УДК 004.056

ИНТЕЛЛЕКТУАЛЬНЫЕ МЕТОДЫ АНАЛИЗА ПОТЕНЦИАЛЬНЫХ УГРОЗ ДЛЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ НА KUBERNETES НА ОСНОВЕ FALCO И PROMETHEUS

Буй А. Т., Университет ИТМО
Научный руководитель: Иванов С. Е., доцент, кандидат физико-математических наук, Университет ИТМО

Введение.

Микросервисная архитектура на базе Kubernetes стала стандартом для облачных приложений за счет масштабирования и автоматического управления жизненным циклом контейнеров. Однако динамичность кластера (автоскейлинг, частые деплои, распределенность сервисов и зависимостей) увеличивает площадь атаки и усложняет мониторинг безопасности. Практика показывает, что традиционные методы, основанные на статических правилах и порогах метрик, часто дают либо пропуски сложных атак, либо высокий уровень ложных срабатываний и «усталость» операторов. Поэтому актуальна разработка прикладной схемы интеллектуального анализа угроз, которая использует данные наблюдаемости и при этом может работать близко к реальному времени.

Основная часть.

В работе рассматривается объединение двух комплементарных источников сигналов: runtime-событий безопасности и метрик мониторинга. В качестве runtime-сенсора используется Falco, фиксирующий потенциально опасные действия в контейнерах и на узле (запуск подозрительных процессов, обращения к чувствительным файлам, сетевые соединения). Семантическая ценность Falco повышается за счет предварительного тегирования на уровне правил: каждое событие содержит rule, priority и набор tags (например, network, discovery, filesystem, crypto, k8s). Количественный контекст состояния сервисов предоставляют метрики Prometheus (CPU, память, сетевой обмен, рестарты и др.), доступные как временные ряды через HTTP API.

Предлагается двухканальный подход к обнаружению угроз. Первый канал решает задачу распознавания известных сценариев (misuse detection) как задачу supervised-классификации на табличных признаках. Для этой цели выбран XGBoost, устойчивый к шуму и хорошо работающий на смешанных признаках. Вектор признаков формируется из атрибутов Falco (rule/priority/tags и Kubernetes-контекст) и агрегатов метрик Prometheus в окне времени вокруг события (например, средние и максимальные значения CPU/памяти, RX/TX). Второй канал ориентирован на неизвестные или слабосигнальные отклонения и реализует anomaly detection в режиме metrics-only с помощью Isolation Forest, что позволяет выявлять нетипичное поведение даже при отсутствии специфических срабатываний Falco.

Для получения корректной разметки данных в лабораторной среде реализована связка Simulator + Collector. Симулятор управляет экспериментом (run) и жизненным циклом сценариев (scenario), фиксируя start/end, целевой namespace/pod и метку класса tag. Collector (сервис в Kubernetes, например на FastAPI) принимает поток событий из Falco через Falcosidekick, нормализует и дедуплицирует данные, сохраняет исходный payload и контекст Kubernetes, после чего выполняет best-effort сопоставление события со сценарием. Приоритетным является marker-based метод, когда идентификатор сценария намеренно «проявляется» в наблюдаемых полях (cmdline, путь файла) и извлекается из output_fields Falco; в противном случае используется резервное сопоставление по пересечению времени и цели (time-overlap). Далее Collector обогащает событие агрегатами метрик Prometheus за заданный интервал вокруг времени события и сохраняет артефакты в хранилище (например, PostgreSQL) в виде

логической схемы experiments, scenarios, security_events, event_labels, event_metrics. Параллельно сервис построения окон метрик формирует скользящие окна для обучения и эксплуатации anomaly-канала.

В эксплуатационном режиме оба канала работают совместно: event-driven классификация уточняет тип угрозы по конкретным Falco-событиям, а window-driven анализ метрик выполняет непрерывный контроль поведения сервисов и дает ранние сигналы о деградации или злоупотреблении ресурсами. Практическая ценность схемы заключается в балансе точности и устойчивости: теги Falco повышают интерпретируемость решения и помогают сократить ложные тревоги, а метрики обеспечивают универсальный фон и позволяют обнаруживать аномалии вне заранее известных паттернов.

Выводы.

Предложенная схема интеллектуального анализа угроз для Kubernetes сочетает два базовых и практичных ML-метода (XGBoost и Isolation Forest) и опирается на стандартные источники наблюдаемости (Falco и Prometheus). Ключевым результатом является воспроизводимый контур формирования размеченного датасета и поток обработки событий: тегирование на уровне Falco-правил, централизованный сбор, привязка к ground-truth сценариям, обогащение метриками и подготовка признаков для обучения. Результаты могут быть использованы для построения системы мониторинга и реагирования с «лестницей» действий: от оповещения и приоритизации инцидентов до ограничительных мер (rate limiting, изоляция pod/namespace) при соблюдении защитных ограничений против ошибочных автоматических действий. Ограничения подхода связаны с дрейфом поведения при деплоях и изменении нагрузки, а также с необходимостью расширять набор сценариев и регулярно обновлять модели; в качестве дальнейшего развития целесообразно учитывать audit logs Kubernetes и улучшать корреляцию событий по контексту.

Список использованных источников.

1. Falco Documentation. The Falco Project. URL: https://falco.org/docs/
2. Prometheus Documentation: Introduction and HTTP API. The Prometheus Authors. URL: https://prometheus.io/docs/
3. Kubernetes Documentation: Auditing. The Kubernetes Authors. URL: https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/
4. XGBoost Documentation. URL: https://xgboost.readthedocs.io/
5. scikit-learn: IsolationForest. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html