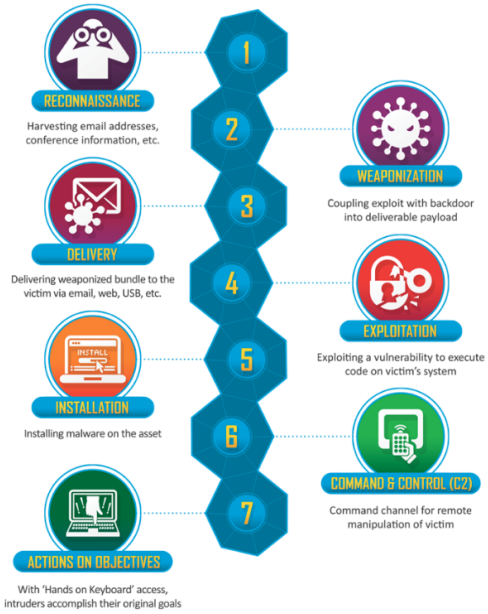
Advanced-Persistent-Threat

APT Prologue

In this scenario, reports of the below graphic come in from your user community when they visit the Wayne Enterprises website, and some of the reports reference "P01s0n1vy." In case you are unaware, P01s0n1vy is an APT group that has targeted Wayne Enterprises. Your goal, as Alice, is to investigate the defacement, with an eye towards reconstructing the attack via the Lockheed Martin Kill Chain.



IP address of web server:

There are 2 indexes:

```
* | stats count by index
```

index	count
botsv1	955807
main	5932

The 1st index does not seem to contain interesting data sources for our investigation:

```
* index=main
| stats count by source
| sort -count
```

source	count
stream:Splunk_HTTPURI	3708
stream:Splunk_HTTPStatus	686
stream:Splunk_HTTPClient	429
stream:Splunk_HTTPResponseTime	429
stream:Splunk_IP	247
stream:Splunk_Tcp	237
stream:Splunk_Udp	79
stream:Splunk_DNSIntegrity	40
stream:Splunk_DNSClientQueryTypes	36
stream:Splunk_DNSRequestResponse	23
stream:Splunk_DNSServerQuery	23
stream:Splunk_DNSServerResponse	23

As far as the second index:

```
* index=botsv1
| stats count by source
| sort -count
| head 10
```

source	count
WinEventLog:Microsoft-Windows-Sysmon/Operational	270597
stream:smb	151568
/var/log/suricata/eve.json	125584
WinEventLog:Security	87430
udp:514	80922
WinRegistry	74720
stream:ip	62083
stream:tcp	28291
stream:http	23936
C:3SVC1\u_ex160810.log	22401

It is now obvious that the interesting information is in the `botsv1` index.

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http
| stats count by src_ip
| sort -count
```

src_ip	count
40.80.148.42	20932
23.22.63.114	1236

IP address: `40.80.148.42`

What web scanner scanned the server?

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http src_ip="40.80.148.42"
| stats count by  src_headers
| sort -count
| head 3
```

Top 3 requests should Acunetix (Free Edition) scanning requests:

```
POST /joomla/index.php/component/search/ HTTP/1.1
Content-Length: 99
Content-Type: application/x-www-form-urlencoded
Cookie: ae72c62a4936b238523950a4f26f67d0=v7ikb3m59romokqmbiet3vphv3
Host: imreallynotbatman.com
Connection: Keep-alive
Accept-Encoding: gzip,deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like
Gecko) Chrome/41.0.2228.0 Safari/537.21
Acunetix-Product: WVS/10.0 (Acunetix Web Vulnerability Scanner - Free Edition)
Acunetix-Scanning-agreement: Third Party Scanning PROHIBITED
Acunetix-User-agreement: http://www.acunetix.com/wvs/disc.htm
Accept: */*

POST /joomla/index.php/component/search/ HTTP/1.1
Content-Length: 101
Content-Type: application/x-www-form-urlencoded
Cookie: ae72c62a4936b238523950a4f26f67d0=v7ikb3m59romokqmbiet3vphv3
Host: imreallynotbatman.com
Connection: Keep-alive
Accept-Encoding: gzip,deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like
Gecko) Chrome/41.0.2228.0 Safari/537.21
Acunetix-Product: WVS/10.0 (Acunetix Web Vulnerability Scanner - Free Edition)
Acunetix-Scanning-agreement: Third Party Scanning PROHIBITED
Acunetix-User-agreement: http://www.acunetix.com/wvs/disc.htm
Accept: */*

POST /joomla/index.php/component/search/ HTTP/1.1
Content-Length: 102
Content-Type: application/x-www-form-urlencoded
Cookie: ae72c62a4936b238523950a4f26f67d0=v7ikb3m59romokqmbiet3vphv3
Host: imreallynotbatman.com
Connection: Keep-alive
Accept-Encoding: gzip,deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like
Gecko) Chrome/41.0.2228.0 Safari/537.21
Acunetix-Product: WVS/10.0 (Acunetix Web Vulnerability Scanner - Free Edition)
```

```
Acunetix-Scanning-agreement: Third Party Scanning PROHIBITED
Acunetix-User-agreement: http://www.acunetix.com/wvs/disc.htm
Accept: */*
```

Asnwer: `acunetix`

What is the IP address of our web server?

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http src_ip="40.80.148.42"
| stats count by dest_ip
| sort -count
```

dest_ip	count
192.168.250.70	20931
192.168.250.40	1

Answer: `192.168.250.70`

What content management system is imreallynotbatman.com using?

From the previous POST requests sent, we can easily guess that the CMS is `Joomla`. We can confirm with the top 10 URIs:

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http src_ip="40.80.148.42"
| stats count by uri
| sort -count
| head 10
```

uri	count
/joomla/index.php/component/search/	14218
/joomla/index.php	798
/	517
/windows/win.ini	33
/joomla/media/jui/js/jquery-migrate.min.js	18
/joomla/media/jui/js/jquery-noconflict.js	18
/joomla/administrator/index.php	17
/joomla/media/jui/js/bootstrap.min.js	17
/joomla/media/system/js/html5fallback.js	13
/joomla/templates/protostar/js/template.js	13

Answer: `joomla`

What address is performing the brute-forcing attack against our website?

Requesting the HTTP methods shows a vast majority of POST requests:

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http src_ip="40.80.148.42"
| stats count by http_method
| sort -count
```

http_method	count
POST	15146
GET	5766
OPTIONS	5
CONNECT	1
PROPFIND	1
TRACE	1

A brute force attack involves POST requests. In addition, it involves a username and a password. Let's identify one of the requests:

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http src_ip="40.80.148.42"
http_method="POST" username
| table dest_content
| head 1
```

Result:

```
[REDACTED]

<form action="/joomla/administrator/index.php" method="post" id="form-login"
class="form-inline">

    [REDACTED]

    <input name="username" tabindex="1" id="mod-login-username" type="text"
class="input-medium" placeholder="Username" size="15" autofocus="true" />

    [REDACTED]

    <input name="passwd" tabindex="2" id="mod-login-password" type="password"
class="input-medium" placeholder="Password" size="15"/>

    [REDACTED]

    <button tabindex="3" class="btn btn-primary btn-block btn-large">
```

```
        <span class="icon-lock icon-white"></span> Log in
</button>

    [REDACTED]

    <input type="hidden" name="option" value="com_login"/>
    <input type="hidden" name="task" value="login"/>
    <input type="hidden" name="return" value="aW5kZXgucGhw"/>
    <input type="hidden" name="da4c70bcedf77f722881e18fb076b963" value="1" />
</fieldset>
</form>

[REDACTED]
    ```
From the above code, we see the structure of the authentication form; it is composed of a username field, a passwd field and a login field.

Let’s search for POST requests involving the username and passwd fields:
```m
index=botsv1 imreallynotbatman.com sourcetype=stream:http http_method="POST" form_data=*username*passwd*
| stats count by src_ip
```

Results:

src_ip	count
23.22.63.114	412
40.80.148.42	1

Interestingly, we see 1 request from `40.80.148.42` vs. 412 requests from `23.22.63.114`, The brute force attack is coming from this latest.

Answer: `23.22.63.114`

What was the first password attempted in the attack?

Based on our previous search, we can easily extract all probed username and passwords vs. time:

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http http_method="POST"
form_data=*username*passwd*
| rex field=form_data "username=(?<u>\w+)"
| rex field=form_data "passwd=(?<p>\w+)"
| table _time, u, p
| sort by _time
| head 5
```

Results:

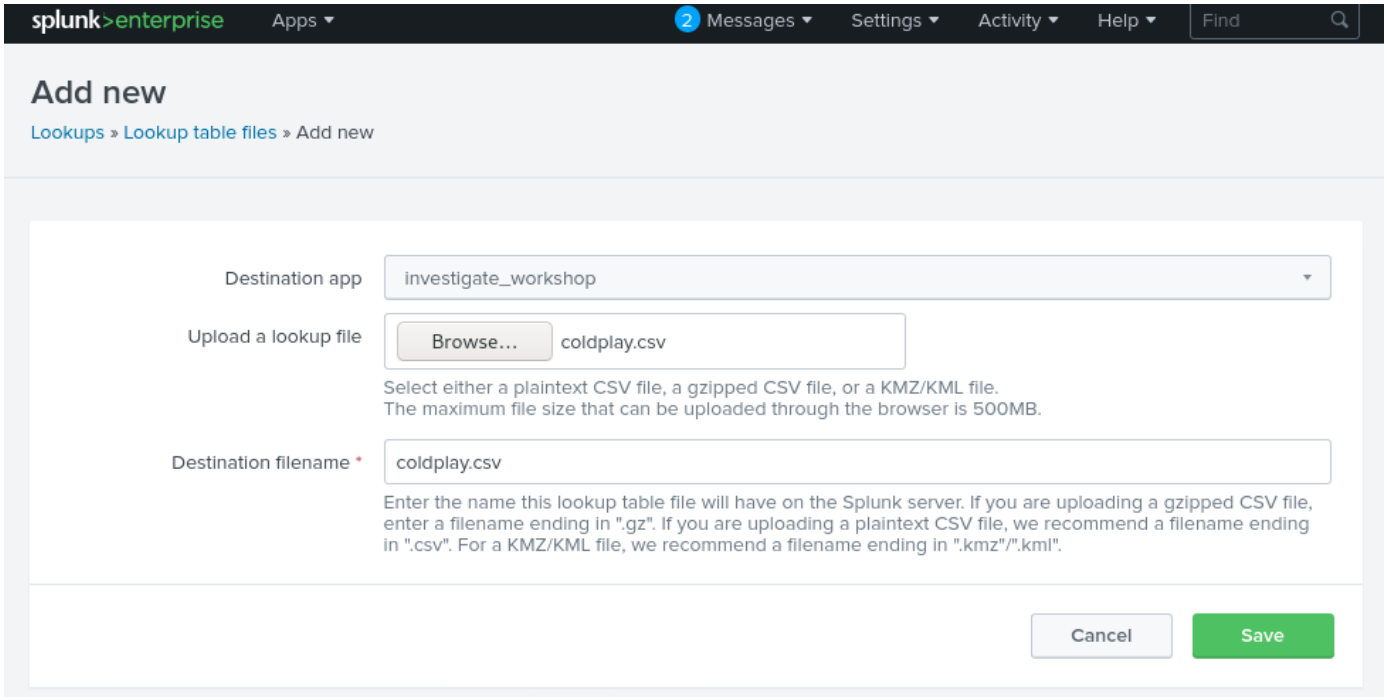
_time	u	p
2016-08-10 21:45:21.226	admin	12345678
2016-08-10 21:45:21.241	admin	letmein
2016-08-10 21:45:21.247	admin	qwerty
2016-08-10 21:45:21.250	admin	1234
2016-08-10 21:45:21.260	admin	123456

Answer: `12345678`

One of the passwords in the brute force attack is James Brodsky's favorite Coldplay song. Which six character song is it?

Go to https://en.wikipedia.org/wiki/List_of_songs_recorded_by_Coldplay and copy the table. Extract all the songs names (1st column) and save the file as coldplay.csv. You can download a copy here

Now in Splunk, go to 'Settings > Lookups > Lookup table files > Add New'.



Enter the following search to check that your file has successfully been imported:

```
| inputlookup coldplay.csv
```

Now, let's search for a common value

```
index=botsv1 sourcetype=stream:http form_data=*username*passwd*
| rex field=form_data "passwd=(?<userpassword>\w+)"
| eval lenpword=len(userpassword)
| search lenpword=6
| eval password=lower(userpassword)
| lookup coldplay.csv song as password OUTPUTNEW song
| search song=*
| table song
```

Answer: `yellow`

What was the correct password for admin access to the content management system running imreallynotbatman.com?

Upon discovering a seemingly correct password, a password brute-forcing engine such as hydra will enter the password a second time to verify that it works.
Let's count the number of occurences for each password, and extract the one(s) with at least 2 occurences.

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http http_method="POST"
form_data=*username*passwd*
| rex field=form_data "passwd=(?<p>\w+)"
| stats count by p
| where count>1
| table p
```

Result: `batman`

What was the average password length used in the password brute forcing attempt rounded to closest whole integer?

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http http_method="POST"
form_data=*username*passwd*
| rex field=form_data "passwd=(?<p>\w+)"
| eval pl=len(p)
| stats avg(pl) as av
| eval avg_count=round(av,0)
| table avg_count
```

Answer: `6`

How many seconds elapsed between the time the brute force password scan identified the correct password and the compromised login rounded to 2 decimal places?

As we have seen previously, 1 of the passwords (`batman`) was used 2 times. Let's extract the timestamps for the occurences of this password.

```
index=botsv1 sourcetype=stream:http form_data=*username*passwd* | rex
field=form_data "passwd=(?<p>\w+)"
| search p="batman"
| table _time, p, src_ip
| sort by _time
```

time	p	src_ip
2016-08-10 21:46:33.689	batman	23.22.63.114
2016-08-10 21:48:05.858	batman	40.80.148.42

Now, let's use `transaction` to compute the delay between these timestamps.

```
index=botsv1 sourcetype=stream:http form_data=*username*passwd* | rex
field=form_data "passwd=(?<p>\w+)"
| search p="batman"
| transaction p
| eval dur=round(duration,2)
| table dur
```

Answer:`92.17`

How many unique passwords were attempted in the brute force attempt?

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http http_method="POST"
form_data=*username*passwd*
| rex field=form_data "passwd=(?<p>\w+)"
| dedup p
| stats count
```

Answer: `412`

What is the name of the executable uploaded by P01s0n1vy?

An upload form is usually structured as follows:

```
<form enctype="multipart/form-data" action="_URL_" method="post">
```

Let's search for `multipart/form-data`:

```
index=botsv1 sourcetype=stream:http dest="192.168.250.70" "multipart/form-data"
| head 1
```

Result:

{"endtime":"2016-08-10T21:52:47.035555Z","timestamp":"2016-08-10T21:52:45.437445Z","accept":"text/html, application/xhtml+xml, */*","accept_language":"en-US","ack_packets_in":1,"ack_packets_out":55,"bytes":77896,"bytes_in":77648,"bytes_out":248,"c_ip":"40.80.148.42","cached":0,"capture_hostname":"demo-01","client_rtt":0,"client_rtt_packets":0,"client_rtt_sum":0,"connection_type":"Keep-Alive","content_disposition":["form-data; name=\"userfile[0]\"; filename=\"3791.exe\"","form-data; name=\"userfile[1]\"; filename=\"agent.php\"","form-data; name=\"userfile[2]\"; filename=\"\"","form-data; name=\"userfile[3]\"; filename=\"\"","form-data; name=\"userfile[4]\"; filename=\"\"","form-data; name=\"userfile[5]\"; filename=\"\"","form-data; name=\"userfile[6]\"; filename=\"\"","form-data; name=\"overwrite_files\"","form-data; name=\"option\"","form-data; name=\"action\"","form-data; name=\"dir\"","form-data; name=\"requestType\"","form-data; name=\"confirm\""],"cookie":"7598a3465c906161e060ac551a9e0276=9qfk2654t4rmhltilkfhe7ua23","cs_cache_control":"no-cache","cs_content_length":77045,"cs_content_type":"multipart/form-data; boundary=--------------------------7e0e42c20990","cs_version":["1.1","1.1"],"data_center_time":1049868,"data_packets_in":55,"data_packets_out":1,"dest_content":"{'action':'upload','message':'Upload successful!','error':'Upload successful!','success':true}","dest_headers":"HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nServer: Microsoft-IIS/8.5\r\nX-Powered-By: PHP/5.5.38\r\nDate: Wed, 10 Aug 2016 21:52:47 GMT\r\nContent-Length: 94\r\n\r\n","dest_ip":"192.168.250.70","dest_mac":"00:0C:29:C4:02:7E","dest_port":80,"duplicate_packets_in":52,"duplicate_packets_out":1,"http_comment":"HTTP/1.1 200 OK","http_content_length":94,"http_content_type":"text/html","http_method":"POST","http_referrer":"http://imreallynotbatman.com/joomla/administrator/index.php?option=com_extplorer&tmpl=component","http_user_agent":"Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko","missing_packets_in":0,"missing_packets_out":0,"network_interface":"eth1","packets_in":56,"packets_out":56,"part_filename":["3791.exe","agent.php"],"reply_time":1049868,"request":"POST /joomla/administrator/index.php HTTP/1.1","request_ack_time":10,"request_time":548242,"response_ack_time":81928,"response_time":0,"sc_date":"Wed, 10 Aug 2016 21:52:47 GMT","server":"Microsoft-IIS/8.5","server_rtt":5934,"server_rtt_packets":26,"server_rtt_sum":154301,"site":"imreallynotbatman.com","src_content":"--------------------------7e0e42c20990\r\nContent-Disposition: form-data; name=\"userfile[0]\"; filename=\"3791.exe\"\r\nContent-Type: application/octet-stream\r\n\r\nMZ\u0000\u0003\u0000\u0000\u0000\u0004\u0000\u0000\u0000

```
cannot be run in DOS
mode.\r\r\n$\u0000\u0000\u0000
```

What is the name of the file that defaced the imreallynotbatman.com website?

In the attack phases, the attacker is likely to have found a vulnerability, and exploited it to download files from the server, to an external server. As we have already identified 2 IP addresses involved in the attack, let's use them as destinations.

Let's search for requests originating from the server, with suricata logs to 23.22.63.114:

```
index=botsv1 sourcetype="suricata" src_ip="192.168.250.70" dest_ip="23.22.63.114"
|  stats count by http.http_method, http.hostname, http.url
|  sort -count
```

Results:

http.http_method	http.hostname	http.url	count
GET	imreallynotbatman.com	/joomla/administrator/index.php	824
POST	imreallynotbatman.com	/joomla/administrator/index.php	411
GET	71.39.18.126	/joomla/agent.php	52
GET	prankglassinebracket.jumpingcrab.com	/poisonivy-is-coming-for-you-batman.jpeg	3

Answer: `poisonivy-is-coming-for-you-batman.jpeg`

This attack used dynamic DNS to resolve to the malicious IP. What fully qualified domain name (FQDN) is associated with this attack?

We already identified the FQDN in the previous request.

Answer: `prankglassinebracket.jumpingcrab.com`

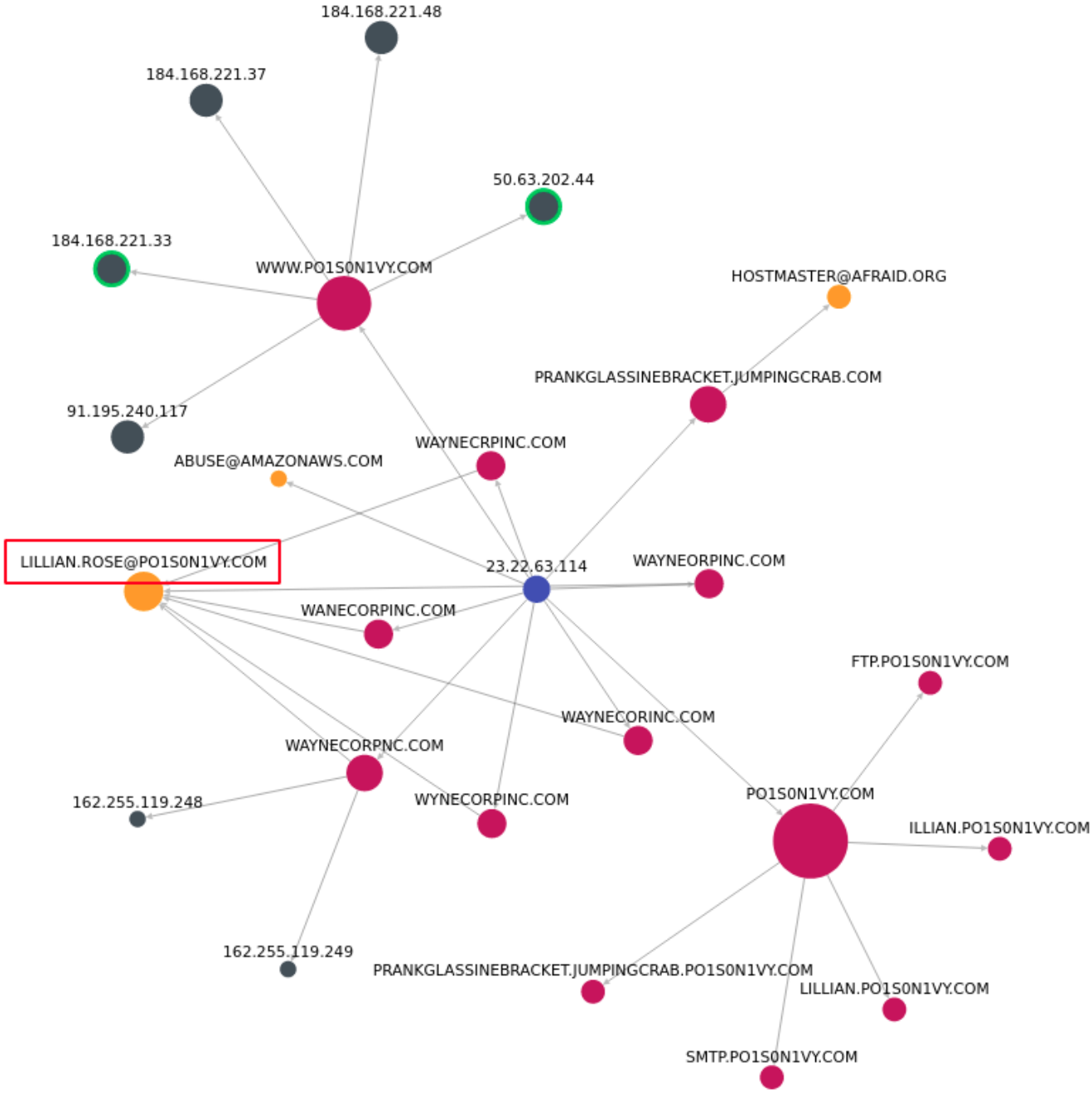
What IP address has P01s0n1vy tied to domains that are pre-staged to attack Wayne Enterprises?

We already identified the IP in the previous request.

Answer: `23.22.63.114`

Based on the data gathered from this attack and common open source intelligence sources for domain names, what is the email address that is most likely associated with P01s0n1vy APT group?

Googling for the IOCs collected so far leads to https://threatcrowd.org/ip.php?ip=23.22.63.114 where we are presented with a relationship diagram involving domains, IPs, emails:



Answer: `lillian.rose@po1s0n1vy.com`

GCPD reported that common TTPs (Tactics, Techniques, Procedures) for the P01s0n1vy APT group if initial compromise fails is to send a spear phishing email with custom malware attached to their intended target. This malware is usually connected to P01s0n1vy's initial attack infrastructure. Using research techniques, provide the SHA256 hash of this malware.

Following online searches leads to https://www.threatminer.org/host.php?q=23.22.63.114 where we are provided with file hashes, 1 of which being identified as malicious by many AV solutions:

- aae3f5a29935e6abcc2c2754d12a9af0
- 39eecefa9a13293a93bb20036eaf1f5e
- c99131e0169171935c5ac32615ed6261 (malicious)
  The last hash (https://www.threatminer.org/sample.php?q=c99131e0169171935c5ac32615ed6261) is associated with the following SHA256:

Answer: `9709473ab351387aab9e816eff3910b9f28a7a70202e250ed46dba8f820f34a8`.

What special hex code is associated with the customized malware discussed in the previous question?

Looking for the hash on Virustotal (https://www.virustotal.com/gui/file/9709473ab351387aab9e816eff3910b9f28a7a70202e250ed46dba8f820f34a8/community) shows an hex string associated to this malware:

```
53 74 65 76 65 20 42 72 61 6e 74 27 73 20 42 65 61 72 64 20 69 73 20
61 20 70 6f 77 65 72 66 75 6c 20 74 68 69 6e 67 2e 20 46 69 6e 64 20
74 68 69 73 20 6d 65 73 73 61 67 65 20 61 6e 64 20 61 73 6b 20 68 69
6d 20 74 6f 20 62 75 79 20 79 6f 75 20 61 20 62 65 65 72 21 21 21
```

What does this hex code decode to?

```
$ echo "53 74 65 76 65 20 42 72 61 6e 74 27 73 20 42 65 61 72 64 20 69 73 20 61 20
70 6f 77 65 72 66 75 6c 20 74 68 69 6e 67 2e 20 46 69 6e 64 20 74 68 69 73 20 6d
65 73 73 61 67 65 20 61 6e 64 20 61 73 6b 20 68 69 6d 20 74 6f 20 62 75 79 20 79
6f 75 20 61 20 62 65 65 72 21 21 21" | xxd -r -p
Steve Brant's Beard is a powerful thing. Find this message and ask him to buy you
a beer!!!
```

Summary

Based on our investigation, we can piece together what our adversary did:

Scanned for vulnerabilities

Found site is running Joomla

Performed a brute force password scan, logged into Joomla, installed file upload modules

Uploaded webshell

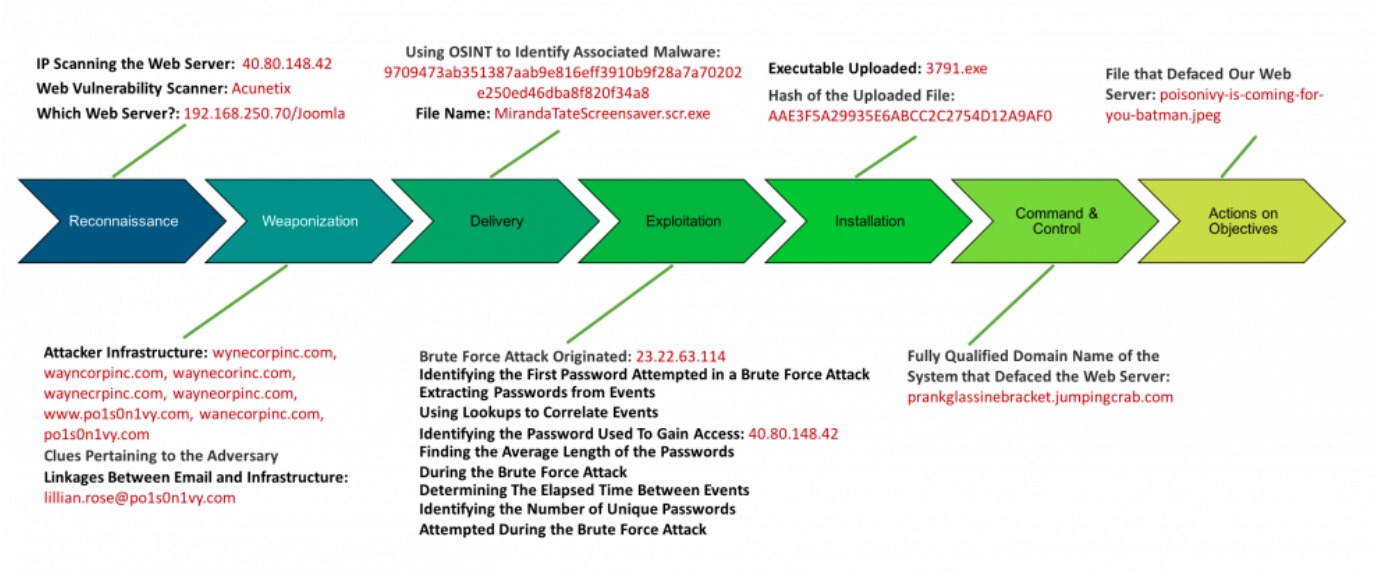
Used webshell to upload reverse TCP shell

Connected via metasploit

Tried to move around but couldn’t get out of locked down Windows 2012R2

Defaced website with downloaded defacement image

Kill Chain



Threat picture - APT