ПАТТЕРН ДЛЯ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ ПРИ УГРОЗЕ НЕКОНТРОЛИРУЕМОГО РОСТА ЧИСЛА ЗАРЕЗЕРВИРОВАННЫХ РЕСУРСОВ

Корнеев Н. В.[1], Трубачева-Гудович А. Е.[2]

DOI: 10.21681/2311-3456-2025-4-35-45

Цель статьи: *разработка паттерна для веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя.*

Метод исследования: *анализ принципов проведения DDoS-атак. Синтез сценариев DDoS-атак по трем видам атак: транспортного уровня, уровня инфраструктуры, уровня приложений. За основу выбран сценарий обеспечения безопасности веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя. Предложен новый механизм защиты, обеспечивающий переадресацию и проверку пользователя на специальном наборе задач и дальнейшую балансировку его запроса к веб-приложению методом IP Hash. Исследование выполнено путем натурного моделирование веб-приложения на основе Docker в средах с поддержкой контейнеризации, его развёртывания и тестирования.*

Результат: *проведен анализ угрозы неконтролируемого роста числа зарезервированных ресурсов и показана актуальность проблемы разработки универсальных шаблонных механизмов безопасности, называемых паттернами. В частности рассмотрены сценарии DDoS-атак на веб-приложения. Предложен сценарий обеспечения безопасности веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя. Построена микросервисная архитектура для обеспечения безопасности веб-приложения. Разработан паттерн для веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя на основе микросервисов, интегрированных в контейнеры. В рамках проведённого исследования был разработан сервис проверки пользователей на языке JavaScript, с виртуализацией на базе Docker, и с балансировщиком нагрузки nginx. Механизм защиты реализован следующим образом. Пользователь перед заходом в веб-приложение перенаправляется на страницу, где требуется выполнить определенную задачу: решить математический пример, распознать правильным образом символы, распознать правильным образом графические объекты. При успешном решении задач пользователь перенаправляется на веб-приложение, проходя перед этим через один из трех балансировщиков нагрузки, использующих метод IP Hash. Разработан программный код сервиса проверки пользователей, включая коды специальных методов и алгоритмы для трех указанных выше задач. Проведено тестирование паттерна безопасности веб-приложения на базе Grafana k6. Разработан программный код теста test.js с реализованным сценарием тестирования, который включает в себя три этапа с различными уровнями нагрузки. В тесте участвовало до 20 виртуальных пользователей одновременно, с постепенным увеличением нагрузки. В результате тестирования не было зафиксировано ни одного сбоя запросов, все 4816 запросов были успешными – это свидетельствует о стабильной работе паттерна безопасности веб-приложения.*

Практическая ценность: *практическая значимость предлагаемого решения включает паттерн для веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя, который можно применить для широкого круга веб-приложений.*

Ключевые слова: *шаблон, DDoS-атака, ботнет, сервис проверки пользователей, балансировщик нагрузки, метод IP Hash, задача распознавания символов, задача распознавания графических объектов, математическая задача, контейнер, тестирование.*

Введение

Изменения в современном ландшафте киберугроз в настоящее время требуют внедрения современных информационно-коммуникационных технологий особенно в сферу информационной безопасности. Применение таких технологий связано с целым комплексом проблем в которых можно выделить сферу компьютерных преступлений. Можно выделить отдельные проблемы соответствующие этой сфере несанкционированный доступ, финансовое мошенничество, подделка сайта, атаки на финансовую инфраструктуру, нарушение работы компьютерных систем, DDoS-атаки [1–7]. Далее в работе мы сконцентрируем свое внимание на характерных особенностях DDoS-атак [7].

Атака типа «распределенный отказ в обслуживании» (DDoS, Distributed Denial of Service attack) является одной из основных атак, от которой страдает организационная группа безопасности. DDoS-атаки

1 Корнеев Николай Владимирович, доктор технических наук, доцент, РГУ нефти и газа (НИУ) имени И. М. Губкина, Москва, Россия. E-mail: niccyper@mail.ru
2 Трубачева-Гудович Анна Евгеньевна, студент РГУ нефти и газа (НИУ) имени И. М. Губкина, Москва, Россия. E-mail: gudovich.an@bk.ru

напрямую нацелены на доступность услуг организации-жертвы [7]. Они могут привести к целому ряду неблагоприятных последствий, которые могут варьироваться от сбоев в обслуживании, споров за ресурсы, ущерба репутации и финансовых потерь до различных расходов, связанных с усилиями по восстановлению. DDoS-атаки становятся все более контролируемыми и изощренными, поскольку злоумышленники постоянно меняют масштаб и модели своих атак [7–10]. Так злоумышленники отправляют огромное количество бессмысленных пакетов или злонамеренно потребляют ресурсы сетевого канала [11], в этом проявляется угроза неконтролируемого роста числа зарезервированных ресурсов.

В последние годы наблюдается значительное увеличение инцидентов DDoS-атак, о чем говорит отчет специалистов компании Positive Technologies за 2023 год. Существенный объем инцидентов связан с деятельностью хактивистов [12], включающую массированные DDoS-атаки и дефейс сайтов [13].

В настоящее время исследователи активно изучают и публикуют работы касающиеся безопасности от DDoS-атак, например [7–11]. Существенным пустым местом в большинстве работ исследователей остается разработка универсальных шаблонных механизмов безопасности, называемыми паттернами. В частности в данной статье мы ставим целью разработку такого паттерна для обеспечения безопасности веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов.

Паттерн безопасности описывает конкретную повторяющуюся проблему безопасности, которая возникает в определенных известных контекстах, а также предлагает хорошо зарекомендовавшую себя общую схему решения такой проблемы безопасности.

Согласно банку угроз безопасности информации ФСТЭК России, такая угроза безопасности информации называется УБИ 059. Угроза заключается в возможности отказа легальным пользователям в выделении компьютерных ресурсов после осуществления нарушителем неправомерного резервирования всех свободных компьютерных ресурсов (вычислительных ресурсов и ресурсов памяти). Данная угроза обусловлена уязвимостями программного обеспечения уровня управления виртуальной инфраструктурой, реализующего функцию распределения компьютерных ресурсов между пользователями. Реализация данной угрозы возможна при условии успешного осуществления нарушителем несанкционированного доступа к программному обеспечению уровня управления виртуальной инфраструктурой, реализующему функцию распределения компьютерных ресурсов между пользователями.

Анализ и методы исследования

К рассматриваемой угрозе относится распределенный отказ в обслуживании, в нашем случае веб-приложения. Данный вид атаки впервые стал известен в конце 1990-х годов. Даже сейчас он является одной из самых больших угроз для любой организации, ведущей бизнес в Интернете [7]. Это способ атаки на сетевую инфраструктуру, включая веб-сайты и онлайн-приложения, путем перегрузки хост-серверов. Это не позволяет законным пользователям получить доступ к услугам. Термин «распределенный» относится к тому, что эти атаки неизменно осуществляются с большого количества скомпрометированных компьютеров или устройств, например «сеть зомби» или ботнет – сеть компьютеров, зараженная вредоносным программным обеспечением. Цель данной атаки состоит в том, чтобы нарушить нормальную работу приложения или сайта, чтобы посетителю казалось, что он находится в автономном режиме.

DDoS-атаки работают по принципу перегрузки сервиса большим объемом запросов. Принцип работы данной атаки изображен на (рис. 1). Злоумышленник создает или покупает достаточно большую «сеть зомби» или ботнет, которые перегружают веб-приложение большим объемом запросов (связь красного цвета), по команде от самого злоумышленника (связь черного цвета) или через активацию такой команды в ботнет на советующих ПК, чтобы уничтожить цель, в нашем случае – веб-приложение на основе веб-сервера nginx. Традиционно «сеть зомби» или ботнет состоит из потребительских или бизнес-ПК, включенных в сеть с помощью вредоносного ПО. В последнее время в ботнет стали включать устройства Интернета вещей, что существенно увеличило количество атак [10], а сами атаки стали комплексными и способными вывести из строя экосистему организации.

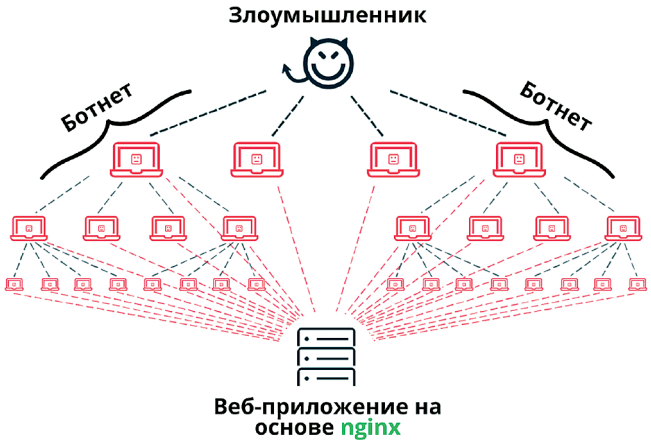

Рис. 1. Сценарий DDoS-атаки с использованием ботнет

Различают несколько видов DDoS-атак:

1. Атаки транспортного уровня. Реализация включает комплекс действий атакующего по перегрузке брандмауэра, сетевой инфраструктуры, включающей подсистемы для распределения нагрузки. Характерной особенностью реализации таких атак является использование сетевого флуда. Сам сетевой флуд – это поток пустых запросов, генерация которых производится постоянно, за счет чего – канал перегружается. Метод взаимодействия формируется за счет клиентских запросов которые направляются к серверу по метод у FIFO (First In, First Out). Этот метод подразумевает последовательную обработку запросов по принципу – первый запрос пришел, он же первый вышел с сервера. Поток пустых запросов, генерация которых производится постоянно при сетевом флуде, вынуждает аппаратные ресурсы сервера функционировать на пределе своих возможностей, и в результате их не хватает даже для завершения обработки первого запроса.

HTTP-флуд. Сценарий такой атаки изображен на (рис. 2). Веб-приложение на основе nginx размещено на сервере который принимает значительное количество HTTP-запросов от пользователей. Однако помимо реальных пользователей запросы генерируют боты. Таким образом сервер перегружается значительным количеством запросов, а веб-приложение на основе nginx, размещённое на сервере, фактически не может обработать запросы от реальных пользователей по причине перегрузки избыточным объемом запросов, которые генерирую боты.

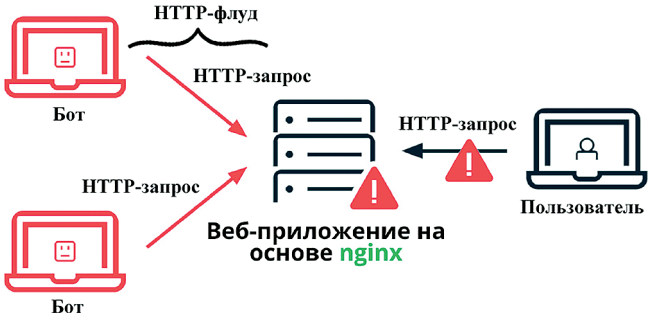

Рис. 2. Сценарий атаки HTTP-флуд

ICMP-флуд. Злоумышленник через бота осуществляет атаку (рис. 3). Суть атаки сводится к целенаправленной загрузке на сервер специальных команд. Для типовой атаки могут быть использованы служебные команды. Каждая такая команда – эхо-запрос. В свою очередь на каждый эхо-запрос сервер должен дать эхо-ответ. В результате сервер перегруженный такими эхо-запросами перестает нормально функционировать, а реальный пользователь не может получить ответ на свой запрос.

Рис. 3. Сценарий атаки ICMP-флуд

SYN-флуд. Сценарий такой атаки изображен на (рис. 4). Суть атаки связана с понятиями SYN-запрос и флагами SYN (Synchronization) и ACK (Acknowledgement). SYN-запрос – это запрос на подключение по протоколу TCP. Пользователь устанавливает флаг SYN посылая на сервер свой пакет, а сервер возвращает пакет с флагами SYN и ACK. Затем пользователь отправляет пакет с флагом ACK, после чего и пользователь и сервер готовы к передаче данных. Такая последовательность действий называется алгоритмом «тройного рукопожатия». В случае с SYN-флудом роль злоумышленника выполняет бот, «сеть зомби» или ботнет. Они перегружают сервер такими запросами и в результате очередь пакетов на сервере переполняется. Дополнительно бот, «сеть зомби» или ботнет подделывают пакеты используя в заголовках служебные команды перенаправляющие пакеты от сервера на вредоносные ссылки. Таким образом каждый бот создает сеть вредоносных ссылок, и в результате сервер перегруженный такими запросами перестает нормально функционировать. Реальный же пользователь в этой ситуации не имеет возможности получить доступ к веб-приложению.

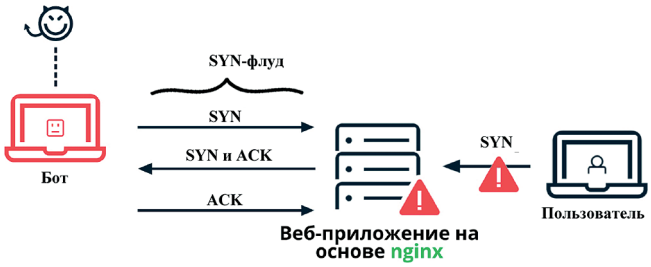

Рис. 4. Сценарий атаки SYN-флуд

UDP-флуд. Сценарий такой атаки изображен на (рис. 5). Веб-приложение на основе nginx размещено на сервере который принимает значительное количество UDP-запросов от пользователей. Однако помимо реальных пользователей запросы генерируют злоумышленники. Здесь, как и предыдущем случае, роль злоумышленника выполняет бот, «сеть зомби» или ботнет, образуя таким образом паразитную сеть. Задача которую они решают сводится к перегрузке полосы пропускания сервера. Таким

образом сервер перегружается значительным количеством запросов UDP, а веб-приложение на основе nginx, размещённое на сервере, фактически не может обработать запросы от реальных пользователей по причине перегрузки избыточным объемом запросов, которые генерирую боты. Дополнительно бот, «сеть зомби» или ботнет подделывают пакеты используя подложный адрес источника инициатора запроса. Таким образом сообщения ICMP с отказами в обслуживании перенаправляются на другие сервера, а ботнет продолжает перегрузки сервера избыточным объемом запросов и реальный пользователь не может получить ответ на свой запрос.

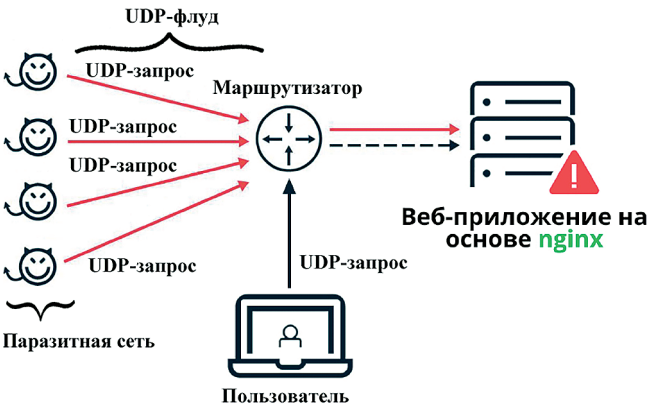

Рис.5. Сценарий атаки UDP-флуд

MAC-флуд. Веб-приложение размещено на сервере который принимает значительное количество пустых пакетов от пользователей. Однако помимо реальных пользователей запросы генерируют боты. Здесь, как и предыдущем случае, роль злоумышленника выполняет бот, «сеть зомби» или ботнет, образуя таким образом паразитную сеть. Задача которую они решают – вызвать перегрузку полосы пропускания сервера за счет пустых пакетов с пустыми MAC-адресами, которые генерируют боты. Реальный же пользователь в этой ситуации не имеет возможности получить доступ к веб-приложению.

2. Атаки уровня инфраструктуры. В настоящее время топология таких атак постоянно расширяется. Она увязывается с комплексом инфраструктурных элементов, в который входят всевозможные элементы микро и макро архитектуры, например подсистема взаимодействия с оперативной памятью, подсистема межпроцессорного взаимодействия, подсистема хранения данных. Главной особенностью атаки уровня инфраструктуры является факт отсутствия перегрузки канала связи.

Атака на вычислительные ресурсы. Эта атака актуальна для систем которые оперируют с большими данными. Перегрузка ядер процессора, как следствие значительного количества запросов от бота, «сети зомби» или ботнет на выполнение сложных вычислительных алгоритмов, вот яркий пример атаки на вычислительные ресурсы. В результате сервер перегруженный такими запросами перестает нормально функционировать. Реальный же пользователь в этой ситуации не имеет возможности получить доступ к веб-приложению.

Атака на дисковое пространство. Веб-приложение размещено на сервере который принимает значительное количество запросов от пользователей с заведомо мусорными данными. Однако помимо реальных пользователей запросы генерируют боты, используя для этого вредоносный код. Здесь, как и в предыдущем случае, роль пользователя выполняет бот, «сеть зомби» или ботнет, образуя таким образом паразитную сеть. Задача которую они решают сводится к переполнению дискового пространства мусорными данными, включающими в себя файлы логов, а также все то, что используется для активного взаимодействия с файловой системой.

Обход системы квотирования. Здесь, как и в предыдущем случае бот, «сеть зомби» или ботнет используя вредоносный код получает доступ к интерфейсу сервера (CGI, Common Gateway Interface). Далее паразитная сеть получает доступ к аппаратной части сервера и тот перестает нормально функционировать. Реальный же пользователь в этой ситуации не имеет возможности получить доступ к веб-приложению.

Неполная проверка пользователя. Отдельный злоумышленник, бот, «сеть зомби» или ботнет использует ресурсы сервера, так как на сервере не реализован современный механизм проверки пользователя. В результате эксплуатация такой уязвимости может проходить бесконечно долго.

Атака второго рода. Веб-приложение размещено на сервере который принимает значительное количество запросов от пользователей. В отдельный момент злоумышленник, бот, «сеть зомби» или ботнет используя вредоносный код вызывает ложный сигнал о перегрузке. Далее паразитная сеть может дополнить сигнал реальной угрозой перегрузки сервера и тот перестает нормально функционировать. Реальный же пользователь в этой ситуации не имеет возможности получить доступ к веб-приложению.

3. Атаки уровня приложений. Атаки этого уровня эксплуатируют уязвимости серверного ПО. Переполнение буфера памяти, как следствие значительного количества ICMP-пакетов от бота, «сети зомби» или ботнет направляемые на сервер, вот яркий пример атаки уровня приложений. В результате сервер перегруженный такими пакетами перестает нормально функционировать, а реальный пользователь не имеет возможности получить доступ к веб-приложению.

Для ограничения зоны применимости разрабатываемого шаблонного механизма защиты определим, что нами будет рассматривается сценарий обеспечения безопасности веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя. Выбор данной атаки обусловлен ее широкой распространенностью для любых веб-приложений и отсутствием у большинства из них современного механизма проверки пользователя, который является по сути своей базовым механизмом доступа к веб-приложению, и позволяет злоумышленнику использовать ресурсы сервера бесконечно долго, в том числе, в обход действующих систем защиты, таким образом истощая ресурсы сервера. В то же время, выбор обусловлен необходимостью постоянного совершенствования механизмов проверки пользователя, с целью отражения все более совершенных атак.

Для отражения таких совершенных атак необходимо реализовать паттерн безопасности, включающий в себя механизм защиты веб-приложения (рис. 6). Механизм защиты будет реализован следующим образом. Пользователь перед заходом в веб-приложение перенаправляется на страницу, где требуется выполнить определенную задачу (решить математический пример, распознать правильным образом символы, распознать правильным образом графические объекты). При успешном решении подобных задач пользователь перенаправляется на основной ресурс (в нашем случае, веб-приложение), проходя перед этим через один из трех балансировщиков нагрузки.

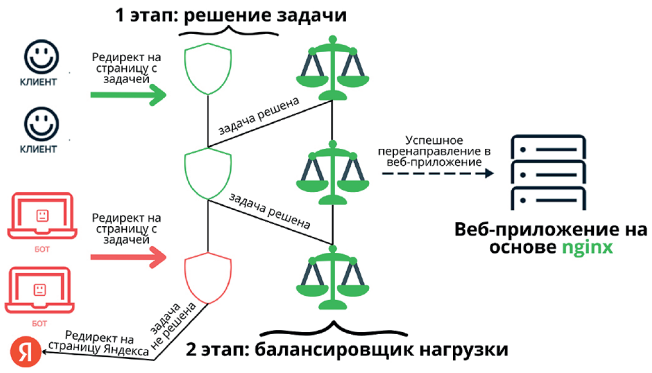

Рис. 6. Сценарий защиты веб-приложения от неполной проверки пользователя

Из (рис. 6) видно, что 1 этап состоит из решения задачи. Для этого необходимо реализовать шаблоны нескольких задач, которые будут включены в паттерн безопасности.

При успешном прохождении первого этапа пользователь перенаправляется на второй этап: к балансировщику нагрузки. Существующие методы балансировки нагрузки, которые присутствуют в nginx [14]:

1. Round Robin. Метод балансировки нагрузки, при котором каждому серверу в кластере предоставляется равная возможность обрабатывать запросы.
2. Round Robin с добавлением веса. Чтобы решить проблему простоя серверов, есть возможность использовать server weights (серверные веса), чтобы указать nginx, какие серверы должны иметь наибольший приоритет.
3. Least Connection. Метод работает путём маршрутизации каждого нового запроса на соединение – на сервер с наименьшим количеством активных соединений. Это гарантирует, что все серверы используются одинаково и ни один из них не перегружен.
4. Least Connection с добавлением веса. Метод работает путем формирования пула активных соединений. Пул формируется со всех серверов. Далее активным соединениям присваивают веса и на их основе балансируют нагрузку на сервера. Метод также позволяет снизить время отклика каждого сервера за счет балансировки нагрузки.
5. IP Hash. Метод балансировки IP Hash использует алгоритм хэширования для определения того, какой сервер должен получить каждый из входящих пакетов. Метод использует IP-адрес источника и IP-адрес назначения и создаёт уникальный хэш-ключ. Затем он используется для распределения клиента между определёнными серверами. Преимущество этого подхода в том, что он может обеспечить более высокую производительность, чем другие методы, такие как Round Robin [15].

В нашем механизме защиты мы используем метод IP Hash. Выбор обусловлен тем, что IP Hash применяется в сценариях, где необходимо поддерживать сессию между клиентом и сервером, что оптимально подходит для веб-приложения.

Разрабатываемый нами паттерн может быть использован не только для обеспечения безопасности веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя, но и интегрирован в систему SIEM (Security Information and Event Management).

Новизна предлагаемого решения определяется возможностью использовать новый механизм защиты, обеспечивающий переадресацию и проверку пользователя на специальном наборе задач и дальнейшую балансировку его запроса к веб-приложению методом IP Hash для обеспечении безопасности веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате

неполной проверки пользователя в облачной информационной инфраструктуре России при переходе на импортозамещение.

Практическая значимость предлагаемого решения включает паттерн для веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя, который можно применить для широкого круга веб-приложений.

Паттерн безопасности для веб-приложения

Для реализации паттерна использован следующий стек технологий: JavaScript; Docker, nginx. Паттерн безопасности (рис. 7) построен на основе микросервисной архитектуры [16, 17]. За основу взят язык JavaScript.

Рис. 7. Микросервисная архитектура паттерна безопасности для веб-приложения

При взаимодействии пользователя с приложением активируется механизм защиты, созданный с помощью JavaScript и описанный ранее, за виртуализацию отвечает Docker, а балансировщик нагрузки nginx позволяет обеспечить стабильную работу веб-приложения.

Развертывание контейнера nginx производилось на операционной системе Windows 10 с помощью Microsoft Visual Studio и Docker Desktop, согласно типовых инструкций настройки и конфигурирования [18, 19, 20].

Для того чтобы веб-сервер был доступен извне контейнера, необходимо соединить порт 80 контейнера с портом операционной системы с помощью команды [20]:

```
docker run -d -p 80:80 nginx.
```

Далее создаём файл vhost.conf в котором пишем следующий код и настраиваем конфигурацию, чтобы сервер открывал страницу нашего сайта [20]:

```
server {
listen 80;
server_name localhost;
index index.html;
root /var/www/public_html;}.
```

Далее передаем конфигурацию сервера внутрь контейнера с помощью команды [20]:

```
docker container run -d -p 80:80 v
"${PWD}/vhost.conf:/etc/nginx/conf.d/
default.conf" nginx.
```

Создаём файл index.html стартовой страницы веб-приложения и запускаем контейнер с помощью команды [20]:

```
docker container run -d -p 80:80 -v
"${PWD}/vhost.conf:/etc/nginx/conf.d/
default.conf" -v "${PWD}/www:/var/www/
public_html" nginx.
```

Для реализации механизма защиты в веб-приложении была создана перенаправляющая страница index.html. Данная страница позволяет в случайном порядке перенаправить пользователя на другую страницу, где располагается одна из задач для решения: index_1.html, index_2.html, index_3.html. Реализованные страницы продемонстрированы на рис. 8, рис. 9, рис. 10.

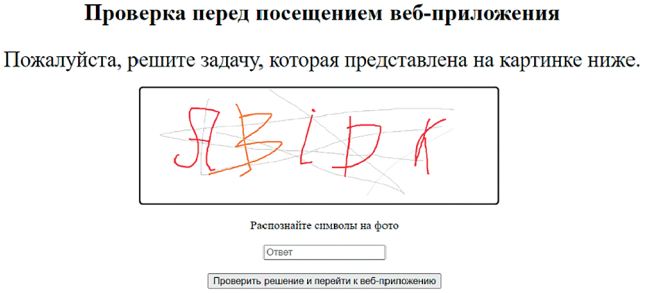

Рис. 8. Страница index_1.html с задачей распознать символы

На странице index_1.html пользователь должен распознать правильным образом символы. На странице index_2.html пользователь должен решить математический пример. На странице index_3.html пользователь должен распознать графические объекты.

Рис. 9. Страница index_2.html с задачей решить математический пример

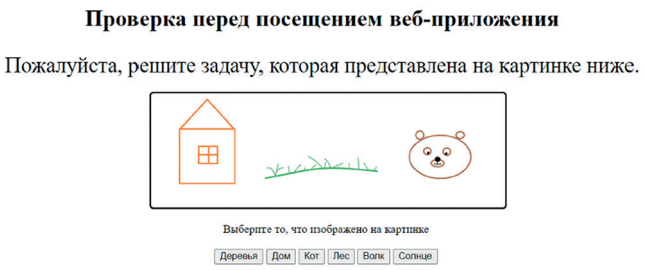

Рис. 10. Страница index_3.html с задачей распознать графические объекты

Согласно рис. 6, при неправильном решении пользователем задачи он перенаправляется на страницу yandex.ru. В случае успешного решения пользователь перенаправляется на балансировщик нагрузки nginx. В этом случае файл конфигурации vhost.conf необходимо дополнить специальным кодом. Итоговый файл vhost.conf содержит следующий код:

```
   upstream backend {
   ip_hash;
   server backend1:80;
   server backend2:80;
   server backend3:80;}
   server {
   listen 80;
   server_name localhost;
   root /var/www/public_html;
   index index.html;
   location / {
   proxy_pass http://backend;
   proxy_set_header Host $host;
   proxy_set_header X-Real-IP $remote_addr;
   proxy_set_header X-Forwarded- For $proxy_
add_x_forwarded_for;
   proxy_set_header X-Forwarded-Proto
$scheme;}}
```

Вначале определен блок upstream с именем backend, который содержит список серверов, в коде выше это backend1, backend2 и backend3. Эти серверы будут обрабатывать входящие запросы, и они распределяются по методу IP Hash, который гарантирует, что запросы от одного и того же IP-адреса всегда будут направляться на один и тот же сервер, что гарантирует безопасность при сохранении сессий пользователя. В основной конфигурации сервера определено, что он слушает порт 80 и отвечает на запросы к домену localhost, а также добавлен блок location, который указывает nginx перенаправлять все входящие запросы на группу серверов backend. При этом используются дополнительные директивы, которые устанавливают заголовки для передачи информации о клиенте на сервер (Host, X-Real-IP, X-Forwarded-For и X-Forwarded-Proto). Эти заголовки передают информацию о реальном IP-адресе клиента и протоколе, что необходимо для корректной и безопасной работы приложений на сервере.

Таким образом, предложенный вариант конфигурации позволяет равномерно распределять нагрузку на несколько серверов, сохраняя при этом согласованность сеансов для пользователей и обеспечивая безопасности веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя.

Предложенный вариант конфигурации включен в код паттерна безопасности веб-приложения, вместе с остальными файлами на языке JavaScript.

Тестирование паттерна безопасности веб-приложения и обсуждение результатов

Для проверки работоспособности паттерна безопасности веб-приложения было выбрано нагрузочное тестирование. Это проверка устойчивости и производительности программного обеспечения под нагрузкой, сопоставимой с реальными условиями использования. В качестве инструмента выбран Grafana k6 – это бесплатный инструмент нагрузочного тестирования с открытым исходным кодом, который упрощает тестирование производительности и делает его более продуктивным для инженерных групп [21].

Была произведена установка Grafana k6 на операционную систему Windows 10 с помощью команды:

```
winget install k6 --source winget.
```

В основной директории проекта был создан файл для тестирования test.js, содержащий следующий код:

```
import http from 'k6/http';
import {sleep} from 'k6';
export const options = {
stages: [
{duration: '1m', target: 20 },
{duration: '3m', target: 20 },
{duration: '1m', target: 0 },],};
export default function (){
http.get('http://nginx1/index.html');
sleep(1);}
```

Программный код теста test.js с реализованным сценарием тестирования, включает в себя три этапа с различными уровнями нагрузки.

На первом этапе, который длится 1 минуту, количество виртуальных пользователей постепенно увеличивается до 20, что позволяет оценить, как веб-приложение справляется с возрастающей нагрузкой. Это помогает определить, насколько эффективно оно адаптироваться к увеличению числа пользователей.

На втором этапе, который длится 3 минуты, нагрузка остается стабильной на уровне 20 виртуальных пользователей. Это позволяет протестировать производительность веб-приложения при постоянной нагрузке, выяснить, насколько стабильно он работает

в условиях активного использования, и выявить возможные проблемы с производительностью при длительной нагрузке.

На третьем этапе, который также длится 1 минуту, нагрузка постепенно снижается до нуля, что позволяет оценить, как веб-приложение справляется с уменьшением нагрузки и возвращается ли он в нормальное состояние после интенсивной работы.

Основная функция в коде отправляет get-запросы к указанному url – 'http://nginx1/index.html, что имитирует действия пользователей, посещающих веб-страницу. Включение паузы в одну секунду между запросами помогает избежать слишком частого обращения к серверу и создает более реалистичное поведение пользователей. Весь этот сценарий позволяет получить полное представление о том, как веб-приложение справляется с различными уровнями нагрузки, и выявить его слабые места или области, требующие оптимизации.

Далее был запущен Grafana k6 с помощью команды:

```
docker run --rm -i --network network1 -v
${PWD}:/app -w /app grafana/k6 run /app/
test.js.
```

Следует отметить, что запуск производился в сети network1, которая была заранее создана командой:

```
docker network create network1.
```

Так как первоначально веб-приложение nginx не было запущено в сети network1, то следует произвести его повторный запуск, но уже в сети network1, командой [20]:

```
docker run -d -p 80:80 -v "${PWD}/vhost.
conf:/etc/nginx/conf.d/default.conf" -v
"${PWD}/www:/var/www/public_html" --network
network1 --name nginx1 nginx.
```

На рис. 11 представлены результаты проведенного нагрузочного тестирования паттерна безопасности веб-приложения.

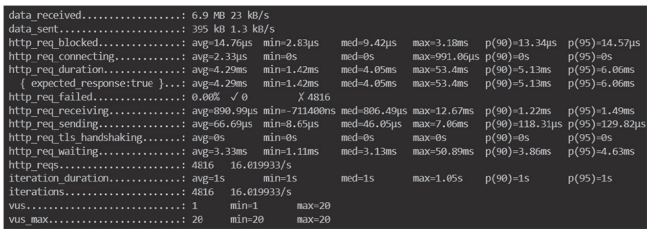

Рис. 11. Результаты нагрузочного тестирования паттерна безопасности веб-приложения

Обсудим основные результаты нагрузочного тестирования. В ходе теста было отправлено и получено определенное количество данных: 6,9 Мбайт данных, полученных от веб-приложения со средней скоростью 23 Кбайт/с, и 395 Кбайт данных, отправленных веб-приложению со средней скоростью 1,3 Кбайт/с.

Метрика http_req_blocked отражает время блокировки HTTP-запросов до их отправки, которое в среднем составило 14,76 микросекунд, а максимальное значение достигало 3,18 миллисекунд.

Время подключения к веб-приложению, которое показывает метрика http_req_connecting практически отсутствовало, что свидетельствует о быстром соединении.

Метрика http_req_duration показывает среднюю длительность HTTP-запросов, которая составила 4,29 миллисекунд, при этом минимальное значение – 1,42 миллисекунды, а максимальное – 53,4 миллисекунд. Для 90 % запросов длительность не превышала 5,13 миллисекунд, а для 95 % – 6,06 миллисекунд.

Время получения ответа, которое показывает метрика http_req_receiving, в среднем составило 890,99 микросекунд, однако к этому значению следует относится осторожно, так как наблюдаются аномалии с отрицательными значениями, что может свидетельствовать о баге в измерении времени по конкретной метрике.

Время отправки запроса, которое показывает метрика http_req_sending, было очень коротким, в среднем 66,69 микросекунд. В тесте не использовались запросы с TLS-шифрованием, поэтому показатели метрики http_reg_tls_handshaking равны нулю.

Время ожидания ответа от веб-приложения, которое показывает метрика http_req_waiting, составило в среднем 3,33 миллисекунды. За все время теста было отправлено 4816 HTTP-запросов с частотой около 16 запросов в секунду. Каждый цикл теста (см. метрику iteration_duration) длился ровно одну секунду, что подтверждает стабильность паузы между запросами. Всего в тесте участвовало до 20 виртуальных пользователей одновременно, с постепенным увеличением нагрузки, что соответствует конфигурации заданного теста.

В результате тестирования не было зафиксировано ни одного сбоя запросов, это означает, что все 4816 запросов были успешными, а это в свою очередь свидетельствует о стабильной работе паттерна безопасности веб-приложения.

Выводы

Рассмотрены сценарии DDoS-атак с использованием ботнет. Построена микросервисная архитектура для обеспечения безопасности веб-приложения. Разработан паттерн для веб-приложения при угрозе неконтролируемого роста числа зарезервированных ресурсов в результате неполной проверки пользователя

на основе микросервисов, интегрированных в контейнеры. В рамках проведённого исследования был разработан сервис проверки пользователей на языке JavaScript, с виртуализацией на базе Docker и с балансировщиком нагрузки nginx. Механизм защиты реализован следующим образом. Пользователь перед заходом в веб-приложение перенаправляется на страницу, где требуется выполнить определенную задачу: решить математический пример, распознать правильным образом символы, распознать правильным образом графические объекты. При успешном решении задач пользователь перенаправляется на веб-приложение, проходя перед этим через один из трех балансировщиков нагрузки, использующих метод IP Hash. Разработан программный код сервиса проверки пользователей, включая коды специальных методов и алгоритмы для трех указанных выше задач. Проведено тестирование паттерна безопасности веб-приложения на базе Grafana k6. Разработан программный код теста test.js с реализованным сценарием тестирования, который включает в себя три этапа с различными уровнями нагрузки. В тесте участвовало до 20 виртуальных пользователей одновременно, с постепенным увеличением нагрузки. В результате тестирования не было зафиксировано ни одного сбоя запросов, все 4816 запросов были успешными – это свидетельствует о стабильной работе паттерна безопасности веб-приложения.

Литература

1. Shameer Mohammed, S. Nanthini, N. Bala Krishna, Inumarthi V. Srinivas, Manikandan Rajagopal, M. Ashok Kumar, A new lightweight data security system for data security in the cloud computing, Measurement: Sensors, Volume 29, 2023, 100856.
2. S. Achar, Cloud computing security for multi-cloud service providers: controls and techniques in our modern threat landscape, International Journal of Computer and Systems Engineering, 16(9), 2022, 379–384.
3. Oludare Isaac Abiodun, Moatsum Alawida, Abiodun Esther Omolara, Abdulatif Alabdulatif, Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: A survey, Journal of King Saud University – Computer and Information Sciences, Volume 34, Issue 10, Part B, 2022, 10217–10245.
4. Chakraborti, A., Curtmola, R., Katz, J., Nieh, J., Sadeghi, A. R., Sion, R., Zhang, Y., Cloud Computing Security: Foundations and Research Directions. Foundations and Trends in Privacy and Security, 3(2), 2022, 103–213.
5. Ukeje, N., Gutierrez, J., Petrova, K., Information security and privacy challenges of cloud computing for government adoption: a systematic review, International Journal of Information Security, Volume 23, 2024, 1459–1475.
6. Fatemeh Khoda Parast, Chandni Sindhav, Seema Nikam, Hadiseh Izadi Yekta, Kenneth B. Kent, Saqib Hakak, Cloud computing security: A survey of service-based models, Computers & Security, Volume 114, 2022, 102580.
7. Anmol Kumar, Mayank Agarwal, Quick service during DDoS attacks in the container-based cloud environment, Journal of Network and Computer Applications, Volume 229, 2024, 103946.
8. Yunhe Cui, Qing Qian, Chun Guo, Guowei Shen, Youliang Tian, Huanlai Xing, Lianshan Yan, Towards DDoS detection mechanisms in Software-Defined Networking, Journal of Network and Computer Applications, Volume 190, 2021, 103156.
9. Anderson Bergamini de Neira, Burak Kantarci, Michele Nogueira, Distributed denial of service attack prediction: Challenges, open issues and opportunities, Computer Networks, Volume 222, 2023, 109553.
10. Shahbaz Ahmad Khanday, Hoor Fatima, Nitin Rakesh, Implementation of intrusion detection model for DDoS attacks in Lightweight IoT Networks, Expert Systems with Applications, Volume 215, 2023, 119330.
11. Man Li, Huachun Zhou, Shuangxing Deng, Parallel path selection mechanism for DDoS attack detection, Journal of Network and Computer Applications, Volume 230, 2024, 103938.
12. Jordana J. George, Dorothy E. Leidner, From clicktivism to hacktivism: Understanding digital activism, Information and Organization, Volume 29, Issue 3, 2019, 100249.
13. Cameron John Hoffman, C. Jordan Howell, Robert C. Perkins, David Maimon, Olena Antonaccio, Predicting new hackers' criminal careers: A group-based trajectory approach, Computers & Security, Volume 137, 2024, 103649.
14. B. Balatamoghna, Aditya Jaganath, S. Vaideeshwaran, Anish Subramanian, K. Suganthi, Integrated balancing approach for hosting services with optimal efficiency - Self Hosting with Docker, Materials Today: Proceedings, Volume 62, Part 7, 2022, 4612–4619.
15. Stephen Jacob, Yuansong Qiao, Yuhang Ye, Brian Lee, Anomalous distributed traffic: Detecting cyber security attacks amongst microservices using graph convolutional networks, Computers & Security, Volume 118, 2022, 102728.
16. Diogo Faustino, Nuno Gonçalves, Manuel Portela, António Rito Silva, Stepwise migration of a monolith to a microservice architecture: Performance and migration effort evaluation, Performance Evaluation, Volume 164, 2024, 102411.
17. Hassaan Siddiqui, Ferhat Khendek, Maria Toeroe, Microservices based architectures for IoT systems - State-of-the-art review, Internet of Things, Volume 23, 2023, 100854.
18. Hubin Yang, Ruochen Shao, Yanbo Cheng, Yucong Chen, Rui Zhou, Gang Liu, Guoqi Xie, Qingguo Zhou, REDB: Real-time enhancement of Docker containers via memory bank partitioning in multicore systems, Journal of Systems Architecture, Volume 151, 2024, 103135.
19. Enrico Cambiaso, Luca Caviglione, Marco Zuppelli, DockerChannel: A framework for evaluating information leakages of Docker containers, SoftwareX, Volume 24, 2023, 101576.
20. Корнеев Н. В., Лазорин Д. С. Паттерн для обеспечения безопасности веб-приложения при угрозе XSS атак в облачной инфраструктуре // Вопросы кибербезопасности. 2024. № 6(64). С. 76–84.
21. Vladimir Ciric, Marija Milosevic, Danijel Sokolovic, Ivan Milentijevic, Modular deep learning-based network intrusion detection architecture for real-world cyber-attack simulation, Simulation Modelling Practice and Theory, Volume 133, 2024, 102916.

PATTERN FOR SECURING WEB APPLICATION UNDER THREAT OF UNCONTROLLED GROWTH IN THE NUMBER OF RESERVED RESOURCES

Korneev N. V.[1], Trubacheva-Gudovich A. E.[2]

Keywords: *template, DDoS attack, botnet, user verification service, load balancer, IP Hash method, character recognition task, graphic object recognition task, mathematical task, container, testing.*

The purpose of this article: *development of a pattern for a web application in case of a threat of uncontrolled growth of the number of reserved resources as a result of incomplete user verification.*

Research method: *analysis of the principles of DDoS attacks. Synthesis of DDoS attack scenarios for three types of attacks: transport layer, infrastructure layer, and application layer. The security scenario of a web application is chosen as the basis for the threat of an uncontrolled increase in the number of reserved resources as a result of incomplete user verification. A new protection mechanism has been proposed that provides redirection and verification of the user on a special set of tasks and further balancing of his request to the web application using the IP Hash method. The research was carried out by full-scale modeling of a Docker-based web application in containerization-enabled environments, its deployment and testing.*

Result: *the analysis of the threat of uncontrolled growth in the number of reserved resources is carried out and the relevance of the problem of developing universal template security mechanisms called patterns is shown. In particular, the scenarios of DDoS attacks on web applications are considered. A scenario for ensuring the security of a web application is proposed when there is a threat of an uncontrolled increase in the number of reserved resources as a result of incomplete user verification. A microservice architecture has been built to ensure the security of a web application. A pattern has been developed for a web application in the event of a threat of uncontrolled growth in the number of reserved resources as a result of incomplete user verification based on microservices integrated into containers. As part of the research, a user verification service was developed in JavaScript, with Docker-based virtualization, and with an nginx load balancer. The protection mechanism is implemented as follows. Before entering the web application, the user is redirected to a page where a specific task is required: to solve a mathematical example, to recognize symbols in the right way, to recognize graphic objects in the right way. Upon successful completion of the tasks, the user is redirected to the web application, passing through one of the three load balancers using the IP Hash method. The program code of the user verification service has been developed, including codes of special methods and algorithms for the three tasks mentioned above. A web application security pattern based on Grafana k6 has been tested. The test program code has been developed.js with an implemented testing scenario that includes three stages with different load levels. Up to 20 virtual users participated in the test at the same time, with a gradual increase in workload. As a result of testing, not a single request failure was recorded, all 4816 requests were successful – this indicates the stable operation of the web application security pattern.*

Practical value: *the practical value of the proposed solution includes a pattern for a web application under the threat of uncontrolled growth in the number of reserved resources as a result of incomplete user verification, which can be applied to a wide range of web applications.*

References

1. *Shameer Mohammed, S. Nanthini, N. Bala Krishna, Inumarthi V. Srinivas, Manikandan Rajagopal, M. Ashok Kumar, A new lightweight data security system for data security in the cloud computing, Measurement: Sensors, Volume 29, 2023, 100856.*
2. *S. Achar, Cloud computing security for multi-cloud service providers: controls and techniques in our modern threat landscape, International Journal of Computer and Systems Engineering, 16(9), 2022, 379–384.*
3. *Oludare Isaac Abiodun, Moatsum Alawida, Abiodun Esther Omolara, Abdulatif Alabdulatif, Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: A survey, Journal of King Saud University – Computer and Information Sciences, Volume 34, Issue 10, Part B, 2022, 10217–10245.*
4. *Chakraborti, A., Curtmola, R., Katz, J., Nieh, J., Sadeghi, A.R., Sion, R., Zhang, Y., Cloud Computing Security: Foundations and Research Directions. Foundations and Trends in Privacy and Security, 3(2), 2022, 103–213.*
5. *Ukeje, N., Gutierrez, J., Petrova, K., Information security and privacy challenges of cloud computing for government adoption: a systematic review, International Journal of Information Security, Volume 23, 2024, 1459–1475.*

1 Nikolay V. Korneev, Doctor of Technical Sciences, Associate Professor, Gubkin Russian State University of Oil and Gas, Moscow, Russia. E-mail: niccyper@mail.ru
2 Anna E. Trubacheva-Gudovich, Student, Gubkin Russian State University of Oil and Gas, Moscow, Russia. E mail: gudovich.an@bk.ru

6. *Fatemeh Khoda Parast, Chandni Sindhav, Seema Nikam, Hadiseh Izadi Yekta, Kenneth B. Kent, Saqib Hakak, Cloud computing security: A survey of service-based models, Computers & Security,Volume 114, 2022, 102580.*
7. *Anmol Kumar, Mayank Agarwal, Quick service during DDoS attacks in the container-based cloud environment, Journal of Network and Computer Applications, Volume 229, 2024, 103946.*
8. *Yunhe Cui, Qing Qian, Chun Guo, Guowei Shen, Youliang Tian, Huanlai Xing, Lianshan Yan, Towards DDoS detection mechanisms in Software-Defined Networking, Journal of Network and Computer Applications, Volume 190, 2021, 103156.*
9. *Anderson Bergamini de Neira, Burak Kantarci, Michele Nogueira, Distributed denial of service attack prediction: Challenges, open issues and opportunities, Computer Networks, Volume 222, 2023, 109553.*
10. *Shahbaz Ahmad Khanday, Hoor Fatima, Nitin Rakesh, Implementation of intrusion detection model for DDoS attacks in Lightweight IoT Networks, Expert Systems with Applications, Volume 215, 2023, 119330.*
11. *Man Li, Huachun Zhou, Shuangxing Deng, Parallel path selection mechanism for DDoS attack detection, Journal of Network and Computer Applications, Volume 230, 2024, 103938.*
12. *Jordana J. George, Dorothy E. Leidner, From clicktivism to hacktivism: Understanding digital activism, Information and Organization, Volume 29, Issue 3, 2019, 100249.*
13. *Cameron John Hoffman, C. Jordan Howell, Robert C. Perkins, David Maimon, Olena Antonaccio, Predicting new hackers' criminal careers: A group-based trajectory approach, Computers & Security, Volume 137, 2024, 103649.*
14. *B. Balatamoghna, Aditya Jaganath, S. Vaideeshwaran, Anish Subramanian, K. Suganthi, Integrated balancing approach for hosting services with optimal efficiency - Self Hosting with Docker, Materials Today: Proceedings, Volume 62, Part 7, 2022, 4612–4619.*
15. *Stephen Jacob, Yuansong Qiao, Yuhang Ye, Brian Lee, Anomalous distributed traffic: Detecting cyber security attacks amongst microservices using graph convolutional networks, Computers & Security, Volume 118, 2022, 102728.*
16. *Diogo Faustino, Nuno Gonçalves, Manuel Portela, António Rito Silva, Stepwise migration of a monolith to a microservice architecture: Performance and migration effort evaluation, Performance Evaluation, Volume 164, 2024, 102411.*
17. *Hassaan Siddiqui, Ferhat Khendek, Maria Toeroe, Microservices based architectures for IoT systems – State-of-the-art review, Internet of Things, Volume 23, 2023, 100854.*
18. *Hubin Yang, Ruochen Shao, Yanbo Cheng, Yucong Chen, Rui Zhou, Gang Liu, Guoqi Xie, Qingguo Zhou, REDB: Real-time enhancement of Docker containers via memory bank partitioning in multicore systems, Journal of Systems Architecture, Volume 151, 2024, 103135.*
19. *Enrico Cambiaso, Luca Caviglione, Marco Zuppelli, DockerChannel: A framework for evaluating information leakages of Docker containers, SoftwareX, Volume 24, 2023, 101576.*
20. *Korneev N. V., Lazorin D. S. Pattern dlya obespecheniya bezopasnosti veb-prilozheniya pri ugroze XSS atak v oblachnoj infrastrukture // Voprosy kiberbezopasnosti. 2024. № 6(64). S. 76–84.*
21. *Vladimir Ciric, Marija Milosevic, Danijel Sokolovic, Ivan Milentijevic, Modular deep learning-based network intrusion detection architecture for real-world cyber-attack simulation, Simulation Modelling Practice and Theory, Volume 133, 2024, 102916.*