链滴

SQLMAP API 使用教程

作者：General

原文链接：https://ld246.com/article/1600820913474

来源网站：链滴

为什么使用SqlmapAPI?

由于SQLMAP每检测一个站点都需要开启一个新的命令行窗口或者结束掉上一个检测任务。虽然 -m 数可以批量扫描URL，但是模式也是一个结束扫描后才开始另一个扫描任务。通过api接口，下发扫任务就简单了，无需开启一个新的命令行窗口。

SqlmapAPI

在sqlmap安装目录下的sqlmapapi.py文件，这个就是sqlmap api，sqlmapapi分为服务端以及客户。

```
C:\Users\yanghelong\Desktop\sqlmap>python sqlmapapi.py -h
Usage: sqlmapapi.py [options]

Options:
  -h, --help            显示帮助信息并
  -s, --server          作为Api服务端运行
  -c, --client          作为Api客户端运行
  -H HOST, --host=HOST  指定服务端IP地址(默认IP : "127.0.0.1")
  -p PORT, --port=PORT  指定服务端端口 (默认端口 : 8775)
  --adapter=ADAPTER     服务端标准接口(默认值 : "wsgiref")
  --username=USERNAME   可选，设置用户名
  --password=PASSWORD   可选，设置密码
```

启动服务端

python sqlmapapi.py -s 命令成功后，在命令行中会返回一些信息。大概的意思是api服务端在本地8775端口上运行，admin token为bca6e58ac03c9cb52f35a87aeaa28dcf，IPC数据库的位置在C:\Users\YANGHE~1\AppData\Local\Temp\sqlmapipc-9lqlm_4d，api服务端已经和IPC数据库连接上了正在使用bottle 框架wsgiref标准接口。

```
 C:\Users\yanghelong\Desktop\sqlmap> python sqlmapapi.py -s
[15:52:26] [INFO] Running REST-JSON API server at '127.0.0.1:8775'..
[15:52:26] [INFO] Admin (secret) token: bca6e58ac03c9cb52f35a87aeaa28dcf
[15:52:26] [DEBUG] IPC database: 'C:\Users\YANGHE~1\AppData\Local\Temp\sqlmapipc-9lql
_4d'
[15:52:26] [DEBUG] REST-JSON API server connected to IPC database
[15:52:26] [DEBUG] Using adapter 'wsgiref' to run bottle
```

但是通过上面的这种方式开启api服务端有一个缺点，当服务端和客户端不在同一台主机时会连接不，因此如果要解决这个问题，可以通过输入以下命令来开启api服务端:

```
python sqlmapapi.py -s -H "0.0.0.0" -p 8775
```

命令成功后，远程客户端就可以通过指定远程主机IP和端口来连接到API服务端。

固定admin token

如果您有特殊的需求需要固定admin token的话，可以修改sqlmap目录下的/lib/utils/api.py文件，下是源代码:

```
DataStore.admin_token = encodeHex(os.urandom(16), binary=False)
```

接口模式

基于命令行的接口模式

```
C:\Users\yanghelong\Desktop\sqlmap> python sqlmapapi.py -c
[16:00:23] [DEBUG] Example client access from command line:
        $ taskid=$(curl http://127.0.0.1:8775/task/new 2>1 | grep -o -I '[a-f0-9]\{16\}') && echo
taskid
        $ curl -H "Content-Type: application/json" -X POST -d '{"url": "http://testphp.vulnweb.c
m/artists.php?artist=1"}' http://127.0.0.1:8775/scan/$taskid/start
        $ curl http://127.0.0.1:8775/scan/$taskid/data
        $ curl http://127.0.0.1:8775/scan/$taskid/log
[16:00:23] [INFO] Starting REST-JSON API client to 'http://127.0.0.1:8775'...
[16:00:23] [DEBUG] Calling 'http://127.0.0.1:8775'
[16:00:23] [INFO] Type 'help' or '?' for list of available commands
api> help
help          显示帮助信息
new ARGS      开启一个新的扫描任务
use TASKID    切换指定taskid任务
data          获取当前任务返回的数据
log           获取当前任务的扫描日志
status        获取当前任务的扫描状态
option OPTION 获取当前任务的选项
options       获取当前任务的所有配置信息
stop          停止当前任务
kill          杀死当前任务
list          显示所有任务列表
version       Fetch server version
flush         清空任务(默认是所有任务)
exit          退出客户端
```

检测GET注入

```
new -u url
eg:
//现请求了/task/new创建一个新的taskid，后由发起/scan/cafeb23befb85bb6/start请求开始任务
api> new -u "http://192.168.64.130/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit"
[16:20:11] [DEBUG] Calling 'http://127.0.0.1:8775/task/new'
[16:20:11] [INFO] New task ID is 'cafeb23befb85bb6'
[16:20:11] [DEBUG] Calling 'http://127.0.0.1:8775/scan/cafeb23befb85bb6/start'
[16:20:11] [INFO] Scanning started
api (cafeb23befb85bb6)>
//通过status 获取该任务的扫描结果，若返回内容中的status字段为terminated，说明扫描完成，若
回内容中的status字段为running，说明扫描还在进行中
api (cafeb23befb85bb6)> status
[16:21:29] [DEBUG] Calling 'http://127.0.0.1:8775/scan/cafeb23befb85bb6/status'
{
    "success": true,
    "status": "running",
    "returncode": null
```

```
}
//输入 data 命令，来获取扫描完成后注入出来的信息，若返回的内容中data字段不为空就说明存在
入。


//不存在注入返回的内容，data字段为空
{
    "success": true,
    "data": [],
    "error": []
}
```

检测POST注入

```
//输入以下命令，在data.txt中加入数据，指定注入的位置，来达到检测POST、cookie、UA等注入
目的：
new -r data.txt
api> new -r C:\Users\yanghelong\Desktop\sql\build1\test.txt
[16:29:30] [DEBUG] Calling 'http://127.0.0.1:8775/task/new'
[16:29:30] [INFO] New task ID is '75708f5e15824979'
[16:29:30] [DEBUG] Calling 'http://127.0.0.1:8775/scan/75708f5e15824979/start'
[16:29:30] [INFO] Scanning started
```

基于命令行的接口模式用起来还是比较方便的，我们只需要通过 new 命令就可以自动创建taskid并始该任务.

基于HTTP协议的接口模式

sqlmapapi.py的h基于http接口调用模式的主要函数，进入到lib/utils/api.py的server类,可以发现通向server提交数据进行与服务的交互。

```
#辅助
@get('/error/401')
@get("/task/new")
@get("/task/<taskid>/delete")

#Admin 命令
@get("/admin/list")
@get("/admin/<token>/list")
@get("/admin/flush")
@get("/admin/<token>/flush")

#sqlmap 核心交互命令
@get("/option/<taskid>/list")
@post("/option/<taskid>/get")
@post("/option/<taskid>/set")
@post("/scan/<taskid>/start")
@get("/scan/<taskid>/stop")
@get("/scan/<taskid>/kill")
@get("/scan/<taskid>/status")
@get("/scan/<taskid>/data")
@get("/scan/<taskid>/log/<start>/<end>")
@get("/scan/<taskid>/log")
```

```
@get("/download/<taskid>/<target>/<filename:path>")
```

@get("/task/new")

新建任务,返回一个随机的taskid

```
@get("/task/new")
def task_new():
    """
    Create a new task
    """
    taskid = encodeHex(os.urandom(8), binary=False)
    remote_addr = request.remote_addr

    DataStore.tasks[taskid] = Task(taskid, remote_addr)

    logger.debug("Created new task: '%s'" % taskid)
    return jsonize({"success": True, "taskid": taskid})
```

@get("/task/<taskid>/delete")

删除任务，在调用时指定taskid，不指定taskid会报错。

```
@get("/task/<taskid>/delete")
def task_delete(taskid):
    """
    Delete an existing task
    """
    if taskid in DataStore.tasks:
        DataStore.tasks.pop(taskid)

        logger.debug("(%s) Deleted task" % taskid)
        return jsonize({"success": True})
    else:
        response.status = 404
        logger.warning("[%s] Non-existing task ID provided to task_delete()" % taskid)
        return jsonize({"success": False, "message": "Non-existing task ID"})
```

@get("/admin/list")

获取所有taskid。

```
def task_list(token=None):
    """
    Pull task list
    """
    tasks = {}
    for key in DataStore.tasks:
        if is_admin(token) or DataStore.tasks[key].remote_addr == request.remote_addr:
            tasks[key] = dejsonize(scan_status(key))["status"]
    logger.debug("(%s) Listed task pool (%s)" % (token, "admin" if is_admin(token) else request
remote_addr))
    return jsonize({"success": True, "tasks": tasks, "tasks_num": len(tasks)})
```

@get("/option/<taskid>/list")

获取特定任务ID的列表选项，调用时请指定taskid，不然会报错。具体代码如下：

```
def option_list(taskid):
    """
    List options for a certain task ID
    """
    if taskid not in DataStore.tasks:
        logger.warning("[%s] Invalid task ID provided to option_list()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    logger.debug("(%s) Listed task options" % taskid)
    return jsonize({"success": True, "options": DataStore.tasks[taskid].get_options()})
```

@post("/option/<taskid>/get")

获取特定任务ID的选项值，调用时请指定taskid，不然会报错。具体代码如下：

```
def option_get(taskid):
    """
    Get value of option(s) for a certain task ID
    """
    if taskid not in DataStore.tasks:
        logger.warning("[%s] Invalid task ID provided to option_get()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    options = request.json or []
    results = {}
    for option in options:
        if option in DataStore.tasks[taskid].options:
            results[option] = DataStore.tasks[taskid].options[option]
        else:
            logger.debug("(%s) Requested value for unknown option '%s'" % (taskid, option))
            return jsonize({"success": False, "message": "Unknown option '%s'" % option})
    logger.debug("(%s) Retrieved values for option(s) '%s'" % (taskid, ",".join(options)))
    return jsonize({"success": True, "options": results})
```

@post("/option/<taskid>/set")

为特定任务ID设置选项值，调用时请指定taskid，不然会报错。具体代码如下：

```
def option_set(taskid):
    """
    Set value of option(s) for a certain task ID
    """
    if taskid not in DataStore.tasks:
        logger.warning("[%s] Invalid task ID provided to option_set()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    if request.json is None:
        logger.warning("[%s] Invalid JSON options provided to option_set()" % taskid)
        return jsonize({"success": False, "message": "Invalid JSON options"})
    for option, value in request.json.items():
        DataStore.tasks[taskid].set_option(option, value)
    logger.debug("(%s) Requested to set options" % taskid)
    return jsonize({"success": True})
```

@post("/scan/<taskid>/start")

开始扫描特定任务，调用时请指定taskid，不然会报错。具体代码如下:

```
def scan_start(taskid):
    """
    Launch a scan
    """
    if taskid not in DataStore.tasks:
        logger.warning("[%s] Invalid task ID provided to scan_start()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    if request.json is None:
        logger.warning("[%s] Invalid JSON options provided to scan_start()" % taskid)
        return jsonize({"success": False, "message": "Invalid JSON options"})
    # Initialize sqlmap engine's options with user's provided options, if any
    for option, value in request.json.items():
        DataStore.tasks[taskid].set_option(option, value)
    # Launch sqlmap engine in a separate process
    DataStore.tasks[taskid].engine_start()
    logger.debug("(%s) Started scan" % taskid)
    return jsonize({"success": True, "engineid": DataStore.tasks[taskid].engine_get_id()})
```

@get("/scan/<taskid>/stop")

停止扫描特定任务，调用时请指定taskid，不然会出现问题。具体代码如下：

```
def scan_stop(taskid):
    """
    Stop a scan
    """
    if (taskid not in DataStore.tasks or DataStore.tasks[taskid].engine_process() is None or Data
tore.tasks[taskid].engine_has_terminated()):
        logger.warning("[%s] Invalid task ID provided to scan_stop()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    DataStore.tasks[taskid].engine_stop()
    logger.debug("(%s) Stopped scan" % taskid)
    return jsonize({"success": True})
```

@get("/scan/<taskid>/status")

该接口可查询扫描状态，调用时请指定taskid，不然会出现问题。具体代码如下：

```
def scan_status(taskid):
    """
    Returns status of a scan
    """
    if taskid not in DataStore.tasks:
        logger.warning("[%s] Invalid task ID provided to scan_status()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    if DataStore.tasks[taskid].engine_process() is None:
        status = "not running"
    else:
        status = "terminated" if DataStore.tasks[taskid].engine_has_terminated() is True else "run
ing"
    logger.debug("(%s) Retrieved scan status" % taskid)
    return jsonize({
        "success": True,
```

```
        "status": status,
        "returncode": DataStore.tasks[taskid].engine_get_returncode()
    })
```

@get("/scan/<taskid>/data")

该接口可获得到扫描结果，调用时请指定taskid，不然会出现问题。具体代码如下：

```
def scan_data(taskid):
    """
    Retrieve the data of a scan
    """
    json_data_message = list()
    json_errors_message = list()
    if taskid not in DataStore.tasks:
        logger.warning("[%s] Invalid task ID provided to scan_data()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    # Read all data from the IPC database for the taskid
    for status, content_type, value in DataStore.current_db.execute("SELECT status, content_typ
, value FROM data WHERE taskid = ? ORDER BY id ASC", (taskid,)):
        json_data_message.append({"status": status, "type": content_type, "value": dejsonize(valu
)})
    # Read all error messages from the IPC database
    for error in DataStore.current_db.execute("SELECT error FROM errors WHERE taskid = ? OR
ER BY id ASC", (taskid,)):
        json_errors_message.append(error)
    logger.debug("(%s) Retrieved scan data and error messages" % taskid)
    return jsonize({"success": True, "data": json_data_message, "error": json_errors_message})
```

@get("/scan/<taskid>/log")

该接口可查询特定任务的扫描的日志，调用时请指定taskid，不然会出现问题。具体代码如下：

```
def scan_log(taskid):
    """
    Retrieve the log messages
    """
    json_log_messages = list()
    if taskid not in DataStore.tasks:
        logger.warning("[%s] Invalid task ID provided to scan_log()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    # Read all log messages from the IPC database
    for time_, level, message in DataStore.current_db.execute("SELECT time, level, message FR
M logs WHERE taskid = ? ORDER BY id ASC", (taskid,)):
        json_log_messages.append({"time": time_, "level": level, "message": message})
    logger.debug("(%s) Retrieved scan log messages" % taskid)
    return jsonize({"success": True, "log": json_log_messages})


def scan_log_limited(taskid, start, end):
    """
    Retrieve a subset of log messages
    """
    json_log_messages = list()
    if taskid not in DataStore.tasks:
```

```
        logger.warning("[%s] Invalid task ID provided to scan_log_limited()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    if not start.isdigit() or not end.isdigit() or end < start:
        logger.warning("[%s] Invalid start or end value provided to scan_log_limited()" % taskid)
        return jsonize({"success": False, "message": "Invalid start or end value, must be digits"})
    start = max(1, int(start))
    end = max(1, int(end))
    # Read a subset of log messages from the IPC database
    for time_, level, message in DataStore.current_db.execute("SELECT time, level, message FR
M logs WHERE taskid = ? AND id >= ? AND id <= ? ORDER BY id ASC", (taskid, start, end)):
        json_log_messages.append({"time": time_, "level": level, "message": message})
    logger.debug("(%s) Retrieved scan log messages subset" % taskid)
    return jsonize({"success": True, "log": json_log_messages})
```

@get("/download/<taskid>/<target>/<filename>/")

下载服务端指定任务的文件。具体代码如下：

```
def download(taskid, target, filename):
    """
    Download a certain file from the file system
    """
    if taskid not in DataStore.tasks:
        logger.warning("[%s] Invalid task ID provided to download()" % taskid)
        return jsonize({"success": False, "message": "Invalid task ID"})
    path = os.path.abspath(os.path.join(paths.SQLMAP_OUTPUT_PATH, target, filename))
    # Prevent file path traversal
    if not path.startswith(paths.SQLMAP_OUTPUT_PATH):
        logger.warning("[%s] Forbidden path (%s)" % (taskid, target))
        return jsonize({"success": False, "message": "Forbidden path"})
    if os.path.isfile(path):
        logger.debug("(%s) Retrieved content of file %s" % (taskid, target))
        with open(path, 'rb') as inf:
            file_content = inf.read()
        return jsonize({"success": True, "file": base64encode(file_content)})
    else:
        logger.warning("[%s] File does not exist %s" % (taskid, target))
        return jsonize({"success": False, "message": "File does not exist"})
```

总结

基于HTTP的接口模式用起来可能比较繁琐，但是对于程序调用接口还是很友善的。总之该模式的流是：

1. 通过GET请求　http://ip:port/task/new 这个地址，创建一个新的扫描任务；

2. 通过POST请求　http://ip:port/scan/<taskid>/start 地址，并通过json格式提交参数，开启一扫描；

3. 通过GET请求　http://ip:port/scan/<taskid>/status 地址，即可获取指定的taskid的扫描状态。个返回值分为两种，一种是run状态（扫描未完成），一种是terminated状态（扫描完成）；

4. 扫描完成后获取扫描的结果。