УДК 004.724; 004.72

Дьяконов В.С.,
старший инженер отдела кибербезопасности ПАО «Сбербанк»
Россия, г.Барнаул
Деева К.С.,
студент
1 курс, факультет «Информационных технологий»
Алтайский Государственный Технический Университет им. И.И. Ползунова
Россия, г. Барнаул
Научный руководитель Шарлаев Е.В.,
кандидат технических наук, доцент кафедры ИВТиИБ Алтайский Государственный Технический Университет им. И.И. Ползунова
Россия, г. Барнаул

РАССЛЕДОВАНИЕ ИНЦЕДЕНТА В УСЛОВИЯХ КОНТЕЙНИРИЗАЦИИ И ОРКЕСТРАЦИИ

Аннотация: *Статья посвящена фундаментальным вызовам, которые технологии контейнеризации (Docker) и оркестрации (Kubernetes) создают для цифровой криминалистики. В работе рассматривается парадигмальный сдвиг от статичной инфраструктуры к эфемерным, распределенным средам, где классические методологии и инструменты сбора доказательств оказываются несостоятельными. Анализируется архитектурная сложность контейнерных систем, включая многоуровневую абстракцию, проблемы сохранения целостности данных и исчезновение "сцены преступления". Предлагается структурированный многоуровневый фреймворк для расследования инцидентов, основанный на проактивном мониторинге, сборе*

артефактов на всех уровнях оркестрации и адаптированных критериях достоверности доказательств в облачных managed-сервисах.

Ключевые слова: *цифровая криминалистика, Docker, Kubernetes, контейнерная форензика, расследование инцидентов, оркестрация, артефакты, облачные среды.*

Annotation: *The article is devoted to the fundamental challenges that containerization (Docker) and orchestration (Kubernetes) technologies pose for digital forensics. The paper examines the paradigm shift from static infrastructure to ephemeral, distributed environments where classical methodologies and evidence collection tools prove inadequate. The architectural complexity of container systems is analyzed, including multi-layered abstraction, data integrity preservation issues, and the disappearance of the "crime scene." A structured multi-level framework for incident investigation is proposed, based on proactive monitoring, artifact collection at all orchestration levels, and adapted evidence admissibility criteria in managed cloud services.*

Key words: *digital forensics, Docker, Kubernetes, container forensics, incident investigation, orchestration, artifacts, cloud environments.*

Современные среды контейнеризации и оркестрации, олицетворяемые технологиями Docker и Kubernetes, представляют собой радикальный отход от статичной инфраструктуры, для исследования которой были разработаны классические методы цифровой криминалистики. рассмотрим фундаментальные вызовы, которые эти технологии создают для специалистов по расследованию инцидентов, фокусируясь на парадигмальных сдвигах, архитектурной сложности и несостоятельности устаревших инструментов.

Классическая криминалистика базируется на принципе целостности и относительной стабильности исследуемой системы (жесткий диск, память, логи сервера). В мире контейнеров эти основы подвергаются ревизии.

Виртуальная машина (ВМ): Представляет собой полноценную изолированную систему с собственной ОС, работающей поверх гипервизора. Форензика ВМ аналогична исследованию физического сервера: есть долгоживущая файловая система, стабильные процессы, четкие сетевые настройки. Снимок (snapshot) ВМ дает полную, статичную картину состояния.

Контейнер: Представляет собой изолированный процесс в пространстве пользователя хостовой ОС, разделяющей с ним ядро. Ключевое отличие — отсутствие своей ОС. Контейнер — это набор изолированных пространств имён (процессов, сети, файловой системы) и контрольных групп (cgroups). Его состояние по умолчанию эфемерно и изменчиво.

Ключевые свойства среды, усложняющие расследование:

- Эфемерность (Ephemerality): Время жизни контейнера может исчисляться секундами или минутами (автоскейлинг, сбои, обновления). Практика «cattle vs. pets» означает, что неисправный контейнер не лечится, а заменяется новым. Это уничтожает критически важные артефакты: содержимое оперативной памяти, временные файлы, сетевые соединения.
- Неизменяемость образов (Immutability): Идеология предполагает, что запущенный контейнер не модифицируется. Все изменения должны вноситься в образ и приводить к пересозданию контейнера. Однако злоумышленник или уязвимость могут изменить состояние запущенного контейнера, создавая расхождение между образом и реальностью. Понимание этого расхождения становится сложной задачей.
- Высокая плотность (High Density): На одном хосте могут одновременно работать десятки или сотни контейнеров от разных приложений. Изоляция артефактов, относящихся к конкретному инциденту, требует тонкой фильтрации данных хостовой ОС.
- Многоуровневая абстракция (Layered Abstraction): Для доступа к данным контейнера необходимо пройти через уровни: оркестратор (K8s) -> среда выполнения (Docker/containerd) -> хостовая ОС. Каждый уровень имеет

свои логи и метаданные, и цепочка событий должна быть восстановлена через их корреляцию.

- Проблема «исчезающей сцены преступления»: Традиционный подход «отключи и исследуй» ведет к полной потере контекста в контейнерной среде. Остановка (docker stop) или удаление контейнера стирает его файловую систему (если она не примонтирована извне) и, что важнее, всю резидентную в памяти информацию. Таким образом, сам акт сохранения доказательств в классическом понимании уничтожает ключевые доказательства.

Kubernetes добавляет дополнительный, критически важный уровень сложности, трансформируя набор контейнеров в интеллектуальную, самоисцеляющуюся распределенную систему.

Модель разделённой ответственности: В managed-сервисах (EKS, GKS, AKS) провайдер облака отвечает за контрольную плоскость (master nodes), безопасность хостовой ОС и гипервизора. Команда клиента (DevOps/SecOps) несет ответственность за конфигурацию рабочей нагрузки (Pods, Deployments), сетевые политики, безопасность образов и данных приложения. Это размывает границы расследования и доступ к данным: журналы уровня гипервизора или физических нод могут быть недоступны для специалиста по ИБ компании.

Многослойность данных: трассировка артефактов:

- Уровень оркестратора (K8s API): Содержит «историю правды» о желаемом состоянии системы. Аудит-логи API-сервера — ключевой источник, показывающий, кто, что, когда и как делал в кластере (создание подов, доступ к секретам). Однако они логируют лишь намерения, а не фактические действия внутри контейнера.
- Уровень среды выполнения: Docker/containerd хранят свои внутренние логи и метаданные, не всегда полностью отражающиеся в событиях K8s.

• Уровень хостовой ОС: Все контейнеры в конечном счете выполняются здесь. Артефакты (процессы, сетевые сокеты, файлы из overlay-файловых систем) смешаны, требуя фильтрации по идентификаторам контейнеров (PID, container_id).

Динамическая сеть и хранение данных:

• Сеть: Традиционный анализ сетевых соединений (netstat) на хосте бесполезен из-за CNI-плагинов, создающих виртуальные сетевые интерфейсы и пространства имён для каждого пода. Service Mesh (Istio, Linkerd) добавляет уровень шифрования (mTLS), делая перехват и анализ трафика между подами крайне сложным без доступа к sidecar-прокси.

• Хранение: Конфиденциальные данные (секреты, конфигурации) хранятся в зашифрованном виде в etcd — распределенном key-value хранилище контрольной плоскости. Их извлечение и анализ требуют доступа к мастер-нодам и ключам шифрования. Постоянные данные приложения (Persistent Volumes) могут находиться на удаленных облачных дисках, отделяя данные от среды выполнения.

Классическая цифровая криминалистика сформировала мощный арсенал методологий (например, процессы, описанные в руководствах NIST или RFC 3227) и инструментов, отточенных на исследовании физических серверов, рабочих станций и статичных виртуальных машин. Однако их применение в динамической, высокоабстрактной среде контейнеров и оркестраторов выявляет фундаментальную архитектурную несовместимость. Это ограничение носит не технический, а парадигмальный характер.

Традиционные инструменты, такие как EnCase, FTK (Forensic Toolkit), Autopsy или даже стандартные средства командной строки Linux, ориентированы на исследование целостных, монолитных систем.

Непонимание концепции образов и слоёв:

• Инструмент: Классический анализатор дисков видит контейнерный образ (например, app.tar) как единый файл-архив. Он не имеет

встроенных парсеров для формата OCI (Open Container Initiative) Image и не может интуитивно отображать иерархию слоёв (layers) Docker/OCI.

- Проблема: Для анализа уязвимости или вредоносного файла в образе эксперт вынужден вручную извлекать и анализировать каждый слой, теряя контекст их наложения (файл в верхнем слое перекрывает файл в нижнем). Специализированные инструменты вроде dive или trivy делают это нативно, а EnCase требует сложных процедур обхода.

Отсутствие контекста контейнерной среды выполнения:

- Инструмент: Анализатор памяти (Volatility) или инструмент для анализа процессов (ps, lsof) на хостовой ОС видят процессы контейнеров как обычные процессы. Они не имеют встроенных плагинов для автоматического связывания хостового PID с идентификатором контейнера (container_id), именем Pod'a и namespace Kubernetes.
- Проблема: Чтобы доказать, что процесс curl, делающий вредоносные HTTP-запросы, принадлежит конкретному контейнеру backend-fraudulent, а не системной службе, аналитик должен вручную сопоставлять PID с выводами crictl ps или docker ps, что крайне трудоемко в масштабе.

Неспособность работать с эфемерными файловыми системами:

- Инструмент: Файловые анализаторы рассчитаны на статические разделы (ext4, NTFS). Файловая система запущенного контейнера часто является комбинацией overlay2 или aufs — объединением read-only слоев образа и read-write слоя контейнера, смонтированных в каталоге хоста (/var/lib/docker/overlay2/...).
- Проблема: Прямой доступ к этим файлам с хоста нарушает их целостность (изменяются метаданные времени доступа). Традиционные инструменты не понимают эту структуру "на лету" и не могут безопасно экспортировать consistent snapshot файловой системы контейнера без его остановки.

Классическая методология следует строгому порядку: идентификация -> сохранение -> анализ -> отчет. В Kubernetes эта последовательность рушится.

Принцип «Сначала сохрани» (Order of Volatility) вступает в конфликт с «не тронь работающую систему»:

- Методология: RFC 3227 предписывает собирать наиболее летучие данные первыми: содержимое RAM, таблицы сетевых соединений. В Kubernetes для этого нужно зайти внутрь работающего контейнера или сделать дамп памяти его процесса с хоста.
- Конфликт: Любое вмешательство (установка инструментов в контейнер, присоединение отладчика) изменяет его состояние, нарушая принцип неизменности доказательств. Более того, сам факт сбора (например, создание дампа памяти) может привести к перезагрузке контейнера из-за исчерпания лимитов памяти.

Невозможность «заморозки» сцены преступления:

- Методология: В физической системе можно изолировать сервер от сети и отключить для создания полного бинарного образа диска.
- Конфликт: В Kubernetes «сервер» — это абстракция. Pod — эфемерный. Его «отключение» (kubectl delete pod) ведет к безвозвратному уничтожению доказательств. А изоляция всего узла нарушает работу десятков других неповинных подов, что неприемлемо в продакшене. Сцена преступления самоуничтожается по дизайну через механизмы самовосстановления (restartPolicy, livenessProbe).

Расследование в Kubernetes — это всегда расследование распределённой системы.

Централизованный сбор и распределенная природа кластера:

- Традиционный подход: Аналитик подключается к одному серверу и собирает с него логи, файлы, дампы памяти.
- Проблема в K8s: Инцидент может затрагивать:

- Множество подов одного Deployment'a, разбросанных по разным узлам.
- Уровень оркестратора (мастер-ноды), где хранятся аудит-логи и секреты.
- Сетевые компоненты (логи Calico, Istio), работающие в своих собственных namespace'ax.
- Итог: Не существует единой SSH-сессии, с которой можно собрать полную картину. Требуется скоординированный параллельный сбор с десятков или сотен источников данных, что невозможно выполнить вручную.

Динамическая идентификация целей для сбора:

- Методология: В статичной среде цель расследования известна и неизменна (сервер srv-01).
- Проблема в K8s: Вредоносный Pod может быть удален системой автоскейлинга или оркестратором через секунды после атаки. Новый Pod, уже чистый, создается на другом узле. Аналитик должен ретроспективно выяснить, на каких узлах он работал, какие идентификаторы контейнеров имел, чтобы найти его следы в централизованных журналах хостов или системах мониторинга. Это задача динамического поиска и корреляции, а не простого сбора.

Для преодоления вызовов, описанных ранее, необходима новая методологическая основа. Рассмотрим структурированный подход, основанный на многоуровневом фреймворке сбора артефактов, адаптированном жизненном цикле расследования и четких критериях достоверности доказательств в распределенной среде.

Фундаментальный принцип — расследование должно вестись сверху вниз, от уровня абстракции оркестратора к уровню хостовой ОС, с четким пониманием связей между слоями. Каждый уровень предоставляет уникальные артефакты, которые вместе формируют целостную картину.

Уровень 1: Оркестратор (Kubernetes Control Plane) – «Источник правды» о намерениях.

Ключевые артефакты:

- Аудит-логи API-сервера (kube-apiserver audit logs): Самый критичный источник. Фиксируют все вызовы к API (кто (user), когда (timestamp), что (requestURI), с какими параметрами (requestObject), был ли успешен (responseStatus)). Позволяют отследить создание подозрительных подов, доступ к secrets, удаление логов, эскалацию привилегий через bind ролей.
- События Kubernetes (kubectl get events -A --sort-by='.lastTimestamp'): Поток событий о состоянии объектов (например, FailedScheduling, PullingImage, Killing). Помогают восстановить хронологию жизни пода.
- Манифесты и состояние объектов: Текущие и желаемые конфигурации (kubectl get -o yaml). Позволяют выявить аномалии: под с привилегированным режимом (privileged: true), монтирование директории хоста, неверные security contexts.
- Secrets и ConfigMaps (только метаданные): Факт их использования подом. Само содержимое, как правило, зашифровано в etcd.

Инструменты сбора: kubectl, kube-audit (для анализа), прямой доступ к лог-файлам мастер-нод или сбор через централизованную систему (Fluentd, Vector) при наличии.

Уровень 2: Контейнерная среда выполнения (Runtime) – «Свидетель событий» внутри контейнера.

Ключевые артефакты:

- Метаданные контейнера: docker inspect или crictl inspect предоставляют детали: ID образа, команду запуска, смонтированные тома, сетевые настройки, переменные окружения (включая потенциальные секреты).

- Стандартные логи приложения (stdout/stderr): Все, что контейнер выводит в консоль, перехватывается драйвером логов (json-file, journald). Это первое место для поиска ошибок или следов эксплойта.
- Измененные файлы контейнера (Diff): Сравнение текущей файловой системы контейнера с исходным образом (docker diff <container_id>). Показывает, какие файлы были созданы, изменены или удалены во время выполнения (например, появление скриптов, конфигов, бинарников).

Инструменты сбора: docker, containerd CLI (ctr, crictl). Критически важно собирать эти данные до остановки контейнера.

Уровень 3: Образ контейнера (Image) – «Исходный код» среды выполнения.

Ключевые артефакты:

- История сборки (Dockerfile через docker history): Показывает последовательность команд (RUN, ADD, COPY), которая привела к созданию образа. Может раскрыть внедрение вредоносного кода на этапе сборки.
- Анализ слоев: Инструменты вроде dive позволяют интерактивно изучать каждый слой образа, выявляя скрытые файлы, большие или подозрительные добавления.
- Статическое сканирование: Поиск известных уязвимостей (CVE) в установленных пакетах (trivy, grype), hard-coded секретов (пароли, ключи API) и соблюдения best practices.

Инструменты сбора: Реестр контейнеров (Harbor, ECR, GCR), инструменты статического анализа SAST/SCA.

Уровень 4: Хостовая ОС (Worker Node) – «Фундамент» для всех контейнеров.

Ключевые артефакты:

- Системные логи (journalctl): Логи ядра, системного демона (systemd), демона среды выполнения (containerd). Содержат записи о создании/удалении контейнеров, ошибках cgroups, сетевых взаимодействиях.
- Файловые системы контейнеров: Данные в директориях /var/lib/docker/overlay2/... или /var/lib/containerd/.... Позволяют получить прямой доступ к файлам контейнера для offline-анализа.
- Процессы и сетевые соединения хоста: ps auxf, ss -tulnp. Позволяют идентифицировать процессы контейнеров и их сетевую активность, но требуют сопоставления PID с container_id (через /proc/<pid>/cgroup).
- Временные файлы и дампы памяти (volatile): Могут быть получены с помощью gcore или crictl debug, но с риском нестабильности.

Инструменты сбора: Стандартные утилиты Linux, специализированные сборщики (например, агенты osquery).

Уровень 5: Сеть и данные (Network & Storage) – «Контекст» взаимодействия и хранения.

Ключевые артефакты:

- Логи CNI-плагина: Показывают создание сетевых интерфейсов, политики firewall, сетевые аномалии.
- Логи Service Mesh (Istio, Linkerd): Позволяют увидеть детализированную картину сервис-сервисного трафика (метрики, трейсы, ошибки), даже если он зашифрован mTLS.
- Дампы сетевого трафика: Захват пакетов (tcpdump) на интерфейсе cni0 или внутри специального диагностического пода (kubectl debug).
- Данные Persistent Volumes (PV): Фактические файлы приложения (БД, загруженный контент), которые переживают пересоздание подов. Могут содержать логи приложения, шелл-скрипты, инструменты злоумышленника.
- Состояние etcd: В крайних случаях — прямое исследование дампа etcd для анализа всех объектов кластера, включая историю.

Расследование превращается в непрерывный цикл с акцентом на проактивность.

Этап 1. Подготовка и проактивный мониторинг (ДО инцидента – ключевой этап).

- Настройка исчерпывающего аудита: Включение аудит-логирования Kubernetes с профилем, захватывающим все запросы (Metadata или RequestResponse).
- Централизованный сбор логов: Развертывание стека (EFK/ELK или Loki/Grafana) для агрегации логов со всех пяти уровней в единое хранилище с длительным сроком сохранения.
- Внедрение Runtime Security: Установка агентов (Falco, Tetragon), которые в реальном времени отслеживают системные вызовы и детектируют аномалии (несанкционированные процессы, изменение бинарников, подозрительные сетевые соединения), выступая как триггер для запуска расследования.
- Создание форензических «рычагов»: Подготовка скриптов и служебных подов (kubectl debug) для быстрого сбора артефактов.

Этап 2. Первоначальный ответ и сохранение состояния (Минуты после обнаружения).

Цель: Зафиксировать максимально летучие данные, не нарушая работу приложения.

Действия:

- Сохранение метаданных оркестратора: Немедленный снимок состояния объектов (kubectl get pod <pod-name> -o yaml > pod.yaml, kubectl describe).
- Экспорт состояния контейнера: docker export или crictl export для создания статичной копии файловой системы работающего контейнера.
- Сбор данных среды выполнения: Копирование логов и метаданных (docker inspect) со всех связанных контейнеров.

- Дамп памяти (если критично): Использование crictl debug или gcore для создания дампа процесса, с пониманием рисков.
- Изоляция через сеть: Применение NetworkPolicy для блокировки входящего/исходящего трафика подозрительного пода вместо его удаления.

Этап 3. Корреляционный анализ и восстановление цепочки событий.

- Сопоставление артефактов: Аналитик использует временные метки и идентификаторы объектов (namespace, pod name, container_id) для связи событий между уровнями. Пример: запрос в аудит-логе о создании пода -> событие PullingImage -> запись в логах containerd на хосте -> появление процесса.
- Использование матрицы MITRE ATT&CK® для Containers: Таксономия атак предоставляет структурированный подход. Например, для техники «Доступ к Kubernetes Dashboard» (T1600) аналитик ищет в аудит-логах аномальные запросы к api/v1/namespaces/kubernetes-dashboard/, а в системных логах — создание соответствующих сетевых соединений.
- Построение временной шкалы (Timeline): Создание единой хронологической последовательности событий от первоначального вектора атаки (образ, секрет) до финального воздействия (майнинг, утечка данных).

В распределенной облачной среде соблюдение принципов целостности, аутентичности и повторяемости доказательств требует новых подходов.

Обеспечение целостности данных в распределённой системе:

- Хеширование в точке сбора: Все собранные артефакты (логи, дампы, yaml-файлы) должны немедленно хешироваться (SHA-256) с фиксацией хеша в защищенном журнале действий.
- Цепочка хранения (Chain of Custody): Необходимо вести автоматизированный журнал, кто, когда и с какой целью инициировал сбор артефакта, используя аутентифицированные идентификаторы (например, IAM-роли в облаке, клиентские сертификаты Kubernetes).

- Использование неизменяемых хранилищ: Артефакты должны сохраняться в объектные хранилища (Amazon S3, Google Cloud Storage) с включенным версионированием и WORM-политиками (Write Once Read Many), предотвращающими удаление или изменение.

Вопросы аутентификации логов и артефактов в managed-сервисах:

Проблема: В EKS/GKE/AKS клиент не имеет прямого доступа к журналам уровня гипервизора или детальным логам мастер-нод. Провайдер управляет контрольной плоскостью.

Решение и проверка:

- Активное использование встроенного аудита: Managed-сервисы предоставляют возможность включать и экспортировать аудит-логи K8s API в облачные сервисы логов (Cloud Audit Logs, CloudTrail). Эти логи аутентифицированы и снабжены метаданными от провайдера.
- Верификация через цифровые подписи провайдера: Необходимо требовать от провайдера, чтобы ключевые системные логи (особенно касающиеся модификации инфраструктуры) были подписаны цифровой подписью для подтверждения их происхождения и неизменности.
- Калибровка времени и синхронизация: Все узлы кластера и системы сбора логов должны синхронизироваться с единым источником времени (NTP), а в метаданные логов должны включаться временные зоны и источники времени для обеспечения корректной временной шкалы в юрисдикционных расследованиях.

Предложенные многоуровневая модель, адаптированный жизненный цикл и критерии достоверности формируют целостный методологический каркас для контейнерной форензики. Этот подход смещает акцент с реагирования на проактивную подготовку, превращая расследование из хаотичного сбора данных в систематизированный, повторяемый процесс, результаты которого могут иметь доказательную силу.

Использованные источники:

1. Цифровая криминалистика : учебник для вузов / под редакцией В. Б. Вехова, С. В. Зуева. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 485 с. — (Высшее образование). — ISBN 978-5-534-21152-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/581669 (дата обращения: 29.06.2026).

2. Сети и телекоммуникации : учебник и практикум для среднего профессионального образования / под редакцией К. Е. Самуйлова, И. А. Шалимова, Д. С. Кулябова. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 464 с. — (Профессиональное образование). — ISBN 978-5-534-17310-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/587334 (дата обращения: 29.06.2026).

3. Основы Kubernetes // kubernetes URL: https://kubernetes.io/ru/docs/tutorials/kubernetes-basics/ (дата обращения: 29.06.2026).

4. Артефакты Linux для пентестера: от credentials до следов атаки // codeby URL: https://codeby.net/threads/artefakty-linux-dlya-pentestera-ot-credentials-do-sledov-ataki.92991/ (дата обращения: 29.06.2026).