



OWASP

Open Web Application
Security Project

REST API安全测试的思路

基于Swagger的REST API自动化安全测试实践

贾玉彬 Gary

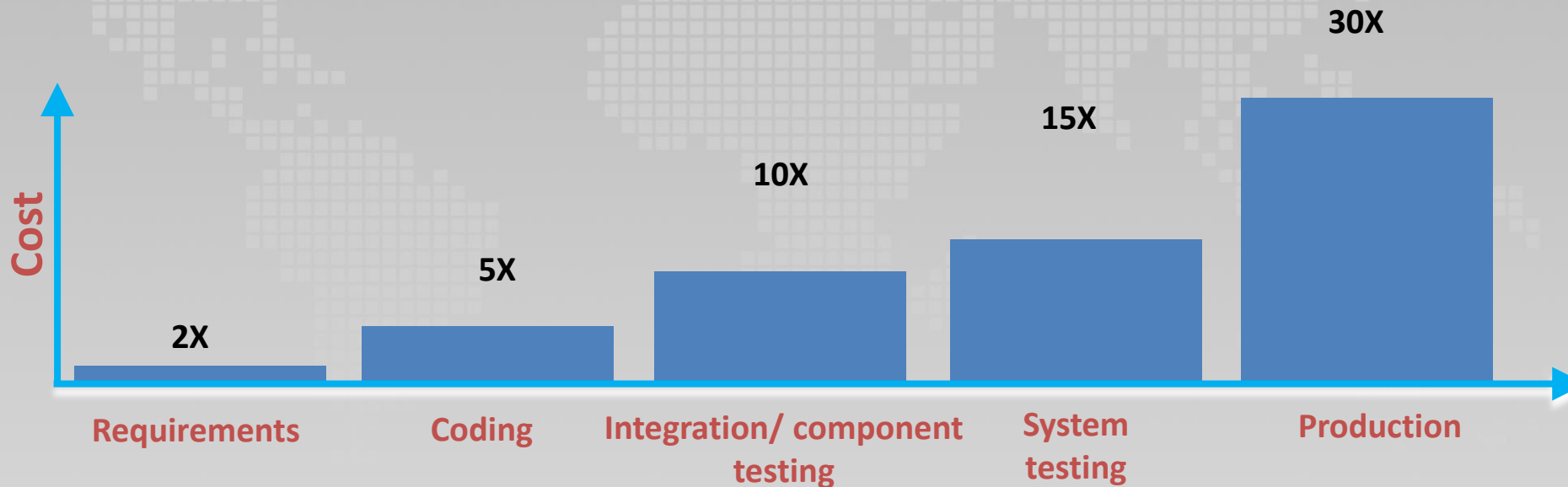
Gary_jia@rapid7.com

大纲

- 由应用安全说起
- 传统REST API安全测试
- 基于Swagger的REST API 的安全测试



在SDLC中尽早发现和修复安全问题



Source: NIST

不同的团队不同的目标



DevSecOps

“Everyone is responsible for security” with the goal of safely distributing security decisions at speed and scale

It does not have to be like this:

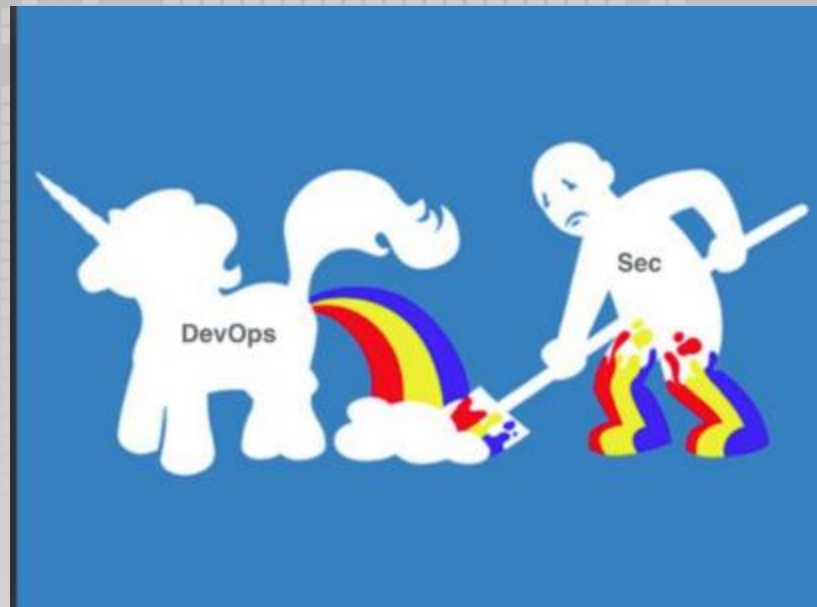


Image : Pete Cheslock at #DevOpsDaysAustin.

传统的REST API安全测试

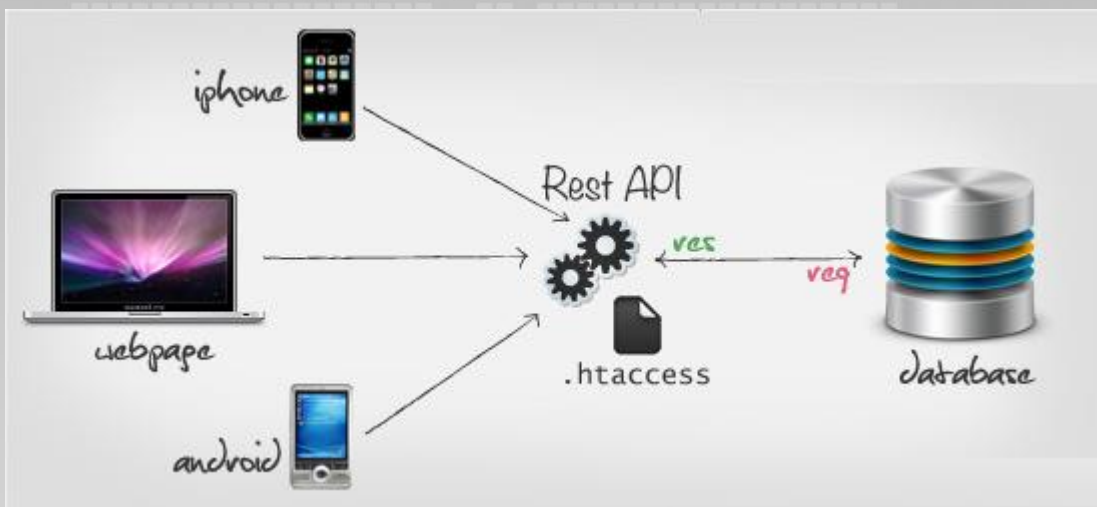


REST API

- REST – REpresentational State Transfer
- REST描述的是客户端和服务端的一种交互形式，REST本身不实用，实用的是如何设计RESTful风格 API
- 传输资源的形式一般是JSON和XML

RESTful结构的好处

RESTful通过一套统一的接口为Web, iOS和Android提供服务



*图片来自互联网

REST API安全问题会带来的后果

- 接口被滥用消耗系统资源
- 数据泄露
- 伪造/篡改数据
- 应用被仿制
- 引入其他安全问题

REST API是可被攻击的

- REST API是可以被常用的攻击方法进行攻击的，如注入、XSS、CSRF、XML Entity攻击等等



REST API测试

- 通常没有WEB界面与之进行交互，通过HTTP的各种终端工具或自己编写HTTP客户端脚本与服务端通讯来进行测试
- 通过变化API调用参数组合，变化API功能调用顺序，实现全面和各种复杂组合的调用
- 针对安全测试也存在同样的问题



以移动应用为例的测试过程（手工）

- 安装配置代理
- 配置测试移动设备
- 准备测试用数据
- 手工测试



安装并设置代理、配置移动设备

The image displays several screenshots from the OWASP ZAP 2.4.3 interface, illustrating the process of setting up a mobile proxy and analyzing traffic.

- Local Proxy Configuration:** The 'Local Proxy' dialog is shown with the following settings:
 - Local Proxy: Local Proxy
 - Address (eg localhost, 127.0.0.1): 192.168.1.109
 - Port (eg 8080): 8081
 - Security Protocols: TLS 1.1, TLS 1.2
- Edit access point:** The 'Edit access point' dialog is shown with the following settings:
 - Name: US
 - APN: tmobile.us
 - epc.tmobile.com
 - Proxy: 192.168.1.109
 - Port: 8081
 - Username: none
- Traffic Analysis:** The 'Request' tab shows a GET request to `http://192.168.1.109/api/category?page=1&per_page=1000 HTTP/1.1`. The 'Response' tab shows the corresponding HTML response from the server.
- Mobile Device View:** A screenshot of a mobile device screen shows the 'Products' page of 'Martha Stewart Crafts'. The products listed are:
 - Martha Stewart Crafts Garland, Pink Pom Pom Small (Price: \$9.0)
 - Martha Stewart Crafts Modern Festive Pennant Garland, 12ft (Price: \$5.6)
 - Martha Stewart Crafts Pom Poms, Pink, 2 Sizes (Price: \$4.8)
- Alerts and History:** The 'Alerts' and 'History' tabs show a list of requests and responses, including the 'SetCookie' response for the product page.

手工测试REST API（SQL注入为例）

- 测试是否存在注入漏洞
- 例如使用PUT方法更新用户档案
- 请求中包含参数值（first_name）

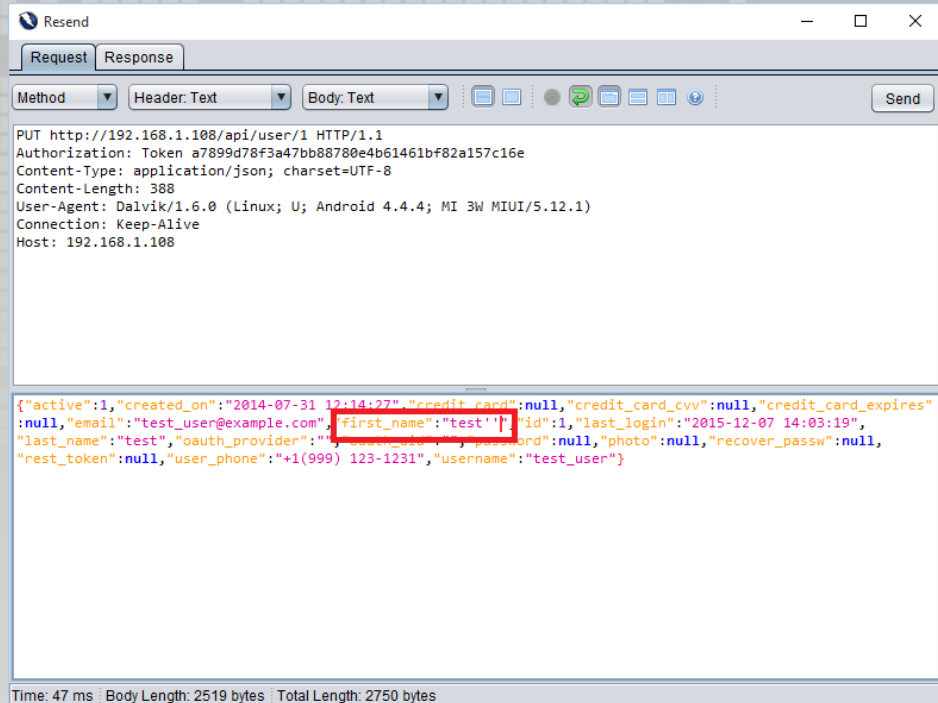
URL: <http://192.168.202.131/api/user/1>
Method: PUT
Parameter name: first_name
Attack values: test''



手工测试REST API（SQL注入为例）

Request 1

使用test”把两个单引号（'）注入到请求



The screenshot shows a REST client window titled "Resend". The "Request" tab is active, displaying a PUT request to `http://192.168.1.108/api/user/1`. The request headers include `Authorization: Token a7899d78f3a47bb88780e4b61461bf82a157c16e`, `Content-Type: application/json; charset=UTF-8`, `Content-Length: 388`, `User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.4; MI 3W MIUI/5.12.1)`, `Connection: Keep-Alive`, and `Host: 192.168.1.108`. The response tab shows a JSON object with the following fields: `active:1`, `created_on:"2014-07-31 12:14:27"`, `credit_card:null`, `credit_card_cvv:null`, `credit_card_expires:null`, `email:"test_user@example.com"`, `first_name:"test"`, `id:1`, `last_login:"2015-12-07 14:03:19"`, `last_name:"test"`, `oauth_provider:""`, `password:null`, `photo:null`, `recover_passw:null`, `rest_token:null`, `user_phone:"+1(999) 123-1231"`, and `username:"test_user"`. The `first_name:"test"` field is highlighted with a red box.

```
PUT http://192.168.1.108/api/user/1 HTTP/1.1
Authorization: Token a7899d78f3a47bb88780e4b61461bf82a157c16e
Content-Type: application/json; charset=UTF-8
Content-Length: 388
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.4; MI 3W MIUI/5.12.1)
Connection: Keep-Alive
Host: 192.168.1.108

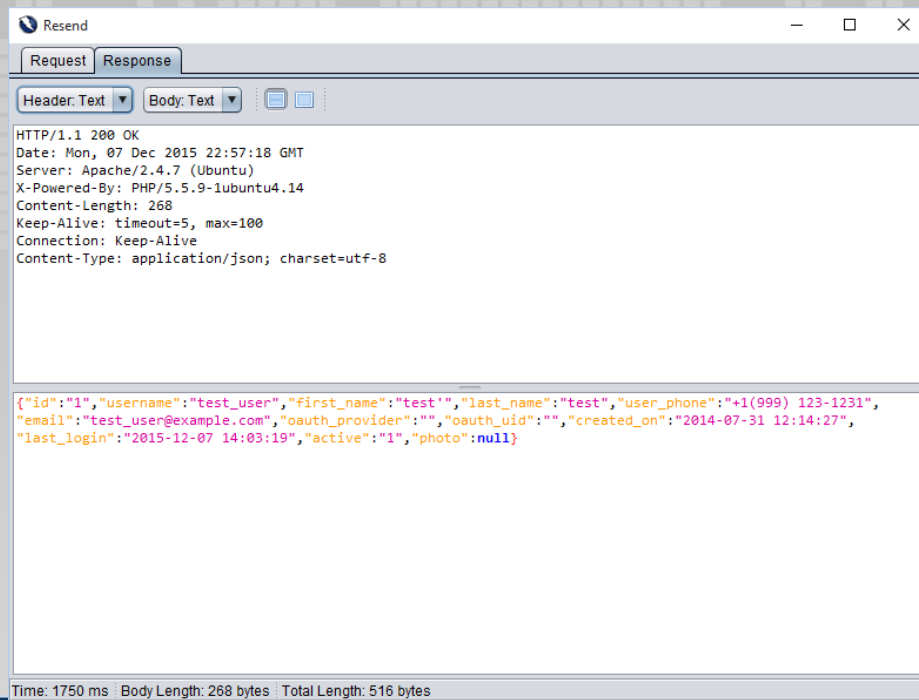
{"active":1,"created_on":"2014-07-31 12:14:27","credit_card":null,"credit_card_cvv":null,"credit_card_expires":null,"email":"test_user@example.com","first_name":"test","id":1,"last_login":"2015-12-07 14:03:19","last_name":"test","oauth_provider":"","password":null,"photo":null,"recover_passw":null,"rest_token":null,"user_phone":"+1(999) 123-1231","username":"test_user"}
```

Time: 47 ms | Body Length: 2519 bytes | Total Length: 2750 bytes

手工测试REST API（SQL注入为例）

Response 1

用户档案成功提交



```
Resend
Request Response
Header: Text Body: Text
HTTP/1.1 200 OK
Date: Mon, 07 Dec 2015 22:57:18 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 268
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8

{"id":1,"username":"test_user","first_name":"test","last_name":"test","user_phone":"+1(999) 123-1231",
"email":"test_user@example.com","oauth_provider":"","oauth_uid":"","created_on":"2014-07-31 12:14:27",
"last_login":"2015-12-07 14:03:19","active":1,"photo":null}

Time: 1750 ms | Body Length: 268 bytes | Total Length: 516 bytes
```


手工测试REST API（XSS为例）

- 测试是否存在XSS漏洞
- 例如使用PUT方法更新用户档案
- 请求中包含参数值（first_name）

URL: <http://192.168.202.131/api/user/1>

Method: PUT

Parameter name: first_name

Attack values: `<script>alert(1)</script>`

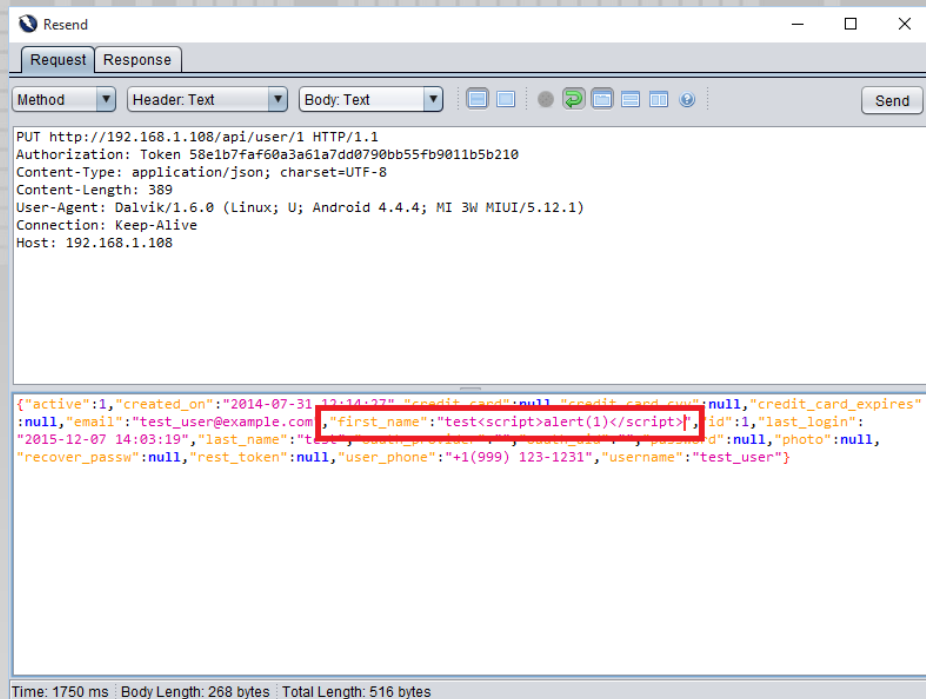


手工测试REST API (XSS为例)

Request

在请求中使用

`<script>alert(1)</script>`把一个脚本标签注入到`first_name`参数值



```
Resend
Request Response
Method Header.Text Body.Text Send
PUT http://192.168.1.108/api/user/1 HTTP/1.1
Authorization: Token 58e1b7faf60a3a61a7dd0790bb55fb9011b5b210
Content-Type: application/json; charset=UTF-8
Content-Length: 389
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.4; MI 3W MIUI/5.12.1)
Connection: Keep-Alive
Host: 192.168.1.108

{"active":1,"created_on":"2014-07-31 12:14:37","credit_card":null,"credit_card_cvv":null,"credit_card_expires":null,"email":"test_user@example.com","first_name":"test<script>alert(1)</script>","id":1,"last_login":"2015-12-07 14:03:19","last_name":"test","password":"test","password_expire":null,"password_reset":null,"photo":null,"recover_passw":null,"rest_token":null,"user_phone":"+1(999) 123-1231","username":"test_user"}
```

Time: 1750 ms · Body Length: 268 bytes · Total Length: 516 bytes

手工测试REST API (XSS为例)

Resend

Request Response

Header: Text Body: Text

```
HTTP/1.1 200 OK
Date: Mon, 07 Dec 2015 23:02:28 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 293
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8
```

```
{
  "id": "1",
  "username": "test_user",
  "first_name": "test<script>alert(1)</script>",
  "last_name": "test",
  "user_phone": "+1(999) 123-1231",
  "email": "test_user@example.com",
  "oauth_provider": "",
  "oauth_uid": "",
  "created_on": "2014-07-31 12:14:27",
  "last_login": "2015-12-07 14:03:19",
  "active": "1",
  "photo": null
}
```

Time: 1578 ms | Body Length: 293 bytes | Total Length: 541 bytes

192.168.1.108/account#profile

HACKAZON

The page at 192.168.1.108 says:

1

Prevent this page from creating additional dialogs.

OK

Your account Logout

Search!

My Account

Home / My Account

My Latest Orders Profile

Order №	Date	Payment Method	Shipping Method	Status
10002797	12/04/2015	creditcard	mail	complete
10000256	12/02/2015	creditcard	mail	complete
10000512	12/02/2015	creditcard	mail	complete
10001024	12/02/2015	creditcard	mail	complete
10001280	12/02/2015	creditcard	mail	complete

Go to my orders

使用DAST工具测试REST API

- 手工测试REST API是个很具挑战性的工作
- 大部分的DAST工具都需要training mode对REST API进行测试（使用复杂的JSON、XML、GWT结构）
- 方法：导入预先录制的流量文件进行测试

以AppSpider + BurpSuite为例

- BurpSuite使用广泛
- AppSpider可以接受各种录制的流量并且可以识别JSON、XML、GWT、AMF参数和值，不需要使用training mode



DEMO

基于SWAGGER的REST API的安全测试



OWASP
Open Web Application
Security Project

基于Swagger的REST API

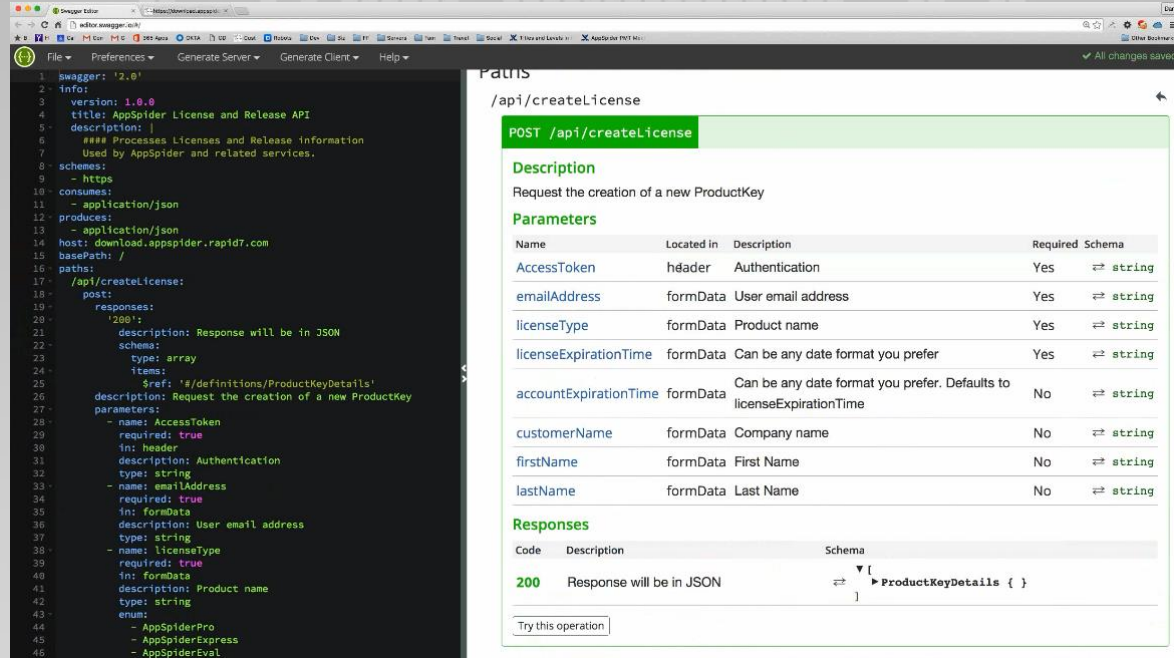
- Swagger是一个简单又强大的REST API文档生成工具
- 标准的、语言无关的，通过它计算机和人类无需阅读代码、文档或者监测网络流量就能发现并理解Web服务



Swagger REST API

```
{
  "swagger": "2.0",
  "info": {
    "version": "0.0.0",
    "title": "Hackazon modern web app"
  },
  "host": "192.168.1.108/api",
  "schemes": [
    "http"
  ],
  "paths": {
    "/category": {
      "get": {
        "parameters": [
          {
            "name": "page",
            "in": "query",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "per_page",
            "in": "query",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "Authorization",
            "in": "header",
            "required": true,
            "default": "Token token_name",
            "schema": {

```



The screenshot shows the Swagger UI interface for the endpoint `/api/createLicense`. The interface is divided into several sections:

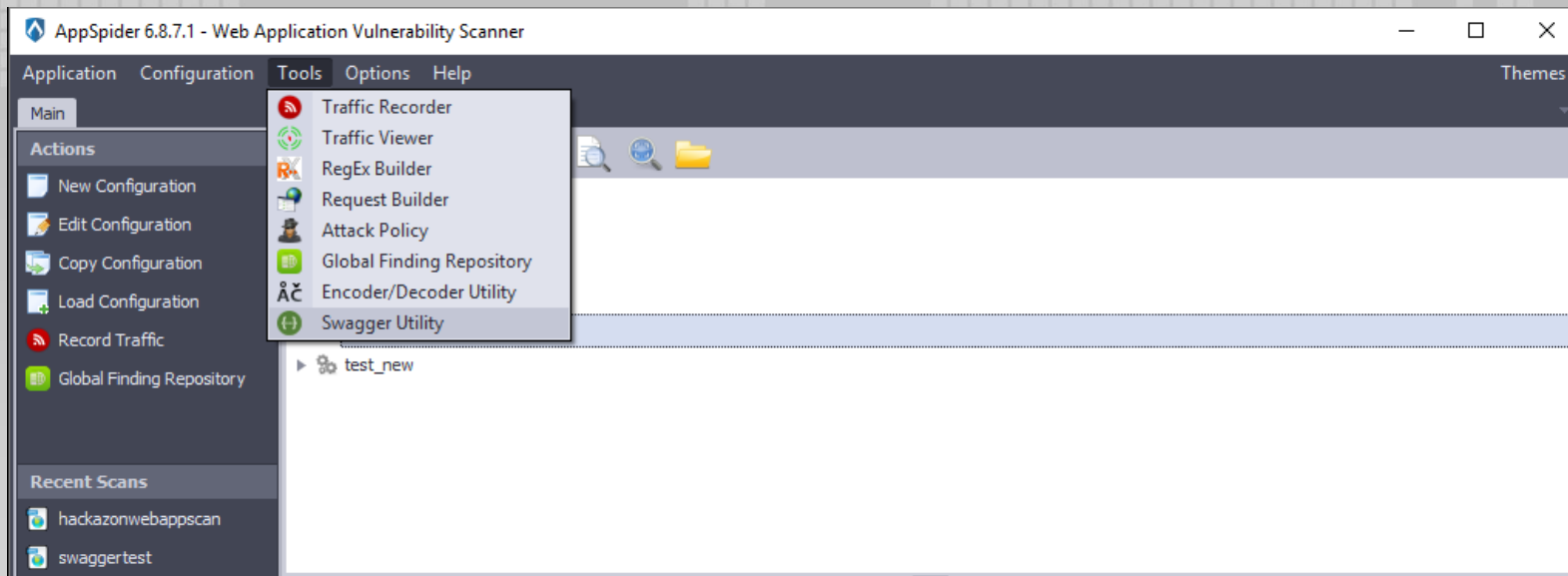
- Paths:** Shows the endpoint `/api/createLicense` with a `POST` method.
- Description:** "Request the creation of a new ProductKey".
- Parameters:** A table listing query and form parameters.
- Responses:** A table showing a `200` response with a JSON schema.
- Try this operation:** A button to execute the API call.

Name	Located in	Description	Required	Schema
AccessToken	header	Authentication	Yes	string
emailAddress	formData	User email address	Yes	string
licenseType	formData	Product name	Yes	string
licenseExpirationTime	formData	Can be any date format you prefer	Yes	string
accountExpirationTime	formData	Can be any date format you prefer. Defaults to licenseExpirationTime	No	string
customerName	formData	Company name	No	string
firstName	formData	First Name	No	string
lastName	formData	Last Name	No	string

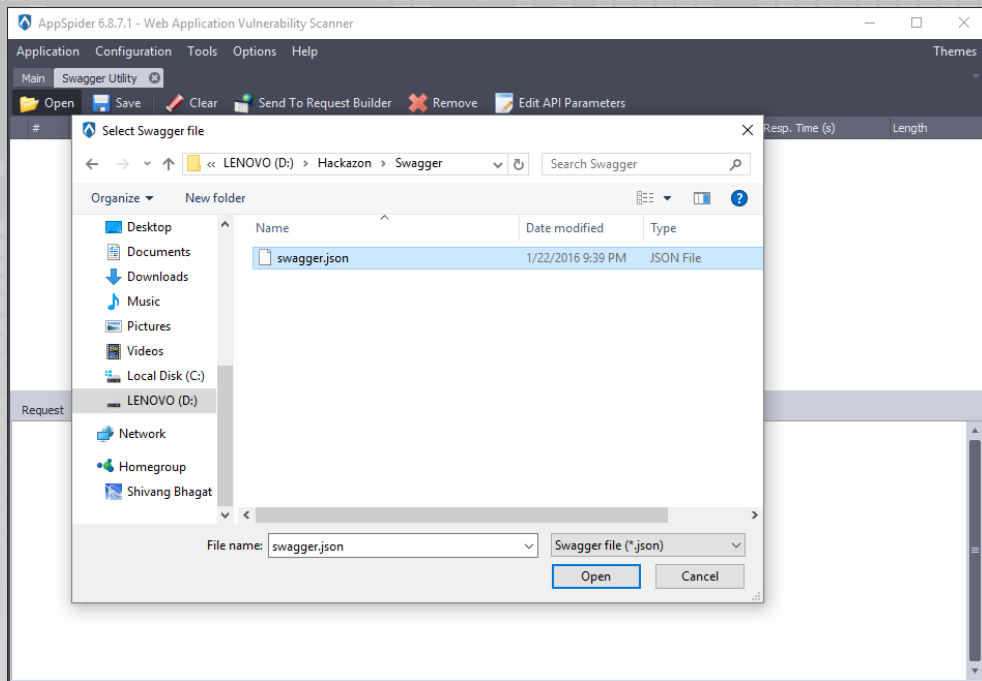
Code	Description	Schema
200	Response will be in JSON	{ ProductKeyDetails { } }

DEMO

运行AppSpider并选择Swagger Utility



导入Swagger JSON文件



API function calls

The screenshot displays the AppSpider 6.8.7.1 interface. The main window shows a table of API calls with columns for #, Status, Code, Method, Host, URL, Resp. Time (s), and Length. Below the table, the 'Request' tab is selected, showing details for a GET request to the URL `http://192.168.1.108/api/category?page=41&per_page=109`.

#	Status	Code	Method	Host	URL	Resp. Time (s)	Length
0	!		GET	192.168.1.108	/api/category?page=41&per_page=109	0.000	0
1	!		GET	192.168.1.108	/api/category/126	0.000	0
2	!		GET	192.168.1.108	/api/user/107	0.000	0
3	!		PUT	192.168.1.108	/api/user/110	0.000	0
4	!		GET	192.168.1.108	/api/user/me	0.000	0
5	!		GET	192.168.1.108	/api/product?page=66&categoryID=16	0.000	0
6	!		GET	192.168.1.108	/api/order?page=18	0.000	0
7	!		POST	192.168.1.108	/api/order	0.000	0
8	!		GET	192.168.1.108	/api/order/47	0.000	0
9	!		GET	192.168.1.108	/api/cart/my?uid=70	0.000	0
10	!		PUT	192.168.1.108	/api/cart/%7bid%7D	0.000	0
11	!		DELETE	192.168.1.108	/api/cart/14	0.000	0
12	!		POST	192.168.1.108	/api/cart/items	0.000	0

Request: `GET http://192.168.1.108/api/category?page=41&per_page=109 HTTP/1.1`
Host: 192.168.1.108
Authorization: 32

操作API参数

The screenshot displays the AppSpider 6.8.7.1 interface. The main window shows a list of API endpoints with columns for #, Status, Code, Method, Host, URL, Resp. Time (s), and Length. A modal window titled "API Parameters Editor" is open, showing the function `/cart/{id}` and its parameters.

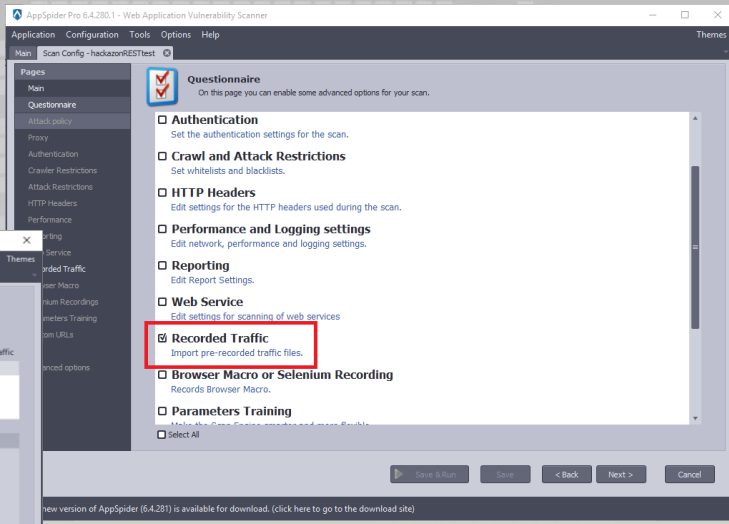
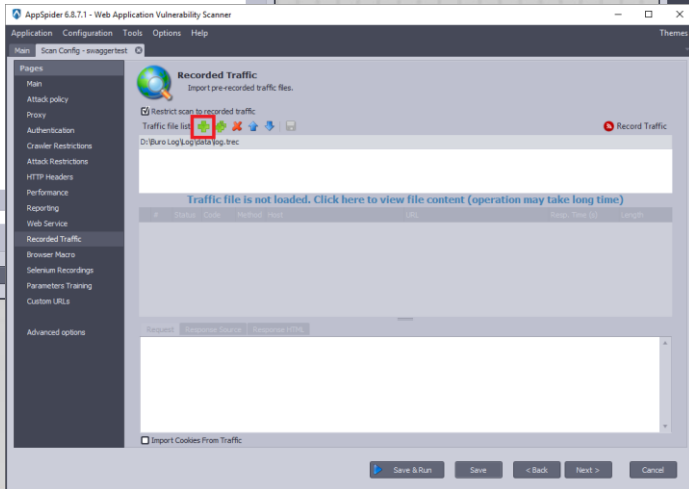
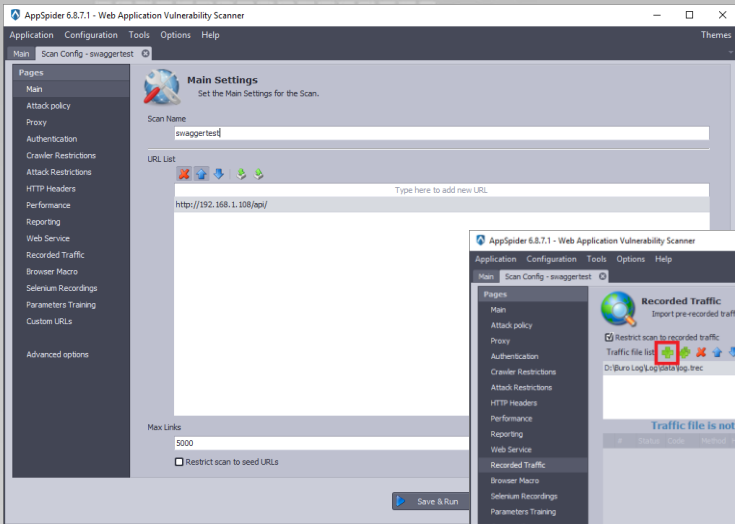
Parameter	Value
Authorization	token e2cd7631cf:00089ed007d4e51eb8c1fbc6c7787
created_at	86
updated_at	56
items_count	0
total_price	6
items_qty	71
uid	39
customer_jd	31
customer_email	18
customer_is_guest	65
payment_method	107
shipping_method	56
shipping_address_jd	14
billing_address_jd	35
last_step	118
Authorization	92
id	14

The background interface shows a table of API endpoints with status icons (red circles with exclamation marks) and a request log at the bottom left:

```
Request Response Source R
GET http://192.168.1.108/ap
Host: 192.168.1.108
Authorization: 32
```



创建新的扫描配置





Scan Results

Scan Name: Webscantest-includeAPIs-reactjs
Date: 8/24/2016 11:24:23 PM
Authenticated User: testuser
Total Links / Attackable Links: 416 / 416
Target URL: http://webscantest.com
Reports: Select Report

Security Status - Partial

Vulnerability	Best Practice	Exposure
		

Summary

A partial scan was performed.

- We crawled **416 links** for which we performed **39,701** requests.
- There are **556 vulnerabilities** detected which can be remediated.
- There are an additional 98 findings such as Best Practices.

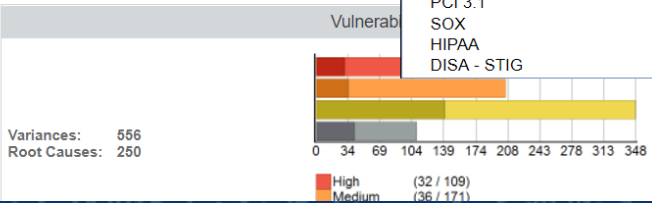
There were issues affecting the scan:

- More than **5%** of the requests failed in this scan. This is a high number of maximum concurrent requests and run the scan again with a lower number of concurrent requests.
- We detected loss of session **1 time**. While it is true the session expiration policy generally does not cause loss of session. This suggests your session expiration policies might be a bit too fragile, compromising usability.

Select Report

- Select Report
- Scan Results Home
- Executive Summary
- Remediation Summary
- Application Threat Modeling
- Reflection Report
- Vulnerabilities
- Remediation Reports:**
 - Application Developer
 - Application Developer by URL
 - Server Administrator
 - Database Administrator
 - Database Administrator by URL
- Best Practices and Compliance Reports:**
 - Best Practices
 - Privacy
 - PCI 3.1
 - SOX
 - HIPAA
 - DISA - STIG

Vulnerabilities



Variations: 556
Root Causes: 250

remediation labor by 55%.

server and significantly compromises the assessment value of this scan. Try setting a lower number (i.e. half to two-thirds of the current value)

generally does not cause loss of session. This suggests your session expiration policies might be a bit too fragile, compromising usability

Vulnerability Reports

Total Vulnerabilities	250 Root Causes
Application & Database	243 Root Causes
Server Administrator	7 Root Causes

应用

Blind SQL Injection

Site: http://webscantest.com:80

URL: http://webscantest.com/datastore/getimage_by_id.php

Attack Type	Original Value	Attacker
Additive Equivalence	1	1+0
Subtractive Equivalence	1	2-1

Vulnerability Validator

Step 1 Step 2 Step 3 Step 4 Step 5 Step 6 Step 7 Step 8 Step 9

Validate for Step 1

Attack Request

GET http://webscantest.com/datastore/getimage_by_id.php?id=3

ADD PARAMETERS

```

GET /datastore/getimage_by_id.php?id=3 HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Charset: *
Accept-Encoding: gzip, deflate
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Host: webscantest.com
Referer: http://webscantest.com/shutterdb/search_get_by_id.php?id=1
Cookies: last_search:4; NB_SRVID:srv140717; TEST_SESSIONID:qg6ko5cfvnde9u75gcvn0i2r1;
firstname:John%27+AND+1%3D0%29--;
  
```

HTTP COMPARE PROXY COOKIES SEND

Attack Response

HIGH

HIGH

VALIDATE

- Original Traffic +
- Attack Traffic #1 +
- Attack Traffic #2 +
- Attack Traffic #3 +
- Attack Traffic #4 +
- Attack Traffic #5 +
- Attack Traffic #6 +
- Attack Traffic #7 +
- Attack Traffic #8 +
- Attack Traffic #9 +

VALIDATE

- Original Traffic +
- Attack Traffic #1 +



总结

- REST API安全测试很重要
- 尽量使用自动化/半自动化工具进行测试
- 使用Swagger生成REST API文档
- 使用支持Swagger的DAST进行安全测试



谢谢



OWASP
Open Web Application
Security Project