УДК 004.056

МЕТОДИКА ТЕСТИРОВАНИЯ НА ПРОНИКНОВЕНИЕ КОНТЕЙНЕРНОЙ ТЕХНОЛОГИИ KUBERNETES

Д.И. Денисенко
Студент группы М22-508 НИЯУ МИФИ, ddlreserch@gmail.com

Аннотация. При разработке клиент-серверных приложений повсеместно используют контейнерные технологии, призванные увеличить скорость разработки и минимизировать количество отказов программного обеспечения. Однако внедрение этих технологий создает уникальные проблемы в области безопасности. Учитывая их сильное влияние на приложения, крайне важно применять структурированную методику тестирования на проникновение, которая учитывает их отличительные особенности. Описанная методика тестирования на проникновение в значительной степени ориентирована на Docker и Kubernetes и предполагает глубокое понимание принципов их работы. В центре внимания методики оказался этап развертывания жизненного цикла разработки программного обеспечения, на котором происходит контроль и управление процессами развертывания контейнерных приложений. Несмотря на то, что методика разработана с учетом особенностей контейнерных сред, она не исключает интеграции с другими методиками, такими как методики тестирования сети или приложений. Методика может быть дополнена другими методиками для обеспечения комплексной оценки безопасности, так как она сосредоточена на среде контейнеризации и не учитывает специфику приложений, работающих в контейнерах. Тем не менее, она является самостоятельной и предоставляет методы тестирования в отсутствие дополнительных методик тестирования.

Ключевые слова: тестирование на проникновение, тестовый стенд, Kubernetes, Docker, кластер, атаки, безопасность, клиент-серверные приложения.

Введение

В области разработки клиент-серверных приложений обеспечение безопасности на протяжении всего жизненного цикла имеет большое значение. Поскольку клиент-серверные приложения становятся все более сложными возникает необходимость внедрения мер безопасности от начала разработки до этапа предоставления их конечному пользователю [1]. Согласно проекту документа от 19.12.2023 ГОСТ Р 56939 «Защита информации. Разработка безопасного программного обеспечения. Общие требования» [2] интеграция процессов обеспечения безопасности в процесс разработки имеет решающее значение для выполнения регулярных проверок кода и устранения возникающих в ПО уязвимостей [3]. В этом контексте конвейер непрерывной интеграции и непрерывной доставки ПО становится важным элементом, который обеспечивает быстрые циклы этапов разработки ПО и устранение ошибок на разных этапах, тем самым повышая уровень безопасности приложений. Технология Kubernetes [4-6] облегчает процессы масштабирования и развертывания приложений в различных средах, занимает основную роль в этом конвейере, обеспечивая надежную среду для контейнеризации приложений и управления ими соответственно. Обеспечение

безопасности должно производится на всех этапах жизненного цикла, для чего в документе содержатся общие требования к каждому этапу цикла. Однако методики оценивания соответствия реализации процессов разработки безопасного ПО требованиям не являются предметом рассмотрения данного документа и выбираются с учетом специфики конкретного приложения. В качестве такой методики возможно использовать методику тестирования на проникновение [7, 8].

Идентификация ресурсов

При обсуждении методики тестирования на проникновение контейнерных технологий важно учитывать две точки зрения: внутренний взгляд изнутри контейнера и внешний взгляд с хоста. Внутри контейнер воспринимается как изолированная среда, сродни автономному процессу. Такая изоляция ограничивает доступ, предоставляя доступ только к тем файлам и ресурсам, которые связаны с контейнером, позволяя выполнять команды исключительно в его пределах.

Внешнее тестирование на проникновение охватывает хост-систему, где как хост-процессы, так и контейнеры отображаются как процессы, находящиеся на хосте. Это позволяет видеть все доступные процессы на хосте, включая взаимодействие с демоном Docker [9, 10] и его дочерними процессами.

Проведение базовой идентификации доступных ресурсов в кластере Kubernetes необходимо для выявления потенциальных уязвимостей. Ниже представлен перечень базовых первостепенных проверок для получения первичной информации о кластере:

- для разрешения DNS-имен в пространстве имен Kubernetes по умолчанию с целью выявления внутренних IP-адресов используется команда

  nslookup kubernetes.default;

- сканирование портов сервера Kubernetes API с помощью команды

  nmap -p- <IP-адрес Kubernetes API>,

  чтобы обнаружить открытые порты и службы, которые могут быть уязвимы;

- пространства имен помогают организовать ресурсы внутри кластера; для их перечисления используется команда

  kubectl get namespaces;

- получение списка модулей в пространстве имен осуществляется выполнением команды

  kubectl get pods -n <namespace>;

- проверка версии сервера Kubernetes определяется посредством команды

kubectl version;

- чтобы составить список и проанализировать все развернутые образы контейнеров и их версии:

kubectl get services -A -o wide;

- дамп всех развертываний для анализа конфигураций выполняется командой

kubectl get deployments -A -o yaml > deployments.yaml;

- действия, разрешенные текущему пользователю, можно определить, выполнив

kubectl auth can-i –list;

- аудит настроек средств контроля доступа на наличие ролей с высокими привилегиями осуществляется путем получения информации с команд;
- вход в модуль для определения особенностей его функционирования осуществляется командой

kubectl exec -ti <pod_name> -n <namespace> -- /bin/sh;

- получение токена сервисной учетной записи модуля, который можно использовать для взаимодействия с API Kubernetes, через команду

kubectl exec -ti <pod> -n <namespace> -- cat /run/secrets/ kubernetes.io/ serviceaccount/token;

- используя ранее найденные токены, получить доступ к секретам в пространствах имен возможно выполнив команду

curl -v -H Authorization: Bearer <jwt_token>https://<master_ip> :<port>/api/v1/namespaces/ <namespace>/secrets;

В сценариях, где выполнение команды *kubectl* затруднено из-за ограничений на загрузку или установку утилиты kubectl, несмотря на наличие доступа к сети и необходимых разрешений или сертификатов для взаимодействия с API-сервером Kubernetes, для управления ресурсами кластера требуются альтернативные способы. Один из таких способов заключается в репликации функциональных возможностей команд *kubectl* путем прямого взаимодействия с API-сервером Kubernetes. Это позволяет выполнять задачи управления кластером без использования *kubectl*. Чтобы эмулировать команды *kubectl* важно понимать структуру запросов, которые генерирует *kubectl*. Для перечисления всех сетевых политик, применяемых в пространствах имен, определения правил маршрутизации трафика и сетевой сегментации необходимо выполнить команду

kubectl get networkpolicies --all-namespaces.

Используя результаты выполнения команды, необходимо проанализировать каждую сетевую политику, чтобы понять структуру маршрутизации трафика.

Повышение привилегий с учетной записью службы

Kubernetes использует для управления контейнерами сертификат учетной записи службы, который привязывается к контейнеру при инициализации модуля. В соответствии со стандартной конфигурацией сертификат и токен учетной записи службы обеспечивают связь с сервером API служб Kubernetes. Изначально учетные данные службы не содержат высоких привилегий. Однако администраторы кластера обладают возможностью повысить привилегии учетной записи службы через команду

kubectl create clusterrolebinding service-account-admin-binding --clusterrole=cluster-admin --serviceaccount=default: default.

Выполнение команды добавляет учетной записи службы роль администратора кластера, предоставляя административные привилегии в кластере.

Тестирование сервера API

Сервер API Kubernetes выступает в качестве центрального интерфейса для управления ресурсами кластера. Получение контроля над главным узлом, на котором хранятся файлы конфигурации и учетные данные администратора для сервера API, означают полную компрометацию кластера. Сервер API требует аутентификации для обеспечения безопасного доступа. Однако при неправильной настройке, когда сервер API запущен с флагами

--insecure-bind-address=0.0.0.0 и *--insecure-port=8080*,

это требование не соблюдается, что приводит к несанкционированному доступу.

Тестирование компонента kubelet

В Kubernetes каждый узел управляется службой kubelet. Порт 10250 служит важным каналом связи между kubelet и API-сервером Kubernetes, позволяя kubelet получать информацию о задачах, которые ему необходимо выполнить. Администраторы Kubernetes могут включить анонимный доступ в определенных конфигурациях, что можно проверить командой

curl -k https://<kubernetes-node-ip>:10250/pods.

Используя конечную точку */pods*, становится возможным получение подробной информации о кластере, такой как пространства имен, идентификаторы модулей и конфигурация контейнера. Кроме того, возможно исполнение команд в модулях, используя команду

curl -k https://<kubernetes-node-ip>:10250/run/<namespace>/<pod>/<container> -d cmd=<command>.

Тестирование панели мониторинга Kubernetes

В Kubernetes панель мониторинга предлагает веб-интерфейс пользователя, облегчающий управление кластерами Kubernetes. Несмотря на наличие аутентификации, поддерживающей вход в систему с помощью файлов конфигурации или токенов, конфигурацией панели мониторинга можно манипулировать, чтобы обойти аутентификацию с помощью опции *enable-skip-login*.

Это позволяет пользователям получать доступ к панели мониторинга без проверки подлинности. Однако доступ к панели мониторинга не означает предоставление привилегий для управления кластером. Kubernetes использует управление доступом на основе ролей для управления разрешениями, где параметры доступа определяется ролями, назначенными различным учетным записям служб. По умолчанию учетная запись службы, связанная с информационной панелью Kubernetes, обладает ограниченными привилегиями, что ограничивает ее способность выполнять задачи по управлению кластером, если явно не настроено иное. Распространенная практика заключается в повышении уровня доступа к панели мониторинга путем привязки роли администратора кластера к учетной записи службы. После применения этой конфигурации доступ к панели мониторинга обеспечивает полный административный контроль над кластером.

Тестирование базы данных etcd

База данных etcd – это распределенное хранилище ключей и значений, используемое Kubernetes и другими системами для хранения важных данных и управления ими. Инструмент *etcdctl* предоставляет интерфейс командной строки для взаимодействия с базой данных, позволяя выполнять различные операции, такие как чтение и запись ключей. Обычно распределенное хранилище ключей использует порт 2379 для взаимодействия с клиентами. Доступ к этому порту через общедоступную сеть без надлежащего контроля доступа может привести к утечке конфиденциальной информации. Kubernetes использует etcd в качестве хранилища данных по умолчанию, где хранятся все данные конфигурации кластера, его состояние и метаданные. Получив доступ к etcd, возможно использовать несколько тактик для повышения привилегий и получения контроля над кластером Kubernetes. Основной метод заключается в извлечении токенов, хранящихся

в etcd, которые используются для аутентификации на сервере API

Заключение

Описанная методика ориентирована на Kubernetes и предполагает глубокое понимание принципов его работы. В центре внимания методики находится этап развертывания жизненного цикла разработки ПО, на котором происходит контроль и управление процессами развертывания контейнерных приложений. Несмотря на то, что методика разработана с учетом особенностей контейнерных сред, она не исключает интеграции с другими методиками и может быть дополнена другими методиками для обеспечения комплексной оценки безопасности. Методика сосредоточена на среде контейнеризации и не учитывает специфику приложений, работающих в контейнерах.

Список литературы

1. ГОСТ Р ИСО/МЭК 27000-2021 «Информационные технологии. Методы и средства обеспечения безопасности. Система менеджмента информационной безопасности. Общий обзор и терминология». – М.: Стандартинформ, 2021.
2. Проект национального стандарта ГОСТ Р 56939 «Защита информации. Разработка безопасного программного обеспечения. Общие требования». [Электронный ресурс]. https://fstec.ru/tk-362/standarty/proekty/proekt-natsionalnogo-standarta-gost-r-56939 (Дата обращения: 7.03.2024).
3. ГОСТ Р 50922-2006 «Защита информации. Основные термины и определения». Введ. 2008-02-01. – М.: Стандартинформ, 2008.
4. Исследование VK Cloud – Kubernetes в России. 2023-2024 гг. [Электронный ресурс]. https://cloud.vk.com/promopage/state-of-kubernetes/. (Дата обращения: 7.03.2024).
5. State of Kubernetes Security Report. 2023 г. [Электронный ресурс]. https://www.redhat.com/en/resources/state-kubernetes-security-report-2023. (Дата обращения: 7.03.2024).
6. Center for Internet Security. CIS Kubernetes Benchmark. 2022, pp. 67-70.
7. Journal of Cybersecurity. Recent advances in penetration testing. 2022, pp. 90-94.
8. OWASP Foundation. Web Security Testing Guide. [Электронный ресурс]. https://owasp.org/www-project-web-security-testing-guide/stable/. (Дата обращения: 27.02.2024).
9. Docker Documentation. Docker Security. [Электронный ресурс]. https://docs.docker.com/engine/security/. (Дата обращения: 25.03.2024).
10. Center for Internet Security. CIS Docker Benchmark. 2022, pp. 78-83.