

< Teach
Me
Skills />

DevSecOps, CI/CD

Вопросы по предыдущим темам или ДЗ

Mini-quize по прошлым темам:

1. Какой главный числовый показатель есть в SIEM системах?
2. С каких систем ИБ вы бы настроили отправку событий в SIEM?
3. Каким средством можно настроить отправку логов с linux систем?
4. Какие типы коннекторов в SIEM вы знаете?
5. Какой вид парсинга данных вы знаете?

Mini-quize по новой теме:

1. Кто такие DevOps?
2. Что такое CI/CD на ваш взгляд?
3. Какие средства ИБ можно внедрить в процесс безопасной разработки?
4. Что на ваш взгляд значит выражение - система контроля версий?
5. Можно ли проверить готовое приложение на уязвимости или неправильную сборку?

План занятия

1. Определения безопасной разработки и непрерывной интеграции
2. Различия Динамического, статического и бинарного анализа
3. SAST и DAST решения на рынке
4. Opensource Анализаторы кода
5. Какие сложности возникают при внедрении DevSecOps

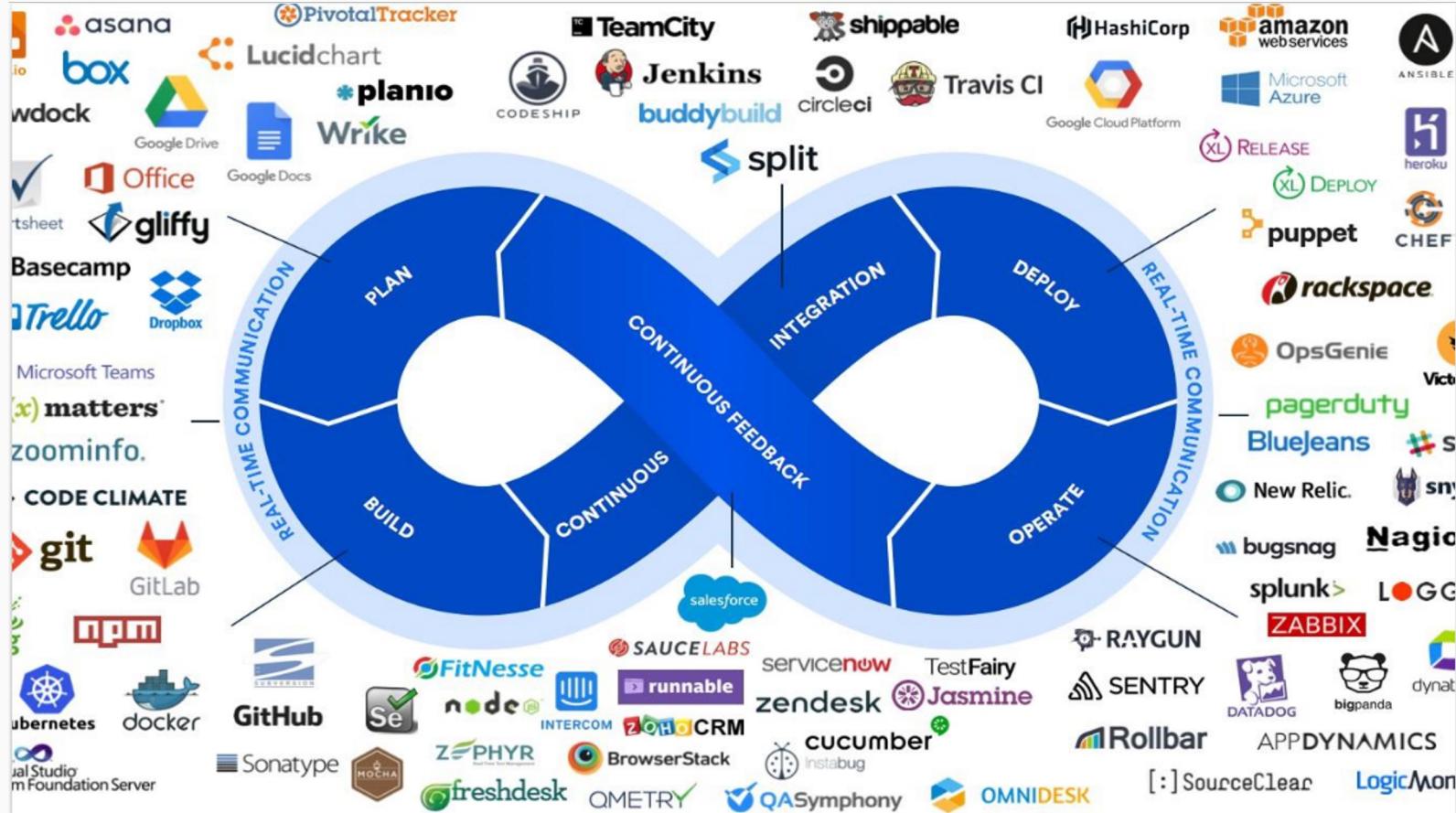
DevOps — это общий большой процесс, в рамках которого происходит активное взаимодействие между разработчиками и IT для повышения качества продукта

CI/CD - методология, относительно которой происходит построение этапов непрерывного цикла разработки и поддержки продукта

CI — это методика DevOps и этап [жизненного цикла DevOps](#), на котором разработчики регистрируют код в общем репозитории кода, часто по несколько раз в день. Желательно, чтобы каждый раз автоматический инструмент сборки проверял изменение или ветку на наличие ошибок и готовность к разработке. Отсюда и главное преимущество — проблемы выявляются на ранних этапах еще до того, как приведут к неприятным последствиям.

За CI следует непрерывная доставка (CD) — своего рода контрольный этап в процессе разработки перед выпуском и развертыванием итогового продукта для пользователей. После подтверждения изменения кода автоматически доставляются в репозиторий.

DevSecOps, CI/CD



Раньше DevOps существовал отдельно от Sec

Сейчас: DevSecOps

- Взаимодействие (Процессы, регламент)
- Люди
- Инструментарий

Направленный на устранение проблем
безопасности

«Цель и смысл внедрения DevSecOps — сформировать такой образ мышления, при котором каждый участник несет ответственность за безопасность.

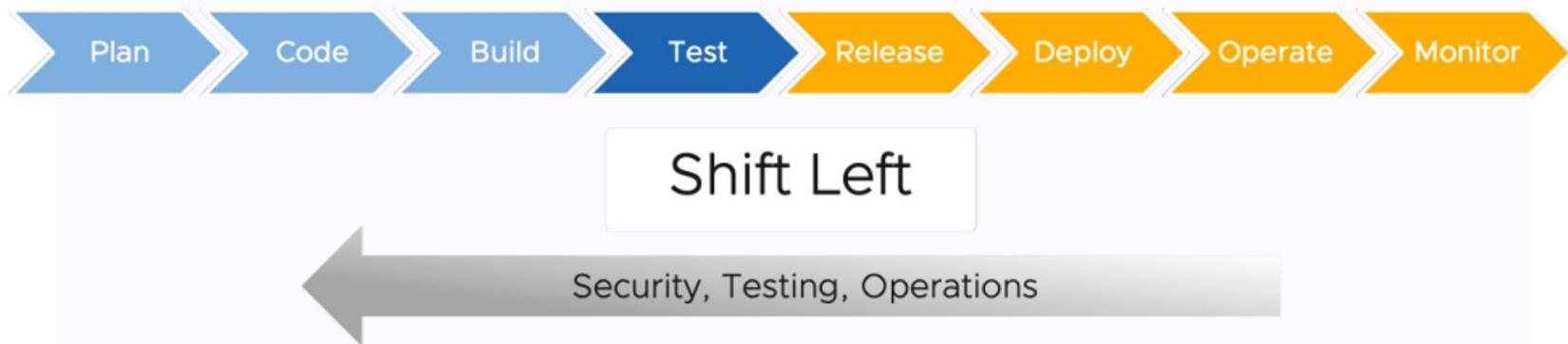
Это обеспечивает быструю и масштабируемую передачу решений по безопасности тем, кто лучше всего владеет контекстом, без ущерба для безопасности»

SSDL (SSDLC, SDL, SDLC - Secure Software Development LifeCycle) - специализированное понятие включающее в себя 2 предыдущих и обозначающее безопасный цикл разработки ПО

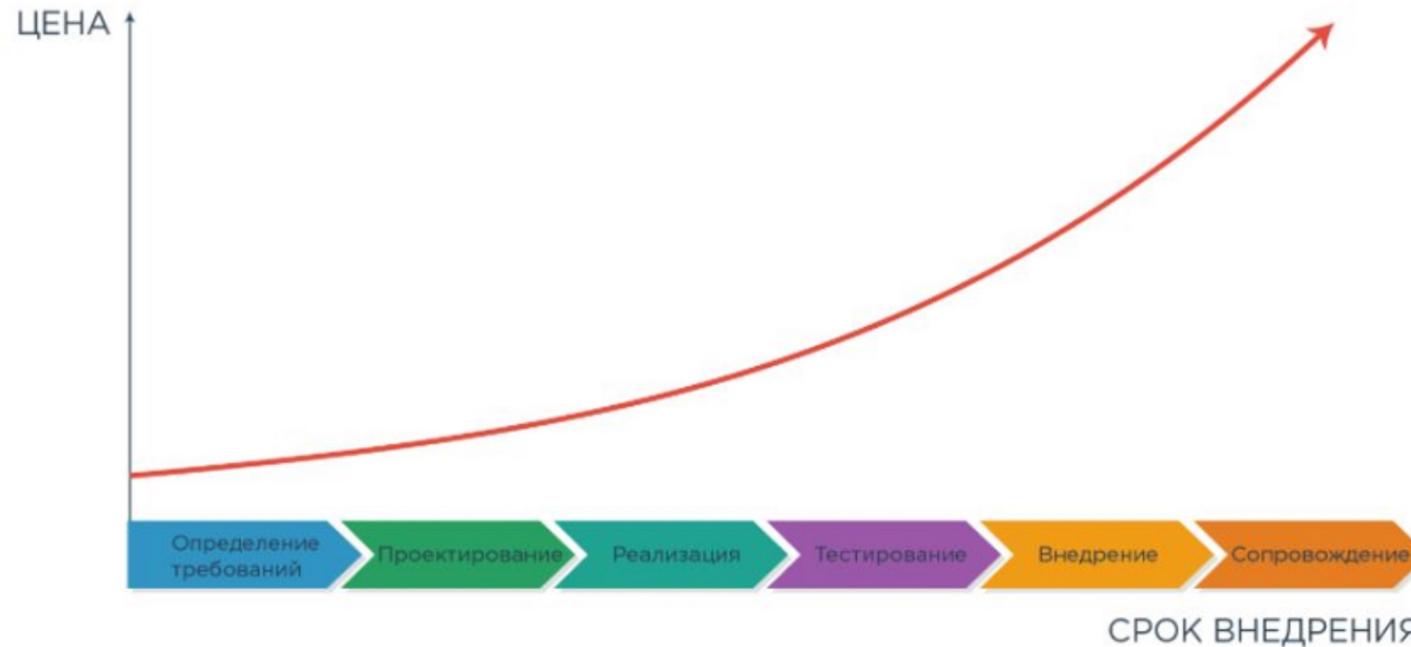
Application Security - общее понятие, включающее в себя все что относится к безопасности приложений и обозначает тип специалиста как отдельную отрасль

Shift-left подход

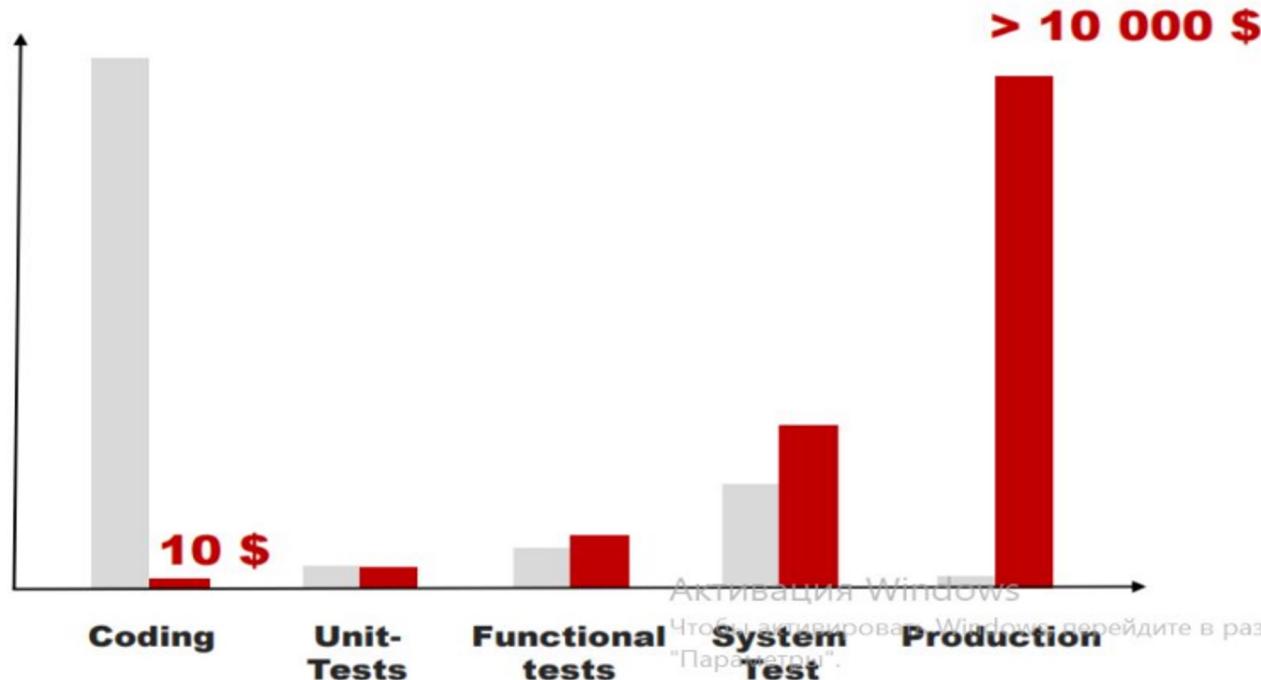
Обеспечить эффективную работу продукта и реализовать меры безопасности на этапах разработки и проектирования



Цена уязвимости



Стоимость бага



Успешное состояние для внедрения Sec в DevOps

- **Этап разработки:** обновление кода в центральном репозиторий
- **Этап CI:** сборка и тестирование (CI собирает приложение, выполняются функциональные тесты, статический анализ кода и модульные тесты безопасности)
- **Этап CD:** после успешного завершения тестов выполняется упаковывание и автоматическое развертывание приложения в рабочей среде.
- **Этап контроля:** выполняется контроль приложения в рабочей среде, чтобы убедиться, что все ее функции работают надлежащим образом.

Continuous Integration

Обновили Code - нужно собрать его в артефакт, перед этим необходимо запустить ряд проверок:

- Анализ кода на предмет наличия в нём лицензий
- Анализ зависимостей кода на предмет наличия в них уязвимостей
- SAST (IDE, CI)
- CodeReview
- Unit-, Автотесты
- Подпись артефактов

[Анализ PyPi](#) (46% Python библиотек содержат потенциально небезопасный код)

Continuous Integration

Так же мы сами можем выбрать политику работы Pipeline:

Останавливать сборку:

- при наличии проблем с автотестами
- при наличии уязвимостей определённого уровня
- при наличии проблем с лицензиями
- ошибки



Доставка и развертывание

Continuous Delivery/Deployment

- Доверие автотестам
- Быстрые релизы

*Некоторые продукты выполняют
деплой в prod несколько раз в день



Проблемы безопасности на этапах CD

- Необходимо тестирование уже запущенного приложения
- Необходима работа над безопасностью используемой инфраструктуры и среды запуска
- Уязвимости самих CI/CD решений

Уязвимости [GitLab](#)

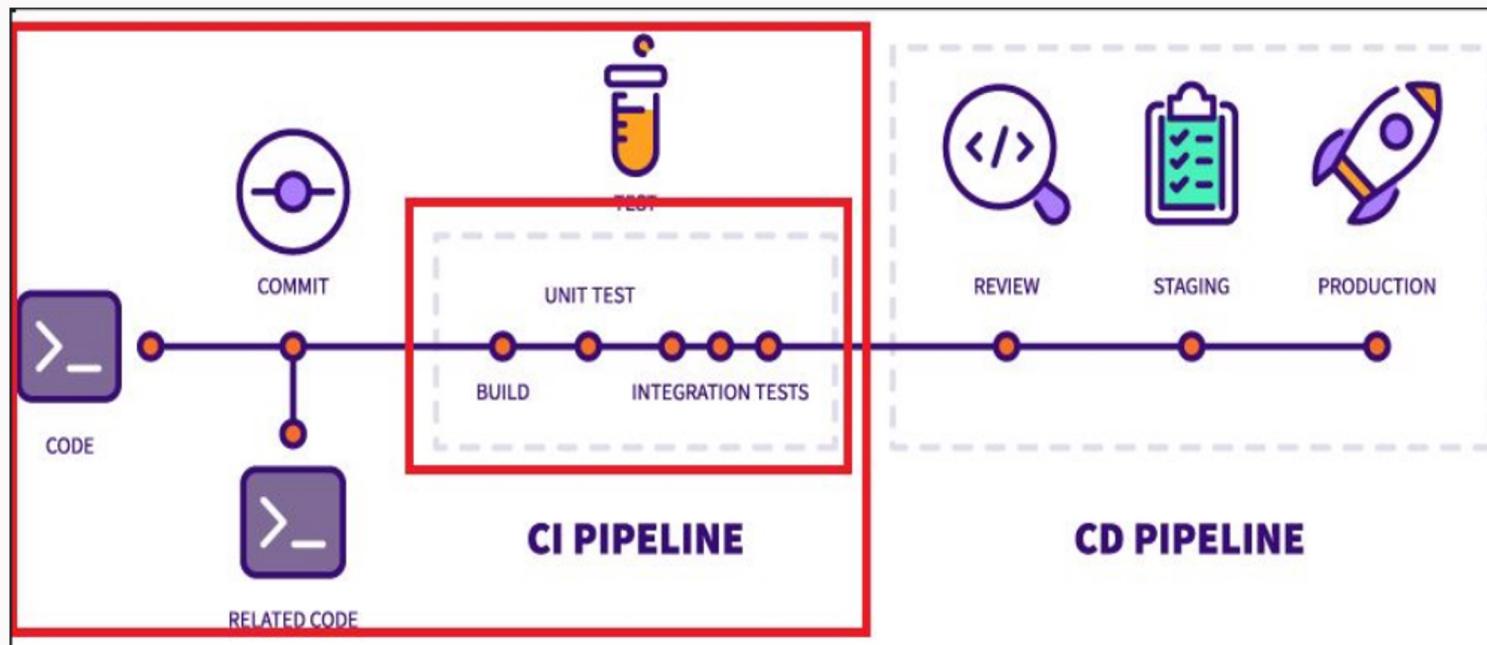
Защита среды контейнеризации

- Cluster Security
- Orchestrator Security
- Image Security
- Run-Time Security
- Access and Key Management Security
- Policy & Privileged Monitoring & Audit



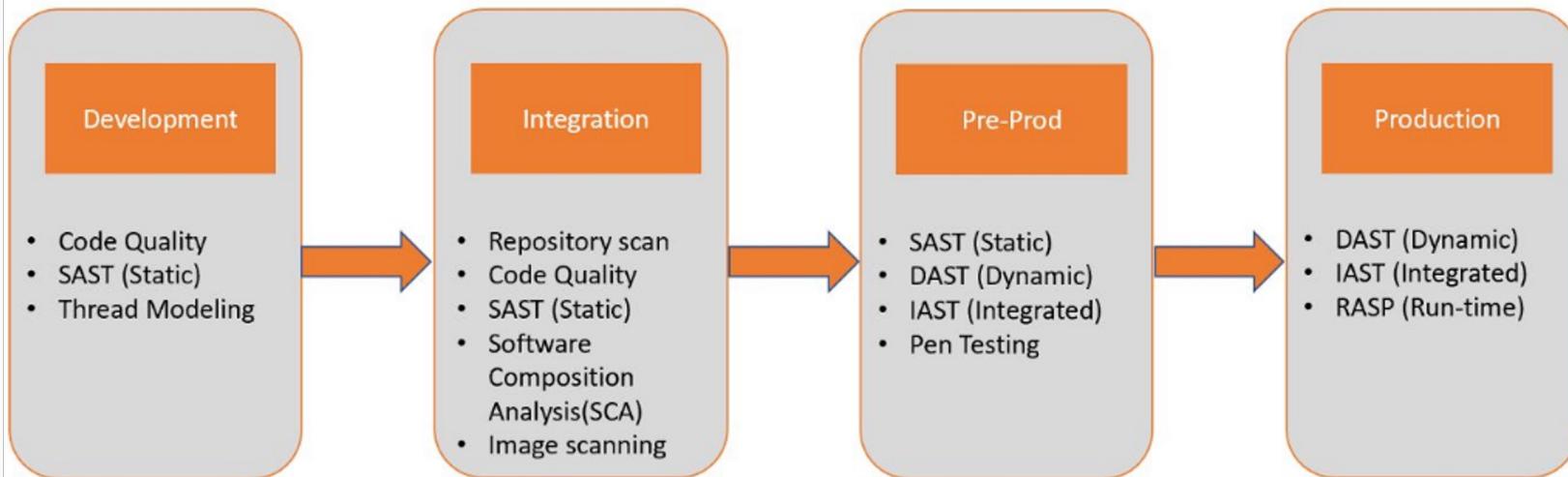
Тестирование и сборка

Базовый CI/CD Pipeline



Куда внедряется безопасность?

Application Security



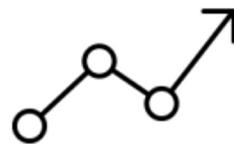
Виды проверок и их отличия

Для чего нужны проверки и анализ?

Проверки и анализ позволяют обнаружить ошибки, баги, codesmell, **уязвимости** на всех этапах жизненного цикла цифрового продукта - от исходного кода до эксплуатации

«Чем раньше обнаружить и исправить ошибку, тем дешевле это обойдется»

Енг Кек Нгуен, «*Testing computer software*»



Виды проверок и их отличия

Какие виды анализа существуют?

В сообществе выделяют **5** основных видов анализа цифрового продукта:

- **SAST** (статический анализ)
- **SCA** (анализ зависимостей)
- **DAST** (динамический анализ)
- **IAST** (интерактивный анализ)
- **RASP** (рантайм-защита)



Виды проверок и их отличия

SAST (Static Application Security Testing) — это анализ исходного кода или его составляющей на наличие уязвимостей без запуска исследуемого приложения на исполнение

Пример инструмента SAST-анализа: «**SonarQube**»:

- Инсталлируется на отдельный сервер приложения
- Требует наличия специального файла свойств в проекте ([sonar-project.properties](#))
- Интегрируется с инструментами SCA-анализа
- Поддерживает множество языков через подключаемые плагины
- Позволяет настроить **Quality Gate**



Виды проверок и их отличия

SCA-анализ

SCA (Software Composition Analysis) – анализ зависимостей проекта

Пример инструмента SCA-анализа: «**OWASP dependency-check**»:

Представляет собой утилиту, запускаемую на хосте, где осуществляется проверка

Устанавливается на постоянную или временную ноду CI/CD-раннера

По результатам проверки генерирует отчет в трех форматах (XML, JSON, HTML)

Интегрируется с инструментами SAST-анализа (например, с SonarQube)



Виды проверок и их отличия



DAST-анализ

DAST (Dynamic Application Security Testing) — динамический анализ исходного кода, тестирование «черного ящика»

Пример инструмента DAST-анализа для web-сайтов: «**OWASP ZAP (Zed Attack Proxy)**»:

- Имеет множество режимов запуска (краулинг, фаззинг, перехват запросов и проч.)
- Позволяет применять пассивное сканирование (без изменений данных)
- Может быть использован как инструмент для атаки (активное сканирование)

Виды проверок и их отличия

IAST-анализ



IAST (Interactive Application Security Testing) — метод интерактивного тестирования безопасности приложений

Пример инструмента IAST-анализа: «Fortify WebInspect» (в рамках единого комплексного решения от Fortify):

- Работает изнутри приложения, может анализировать код, потоки, конфиги, запросы и ответы, фреймворки
- Выдает более точные результаты, нежели SAST и DAST по отдельности
- Способен создать программу безопасности приложения на базе решения, а не отдельного продукта
- Покрывает более широкий спектр правил безопасности

Виды проверок и их отличия

RASP-анализ



RASP (Runtime Application Security(-Self) Protection) — защита безопасности приложения во время выполнения.

Пример инструмента RASP-анализ: «Application Security Monitoring» (AppDynamics by Cisco):

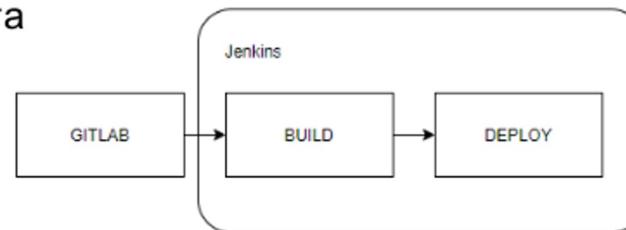
- Работает внутри приложения, как и IAST
- Подключается к приложению или его среде выполнения и контролирует его исполнение
- Защищает приложение без дополнительных брандмауэров или прокси
- Автоматически блокирует угрозы в режиме реального времени для защиты данных клиентов, корпоративной интеллектуальной собственности
- Упрощает жизненный цикл исправлений уязвимостей и предотвращает известные эксплойты

С чего начинать внедрение

С чего начинать внедрение инструментов анализа?

Для начала, донесите до руководства компании (или команды) текущую ситуацию со сборкой и доставкой исходного кода продукта:

- Код хранится в Gitlab
- Сборка и деплой на стороне Jenkins
- Проверки исходного кода и артефактов отсутствуют
- ИБ не участвует в цикле разработки продукта

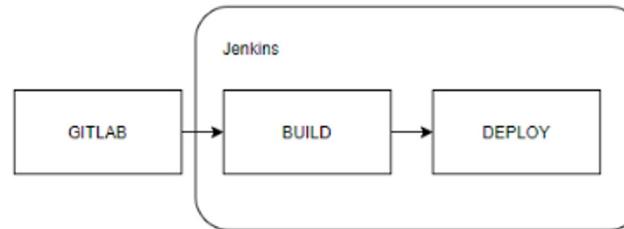


С чего начинать внедрение

С чего начинать внедрение инструментов анализа?

Выделите и подсветите проблемы текущего подхода:

- **Отсутствуют** проверки исходного кода и артефактов на предмет уязвимостей
- Сотрудник ИБ **не участвует** в процессе разработки
- Сотрудник ИБ вынужден **повторно** прорабатывать модель угроз приложения, из-за того что не участвует в процессе
- Стоимость и сроки исправления ошибок и уязвимостей **увеличивается**
- Приложение **не защищено** от кибер-угроз после выпуска



С чего начинать внедрение

С чего начинать внедрение инструментов анализа?

Донесите мысль о том, что внедрение проверок важно и необходимо, а сам процесс – итеративен и прост:

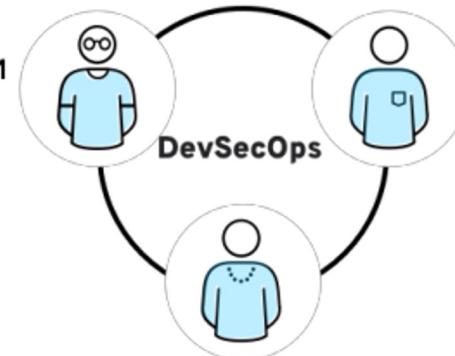
- **Интеграция** в существующий процесс проверок по информационной безопасности необходим
- **Внедрение** практик DevSecOps это ключ к безопасности
- Практика активного взаимодействия разработки и ИБ **на всех этапах** жизненного цикла продукта
- **Поэтапное** наращивание количества проверок
- Развитие **культуры** разработки безопасного продукта, развитие кибер-иммунитета

С чего начинать внедрение

С чего начинать внедрение инструментов анализа?

Не забудьте упомянуть о плюсах перехода к практикам DevSecOps, которые:

- Позволяют **расширить** существующий цикл разработки продукта безболезненно
- **Интегрируют** проверки по информационной безопасности
- Налаживают **взаимодействие** между командами
- Позволяют **своевременно** обнаруживать проблемы
- **Сокращают** длительность и стоимость исправления ошибок
- Увеличивают **ценность** продукта и его защищенность
- **Митигируют** риски появления и эксплуатации уязвимостей в продукте



Подробности SAST-SCA-анализа

Как внедрить совокупный SAST-SCA-анализ

SAST (Static Application Security Testing) — это анализ кода или его части на наличие уязвимостей без запуска исследуемого приложения на исполнение.

Может быть реализован с помощью сервисов **SonarQube** и **Dependency-check**. Для примера, возьмем сборщик **Jenkins**

Краткий план основных действий:

- Развернуть сервис [Sonarqube](#) (например, в k8s)
- Настроить [Quality Gate](#) в [Sonarqube](#)
- Добавить в проект файл свойств [Sonarqube](#) ([sonar-project.properties](#))
- Скачать и установить утилиту [Dependency-check](#)
- Скачать плагин в [Sonarqube](#) для интеграции с [Dependency-check](#)
- Скачать в [Jenkins](#) плагины для работы с [Sonarqube](#) и [Dependency-check](#)
- Настроить в [Jenkins](#) подключение к [Sonarqube](#) и [Dependency-check](#)
- Добавить в [пайплайн](#) джоб статического анализа

Подробности SAST-SCA-анализа

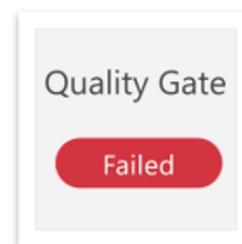
Quality Gate в SonarQube

Quality Gate – набор правил, определяющий **требования по безопасности** к программному продукту

Может включать в себя различные правила, объединенные логическим «И»:

- Степень покрытия тестами
- Количество найденных уязвимостей
- Количество найденных багов
- Количество обнаруженных code-smell срабатываний
- Общий уровень безопасности (Security Rating)

И многое другое...



SAST/DAST-анализ мобильных приложений

Mobile Security Framework (MobSF) – инструмент для проведения статического (Android и iOS) и динамического (только для Android) анализа мобильных приложений

Достоинства:

- Возможность проведения интеллектуального статического и динамического анализа мобильных приложений
- Открытый исходный код
- Поддержка множества вариантов развертывания (от standalone под windows до развертывания в k8s)
- Простота установки и использования
- Возможность интеграции с CI/CD-конвейерами



SAST/DAST-анализ мобильных приложений

На что обратить внимание при сканировании мобильных приложений?

- Под какую версию ОС рассчитано использование мобильного приложения?
- Каким сертификатом подписано приложение?
- Какие пермишены имеет приложение, совпадают ли они со спецификой использования?
- Не содержится ли уязвимостей в исходном коде приложения?
- Анализ манифестов и базовых конфигов не предполагает ли незащищенных соединений?
- Нет ли возможности у приложения получить сенситивную информацию о пользователе или его устройстве?

Основные этапы DevSecOps

- Планирование (Planing)
- Написание кода (Code)
- Тестирование и Сборка (Build & Testing)
- Доставка и развертывание (Delivery/Deployment)
- Мониторинг и защита приложения (Security Operations)

Составляющие планирования

- Оценка стоимости приложения
- Архитектурный анализ
- Моделирование угроз
- Анализ стека технологий (Inventory)
- Анализ требований законодательства (в зависимости от обрабатываемых данных)
- Анализ текущих процессов с точки зрения защищенности
- Обработка обратной связи с текущей итерации цикла DevOps



* Получив все текущие показатели - можем оценить уровень зрелости компании и заложить перспективы для повышения (Road Map)



Участники

- CISO
- Security - архитектор
- Security champions (как источник обратной связи)

Основная задача - обоснование необходимости повышения затрат на процесс.

Предусмотрение

- Patch-management процесс
- Приоритизация
- План по критическим ситуациям (Emergency Issues)
- Внутренний регламент
- Политика релизов



Написание кода

Разработка ПО и код

В основе всего лежит разработка программного обеспечения:

- языки программирования;
- библиотеки и фреймворки;
- инструменты сборки и управления зависимостями;
- системы управления версиями;
- инструменты анализа.
- среды запуска готового приложения

Потенциально возможный сценарий, работая в организации в рамках какой либо проверки 1ого из 50 проектов компании мы обнаруживаем:

- 1.Одна из используемых библиотек подвержена SQL-инъекции, также во фронт части приложения имеются проблемы с GET параметрами и валидацией инпута
- 2.Используются хэш функции MD-5 в БД
- 3.Неправильно используется функция округления в приложении для перевода средств (round())

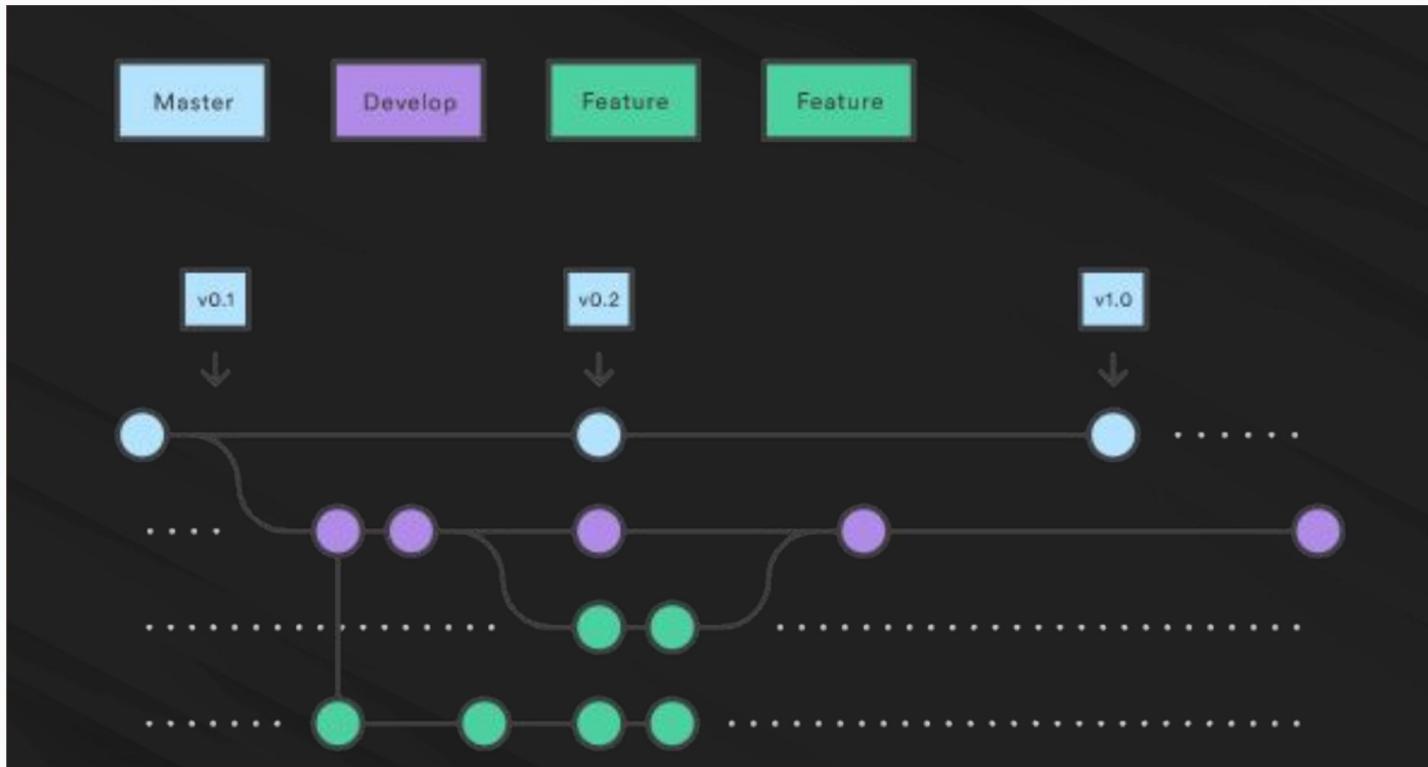
Основные проблемы

- Уязвимости в коде и зависимостях(CWE)
- Баги
- Мёртвый код
- Проблемы с шифрованием
- Неправильные конфигурации фреймворков
- Низкий уровень квалификации разработчиков
- Долгий процесс исправления уязвимостей в зависимостях для прода
- Повышенные привилегии

Решения

- Обучение разработчиков и Security Champions [основам безопасности от OWASP](#)
- Договоренности между разработчиками и InfoSec [Top 10 Secure Coding Practices](#)
- Митапы
- Использование Sec-инструментов в IDE
- Митигация через автотесты на этапах CI

Системы контроля версий (VCS)



Системы контроля версий (VCS)

Что мы хотим от VCS?

- сохранность данных;
- быстродействие;
- удобство использования;
- функциональность.

Однако в утилитах для управления версиями кода базово
отсутствуют механизмы
безопасности.

Какие проблемы могут быть в процессе централизованного хранения кода?

- Утечка исходных кодов/секретов приложений
- Внедрение программных закладок в код
- Проникновение в инфраструктуру
- Компрометация ПК разработчиков

Основные проблемы безопасности

- Хранение секретов в коде и в результате commit в VCS
- Проблемы аутентификации (Weak authentication, Authorization)
- Отсутствие ПРД (разграничения доступа, по дефолту)
- Проблемы конфигурации (публичная доступность, проблемы с паролями)
- Уязвимости в системах контроля версий

Решение проблем

- Secret Finding - Commit Hooks (main/master)
- Хранилище секретов и Environment переменные
- Подпись коммитов (PGP)
- Protected Branches, Code Review - снижение риска внедрения закладок, ошибок
- Внедрение требований ИБ/Промежуточный Compliance Check
- Threat Intelligence/Patch Management

Мониторинг и защита

Проблемы безопасности работающего приложения

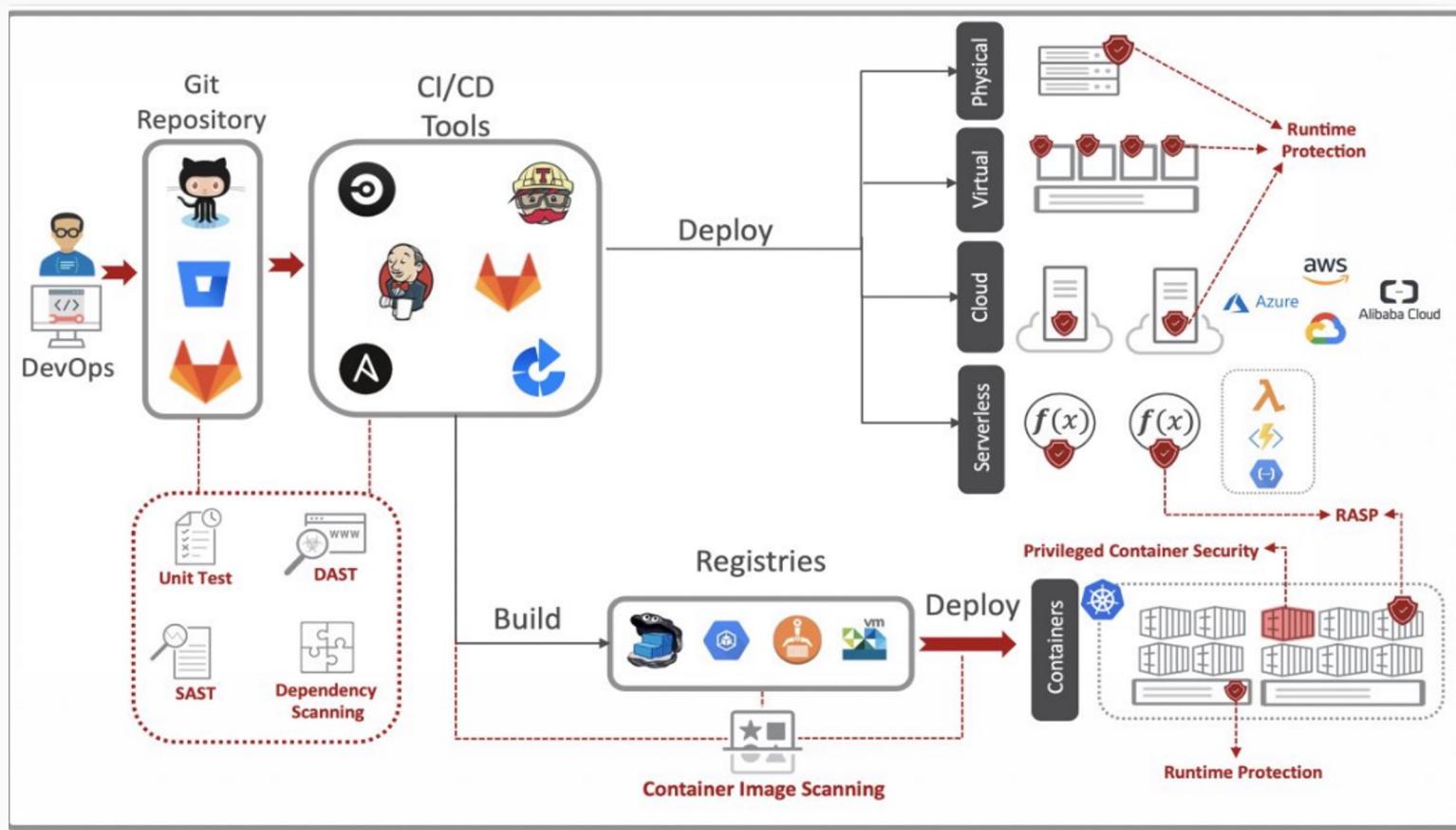
- Уязвимости которые были упущены или неизвестны в процессе CI/CD
- Zero-day уязвимости
- Миссконфиги



Защита приложения в RunTime

- WAF/IPS (Attack Prevention)
- Runtime Application Self-Protection (RASP)
- Log Management & SIEM (Monitoring)
- Incident Response (SOC)
- Vulnerability/Patch Management
- DDoS protection
- Bug Bounty - программы

DevSecOps, CI/CD



Инструменты

Threat modeling

- ThreatModeler
- OWASP Treat Dragon
- Microsoft Threat Modeling Tool



ThreatModeler
SECURITY STARTS HERE

Secret Finding

Поиск секретов (Secret Finding)

GitHound

Анализ:

- branches
- commits



GitGuardian [automated secrets detection]

Хранение секретов

- ASP.NET Core
- Hashicorp Vault
- Jenkins Credentials/Git Env Variables
- AWS Secrets Manager



Secret
Manager



AWS Secrets
Manager

```
node {  
    withCredentials( [usernamePassword( credentialsId: 'googlecompute',  
                                         usernameVariable: 'USERNAME',  
                                         passwordVariable: 'PASSWORD')]) {  
        // build project  
        sh 'mvn clean install'  
    }  
}
```

Container Security



NIST 800-190 - стандарт
описывающий
многоуровневую защиту
контейнеризации

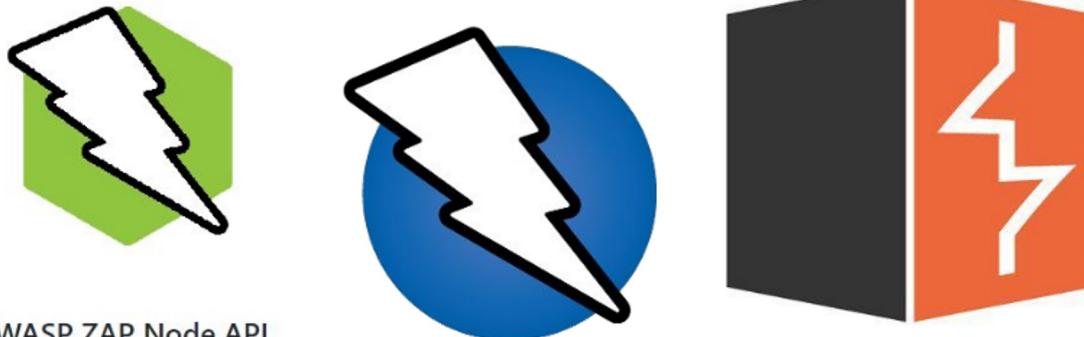


Open source SAST

- Node.js - **nodejsscan** 
 - Python -  **Bandit**
 - Ruby on Rails -
 - Java (Maven & Gradle) -
 - Scala (sbt) -
 - Go -  **GoSec**
 - PHP - [FloeDesignTechnologies/phpcs-security-audit](https://github.com/FloDesignTechnologies/phpcs-security-audit)
 - .NET - **Security Code Scan**
 - C/C++ - 
FlowFinder-Action
- 
- 

DAST

- Базируется на запросах к web-приложению/сервису
- Идентифицирует уязвимости OWASP TOP 10
- Используется для проверки и своевременного Virtual Patching
- Синергия с SAST - приоритизация устранения exploitable уязвимостей.



Фазинг

2. Фаззинг

- AFL (форк – AFL++)
- LibFuzzer (Honggfuzz)
- KLEE*

3. Санитайзеры (Sanitizers)

Fuzzing (Fuzz testing)

- AFL (fork AFL++)
- LibFuzzer (Honggfuzz)
- KLEE
- Sanitizers

Audit & Monitoring (SIEM)

- Хранение, обработка и представление логов
- Триггеры и корреляционный анализ
- Оповещения об инцидентах

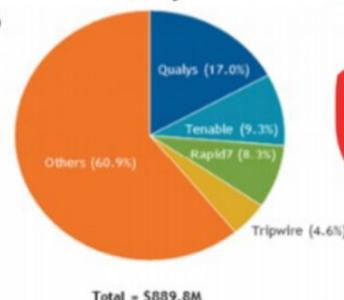
Alert & Monitoring (WAF)



Vulnerability Management

- Процесс и тип решений в виде платформы
- Позволяет объединить результаты найденных уязвимостей на различных этапах процесса (подобно GitLab Security & Compliance)

Worldwide Device
Vulnerability Assessment
Revenue Share by Vendor,
2016



Средства

		PERIODIC TABLE OF DEVOPS TOOLS (V3)																																			
1	Os	Gl GitLab		Os Open Source		Fr Free		Fm Freemium		Pd Paid		En Enterprise		Source Control Mgmt.		Deployment		Analytics		Monitoring		Containers		Release Orchestration		Security		Collaboration		2	En						
3	Fm	Gh GitHub		Dt Datical		Xlr Xebialabs XL Release		Aws AWS		Az Azure		Gc Google Cloud		Op OpenShift		Sp Splunk		Sg Sumo Logic		Dk Docker		Ur UrbanCode Release		Af Azure Functions		Ld Lambda		Ic IBM Cloud		Fd Fluentd							
11	Os	Sv Subversion		Db DEMaestro		Xld Xebialabs XL Deploy		Ud UrbanCode Deploy		Ku Kubernetes		Cc CCD Director		Pr Putora Release		Al Alibaba Cloud		Os OpenStack		Ps Prometheus		Cp AWS CodePipeline		Cy Cloud Foundry		It JTRS											
19	En	Cw ISWP		Dp Delphix		Jn Jenkins		Cs Codeship		Fn FitNesse		Ju JUnit		Ka Karma		Su SoapUI		Ch Chef		Tf Terraform		Oc Octopus Deploy		Go GoCD		Ms Mesos		Gke GKE		Om OpenMake							
37	Pd	At Artifactory		Rg Ridgeata		Ba Bamboo		Vs VSTS		Se Selenium		Jm JMeter		Ja Jasmine		Sl Sauce Labs		An Ansible		Ru Rudder		Cd AWS CodeDeploy		Ec ElectricCloud		Ra Rancher		Aks AKS		Rk Rkt							
55	Pd	Nx Nexus		Fw Flyway		Tr Travis CI		Tc TeamCity		Ga Gatling		Tn TestNG		Tt Tricentis Tosca		Pe Perfecto		Pu Puppet		Pa Packer		Rm Rancher		En En		67 Os		68 Pd		69 Os		70 Pd					
73	Fm	Bb BitBucket		Pf Perforce		Cr Circle CI		Cb AWS CodeBuild		Cu Cucumber		Mc Mocha		Lo Locust.io		Mf Micro Focus UFT		Sa Salt		Ce CFEngine		Eb ElasticBox		Ca CA Automic		De Docker Enterprise		Ae AWS ECS		Cf Codefresh		Hm Helm		Aw Apache OpenWhisk		90 Os	
91	En	Xli Xebialabs XL Impact		Ki Kibana		Nr New Relic		Dt Dynatrace		Dd Datadog		Ad AppDynamics		El ElasticSearch		Ni Nagios		Zb Zabbix		Zn Zenoss		101 En		102 En		Wp Signal Sciences WPP		103 En		104 Os		105 Os					
106	En	Sw ServiceNow		Jr Jira		Tl Trello		Sk Slack		St Stride		Cn CollabNet VersionOne		Ry Remedy		Ac Agile Central		Og OpsGenie		Pd Pagerduty		Sn Snort		Tw Tripwire		Ck CyberArk		Vc Veracode		Ff Fortify SCA		120 En					



Follow @xebialabs

Спасибо за внимание!