Meerkat Writeup

Prepared by: Cyberjunkie

Machine Author(s): Sebh24

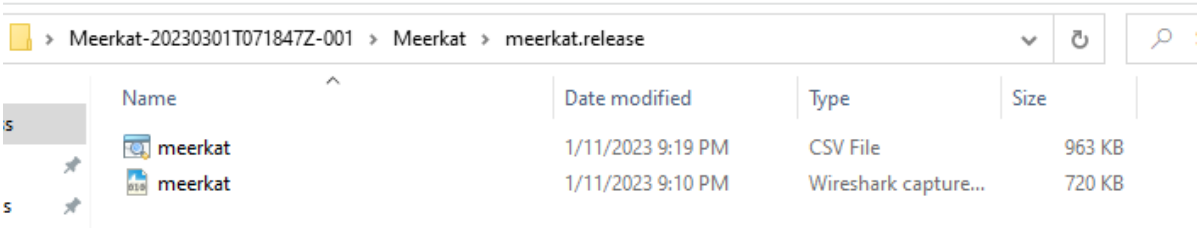
Difficulty: Easy

Scenario

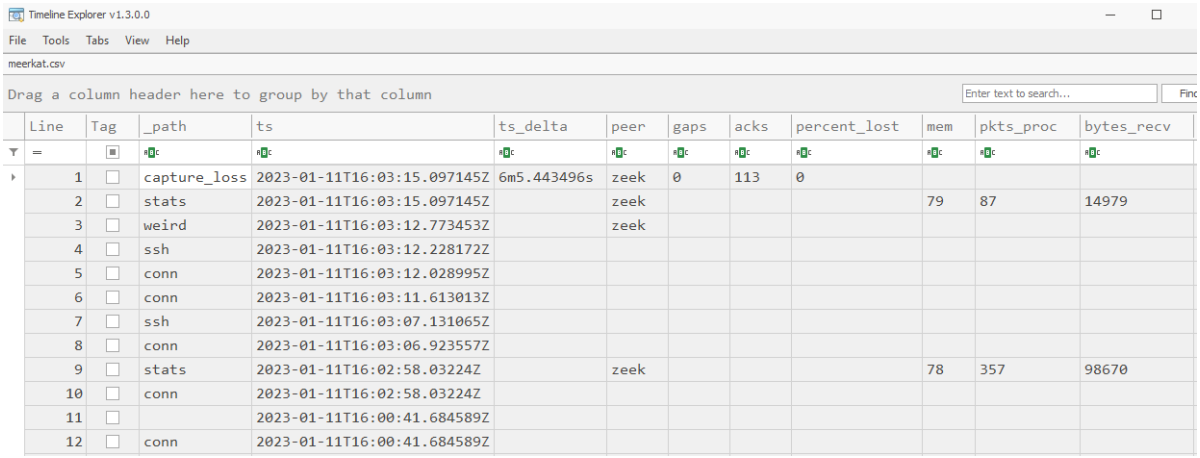
As a fast growing startup, Forela have been utilising a business management platform. Unfortunately our documentation is scarce and our administrators aren't the most security aware. As our new security provider we'd like you to take a look at some PCAP and log data we have exported to confirm if we have (or have not) been compromised.

Initial Analysis

We are provided with a pcap file and suricata/zeek log file.



We will use the pcap and the recorded network log file to answers the questions in a quick way. Lets start by opening the processed log file in timeline explorer and the pcap file in wireshark.


Timeline Explorer v1.3.0.0

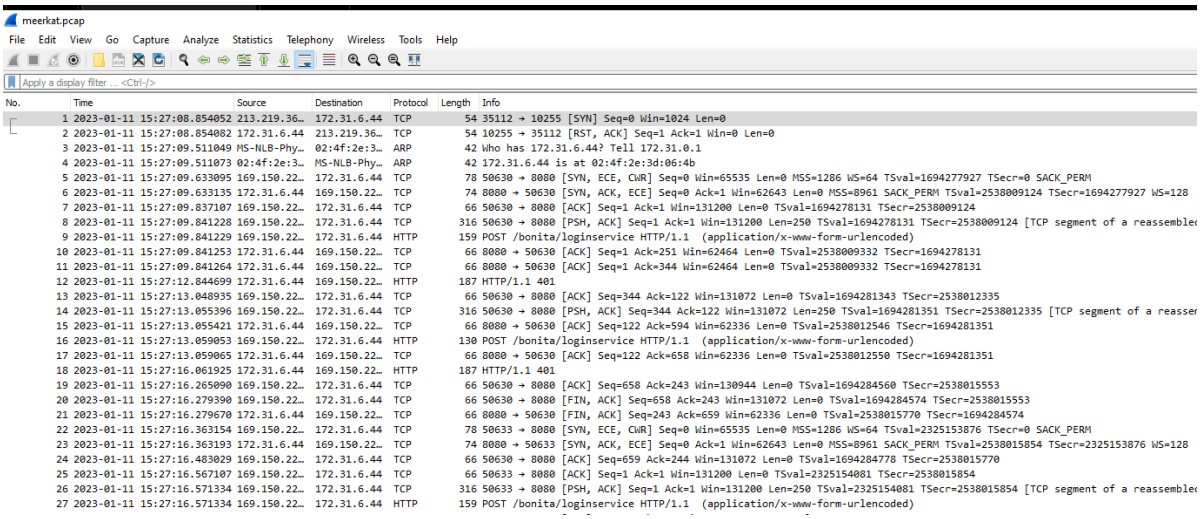
File Tools Tabs View Help

meerkat.csv

Drag a column header here to group by that column

Enter text to search...

Line	Tag	_path	ts	ts_delta	peer	gaps	acks	percent_lost	mem	pkts_proc	bytes_recv
1		capture_loss	2023-01-11T16:03:15.097145Z	6m5.443496s	zeek	0	113	0			
2		stats	2023-01-11T16:03:15.097145Z		zeek				79	87	14979
3		weird	2023-01-11T16:03:12.773453Z		zeek						
4		ssh	2023-01-11T16:03:12.228172Z								
5		conn	2023-01-11T16:03:12.028995Z								
6		conn	2023-01-11T16:03:11.613013Z								
7		ssh	2023-01-11T16:03:07.131065Z								
8		conn	2023-01-11T16:03:06.923557Z								
9		stats	2023-01-11T16:02:58.03224Z		zeek				78	357	98670
10		conn	2023-01-11T16:02:58.03224Z								
11			2023-01-11T16:00:41.684589Z								
12		conn	2023-01-11T16:00:41.684589Z								


meerkat.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

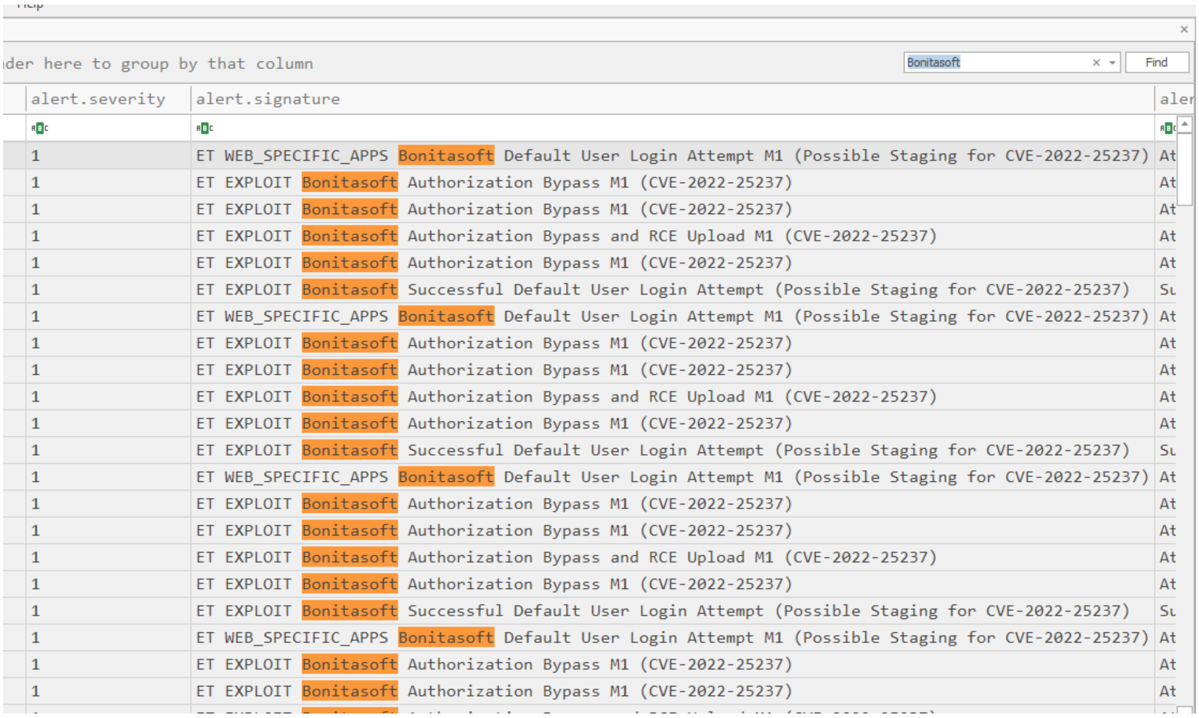
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-01-11 15:27:08.854052	213.219.36…	172.31.6.44	TCP	54	35112 → 10255 [SYN] Seq=0 Win=1024 Len=0
2	2023-01-11 15:27:08.854082	172.31.6.44	213.219.36…	TCP	54	10255 → 35112 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	2023-01-11 15:27:09.511049	MS-NLB-Phy…	02:4f:2e:3…	ARP	42	Who has 172.31.6.44? Tell 172.31.0.1
4	2023-01-11 15:27:09.511073	02:4f:2e:3…	MS-NLB-Phy…	ARP	42	172.31.6.44 is at 02:4f:2e:3d:06:4b
5	2023-01-11 15:27:09.633095	169.150.22…	172.31.6.44	TCP	78	50630 → 8080 [SYN, ECE, CWR] Seq=0 Win=65535 Len=0 MSS=1286 WS=64 TSval=1694277927 TSecr=0 SACK_PERM
6	2023-01-11 15:27:09.633135	172.31.6.44	169.150.22…	TCP	74	8080 → 50630 [SYN, ACK, ECE] Seq=0 Ack=1 Win=62643 Len=0 MSS=8961 SACK_PERM TSval=2538009124 TSecr=1694277927 WS=128
7	2023-01-11 15:27:09.837107	169.150.22…	172.31.6.44	TCP	66	50630 → 8080 [ACK] Seq=1 Ack=1 Win=131200 Len=0 TSval=1694278131 TSecr=2538009124
8	2023-01-11 15:27:09.841228	169.150.22…	172.31.6.44	TCP	316	50630 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=131200 Len=250 TSval=1694278131 TSecr=2538009124 [TCP segment of a reassemble
9	2023-01-11 15:27:09.841229	169.150.22…	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
10	2023-01-11 15:27:09.841253	172.31.6.44	169.150.22…	TCP	66	8080 → 50630 [ACK] Seq=1 Ack=251 Win=62464 Len=0 TSval=2538009332 TSecr=1694278131
11	2023-01-11 15:27:09.841264	172.31.6.44	169.150.22…	TCP	66	8080 → 50630 [ACK] Seq=1 Ack=344 Win=62464 Len=0 TSval=2538009332 TSecr=1694278131
12	2023-01-11 15:27:12.844699	172.31.6.44	169.150.22…	HTTP	187	HTTP/1.1 401
13	2023-01-11 15:27:13.048935	169.150.22…	172.31.6.44	TCP	66	50630 → 8080 [ACK] Seq=344 Ack=122 Win=131072 Len=0 TSval=1694281343 TSecr=2538012335
14	2023-01-11 15:27:13.055396	169.150.22…	172.31.6.44	TCP	316	50630 → 8080 [PSH, ACK] Seq=344 Ack=122 Win=131072 Len=250 TSval=1694281351 TSecr=2538012335 [TCP segment of a reasser
15	2023-01-11 15:27:13.055421	172.31.6.44	169.150.22…	TCP	66	8080 → 50630 [ACK] Seq=122 Ack=594 Win=62336 Len=0 TSval=2538012546 TSecr=1694281351
16	2023-01-11 15:27:13.059053	169.150.22…	172.31.6.44	HTTP	130	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
17	2023-01-11 15:27:13.059065	172.31.6.44	169.150.22…	TCP	66	8080 → 50630 [ACK] Seq=122 Ack=658 Win=62336 Len=0 TSval=2538012550 TSecr=1694281351
18	2023-01-11 15:27:16.061925	172.31.6.44	169.150.22…	HTTP	187	HTTP/1.1 401
19	2023-01-11 15:27:16.265090	169.150.22…	172.31.6.44	TCP	66	50630 → 8080 [ACK] Seq=658 Ack=243 Win=130944 Len=0 TSval=1694284560 TSecr=2538015553
20	2023-01-11 15:27:16.279390	169.150.22…	172.31.6.44	TCP	66	50630 → 8080 [FIN, ACK] Seq=658 Ack=243 Win=131072 Len=0 TSval=1694284574 TSecr=2538015553
21	2023-01-11 15:27:16.279670	172.31.6.44	169.150.22…	TCP	66	8080 → 50630 [FIN, ACK] Seq=243 Ack=659 Win=62336 Len=0 TSval=2538015770 TSecr=1694284574
22	2023-01-11 15:27:16.363154	169.150.22…	172.31.6.44	TCP	78	50633 → 8080 [SYN, ECE, CWR] Seq=0 Win=65535 Len=0 MSS=1286 WS=64 TSval=2325153876 TSecr=0 SACK_PERM
23	2023-01-11 15:27:16.363193	172.31.6.44	169.150.22…	TCP	74	8080 → 50633 [SYN, ACK, ECE] Seq=0 Ack=1 Win=62643 Len=0 MSS=8961 SACK_PERM TSval=2538015854 TSecr=2325153876 WS=128
24	2023-01-11 15:27:16.483029	169.150.22…	172.31.6.44	TCP	66	50630 → 8080 [ACK] Seq=659 Ack=244 Win=131072 Len=0 TSval=1694284778 TSecr=2538015770
25	2023-01-11 15:27:16.567107	169.150.22…	172.31.6.44	TCP	66	50633 → 8080 [ACK] Seq=1 Ack=1 Win=131200 Len=0 TSval=2325154081 TSecr=2538015854
26	2023-01-11 15:27:16.571334	169.150.22…	172.31.6.44	TCP	316	50633 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=131200 Len=250 TSval=2325154081 TSecr=2538015854 [TCP segment of a reassemble
27	2023-01-11 15:27:16.571334	169.150.22…	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)

Questions

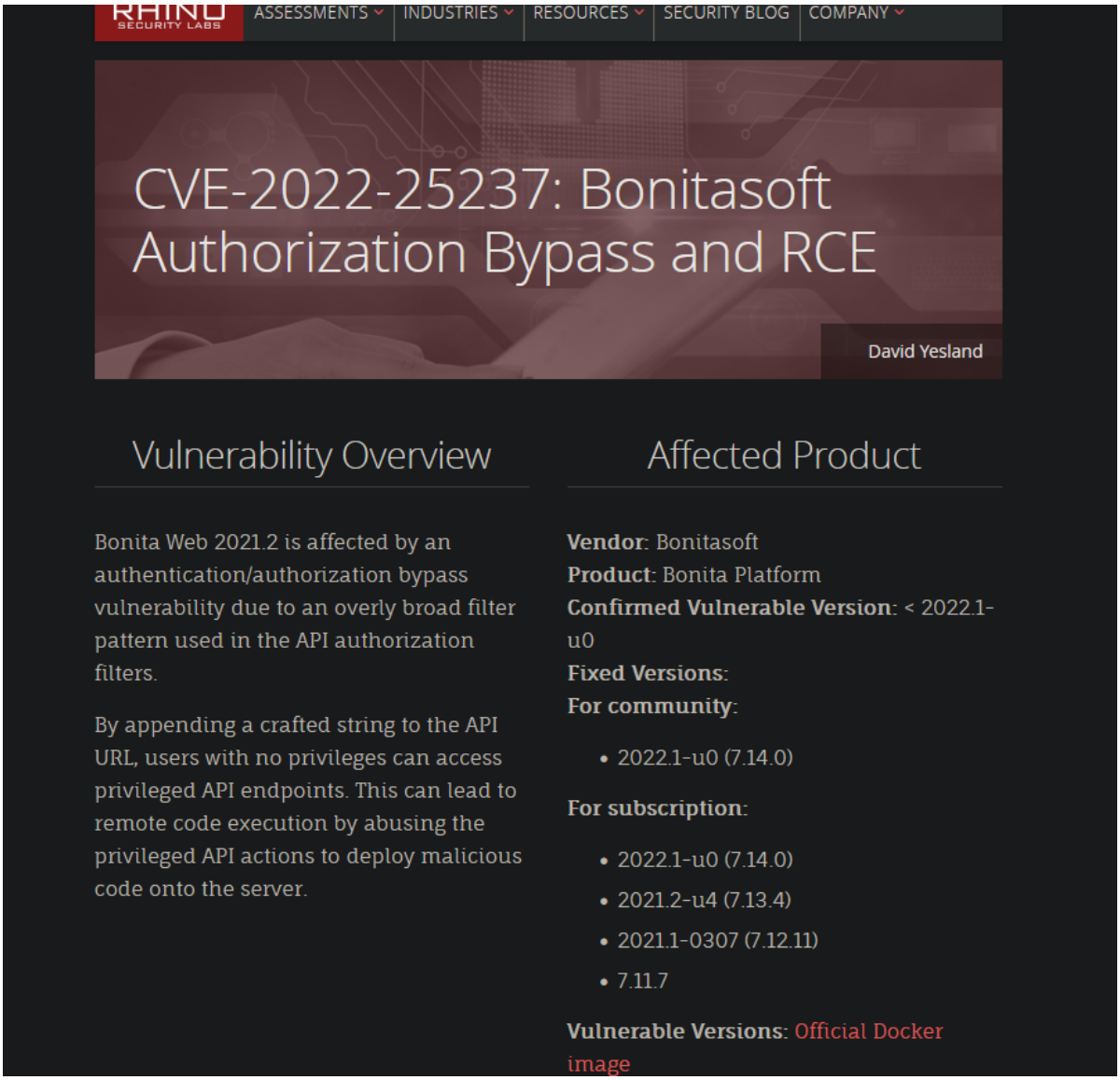
1-We believe our Business Management Platform server has been compromised. Please can you confirm the application name and version running on the server (we aren't very good at documentation).

When analyzing log file , we spot multiple alerts for an exploit for Bonitasoft authorization bypass. We can also see the associated CVE.



alert.severity	alert.signature	aler
1	ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass and RCE Upload M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Successful Default User Login Attempt (Possible Staging for CVE-2022-25237)	Su
1	ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass and RCE Upload M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Successful Default User Login Attempt (Possible Staging for CVE-2022-25237)	Su
1	ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass and RCE Upload M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Successful Default User Login Attempt (Possible Staging for CVE-2022-25237)	Su
1	ET WEB_SPECIFIC_APPS Bonitasoft Default User Login Attempt M1 (Possible Staging for CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At
1	ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)	At

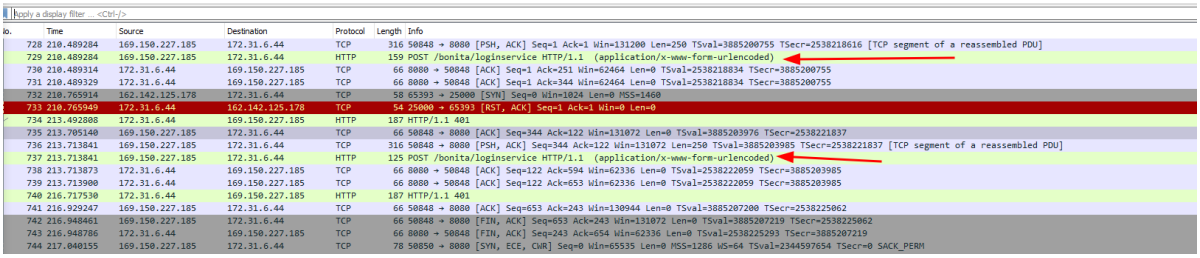
Now that we know the cve , we can research on it and find the vulnerable version of the application



Answer : 2022.1-u0

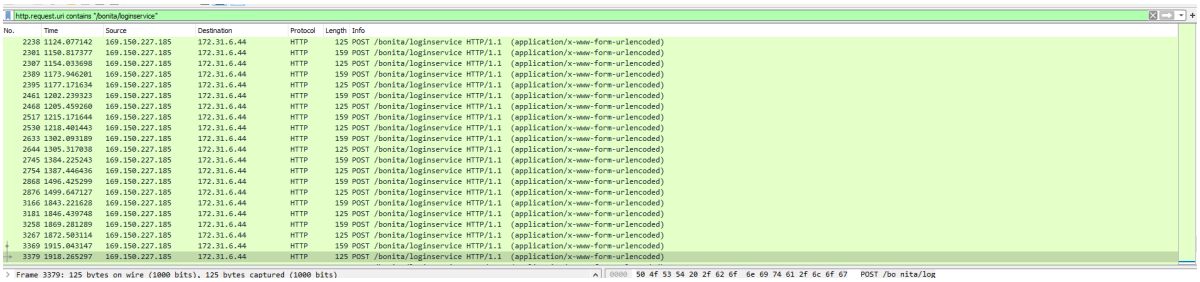
2- We believe the attacker may have used a subset of the brute forcing attack category - what is the name of the attack carried out?

On opening the pcap, we can see post requests to a login endpoint



No.	Time	Source	Destination	Protocol	Length	Info
728	210.489284	169.150.227.185	172.31.6.44	TCP	316	50848 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=131200 Len=250 TSval=3885200755 TSecr=2538218616 [TCP segment of a reassembled PDU]
729	210.489284	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
730	210.489314	172.31.6.44	169.150.227.185	TCP	66	8080 → 50848 [ACK] Seq=1 Ack=251 Win=62464 Len=0 TSval=2538218834 TSecr=3885200755
731	210.489329	172.31.6.44	169.150.227.185	TCP	66	8080 → 50848 [ACK] Seq=1 Ack=344 Win=62464 Len=0 TSval=2538218834 TSecr=3885200755
732	210.765914	162.142.125.178	172.31.6.44	TCP	58	65393 → 25000 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
733	210.765949	172.31.6.44	162.142.125.178	TCP	54	25000 → 65393 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
734	213.492808	172.31.6.44	169.150.227.185	HTTP	187	HTTP/1.1 401
735	213.705140	169.150.227.185	172.31.6.44	TCP	66	50848 → 8080 [ACK] Seq=344 Ack=122 Win=131072 Len=0 TSval=3885203976 TSecr=2538221837
736	213.713841	169.150.227.185	172.31.6.44	TCP	316	50848 → 8080 [PSH, ACK] Seq=344 Ack=122 Win=131072 Len=250 TSval=3885203985 TSecr=2538221837 [TCP segment of a reassembled PDU]
737	213.713841	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
738	213.713873	172.31.6.44	169.150.227.185	TCP	66	8080 → 50848 [ACK] Seq=122 Ack=594 Win=62336 Len=0 TSval=2538222059 TSecr=3885203985
739	213.713900	172.31.6.44	169.150.227.185	TCP	66	8080 → 50848 [ACK] Seq=122 Ack=653 Win=62336 Len=0 TSval=2538222059 TSecr=3885203985
740	216.717530	172.31.6.44	169.150.227.185	HTTP	187	HTTP/1.1 401
741	216.929247	169.150.227.185	172.31.6.44	TCP	66	50848 → 8080 [ACK] Seq=653 Ack=243 Win=130944 Len=0 TSval=3885207200 TSecr=2538225062
742	216.948461	169.150.227.185	172.31.6.44	TCP	66	50848 → 8080 [FIN, ACK] Seq=653 Ack=243 Win=131072 Len=0 TSval=3885207219 TSecr=2538225062
743	216.948786	172.31.6.44	169.150.227.185	TCP	66	8080 → 50848 [FIN, ACK] Seq=243 Ack=654 Win=62336 Len=0 TSval=2538225293 TSecr=3885207219
744	217.040155	169.150.227.185	172.31.6.44	TCP	78	50850 → 8080 [SYN, ECE, CWR] Seq=0 Win=65535 Len=0 MSS=1286 WS=64 TSval=2344597654 TSecr=0 SACK_PERM

Lets filter out requests only for this endpoint to correctly identify the attacker ip and the bruteforce technique used. We use this because in real scenario, there are hundered of legitimate login requests on a public webservice, so we need to pinpoint the attacker ip which will have a lot of consecative requests to login endpoint as this is not a normal behavior (legit login requests for 1 user may be 3 4 requests max).



No.	Time	Source	Destination	Protocol	Length	Info
2238	1124.077142	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2301	1150.817377	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2307	1154.033698	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2389	1173.946201	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2395	1177.171634	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2461	1202.239323	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2468	1205.459260	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2517	1215.171644	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2530	1218.401443	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2633	1302.093189	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2644	1305.317038	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2745	1384.225243	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2754	1387.446436	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2868	1496.425289	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
2876	1499.647127	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3166	1843.221628	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3181	1846.439748	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3258	1869.281289	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3267	1872.503114	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3369	1915.043147	169.150.227.185	172.31.6.44	HTTP	159	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3379	1918.265297	169.150.227.185	172.31.6.44	HTTP	125	POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)

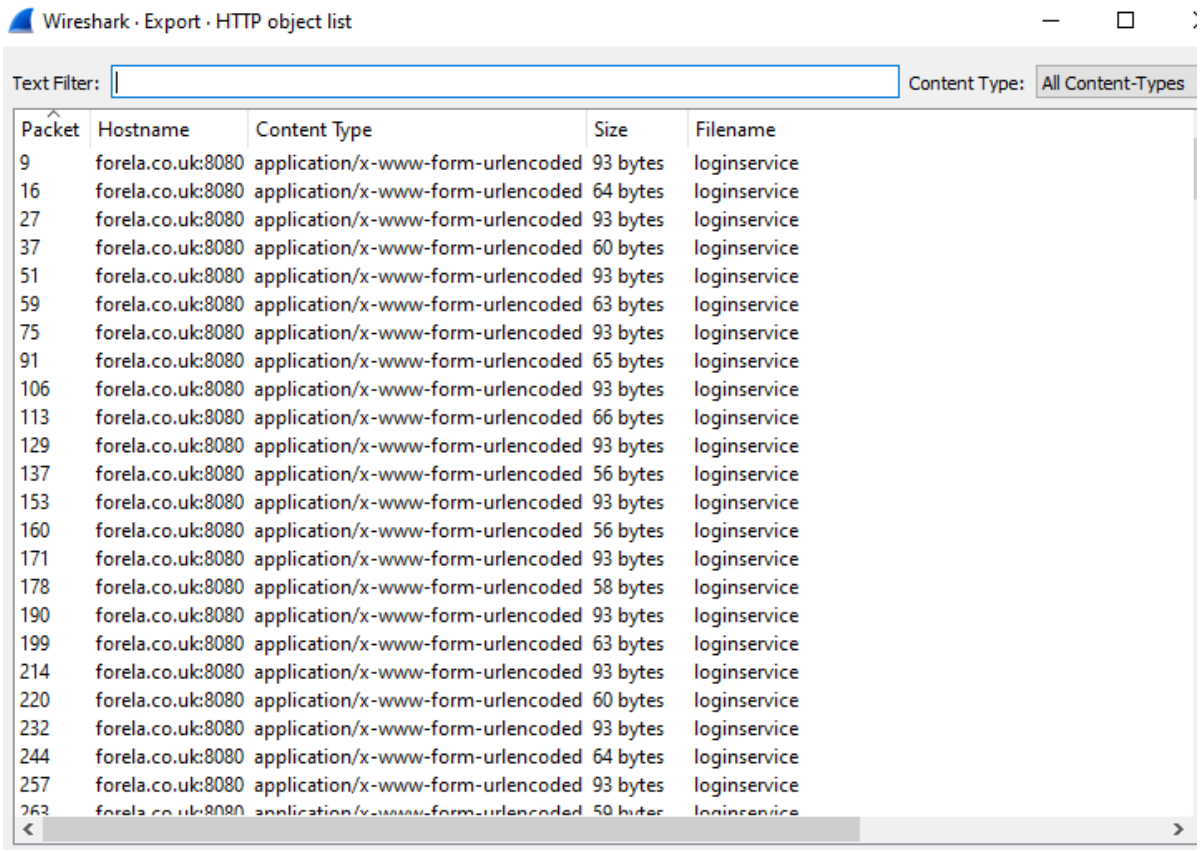
We filter for the “/bonita/loginservice” endpoint and find the IP “169.150.227.185” consecutively. Now if we explore some of these requests , we can find that set of credentials are being used and each username and password sets are unique in each request. In classic bruteforce attempts, every password is tried for every user account and vice versa. Here each set is unique which makes it a credential stuffing attack. Another major point to support this theory ois that passwords are not common passwords or made via dictionary, they are likely valid credentials from a known data breach or collected via phishing campaign.

Answer: Credential stuffing.

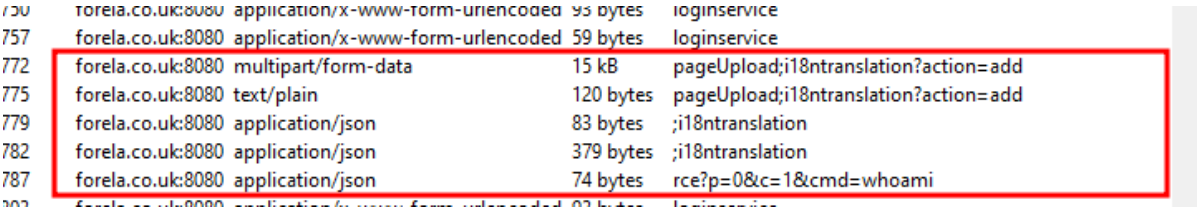
3- Does the vulnerability exploited have a CVE assigned - and if so, which one?

Since so far we don't have any knowledge of CVE, we need to look at endpoint interactions or we can look at suricata logs which enriches the network telemtery with Threat intel.

First lets explore any rce or some kind of interaction with enpoint. Lets look at exported objects from the pcap.



Here we see that most of these are from the post requests from the bruteforce attempt. Lets scroll down to see what occured after the bruteforce.

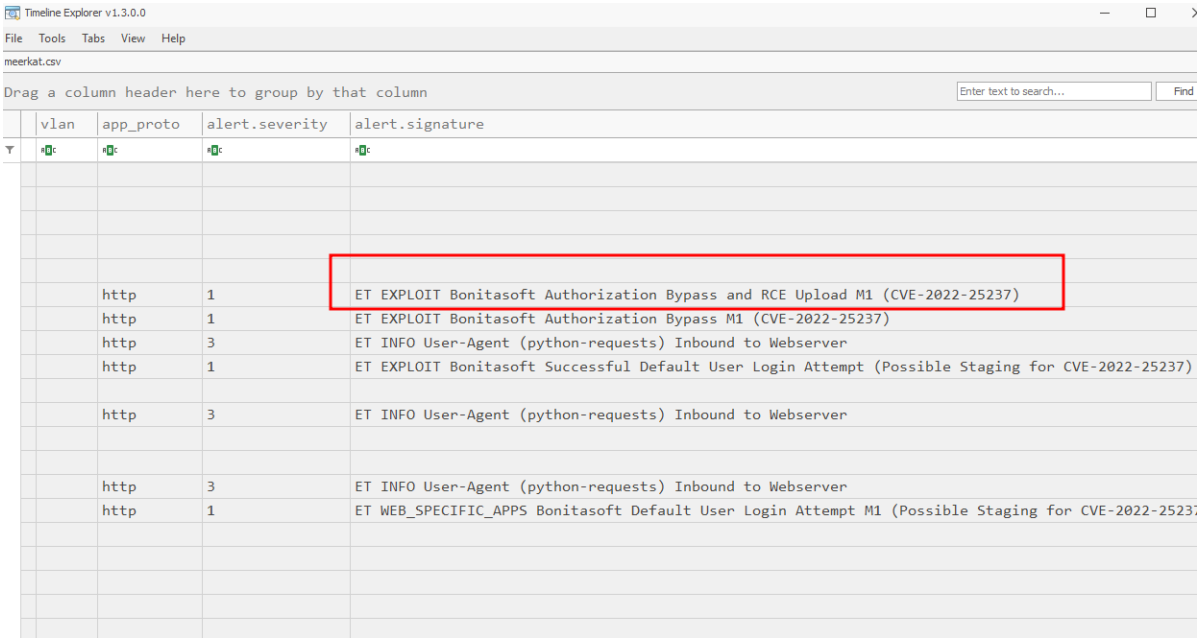


we spot some other endpoints and a suspicious get parameter named rce and a command executed “whoami” which is often the first command to verify rce.

From the look of these endpoint it looks like vulnerability was exploited allowing attacker to upload a Get parameter webshell allowing RCE from url. Lets go to the relevent packet and view its content.

It seems that attacker got successfull login and uploaded a file named rce_api_extension.zip. Lets use google for the the above endpoint to find any cve. We googled the endpoint uri and some results popped up

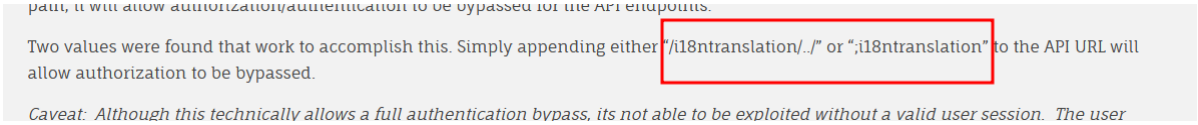
We can also find this from our suricata log file.



Answer : CVE-2022-25237

4- What is this API url abused by the exploit?

If we research on the cve, we find the details which api is vulnerable and is abused.

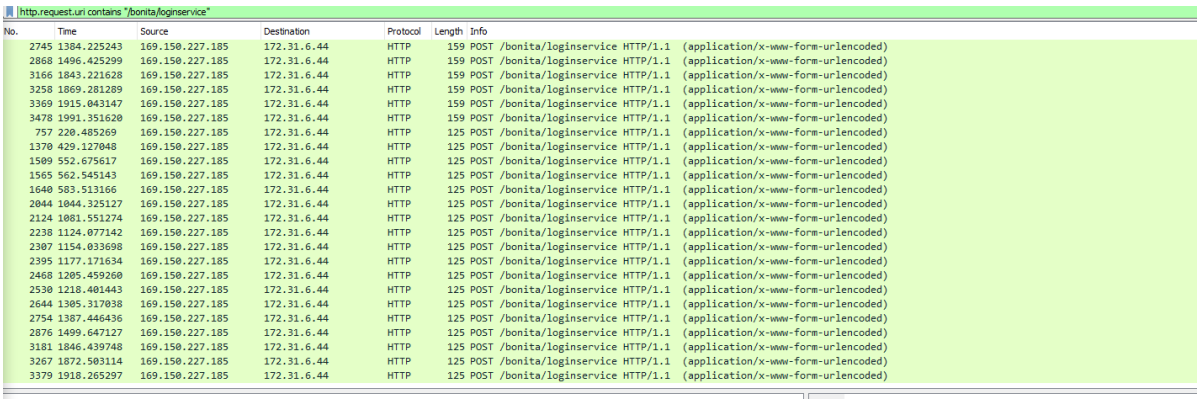


[Source : https://rhinosecuritylabs.com/application-security/cve-2022-25237-bonitasoft-authorization-bypass/](https://rhinosecuritylabs.com/application-security/cve-2022-25237-bonitasoft-authorization-bypass/)

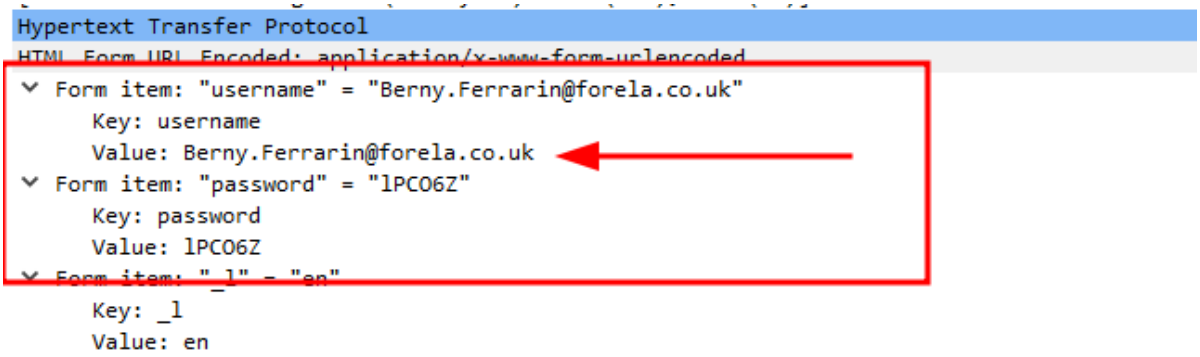
Answer : / i18ntranslation/

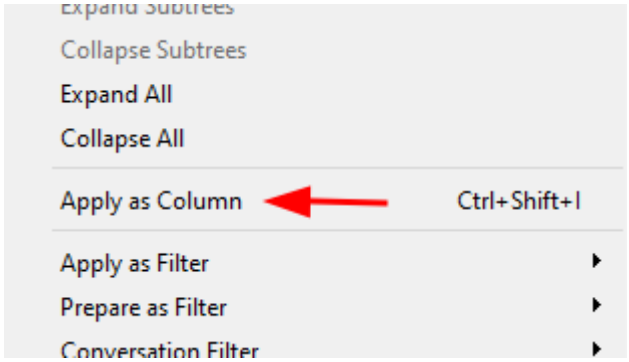
5- How many combinations of usernames and passwords were used in the credential stuffing attack?

We need to find unique set of username:passwords used in attack. First filter the requests only going towards the login endpoint.

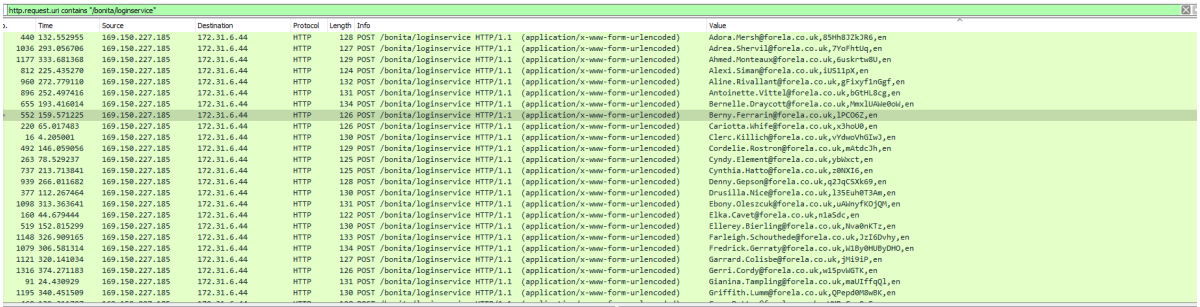


Now open any request packet and view the form fields where username and passwords are visible.

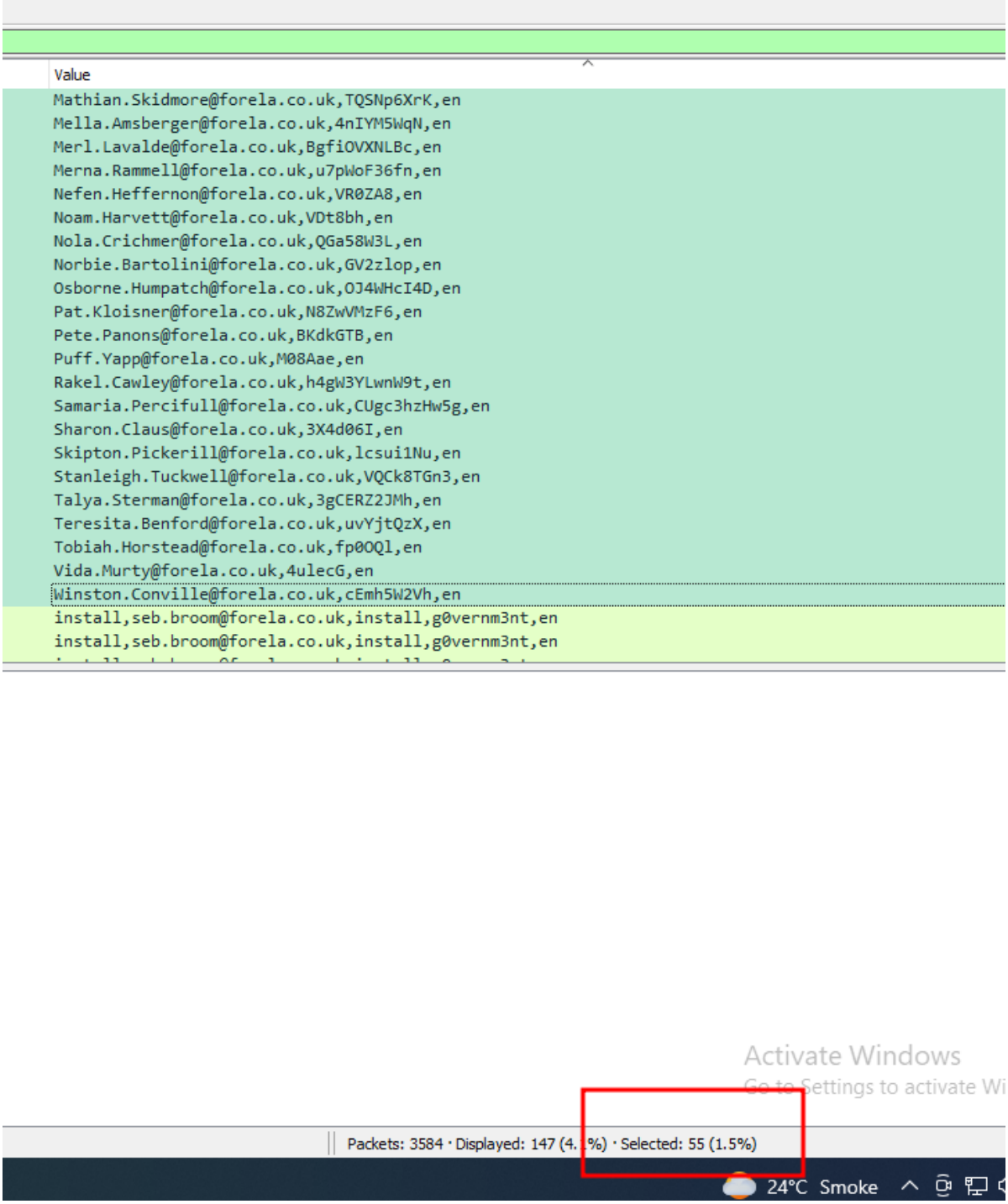




Now this data will be displayed alongside other information.



Now select all the packets with different usernames.



These are all the set of credentials which were unsuccessful, and they are 55. Since it was a credential stuffing attack, we only need to count each username one time as the username:password are not recurring in this attack. Then we have another last set of creds, which are valid set of credentials and used by attacker each time when command is executed, to compromise the server.

```
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
seb.broom@forela.co.uk,g0vernm3nt,en
```

As we can see these creds are recurring, meaning that attacker got access using this set of creds and used the session to perform there malicious actions.

So in total we have 56 username:password combinations

Answer : 56

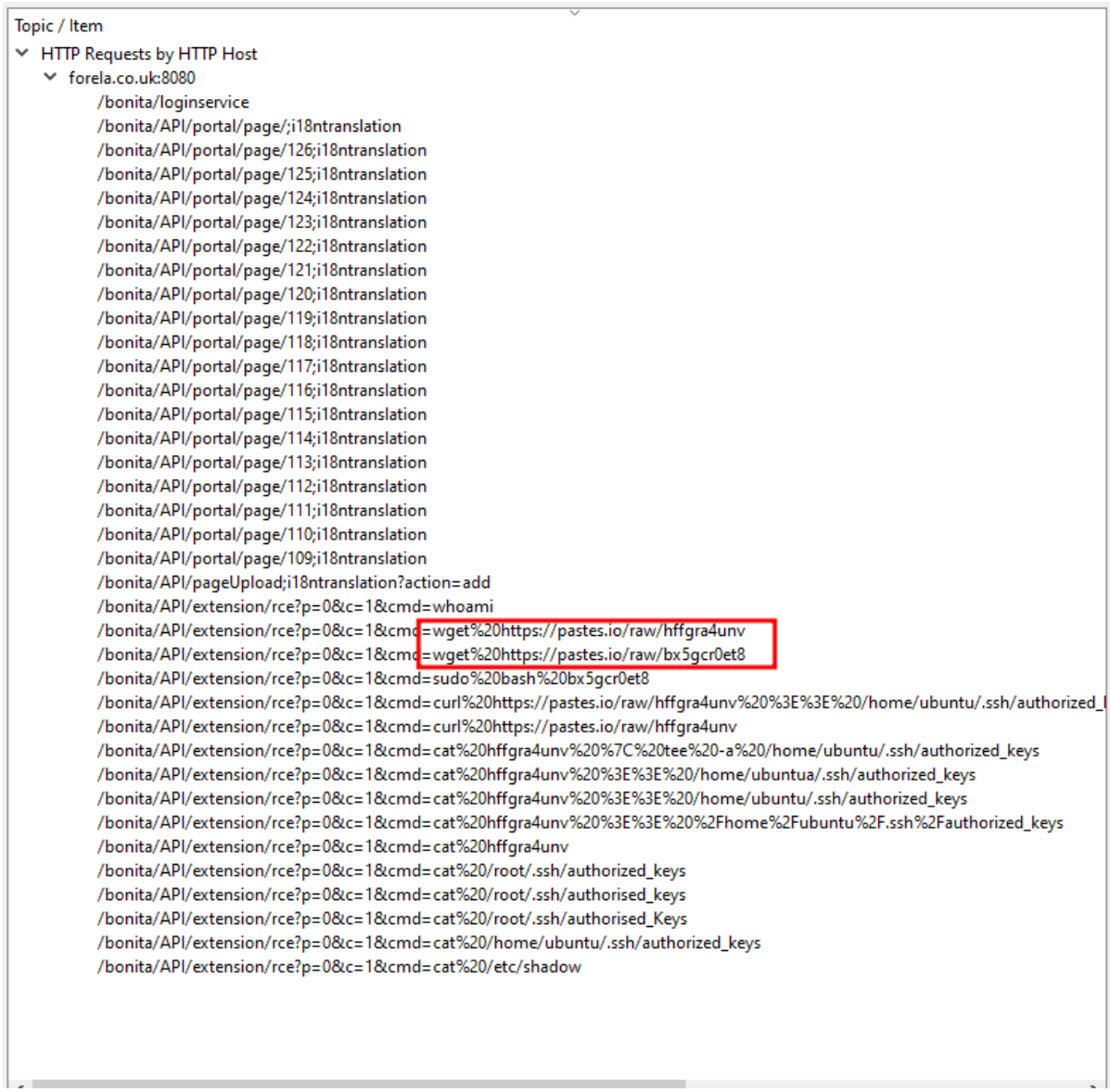
6- Which username and password combination was successful?

As already seen in question3, we identified the set of credentials from the request where attacker was able to abuse the vulnerable api to get rce.

Answer : seb.broom@forela.co.uk : g0vernm3nt

7- If any, which text sharing site did the attacker utilize?

We can see all the unique http requests to the host from statistics->http->requests.

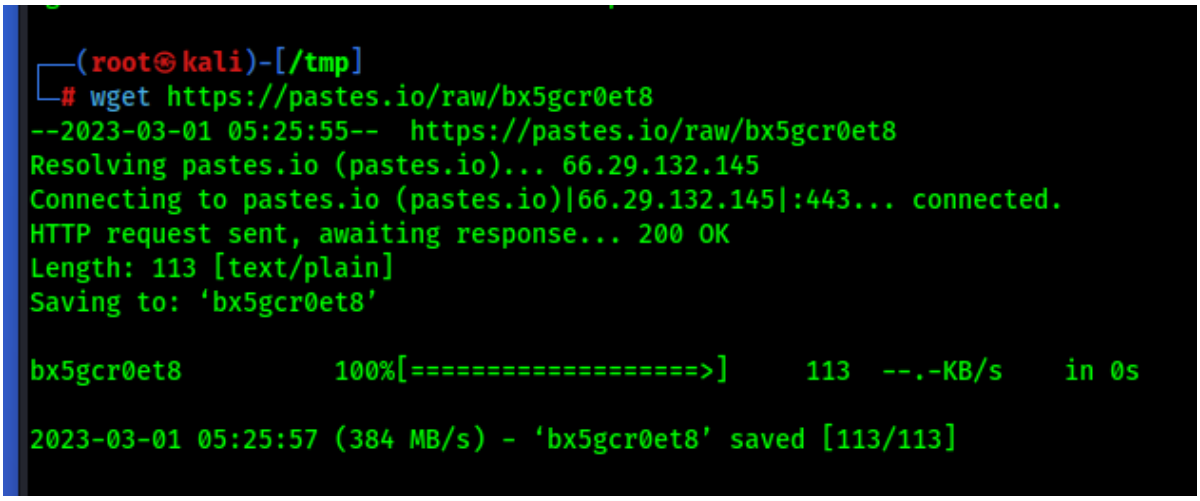


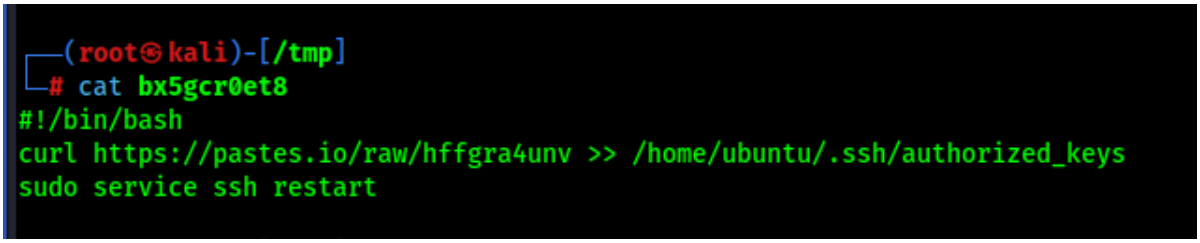
Here we spotted that attack fetched some files from the text sharing site “pastes.io” using the webshell. So the files were downloaded on the compromised webserver. Attackers use opensource services like pastebin pastes.io to evade domain blacklists, as these sites are not neccassarily malicious but are used by attackers.

Answer: https[:]//[pastes.io]/

8- Please provide the file hash of the script used by the attack to gain persistent access to our host.

We saw that after getting rce, attacker fetched 2 ascii/txt files from pastes.io website using wget .And after fetching them attacker execute the files with bash meaning its probably a bash script. So lets fetch the files on our linux vm and hash it.

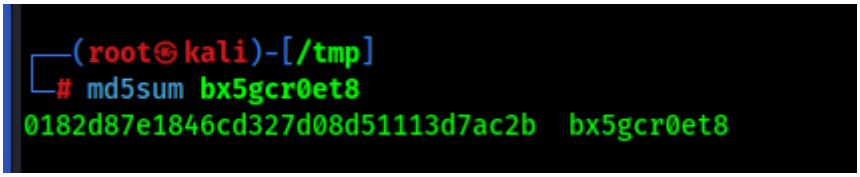




As we can see that attacker added shabang header with bash making it execytable by bash.

The script fetches another file from pastes.io and directly appends it to the user “ubuntu” ssh keys. The other file would most probably have ssh public key contents in it and now attacker can access the server through using his “private” key of the added public key of user ubuntu. This is one of a persistence technique.

Lets calculate the hash of this file and submit the answer



Answer : 0182d87e1846cd327d08d51113d7ac2b

9- Please provide the file hash of the public key used by the attacker to gain persistence on our host

Now let us fetch the second file

```
└─# wget https://pastes.io/raw/hffgra4unv
--2023-03-01 05:41:58--  https://pastes.io/raw/hffgra4unv
Resolving pastes.io (pastes.io)... 66.29.132.145
Connecting to pastes.io (pastes.io)|66.29.132.145|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 380 [text/plain]
Saving to: 'hffgra4unv'

hffgra4unv          100%[===================>]     380  --.-KB/s    in 0s

2023-03-01 05:41:59 (12.7 MB/s) - 'hffgra4unv' saved [380/380]


┌──(root㉿kali)-[/tmp]
└─# cat hffgra4unv
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCgruRMq3DMroGXrcPeeuEqQq3iS/sAL3gryt+nUqbB
A/M+KG4ElCvJS4gP2os1b8FMk3ZwvrVTdpEKW6wdGqPl2wxznBjOBstx6OF2yp9RIOb3c/ezgs9zvnaO
07YC8Sm4nkkXHgkabqcM7rHEY4Lay0LWF9UbxueSAHIJgQ2ADbKSnlg0gMnJTNRwKbqesk0ZcG3b6icj
6nkKykezBLvWc7z4mkSm28ZVTa15W3HUWSEWRbGgJ6eMBdi7WnWXZ92SYDq0XUBV2Sx2gjoDGHwcd6I0
q9BU52wWYo3L3LaPEoTcLuA+hnn82086oUzJfmEUtWGlPAXfJBN7vRIMSvsN

┌──(root㉿kali)-[/tmp]
```

As expected, this is the publick key created by the attacker, and attacker can now use the private pair of the key to login on the server whenever they want, providing them persistence access.

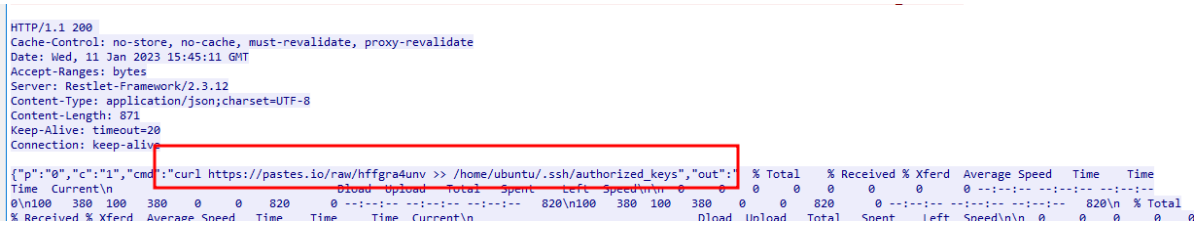
Lets hash this file to answer the question

```
┌──(root㉿kali)-[/tmp]
└─# md5sum hffgra4unv
dbb906628855a433d70025b6692c05e7  hffgra4unv
```

Answer : dbb906628855a433d70025b6692c05e7

10- Can you confirmed the file modified by the attacker to gain persistence?

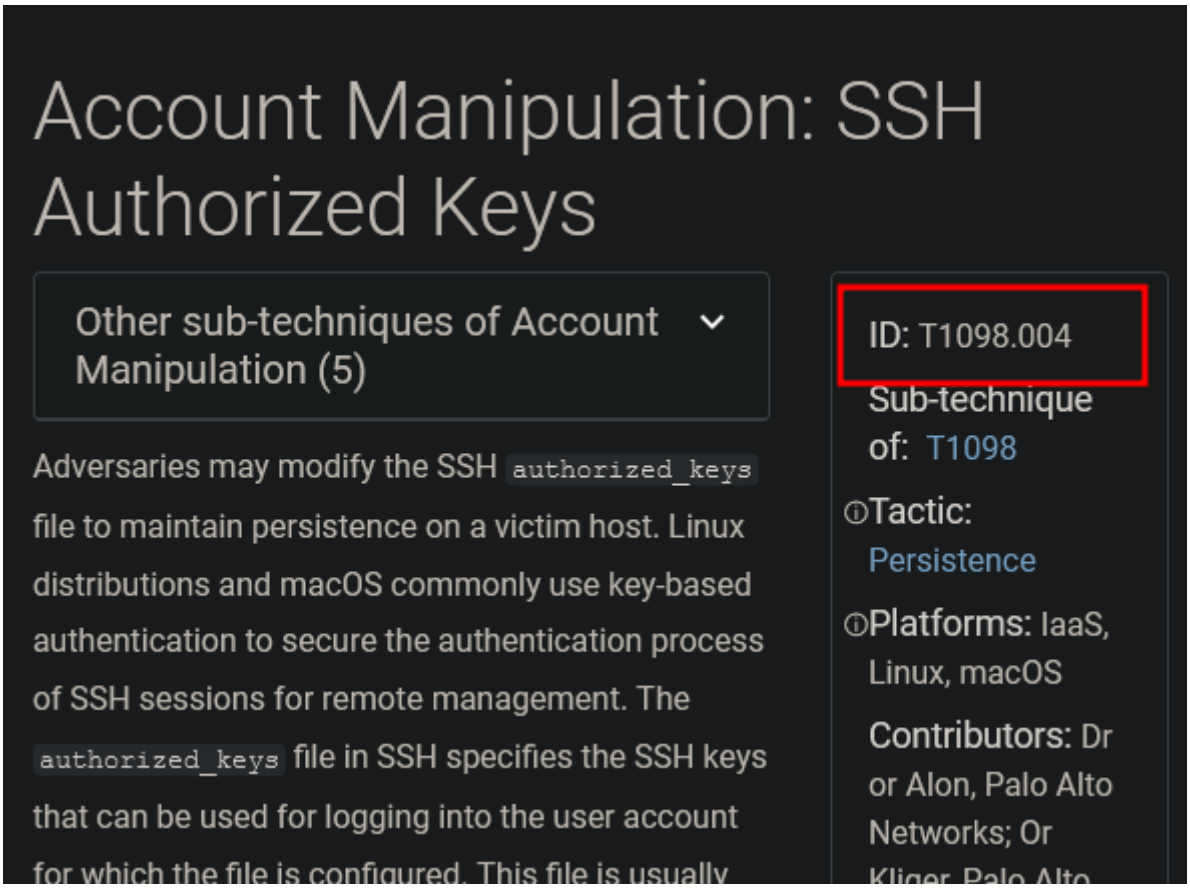
As seen in the http requests with the “rce” get parameter, we saw that attacker appended contents of their public ssh keys(fetched directly from pastes,io) to user ubuntu authorized_keys file which stores the public key of allowed ssh connections.



Answer : /home/ubuntu/.ssh/authorized_keys

11-Can you confirm the MITRE technique ID of this type of persistence mechanism?

As can be seen in mitre att&ck



https://attack.mitre.org/techniques/T1098/004/

Answer : T1098.004