Greenbone

Test Automation & QA Strategy

for Short Iterations

Tim Steffens

Software / QA Engineer

ABOUT ME

- timsteffens.de

- Father & husband
- ~20 years Software Engineering
- Quest
  - ✓ how to build software that "works well"?
- Fan of
  - ✓ clean code, TDD & functional programming
  - ✓ music (Metal)
  - ✓ outdoors & hiking

ABOUT GREENBONE

What sets us apart

Innovation: Leading in cybersecurity with scalable, cutting-edge solutions that identify and mitigate vulnerabilities.

Value Driven Culture: Work environment based on sincere appreciation, honoring individuality

Employee Empowerment: Recognizing our employees as the primary drivers of our success and committing to nurture growth through continuous learning and comprehensive benefits

ABOUT YOU!

- Software Engineer?
- Testing Engineer?
- Architect?
- Project Lead?
- None of those?

- Worked with test automation?

PLEASE INTERRUPT!

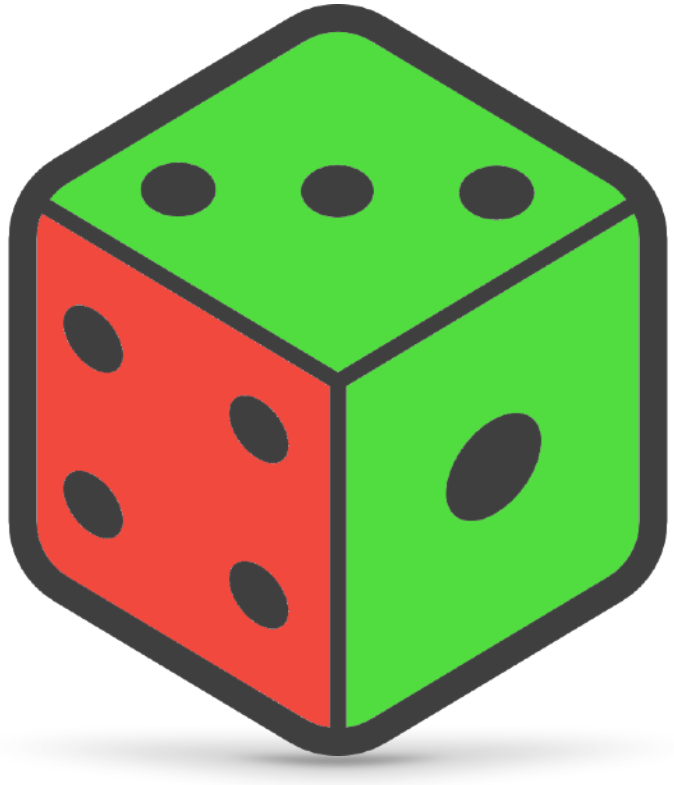
AGENDA

- Setting the Scene
- QA Strategy
- Implementation, Challenges & Learnings
- Outlook

DICE

- Recurring theme in Test Automation: **"the dice"**
  - ✓ AKA **"flaky tests"**
  - ✓ Two runs green, next red - despite no code change ...

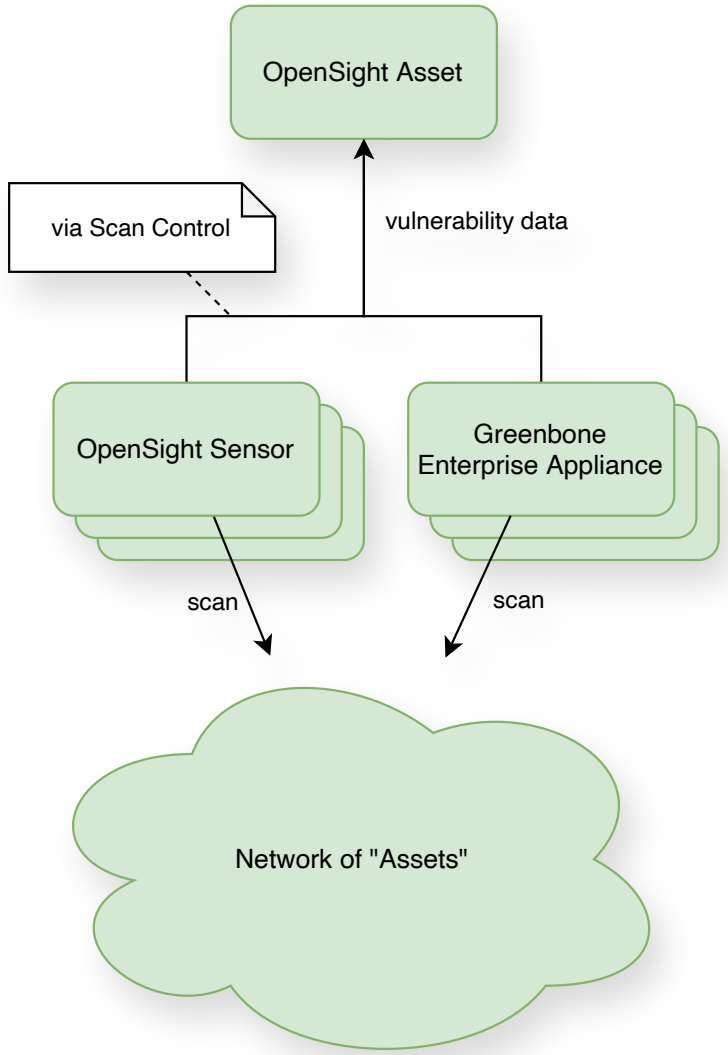
- **Who** has already experienced **flaky tests**?

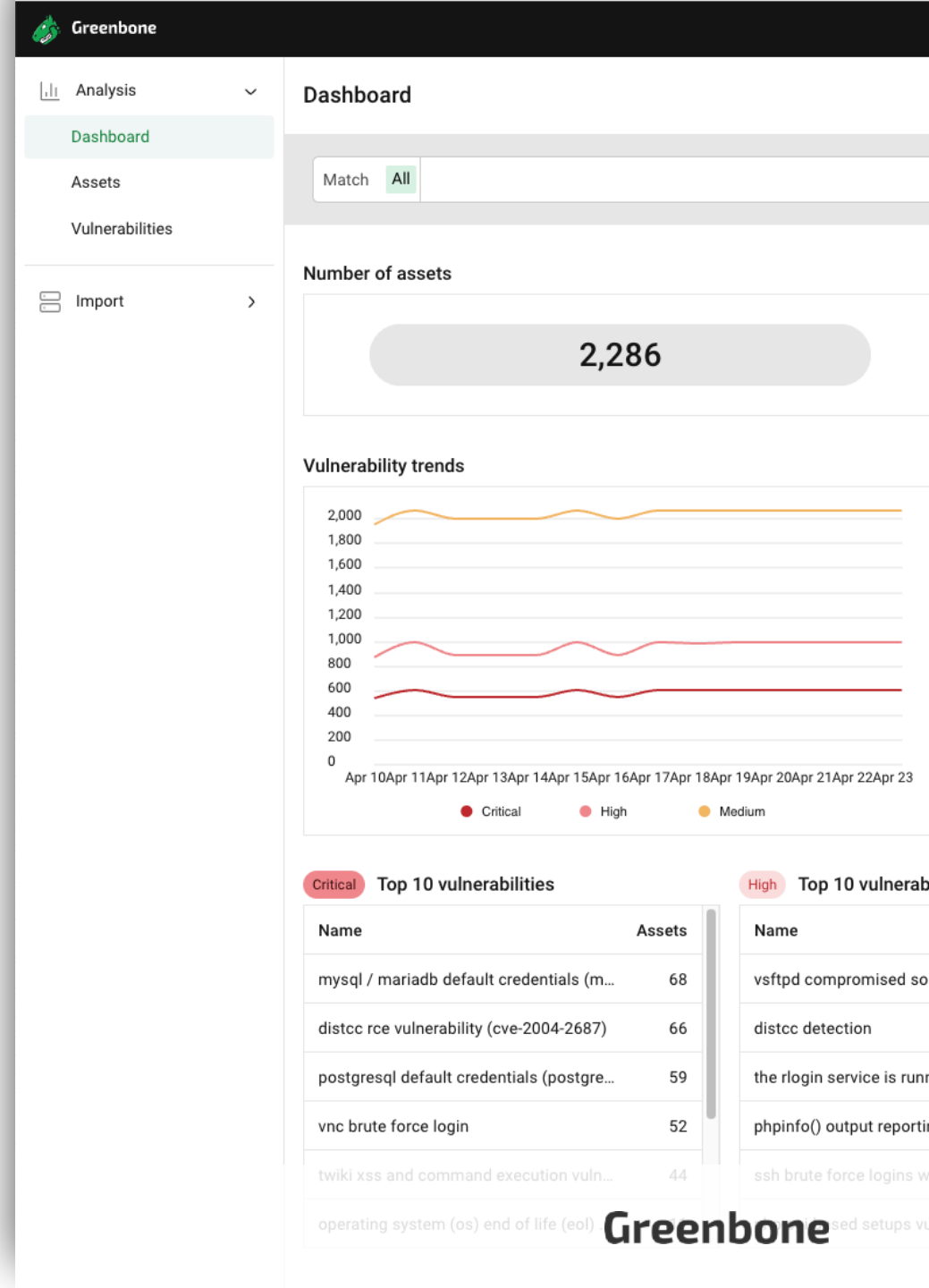


DISCLAIMER

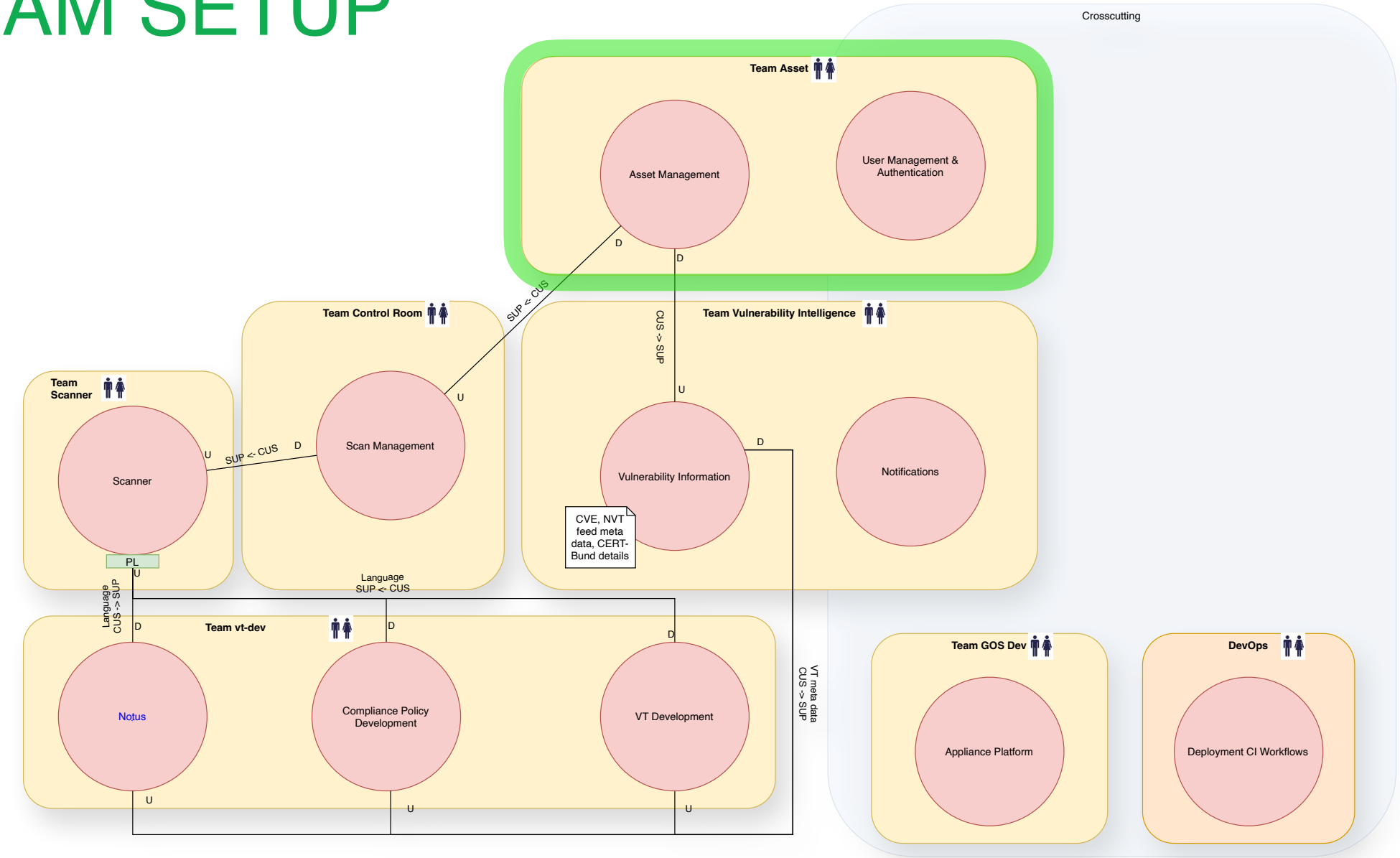
- Out of scope:
  - NFA: Security Testing, Load Testing
  - Code Coverage
  - QA development practices (e.g. Pair Programming, Code Reviews)
  - Frontend Unit Tests

SETTING: PRODUCT





TEAM SETUP



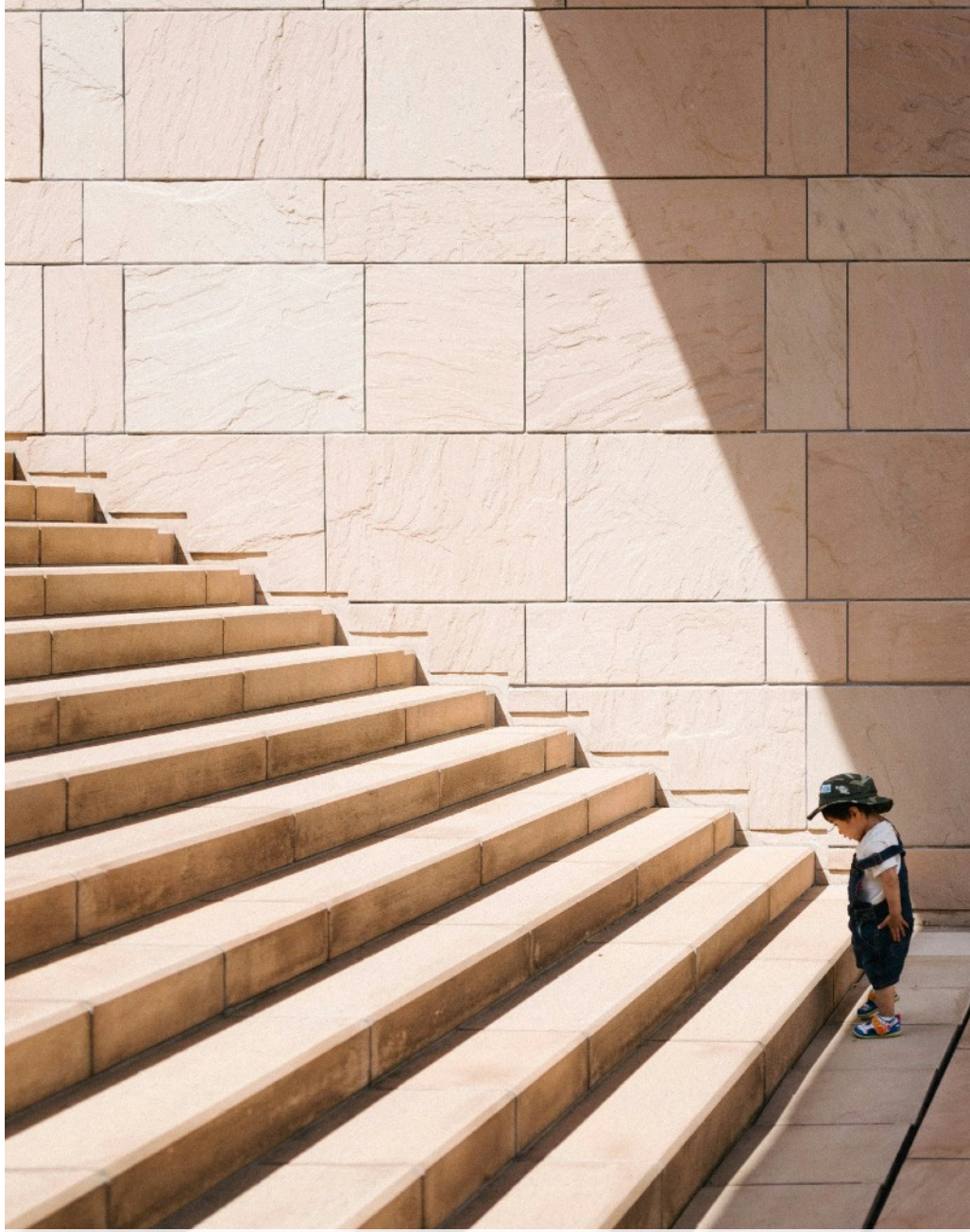
Greenbone

SETTING: AGILE

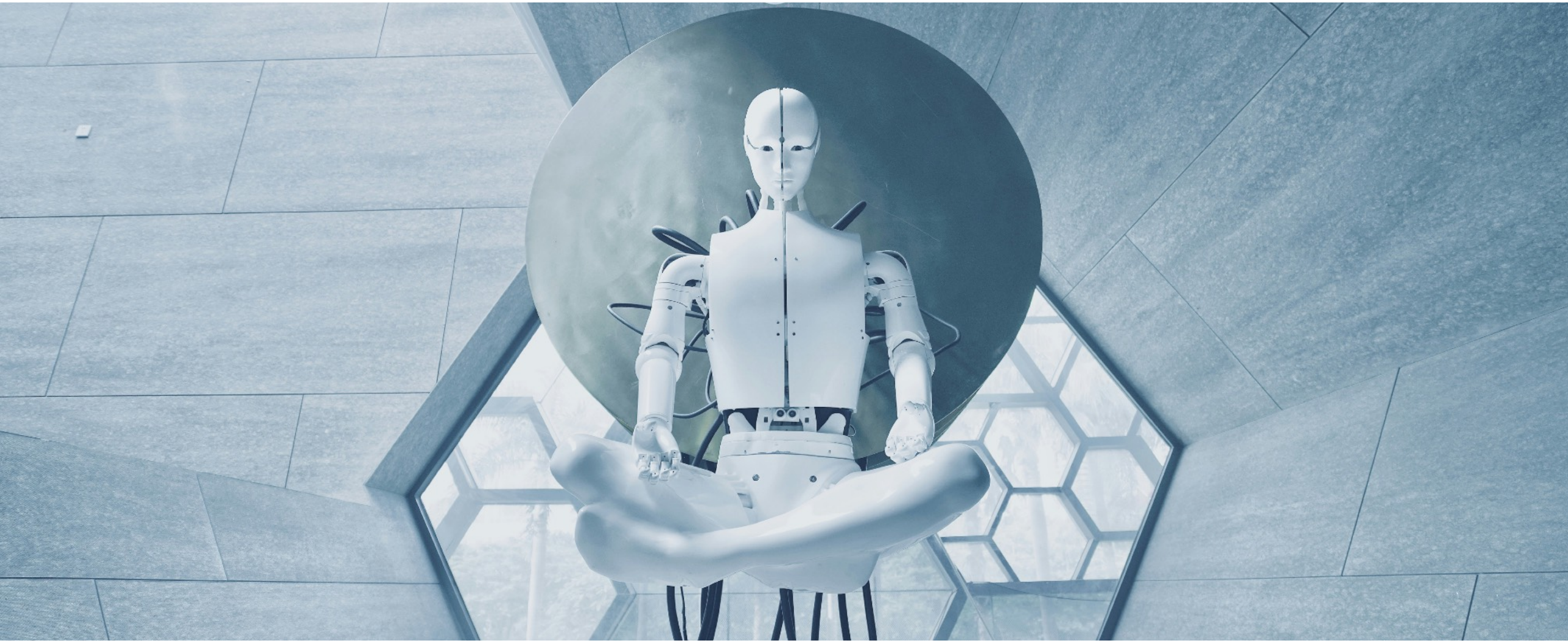
- Agile Approach
  - ✓ **Scrum**
  - ✓ short iterations: **2 week sprints**
  - ✓ **multiple increments** released per sprint
  - ✓ optimized for **fast feedback**

CHALLENGES

- Too few resources to **test** everything **manually**
- Especially **regression**
  - ✓ needs to be repeated **several times a week**
  - ✓ should as **reliable** as possible
  - ✓ reasonable **effort**



THE ANSWER: TEST AUTOMATION

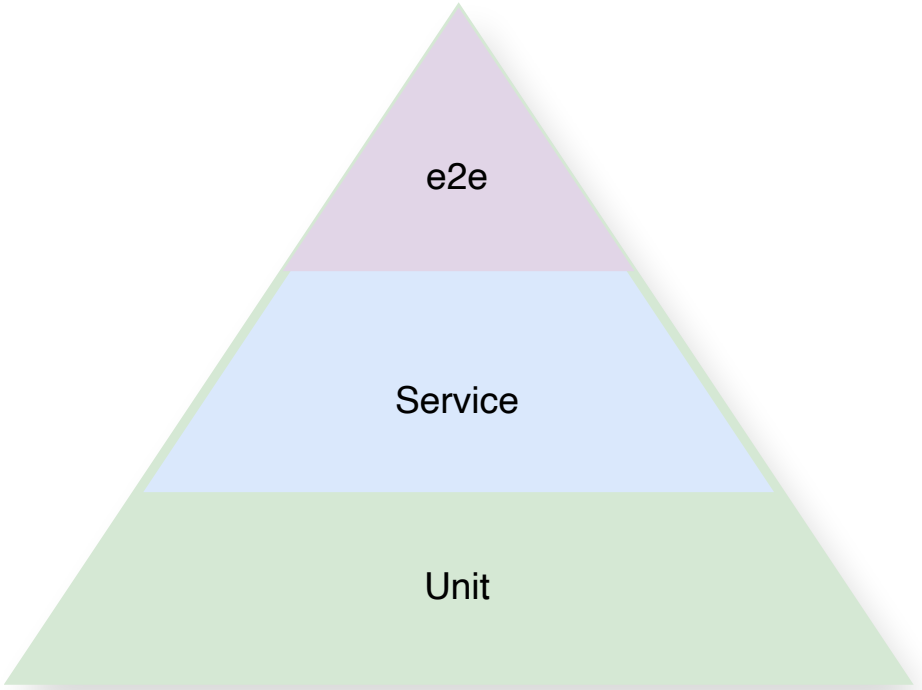


QA STRATEGY

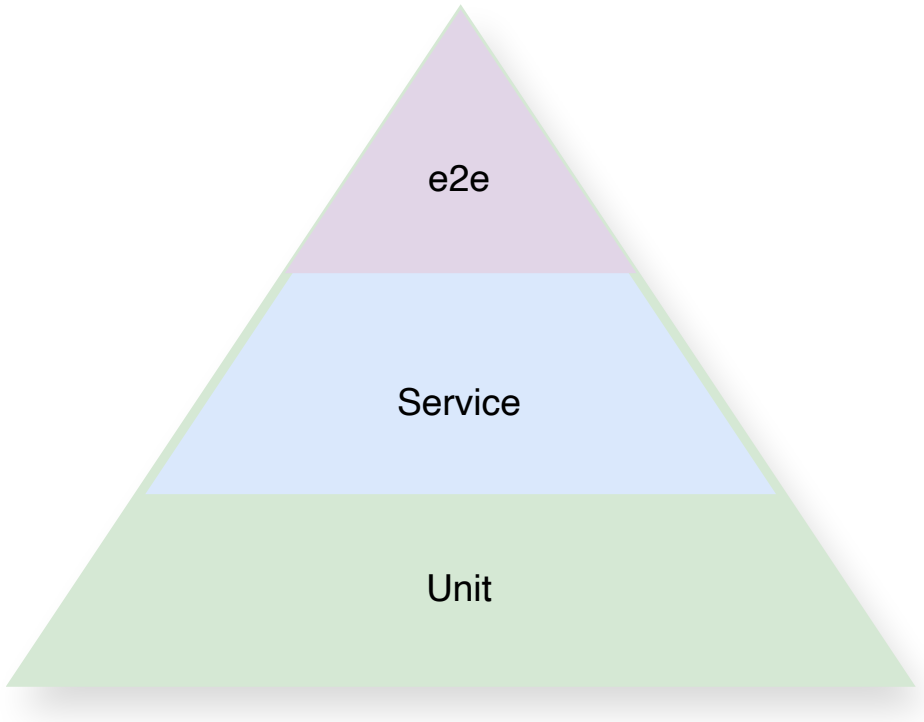
- **automate as much as possible**
- test things on the **right level**

- **code reviews** are mandatory
- in case **manual tests** are needed for regression they must be **well-documented** and easily **reproducible**
- for each **bug** that is found a **test** must be created
- (...)

TEST LEVELS

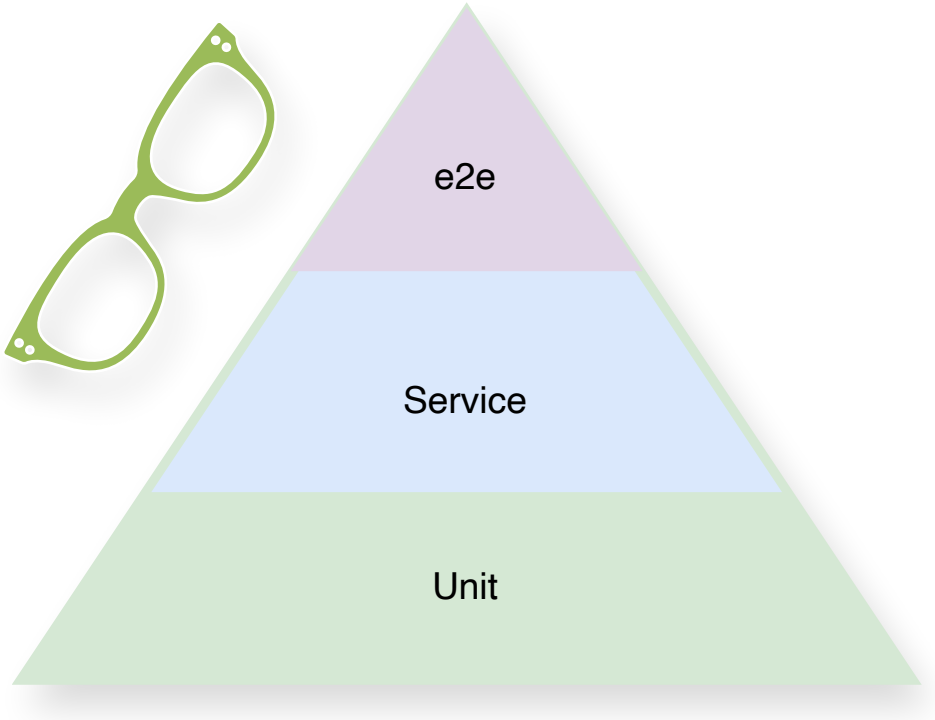


TEST LEVELS



	Automated	Manual
e2e	✓	✓
Service	✓	
Unit	✓	

TEST LEVELS



TEST MATRIX

Use Case	Test Case	Unit Test	BE Service Test	FE Service Test	Full Service Test	e2e Test	manual test
Edit Appliance	edit appliance with missing fingerprint	+ for appliance controller + for appliance service	+ add test case (example test case for failure)	+ add test case (test case stops after receiving the fingerprint)	-	-	-
	edit appliance with correct fingerprint	+ for appliance controller + for appliance service	+ add test case (example test case for success)	+ add test case	-	-	-
	edit appliance with incorrect fingerprint	+ for appliance controller + for appliance service	-	+ add test case (test case stops after receiving the error)	-	-	-
	get appliance where the host key has changed and the fingerprint is no longer valid	+ for appliance controller + for appliance service + for import service(s)	(too difficult to simulate)	+ add test case	-	-	test manually

...

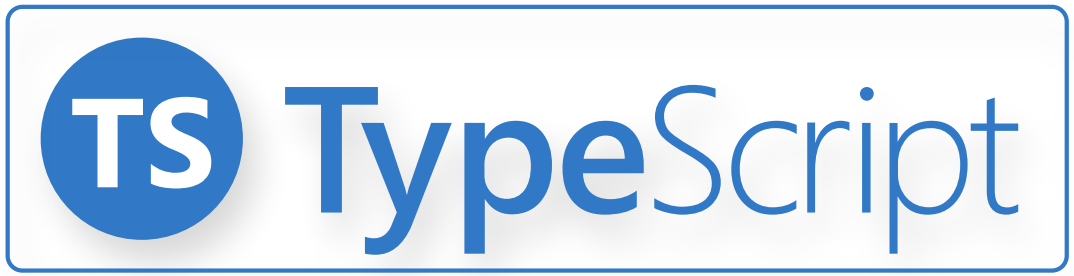
QA STRATEGY

Implementation

- **QA Strategy** workshop per team
- Dedicated **QA Engineer** per team
- **QA CoP**

E2E TESTS

E2E TESTS



E2E TESTS


```
import "cypress-wait-until"

describe('create a user and log in', () : void  => {
    after(() : void  => {
        cy.programmaticLogout()
        cy.deleteUser(Cypress.env("new_user"))
    })


    it('Creating a user on operating system level should work', () : void  => {
        cy.createUser(Cypress.env("new_user"), Cypress.env("temp_password"))
    })


    it('Login via UI should work and lead to the landing page of OpenSight Asset', () : void  => {
        cy.visit(Cypress.env('redirect_uri'))
        cy.waitForStableDOM()

        cy.loginViaUI(Cypress.env("new_user"), Cypress.env("temp_password"))
        cy.updatePassword(Cypress.env("password"))
            .then(() : void  => {
                // check landing page for validation
                cy.contains("h2", "Dashboard")
            })
    })

```

E2E TESTS

GitHub

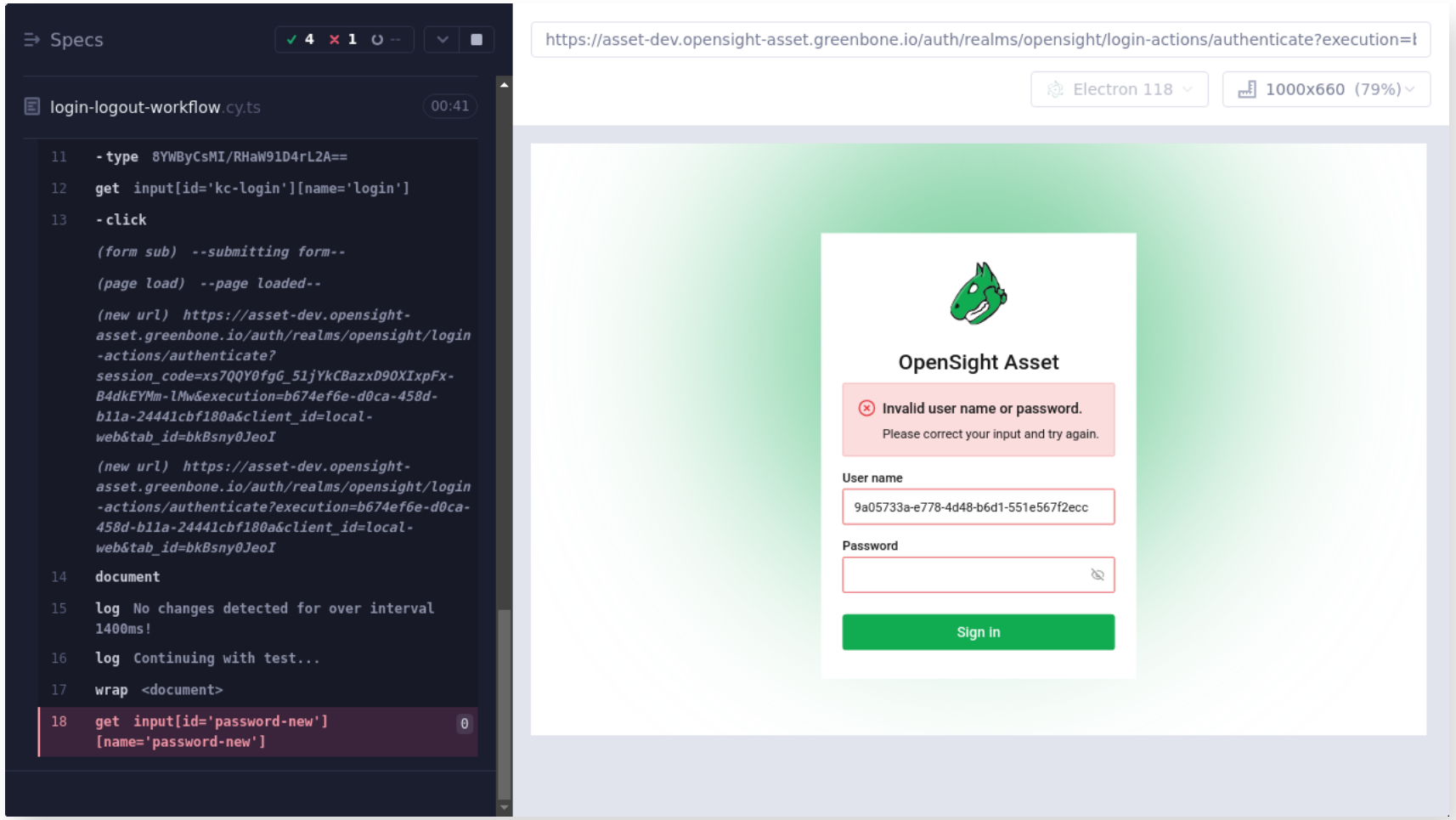
```yaml
name: e2e tests dev environment

on:
  schedule:
    - cron: '30 00 * * 1-5' # Mon-Fri at 01:30 UTC
  workflow_dispatch:
  pull_request:
    branches: [ main ]

jobs:
  e2e-tests:
    name: Run e2e tests against the dev system
    runs-on:
      - self-hosted-generic
      - self-hosted
    container:
      image: ghcr.io/greenbone/opensight-e2e:latest
      options: --init
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Install dependencies
        run: chown -R root ./ && npm ci
      - name: Run tests
        env:
          CYPRESS_ssh_pass: ${{ secrets.SSH_PASSWORD }}
          CYPRESS_password: ${{ secrets.PASSWORD }}
          CYPRESS_appliance_password: ${{ secrets.APPLIANCE_PASSWORD }}
        run: make start-e2e-tests ENV=dev
      - name: Upload report folder
        if: always()
        uses: actions/upload-artifact@v4
```

E2E TESTS

GitHub

```yaml
    runs-on:
      - self-hosted-generic
      - self-hosted
    container:
      image: ghcr.io/greenbone/opensight-e2e:latest
      options: --init
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Install dependencies
        run: chown -R root ./ && npm ci
      - name: Run tests
        env:
          CYPRESS_ssh_pass: ${{ secrets.SSH_PASSWORD }}
          CYPRESS_password: ${{ secrets.PASSWORD }}
          CYPRESS_appliance_password: ${{ secrets.APPLIANCE_PASSWORD }}
        run: make start-e2e-tests ENV=dev
      - name: Upload report folder
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: e2e-test-dev-reports
          path: cypress/reports/
          retention-days: 4
      - name: Upload screenshot folder
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: e2e-test-dev-screenshots
          path: cypress/screenshots/
          retention-days: 4
```

E2E TESTS



Greenbone

E2E TEST CHALLENGES

↯ Asynchronism ↯ 

- certain computations are done **asynchronously**
  - ➡ e.g. ui rendering, asynchronous messages
  - ➡ **time interval** before results are available **may vary** strongly depending on system and load

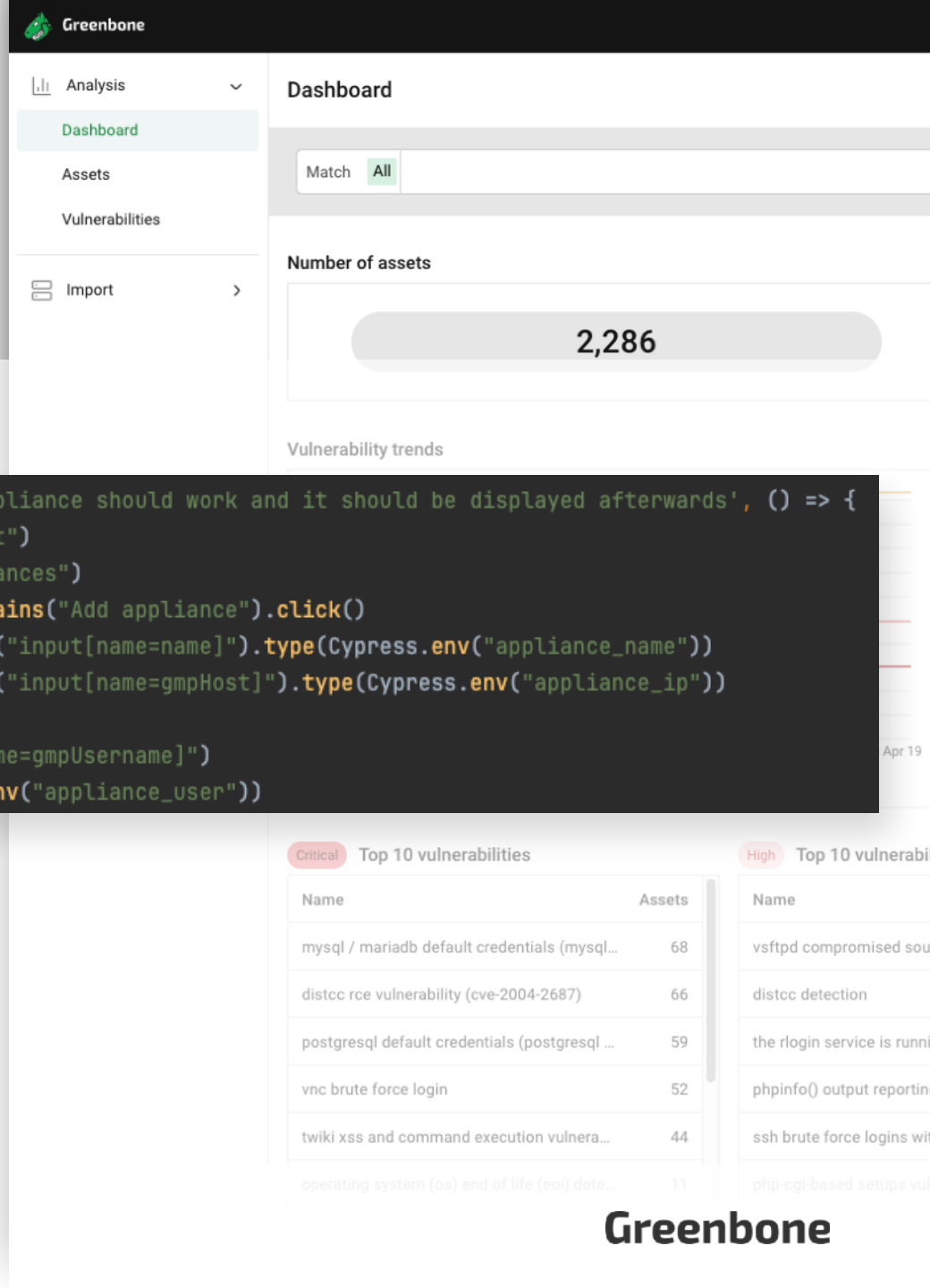
E2E TEST CHALLENGES

↯ Asynchronism ↯

- certain **computations** are done
  - ➡ e.g. **ui rendering**, asynchronous
  - ➡ **time interval** before **results** are
    strongly depending on system an

```
18  it('Creating a valid appliance should work and it should be displayed afterwards', () => {
19      cy.enterPage("Import")
20      cy.enterPage("Appliances")
21      cy.get("span").contains("Add appliance").click()
22      cy.get("form").find("input[name=name]").type(Cypress.env("appliance_name"))
23      cy.get("form").find("input[name=gmpHost]").type(Cypress.env("appliance_ip"))
24      cy.get("form")
25          .find("input[name=gmpUsername]")
26          .type(Cypress.env("appliance_user"))
```

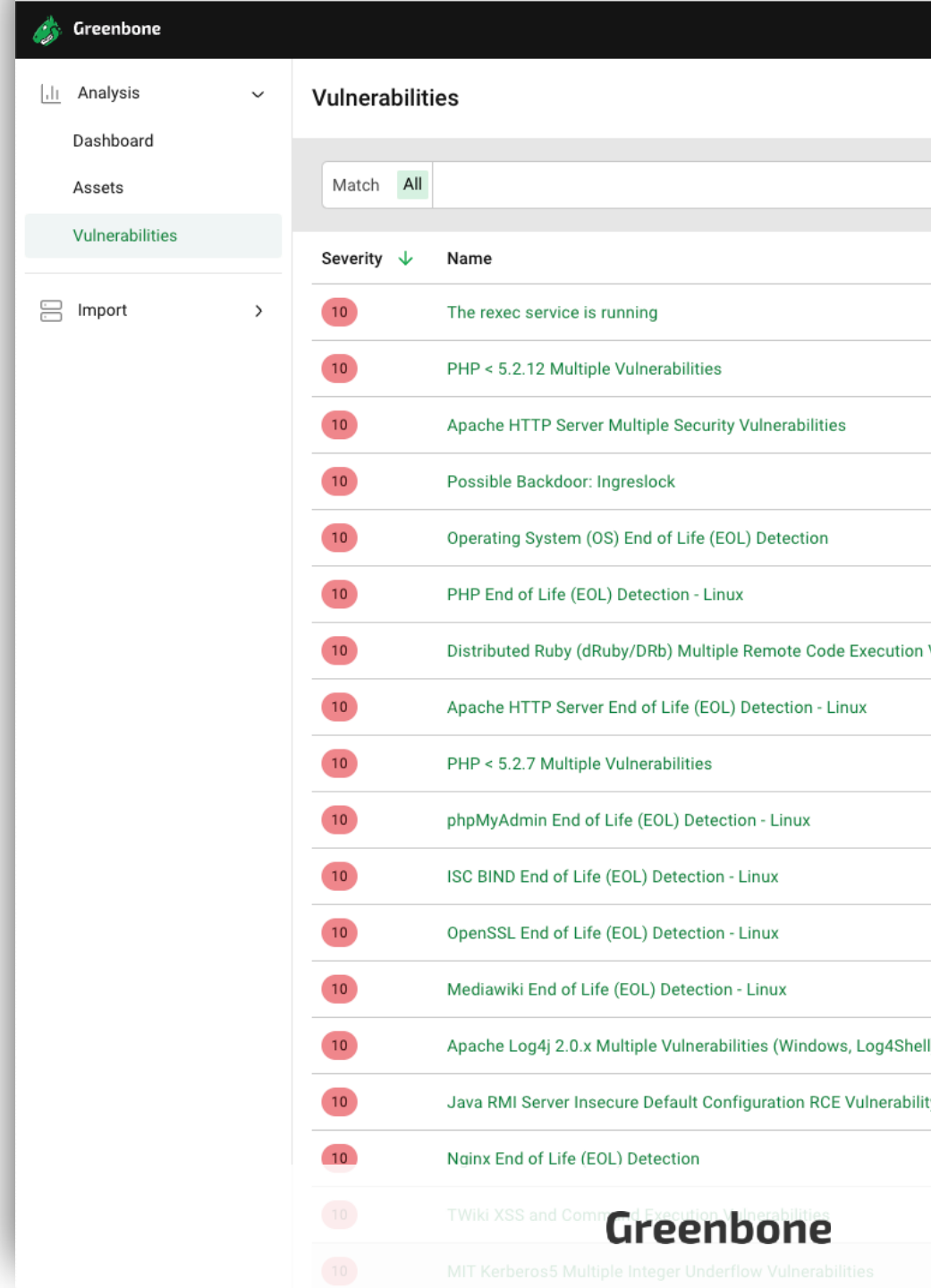
- our approach @ 
  - ➡ **wait for** conditions with **timeout** before checking assertions
  - ➡ **cypress** has **built-in waits** with default timeouts!



E2E TEST CHALLENGES

↯ Changing Data ↯ 

- **data** imported from **real systems changes** throughout time
  - ➡ **how many** and **which** entries will be there?



E2E TEST CHALLENGES

↯ Changing Data ↯

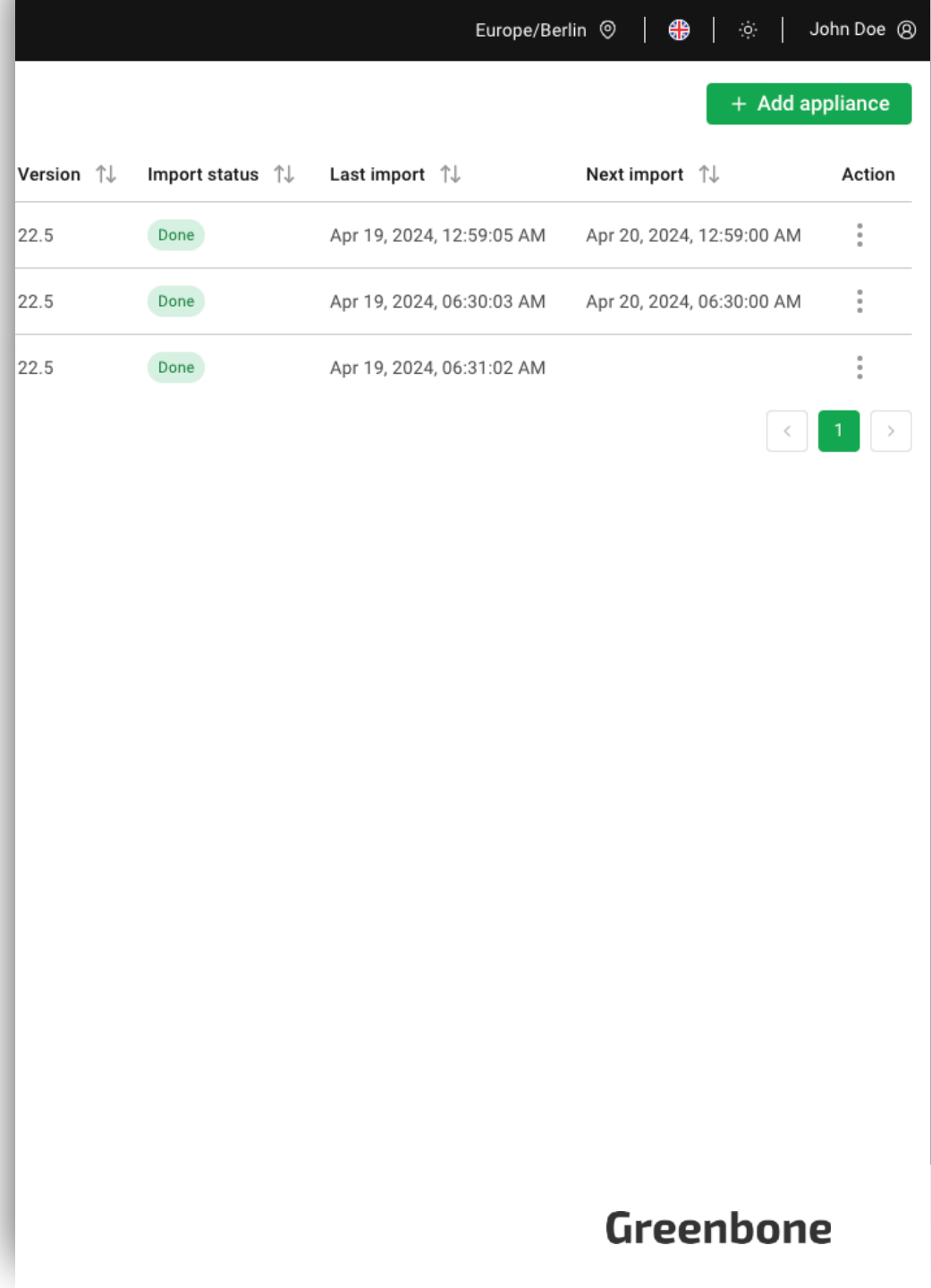
- **data** imported from **real systems** throughout time
  - ➡ **how many** and **which** entries will be
- our approach @
  - ➡ **assert** only what **you can know**
  - ➡ validate **structure**

```
35  Cypress.Commands.add("checkNumberOfImportedItems", (page: string) => {
36      cy.enterPage(page)
37      cy.deleteFilterTerms()
38      cy.enterFilterTerm(["Appliance ", "contains", Cypress.env("appliance_name")])
39      cy.waitForStableDOM()
40      cy.get("div[data-testid='pagination-container']", {timeout: 5000})
41          .contains("span", "items")
42          .invoke('text')
43          .then((text) => {
44              const regex = /(\d+) items/;
45              const matches = text.match(regex);
46              if (matches && matches.length === 2) {
47                  cy.expect(parseInt(matches[1])).to.be.greaterThan(0)
48              }
49          })
50
51      cy.deleteFilterTerms()
52  })
```

E2E TEST CHALLENGES

↯ Interference in Parallel Runs ↯

- **same resources** / entities / identifiers used for **each run**
  - ➡ e.g. same user
  - ➡ **test data** can be **influenced by another test run**

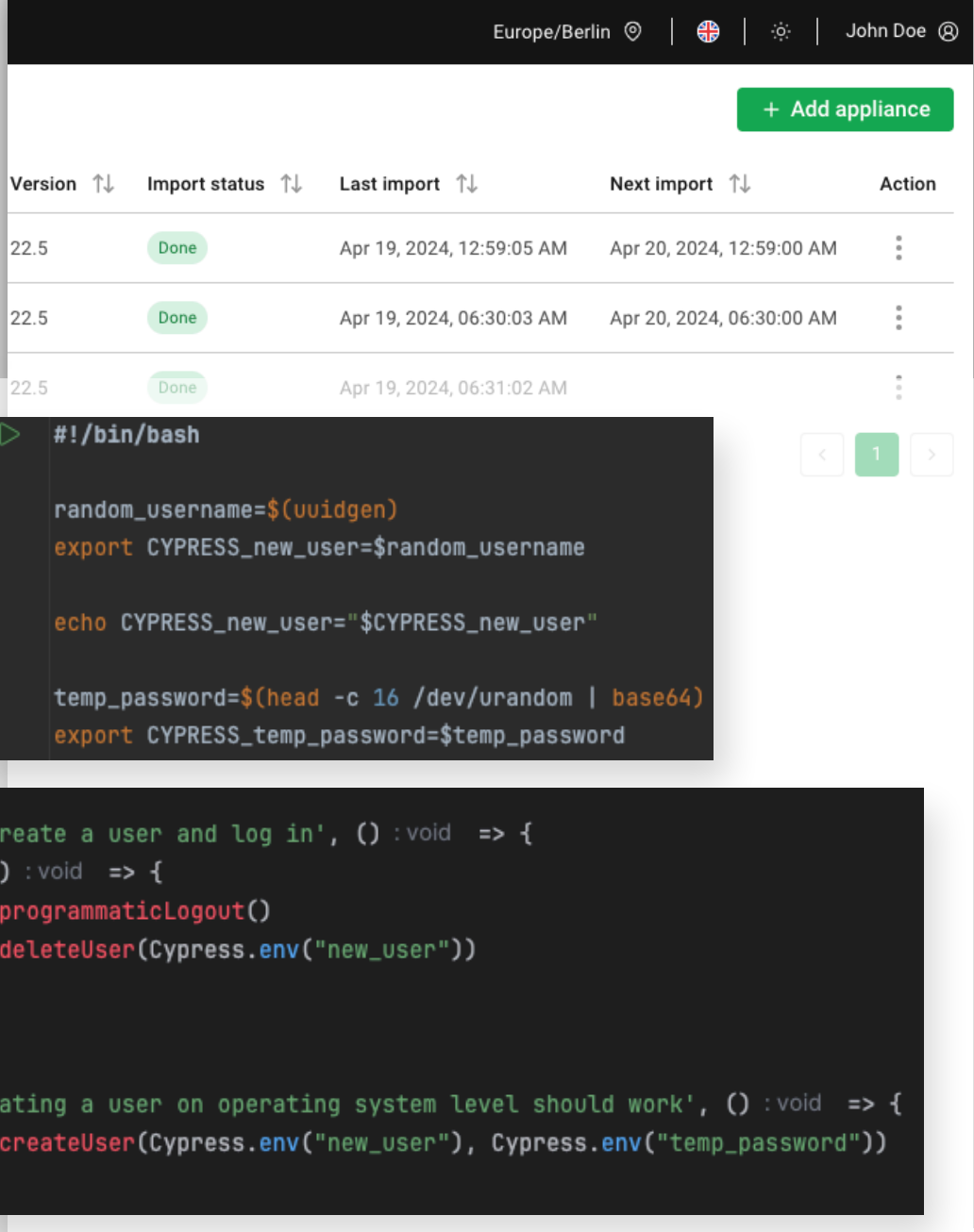


E2E TEST CHALLENGES

↯ Interference in Parallel Runs ↯

- **same resources** / entities / identifiers used for **each run**
  - ➡ e.g. same user
  - ➡ **test data** can be **influenced by another test**

- our approach @
  - ➡ **generate** entities with random identifiers
  - ➡ and **delete** them after **each run**

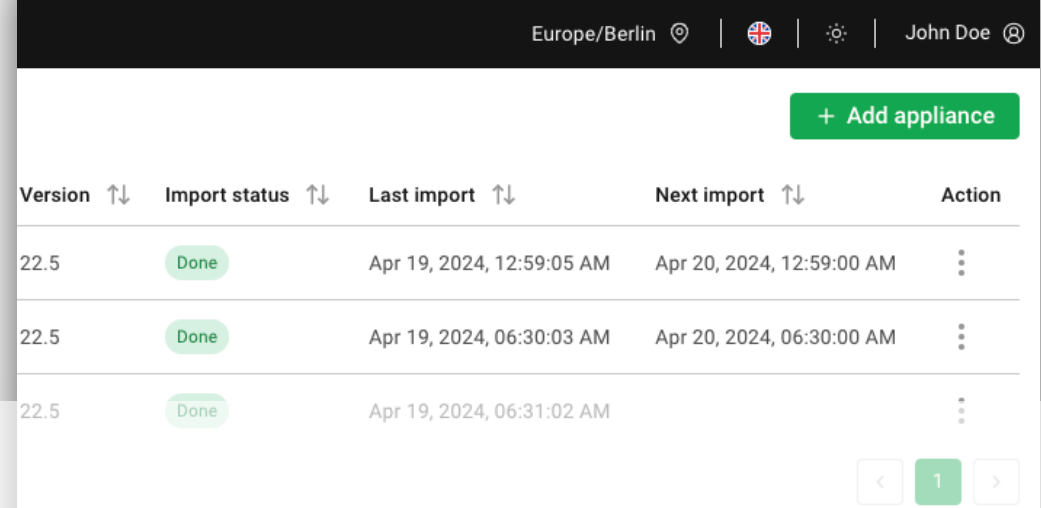


E2E TEST CHALLENGES

↯ Interference in Parallel Runs ↯

- **same resources** / entities / identifiers used for **each run**
  - ➡ e.g. same user
  - ➡ **test data** can be **influenced by another test run**

- our approach @ 
  - ➡ **generate** entities with random identifiers
  - ➡ and **delete** them after **each run**



- ↯ **unique resources** ↯
  - ➡ e.g. single machine

- possible approaches
  - ➡ provide set and **lend instances to test runs**
  - ➡ write test that **adapt to situation** (e.g. start import if none is running)

Greenbone

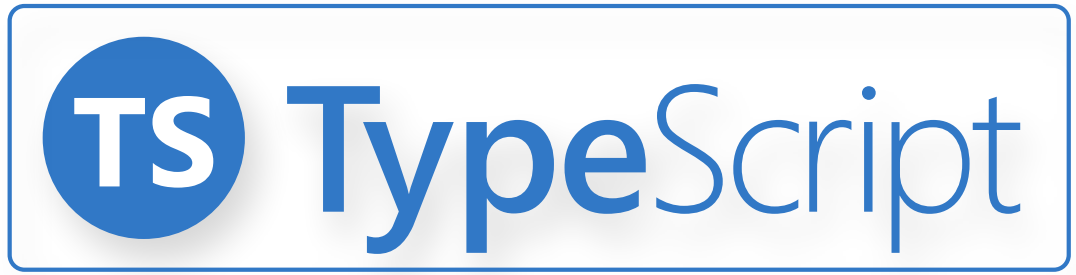
E2E TESTS: WRAP-UP

- asynchronism: use **wait for** constructs
- no control of data: **validate structure**, assert what you can know
- parallel runs: **generate entities** with random identifiers, delete them after each run

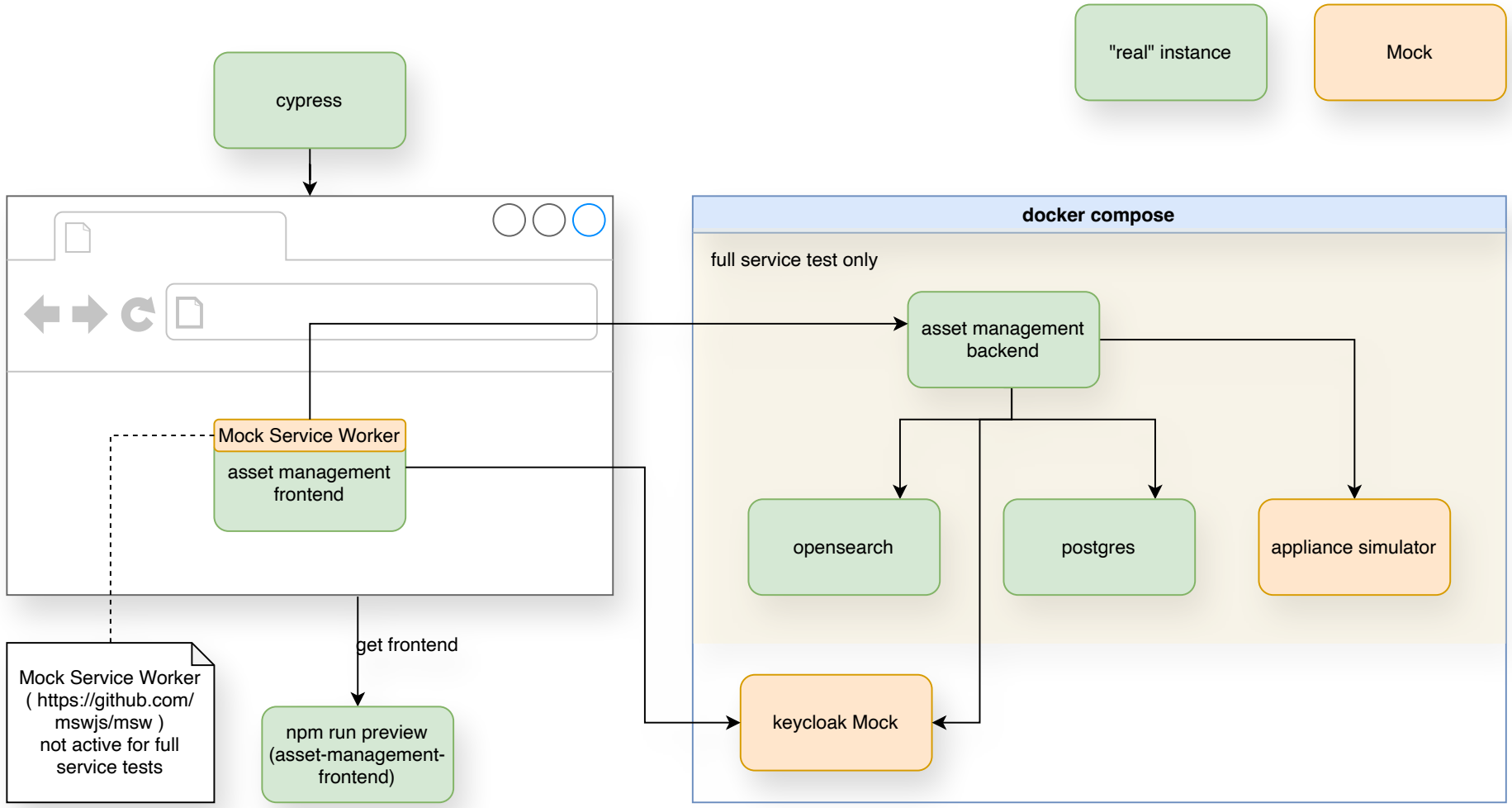
- keep **test durations** low
  - ➡ e.g. **use API calls** for test setup
- **expect efforts** for maintenance

FRONTEND SERVICE TESTS

FRONTEND SERVICE TESTS



FRONTEND SERVICE TESTS



Greenbone

FRONTEND SERVICE TESTS

Page Object Pattern

```
class VulnerabilityDetailsPage {
    elements = {
        breadcrumbCurrentPage: (text: string) =>
            cy.contains("div", `Vulnerability details: ${text}`),
        breadcrumbPreviousPage: () =>
            cy
                .contains("div", "Vulnerability details: ")
                .parent()
                .siblings("a"),
        vulnerabilityName: (text: string) => cy.contains("div", text),
        divContainer: () => cy.get("div"),
        linkToAffectedAssets: () => cy.contains("a", "here"),
    }

    verifyThatBreadcrumbContainsText(text: string) {
        this.elements.breadcrumbCurrentPage(text).should("exist")
    }

    verifyThatVulnerabilityNameContainsText(text: string) {
        this.elements.vulnerabilityName(text).should("exist")
    }
```

FRONTEND SERVICE TESTS

cypress

Page Object Pattern

```
describe("Frontend service tests for checking the vulnerabilities details page", () => {
    const vulnerabilityDetailsPage = require("../../pages/vulnerabilityDetailsPage")
    const vulnerabilityOverviewPage = require("../../pages/vulnerabilityOverviewPage")
    const pageLayout = require("../../pages/pageLayout")
    const vulnerabilityName = "The rexec service is running"

    before(() => {
        pageLayout.visitRootPage()
    })

    beforeEach(() => {
        pageLayout.enterPage("/vulnerabilities")
        vulnerabilityOverviewPage.cleanFilterContainer()
        vulnerabilityOverviewPage.navigateToVulnDetailsByIndex(0)
    })

    it("vulnerability details page should show correct information", () => {
        vulnerabilityDetailsPage.verifyThatBreadcrumbContainsText(
            vulnerabilityName
        )
        pageLayout.verifyThatPageTitleExists("Vulnerability details")
        vulnerabilityDetailsPage.verifyThatVulnerabilityNameContainsText(
            vulnerabilityName
        )
    })
```

FRONTEND SERVICE TESTS


```yaml
jobs:
  run-service-test:
    runs-on: "ubuntu-latest"
    strategy:
      matrix:
        node-version:
          - 18.x
          - 20.x
```

...

```yaml
      - name: setup service test environment
        run: make start-service-test-environment

      - name: run service tests
        run: |
          npm run preview&
          sleep 5s
          ./node_modules/.bin/cypress run --reporter mochawesome --spec "service-tests/cypress/tests/service-tests/*.cts"
```

...

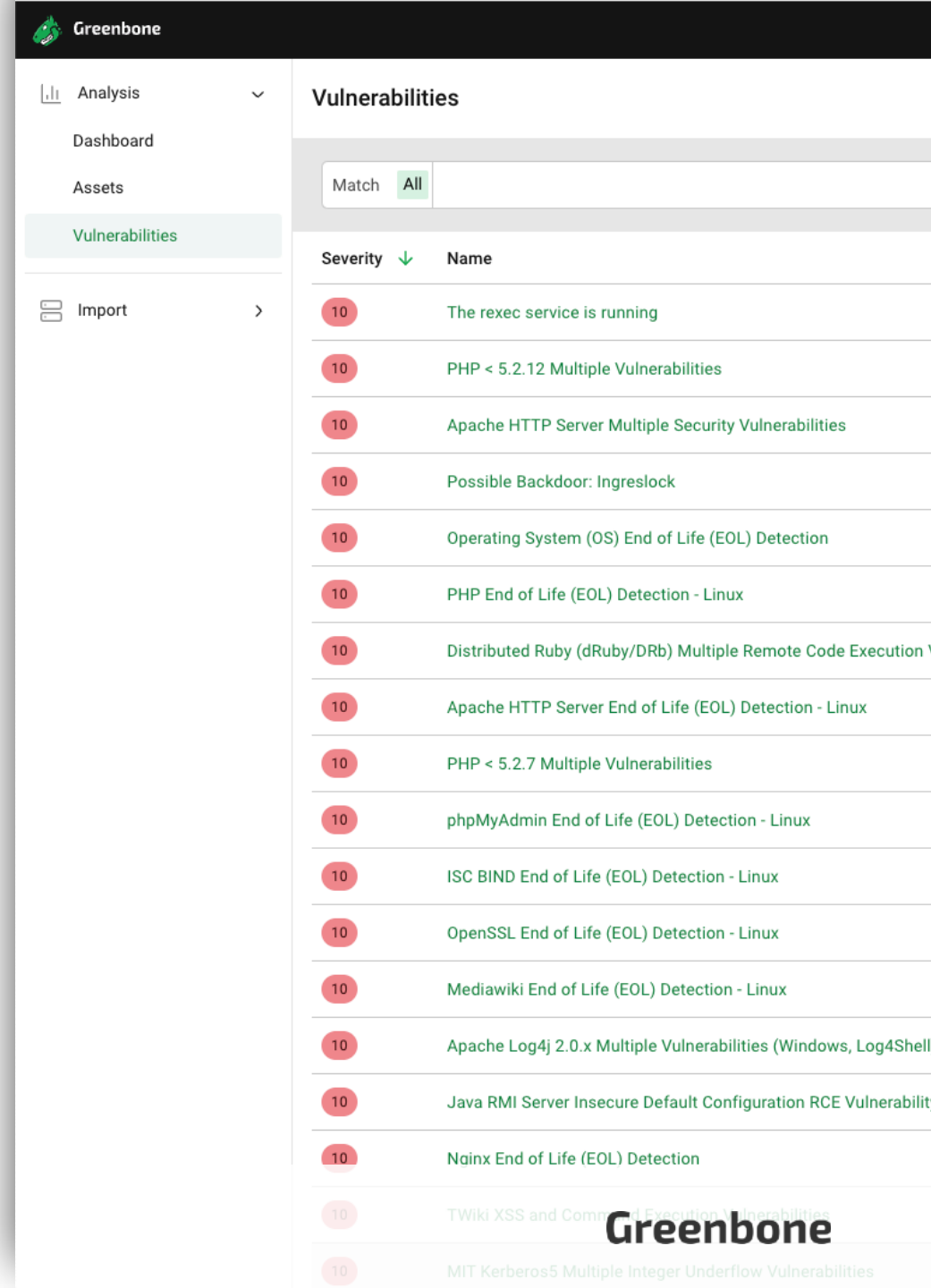
```yaml
      - name: Upload report folder
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: FrontendServiceTestReports_${{ matrix.node-version }}
          path: service-tests/cypress/reports/
          retention-days: 1
```

...

FE S TEST CHALLENGES

↯ Changing Data ↯ 

- **data** imported from **real systems changes** throughout time
  - ➡ **how many** and **which** entries will be there?

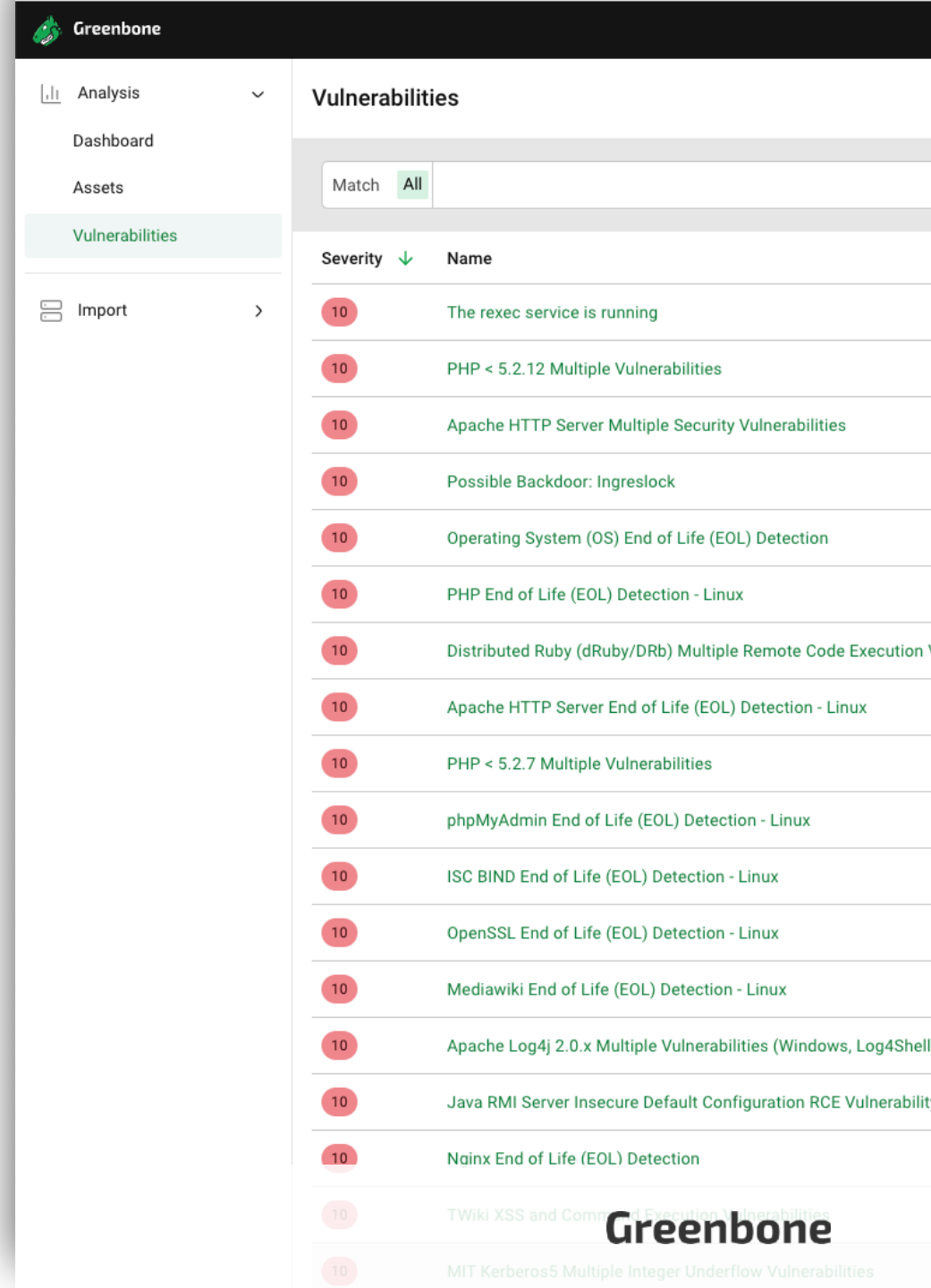


FE S TEST CHALLENGES

↯ Changing Data ↯ 

- **data** imported from **real systems changes** throughout time
  - ➡ **how many** and **which** entries will be there?

- our approach @ 
  - ➡ **mock** external systems
  - ➡ simplifies test setup
  - ➡ makes test shorter & more stable (reliable testdata!)



FE S TEST CHALLENGES

↯ Code Maintainability Decreases ↯

- code may become **difficult to read**
  - ➡ often selectors on the DOM are **not self-explanatory**
- code changes may become **cumbersome**
  - ➡ often **selectors** are **repeated** throughout the code
  - ➡ can be a **lot of effort** to **change** all locations

```
it("sorting assets by appliance name should work correctly", () => {
    cy.contains("h4", "Appliance name").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr")
        .eq(0)
        .should("contain", Cypress.env("appliance_name"))
    cy.contains("h4", "Appliance name").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr")
        .eq(0)
        .should("contain", Cypress.env("appliance_name"))
})

it("sorting assets by last scan should work correctly", () => {
    cy.contains("h4", "Last scan").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr").eq(0).should("contain", "Feb. 07, 2023 08:42:02am")
    cy.contains("h4", "Last scan").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr").eq(0).should("contain", "Feb. 07, 2023 09:36:46am")
})

it("the pagination should work for different sizes and pages", () => {
    cy.contains("span", "1-10 of 15 items").should("exist")
    cy.get("tbody>tr").then((rows) => {
        cy.expect(rows.length).to.equal(10)
    })
    cy.get("input[data-testid='paging-step-select'][value='10']").should(
        "exist"
    )

    // navigate to page 2
    cy.get("div[data-testid='pagination-container']").within(() => {
        cy.contains("button", "2").click().waitForStableDOM()
    })
    cy.contains("span", "11-15 of 15 items").should("exist")
    cy.get("tbody>tr").then((rows) => {
        cy.expect(rows.length).to.equal(5)
    })
    cy.get("tbody>tr").eq(0).should("contain", "192.168.123.36")

    // change paging size to 20
```

FE S TEST CHALLENGES

↯ Code Maintainability Decreases ↯

- code may become **difficult to read**
  - ➡ often selectors on the DOM are **not self-explanatory**
- code changes may become **cumbersome**
  - ➡ often **selectors** are **repeated** throughout the code
  - ➡ can be a **lot of effort** to **change** all locations

- our approach @ 
  - ➡ use **page object pattern**

```
it("sorting assets by appliance name should work correctly", () => {
    cy.contains("h4", "Appliance name").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr")
        .eq(0)
        .should("contain", Cypress.env("appliance_name"))
    cy.contains("h4", "Appliance name").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr")
        .eq(0)
        .should("contain", Cypress.env("appliance_name"))
})

it("sorting assets by last scan should work correctly", () => {
    cy.contains("h4", "Last scan").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr").eq(0).should("contain", "Feb. 07, 2023 08:42:02am")
    cy.contains("h4", "Last scan").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr").eq(0).should("contain", "Feb. 07, 2023 09:36:46am")
})

it("the pagination should work for different sizes and pages", () => {
    cy.contains("span", "1-10 of 15 items").should("exist")
    cy.get("tbody>tr").then((rows) => {
        cy.expect(rows.length).to.equal(10)
    })
    cy.get("input[data-testid='paging-step-select'][value='10']").should(
        "exist"
    )

    // navigate to page 2
    cy.get("div[data-testid='pagination-container']").within(() => {
        cy.contains("button", "2").click().waitForStableDOM()
    })
    cy.contains("span", "11-15 of 15 items").should("exist")
    cy.get("tbody>tr").then((rows) => {
        cy.expect(rows.length).to.equal(5)
    })
    cy.get("tbody>tr").eq(0).should("contain", "192.168.123.36")

    // change paging size to 20
```

FE S TEST CHALLENGES

↯ Code Maintainability Decreases ↯

- code
  - ➡ ofte
- code
  - ➡ ofte
  - ➡ can
- our a
  - ➡ use

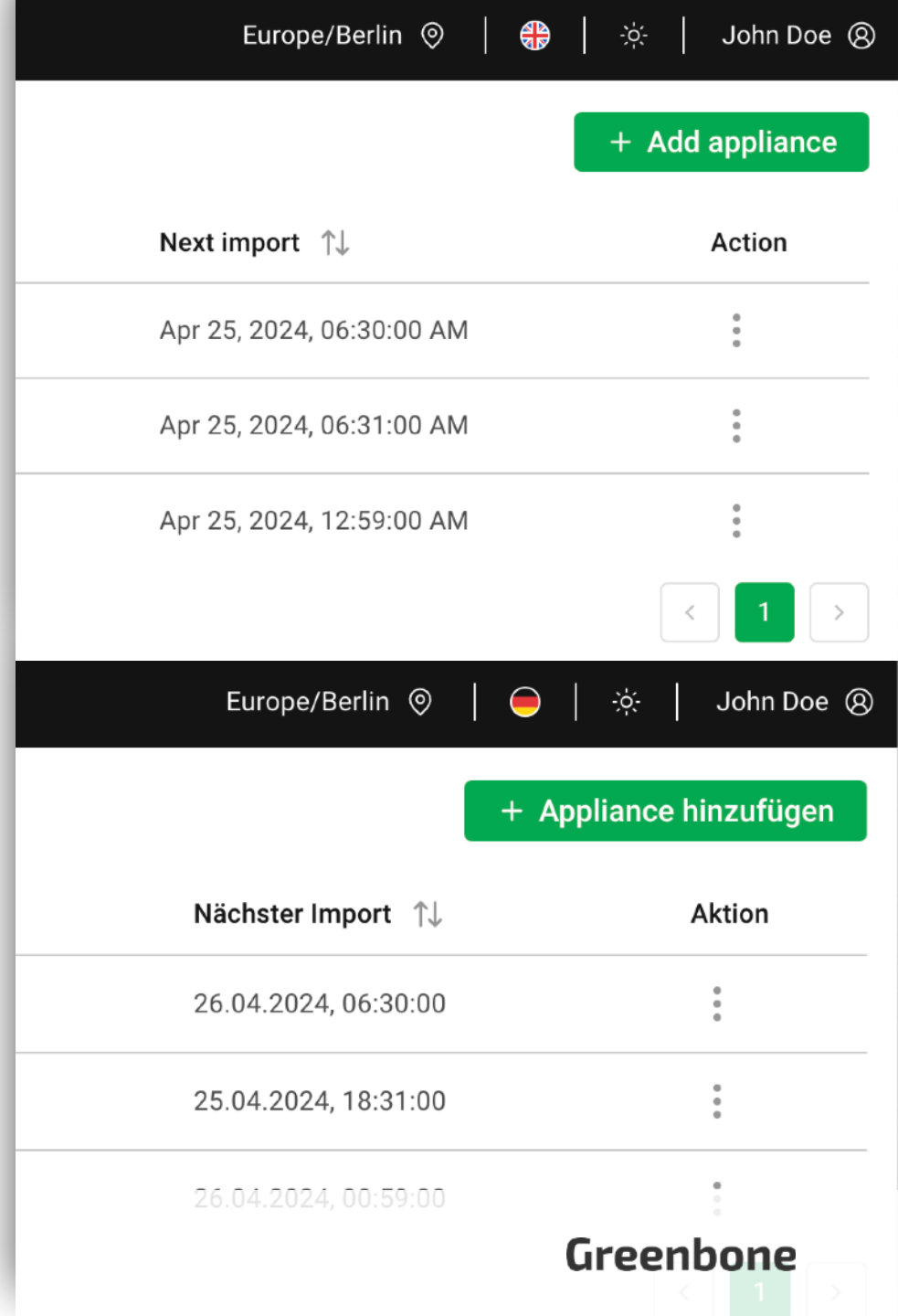
```
it("sorting assets by last scan should work correctly", () => {
    affectedAssetsPage.tableController().sortHeader("Last scan")
    affectedAssetsPage.verifyTharFirstRowsContainsText(
        "Feb 07, 2023, 08:42:02 AM"
    )
    affectedAssetsPage.tableController().sortHeader("Last scan")
    affectedAssetsPage.verifyTharFirstRowsContainsText(
        "Feb 07, 2023, 09:36:46 AM"
    )
})

it("the pagination should work for different sizes and pages", () => {
    affectedAssetsPage.paginationTextEquals("1-10 of 15 items")
    affectedAssetsPage.tableController().verifyRowAmountIsEqualTo(10)
    affectedAssetsPage.tableController().verifySelectedPageSize(10)

    affectedAssetsPage.tableController().gotoPage(2)
    affectedAssetsPage.paginationTextEquals("11-15 of 15 items")
    affectedAssetsPage.tableController().verifyRowAmountIsEqualTo(5)
    affectedAssetsPage.verifyTharFirstRowsContainsText("192.168.123.36")

    affectedAssetsPage.tableController().setPageSize(20)
    affectedAssetsPage.paginationTextEquals("1-15 of 15 items")
    affectedAssetsPage.tableController().verifySelectedPageSize(20)
```

```
it("sorting assets by appliance name should work correctly", () => {
    cy.contains("h4", "Appliance name").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr")
        .eq(0)
        .should("contain", Cypress.env("appliance_name"))
    cy.contains("h4", "Appliance name").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr")
        .eq(0)
        .should("contain", Cypress.env("appliance_name"))
})

it("sorting assets by last scan should work correctly", () => {
    cy.contains("h4", "Last scan").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr").eq(0).should("contain", "Feb. 07, 2023 08:42:02am")
    cy.contains("h4", "Last scan").click()
    cy.waitForStableDOM()
    cy.get("tbody>tr").eq(0).should("contain", "Feb. 07, 2023 09:36:46am")
})

it("the pagination should work for different sizes and pages", () => {
    cy.contains("span", "1-10 of 15 items").should("exist")
    cy.get("tbody>tr").then((rows) => {
        cy.expect(rows.length).to.equal(10)
    })
    cy.get("input[data-testid='paging-step-select'][value='10']").should(
        "exist"
    )

    // navigate to page 2
    cy.get("div[data-testid='pagination-container']").within(() => {
        cy.contains("button", "2").click().waitForStableDOM()
    })
    cy.contains("span", "11-15 of 15 items").should("exist")
    cy.get("tbody>tr").then((rows) => {
        cy.expect(rows.length).to.equal(5)
    })
    cy.get("tbody>tr").eq(0).should("contain", "192.168.123.36")

    // change paging size to 20
```

FE S TEST CHALLENGES

↯ time and translation varies ↯

- **i18n**: different time zones, date formats, translations
  - ➡ may **differ** between systems
  - ➡ **tests may fail** when checking for date, time or translations

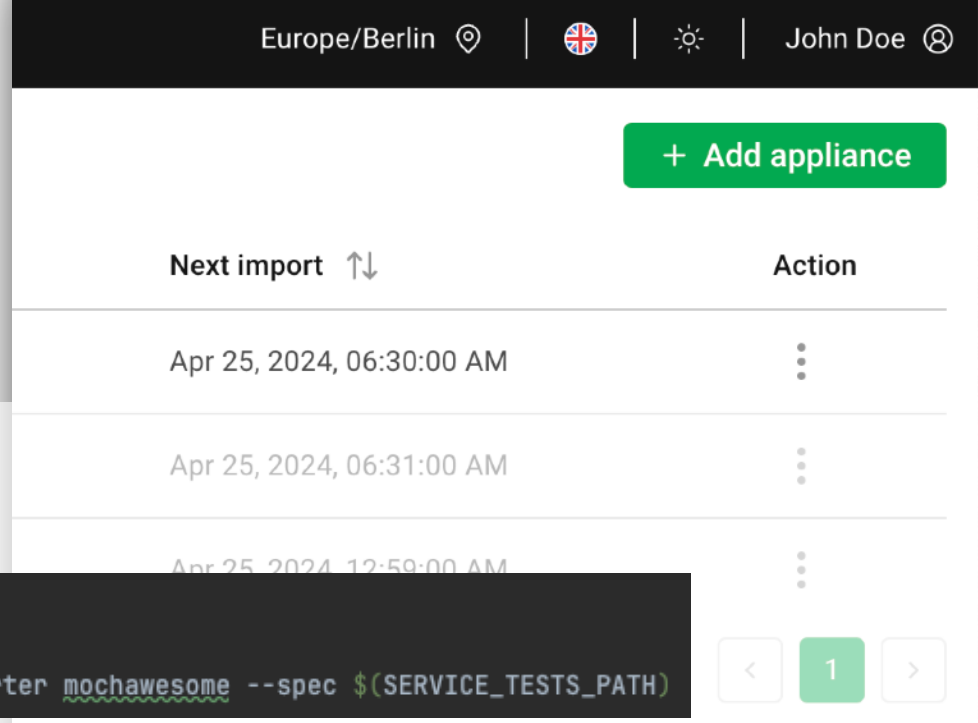


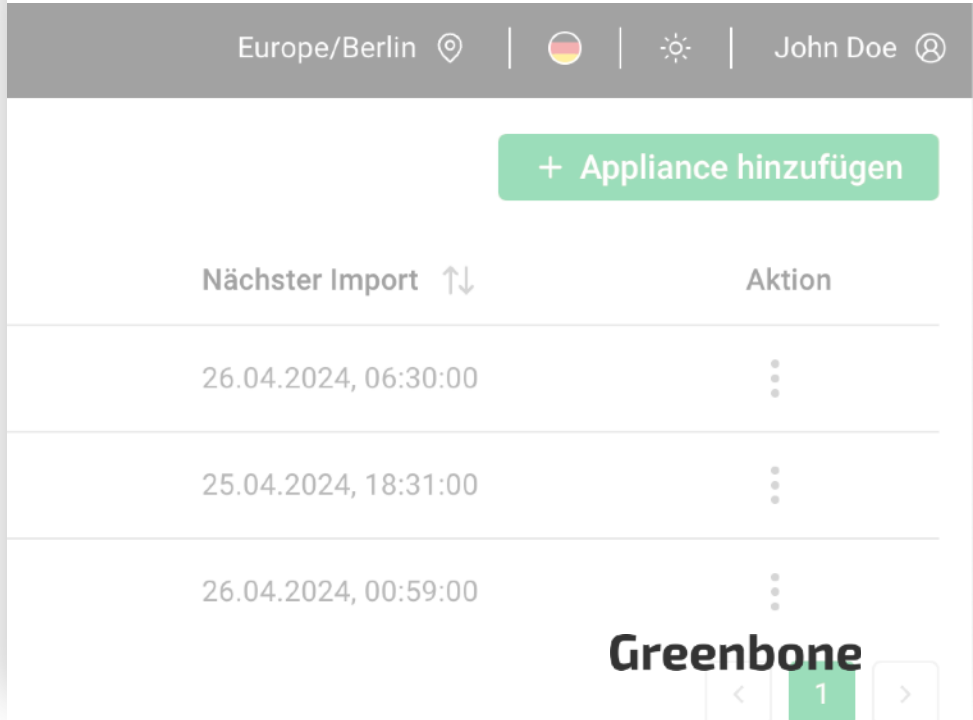
FE S TEST CHALLENGES

↯ time and translation varies ↯

- **i18n**: different time zones, date formats, translations
  - ➡ may **differ** between
  - ➡ **tests may fail** when checking for date, time or translations

```
56  .PHONY: start-service-tests
57  start-service-tests:
58      TZ=UTC ./node_modules/.bin/cypress run --reporter mochawesome --spec $(SERVICE_TESTS_PATH)
```





- our approach @ 
  - ➡ **set time zone** explicitly
  - ➡ **select locale** explicitly (or **rely on default**)

FE S TESTS: WRAP-UP

- consider **full service tests** (with backend)
- **mock** systems outside scope (except data stores)
- use **page object** pattern
- set **time zone / locale** explicitly

- like e2e
  - ➡ asynchronism: use **wait for** constructs
  - ➡ keep **test durations** low
  - ➡ **expect efforts** for maintenance

BACKEND SERVICE TESTS

BACKEND SERVICE TESTS

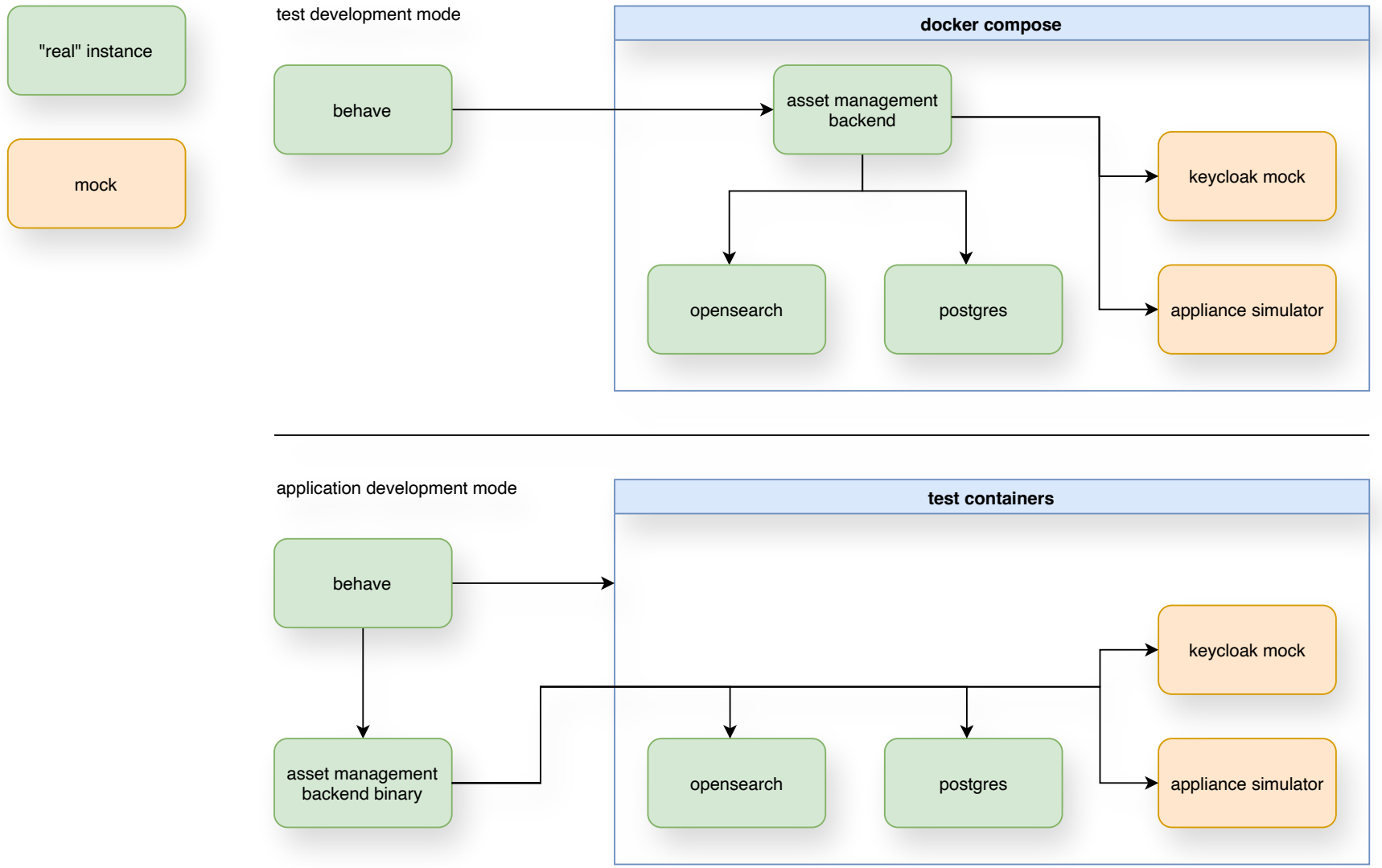
GitHub

python™

docker®

Testcontainers

BACKEND SERVICE TESTS



Greenbone

BACKEND SERVICE TESTS

```
98    @fixture.appliance.creation
99    @fixture.appliance.cleanup
100   Scenario: Updating an existing appliance with invalid ssh key fingerprint should deliver a 400 status and the expected error (via PUT /appliances/{applianceId})
101     Given the user is logged in
102     And the appliance 'Prepared Test Appliance' is available
103     When the existing appliance 'Prepared Test Appliance' is updated using invalid fingerprint
104     Then a 400 status is delivered
105     And the response has the text 'greenbone/asset/host-key-invalid' for the field 'type'
106     And the response has the text 'SHA256:yyq1KasQ622bc5Ez5nH/fh1Mcse2ZFhewcG4VH3Mir4' for the custom field 'retrievedSshFingerprint'
```

```
36  @step(u'a (?P<status_code>\\d+) status is delivered')
37  def step_check_error_code(context, status_code):
38      wait.wait_for_context_response(context, condition=lambda r: assert_that(r.status_code)
39                                     .is_equal_to(int(status_code)))
```

BE S TESTS: WRAP-UP

- like frontend service tests:
  - **mock systems** outside scope (except data stores)
  - asynchronism: use **wait for** constructs
  - keep **test durations** low
  - **expect efforts** for maintenance

OUTLOOK

- Sometimes we get "We detected that the **Electron** Renderer process just **crashed**." in the CI 
  - ➡ we need to investigate
- run e2e test after each deployment
- synchronize QA approach in whole company
- automated cross browser test
- contract driven testing (e.g. pact)
- metrics for quality (e.g. escaped bugs)

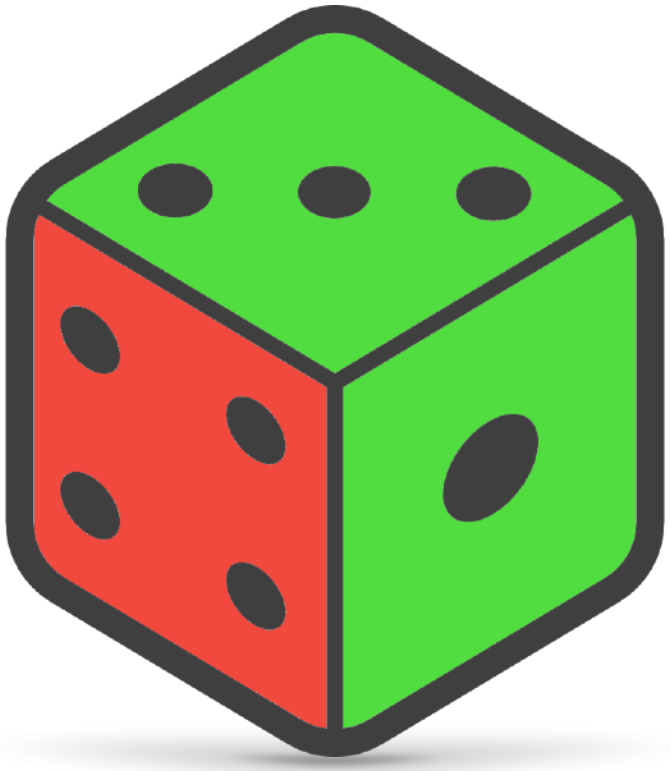
EXPLORATORY TESTING

- **Scripted** (espec. automated) **testing is important** e.g. for regression and possibly short feedback cycles
- But think of your scripted tests as **threads of a net**
- They catch whatever you thought of, but there are always **things** that are **not caught**
- Given the **unlimited space of possibilities** it is impossible to weave the net fine enough to catch every possibility
- This is where **exploratory testing** comes in handy!

https://www.goodreads.com/book/show/15980494-explore-it

BEWARE OF THE DICE

don't ignore flaky tests!



Thank You!

Questions?

LET’S CREATE A FUTURE TOGETHER

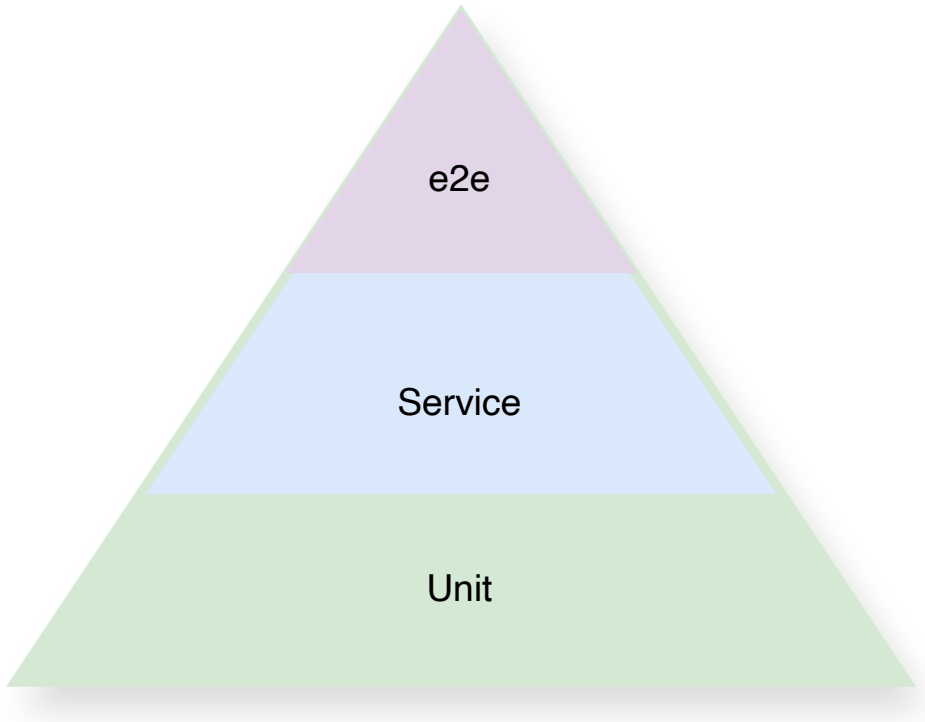
We are more than just a business: We are a team of creative thinkers, innovative problem solvers, dedicated professionals – and passionate humans.

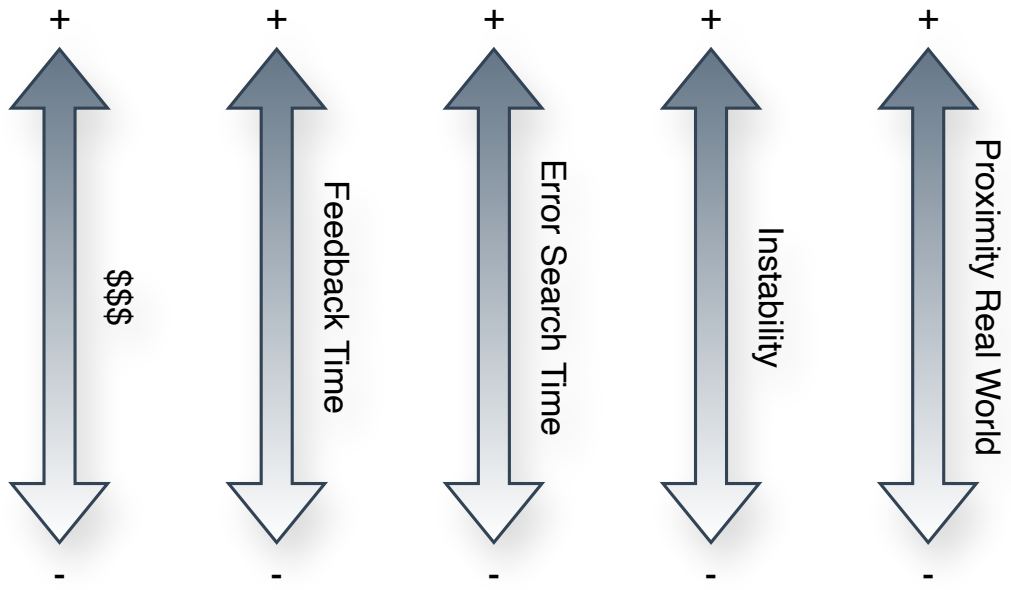
Be part of the Solution: find our job postings here

Our key benefits:
Remote Work options: 100% remote work possible
Flexible working hours
Opportunities for participation in company decisions

Backup

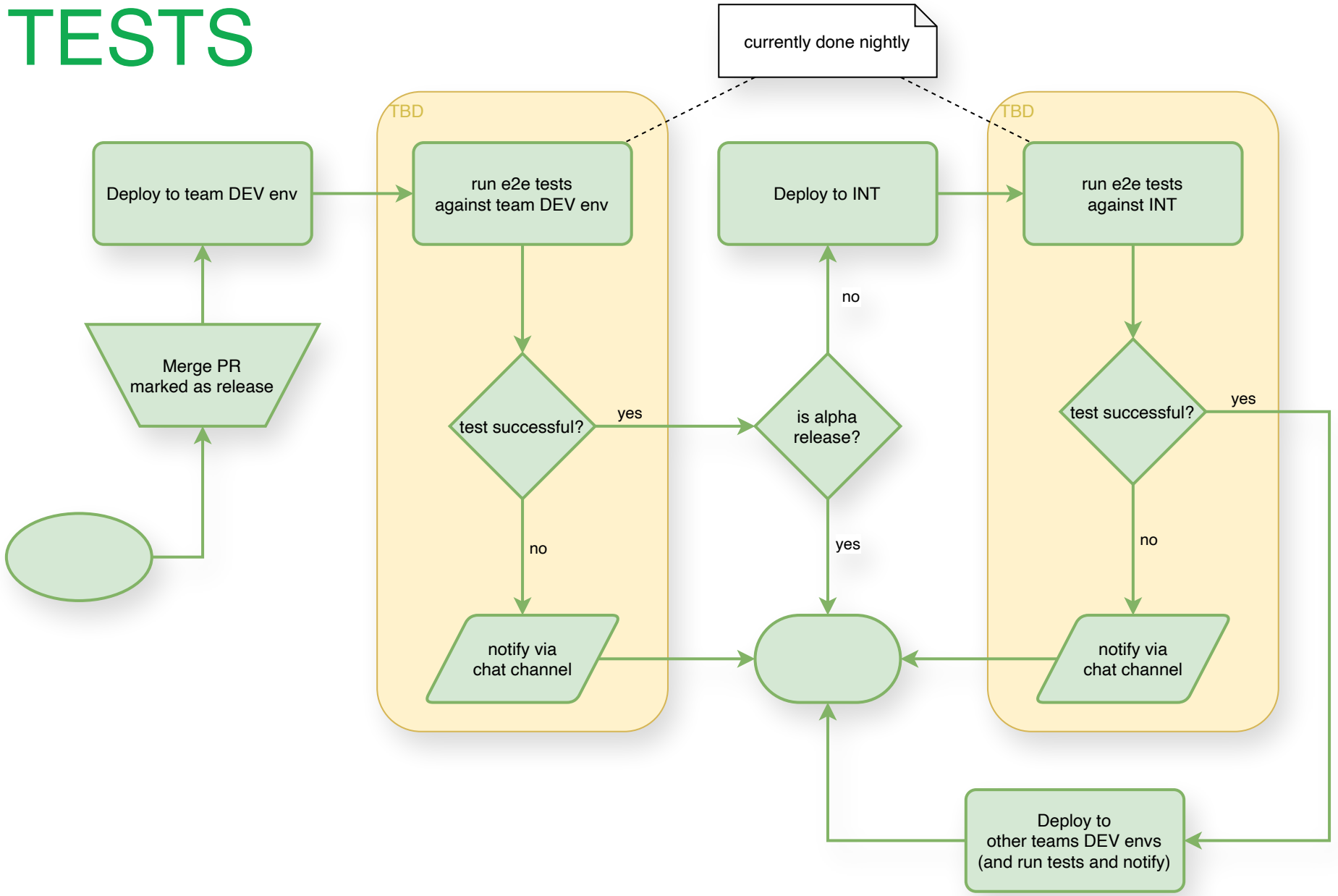
TEST PYRAMID





E2E TESTS

E2E TESTS



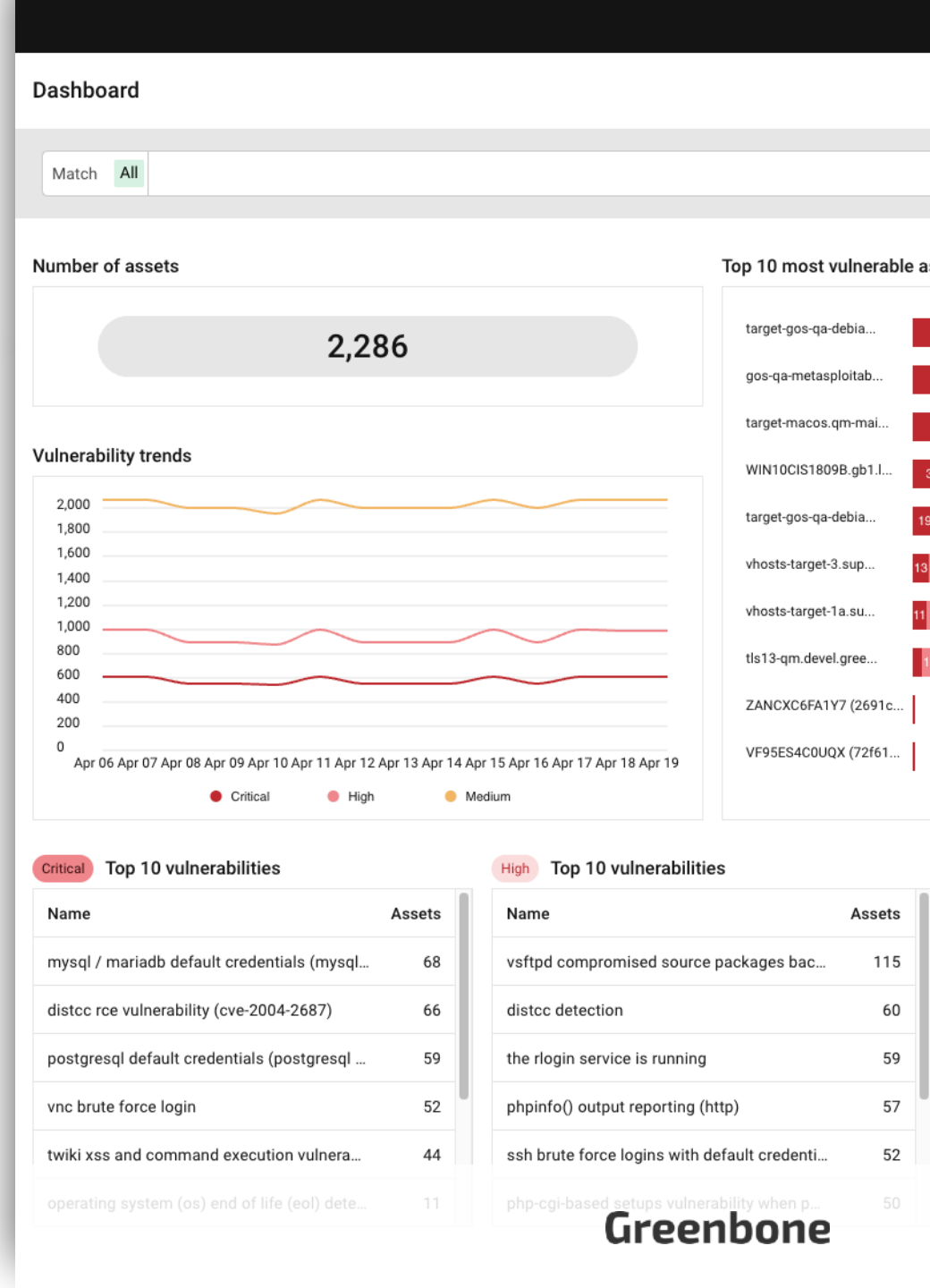
Greenbone

E2E TEST CHALLENGES

↯ Active Users on Test System ↯

- **test systems** are also used for **manual testing** and presentations

➡ test outcomes can be **influenced** by **user actions**



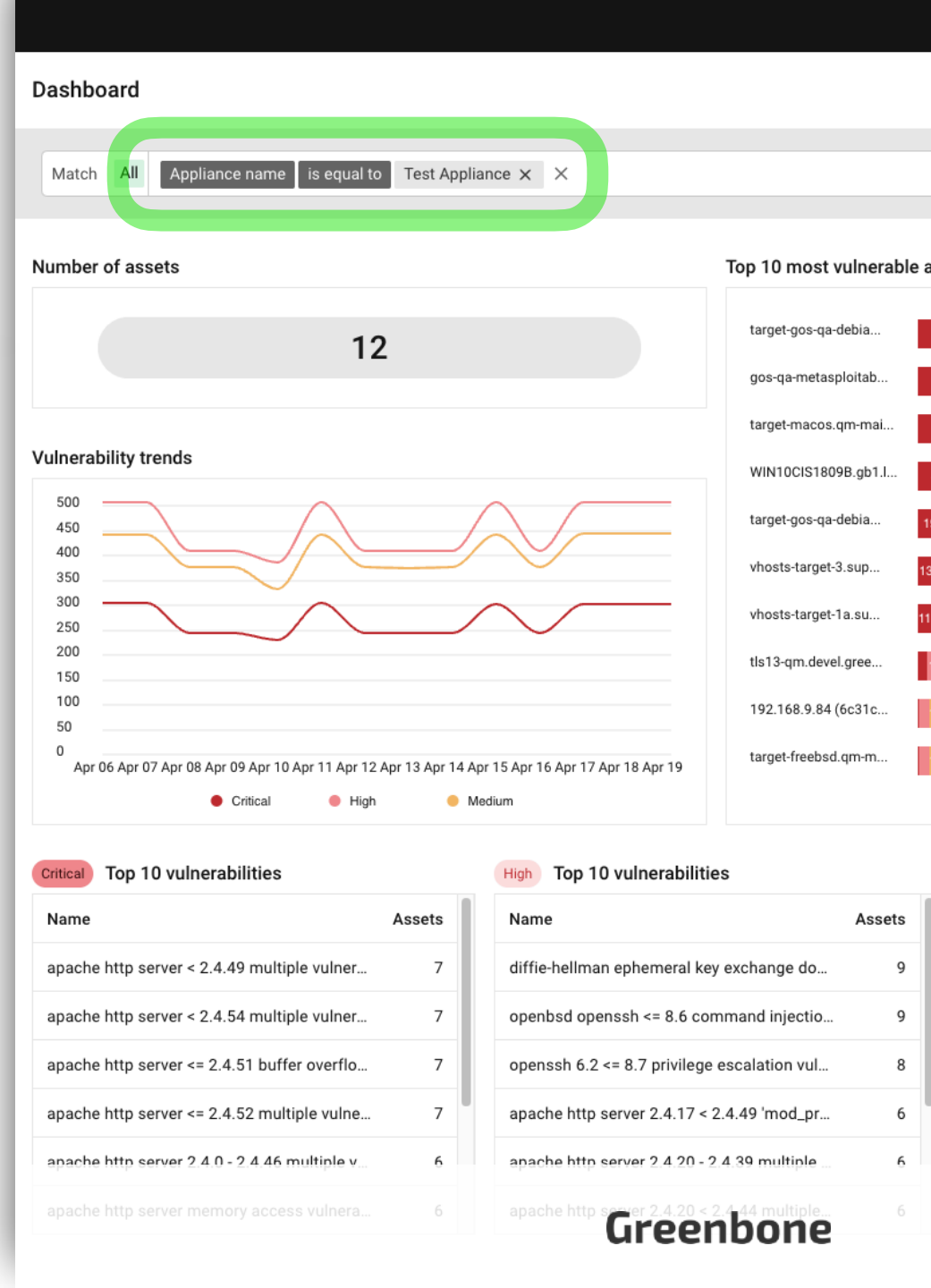
Greenbone

E2E TEST CHALLENGES

↯ Active Users on Test System ↯

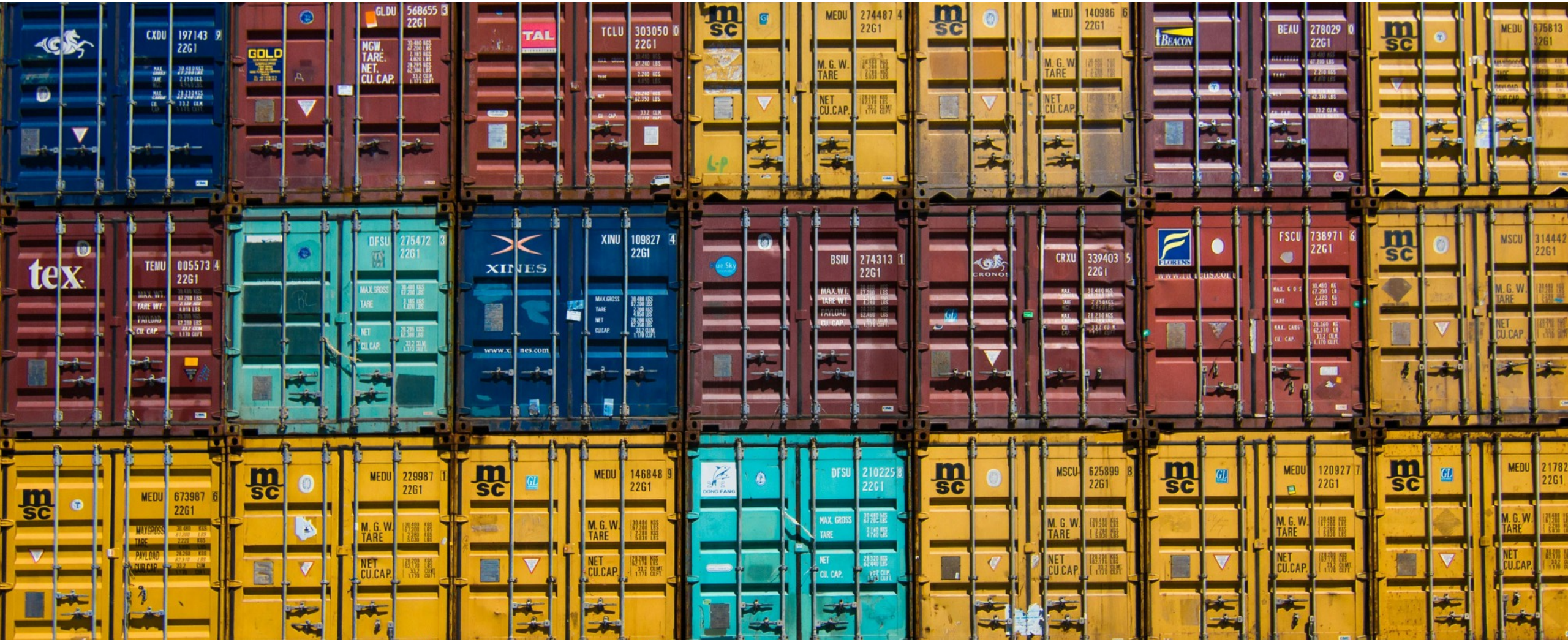
- **test systems** are also used for **manual testing** and presentations
  - ➡ test outcomes can be **influenced** by **user actions**

- our approach @ 
  - ➡ **label** test data and **filter** for it
  - ➡ conventions on **what** (not) **to change**
  - ➡ alternative: separate envs ($$$), create separate accounts



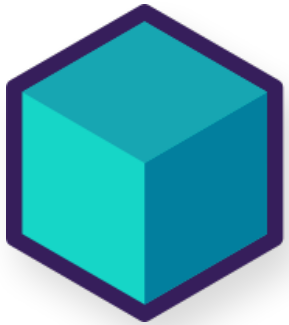
Greenbone

BACKEND UNIT TESTS



Greenbone

BACKEND UNIT TESTS



Testcontainers

BACKEND UNIT TESTS

```
327 func TestVulnerabilityExport(t *testing.T) {
328     testCases := map[string]testCase{
329         "happy path": {
330             acceptHeader:              "text/csv",
331             requestBody:               vulnerabilityRequest(),
332             expectedStatusCode:        http.StatusOK,
333             mockedVulnerabilitySearch: getMockedVulnerabilitySearch(t),
334             bodyValidation: func(mvc rest_test.MockMvc) {
335                 mvc.ExpectBody( body: "Severity,Name,QoD,Total Assets,Solution type\n" +
336                     "7.5,FTP Brute Force Logins Reporting,95,1,Mitigation\n")
337             },
338         },
339         "invalid media type": {
340             acceptHeader:       "application/json",
341             expectedStatusCode: http.StatusNotAcceptable,
342             bodyValidation: func(mvc rest_test.MockMvc) {
343                 mvc.ExpectJson( json: "{\"title\":\"Invalid media type was specified in the Accept header (should be text/csv)\", \"type\":\"greenbone/generic-error\"}")
344             },
345         },
346         "invalid body": {
```

BACKEND UNIT TESTS

```
for name := range testCases {
    t.Run(name, func(t *testing.T) {
        mockMvc := setupExport(testCases[name])
        mockMvc = mockMvc.
            Perform(t, rest_test.Post( url: "/api/asset-management/export/vulnerabilities").
                Header( key: "Accept", []string{
                    testCases[name].acceptHeader,
                }).
                Content(testCases[name].requestBody)).
            ExpectStatusCode(testCases[name].expectedStatusCode)
        testCases[name].bodyValidation(mockMvc)
    })
}
```

BACKEND UNIT TESTS - OPENSEARCH

```
56  "paging and sorting": {
57      oid: "ba312986-9667-413a-95e8-030dcb039b37",
58      resultSelector: query.ResultSelector{
59          Filter: &filter.Request{},
60          Sorting: &sorting.Request{
61              SortColumn:    "name",
62              SortDirection: "asc",
63          },
64          Paging: &paging.Request{
65              PageIndex: 0,
66              PageSize:  2,
67          },
68      },
69      expectedRequest:       string(readFile(t, name: "testJsons/vulnerabilityByAssetRequestWithPagingAndSorting.json")),
70      expectedResponse:      string(readFile(t, name: "testJsons/vulnerabilityByAssetResponse.json")),
71      expectedResults:       vulnerabilitiesByAssets(),
72      expectedTotalRowCount: 286,
73      expectedError:         nil,
74  },
```

BACKEND UNIT TESTS - OPENSEARCH

```
"paging and sorting": {
    oid: "ba312986-9667-413a-95e8-030dcb039b37",
    resultSelector: query.ResultSelector{
        Filter: &filter.Request{},
        Sorting: &sorting.Request{
            SortColumn:    "name",
            SortDirection: "asc",
        },
        Paging: &paging.Request{
            PageIndex: 0,
            PageSize:  2,
        },
    },
    expectedRequest:       string(readFile(t, name: "testJsons/vulnerability
    expectedResponse:      string(readFile(t, name: "testJsons/vulnerability
    expectedResults:       vulnerabilitiesByAssets(),
    expectedTotalRowCount: 286,
    expectedError:         nil,
},
```

```
{
  "from": 0,
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "asset.id": {
              "value": "ba312986-9667-413a-95e8-030dcb039b37"
            }
          }
        }
      ]
    }
  },
  "size": 2,
  "sort": [
    {
      "vulnerabilityTest.name.keyword": {
        "order": "asc"
      }
    }
  ]
}
```

BE UNIT TESTS: WRAP-UP

- Make use of **data tables** / data driven testing

- make it possible to **verify test expectations**
- use **testcontainers** only **where unavoidable**

- like backend service tests:
  - ➡ **mock functions** outside scope (except data stores)
  - ➡ asynchronism: use **wait for** constructs
  - ➡ keep **test durations** low

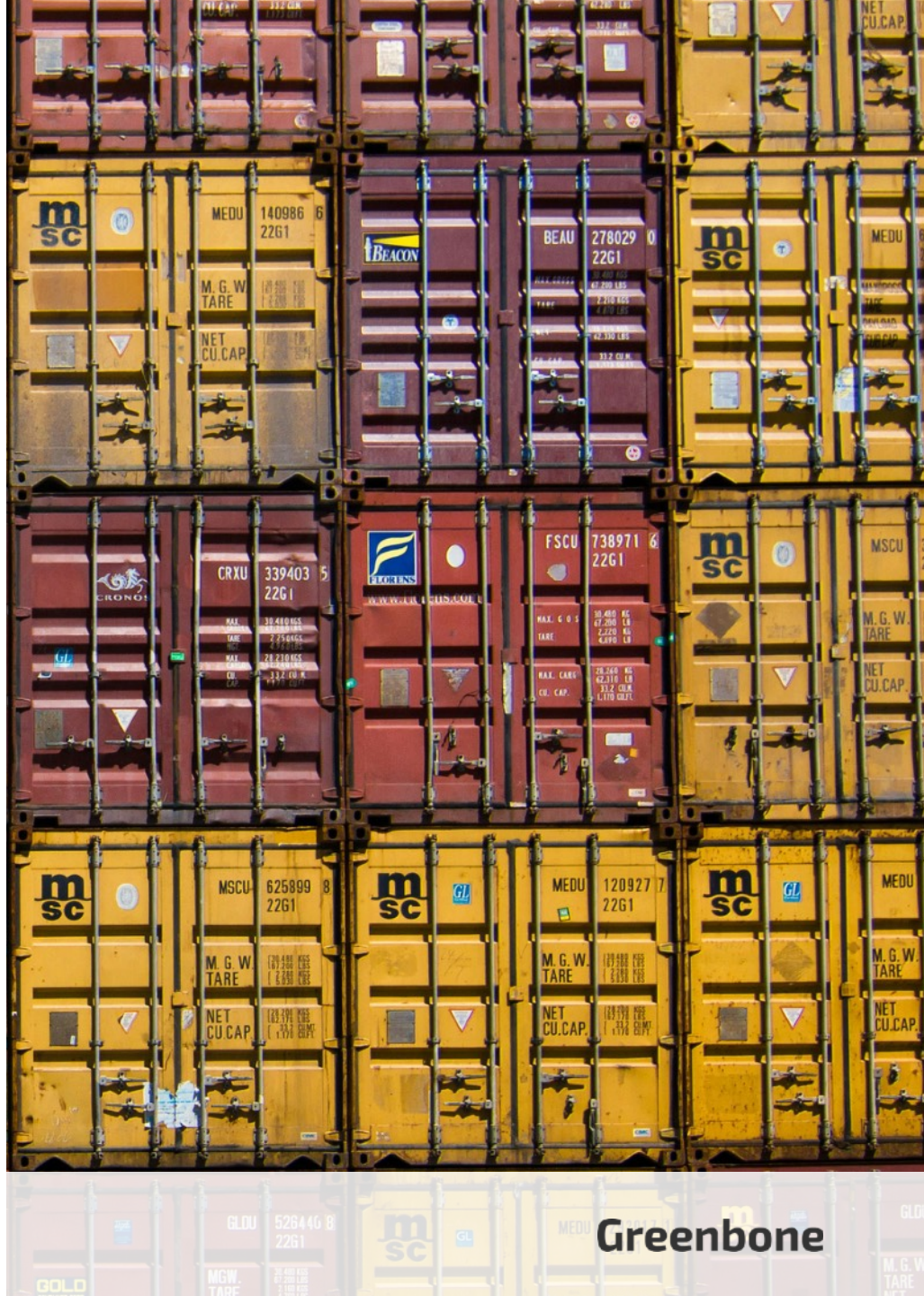


IMAGE CREDITS


Foto von Nicholas Green auf Unsplash


Foto von Jonathan Safa auf Unsplash

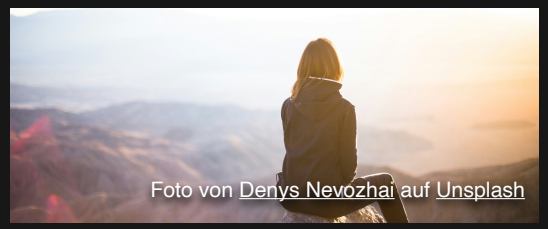


Foto von Denys Nevozhai auf Unsplash


Foto von Nicolas Hoizey auf Unsplash


Foto von Jukan Tateisi auf Unsplash


Foto von Aideal Hwa auf Unsplash


Foto von Dipesh Shrestha auf Unsplash

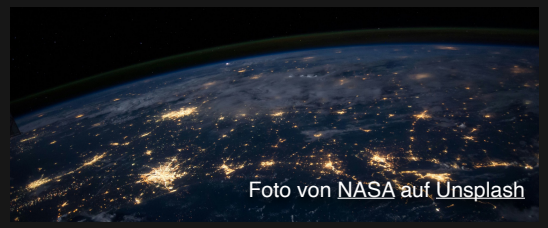


Foto von NASA auf Unsplash


Foto von Killian Cartignies auf Unsplash

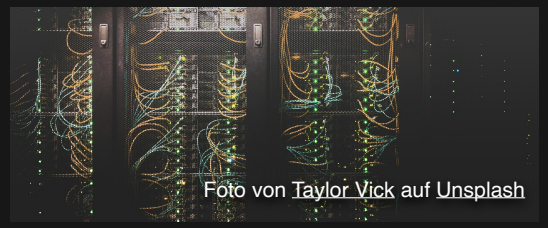


Foto von Taylor Vick auf Unsplash


Foto von Guillaume Bolduc auf Unsplash


Foto von Jonas Jacobsson auf Unsplash