RAPID7

Hackazon

User's Guide

Contents

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.

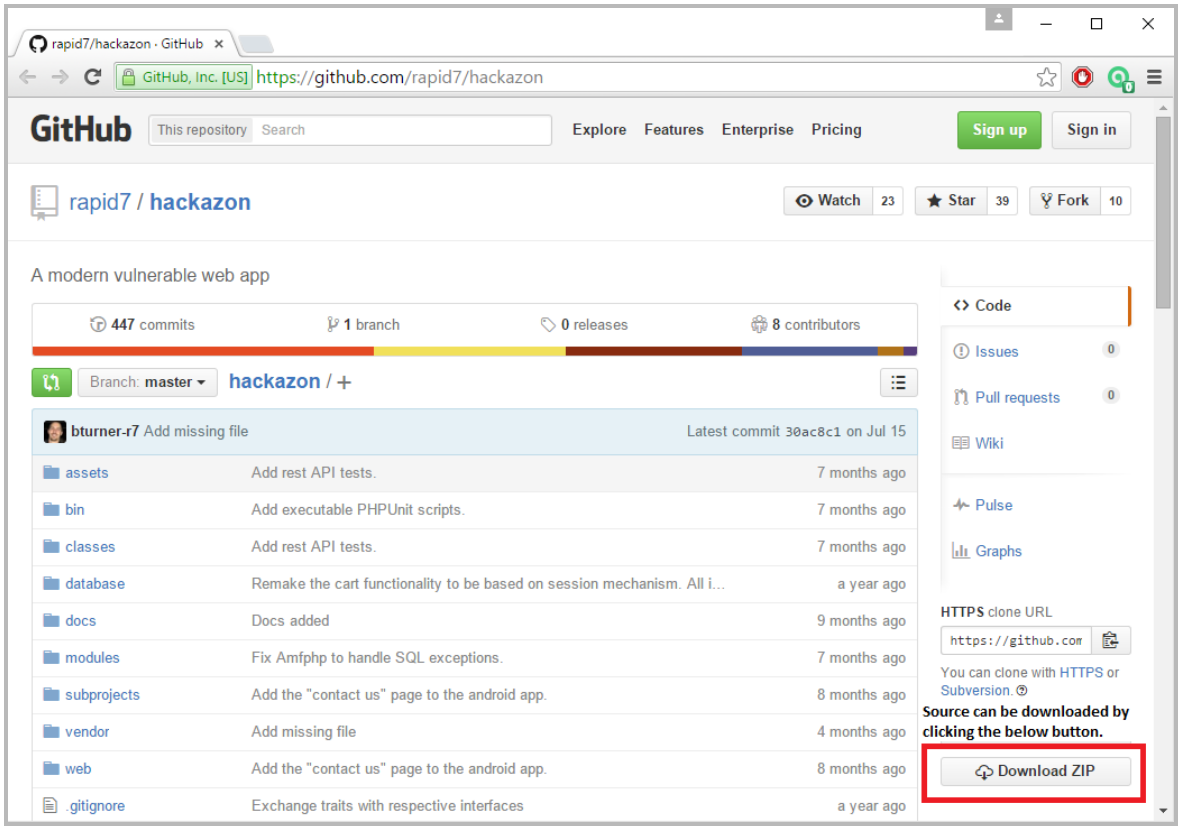
Disclaimer: Hackazon is riddled with vulnerabilities by design. Use of Hackazon can cause system compromise and Rapid7 accepts no liability for the same. We strongly advise users not to use the application on production systems. Any download, installation, or use of Hackazon is entirely at the user’s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

Hackazon setup for a Windows machine

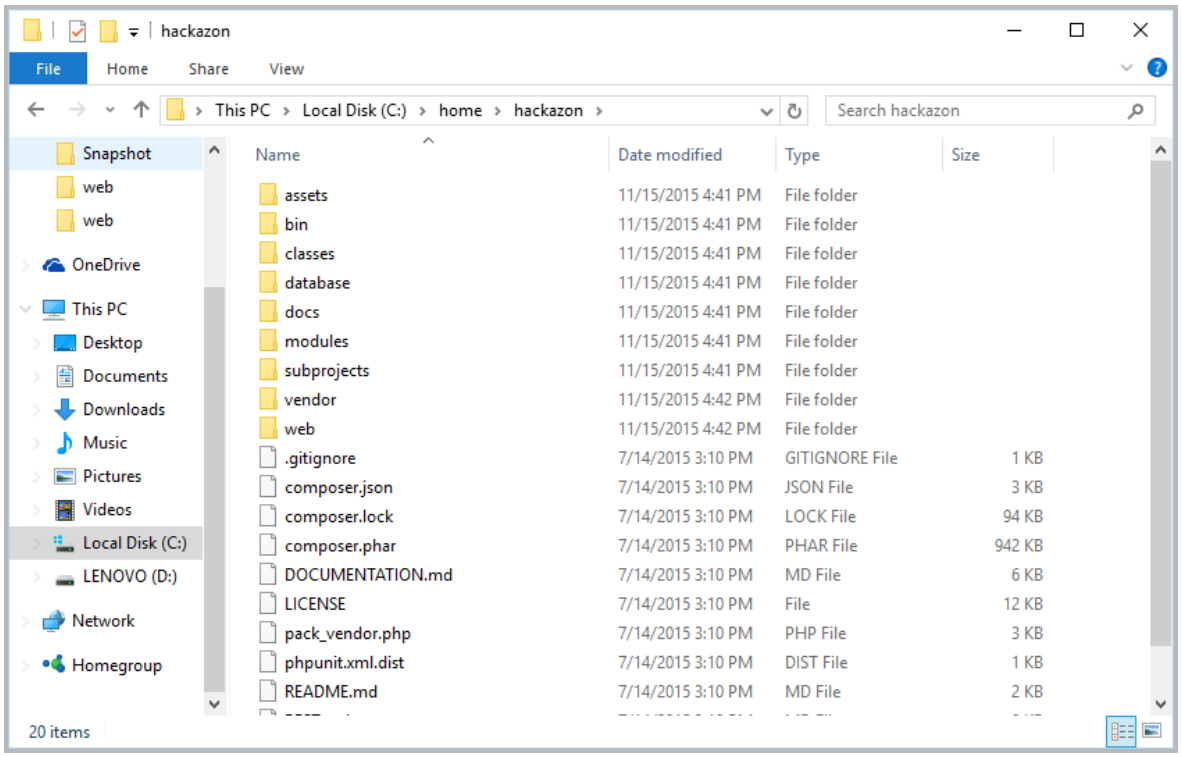
Hackazon is available on the Rapid7 GitHub page and can be downloaded from the following link:

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\.

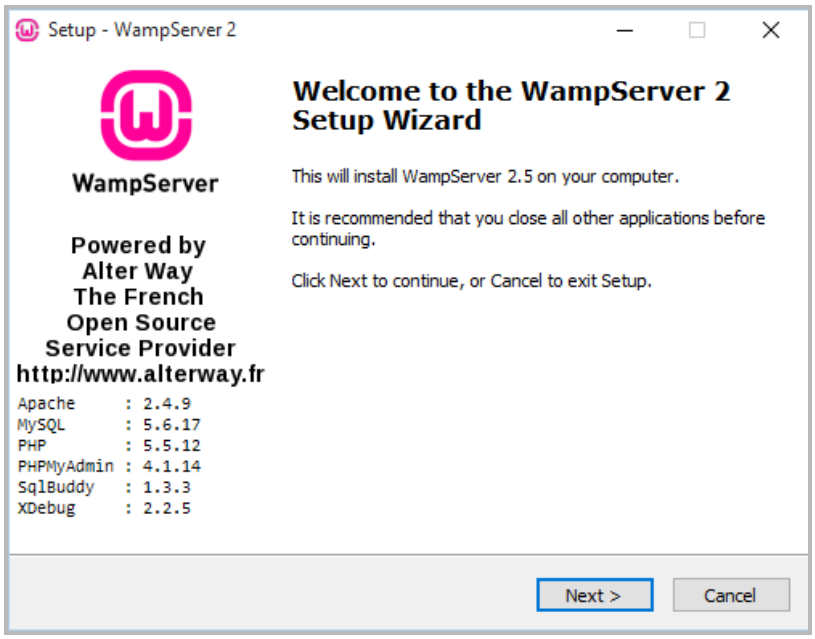


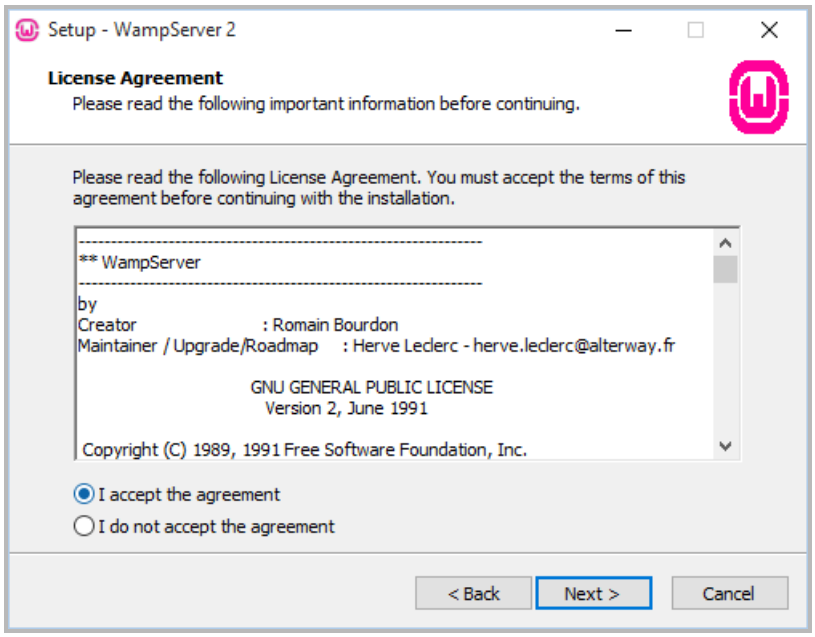
WampServer setup

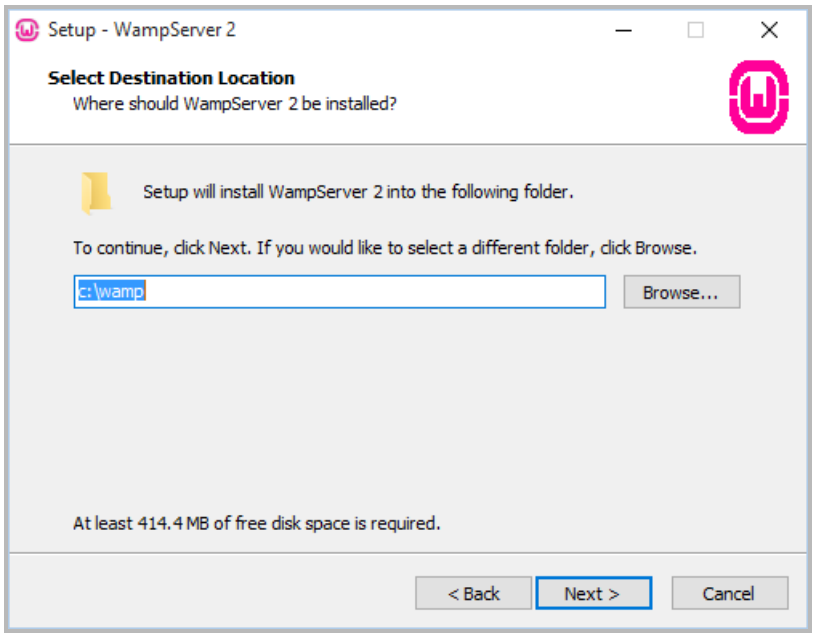
Hackazon is a PHP web application and requires PHP framework, an Apache server, and a MySQL database. For an all-in-one, Windows web development environment, you can use WampServer. It allows you to create web applications with PHP framework, an Apache server, and a MySQL database. WampServer can be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

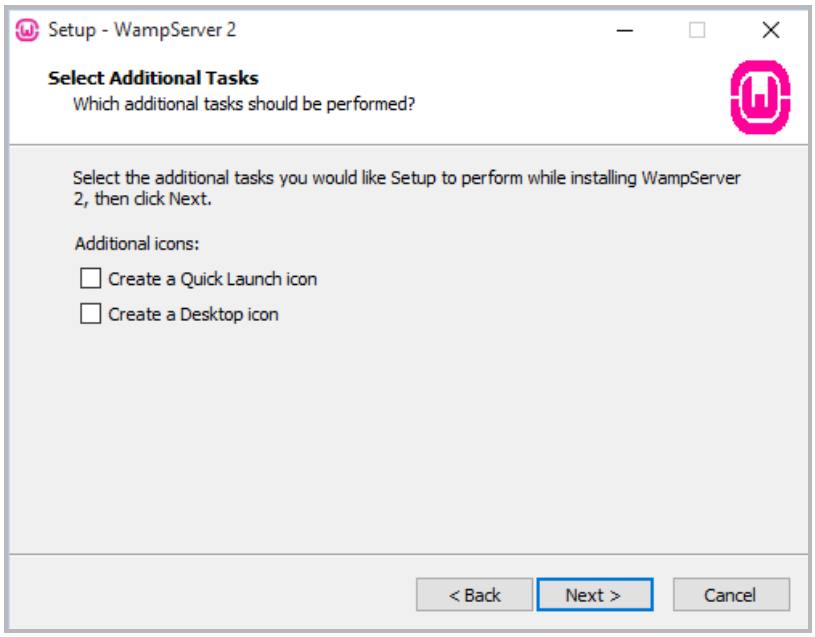
WampServer: http://www.wampserver.com/en/download-wampserver-64bits/

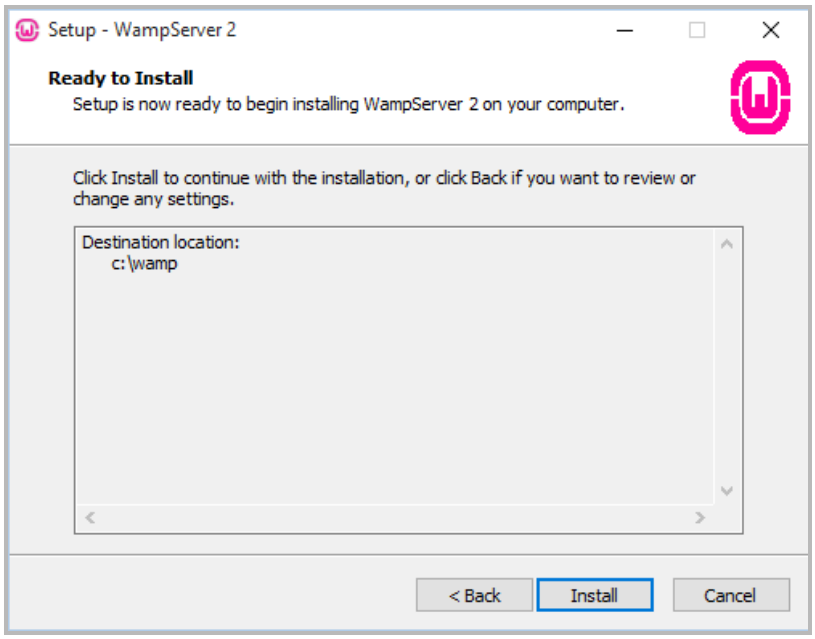
1. Complete WampServer Setup Wizard.

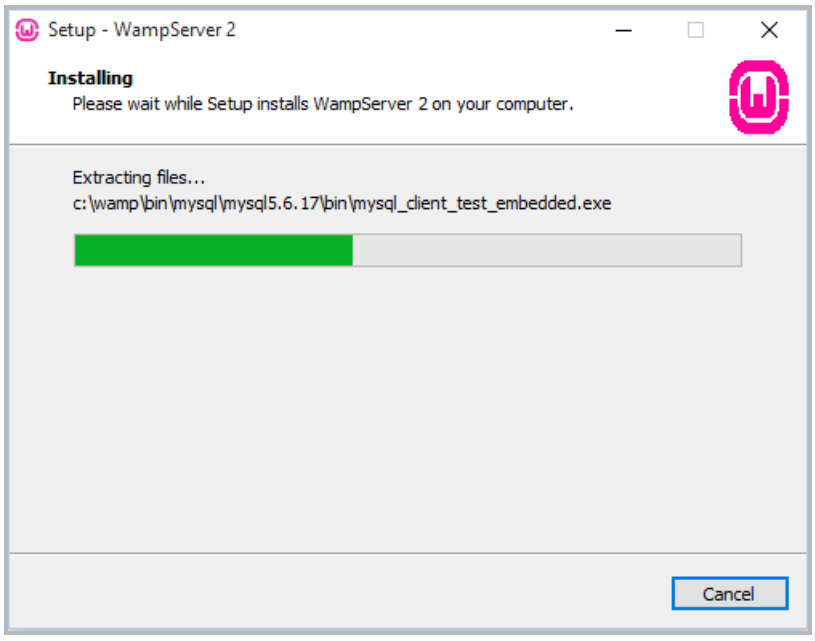

Setup - WampServer 2
WampServer
Powered by
Alter Way
The French
Open Source
Service Provider
http://www.alterway.fr
Apache : 2.4.9
MySQL : 5.6.17
PHP : 5.5.12
PHPMyAdmin : 4.1.14
SqlBuddy : 1.3.3
XDebug : 2.2.5
Welcome to the WampServer 2 Setup Wizard
This will install WampServer 2.5 on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit Setup.
Next >
Cancel

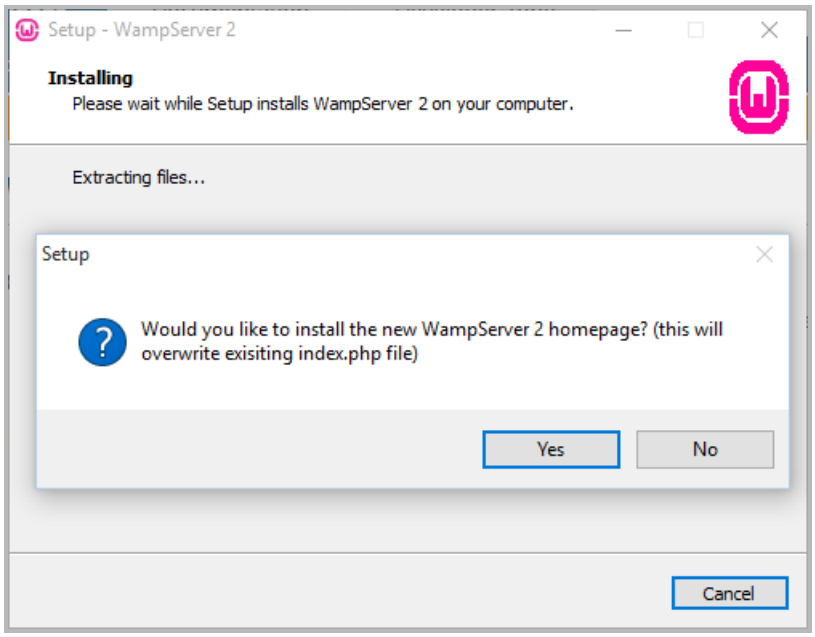

Setup - WampServer 2
License Agreement
Please read the following important information before continuing.
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.
** WampServer
by
Creator : Romain Bourdon
Maintainer / Upgrade/Roadmap : Herve Leclerc - herve.leclerc@alterway.fr
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
I accept the agreement
I do not accept the agreement
< Back
Next >
Cancel


Setup - WampServer 2
Select Destination Location
Where should WampServer 2 be installed?
Setup will install WampServer 2 into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
c:\wamp
Browse...
At least 414.4 MB of free disk space is required.
< Back
Next >
Cancel

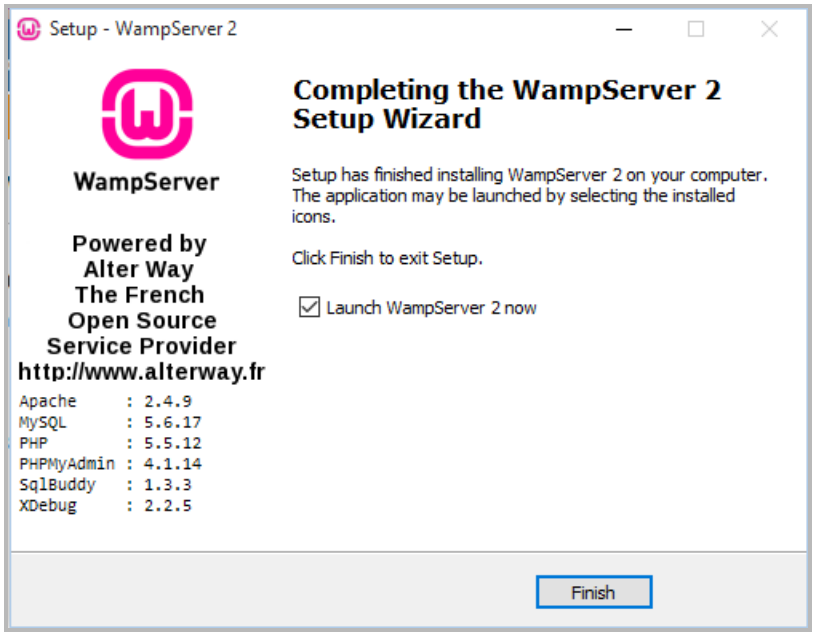

Setup - WampServer 2
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing WampServer 2, then click Next.
Additional icons:
Create a Quick Launch icon
Create a Desktop icon
< Back
Next >
Cancel


Setup - WampServer 2
Ready to Install
Setup is now ready to begin installing WampServer 2 on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
c:\wamp
< Back
Install
Cancel

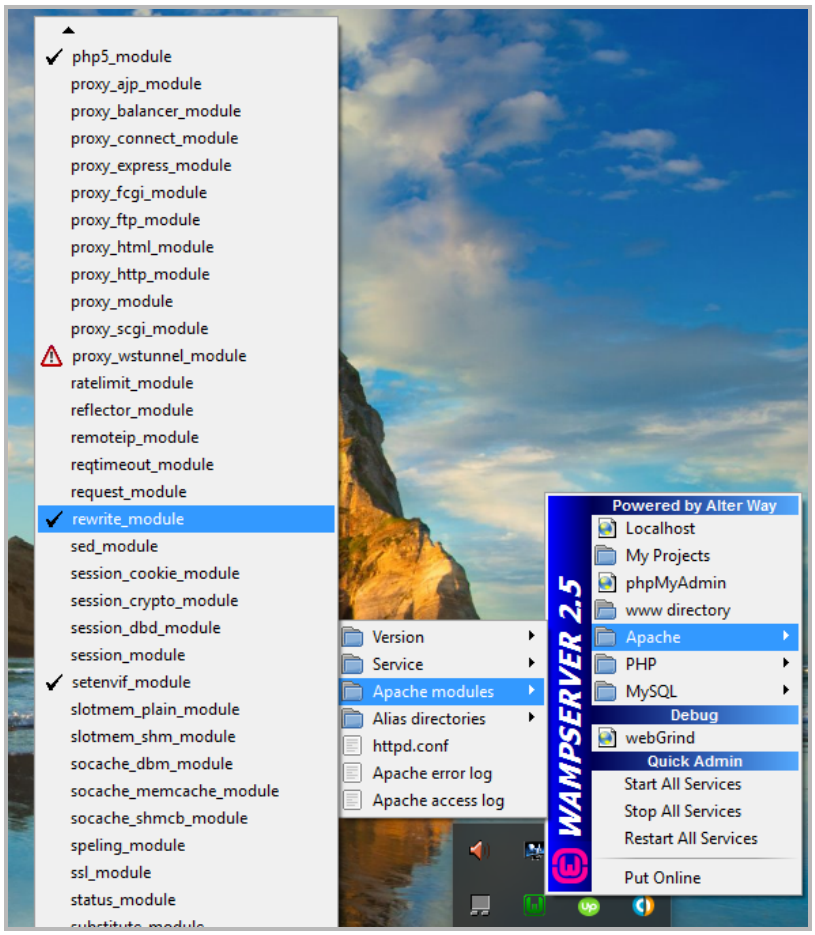

Setup - WampServer 2
Installing
Please wait while Setup installs WampServer 2 on your computer.
Extracting files...
c:\wamp\bin\mysql\mysql5.6.17\bin\mysql_client_test_embedded.exe
Cancel



2. Launch WampServer.



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



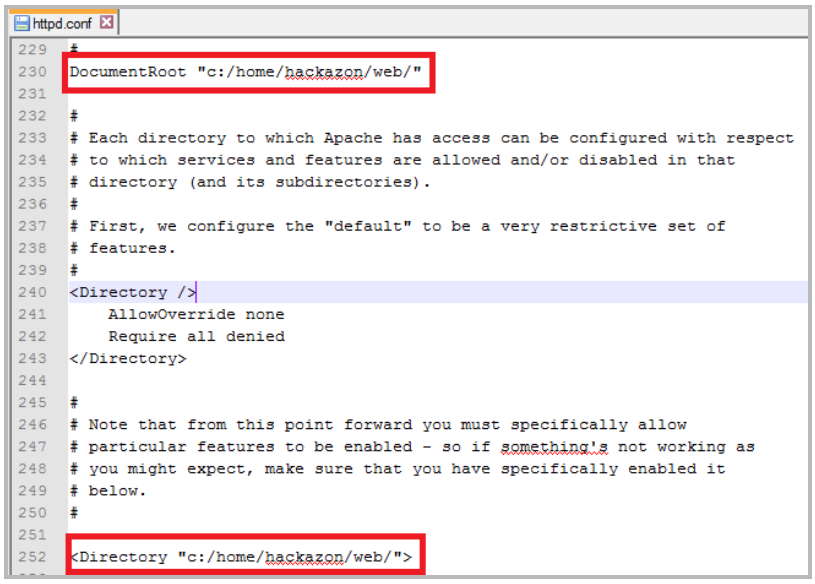
Modify the file, C:\wamp\bin\apache\apache2.4.9\conf\httpd.conf:

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

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

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

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



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:

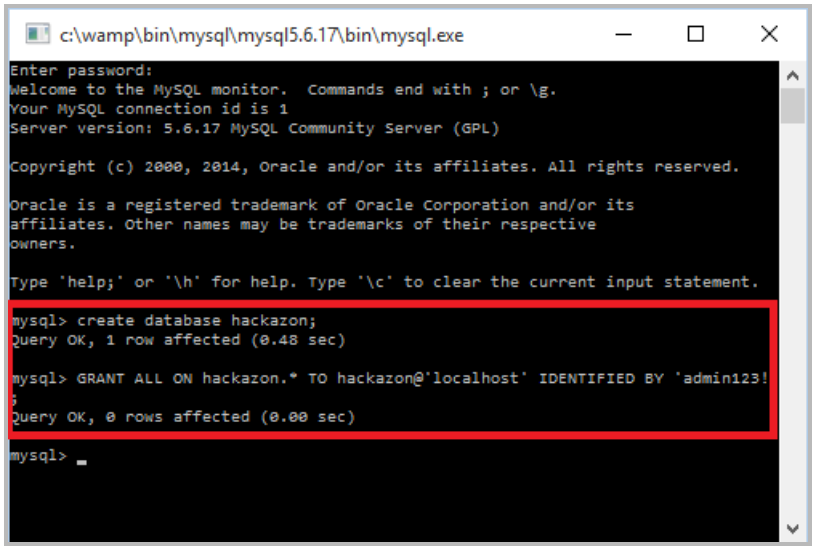
```
create database hackazon;
```

12. Assign database credentials:

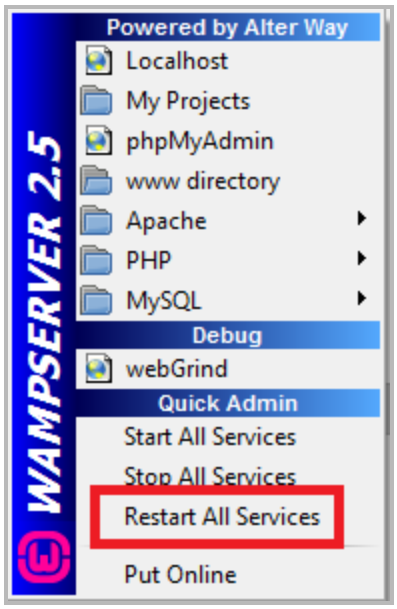
```
GRANT ALL ON hackazon.* TO hackazon@'localhost' IDENTIFIED BY
'InsertPasswordHere';
```

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.



12. Navigate to **WampServer -> Restart All Services**.

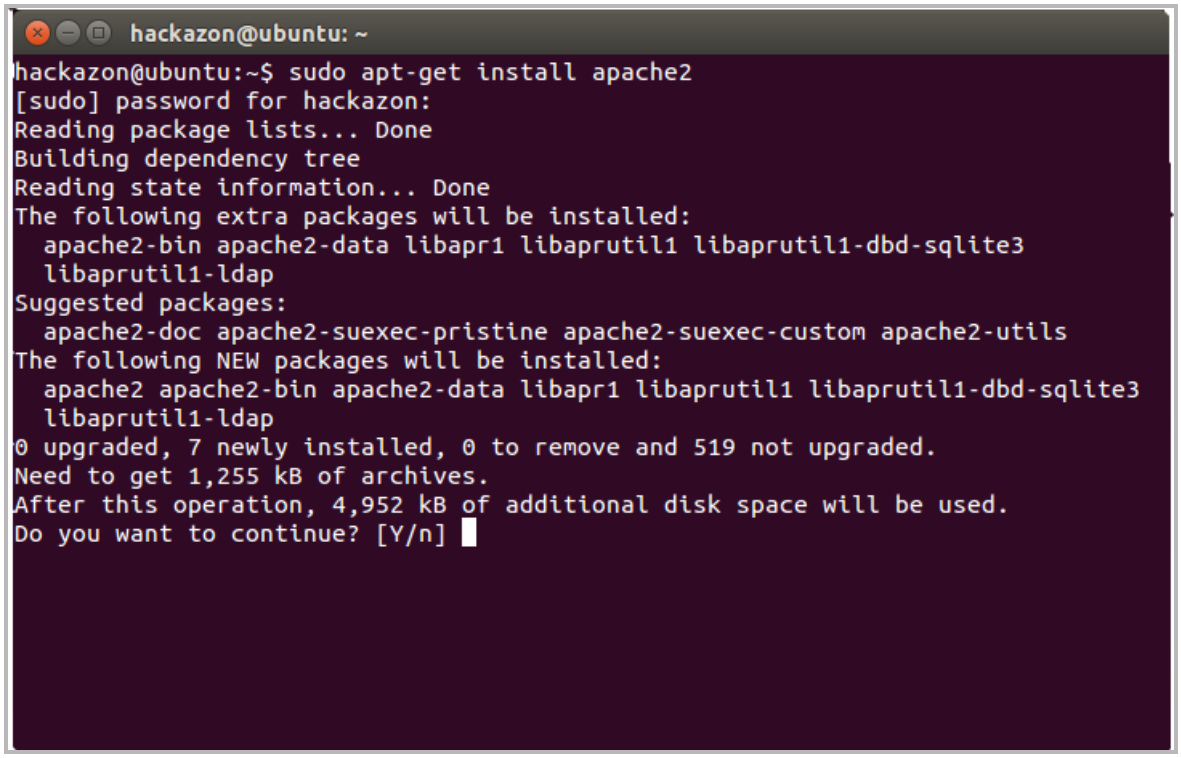


13. Proceed to the Hackazon installation wizard to continue.

Hackazon setup for a Linux (Ubuntu) machine

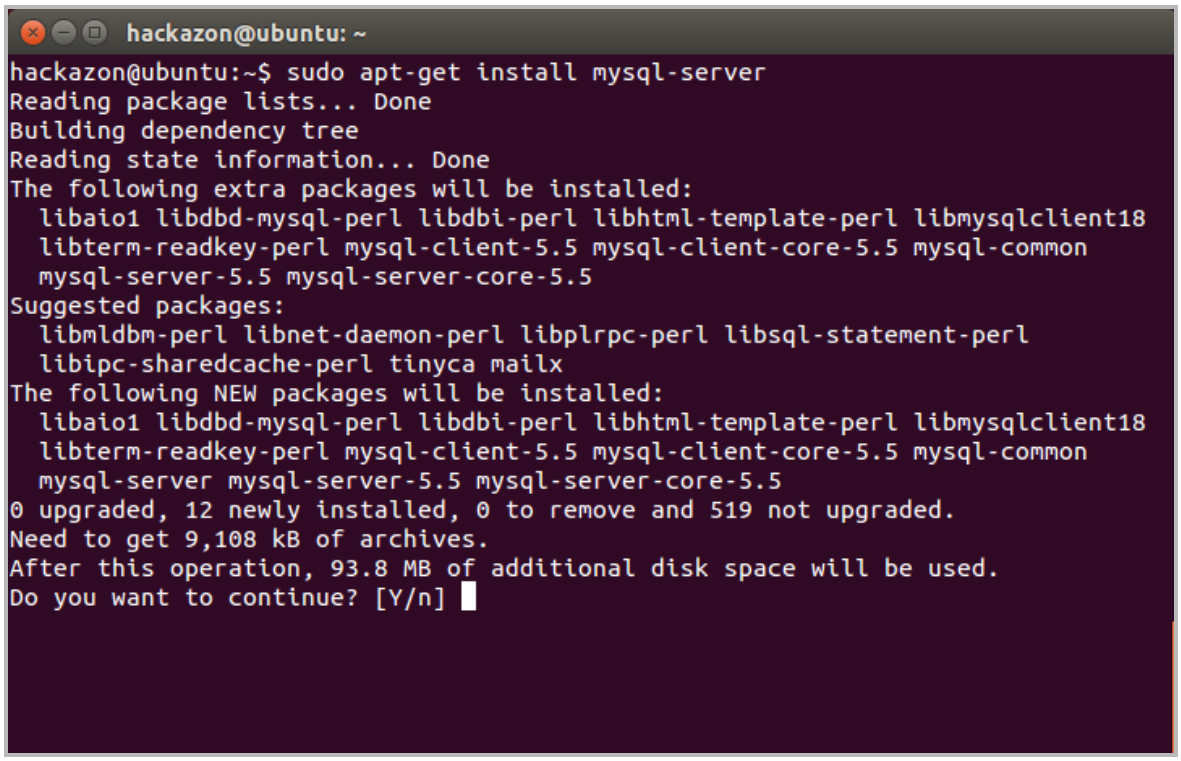
1. Hold **Ctrl + Alt + T** on your keyboard to open a terminal and enter the following commands.
2. Install Apache server:

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

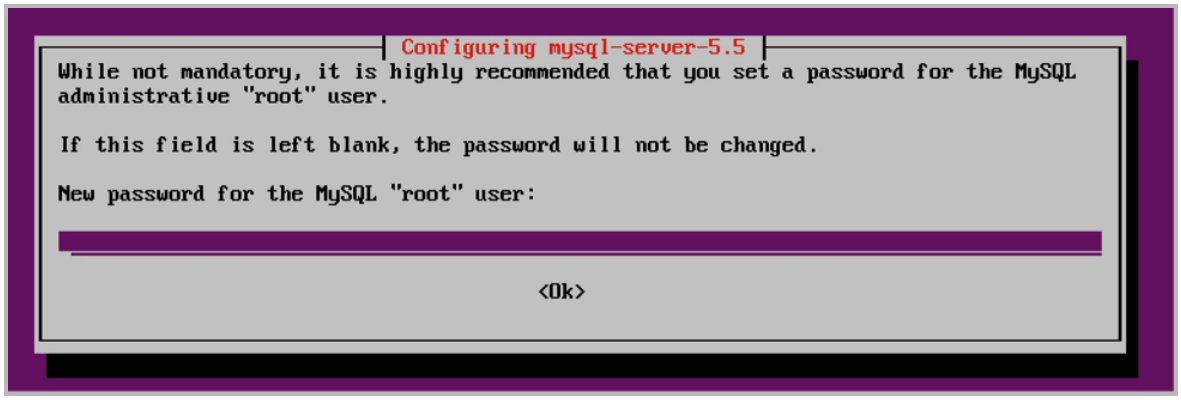


3. Install MySQL database server:

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

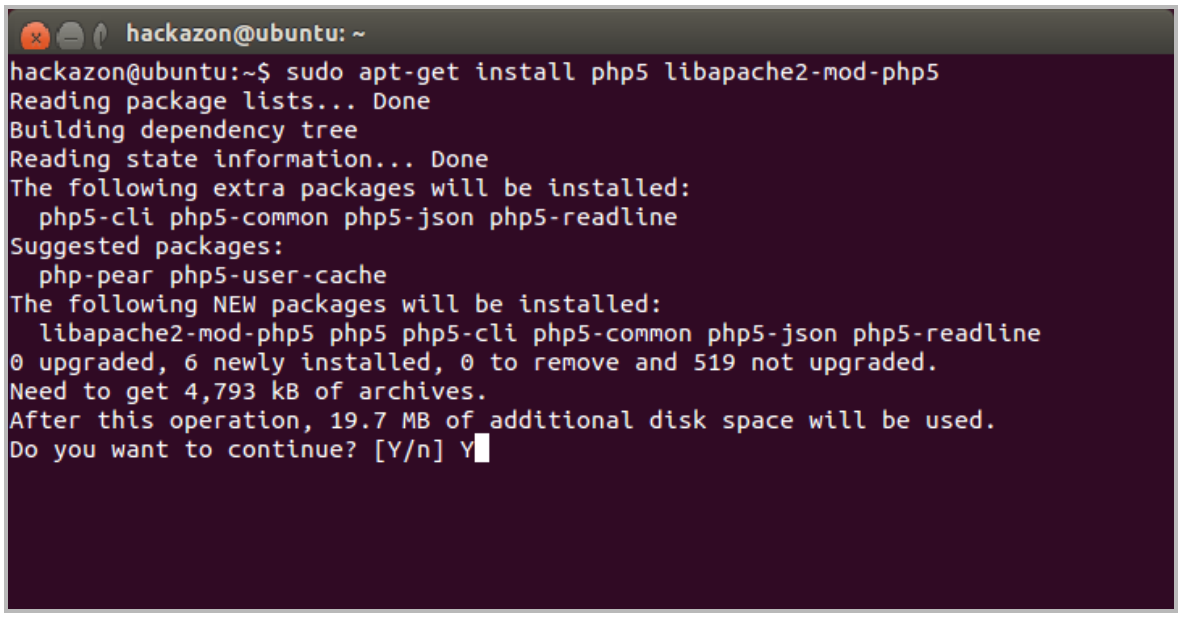


4. Give the MySQL root user a password.



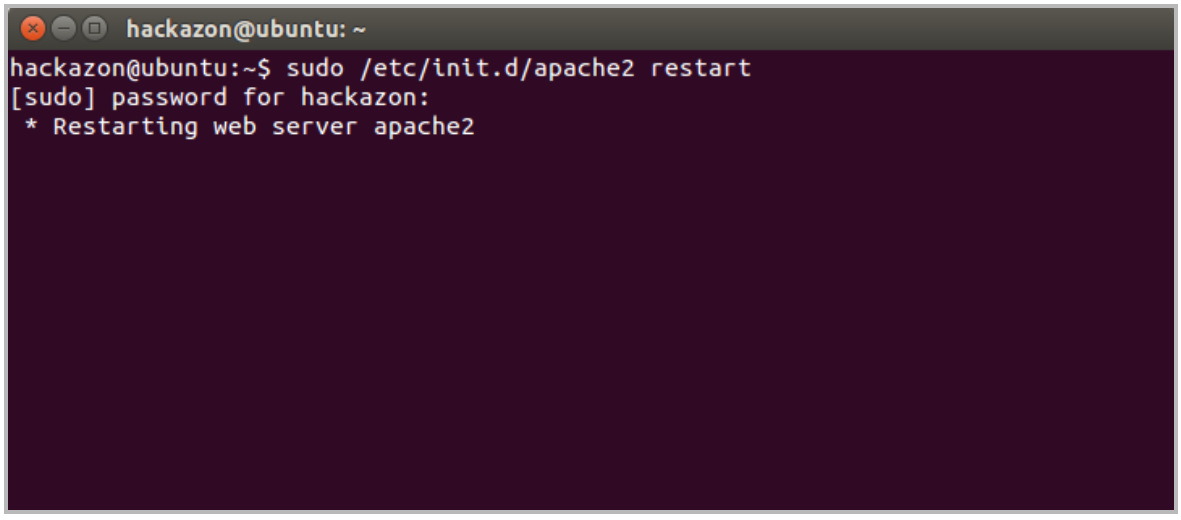
5. Install PHP framework:

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



6. Restart the apache server:

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



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:

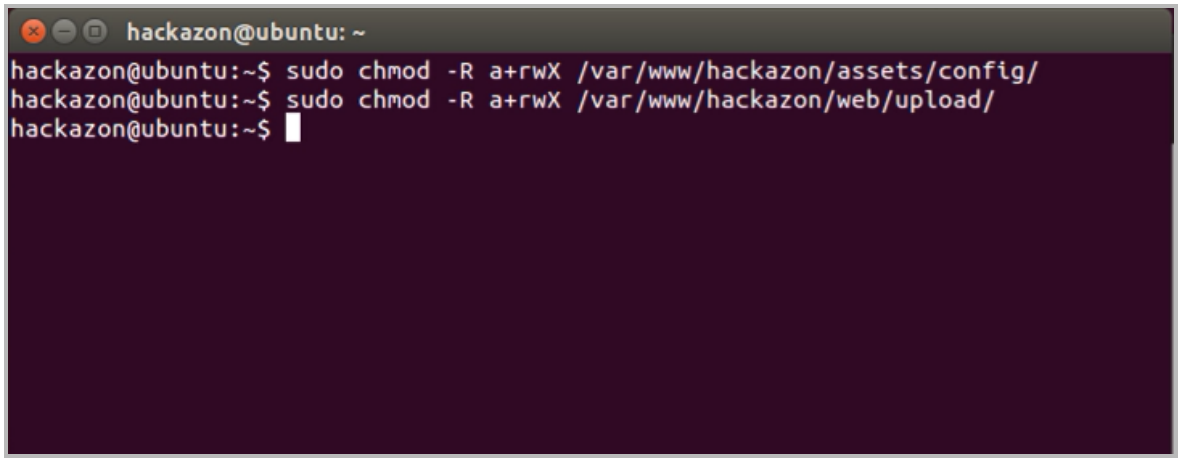
```
sudo unzip master.zip
```

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/
```



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

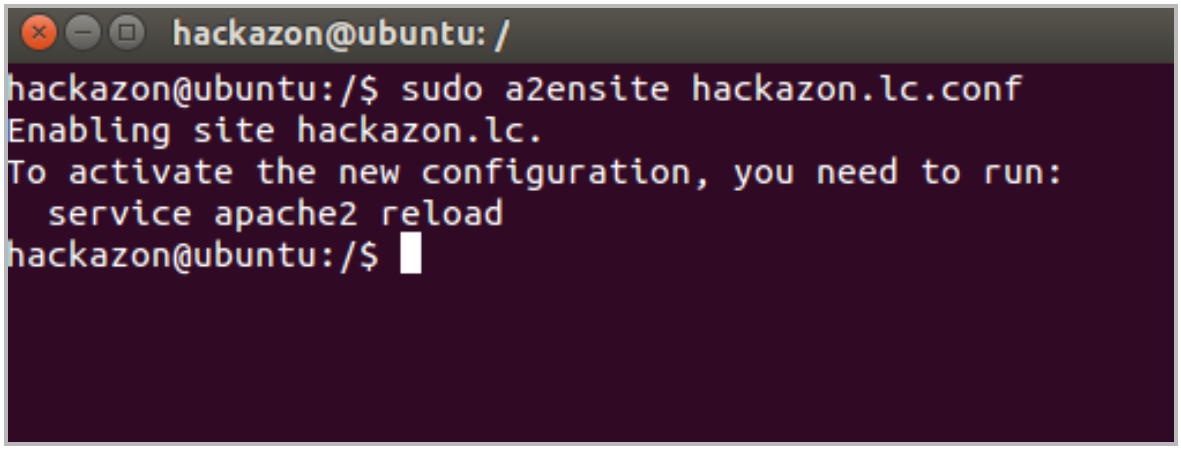
```
sudo gedit
```

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>
```

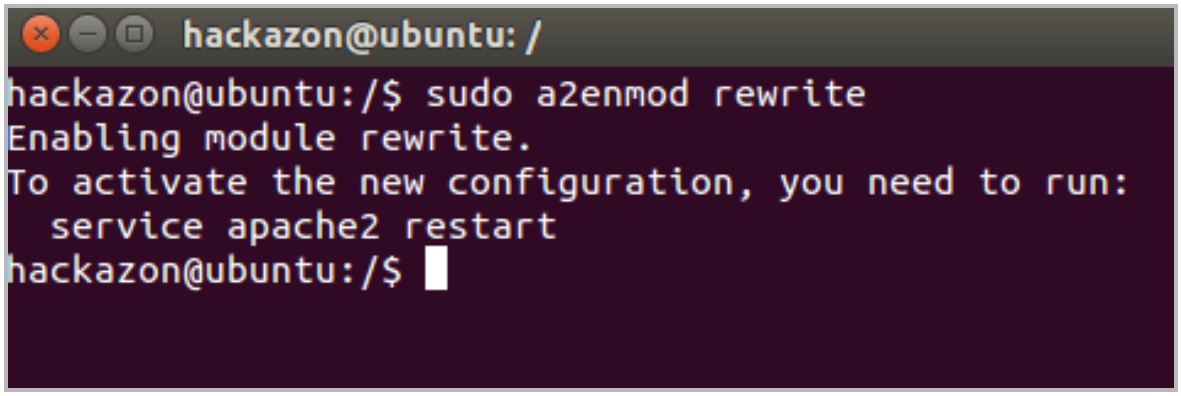
15. Save file as hackazon.lc.conf in /etc/apache2/sites-available/
16. Enable the newly created site hackazon.lc.conf:

```
sudo a2ensite hackazon.lc.conf
```



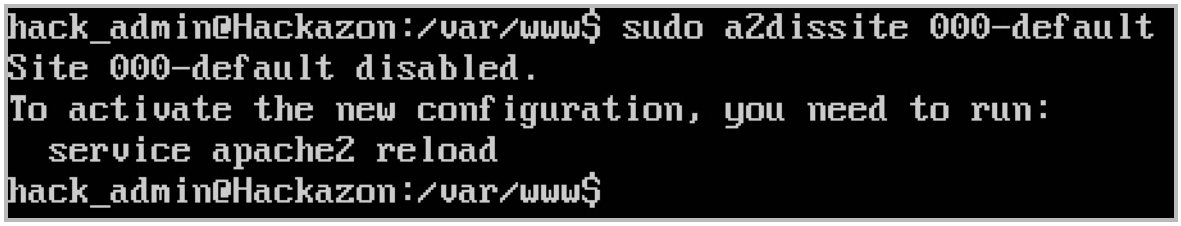
17. Enable Apache rewrite module:

```
sudo a2enmod rewrite
```



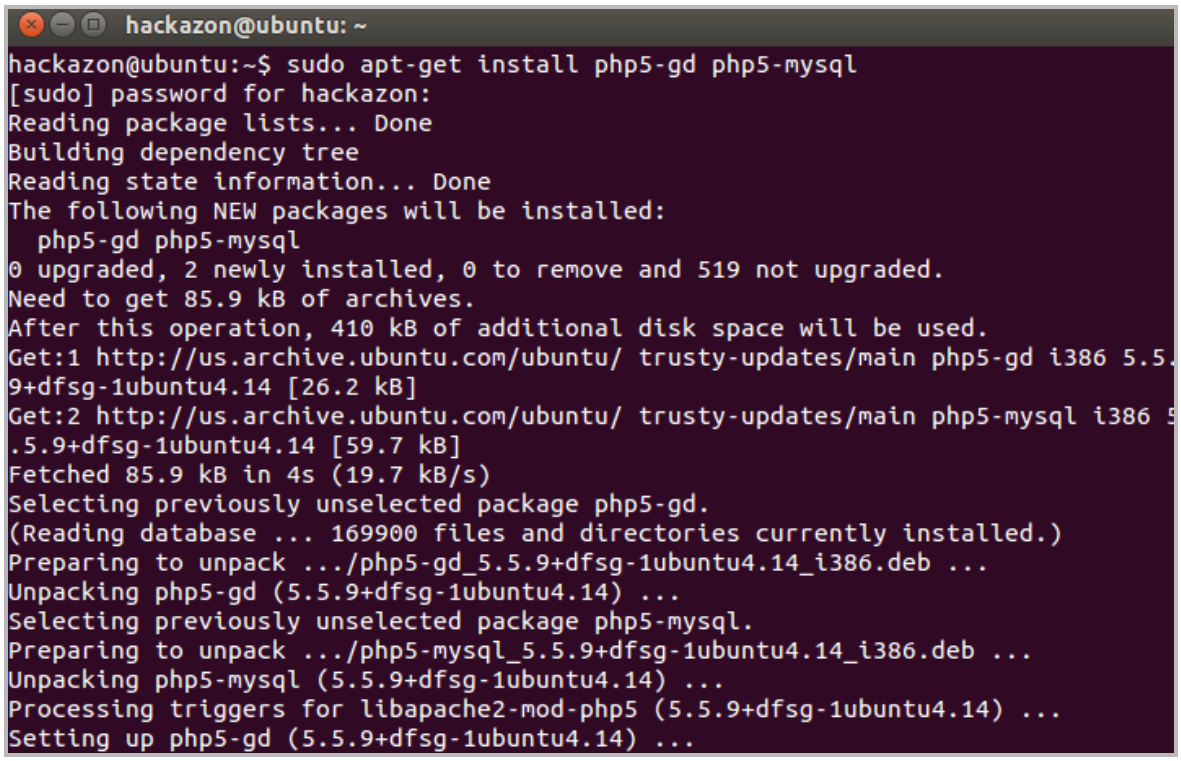
18. Disable Apache default site:

```
sudo a2dissite 000-default
```



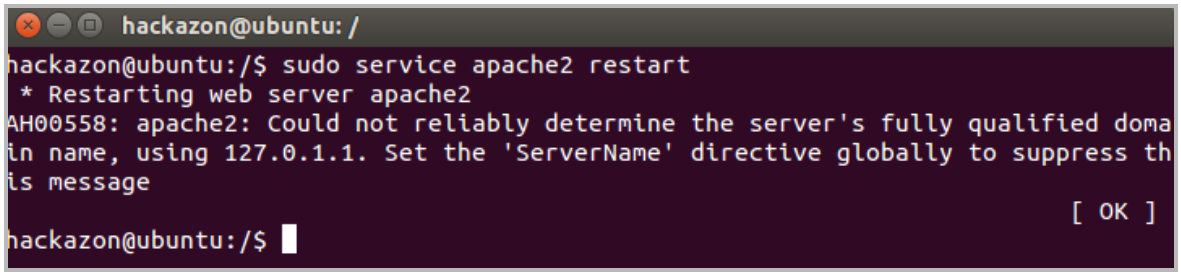
19. Install pdo_mysql drivers:

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



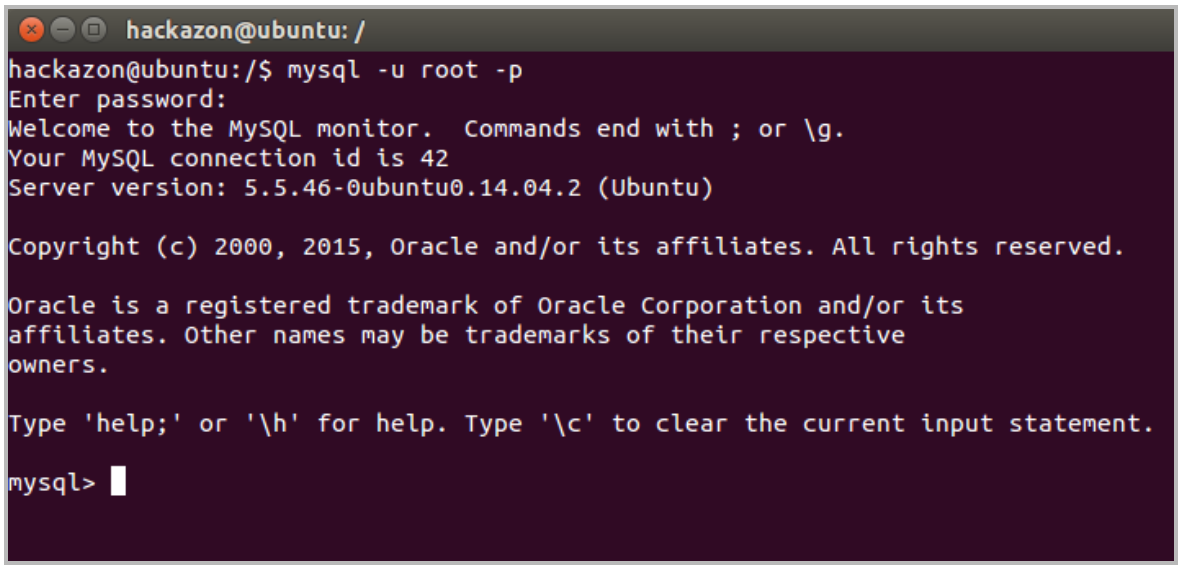
20. Restart the Apache server:

```
sudo service apache2 restart
```



21. Open MySQL database:

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



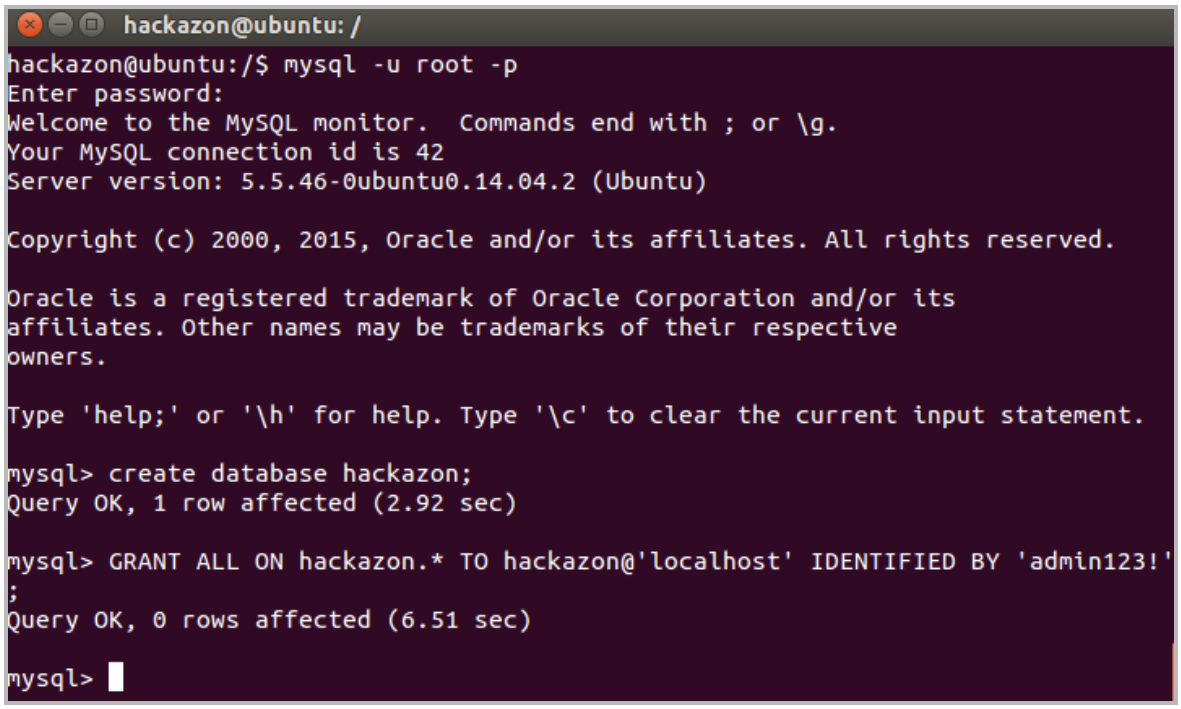
22. Create Hackazon database:

```
create database hackazon;
```

23. Assign database credentials and exit:

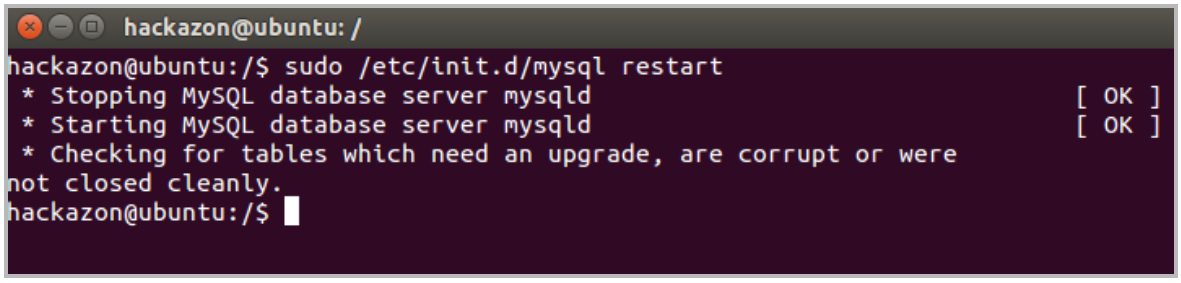
```
GRANT ALL ON hackazon.* TO hackazon@'localhost' IDENTIFIED BY
'InsertPasswordHere';

exit
```

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



24. Restart the MySQL server:

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

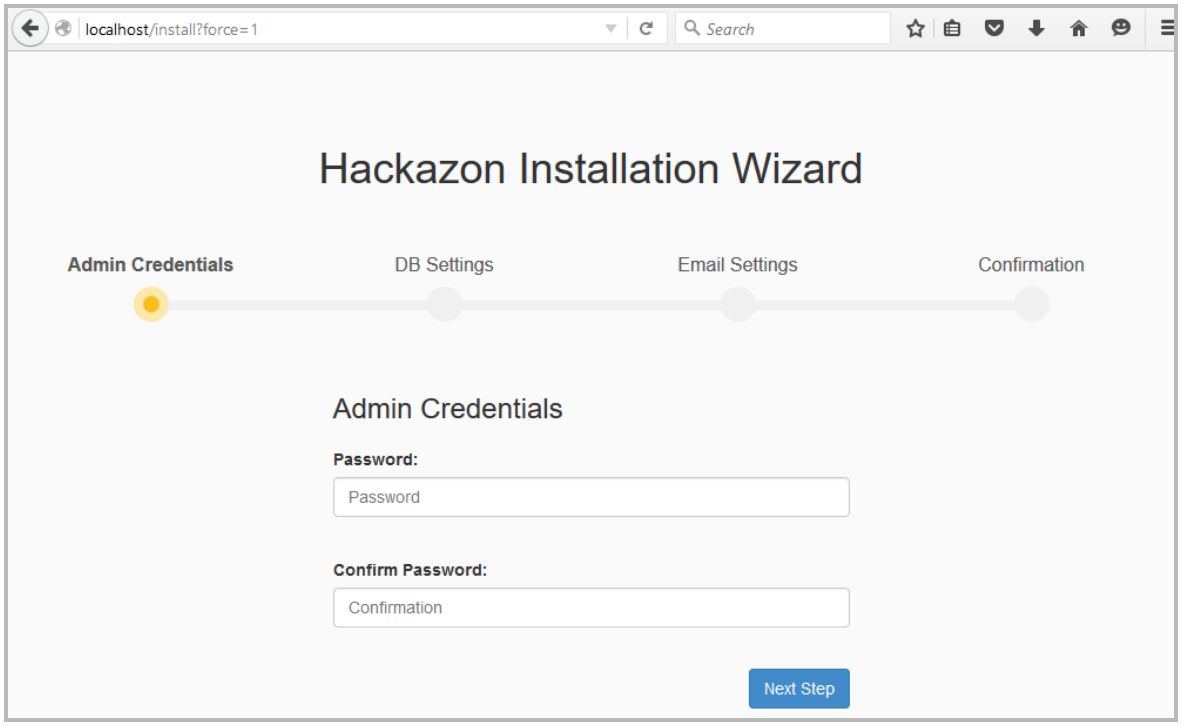


25. When you are finished, proceed to the Hackazon installation wizard.

Hackazon installation wizard

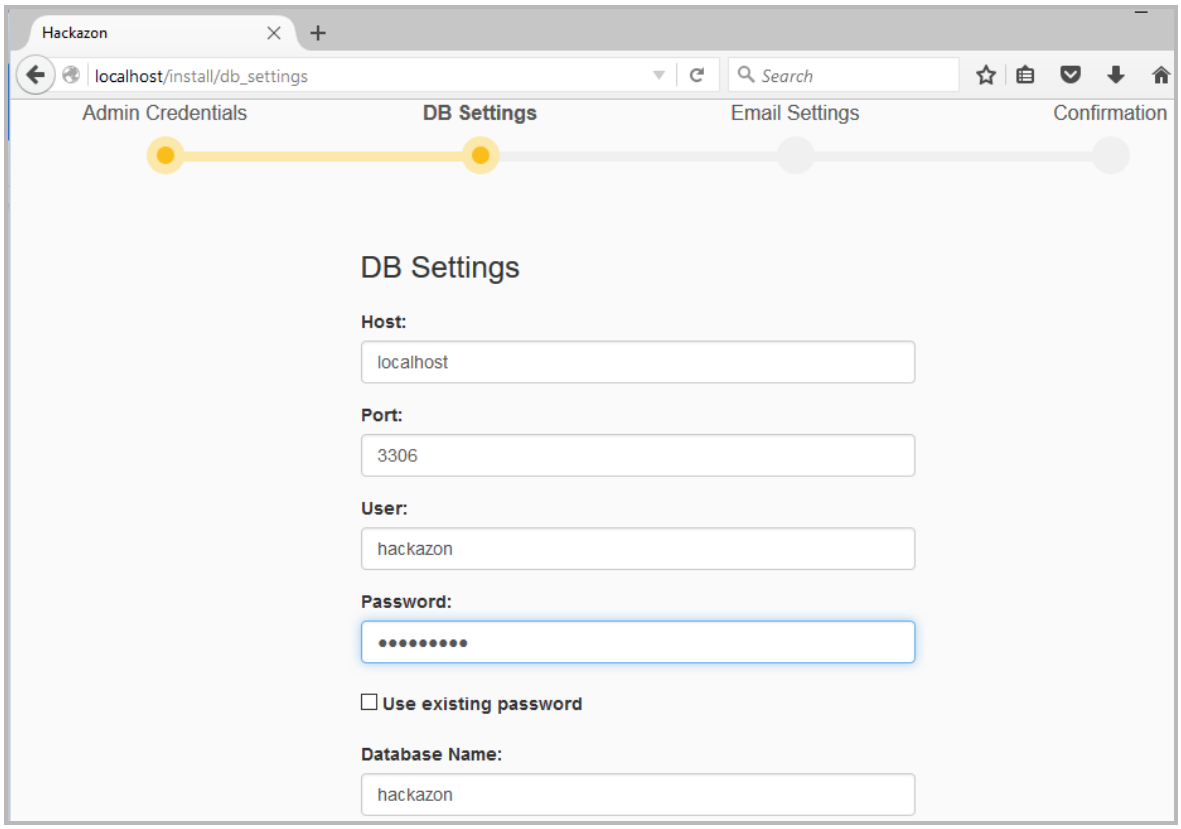
Now that you've configured your Windows or Linux machine using our setup instructions, Hackazon is ready to be installed.

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

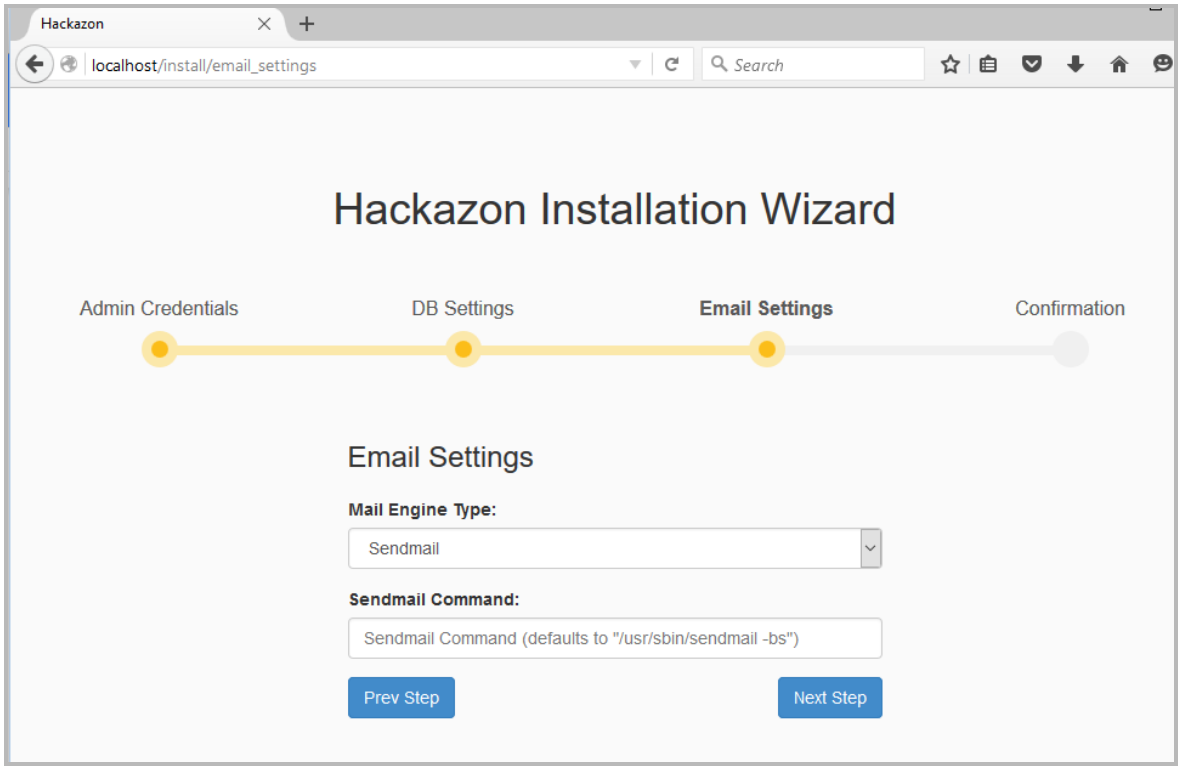


Enter database password and click the **Next Step** button.

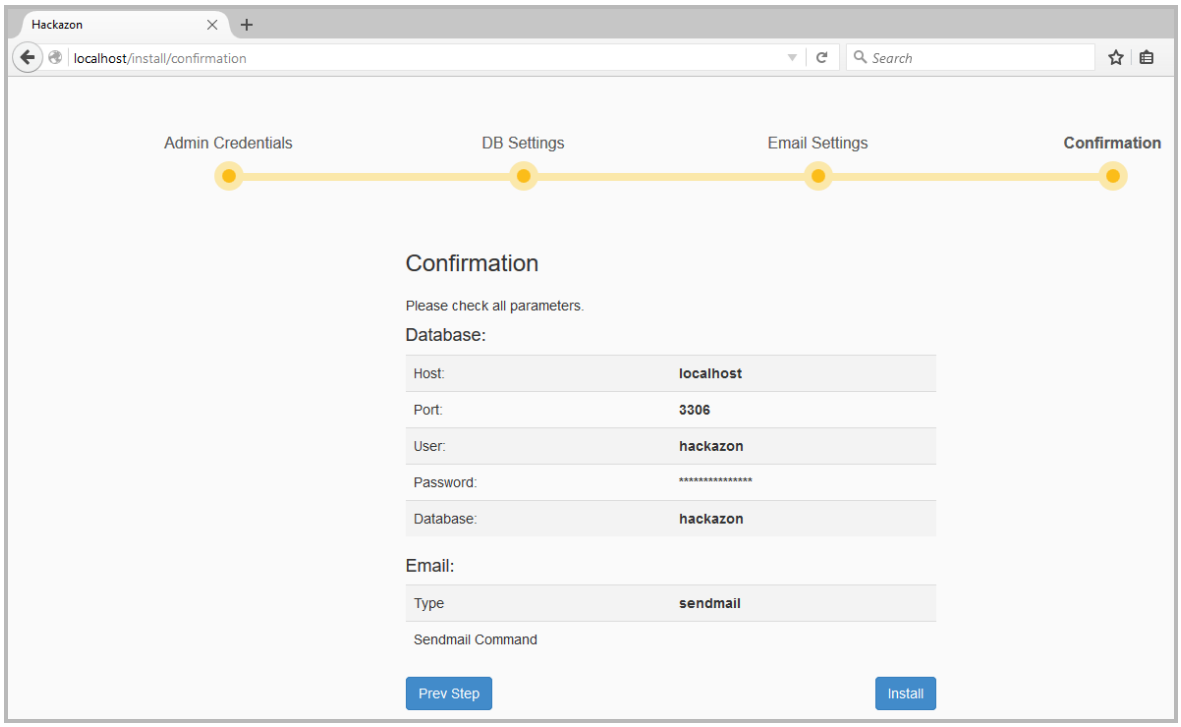
Note: Database credentials were assigned during MySQL configuration.



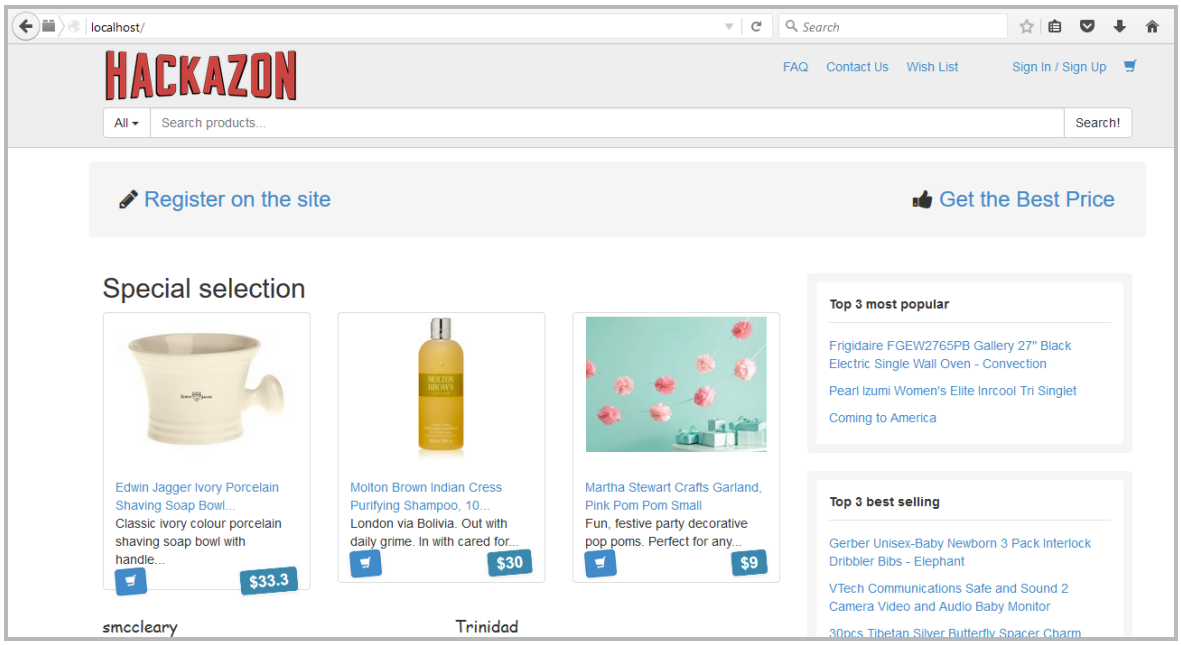
3. Apply *Email Settings* and click the **Next Step** button.



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



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



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

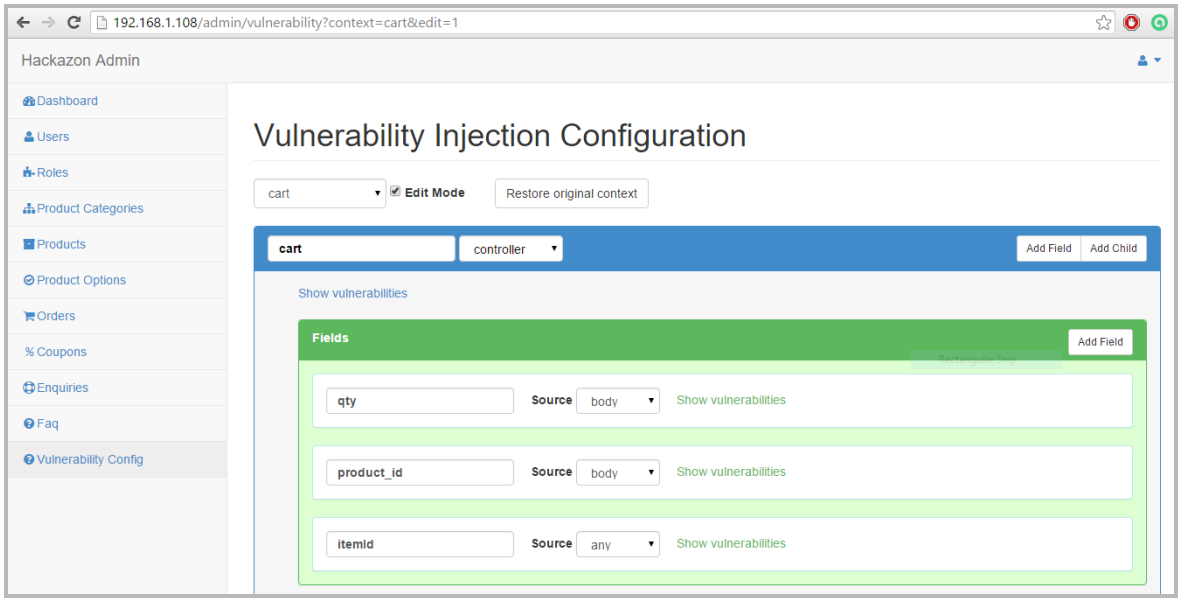
Application features

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

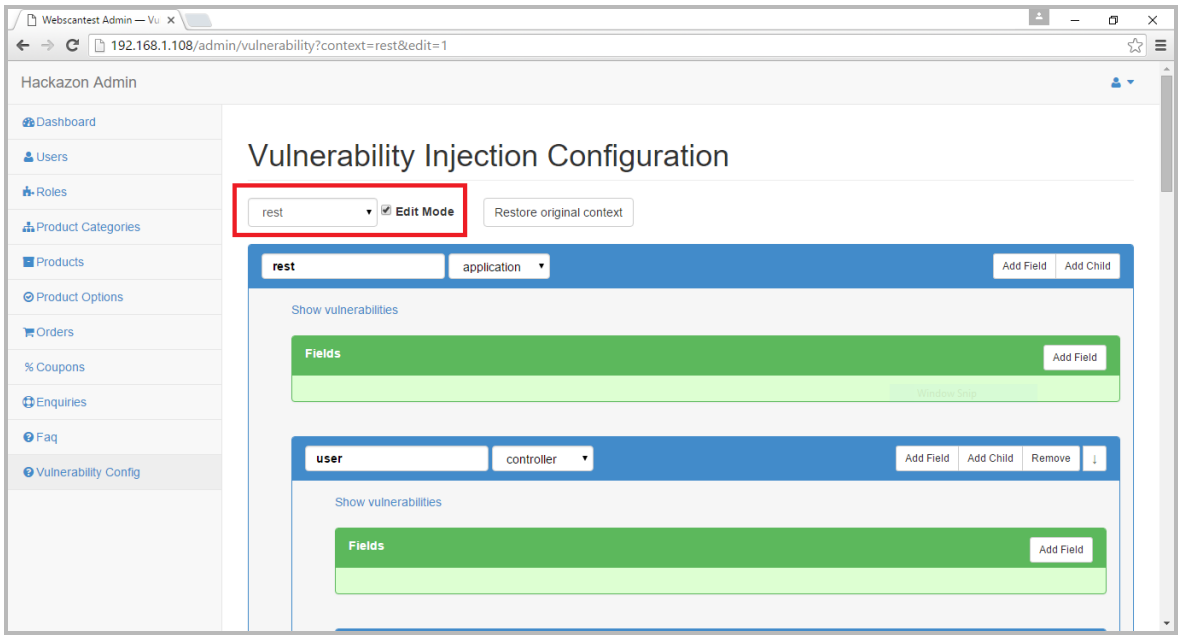
1. **Dashboard:** The dashboard component shows the vulnerabilities that persist in the application including vulnerable URL, Field, vulnerabilities, and details.
2. **User:** The application allows users to Add, Edit, and Update users.
3. **Roles:** The application allows users to Add, Edit, and Update 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

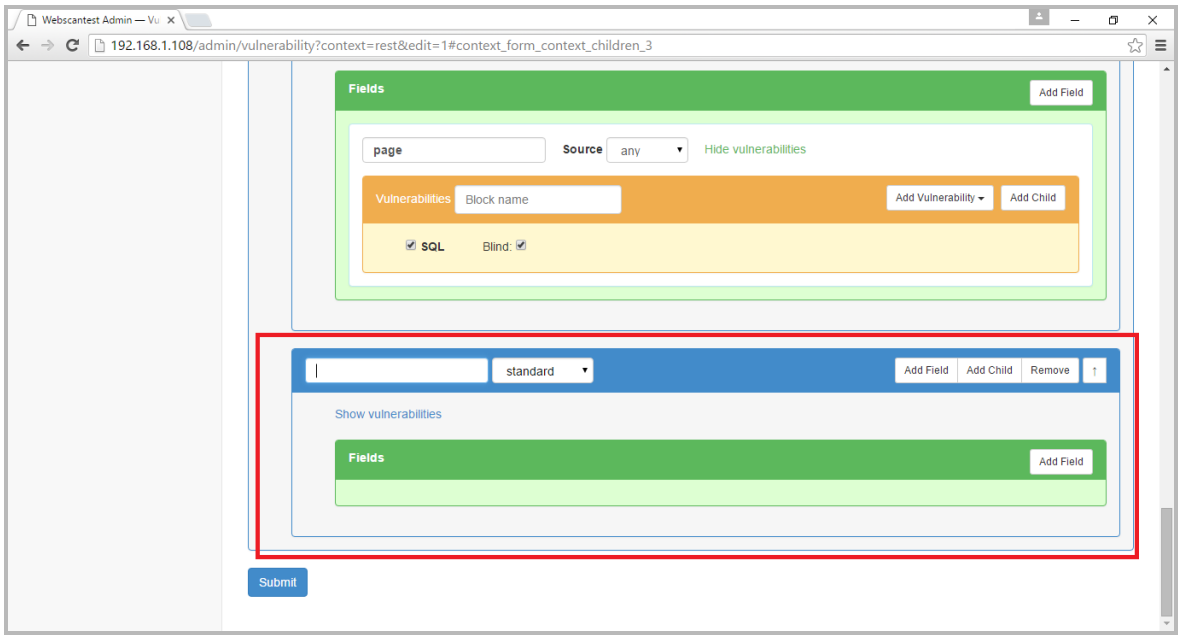
The Hackazon application has a RESTful API in which users can view products. Here is an example on how to create SQL Injection vulnerability in the Hackazon application.

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.

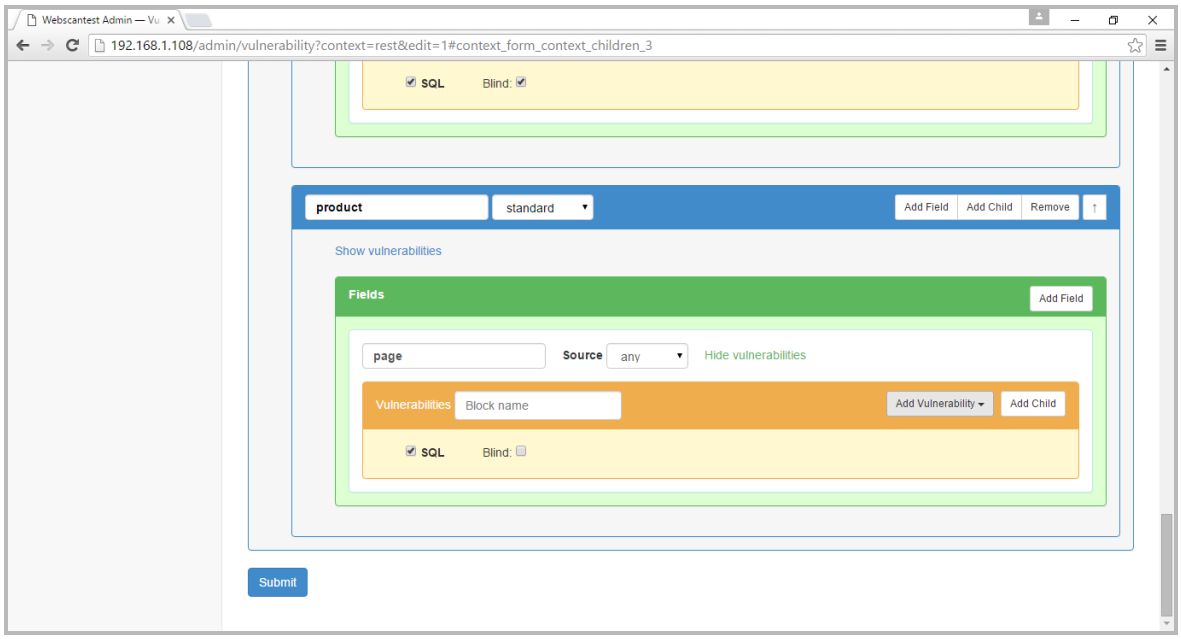


3. Click the **Add Child** button.

The application will generate an empty child box



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



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

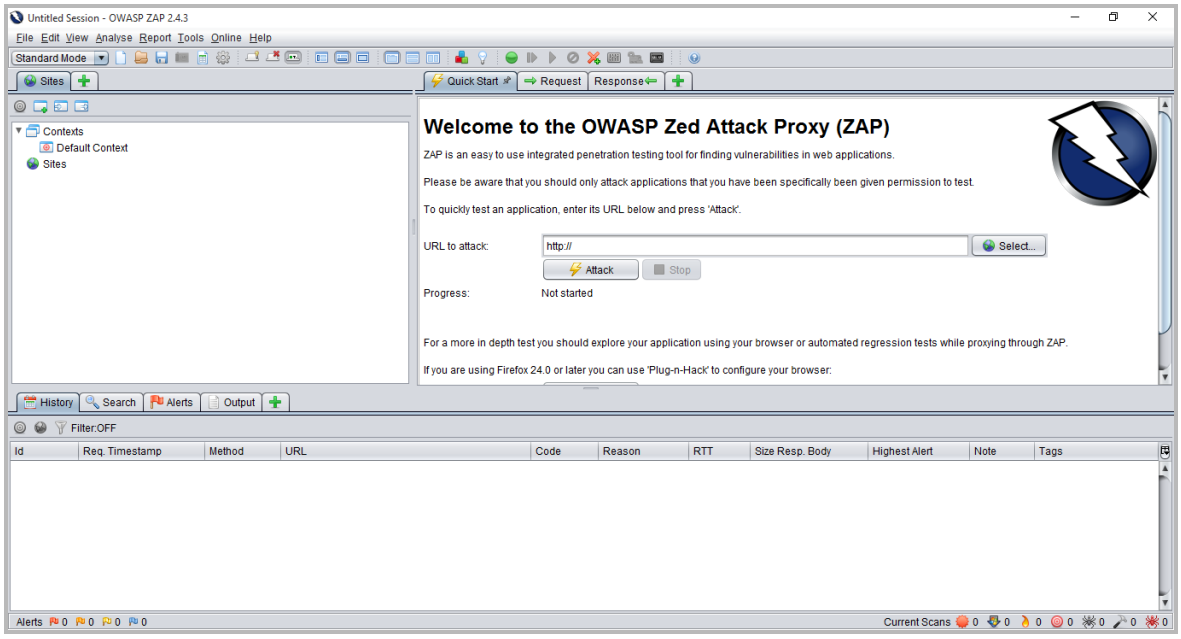
Performing a manual vulnerability assessment requires a browser and a proxy tool. Burp Suite's Proxy tool and OWASP ZAP (Zed Attack Proxy) are proxy tools that are commonly used in the security testing arena. A Java Runtime Environment is required to install and setup both tools. OWASP ZAP, Burp Suite,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

Burp Suite: https://portswigger.net/burp/downloadfree.html

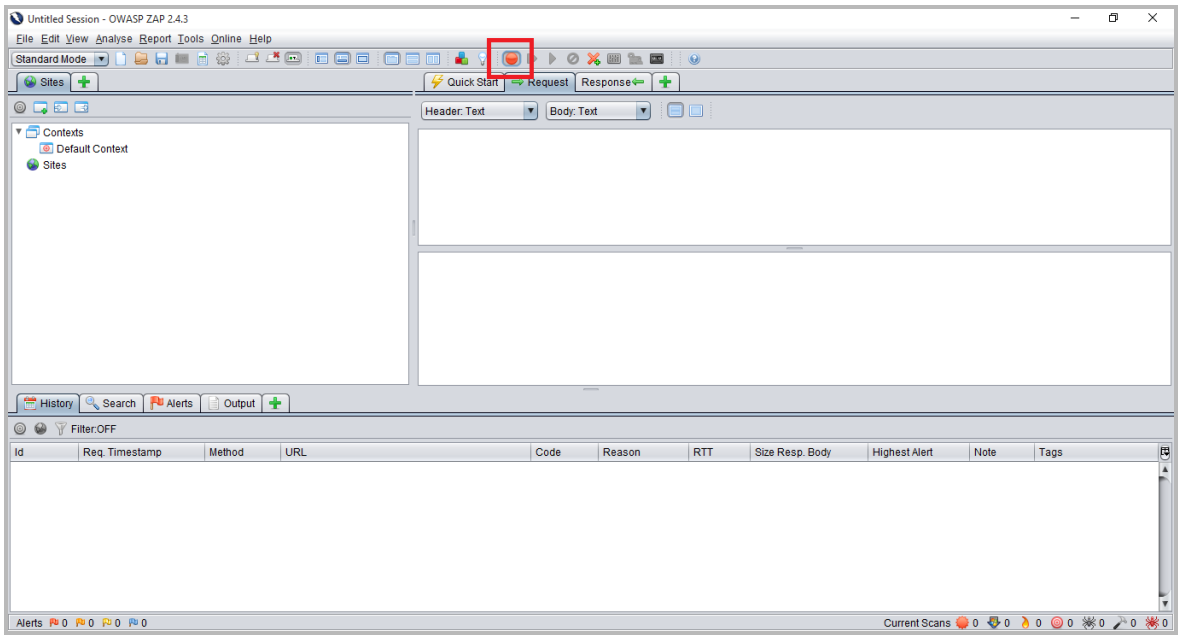
OWASP ZAP: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

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

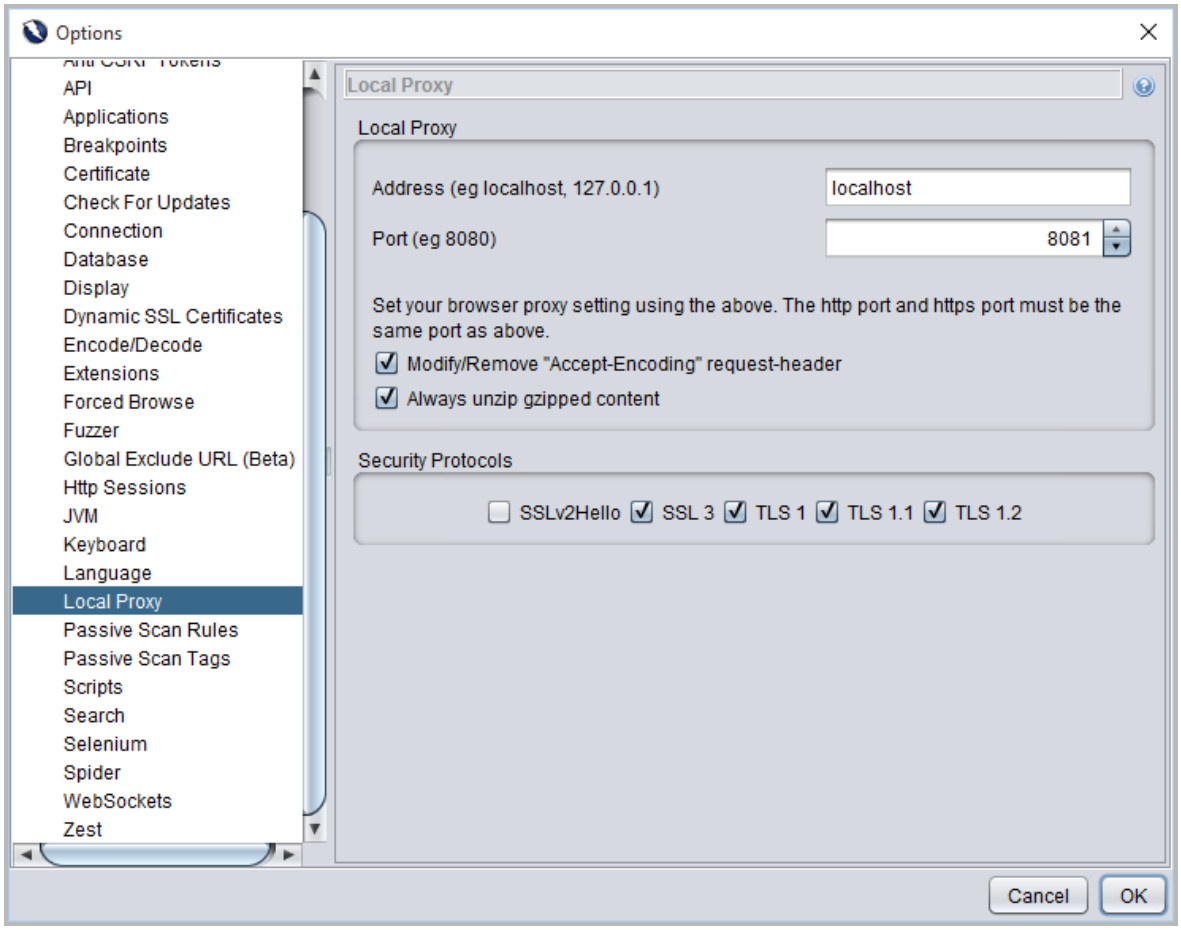
The sites tab will display the tree map of the application.



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

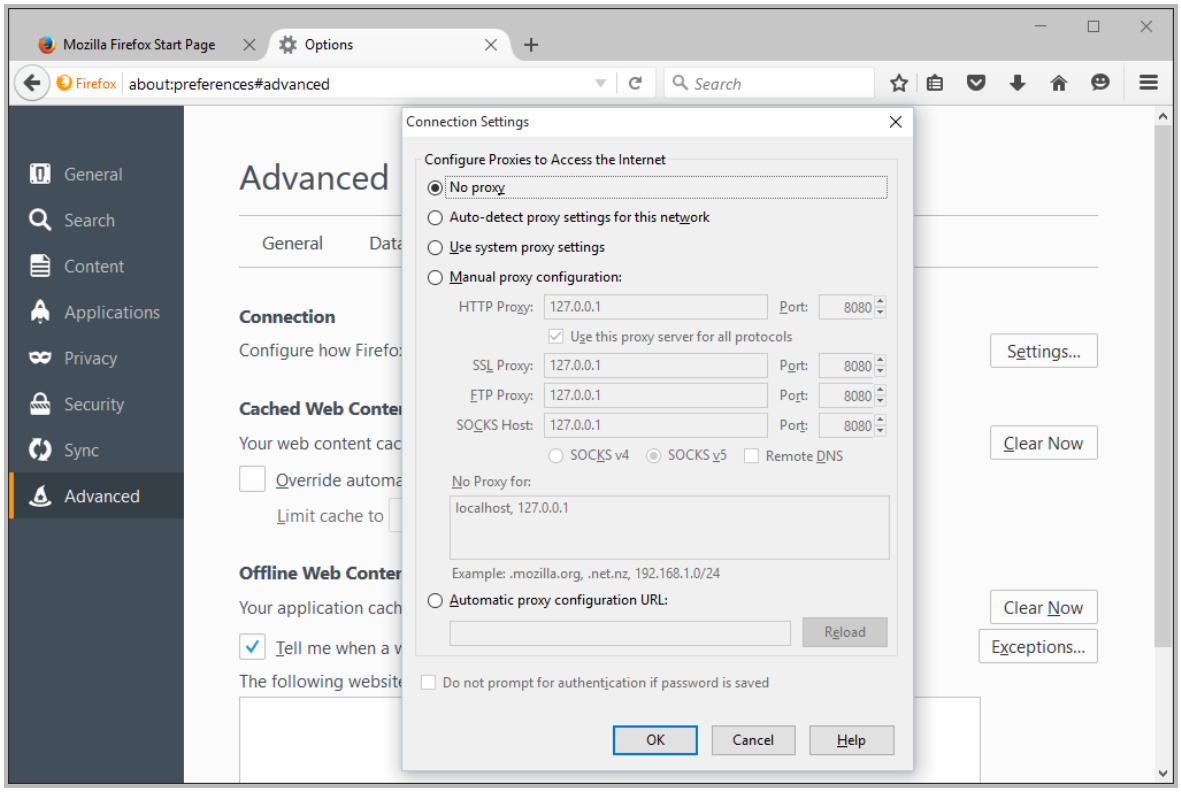


2. To bind ports, navigate to **Options -> Local Proxy** and enter your port number.

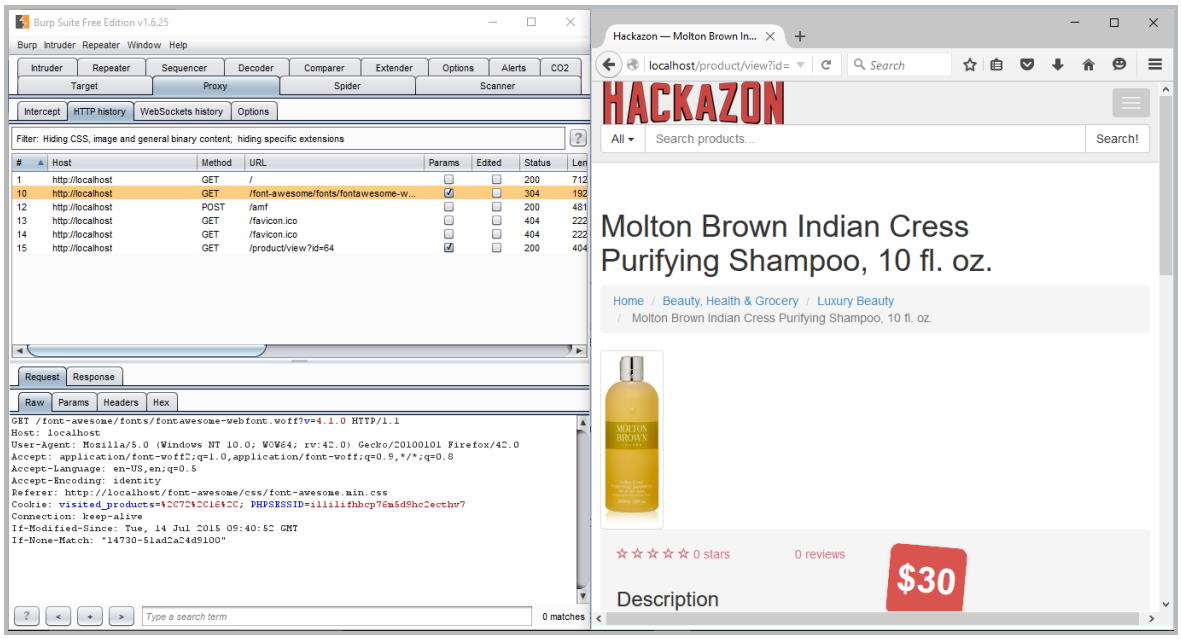


Furthermore, your browser must be configured to use the web proxy.

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.



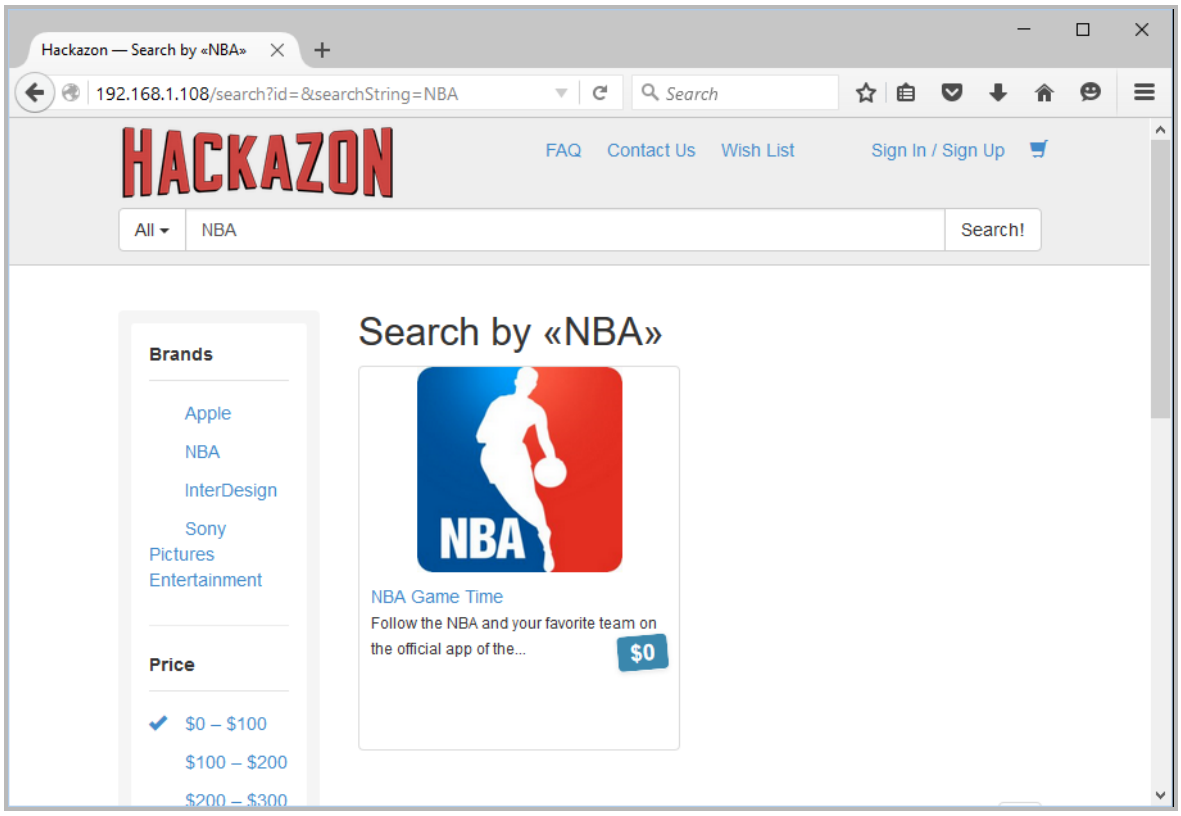
How to find vulnerabilities from the Hackazon application

Cross-site scripting

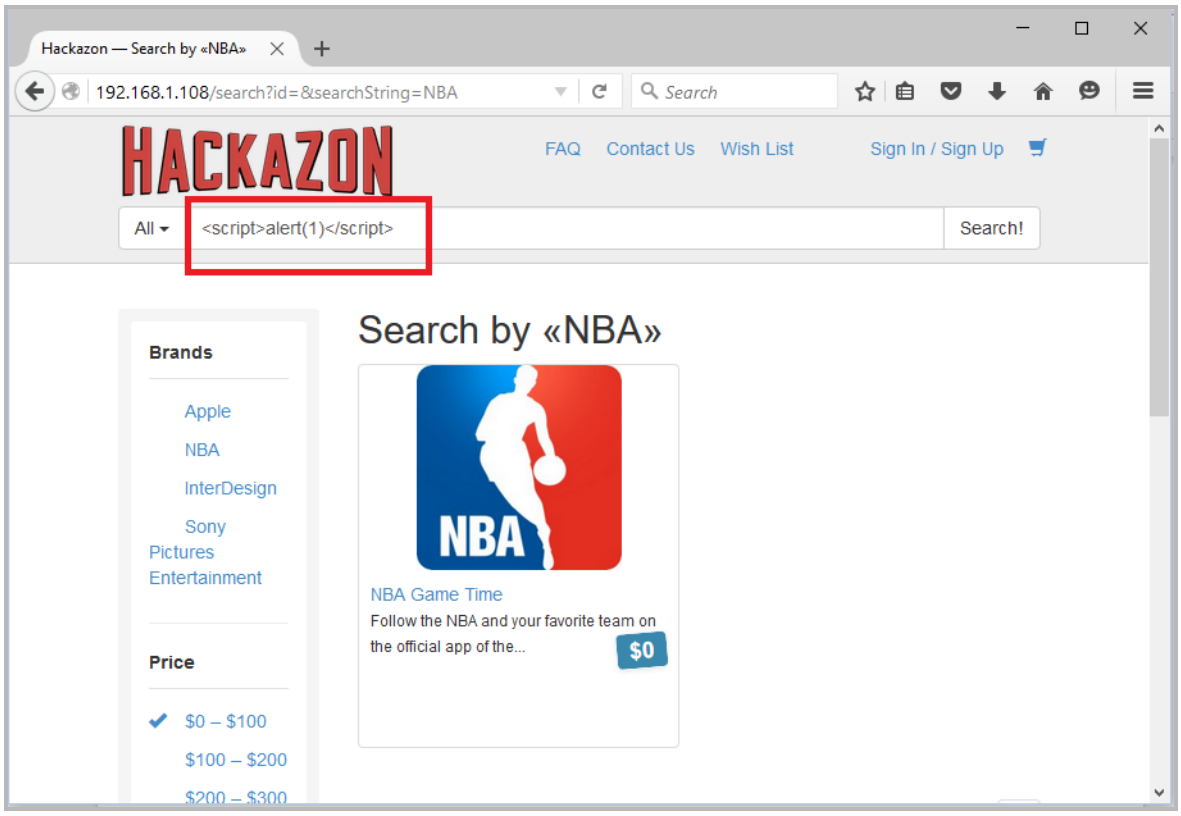
The following example demonstrates the search functionality of the application.

URL: http://192.168.1.108/search?id=&searchString=NBA
Parameter name: searchString
Attack value: <script>alert(1)</script>

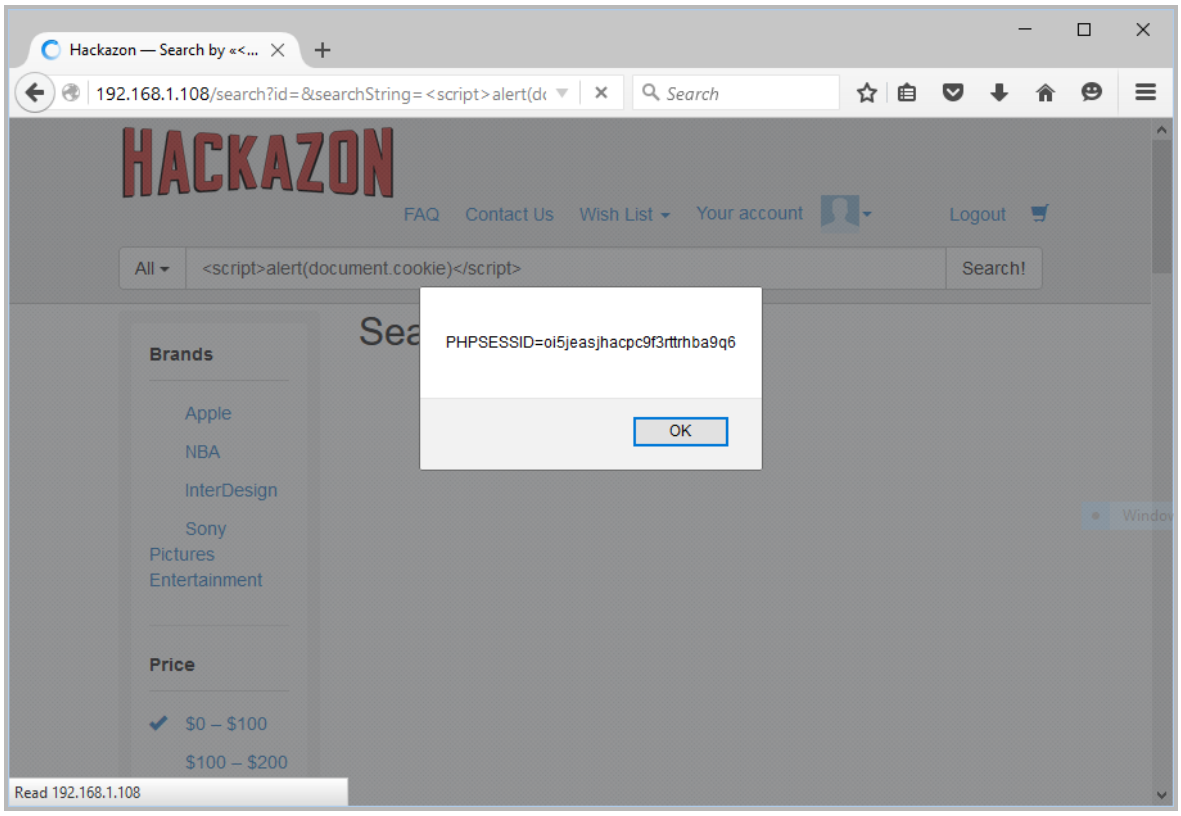
1. Search for the keyword **NBA**. The result will be based on that input.



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



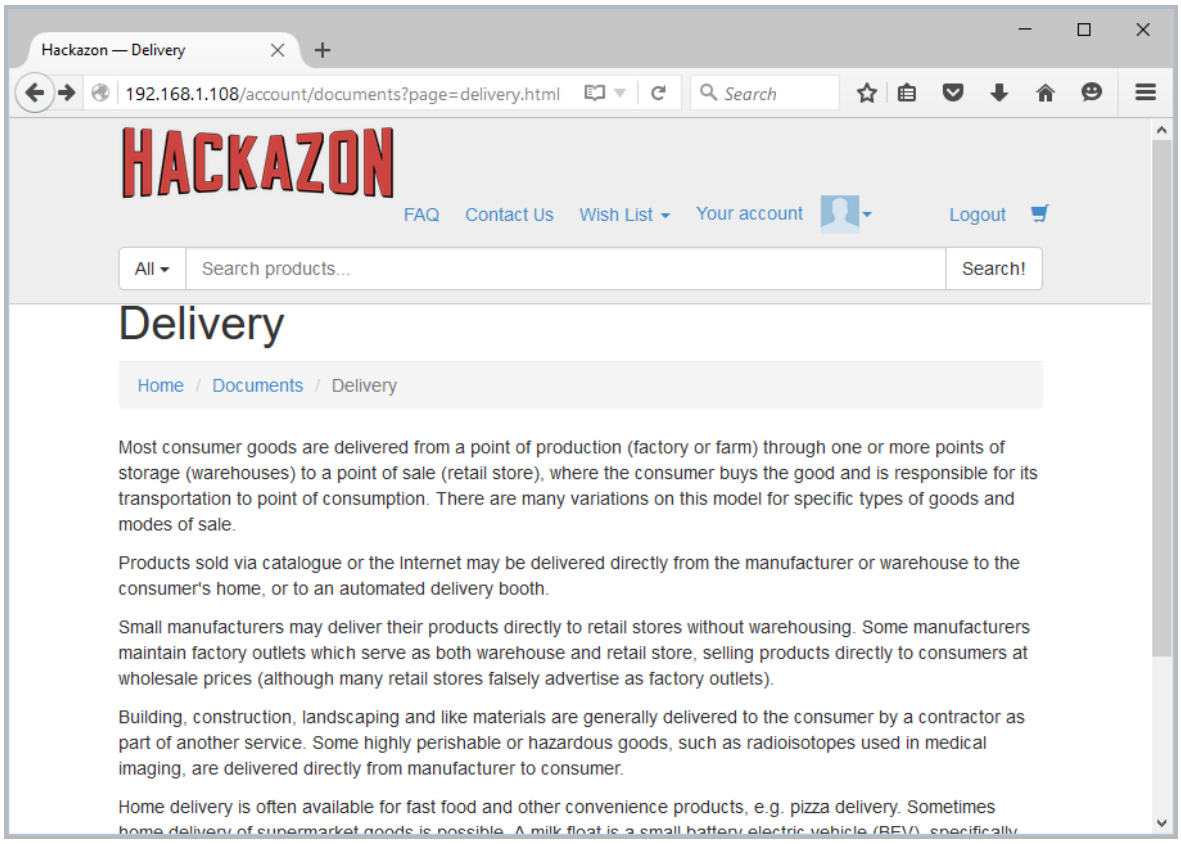
The JavaScript injected into the code was executed.



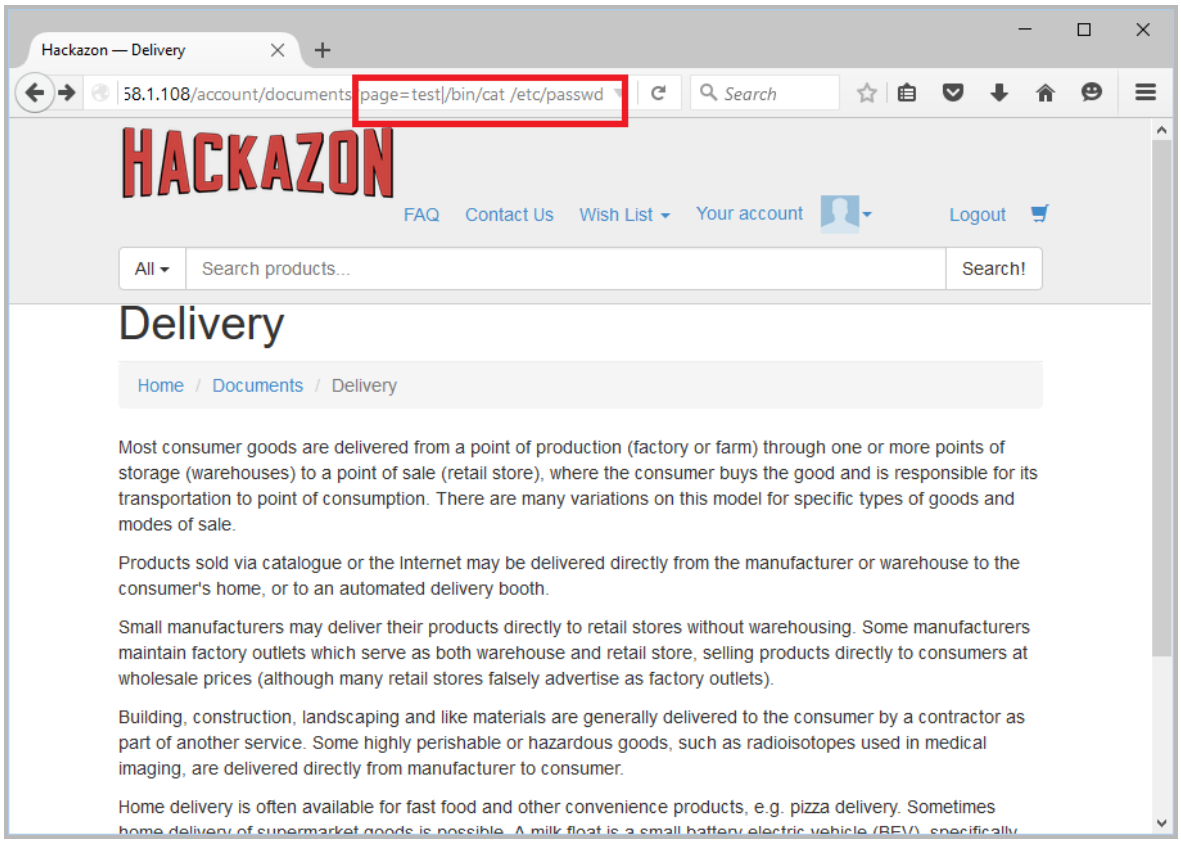
OSCommand injection

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

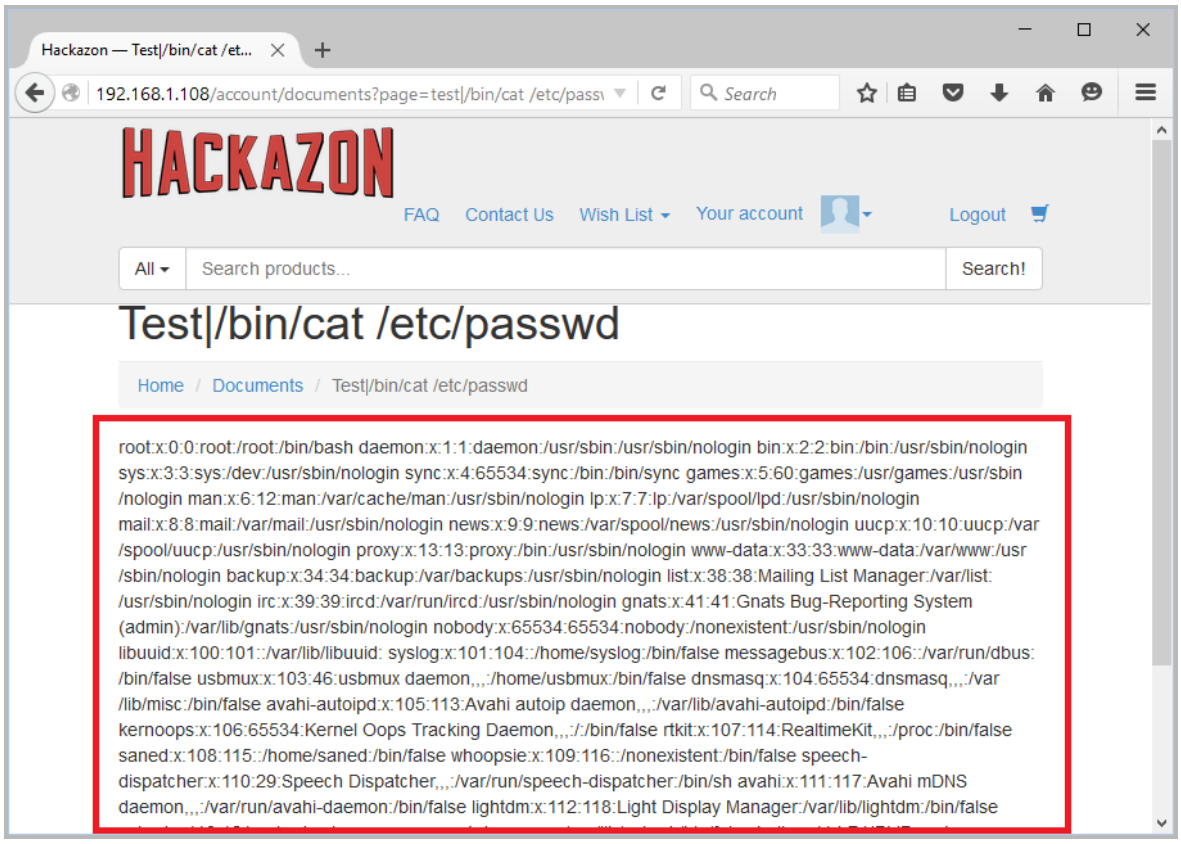
URL: http://192.168.1.108/account/documents?page=delivery.html
Parameter name: page
Attack value: test|/bin/cat /etc/passwd



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



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

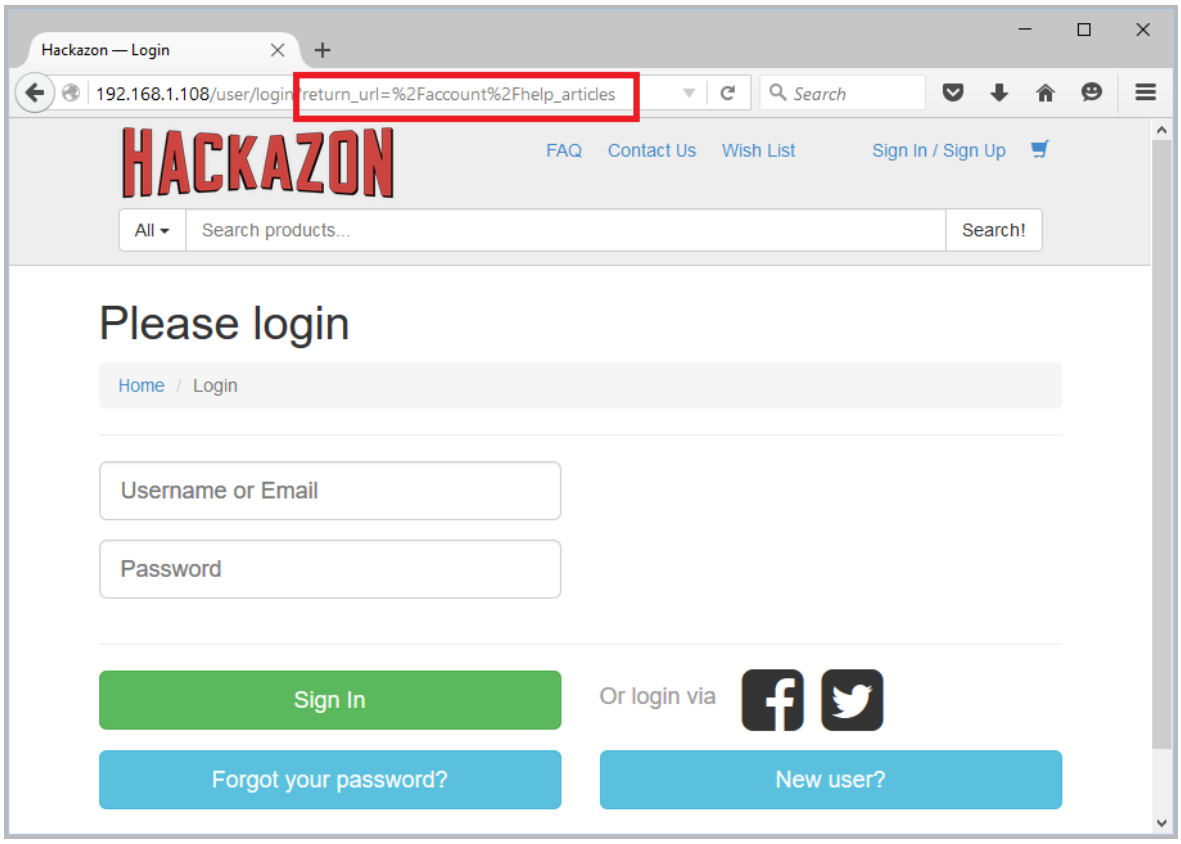


Unvalidated redirect

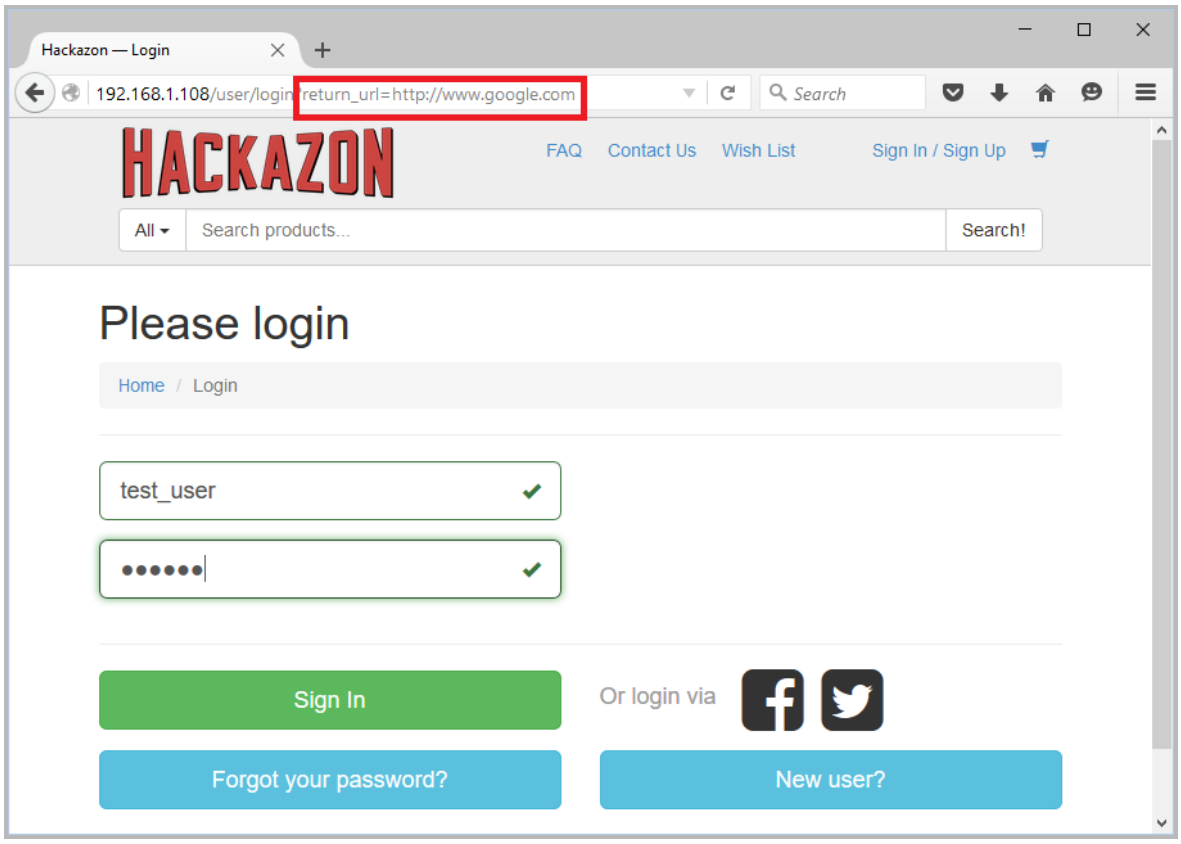
The following example demonstrates how the Hackazon application has functionality to redirect to an internal application page after a login. This vulnerability is used in phishing attacks to get users to visit malicious sites without realizing it.

URL: http://192.168.1.108/user/login?return_url=%2Faccount%2Fhelp_articles
Parameter name: return_url
Attack value: http://www.google.com

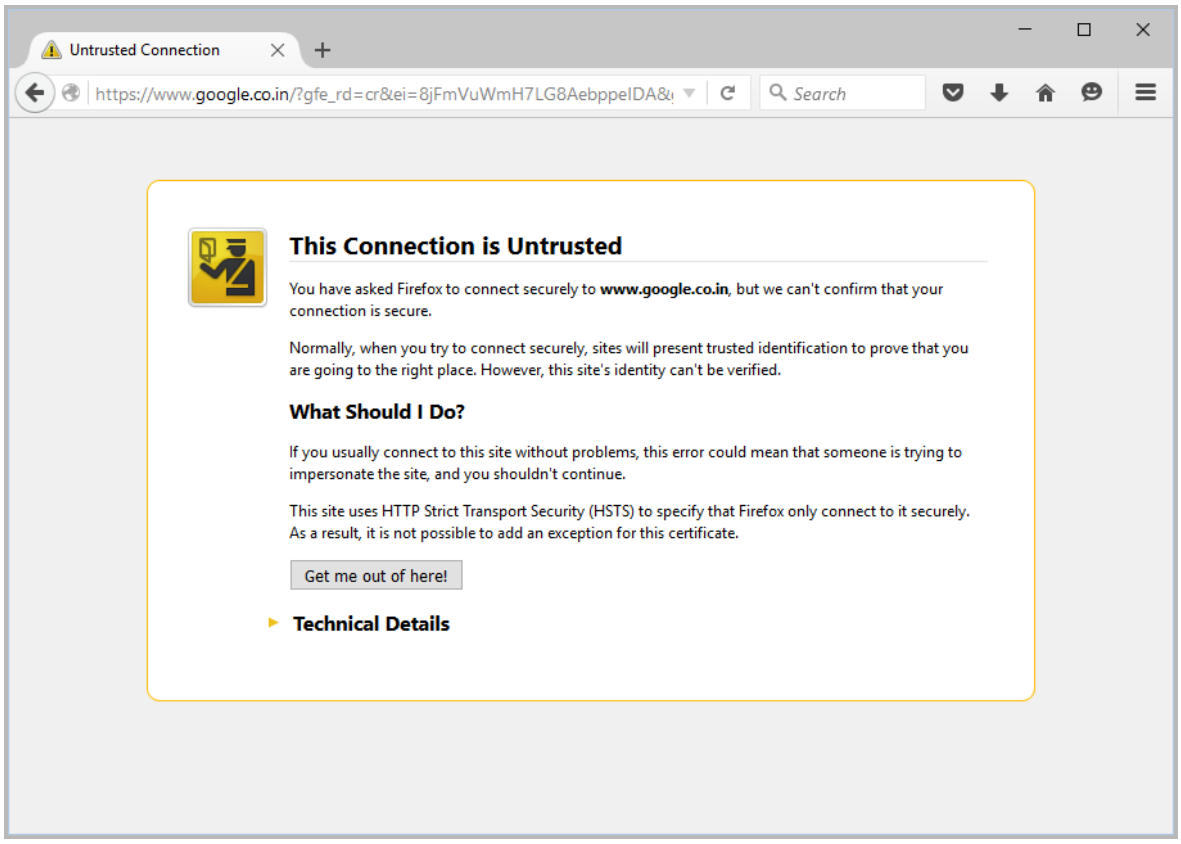
The return_url parameter value is **/account/help_articles**.



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.


Untrusted Connection
https://www.google.co.in/?gfe_rd=cr&ei=8jFmVuWmH7LG8AebppeIDA&
Search
This Connection is Untrusted
You have asked Firefox to connect securely to www.google.co.in, but we can't confirm that your connection is secure.
Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.
What Should I Do?
If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.
This site uses HTTP Strict Transport Security (HSTS) to specify that Firefox only connect to it securely. As a result, it is not possible to add an exception for this certificate.
Get me out of here!
Technical Details

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

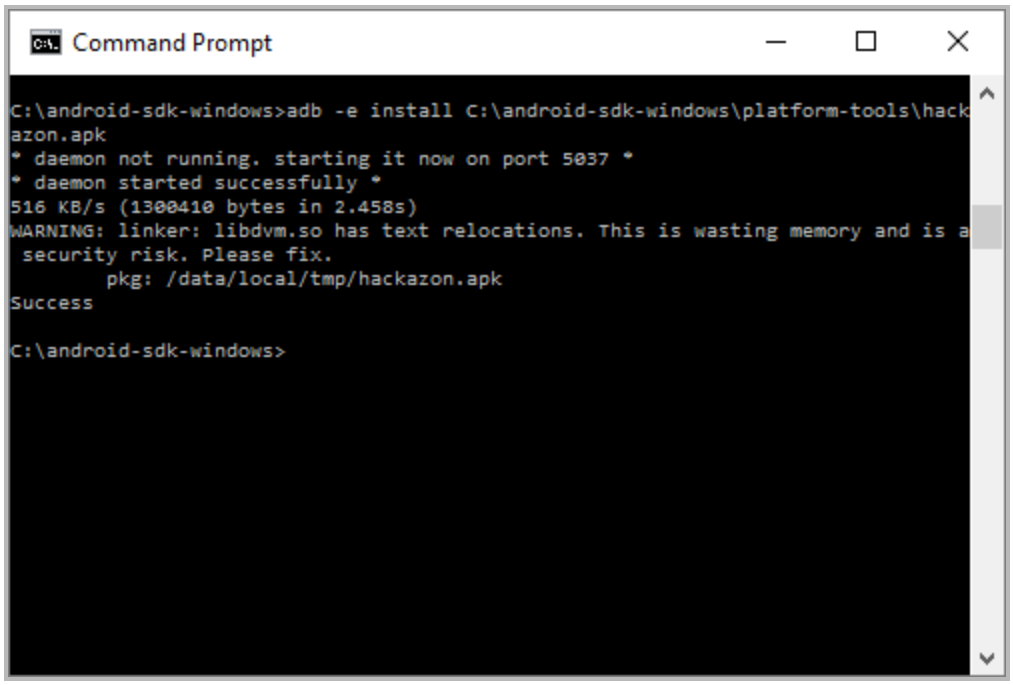
Install Hackazon application in the Android emulator

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.



Configuring the proxy

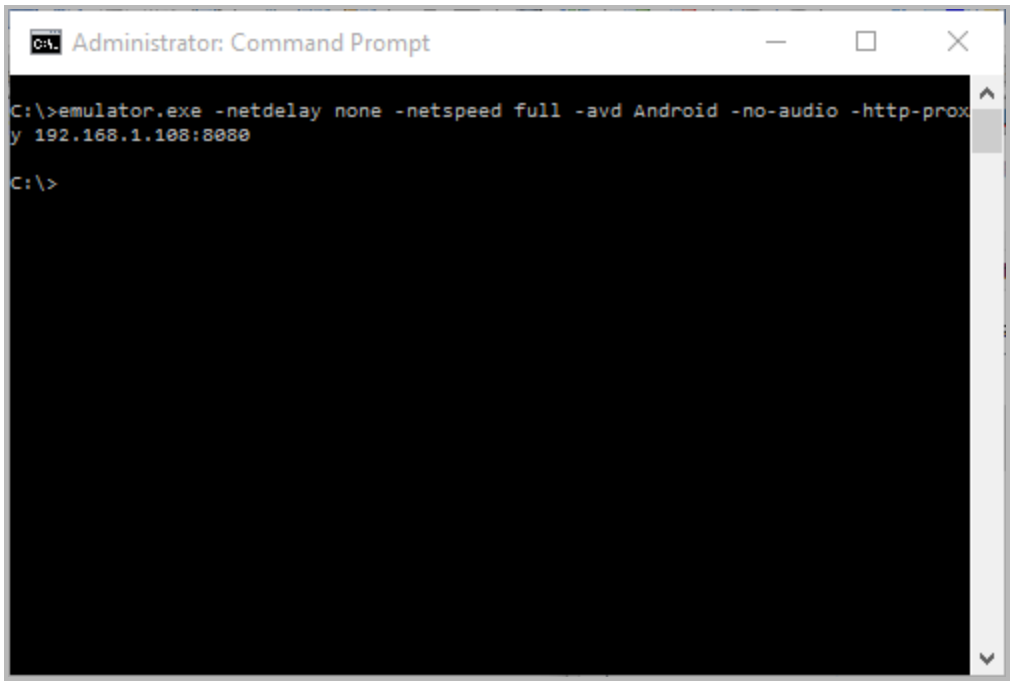
Configuration of the proxy can be performed in two ways. Either at the emulator layer or in the Android Operating System.

Configure the proxy at the emulator layer

Run and set the proxy using a command line tool:

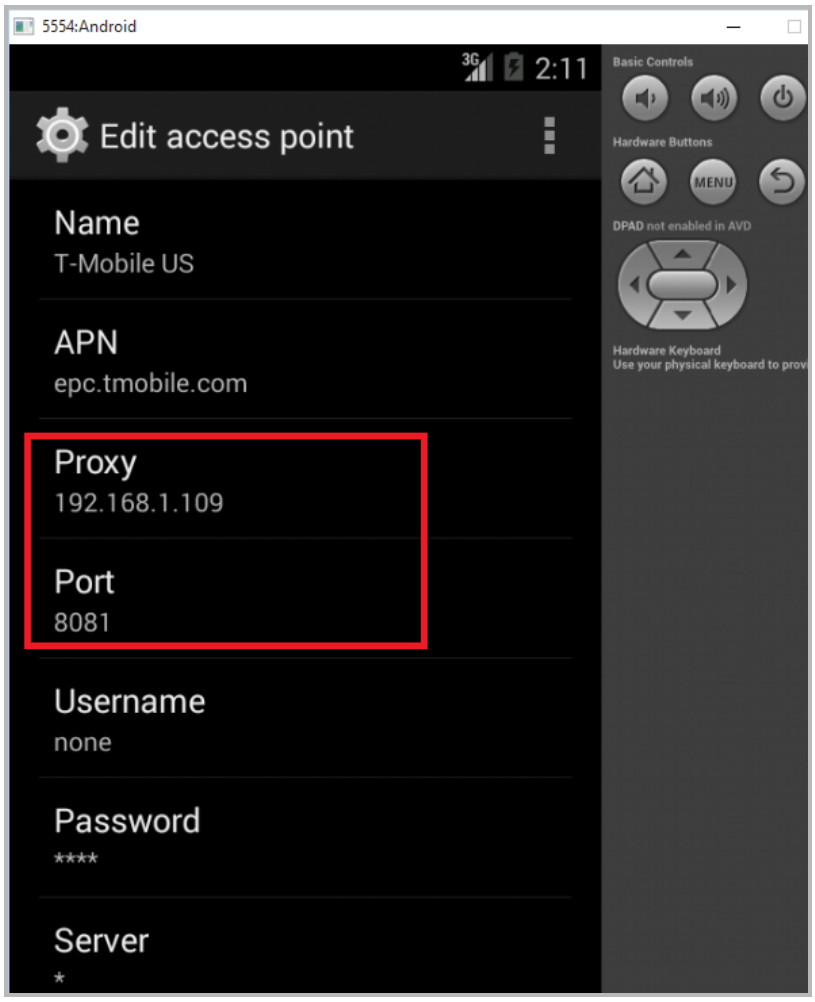
```
-avd Android -http-proxy 192.168.56.101:8080
```

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



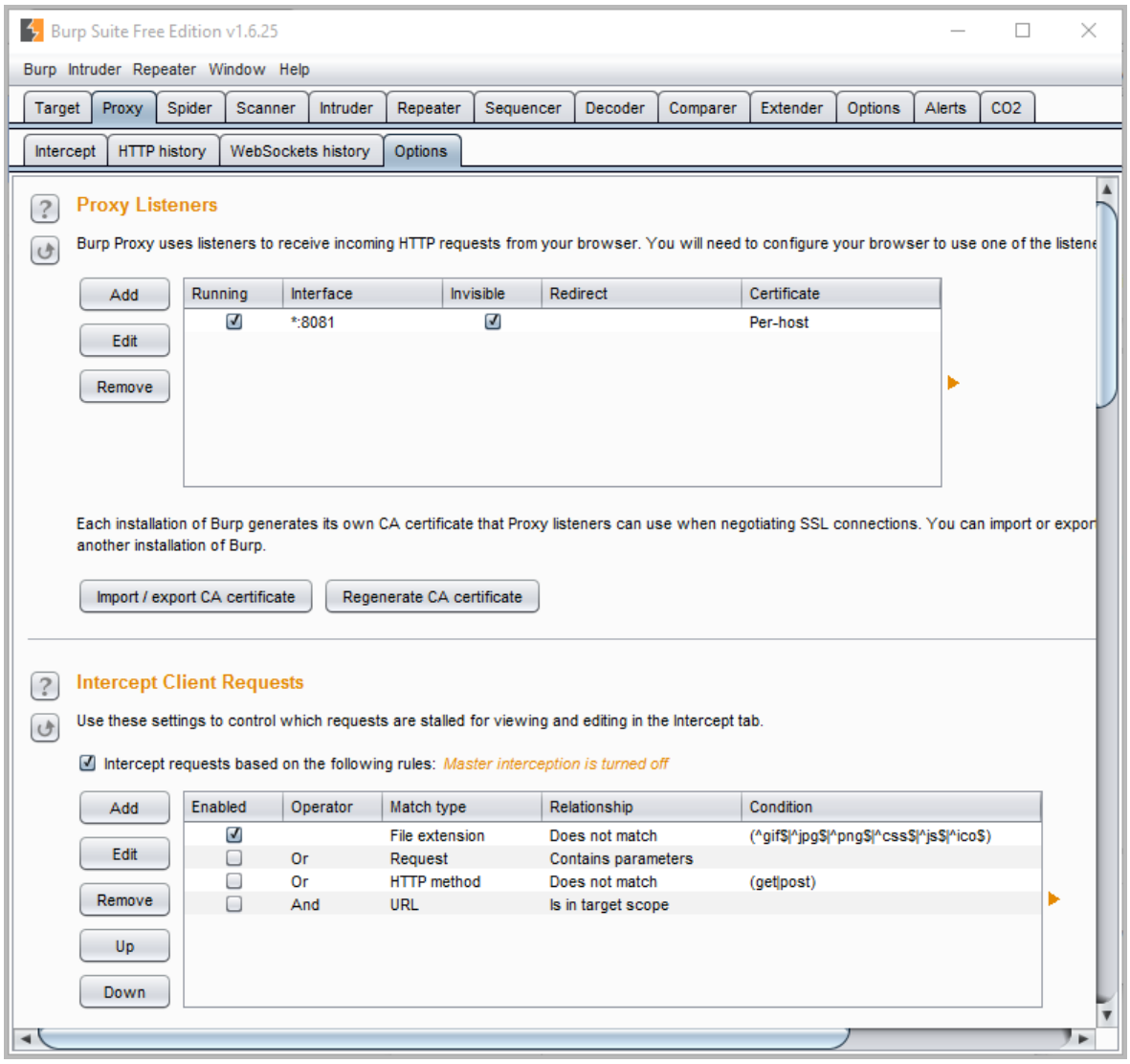
Configure the proxy in the Android operating system

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



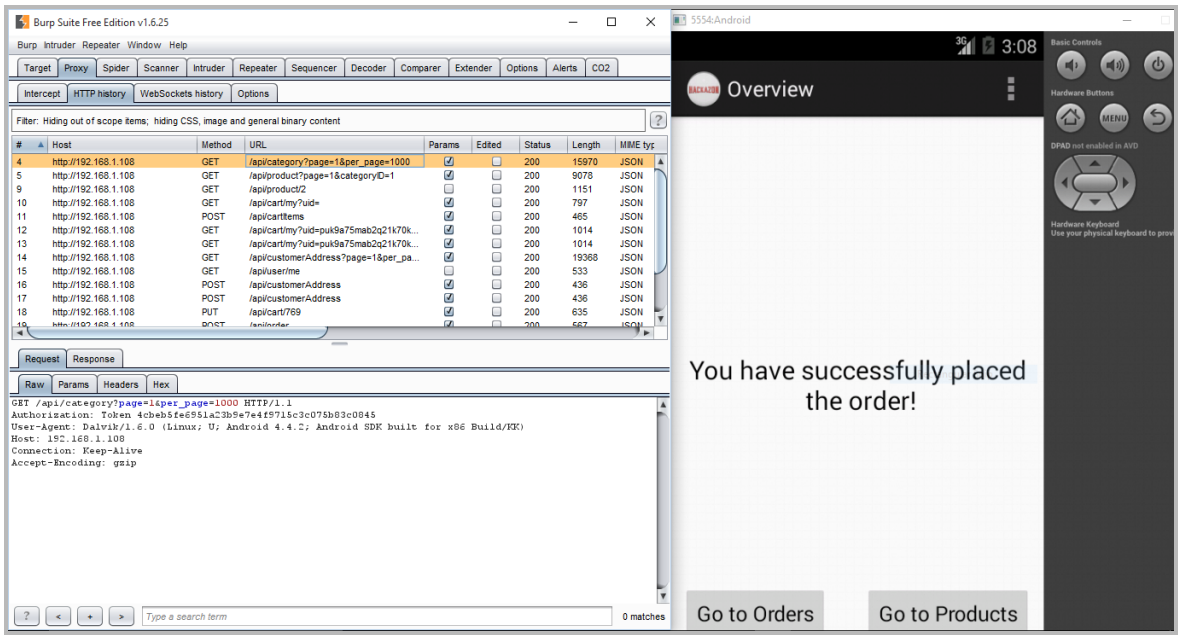
Capture mobile application traffic

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

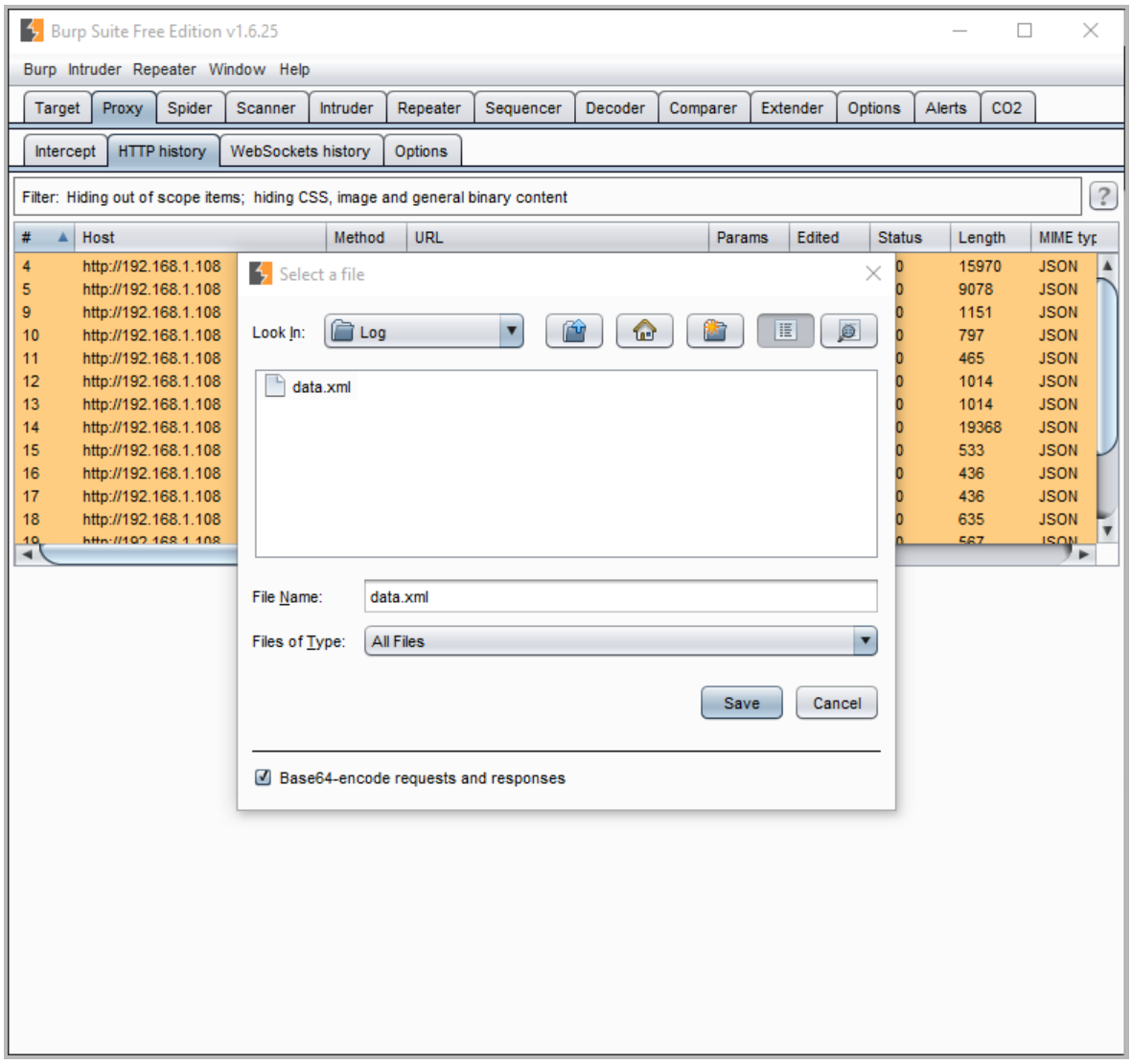


3. Launch the Hackazon application.
4. Log into the application using the default login credentials.
5. Manually crawl the application.

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



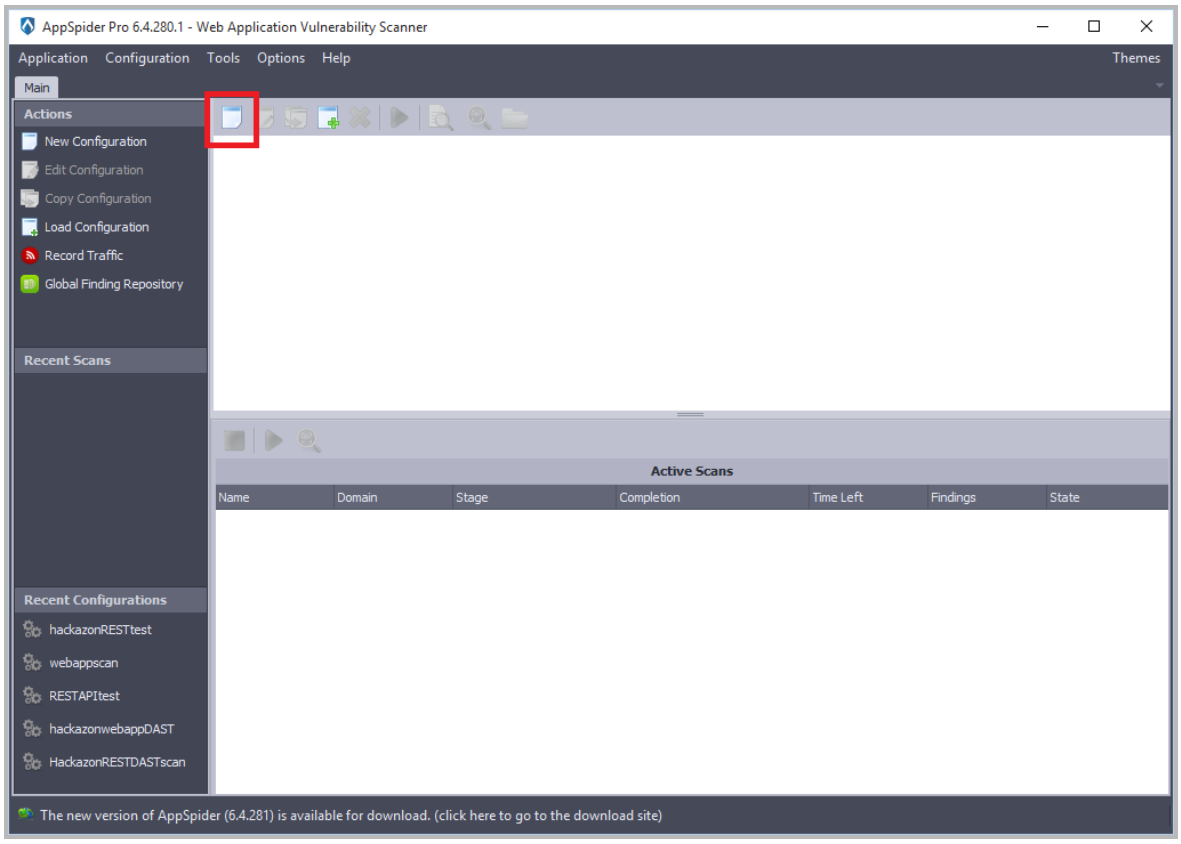
6. When you are finished recording, name your .xml file and click the **Save** button.



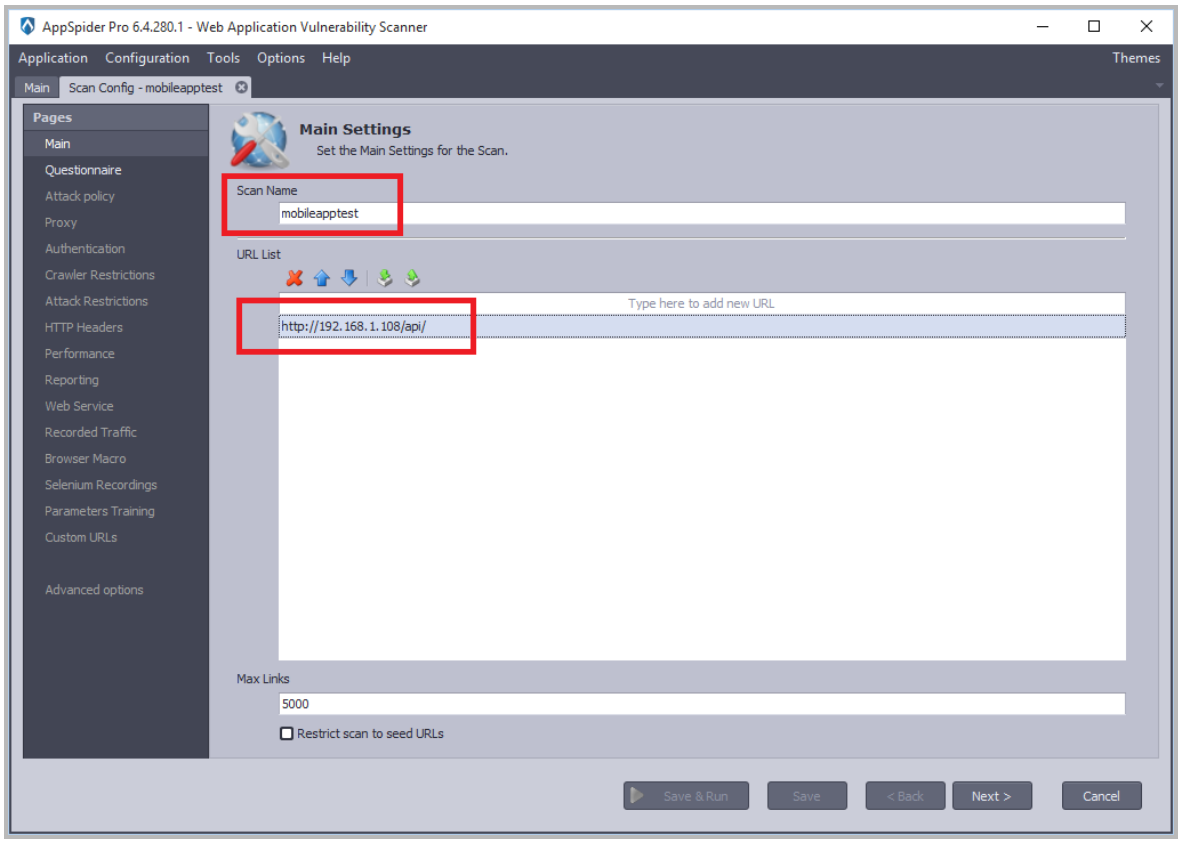
Import recorded traffic into AppSpider

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.

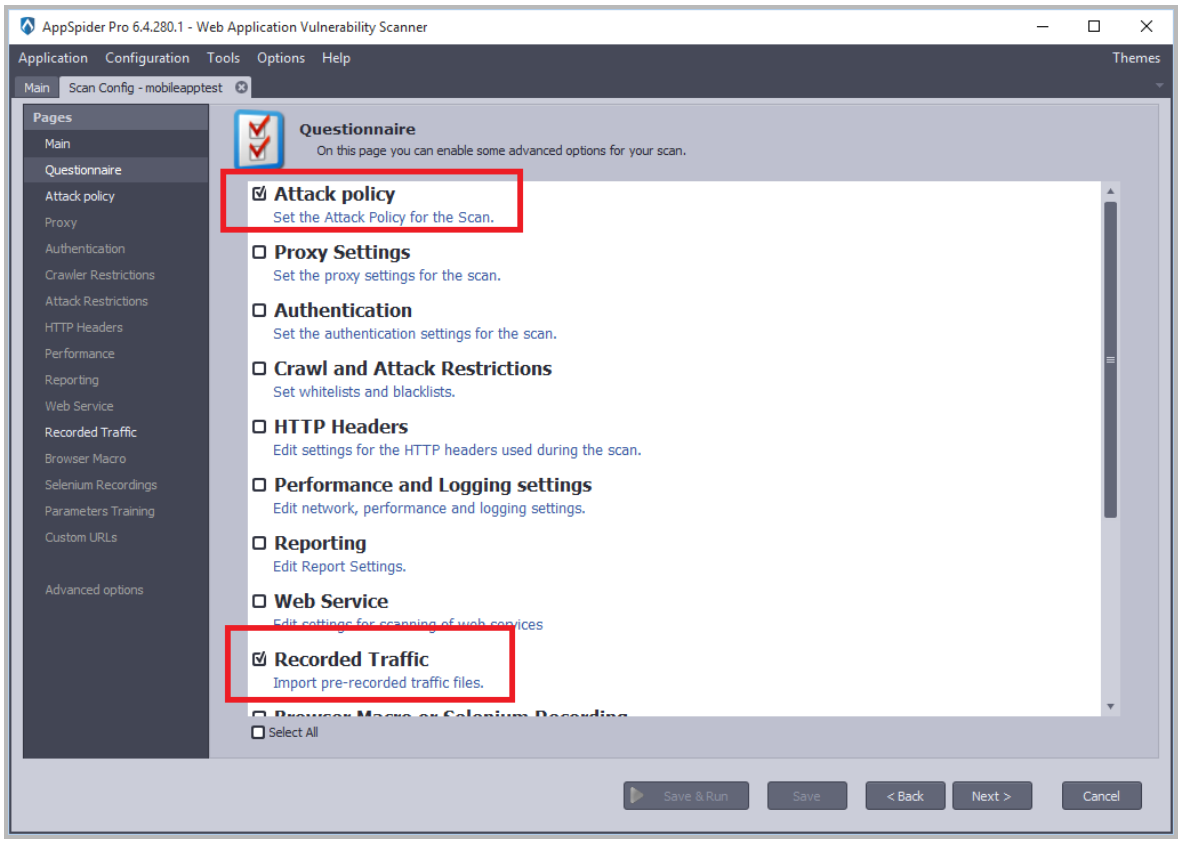
1. Open AppSpider
2. Select **New Configuration** from the *Actions* panel in AppSpider.



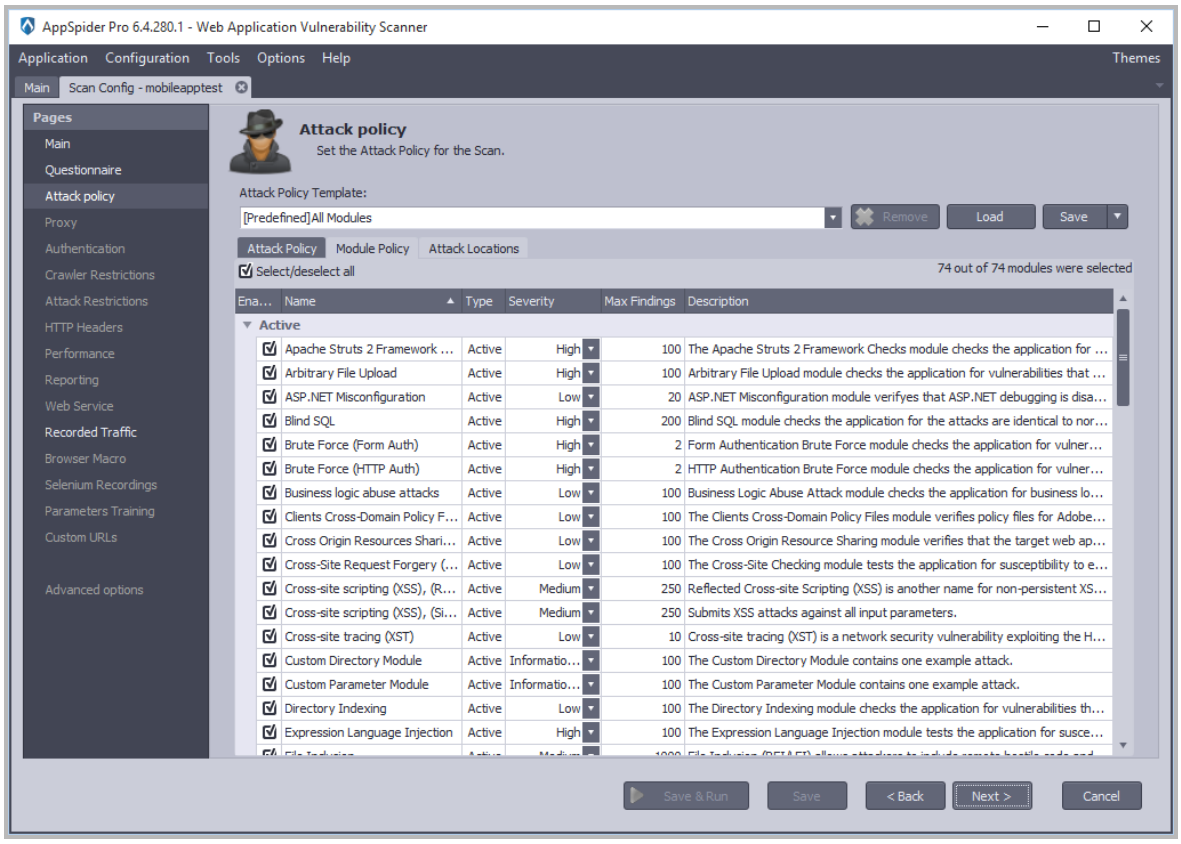
3. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.



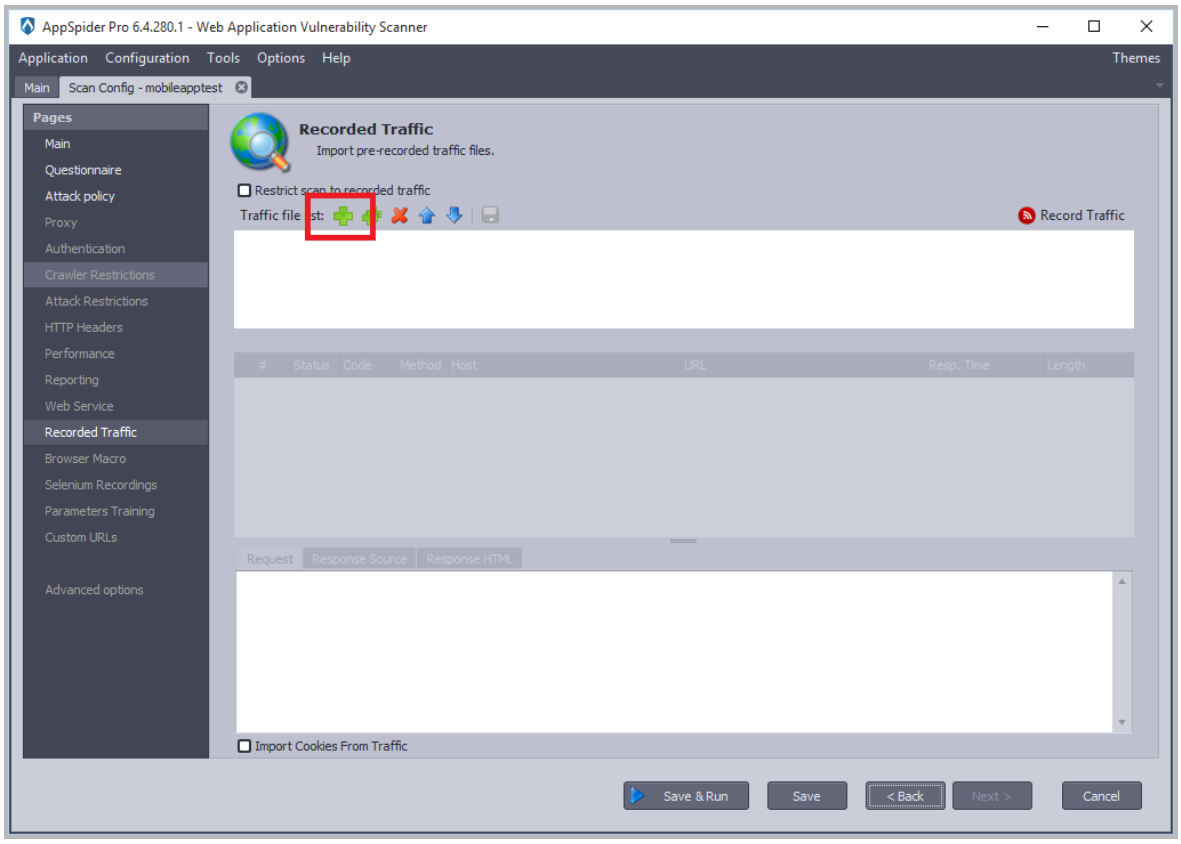
4. Select the check box for **Attack policy** and **Recorded Traffic** then click the **Next** button.



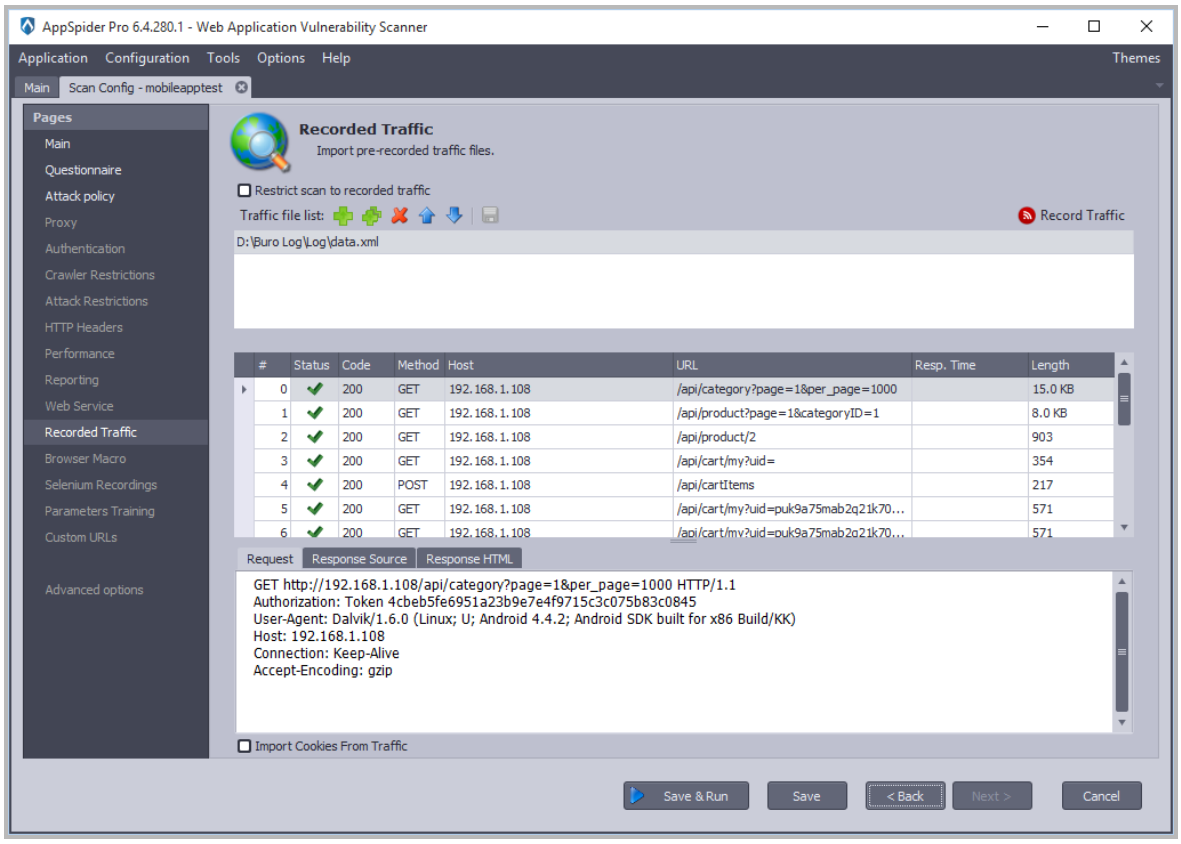
5. Select and load an *Attack Policy Template* then click the **Next** button.



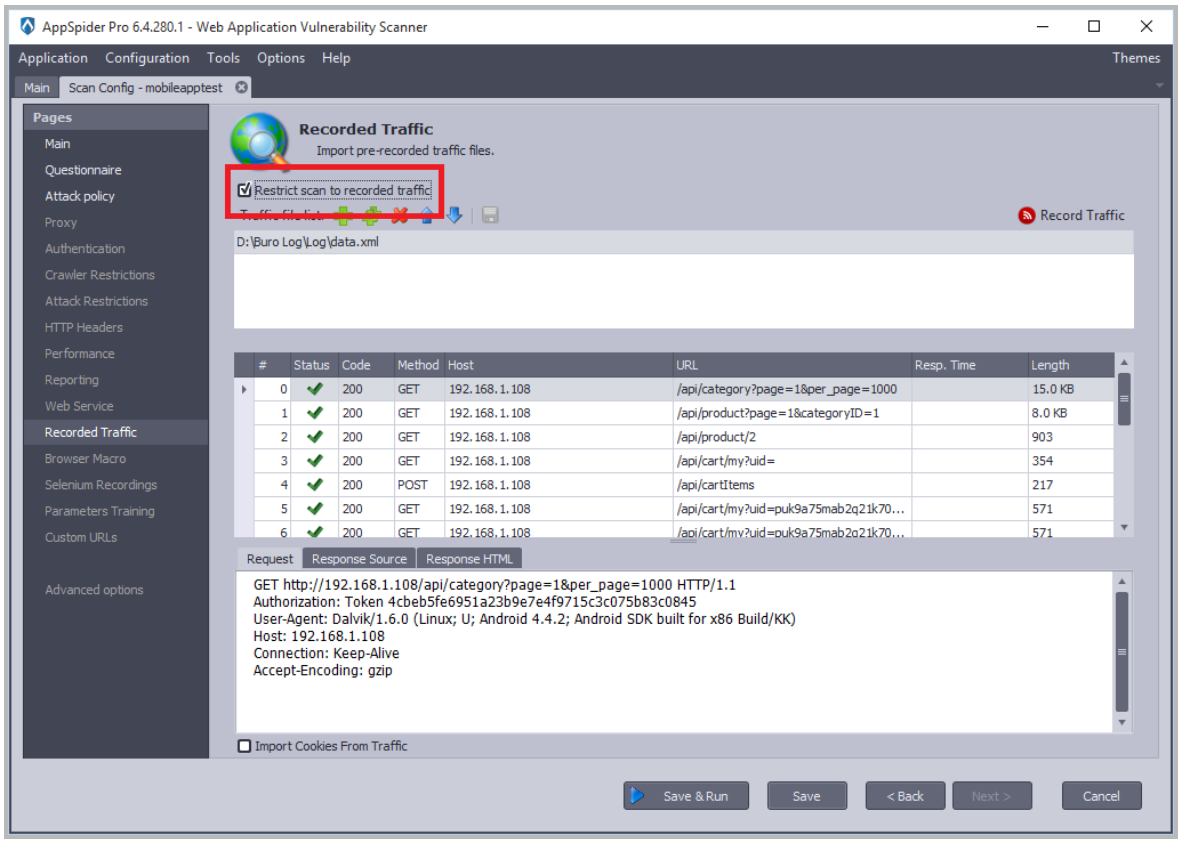
6. Click the **Import Traffic (+)** icon to load the recorded traffic file.



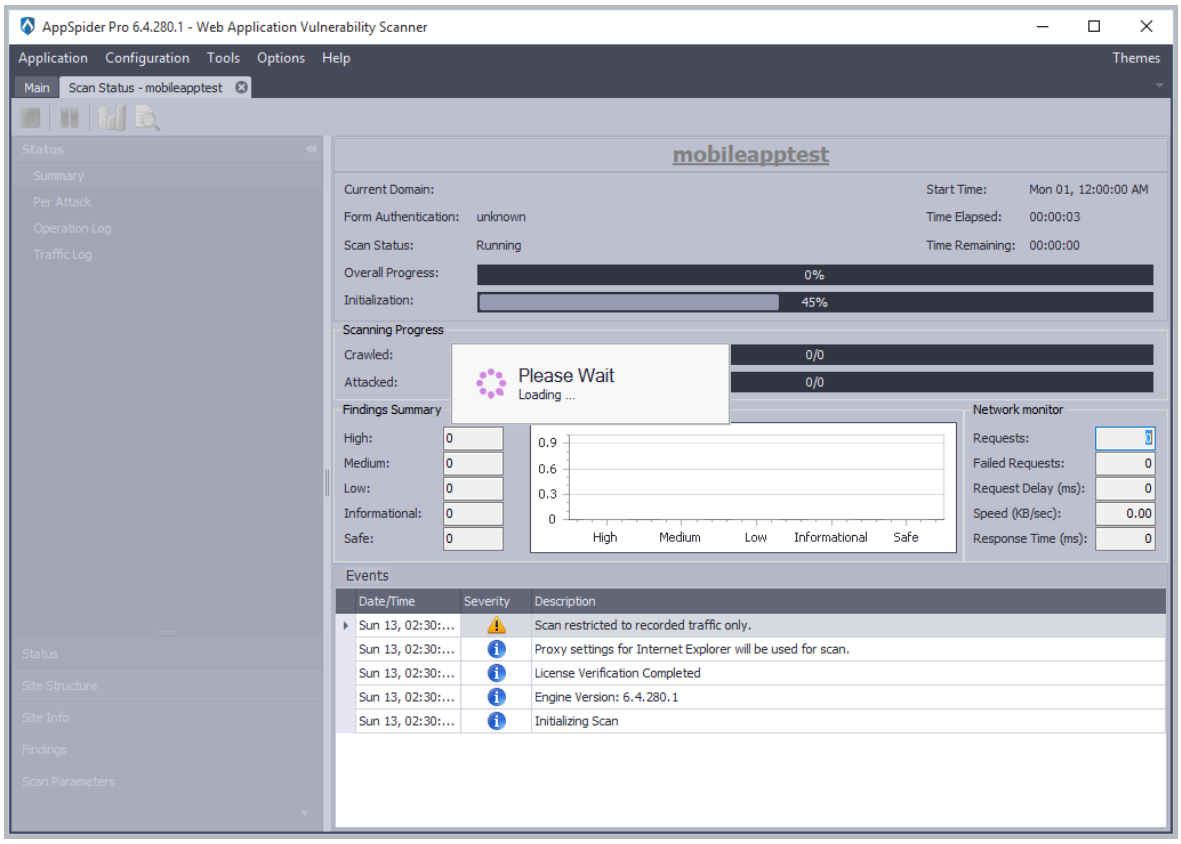
The pre-recorded traffic is now visible in AppSpider.



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

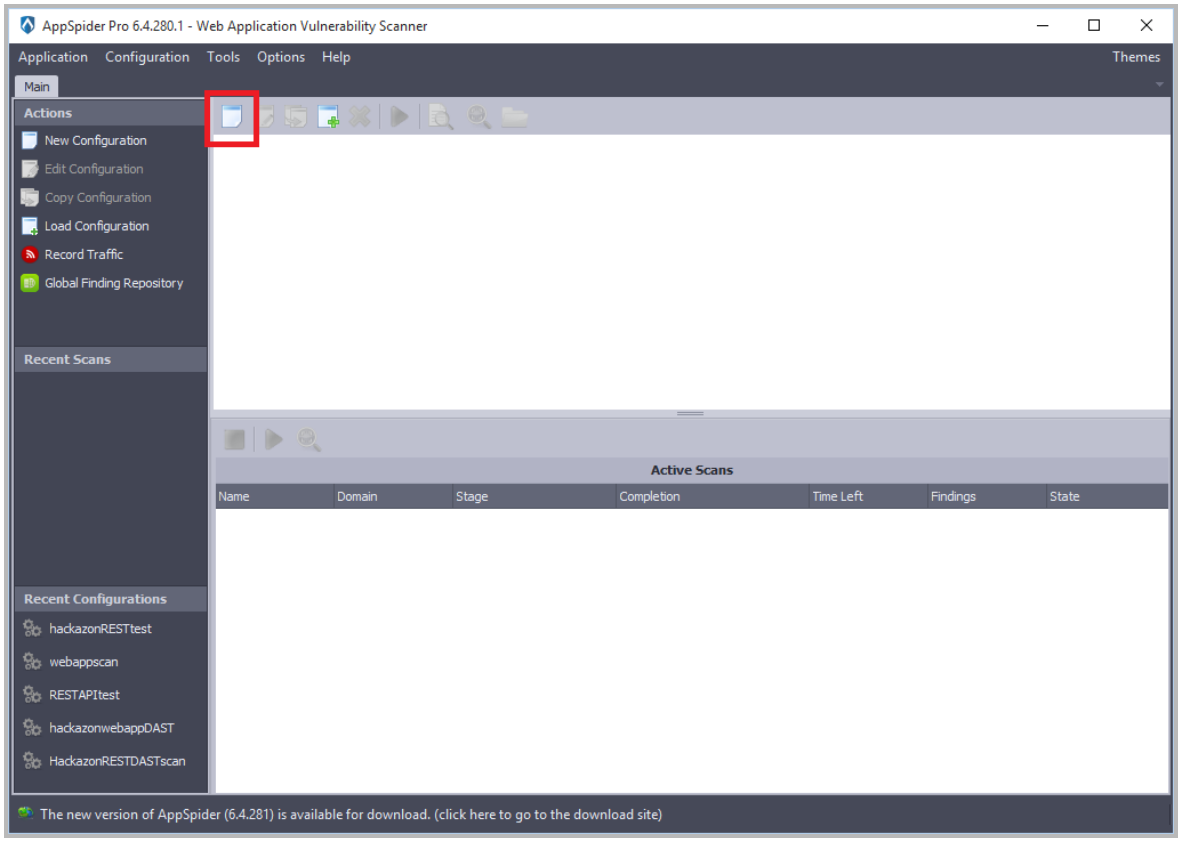


8. Click the **Save and Run** button and AppSpider will start scanning the mobile application.

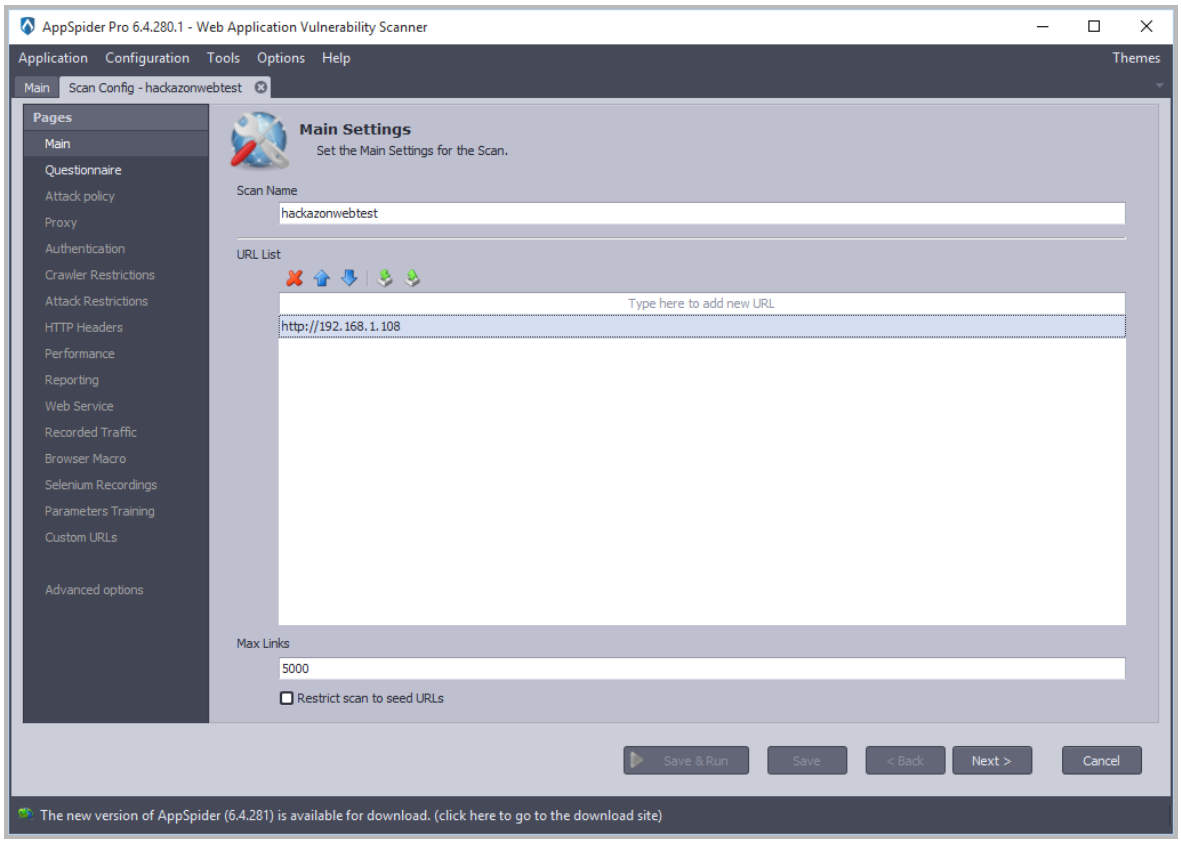

AppSpider Pro 6.4.280.1 - Web Application Vulnerability Scanner
Application
Configuration
Tools
Options
Help
Themes
Main
Scan Status - mobileapptest
Status
Summary
Per Attack
Operation Log
Traffic Log
mobileapptest
Current Domain:
Form Authentication: unknown
Scan Status: Running
Start Time: Mon 01, 12:00:00 AM
Time Elapsed: 00:00:03
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
0.9
0.6
0.3
0
High
Medium
Low
Informational
Safe
Network monitor
Requests: 0
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:...
Proxy settings for Internet Explorer will be used for scan.
Sun 13, 02:30:...
License Verification Completed
Sun 13, 02:30:...
Engine Version: 6.4.280.1
Sun 13, 02:30:...
Initializing Scan
Status
Site Structure
Site Info
Findings
Scan Parameters

How to test the Hackazon web application using AppSpider

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

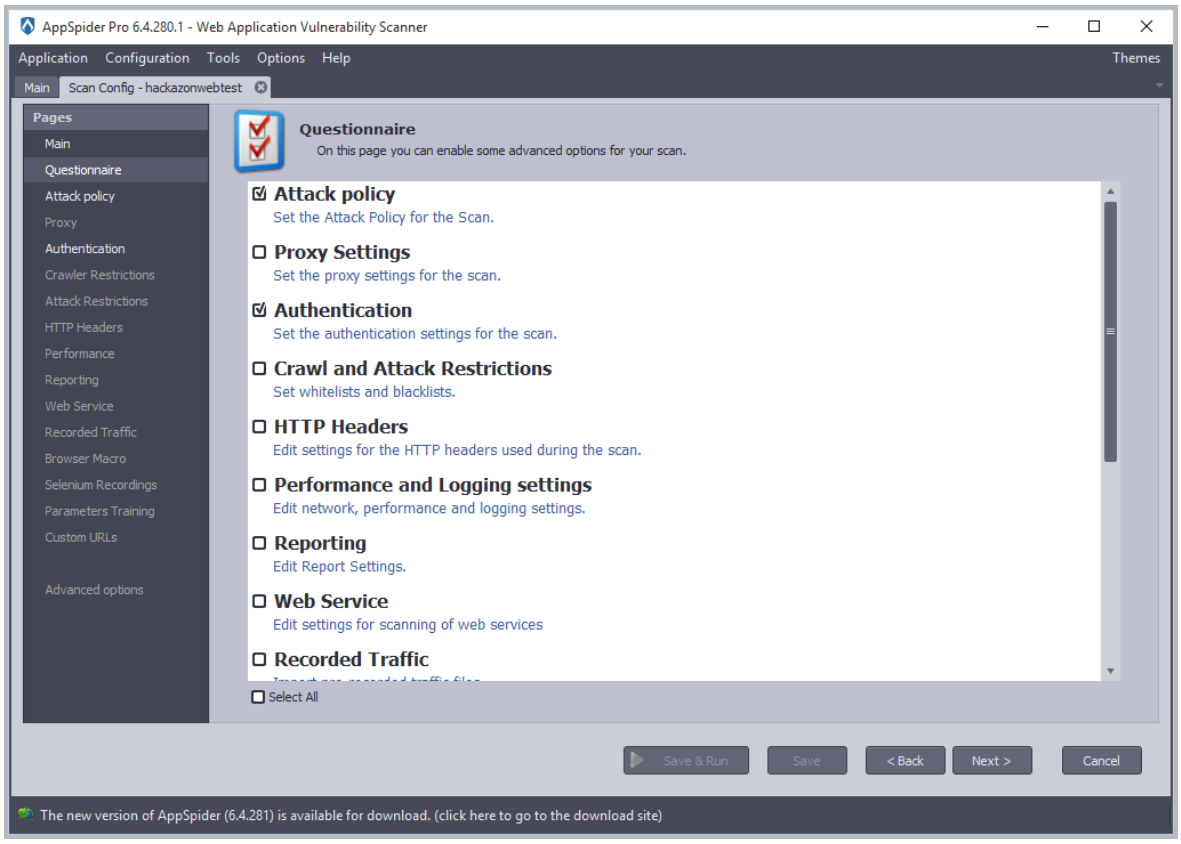


3. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.



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.

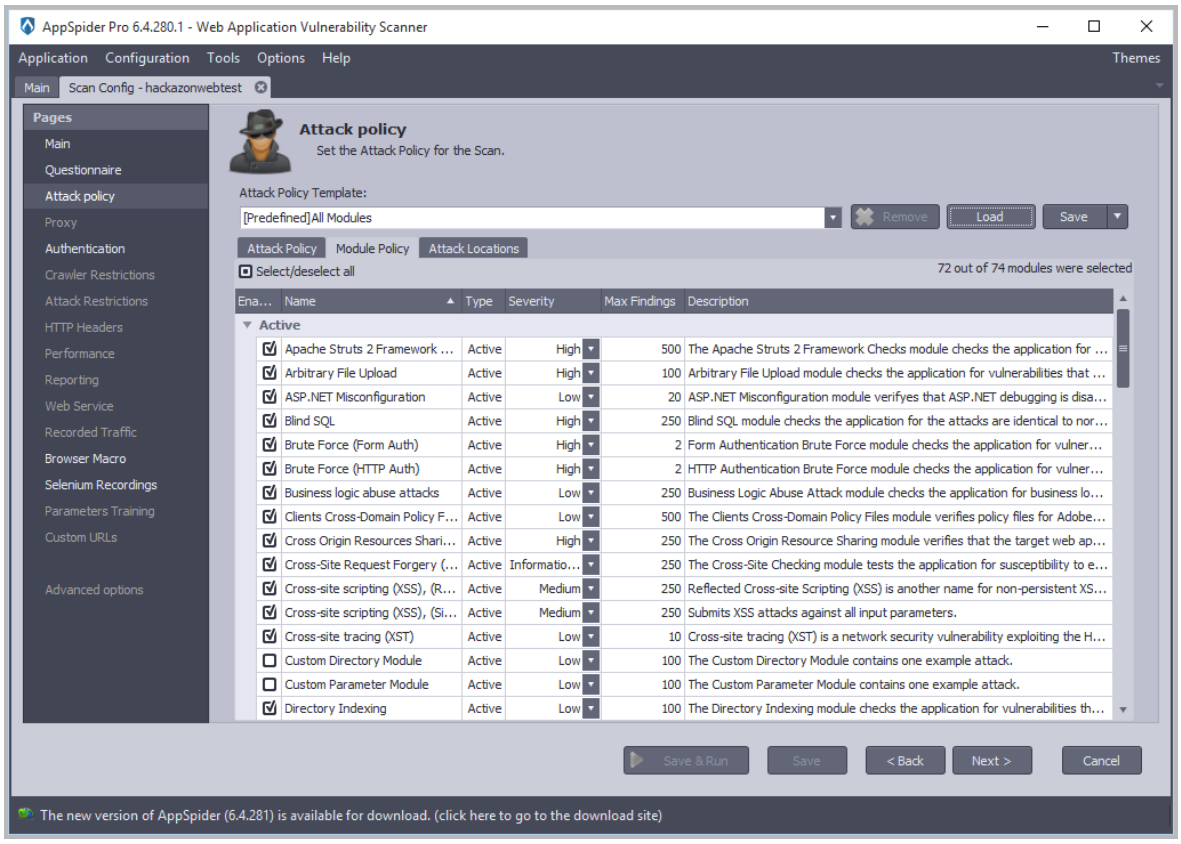


Attack policy

An attack policy can contain over 80 different attack modules. You can choose from any of AppSpider's predifined 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.

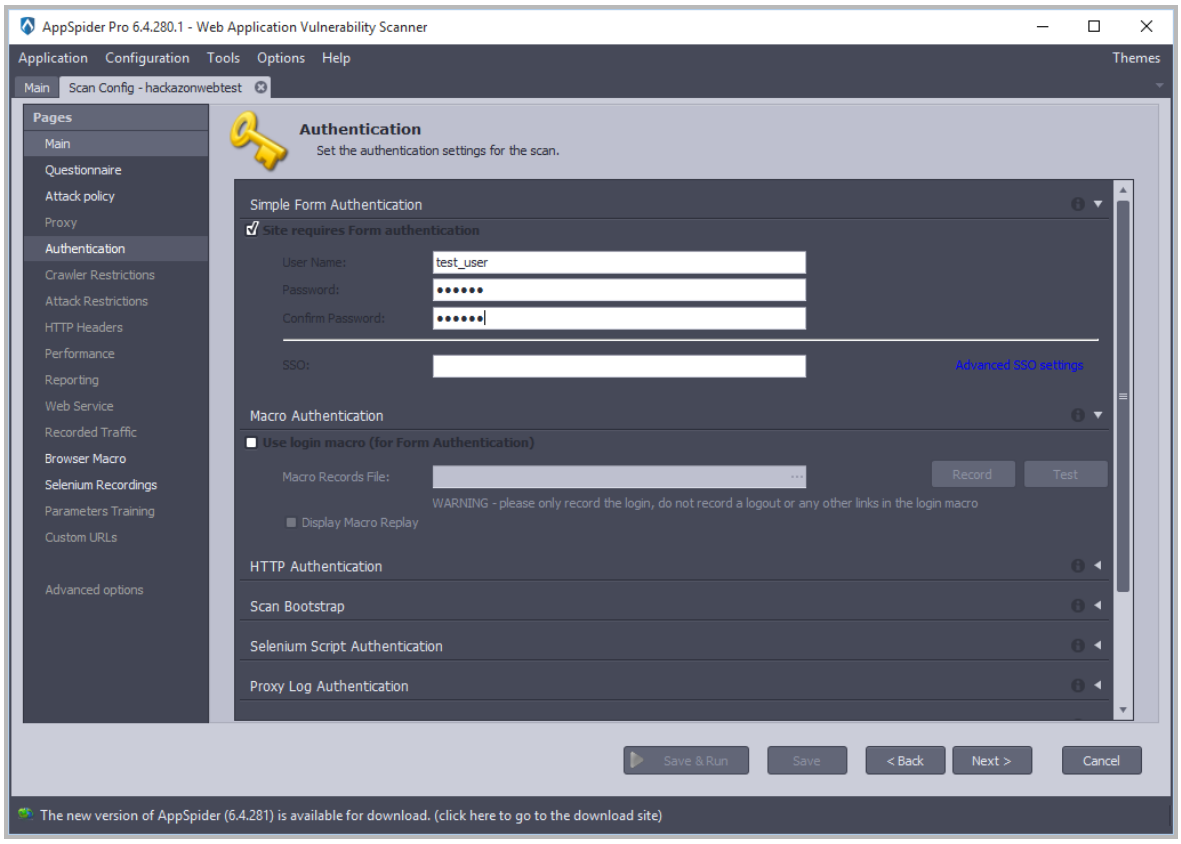
1. Select the **All Modules** template, to test the Hackazon application against all threats, and click the **Load** button.
2. Click on the **Next** button.



Authentication

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

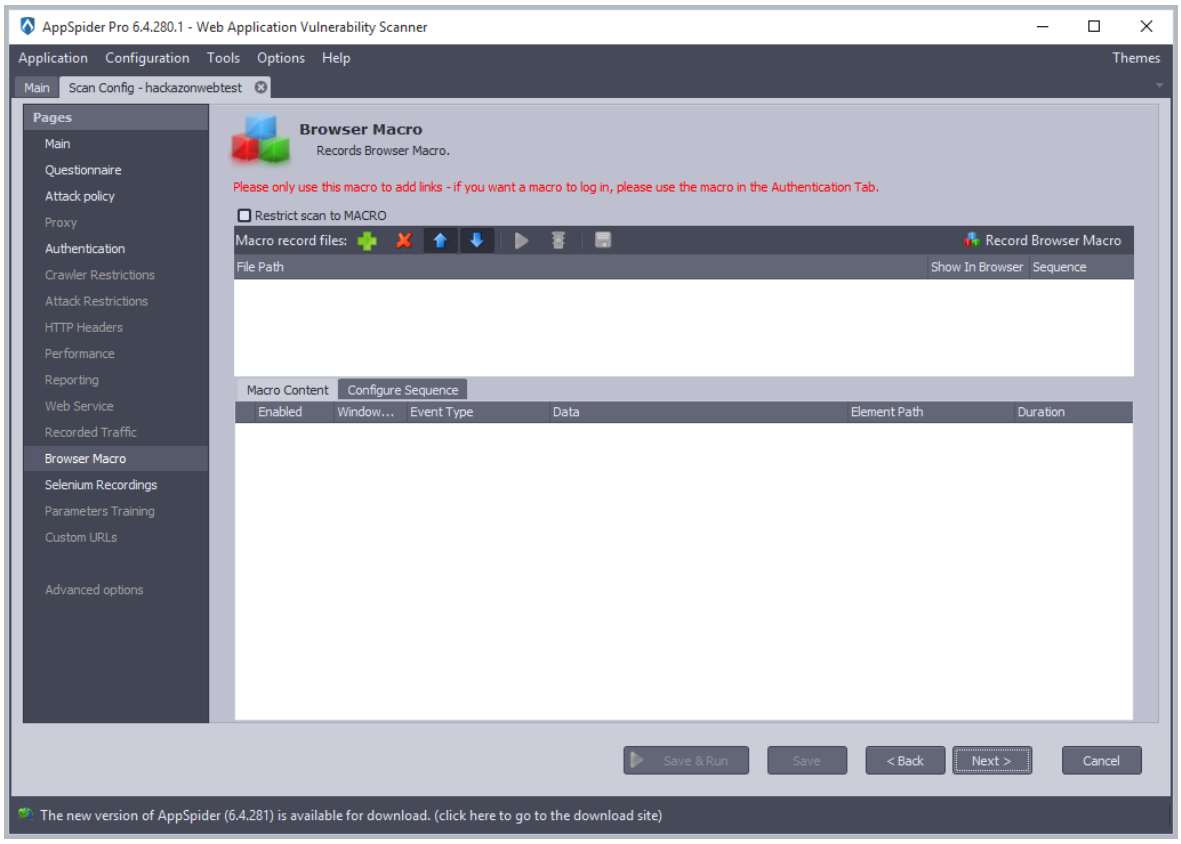
1. Select **Simple Form Authentication**, as the Hackazon application is using an HTML form-based authentication technique.
2. Enter **User Name** and **Password** then click the **Next** button.



Browser Macro

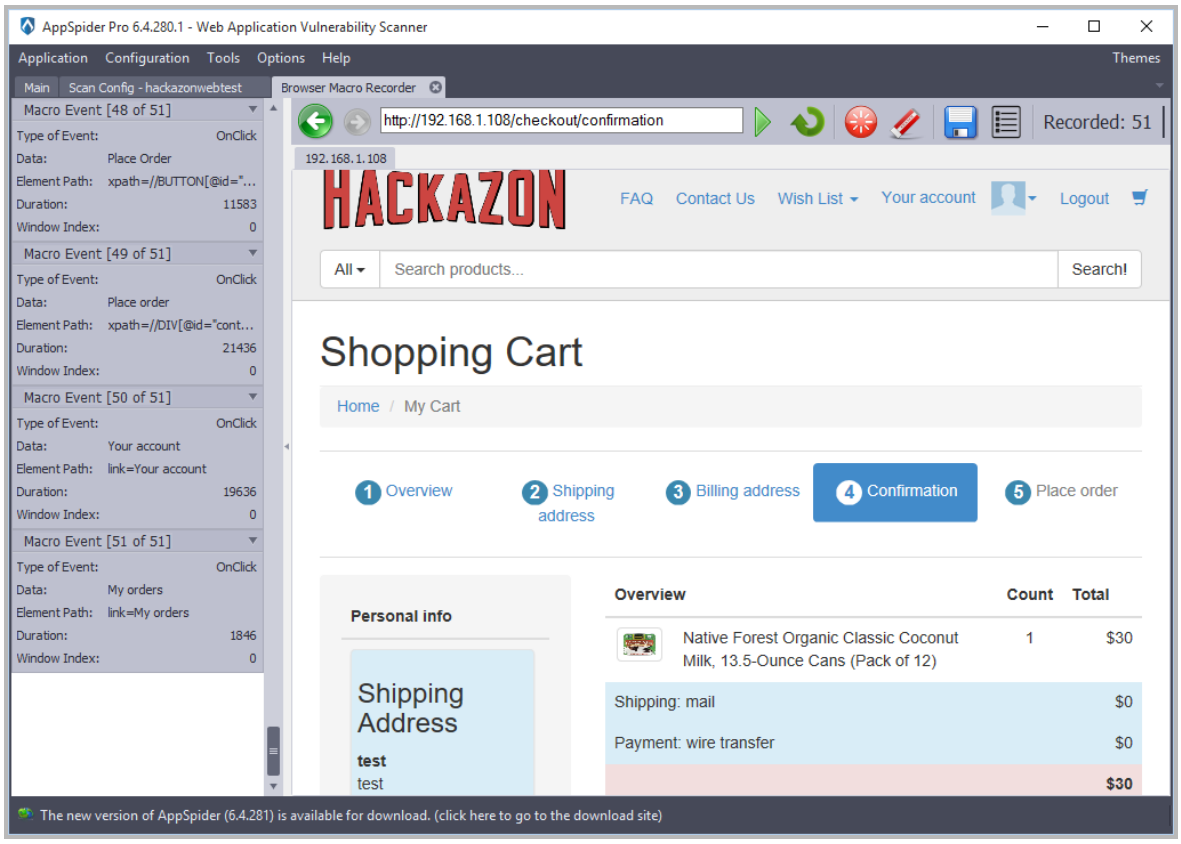
A macro is a sequence of actions (e.g. menu selections, link executions, value entries, etc.) that get replayed exactly as they were inputted by the user. The browser macro in AppSpider allows you to record or import pre-recorded macro files.

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

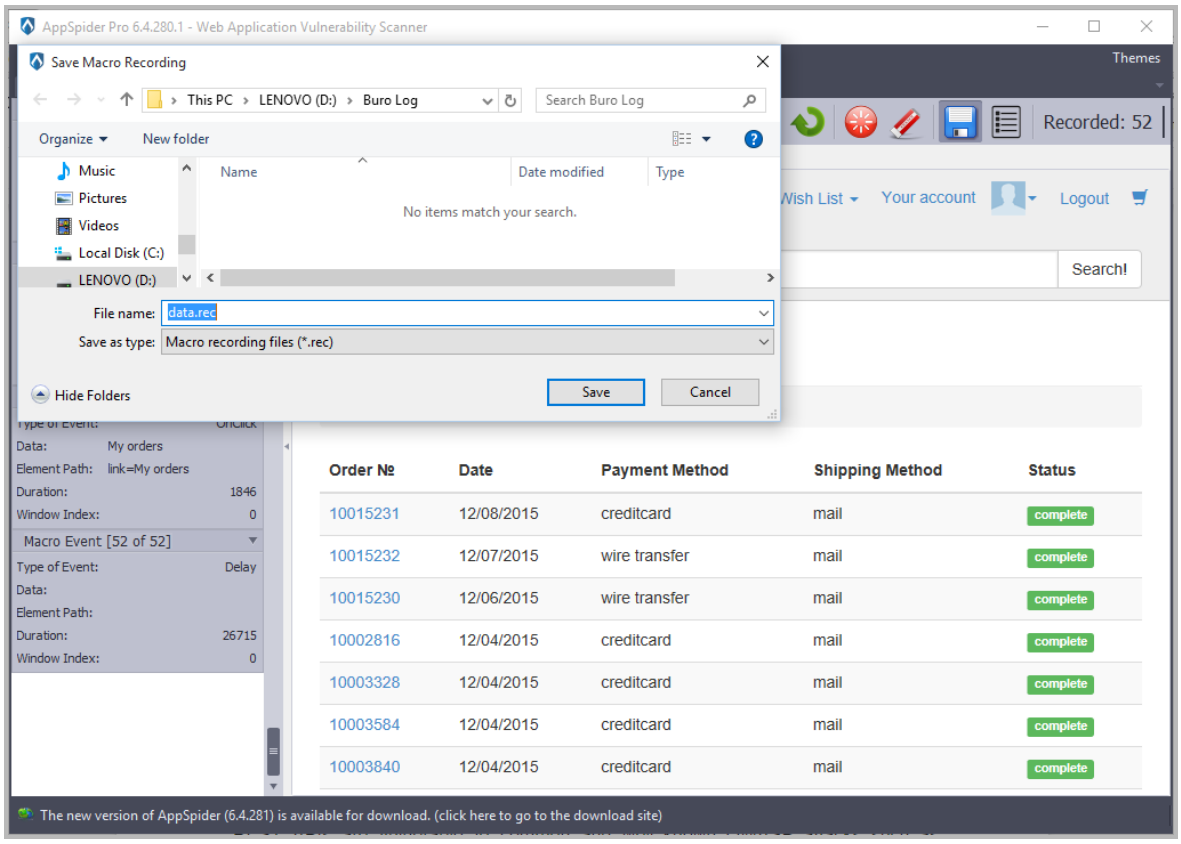


2. Manually crawl the application.

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

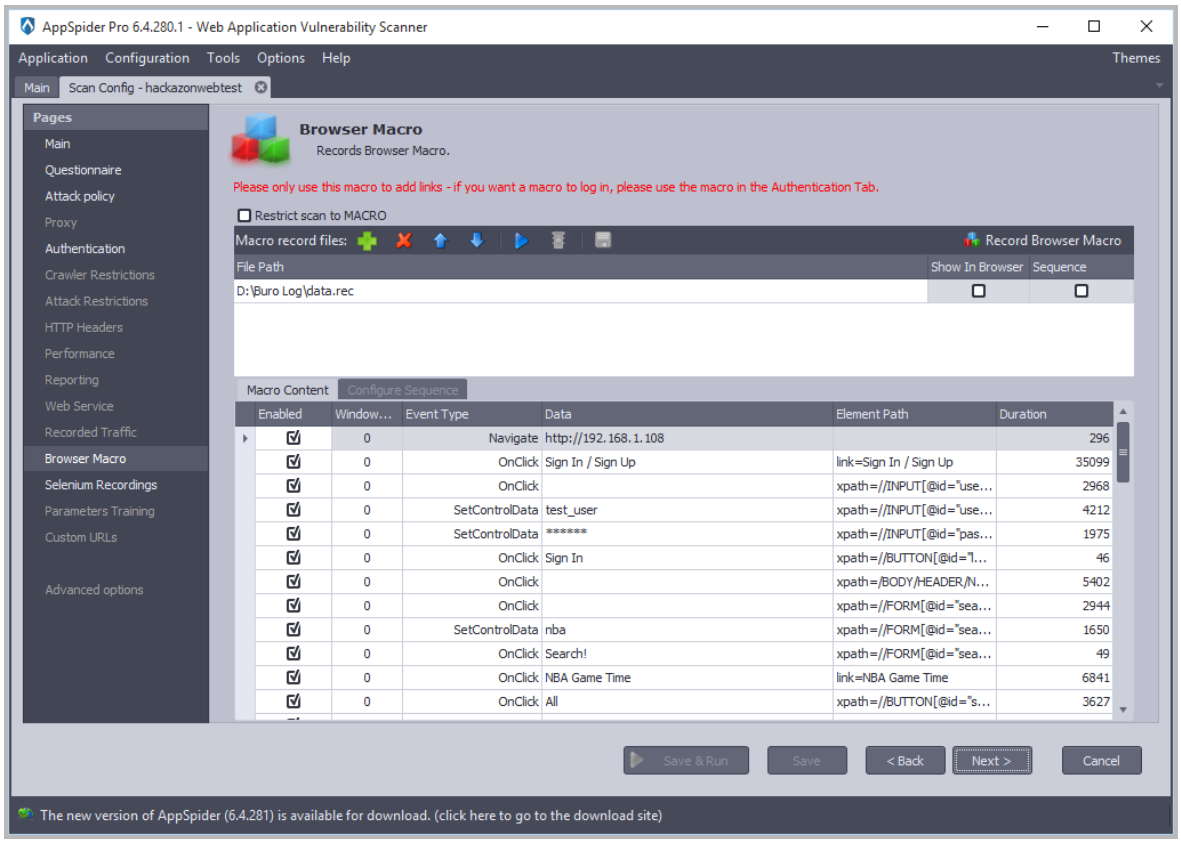


3. When you are finished, click the **Save** icon and save the macro recording file.



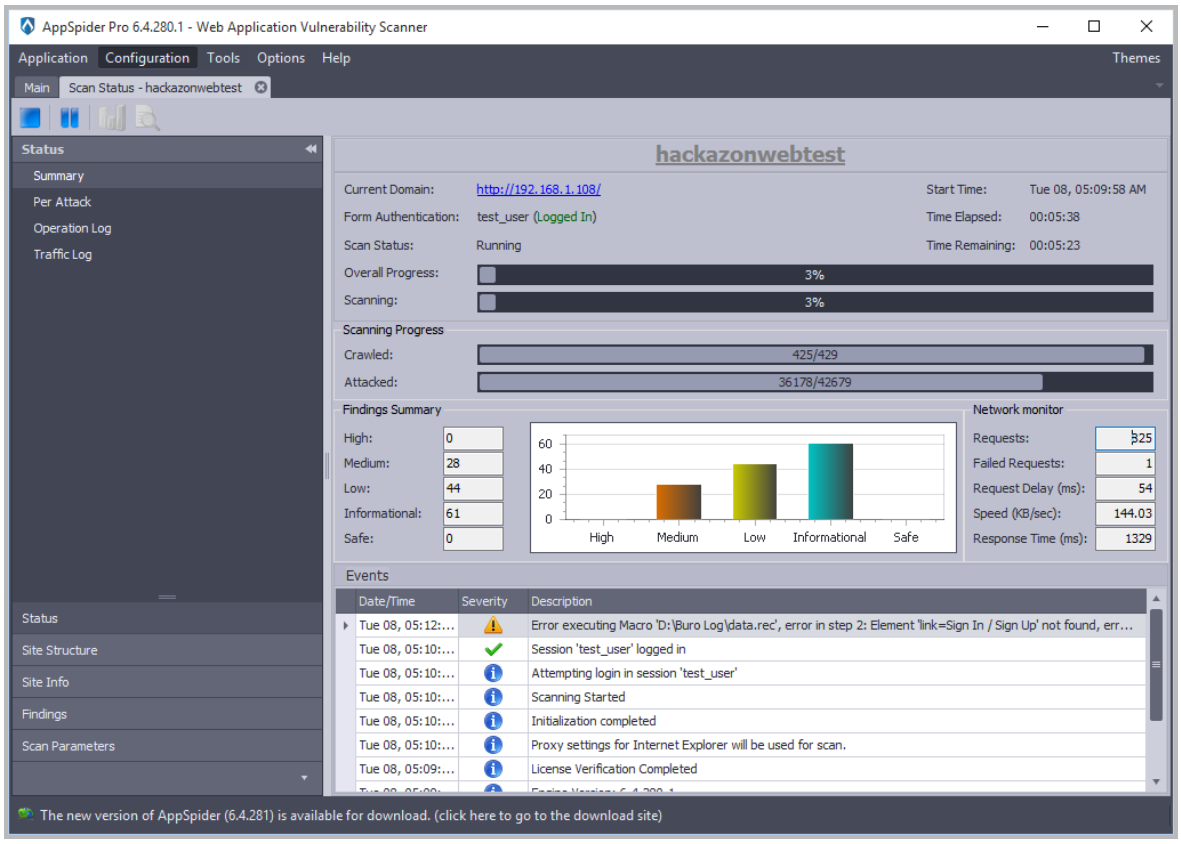
The macro recording file will be imported automatically into AppSpider and the recorded HTTP traffic will be visible in the *Browser Macro* panel.

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

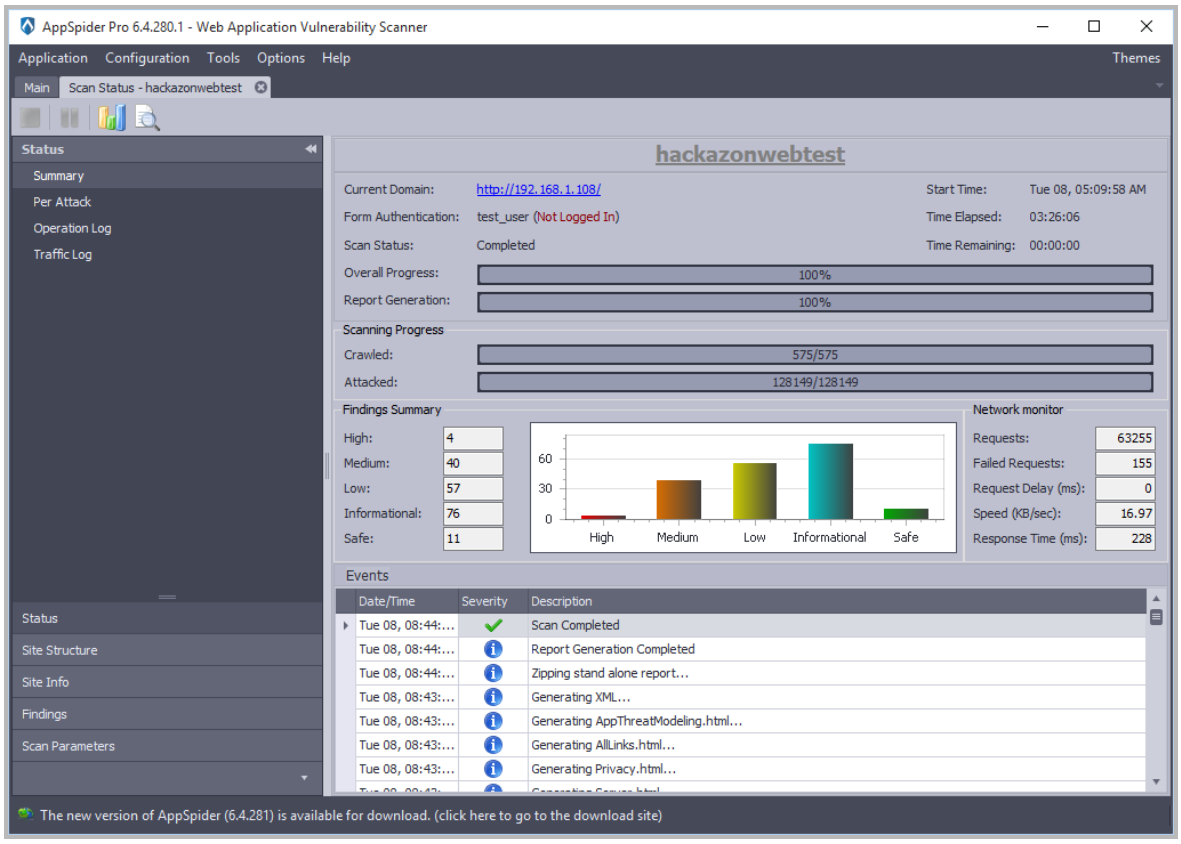


Scan summary

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

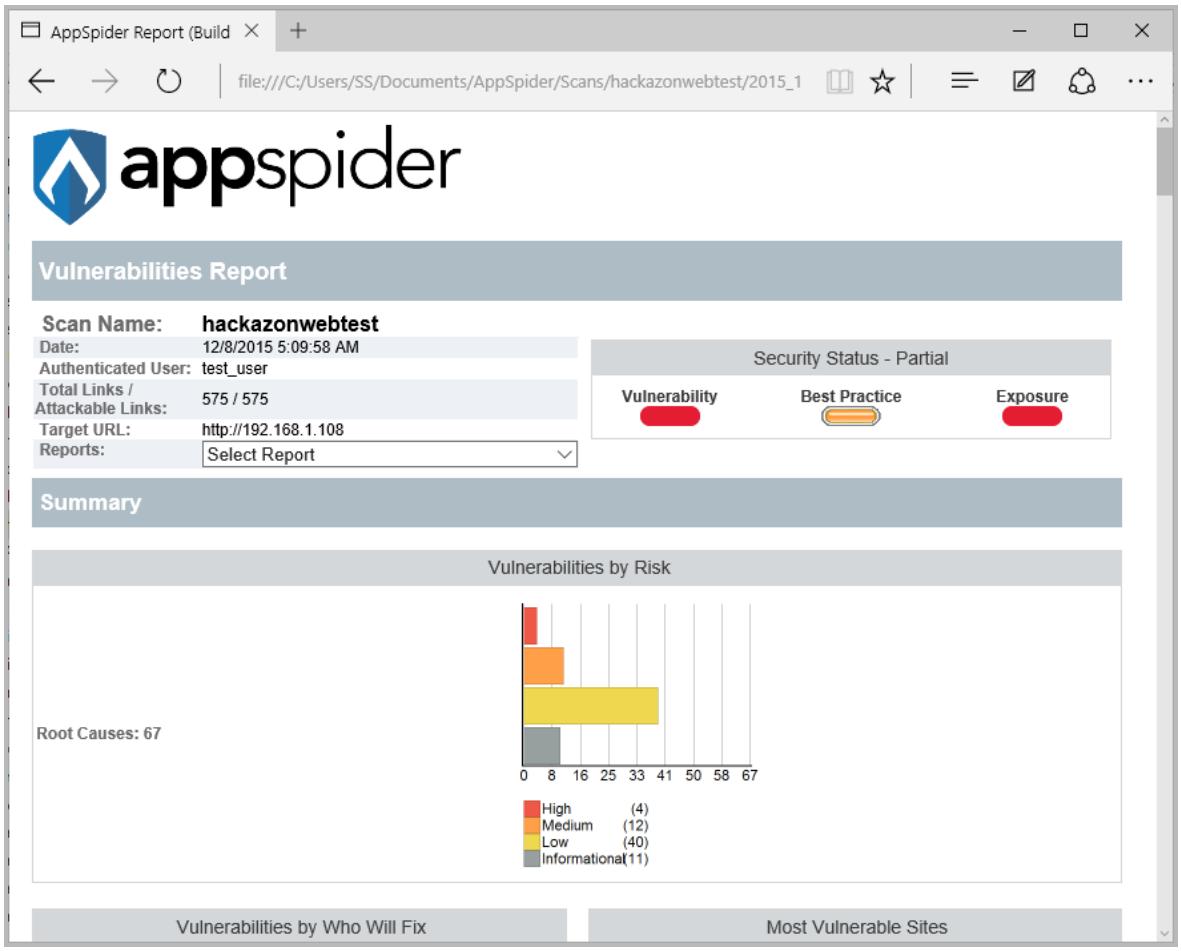


When the scan is complete, AppSpider will provide a summary of findings. For this scan, AppSpider found 4 High, 40 Medium, 57 Low, and 76 Informational vulnerabilities during the scan.

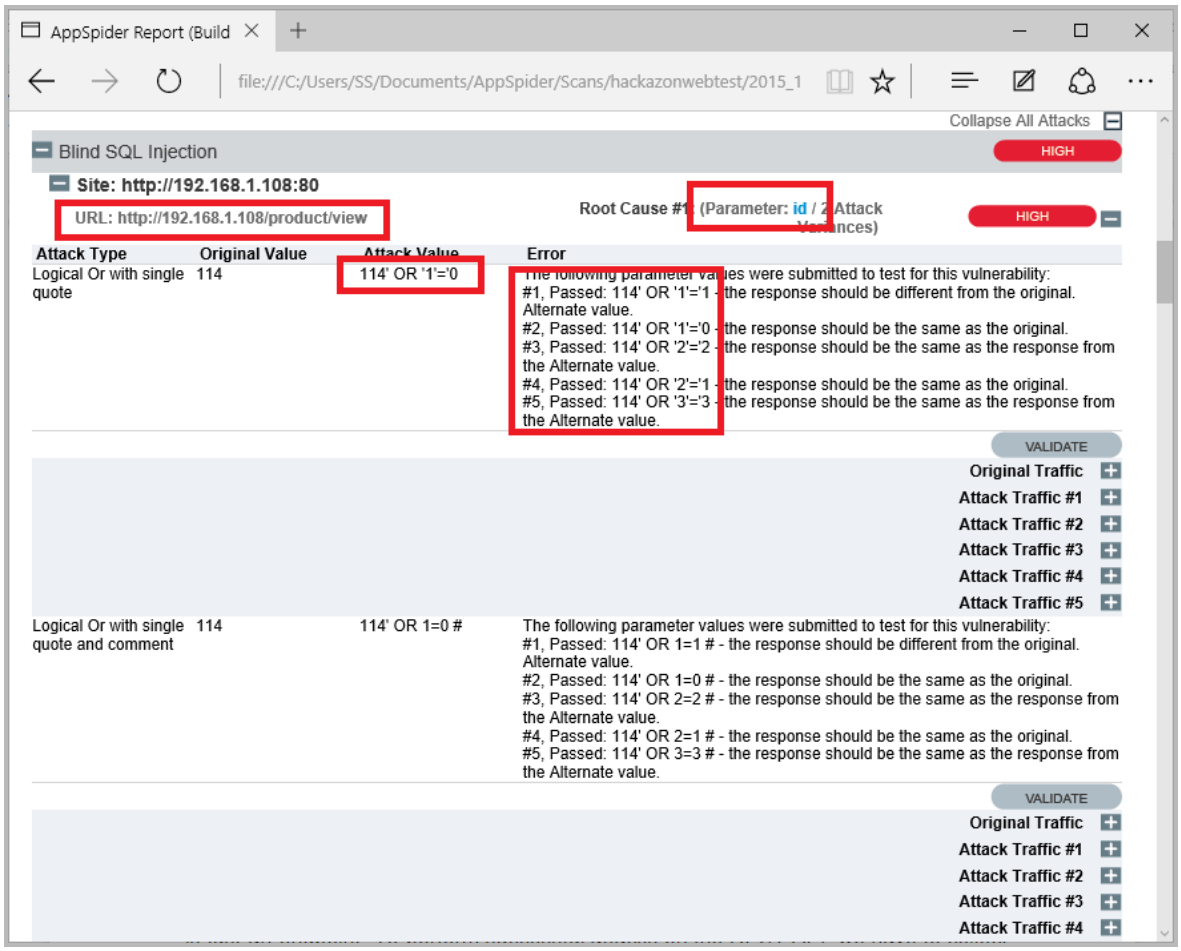


Reporting

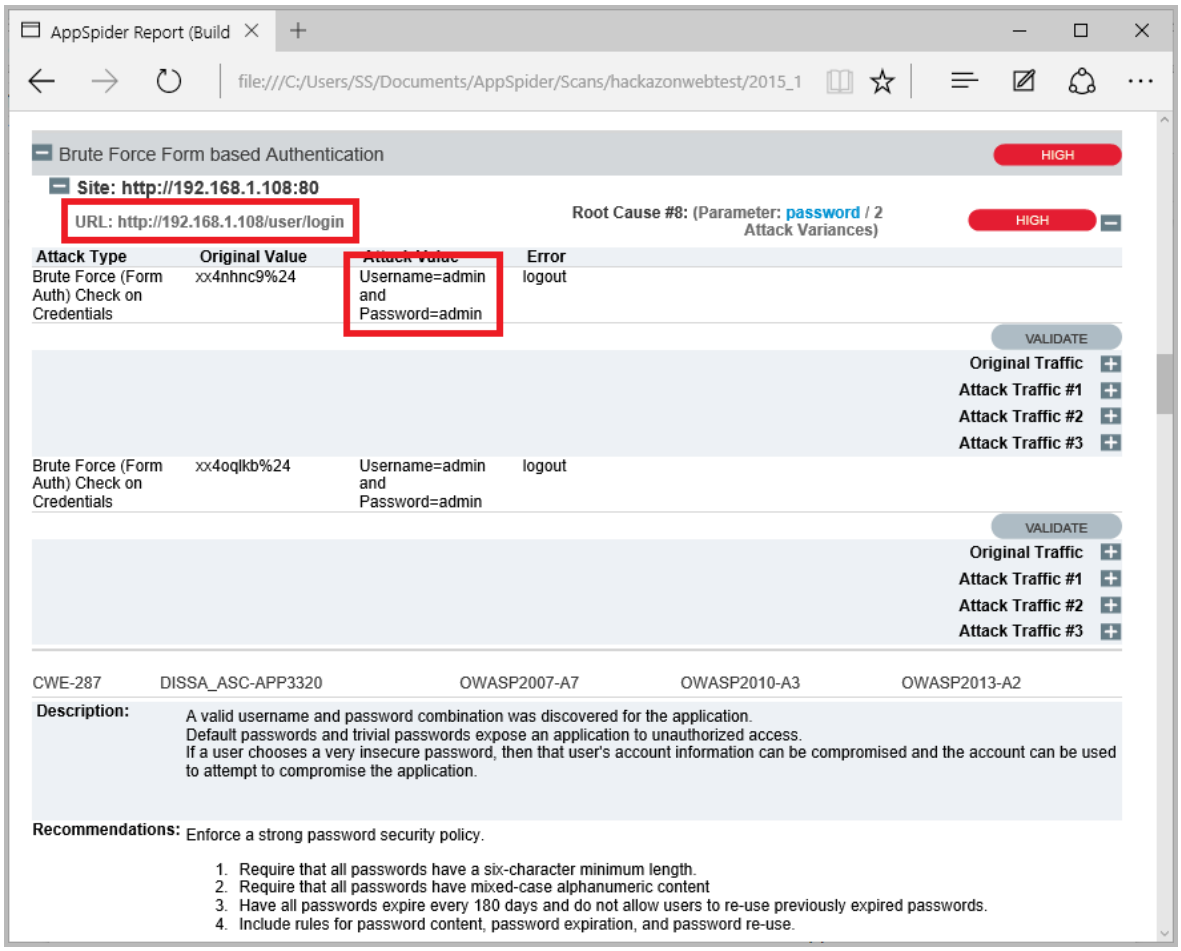
AppSpider also generates an HTML report after every completed scan.



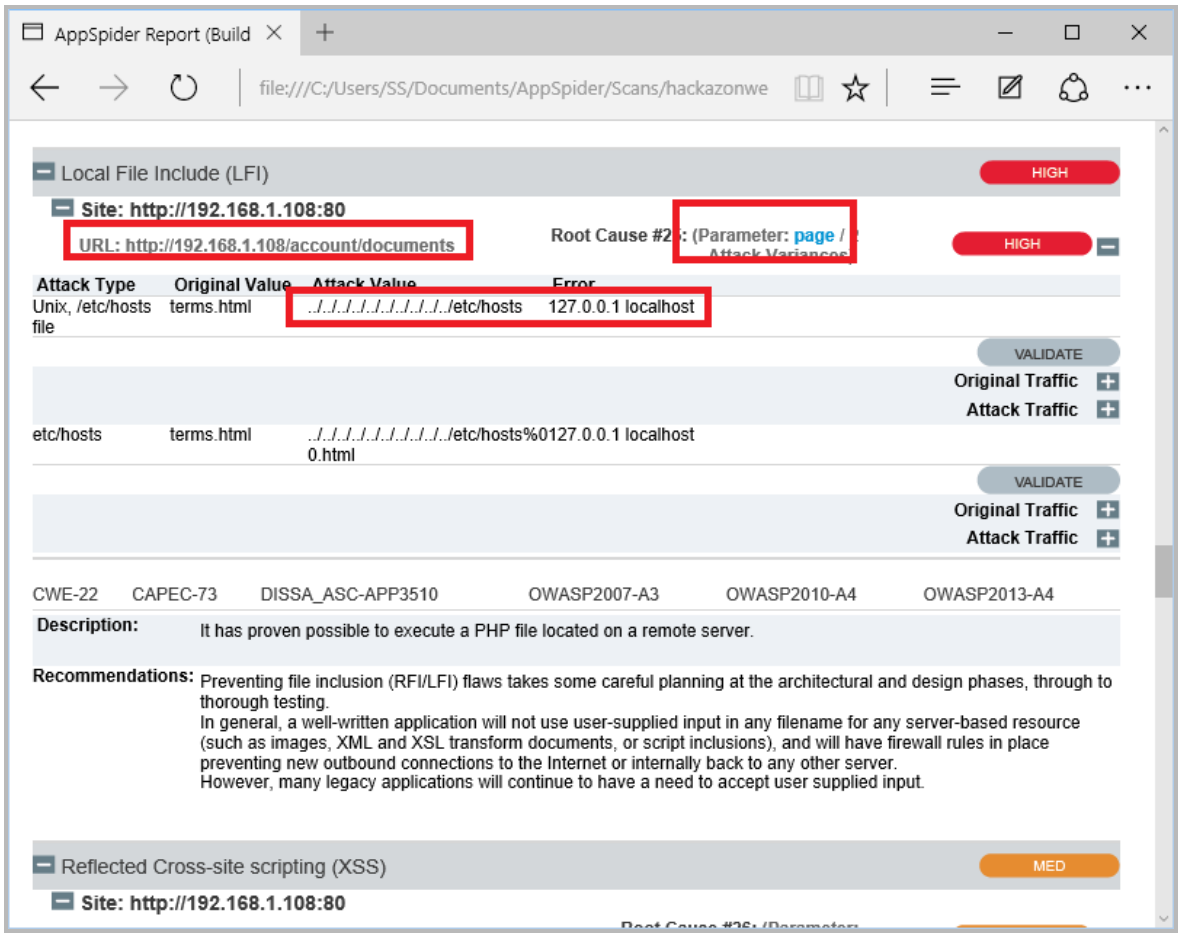
AppSpider found a blind SQL injection attack. Based on the application response, the attack asks the database true or false questions and determines the answer. AppSpider used logical OR with single quote to identify the vulnerability.



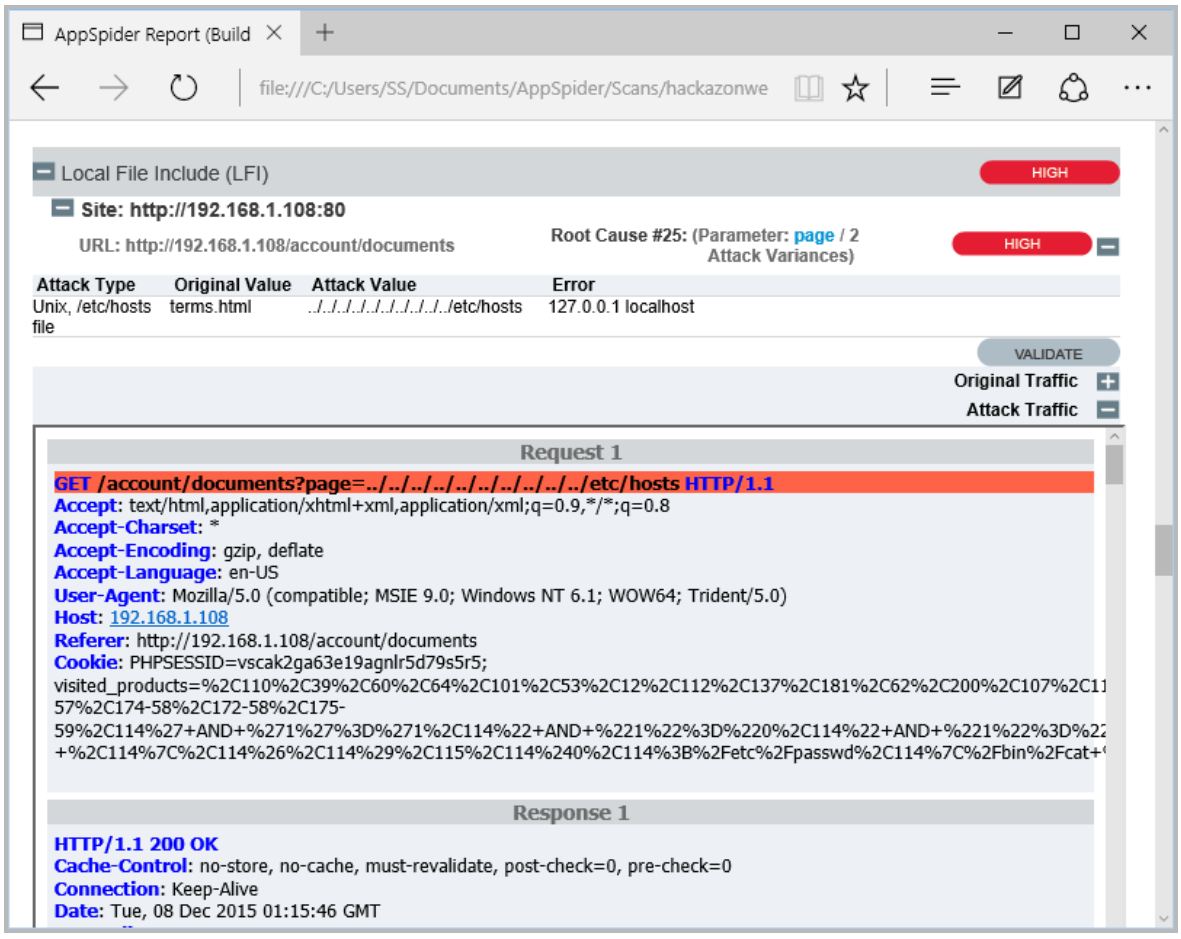
In another case, AppSpider has found login credentials by way of a brute force attack. The default username and password for the Hackazon application is admin (Username) and admin (Password). For this attack, AppSpider tried commonly-used usernames and passwords to guess as to the value of the desired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



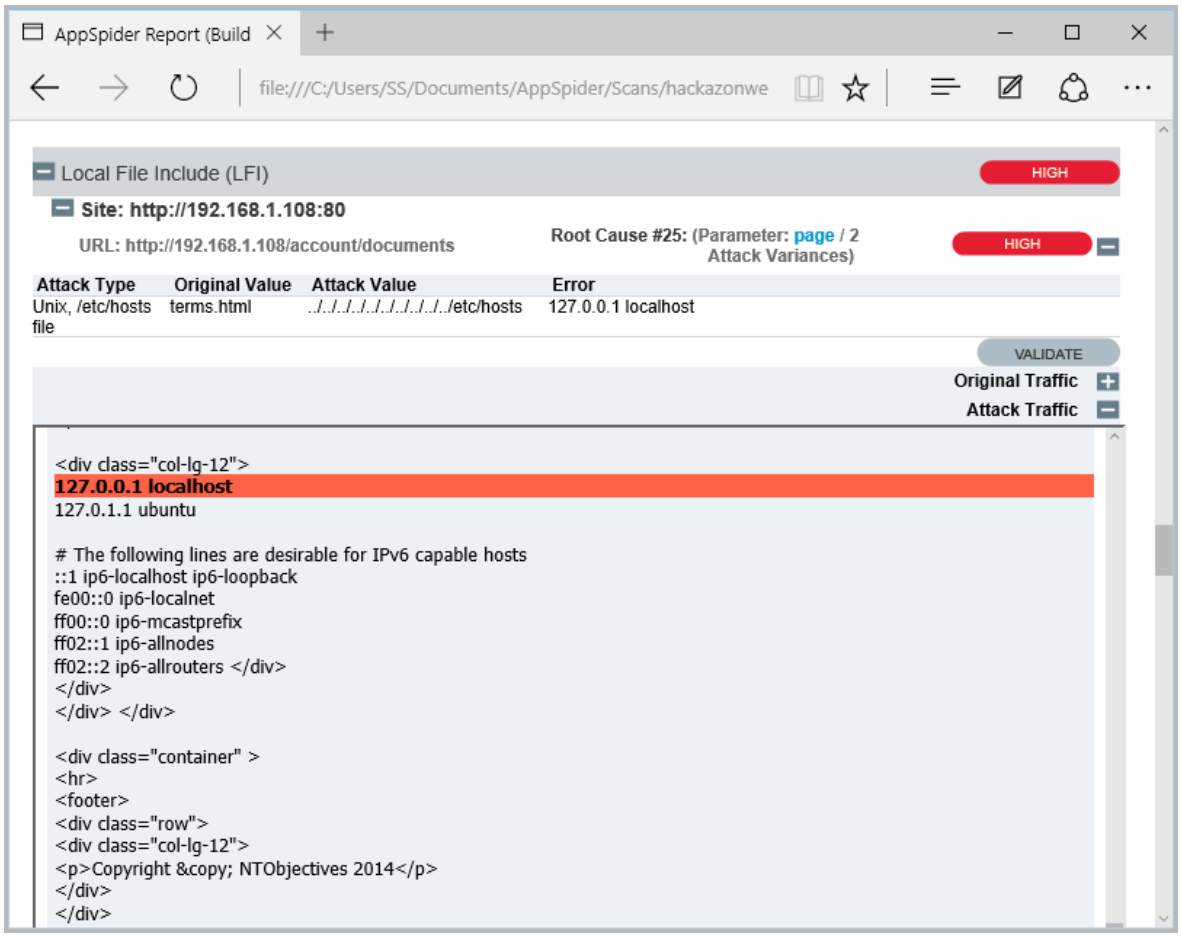
AppSpider also found a Local File Inclusion vulnerability (LFI). The File Inclusion vulnerability allows an attacker to include a file, usually exploiting dynamic file inclusion mechanisms implemented in the target application.



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



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
file:///C:/Users/SS/Documents/AppSpider/Scans/hackazonwe
Local File Include (LFI)
HIGH
Site: http://192.168.1.108:80
URL: http://192.168.1.108/account/documents
Root Cause #25: (Parameter: page / 2 Attack Variances)
HIGH
Attack Type
Original Value
Attack Value
Error
Unix, /etc/hosts file
terms.html
../../../../../../../../../etc/hosts
127.0.0.1 localhost
VALIDATE
Original Traffic
Attack Traffic

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

<hr>
<footer>

Copyright © NTObjectives 2014

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.

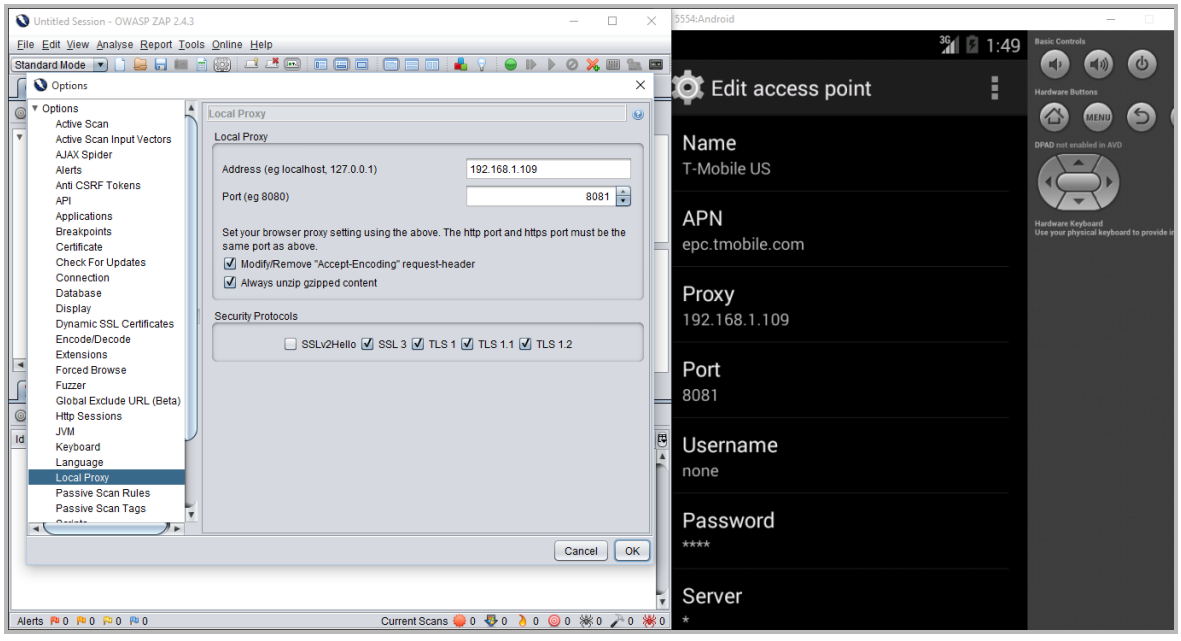
REST APIs are vulnerable to common and well known OWASP attacks such as injection, CSRF, cross-site scripting, XML External Entity attacks, etc.

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:

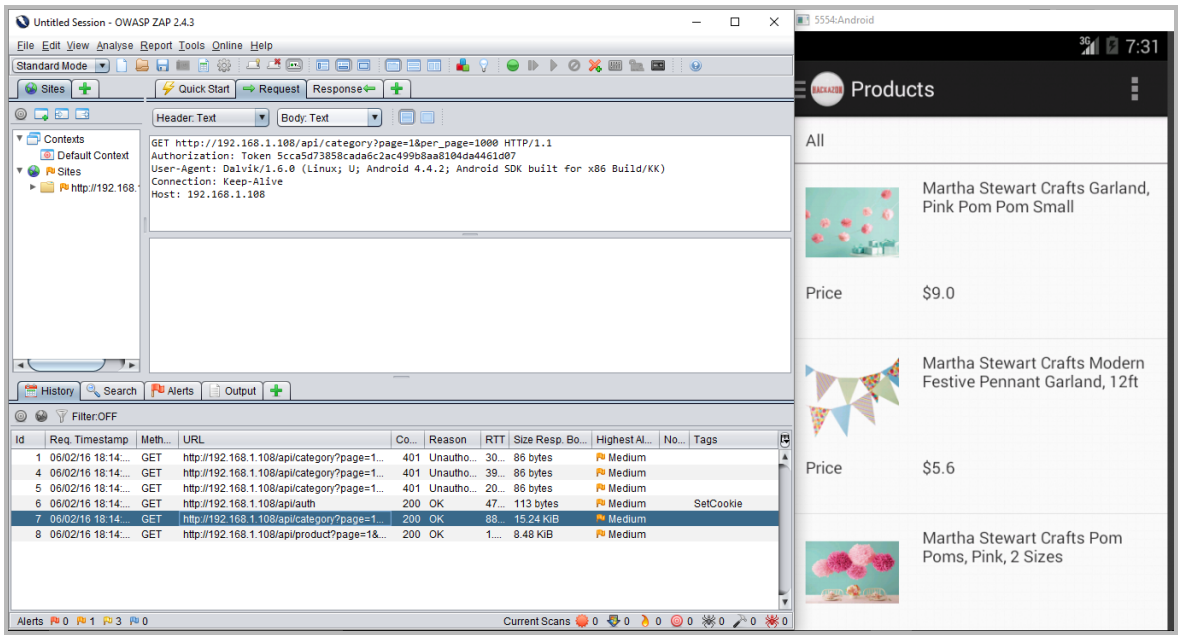
OWASP ZAP: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

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

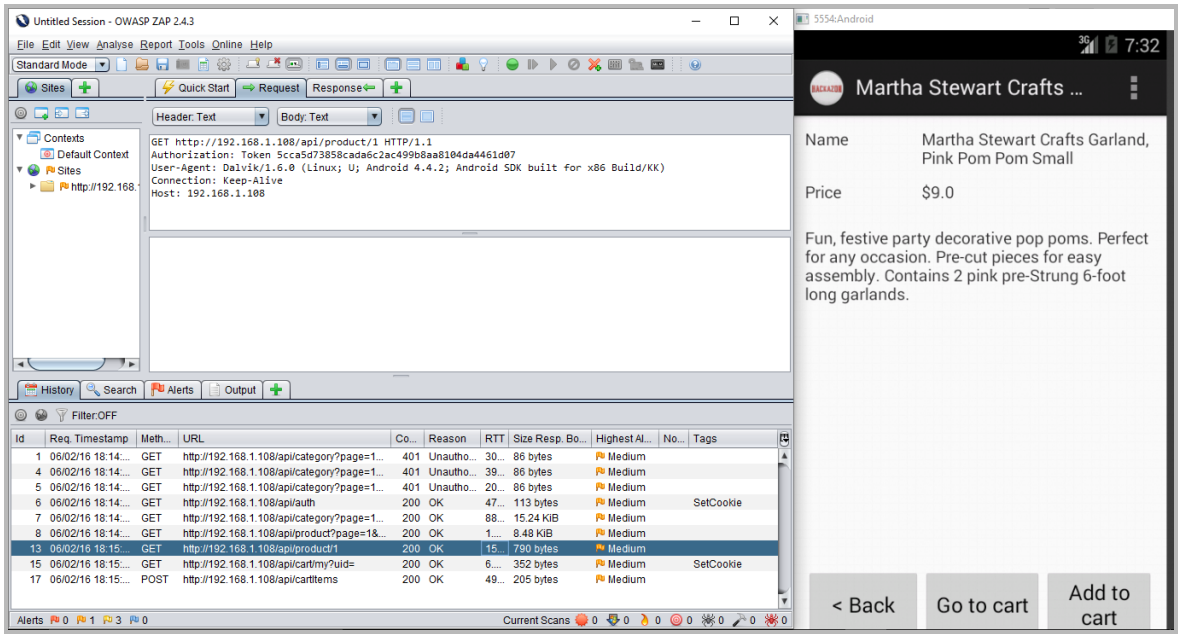
Example of proxy setup in OWASP ZAP and Android emulator.



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



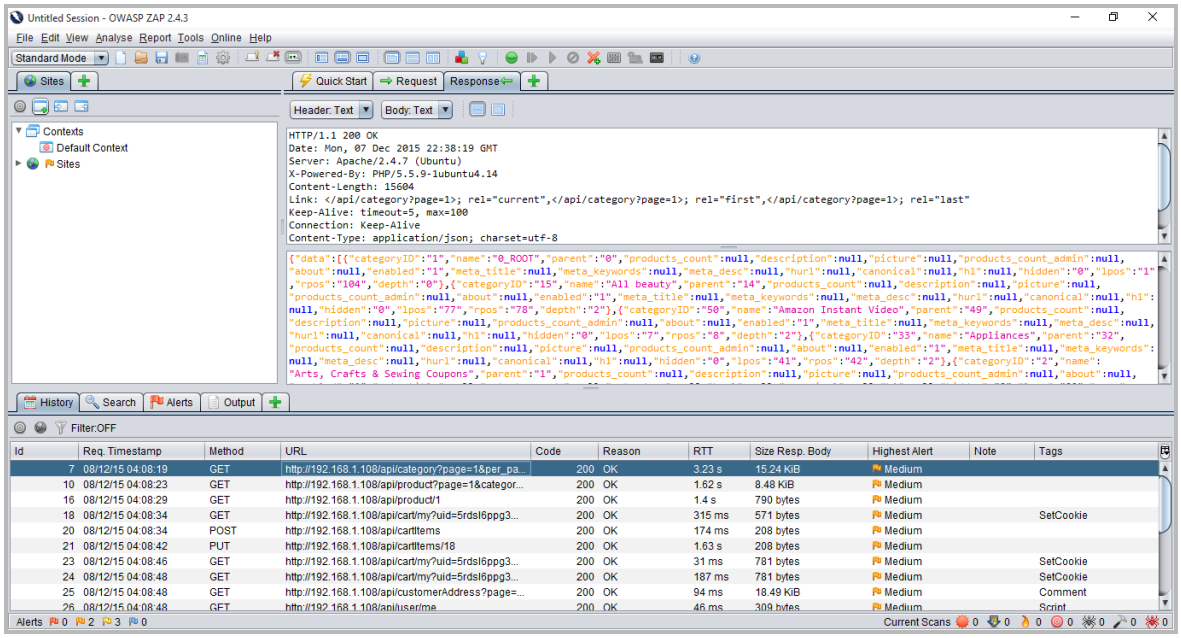
The REST data is being captured by OWASP ZAP.



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.



Blind SQL injection

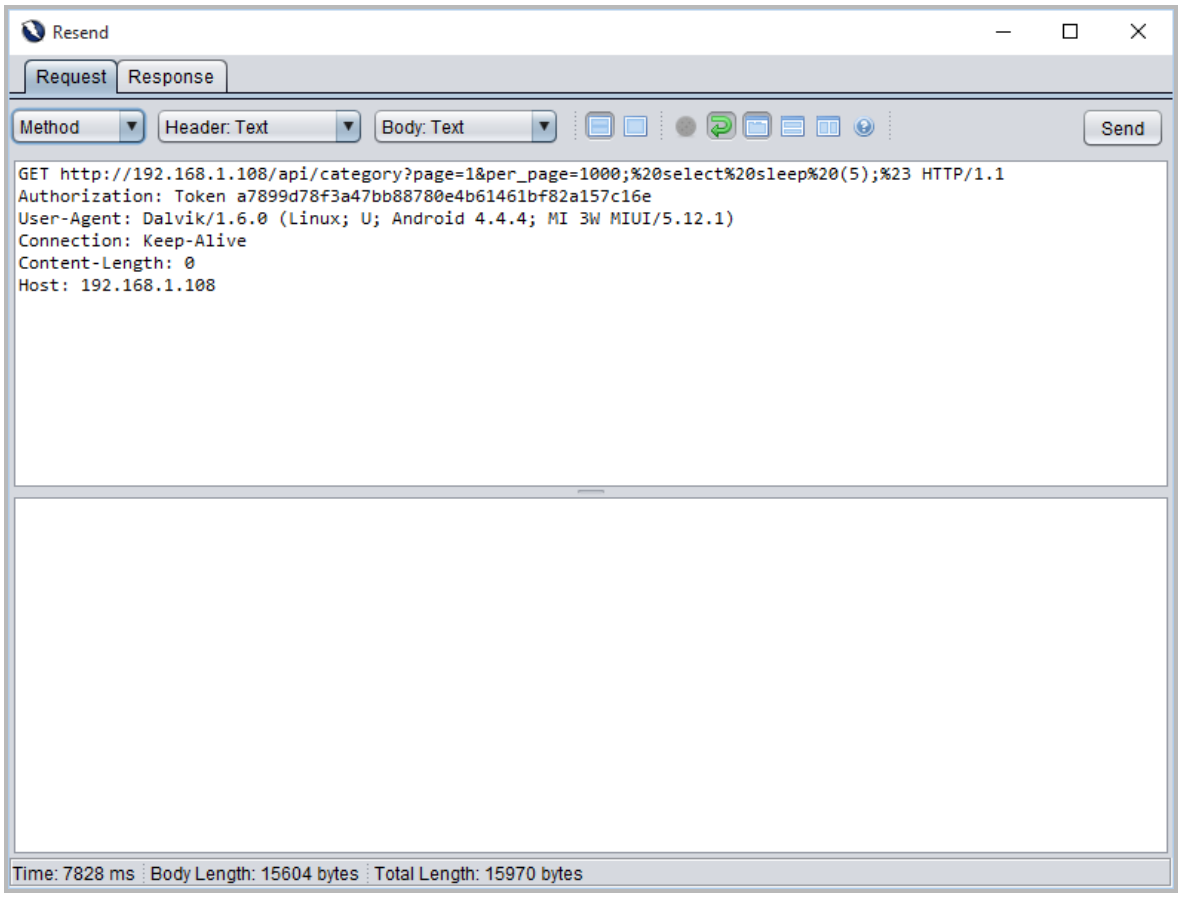
The following example demonstrates Time-Based blind SQL injection in the REST API. The request contains two parameter values, page and per_page. Based on the responses of the following three requests, we can conclude that the per_page parameter is vulnerable to.

URL: http://192.168.1.108/api/category?page=1&per_page=2
Parameter name: per_page
Attack values:

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

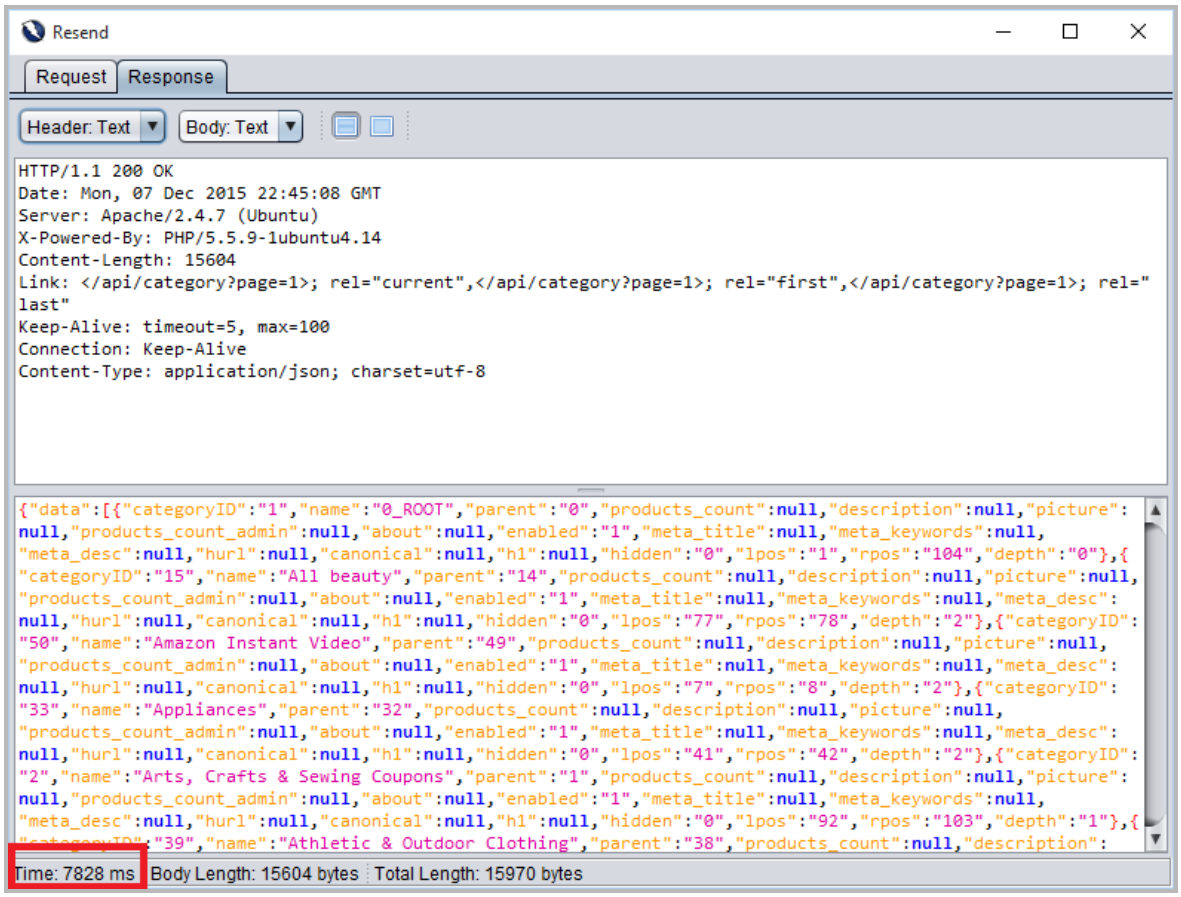
Request 1

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



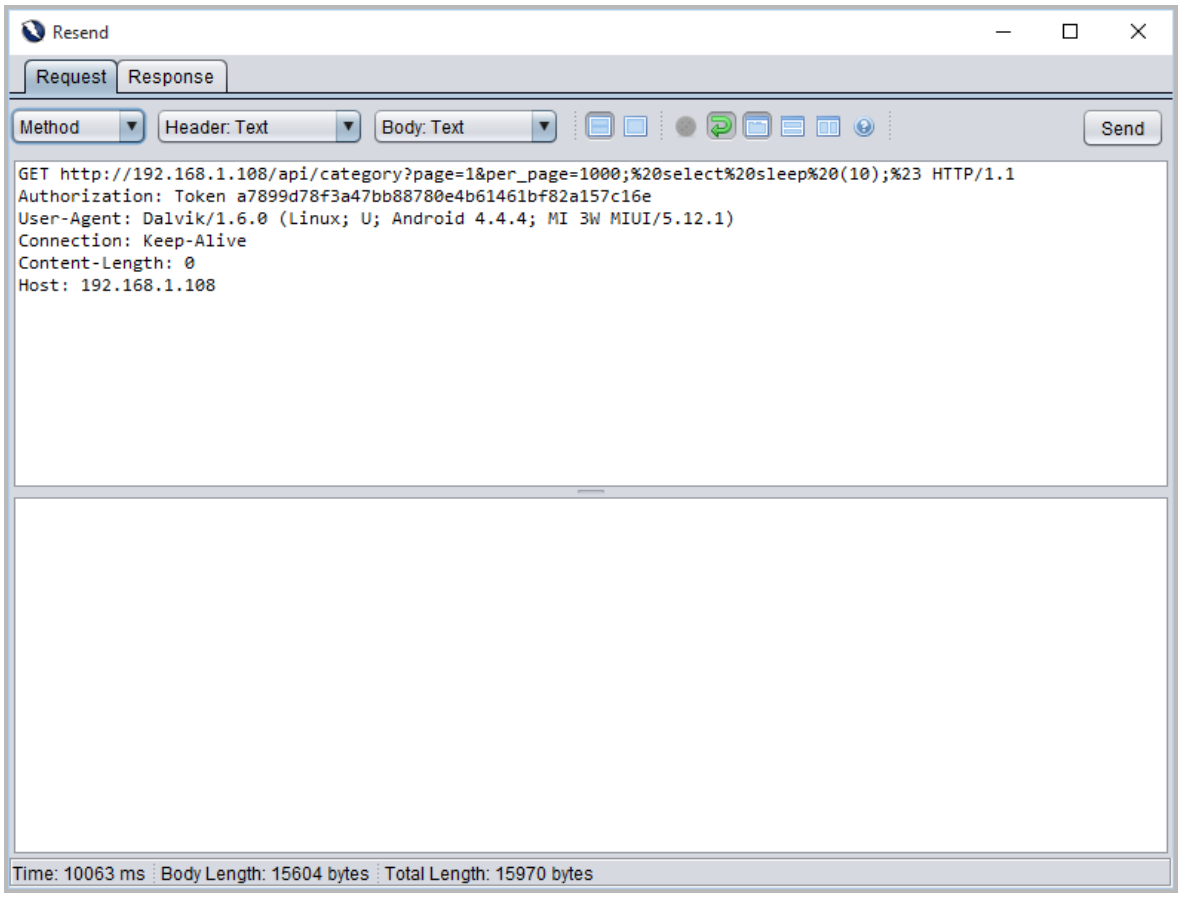
Response 1

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



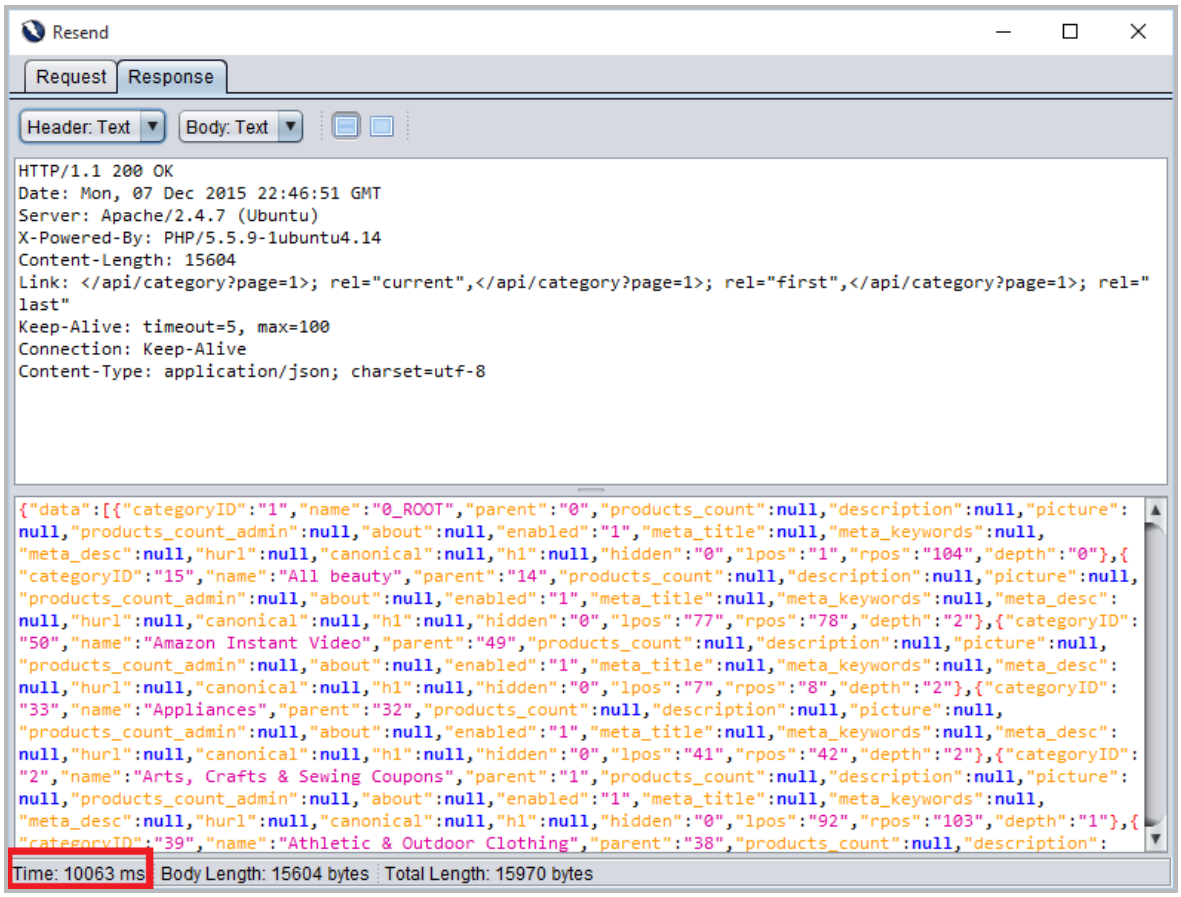
Request 2

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



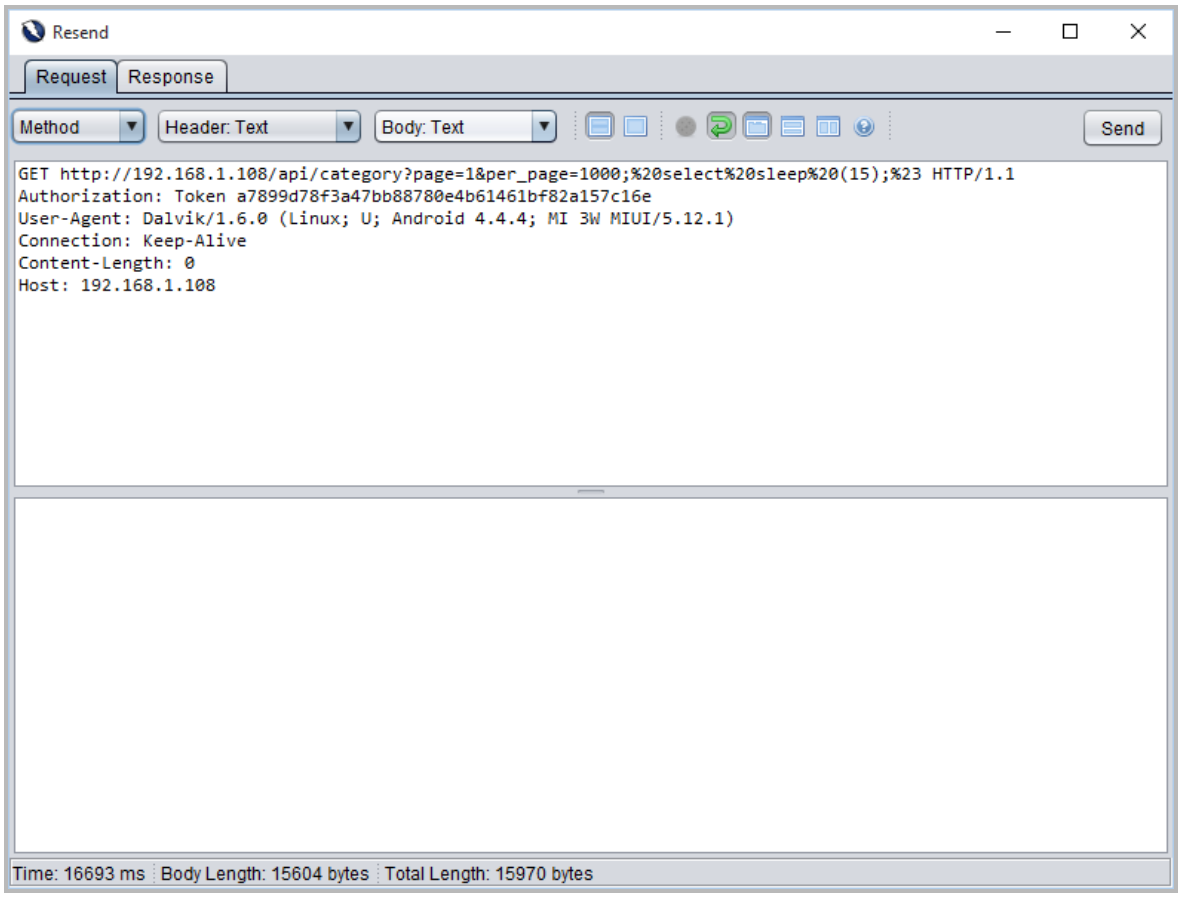
Response 2

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



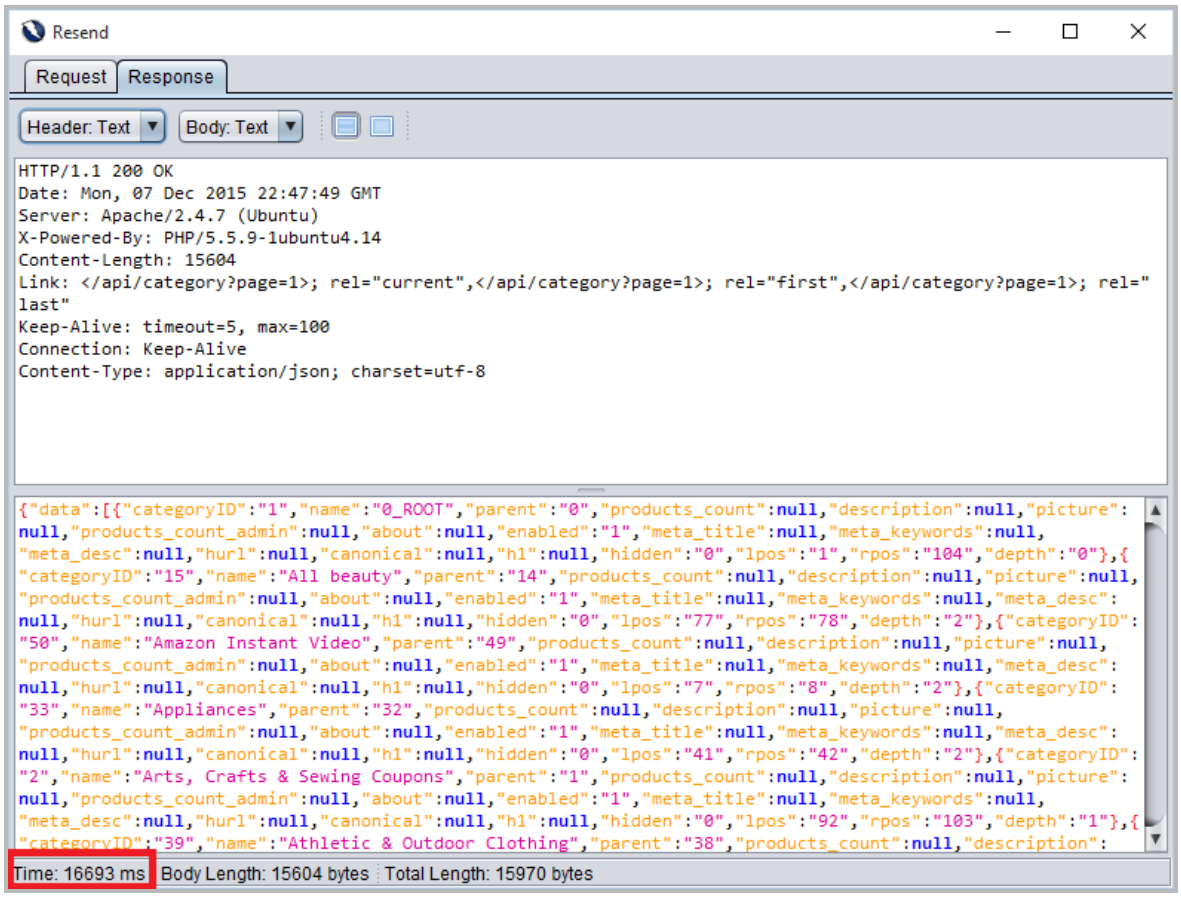
Request 3

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.



SQL injection

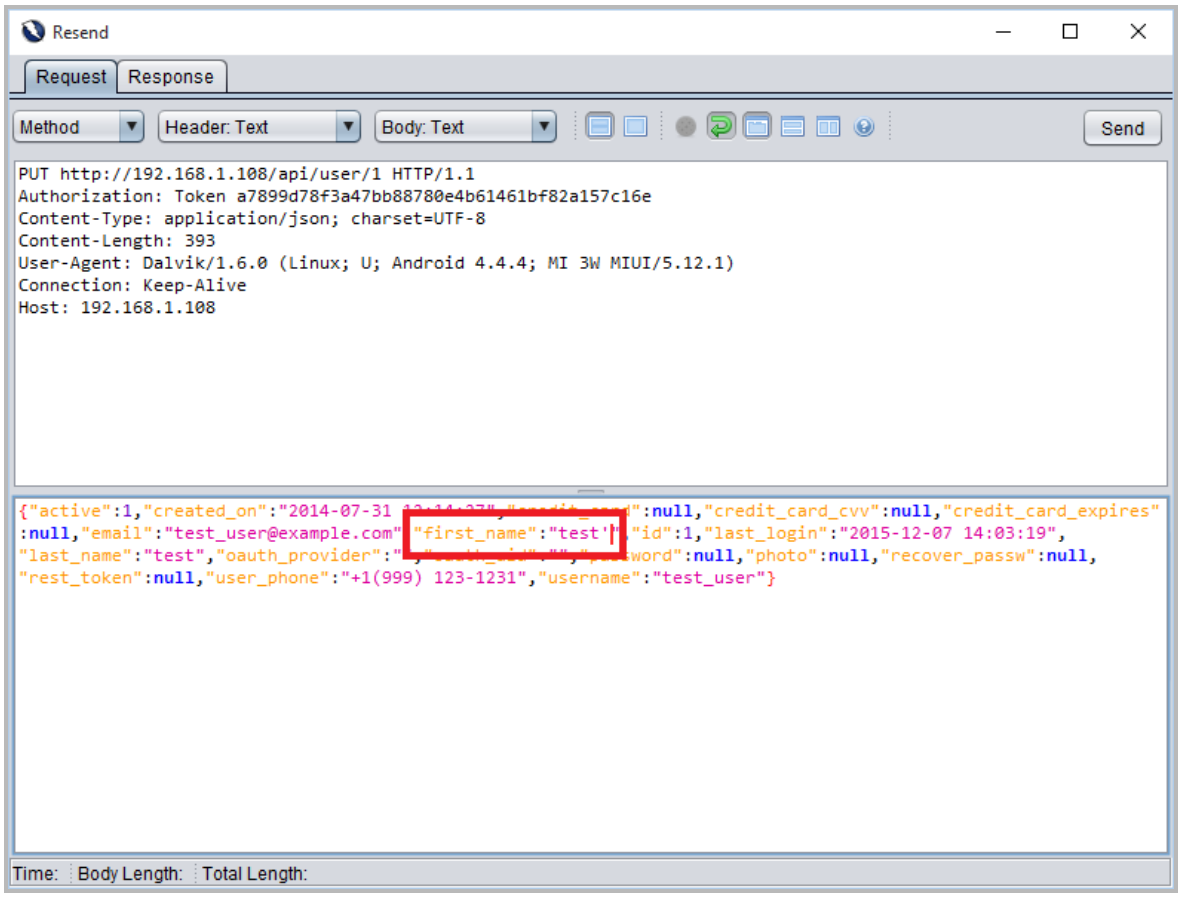
The REST API of the Hackazon application, which is vulnerable to SQL injection, is using the PUT method to update the user's profile. The request contains the parameter value, first_name. Based on this exercise, we can conclude that the first_name parameter is vulnerable to SQL Injection.

URL: http://192.168.202.131/api/user/1
Method: PUT
Parameter name: first_name
Attack values:

1. test'
2. test''

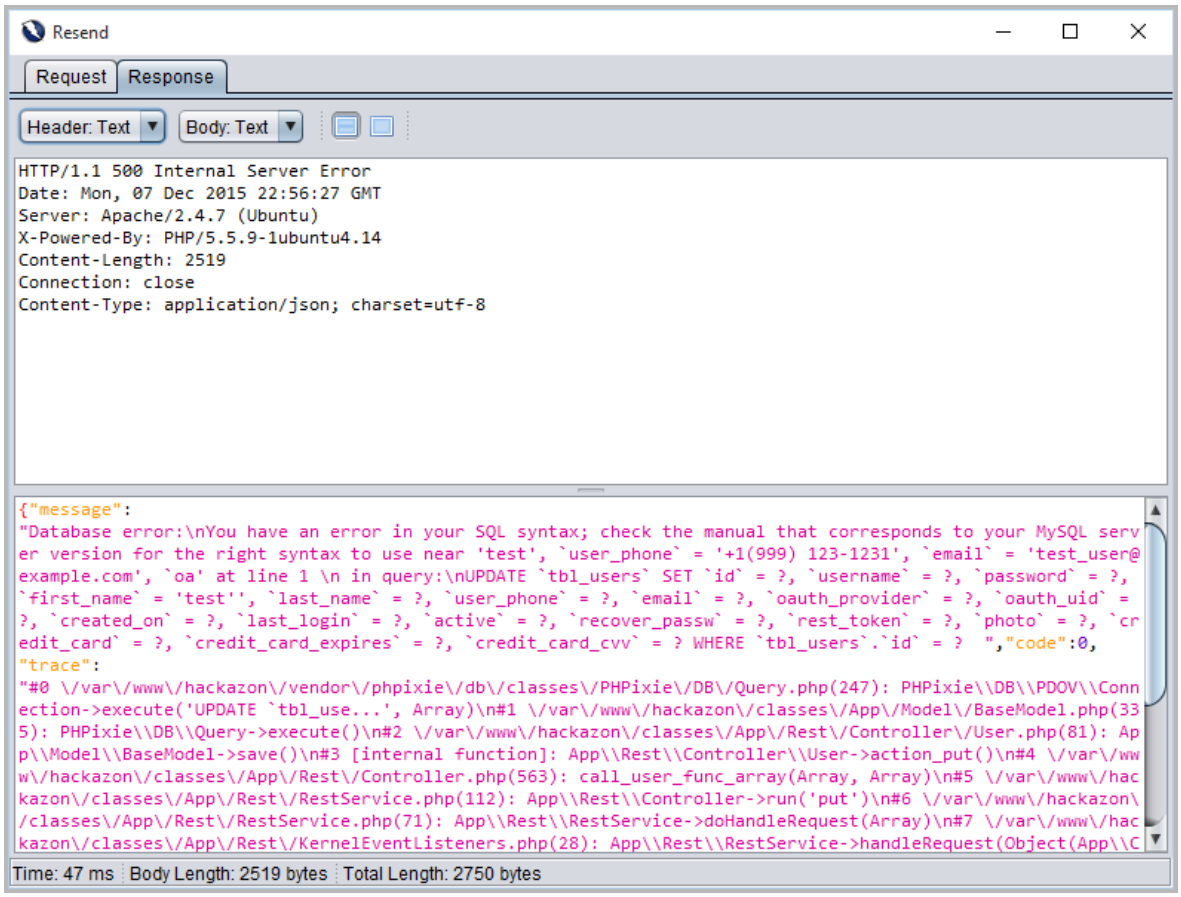
Request 1

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



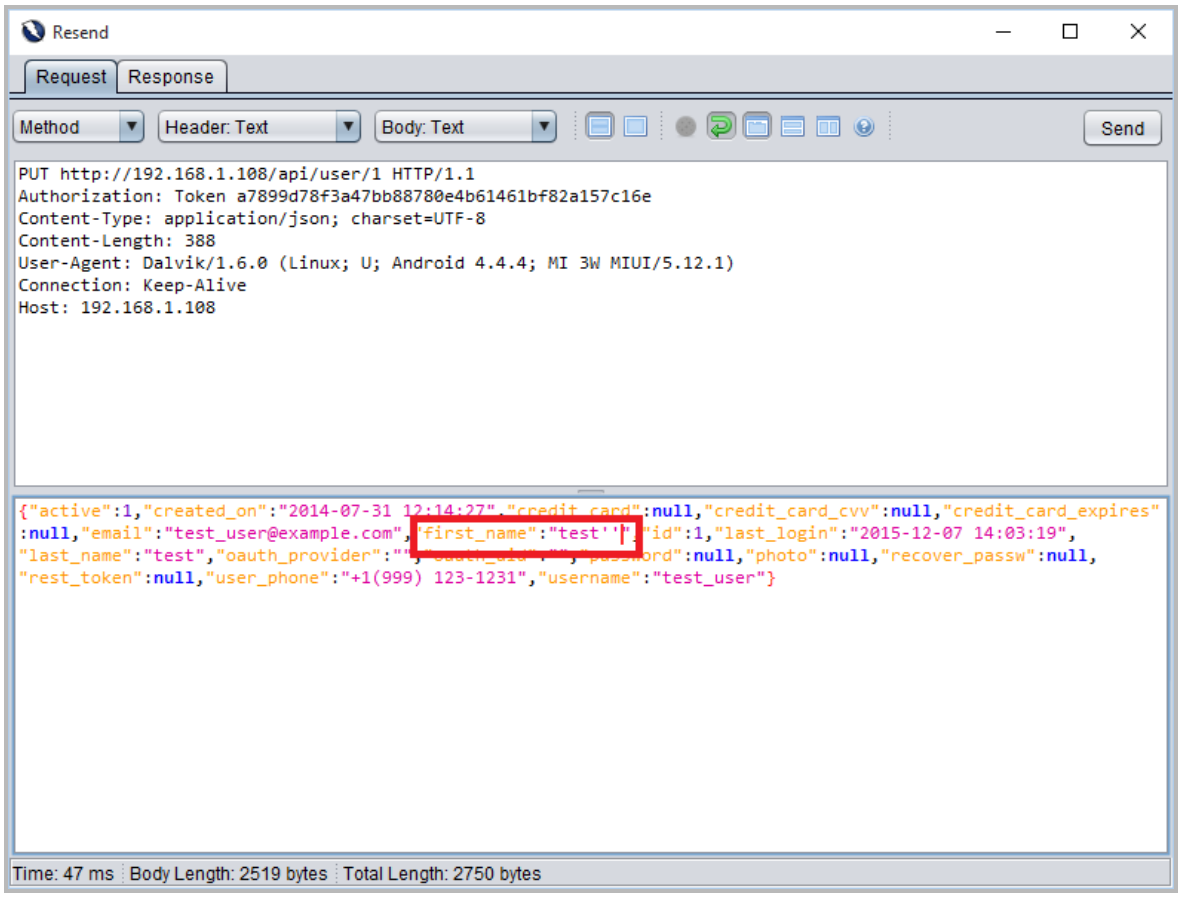
Response 1

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.



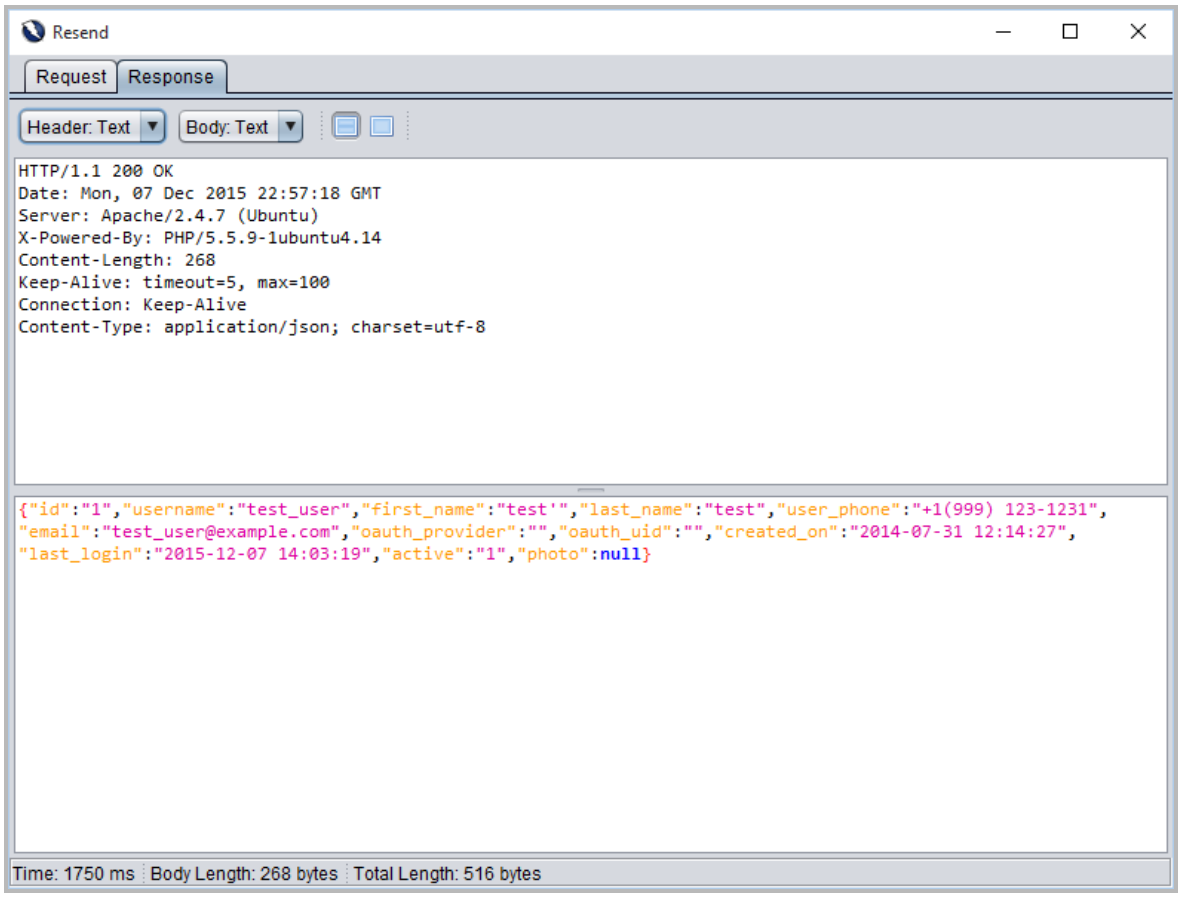
Request 2

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



Response 2

The user profile was submitted successfully.



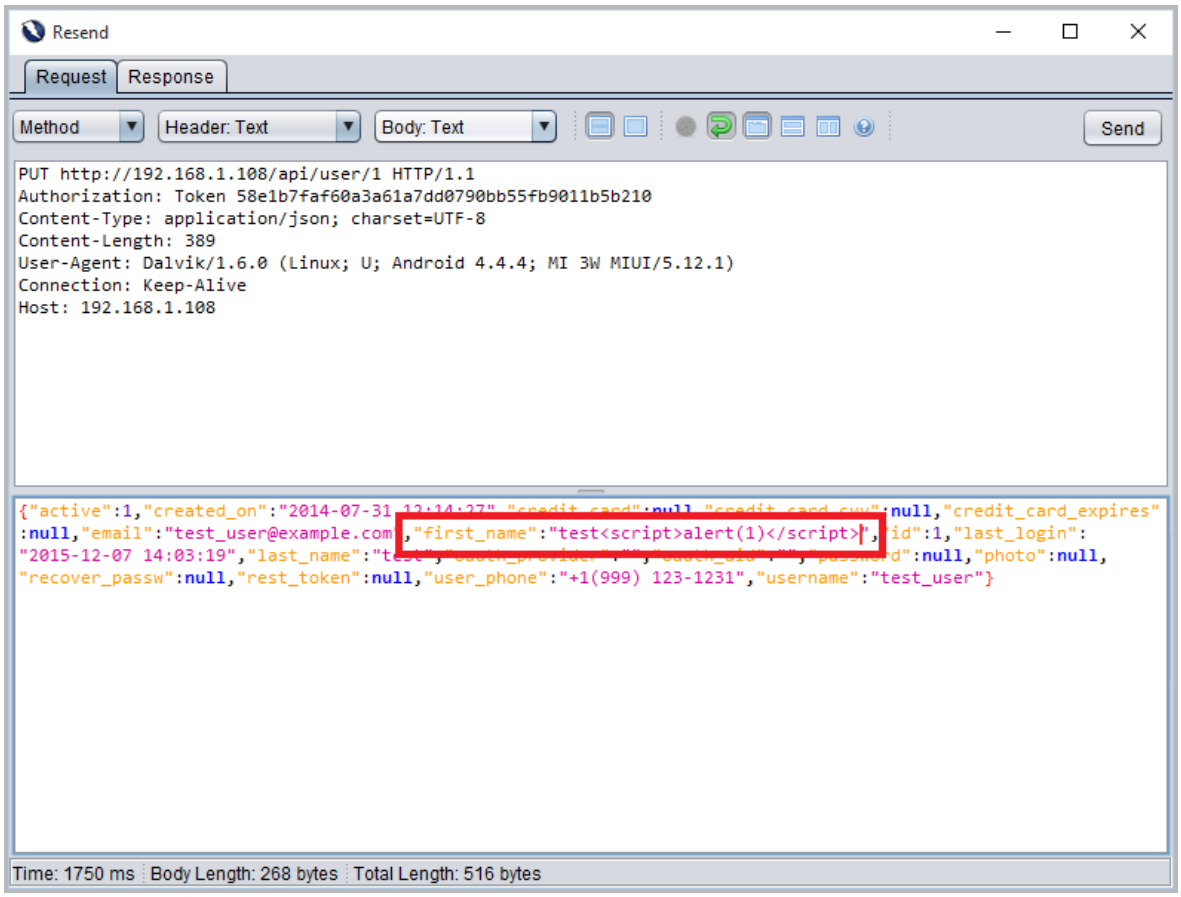
Cross-site scripting

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>

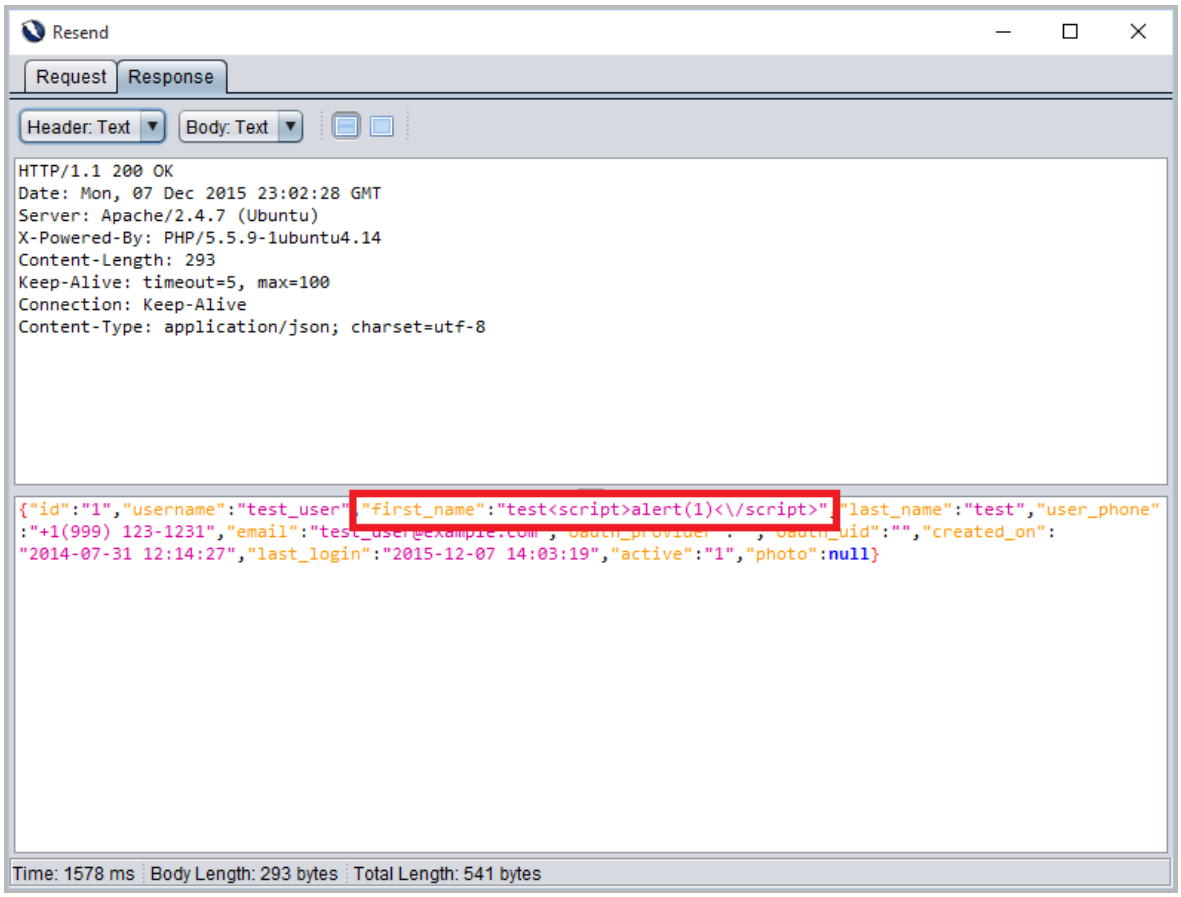
Request

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

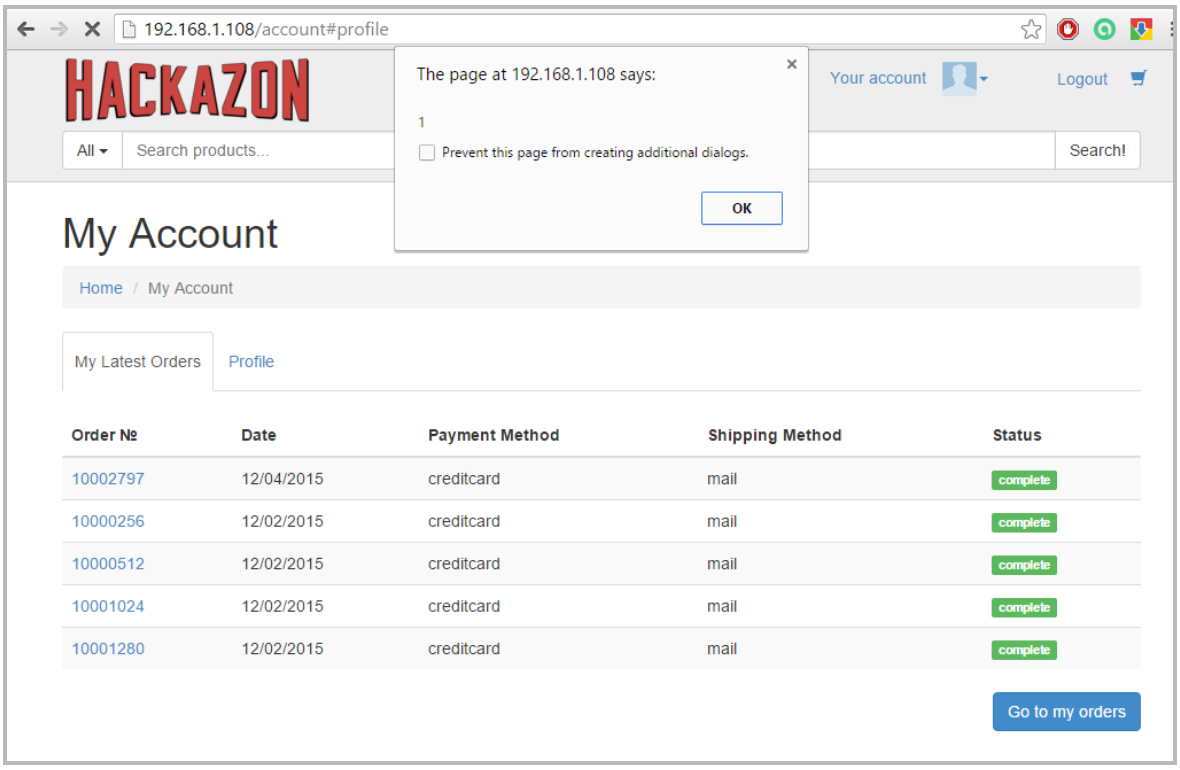


Response

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



The 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.

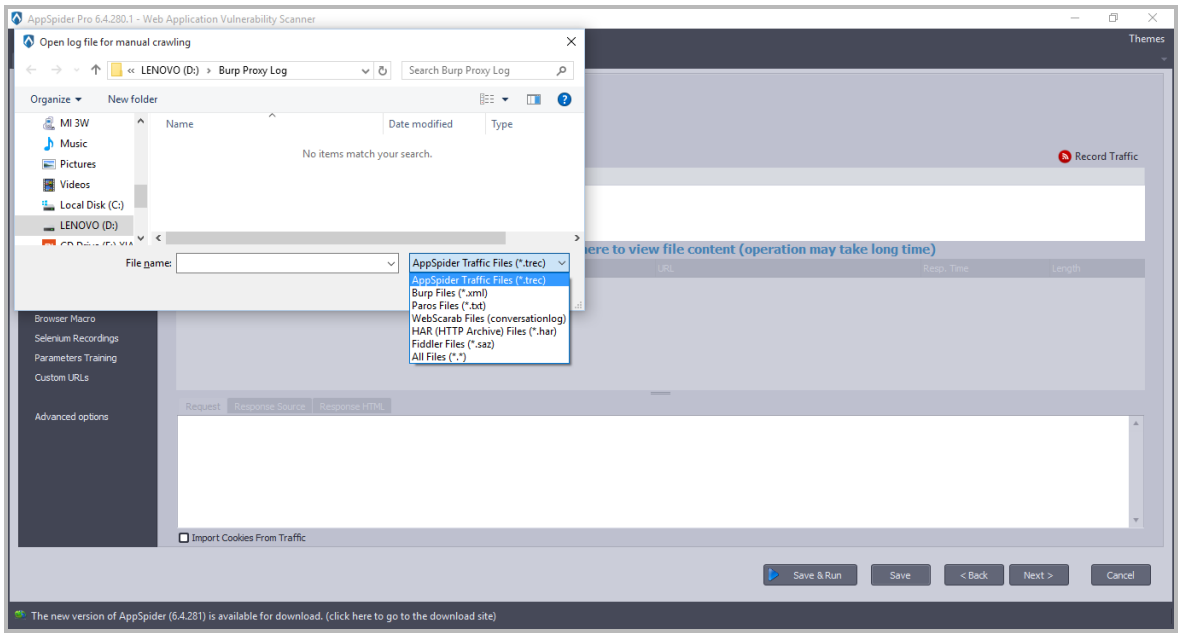


Testing a REST API using AppSpider

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.

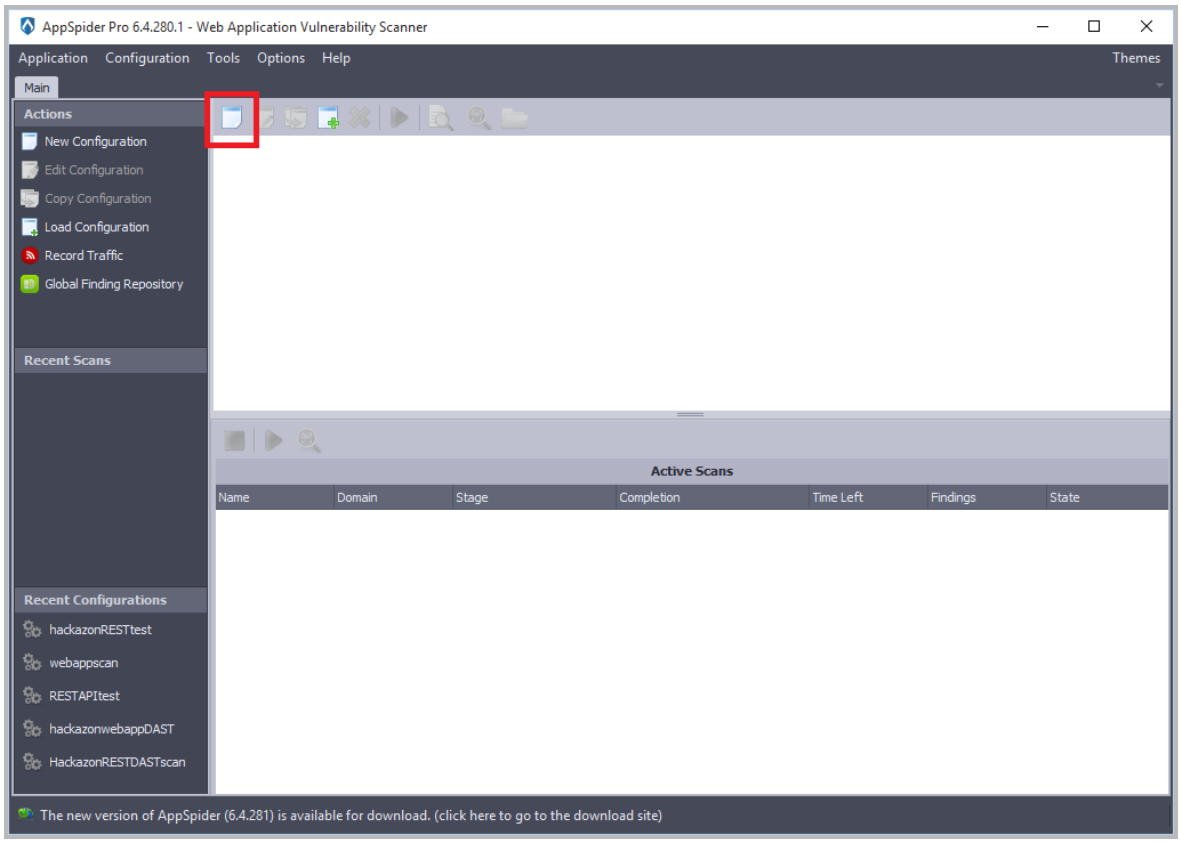
Record and import traffic

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

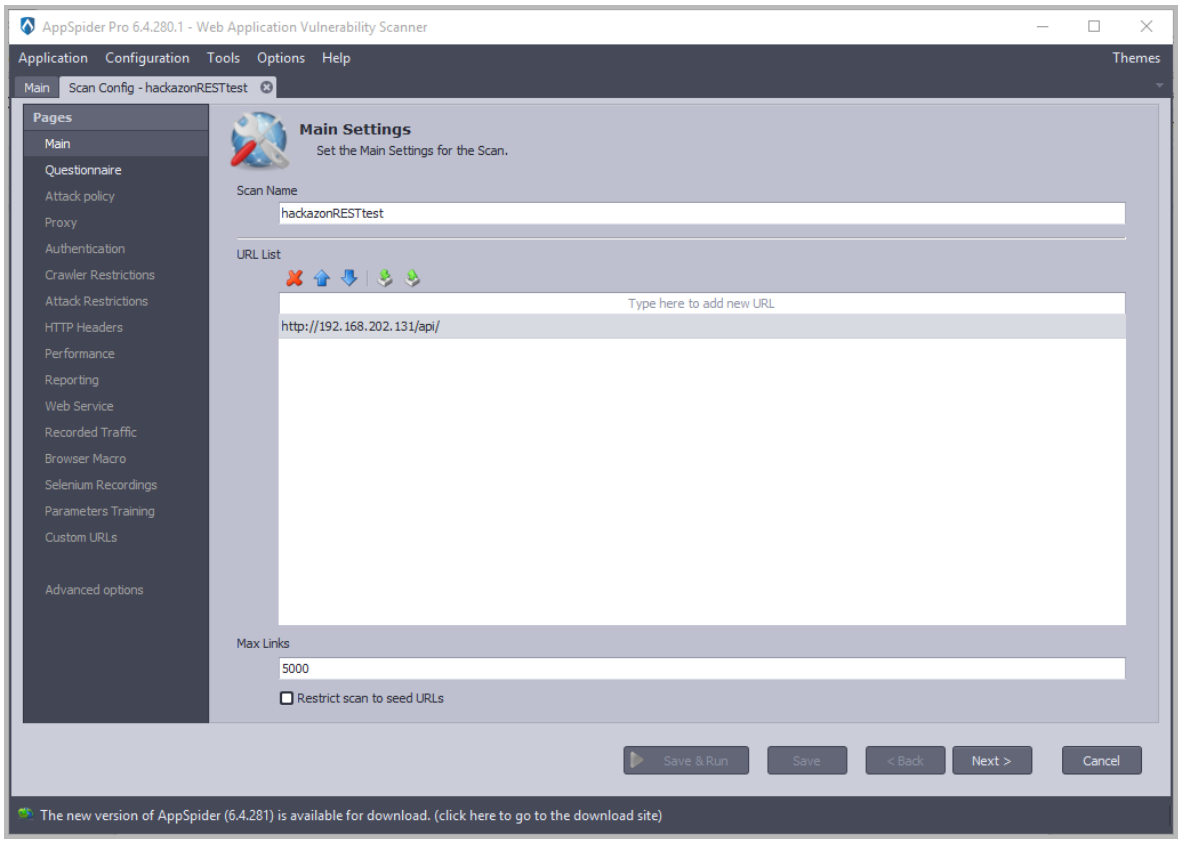


Configure and run a scan

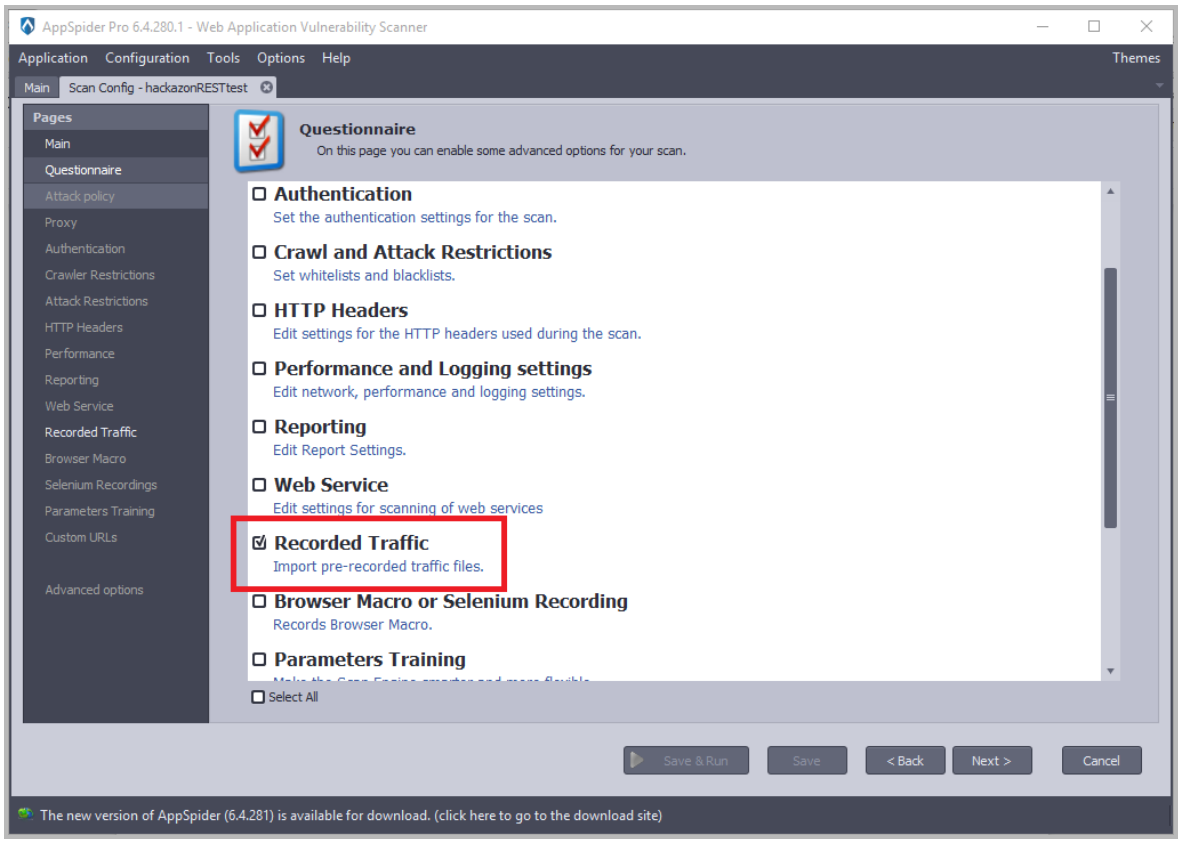
1. Open AppSpider.
2. Select **New Configuration** from the *Actions* panel in AppSpider.



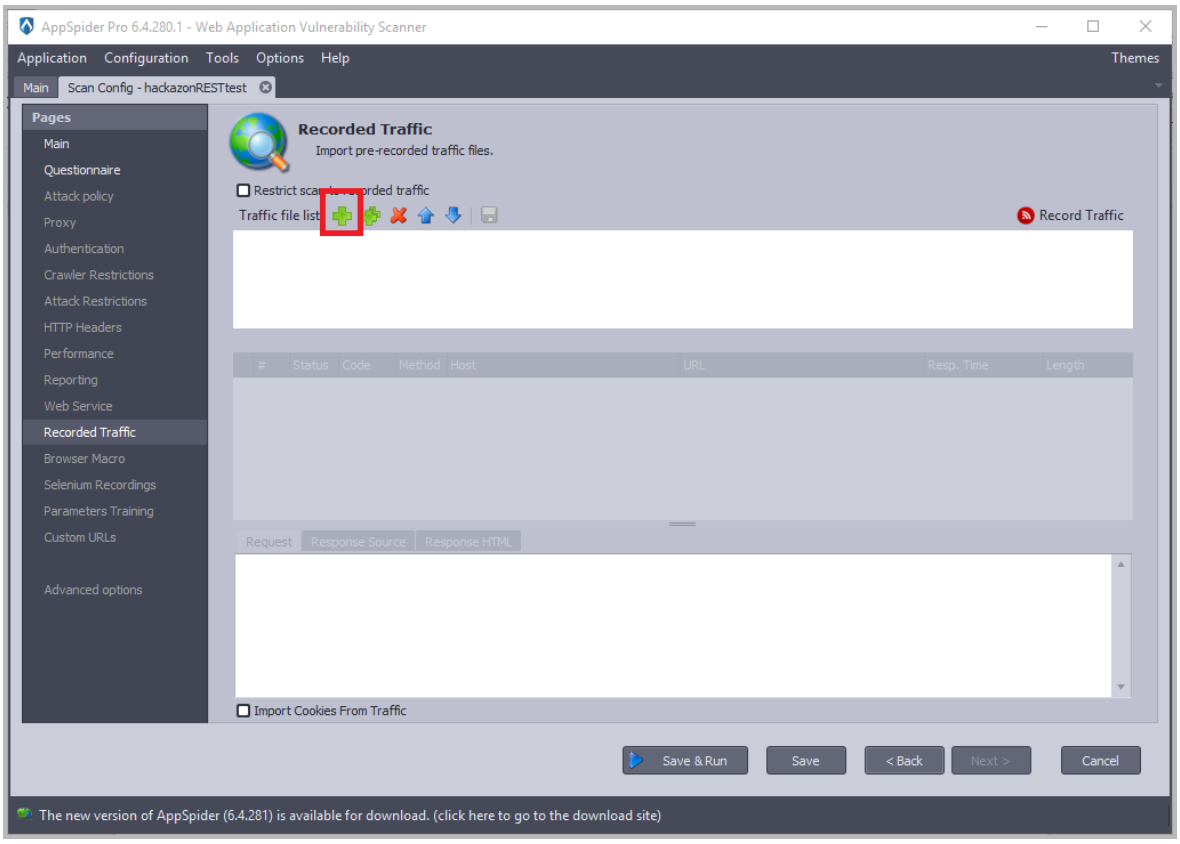
3. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.



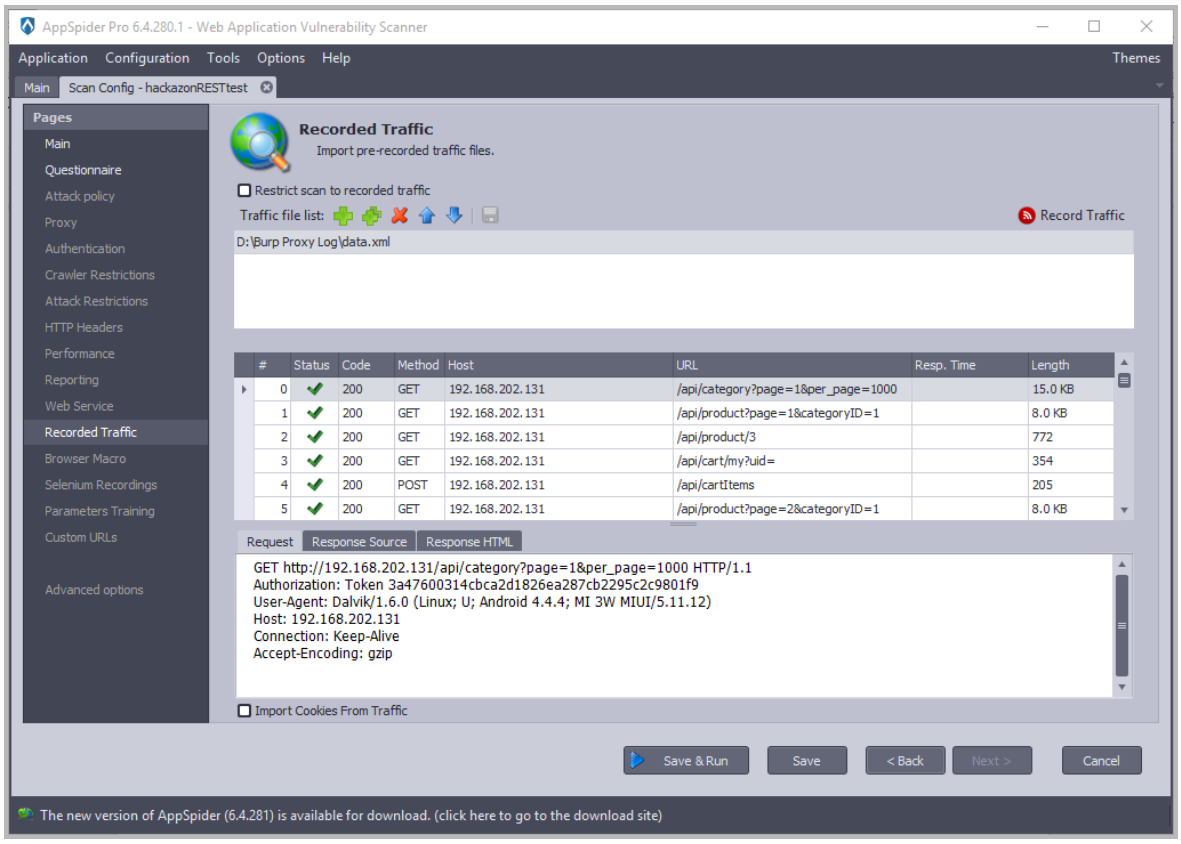
4. Select the check box for **Recorded Traffic** on the *Questionnaire* and click the **Next** button.



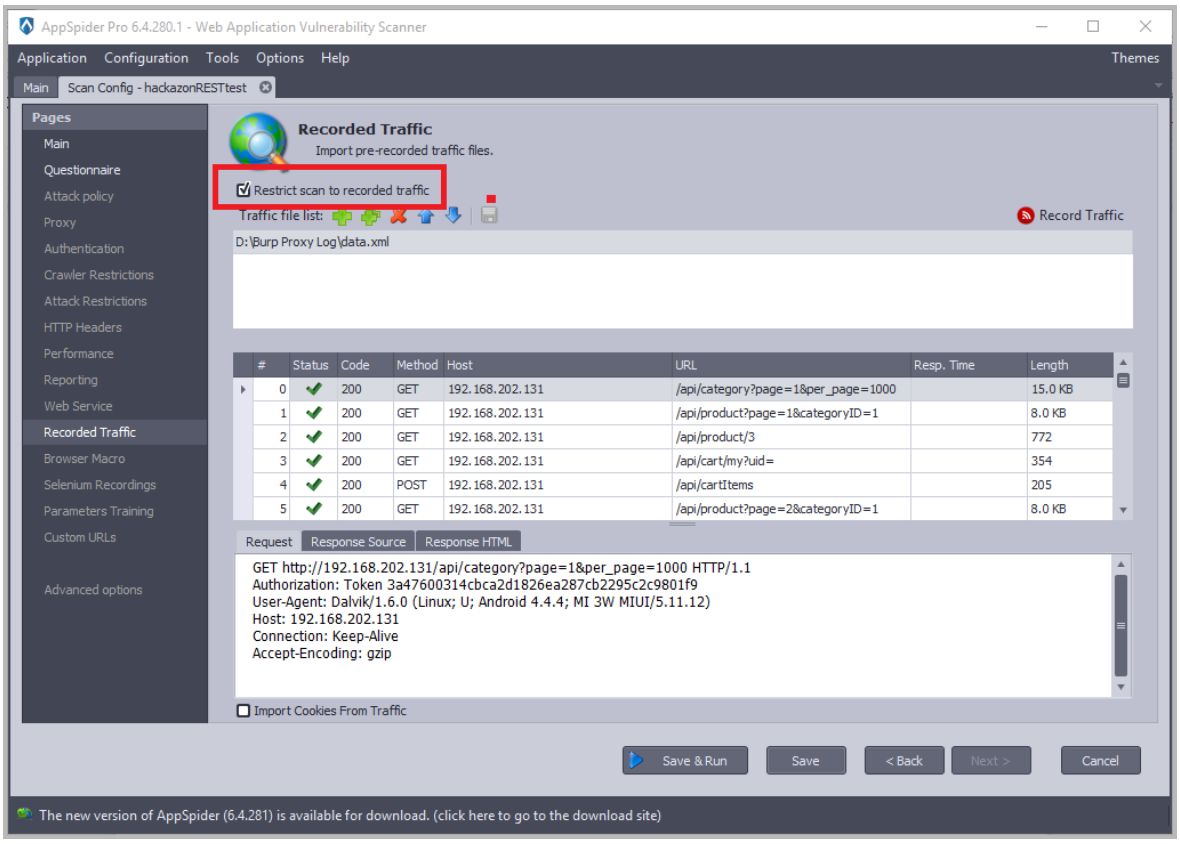
5. Click the **Add File (+)** icon and import the . xml file containing the recorded traffic.



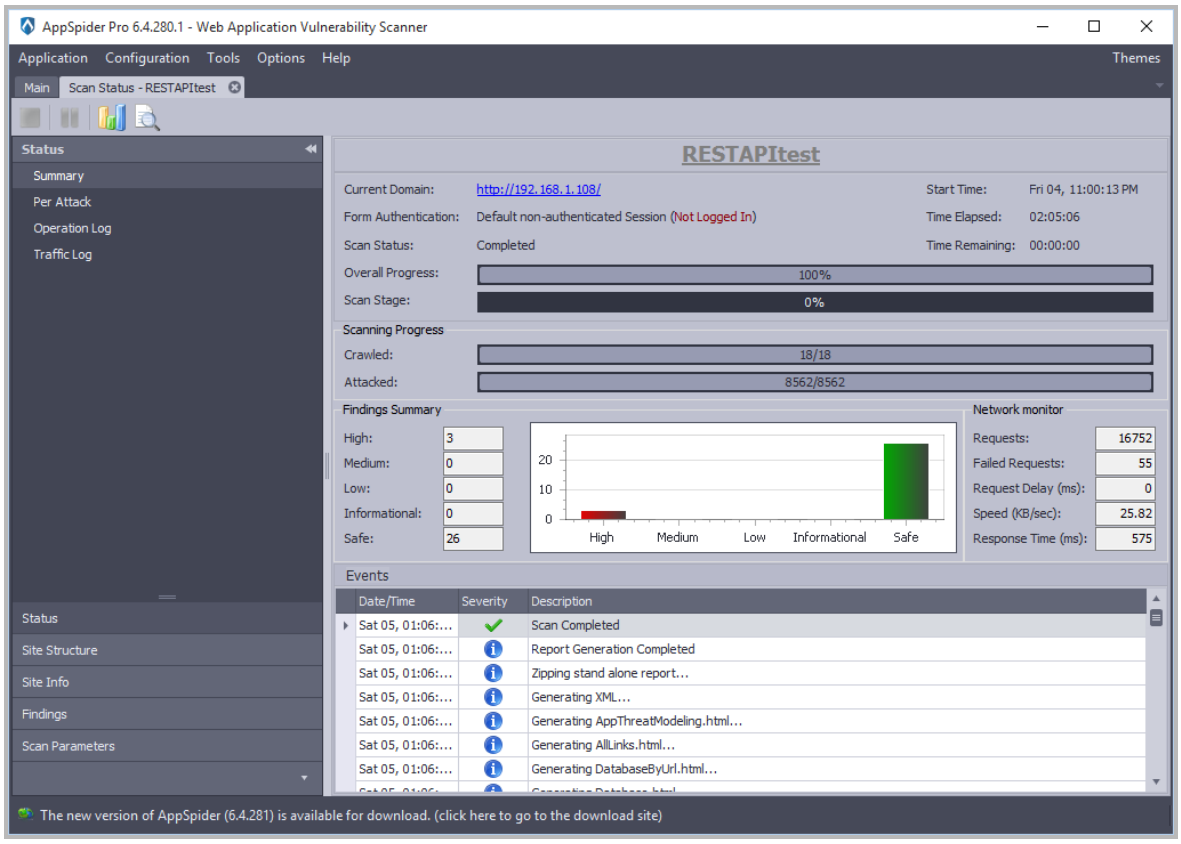
6. When prompted, click on **Click here to view file content** to view the recorded traffic.



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.

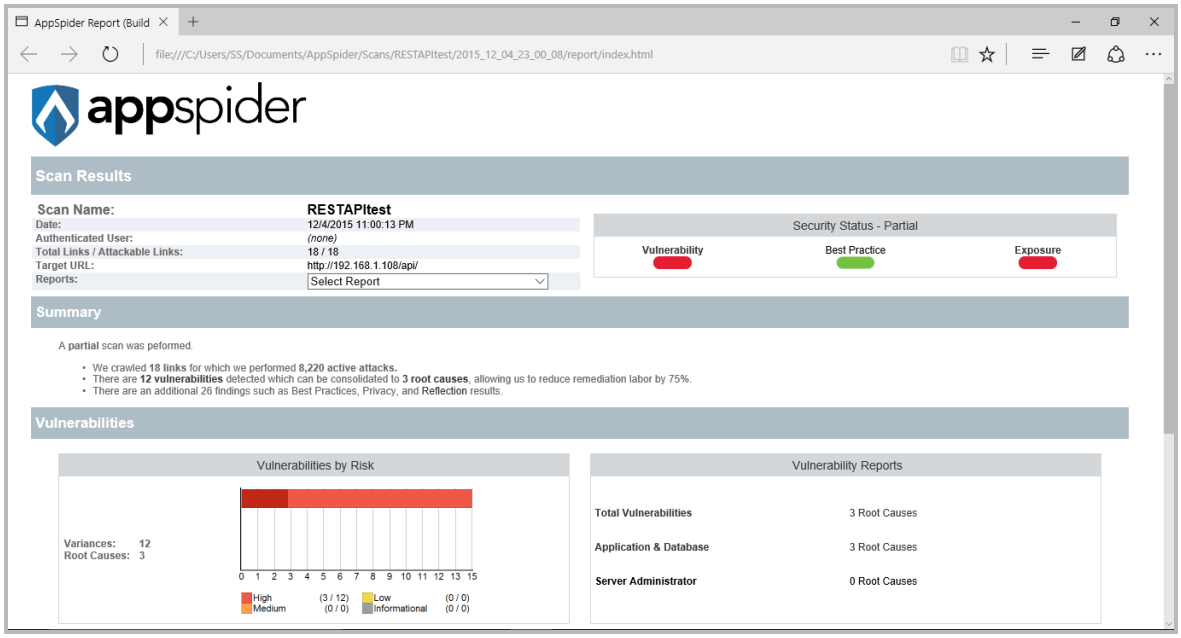


During the scan, AppSpider will provide live results in the *Scan Status* panel.



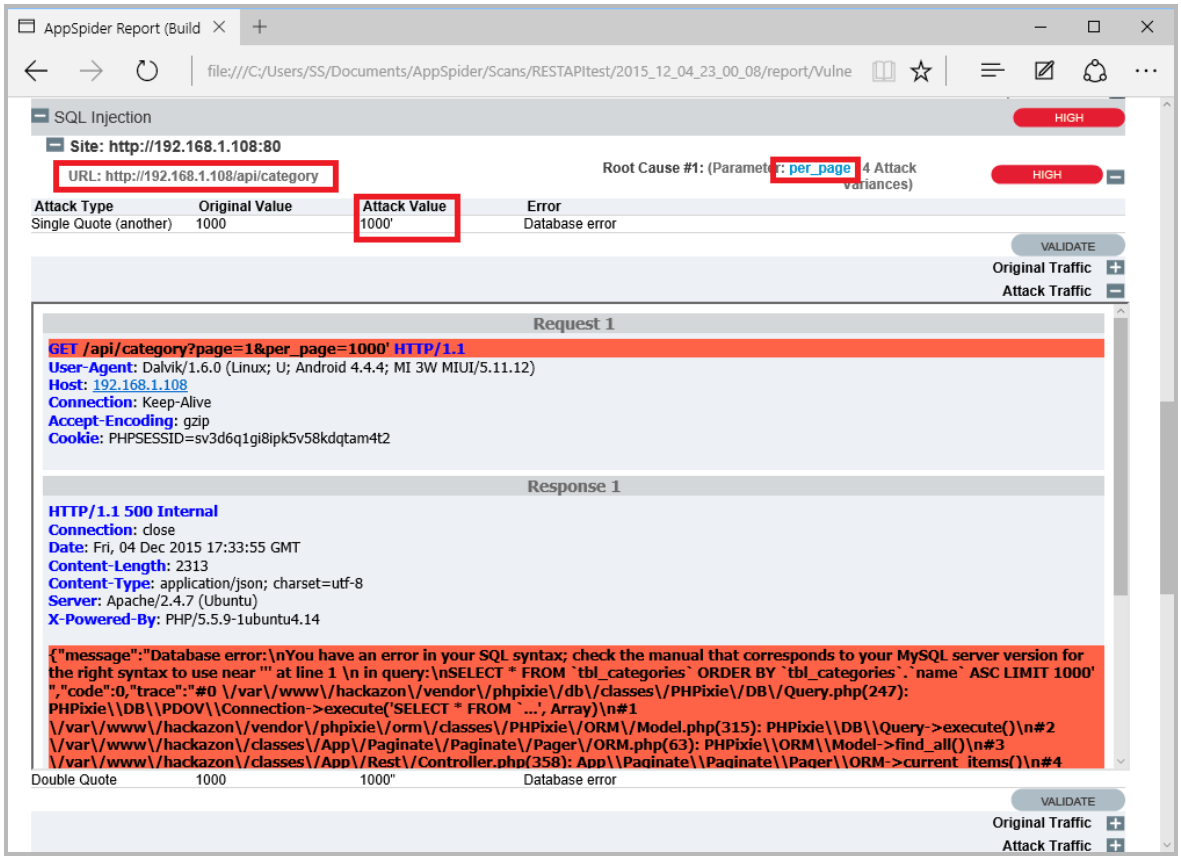
Reporting

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.



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.



How to create a custom attack module

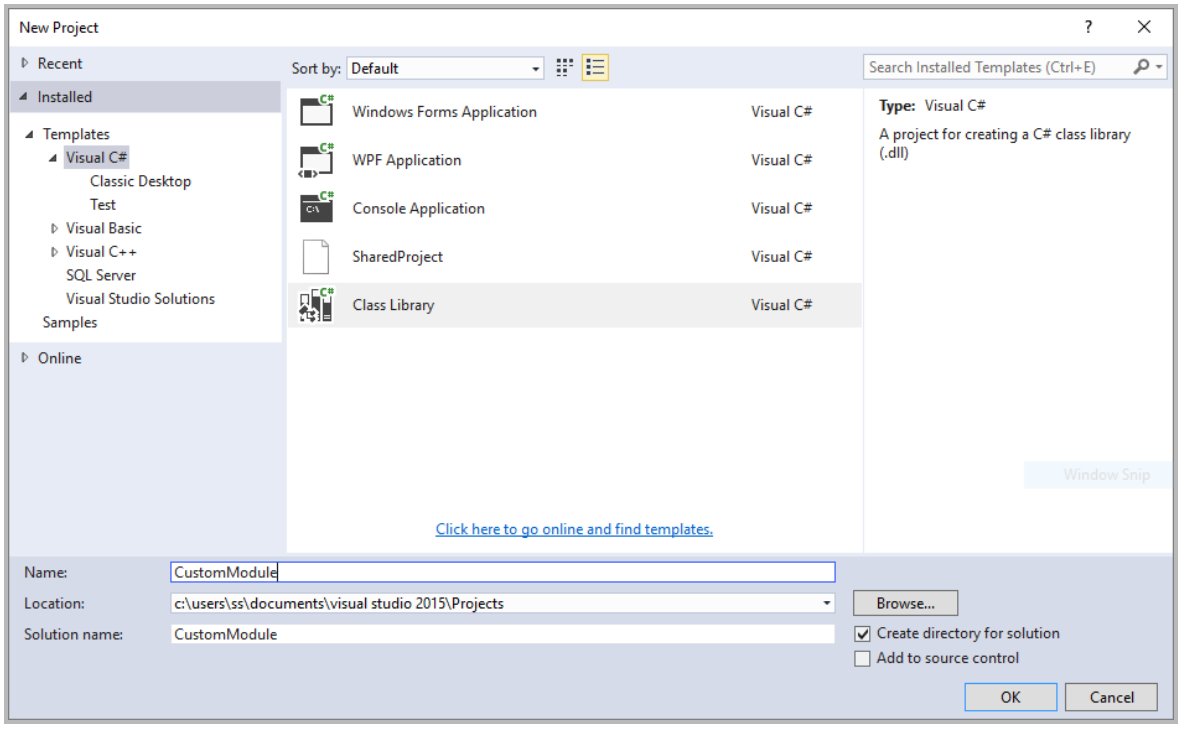
AppSpider has over 80 attack modules. In some cases, you may want to create a custom attack specific to your environment. AppSpider provides unique functionality which allows you to implement custom attack modules based on your application environment.

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

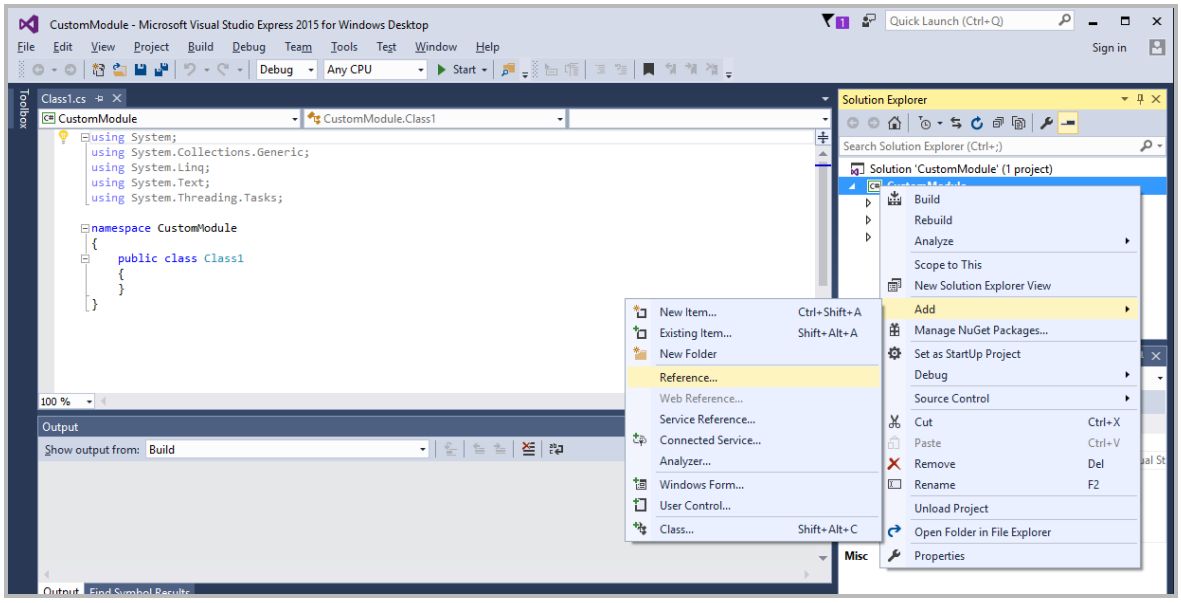
1. Open VS Express and select **New Project**.
2. In the left pane of the *New Project* dialog box, navigate to **Installed ->Templates -> Visual C# -> Class Library.**
3. Enter the **Name** and **Location** of your project, and click the **OK** button to save your class library.



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:

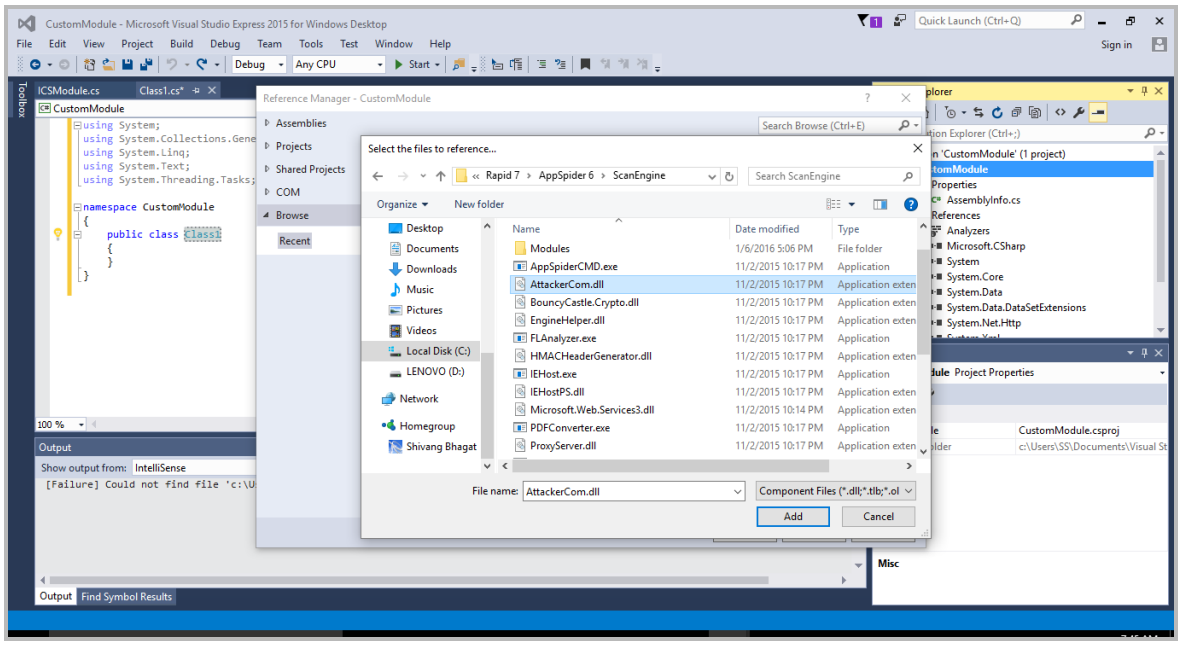
1. Right-click on your project in the *Solution Explorer* and navigate to **Add -> References**.



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.

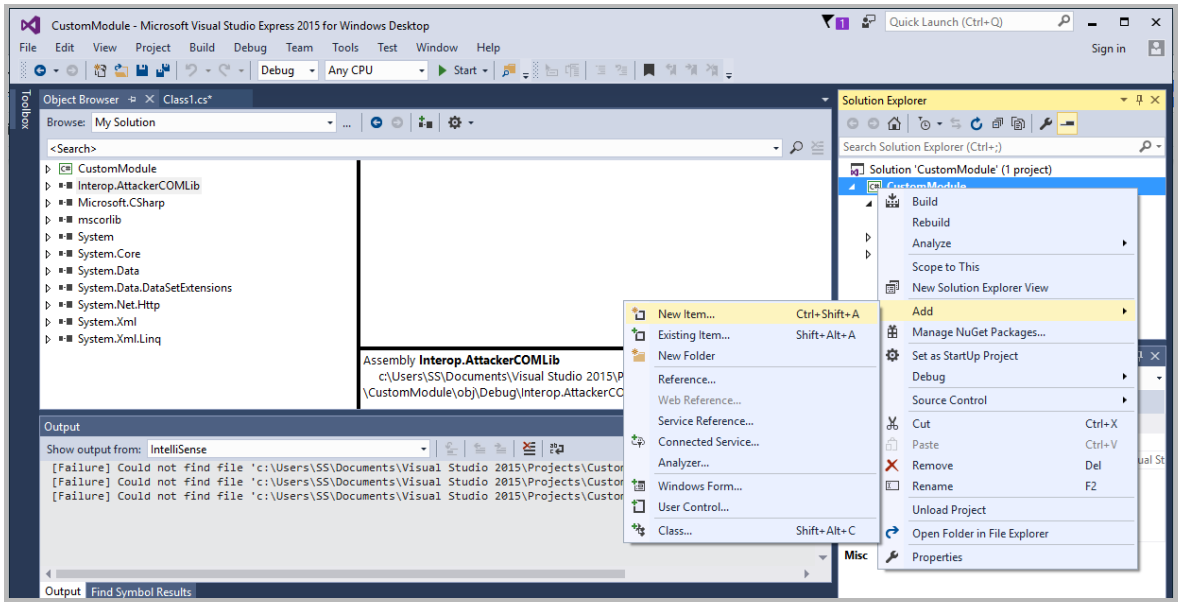
4. Click the **Add** button to continue.



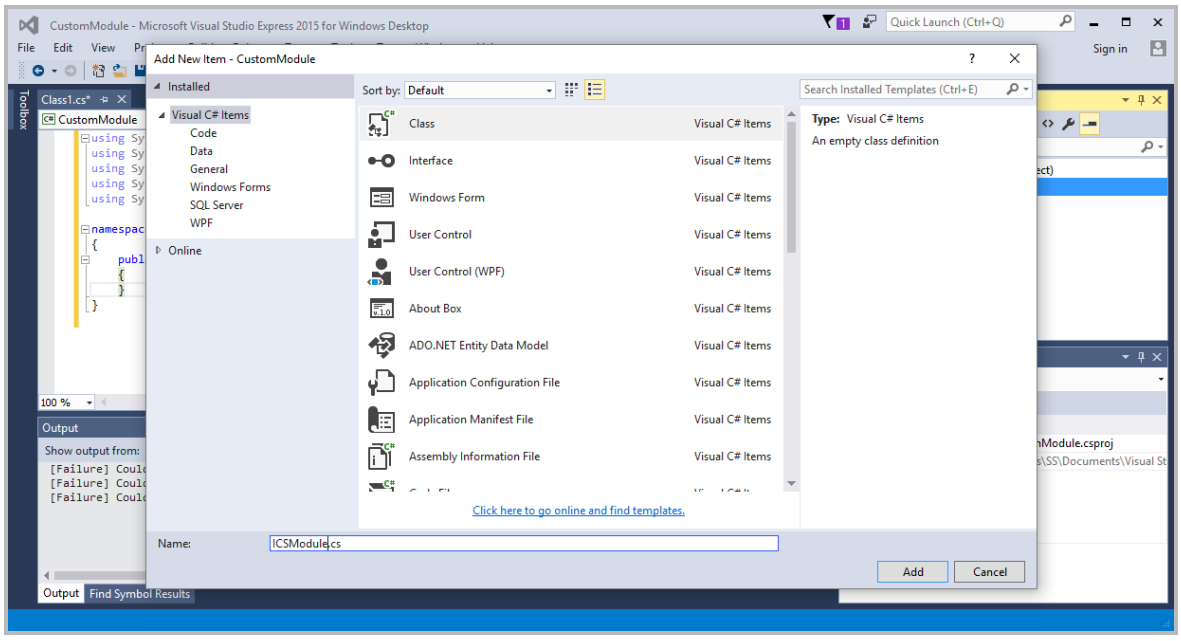
Creating classes

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**.

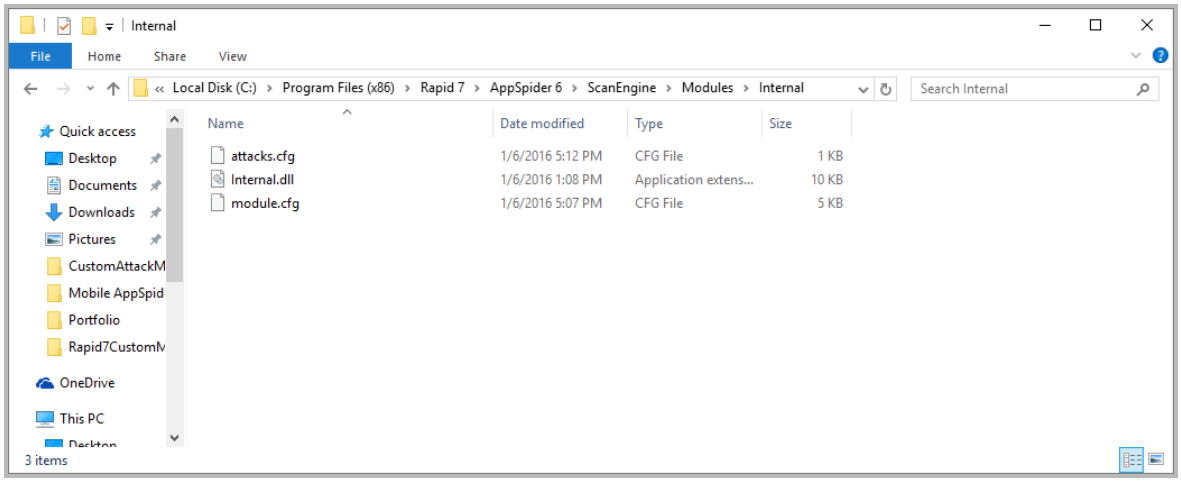
Note: The path to the *Modules* folder location is C:Program Files (x86)\Rapid7 \AppSpider6\ScanEngine\Modules.

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:

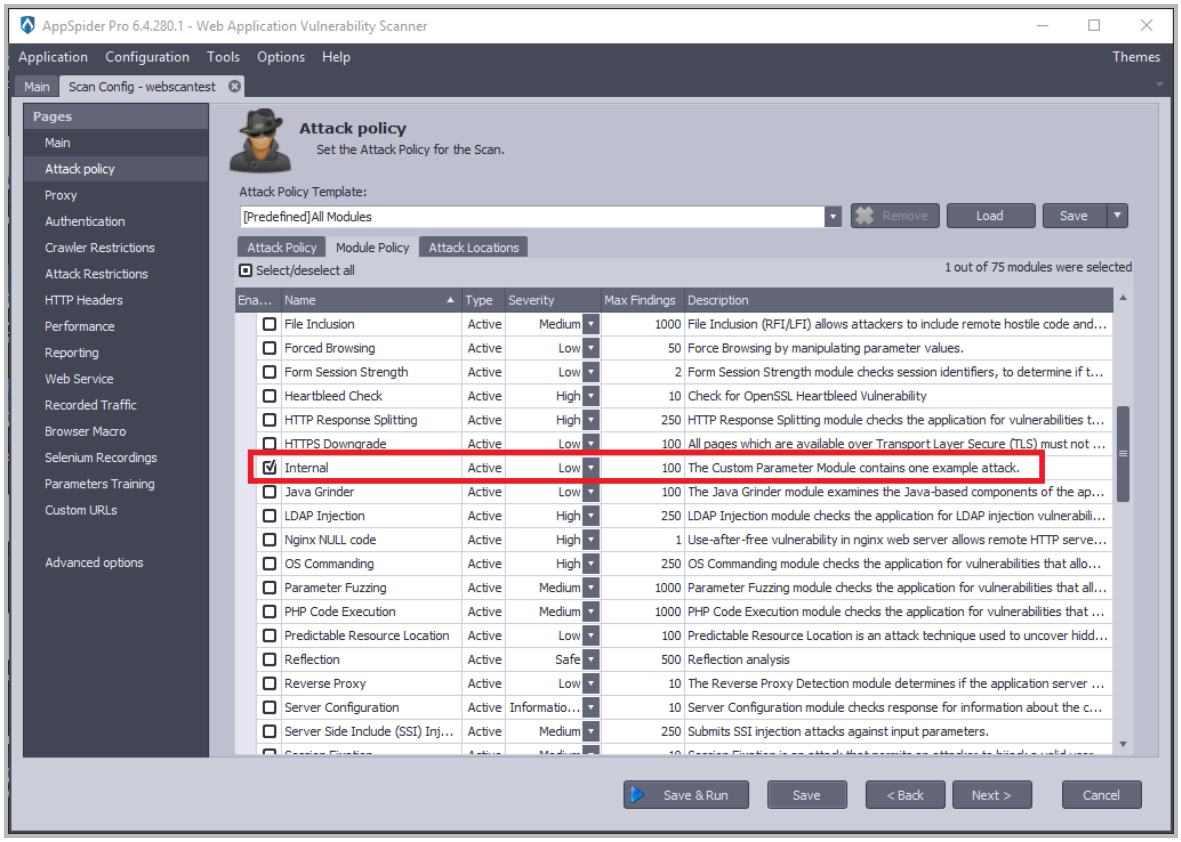
```
<AttackModulePolicy>
<Enabled>1</Enabled>
<ModuleId>7DEE1967063D4BE08061028D3E707FCE</ModuleId>
<ModulePriority>Medium</ModulePriority>
<Severity>Low</Severity>
<MaxVulnLimit>100</MaxVulnLimit>
<MaxVarianceLimit>1</MaxVarianceLimit>
<PassiveAnalysisOnAttacks>0</PassiveAnalysisOnAttacks>
<EnforceEncoding>0</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.



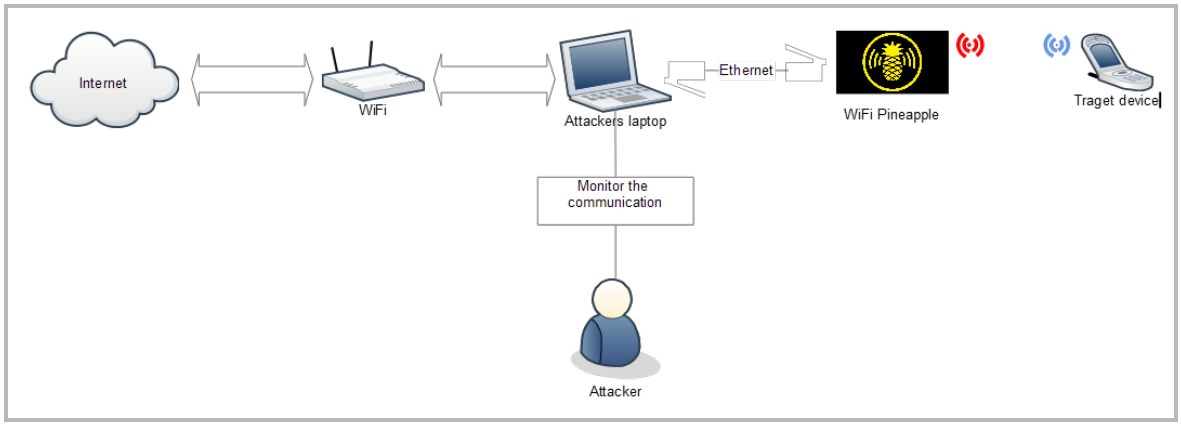
How to conduct mobile application testing using the WiFi Pineapple

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.

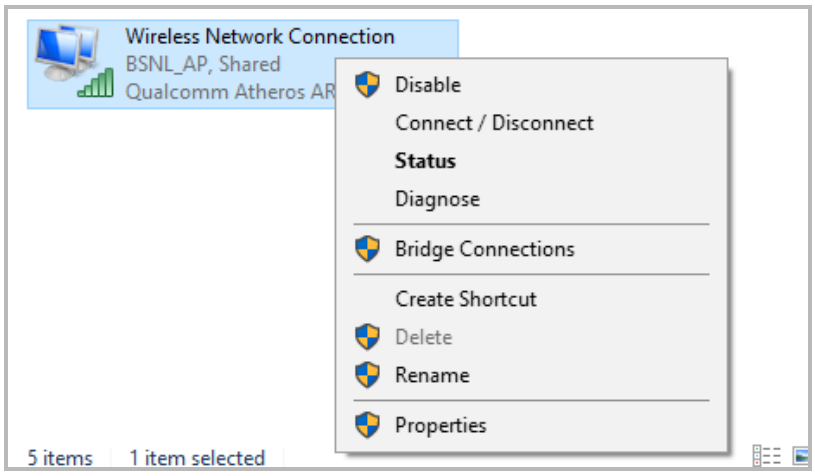
Once the target device connects to this rogue access point, you will able to monitor the traffic of the target device.



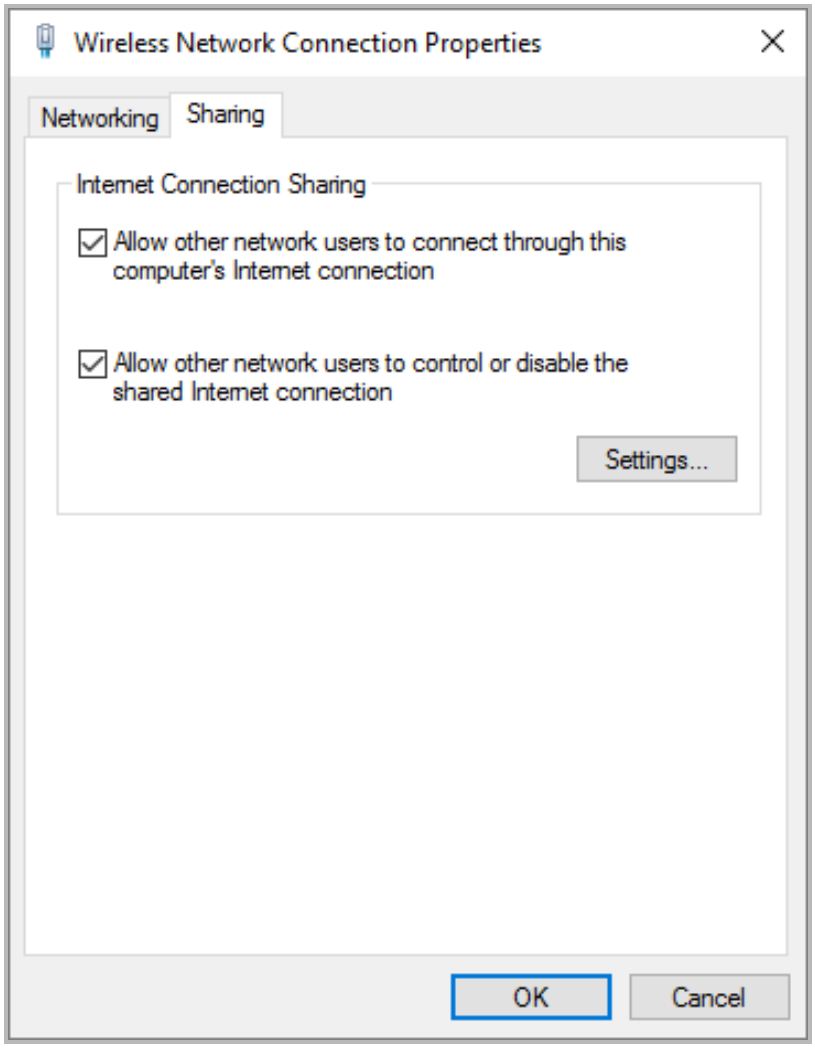
WiFi Pineapple setup with your machine

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.

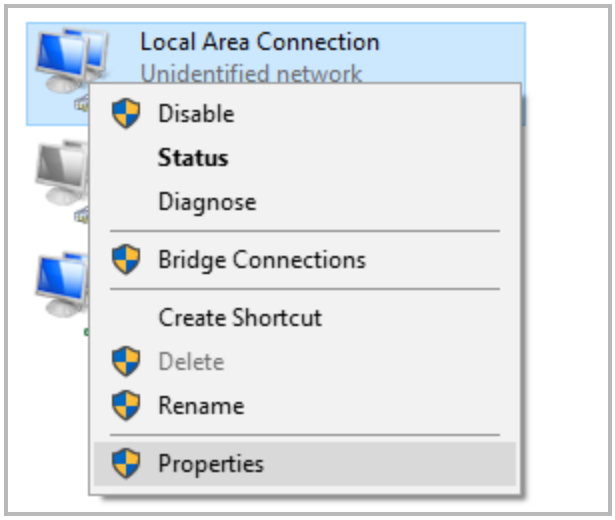
1. Open Network Connections. Right-click the Internet facing adapter and select **Properties**.



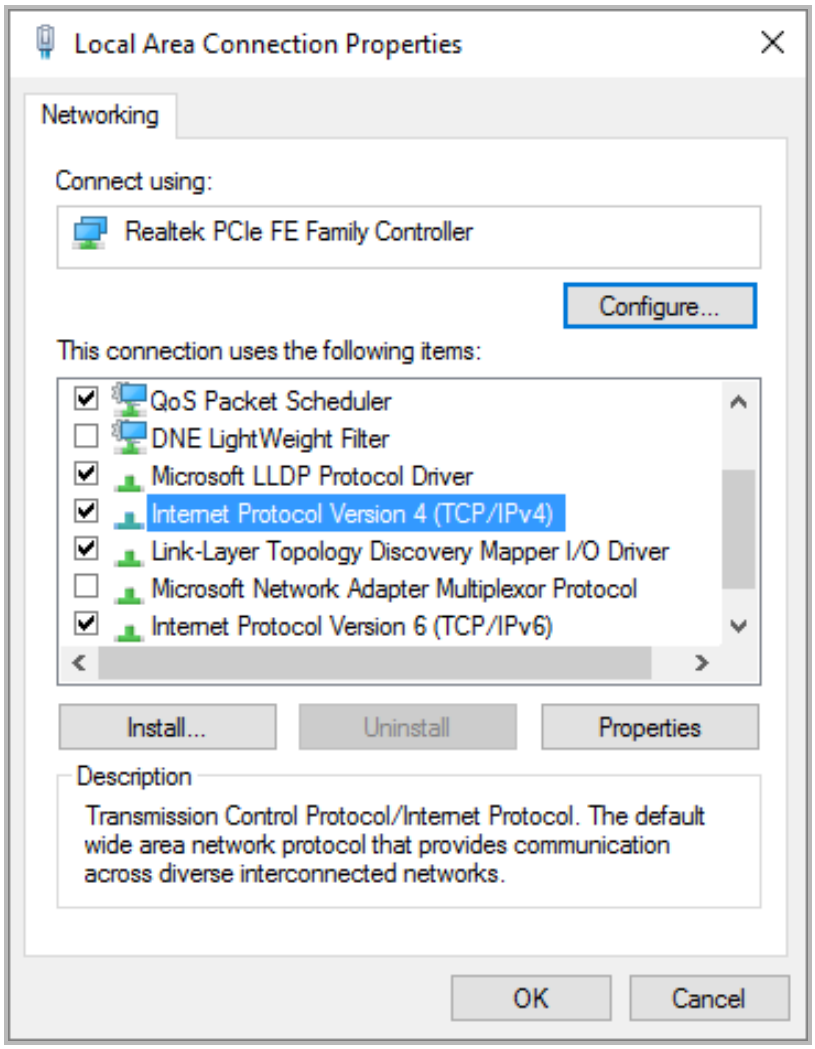
2. From the *Sharing* tab in the *Wireless Network Connection Properties*, select the **Allow other network users to connect through this computer's Internet connection** check box and click **OK**.



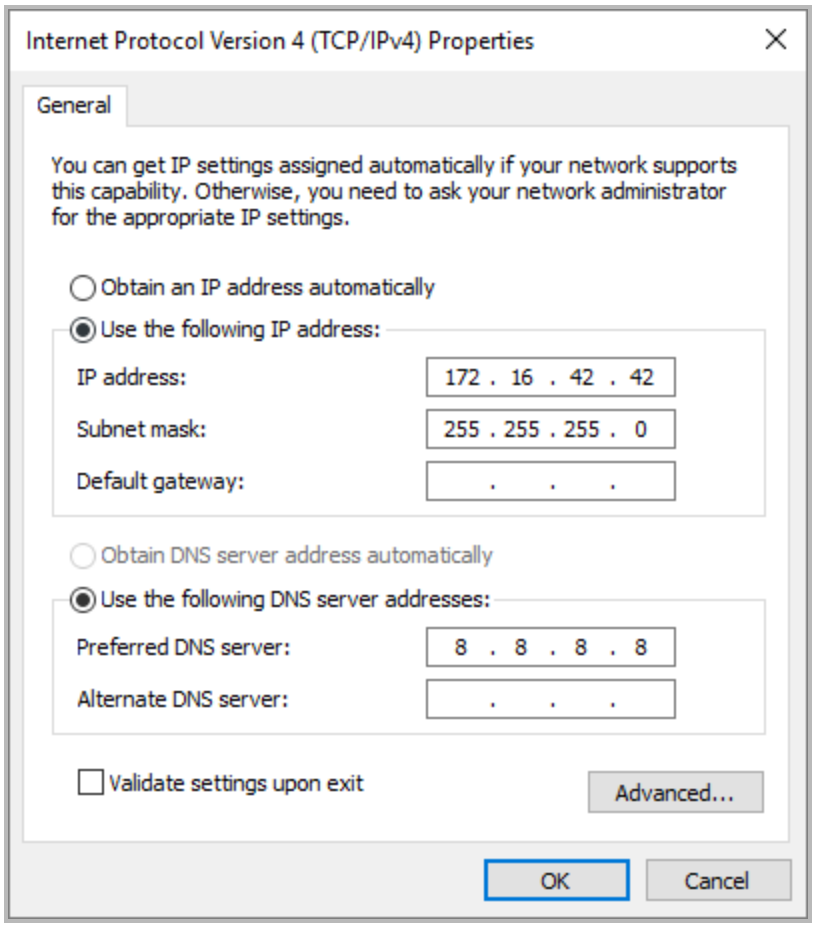
3. Select **Properties** of the WiFi Pineapple-facing adapter.



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

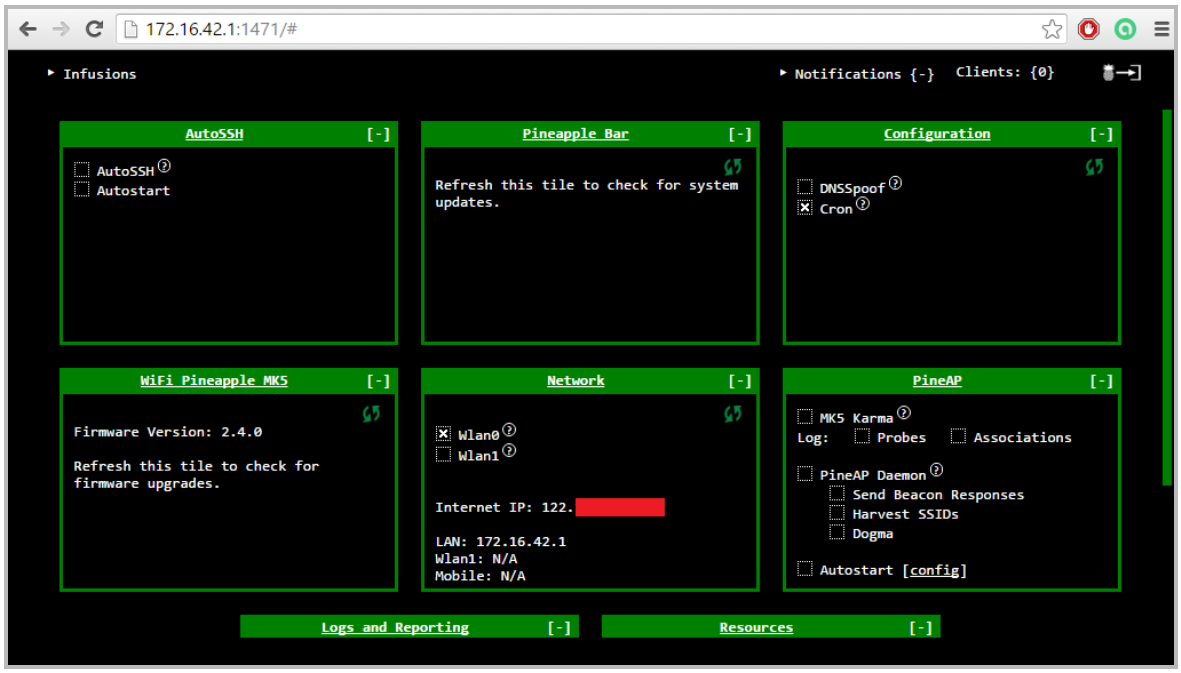


5. Select **Use the following IP address**
6. Specify **172.16.42.42** for the *IP address* and **255.255.255.0** for the *Subnet mask*. Leave the *Default gateway* empty.
7. Select **Use the following DNS server addresses**, enter your *preferred DNS server* (e.g. Google's 8.8.8.8), and click **OK**.



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

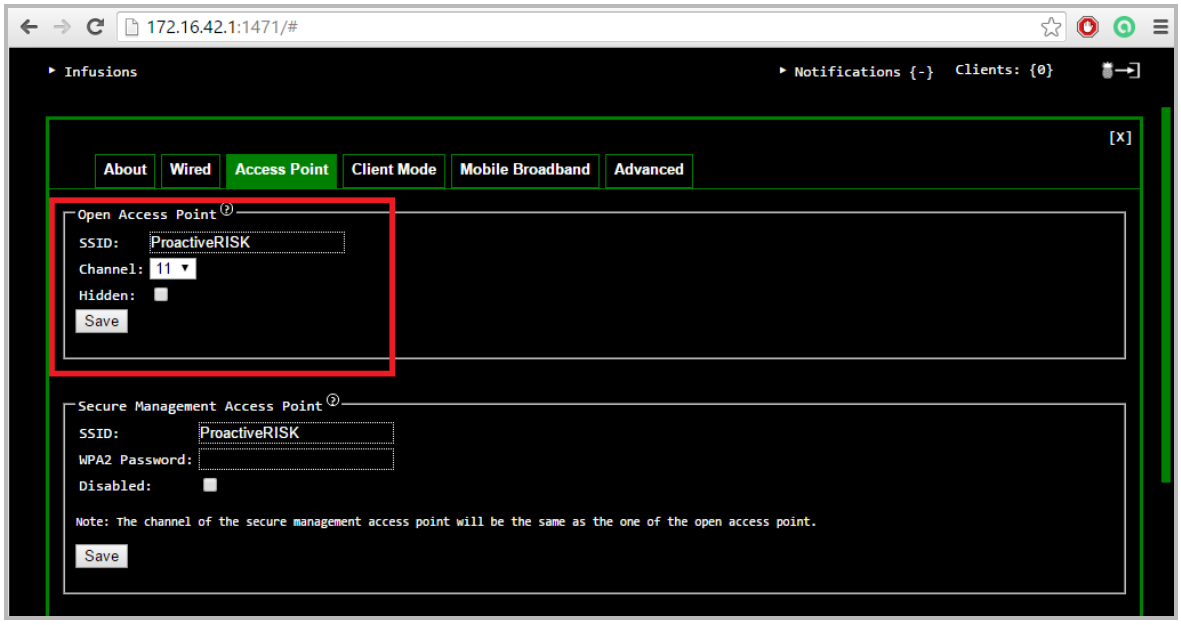
1. Open and log into the WiFi Pineapple.
2. Click the **Show** link of the *Networ*k module and it will display the internet address of the machine.



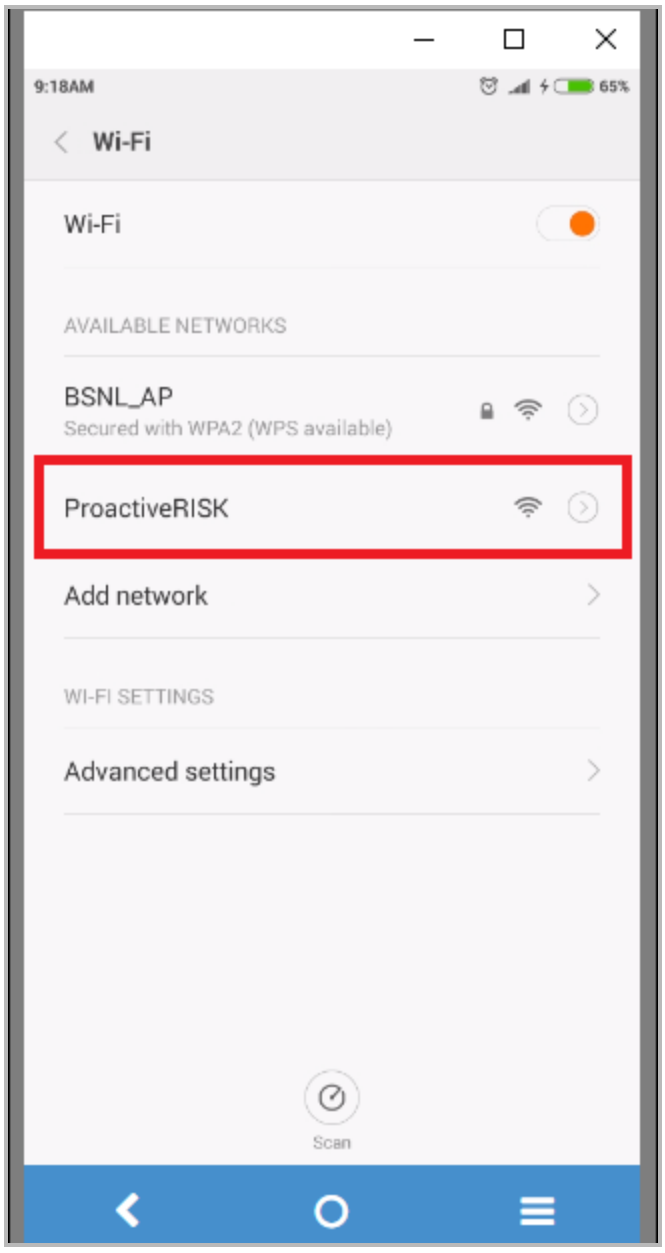
Create an open wireless network

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.

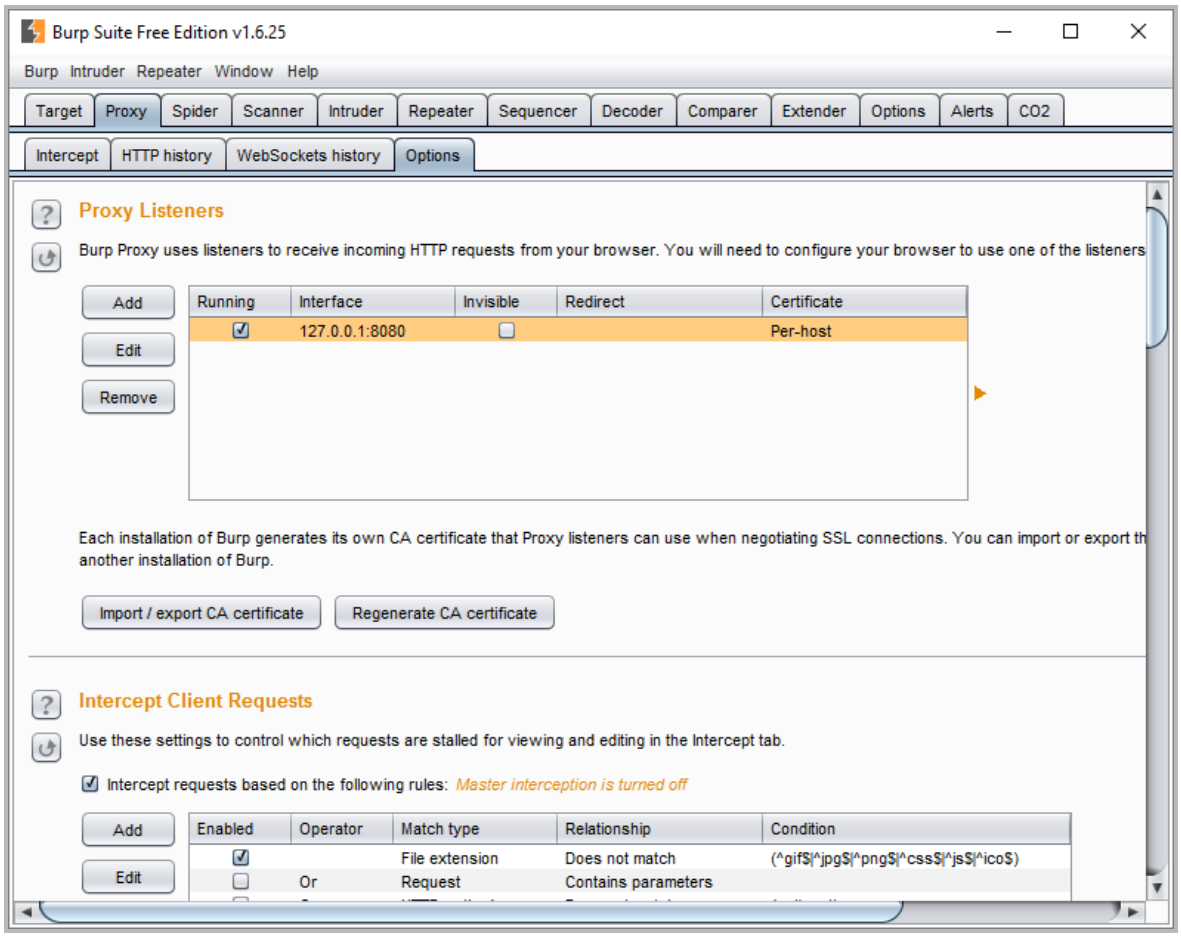


3. SSH into the WiFi Pineapple from your laptop using PuTTY or WinSCP.

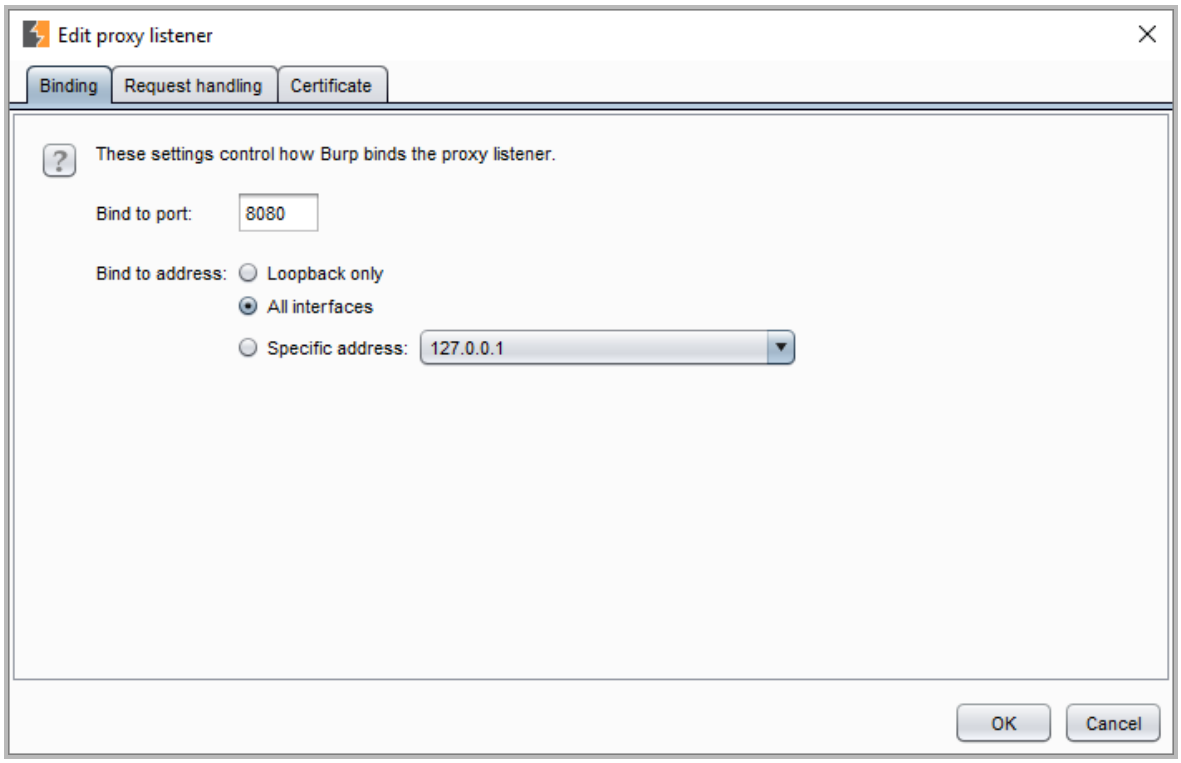
Once SSH connection has been established, execute following commands to configure the WiFi Pineapple to forward traffic.

```
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
```

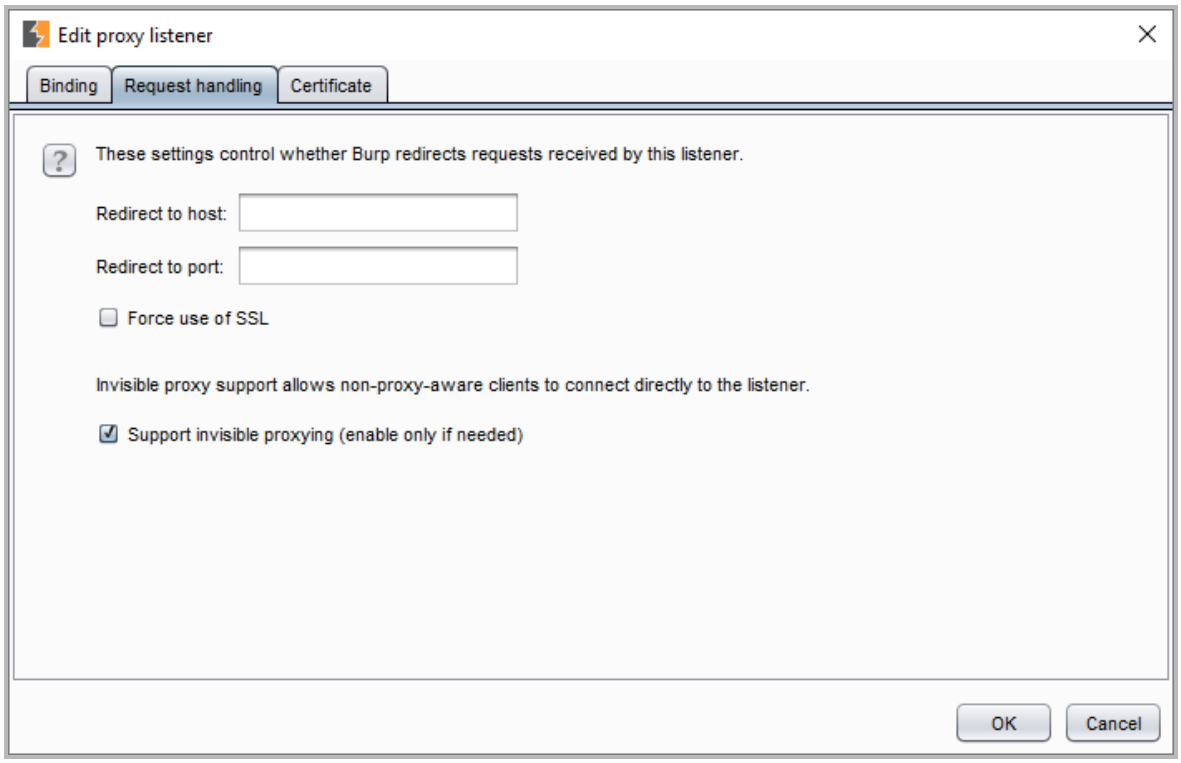
4. Open Burp Suite.
5. Navigate to **Proxy -> Options** to configure a proxy listener.
6. Select the check box for **127.0.0.1:8080** and click the **Edit** button.



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

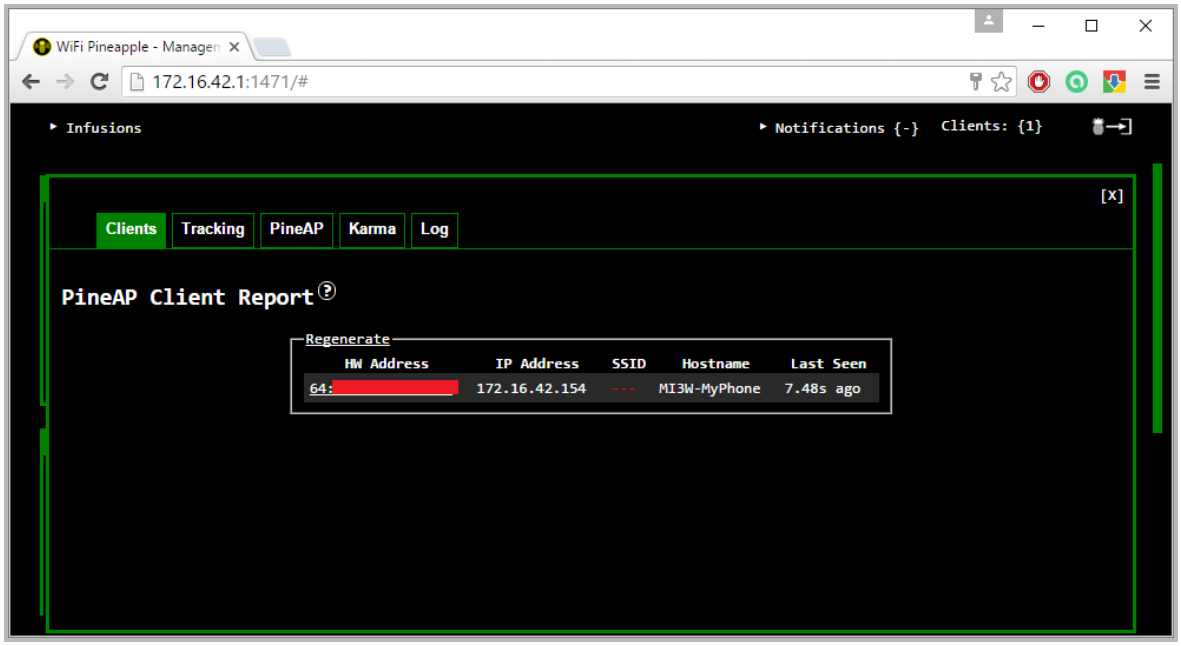


8. Select the **Request handling** tab.
9. Select the **Support invisble proxying** checkbox and click the **OK** button.
10. When prompted, select **Yes** on the confirmation pop-up to complete the setup to intercept the mobile traffic using the Burp Proxy tool.

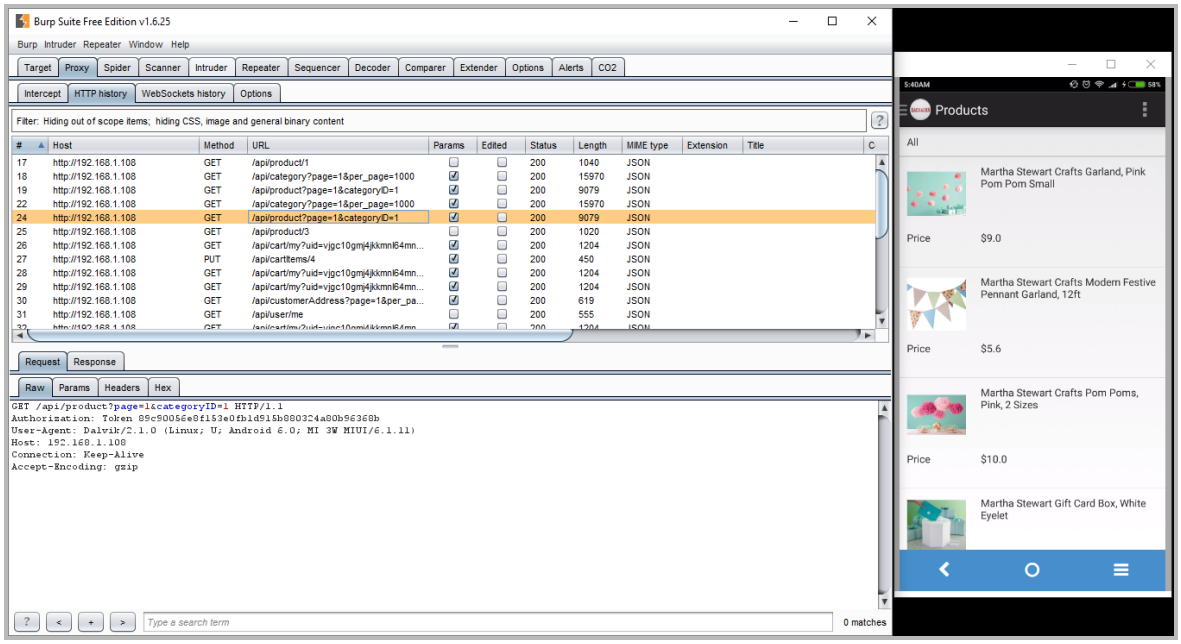


Monitor mobile application traffic

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.

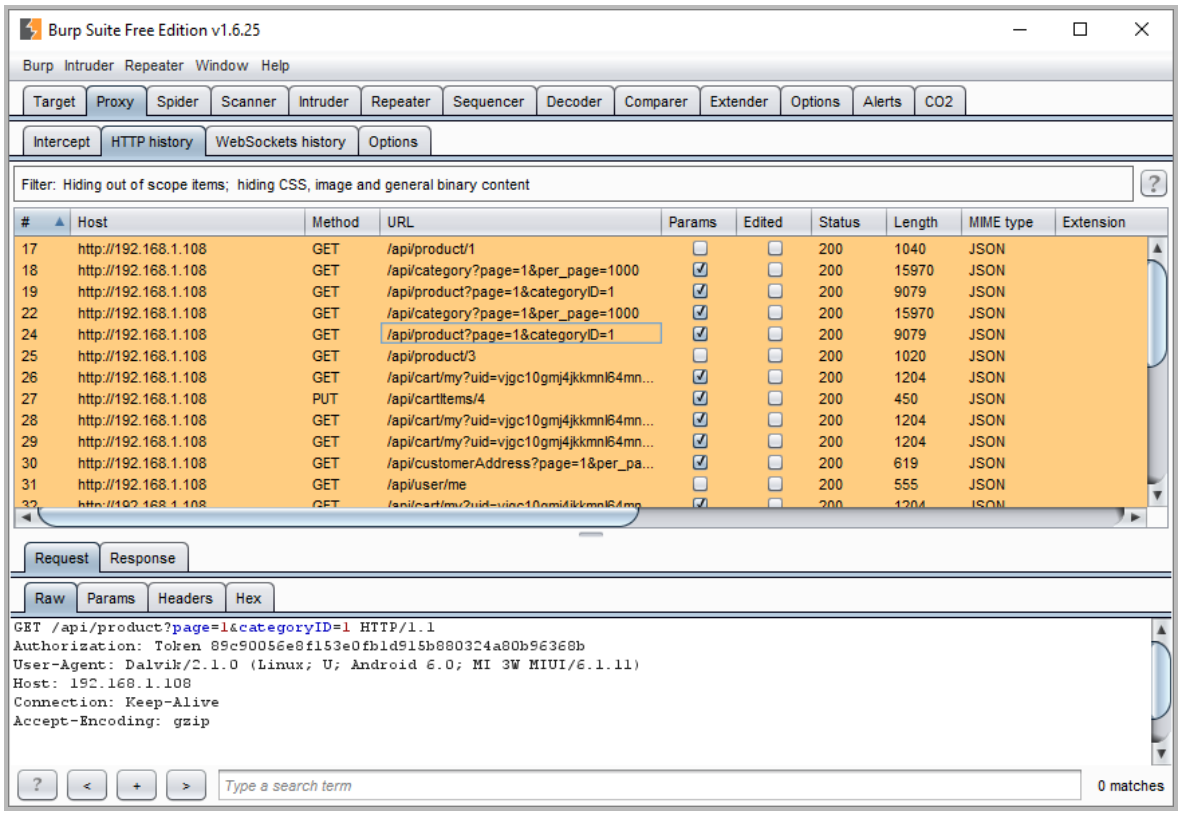


Let's assume that target device is using the Hackazon mobile application to purchase a few things. All the traffic will be monitored during this session.

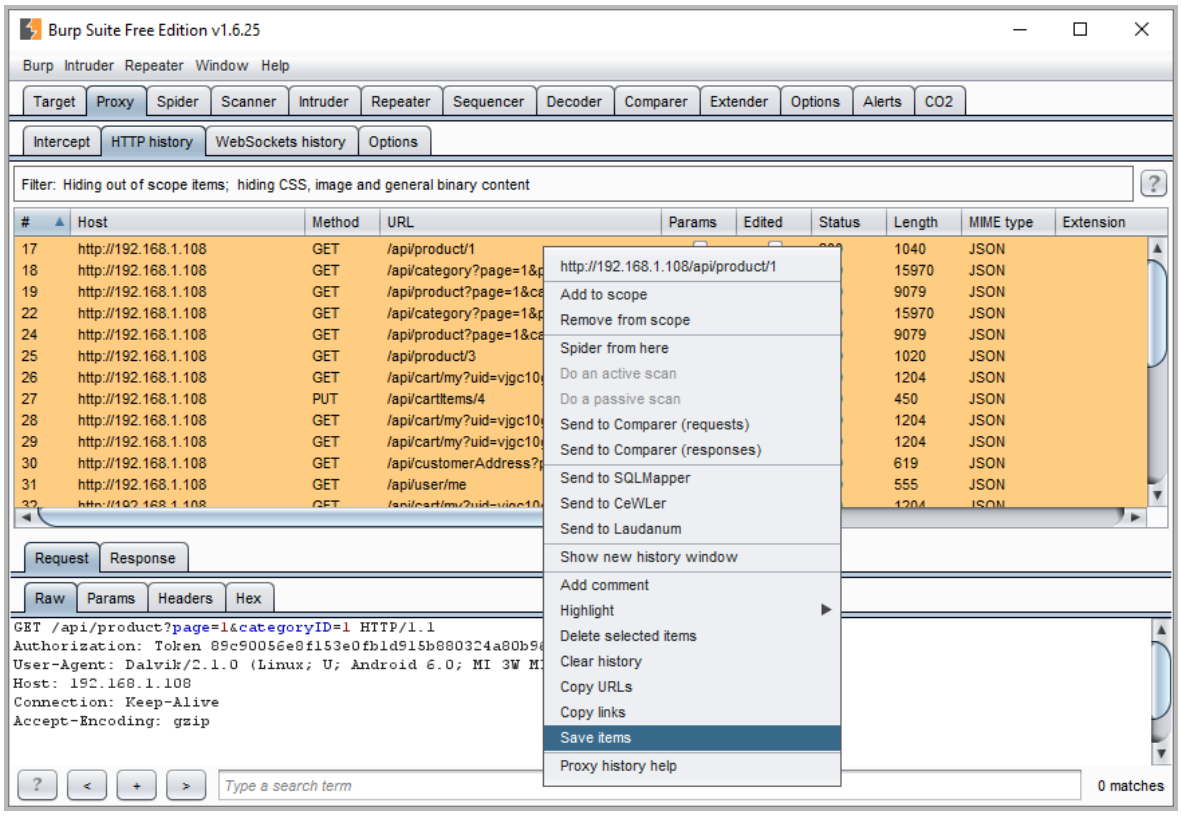


Now that you have the Hackazon mobile application traffic from the active session, the traffic can be saved, in .xml format, for use in AppSpider

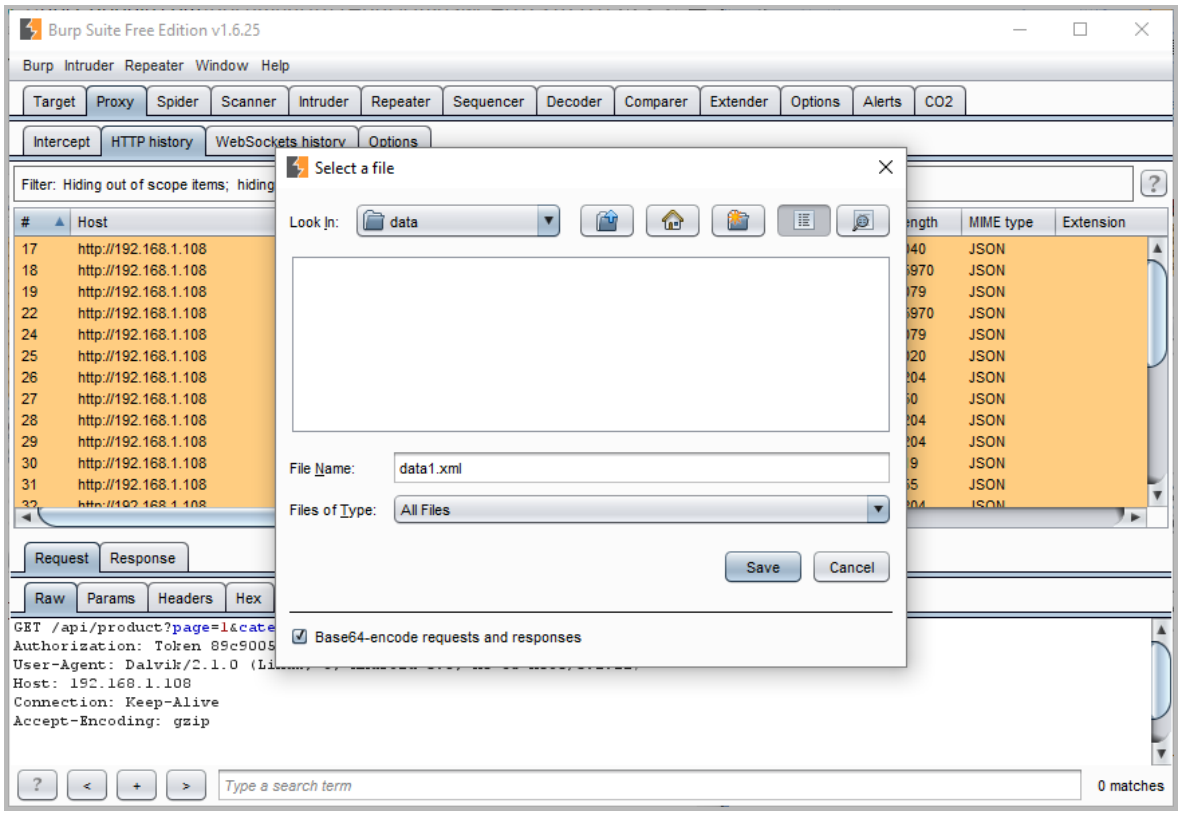
1. Hold **Ctrl + A** on your keyboard to select all of the HTTP Proxy traffic data.



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

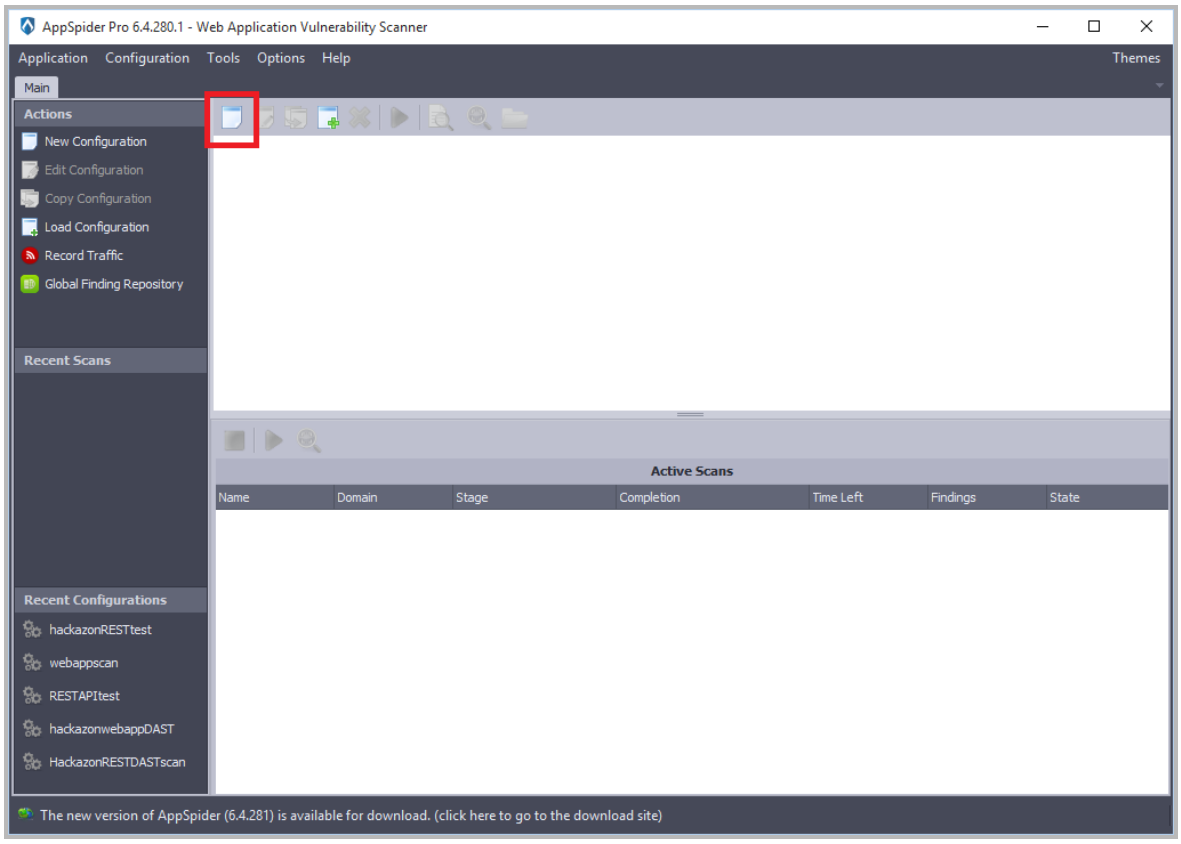


3. Name your .xml file and click the **Save** button.

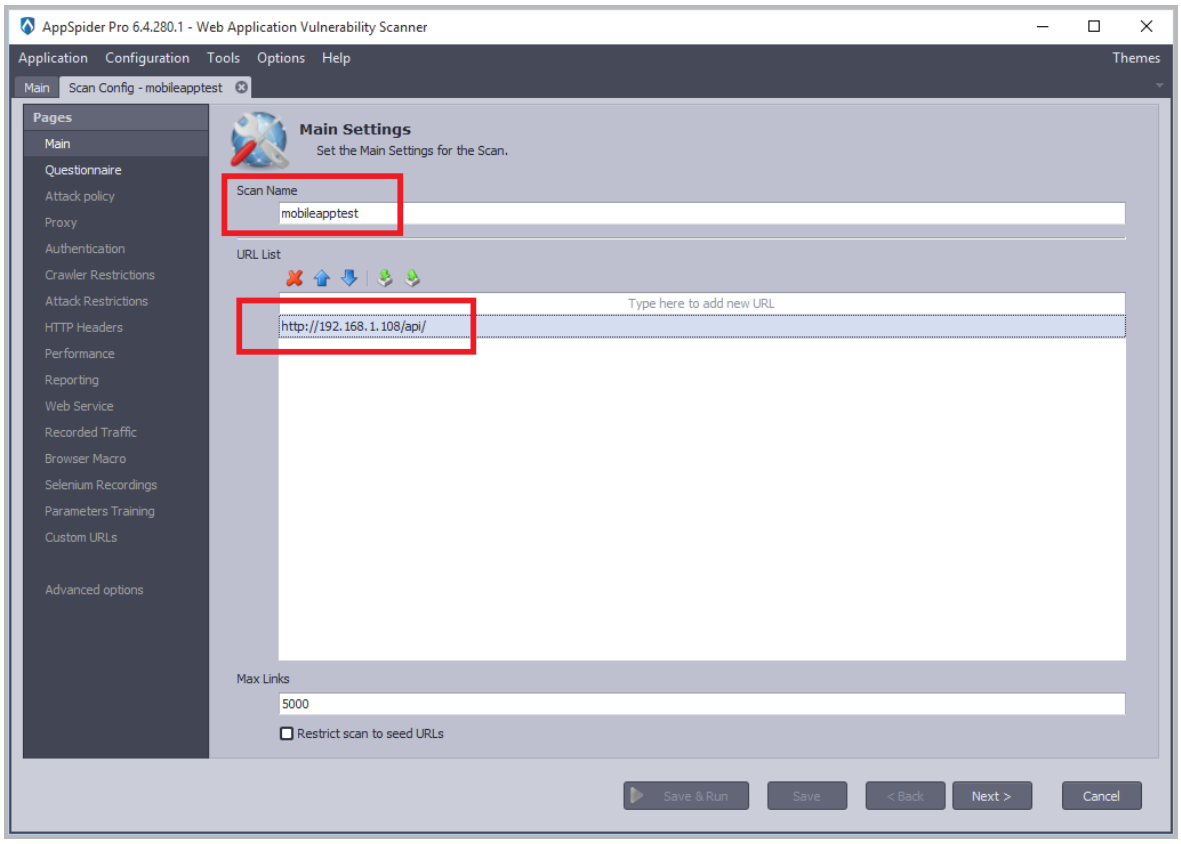


Import recorded traffic into AppSpider

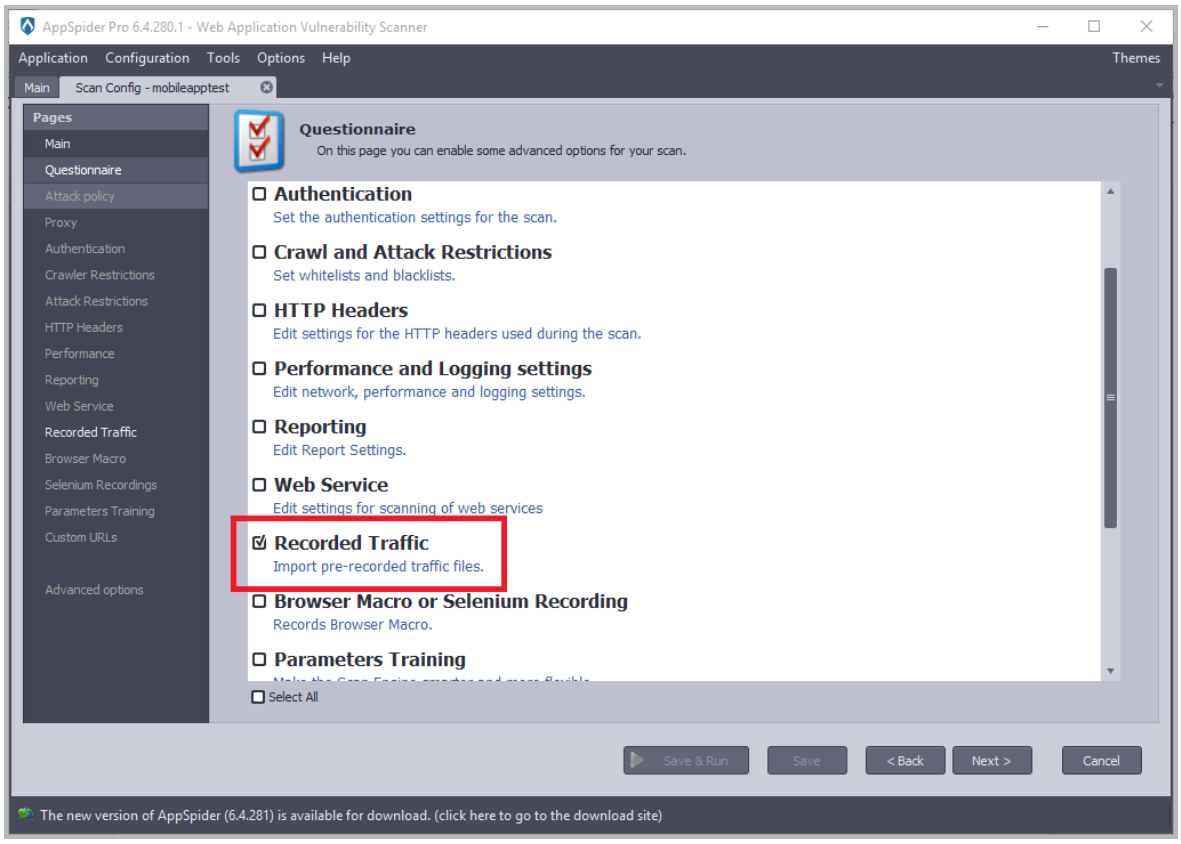
1. Open AppSpider
2. Select **New Configuration** from the *Actions* panel in AppSpider.



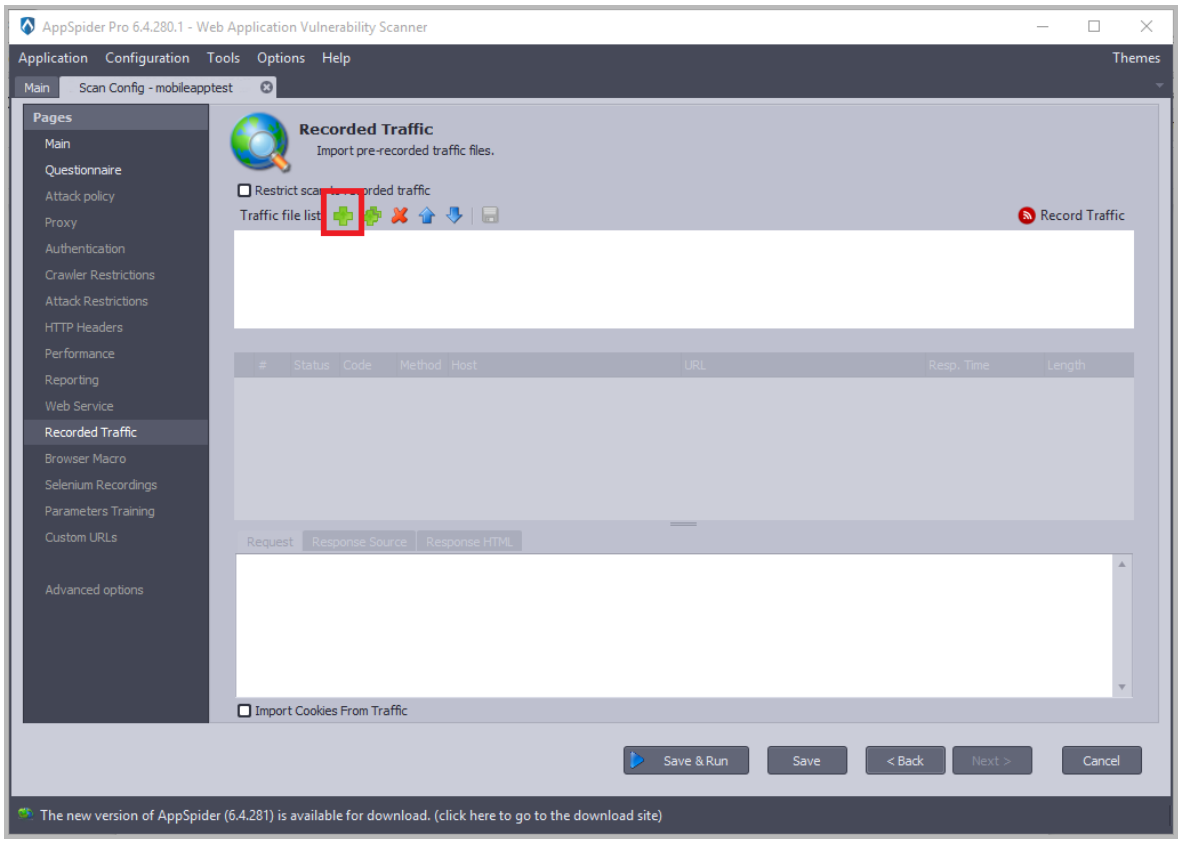
3. Enter a **Scan Name** and **URL** for your scan then click the **Next** button.



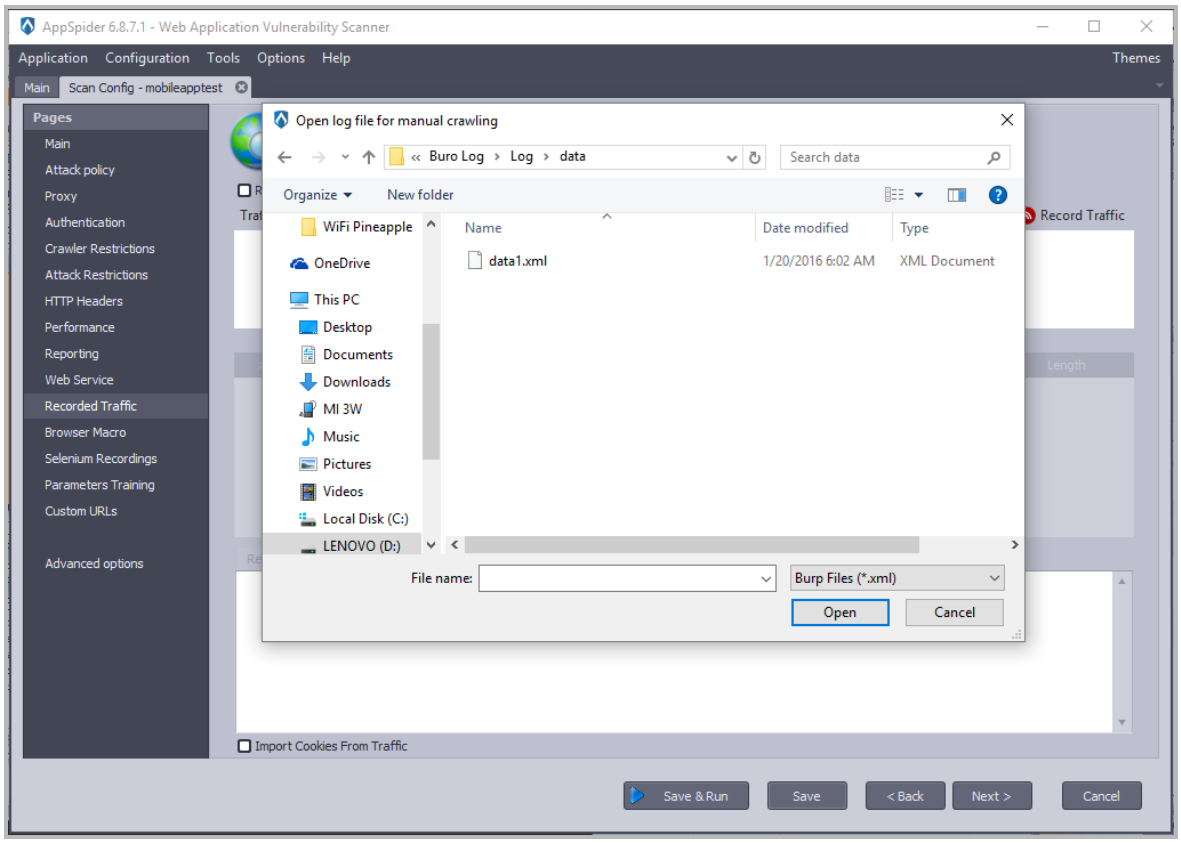
4. Select the check box for **Recorded Traffic** on the *Questionnaire* and click the **Next** button.



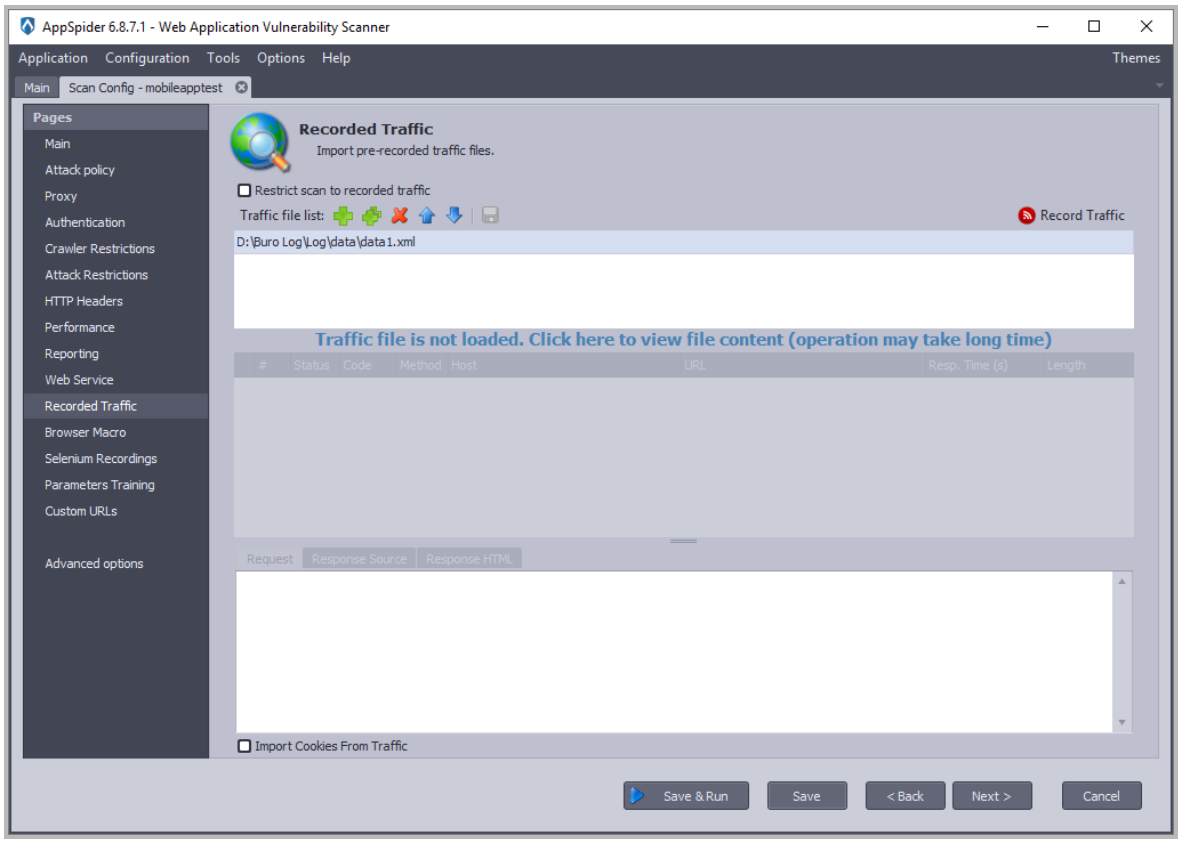
5. Click the **Add File (+)** icon and import the .xml file containing the recorded traffic.



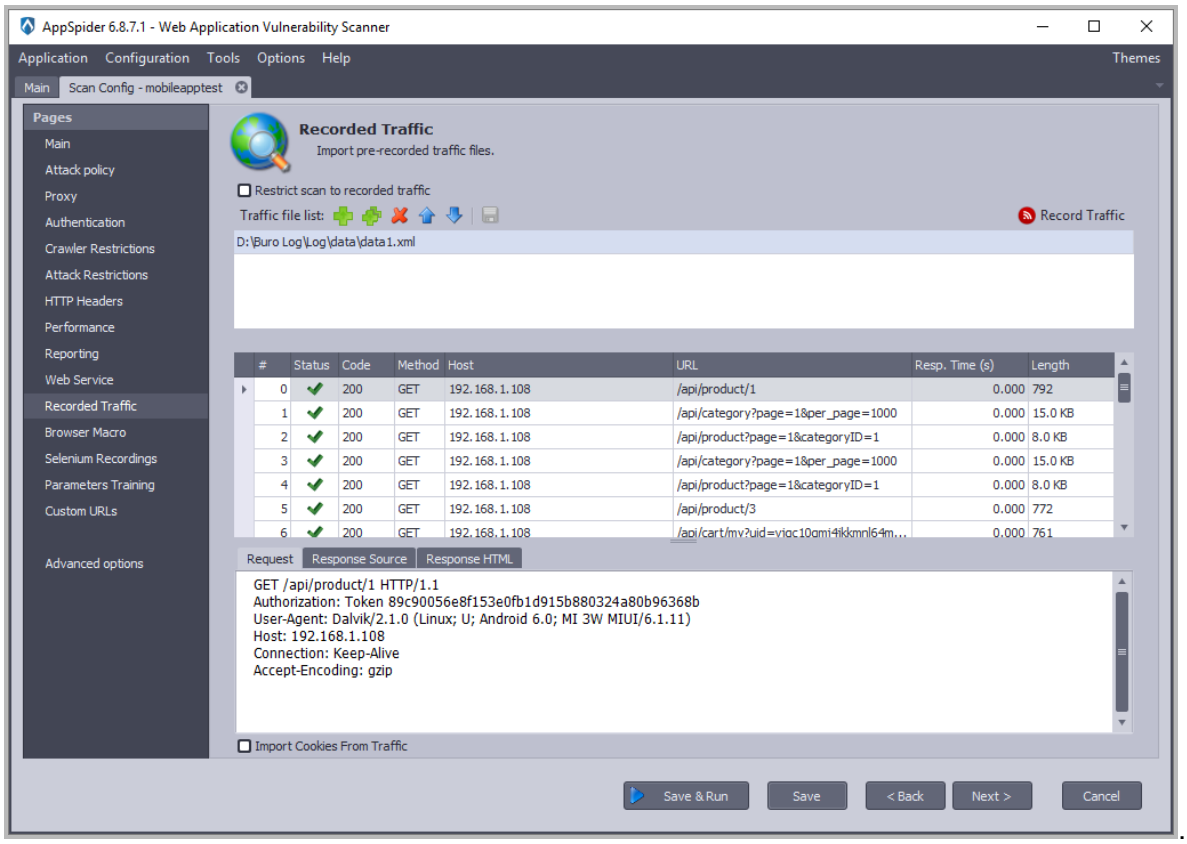
6. Change the file type to *Burp Files (*.xml)*, highlight the file you created with the Burp Proxy tool, and click the **Open** button.



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.


AppSpider 6.8.7.1 - Web Application Vulnerability Scanner
Application Configuration Tools Options Help
Themes
Main
Scan Config - mobileapptest
Pages
Main
Attack policy
Proxy
Authentication
Crawler Restrictions
Attack Restrictions
HTTP Headers
Performance
Reporting
Web Service
Recorded Traffic
Browser Macro
Selenium Recordings
Parameters Training
Custom URLs
Advanced options
Recorded Traffic
Import pre-recorded traffic files.
Restrict scan to recorded traffic
Traffic file list:
Record Traffic
D:\Buro Log\Log\data\data1.xml
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/my?uid=viqc10qmi4ikkmnl64m... 0.000 761
Request
Response Source
Response HTML
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
Import Cookies From Traffic
Save & Run
Save
< Back
Next >
Cancel

AppSpider Swagger Utility

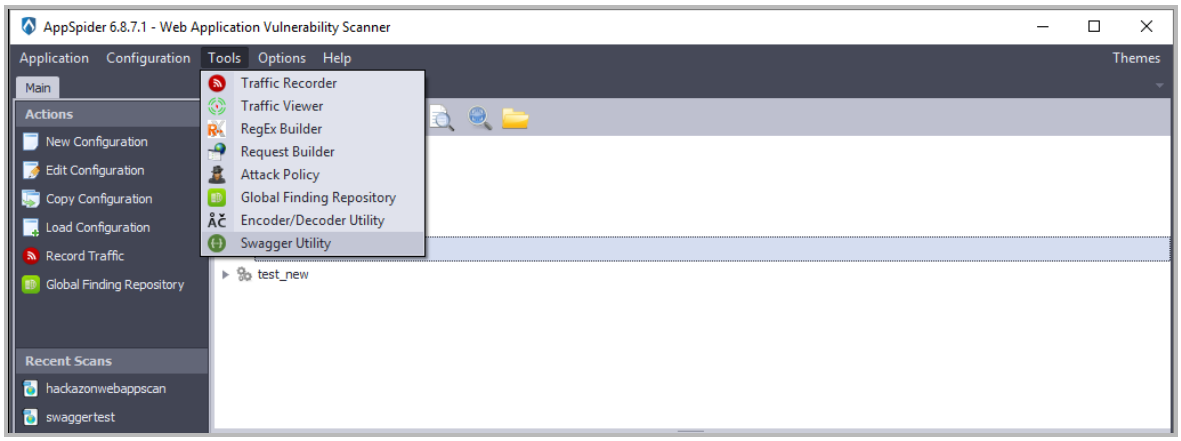
AppSpider Pro has a new functionality. The Swagger Utility allows you to upload Swagger REST API documents to enable the API to be scanned in AppSpider.

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.

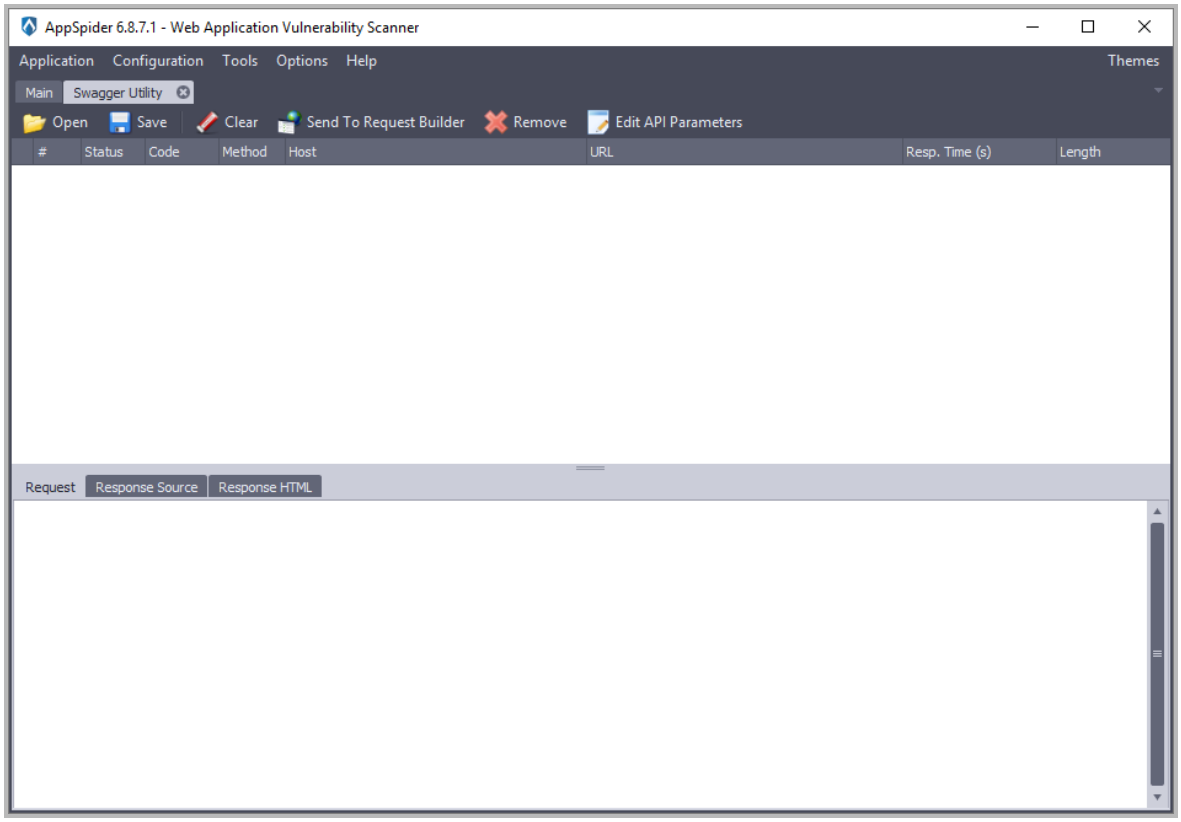
```
{
    "swagger": "2.0",
    "info": {
        "version": "0.0.0",
        "title": "Hackazon modern web app"
    },
    "host": "192.168.1.108/api",
    "schemes": [
        "http"
    ],
    "paths": {
        "/category": {
            "get": {
                "parameters": [
                 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

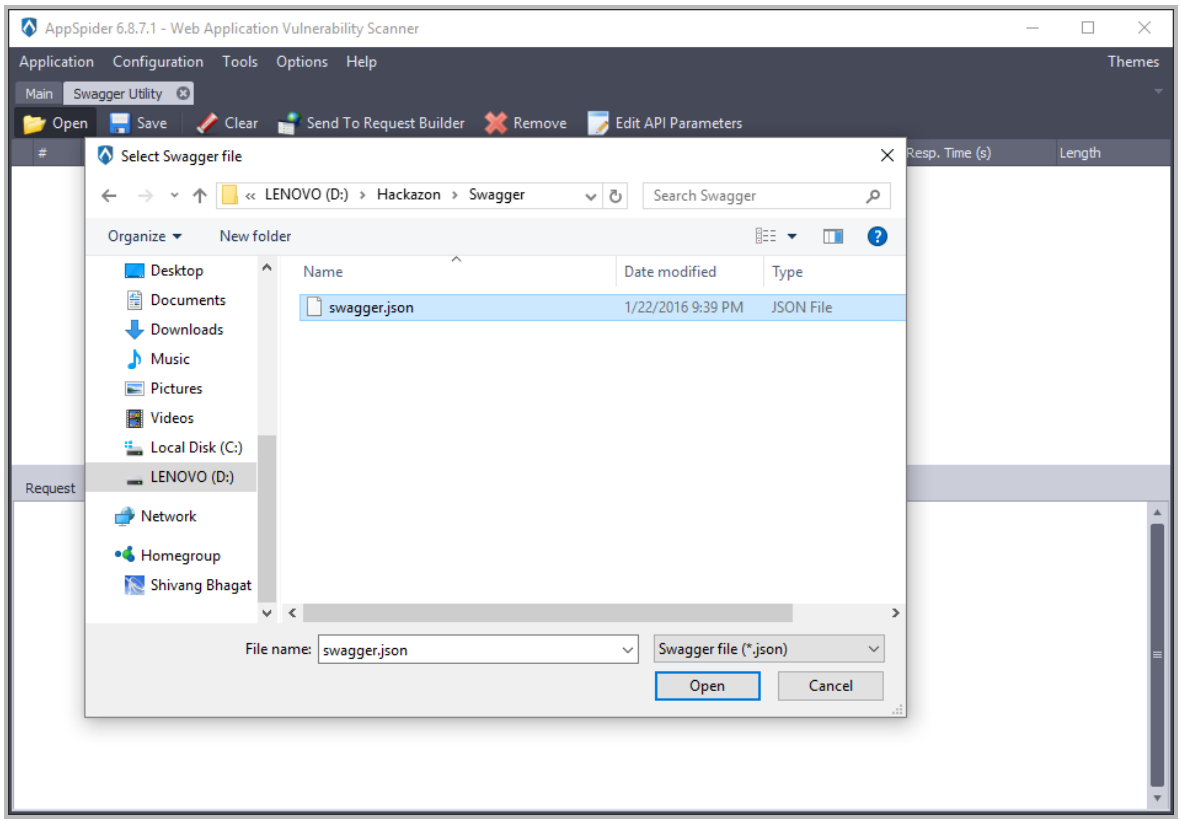
1. Open AppSpider and select **Swagger Utility** from the *Tools* menu.



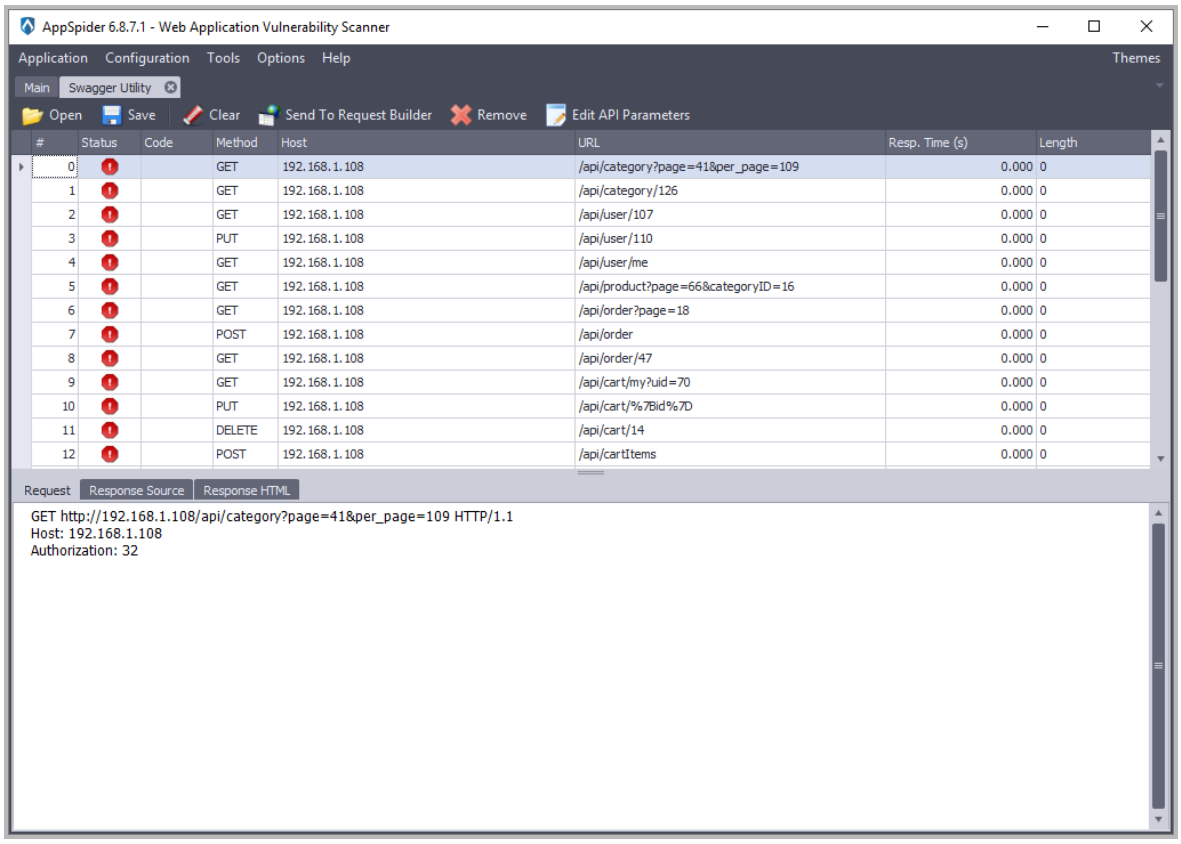
A new tab will open with the Swagger Utility.



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.

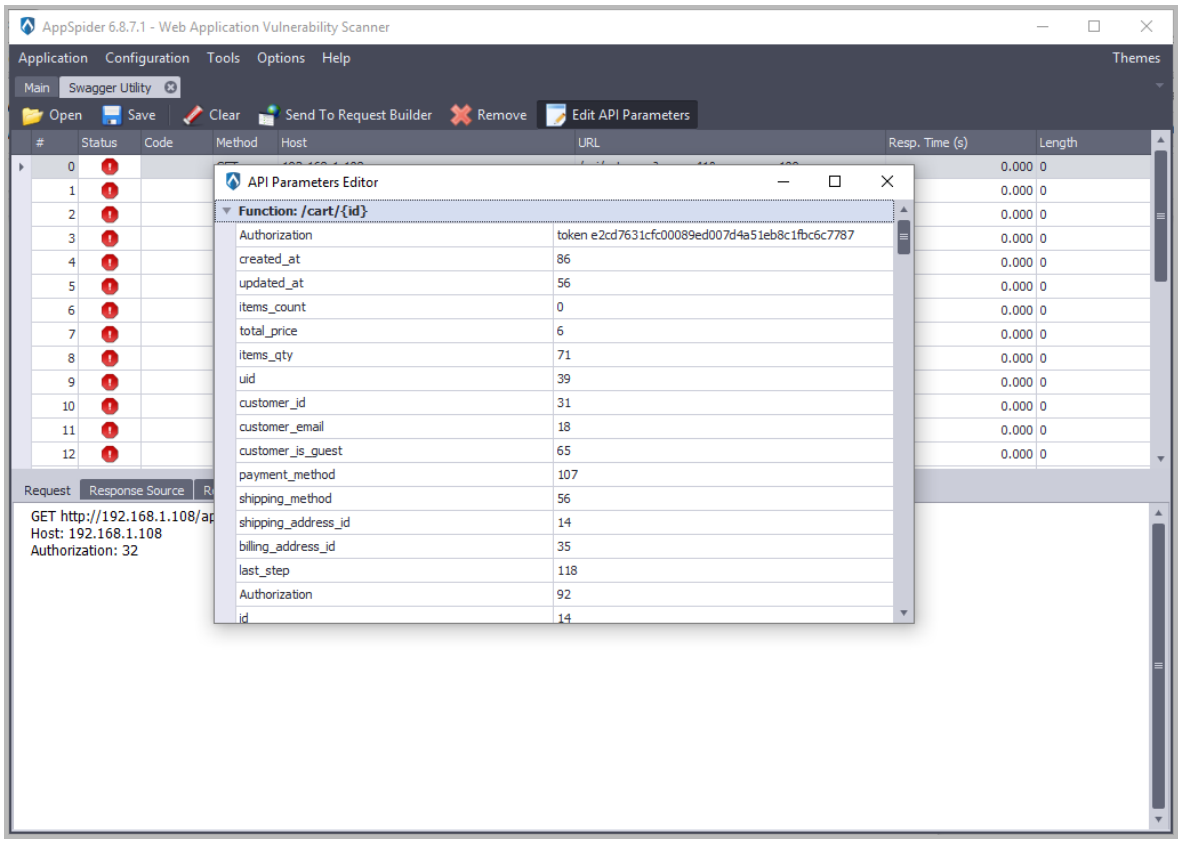


4. API function calls will be displayed in the traffic viewer window.



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.

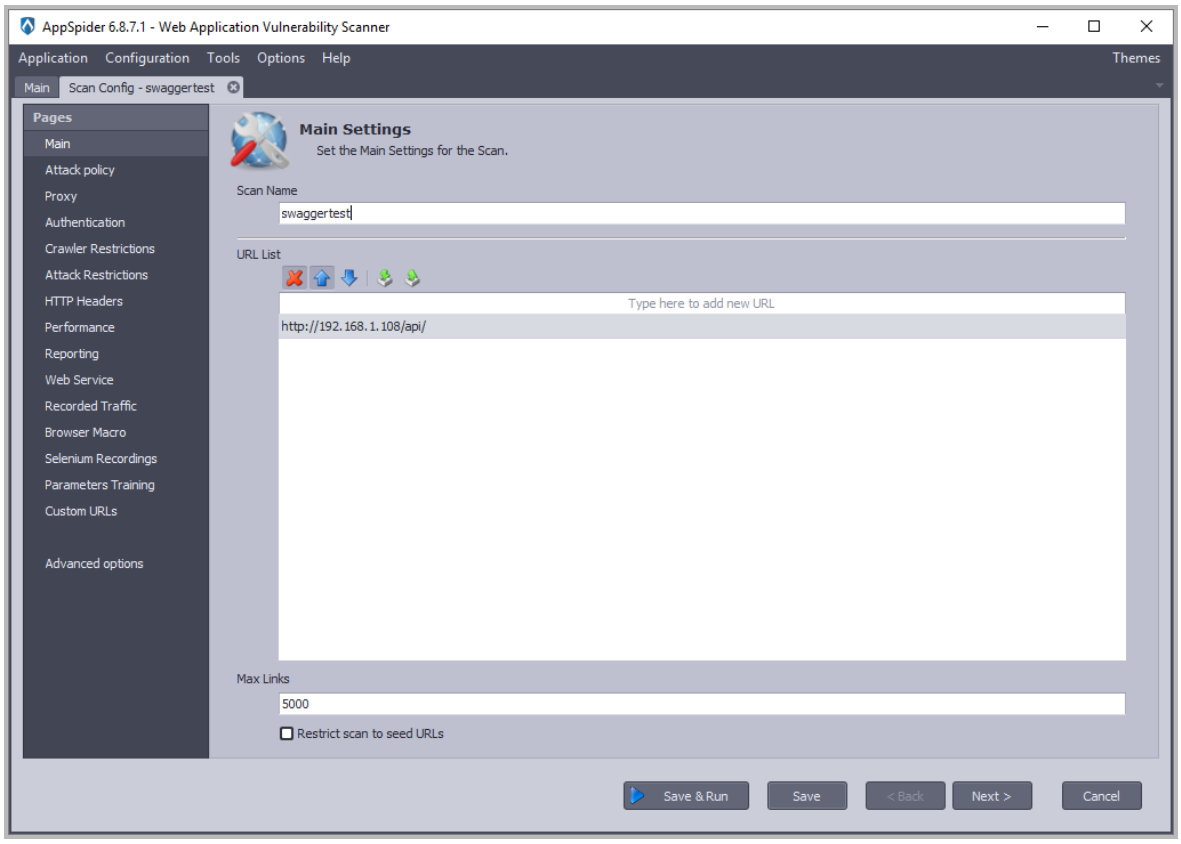
This file is now ready to be added to a scan configuration.



Creating a new scan configuration

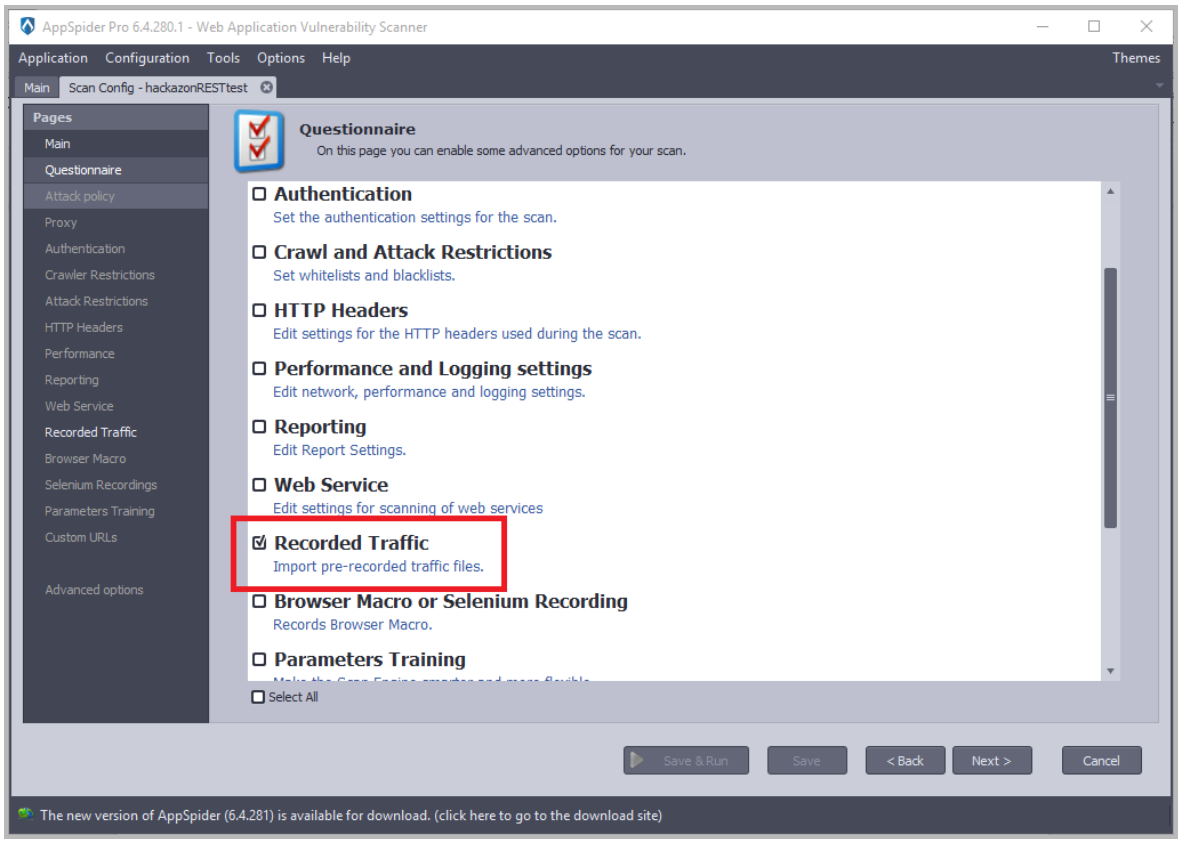
You can create a new scan configuration using the traffic recording file that you created with the Swagger Utility tool. It is recommended that you include the base URL of the REST API in the URL list when you create the scan configuration. This ensures that the same domain restrictions that apply to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

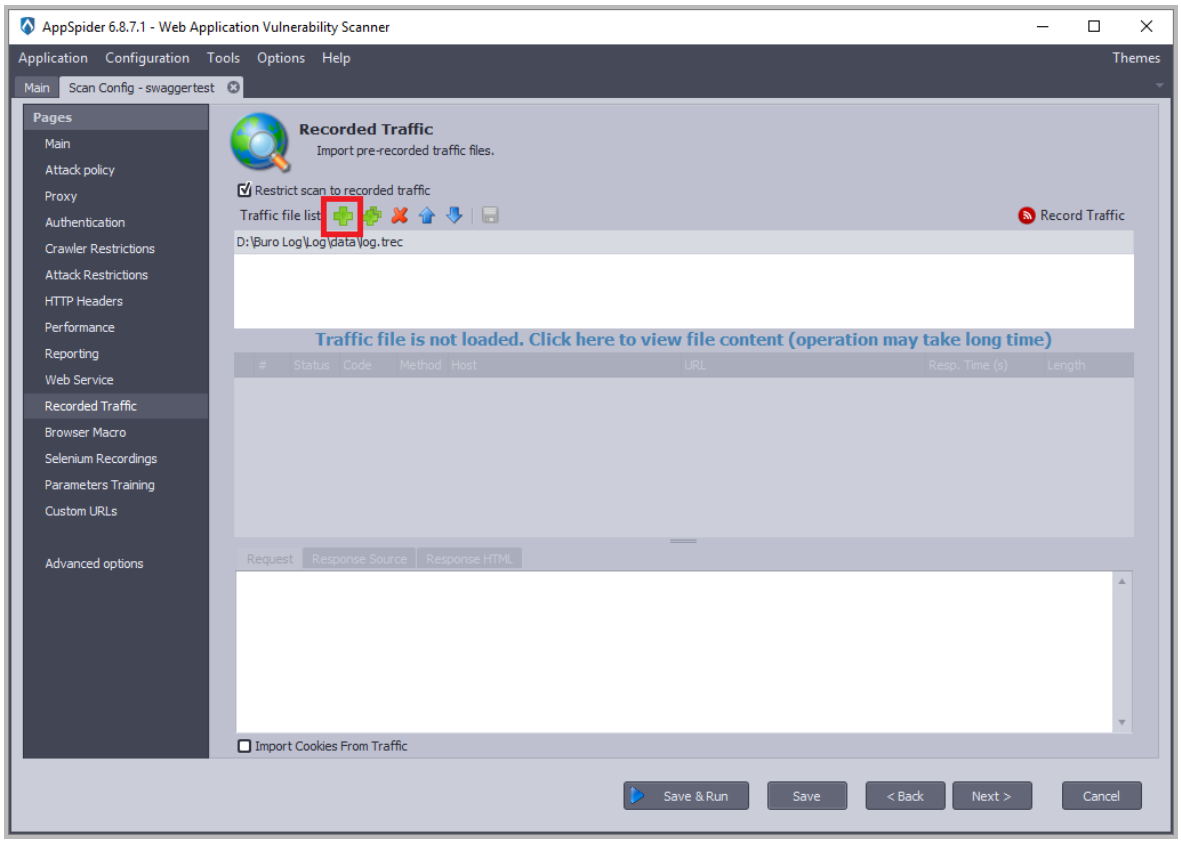


The *Questionnaire* allows users to enable advanced options for the scan configuration.

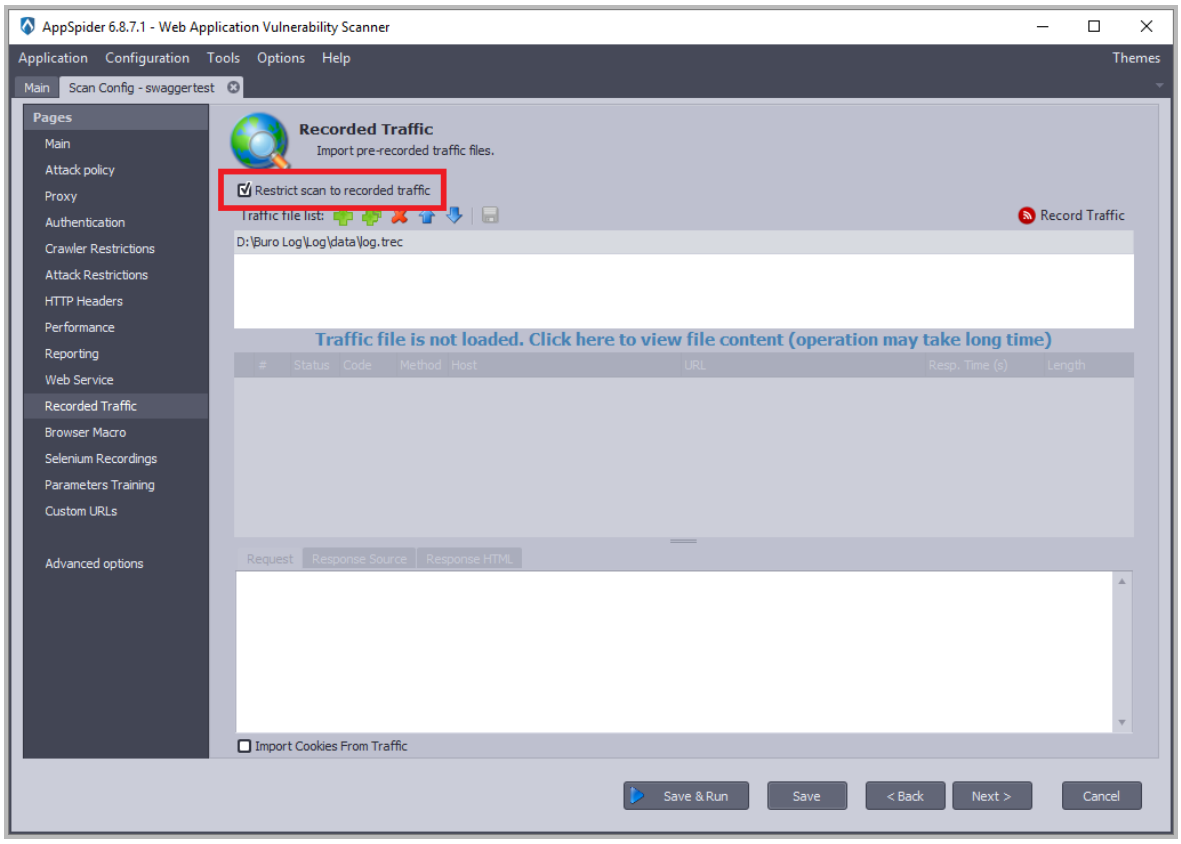
3. Select the check box for **Recorded Traffic** on the *Questionnaire* and click the **Next** button.



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.


AppSpider 6.8.7.1 - Web Application Vulnerability Scanner
Application
Configuration
Tools
Options
Help
Themes
Main
Scan Config - swaggertest
Pages
Main
Attack policy
Proxy
Authentication
Crawler Restrictions
Attack Restrictions
HTTP Headers
Performance
Reporting
Web Service
Recorded Traffic
Browser Macro
Selenium Recordings
Parameters Training
Custom URLs
Advanced options
Recorded Traffic
Import pre-recorded traffic files.
Restrict scan to recorded traffic
Traffic file list:
Record Traffic
D:\Buro Log\Log\data\log.trec
Traffic file is not loaded. Click here to view file content (operation may take long time)
#
Status
Code
Method
Host
URL
Resp. Time (s)
Length
Request
Response Source
Response HTML
Import Cookies From Traffic
Save & Run
Save
< Back
Next >
Cancel