FORTIS

Безопасность приложений на скорости DevOps

Bright DAST - инструмент автоматизации тестирования защищенности приложений и API, для команд безопасности и разработки

Инструменты для выпуска безопасного продукта

- ❑ SAST
- ❑ DAST
- ❑ Pentest / Bug bounty
- ❑ Compliance

Проблемы legacy DAST решений

- ❑ Высокий false positive/negative rate
- ❑ Отсутствие поддержки не HTTP протоколов
- ❑ Интеграция DAST на поздних стадиях разработки
- ❑ Много времени на анализ и интерпретацию результатов
- ❑ Нет понимания бизнес логики

Почему компании продолжают выпускать уязвимый код?

86%

Компаний разработчиков продолжают регулярно выпускать уязвимый код

Соотношение разработчиков/специалистов ИБ во многих компаниях ~ 10/1 - 100/1

~25%

Разработчиков отвечают за безопасность в процессе

~ 25%

Уязвимостей находятся разработчиками

< 19%

Компаний имеют DAST с интеграцией в CI/CD и < 10% делают его инструментом разработчиков

69% уязвимостей обнаруживаются в собственном коде компаний (только 31% в стороннем)

**Согласно [Cost of a Data Breach Report 2021 | IBM]*

Требования к современному DAST

- ❑ Широкие возможности управления и интеграции
- ❑ Минимизирование нагрузки на инфраструктуру
- ❑ Легко масштабируемое
- ❑ Простота использования

Особенности Bright DAST

- ❑ No false positive/No false negative
- ❑ Crawler собственной разработки
- ❑ Поддержка различных Web протоколов
- ❑ Поддержка систем проектного управления
- ❑ Возможность работы с системой всех участников цикла разработки
- ❑ Агностичная архитектура
- ❑ Проверки бизнес-логики

NexDAST - подтверждение уязвимостей

- ❑ Валидация уязвимости виде выполненной команды и ответа от сервера и скриншота
- ❑ Полная информация об уязвимости и рекомендации по исправлению

Severity	Discovered	Status
High	15/9/2021, 17:26	Unresolved

CWE ID	CVSS
CWE-79	6.5 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N)

Details

This vulnerability was found in the Query by changing the value of the parameter "".
The value '<script>setTimeout(function(){alert(993113)},1000);</script>' was injected, which caused the target to execute an alert function with the value 993113, which verified that the JavaScript injection was successful.

Remediation suggestions

To remedy this vulnerability, a proper input validation and sanitization should be applied to the "" parameter.
The best course of action is not to blacklist or disallow specific characters, but instead to verify that the input has the relevant types and structure.
For example, the value of the above parameter is detected as type: MultiParse::DataType::String. A good start is to verify that it is not a different type of data structure.
Another good approach is to validate that integer fields accept only integers, text fields only accept alphabetical characters (if possible) and types (such as JSON or other formatted objects) are verified and parsed before being digested and reflected back to the user.

Request

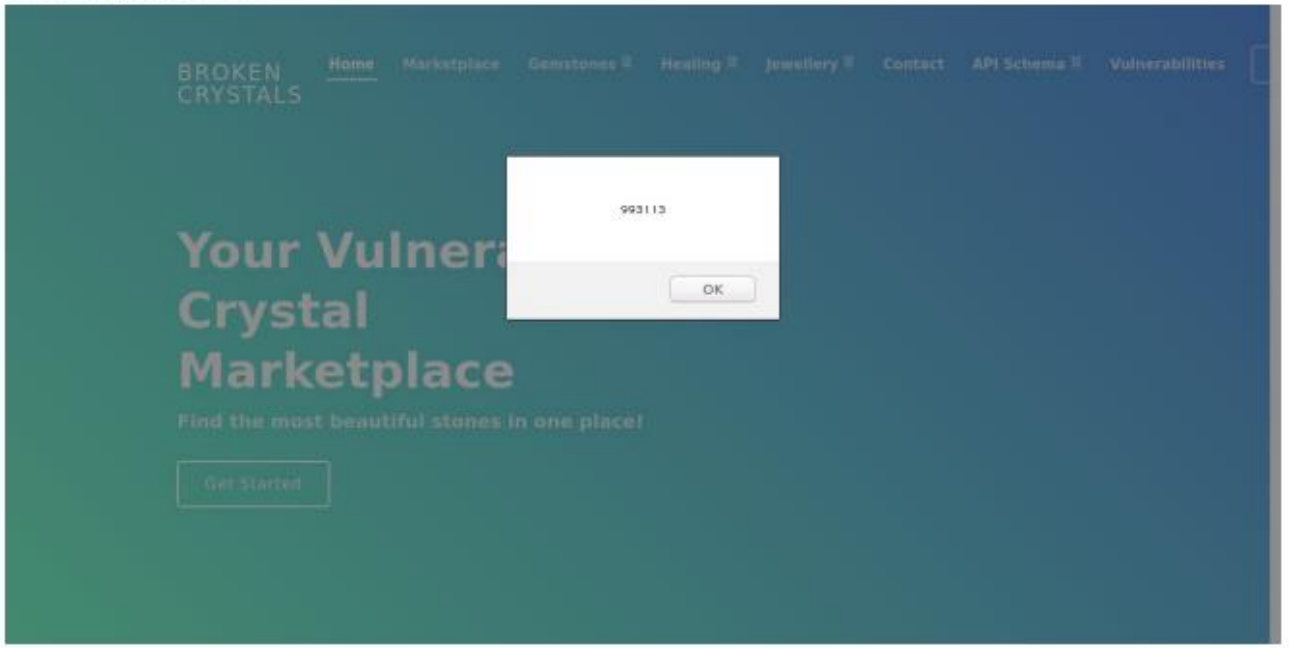
Url

```
https://brokencrystals.com/?name=%3Cscript%3EsetTimeout%28function%28%29%7Balert%28993113%29%7D%2C1000%29%3B%3C%2Fscript%3E&email=Your+Email&subject=Subject&message=Message
```

NeuraLegion

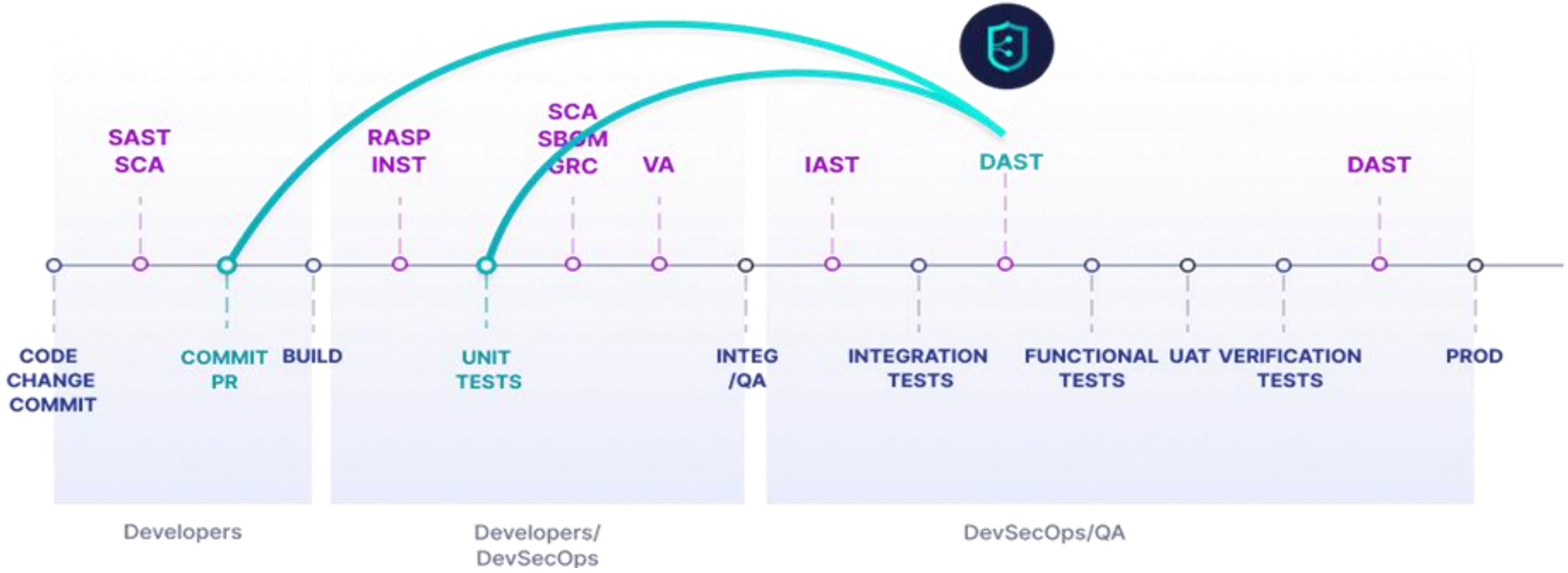
Screenshots

1. rXSS Execution POC

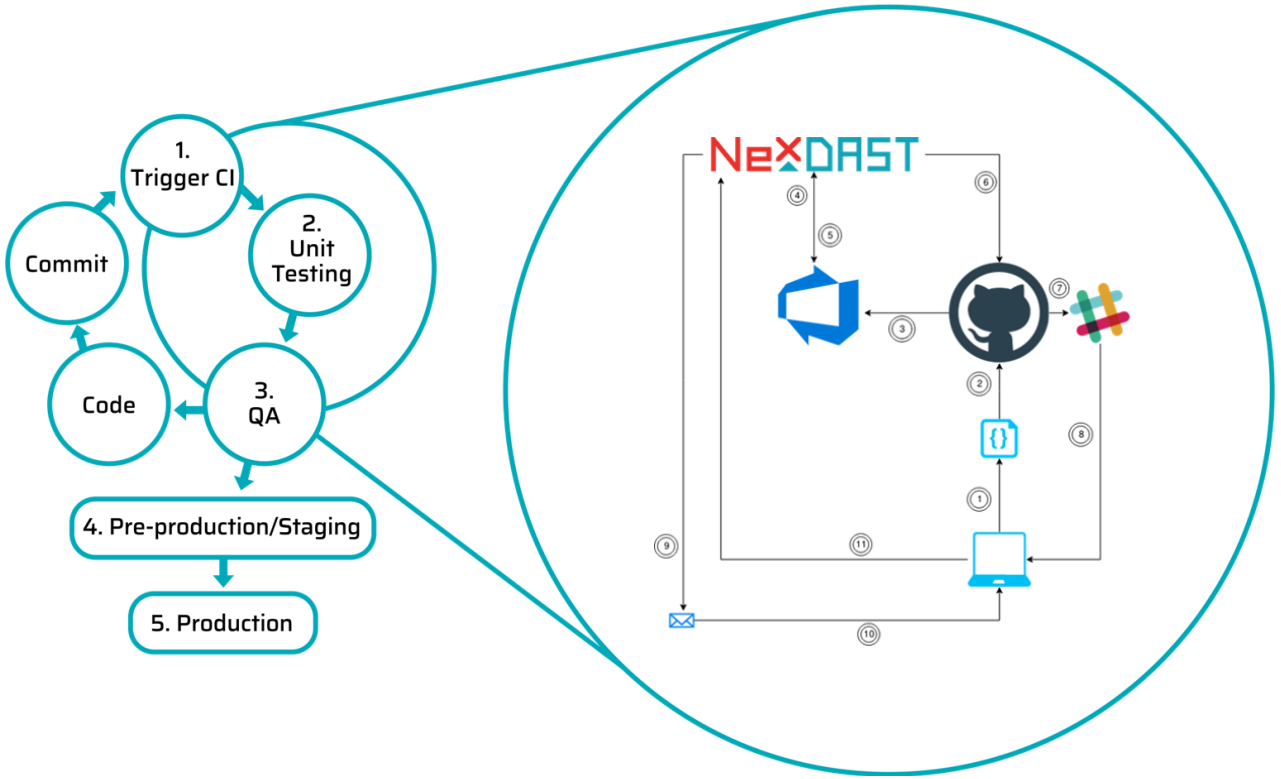


NexDAST – shift-left Security

- Смещает безопасность левее, чтобы встроиться в модульные и интеграционные тесты (автоматизация создания модульного теста безопасности)



NexDAST – автоматизация проверок в SDLC



1. Code
2. Commit
3. Push commits to GitHub, GitLab
4. Trigger CD
5. Initiate NexDAST scan
6. Build pass/fail
7. Open tickets in Ticketing system
8. GitHub > Slack integration

9. Results shared with security team

Azure DevOps · Jenkins · circleci · JFrog · GitHub · Jira · slack · OPEN API

NexDAST – запуск проверок из среды разработки

Quick Start

Starting with NexPloit-CLI is easy.

First, install NexPloit-CLI globally:

```
$ npm install @neuralegion/nexploit-cli -g
```

Next, go to the directory of your project and run the command:

```
$ nexploit-cli -h
```

If you already have a prepared mock file, you can generate a HAR file with the following command:

```
$ nexploit-cli archive:generate \
    --mockfile .nexmock \
    --archive archive.har \
    --target https://nexploit.app/ \
    --header "authorization: Bearer eyJhbGciOiJIUzUxM"
```

Where `mockfile` is the path to your mockfile and `archive` is the path to save the HAR file to.

- Интеграция на этапе модульного тестирования, для запуска сканирования с помощью нашего инструмента с открытым исходным кодом [nexploit-cli]

```
fetchMock.mock('http://example.com', 200);
const res = await fetch('http://example.com');
assert(res.ok);
fetchMock.restore();
```

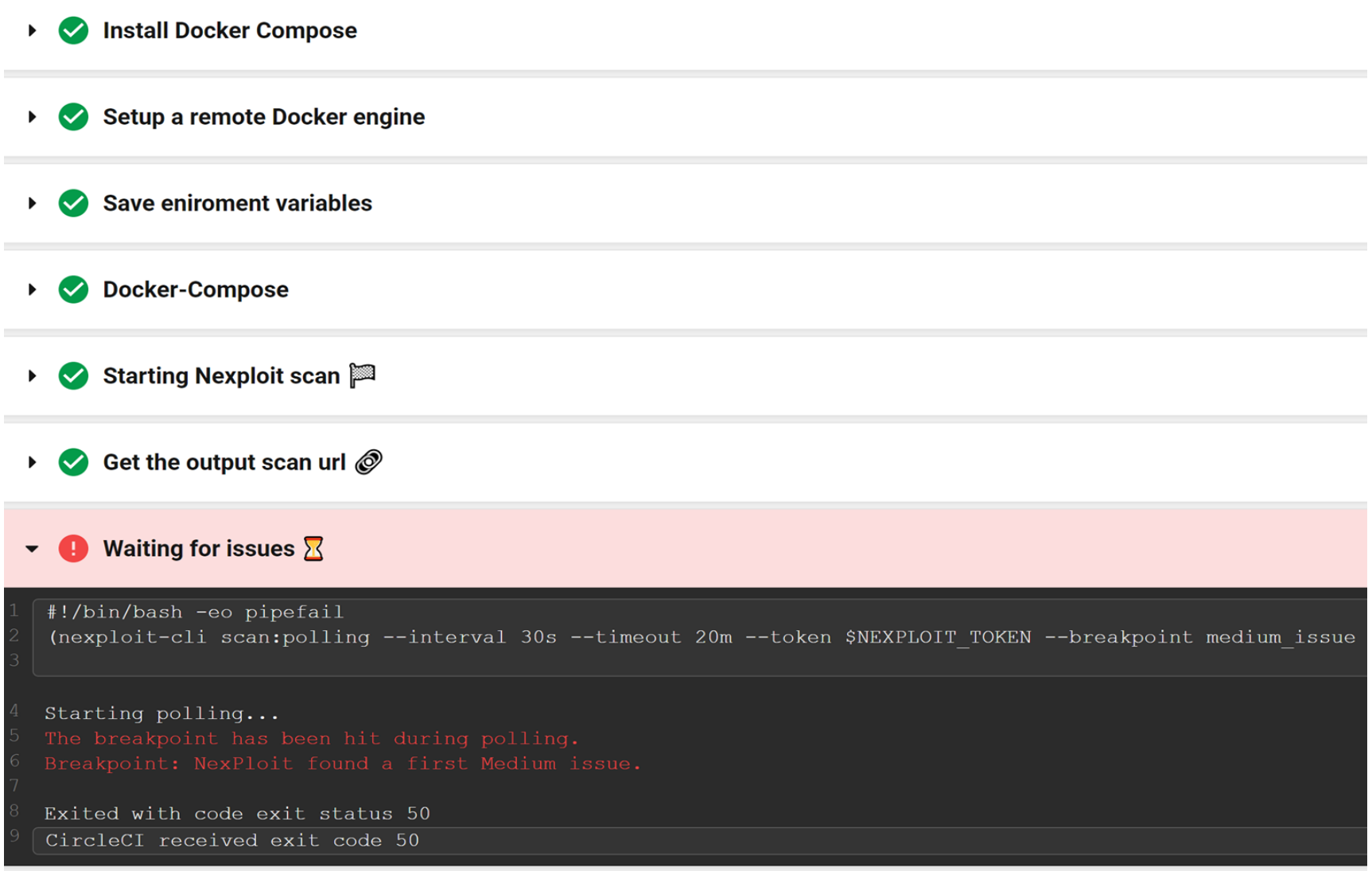
NexDAST – настройка проверок в pipeline

- ❑ Используйте github actions YAML файлы для контроля настроек сканирования
- ❑ Используйте CircleCI YAML файлы для настройки и контроля потока сканирования

```yaml
version: 2.1
orbs:
  nexploit: neuralegion/nexploit@2.0
jobs:
  build:
    machine: true
    steps:
      - nexploit/scan:
          scan_name: CircleCI Scan
          api_key: NEXPLOIT_API_KEY
          discovery_types: crawler
          crawlers: 'https://www.random1.org/,https://www.random2.org/'
          host_filters: 'random1.org,random2.org'
          headers: 'Content-Type: application/json;Keep-Alive: timeout=5, max=1000'
          type: appscan
          protocol: http
```

```yaml
start_and_wait_scan:
  runs-on: ubuntu-latest
  name: A job to run a scan
  steps:
  - name: Start NexDAST Scan 🏁
    id: start
    uses: NeuraLegion/run-scan@master
    with:
      api_token: ${{ secrets.NEXPLOIT_TOKEN }}
      name: GitHub scan ${{ github.sha }}
      discovery_types: |
        [ "crawler"]
      crawler_urls: |
        [ "https://juice-shop.herokuapp.com" ]
      hosts_filter: |
        [ "juice-shop.herokuapp.com" ]
      wait_for: on_high
  - name: Get the output scan url
    run: echo "The scan was started on ${{ steps.start.outputs.url }}"
  - name: Wait for any issues ⌛
    id: wait
    uses: NeuraLegion/wait-for@master
    with:
      api_token: ${{ secrets.NEXPLOIT_TOKEN }}
      scan: ${{ steps.start.outputs.id }}
      wait_for: any
      timeout: 55
```

NexDAST - взгляд со стороны разработчика



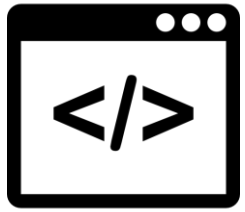
NexDAST – для команд безопасности и разработки

Bright

AppSec Flow

- ❑ Настройка сканов, АО, pipeline
- ❑ Анализ результатов на качество и покрытие
- ❑ Рекомендации по исправлению
- ❑ Итерации с разработчиками для достижения бизнес-целей

Developer Flow



- ❑Полная автоматизация
- ❑ Гибкое внесение изменений
- ❑ Информирование как только обнаружена уязвимость
- ❑ Исправление уязвимостей основных на доказательствах

FORTIS

Bright

Вопросы? neuralegion@fortis.ru

Всеволод Смирнов
v.Smirnov@fortis.ru
+79133986706

Марат Юнусов
+79168740004