code intelligence

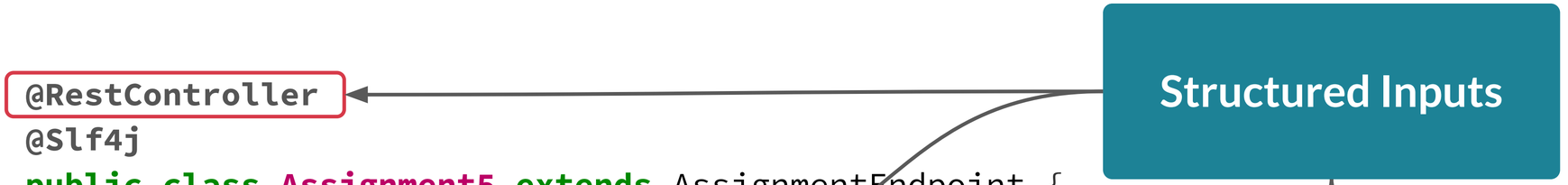
Coverage-Guided Fuzzing for Web Applications

Khaled Yakdan
CTO & Co-Founder

Why?

- Most critical code in web services
- Enormous security impact
- Critical OWASP vulnerabilities

Typical Challenges of Web Fuzzing in a Single Piece of Code




```
@RestController
@Slf4j
public class Assignment5 extends AssignmentEndpoint {
    //...
    @PostMapping("/challenge/5")
    @ResponseBody
    public AttackResult login(@RequestParam String username_login, @RequestParam String password_login) {
        //...
        if (!"Larry".equals(username_login)) {
            return failed(this).feedback("user.not.larry").feedbackArgs(username_login).build();
        }
        //...
        PreparedStatement stmt = connection.prepareStatement("select password from users where userid = '" +
username_login + "' and password = '" + password_login + "'");
        ResultSet resultSet = statement.executeQuery();
        //...
    }
}
```

Typical Challenges of Web Fuzzing in a Single Piece of Code

```java
@RestController
@Slf4j
public class Assignment5 extends AssignmentEndpoint {
    //...
    @PostMapping("/challenge/5")
    @ResponseBody
    public AttackResult login(@RequestParam String username_login, @RequestParam String password_login) {
        //...
        if (!"Larry".equals(username_login)) {
            return failed(this).feedback("user.not.larry").feedbackArgs(username_login).build();
        }
        //...
        PreparedStatement stmt = connection.prepareStatement("select password from users where userid = '" +
username_login + "' and password = '" + password_login + "'");
        ResultSet resultSet = statement.executeQuery();
        //...
    }
}
```

Bug Detectors for OWASP Vulnerabilities

Typical Challenges of Web Fuzzing in a Single Piece of Code

```java
@RestController
@Slf4j
public class Assignment5 extends AssignmentEndpoint {
    //...
    @PostMapping("/challenge/5")
    @ResponseBody
    public AttackResult login(@RequestParam String username_login, @RequestParam String password_login) {
        //...
        if (!"Larry".equals(username_login)) {
            return failed(this).feedback("user.not.larry").feedbackArgs(username_login).build();
        }
        //...
        PreparedStatement stmt = connection.prepareStatement("select password from users where userid = '" +
username_login + "' and password = '" + password_login + "'");
        ResultSet resultSet = statement.executeQuery();
        //...
    }
}
```

Many Such Checks

Challenge 4: Containers, Kubernetes and more Complexity for Deployment

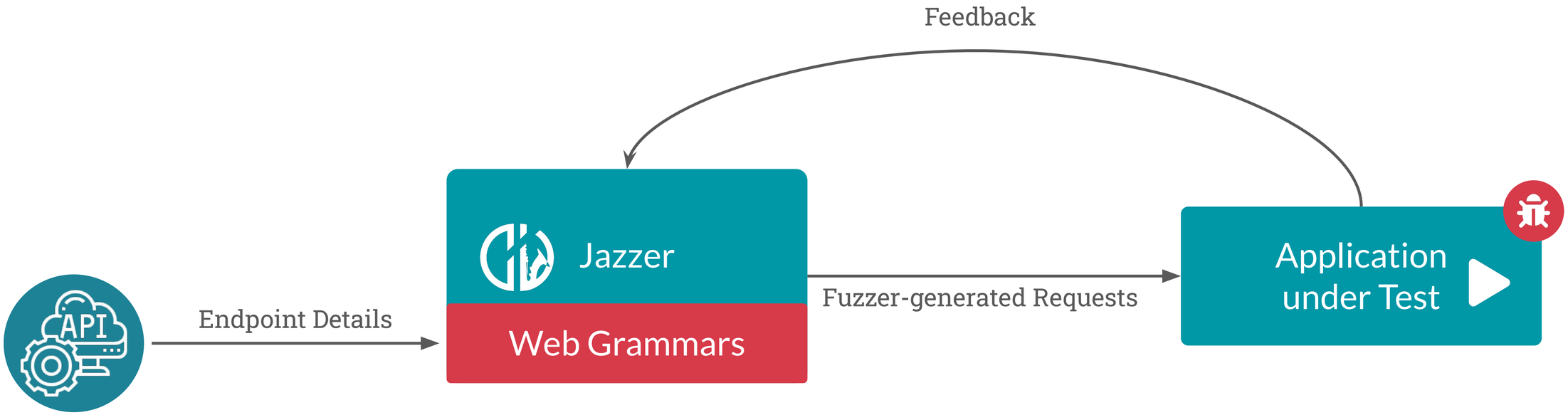
```yaml
version: '2.0'

services:
  webgoat:
    image: webgoat/webgoat-8.0
    user: webgoat
    environment:
      - WEBWOLF_HOST=webwolf
      - WEBWOLF_PORT=9090
      - spring.datasource.url=jdbc:postgresql://webgoat_db:5432/webgoat?user=webgoat&password=webgoat
      - spring.datasource.username=webgoat
      - spring.datasource.password=webgoat
      - spring.datasource.driver-class-name=org.postgresql.Driver
      - spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQL10Dialect
      - webgoat.server.directory=/home/webgoat/.webgoat/
      - webgoat.user.directory=/home/webgoat/.webgoat/
    ports:
      - "8080:8080"
  webwolf:
    image: webgoat/webwolf
    environment:
      - spring.datasource.url=jdbc:postgresql://webgoat_db:5432/webgoat?user=webgoat&password=webgoat
      - spring.datasource.username=webgoat
      - spring.datasource.password=webgoat
      - spring.datasource.driver-class-name=org.postgresql.Driver
      - spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQL10Dialect
    ports:
      - "9090:9090"
  webgoat_db:
    image: postgres:10.12
# Uncomment to store the state of the database on the host.
#    volumes:
#      - ./database:/var/lib/postgresql
    environment:
      - POSTGRES_PASSWORD=webgoat
      - POSTGRES_USER=webgoat
      - POSTGRES_DB=webgoat
    ports:
      - "5432:5432"
```

Challenge 1: We need structured inputs

- Web protocols are pretty standardized
  - REST
  - Web Forms
  - SOAP
  - GraphQL
- Solution
  - Structure-aware fuzzing

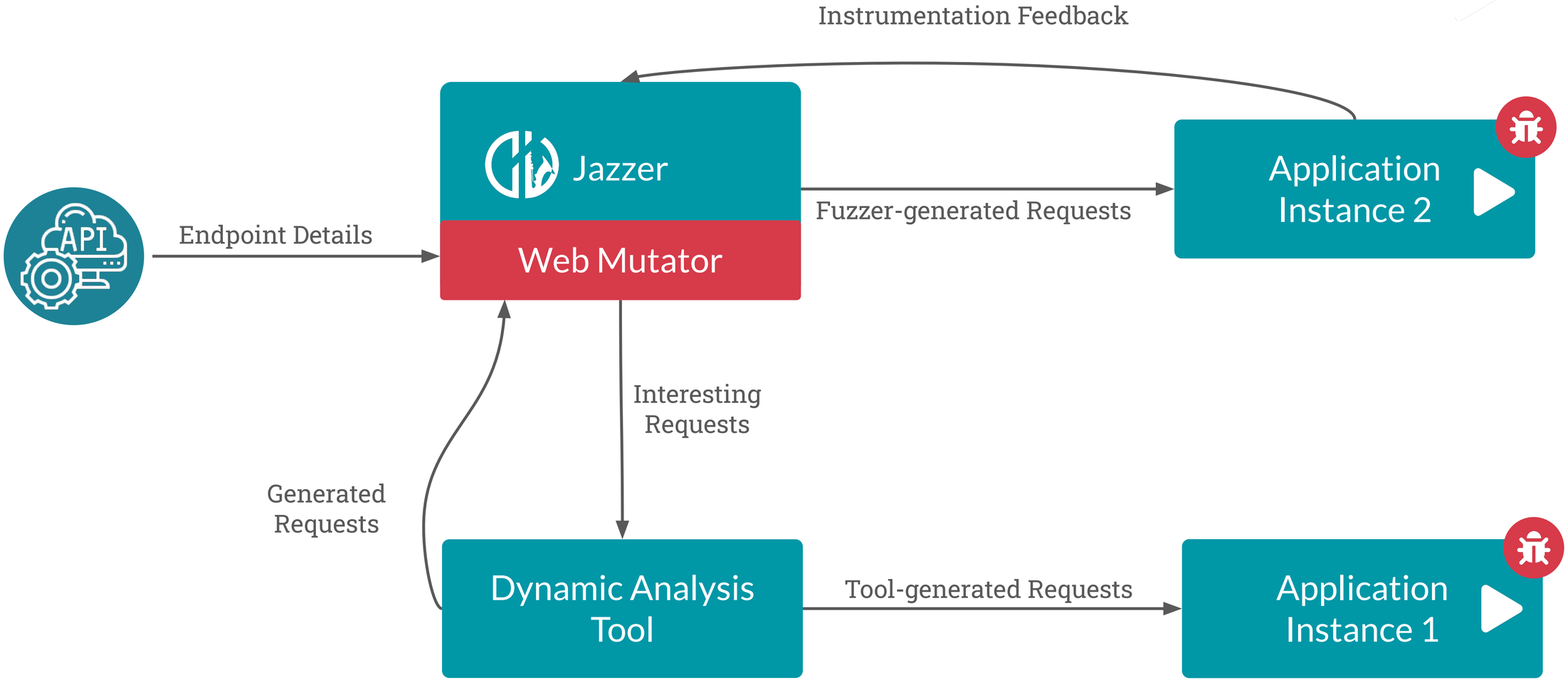
Solution 1: Let’s Bring Structure-aware Fuzzing into Web



Challenge 2: We need to find OWASP Issues

- We’re interested in special classes of security issues
  - OWASP Top 10
  - OWASP API Top 10
- Blackbox testing has false positives
- Solution:
  - Instrumentation-based bug detections
  - Integrating with other DAST tools

Solution 2: Combine Coverage-based Fuzzing w/ Existing Tooling



Solution 2: Instrumentation-based Bug Detectors

```
@RestController
@Slf4j
public class Assignment5 extends AssignmentEndpoint {
    //...
    @PostMapping("/challenge/5")
    @ResponseBody
    public AttackResult login(@RequestParam String username_login, @RequestParam String password_login) {
        //...
        if (!"Larry".equals(username_login)) {
            return failed(this).feedback("user.not.larry").feedbackArgs(username_login).build();
        }
        //...
        PreparedStatement stmt = connection.prepareStatement("select password from users where userid = '" +
username_login + "' and password = '" + password_login + "'");
        ResultSet resultSet = statement.executeQuery();
        //...
    }
}
```

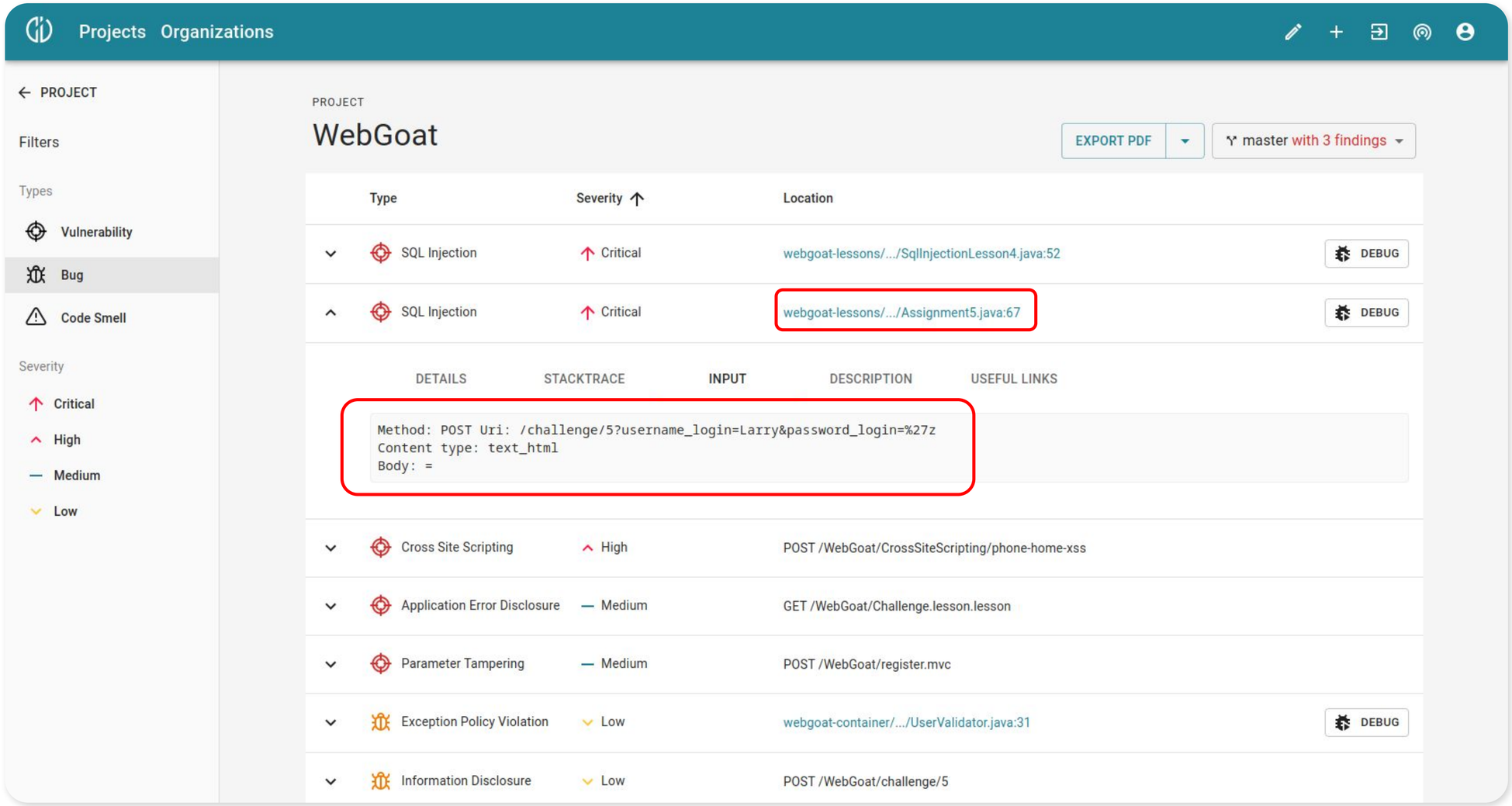
Intercept calls at runtime and perform checks

Challenge 3: How to handle checks?

```
@RestController
@Slf4j
public class Assignment5 extends AssignmentEndpoint {
    //...
    @PostMapping("/challenge/5")
    @ResponseBody
    public AttackResult login(@RequestParam String username_login, @RequestParam String password_login) {
        //...
        if (!"Larry".equals(username_login)) {
            return failed(this).feedback("user.not.larry").feedbackArgs(username_login).build();
        }
        //...
        PreparedStatement stmt = connection.prepareStatement("select password from users where userid = '" +
username_login + "' and password = '" + password_login + "'");
        ResultSet resultSet = statement.executeQuery();
        //...
    }
}
```

Solution 3: Jazzer handles this out of the box

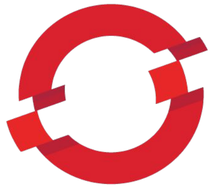
Result: Effective Testing and Detailed Bug Reporting



Challenge 4: Complex Setup & Deployment

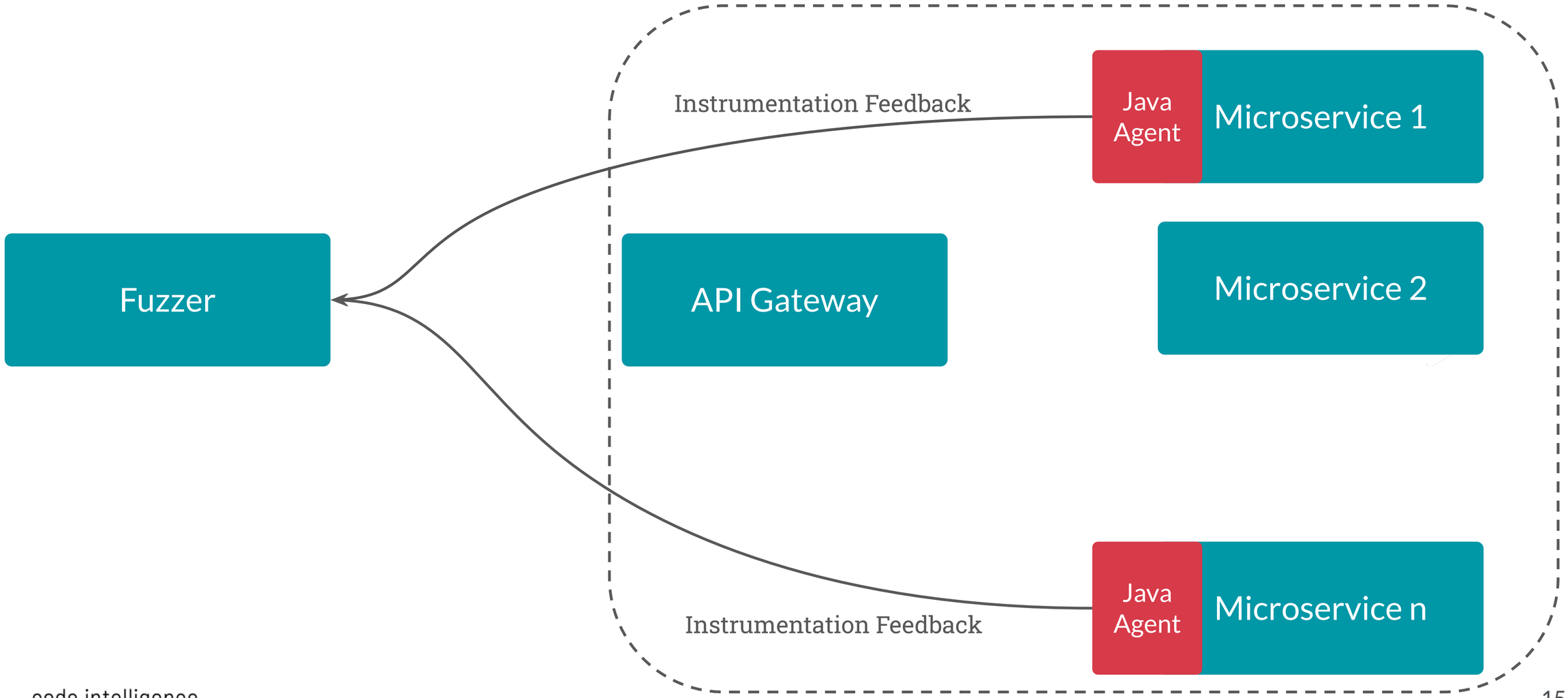
- Microservices architecture
- Complex deployments
  - Multiple processes
  - Kubernetes
  - OpenShift
  - Special protocol for internal communication

kubernetes

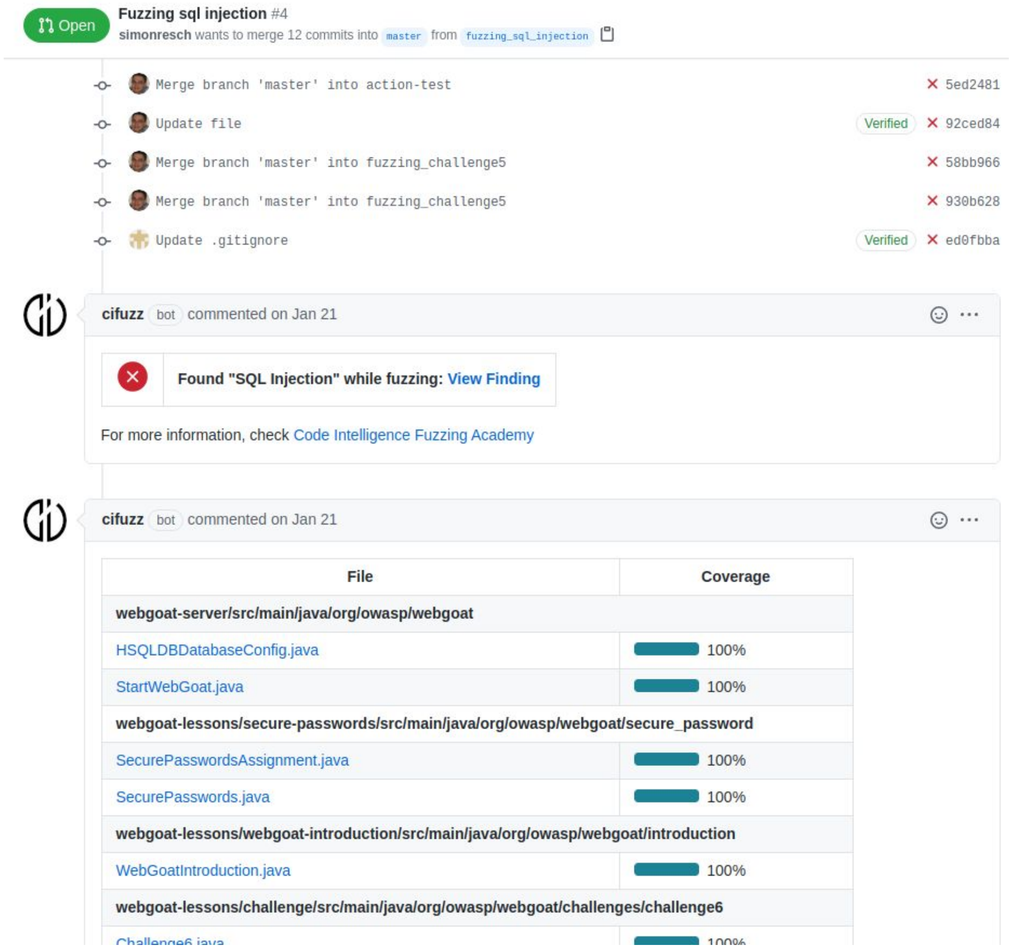


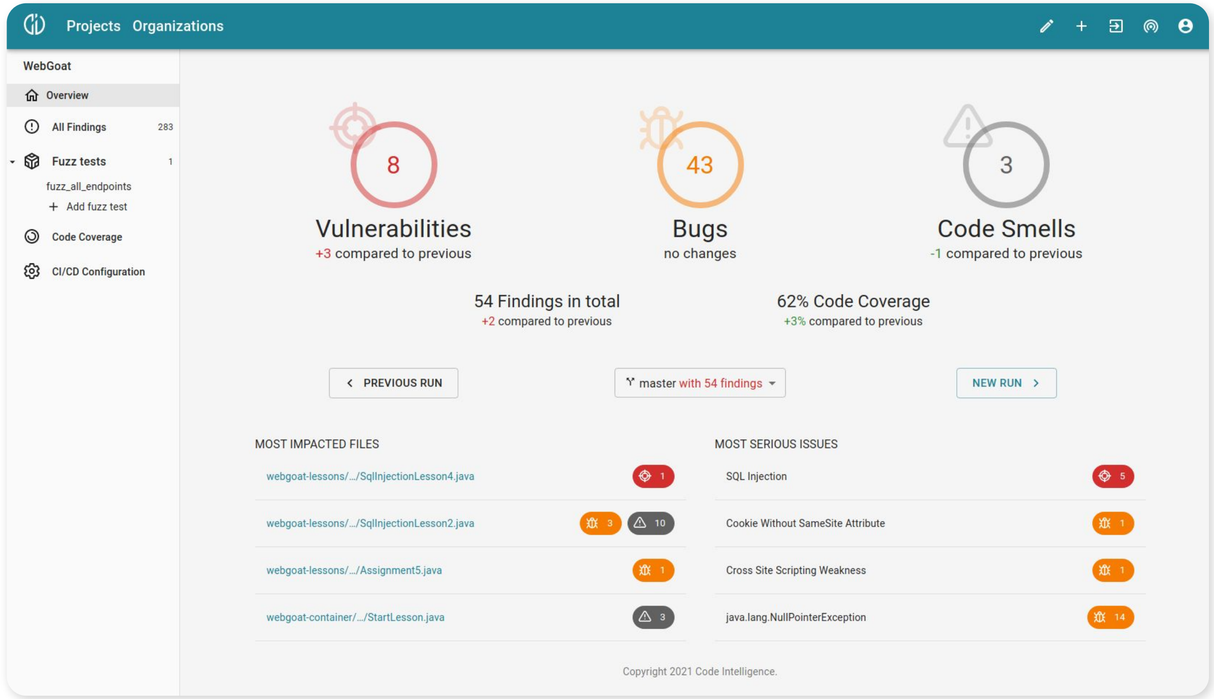
OPENSHIFT

Solution 4: How to handle complex deployments



Fuzzing most Effective when Performed Continuously!





Takeaway: Instrumented Fuzzing Improves Web Testing Immensely

- Higher code coverage
- Improved bug detection
- Perfect fit for CI/CD integration