Burp Suite Bootcamp

Written by Joe McCray

Contributors:

Tino Brants
Aaron Levin

Table of Contents

Section 1: Basics **3**

Lab 1: VM Download and Setup 3
Lab 2: Configuring Burp Suite 4
Lab 3: Running Burp Suite 15
Lab 4: Adding RSnake RFI List 21
Lab 5: Dealing with reCAPTCHA 38
Lab 6: Working with NBFS-Encoded WCF Communications in BurpSuite 41
Lab 7: Testing SOAP Web Services by Integrating SoapUI and BURP 44
Lab 8: Using Burp & Wsdler to Hacking Web Services 55
Lab 9: Burp Suite Through Tor/Privoxy 61
Lab 10: Masking Nikto Headers 63

Section 3: Extending Burp Suite **68**

Lab 11: Burp Python 68
Lab 12: BurpExtender-w3af 78

Section 4: Burp Suite & Mobile (Additional Material) **83**

Lab 13: Burp on iPhone 83
Lab 15: SSL issues and Android 93
Lab 16: Burping Android 99

Section 1: Basics

Lab 1: VM Download and Setup

The attack virtual machine used in this workshop can be downloaded from:
https://s3.amazonaws.com/StrategicSec-VMs/StrategicsecUbuntu-v3.zip
username: strategicsec
password: strategicsec

Download, extract, and login to this virtual machine with the credentials provided above.

Flush(delete) iptables and cd to the toolz directory:

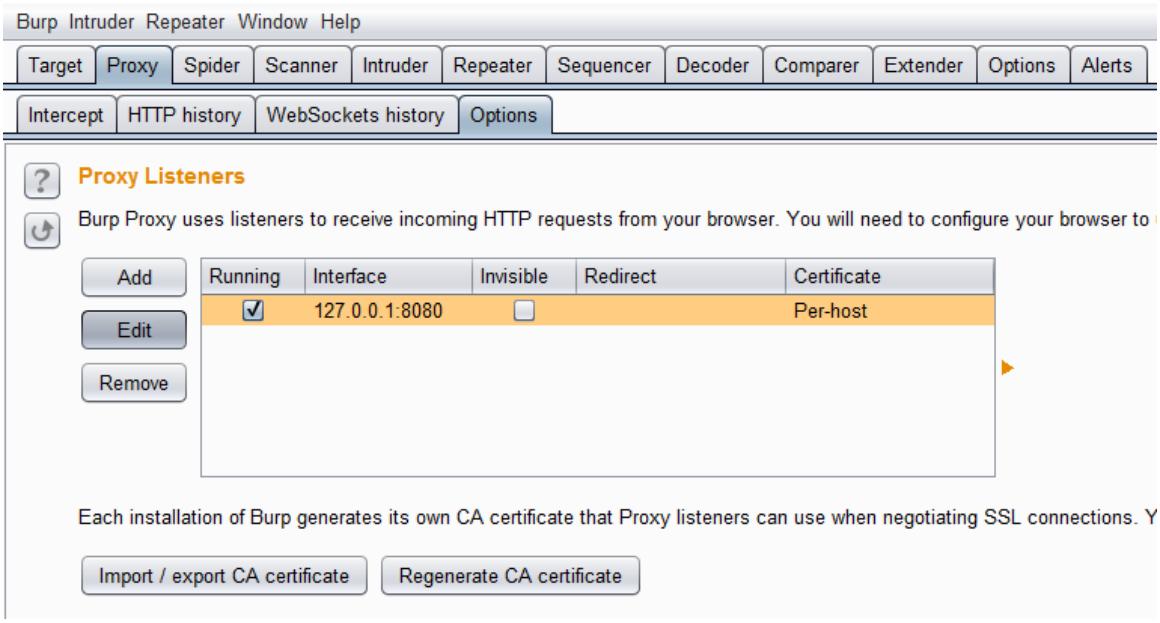
```
sudo /sbin/iptables -F
cd /home/strategicsec/toolz
```

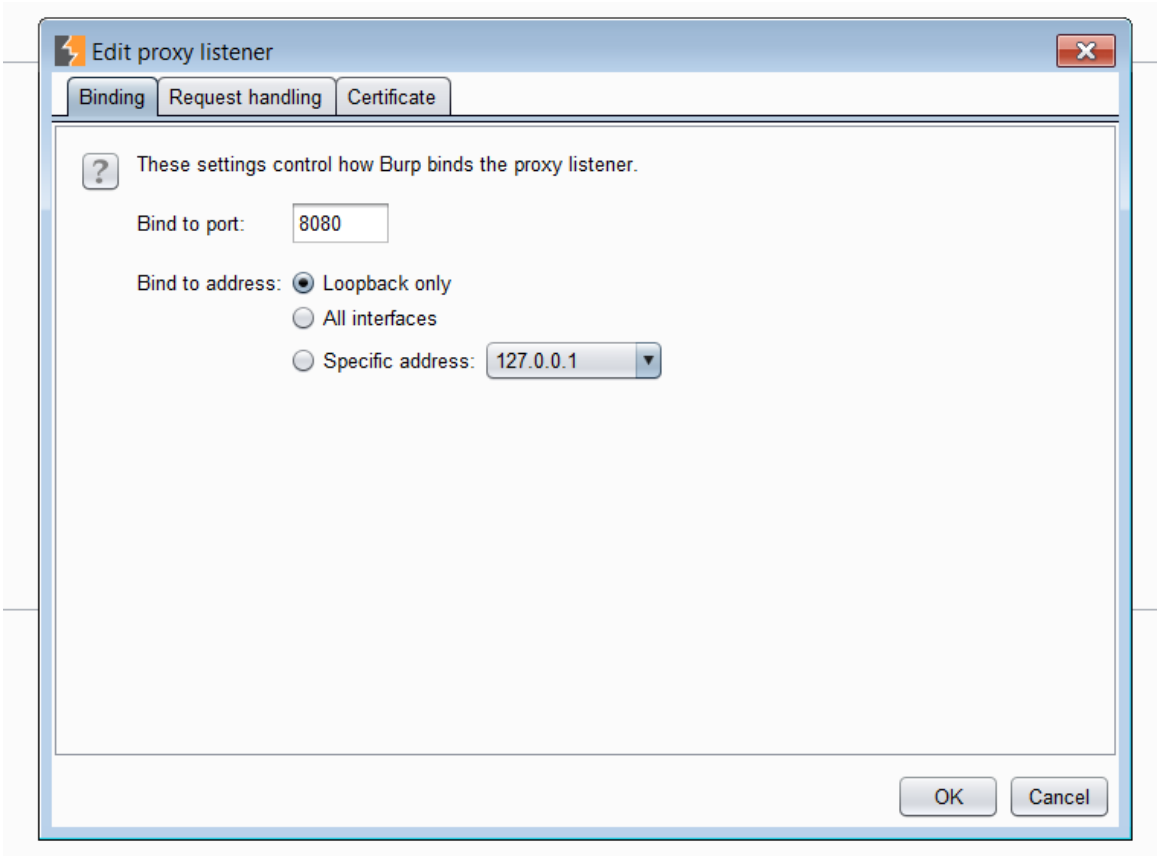
Lab 2: Configuring Burp Suite

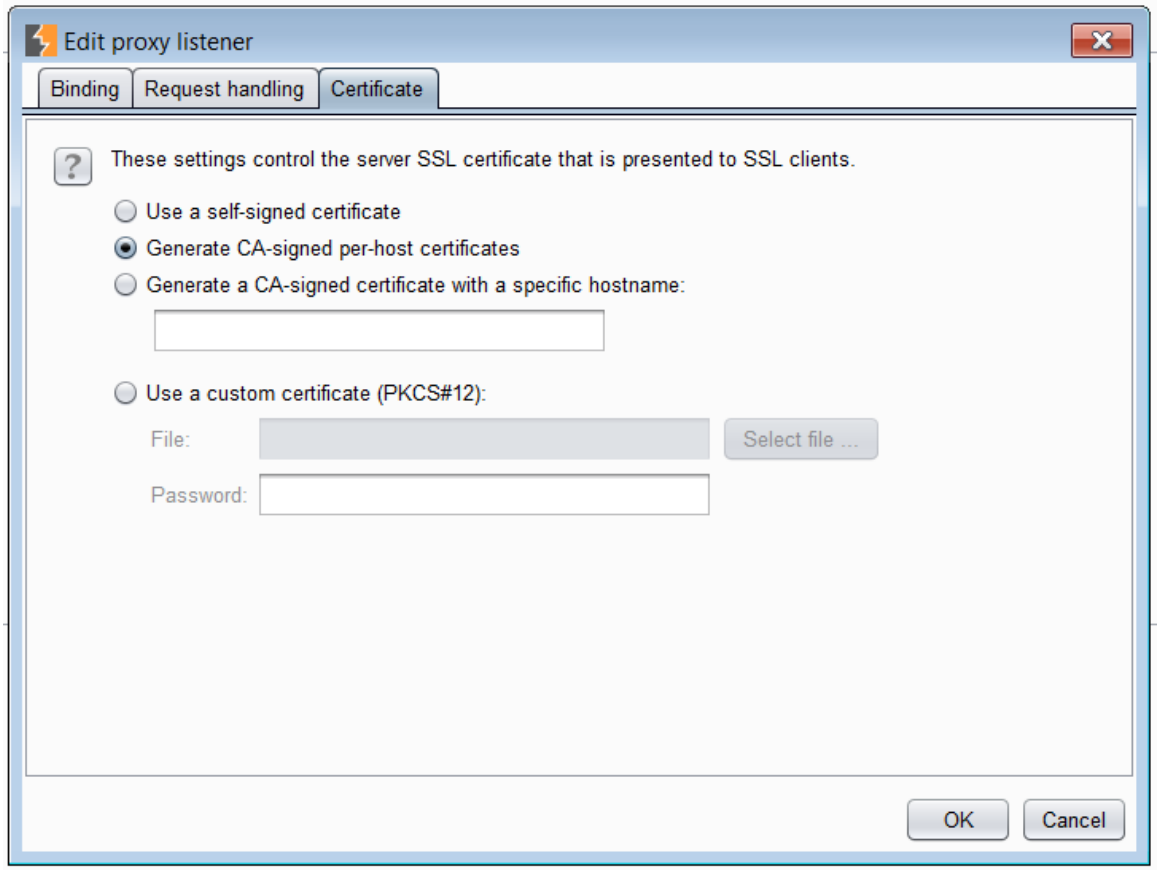
Download latest free version of Burp at
http://www.portswigger.net/burp/download.html
Make sure that burpsuite_free_v1.6.31.jar is set as executable (chmod +x burpsuite_free_v1.6.31.jar) and then run:

java -jar burpsuite_free_v1.6.31.jar

- Click the "Proxy" tab
- Click the "Options" sub tab
- Click “Edit” in the “Proxy Listeners” section
- In the “Edit proxy listener” pop up select “Binding Tab” select “loopback only”
- In the same pop up make sure that the bind port is 8080
- In the same pop up select the “Certificate” tab
- Ensure that burp is configured to "generate CA-signed per-host certificates"

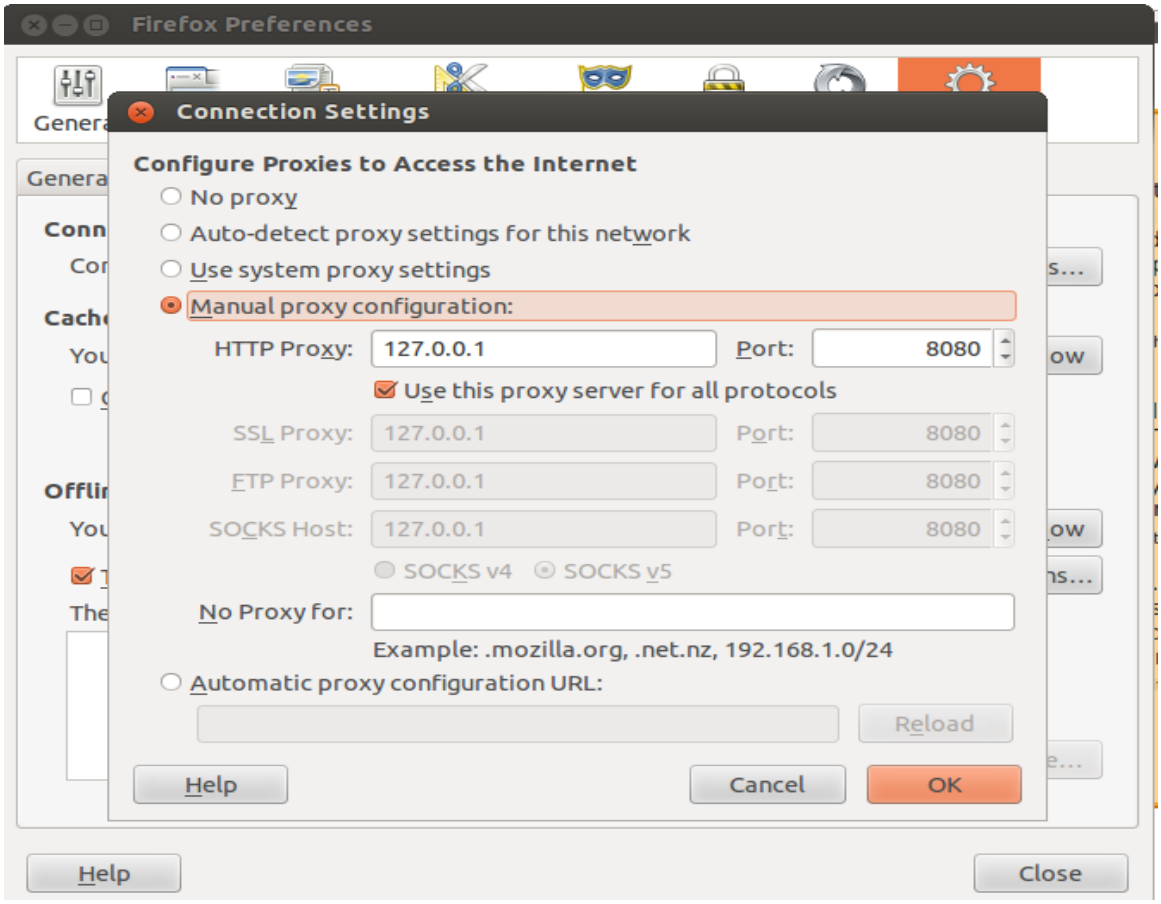



Edit proxy listener
Binding
Request handling
Certificate
These settings control how Burp binds the proxy listener.
Bind to port:
8080
Bind to address:
Loopback only
All interfaces
Specific address:
127.0.0.1
OK
Cancel



Open Firefox

- Click "Edit"
- Click “Preferences"
- Click the "Advanced" tab
- Click the "Network" sub tab
- Click the connection "settings" button
- Click "manual proxy configuration"
  set it to 127.0.0.1 port 8080
  check "Use this proxy server for all protocols"
- Remove both the "localhost, 127.0.0.1" text from the "No Proxy For:" line

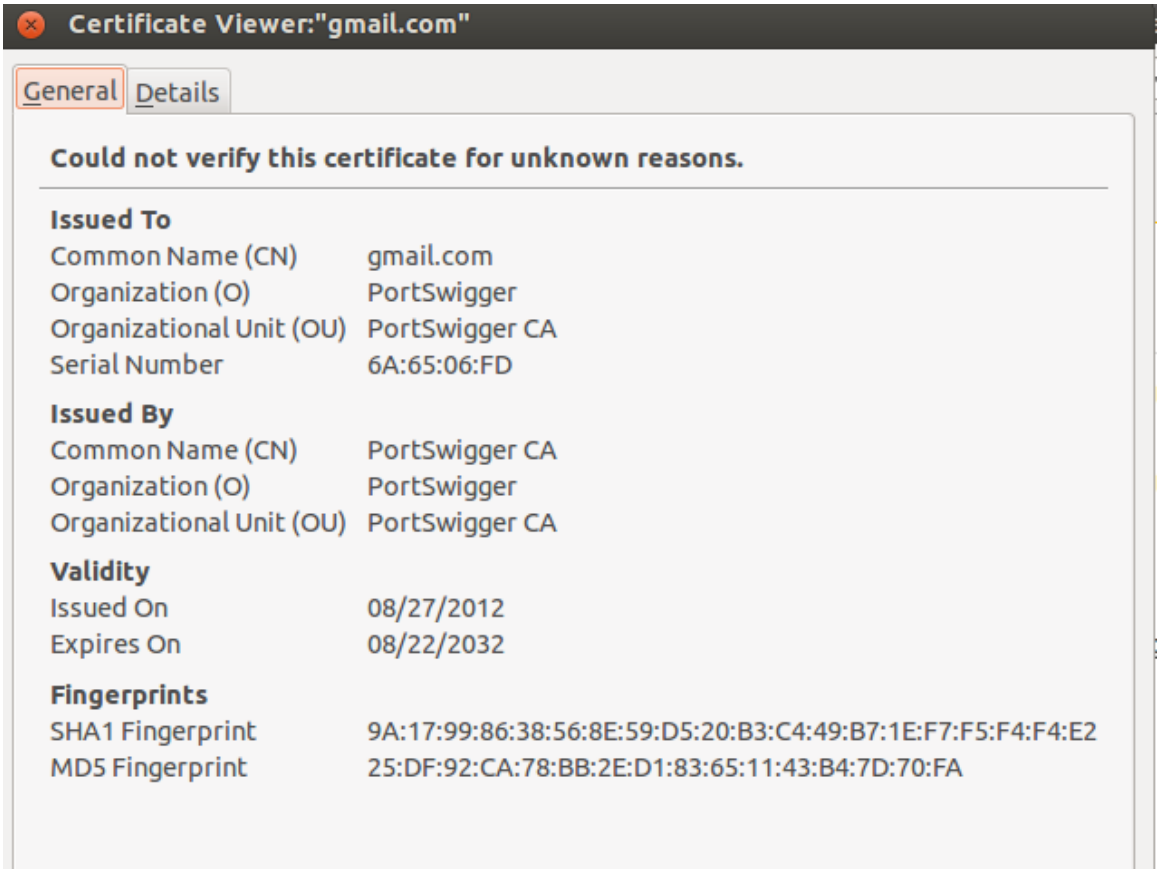


Configure your browser to use Burp as its proxy, and configure Burp's proxy listener to generate CA-signed per-host certificates.

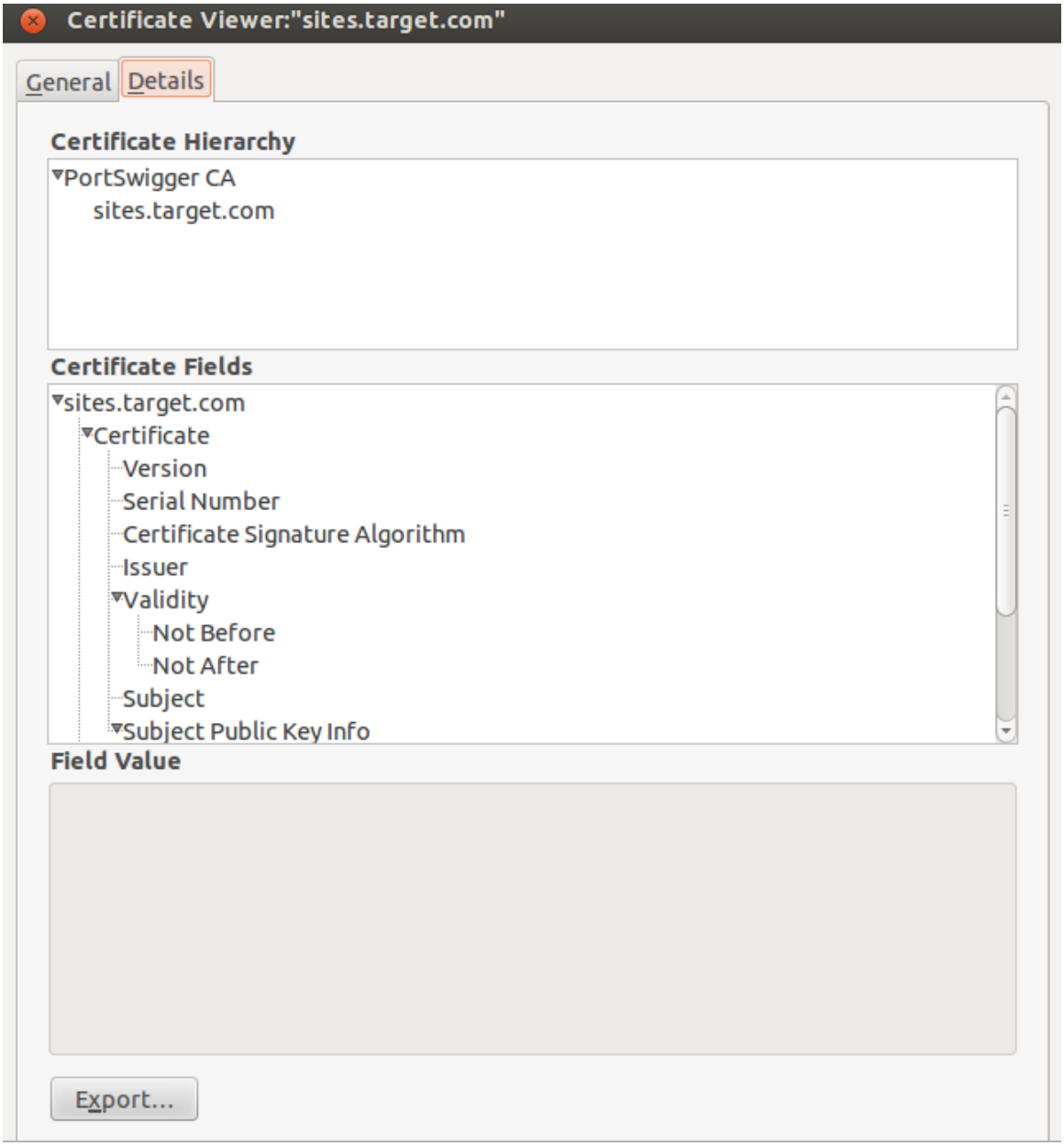
Visit any SSL-protected URL.

On the “**This Connection is Untrusted**” screen, click on “**Add Exception**”

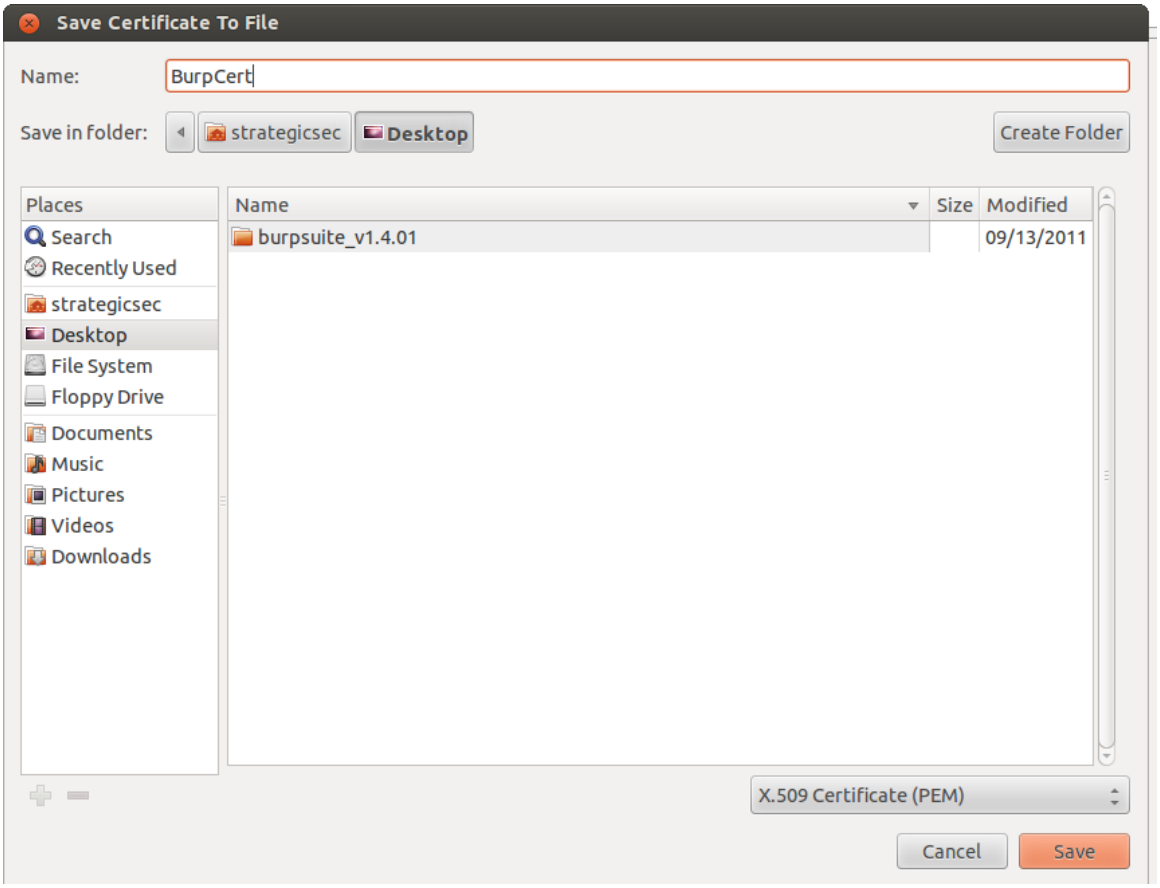
Click "**Get Certificate**", then click "**View**".


Certificate Viewer:"gmail.com"
General
Details
Could not verify this certificate for unknown reasons.
Issued To
Common Name (CN) gmail.com
Organization (O) PortSwigger
Organizational Unit (OU) PortSwigger CA
Serial Number 6A:65:06:FD
Issued By
Common Name (CN) PortSwigger CA
Organization (O) PortSwigger
Organizational Unit (OU) PortSwigger CA
Validity
Issued On 08/27/2012
Expires On 08/22/2032
Fingerprints
SHA1 Fingerprint 9A:17:99:86:38:56:8E:59:D5:20:B3:C4:49:B7:1E:F7:F5:F4:F4:E2
MD5 Fingerprint 25:DF:92:CA:78:BB:2E:D1:83:65:11:43:B4:7D:70:FA

In the **"Details**" tab, select the root certificate in the tree (PortSwigger CA).

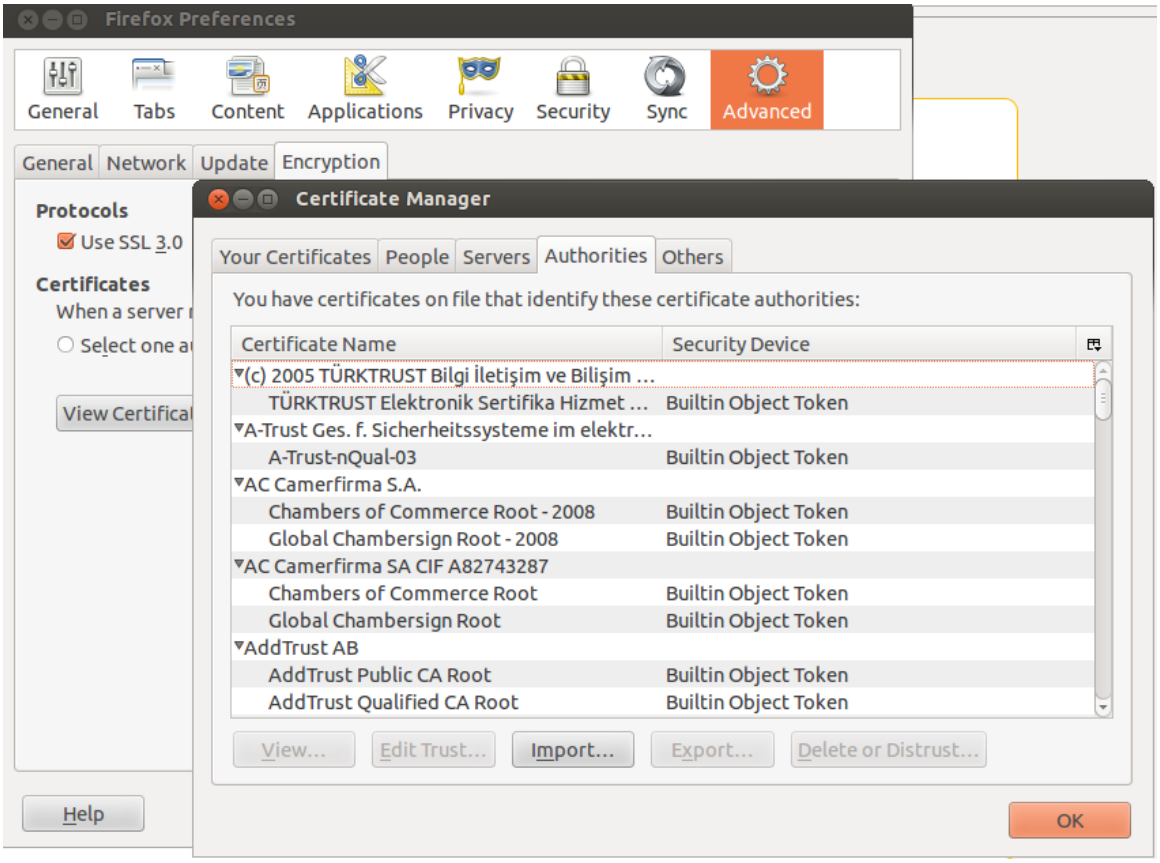


Click "Export" and save the certificate as "BurpCert" on the Desktop.

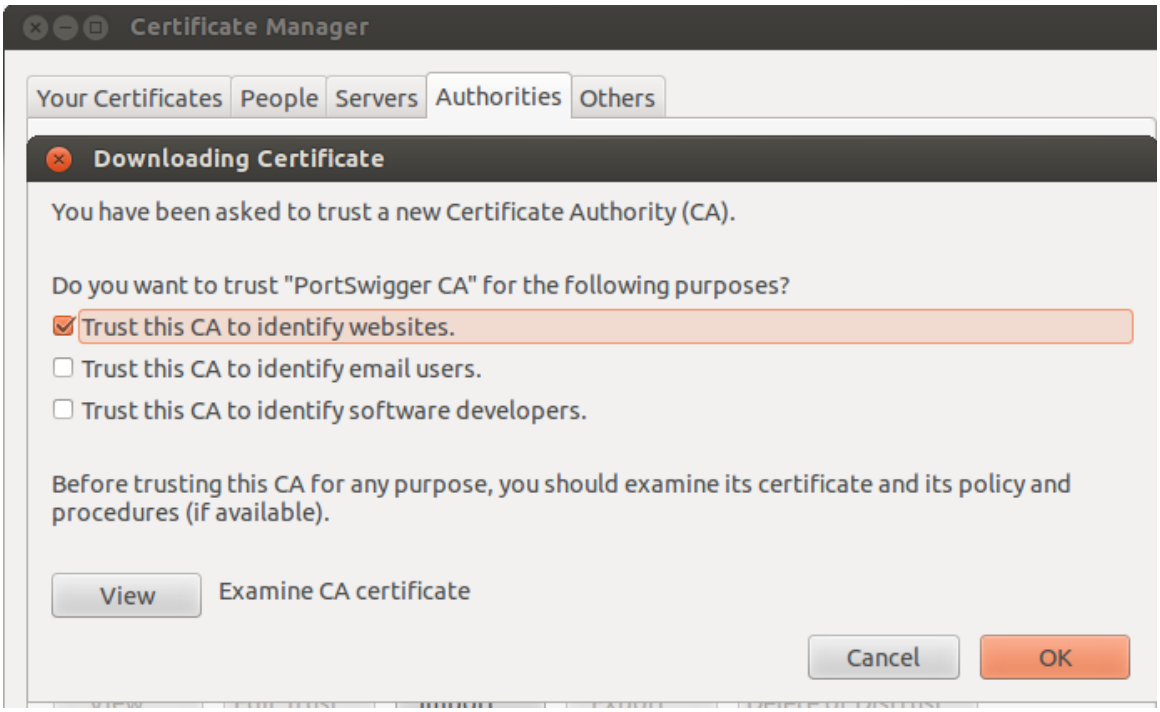


Close Certificate Viewer dialog and click “**Cancel**” on the “**Add Security Exception**” dialog
Go to Edit | Preferences
Click “**Advanced**” and go to “**Certificates**” tab
Click “**View Certificates**”

Click "**Import**" and select the certificate file that you previously saved.

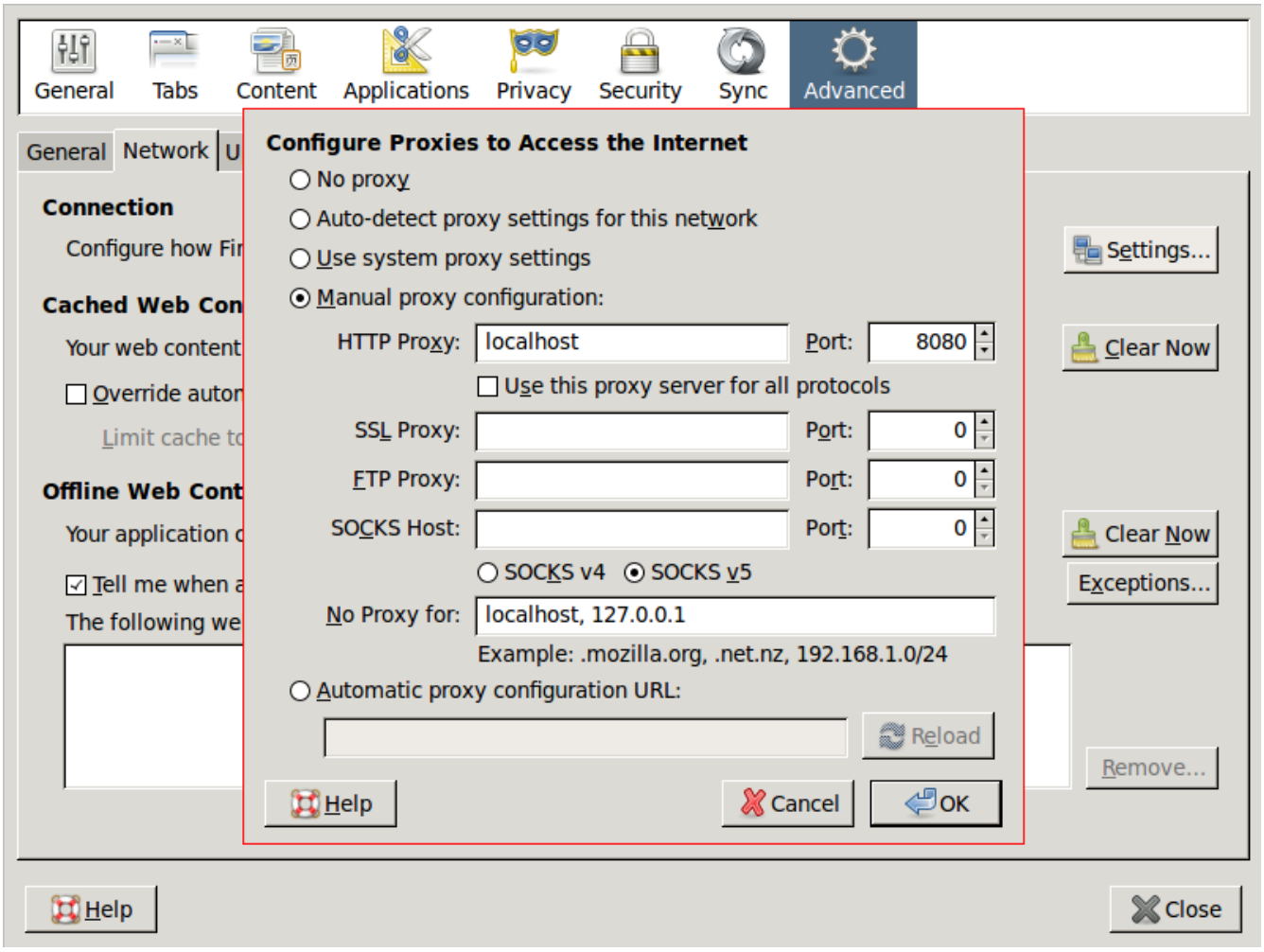


On the "**Downloading Certificate**" dialog, check the box "**Trust this CA to identify web sites**", and click "**OK**".

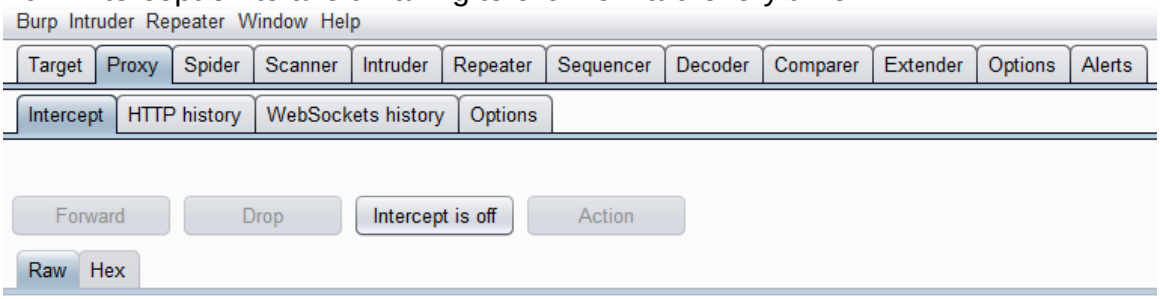


Close all dialogs and restart Firefox

Set Firefox Proxy

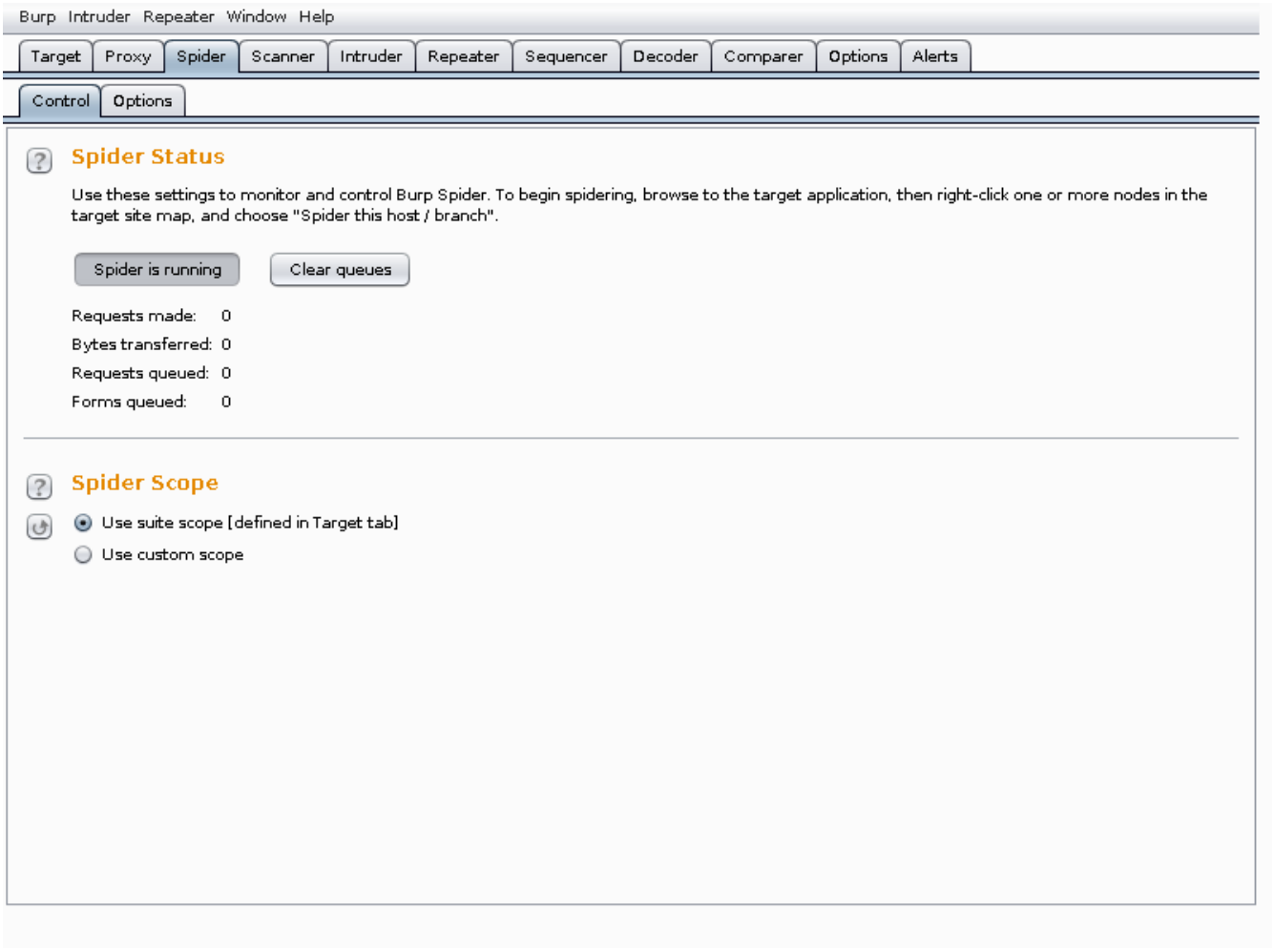


Turn intercept off to avoid having to click forward every time:



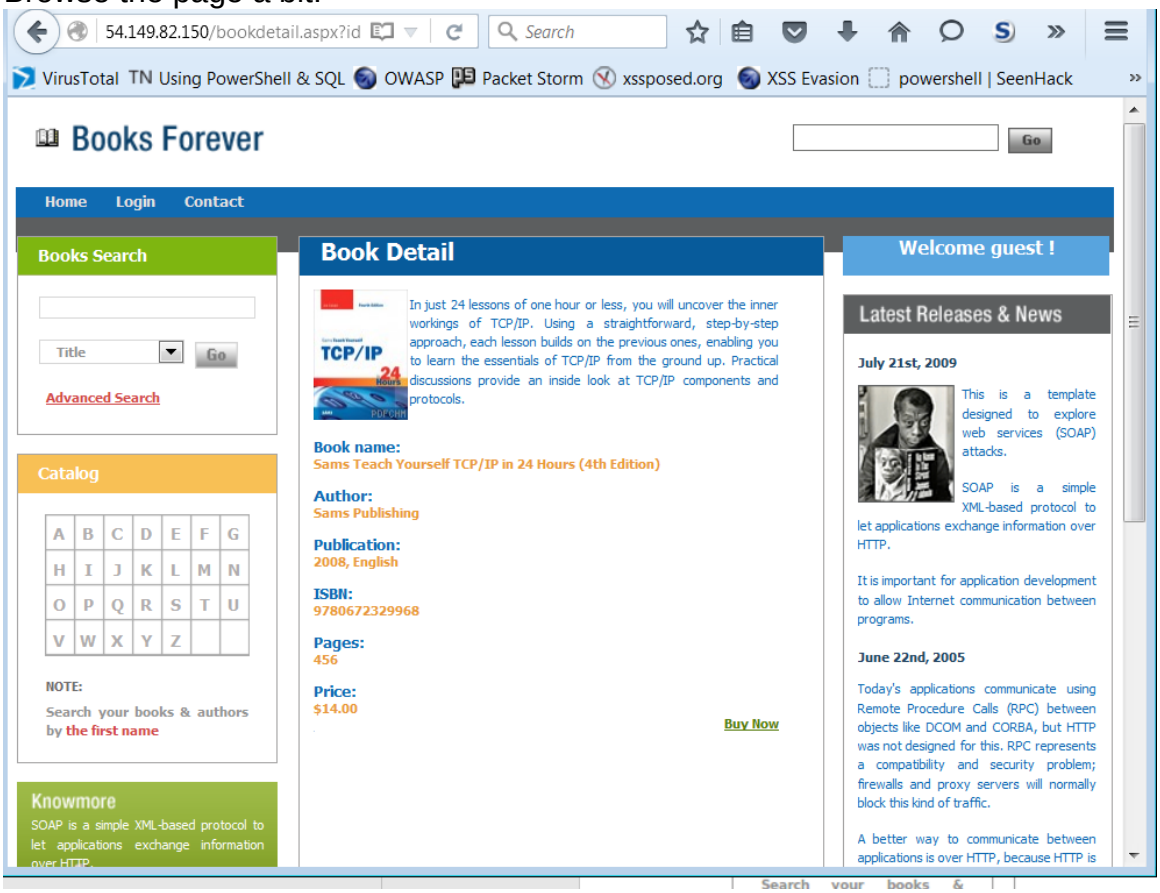
Lab 3: Running Burp Suite

Make sure Spider is running if we need it

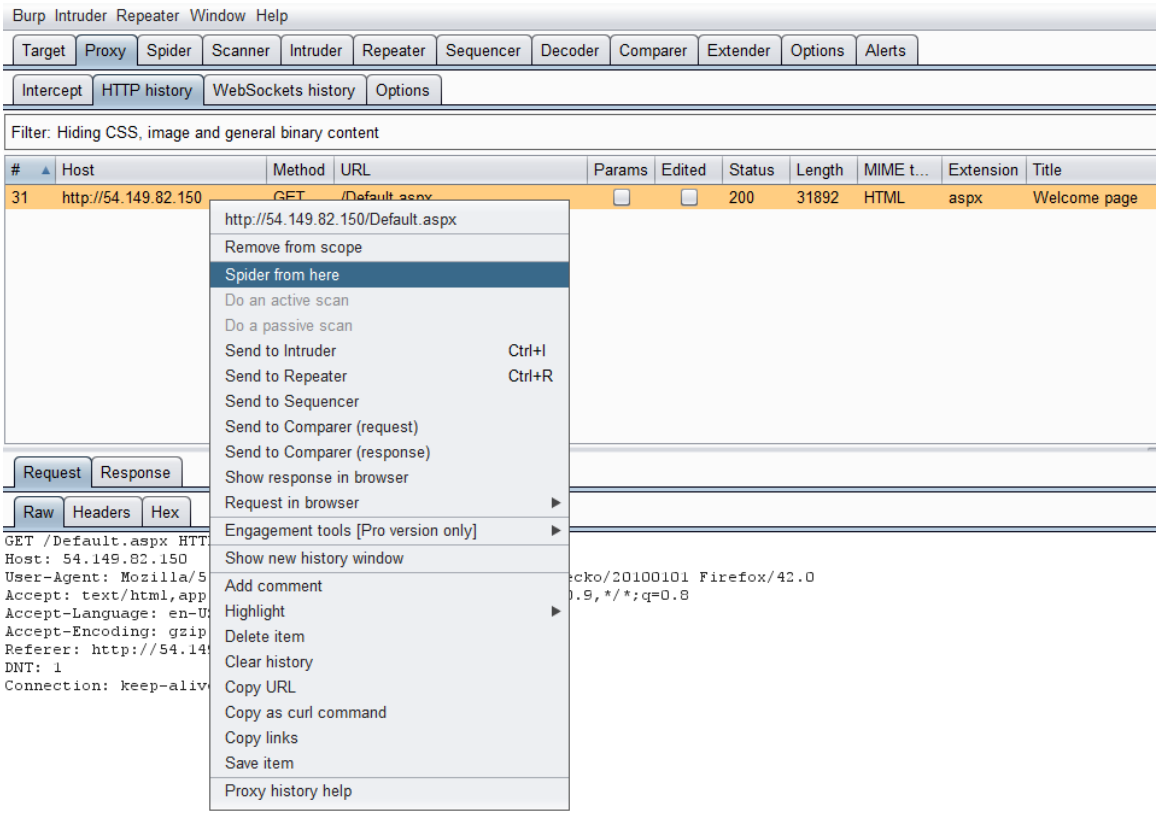


Navigate page to start information collection:

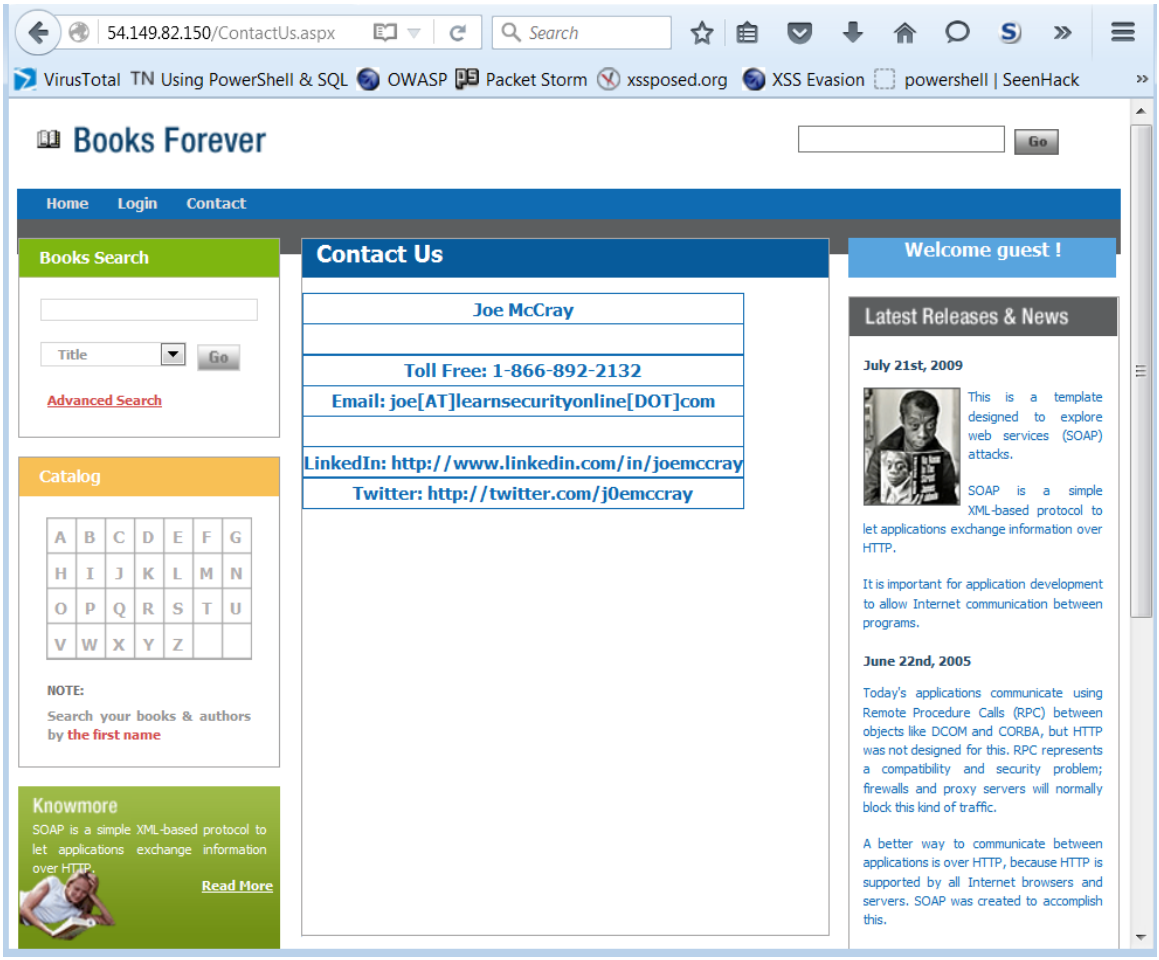
Browse the page a bit:



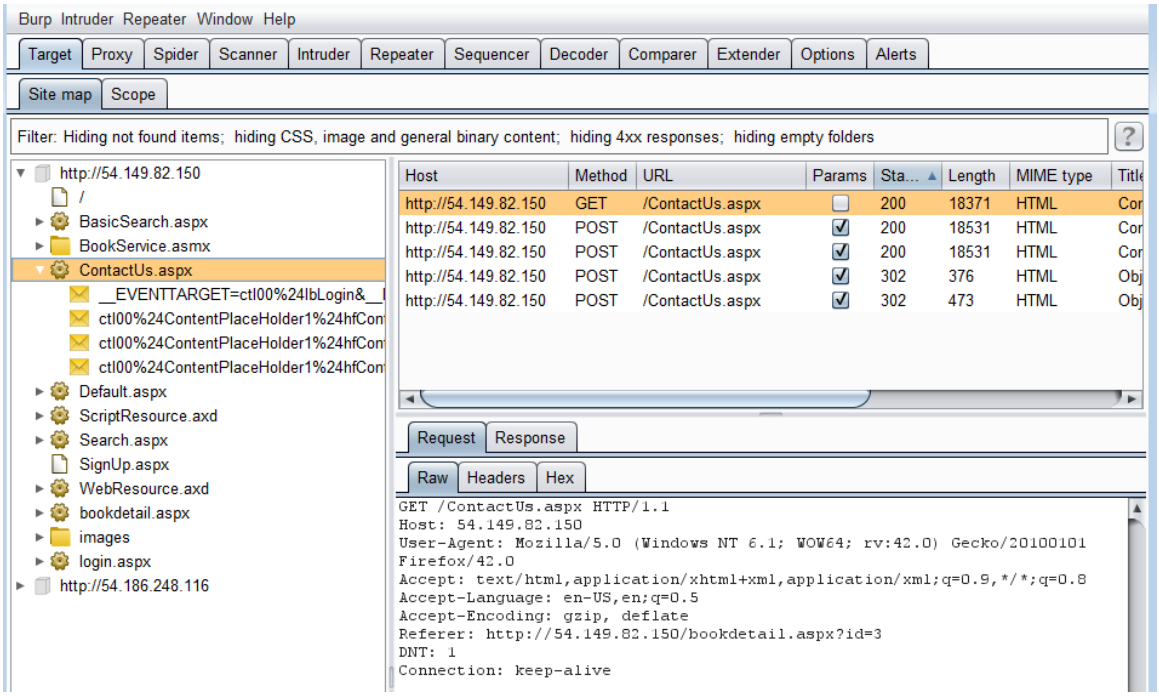
Spider the page (Spider from here):



Try some other pages:

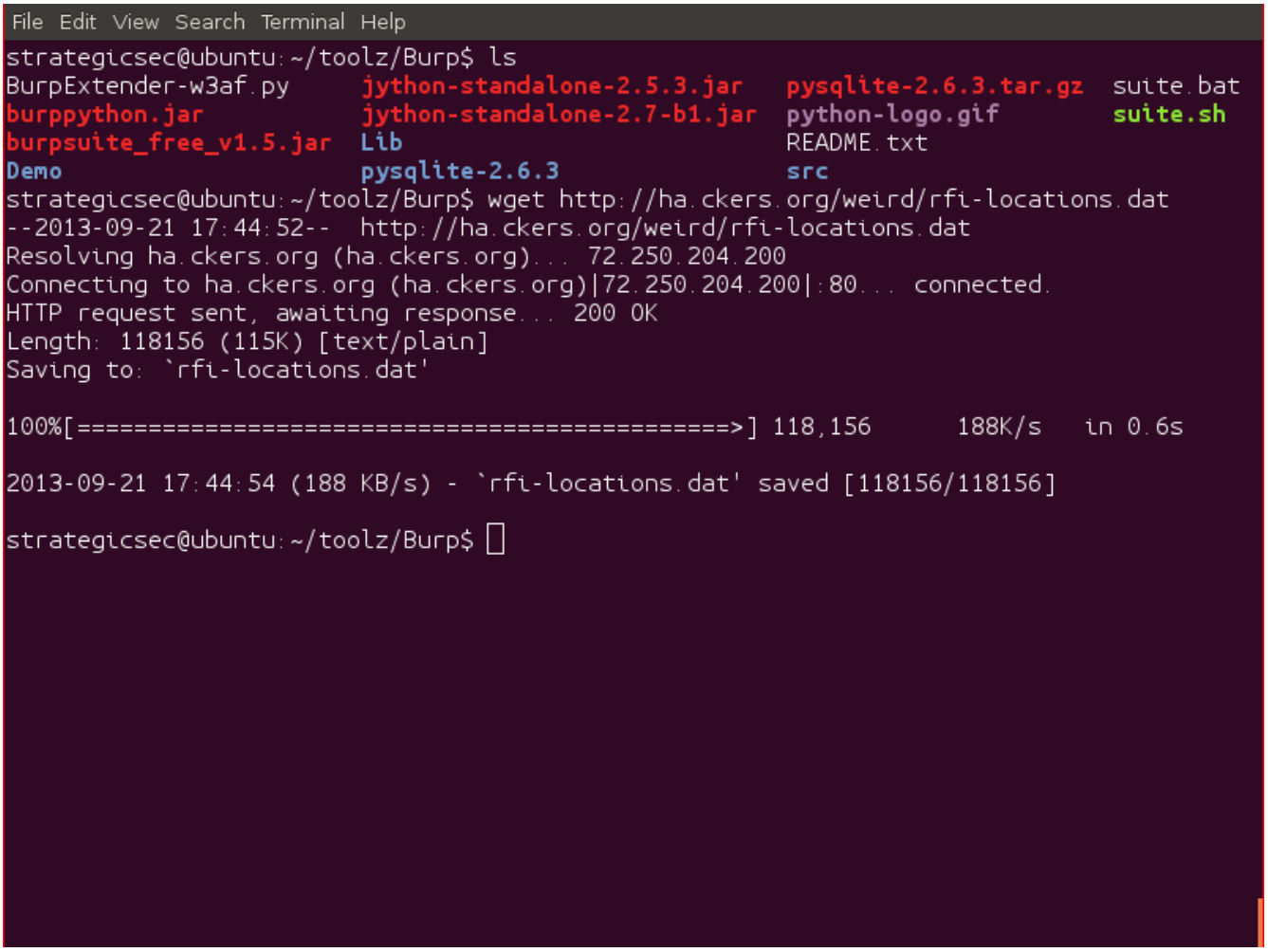


Results:


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Site map Scope
Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders
http://54.149.82.150
/
BasicSearch.aspx
BookService.asmx
ContactUs.aspx
__EVENTTARGET=ctl00%24lbLogin&__
ctl00%24ContentPlaceHolder1%24hfCon
ctl00%24ContentPlaceHolder1%24hfCon
ctl00%24ContentPlaceHolder1%24hfCon
Default.aspx
ScriptResource.axd
Search.aspx
SignUp.aspx
WebResource.axd
bookdetail.aspx
images
login.aspx
http://54.186.248.116
Host Method URL Params Sta... Length MIME type Titl
http://54.149.82.150 GET /ContactUs.aspx 200 18371 HTML Cor
http://54.149.82.150 POST /ContactUs.aspx 200 18531 HTML Cor
http://54.149.82.150 POST /ContactUs.aspx 200 18531 HTML Cor
http://54.149.82.150 POST /ContactUs.aspx 302 376 HTML Obj
http://54.149.82.150 POST /ContactUs.aspx 302 473 HTML Obj
Request Response
Raw Headers Hex
GET /ContactUs.aspx HTTP/1.1
Host: 54.149.82.150
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:42.0) Gecko/20100101
Firefox/42.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://54.149.82.150/bookdetail.aspx?id=3
DNT: 1
Connection: keep-alive

Lab 4: Adding RSnake RFI List

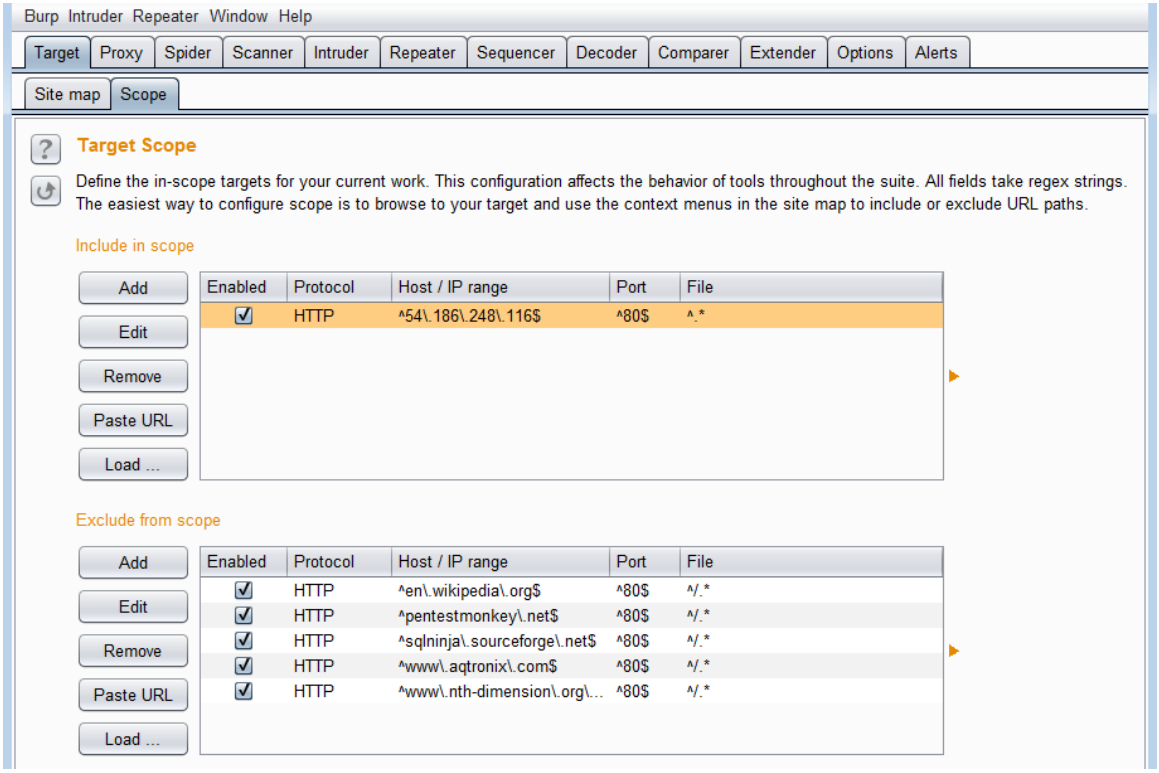
Get rsnake rfi list:



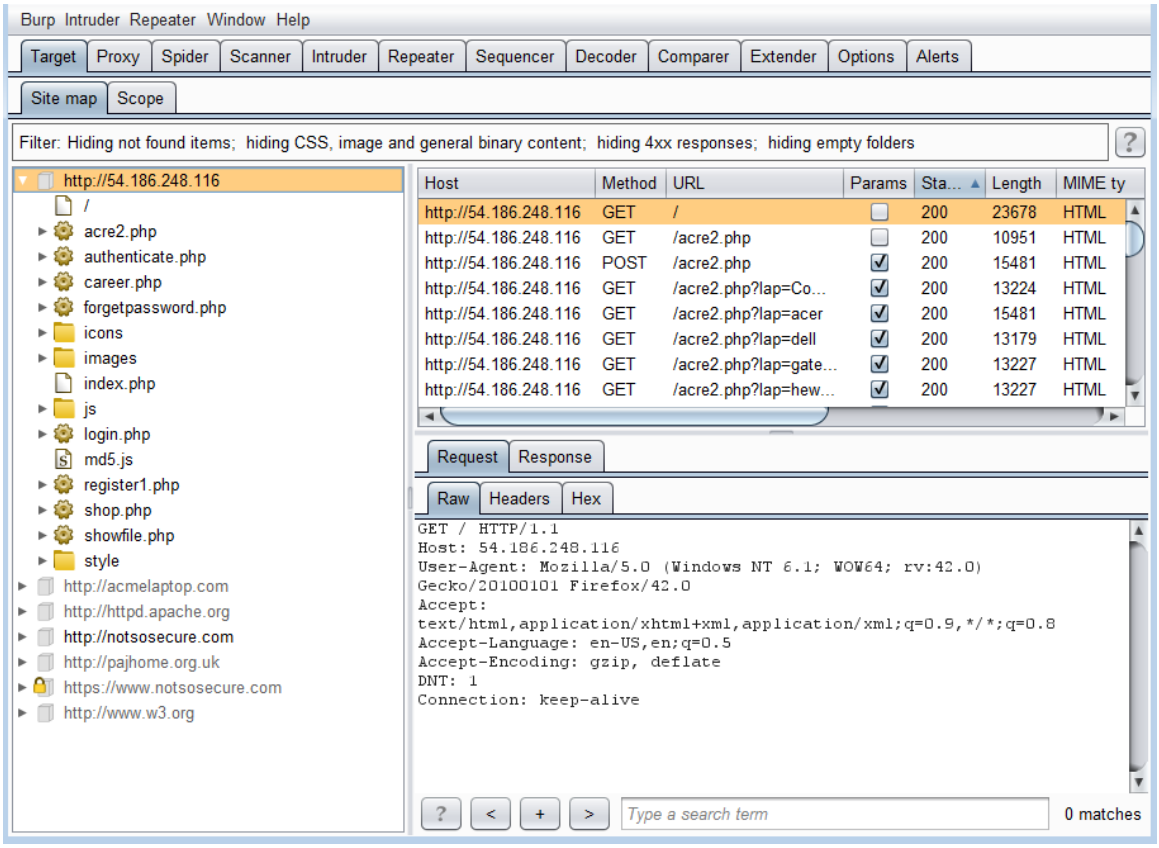
Process the list:

```
File  Edit  View  Search  Terminal  Help
strategicsec@ubuntu:~/toolz/Burp$ ls
BurpExtender-w3af.py     jython-standalone-2.5.3.jar   pysqlite-2.6.3.tar.gz  suite.bat
burppython.jar           jython-standalone-2.7-b1.jar  python-logo.gif        suite.sh
burpsuite_free_v1.5.jar  Lib                           README.txt
Demo                     pysqlite-2.6.3                src
strategicsec@ubuntu:~/toolz/Burp$ wget http://ha.ckers.org/weird/rfi-locations.dat
--2013-09-21 17:44:52--  http://ha.ckers.org/weird/rfi-locations.dat
Resolving ha.ckers.org (ha.ckers.org)... 72.250.204.200
Connecting to ha.ckers.org (ha.ckers.org)|72.250.204.200|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 118156 (115K) [text/plain]
Saving to: `rfi-locations.dat'

100%[==============================================>] 118,156      188K/s   in 0.6s

2013-09-21 17:44:54 (188 KB/s) - `rfi-locations.dat' saved [118156/118156]

strategicsec@ubuntu:~/toolz/Burp$ cat rfi-locations.dat | grep -v "^#" | awk -F '?' '{pr
int $1}' | sort -u > rsnake_list.txt
strategicsec@ubuntu:~/toolz/Burp$
```

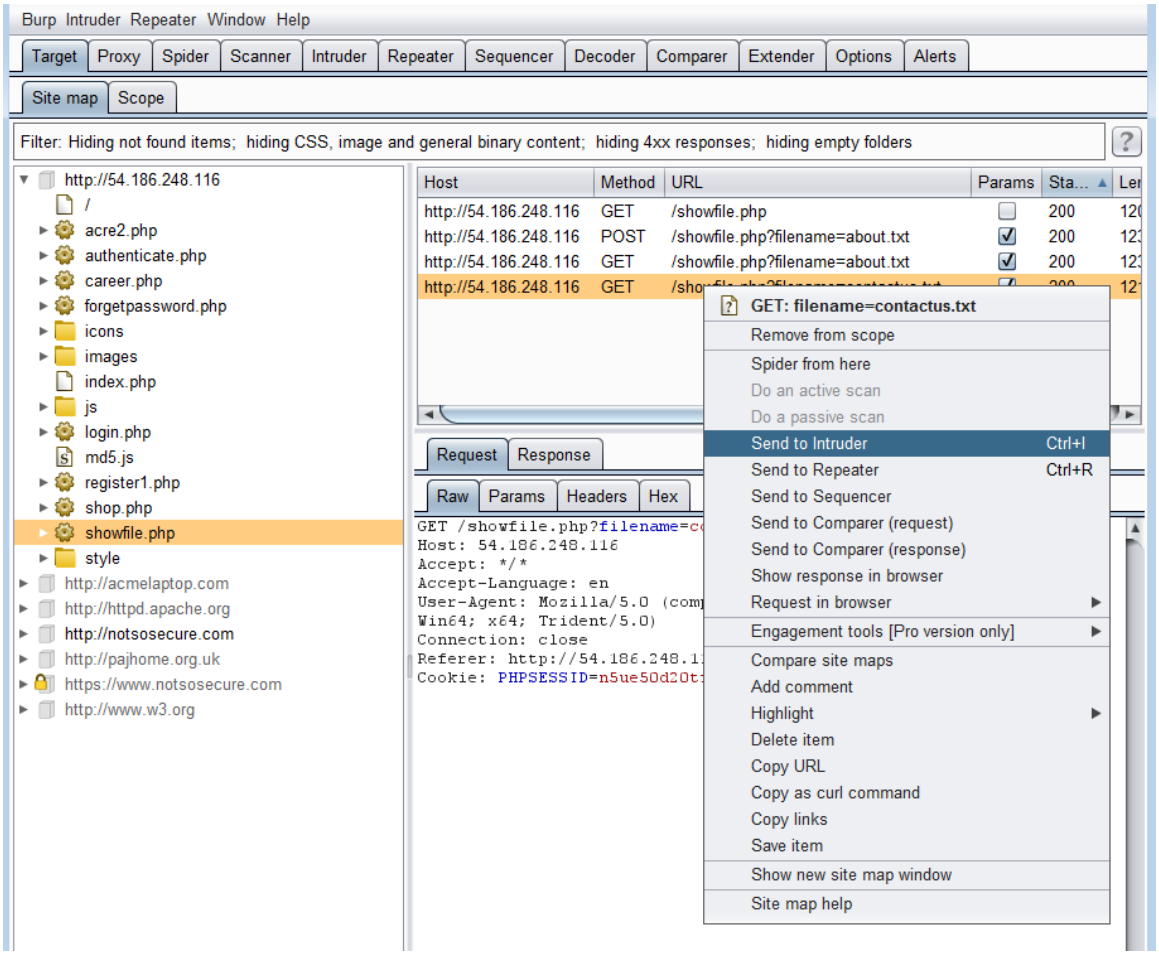
Set target to http://54.186.248.116 it uses PHP:



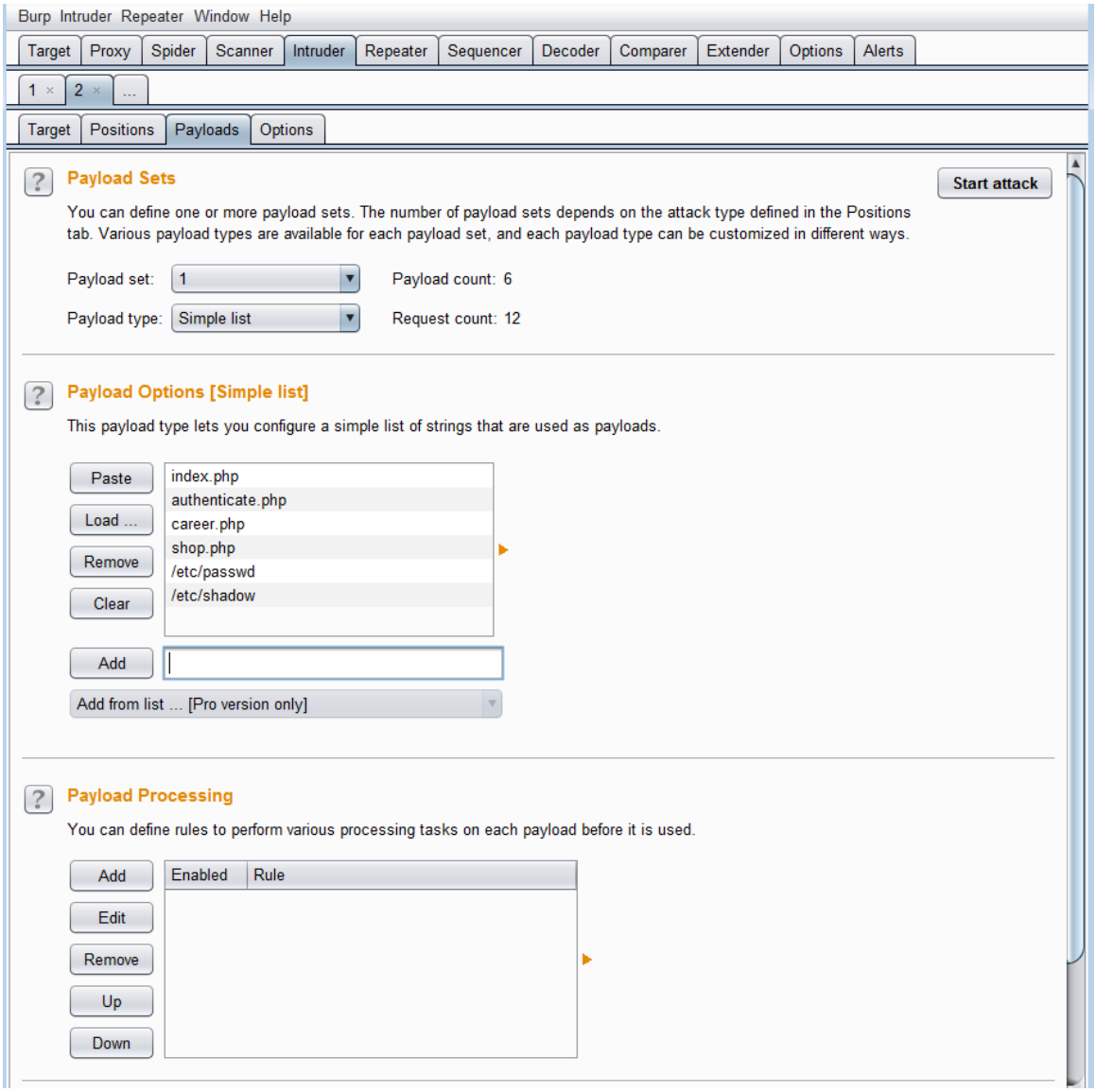
Spider it first:



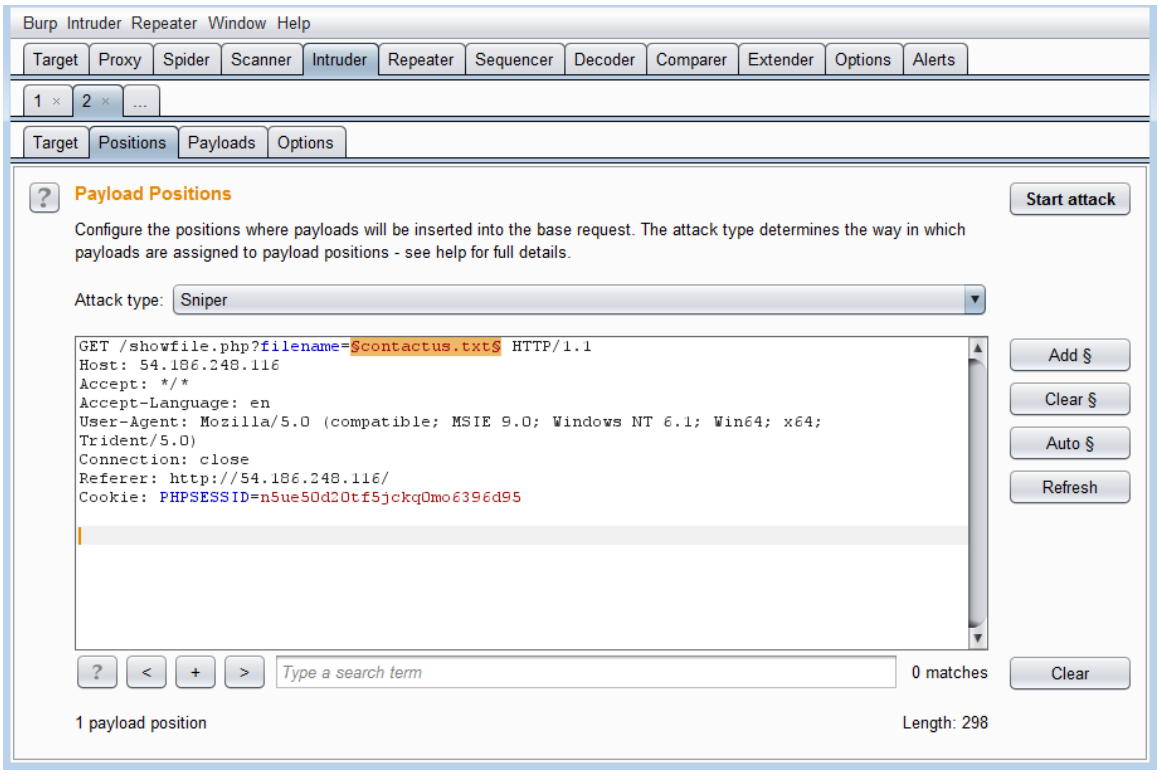
See rfi potential in showfile.php so send it to Intruder module:



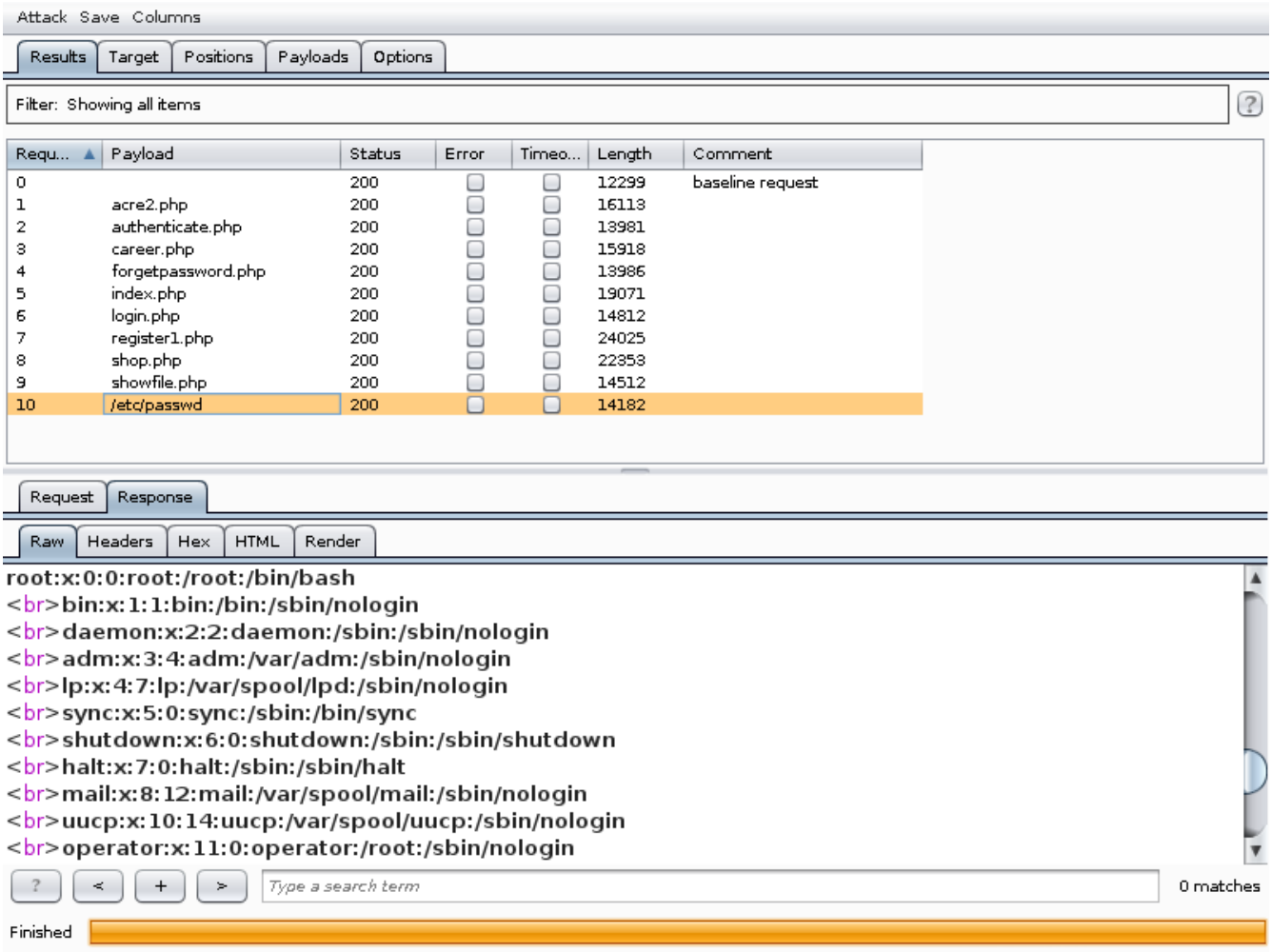
Add manual payloads (rsnake's are too big):



Select payload position:



Attack:


Attack Save Columns
Results Target Positions Payloads Options
Filter: Showing all items
Requ... Payload Status Error Timeo... Length Comment
0 200 12299 baseline request
1 acre2.php 200 16113
2 authenticate.php 200 13981
3 career.php 200 15918
4 forgetpassword.php 200 13986
5 index.php 200 19071
6 login.php 200 14812
7 register1.php 200 24025
8 shop.php 200 22353
9 showfile.php 200 14512
10 /etc/passwd 200 14182
Request Response
Raw Headers Hex HTML Render
root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

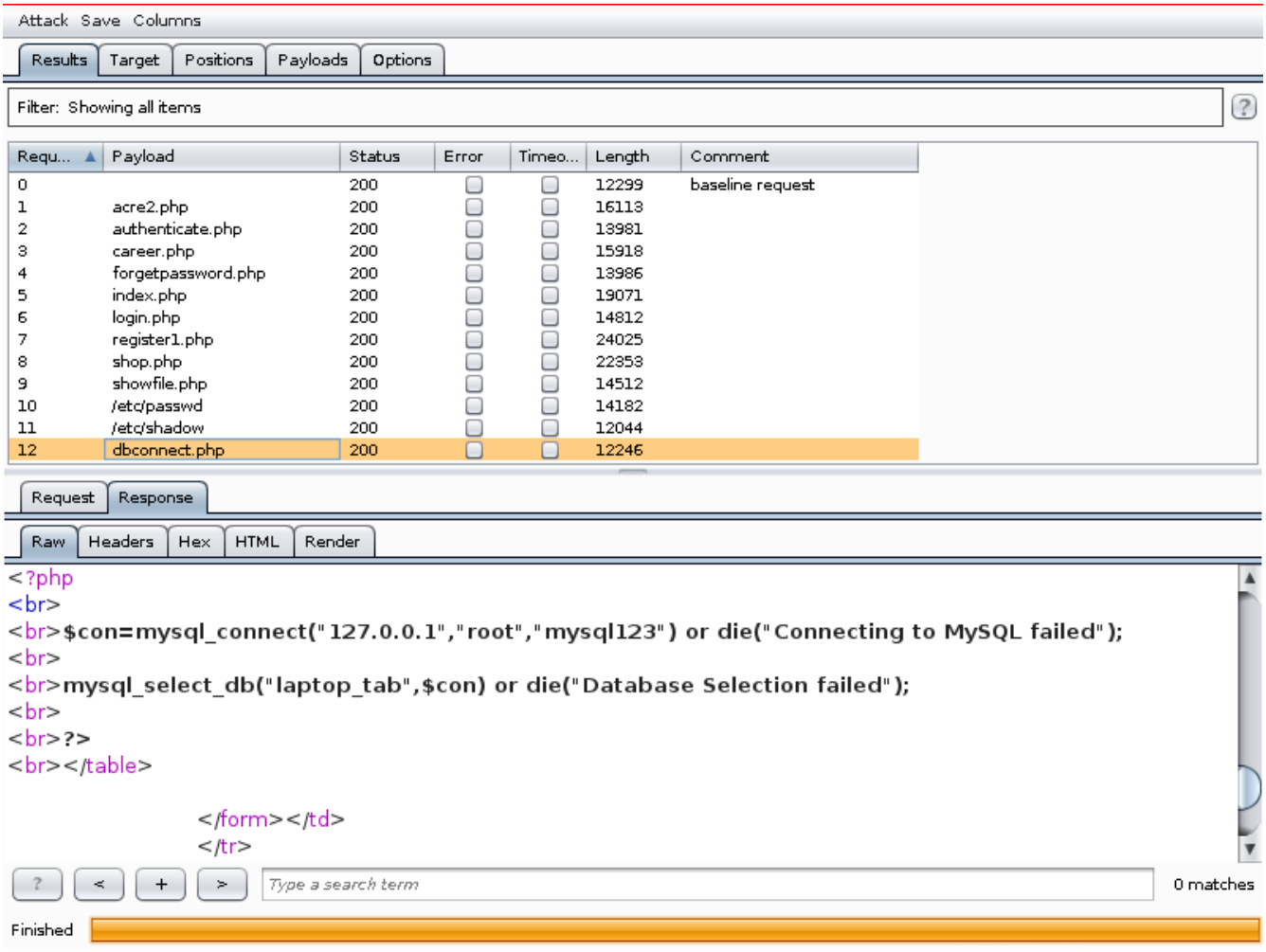
halt:x:7:0:halt:/sbin:/sbin/halt

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin
Type a search term
0 matches
Finished

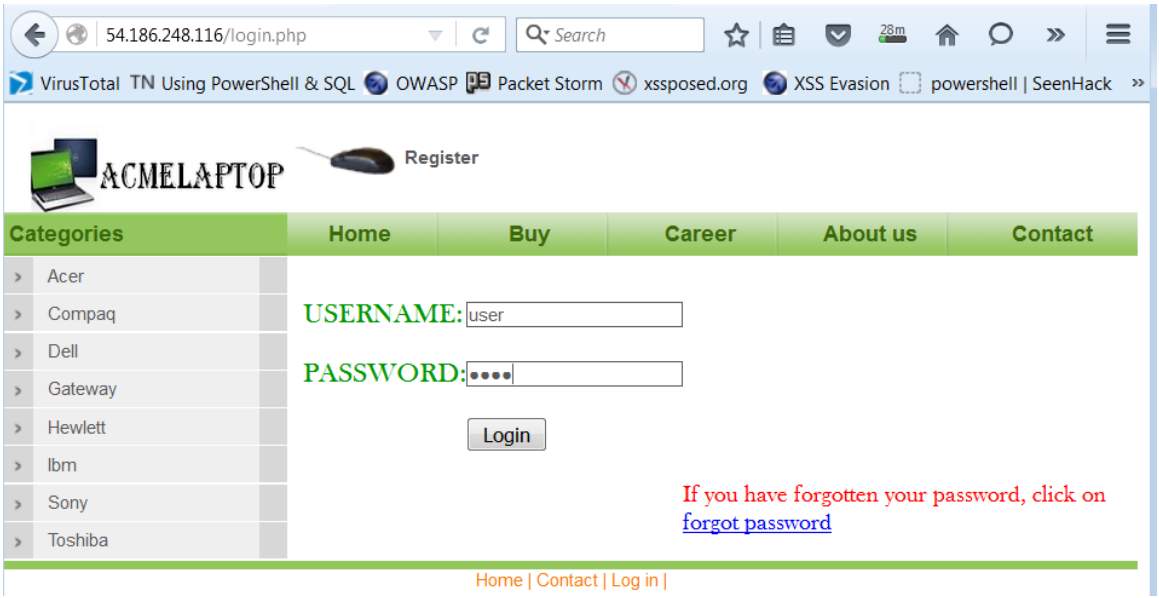
Get MySQL password from dbconnect.php:



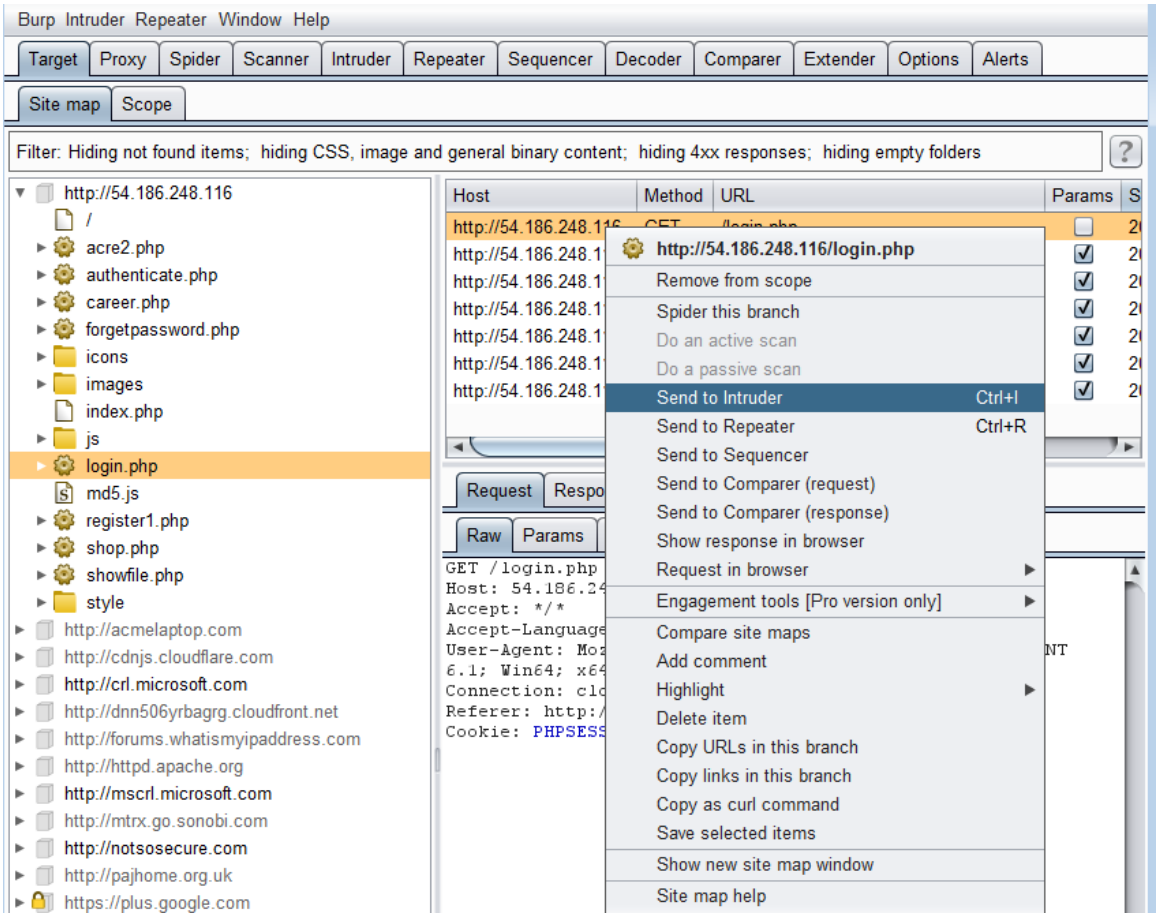
Root?!, check ssh:

```
File  Edit  View  Search  Terminal  Help
ssh strategicsec@ubuntu:~$ ssh 10.10.10.107
The authenticity of host '10.10.10.107 (10.10.10.107)' can't be established.
RSA key fingerprint is 60:51:b1:55:a8:4c:f0:88:50:a1:2a:e0:46:5f:87:06.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.107' (RSA) to the list of known hosts.
strategicsec@10.10.10.107's password:
Permission denied, please try again.
strategicsec@10.10.10.107's password:

strategicsec@ubuntu:~$ ssh root@10.10.10.107
root@10.10.10.107's password:
Last login: Sun Jun 28 14:38:23 2009 from 192.168.110.1
[root@fedora10server ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog  src
[root@fedora10server ~]# whoami
root
[root@fedora10server ~]#
```

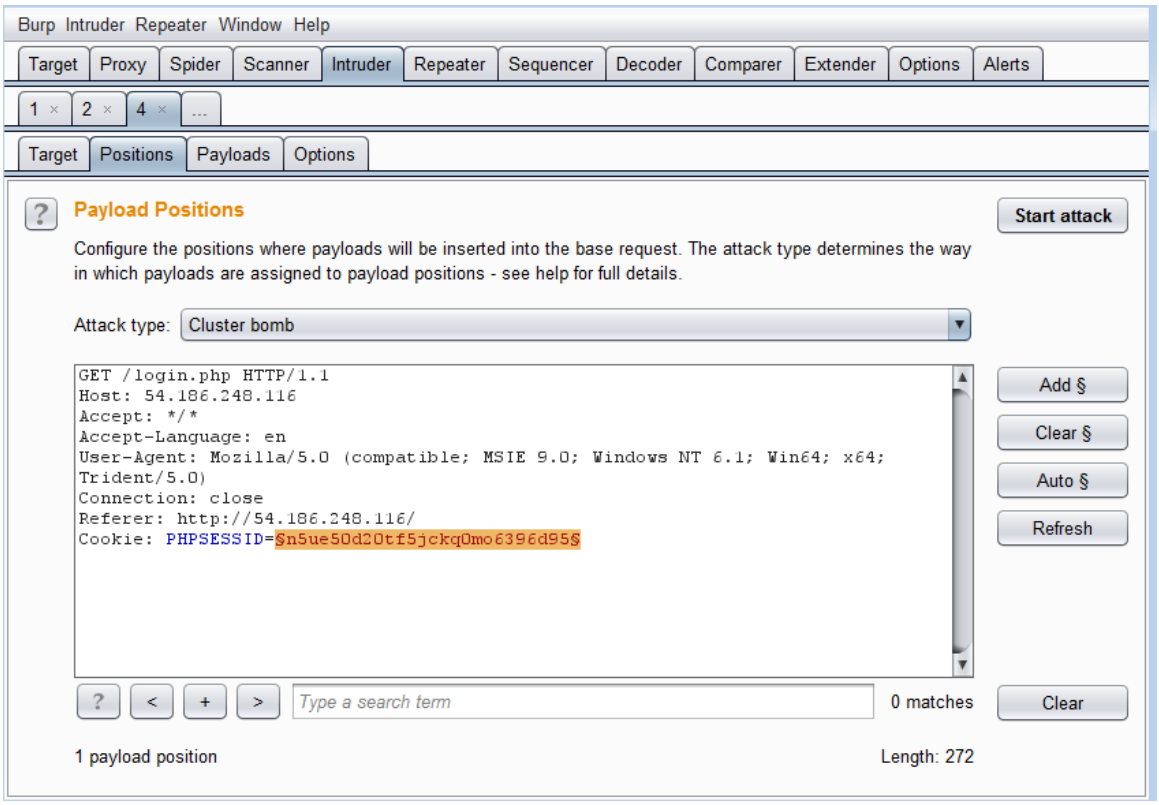
Goto login page:



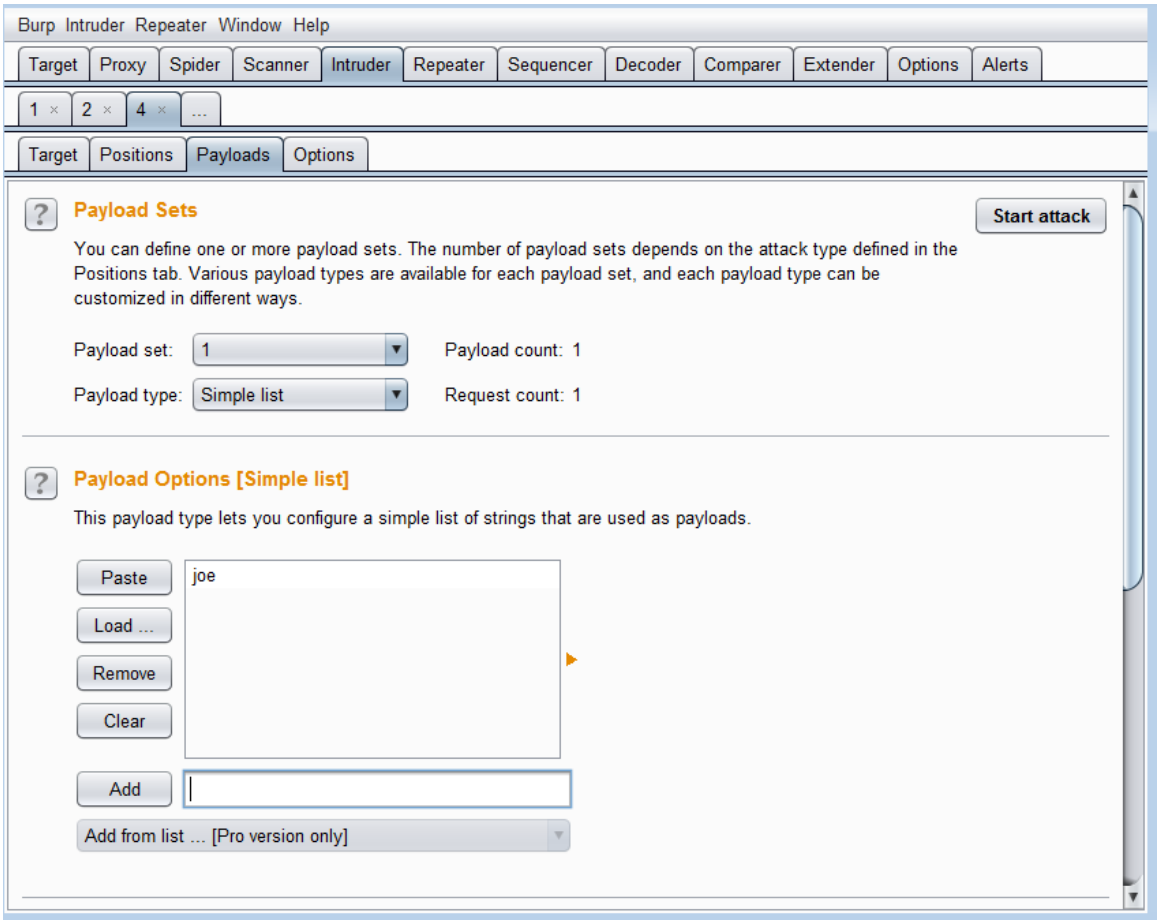
Send the page to the Intruder module:



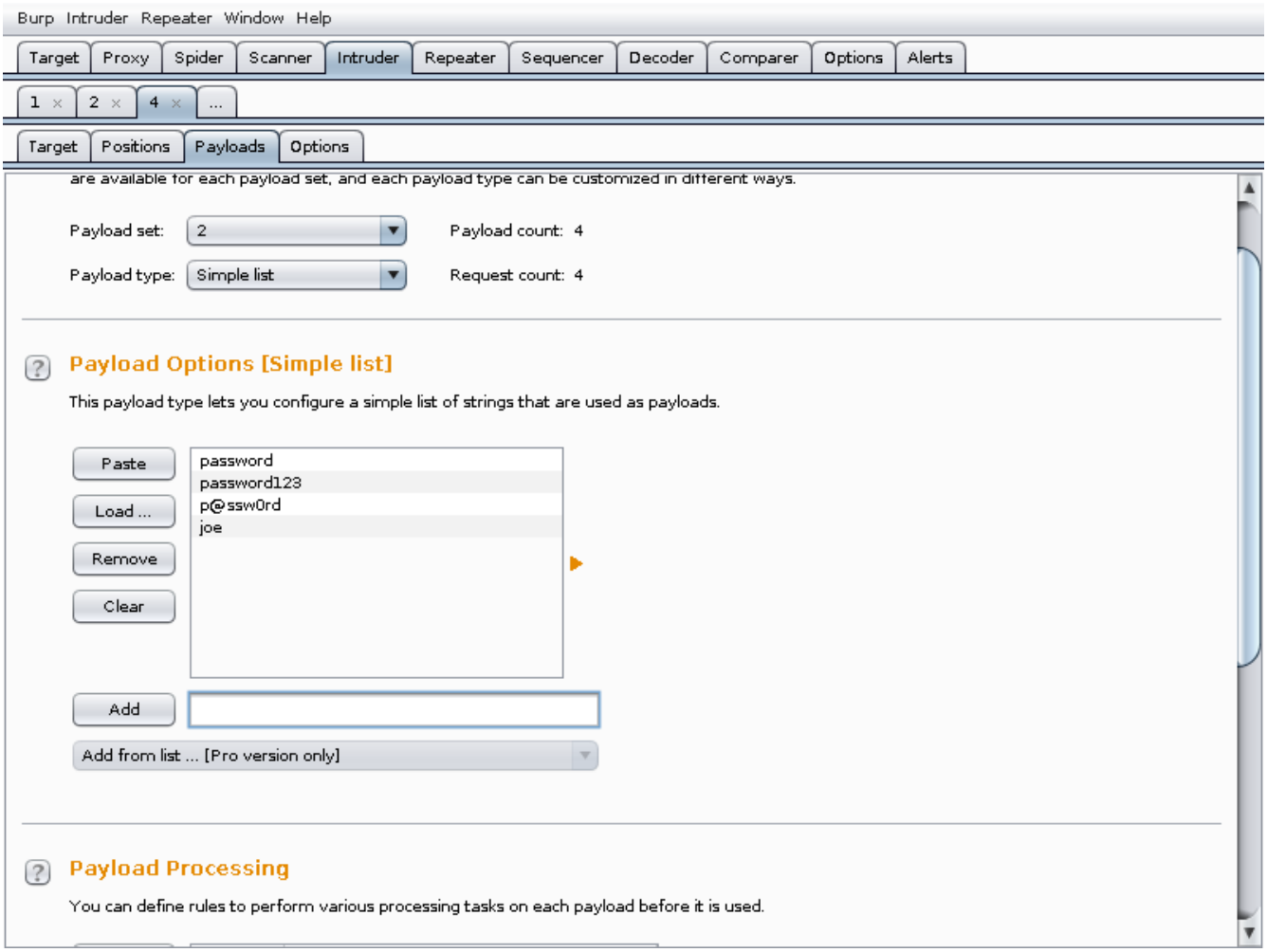
Set attack to cluster bomb and set positions:



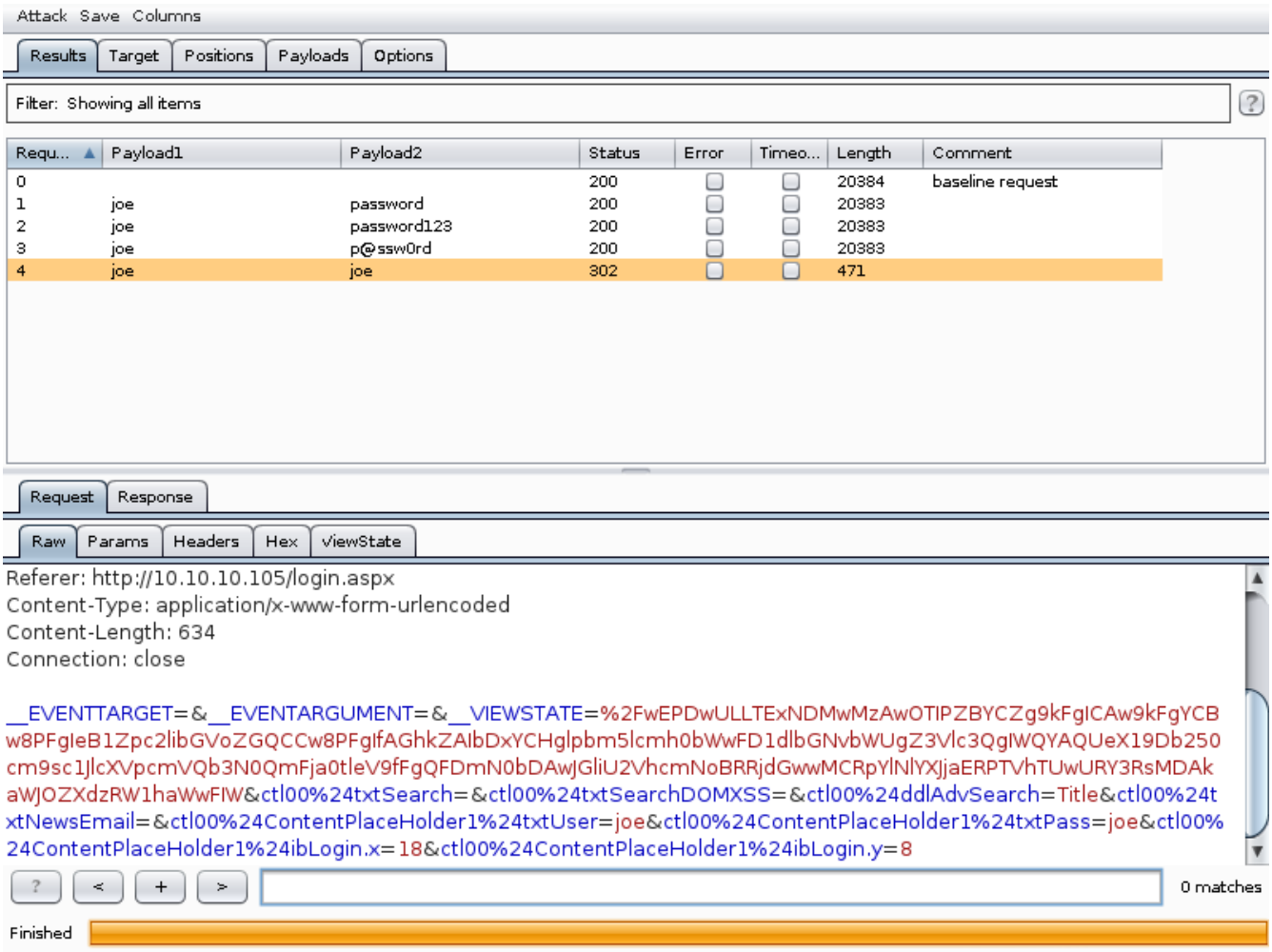
We know of user joe existence on 10.10.10.107 so let's put joe as username:



Put a custom list in the second:



Found password:


Attack Save Columns
Results Target Positions Payloads Options
Filter: Showing all items
Requ... Payload1 Payload2 Status Error Timeo... Length Comment
0 200 20384 baseline request
1 joe password 200 20383
2 joe password123 200 20383
3 joe p@ssw0rd 200 20383
4 joe joe 302 471
Request Response
Raw Params Headers Hex ViewState
Referer: http://10.10.10.105/login.aspx
Content-Type: application/x-www-form-urlencoded
Content-Length: 634
Connection: close
__EVENTTARGET=&__EVENTARGUMENT=&__VIEWSTATE=%2FwEPDwULLTExNDMwMzAwOTlPZBYCZg9kFgICAw9kFgYCBw8PFgIeB1Zpc2libGVoZGQCCw8PFgIfAGhkZAIbDxYCHglpbm5lcmh0bWwFD1dlbGNvbWUgZ3Vlc3QgIWQYAQUeX19Db250cm9sc1JlcXVpcmVQb3N0QmFja0tleV9fFgQFDmN0bDAwJGliU2VhcmNoBRRjdGwwMCRpYlNlYXJjaERPTVhTUwURY3RsMDAkaWJOZXdzRW1haWwFIW&ctl00%24txtSearch=&ctl00%24txtSearchDOMXSS=&ctl00%24ddlAdvSearch=Title&ctl00%24txtNewsEmail=&ctl00%24ContentPlaceHolder1%24txtUser=joe&ctl00%24ContentPlaceHolder1%24txtPass=joe&ctl00%24ContentPlaceHolder1%24ibLogin.x=18&ctl00%24ContentPlaceHolder1%24ibLogin.y=8
? < + >
0 matches
Finished

Try joe:joe:

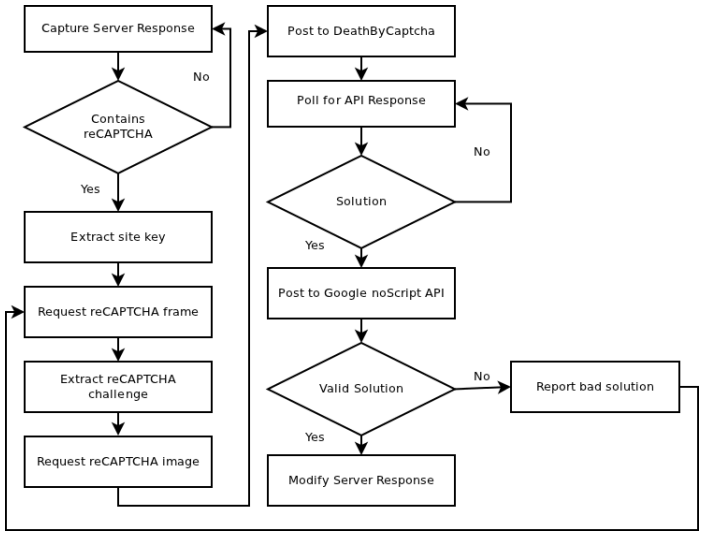

54.186.248.116/index.php
Search
VirusTotal
Using PowerShell & SQL
OWASP
Packet Storm
xssposed.org
XSS Evasion
powershell | SeenHack
ACMELAPTOP
Hey ! Joe Wel-Come
Categories
Home
Buy
Career
About us
Contact
Acer
Compaq
Dell
Gateway
Hewlett
Ibm
Sony
Toshiba
Eee PC S101
Acme is an online website where you can sell your laptop or services
Best Featured Products
Acer
acer | Intel Core2 Duo 2.0GHz / Glossy 15.6" WXGA high-definition widescreen display / 2GB DDR2 SDRA
Compaq
Compaq | Intel Core 2 Duo 2.1GHz / 4GB memory / 320GB hard drive / DVD RW/DVD-RAM/DVD+R Double Layer

Lab 5: Dealing with reCAPTCHA

Using Burp Extender to automatically solve reCAPTCHA

There are a number of CAPTCHA solving businesses, which sell the service of human-generated solutions to CAPTCHA images. Phil at idontplaydarts.com built this particular Burp extension to interface with the service at DeathByCaptcha.com, to automatically solve reCAPTCHA challenges and pass the solutions to the tools within Burp Suite. Currently, this service charges $1.39 for 1000 CAPTCHA solutions. At the time of this writing, the stats on the DeathByCaptcha.com website are showing solution times of about 9 seconds, and an accuracy rate of 95.7%. The author of the extension chose to focus on reCAPTCHA with this script due to its widespread usage, as well as the fact that solutions received from the solving service can be validated via Google's servers prior to releasing it back to the target.

Essentially, this extension takes incoming HTTP responses, and scans through them for the reCAPTCHA script. If it finds a reCAPTCHA challenge, it pulls from it the site key which it then uses to obtain an iframe with a link to the CAPTCHA image. It uses that link to grab the reCAPTCHA JPEG image and send it to DeathByCaptcha, and also pulls the reCAPTCHA challenge field from the HTML in the iframe. Once a solution is received, it's posted to the iframe location. The extension then takes the reply from that post, extracts the challenge response, and inserts it into the initial HTTP response. At that point, the initial HTTP response in Burp Suite has in it the challenge and response codes. The author of the script provides a flowchart that illustrates the process[1]:



The plugin can be downloaded from the idontplaydarts.com site.[2] Run these commands to compile the extension:

Javac.exe BurpExtender.java
Jar.exe –cf BurpExtender.jar BurpExtender.class

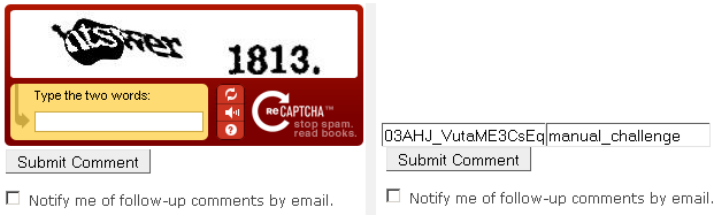
[1] https://www.idontplaydarts.com/wp-content/uploads/2012/01/HSRA.png
[2] https://www.idontplaydarts.com/wp-content/uploads/2012/01/BurpExtender-reCAPTCHA.zip

This will create the burpExtender.jar in the current directory.

To run the extension, you need to pass it your username and password for the DeathByCaptcha service. Note that this will not run without access to an account there, as it relies on that service's API. With the extension in the same directory as Burp Suite, use the following to launch it:

Java –Xmx512m –classpath "" burp.StartBurp "username_here" "password_here"*

With this extension running, the Burp Proxy will now replace reCAPTCHA instances with challenge/response input boxes. Note that the page will not load while the extension is utilizing the CAPTCHA solving API (an average of 9 seconds according to the service's website at this time). The following screenshot is from the author's website, showing a before and after image of a reCAPTCHA solved with this extension.[3]



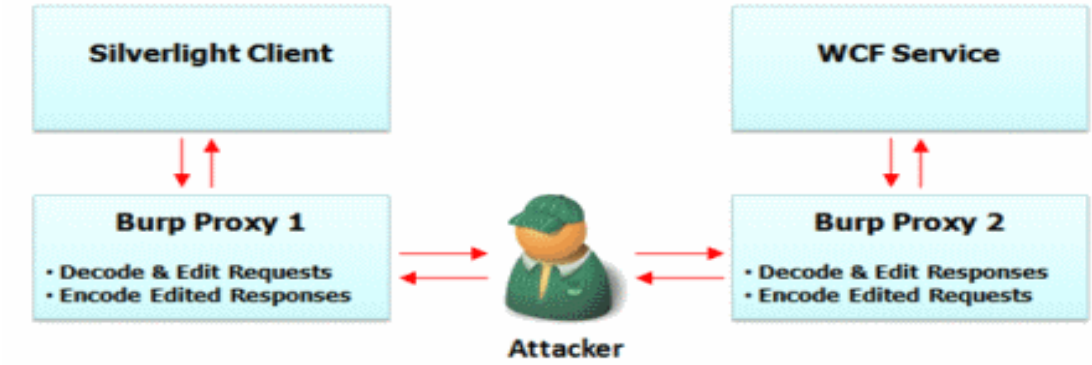
[3] https://www.idontplaydarts.com/wp-content/uploads/2012/01/beforeafter.png

Lab 6: Working with NBFS-Encoded WCF Communications in BurpSuite

Silverlight web applications that utilize WCF (Windows Communication Foundation) APIs, often encode their communications with the endpoint using ".NET Binary Format for SOAP" or NBFS[4]. This presents a challenge to an attacker attempting to modify requests and/or responses in real-time, as the message recipient will throw an exception if the message is not encoded correctly. This encoding format is not built-in to Burp Suite, so working with communication to and from these types of applications can be difficult.

Over the years, several solutions have been developed to work with NBFS-encoded communications. Some of the earliest examples were NBFS inspectors written for the Fiddler proxy. Brian Holyfield took one of these inspectors (built by Richard Berg) and used some of the code in it to build a BurpExtender plugin that was able to both encode and decode NBFS communications. The plugin is discussed in more detail on the GDS Security Blog[5].

This solution allows BurpSuite to view and edit the data in WCF requests and responses. Essentially what it does, is examine data as it comes in for the header that indicates it is NBFS. If that header is present, it passes the data to a library for decoding, which results in BurpSuite receiving the plain-text. The plugin also examines outgoing requests, and if it needs to be encoded it passes it to the



library for encoding on the way out.

4 http://msdn.microsoft.com/en-us/library/cc219175.aspx
5 http://blog.gdssecurity.com/labs/2009/11/19/wcf-binary-soap-plug-in-for-burp.html

It does have some quirks due to the timing of the events the plugin uses. Essentially, the processing takes place prior to the user being able to edit the response. While one instance of Burp is sufficient for editing requests, if one wants to edit NBFS-encoded responses, it is necessary to chain together two instances of the BurpSuite Proxy. This can be accomplished using the "Upstream Proxy Servers" setting in Options>Connections in Burp Suite. The entry on the GDS Security Blog provides the following diagram and explanation of this process:

"The purpose of chaining two proxies together is as follows:

- The first instance handles decoding requests, intercepting (and editing) requests, and re-encoding edited responses. Set this instance to intercept REQUESTS only (not responses) and to use the 2nd proxy as the next hop.
- The second instance handles re-encoding edited requests, decoding responses, and intercepting (and editing) responses. Set this instance to intercept RESPONSES only (not requests). "

The plugin can be downloaded here[6]. Place the BurpExtender.jar in the same directory as Burp Suite, and run it with:

```
java -Xmx512m -classpath BurpExtender.jar:./burpsuite_free_v1.6.31.jar
burp.StartBurp
```

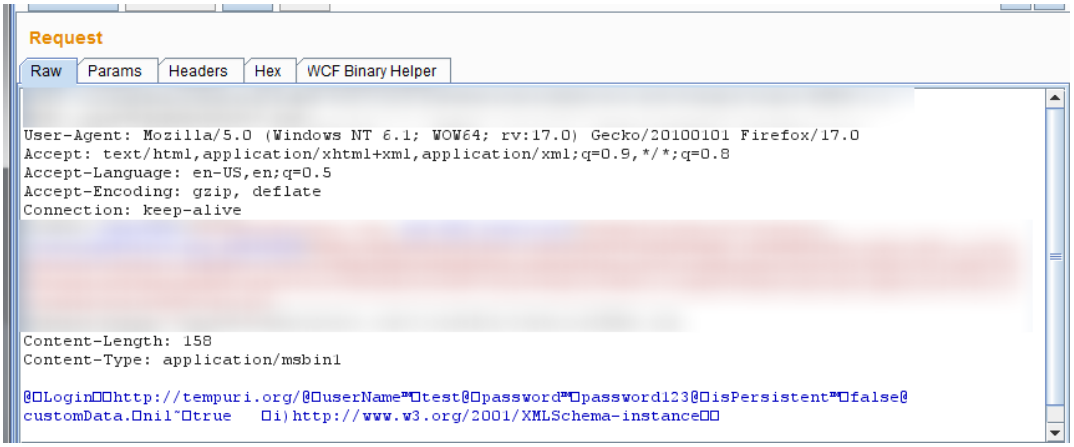
Another option...

With the new BurpExtender Framework API (available currently only in the Pro version of Burp Suite), there is a newer and slightly more straightforward option. Nick Coblentz took Brian Holyfield's code, and modified it to create a Python extension for BurpSuite. It Coblentz's Python extension can be downloaded at the GitHub page here[7].

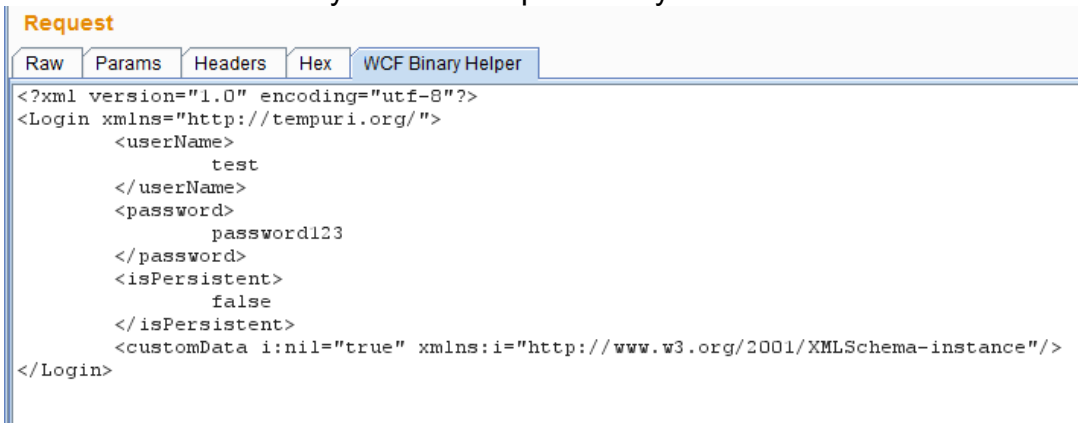
Below are some screenshots of this plugin in action, taken from the Security PS Blog[8] post about the plugin.

WCF Binary SOAP Request:

6 https://github.com/GDSSecurity/WCF-Binary-SOAP-Plug-In/tree/master/burp_wcf_plugin
7 https://gist.github.com/sekhmetn/4420532
8 http://blog.securityps.com/2013/02/burp-suite-plugin-view-and-modify-wcf.html

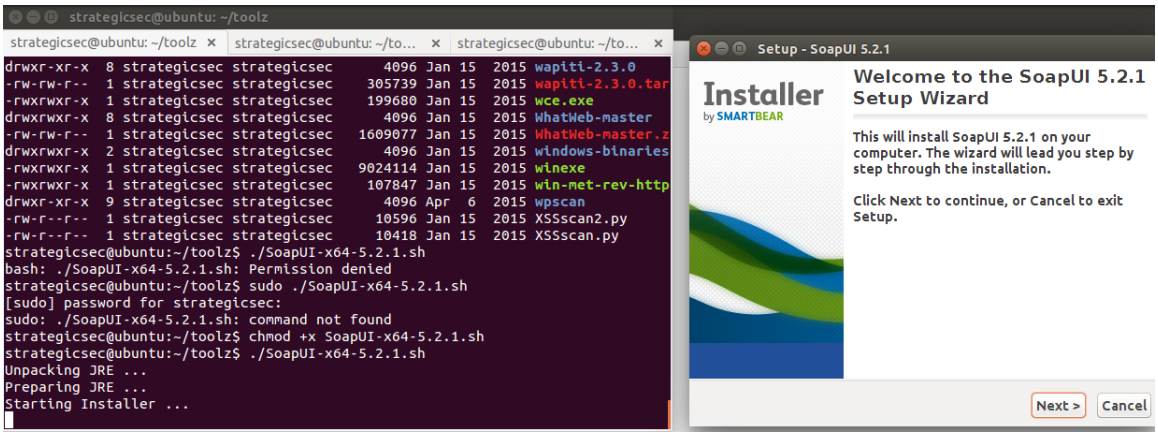


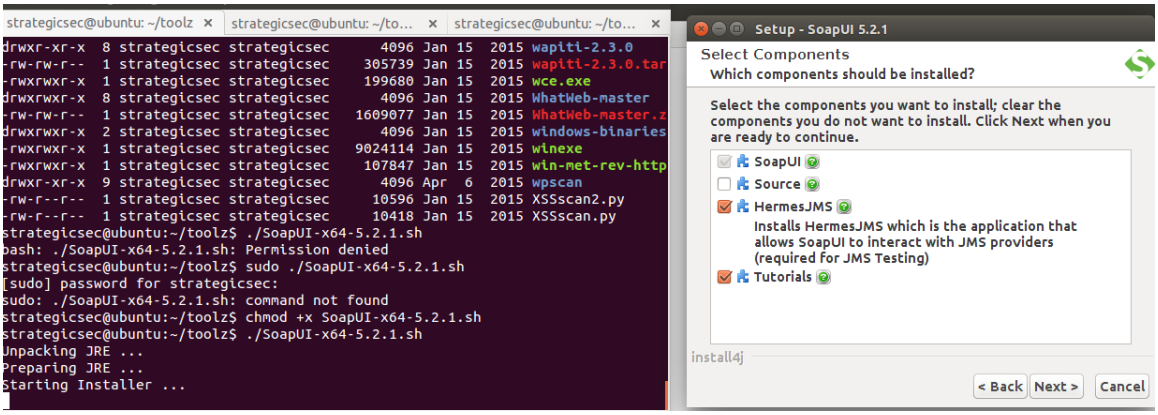
SOAP Binary → XML Request Body



Lab 7: Testing SOAP Web Services by Integrating SoapUI and BURP

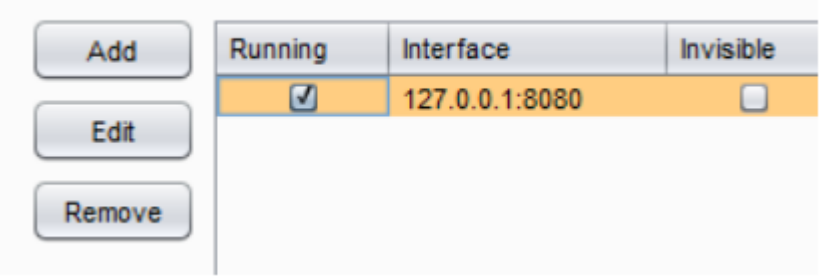
1. Download SoapUI from http://www.soapui.org/downloads/latest-release.html
2. Copy SoapUI.sh to the toolz directory
3. Make SoapUI executable: chmod +x SoapUI-x645.2.1.sh
4. Install SoapUI by entering ./SoapUI-x645.2.1.sh into the terminal
5. It's not necessary to install the source code. The components that you need are already selected by default



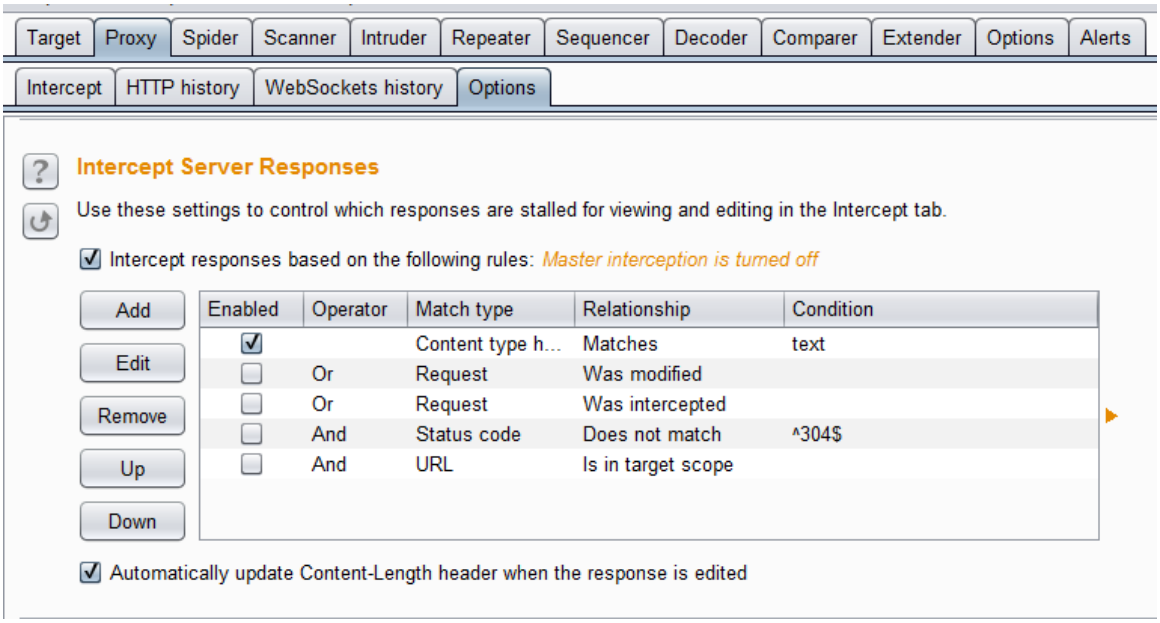


6. You can use the defaults for the rest of the install
7. After SoapUI is installed it will open automatically
8. Close SoapUI so we can configure Burp first.

9. Open Burp and make sure that the proxy is running on 127.0.0.1:8080



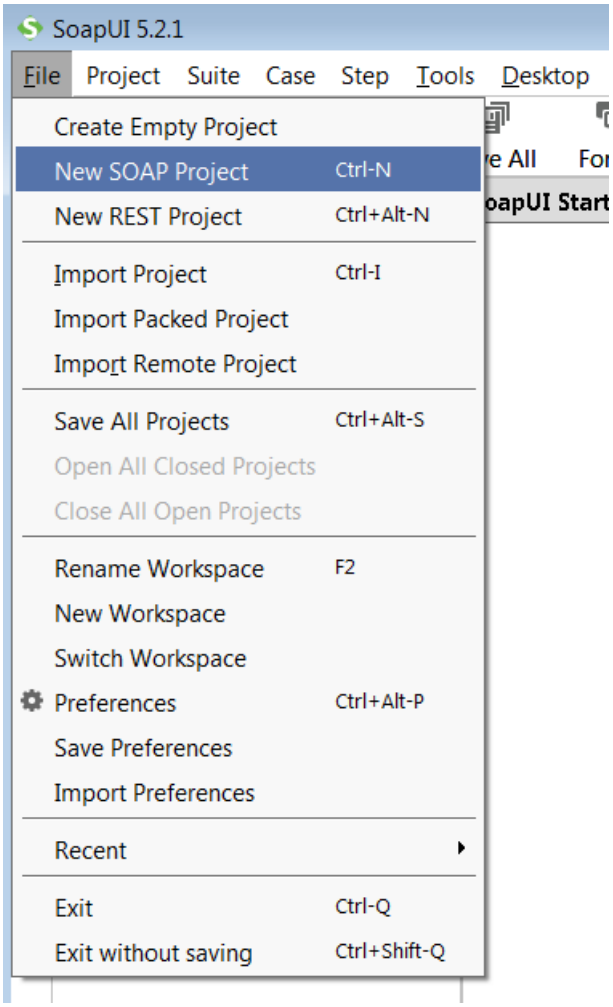
10. Next lets enable “Intercept Server Responses”
11. Make sure that the “Intercept response based on the following rules” is checked



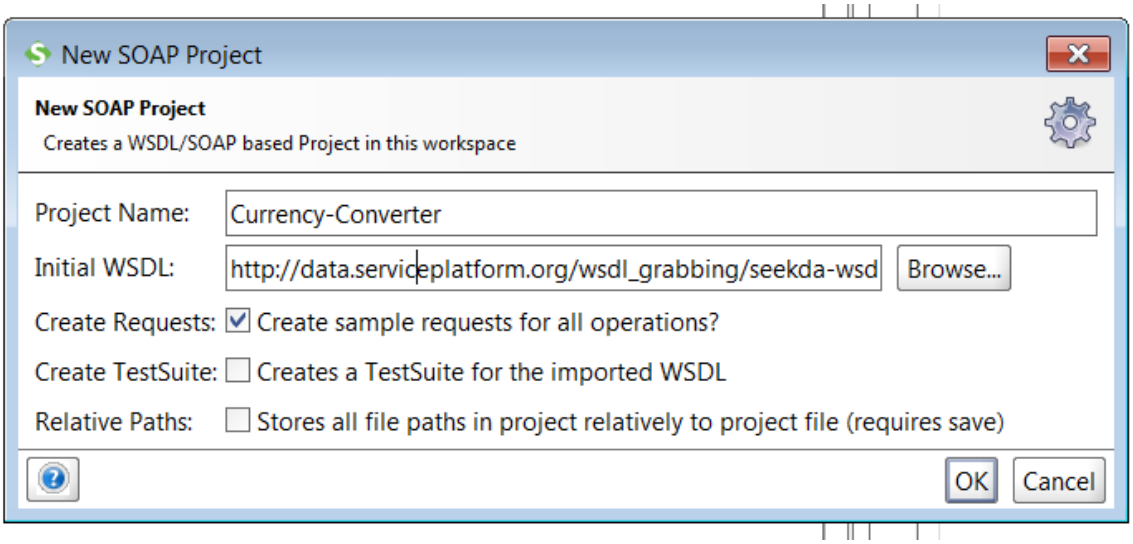
12. We’re ready to integrate Burp and SoapUI
13. Open SoapUI by clicking the desktop Icon

Creating a SoapUI Project

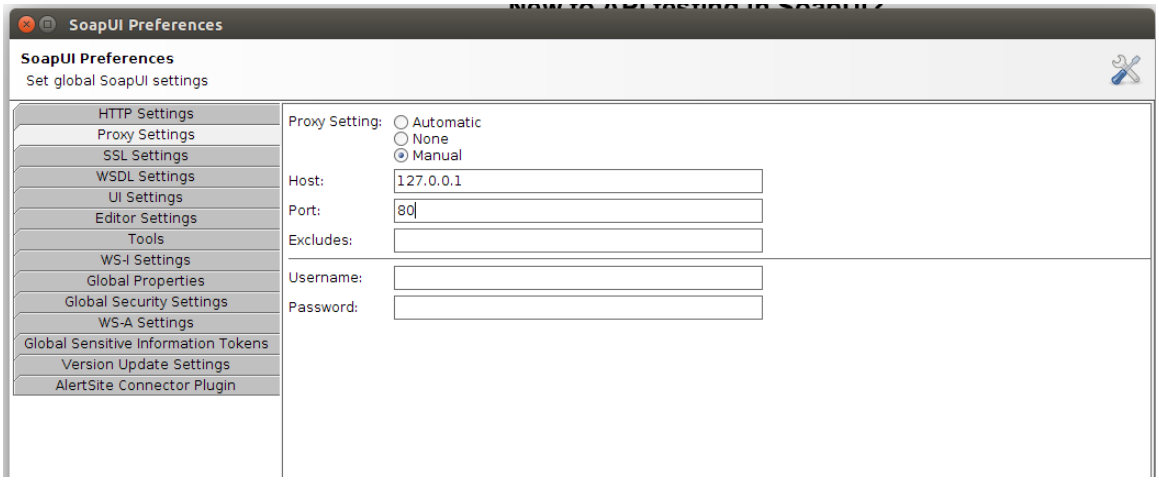
1. Go to the “File” menu and select “New SOAP Project”



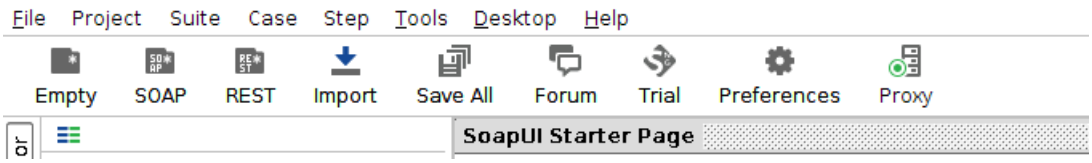
2. Set the name of the project name to Currency-Converter and the initial WSDL to http://data.serviceplatform.org/wsdl_grabbing/seekda-wsdls.with_ini/36-CurrencyConvertor.wsdl then click “ok”



3. Set the Proxy setings in SoapUI with a host of 127.0.0.1 port 80

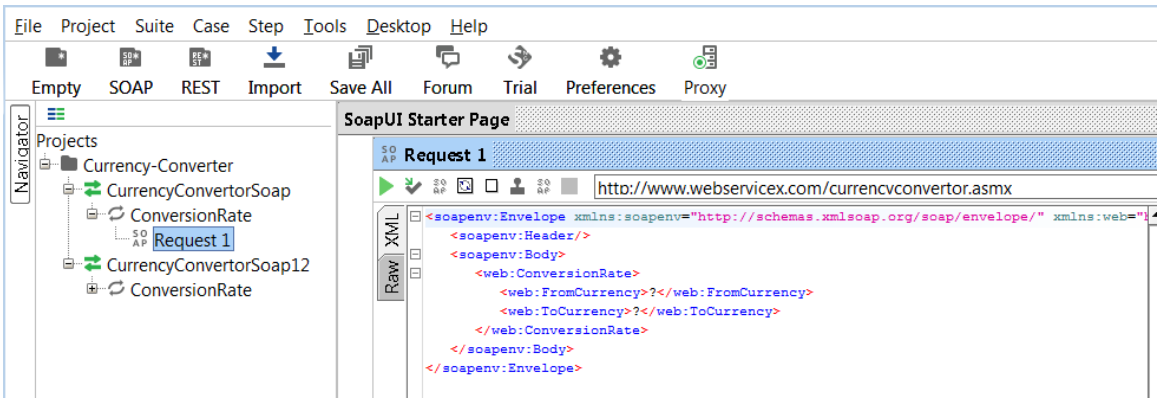


4. Now turn the proxy on in SoapUI. Green is on and red is off



Now we've completed the integration and are ready to capture requests.

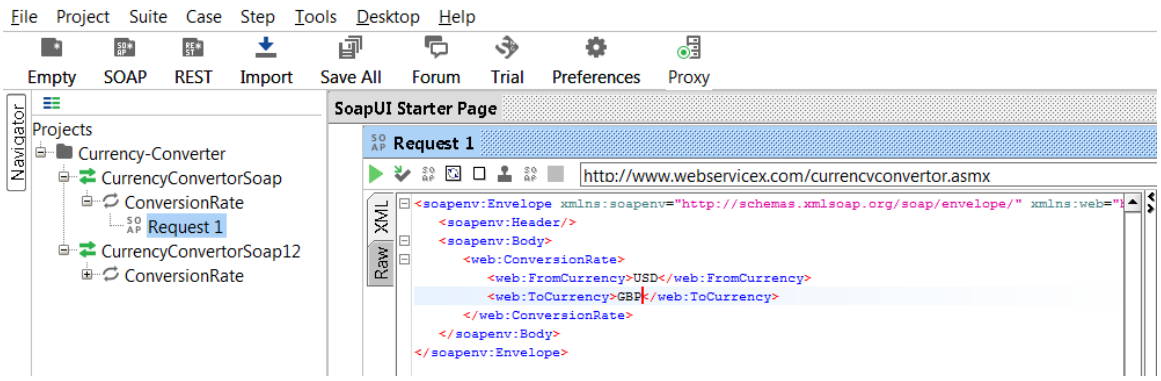
Expand the Conversion Rate and click request1



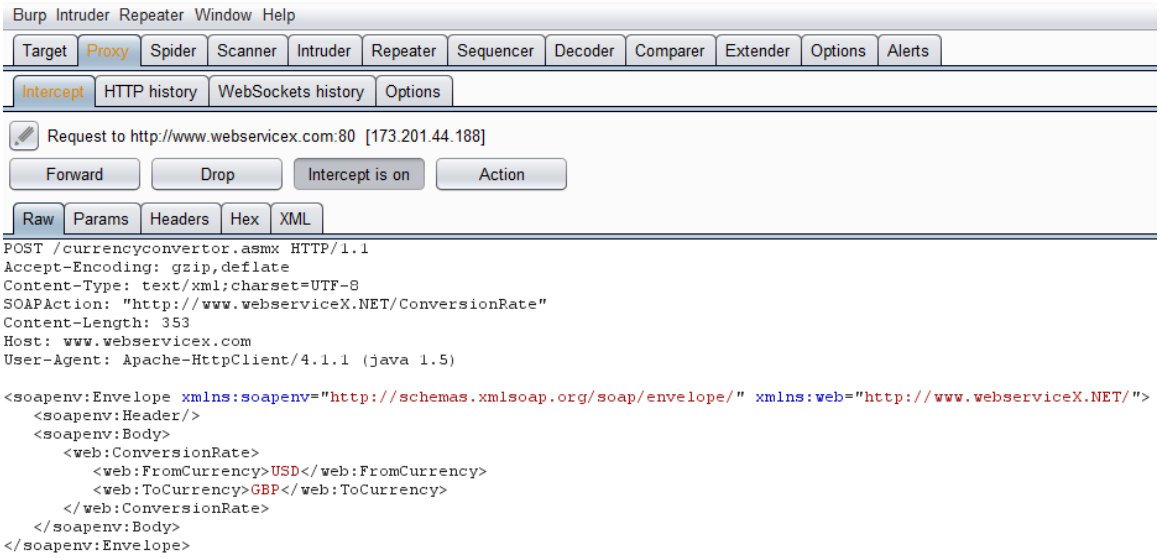
On the right side you'll see the XML of the soap request which takes 2 parameters- "From Currency" and "To Currency"

Substitue the 1st question mark in the XMLwith "USD"
Substitute the 2nd question mark in the XML with "GBP"

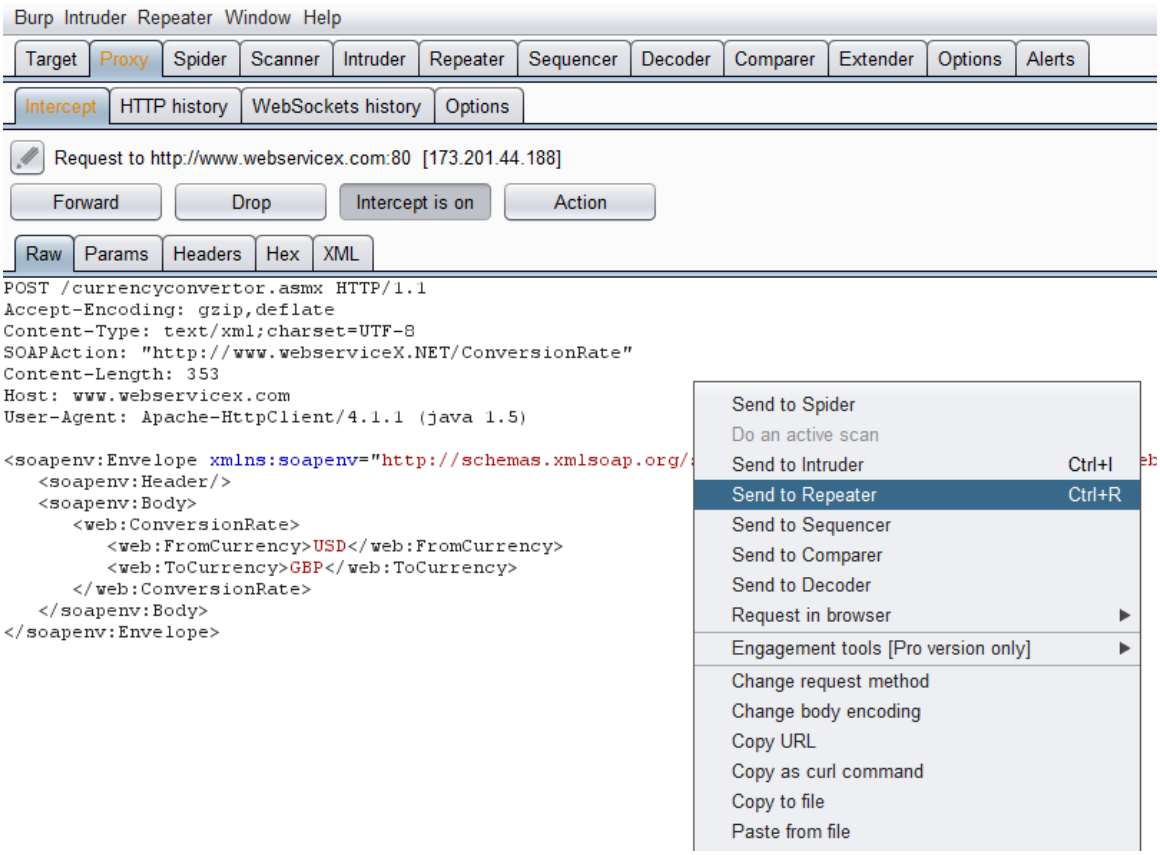
Click the green arrow to send the request. Make sure that Burp is set to intercept requests



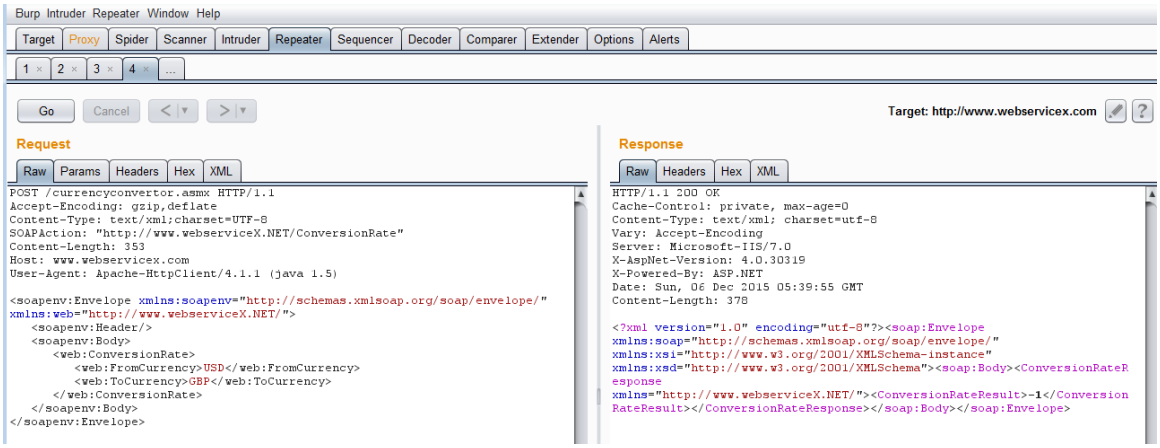
Burp intercepted the request. You'll the currencies that wre set are in the "Raw" request



Now we send the request to Burp's repeater

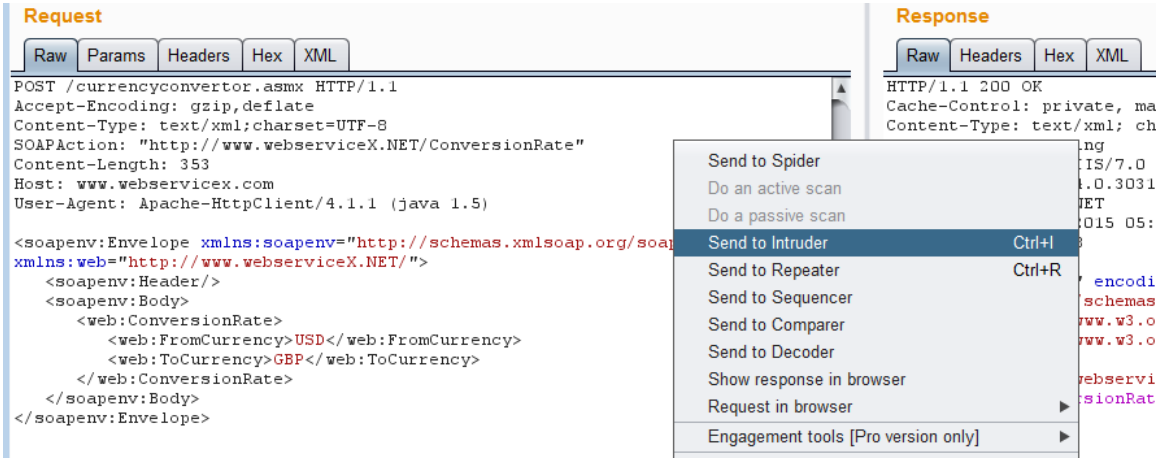


Click "Go" in the repeater and inspect the response. We know the integration is working.

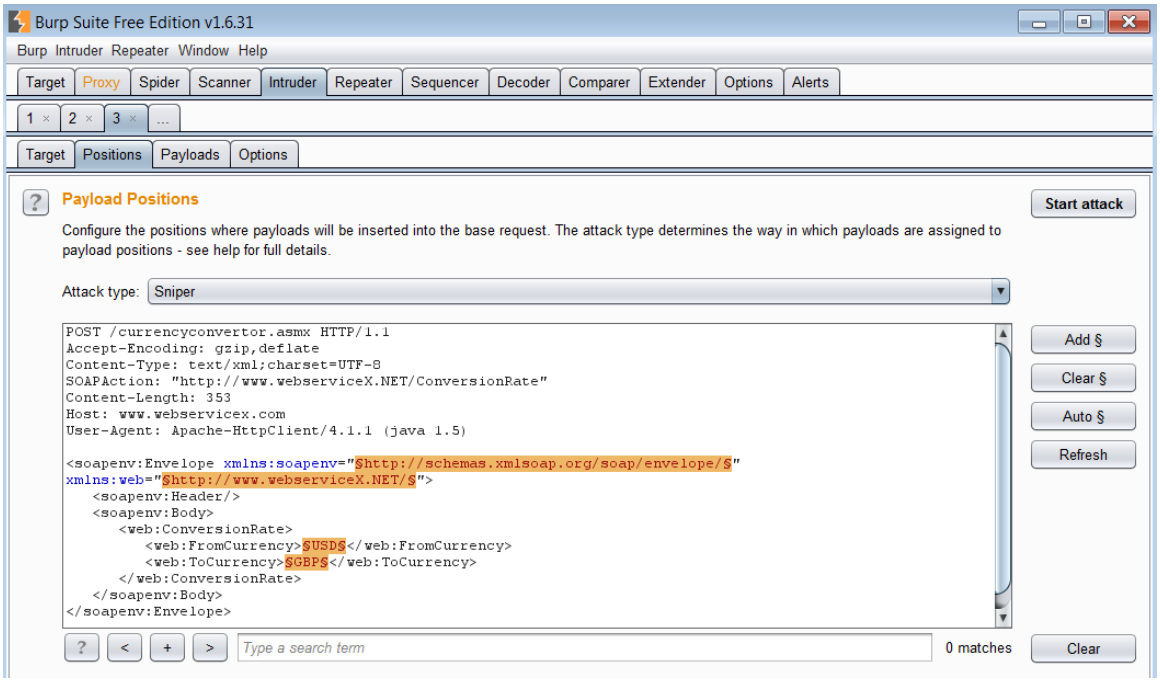


We received a 200 OK and can see that the server is running IIS 7 & ASP.NET 4.0

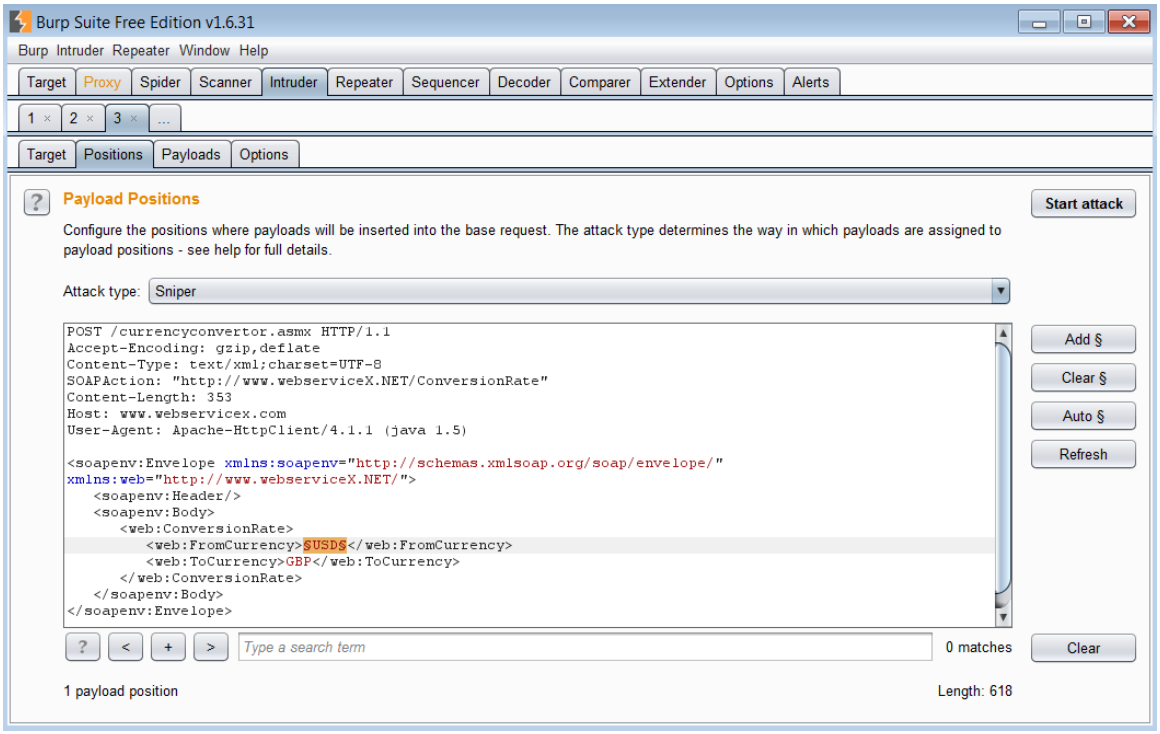
Next we'll fuzz the data by sending the request to the intruder.



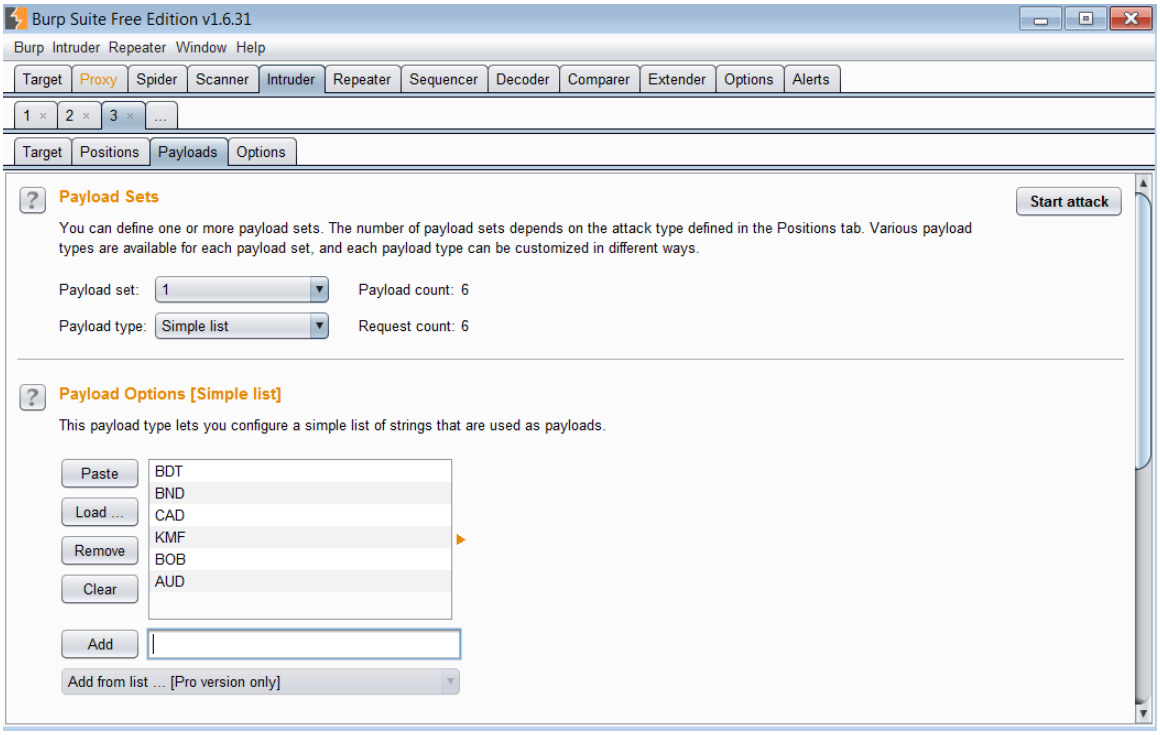
The intruder auto-selected values to change. Click the "Clear" button



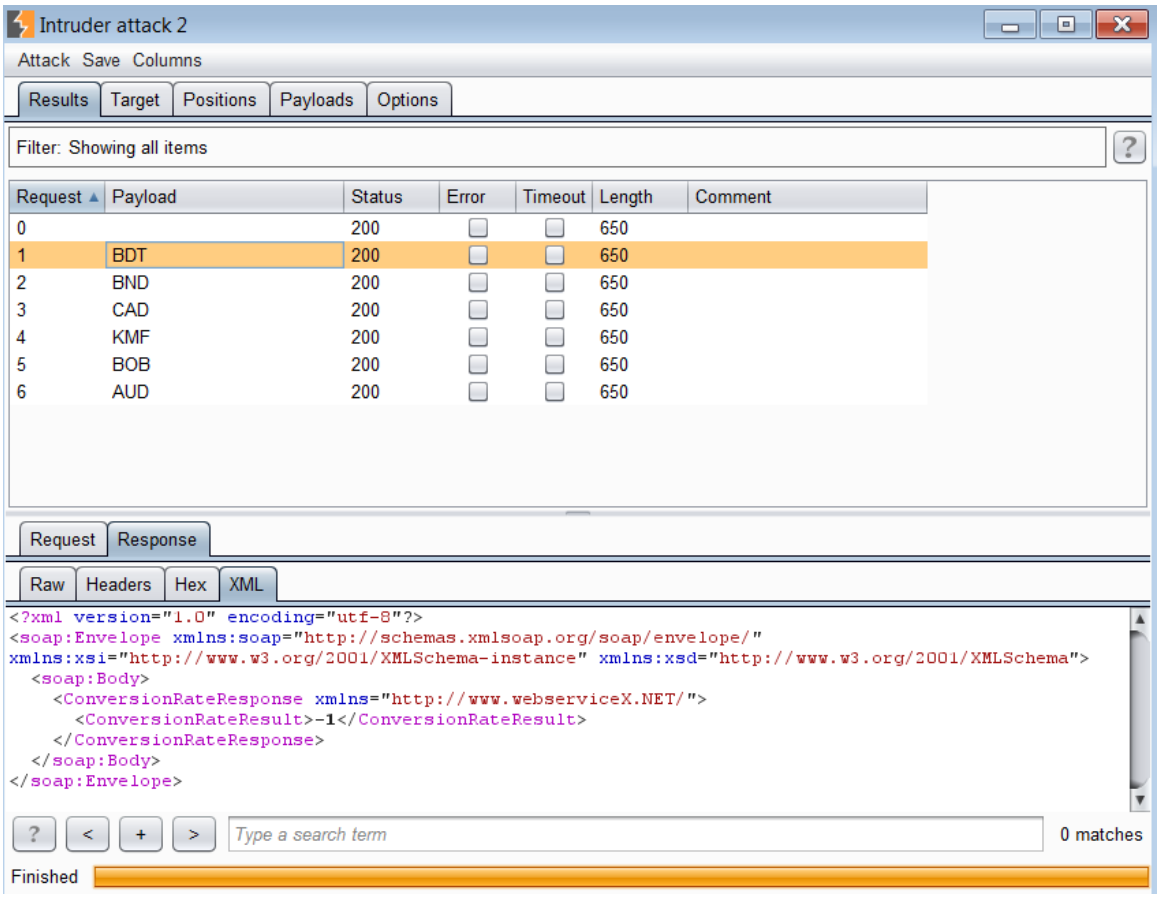
Highlight the “From Currency” Field and click “Add” This is the value that we’ll put into our payload to have the intruder use the values that we enter into the payload.



Load some values into the payload



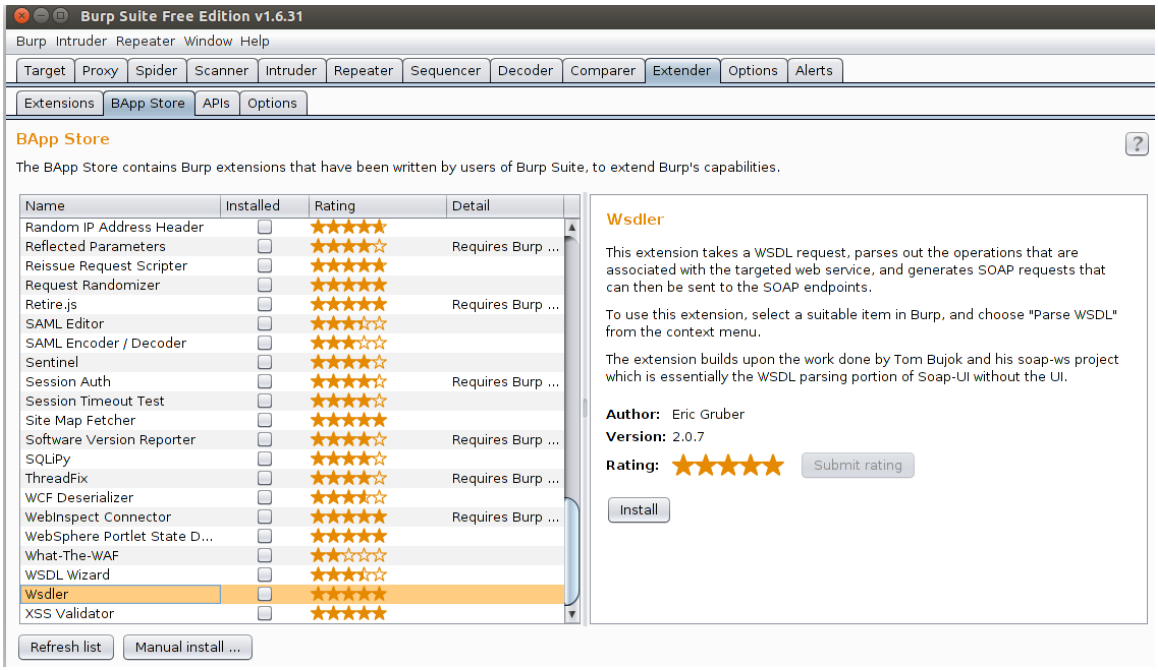
Not much of interest came back. It is only a currency converter



Try another target, for example a bank. That should yield some interesting results.

Lab 8: Using Burp & Wsdler to Hacking Web Services

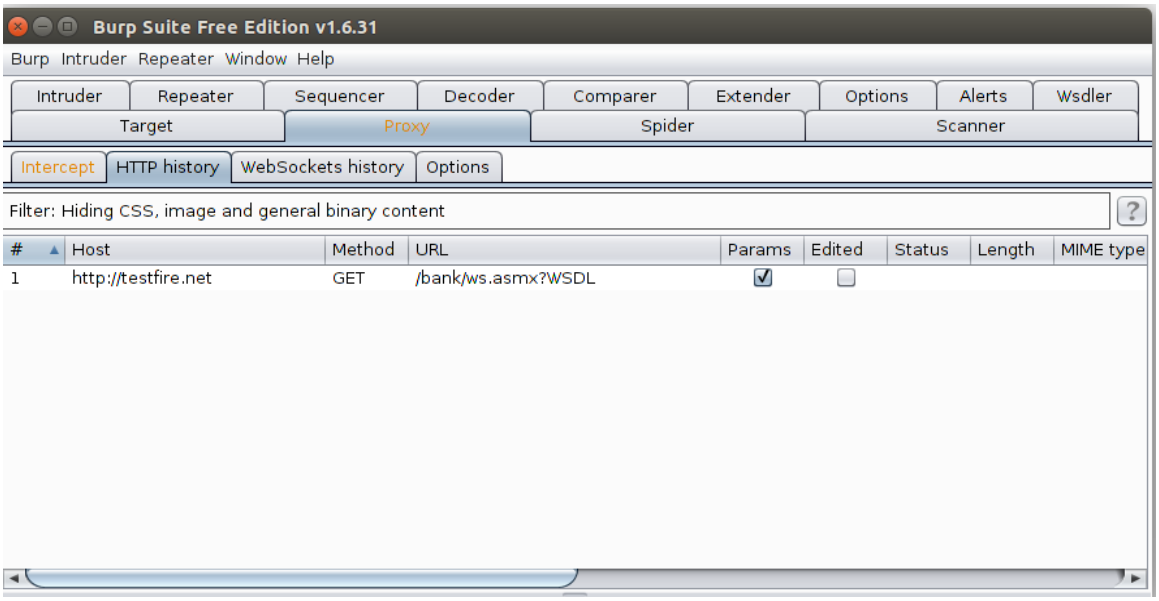
Open Up Burpsuite and click the “Extender” tab and “BApp Store” sub tub



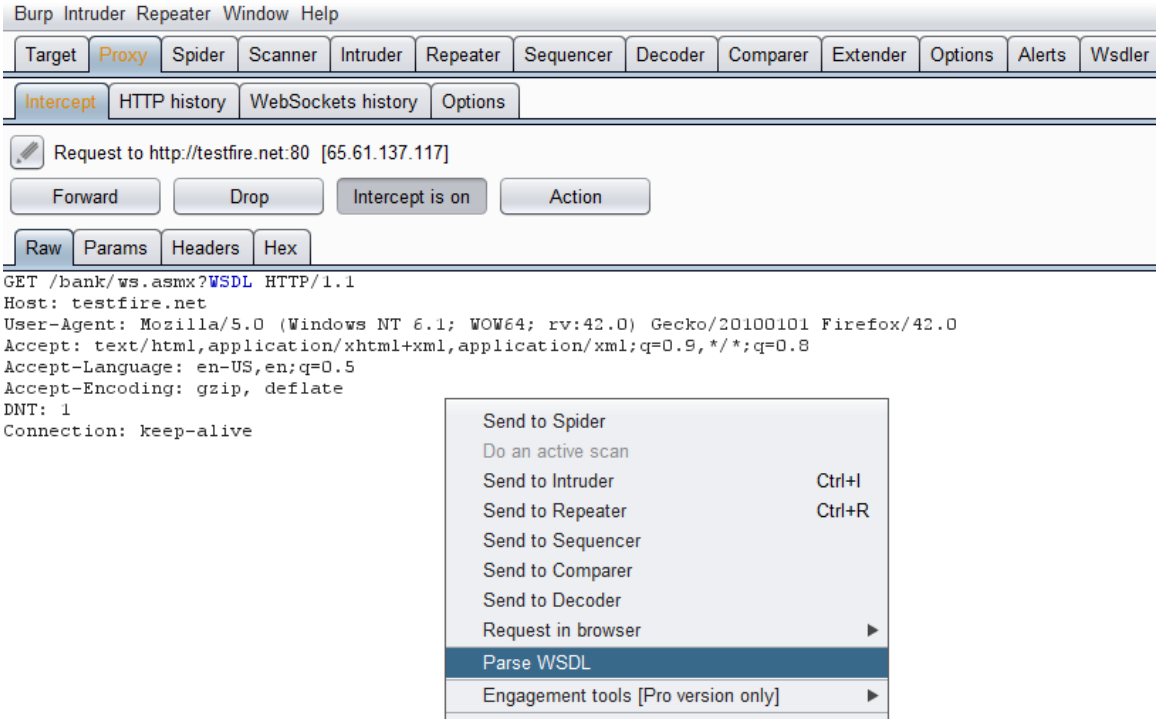
Scroll down through the BApp Store until you find Wsdler. Select it and click “Install” on the right.

Double check t make sure that “intercept” is enabled

Open up your browser and navigate to the URL to the WSDL below
http://www.testfire.net/bank/ws.asmx?WSDL

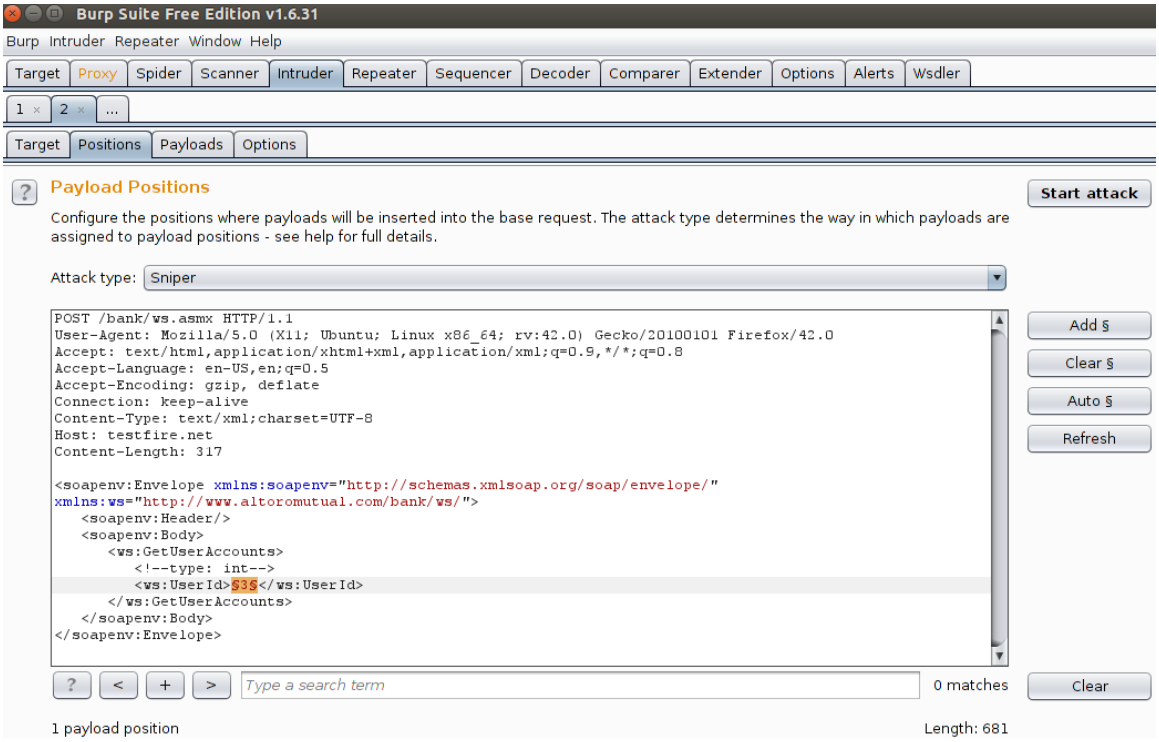


Select Parse WSDL



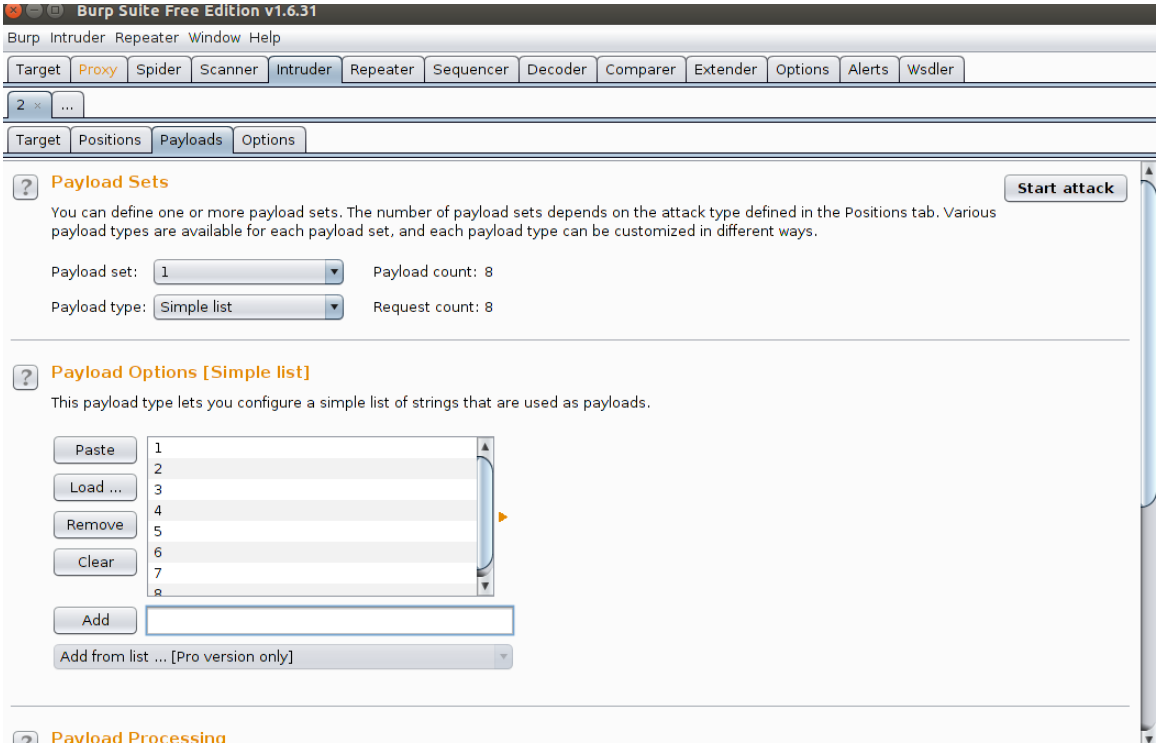
After parsing the WSDL click on the “Wsdler” tab again. Each record has two bindings and an endpoint.

In the request we can see an operation named “GetUserAccount”. That seems interesting. Send the request to the intruder. You may notice a highlighted value of User Id. Click the “Clear” button. We’re only interested in the User Id field. Select the User Id Field and click “Add”.

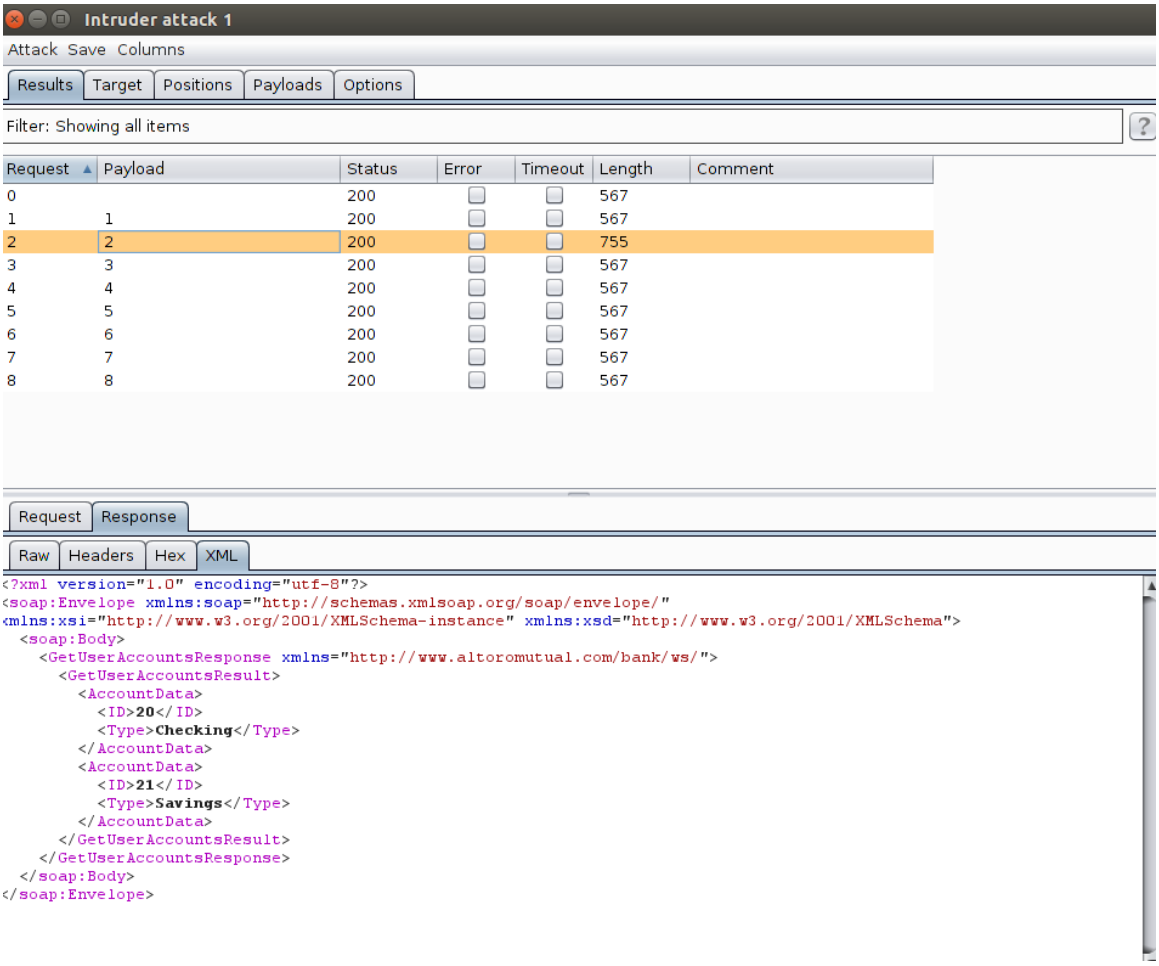


We could have chosen any of the other tabs as Wsdler is a Burp Extender that utilized standard Burp Requests.

Click “Payloads” and add some integers.



Start the attack and see what we find.



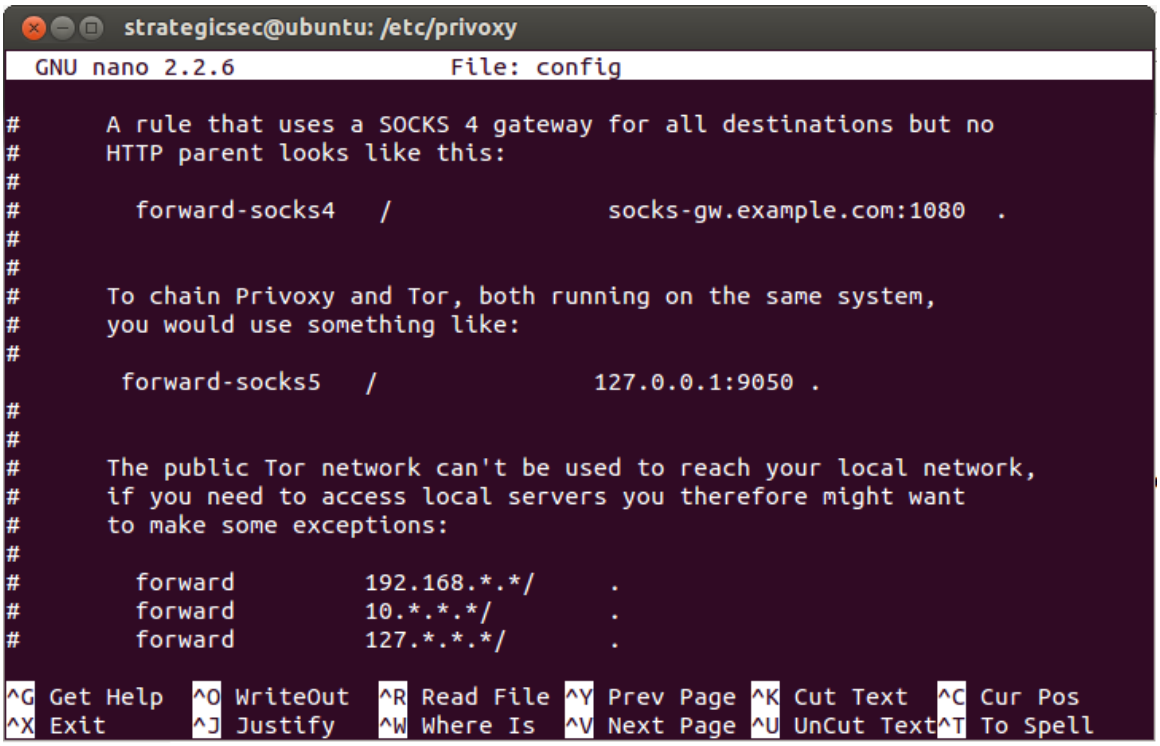
ID #2 looks interesting. Its length is greayer than the others. It looks like we found a checking account with an ID and a savings account with an ID.

Clear the auto-selected balues. Section 2: Masking Yourself With Burp Suite

Lab 9: Burp Suite Through Tor/Privoxy

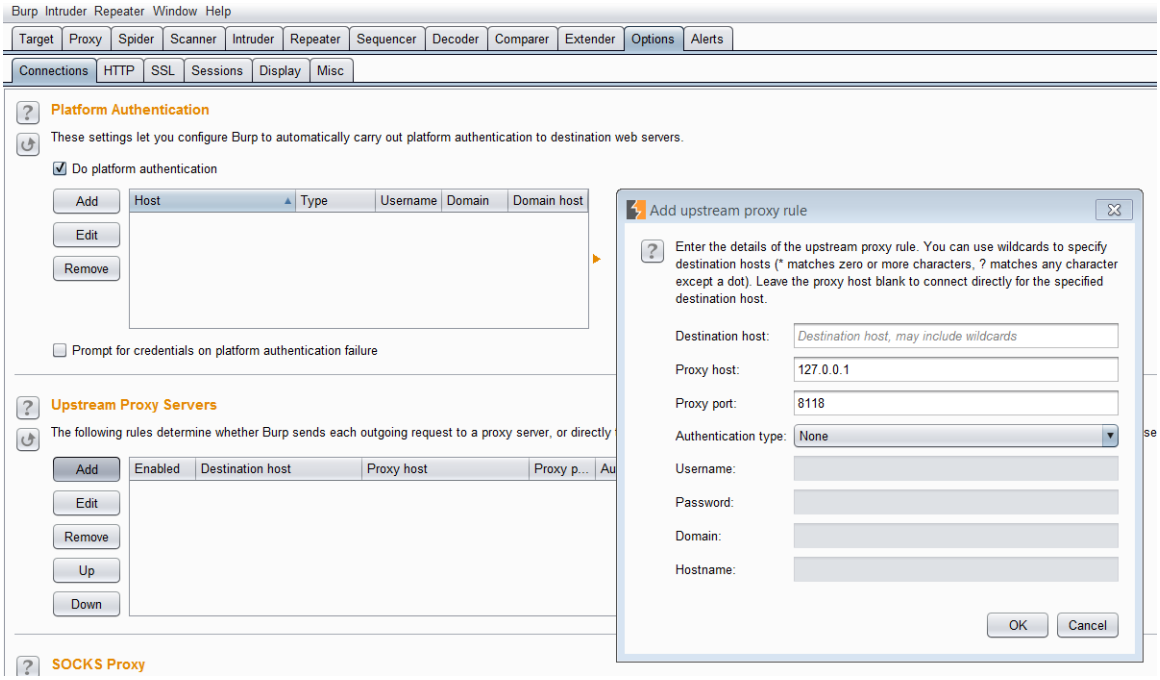
Since we've already installed Tor and configured it, privoxy should be working fine. But we need to configure a few things before everything will work properly.

Open the file up in your favorite text editor and search for "9050"



Once you've found the line that says "forward-socks5 / 127.0.0.1:9050", go ahead and uncomment it.

Now we need to configure the proxy settings. Settings are under the “options” tab. After clicking the “options” tab click the “connections” tab and go to “Upstream Proxy Servers”. Click “add” In the popup enter the proxy host as 127.0.0.1 on proxy port 8118 and then click “ok”



Once you’re finished with this, the final step is to fire up Tor and Privoxy.

Lab 10: Masking Nikto Headers

In this lab we are going to be masking the Nikto User-Agent in the request header. Navigate to the directory where you've stored Nikto. In this directory you'll notice a nikto.conf file.

```
strategicsec@ubuntu: ~/toolz/nikto-2.1.1
strategicsec@ubuntu:~/toolz/nikto-2.1.1$ ls
docs  nikto.conf  nikto.pl  plugins  templates
strategicsec@ubuntu:~/toolz/nikto-2.1.1$
```

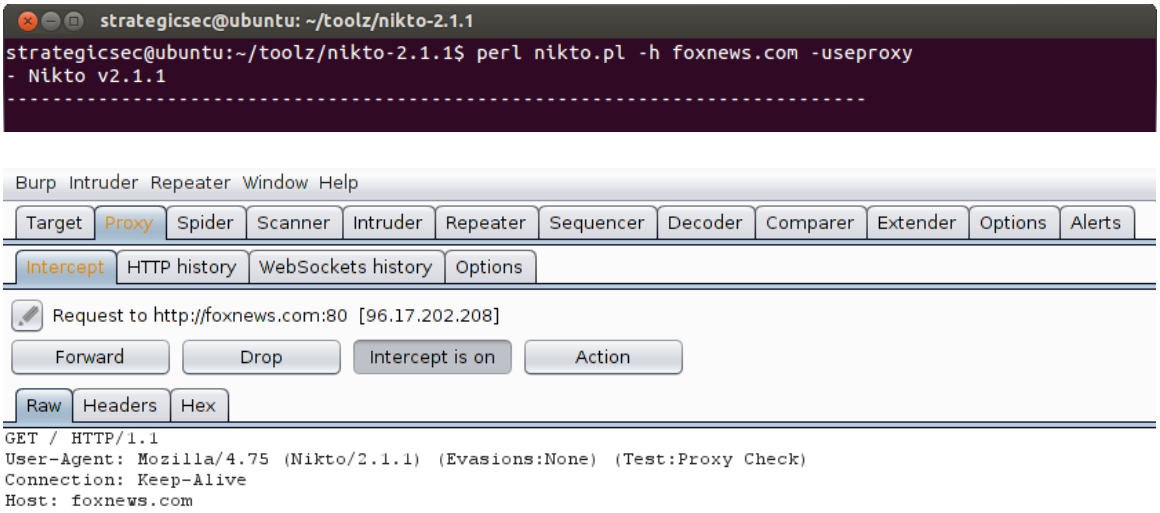
Open up the config file in your favorite text editor and look for the lines referencing proxy options.

```
strategicsec@ubuntu: ~/toolz/nikto-2.1.1
  GNU nano 2.2.6                File: nikto.conf

#PROMPTS=no

# cirt.net : set the IP so that updates can work without name resolution
CIRT=174.142.17.165

##############################################################################$
# PROXY STUFF
##############################################################################$
#PROXYHOST=127.0.0.1
#PROXYPORT=3128
#PROXYUSER=proxyuserid
#PROXYPASS=proxypassword

##############################################################################$
# COOKIE STUFF
##############################################################################$
# send a cookie with all requests, helpful if auth cookie is needed
#STATIC-COOKIE=cookiename=cookievalue


^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Uncomment the two lines for “PROXYHOST” and “PROXYPORT” you will also have to change the “PROXY” port to go through Burp.

```
##############################
# PROXY STUFF
##############################
PROXYHOST=127.0.0.1
PROXYPORT=8080
#PROXYUSER=proxyuserid
#PROXYPASS=proxypassword
```

If we run Nikto we can see what the user agent looks like.



Now to modify Nikto’s User-Agent to do this we need the “mechanize.rb” rubygem. If you are on Fedora you can simply use yum to install it. If not you can download it at (http://mechanize.rubyforge.org) or use the command:

```
sudo gem install mechanize
```

If you’ve installed it via gem install then navigate to the folder:

"/usr/lib/ /gems/1.8/gems/mechanize-2.5.1/lib/"

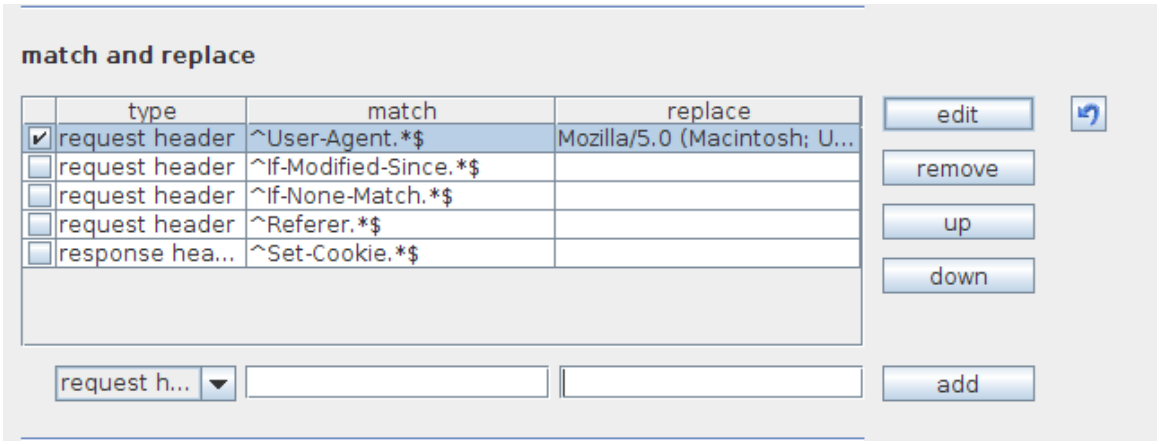
```
  AGENT_ALIASES = {
    'Mechanize' => "Mechanize/#{VERSION} Ruby/#{ruby_version} (http://github.com/tenderlove/mechanize/)",
    'Linux Firefox' => 'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.1) Gecko/20100122 firefox/3.6.1',
    'Linux Konqueror' => 'Mozilla/5.0 (compatible; Konqueror/3; Linux)',
    'Linux Mozilla' => 'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4) Gecko/20030624',
    'Mac FireFox' => 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6',
    'Mac Mozilla' => 'Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.4a) Gecko/20030401',
    'Mac Safari 4' => 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_2; de-at) AppleWebKit/531.21.8 (KHTML, like Gecko)
Version/4.0.4 Safari/531.21.10',
    'Mac Safari' => 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2) AppleWebKit/534.51.22 (KHTML, like Gecko)
Version/5.1.1 Safari/534.51.22',
    'Windows IE 6' => 'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)',
    'Windows IE 7' => 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)',
    'Windows IE 8' => 'Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR
2.0.50727)',
    'Windows IE 9' => 'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)',
    'Windows Mozilla' => 'Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.4b) Gecko/20030516 Mozilla Firebird/0.6',
    'iPhone' => 'Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en) AppleWebKit/420+ (KHTML, like Gecko) Version/3.0
Mobile/1C28 Safari/419.3',
  }
```

In the mechanized.rb file you can see the different user agents. From this list we need to make a separate user-agent.txt file. You may want to clean it up a little bit.

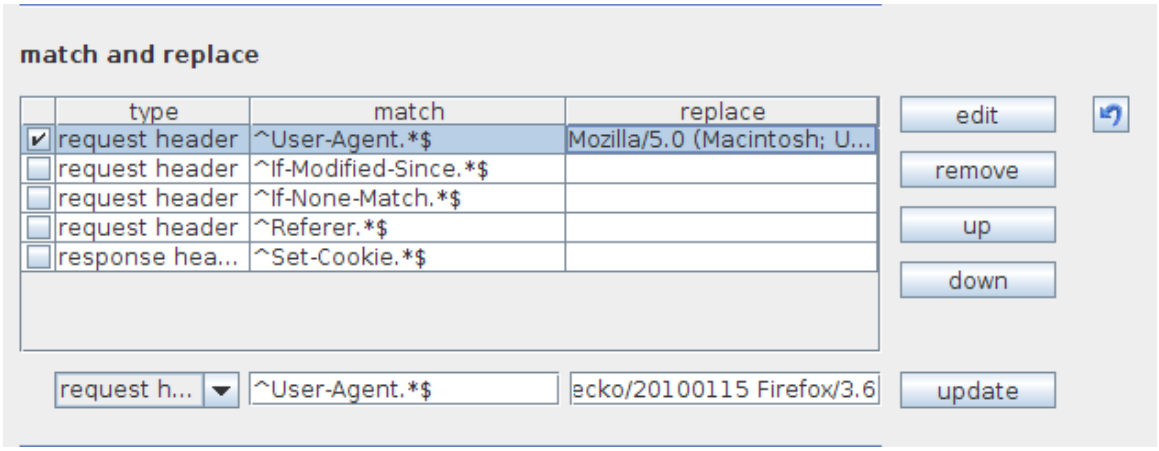
user-agent.txt

```
'Linux Firefox' => 'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.1) Gecko/20100122 firefox/3.6.1',
'Linux Konqueror' => 'Mozilla/5.0 (compatible; Konqueror/3; Linux)',
'Linux Mozilla' => 'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4) Gecko/20030624',
'Mac FireFox' => 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6',
'Mac Mozilla' => 'Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.4a) Gecko/20030401',
'Mac Safari 4' => 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_2; de-at) AppleWebKit/531.21.8 (KHTML, like Gecko)
Version/4.0.4 Safari/531.21.10',
'Mac Safari' => 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2) AppleWebKit/534.51.22 (KHTML, like Gecko) Version/5.1.1
Safari/534.51.22',
'Windows IE 6' => 'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)',
'Windows IE 7' => 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)',
'Windows IE 8' => 'Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR
2.0.50727)',
'Windows IE 9' => 'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)',
'Windows Mozilla' => 'Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.4b) Gecko/20030516 Mozilla Firebird/0.6',
'iPhone' => 'Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en) AppleWebKit/420+ (KHTML, like Gecko) Version/3.0 Mobile/1C28
Safari/419.3',
```

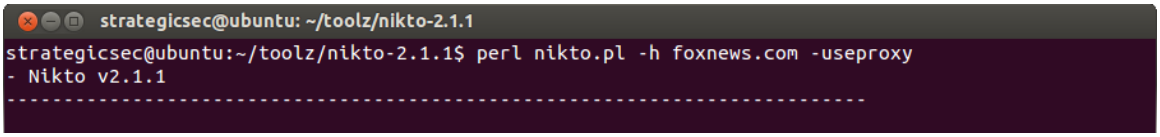
Now we have to can change the user-agent. Go into Burp and navigate to Proxy -> Options and scroll down to "Match and Replace."



Just copy and paste in the user-agent information from your user-agent.txt file. I am going to use the Mac Firefox user-agent.



Make sure that the request header box is checked. Now run Nikto again.

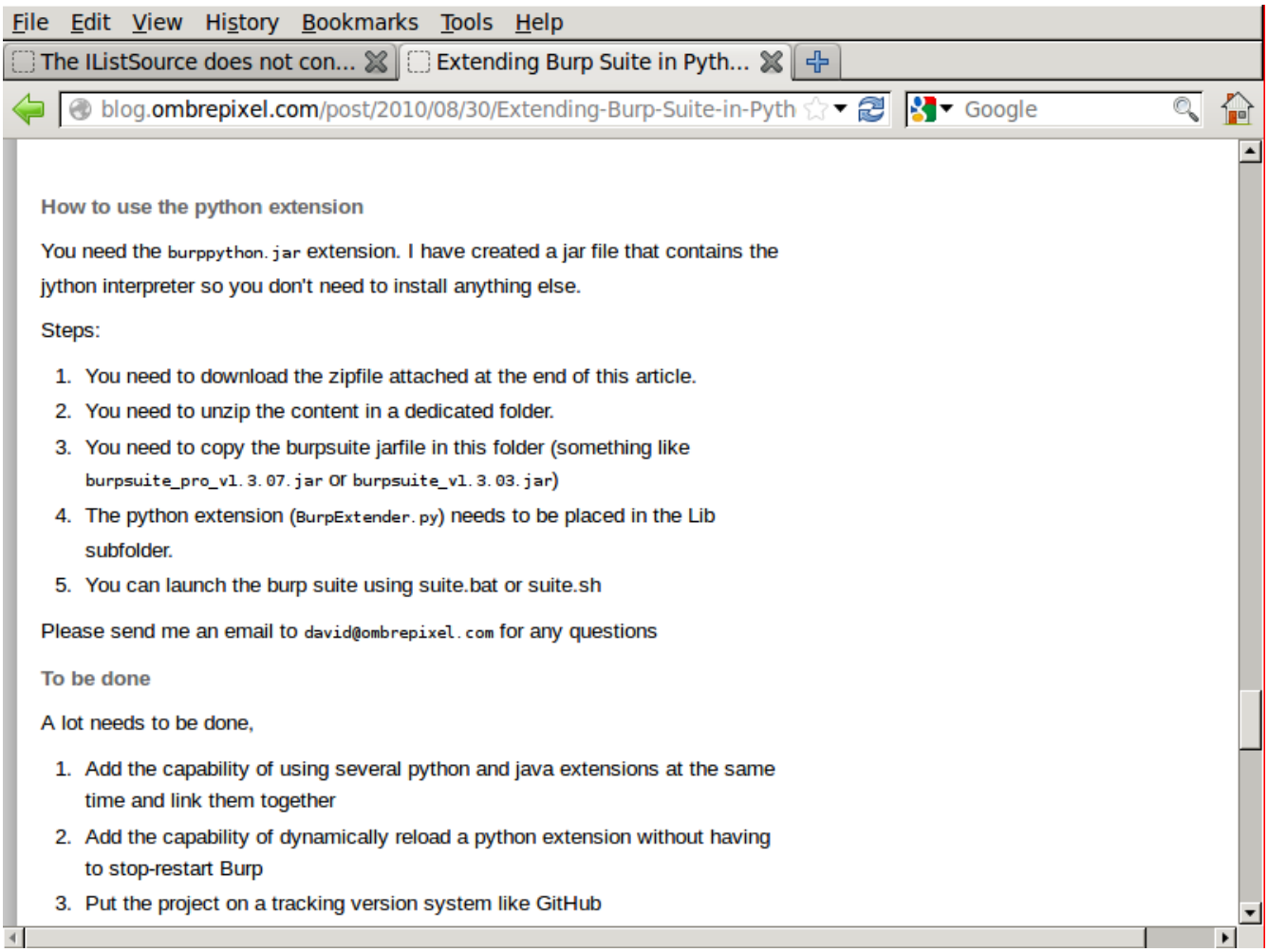


Once you navigate to a web page, you should see the activity under the “Proxy” tab and then under the “Intercept” tab:

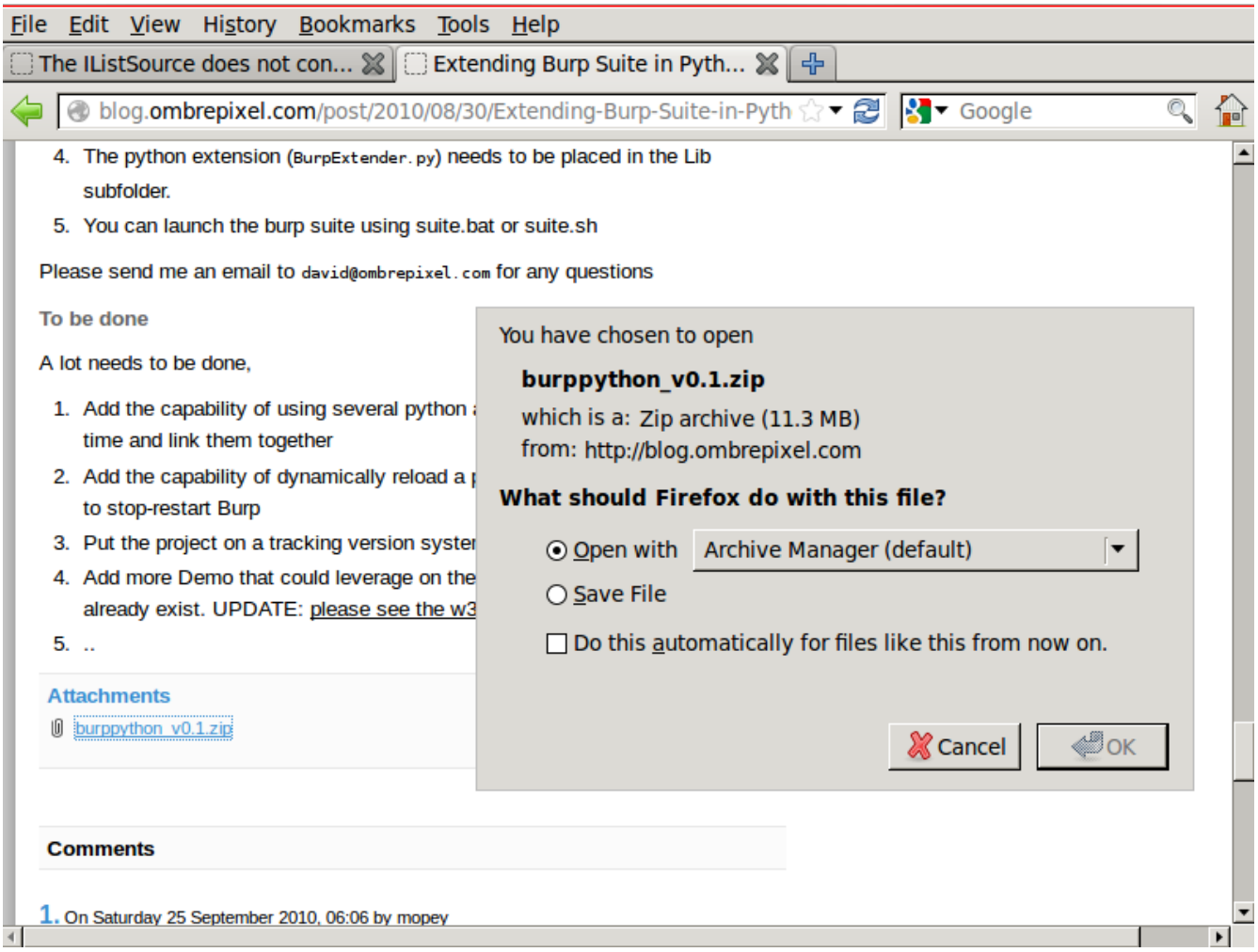
Section 3: Extending Burp Suite

Lab 11: Burp Python

Download burppython:



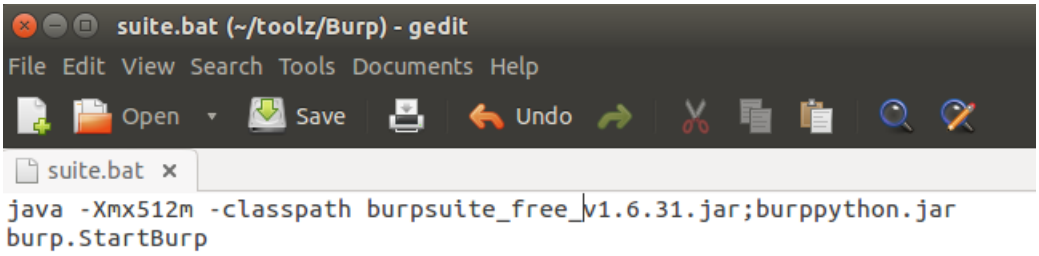
Download it:


File Edit View History Bookmarks Tools Help
The IListSource does not con...
Extending Burp Suite in Pyth...
blog.ombrepixel.com/post/2010/08/30/Extending-Burp-Suite-in-Pyth
Google
4. The python extension (BurpExtender.py) needs to be placed in the Lib subfolder.
5. You can launch the burp suite using suite.bat or suite.sh
Please send me an email to david@ombrepixel.com for any questions
To be done
A lot needs to be done,
1. Add the capability of using several python
time and link them together
2. Add the capability of dynamically reload a
to stop-restart Burp
3. Put the project on a tracking version syste
4. Add more Demo that could leverage on the
already exist. UPDATE: please see the w3
5. ..
Attachments
burppython_v0.1.zip
You have chosen to open
burppython_v0.1.zip
which is a: Zip archive (11.3 MB)
from: http://blog.ombrepixel.com
What should Firefox do with this file?
Open with
Archive Manager (default)
Save File
Do this automatically for files like this from now on.
Cancel
OK
Comments
1. On Saturday 25 September 2010, 06:06 by mopey

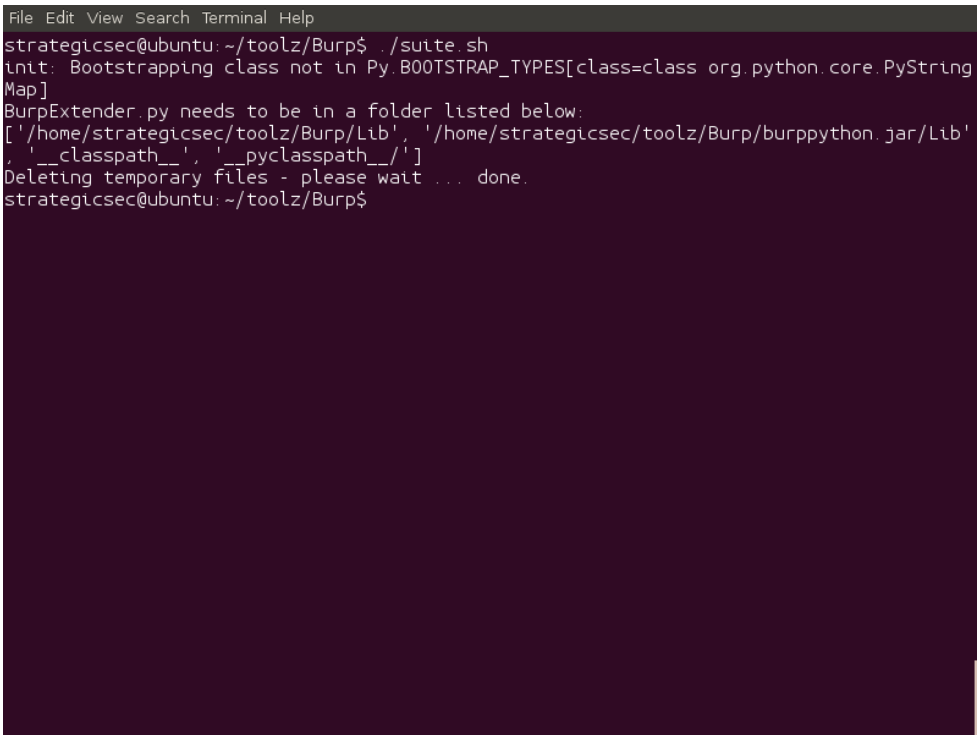
Extract it:

```
File  Edit  View  Search  Terminal  Help
payloadz
strategicsec@ubuntu:~/toolz$ unzip burppython_v0.1.zip
Archive:  burppython_v0.1.zip
   creating: Burp/
  inflating: Burp/burppython.jar
   creating: Burp/Demo/
  inflating: Burp/Demo/BurpExtender-interactive.py
  inflating: Burp/Demo/BurpExtender-menu.py
  inflating: Burp/Demo/BurpExtender-minimal.py
   creating: Burp/Lib/
  inflating: Burp/Lib/BurpExtender.py
  inflating: Burp/python-logo.gif
  inflating: Burp/README.txt
   creating: Burp/src/
   creating: Burp/src/burp/
  inflating: Burp/src/BurpExtender.java
  inflating: Burp/src/burp/IBurpExtender.java
  inflating: Burp/src/burp/IBurpExtenderCallbacks.java
  inflating: Burp/src/burp/IHttpRequestResponse.java
  inflating: Burp/src/burp/IMenuItemHandler.java
  inflating: Burp/src/burp/IScanIssue.java
  inflating: Burp/src/burp/IScanQueueItem.java
  inflating: Burp/suite.bat
  inflating: Burp/suite.sh
strategicsec@ubuntu:~/toolz$ mv burpsuite_free_v1.5.jar Burp/
strategicsec@ubuntu:~/toolz$ cd Burp/
strategicsec@ubuntu:~/toolz/Burp$ ls
burppython.jar           Demo  python-logo.gif  src        suite.sh
burpsuite_free_v1.5.jar  Lib   README.txt       suite.bat
strategicsec@ubuntu:~/toolz/Burp$ chmod u+x suite.sh
strategicsec@ubuntu:~/toolz/Burp$
```

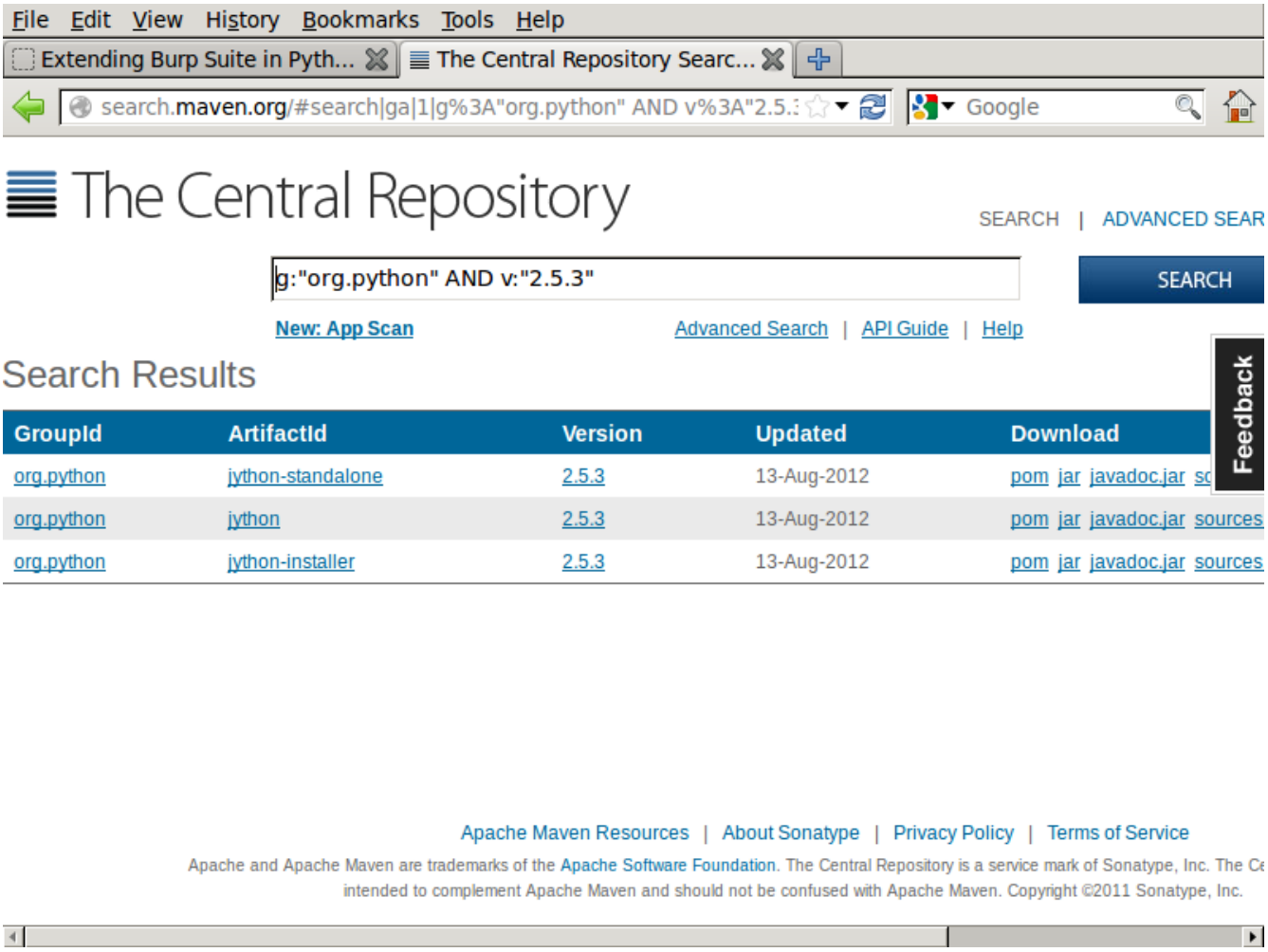
Edit suite.sh and change burpsuite version:



Get an error:

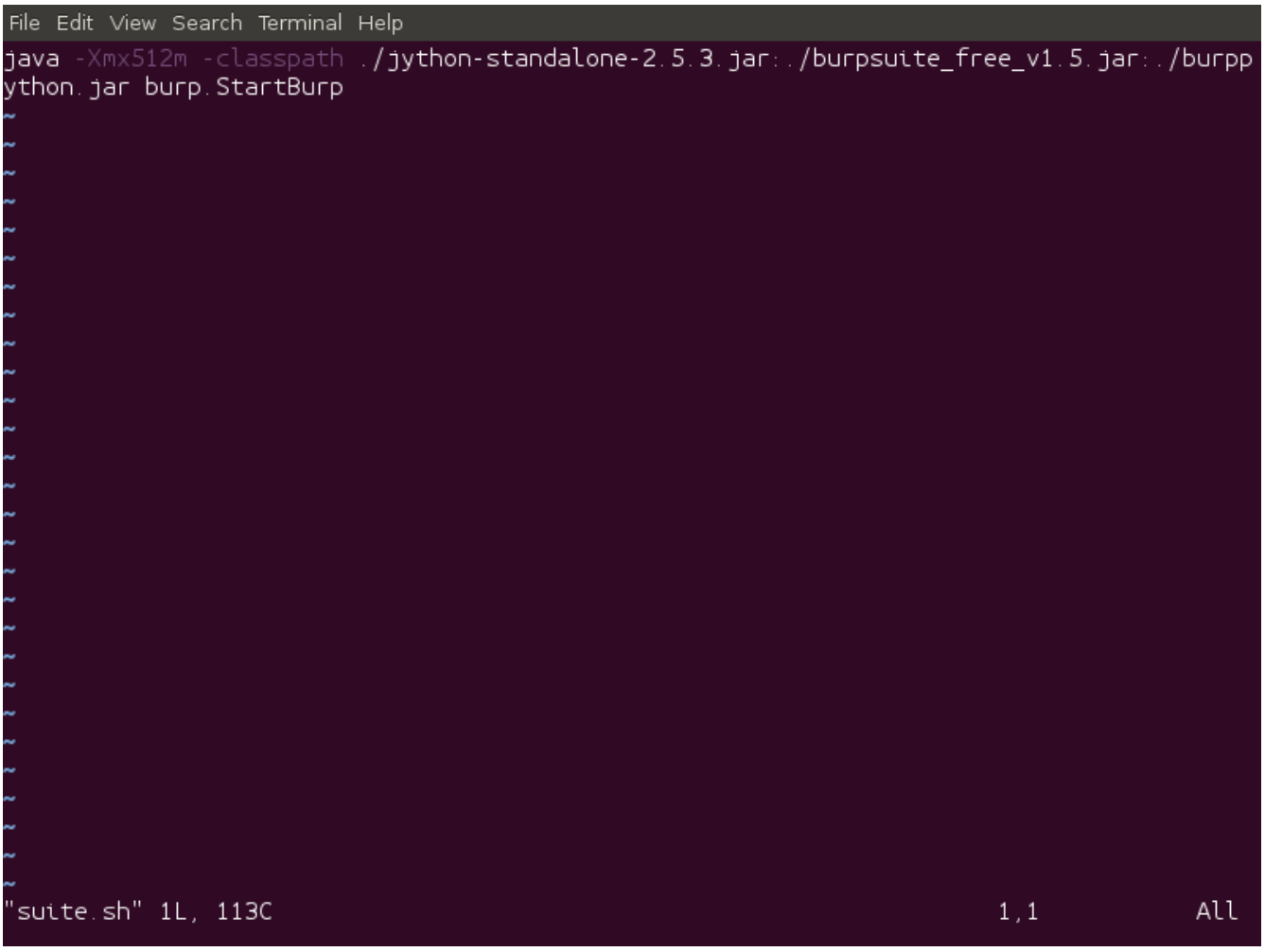


Get new jython version:

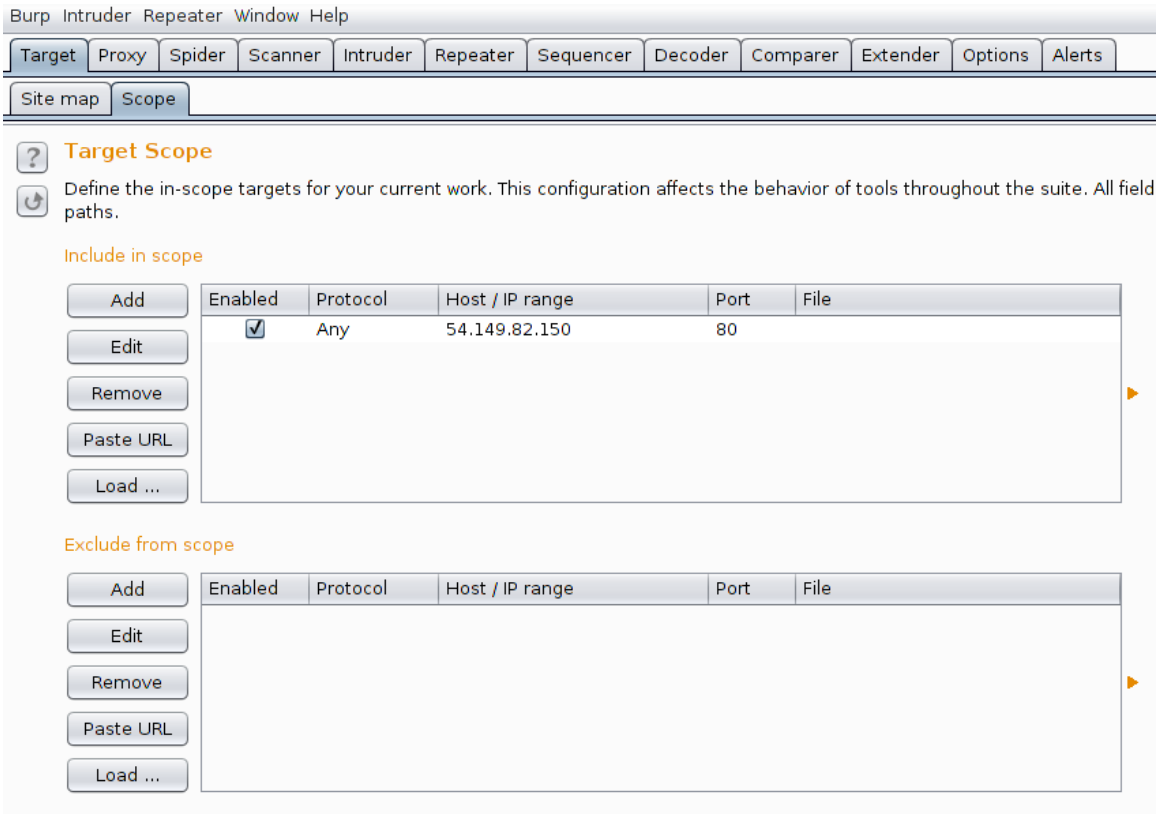


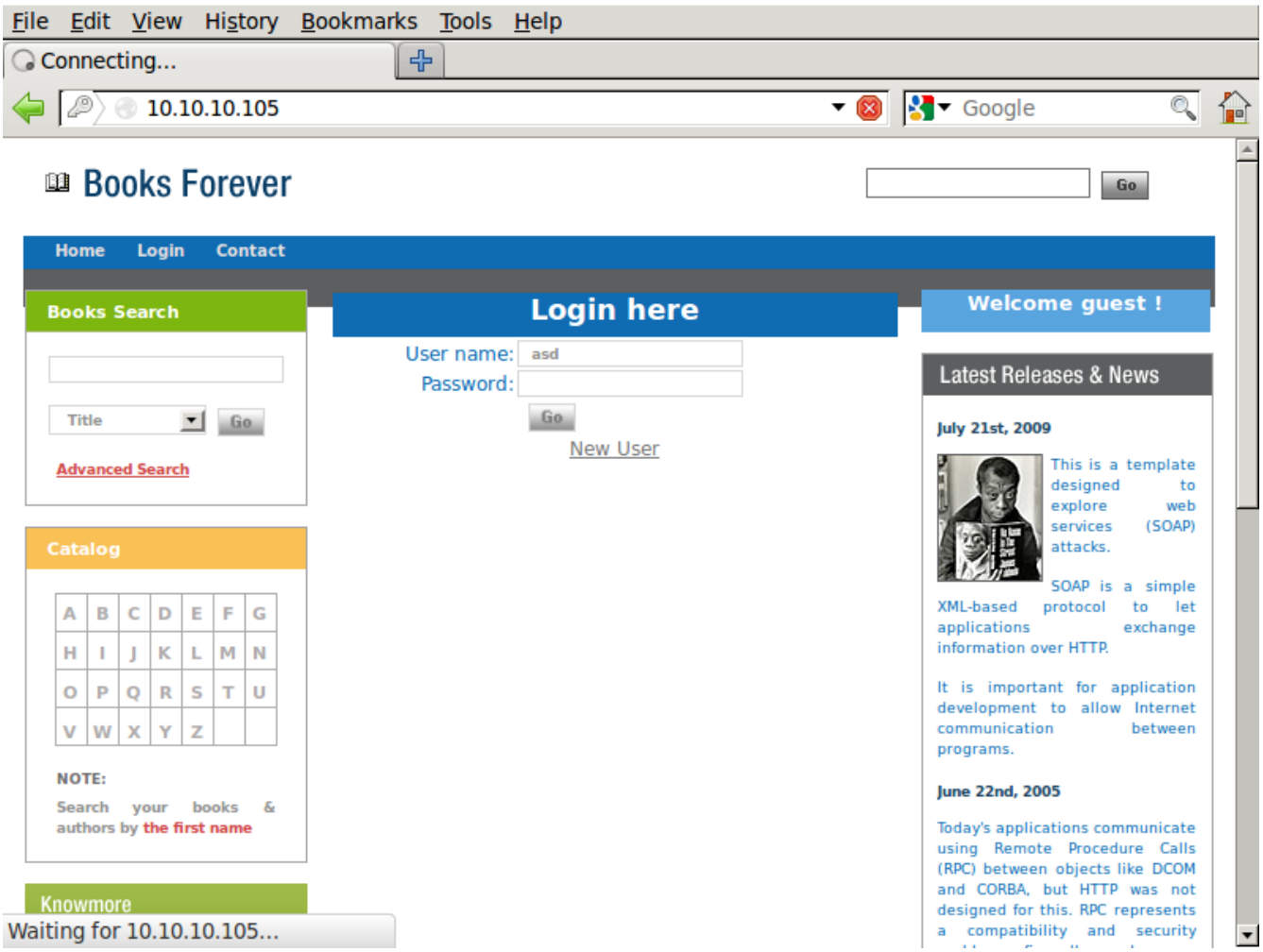
Put jython in folder:

Edit suite.sh and add jython to classpath, also copy Demo/BurpExtender-interactive.py to Lib/BurpExtender.py:

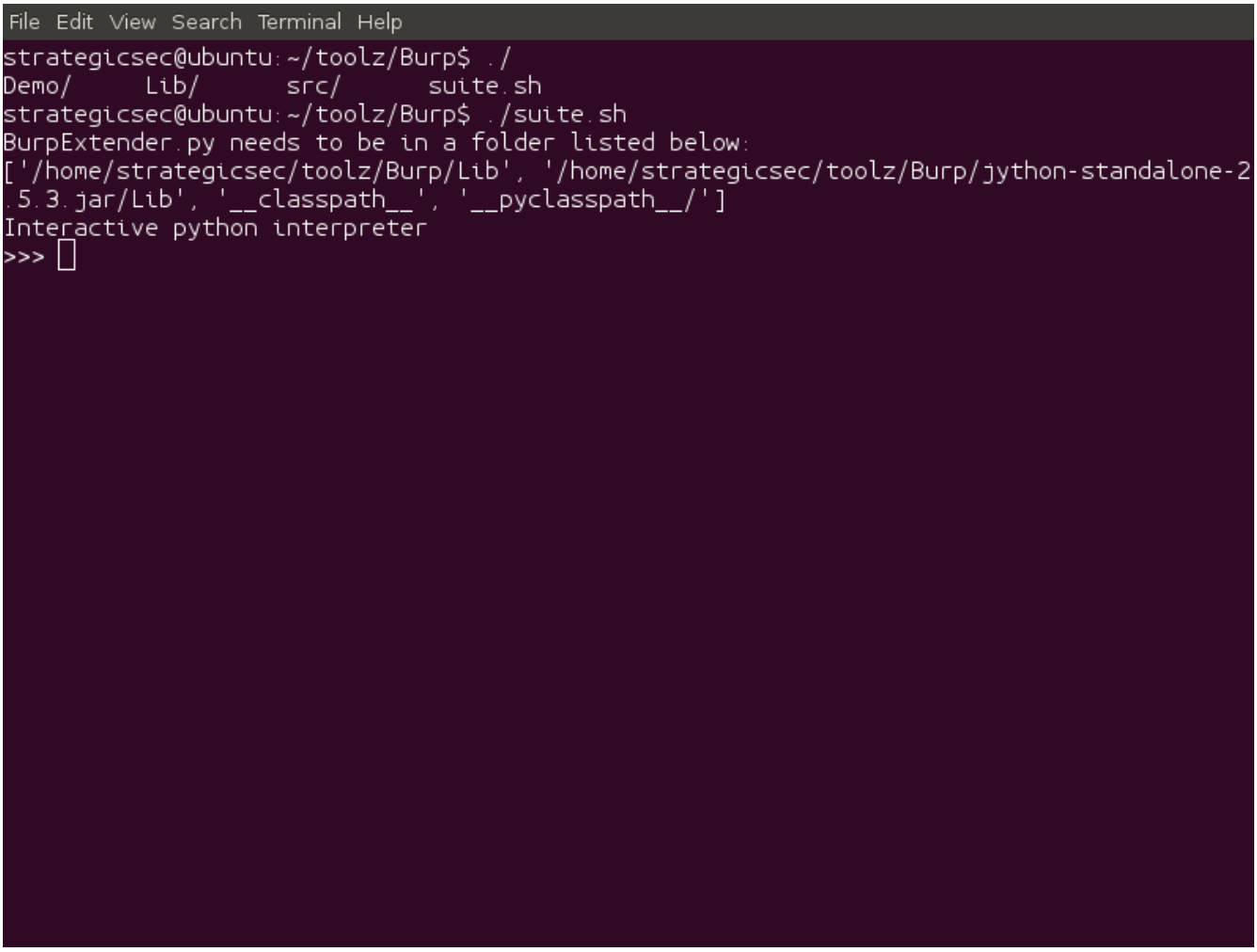


Add target:


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Site map Scope
Target Scope
Define the in-scope targets for your current work. This configuration affects the behavior of tools throughout the suite. All field paths.
Include in scope
Add
Edit
Remove
Paste URL
Load ...
Enabled Protocol Host / IP range Port File
Any 54.149.82.150 80
Exclude from scope
Add
Edit
Remove
Paste URL
Load ...
Enabled Protocol Host / IP range Port File



Check out console window:


File Edit View Search Terminal Help
strategicsec@ubuntu:~/toolz/Burp$ ./
Demo/ Lib/ src/ suite.sh
strategicsec@ubuntu:~/toolz/Burp$ ./suite.sh
BurpExtender.py needs to be in a folder listed below:
['/home/strategicsec/toolz/Burp/Lib', '/home/strategicsec/toolz/Burp/jython-standalone-2
.5.3.jar/Lib', '__classpath__', '__pyclasspath__/']
Interactive python interpreter
>>>

Test shell:

```
File  Edit  View  Search  Terminal  Help
strategicsec@ubuntu:~/toolz/Burp$ ./suite.sh
BurpExtender.py needs to be in a folder listed below:
['/home/strategicsec/toolz/Burp/Lib', '/home/strategicsec/toolz/Burp/jython-standalone-2
.5.3.jar/Lib', '__classpath__', '__pyclasspath__/']
Interactive python interpreter
>>> print 5
5
>>> pprint(dir())
['httpMethod',
 'interceptAction',
 'message',
 'messageIsRequest',
 'messageReference',
 'pprint',
 'remoteHost',
 'remotePort',
 'resourceType',
 'responseContentType',
 'self',
 'serviceIsHttps',
 'statusCode',
 'uUrl',
 'url']
>>> pprint(dir(self))
['ACTION_DONT_INTERCEPT',
 'ACTION_DONT_INTERCEPT_AND_REHOOK',
 'ACTION_DO_INTERCEPT',
 'ACTION_DO_INTERCEPT_AND_REHOOK',
 'ACTION_DROP',
 'ACTION_FOLLOW_RULES',
 'ACTION_FOLLOW_RULES_AND_REHOOK',
```

Lab 12: BurpExtender-w3af

Download BurpExtender-w3af and copy it as Lib/BurpExtender.py:

```
File  Edit  View  Search  Terminal  Help
strategicsec@ubuntu:~/toolz/Burp$ wget http://blog.ombrepixel.com/public/BurpExtender-w3
af.py
--2013-09-21 16:58:08--  http://blog.ombrepixel.com/public/BurpExtender-w3af.py
Resolving blog.ombrepixel.com (blog.ombrepixel.com)... 217.70.184.34, 2001:4b98:b:a::b10
9
Connecting to blog.ombrepixel.com (blog.ombrepixel.com)|217.70.184.34|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6620 (6.5K) [application/octet-stream]
Saving to: `BurpExtender-w3af.py'

100%[======================================>] 6,620       --.-K/s   in 0.02s

2013-09-21 16:58:08 (291 KB/s) - `BurpExtender-w3af.py' saved [6620/6620]

strategicsec@ubuntu:~/toolz/Burp$ cp BurpExtender-w3af.py Lib/
BurpExtender.py        BurpExtender$py.class  .BurpExtender.py.swp
strategicsec@ubuntu:~/toolz/Burp$ cp BurpExtender-w3af.py Lib/BurpExtender.py
strategicsec@ubuntu:~/toolz/Burp$
```

Get error in kb:

```
File  Edit  View  Search  Terminal  Help
strategicsec@ubuntu:~/toolz/Burp$ ./suite.sh
BurpExtender.py needs to be in a folder listed below:
['/home/strategicsec/toolz/Burp/Lib', '/home/strategicsec/toolz/Burp/jython-standalone-2
.5.3.jar/Lib', '__classpath__', '__pyclasspath__/']
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/home/strategicsec/toolz/Burp/Lib/BurpExtender.py", line 27, in <module>
    import core.data.kb.knowledgeBase as kb
ImportError: No module named core
Deleting temporary files - please wait ... done.
strategicsec@ubuntu:~/toolz/Burp$
```

Edit BurpExtender.py and change knowledgeBase to knowledge_base:

```
File  Edit  View  Search  Terminal  Help
# BurpExtender.py - use w3af plugins (http://w3af.sourceforge.net) with Burp Suite
# Author: David Robert david@ombrepixel.com
# Version 0.1 09/09/2010

# ========= You need to edit below =============

# Here you define the name of the plugins you want (category.plugin)
plugins = ['grep.domXss',  'grep.error500', 'grep.errorPages', 'grep.feeds',
           'grep.fileUpload','grep.hashFind', 'grep.httpAuthDetect',
           'grep.privateIP', 'grep.ssn', 'grep.strangeHeaders',
           'grep.strangeHTTPCode', 'grep.strangeReason', 'grep.svnUsers',
           'grep.wsdlGreper']

# Here you should define the location of your w3af installation
w3afPath="/home/strategicsec/toolz/w3af"
# Example for Unix "/usr/local/w3af/w3af"
# ========= You need to edit above =============

import sys
import urllib2
sys.path.append(w3afPath)

# Burp Suite related
from burp import IBurpExtender

# w3af srelated
import core.data.kb.knowledge_base as kb
from core.data.parsers.urlParser import getPathQs
from core.data.url.httpResponse import httpResponse
from core.controllers.misc.factory import factory
"Lib/BurpExtender.py" 159L, 6614C written                     27,31         Top
```

Try again:

```
File  Edit  View  Search  Terminal  Help
strategicsec@ubuntu:~/toolz/Burp$ ./suite.sh
BurpExtender.py needs to be in a folder listed below:
['/home/strategicsec/toolz/Burp/Lib', '/home/strategicsec/toolz/Burp/jython-standalone-2
.5.3.jar/Lib', '__classpath__', '__pyclasspath__/']
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/home/strategicsec/toolz/Burp/Lib/BurpExtender.py", line 27, in <module>
    import core.data.kb.knowledgeBase as kb
ImportError: No module named knowledgeBase
Deleting temporary files - please wait ... done.
strategicsec@ubuntu:~/toolz/Burp$ vim /home/strategicsec/toolz/Burp/Lib/BurpExtender.py
strategicsec@ubuntu:~/toolz/Burp$ ./suite.sh
BurpExtender.py needs to be in a folder listed below:
['/home/strategicsec/toolz/Burp/Lib', '/home/strategicsec/toolz/Burp/jython-standalone-2
.5.3.jar/Lib', '__classpath__', '__pyclasspath__/']
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/home/strategicsec/toolz/Burp/Lib/BurpExtender.py", line 27, in <module>
    import core.data.kb.knowledge_base as kb
  File "/home/strategicsec/toolz/w3af/core/data/kb/knowledge_base.py", line 77
SyntaxError: 'with' will become a reserved keyword in Python 2.6
Deleting temporary files - please wait ... done.
strategicsec@ubuntu:~/toolz/Burp$
```

Ah, get beta newest jython:

```
strategicsec@ubuntu:~/toolz/Burp$ ls
BurpExtender-w3af.py  burppython_v0.1.zip         Demo                          Lib             README.txt  suite.bat   suite.sh
burppython.jar        burpsuite_free_v1.6.31.jar  jython-standalone-2.7.1b2.jar  python-logo.gif  src         suite.bat~  suite.sh~
strategicsec@ubuntu:~/toolz/Burp$
```

Now problems with sqlite3:

```
File  Edit  View  Search  Terminal  Help
strategicsec@ubuntu:~/toolz/Burp$ ls
BurpExtender-w3af.py     Demo                        Lib              src
burppython.jar           jython-standalone-2.5.3.jar  python-logo.gif  suite.bat
burpsuite_free_v1.5.jar  jython-standalone-2.7-b1.jar  README.txt       suite.sh
strategicsec@ubuntu:~/toolz/Burp$ vim suite.sh
strategicsec@ubuntu:~/toolz/Burp$ ./suite.sh
BurpExtender.py needs to be in a folder listed below:
['/home/strategicsec/toolz/Burp/Lib', '/home/strategicsec/toolz/Burp/jython-standalone-2
.7-b1.jar/Lib', '__classpath__', '__pyclasspath__/']
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/home/strategicsec/toolz/Burp/Lib/BurpExtender.py", line 27, in <module>
    import core.data.kb.knowledge_base as kb
  File "/home/strategicsec/toolz/w3af/core/data/kb/knowledge_base.py", line 28, in <modu
le>
    from core.data.db.dbms import get_default_persistent_db_instance
  File "/home/strategicsec/toolz/w3af/core/data/db/dbms.py", line 26, in <module>
    import sqlite3
ImportError: No module named sqlite3
```

Giving up, would take more time to implement this is jython.

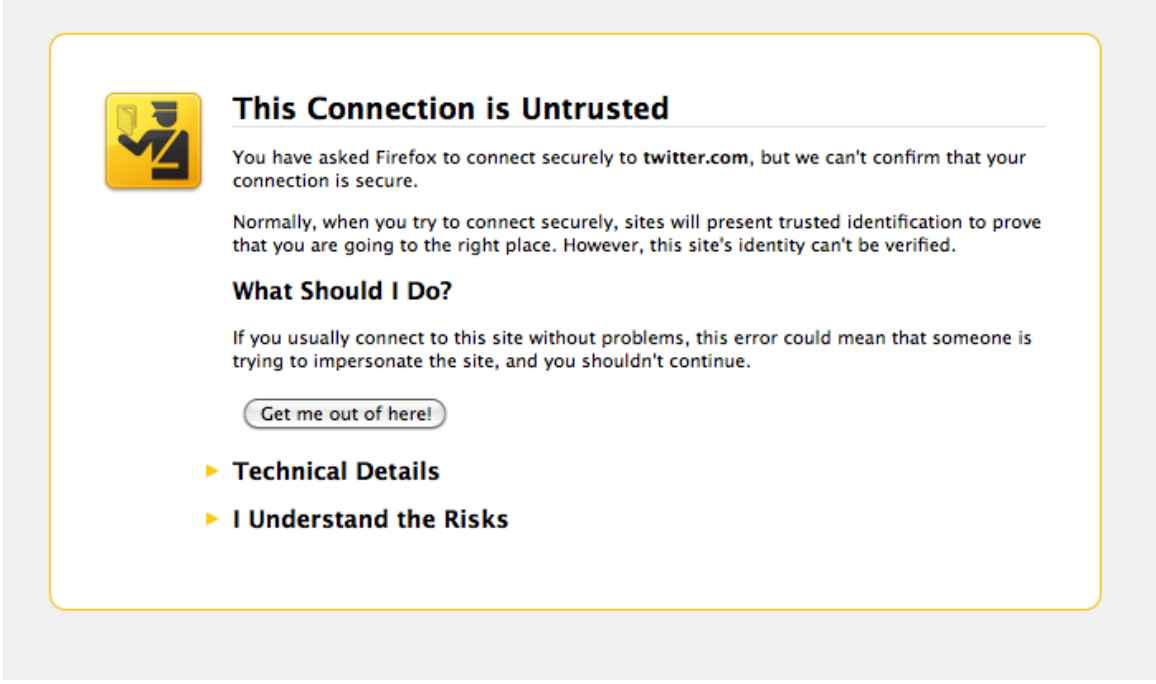
Section 4: Burp Suite & Mobile (Additional Material)

NOTE: Although we will not be covering mobile in this class I have added this section to the course to give you some extra material. I apologize for the inconvenience of not being able to run the 1st day of the class due to isses with the payment system and webinar software, but I'll have a makeup class on next Sunday.

Lab 13: Burp on iPhone

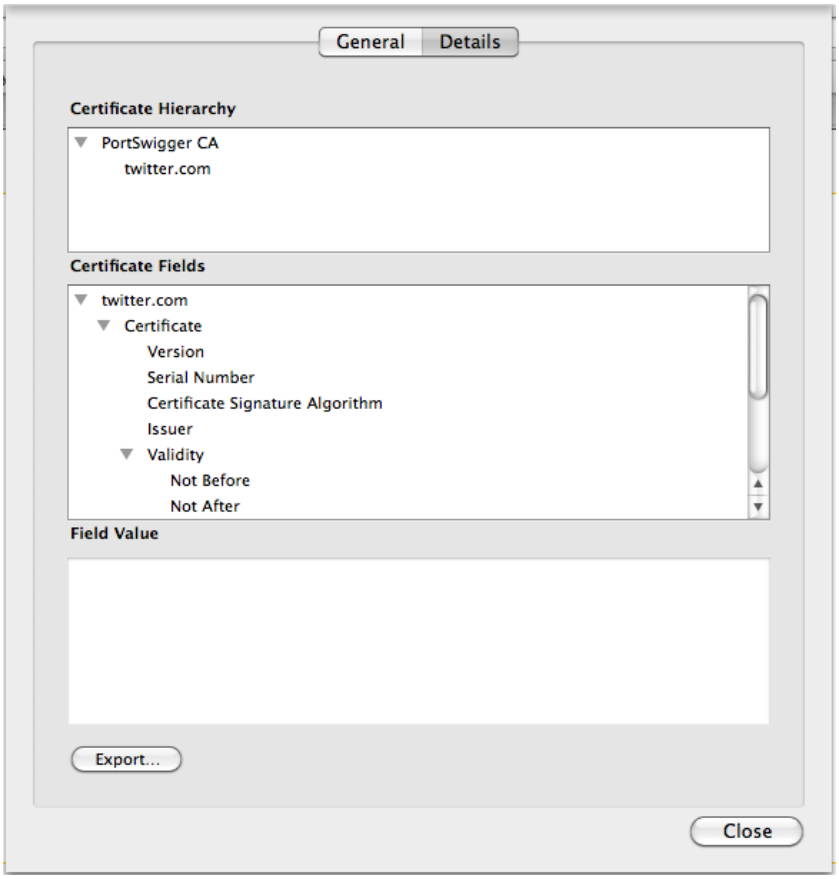
In this lab we will see how to use the burp proxy with your iPhone.

Start the BurpSuite and and export the certificate. This can be done by visiting a page using https, for example: https://twitter.com



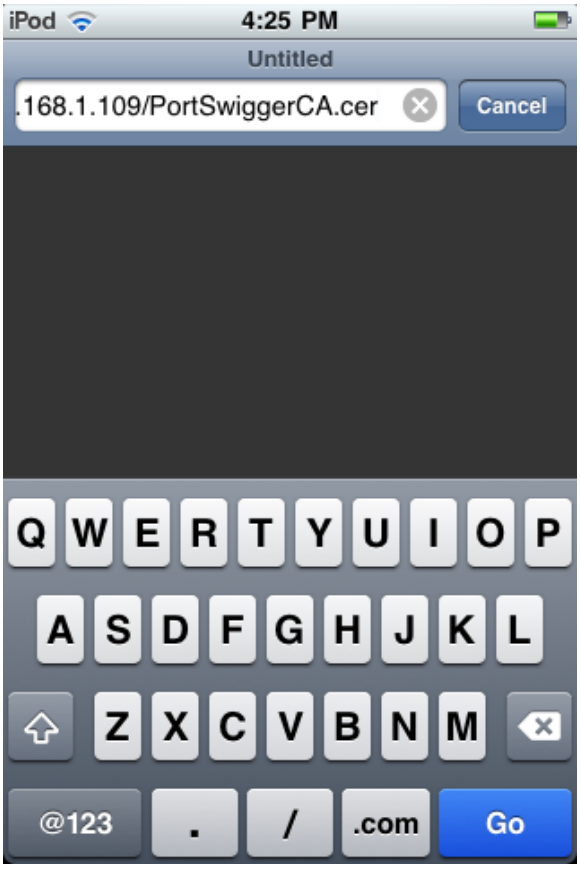
Click **I Understand the Risks** -> **Add Exception**

Click **View** -> **Details**

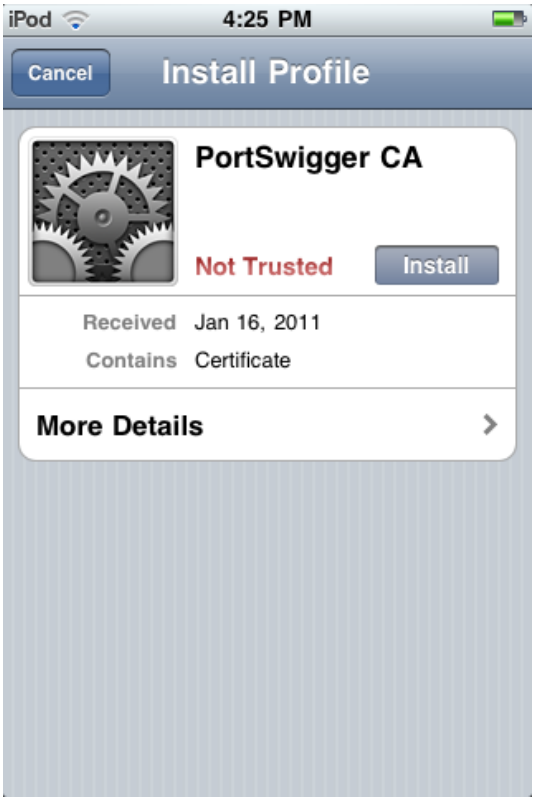


Now **Export** the **PortSwigger CA** certificate. When you are asked to save it, save it as a **.cer** file. This is what is accepted via the iPhone.

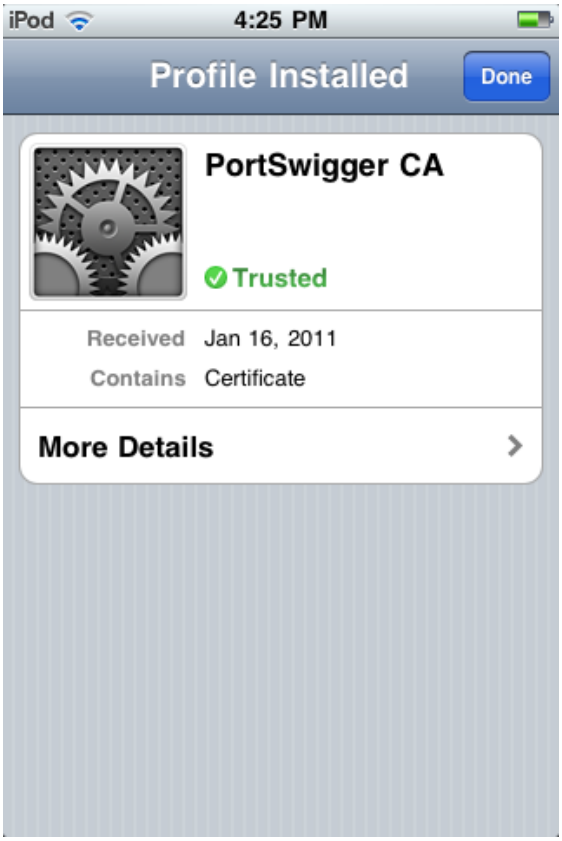
Once you have the certificate in hand, start a webserver to “serve up” the new certificate. Back on the iPhone, browse to the webserver where you are hosting the new certificate.



You should now be prompted to install the new certificate.



After installation, the certificate now becomes trusted.



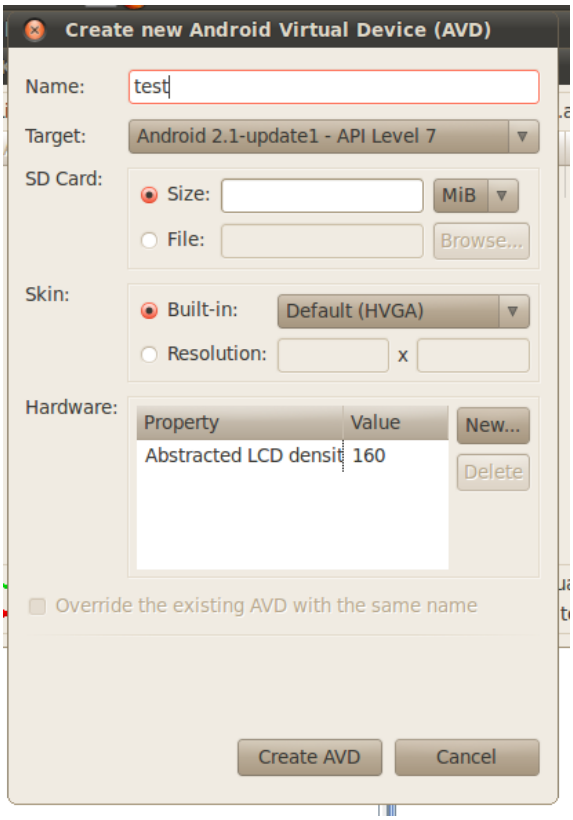
With the new certificate installed, you have to setup your proxy settings. This can be done by going into **Settings** -> **WiFi** -> then clicking the blue arrow to the right of your wifi network. Set the IP address as the IP of the machine that is running Burp.

You can now run your iPhone apps through a burp proxy!

Lab 14: Burp with Android

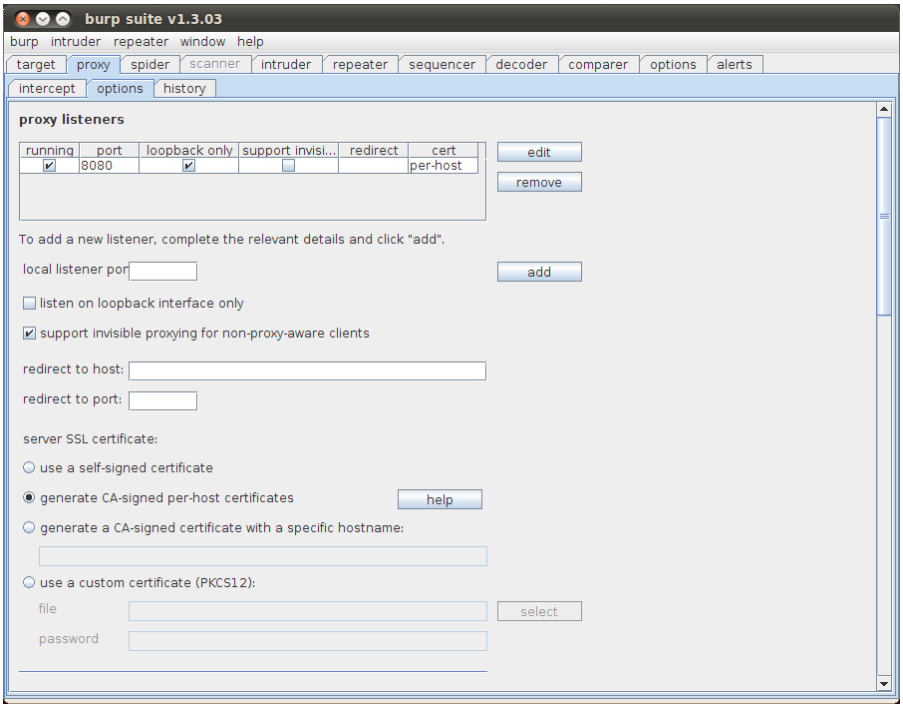
For this lab, we will be using the Android SDK to create our environment.

Create a new AVD using **Android 2.1 – API Level 7**



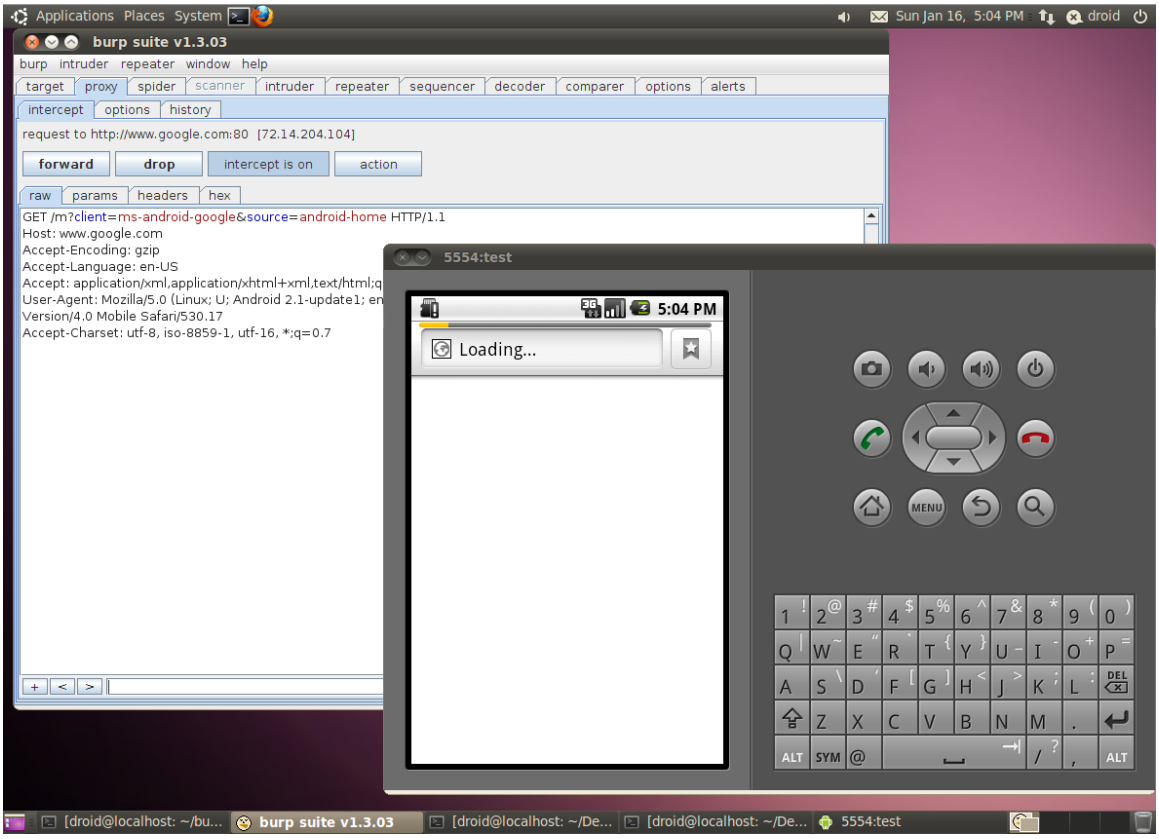
Once, you've given it a name and set your Target information. Click **Create AVD**

Start **BurpSuite** and change its proxy from **Listen on loopback interface only** to **Support invisible proxy for non-aware clients**



Now start the AVD with the following flags. This will start the AVD named **test** and use the following as it's http proxy server

./emulator –avd test –http-proxy http://127.0.0.1:8080


Applications Places System
Sun Jan 16, 5:04 PM droid
burp suite v1.3.03
burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
intercept options history
request to http://www.google.com:80 [72.14.204.104]
forward drop intercept is on action
raw params headers hex
GET /m?client=ms-android-google&source=android-home HTTP/1.1
Host: www.google.com
Accept-Encoding: gzip
Accept-Language: en-US
Accept: application/xml,application/xhtml+xml,text/html;q
User-Agent: Mozilla/5.0 (Linux; U; Android 2.1-update1; en
Version/4.0 Mobile Safari/530.17
Accept-Charset: utf-8, iso-8859-1, utf-16, *;q=0.7
5554:test
5:04 PM
Loading...
MENU
[droid@localhost: ~/bu...
burp suite v1.3.03
[droid@localhost: ~/De...
[droid@localhost: ~/De...
5554:test

Lab 15: SSL issues and Android

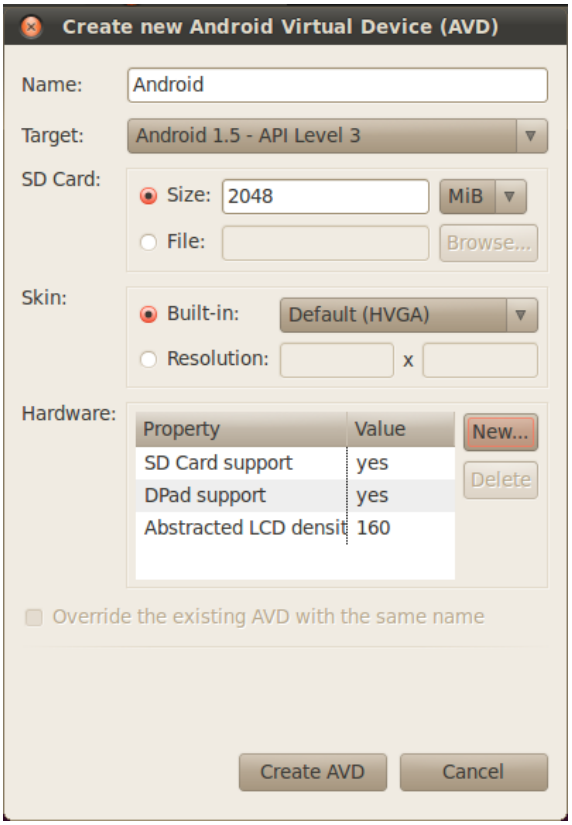
In the previous lab, if you messed around with Burp and Android a bit more. You will find issues as you start messing with some of the applications and web pages. Any site that is using SSL will error out because the certificates no longer match. Well, in this lab we will show you how to fix that. A prerequisite for this lab is to get another **PortSwiggerCA** certificate. This time save it as **PortSwiggerCA.crt** (see an above lab for a tutorial on this).

Now that you have the certificate, we have another problem we have to contend with. Our emulator does not have a marketplace. We can fix this with a simple tweak to our AVD.

Visit the following link and download the **system image** that matches your AVD. In our case, we will be working with the **1.5** image.

http://developer.htc.com/google-io-device.html#s3

Once you've downloaded the file, create an AVD with the following settings in your Android emulator.



Click **Create AVD** and unzip the system image file you've just downloaded. We need to copy the **system.img** file into our **~/.android/Android.avd/** directory.

```
droid@localhost:~/Downloads$ cp system.img /home/droid/.android/avd/Android.avd/
droid@localhost:~/Downloads$
```

With the system image in place, we need to edit the certificates on the device. Use the Android emulator to start the device.

With the new emulator started, navigate to the following URL and download the **.jar** file.

http://wiki.cacert.org/ImportRootCert#Android_Phones

This **.jar** file needs to be placed in the following directory (Ubuntu):

/usr/lib/jvm/java-6-sun-1.6.0.22/jre/lib/ext/

Next we have to grab the **cacerts.bks** from the phone:

./adb pull /system/etc/security/cacerts.bks cacerts.bks

Now we need to import the **PortSwiggerCA** cert into **cacerts.bks**:

```
droid@localhost:~/Desktop/android-sdk-linux_86/platform-tools$ keytool -keystore cacerts.bks -storetype BKS -p
rovider org.bouncycastle.jce.provider.BouncyCastleProvider -storepass changeit -importcert -trustcacerts -alia
s PortSwiggerCA -file /home/droid/Desktop/PortSwiggerCA.crt
Owner: CN=PortSwigger CA, OU=PortSwigger CA, O=PortSwigger, L=PortSwigger, ST=PortSwigger, C=PortSwigger
Issuer: CN=PortSwigger CA, OU=PortSwigger CA, O=PortSwigger, L=PortSwigger, ST=PortSwigger, C=PortSwigger
Serial number: 4d336899
Valid from: Sun Jan 16 16:52:25 EST 2011 until: Sat Jan 11 16:52:25 EST 2031
Certificate fingerprints:
         MD5:  F7:CE:6E:49:B9:A0:A6:7D:A1:36:6A:96:30:DD:C8:E2
         SHA1: 69:07:26:EF:83:8B:E8:15:FD:30:BE:58:BF:58:2D:9B:E4:A5:69:C0
         Signature algorithm name: SHA1withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:0
]

#2: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: EF 50 55 44 41 A4 AA AB   63 76 00 35 E8 DA 21 D9  .PUDA...cv.5..!.
0010: 4D 60 13 27                                        M`.'
]
]

Trust this certificate? [no]:  y
Certificate was added to keystore
```

keytool –keystore cacerts.bks –storetype BKS –provider org.bouncycastle.jce.provider .BouncyCastleProvider –storepass changeit –importcert –trustcacerts –alias PortSwiggerCA –file /home/droid/Desktop/PortSwiggerCA.crt

When you are prompted, make sure that you **Trust this certificate**. Once this is completed **remount** the device.

./adb remount

We are almost done with the setup. Unfortunately, 1.5 has issues a few issues. The first is that it runs out of space quickly. Close the emulator and start it with the following flags:

./emulator –avd Android –partition-size 512

The other issue is that the **/system** directory is currently **read-only**. We need to remount this partition as **read/write**, this can be done by using **ADB** to "hop" onto the device.

```
droid@localhost:~/Desktop/android-sdk-linux_86/platform-tools$ ./adb shell
# mount
rootfs / rootfs ro 0 0
tmpfs /dev tmpfs rw,mode=755 0 0
devpts /dev/pts devpts rw,mode=600 0 0
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
tmpfs /sqlite_stmt_journals tmpfs rw,size=4096k 0 0
/dev/block/mtdblock0 /system yaffs2 ro 0 0
/dev/block/mtdblock1 /data yaffs2 rw,nosuid,nodev 0 0
/dev/block/mtdblock2 /cache yaffs2 rw,nosuid,nodev 0 0
/dev/block//vold/179:0 /sdcard vfat rw,dirsync,nosuid,nodev,noexec,uid=1000,gid=
1000,fmask=0711,dmask=0700,allow_utime=0022,codepage=cp437,iocharset=iso8859-1,s
hortname=mixed,utf8 0 0
```

Issuing the **mount** command, we can see all of the different mounted partitions. The only one we are really worried about at the moment is **/system**. Issue the following command to remount it as **read/write**.

```
# mount -o rw,remount -t yaffs2 /dev/block/mtdblock0 /system
# mount
rootfs / rootfs ro 0 0
tmpfs /dev tmpfs rw,mode=755 0 0
devpts /dev/pts devpts rw,mode=600 0 0
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
tmpfs /sqlite_stmt_journals tmpfs rw,size=4096k 0 0
/dev/block/mtdblock0 /system yaffs2 rw 0 0
/dev/block/mtdblock1 /data yaffs2 rw,nosuid,nodev 0 0
/dev/block/mtdblock2 /cache yaffs2 rw,nosuid,nodev 0 0
/dev/block//vold/179:0 /sdcard vfat rw,dirsync,nosuid,nodev,noexec,uid=1000,gid=
1000,fmask=0711,dmask=0700,allow_utime=0022,codepage=cp437,iocharset=iso8859-1,s
hortname=mixed,utf8 0 0
```

You will now notice that the **/system** partition is now **read/write**. Now we can push the certificate back onto the machine. Exit out of the shell and issue the following command:

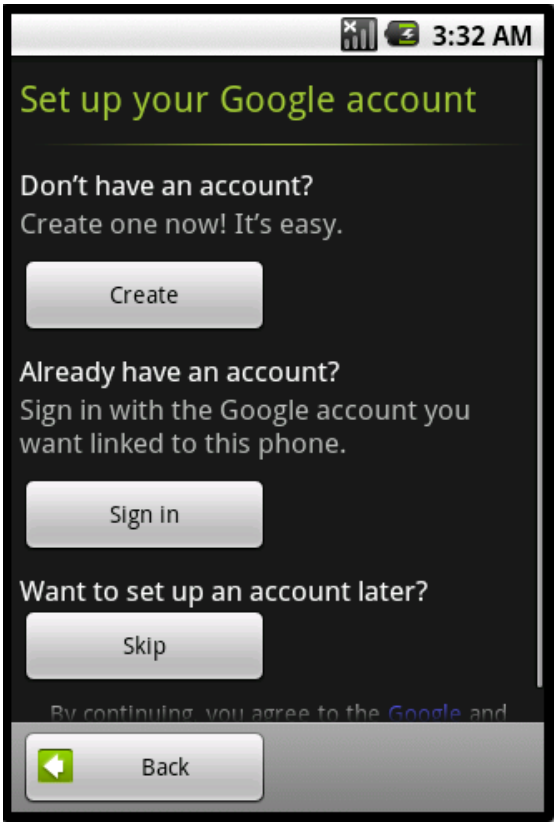
./adb push cacerts.bks /system/etc/security/

```
droid@localhost:~/Desktop/android-sdk-linux_86/platform-tools$ ./adb push cacerts.bks /system/etc/security
1281 KB/s (52650 bytes in 0.040s)
droid@localhost:~/Desktop/android-sdk-linux_86/platform-tools$
```

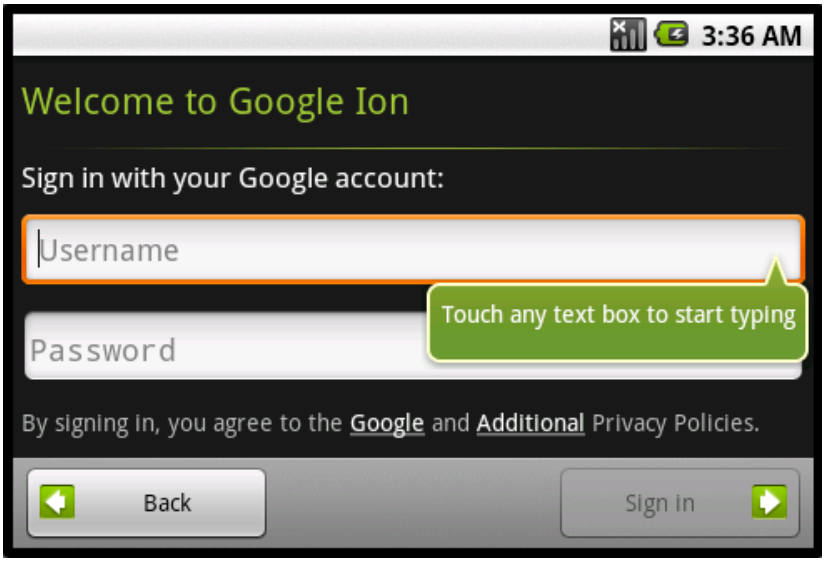
Now close, the emulator one more time. This time we will proxy it's traffic through burp.

./emulator –avd Android –http-proxy http://127.0.0.1:8080

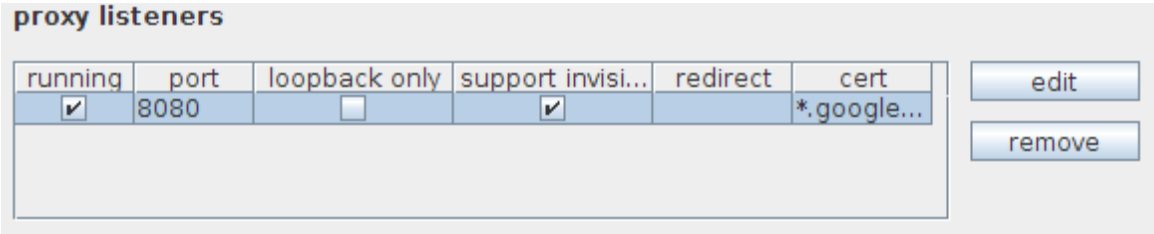
In order to use the Droid device now (because we have the marketplace installed) you have to fill in all of your google information.



When it prompts you to slide the device, hit **CTRL + F11** to do this.



After you've filled in your information and clicked **Sign in**, you'll notice that it errored out. This is because the certificates do not match. Edit burp to generate its own certificate for the ***.google.com** domain and you will able to authenticate!



This technique should work with other applications as well!

Lab 16: Burping Android

Setting up the Virtual Machine

Download the VM

- http://dl.dropbox.com/u/1791263/MobileAppVM.rar

Turn on the Virtual Machine (MobileAppVM)

-Username: Joe
-Password: Secure123

Download Burp Suite

-http://www.portswigger.net/burp/download.html

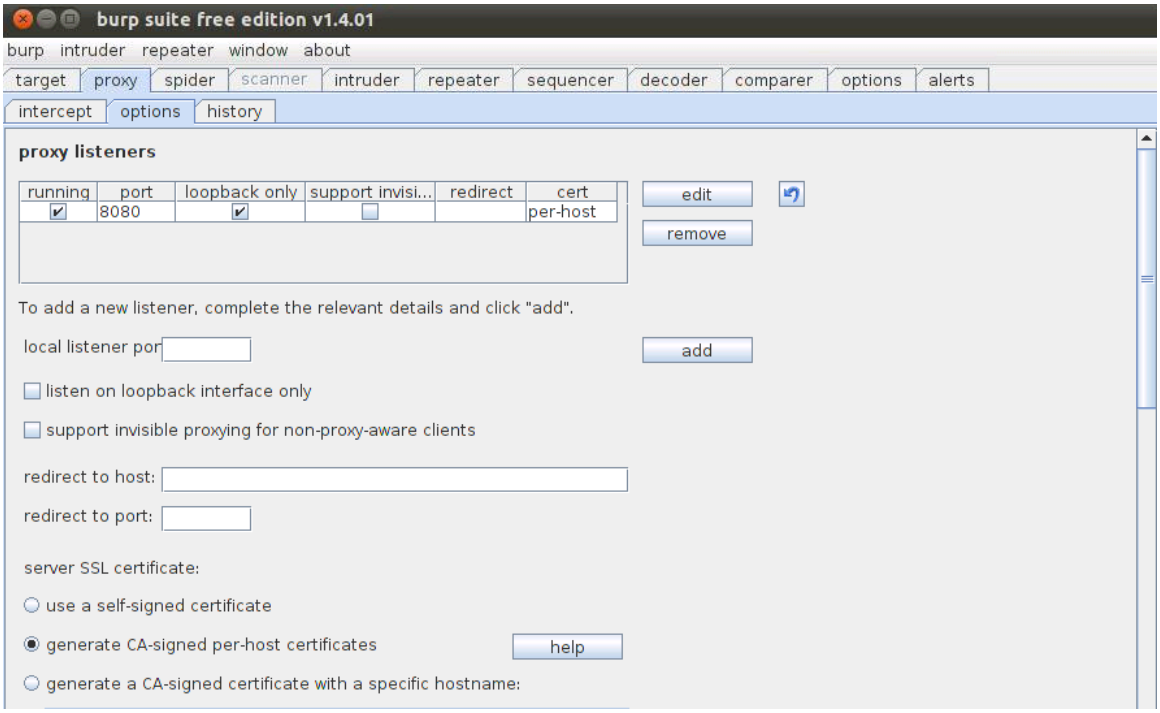
Unzip it (via terminal)

-cd ~/Desktop
-unzip burpsuite_v1.4.01.zip
-cd burpsuite_v1.4.01

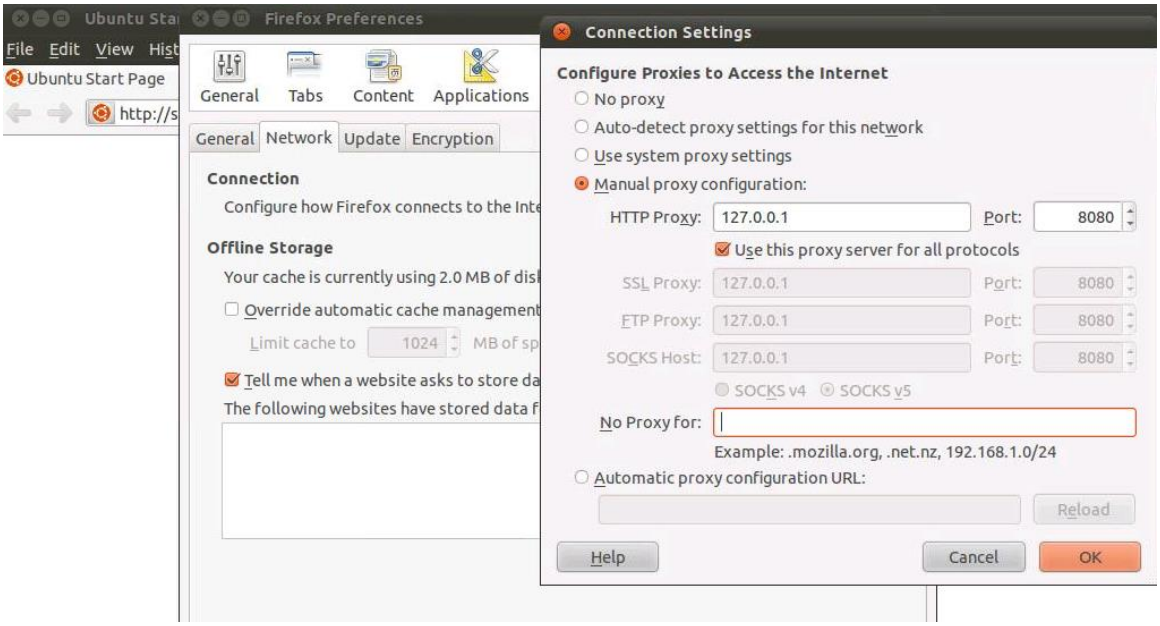
```
joe@StrategicAssesment:~$ cd ~/Desktop/
joe@StrategicAssesment:~/Desktop$ unzip burpsuite_v1.4.01.zip
Archive:  burpsuite_v1.4.01.zip
   creating: burpsuite_v1.4.01/
  inflating: burpsuite_v1.4.01/burpsuite_v1.4.01.jar
  inflating: burpsuite_v1.4.01/readme - running burp.txt
 extracting: burpsuite_v1.4.01/suite.bat
joe@StrategicAssesment:~/Desktop$ cd burpsuite_v1.4.01/
joe@StrategicAssesment:~/Desktop/burpsuite_v1.4.01$
```

-Setup Burp

-java –jar burpsuite_v1.4.01.jar
-Click on OK for the openJDK warning
-Accept the agreement
-Click the “Proxy” tab
-Click the “Options” sub tab
-Ensure that burp is configured to “generate CA-signed per-host certificates”



-Configure Firefox
- -Click “edit”
- -Click “preferences”
- -Click the “Advanced” tab
- -Click the “Network” sub tab
- -Click the connection “settings” button
- -Click “manual proxy configuration”
  - -set it to 127.0.0.1 port 8080
  - -Check “Use this proxy server for all protocols”
- -Remove both the “localhost, 127.0.0.1” text from the “No Proxy For” line



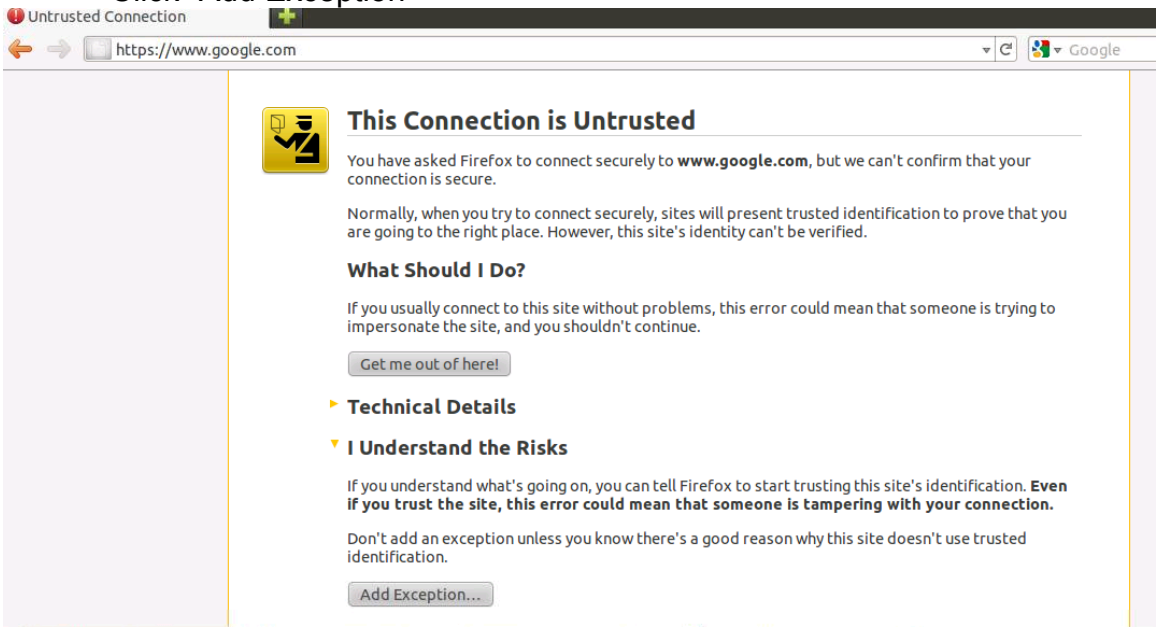
-Configure Firefox to use Burp as proxy and configure Burp's proxy listener to generate CA-signed per-host certificates.

-Visit any SSL-protected

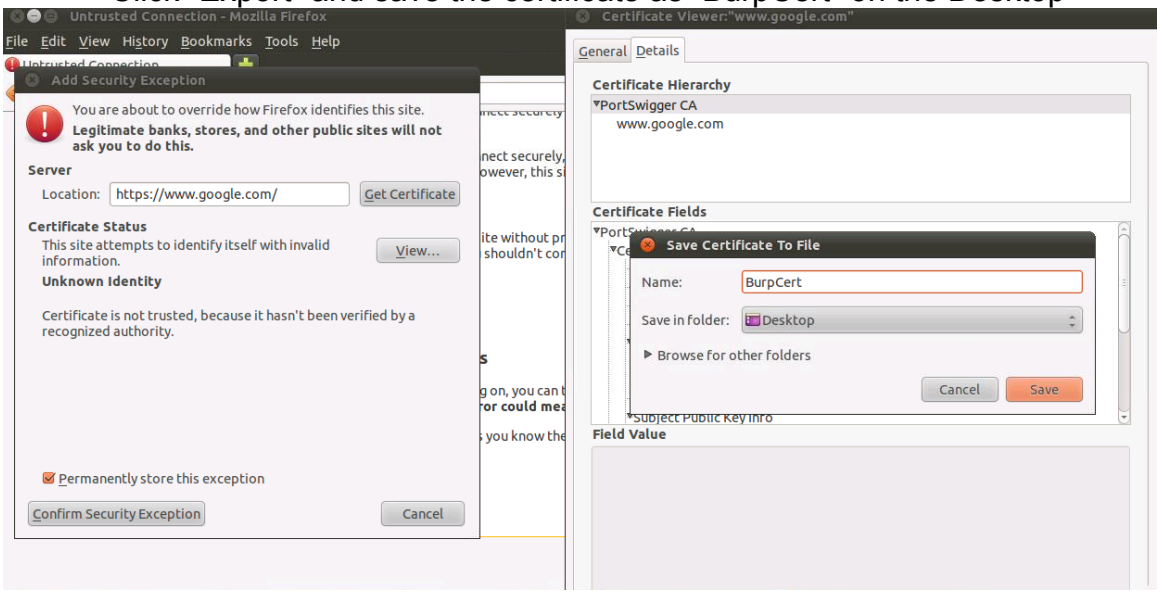
- https://www.google.com

-Expand the "I Understand the Risks" section

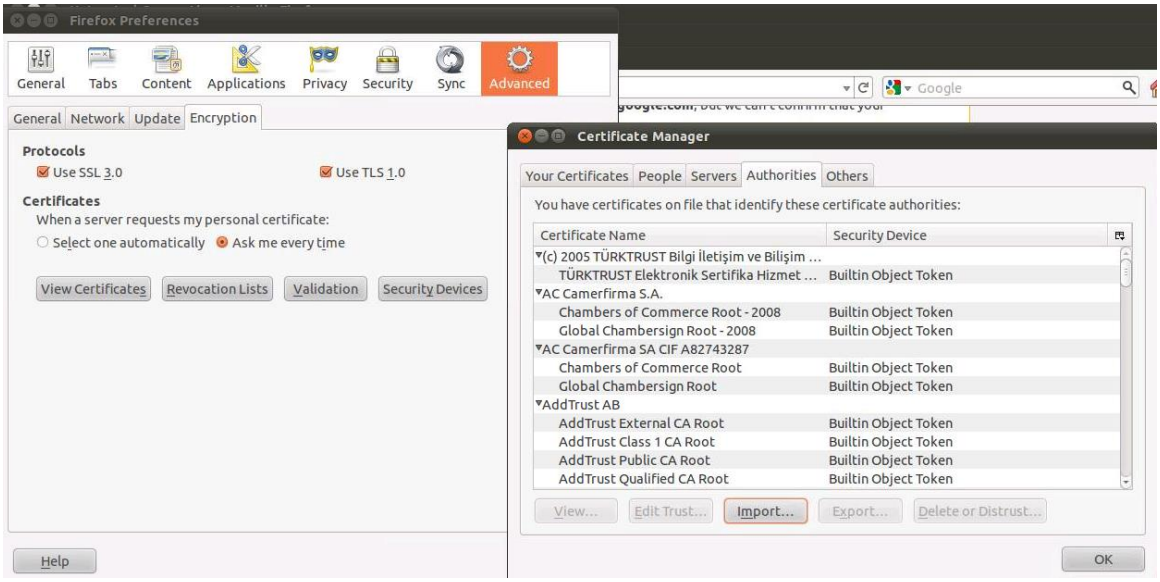
-Click "Add Exception"



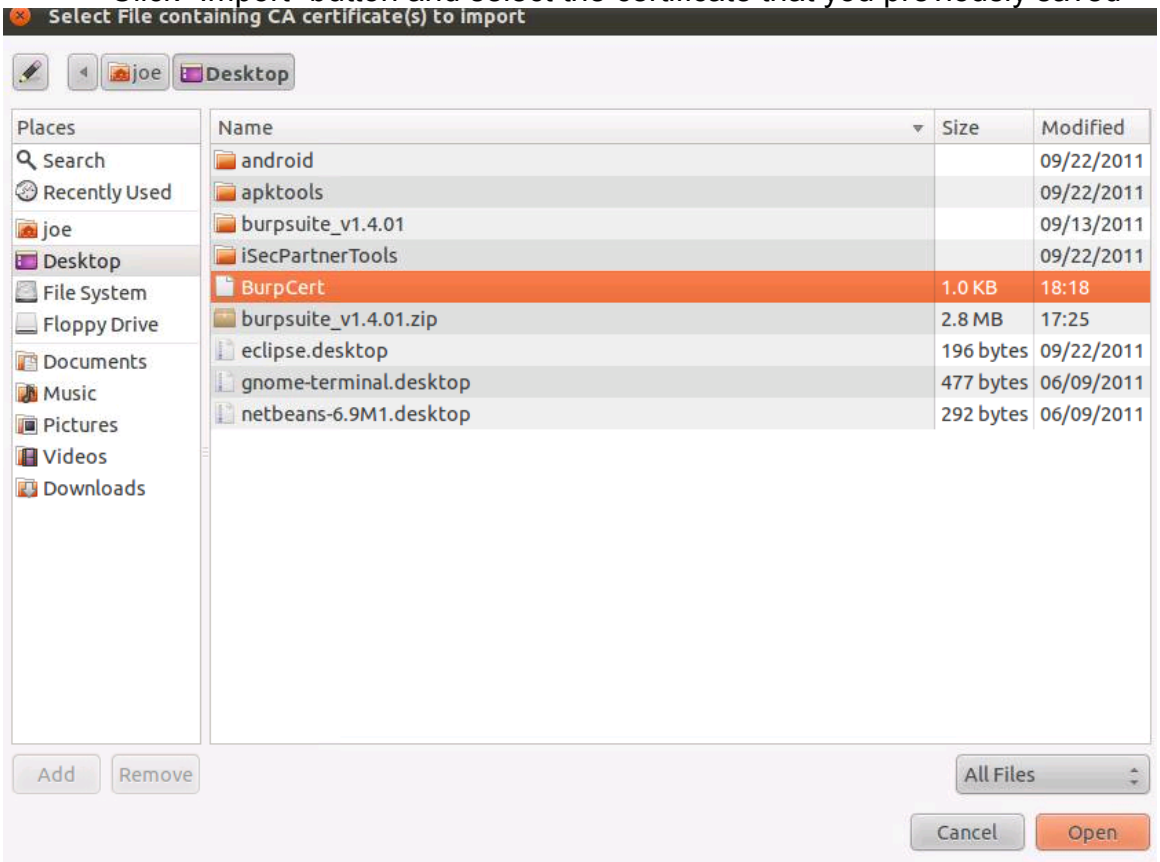
-Click “Get Certificate”
-Click “View”
-Select the root certificate in the tree (PortSwigger CA)
-Click “Export” and save the certificate as “BurpCert” on the Desktop



-Click “Close” on the Certificate Viewer dialog, and “Cancel” on th “Add Security Exception” dialog.
-Click on “Edit”
-Click on “Preferences”
-Click on “Advanced”
-Click on “Encryption” tab
-Click “View Certificates”
-Click “Authorities” tab



-Click “Import” button and select the certificate that you previously saved

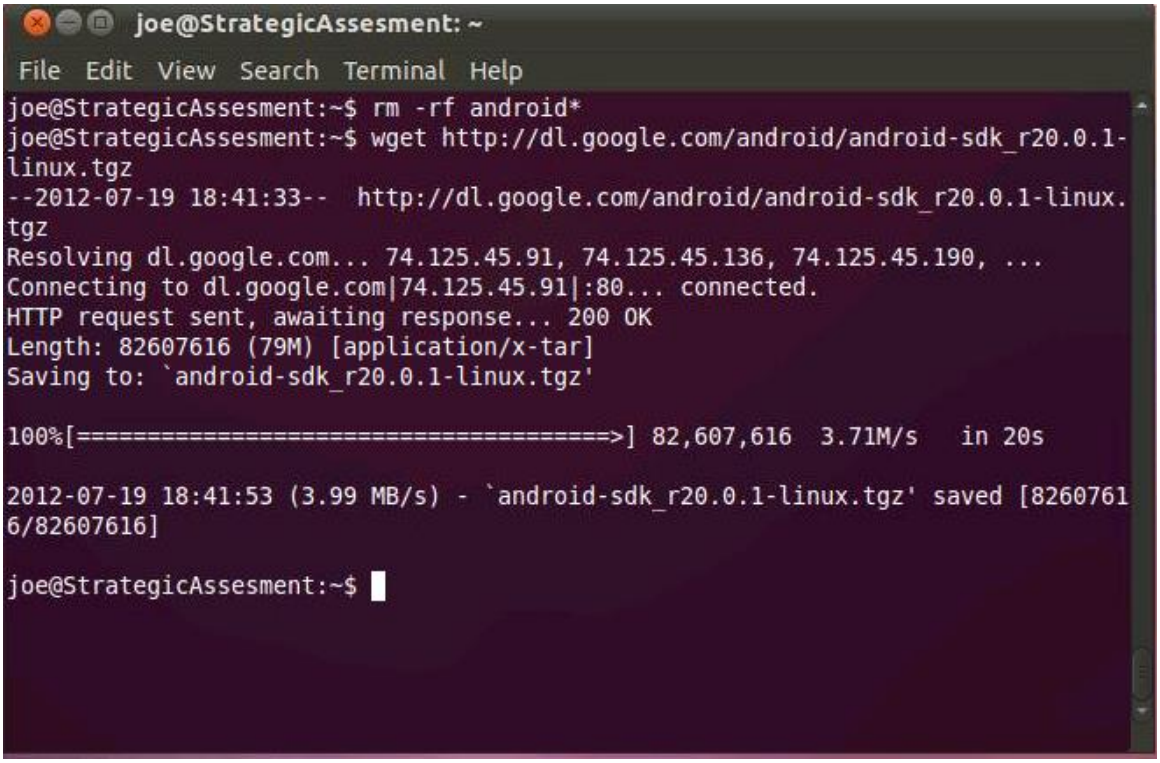


-On the “Downloading Certificate” dialog, check the box “Trust this CA to identify web sites” and click “OK”

-Close all dialogs and restart Firefox

Setup the Android SDK

-Remove old files
    -rm –rf android*
-Download the SDK
    -wget http://dl.google.com/android/android-sdk_r20.0.1-linux.tgz



-Extract the files

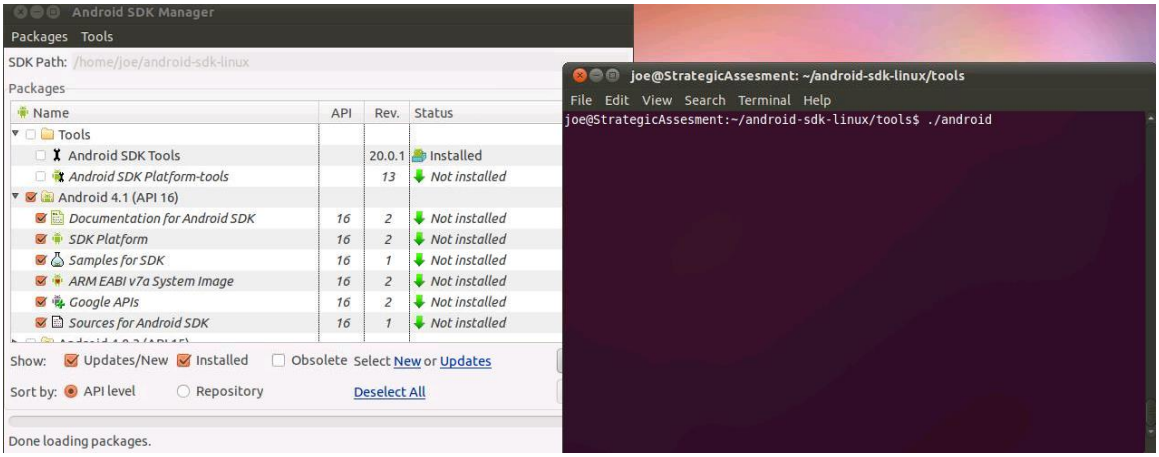
-tar –zxvf android-sdk_r20..0.1-linux.tgz

-Change into the tools directory

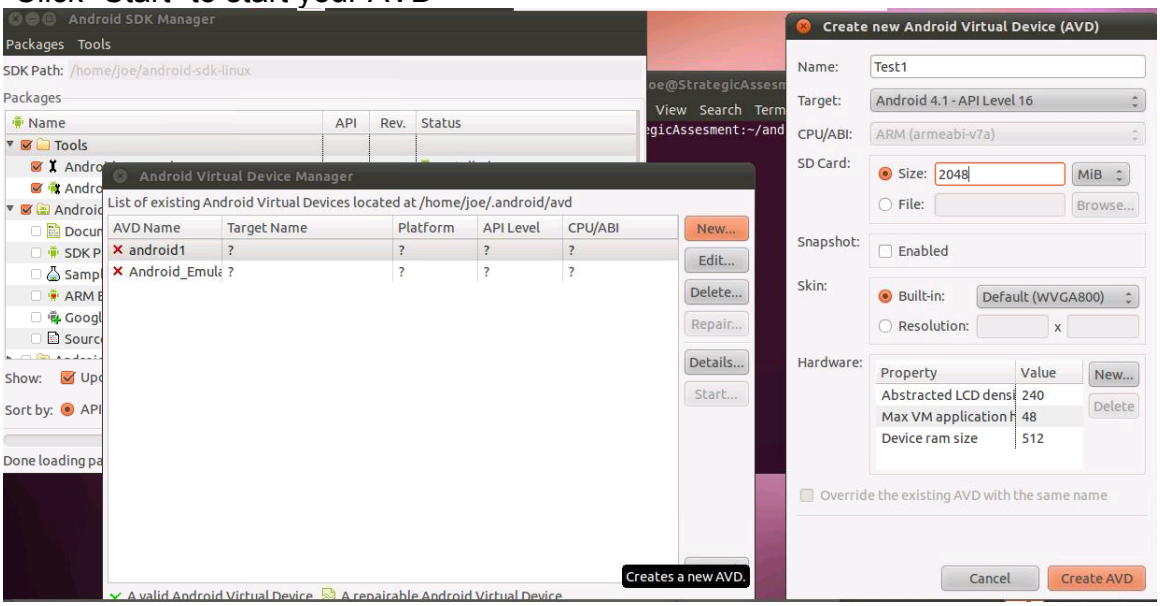
-cd android-sdk-linux/tools

-Start the SDK Manager

- ./android



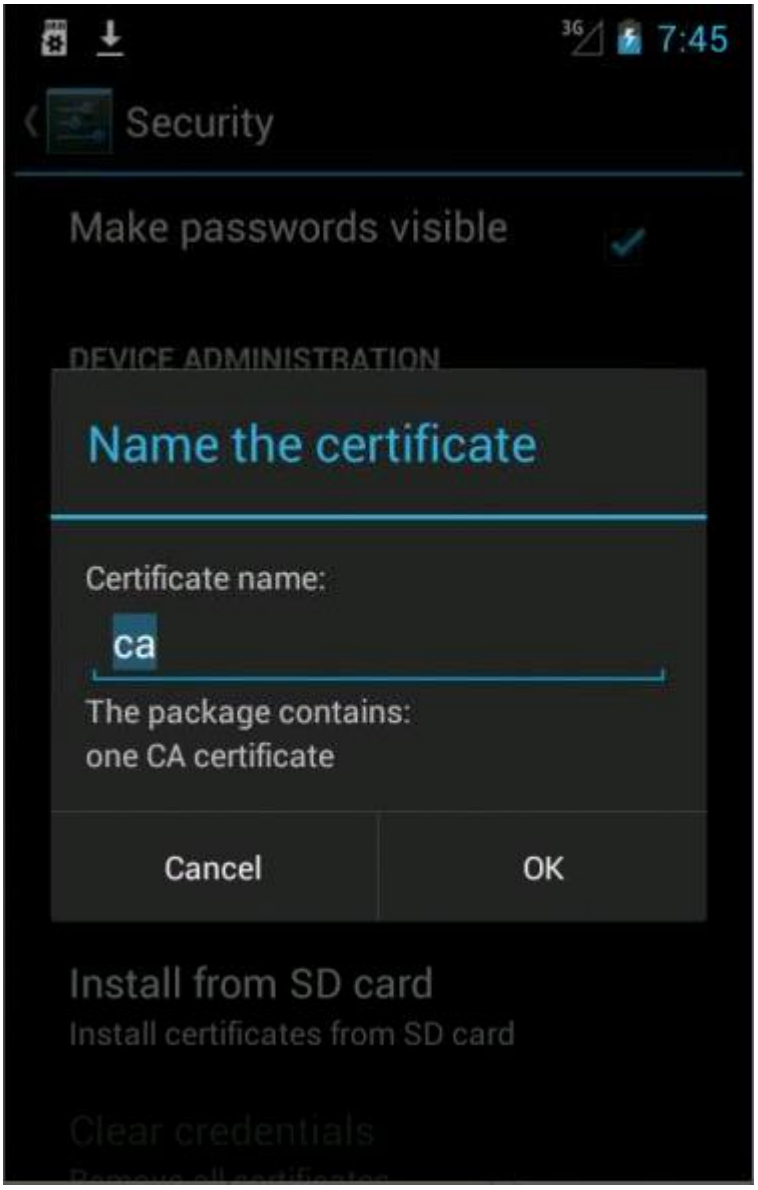
-From the SDK Manger window click the checkbox for Tools and Android SDK
-Click install packages in the lower right hand corner to install the tools
-From the file menu click “Tools”
-Click “Manage AVDs”
-Click “New”
-Type “Test1” in Name
-Select the default for target
-Set SD Card size to “2048”
-Click “Create AVD”
-Click “Start” to start your AVD



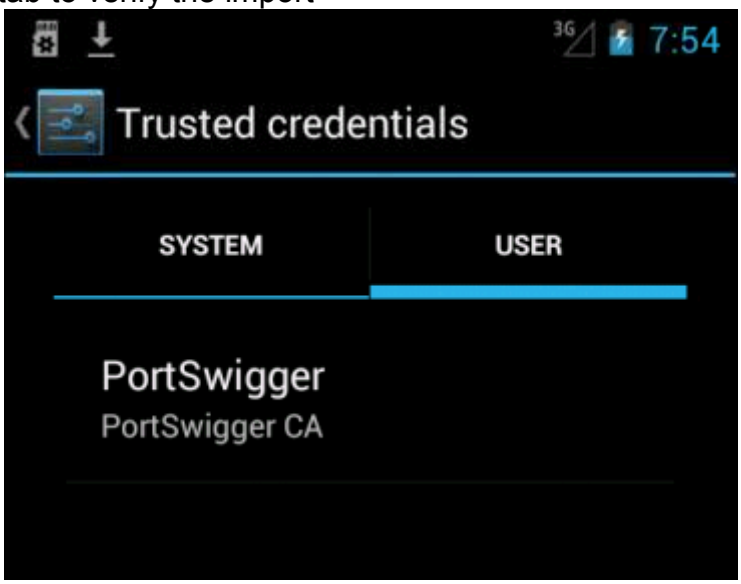
Setup Proxying in Burp (The new way, not the old way)

-Open a terminal and go to the desktop
    -cd ~/Desktop
-copy burp certificate
    -cp BurpCert ca.cer
-Start Python Simple HTTP Server
    -python –m SimpleHTTPServer
-In the Android Emulator navigate to the ca.cer we created
    -http://xxx.xxx.xxx.xxx:8000/ca.cer (xxx is the local machine IP)
    -A download will start for the ca.cer
-On your AVD go back to the home screen and open the app drawer
- Click on “Settings” (It’s an app)

-Click on “Security”
-Click “Install from SD Card”



-Click “OK”
-Click “OK” on the dialog box for setting a password
-Click “Pin”
-Enter “1234” twice
-Click “Trusted Credentials”
-Click “Users” tab to verify the import



-Turn off the Android emulator
-Install Mercurial from the terminal
    -sudo apt-get install –y mercurial
    -Password is Secure123
-Get android proxy
    -cd ~/Desktop
    -hg clone https://code.google.comp/androidproxy/
    -cd androidproxy

```
joe@StrategicAssesment:~$ cd ~/Desktop/
joe@StrategicAssesment:~/Desktop$ hg clone https://code.google.com/p/androidprox
y/
warning: code.google.com certificate with fingerprint c4:0c:c8:54:d7:c9:fc:5f:b5
:8b:38:53:31:6d:05:11:b4:53:9d:1b not verified (check hostfingerprints or web.ca
certs config setting)
destination directory: androidproxy
requesting all changes
adding changesets
adding manifests
adding file changes
added 5 changesets with 7 changes to 3 files
updating to branch default
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
joe@StrategicAssesment:~/Desktop$ cd androidproxy/
joe@StrategicAssesment:~/Desktop/androidproxy$
```

-Configure IPTables

-sudo iptables –F

-sudo iptables -t nat -A PREROUTING –p tcp --dport 80 –j REDIRECT --to-port 8007

- sudo iptables -t nat -A PREROUTING –p tcp --dport 443 –j REDIRECT --to-port 8007

-Start android proxy

-sudo Python main.py

```
joe@StrategicAssesment:~/Desktop/androidproxy$ sudo python main.py
AndroidProxy  ---   (C) Mathy Vanhoef (Made While Intern @ Ernst & Young)
This program comes with ABSOLUTELY NO WARRANTY.

DNS server will listen on localhost:65
HTTP Proxy will listen on localhost:8007

Physical device: Configure your computer as router and dns server and execute
        iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 80
07
        iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 8
007

Start emulator using command: emulator @AvdName -http-proxy http://localhost:800
7 -dns-server localhost

Don't forget to start your normal proxy on localhost:8080

        ==== Android Proxy Up and Running ====
```

-Restart the Android emulator (AVD) from the terminal

-cd ~/android-sdk-linux/tools

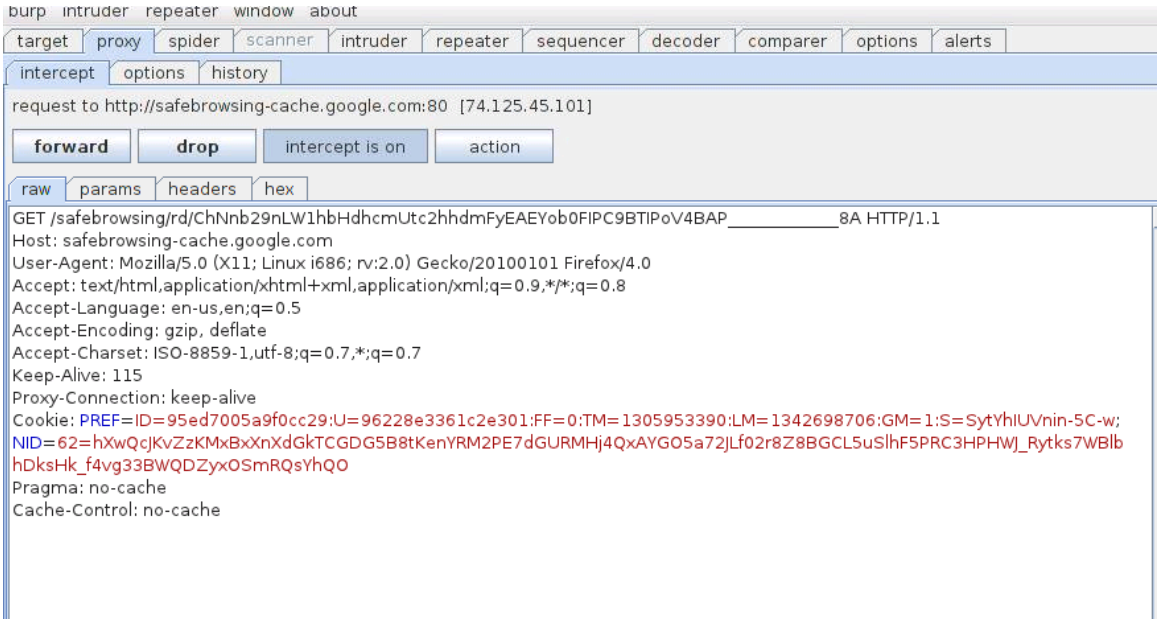
-emulator –avd Test1 -http-proxy http://127.0.0.1:8007 –dns-server localhost

-Verify that Android emulators web browser uses burp

-Open Android Browser

-Navigate to https://www.google.com

-Switch over to burp and notice that the traffic is being captured



-Forward the request and check back with your Android Emulator


8:20
https://www.google.com/
Google
Images
Places
News
more
security rookies
The Security Rookies
security-rookies.com/
The Security Rookies – aka – "the rookies" are a group of people that are interested in learning ...
the Security Rookies Website
security-rookies.com/2011/08/14/we...
Welcome to the Security Rookies Website. Aug 14. Posted by ...
About
security-rookies.com/about/
About. This website was developed by Strategic Security with goal of ...