ПАТТЕРН ДЛЯ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЯ ПРИ УГРОЗЕ XSS АТАК В ОБЛАЧНОЙ ИНФРАСТРУКТУРЕ

Корнеев Н. В.[1], Лазорин Д. С.[2]

DOI: 10.21681/2311-3456-2024-6-76-84

Цель статьи: *разработка шаблонного механизма защиты для обеспечения безопасности веб-приложения при угрозе межсайтового скриптинга.*

Метод исследования: *анализ принципа работы межсайтового скриптинга, в частности хранимого XSS. Синтез межсайтового сценария с помощью двух уровней защиты: кодирование данных на выходе и подтверждение ввода по прибытии. С использованием методов экранирования в Unicode, блокировки и применения нескольких уровней кодирования в правильном порядке для недопустимого ввода вредоносного кода предложены операции замены опасных символов на подходящие HTML-мнемоники. Исследование выполнено путем натурного моделирования веб-приложения на основе Docker в средах с поддержкой контейнеризации, его развёртывания и тестирования при угрозе межсайтового скриптинга.*

Результат: *проведен анализ облачной безопасности веб-приложений и показана актуальность проблемы разработки универсальных шаблонных механизмов безопасности, называемыми паттернами для защиты веб-приложения от XSS атак. В частности, рассмотрены принципы работы межсайтового скриптинга, типы XSS атак, и шаблонный механизмы защиты веб-приложения от XSS атак. Определена последовательность действий пользователя при входе в типовое веб-приложение. Построена микросервисная архитектура паттерна для защиты веб-приложения от XSS атак. Разработан паттерн для защиты веб-приложения от XSS атак на основе микросервисов с учётом сервиса безопасности, включающего механизмы защиты. На практическом примере реального веб-приложения развернуто 4 контейнера (nginx, php, mysql, phpmyadmin) и настроено взаимодействие между ними. Реализована форма регистрации и форма авторизации типового веб-приложения. Произведена XSS атака на веб-приложение кодом JavaScript без механизма защиты. В коде php реализован сервис безопасности для защиты от XSS атак с помощью встроенной функции htmlspecialchars. Приведен программный код сервиса в виде функции. Программный код сервиса безопасности включает функцию htmlspecialchars с кодом конфигурации и функциями взаимодействия с описанными выше контейнерами. Произведена повторная XSS атака в результате которой код JavaScript не был выполнен, данные безопасно были извлечены из базы данных. В результате в базе данных получена строка кода, которая является признаком диагностической ошибки веб-приложения, и может служить маркером для мониторинга блокированной XSS атаки. Для мониторинга XSS атаки использовано открытое программное обеспечение Kubernetes, Prometheus, Grafana, cAdvisor и Node Exporter. Созданы манифесты для реализации базовой конфигурации кластера, состоящего из одного пода с четырьмя контейнерами. В результате развёрнута система мониторинга XSS атаки и показана возможность диагностировать пики изменения нагрузки, как признаки блокированной XSS атаки.*

Практическая ценность: *практическая ценность предлагаемого решение включает шаблонный механизм защиты в виде паттерна. Паттерн можно применить для широкого круга веб-приложений, в том числе перенести разрабатываемое решение на любую отрасль: топливно-энергетическую, экономическую и не только, ввиду кроссплатформенности самого решения.*

Ключевые слова: *облачные вычисления, набор данных, шаблон, вредоносный код, сервис безопасности, контейнер, диагностическая ошибка, маркер, манифест, кластер, система мониторинга*

Введение

В последние десятилетия данные оказались незаменимыми для всех аспектов человеческого существования. Разработка нескольких приложений привела к экспоненциальному расширению объема информации. Эта информация может быть зашифрована и храниться в безопасных местах. Этому помогают облачные вычисления – технологии, которые можно использовать для хранения этих огромных наборов данных [1]. Киберпространство или облачные вычисления резко увеличили практическую полезность компьютеров и периферийных устройств [2]. Они широко используются в финансах, управлении бизнесом, телекоммуникациях, транспорте, образовании, здравоохранении и других сферах нашей повседневной жизни [3]. Это также позволяет пользователям эффективно общаться, совместно использовать программное обеспечение, оборудование, а также ресурсы данных через сетевые протоколы [4].

В эпоху облачных вычислений существует несколько возможностей, позволяющих временно

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 Лазорин Данил Сергеевич, студент, РГУ нефти и газа (НИУ) имени И. М. Губкина, Москва, Россия. E-mail: lazorindanya@yandex.ru

кэшировать данные, хранящиеся удаленно, на настольных компьютерах, мобильных телефонах или других интернет-устройствах [5]. Отрасли программного обеспечения, а также частные лица, которые хранят свои данные в облаке, используют гибкий подход, который дает некоторые преимущества, такие как избежание капитальных затрат на личное обслуживание, оборудование и программное обеспечение [6].

Безопасность и надежность – две основные проблемы в облачных вычислениях. Данные клиента в облаке могут быть доступны другим клиентам, поэтому возникают проблемы с безопасностью данных клиентов. Для обеспечения безопасности облачных данных доступно множество методов и алгоритмов. За последние годы многими исследователями было предложено и реализовано много работ, касающихся облачной безопасности. Однако пробелом остается разработка универсальных шаблонных механизмов безопасности, называемыми паттернами. В частности, в данной статье мы ставим целью разработку шаблонного механизма защиты для обеспечения безопасности веб-приложения при угрозе межсайтового скриптинга.

Анализ и методы исследования

Межсайтовый скриптинг (XSS) [7] – это уязвимость веб-безопасности, которая позволяет злоумышленнику поставить под угрозу взаимодействие пользователей с уязвимым приложением. Это позволяет злоумышленнику обойти одну и ту же политику происхождения, которая предназначена для отделения разных веб-сайтов друг от друга. Уязвимости межсайтового скриптинга обычно позволяют злоумышленнику маскироваться под пользователя-жертву, выполнять любые действия, которые пользователь может выполнить, и получать доступ к любым данным пользователя. Если пользователь-жертва имеет привилегированный доступ к приложению, злоумышленник может получить полный контроль над всеми функциями и данными приложения.

Принцип работы межсайтового скриптинга заключается в манипулировании уязвимым веб-сайтом таким образом, чтобы он возвращал пользователям вредоносный код JavaScript. Когда вредоносный код выполняется внутри браузера жертвы, злоумышленник может полностью поставить под угрозу взаимодействие с веб-приложением.

На практике существуют три основных типа XSS атак:

1. Отраженный XSS (Reflected XSS) [8], где вредоносный скрипт поступает из текущего HTTP-запроса.
2. Хранимый XSS (Stored XSS) [9], где вредоносный скрипт поступает из базы данных сайта.
3. XSS на основе DOM (DOM Based XSS) [10], где уязвимость существует в коде на стороне клиента, а не в коде на стороне сервера.

В данной статье мы ограничимся рассмотрением только второго типа атаки ввиду широкой распространённости, а также ввиду указанной широкой практической применимости полученных результатов.

Хранимый XSS (также известный как постоянный XSS или XSS второго порядка) возникает, когда приложение получает данные из ненадежного источника и включает эти данные в свои последующие HTTP-ответы небезопасным способом.

Соответствующие данные могут быть отправлены в приложение через HTTP-запросы; например, при авторизации или регистрации пользователя. В других случаях данные могут поступать из других ненадежных источников; например, приложение веб-почты, отображающее сообщения, полученные по SMTP, маркетинговое приложение, отображающее сообщения в социальных сетях, или приложение мониторинга сети, отображающее пакетные данные из сетевого трафика.

Для ограничения зоны применимости разрабатываемого шаблонного механизма защиты определим, что нами будет рассматривается веб-приложение, которое строится на основе Docker – это программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Также будут использованы следующие сервисы: веб-сервер NGINX, база данных MySQL, phpMyAdmin для управления базой данных, менеджер FastCGI Process Manager (FPM) для выполнения скриптов сайта. Для каждого сервиса будет запущен отдельный контейнер и настроено взаимодействие между ними с целью проверки механизма защиты.

В веб-приложении будет присутствовать главная страница с авторизацией пользователей, которая, при успешной регистрации, отправляет логин и пароль в базу данных. Далее будет происходить переадресация пользователя на страницу его профиля, где располагаются его персональные данные.

Предотвращение межсайтового сценария обычно можно обеспечить с помощью двух уровней защиты:

1. Кодирование данных на выходе. Кодирование следует применять непосредственно перед записью данных, управляемых пользователем, на страницу, поскольку контекст, в который записываются данные, определяет, какой тип кодировки необходимо использовать.

Например, значения внутри строки JavaScript требуют другого типа экранирования, чем значения в контексте HTML. В контексте HTML следует преобразовать значения, не внесенные в белый список,

в объекты HTML, а в контексте строки JavaScript небуквенно-цифровые значения должны быть экранированы в Unicode. Иногда может потребоваться применение нескольких уровней кодирования в правильном порядке.

2. Подтверждение ввода по прибытии. Кодирование является наиболее важной линией защиты от XSS, но его недостаточно для предотвращения XSS уязвимостей в любом контексте. Необходимо как можно более строго проверять вводимые данные в тот момент, когда они впервые получены от пользователя. В идеальном сценарии проверка ввода должна работать путем блокировки недопустимого ввода. Альтернативный подход, заключающийся в попытке очистить недопустимый ввод, чтобы сделать его действенным, более подвержен ошибкам, и его следует избегать, где это возможно.

Предотвращение XSS в PHP имеет свои особенности. В PHP есть встроенные функции для кодирования сущностей: htmlspecialchars, htmlentities. Следует вызвать одну из этих функцию, чтобы избежать ввода данных внутри контекста HTML. Функцию htmlentities следует вызывать с тремя аргументами:

1. Входная строка.
2. Флаг ENT_QUOTES, который указывает, что все кавычки должны быть закодированы.
3. Набор символов, который в большинстве случаев должен быть в UTF.

Приведем пример 1:

```
<?php echo
htmlentities($input,ENT_QUOTES,'UTF- 8');?>.
```

В контексте строки JavaScript необходимо экранировать ввод в Unicode, как уже было описано ранее.

Функцию htmlspecialchars следует вызывать следующим образом:

1. Проверить, была ли отправлена переменная $_POST['name'].
2. Если переменная была отправлена, то присвоить значение переменной результату функции, которая принимает входную строку (значение $_POST['name']) и флаг ENT_QUOTES, чтобы кодировать кавычки, если таковые имеются в строке. В большинстве случаев набор символов должен быть в UTF-8, поэтому можно указать его третьим аргументом.
3. Если переменная $_POST['name'] не была отправлена или была отправлена пустая строка, присвоить переменной пустую строку.

Приведем пример 2:

```
$name=isset($_POST['name'])?htmlspecialchars
($_POST['name'], ENT_QUOTES, 'UTF-8'):'';.
```

Подводя итог анализу, следует сказать, что для реализации поставленной нами цели для каждого конкретного веб-приложения следует действовать по следующему алгоритму:

1. Провести анализ в области межсайтового скриптинга для условий эксплуатации конкретного веб-приложения.
2. Разработать паттерн веб-приложения, который позволит продемонстрировать схему защиты от угрозы межсайтового скриптинга.
3. Разработать контейнер, который включает в себя все необходимые зависимости, библиотеки и системные инструменты для запуска веб-приложения.
4. Произвести тестирование и мониторинг разработанного контейнера на кластере, с целью проверки его работоспособности от угрозы межсайтового скриптинга.

Далее в статье мы покажем реализацию данного алгоритма для типового решения, подходящего для большинства веб-приложений, а при необходимости, он может быть расширен с учетом особенностей практической реализации в каждом конкретном случае.

В рассматриваемом случае последствиями реализации угроз является нарушение конфиденциальности, целостности, доступности.

Разрабатываемый нами паттерн может быть использован не только для защиты описанного веб-приложения, но и в более широком контексте – для обеспечения безопасности веб-приложений в целом, например, в системе SIEM (Security Information and Event Management) с учетом доработки по описанному выше алгоритму.

Новизна предлагаемого решения определяется возможностью обеспечения безопасности веб-приложения при угрозе XSS атак в облачной информационной инфраструктуре России при переходе на импортозамещение.

Практическая значимость предлагаемого решение включает шаблонный механизм защиты в виде паттерна, который можно применить для широкого круга веб-приложений, в том числе перенести разрабатываемое решение на любую отрасль: топливно-энергетическую, экономическую и не только, ввиду кроссплатформенности самого решения.

Разработка паттерна веб-приложения

Для разработки паттерна веб-приложения, необходимо определить последовательность действий пользователя при входе в веб-приложение. Это ограничивает условие эксплуатации самого паттерна, с одной стороны, а с другой дает нам формализовать рассматриваемую задачу. Рассмотрим последовательность действий пользователя поэтапно.

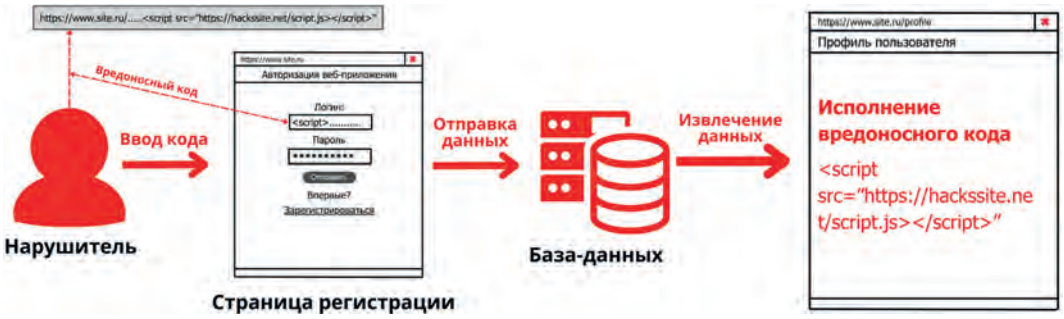

Рис. 1. Сценарий XSS атаки на этапе 2

Рис. 2. Сценарий защиты веб-приложения от XSS атаки

Этап 1. При открытии веб-приложения пользователь находится на главной странице: авторизации веб-приложения. Если же он впервые оказался в веб-приложении, то необходимо произвести регистрацию.

Этап 2. Если пользователь прошел регистрацию, то данные вносятся в базу данных и происходит переадресация на страницу с профилем. Если пользователь уже зарегистрирован, происходит проверка введенных данных, затем дальнейшая переадресация.

Этап 3. Далее пользователь попадает на страницу своего профиля, где отображается информация (персональные данные), извлекаемая из базы данных.

Системная архитектура паттерна веб-приложения строится нами на основе микросервисов. Сама архитектура микросервисов (Microservice Architecture, MA) представляет собой новую парадигму архитектуры программного обеспечения, которая направлена на решение ограничений традиционного монолитного программного обеспечения путем разложения всего программного обеспечения на независимо развертываемые и масштабируемые более мелкие части, называемые сервисами или микросервисами [11, 12]. Идея этой декомпозиции заключается в разделении различных функций системы. В веб-приложении за свой функционал отвечают следующие сервисы: сервис регистрации, сервис авторизации, сервис профиля.

Отметим, что в данном случае, как и в реальных существующих практических решениях, никакой другой обработки данных веб-приложение не осуществляет, поэтому злоумышленник легко может отправить сообщение, атакующее других пользователей. Исходя из этого может возникнуть XSS атака на этапе 2. Подобный сценарий изображен на (рис. 1).

Для решения данной проблемы необходимо реализовать дополнительный сервис, включающий в себя механизм защиты веб-приложения (рис. 2), включающий в себя операции замены опасных символов на подходящие HTML-мнемоники. На (рис. 3) продемонстрирована архитектура предлагаемого паттерна веб-приложения на основе микросервисов с сервисом безопасности для решения указанной проблемы.

Рис.3. Микросервисная архитектура паттерна веб-приложения

Разработка контейнеров для сервисов

Разработка контейнеров для работы сервисов производилась на операционной системе Windows 10 Pro с помощью Microsoft Visual Studio и Docker Desktop согласно типовым инструкциям настройки и конфигурирования [13, 14].

Нами была создана основная директория docker-projects, в которой были созданы каталоги images/php (директория для созданных вручную образов, образ php со всеми необходимыми модулями),

mysql-data (директория для физического хранения файлов баз-данных), www (корневая директория веб-приложения для хранения всех php-скриптов и ресурсов сайта). Далее последовательно выполнена настройка всех контейнеров для совместной работы согласно типовым инструкциям настройки и конфигурирования.

1. Контейнер веб-сервера nginx. Для того чтобы веб-сервер был доступен вне контейнера, необходимо соединить порт 80 контейнера с портом операционной системы. Это можно выполнить с помощью команды:

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

Создаём файл vhost.conf и настраиваем конфигурацию, чтобы веб-сервер открывал страницу нашего сайта.

Далее передаем конфигурацию веб-сервера внутрь контейнера. Это делается с помощью команды:

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

В директории www создаём файл index.html. Далее запускаем контейнер с помощью команды:

```
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".
```

Далее необходимо проверить, отобразилась ли страница index.html.

2. Контейнер php. Необходимо создать образ с помощью типовых инструкций по сборке в специальном файле Dockerfile, который создаётся в директории images/php/. Далее собираем контейнер с помощью команды:

```
docker build -t php8lfpm:l.0
"${PWD}/images/php".
```

Необходимо настроить взаимодействие между контейнерами php81fpm и nginx. Команда:

```
docker inspect <id_container> |
grep IPAddress,
```

выведет все параметры выбранного контейнера (в нашем случае php81fpm). Настроим IP-адрес (порт 9000).

При запуске контейнеров IP-адреса попадают в сеть по умолчанию, в которой к контейнеру можно получить доступ по его IP-адресу, но нельзя по его имени. Чтобы иметь возможность использовать имена контейнеров нужно запускать их в пользовательской сети, предварительно создав её. Для этого используется команда:

```
docker network create networkl.
```

Далее запускаем контейнеры в созданной сети командой:

```
Docker run –d –p 80:80 -v
"${PWD}/vhost.conf:/etc/nginx/conf.d/
default.conf”-v "${PWD}/www:/var/www/public_
html" --network networkl --name nginxl
nginx, docker run -d -v "${PWD}/www:/var/
www/public_html" --network networkl --name
phpl php8lfpm.
```

3. Контейнер MySQL. Необходимо создать контейнер с помощью команды:

```
docker run -d -v "${PWD}/mysql-data:/
var/lib/mysql" -e MYSQL_ROOT_PASSWORD=root
--network network1 --name mysql1 mysql,
```

и проверить подключение к тестовой базе данных. В нашем случае проверка показала, что подключение произошло успешно и данные из БД извлекаются.

4. Контейнер phpMyAdmin. Необходимо создать контейнер с помощью команды:

```
docker run -d -p 1500:80 -e PMA_
HOST=mysql1 --network networkl --name
phpmyadminl phpmyadmin.
```

В браузере перейдем по адресу localhost:1500. Отобразится панель управления phpMyAdmin.

Таким образом, было создано 4 контейнера и настроено взаимодействие между ними.

Для проверки защиты веб-приложения от угрозы межсайтового скриптинга проведем настройку конфигураций страниц веб-приложения, которая позволит продемонстрировать схему защиты.

Конфигурация страниц веб-приложения и проведение тестовой XSS атаки

Была реализована форма регистрации (рис. 4) и форма авторизации (рис. 5). При регистрации в скрипте php осуществляется проверка введённых данных и внесение их в базу данных, далее перенаправление на страницу с авторизацией. При авторизации происходит сверка введенных данных с базой данных и перенаправление на страницу с профилем. В профиле имя пользователя используется из базы данных. Введены ограничения на минимальное и максимальное количество символов в полях: от 3 до 255 символов, при выходе за данные пределы появляется сообщение об ошибке. При вводе разных паролей появляется сообщение об ошибке. При вводе почтового адреса, который уже внесён в базу данных, появляется сообщение об ошибке.

Проведем тестовую XSS атаку поэтапно:

Этап 1. При регистрации в поле ФИО (рис. 4) укажем JavaScript код:

```
<script>alert(’Код’)</script>,
```

и заполним остальные поля так, как показано на (рис. 4).

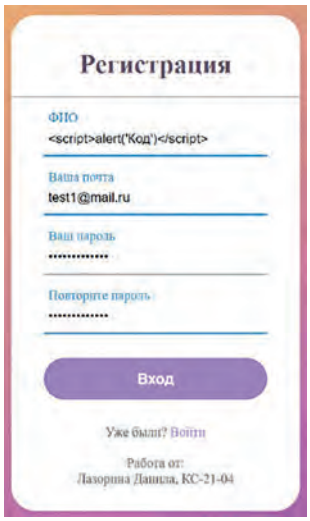

Рис. 4. Форма регистрации

Этап 2. Регистрация на этапе 1 прошла успешно, далее произведем авторизацию так, как показано на (рис. 5).

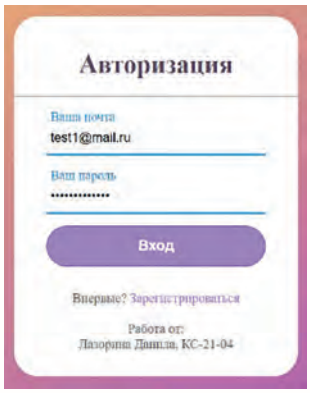

Рис. 5. Форма авторизации

Этап 3. После авторизации выполняется код, указанный в поле ФИО при регистрации (рис. 5). Следовательно, уязвимость в веб-приложении присутствует.

Этап 4. Переход в профиль. Отметим следующее: имя пользователя уже не отображается – XSS атака реализована (рис. 6).

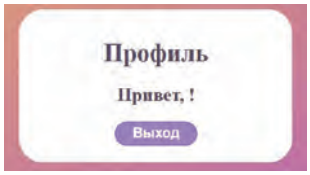

Рис. 6. Профиль пользователя с успешной реализацией XSS атаки

Реализация сервиса безопасности от XSS атаки

Как видно на этапе 3, проблема заключается в том, что переменные переносятся в базу данных такими, какими ввёл их пользователь, без дополнительной обработки. Необходимо осуществить механизм защиты от подобного рода атак. В этом могут помочь HTML-мнемоники – кодовое представление символа в HTML, который начинается со знака амперсанда и завершается точкой с запятой. Теги <script> состоят из треугольных скобок, следовательно, их необходимо заменить на мнемоники. Благодаря этому текст не будет трактоваться браузером как HTML-тег.

В php-сценариях на этапах регистрации и авторизации необходимо реализовать сервис безопасности, включающий в себя вызов функции, которая для переданной строки выполнит фильтрацию и заменит все опасные символы в ней на подходящие HTML-мнемоники. Такая функция в нашем случае называется htmlspecialchars.

Приведем пример кода для обработки поля ФИО пользователя с помощью данной функции:

```
$name=isset($_POST['name'])?htmlspecialch
ars($_POST['name']):'';.
```

Повторим этап 1 и этап 2. Результат авторизации показан на (рис.7).

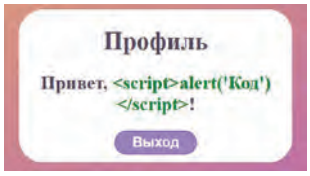

Рис. 7. Профиль пользователя с блокированной XSS атакой

Как видно из (рис. 7), код JavaScript не был выполнен, данные безопасно были извлечены из базы данных. В БД данные в поле ФИО для пользователя выглядят строкой кода:

```
&lt;script&gt;alert(&#039;Код&#039;)&lt;/
script&gt,
```

что является признаком диагностической ошибки веб-приложения, и может служить маркером для мониторинга блокированной XSS атаки.

Таким образом, в коде php реализован сервис безопасности с помощью встроенной функции htmlspecialchars, с его помощью удалось реализовать механизм защиты от XSS атаки. Программный код в виде функции:

```
function sanitizeInput($input) {
return isset($input) ?
htmlspecialchars($input) : “” ;
}
$name = sanitizeInput($_POST['name']).
```

Программный код сервиса безопасности включает встроенную функцию htmlspecialchars с кодом конфигурации и функциями взаимодействия с контейнерами.

Мониторинг XSS атаки и обсуждение результатов

Для мониторинга XSS атаки может служить указанный выше маркер в виде соответствующей диагностической ошибки веб-приложения. Для реализации системы мониторинга веб-приложения на выделенном кластере нами использовано открытое программное обеспечение Kubernetes [15]. В Kubernetes наименьшей единицей является «под» – это абстрактный объект, представляющий собой группу из одного или нескольких контейнеров приложения (в нашем случае, Docker).

Для реализации базовой конфигурации кластера, состоящего из одного пода с четырьмя контейнерами, создан специальный манифест config.yaml. Манифест описывает развертывание веб-приложения, состоящего из веб-сервера nginx, сервера обработки php (php-fpm), базы данных mysql и интерфейса администрирования базы данных phpmyadmin, а также настройку сервиса для доступа к веб-серверу nginx. Развёртывание кластера осуществляется командой:

```
kubectl create -f config.yaml.
```

Для реализации системы мониторинга нами использованы: Prometheus и Grafana [16], а также cAdvisor и Node Exporter [17].

Prometheus – это набор инструментов для мониторинга и оповещения систем с открытым исходным кодом.

Grafana – это платформа для мониторинга, анализа данных и визуализации собранных данных с открытым исходным кодом. Grafana упрощает мониторинг и анализ состояния системы для разработчиков и администраторов. Таким образом, вместе они обеспечивают мощный инструментарий для эффективного контроля и оптимизации работы IT-инфраструктуры любого уровня.

cAdvisor – предоставляет данные по использованию ресурсов и производительности запущенных контейнеров. Формирует метрики в читаемом для Prometheus формате.

Node Exporter – это агент сбора метрик системы, таких как использование процессора, памяти, дисков, сети, а также самостоятельных метрик в виде соответствующей диагностической ошибки веб-приложения.

Для упрощения развёртывания системы мониторинга нами использован Docker Compose, который обеспечивает управление всеми этапами в жизненном цикле определенной службы: запуском, остановкой и перестроением служб; просмотром состояния службы, потоковой передачей журналов. Созданы манифест docker-compose.yml, директория prometheus с манифестом prometheus.yml, директория grafana с манифестом config.monitoring, поддиректория provisioning/dashboards с манифестом dashboard.yml и поддиректория provisioning/datasources с манифестом datasource.yml.

При использовании Docker Compose также потребуется установить переменные в Windows PowerShell, чтобы преобразовать пути Windows, присутствующие в сопоставлениях томов Docker Compose.

На (рис. 8) показана витрина Docker Containers в Grafana, которая отображает информацию о наших контейнерах Docker, запущенных на хостовой машине. Здесь демонстрируются данные о ресурсах, потребляемых каждым контейнером, такие как CPU, память, сеть и дисковое пространство, а также другие метрики, в том числе самостоятельные метрики

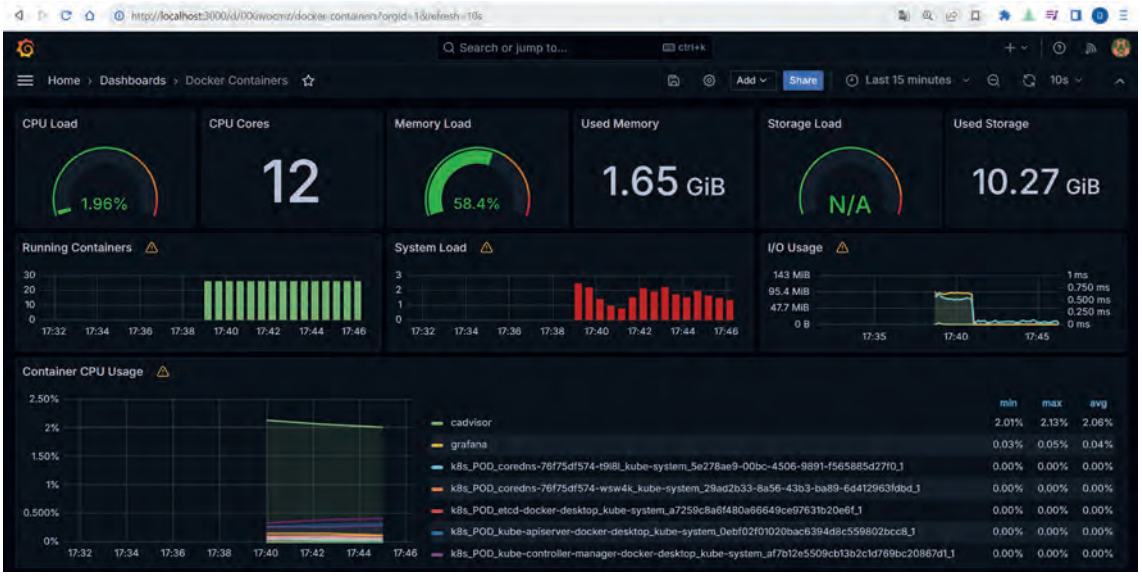

Рис. 8. Витрина Docker Containers в Grafana

в виде соответствующей диагностической ошибки веб-приложения, которые, собираются с помощью инструментов мониторинга.

Пик изменения нагрузки на (рис. 8) в 17:40 является признаком диагностической ошибки веб-приложения, и может служить маркером для мониторинга блокированной XSS атаки. В то же время можно настроить дополнительные метрики, более чувствительные к соответствующей диагностической ошибки веб-приложения.

Выводы

Построена микросервисная архитектура паттерна для защиты веб-приложения от XSS атак для широко круга веб-приложений в облачной инфраструктуре. Разработан паттерн для защиты веб-приложения от XSS атак на основе микросервисов, интегрированных в контейнеры, с учётом сервиса безопасности, включающего механизмы защиты от XSS-атак, которые могут быть использованы как шаблон по построению аналогичных систем безопасности от XSS атак. В коде php реализован сервис безопасности от XSS атак с помощью встроенной функции htmlspecialchars. Программный код сервиса безопасности включает встроенную функцию htmlspecialchars с кодом конфигурации и функциями взаимодействия с контейнерами. Произведена практическая проверка работы сервиса безопасности путем натурной имитации XSS атаки на типовое веб-приложение, которая показала его эффективность в отношении отражения XSS атаки. Для мониторинга XSS-атаки на веб-приложение использовано открытое программное обеспечение. Созданы манифесты для реализации базовой конфигурации кластера, состоящего из одного пода с четырьмя контейнерами. В результате развёрнута система мониторинга XSS атаки и показана возможность диагностировать пики изменения нагрузки, как признаки блокированной XSS атаки, которая может быть использована в системах SIEM.

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, Issue 2/2024, 2024, https://doi.org/10.1007/s10207-023-00797-6.*
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. *Faizan Younas, Ali Raza, Nisrean Thalji, Laith Abualigah, Raed Abu Zitar, Heming Jia, An efficient artificial intelligence approach for early detection of cross-site scripting attacks, Decision Analytics Journal, Volume 11, 2024, 100466.*
8. *Wenbo Wang, Peng Yi, Huikai Xu, DoubleR: Effective XSS attacking reality detection, Computer Networks,Volume 251, 2024, 110567.*
9. *Abdelhakim Hannousse, Salima Yahiouche, Mohamed Cherif Nait-Hamoud, Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey, Computer Science Review, Volume 52, 2024, 100634.*
10. *Josh Hickling, What is DOM XSS and why should you care?, Computer Fraud & Security, Volume 2021, Issue 4, 2021, 6–10.*
11. *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.*
12. *Hassaan Siddiqui, Ferhat Khendek, Maria Toeroe, Microservices based architectures for IoT systems – State-of-the-art review, Internet of Things, Volume 23, 2023, 100854.*
13. *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.*
14. *Enrico Cambiaso, Luca Caviglione, Marco Zuppelli, DockerChannel: A framework for evaluating information leakages of Docker containers, SoftwareX, Volume 24, 2023, 101576.*
15. *Gianluca Turin, Andrea Borgarelli, Simone Donetti, Ferruccio Damiani, Einar Broch Johnsen, S. Lizeth Tapia Tarifa, Predicting resource consumption of Kubernetes container systems using resource models, Journal of Systems and Software, Volume 203, 2023, 111750.*
16. *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.*
17. *Miguel Correia, Wellington Oliveira, José Cecílio, Monintainer: An orchestration-independent extensible container-based monitoring solution for large clusters, Journal of Systems Architecture, Volume 145, 2023, 103035.*

PATTERN FOR SECURING WEB APPLICATIONS AGAINST XSS ATTACKS IN CLOUD INFRASTRUCTURE

Korneev N. V.[3], Lazorin D. S.[4]

The purpose of this article: *To develop a template protection mechanism to ensure the security of a web application in the event of a cross-site scripting threat.*

3 Nikolai V. Korneev, Dr.Sc., Professor, Gubkin Russian State University of Oil and Gas (National Research University), Moscow, Russia. E-mail: niccyper@mail.ru
4 Danil S. Lazorin, student, Gubkin Russian State University of Oil and Gas (National Research University), Moscow, Russia. E-mail: lazorindanya@yandex.ru

Research method: *Analysis of the principle of operation of cross-site scripting, in particular stored XSS. Synthesis of a cross-site script using two levels of protection: data encoding at the output and confirmation of input on arrival. Using methods of escaping in Unicode, blocking and applying several encoding levels in the correct order for invalid input of malicious code, operations are proposed to replace dangerous characters with suitable HTML mnemonics performed by full-scale modeling of a Docker-based web application in containerization-enabled environments, its deployment and testing under the threat of cross-site scripting.*

Result: *The analysis of cloud security of web applications is carried out and the relevance of the problem of developing universal template security mechanisms, called patterns for protecting a web application from XSS attacks, is shown. In particular, the principles of cross-site scripting, types of XSS attacks, and template mechanisms for protecting a web application from XSS attacks are considered pattern to protect the web application from XSS attacks. A pattern has been developed to protect a web application from XSS attacks based on microservices, taking into account a security service that includes protection mechanisms. On a practical example of a real web application, 4 containers (nginx, php, mysql, phpmyadmin) are deployed and interaction between them is configured. A registration form and an authorization form for a standard web application have been implemented. An XSS attack was performed on a web application using JavaScript code without a protection mechanism. The php code implements a security service to protect against XSS attacks using the built-in htmlspecialchars function. The program code of the service in the form of a function is given. The security service code includes an htmlspecialchars function with configuration code and interaction with the containers described above. A second XSS attack was performed, as a result of which the JavaScript code was not executed, the data was safely retrieved from the database. As a result, a line of code is obtained in the database, which is a sign of a diagnostic error of the web application, and can serve as a marker to monitor the XSS blocked attack. Open source software Kubernetes, Prometheus, Grafana, cAdvisor and Node Exporter were used to monitor the XSS attack. Manifests have been created to implement a basic cluster configuration consisting of a single pod with four containers. As a result, an XSS attack monitoring system was deployed and the ability to diagnose spikes in load changes as signs of a blocked XSS attack was shown.*

Practical value: *The practical value of the proposed solution includes a template protection mechanism in the form of a pattern. The pattern can be applied to a wide range of web applications, including transferring the developed solution to any industry: fuel and energy, economic and not only, due to the cross-platform nature of the solution itself.*

Keywords: *cloud computing, dataset, template, malicious code, security service, container, diagnostic error, marker, manifest, cluster, XSS attack monitoring system.*

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, Issue 2/2024, 2024, https://doi.org/10.1007/s10207-023-00797-6.*
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. *Faizan Younas, Ali Raza, Nisrean Thalji, Laith Abualigah, Raed Abu Zitar, Heming Jia, An efficient artificial intelligence approach for early detection of cross-site scripting attacks, Decision Analytics Journal, Volume 11, 2024, 100466.*
8. *Wenbo Wang, Peng Yi, Huikai Xu, DoubleR: Effective XSS attacking reality detection, Computer Networks,Volume 251, 2024, 110567.*
9. *Abdelhakim Hannousse, Salima Yahiouche, Mohamed Cherif Nait-Hamoud, Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey, Computer Science Review, Volume 52, 2024, 100634.*
10. *Josh Hickling, What is DOM XSS and why should you care?, Computer Fraud & Security, Volume 2021, Issue 4, 2021, 6–10.*
11. *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.*
12. *Hassaan Siddiqui, Ferhat Khendek, Maria Toeroe, Microservices based architectures for IoT systems – State-of-the-art review, Internet of Things, Volume 23, 2023, 100854.*
13. *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.*
14. *Enrico Cambiaso, Luca Caviglione, Marco Zuppelli, DockerChannel: A framework for evaluating information leakages of Docker containers, SoftwareX, Volume 24, 2023, 101576.*
15. *Gianluca Turin, Andrea Borgarelli, Simone Donetti, Ferruccio Damiani, Einar Broch Johnsen, S. Lizeth Tapia Tarifa, Predicting resource consumption of Kubernetes container systems using resource models, Journal of Systems and Software, Volume 203, 2023, 111750.*
16. *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.*
17. *Miguel Correia, Wellington Oliveira, José Cecílio, Monintainer: An orchestration-independent extensible container-based monitoring solution for large clusters, Journal of Systems Architecture, Volume 145, 2023, 103035.*