OWASP
Open Web Application Security Project

REST API安全测试的思路

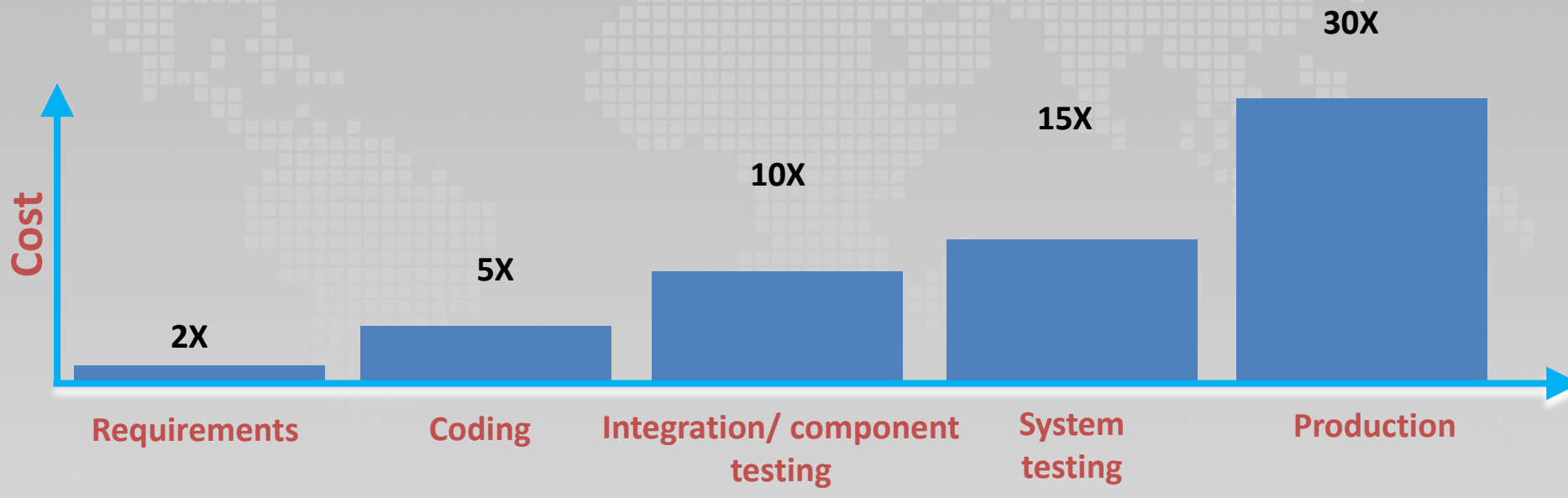
基于Swagger的REST API自动化安全测试实践

贾玉彬 Gary
Gary_jia@rapid7.com

大纲

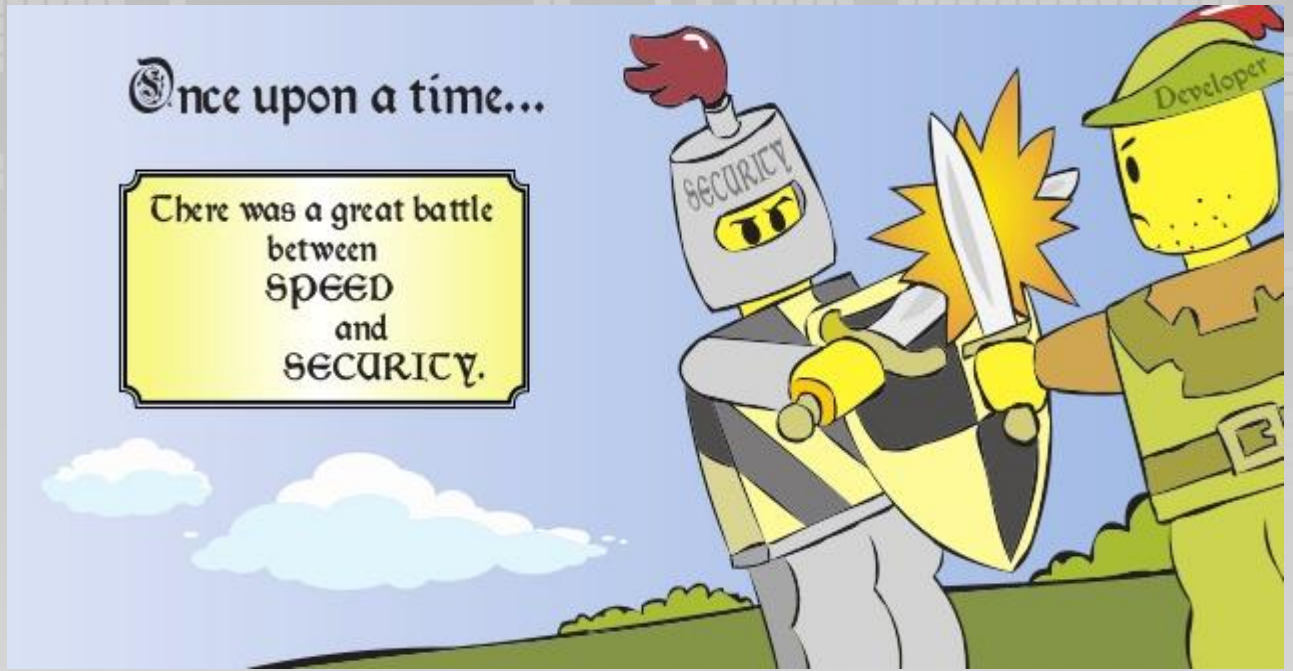
- 由应用安全说起
- 传统REST API安全测试
- 基于Swagger的REST API 的安全测试

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


Once upon a time...
There was a great battle between SPEED and SECURITY.
SECURITY
Developer

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:

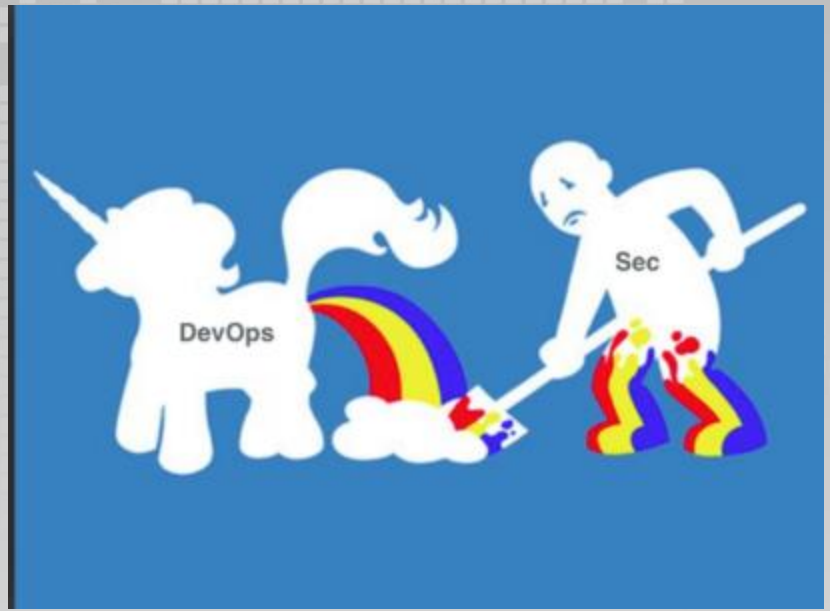


Image : Pete Cheslock at #DevOpsDaysAustin.

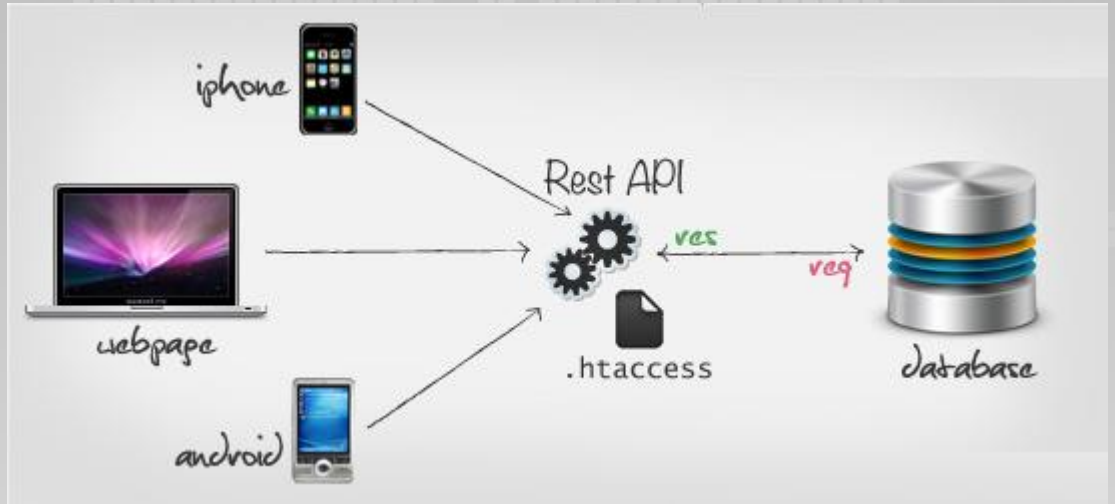
传统的REST API安全测试

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

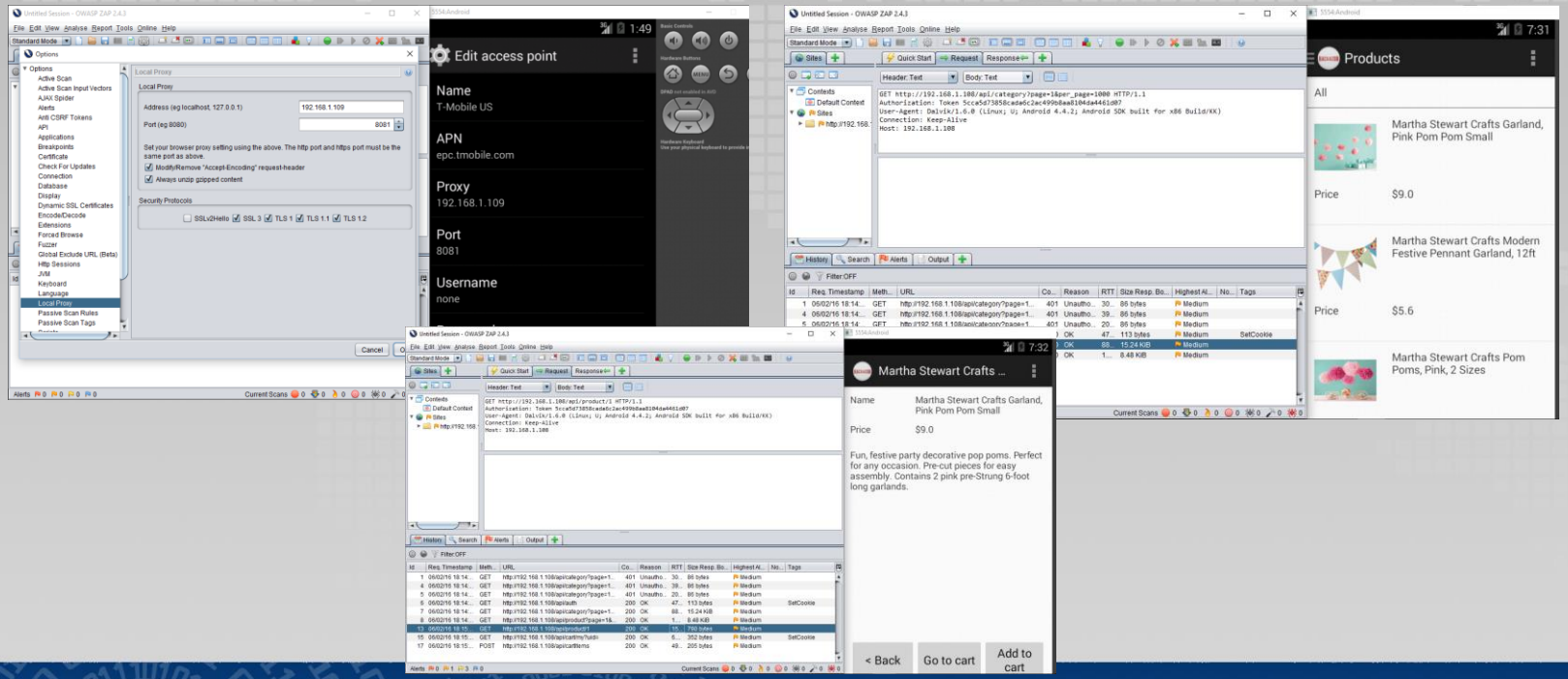
REST API测试

- 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



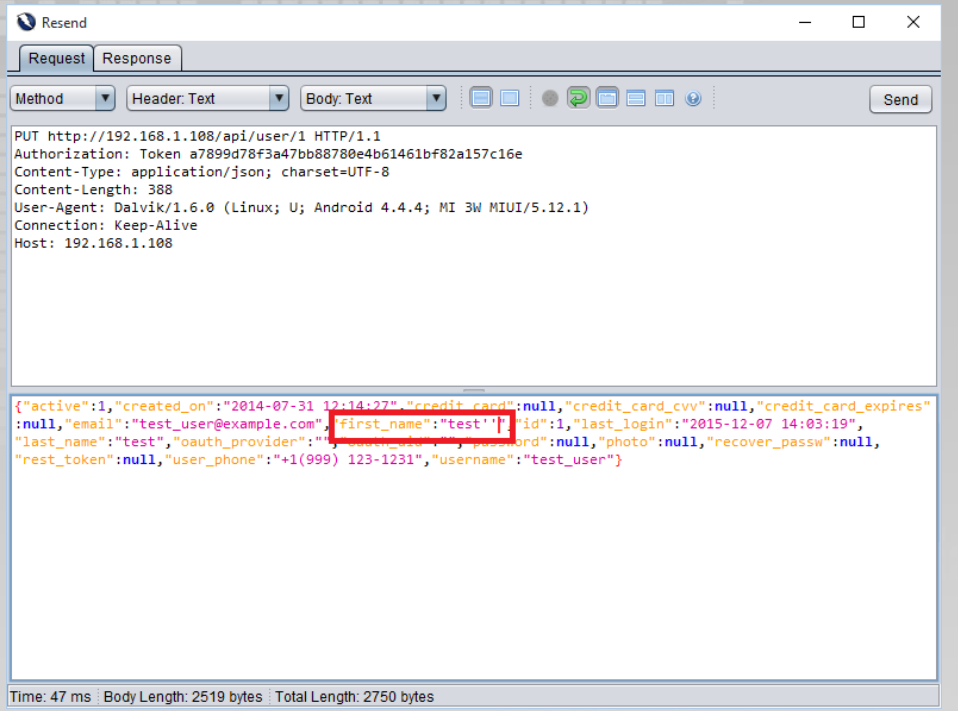
手工测试REST API（SQL注入为例）

- 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''

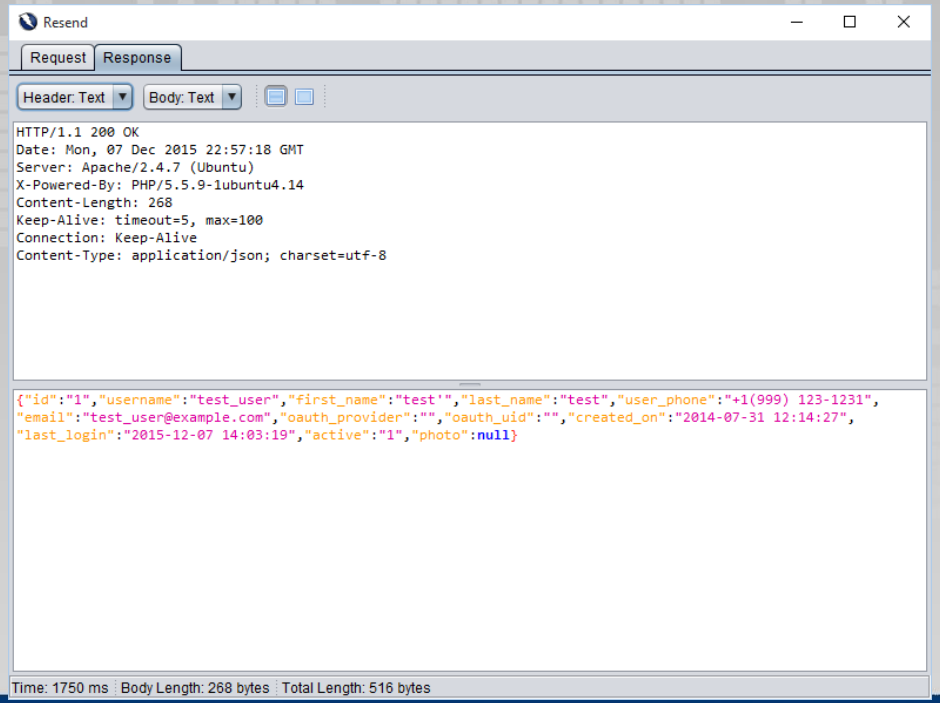
手工测试REST API（SQL注入为例）

Request 1
使用test”把两个单引号（’）注入到请求



手工测试REST API（SQL注入为例）

Response 1
用户档案成功提交



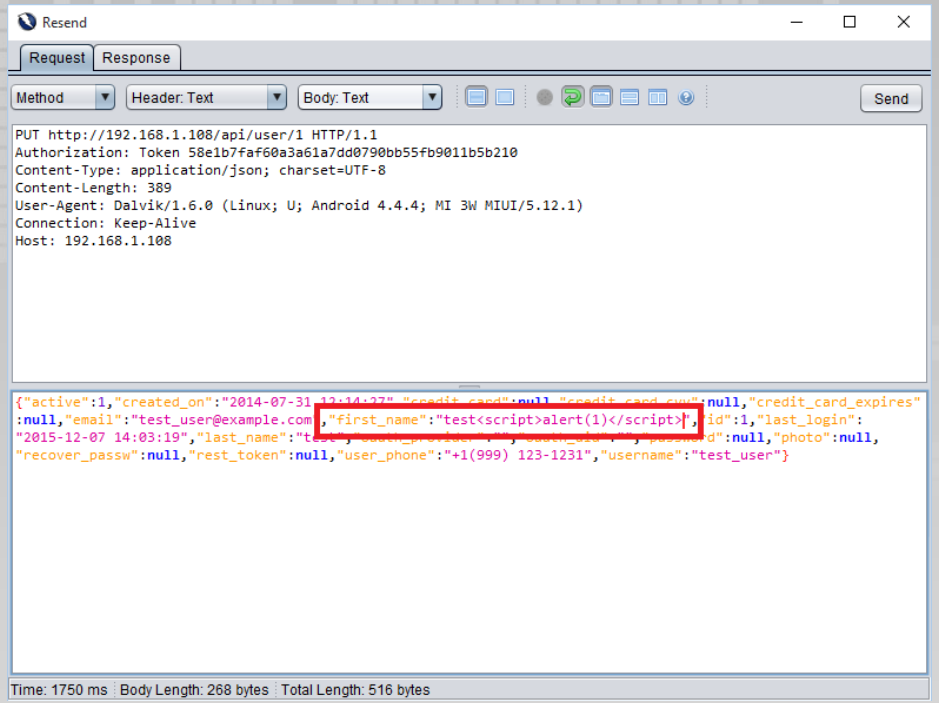
手工测试REST API（XSS为例）

- 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)

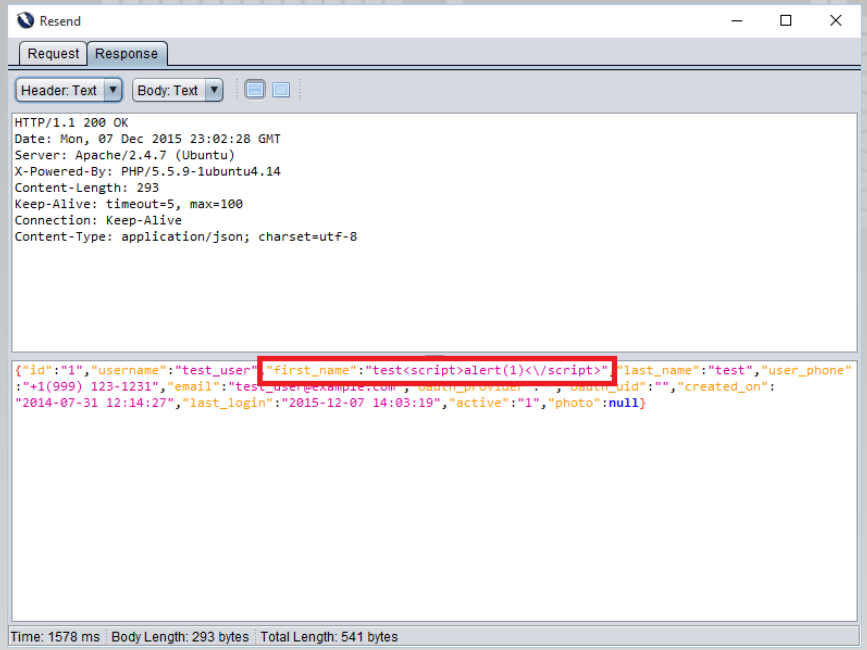
URL: http://192.168.202.131/api/user/1
Method: PUT
Parameter name: first_name
Attack values: <script>alert(1)</script>

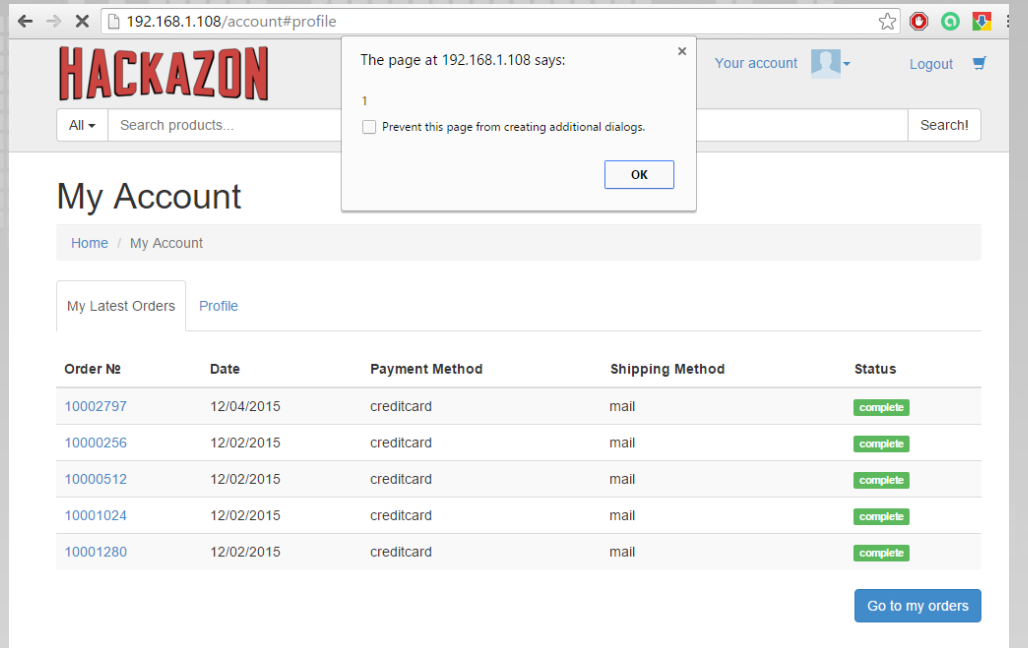
手工测试REST API（XSS为例）

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



手工测试REST API（XSS为例）





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

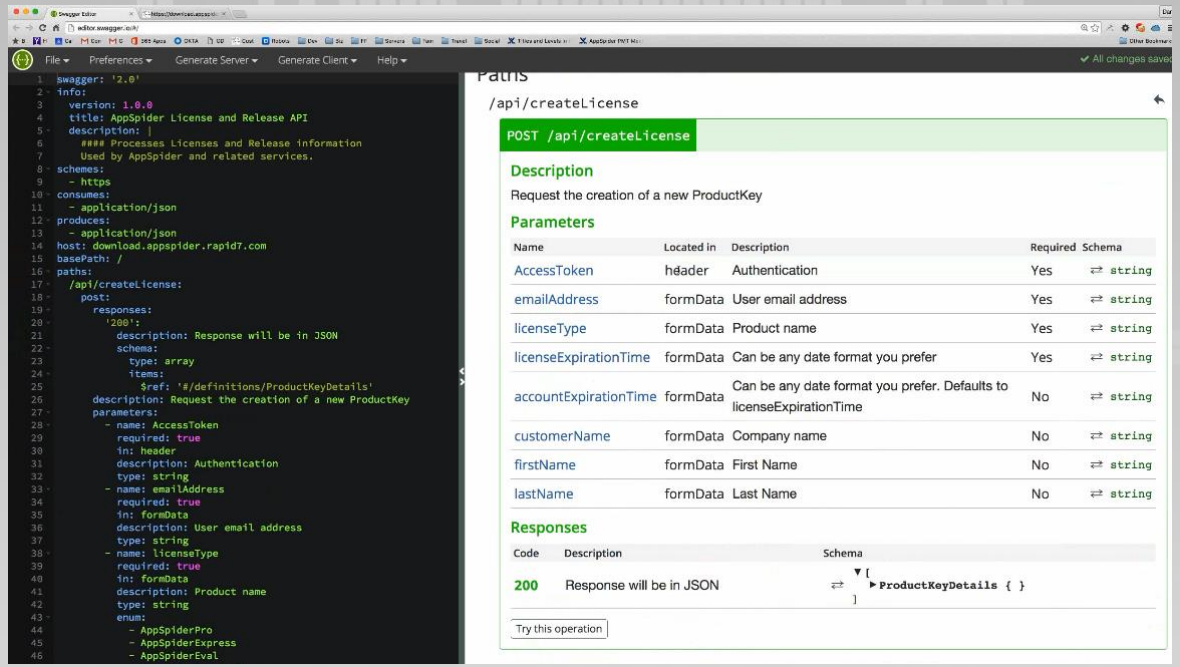
基于SWAGGER的REST API的安全测试

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

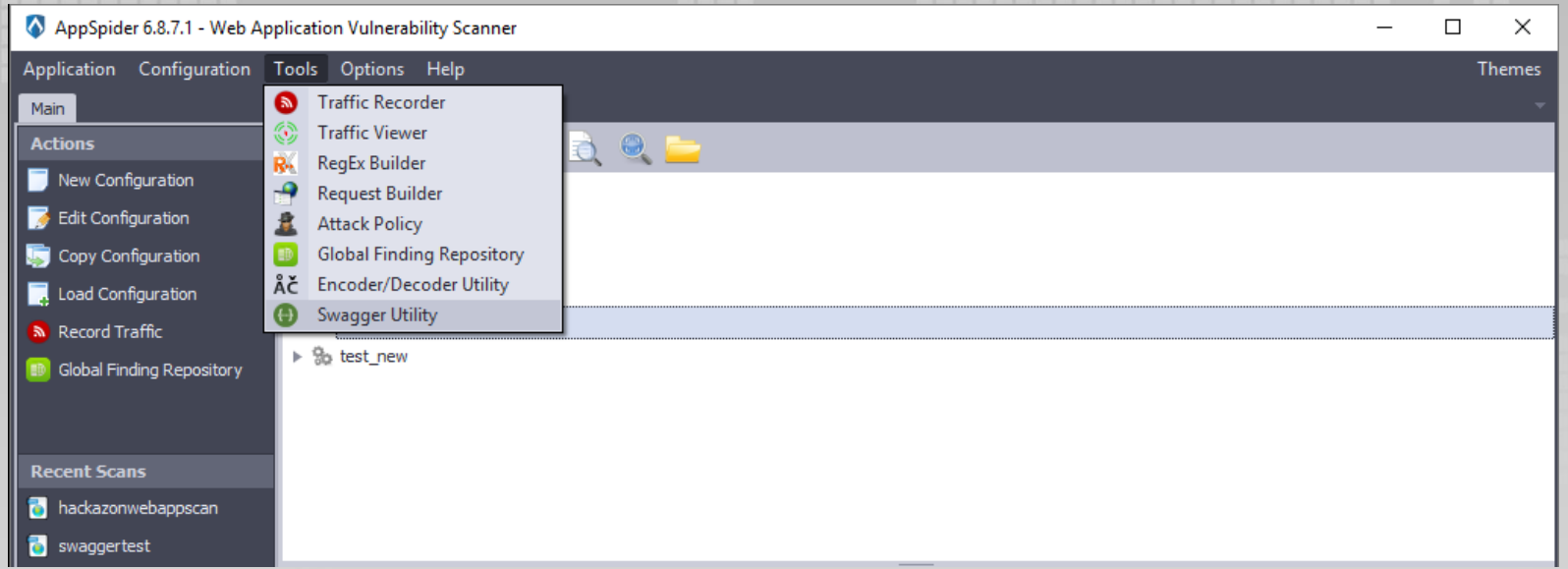
Swagger REST API

```
{
    "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": {
```

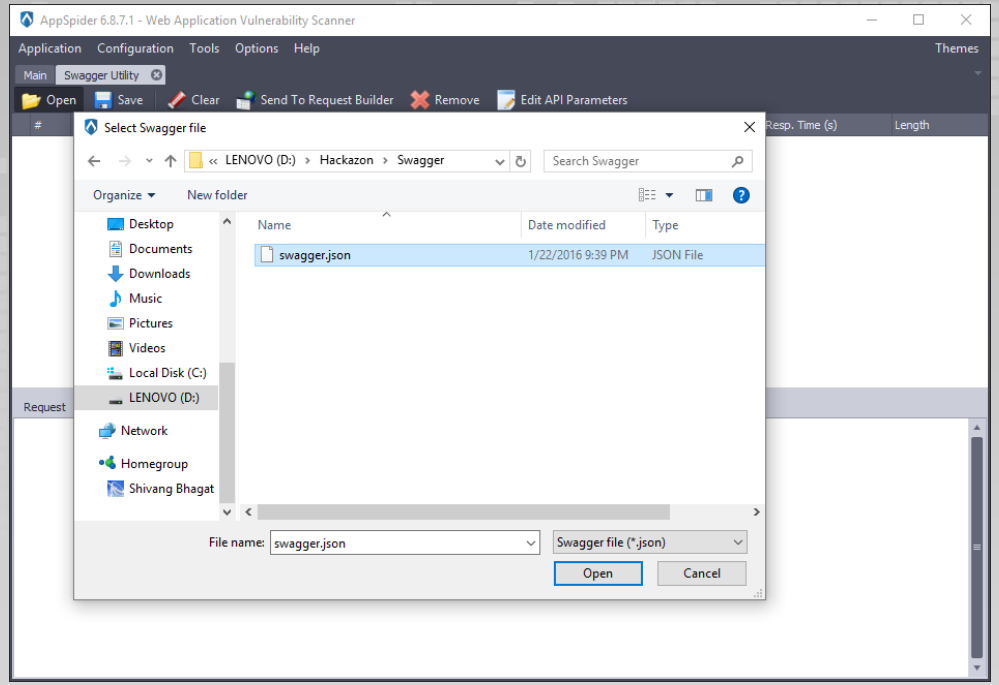


DEMO

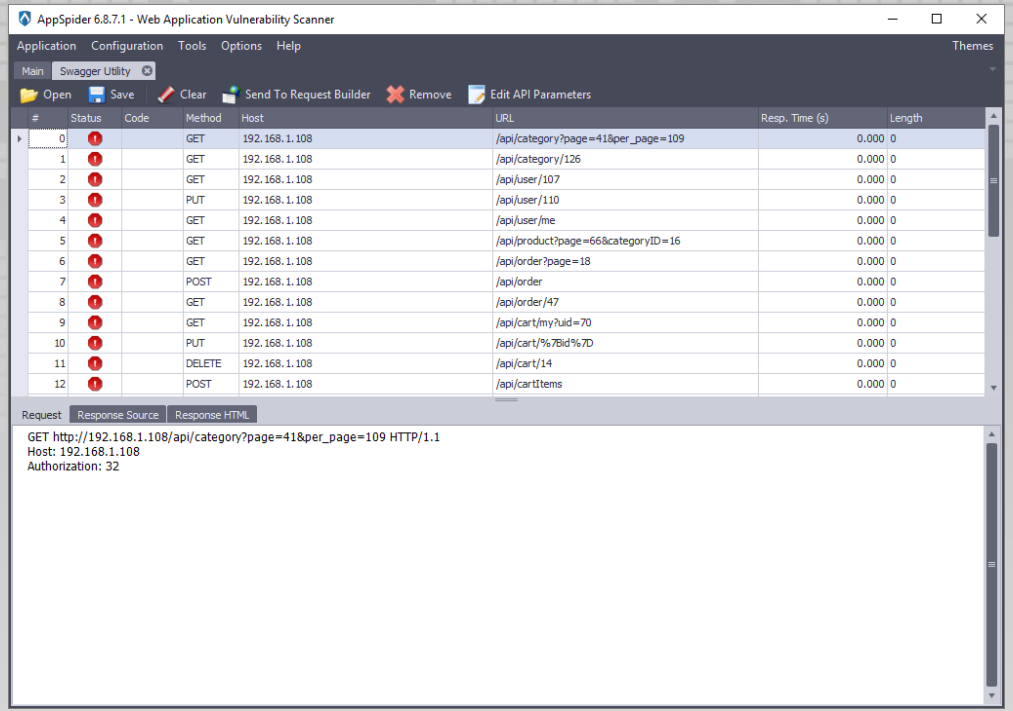
运行AppSpider并选择Swagger Utility



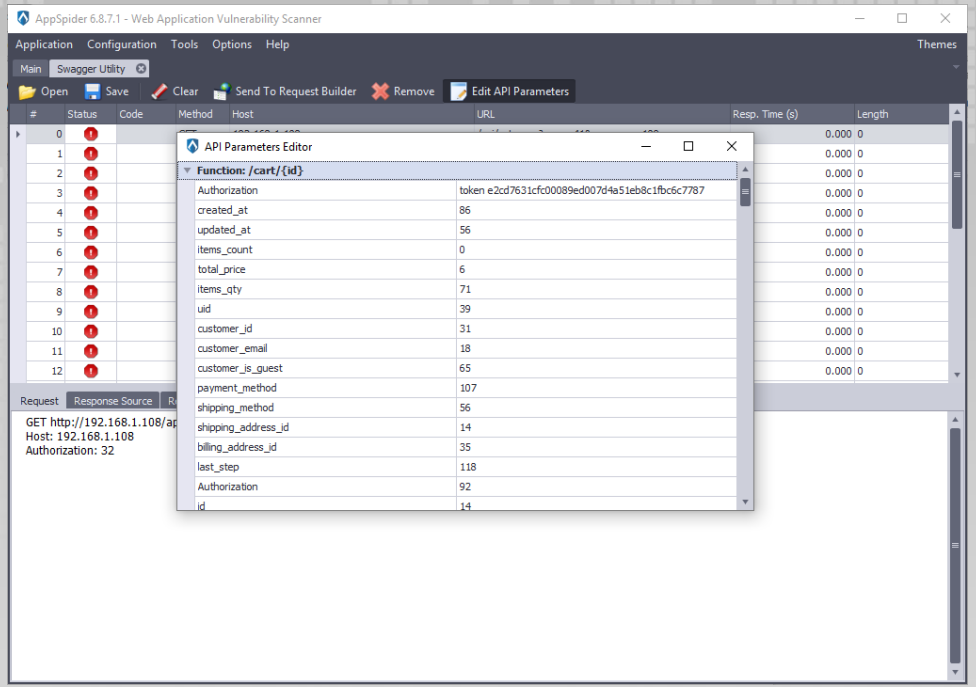
导入Swagger JSON文件



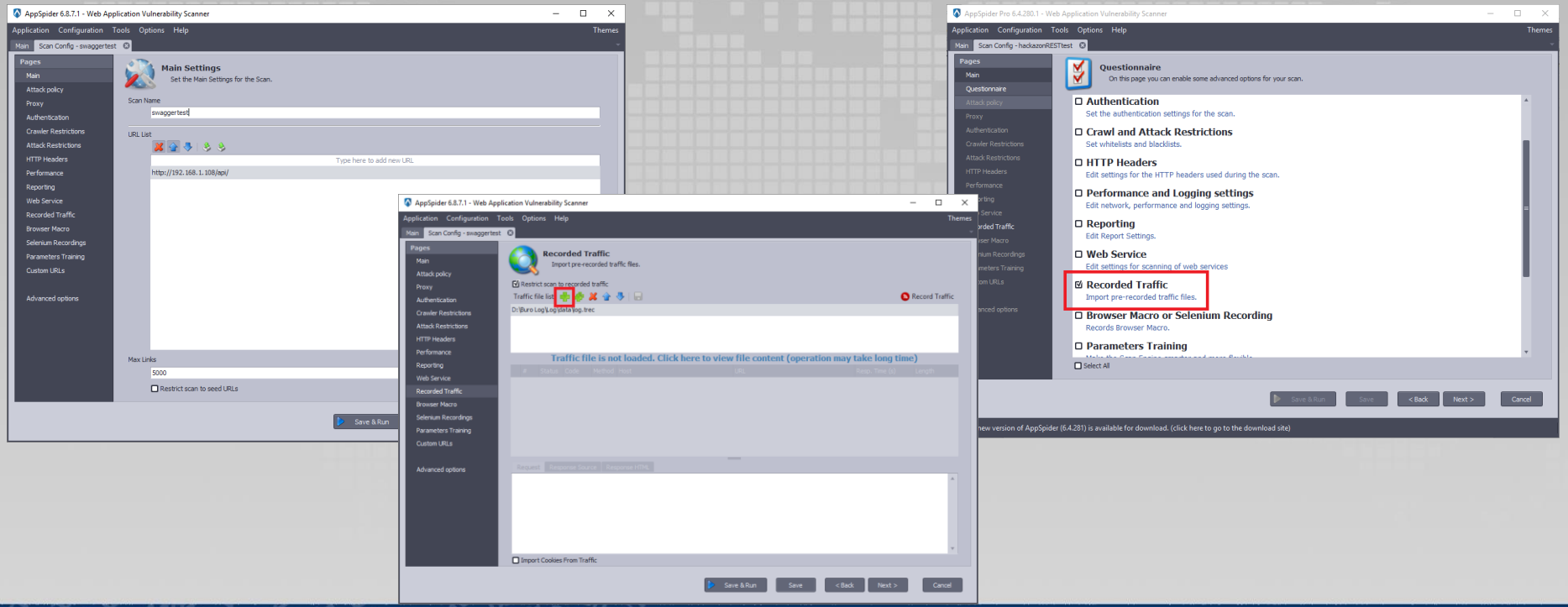
API function calls



操作API参数


AppSpider 6.8.7.1 - Web Application Vulnerability Scanner
Application
Configuration
Tools
Options
Help
Themes
Main
Swagger Utility
Open
Save
Clear
Send To Request Builder
Remove
Edit API Parameters
Status Code Method Host URL Resp. Time (s) Length
0 0.000 0
1 0.000 0
2 0.000 0
3 0.000 0
4 0.000 0
5 0.000 0
6 0.000 0
7 0.000 0
8 0.000 0
9 0.000 0
10 0.000 0
11 0.000 0
12 0.000 0
Request
Response Source
GET http://192.168.1.108/ap
Host: 192.168.1.108
Authorization: 32
API Parameters Editor
Function: /cart/{id}
Authorization token e2cd7631cfc00089ed007d4a51eb8c1fbc6c7787
created_at 86
updated_at 56
items_count 0
total_price 6
items_qty 71
uid 39
customer_id 31
customer_email 18
customer_is_guest 65
payment_method 107
shipping_method 56
shipping_address_id 14
billing_address_id 35
last_step 118
Authorization 92
id 14

创建新的扫描配置



www.webscantest.com/report/

应用

appspider

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

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

Security Status - Partial

Vulnerability	Best Practice	Exposure

Summary

A **partial** scan was peformed.

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

e remediation labor by 55%.

There were issues affecting the scan:

- More than **5%** of the requests failed in this scan. This
of maximum concurrent requests and run the scan ag
- We detected loss of session **1 time.** While it is true tha

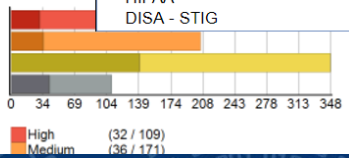
server and significantly compromises the assessment value of this scan. Try setting a lower number (i.e. half to two-thirds of the current value)

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

Vulnerabilities

Vulnerab

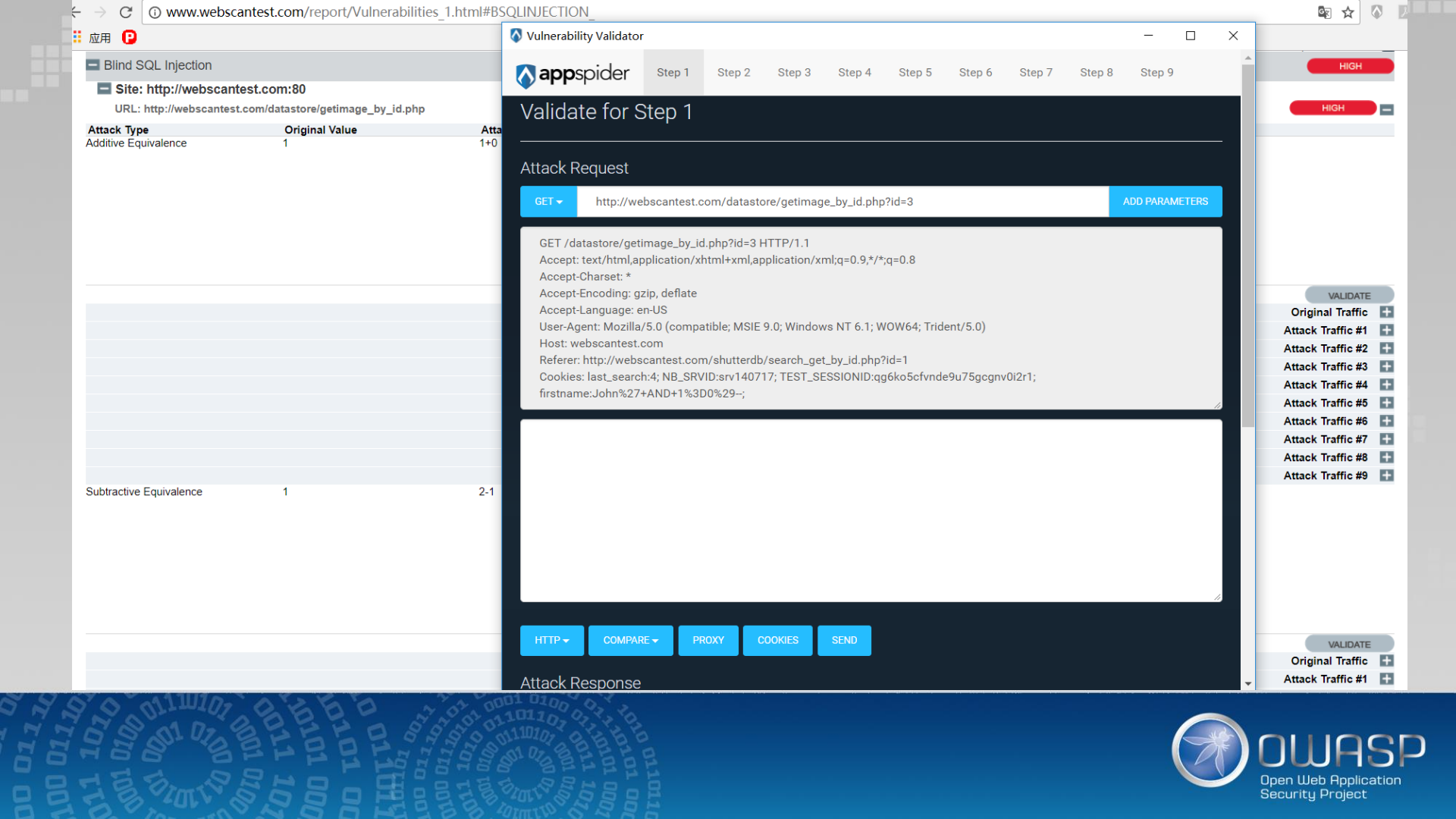
Variances: 556
Root Causes: 250



Vulnerability Reports

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

OWASP Open Web Application Security Project


www.webscantest.com/report/Vulnerabilities_1.html#BSQLINJECTION_
应用
Blind SQL Injection
Site: http://webscantest.com:80
URL: http://webscantest.com/datastore/getimage_by_id.php
Attack Type
Original Value
Additive Equivalence
1
1+0
Subtractive Equivalence
1
2-1
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
Vulnerability Validator
appspider
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:qg6ko5cfvnde9u75gcgnv0i2r1;
firstname:John%27+AND+1%3D0%29--;
HTTP
COMPARE
PROXY
COOKIES
SEND
Attack Response
OWASP
Open Web Application
Security Project

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