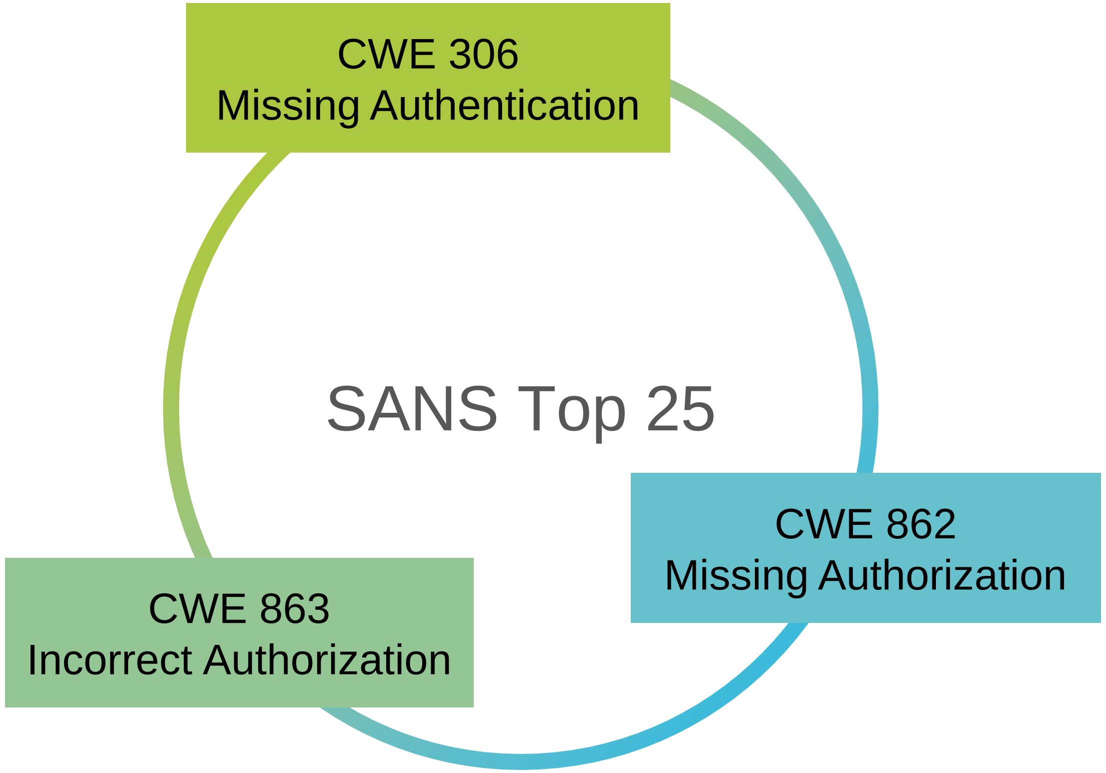
AUTHCHECK: PROGRAM-STATE ANALYSIS FOR ACCESS-CONTROL VULNERABILITIES

10th Workshop on Tool for Automatic Program Analysis (TAPAS)

Goran Piskachev, Tobias Petrasch, Johannes Späth, Eric Bodden

8. October 2019, Porto

Access-control vulnerabilities are still highly relevant according to security rankings

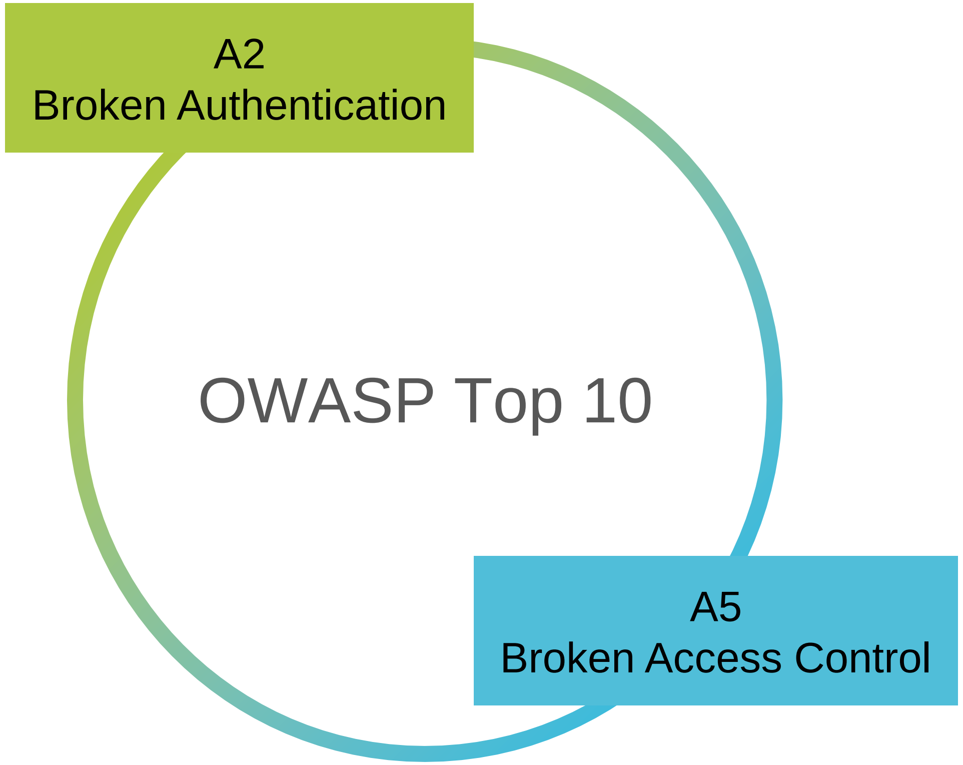


SANS Top 25

CWE 306
Missing Authentication

CWE 862
Missing Authorization

CWE 863
Incorrect Authorization



OWASP Top 10

A2
Broken Authentication

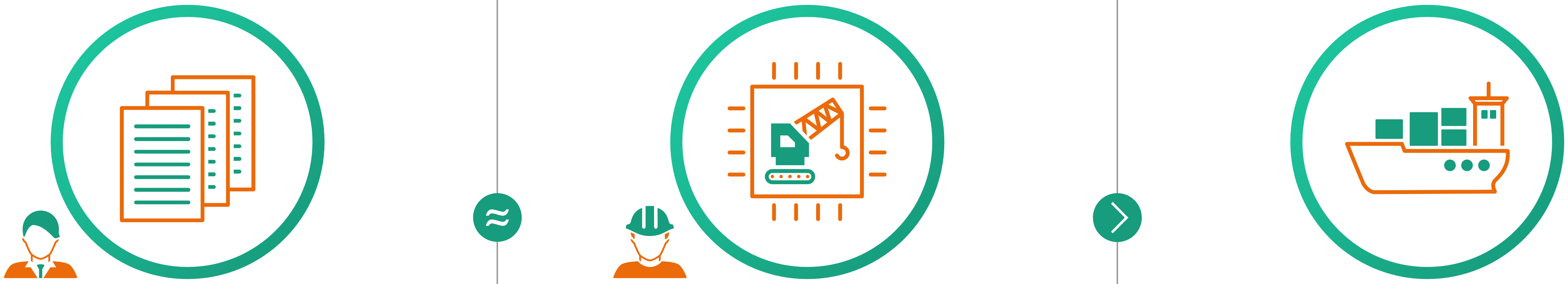
A5
Broken Access Control

Differences in specification and implementation can lead to shipment of vulnerable products

Specification

Implementation

Shipment



Requirements engineer writes specification and hands over to software engineer

Software engineer receives specification, but misinterprets it while implementing (e.g. bug)

Leads to potential vulnerable product shipment as specification is not followed

Specification of example *ToDo* application provides four resources

Illustrative

Specification

Requirements engineer writes specification and hands over to software engineer

Example

HTTP	URI	Resource	Description	Access rule
GET	/version	version()	Return's applications version	No rule
GET	/profile	profile()	Returns user profile	Authenticated
GET	/task	retrieveAll()	Returns list of all tasks	*USER* or *ADMIN*
POST	/task	create()	Create new task	*ADMIN*

Focus of next slide

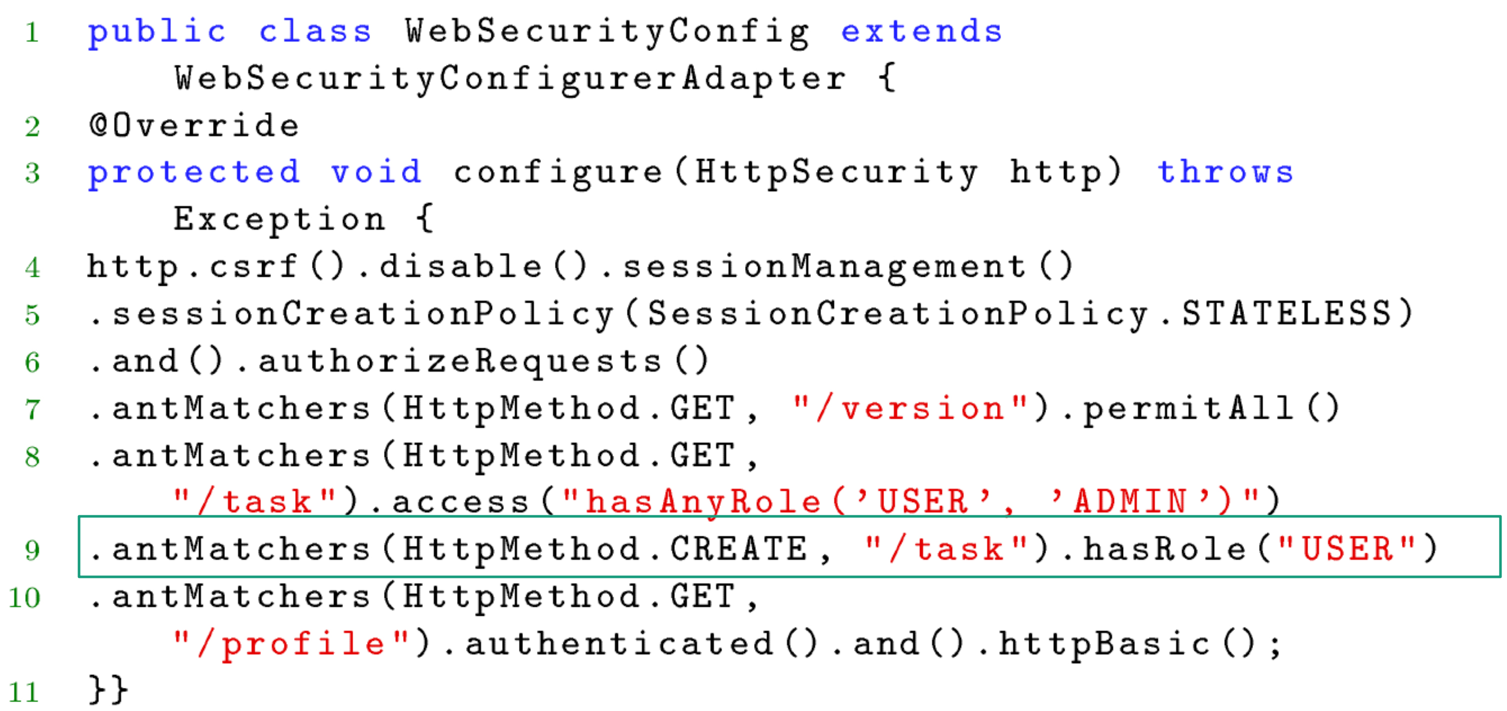
Implementation of example *ToDo* application has one error

Illustrative

Implementation

Software engineer receives specification, but not following it while implementing (e.g. bug)

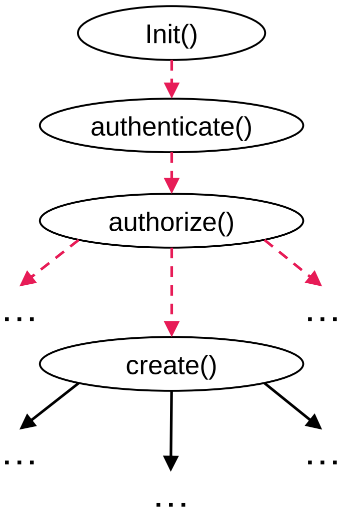
Example



```
public class WebSecurityConfig extends
    WebSecurityConfigurerAdapter {
@Override
protected void configure(HttpSecurity http) throws
    Exception {
http.csrf().disable().sessionManagement()
.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
.and().authorizeRequests()
.antMatchers(HttpMethod.GET, "/version").permitAll()
.antMatchers(HttpMethod.GET,
    "/task").access("hasAnyRole('USER', 'ADMIN')")
.antMatchers(HttpMethod.CREATE, "/task").hasRole("USER")
.antMatchers(HttpMethod.GET,
    "/profile").authenticated().and().httpBasic();
}}
```

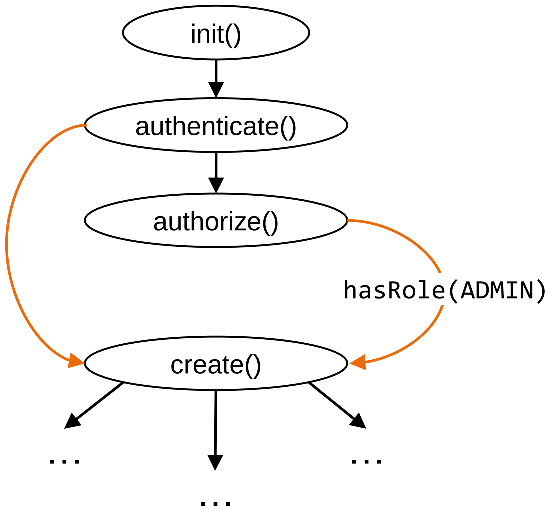
AuthCheck prevents shipping vulnerable products by running through three phases

Callgraph Construction



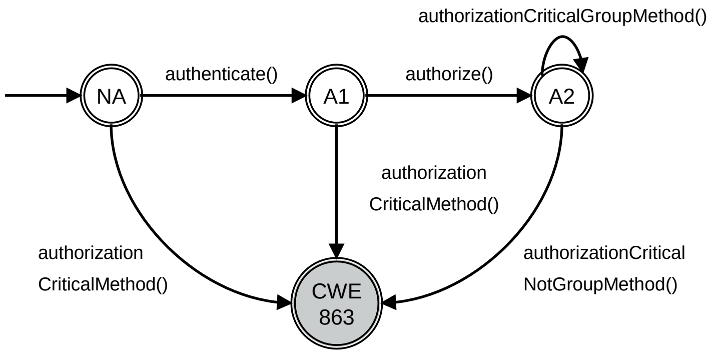
Generates callgraph and abstracts authentication and authorization related methods

Callgraph Extension



Creates edges between abstracted methods based on inter-procedural analysis

Program-State Analysis


authorizationCriticalGroupMethod()
NA
authenticate()
A1
authorize()
A2
authorization CriticalMethod()
authorization CriticalMethod()
authorizationCritical NotGroupMethod()
CWE 863

Checks generated callgraph with finite state machines to detect vulnerabilities

Callgraph construction phase generates callgraph with CHA and abstracts authentication and authorization related methods

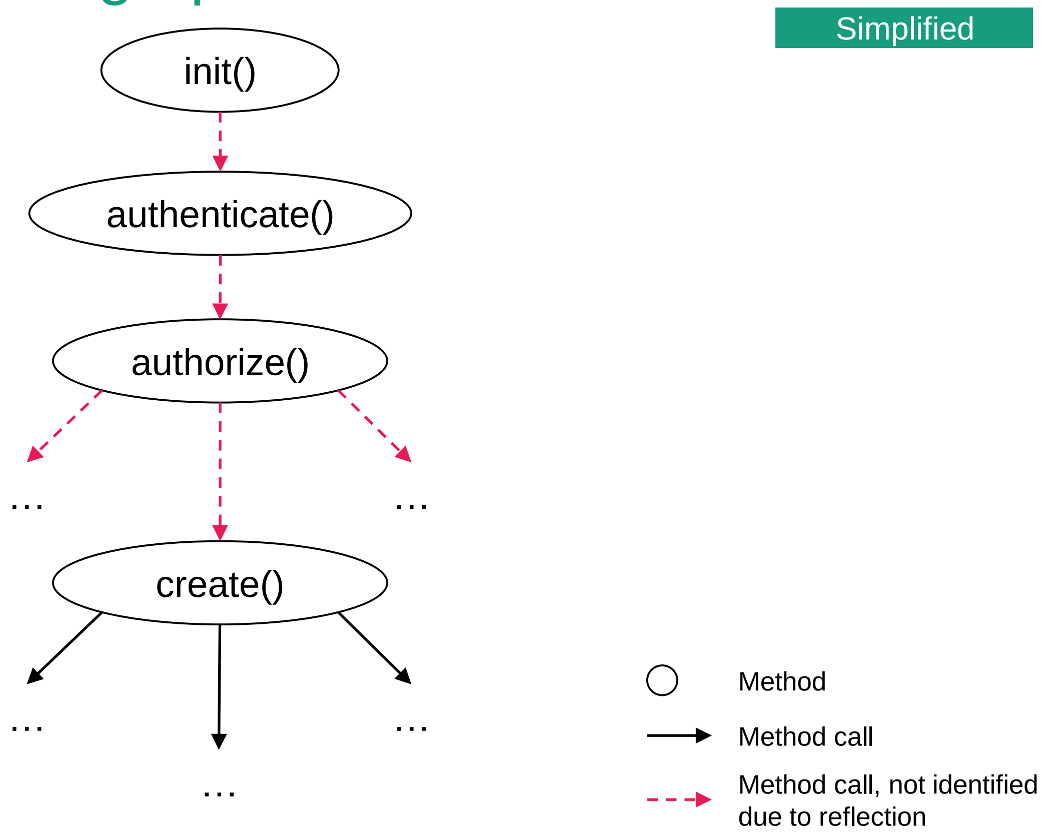
Problems

- Missing edges due to reflection
- High complexity of authentication and authorization mechanisms in Spring

Steps

1. Abstract authentication and authorization related methods based on Spring lifecycle
2. Use controller methods as entry points for call graph generation
3. Generate call graph using CHA Algorithm

Callgraph



Callgraph extension phase creates edges between abstracted methods based on inter-procedural analysis and lifecycle of Spring framework

Problems

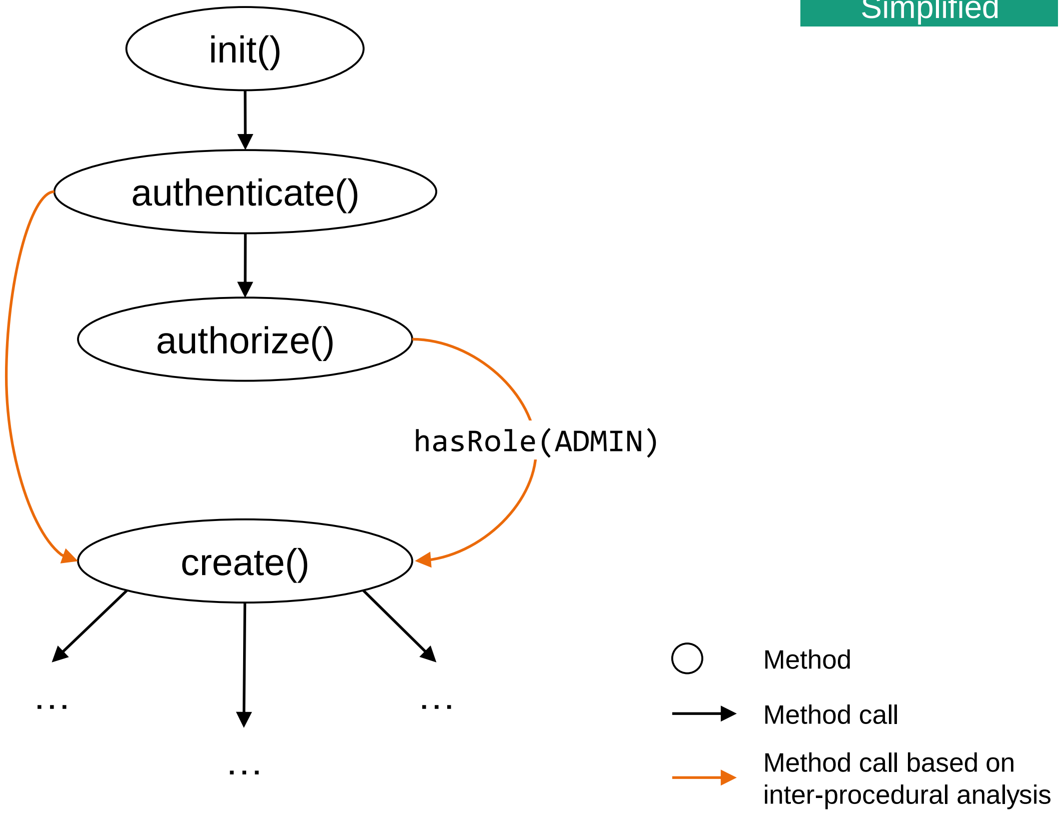
- Missing edges due to reflection
- Annotations of needed access rights missing

Steps

1. Add edges between init(), authenticate() and authorize()
2. Extract configuration method of Sprint with inter-procedural analysis
3. Add edges between authenticate() or authorize() and controller methods

Callgraph

Simplified



Program state analysis phase uses finite state machines to check for specific vulnerabilities

Example CWE 863

Problems

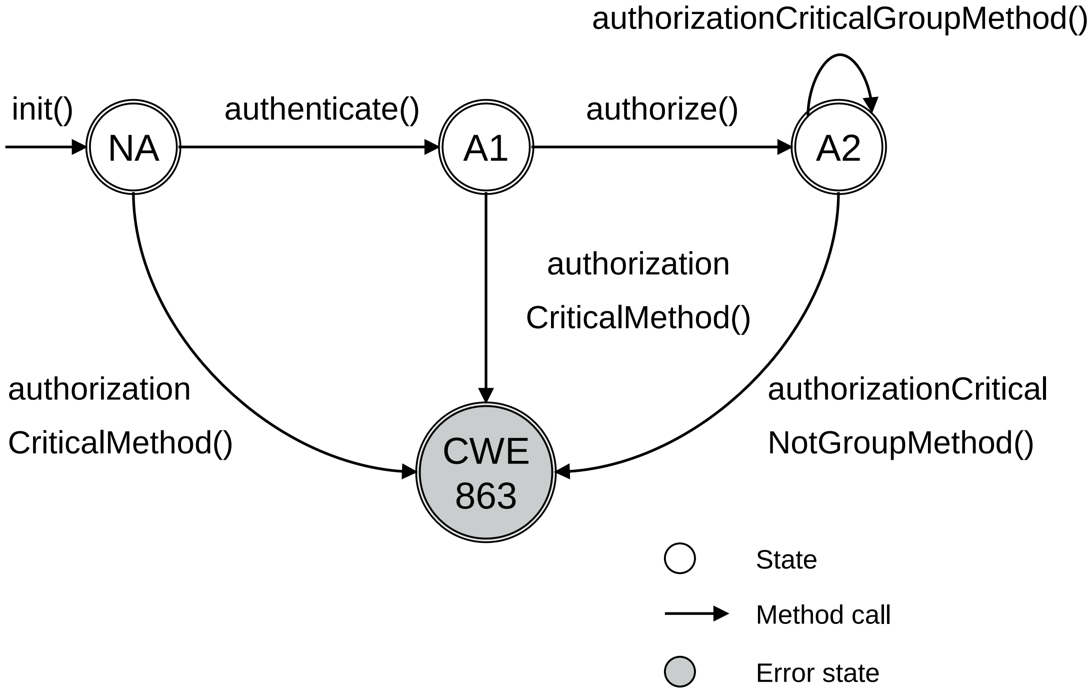
- High complexity in classification of methods

Steps

1. Generate all possible paths with DFS algorithm
2. For each path run state machine and check for error states
3. Collect errors into report

State Machine

Illustrative


authorizationCriticalGroupMethod()
init()
NA
authenticate()
A1
authorize()
A2
authorization CriticalMethod()
authorization CriticalMethod()
authorizationCritical NotGroupMethod()
CWE 863
○ State
→ Method call
● Error state

Classfication as *authorization critical and group belonging* is done by evaluating truthtables

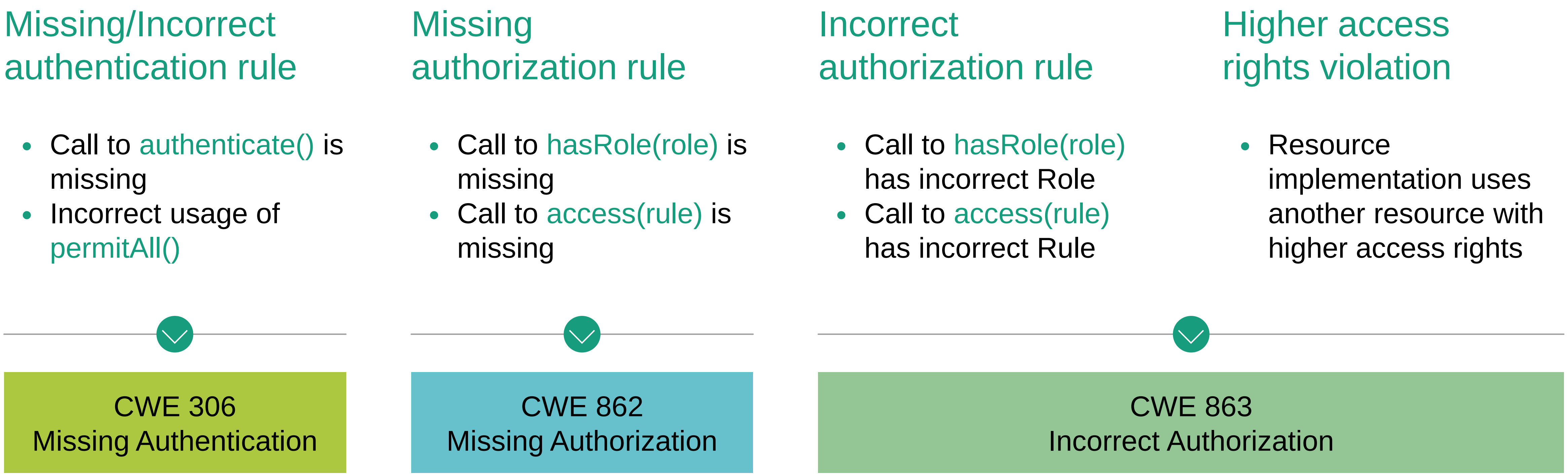
Truthtable		Specification	Program

hasRole(*ADMIN*)	hasRole(*USER*)	hasRole(*ADMIN*)
		or
		hasRole(*USER*)
			hasRole(*ADMIN*)

0	0	0	0
Error 0	1	1	0
1	0	1	1
1	1	1	1

Algorithm complexity is **$O(2^{|G|})$**

AuthCheck can detect four implementation errors that cause access-control vulnerabilities



Missing/Incorrect authentication rule

- Call to authenticate() is missing
- Incorrect usage of permitAll()

Missing authorization rule

- Call to hasRole(role) is missing
- Call to access(rule) is missing

Incorrect authorization rule

- Call to hasRole(role) has incorrect Role
- Call to access(rule) has incorrect Rule

Higher access rights violation

- Resource implementation uses another resource with higher access rights

CWE 306
Missing Authentication

CWE 862
Missing Authorization

CWE 863
Incorrect Authorization

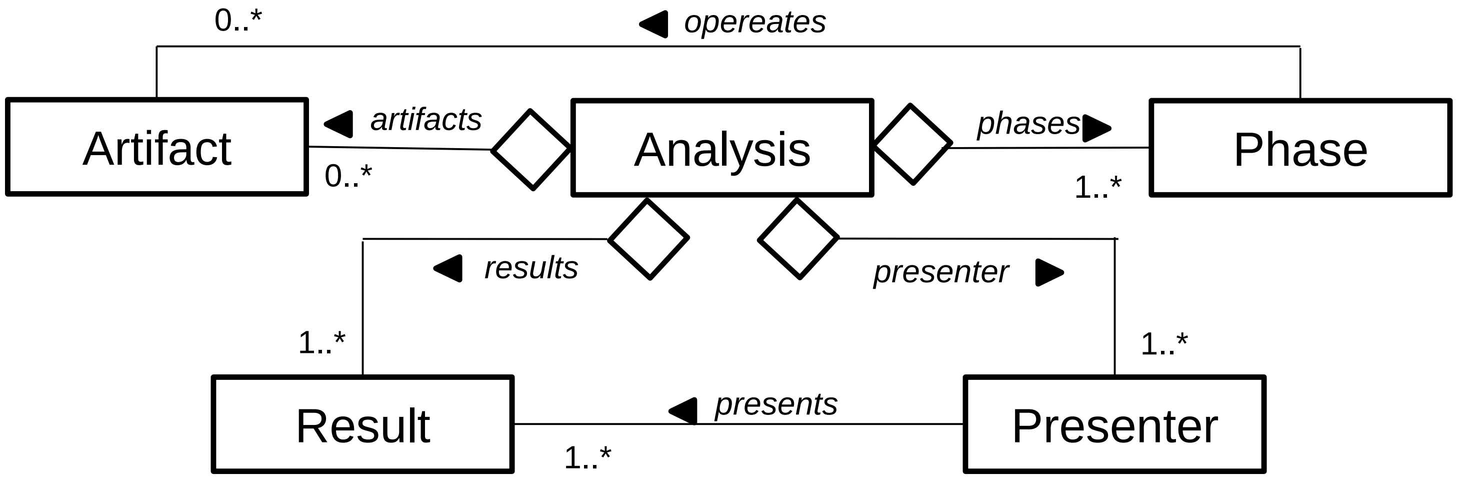
TODO example implemented as Spring Application with minimal code to demonstrate the four types of implementation errors

Implementation is based on flexible pipeline architecture that can be parallelized in the future

Summary

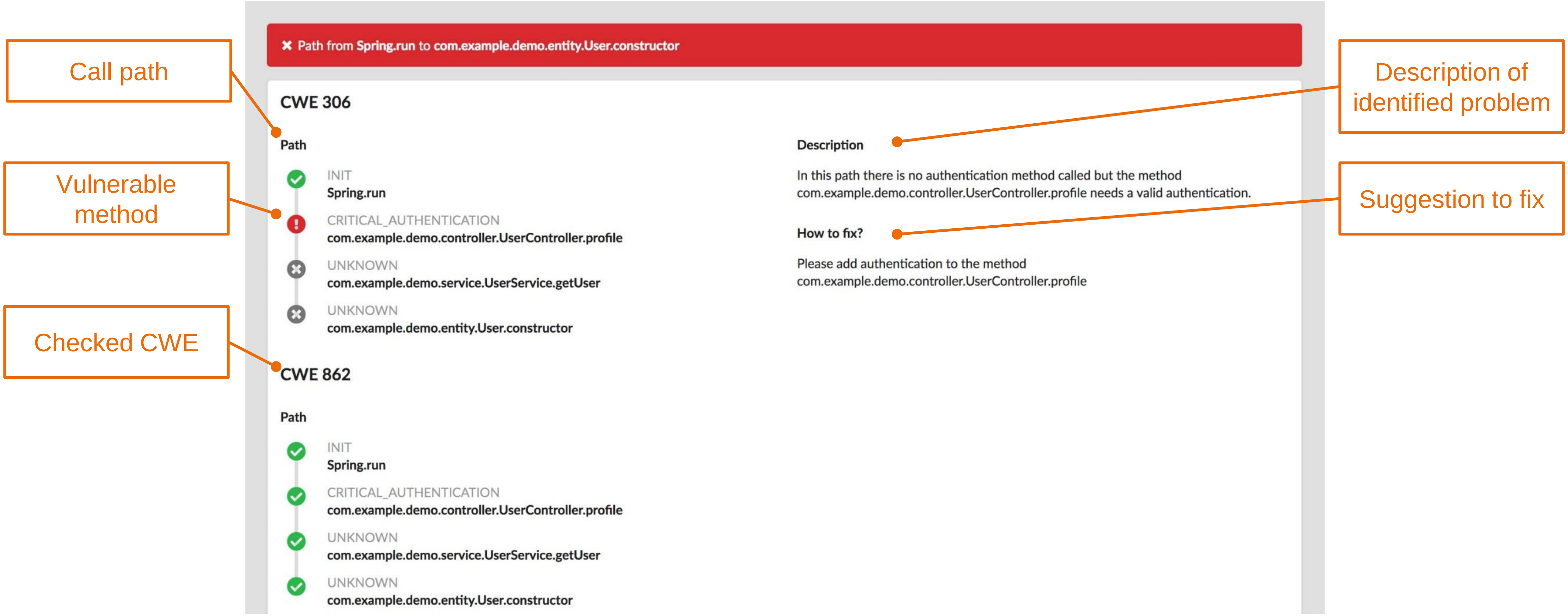
- Based on pipeline architecture
- Runs phases sequentially
- Uses artifacts as input and output
- Can be parallelized in the future

Architecture Constructs



Vulnerability report is generated as result of AuthCheck’s analysis and features visualizations of call paths and vulnerable methods

Illustrative



✖ Path from **Spring.run** to **com.example.demo.entity.User.constructor**

CWE 306

Path

✓ INIT
Spring.run

! CRITICAL_AUTHENTICATION
com.example.demo.controller.UserController.profile

✖ UNKNOWN
com.example.demo.service.UserService.getUser

✖ UNKNOWN
com.example.demo.entity.User.constructor

Description

In this path there is no authentication method called but the method com.example.demo.controller.UserController.profile needs a valid authentication.

How to fix?

Please add authentication to the method com.example.demo.controller.UserController.profile

CWE 862

Path

✓ INIT
Spring.run

✓ CRITICAL_AUTHENTICATION
com.example.demo.controller.UserController.profile

✓ UNKNOWN
com.example.demo.service.UserService.getUser

✓ UNKNOWN
com.example.demo.entity.User.constructor

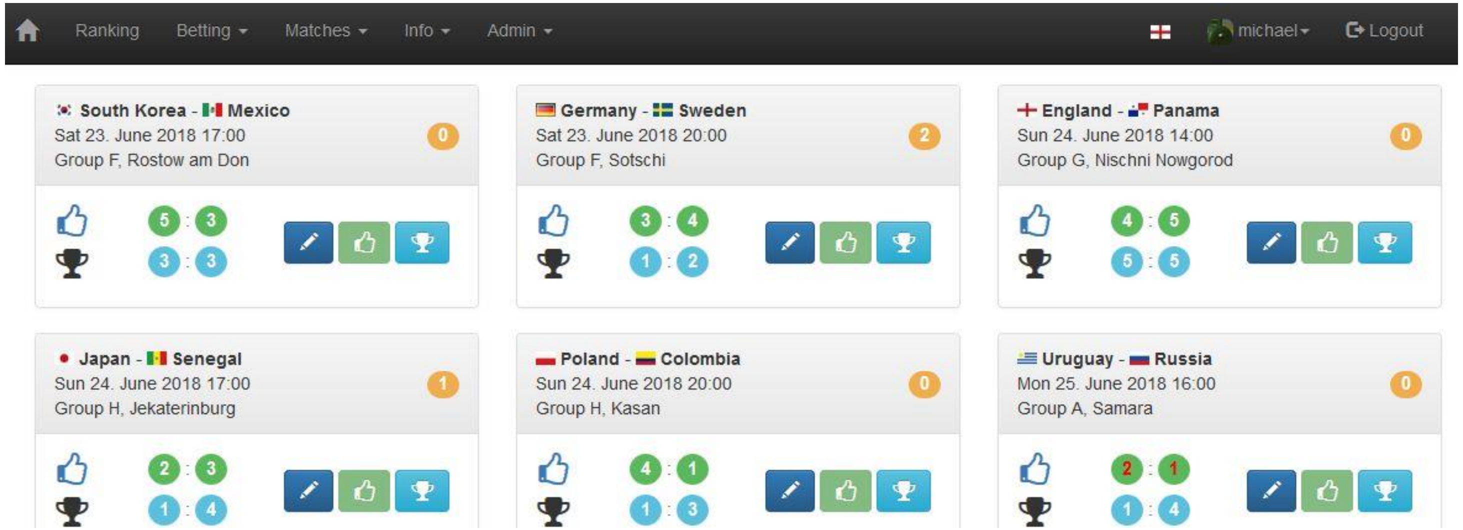
Evaluation is based on analysis of real-world example FredBet by using test catalogue

Application Facts

- Betting System
- Spring Boot, Bootstrap and Spring Security
- Open Source
- Actively maintained

- 22 Controllers
- 37 Resources
- 4 Groups with 28 permission types

Screenshot



South Korea - Mexico
Sat 23. June 2018 17:00
Group F, Rostow am Don

Germany - Sweden
Sat 23. June 2018 20:00
Group F, Sotschi

England - Panama
Sun 24. June 2018 14:00
Group G, Nischni Nowgorod

Japan - Senegal
Sun 24. June 2018 17:00
Group H, Jekaterinburg

Poland - Colombia
Sun 24. June 2018 20:00
Group H, Kasan

Uruguay - Russia
Mon 25. June 2018 16:00
Group A, Samara

1. https://github.com/fred4jupiter/fredbet

AuthCheck is evaluated by deriving specification from implementation and introducing four types of errors

Create Specification

Derive specification from implementation

Introduce Errors

Introduce 5 errors of all four types by twisting specification in 5 controllers

All errors detected

Experiments on 8 GB RAM and i5-6200U CPU (2,3 GHz)

Total analysis time for all controllers

- w/ errors: 68,874 sec
- w/o errors: 71,953 sec

Create specification

AuthCheck reads specification models in a simple JSON format that features groups, resources and access control rules

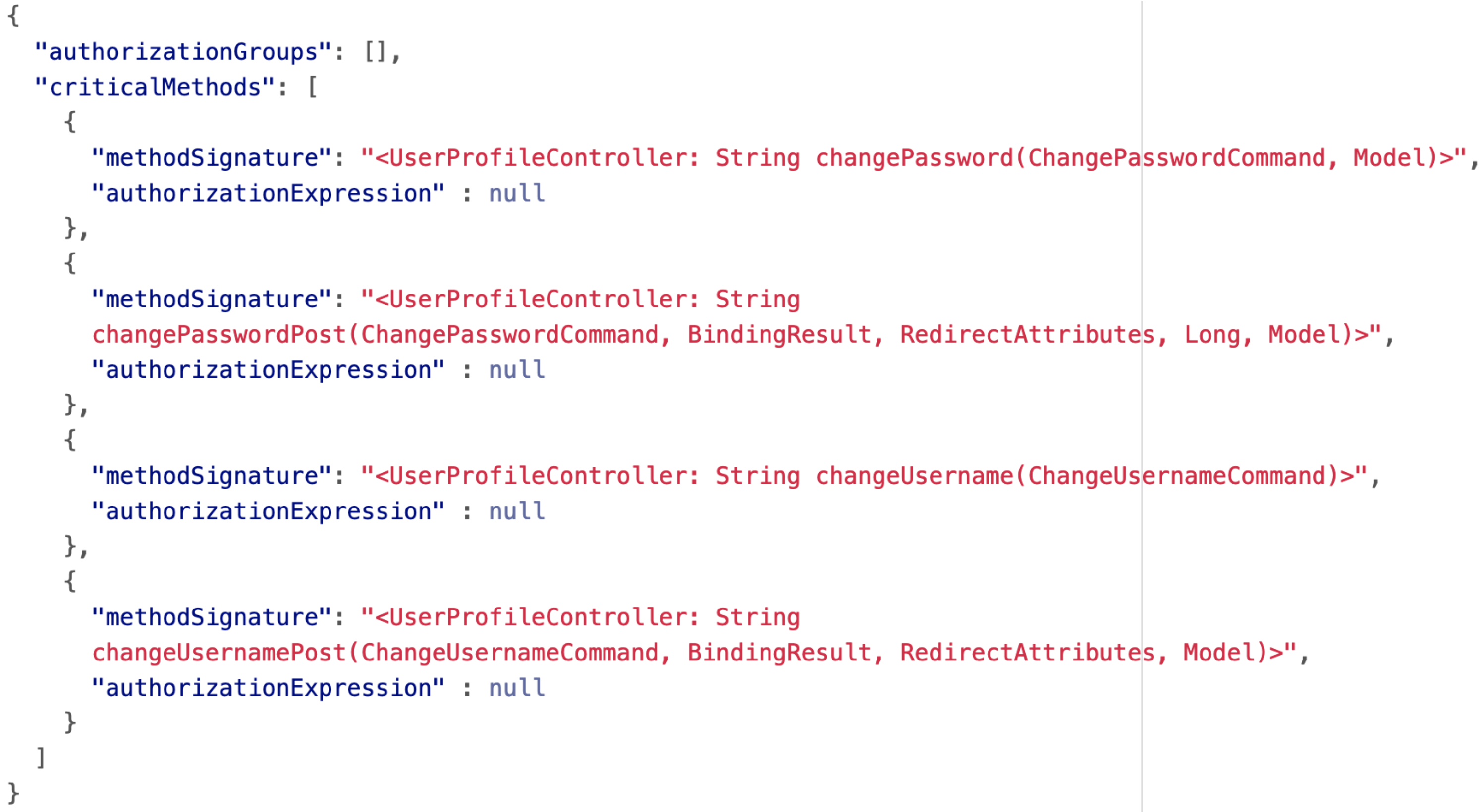
Controllers

- AdminController
- BetContoller
- ConfigurationController
- CreateEditMatchController
- DatabaseBackupController
- ExcelExportController
- ExcelImportController
- ExtraBetController
- HomeController
- ImageCroppingController
- ImageGalleryController
- ImageGroupController
- ImageUploadController
- InfoController
- **MatchController (ROOT)**
- MatchResultController
- PointsFrequencyController
- RankingController
- RuntimeConfigurationController
- SystemInfoController
- UserController
- **UserProfileController**

Example

Simplified

UserProfileController linked from MatchController



```
{
  "authorizationGroups": [],
  "criticalMethods": [
    {
      "methodSignature": "<UserProfileController: String changePassword(ChangePasswordCommand, Model)>",
      "authorizationExpression" : null
    },
    {
      "methodSignature": "<UserProfileController: String
      changePasswordPost(ChangePasswordCommand, BindingResult, RedirectAttributes, Long, Model)>",
      "authorizationExpression" : null
    },
    {
      "methodSignature": "<UserProfileController: String changeUsername(ChangeUsernameCommand)>",
      "authorizationExpression" : null
    },
    {
      "methodSignature": "<UserProfileController: String
      changeUsernamePost(ChangeUsernameCommand, BindingResult, RedirectAttributes, Model)>",
      "authorizationExpression" : null
    }
  ]
}
```

Errors are purposely introduced by making changes in the specification models

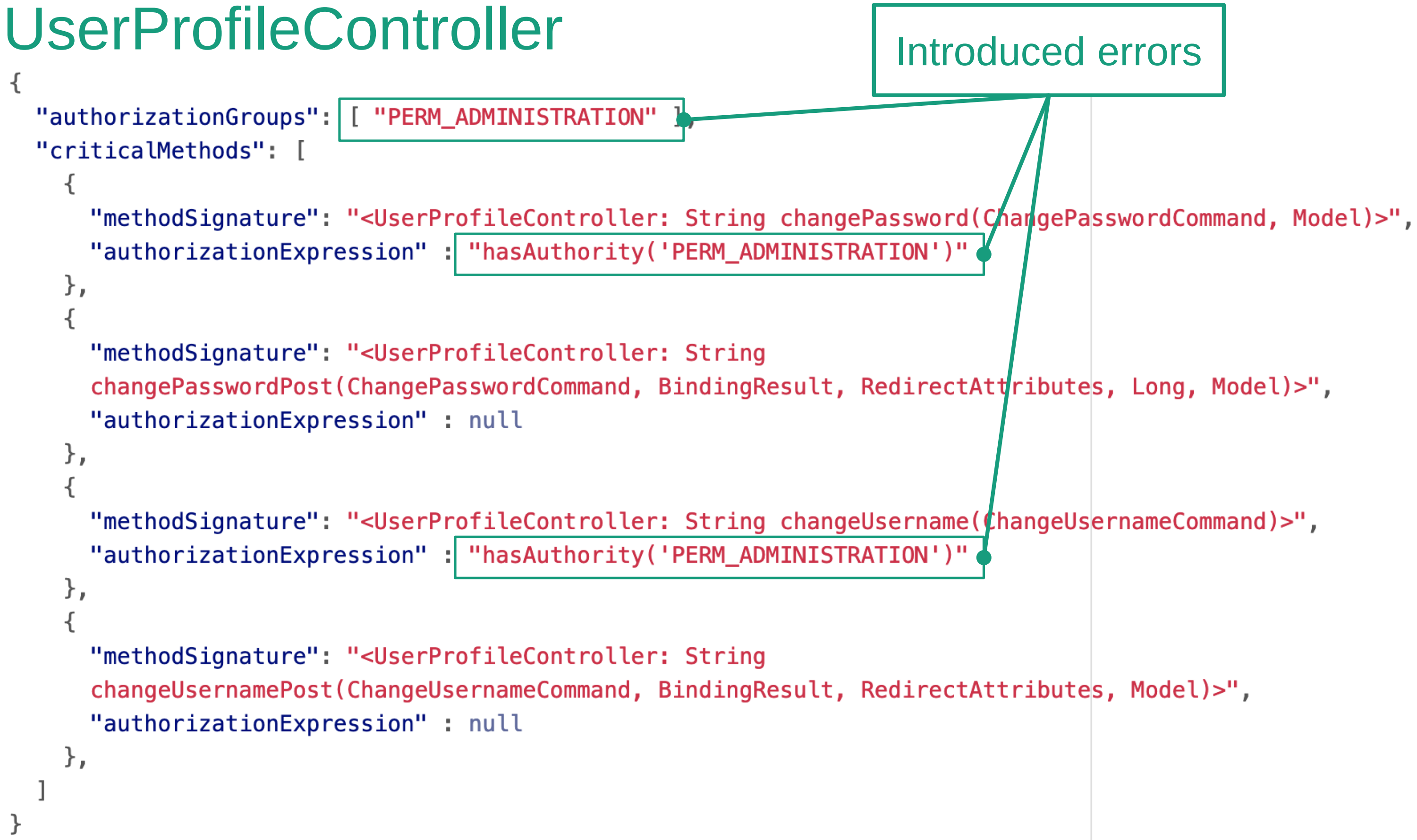
Making changes

- UserProfileController is linked from MatchController
- Adding permission for changing password and changing username
- Example for CWE 862 (Missing authorization)

Example

Simplified

UserProfileController



Introduced errors

```
{
  "authorizationGroups": [ "PERM_ADMINISTRATION" ],
  "criticalMethods": [
    {
      "methodSignature": "<UserProfileController: String changePassword(ChangePasswordCommand, Model)>",
      "authorizationExpression" : "hasAuthority('PERM_ADMINISTRATION')"
    },
    {
      "methodSignature": "<UserProfileController: String
      changePasswordPost(ChangePasswordCommand, BindingResult, RedirectAttributes, Long, Model)>",
      "authorizationExpression" : null
    },
    {
      "methodSignature": "<UserProfileController: String changeUsername(ChangeUsernameCommand)>",
      "authorizationExpression" : "hasAuthority('PERM_ADMINISTRATION')"
    },
    {
      "methodSignature": "<UserProfileController: String
      changeUsernamePost(ChangeUsernameCommand, BindingResult, RedirectAttributes, Model)>",
      "authorizationExpression" : null
    },
  ]
}
```

AuthCheck’s implementation is lacking scalability and is framework-dependent; Evaluation with industry and further improvements follow

Limitations

Limited scalability

- Algorithms grow exponentially with number of groups
- In practice, number of groups is small

Framework-dependent implementation

- Callgraph generation and parsers are framework-dependent and limited to Java Spring framework

Missing specifications

- Missing specifications result in high evaluation effort
- Directly implemented by developer

Outlook

Implement SARIF output for results

- Integration with third-party tools

UX improvements

- Integration into CI/CD pipelines

Tool evaluation with industry

- Specifications and applications provided by industry to evaluate real-world examples
- More information on https://appsecure.nrw/