации обучения. Чаще всего применяются два подхода для уменьшения шага обучения:

1) Линейное уменьшение. Шаг обучения уменьшается линейно по мере продвижения итераций. Например, на каждой итерации шаг обучения может быть уменьшен на фиксированную величину или в масштабируемом показателе, предназначенном для учета скорости сходимости.
2) Экспоненциальное уменьшение. Шаг обучения экспоненциально уменьшается с увеличением числа итераций или эпох обучения. Это может быть реализовано, например, с использованием экспоненциального коэффициента снижения, который определяет скорость уменьшения шага обучения.

Уменьшение шага обучения позволяет улучшить стабильность и сходимость обучения. По мере приближения к конечному состоянию карты Кохонена, уменьшение шага обучения помогает избежать сильных флуктуаций весов и способствует более плавному обновлению. Это особенно важно при работе с большими объемами данных или сложными пространствами признаков.

После завершения обучения карта Кохонена может быть использована для кластеризации и визуализации данных. Близкие нейроны на карте обозначают похожие образцы, а удаленные нейроны представляют различные образцы.

Самоорганизующаяся карта Кохонена относится к классу нейронных сетей без учителя и часто применяется в задачах анализа данных, обработки изображений, компьютерного зрения, кластеризации и картирования. Она позволяет нам получить интуитивное понимание сложных данных и их структуры.

2. Использование нейронных сетей в системах WAF

Анализ трафика веб-приложения в WAF может быть выполнен несколькими способами:

1. Разбор и обработка заголовков. WAF анализирует заголовки HTTP-запросов и ответов, чтобы получить информацию о клиенте, сервере, используемых методах запроса, типах содержимого и дополнительных параметрах. Это позволяет WAF распознавать типичные признаки атак или потенциально опасные конфигурации.
2. Парсинг запросов. WAF разбирает содержимое HTTP-запросов, чтобы достать параметры, URL-адреса, куки и другую информацию. Это помогает определить, с помощью каких параметров пользователь обращается к веб-приложению и выявить потенциально опасные или злонамеренные значения.
3. Анализ содержимого. WAF анализирует тело HTTP-запросов и ответов, чтобы распознать злонамеренный код, вредоносные скрипты или другие опасные содержимые. Для этого может быть применен алгоритм обнаружения аномалий, сигнатурное сопоставление или даже применение машинного обучения для определения необычного поведения и атак.
4. Фильтрация данных. WAF может применять предопределенные фильтры и правила для проверки параметров запросов и данных на соответствие безопасности. Например, он может блокировать запросы, содержащие SQL-инъекции, XSS-атаки или другие уязвимости. Фильтрация может осуществляться с использованием регулярных выражений, ключевых слов или других подходов.
5. Обнаружение и предотвращение атак. С помощью методов обучения, анализа аномалий или сигнатурное соответствие, WAF может обнаруживать и предотвращать известные и неизвестные атаки на веб-приложение. Это может включать обнаружение SQL-инъекций, подбора паролей, кросс-сайтовой сценарной атаки и других видов атак.

Анализ трафика WAF происходит в реальном времени, поэтому система должна быть очень производительной и быстрой для эффективной защиты веб-приложения от угроз безопасности и атак [5].

WAF служит важным инструментом для обеспечения безопасности веб-приложений, предотвращения атак и обнаружения уязвимостей. Он работает как дополнительный слой защиты, помогающий предотвратить утечку данных, повреждения сайта и другие проблемы безопасности [6].

Карту Кохонена можно использовать в системах WAF для обнаружения и предотвращения атак на веб-сайты. Вот несколько способов [7]:

1. Обнаружение аномального трафика. Карта Кохонена может быть обучена на нормальном трафике веб-сайта, чтобы выявить типичные образцы поведения пользователей. Затем, в реальном времени, она может анализировать входящий трафик и идентифицировать аномальное поведение, которое может указывать на попытку атаки. Например, если входящий запрос отличается от типичных запросов, нейронная сеть может сигнализировать о возможном взломе или атаке на сайт.
2. Отслеживание и обнаружение атак. Карта Кохонена может быть обучена различать образцы трафика, связанные с известными атаками, такими как SQL-инъекции, XSS, CSRF и другие. Она может быть настроена на обнаружение характеристических признаков, связанных с такими атаками, и предотвращение их выполнения или блокировку соответствующего входящего трафика.
3. Защита от ботов и сканеров уязвимостей. Карта Кохонена может помочь в обнаружении ботов и сканеров уязвимостей, которые могут искать уязвимости веб-сайта для дальнейших атак. Используя способность к распознаванию образцов и аномалий в трафике, нейронная сеть может помочь в отслеживании и блокировке подозрительных активностей.
4. Анализ логов и инцидентов. Нейронная сеть может быть использована для анализа логов событий веб-приложения и обнаружения аномалий, нетипичного поведения или паттернов, связанных с безопасностью. Она может помочь в обнаружении новых уязвимостей, атак или аномальных событий, которые могут оставаться незамеченными при использовании традиционных методов анализа логов.

Важно отметить, что нейронная сеть должна быть обучена на репрезентативных данных и регулярно обновляться, чтобы адаптироваться к изменяющимся угрозам. Она может использоваться в сочетании с другими методами обнаружения и защиты, образуя комплексную систему безопасности веб-приложений [8, 9].

3. Применение нейронной сети Кохонена для выявления атак на веб-приложение

На рисунке 1 изображена схема работы WAF, построенного на микросервисной архитектуре и использующего нейронную сеть Кохонена для выявления атак [10, 11]. Клиент передает запрос на внешний веб-сервер NGINX. NGINX средствами модуля NJS направляет запрос модулю анализа SOM и целевому веб-приложению. Веб-приложение возвращает ответ веб-серверу NGINX. NGINX средствами модуля NJS направляет ответ модулю анализа SOM и клиенту. Модуль анализа SOM выполняет две функции: сообщает о результатах анализа аналитику в режиме реального времени и записывает запросы, ответы и результаты анализа в базу данных Storage [12, 13].

NJS (Ngx JavaScript) – это встроенный язык программирования, который используется в веб-сервере nginx для написания дополнительной логики обработки запросов и управления конфигурацией сервера. NJS базируется на языке JavaScript и предоставляет возможность расширения функциональности сервера nginx путем выполнения JavaScript-кода на этапе обработки запросов [14].

Первичное обучение происходит по данным, полученным от клиентов без анализа и фильтрации. Переобучение реализовано через базу данных. То есть нейронная сеть в момент инициализации подключается к базе данных, получает список запросов, помеченных как легитимные, затем обучается на них и анализирует входящий от клиентов трафик.

Описанная схема WAF не подразумевает активную защиту, то есть блокировку запросов и ответов, так как нейронные сети имеют высокую вероятность ложных срабатываний, что может повлечь негативное влияние на бизнес-процессы.

Модуль анализа SOM представляет из себя приложение, написанное на языке программирования Golang, реализующее работу нейронной сети Кохонена [15]. Каждый модуль схемы рисунка 1 – отдельное программное средство, запущенное в контейнере docker [16]. Приложение строит нейронную сеть со следующими параметрами:

- ✓ тип сетки – плоская;
- ✓ топология – шестиугольник;
- ✓ размер сетки – 20 на 20;
- ✓ инициализация весов производится значениями типа int, верхний предел – 500;
- ✓ выбор победителя осуществляется по евклидовому расстоянию;
- ✓ функция, используемая для обновления весов – Гауссова;
- ✓ уменьшение шага обучения – экспоненциальное [17].

Для проекции запросов к веб-приложению на числовой вектор были выбраны метрики, которые могут кардинально отличаться, если сравнить легитимный и вредоносный запросы. Это неполный список метрик, их можно расширить, однако в эксперименте брались именно эти:

- ✓ количество букв в поле запроса;
- ✓ количество цифр в поле запроса;
- ✓ количество специальных символов в поле запроса;
- ✓ общее количество символов;
- ✓ энтропия запроса.

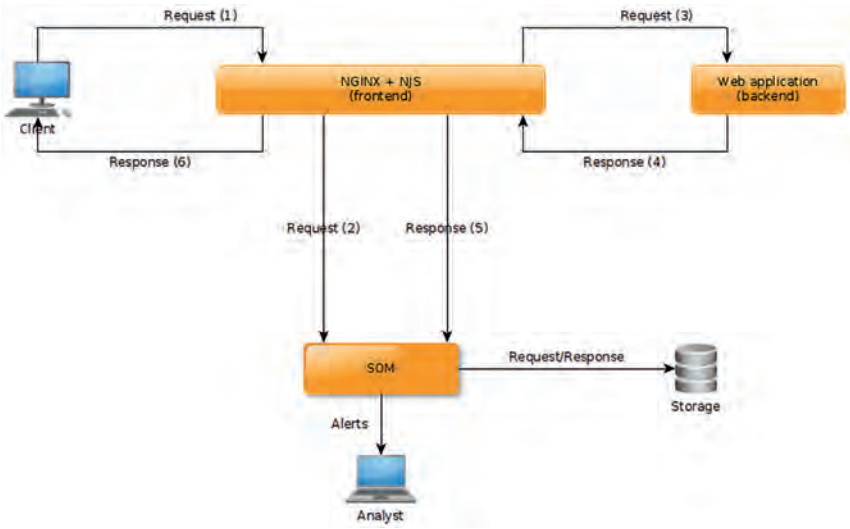

Рис. 1. Схема работы WAF, использующего нейронную сеть Кохонена

Модуль анализа SOM работает по следующему алгоритму:

1. Получает из базы или в режиме реального времени запросы клиентов к веб-приложению;
2. По запросам вычисляет необходимые для анализа метрики, формирует числовые векторы;
3. Нейронная сеть обучается на полученных векторах, в результате получаем карту Кохонена и эталонную ошибку квантования.
4. Для каждого запроса клиента в режиме реального времени вычисляется ошибка квантования. Если они выше эталонной, то запрос считается нелегитимным.

Ошибка квантования в самоорганизующейся карте Кохонена является показателем расстояния между входным образцом данных и вектором весов победившего нейрона на карте. В процессе обучения карты Кохонена каждый входной образец данных сопоставляется с ближайшим нейроном на карте, который называется победителем. Ошибка квантования измеряет расстояние между входным образцом и вектором весов победившего нейрона в пространстве признаков. Чем ниже ошибка квантования, тем более точно карта Кохонена отображает и классифицирует входные данные. Уменьшение ошибки квантования является одной из задач обучения нейронной сети и зависит от эффективности алгоритма обновления весов в процессе обучения. Ошибка квантования является метрикой, которая помогает оценить эффективность обучения нейронной сети и может использоваться для выбора наилучшей конфигурации нейронной сети или для определения условия остановки обучения. Если ошибка квантования выше эталонной, то входные данные являются нелегитимными.

Для тестирования выбрано веб-приложение Mutillidae. Mutillidae – это веб-приложение с открытым исходным кодом, разработанное OWASP (Open Web Application Security Project) в качестве целей для тестирования и практики навыков эксплуатации уязвимостей. Оно представляет собой настраиваемое и легко устанавливаемое окружение, содержащее различные типы уязвимостей, обнаруженных в реальных веб-приложениях. Mutillidae создано с целью обучить и эмулировать реальные среды уязвимых веб-приложений, чтобы разработчики, тестировщики и исследователи могли изучить и практиковать методы обнаружения и решения уязвимостей, таких как SQL-инъекции, XSS, подделка запросов между сайтами (CSRF) и другие. Mutillidae широко используется в обучающих курсах и тренировках, связанных с безопасностью приложений и тестированием уязвимостей. Это полезный ресурс для тех, кто хочет получить опыт в обнаружении и эксплуатации уязвимостей в веб-приложениях [18].

Генератором легитимных запросов выступил модуль обхода веб-приложения сканера безопасности OWASP ZAP. Генератором вредоносных запросов выступил модуль тестирования на проникновение сканера безопасности OWASP ZAP. Для простоты в качестве входных данных взяты только запросы типа GET, поле URN [19].

OWASP ZAP (Zed Attack Proxy) – это инструмент для тестирования безопасности веб-приложений с открытым исходным кодом. Он разработан и поддерживается сообществом Open Web Application Security Project (OWASP) и предоставляет возможности для обнаружения и эксплуатации уязвимостей в веб-приложениях [20].

Результаты тестирования представлены в таблице 1. Выборка запросов недостаточно большая для эксплуатации в промышленной среде, но с учетом однородности запросов выводы об эффективности подхода носят объективный характер.

Высокая нагрузка на CPU обусловлена использованием одного ядра многоядерного процессора. По факту нагрузку нельзя считать высокой. При этом оперативная память практически не нагружена. Это связано с тем, что модуль держит только в оперативной памяти только сеть Кохонена 20 на 20.

Таблица 1.

Результаты тестирования карты Кохонена

Нагрузка на CPU, %	100
Нагрузка на оперативную память, %	<1
Число запросов, на которых обучалась нейронная сеть	1450
Число проанализированных запросов (все запросы считаются вредоносными)	7331
Среднее время анализа одного запроса, микросекунды	113
Потенциальная скорость обработки запросов, запросы в секунду	8849
Процент ошибок, %	12,5

Вывод

Самоорганизующаяся карта успешно выполняет задачу анализа HTTP трафика на наличие векторов атак и попыток эксплуатации уязвимостей. Необходимые вычислительные мощности для работы анализатора оказались низкими, что является преимуществом, так как это позволяет снизить затраты на вычислительный процесс. В целом, результаты исследования указывают на эффективность и перспективность использования самоорганизующейся карты Кохонена для защиты веб-приложений.

Чтобы повысить эффективность анализа трафика и улучшить точность обнаружения атак, рекомендуется использовать нейронную сеть совместно с шаблонным анализом. Комбинируя шаблонный анализ с нейронной сетью, можно достичь более надежного обнаружения атак и уменьшить число ложных срабатываний. Такой подход значительно повышает безопасность системы и защищает веб-приложения от широкого спектра атак. Таким образом, рекомендуется использование шаблонного анализа трафика в сочетании с самоорганизующейся картой Кохонена для достижения наилучших результатов в обнаружении атак и защите веб-приложения.

Литература

1. *Clincy, V. Web Application Firewall: Network Security Models and Configuration. / V. Clincy, H. Shahriar // IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). – 2018. – P. 835-836. – DOI 10.1109/COMPSAC.2018.00144.*
2. *Коллинз, М. Защита сетей. Подход на основе анализа данных / Майкл Коллинз; пер. с англ. А. В. Добровольская. – М.: ДМК Пресс, 2020. – 308 с.: ил. – ISBN 978-5-97060-649-0.*
3. *Остроух, А. В. Системы искусственного интеллекта / А. В. Остроух, Н. Е. Суркова. – 3-е изд., стер. – Санкт-Петербург: Лань, 2023. – 228 с. – ISBN 978-5-507-46441-8.*
4. *Dogo, E. Sensed Outlier Detection for Water Monitoring Data and a Comparative Analysis of Quantization Error Using Kohonen Self-Organizing Maps. / E. Dogo, N. Nwulu, B. Twala, C. Aigbavboa // International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS) – 2018. – DOI 10.1109/COMPSAC.2018.00144.*
5. *Брюхомицкий, Ю. А. Искусственные иммунные системы в информационной безопасности [Текст] / Ю. А. Брюхомицкий; Южный федеральный университет. –Таганрог: Издательство Южного федерального университета, 2019. – 147 с. – ISBN 978-5-9275-3212-4.*
6. *Чио, К. Машинное обучение и безопасность / Кларенс Чио, Дэвид Фримэн; пер. с анг. А. В. Снастина. – М.: ДМК Пресс, 2020. – 388 с.: ил. – ISBN 978-5-97060-713-8.*
7. *Arul, E. Firmware Attack Detection on Gadgets Using Kohonen's Self Organizing Feature Maps (KSOFM). / E. Arul, P. Angusamy // Third International Conference on Smart Systems and Inventive Technology (ICSSIT). – 2020. – P. 21–26. – DOI 10.1109/ICSSIT48917.2020.9214115.*
8. *Zhukovytskyy, I. Study of Combined Approach Possibilities to Detecting Network Attacks Using Artificial Intelligence Mechanisms. / I. Zhukovytskyy, V. Pakhomova, I. Tsykalo, D. Bikovska // 12th International Conference on Dependable Systems, Services and Technologies (DESSERT). – 2022. – P. 1–4. – DOI: 10.1109/DESSERT58054.2022.10018718*
9. *Belej, O. Using Hybrid Neural Networks to Detect DDOS Attacks. / O. Belej, L. Halkiv // IEEE Third International Conference on Data Stream Mining & Processing (DSMP). – 2020. – DOI: 10.1109/DSMP47368.2020.9204166.*
10. *Дэвис, К. Шаблоны проектирования для облачной среды / Корнелия Дэвис; пер. с анг. Д. А. Беликова. – М.: ДМК Пресс, 2020. – 388 с.: ил. – ISBN 978-5-97060-807-4.*
11. *Белл, Л. Безопасность разработки в Agile-проектах: Обеспечение безопасности в конвейере непрерывной поставки / Л. Белл, М. Брантон-Сполл, Р. Смит, Д. Бэрд ; пер. с англ. А. А. Слинкин.– М.: ДМК Пресс, 2018.– 448с. :ил. – ISBN 978-5-97060-648-3.*
12. *Айвалиотис, Д. Администрирование сервера NGINX / Д. Айвалиотис .– Москва : ДМК Пресс, 2018 .– 289 с. – ISBN 978-5-97060-610-0.*
13. *Дерек де Йонге. NGINX. Книга рецептов / Дерек де Йонге; пер. с англ. Д. А. Беликова.– М.: ДМК Пресс, 2020.– 176с.: ил. – ISBN 978-5-97060-790-9.*
14. *Muzaki, R. A. Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall / R. Muzaki, B. Obrina, M. Hasditama, H. Ritchi // International Workshop on Big Data and Information Security (IWBIS). – 2020. – P. 5–16. – DOI 10.1109/IWBIS50925.2020.9255601.*
15. *Батчер, М. Go на практике [Текст] / Мэтт Батчер, Мэтт Фарина; пер. с англ. Р. Н. Рагимова; науч. ред. А. Н. Киселев. – М.: ДМК Пресс, 2017. – 374 с.: ил.– ISBN 978-5-97060-477-9 (рус.). – ISBN 978-1-63343-007-5 (анг.).*
16. *Милл, Иан. Docker на практике / Иан Милл, Эйдан Хобсон Сейерс; пер. с англ. Д.А. Беликов. – М.: ДМК Пресс, 2020.– 516 с.: ил. – ISBN 978-5-97060-772-5.*
17. *Омельяненко. Эволюционные нейросети на языке Python / Омельяненко. – Москва : ДМК Пресс, 2020.– 311 с.*
18. *Эдриан Прутяну. Как стать хакером: Сборник практических сценариев, позволяющих понять, как рассуждает злоумышленник / Эдриан Прутяну; пер. с англ. Д. А. Беликова – М.: ДМК Пресс, 2020.– 380 с.: ил. – ISBN 978-5-97060-802-9.*
19. *Shubham, L. Secure Web development using OWASP Guidelines. / L. Shubham, Kumar A., Dr. T. Subbulakshmi // 5th International Conference on Intelligent Computing and Control Systems (ICICCS). – 2021. – P. 323–332. – DOI 10.1109/ICICCS51141.2021.9432179.*
20. *Fredj, O. An OWASP Top Ten Driven Survey on Web Application Protection Methods. / O. Fredj, O. Cheikhrouhou, M. Krichen, H. Hamam, A. Derhab // Risks and Security of Internet and Systems. – 2021. – P. 235–252. – DOI 10.1007/978-3-030-68887-5_14.*