

# **Spectra Guide v2.2**

2023.05

# 목차

---

- **Spectra Web Guide**
- **Spectra Open API Guide**
- **Jenkins Plugin Guide**

# Spectra Web Guide

2023.05

## 0. 프로젝트 구조

Spectra Web의 경우 다음과 같은 프로젝트 구조를 지원합니다.

(1) 하나의 sol 파일

a.sol

(2) 여러 sol파일의 압축파일 (.zip 확장자만 지원)

sample.zip

└ a.sol

└ b.sol

└ c.sol

(3) hardhat, truffle, brownie의 표준 구조를 취하는 압축파일  
((O)는 optional을 의미합니다.)

sample.zip

└ contracts/

└ (O) node\_modules/

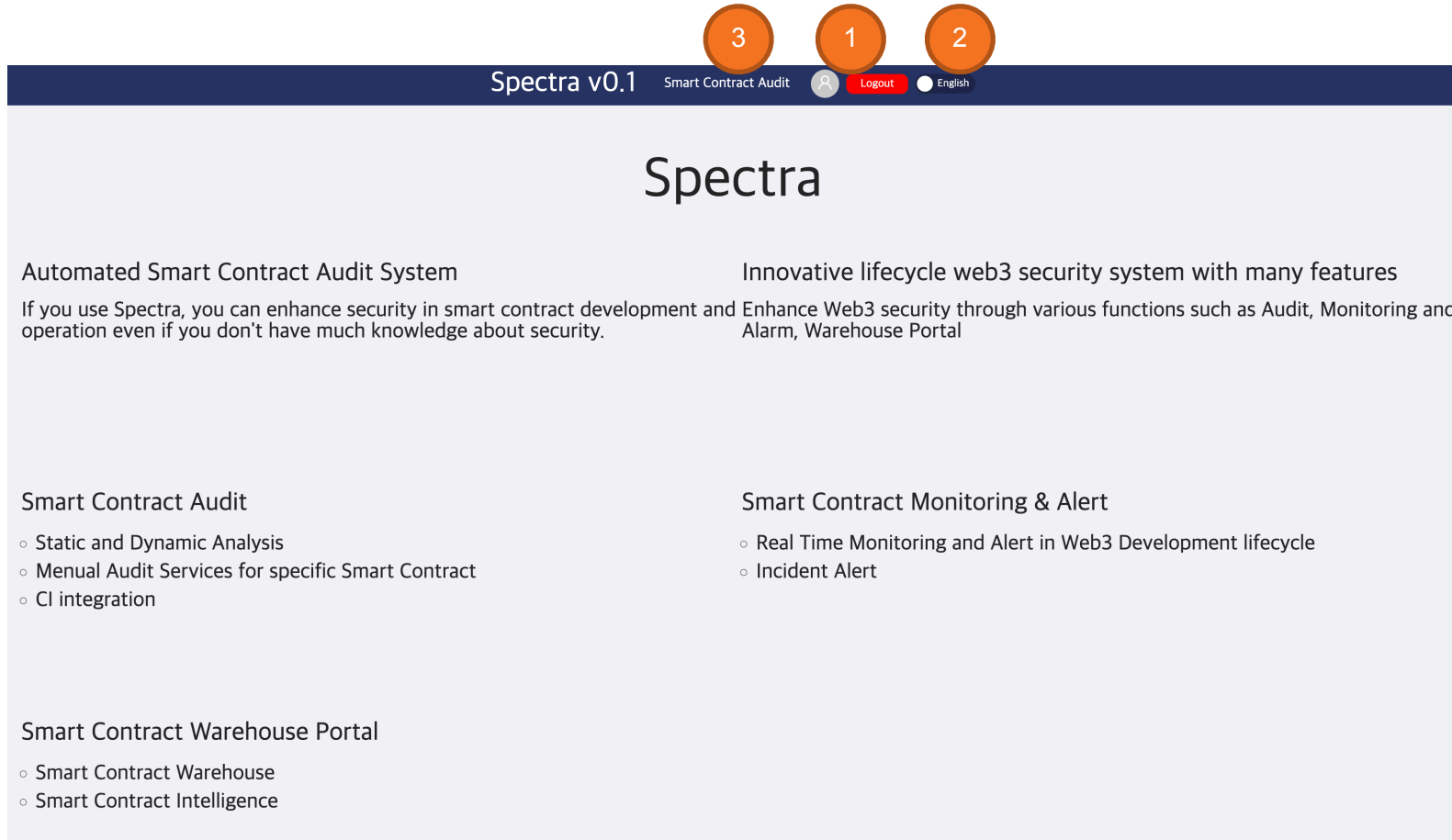
└ (O) package.json

└ (O) hardhat.config.js 혹은 truffle-config.js 혹은 brownie-config.yaml

# 1. Introduction

<https://www.spectra-space.io> 로 접속하면 Spectra 메인 페이지로 이동할 수 있습니다.

Smart Contract 파일(.sol)을 직접 업로드하여 Static Audit을 수행할 수 있고 이에 대한 Report를 확인할 수 있습니다.



## 1 로그인 버튼

Spectra 유저 로그인 페이지로 이동합니다.  
로그인 상태인 경우 로그아웃(logout)으로 표시됩니다.

## 2 한/영 전환 버튼

언어를 한글 또는 영어로 변경합니다.

## 3 Smart Contract Audit 메뉴


Smart Contract Audit 수행을 위한 페이지로 이동합니다.


## 2. Login

관리자로부터 발급받은 ID(이메일주소) 및 Password로 로그인할 수 있습니다.

신규 계정 발급은 반드시 **Admin** 관리자로부터 요청해주시기 바랍니다.

### Login



 Sign In

### 3. Smart Contract Audit


상단의 Smart Contract Audit 메뉴를 클릭하면 Audit을 수행하거나, Audit 히스토리를 조회할 수 있습니다.

**파일 업로드 버튼 :** Audit을 수행할 Smart Contract 파일(.sol 또는 컴파일 가능한 프로젝트 압축파일)을 업로드

**Audit 수행 버튼 :** 업로드한 파일을 기준으로 Audit 수행을 시작

Smart Contract Audit Upload the smart contract files(.zip) to perform the security audit.

Upload files Start Audit

Smart Contract	Audit Time	Status
 No Data		

**Audit History :** 현재 로그인한 계정의 Audit 히스토리가 보여지는 영역

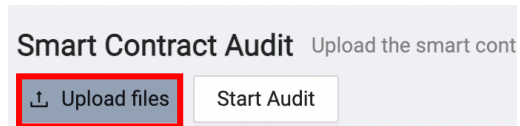
### 3. Smart Contract Audit (step 1. File upload)

Audit 을 수행할 파일을 업로드합니다.

(업로드 규칙1) 이미 업로드한 파일을 중복해서 업로드할 수 없음

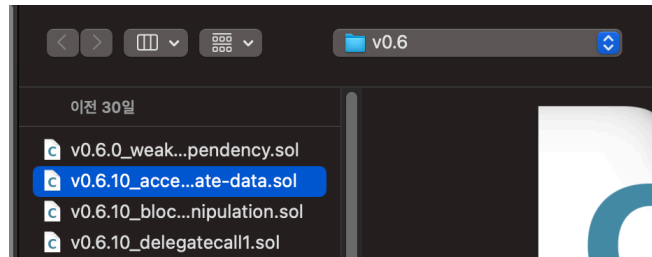
(업로드 규칙2) Audit을 수행한 파일인 경우, 다시 업로드할 수 있음

#### (1) 업로드버튼 클릭

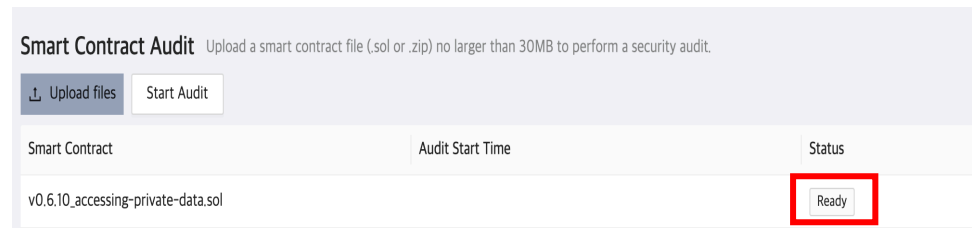


#### (2) 파일 선택

(파일 제한) .sol 또는 컴파일 가능한 프로젝트의 .zip파일  
(파일 개수) 1개 이상



#### (3) 업로드된 파일 확인 (Status : Ready)



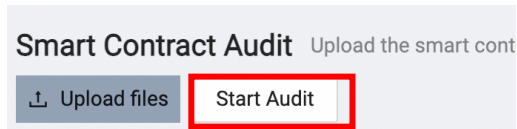


### 3. Smart Contract Audit (step 2. Audit Start)

Audit 을 수행합니다.

Audit 성공인 경우, Smart Contract 항목이 링크 표시됩니다. (컴파일 오류 발생 시, Audit 결과가 Fail로 나타날 수 있습니다.)

#### (1) Start Audit 버튼 클릭



#### (2) Audit 시작 및 완료 (10초 이내)

⬆️ Upload files

Start Audit

Smart Contract	Audit Start Time	Status
<a href="#">v0.6.10_accessing-private-data.sol</a>	2023-05-04 18:11:49	Success

### 3. Smart Contract Audit (step 3. View Audit Report)

Audit 결과가 성공인 경우, Audit 보고서를 확인할 수 있습니다.

(1) Smart Contract 링크 클릭



(2) Audit Report 화면으로 이동

Smart Contract

[v0.6.10\\_accessing-private-data.sol](#)

← Audit Status target : v0.6.10\_delegatecall2.sol

40.0%

20.0%

40.0%

Total  
5

탐지된 이슈의 심각도별 비율

Audit Start Time2023-05-04 14:20:14

Audit 요청시간

Number of issues by severity levelhigh 1medium 2info 2

탐지된 이슈의 심각도별 개수

탐지된 이슈, 클릭하여 상세 보기

> Rule ID : unrestricted-delegatecall	Severity : high	Detected Files (1): Select a file
> Rule ID : missing-low-level-call-return-check	Severity : medium	Detected Files (1): Select a file
> Rule ID : missing-zero-address-check	Severity : medium	Detected Files (1): Select a file
> Rule ID : low-level-calls	Severity : info	Detected Files (1): Select a file
> Rule ID : constable-states	Severity : info	Detected Files (1): Select a file

# 3. Smart Contract Audit (step 3. View Audit Report)

Rule ID: unrestricted-delegatecall

Severity: high

Detected Files (1): v0.6.10\_delegatecall2.sol

Lines을 클릭하여 탐지된 영역으로 바로가기

탐지된 파일 중, 조회할 파일을 선택

Lines

41 - 43

```
32     address public lib;
33     address public owner;
34     uint public someNumber;
35
36     constructor(address _lib) public {
37         lib = _lib;
38         owner = msg.sender;
39     }
40
41     function doSomething(uint _num) public {
42         lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
43     }
44 }
45
46 // contract Attack {
47 //     // Make sure the storage layout is the same as HackMe
48 //     // This will allow us to correctly update the state variables
49 //     address public lib;
50 //     address public owner;
51 //     uint public someNumber;
52 //
53 //     HackMe public hackMe;
54 //
55 //     constructor(HackMe hackMe) public {
```

탐지된 영역은 하이라이트 처리

Description 탐지된 취약점에 대한 설명

Calling into untrusted contracts is very dangerous, as the code at the target address can change any storage values of the caller and has full control over the caller's balance.

Countermeasure 탐지된 취약점에 대한 recommendation

1. Control what is executed with delegatecall  
: Use permissions or authentication or some other form of control for specifying or changing what functions and contracts are executed with delegatecall.  
2. Only call delegatecall on addresses that have code  
: check address  
==>  
function isContract (address account) internal view returns (bool) {  
 uint size;  
 assemble { size := extcodesize(account) }  
 return size > 0;  
}

### 3. Smart Contract Audit (step 4. Delete audit history)

Audit 내역을 삭제할 수 있습니다

삭제 버튼 클릭

Smart Contract	Audit Start Time	Status	
<a href="#">v0.6.10_accessing-private-data.sol</a>	2023-05-04 18:11:49	<a href="#">Success</a>	<a href="#">삭제</a>
<a href="#">v0.6.10_delegatecall2.sol</a>	2023-05-04 14:20:14	<a href="#">Success</a>	<a href="#">삭제</a>
<a href="#">v0.6.10_front-running.sol</a>	2023-05-04 11:32:07	<a href="#">Success</a>	<a href="#">삭제</a>

# Spectra Open API Guide

2023.05

## 0. 프로젝트 구조

Spectra Open API의 경우 다음과 같은 프로젝트 구조를 지원합니다.

(1) 하나의 sol 파일

a.sol

(2) 여러 sol파일의 압축파일 (.zip 확장자만 지원)

sample.zip

└ a.sol

└ b.sol

└ c.sol

(3) hardhat, truffle, brownie의 표준 구조를 취하는 압축파일  
((O)는 optional을 의미합니다.)

sample.zip

└ contracts/

└ (O) node\_modules/

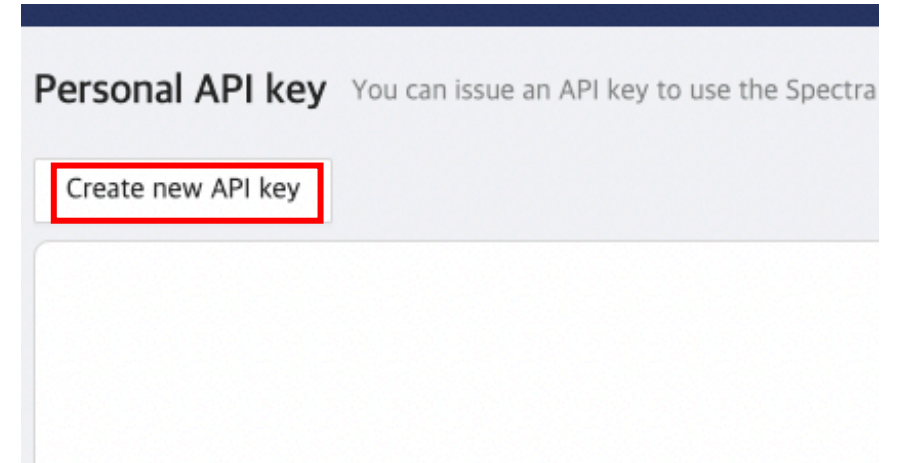
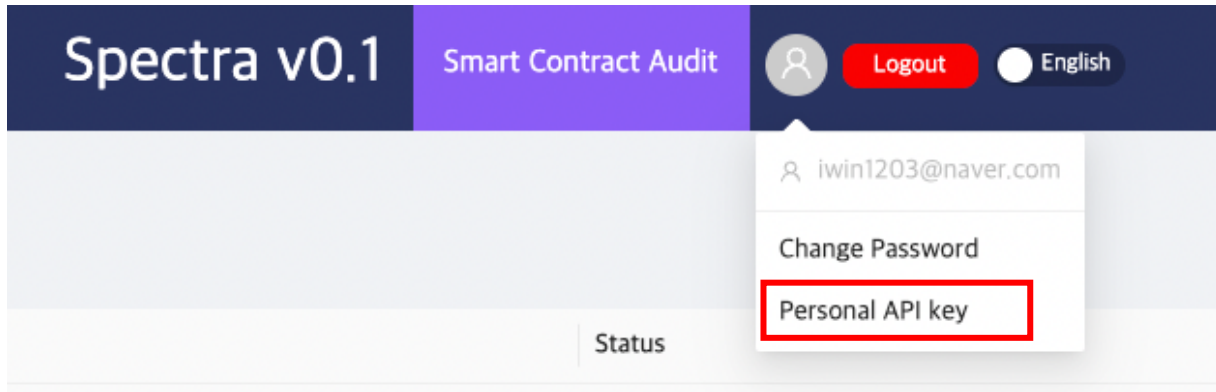
└ (O) package.json

└ (O) hardhat.config.js 혹은 truffle-config.js 혹은 brownie-config.yaml

# 1. Setup

Spectra는 CI/CD 툴 Jenkins의 Open API를 제공합니다.

Spectra Web에서 프로필 아이콘 클릭 > Personal API key > Create new API key를 클릭하여 api key 발급 창으로 이동합니다.



# 1. Setup

Spectra는 CI/CD 툴 Jenkins의 Open API를 제공합니다.

Expiration과 role을 선택하여 API key를 발급받습니다.

Create New API key

Key 만료 기한

\* Expiration : 90 days Select date

\* roles : ☒ request audit ☒ view audit

권한: audit 요청 권한 / 결과 조회

Cancel

OK



Personal API key You can issue an API key to use the Spectra service.

Create new API key

발급된 Key

5f127d95-244e-42d7-933c-5b3b1cc7a747

roles : view-audit,request-audit created : 2023-05-16 expiration date : 2023-07-15



## 2. API Specification

방식: http/REST

엔드포인트: [POST] <https://api.spectra-space.io/v1/request>

**Request Field:**

header

X-Apikey: <api key>  
Content-Type: multipart/form-data

KEY	VALUE
X-Apikey	<api key>
Content-Type	multipart/form-data;

body (multipart/form-data)

file : <file to audit>

KEY	VALUE
file	Select Files

## 2. API Specification

---

### Response

401 Unauthorized: API Key가 올바르지 않거나 만료된 경우

404 Bad Request: Request Body가 잘못된 경우

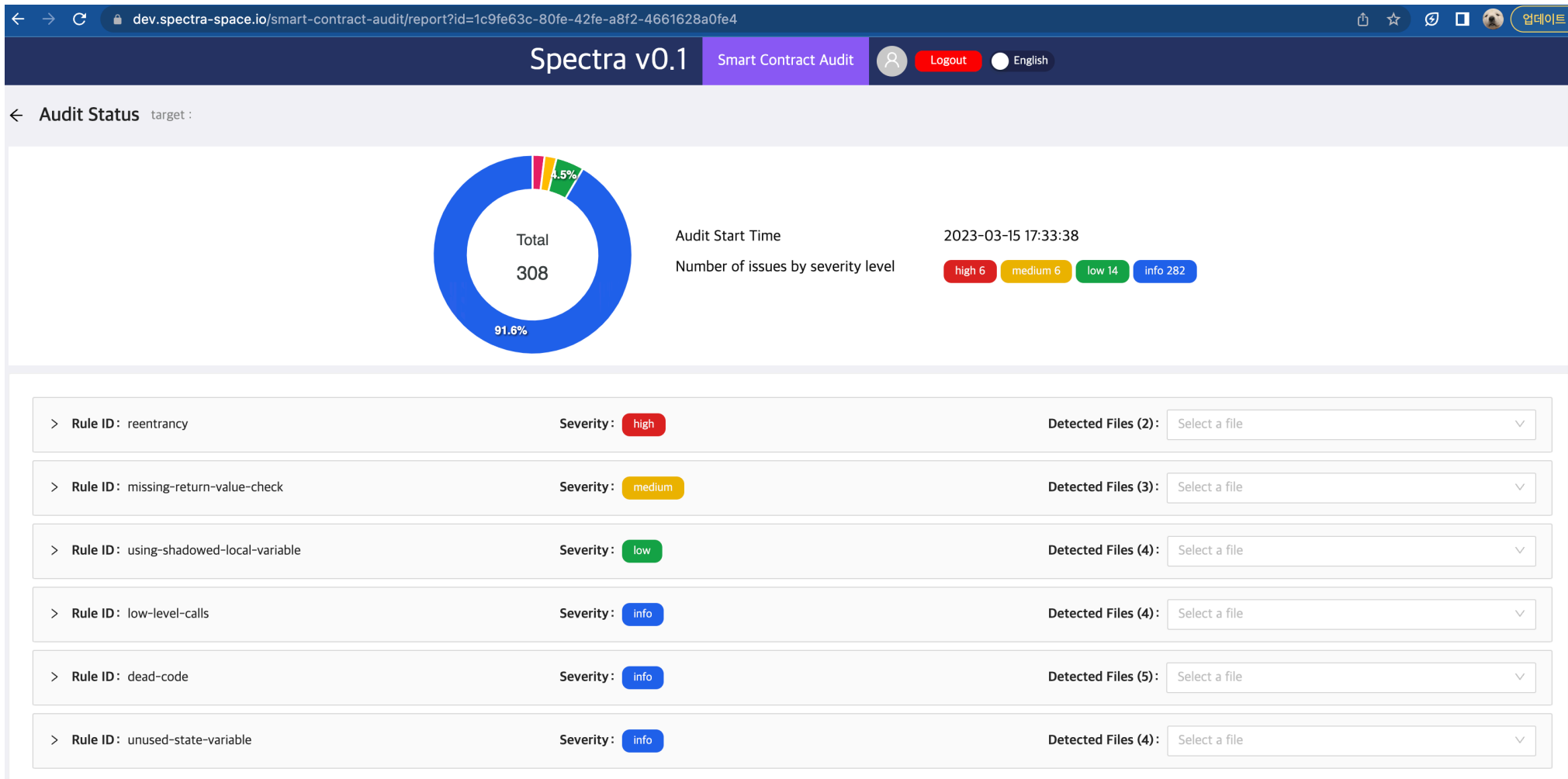
200 Ok: 분석이 성공적으로 진행된 경우.

### Result

‘200 Ok’가 리턴되었다면, Spectra Web에서 설정한 이메일로 링크가 전송됩니다. 해당 링크를 클릭하면 audit 결과를 확인할 수 있습니다.

### 3. 결과 확인하기

빌드에 성공한 경우, Spectra Web에서 설정한 이메일로 결과 링크가 주어집니다. 해당 링크를 클릭하여 결과를 볼 수 있습니다.



# Spectra Jenkins Plugin Guide

2023.05

## 0. 프로젝트 구조

Spectra Jenkins Plugin의 경우, 다음과 같은 프로젝트 구조를 지원합니다.

(1) 한 개 이상의 sol 파일

```
a.sol  
b.sol  
c.sol
```

(2) hardhat, truffle, brownie의 표준 구조

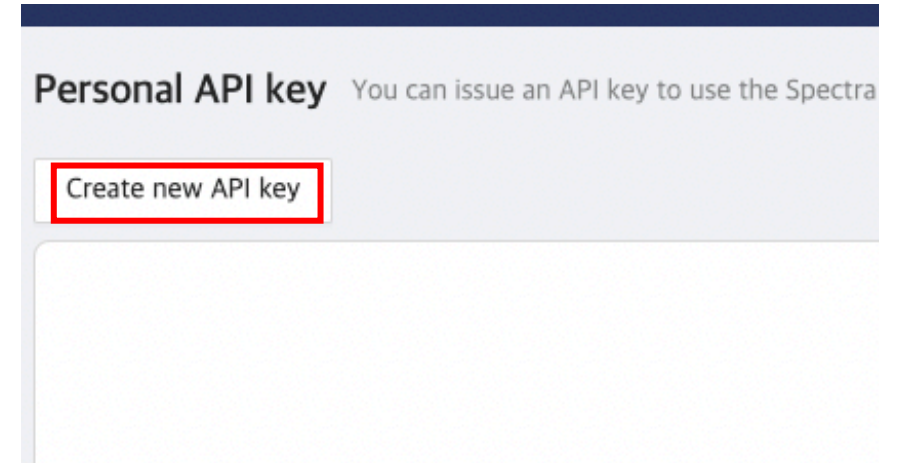
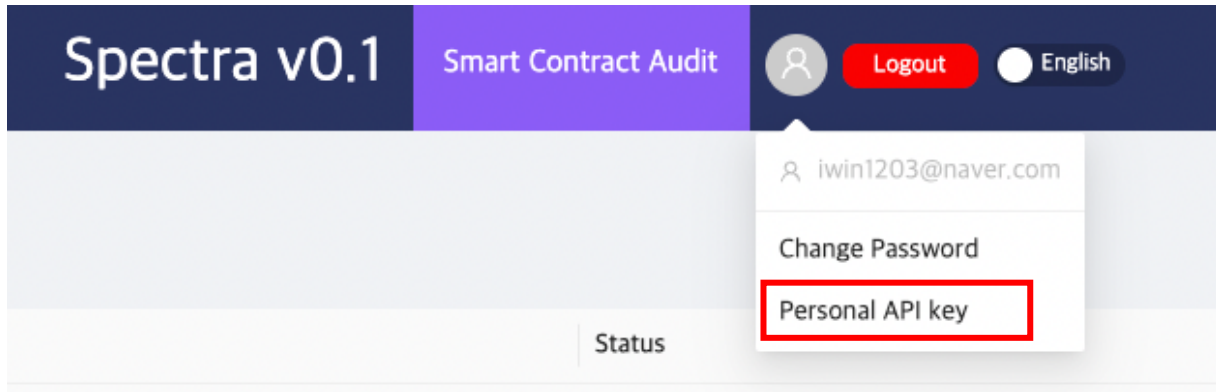
주의) 다음 중 필수는 **"contracts"** 경로이며, 그 외는 선택적입니다.

```
contracts/  
└─ a.sol  
node_modules/  
└─ @openzeppelin/  
package.json  
hardhat.config.js 혹은 truffle-config.js 혹은 brownie-config.yaml
```

# 1. API Key Setup

Spectra는 CI/CD 툴 Jenkins의 플러그인을 제공합니다. \* 본 가이드라인은 Jenkins를 활용하고 있는 프로젝트를 대상으로 합니다.

Spectra Web에서 프로필 아이콘 클릭 > Personal API key > Create new API key를 클릭하여 api key 발급 창으로 이동합니다.



# 1. API Key Setup

Spectra는 CI/CD 툴 Jenkins의 Open API를 제공합니다.

Expiration과 role을 선택하여 API key를 발급받습니다.

Create New API key

X

Key 만료 기한

\* Expiration : 90 days Select date

\* roles : ☒ request audit ☒ view audit

권한: audit 요청 권한 / 결과 조회

Cancel OK



Personal API key You can issue an API key to use the Spectra service.

Create new API key


발급된 Key

5f127d95-244e-42d7-933c-5b3b1cc7a747

roles : view-audit,request-audit created : 2023-05-16 expiration date : 2023-07-15

## 2. Installation

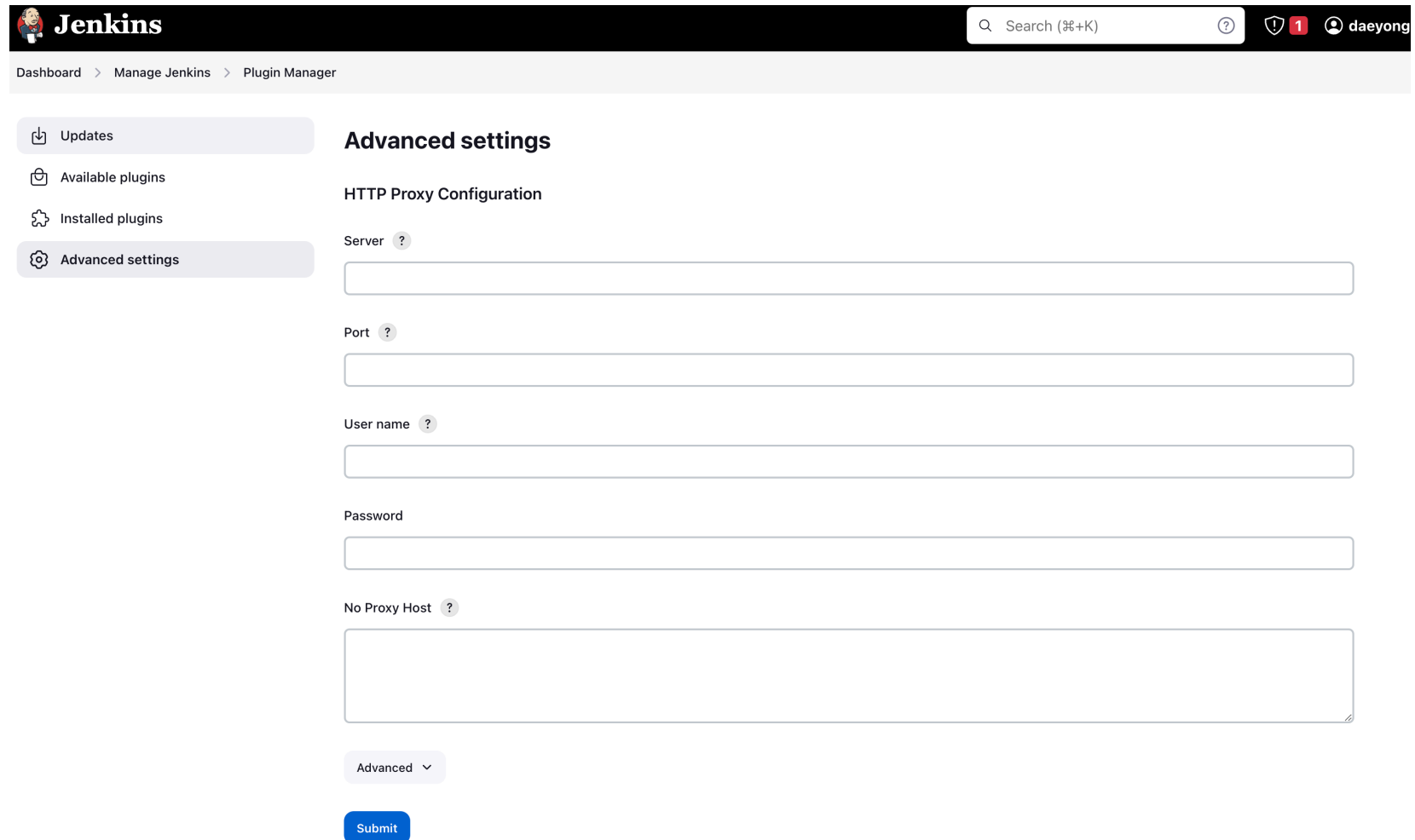
<https://github.com/spark63/spectra-jenkins-plugin> 레포지토리에서 **spectra-plugin.hpi** 파일을 다운로드합니다.

	GanziDaeyong dev: updated hpi	2f39c19 3 hours ago	🕒 25 commits
📁	.github	dev: created structure	3 weeks ago
📁	.mvn	dev: created structure	3 weeks ago
📁	src/main	fix: updated invalid rule template value	4 hours ago
📄	.gitignore	dev: gitignore updated	2 weeks ago
📄	Jenkinsfile	dev: created structure	3 weeks ago
📄	LICENSE.md	dev: created structure	3 weeks ago
📄	README.md	dev: updated README	4 hours ago
📄	img.png	dev: updated README	4 hours ago
📄	pom.xml	dev: updated apache http version	yesterday
📄	spectra-plugin.hpi	dev: updated hpi	3 hours ago



## 2. Installation

Jenkins에서, Dashboard > Manage Jenkins > Plugin Manager > Advanced settings로 이동합니다.



The screenshot shows the Jenkins web interface. At the top is a black header with the Jenkins logo, a search bar, and a user profile. Below the header is a breadcrumb trail: Dashboard > Manage Jenkins > Plugin Manager. On the left is a sidebar with four menu items: Updates, Available plugins, Installed plugins, and Advanced settings (which is highlighted). The main content area is titled 'Advanced settings' and contains the 'HTTP Proxy Configuration' section. This section has five input fields: 'Server', 'Port', 'User name', 'Password', and 'No Proxy Host'. Each field has a small help icon to its right. At the bottom of the form is a 'Submit' button. There is also a small 'Advanced' dropdown menu above the submit button.

Jenkins

Search (⌘+K)

daeyong

Dashboard > Manage Jenkins > Plugin Manager

Updates

Available plugins

Installed plugins

Advanced settings

### Advanced settings

#### HTTP Proxy Configuration

Server ?

Port ?

User name ?

Password

No Proxy Host ?

Advanced

Submit

## 2. Installation

Deploy Plugin에서, 다운로드한 .hpi 파일을 선택 후 Deploy 버튼을 클릭합니다.

### Deploy Plugin

You can select a plugin file from your local system or provide a URL to install a plugin from outside the central plugin repository.

File



파일 선택

spectra-plugin.hpi

Or

URL

Deploy

### 3. Network Setup

---

Jenkins plugin은 다음 IP에 대한 인바운드 / 아웃바운드 허용이 필요합니다.

인바운드: 223.130.162.168 (포트: 유저의 젠킨스 서버 포트와 동일)

아웃바운드: 223.130.162.168 (포트: 443)

## 4. Freestyle Project에 적용하기

작성해둔 freestyle project의 configure에서, Build Steps - Spectra 플러그인을 선택합니다.

### Build Steps

Add build step ^

Filter

- Aqua MicroScanner
- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Dependency-Check
- Invoke Snyk Security task
- Invoke top-level Maven targets
- JSLint
- NeuVector Vulnerability Scanner
- Set build status to "pending" on GitHub commit
- Spectra**
- Vulnerability scan with gype

### Build Steps

≡ Spectra

API Key

Put API key obtained from Spectra website.

abcd123

Add build step v

발급받은 API Key를 입력합니다.

## 4. Freestyle Project에 적용하기

빌드 후 조치 (Post Build Action)에도 동일한 방식으로 Spectra 플러그인을 적용할 수 있습니다.

The diagram illustrates the steps to add Spectra as a Post Build Action in a Freestyle Project. On the left, a list of available actions is shown, with 'Spectra' selected. Below the list is a button labeled '빌드 후 조치 추가 ▲'. An orange arrow points from this button to the right, where a configuration form for the 'Spectra' action is displayed. The form includes a title 'Spectra', a label 'API Key', and a description 'Put API key obtained from Spectra website.' Below the description is a text input field. At the bottom of the form is a button labeled '빌드 후 조치 추가 ▼'. At the bottom left of the left panel, there are two buttons: '저장' (Save) and 'Apply'.

Record fingerprints of files to track usage  
Report Violations  
Git Publisher  
Spectra  
E-mail Notification  
Scan host/instances with Qualys VM  
Scan web applications with Qualys WAS  
Set GitHub commit status (universal)  
Set build status on GitHub commit [deprecated]

빌드 후 조치 추가 ▲

저장 Apply

빌드 후 조치

≡ Spectra

API Key

Put API key obtained from Spectra website.

빌드 후 조치 추가 ▼

## 4. Freestyle Project에 적용하기

프로젝트 빌드 후 Spectra 분석 성공 여부는 Console Output에서 확인할 수 있습니다.

↑ 프로젝트로 돌아가기

☰ 상태

</> 바뀐점

📄 Console Output

📄 View as plain text

📝 빌드 정보 수정

### 콘솔 출력

```
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace /Users/jeongdaeyong/Desktop/ongoing-task/spectra-ci/spectra-plugin/work/workspace/guide
[Spectra] Ready
ERROR: Step 'Spectra' aborted due to exception:
java.io.IOException: [Spectra] Invalid API key.
    at io.jenkins.plugins.SpectraPublisher.perform(SpectraPublisher.java:48)
    at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
```

컨트랙트 분석에 성공한 경우 Spectra Web에서 설정한 이메일로 결과가 전송됩니다.  
API Key가 유효하지 않은 경우 빌드 실패하며 관련 오류는 Console Output에 기록됩니다.

# 5. Pipeline Project에 적용하기

작성해둔 pipeline project의 configure에서, Pipeline script 작성 창으로 이동합니다.

Configuration

General

Advanced Project Options

Pipeline

Advanced Project Options

고급...

Pipeline

Definition

Pipeline script

Script ?

1

try sample Pipeline...

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

## 5. Pipeline Project에 적용하기

다음과 같이 'Spectra' stage를 원하는 위치에 삽입해줍니다.

```
pipeline {
  agent any
  stages {
    stage('Stage A') {
      ...
    }
    stage('Spectra') {
      steps {
        step([$class: 'SpectraBuilder', apikey: 'abcd1234'])
      }
    }
  }
}
```

**apikey: {발급받은 api key} 형식으로 입력합니다. 'apikey'의 철자 및 대소문자에 유의해주세요.**



## 5. Pipeline Project에 적용하기

프로젝트 빌드 후 Spectra 분석 성공 여부는 Console Output에서 확인할 수 있습니다.

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#1'

Restart from Stage

Replay

Pipeline Steps

Workspaces



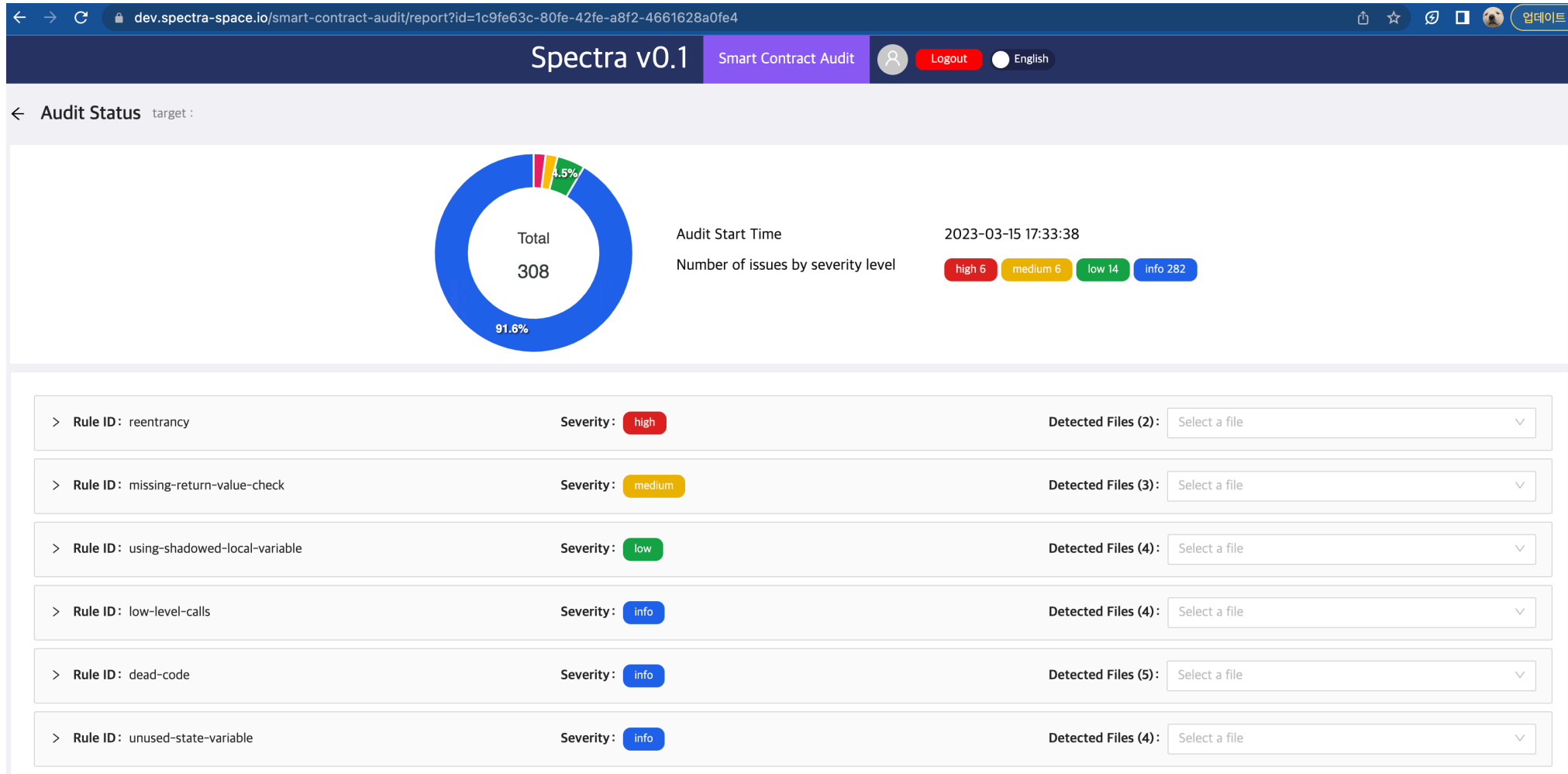
```
Started by user unknown or anonymous
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /Users/jeongdaeyong/Desktop/ongoing-task/spectra-ci/spectra-plugin/work/workspace/plguide
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Spectra)
[Pipeline] step
[Spectra] Ready
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
java.io.IOException: [Spectra] Invalid API key.
    at io.jenkins.plugins.SpectraBuilder.perform(SpectraBuilder.java:55)
    at org.jenkinsci.plugins.workflow.steps.CoreStep$Execution.run(CoreStep.java:101)
    at org.jenkinsci.plugins.workflow.steps.CoreStep$Execution.run(CoreStep.java:71)
    at org.jenkinsci.plugins.workflow.steps.SynchronousNonBlockingStepExecution.lambda$start$0(SynchronousNonBlockingStepExecution.java:47)
    at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
```

컨트랙트 분석에 성공한 경우 Spectra Web에서 설정한 이메일로 결과가 전송됩니다.  
API Key가 유효하지 않은 경우 빌드 실패하며 관련 오류는 Console Output에 기록됩니다.

Finished: FAILURE

## 6. 결과 확인하기

빌드에 성공한 경우, Spectra Web에서 설정한 이메일로 결과 링크가 주어집니다. 해당 링크를 클릭하여 결과를 볼 수 있습니다.



**E O D**

