A DETAILED GUIDE FOR THE INSTALLATION AND RUNNING OF THE ARCULUS ZERO TRUST DRONE TESTBED AND APPLICATION

I. (Optional) Installing the Community Honey Network (CHN) Server

The CHN Server is an open-source application used to deploy honeypots on any devices on our network. These honeypots can be used to monitor activity from malicious users, develop threat intelligence based on these actions, build attack signatures, and achieve active defense by pretense. While this is an additional plugin that delivers utility to Arculus, its installation is optional, and all other functionalities will work just fine without it.

1. First, on a new Ubuntu device to host the CHN Server, update the APT package library using the command **`sudo apt update`**. Then install the required packages using the command **`sudo apt-get install -y python3-validators docker.io.`** Install Docker Compose using the below commands:

   ```
   sudo curl -L
   "https://github.com/docker/compose/releases/download/v2.0.1/docker-compose-
   $(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

   sudo chmod +x /usr/local/bin/docker-compose

   sudo systemctl restart docker
   ```

```
ubuntu@ip-172-31-7-86:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [318 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [82.9 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [5676 B]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [319 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [134 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [12.6 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [265 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [63.3 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [3668 B]
Get:14 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [247 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [107 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [9220 B]
Fetched 1819 kB in 1s (1789 kB/s)
Reading package lists... Done
```

```
ubuntu@ip-172-31-7-86:~$ sudo apt-get install -y python3-validators docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (24.0.7-0ubuntu4).
The following NEW packages will be installed:
  python3-decorator python3-validators
0 upgraded, 2 newly installed, 0 to remove and 21 not upgraded.
Need to get 25.3 kB of archives.
After this operation, 114 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-decorator all 5.1.1-5 [10.1 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 python3-validators all 0.20.0-2 [15.2 kB]
Fetched 25.3 kB in 0s (709 kB/s)
Selecting previously unselected package python3-decorator.
(Reading database ... 106389 files and directories currently installed.)
Preparing to unpack .../python3-decorator_5.1.1-5_all.deb ...
Unpacking python3-decorator (5.1.1-5) ...
Selecting previously unselected package python3-validators.
Preparing to unpack .../python3-validators_0.20.0-2_all.deb ...
Unpacking python3-validators (0.20.0-2) ...
Setting up python3-decorator (5.1.1-5) ...
Setting up python3-validators (0.20.0-2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...
```

2. Clone the CHN Server git repository and enter the root directory of the code.

```
ubuntu@ip-172-31-21-212:~$ sudo mkdir -p /opt && sudo git clone https:/
Cloning into '/opt/chnserver'...
remote: Enumerating objects: 221, done.
remote: Counting objects: 100% (99/99), done.
remote: Compressing objects: 100% (69/69), done.
remote: Total 221 (delta 61), reused 63 (delta 29), pack-reused 122
Receiving objects: 100% (221/221), 58.03 KiB | 6.45 MiB/s, done.
Resolving deltas: 100% (130/130), done.
ubuntu@ip-172-31-21-212:~$ cd /opt/chnserver
ubuntu@ip-172-31-21-212:/opt/chnserver$
```

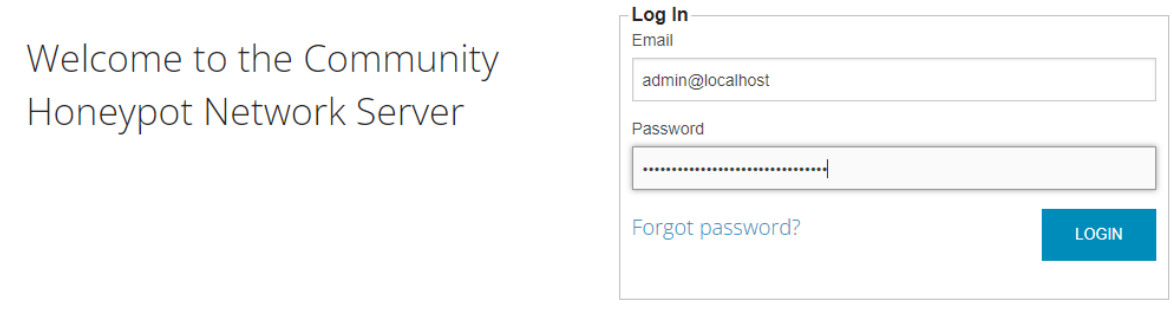
3. Configure the CHN Server using the command **`./guided_docker_compose.py`**, supply it with the public domain name of the host machine and set all other options as shown below.

```
ubuntu@ip-172-31-21-212:/opt/chnserver$ sudo ./guided_docker_compose.py
Please enter the URL where you'd like your CHN web console available.  Note that the domain must be resolvab
Domain: ec2-54-221-98-176.compute-1.amazonaws.com
Please enter a Certificate Strategy.  This should be one of:

CERTBOT: Signed certificate by an ACME provider such as LetsEncrypt.  Most folks will want to use this.  You
tion here
BYO: Bring Your Own.  Use this if you already have a signed cert, or if you want a real certificate without
SELFSIGNED: Generate a simple self-signed certificate
Certificate Strategy: SELFSIGNED
Wrote file to config/sysconfig/chnserver.env

How many days of honeypot data should be maintained in the database (default 30 days)?
Number of Days: 30
Wrote file to config/sysconfig/mnemosyne.env
Do you wish to enable logging to a remote CIFv3 server? [y/N]: N
Do you wish to enable logging to a local file? [y/N]: y

splunk: Comma delimited key/value logging format for use with Splunk
json: JSON formatted log format
arcsight: Log format for use with ArcSight SIEM appliances
json_raw: Raw JSON output from hpfeeds. More verbose that other formats, but also not normalized. Can genera
Logging Format: json
Wrote file to config/sysconfig/hpfeeds-logger.env
Do you wish to enable intelligence feeds from a remote CIF instance? [y/N]: N
ubuntu@ip-172-31-21-212:/opt/chnserver$
```

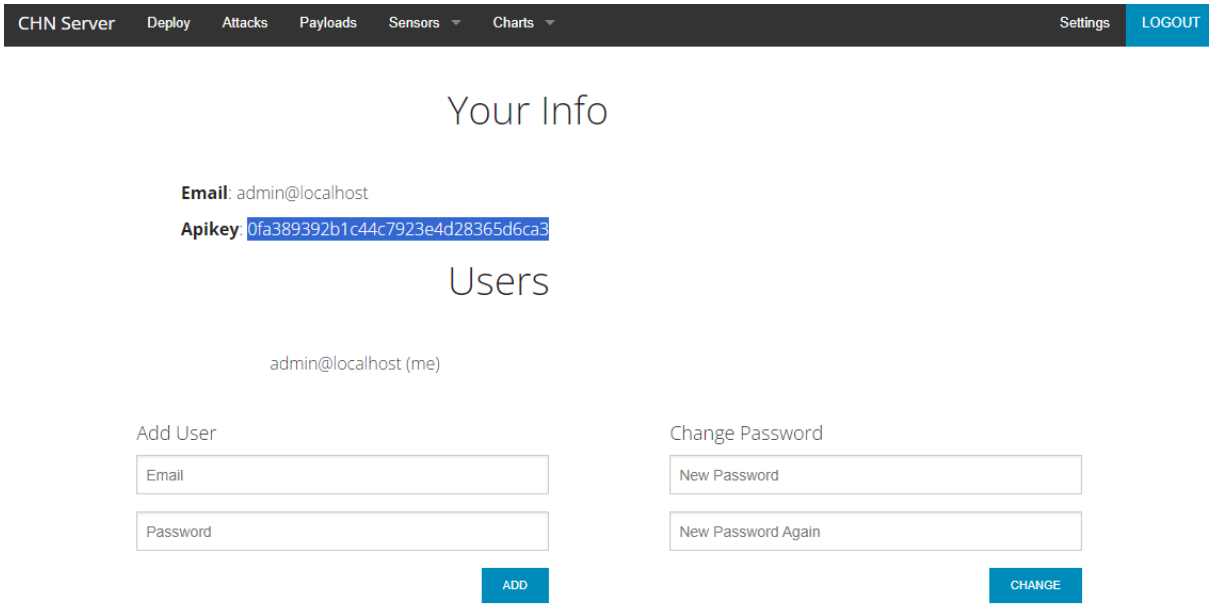
4. Quickstart the CHN Server on Docker Compose using the command **`sudo docker-compose up -d`**, and fetch the user credentials using the command **`grep SUPERUSER /opt/chnserver/config/sysconfig/chnserver.env`**

```
ubuntu@ip-172-31-21-212:/opt/chnserver$ sudo docker-compose up -d
[+] Running 6/6
 ⠿ Container chnserver-redis-1           Started
 ⠿ Container chnserver-mongodb-1         Started
 ⠿ Container chnserver-hpfeeds3-1        Started
 ⠿ Container chnserver-chnserver-1       Started
 ⠿ Container chnserver-mnemosyne-1       Started
 ⠿ Container chnserver-hpfeeds-logger-1  Started
ubuntu@ip-172-31-21-212:/opt/chnserver$ grep SUPERUSER /opt/chnserver/config/sysconfig/chnserver.env
SUPERUSER_EMAIL=admin@localhost
SUPERUSER_PASSWORD=5j1VQOwhdpEf7fSiLUNddW7FPzYOl4Fy
ubuntu@ip-172-31-21-212:/opt/chnserver$
```

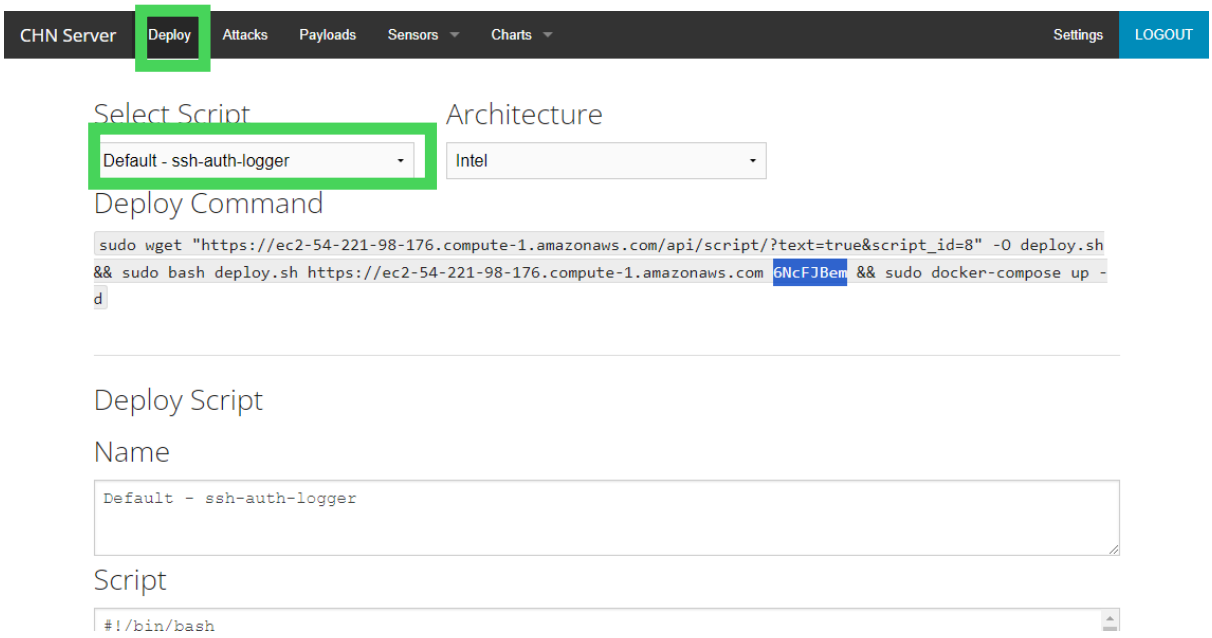
5. Access the CHN Server web application using the public domain of the host machine, and login using the credentials fetched in the previous step.



6. Once logged in, go to the settings page of the application and note down the API key of the CHN Server. This will be used in the future setup of the Arculus application.



7. Click the Deploy tab on the top bar and select script for any honeypot type. A script will be generated containing an 8-letter string as shown in the screenshot below. This is the deploy key of the CHN Server. This will allow authorized deployment of honeypots through Arculus while utilizing the capabilities of the CHN Server.



II. Setting up the Arculus testbed and application

1. On a new Ubuntu host machine, configure the below inbound firewall rules to allow effective communication among the key components of the Arculus testbed and application.

`	**Transport-layer Protocol**	**Ports**	**Description**
HTTPS	TCP	443	HTTPS communication
HTTP	TCP	80	HTTP communication
SSH	TCP	22	Remote SSH access
TCP	TCP	179	K3s worker nodes communication access
TCP	TCP	3000 – 3003	Web application UI and API access
TCP	TCP	5000	Honeypot deployment APIs
TCP	TCP	6006	Documentation pages
TCP	TCP	10250	Kubernetes master node communication
UDP	UDP	8285, 8472	K3s worker nodes communication access

Here is an implementation of these firewall settings on AWS EC2 resources configured using security group rules.

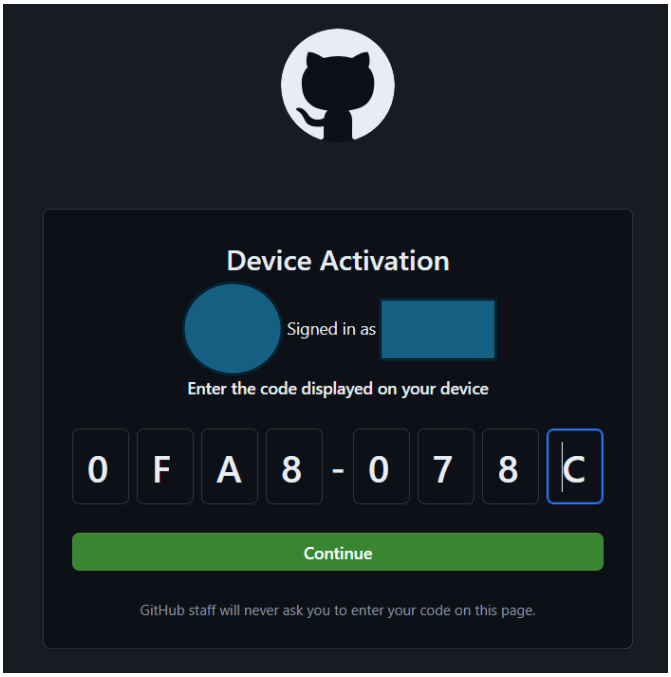
Inbound rules Info

Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
HTTPS	TCP	443	Custom	0.0.0.0/0		Delete
HTTP	TCP	80	Custom	0.0.0.0/0		Delete
Custom UDP	UDP	8285	Custom	0.0.0.0/0	kubernetes communication	Delete
SSH	TCP	22	Custom	0.0.0.0/0		Delete
Custom TCP	TCP	6006	Custom	0.0.0.0/0	documentation pages	Delete
Custom TCP	TCP	179	Custom	0.0.0.0/0	Worker nodes comm	Delete
Custom TCP	TCP	5000	Custom	0.0.0.0/0	Honeypots	Delete
Custom TCP	TCP	10250	Custom	0.0.0.0/0	Master node	Delete
Custom UDP	UDP	8472	Custom	0.0.0.0/0	kubernetes worker communications	Delete
Custom TCP	TCP	3000 - 3003	Custom	0.0.0.0/0	applications ports	Delete

2. Install “gh”, a GitHub utility for easy authentication to be able to clone the private Arculus repository as an authorized user. Then, using the command **`gh auth login`**, authenticate the user through a browser using a one-time code.

```
ubuntu@ip-172-31-26-207:~$ sudo apt install gh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  gh
0 upgraded, 1 newly installed, 0 to remove and 26 n
Need to get 8826 kB of archives.
After this operation, 45.4 MB of additional disk sp
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubunt
Fetched 8826 kB in 0s (32.3 MB/s)
Selecting previously unselected package gh.
(Reading database ... 67739 files and directories c
Preparing to unpack .../gh_2.45.0-1ubuntu0.1_amd64.
Unpacking gh (2.45.0-1ubuntu0.1) ...
Setting up gh (2.45.0-1ubuntu0.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...
```

```
ubuntu@ip-172-31-26-207:~$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 0FA8-078C
Press Enter to open github.com in your browser...
! Failed opening a web browser at https://github.com/login/device
  exec: "xdg-open,x-www-browser,www-browser,wslview": executable file not
  Please try entering the URL in your browser manually
```



3. Clone the GitHub repository as an authorized user and enter the root directory of the codebase.

```
ubuntu@ip-172-31-26-207:~$ git clone https://github.com/mizzouceri/Arculus.git
Cloning into 'Arculus'...
remote: Enumerating objects: 688, done.
remote: Counting objects: 100% (688/688), done.
remote: Compressing objects: 100% (597/597), done.
remote: Total 688 (delta 76), reused 688 (delta 76), pack-reused 0
Receiving objects: 100% (688/688), 36.51 MiB | 48.68 MiB/s, done.
Resolving deltas: 100% (76/76), done.
ubuntu@ip-172-31-26-207:~$ cd Arculus/
ubuntu@ip-172-31-26-207:~/Arculus$
```

4. Setup the execute permissions of the automated application script using the command **`sudo chmod +x arculus-setup.sh`** and execute the script using the command **`./arculus-setup.sh`**

```
ubuntu@ip-172-31-26-207:~/Arculus$ sudo chmod +x arculus-setup.sh
./arculus-setup.sh
Starting setup process...
Running initial setup script...
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
```

5. Furnish configuration information as prompted. First, provide a password to protect the Arculus MySQL database. This will be set to the database, as well as securely shared with the application to make the database accessible to the backend.

```
[INFO]  systemd: Starting k3s
Enter database password:
cericenter
Installing Docker and NPM...
Hit:1 http://us-east-1.ec2.archive.ubunt
Hit:2 http://us-east-1.ec2.archive.ubunt
Hit:3 http://us-east-1.ec2.archive.ubunt
Hit:4 http://security.ubuntu.com/ubuntu
Reading package lists... Done
```

6. Use a Random Key Generator tool to generate a 256-bit encryption key and supply it to the application. This is the secret that will be used for secure signing of the JWT authentication tokens. Further, provide the details (Server URL, API key and deploy key) of the CHN Server for letting Arculus access the capabilities of the CHN Server.

```
Starting MySQL container in detached mode...
eed2479125fb4545c020d26124c37b1f8c7205d68a1259892e12cc48db906da3
Enter a 256-bit encryption key:
ICKEGP6S0uhdFcXZfSatHSqRyvkse8oF
Enter CHN Server URL (leave blank if not available):
ec2-44-193-76-143.compute-1.amazonaws.com
Enter CHN API key (leave blank if not available):
d568c953fe7c41b6b5549b2b3e105230
Enter CHN deploy key (leave blank if not available):
LEYMtqP9
```

7. After all the configuration steps, the Node.js backend and the React.js UI will be deployed, and the application will be accessible at **`http://{arculusHostMachinePublicIPAddress}/3000`**

```
[PM2] Spawning PM2 daemon with pm2_home=/home/ubuntu/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ubuntu/Arculus/arculus-gcs-node/index.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	namespace	version	mode	pid	uptime	↺	status
0	node-app	default	1.0.0	fork	13320	0s	0	online

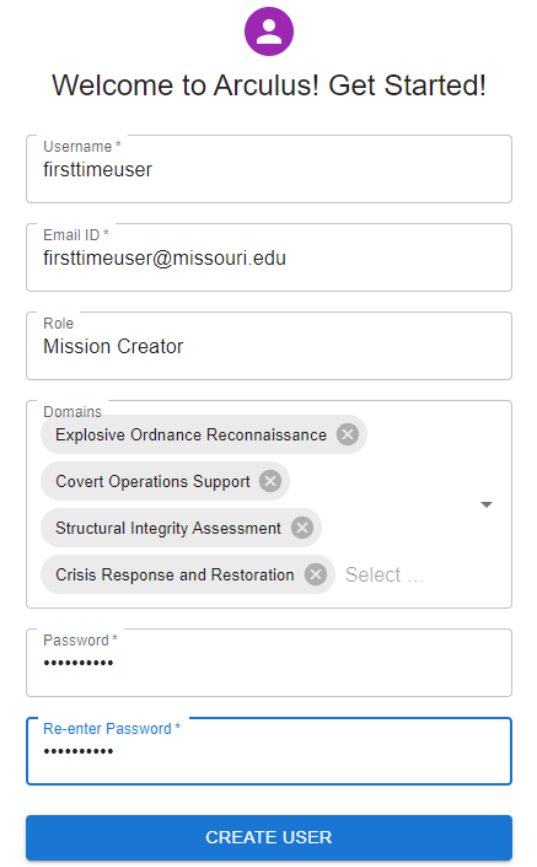
```
Starting UI server with PM2...
[PM2] Starting /home/ubuntu/Arculus/arculus-gcs-ui/server.js in fork_mode (1 instance)
[PM2] Done.
```

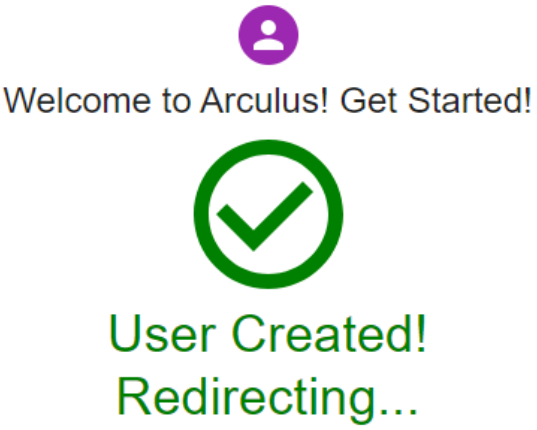
id	name	namespace	version	mode	pid	uptime	↺	status
0	node-app	default	1.0.0	fork	13320	2m	0	online
1	ui-server	default	0.1.0	fork	17925	0s	0	online

```
Arculus successfully setup! Access the Arculus UI at http://52.23.225.56:3000
```

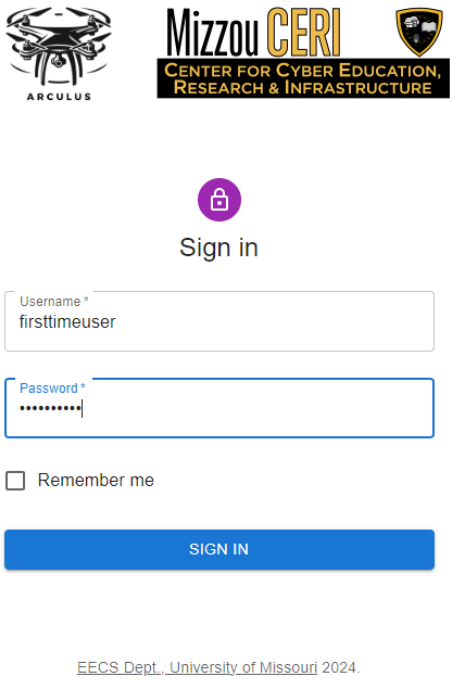
III. Navigating through the User and Device Management Dashboards of the Arculus Application

1. On visiting the UI's web URL for the first time before creating any users, the application takes us to a "Getting Started" page where users will be prompted to create the first ever user, also a super user. Once the first user is created, this page will be non-functional and only insiders with "Mission Creator" privileges will be able to add new users to the application.

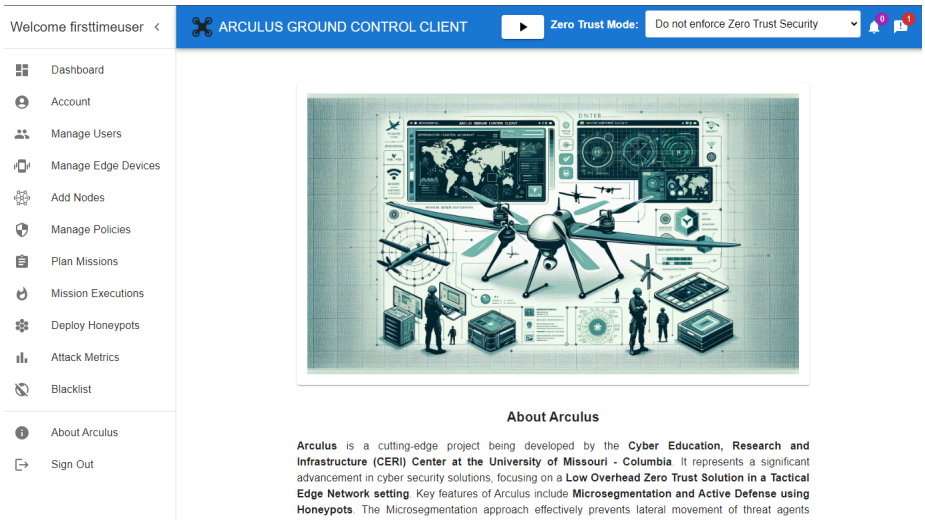




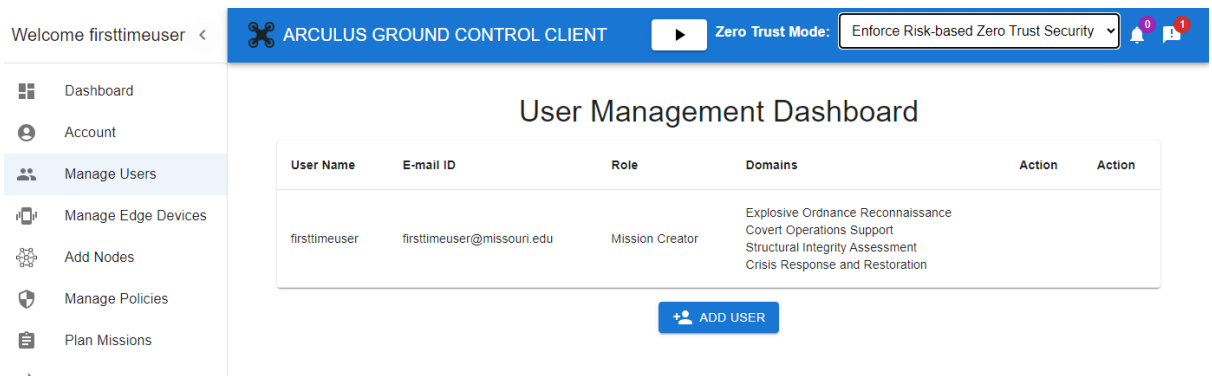
2. After the user is created, the page redirects to the login page, where the first ever user or subsequently created users can log in using their credentials.

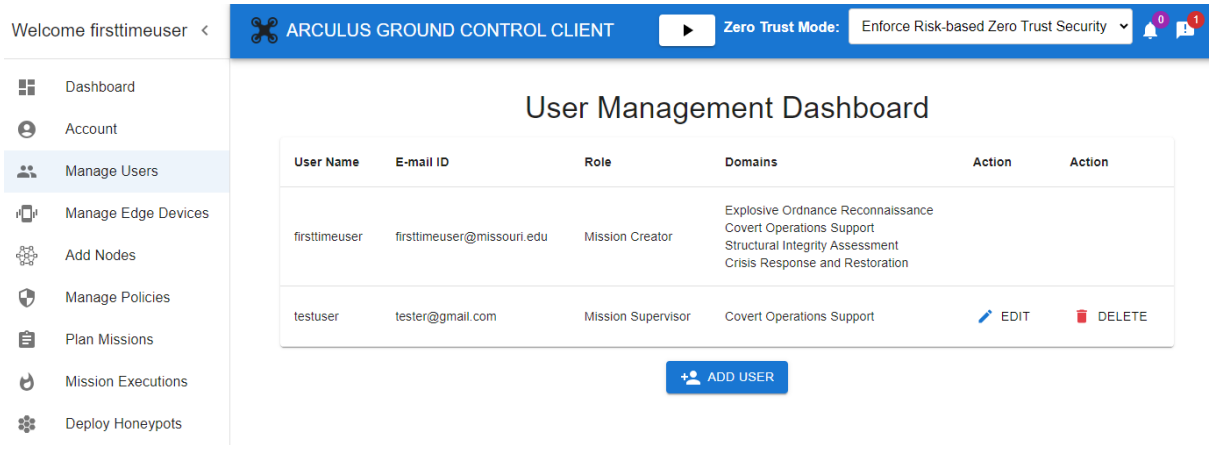


3. On successful password authentication, the user is taken to the “About” page.



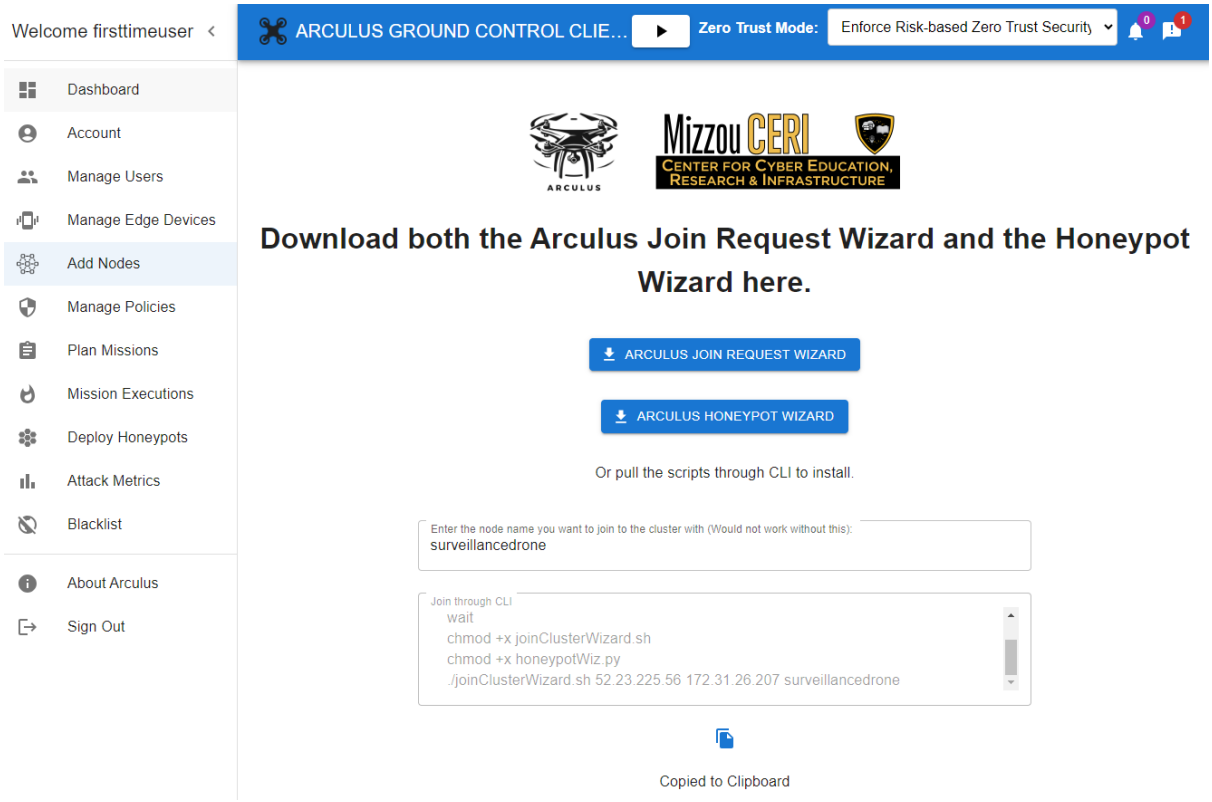
4. The “Manage Users” tab of the Arculus sidebar takes the user to the “User Management Dashboard” where (s)he (only mission creators) can view existing users, create new users, configure and edit their roles and domains, or delete the users. For a demo purpose, we created a user named “testuser” configured with a “Mission Supervisor” role.





5. On signing in as testuser (a Mission Supervisor), we can observe that there are limited dashboards (s)he can navigate to, demonstrating a robust Role-based Access Control for users' interaction with the UI.

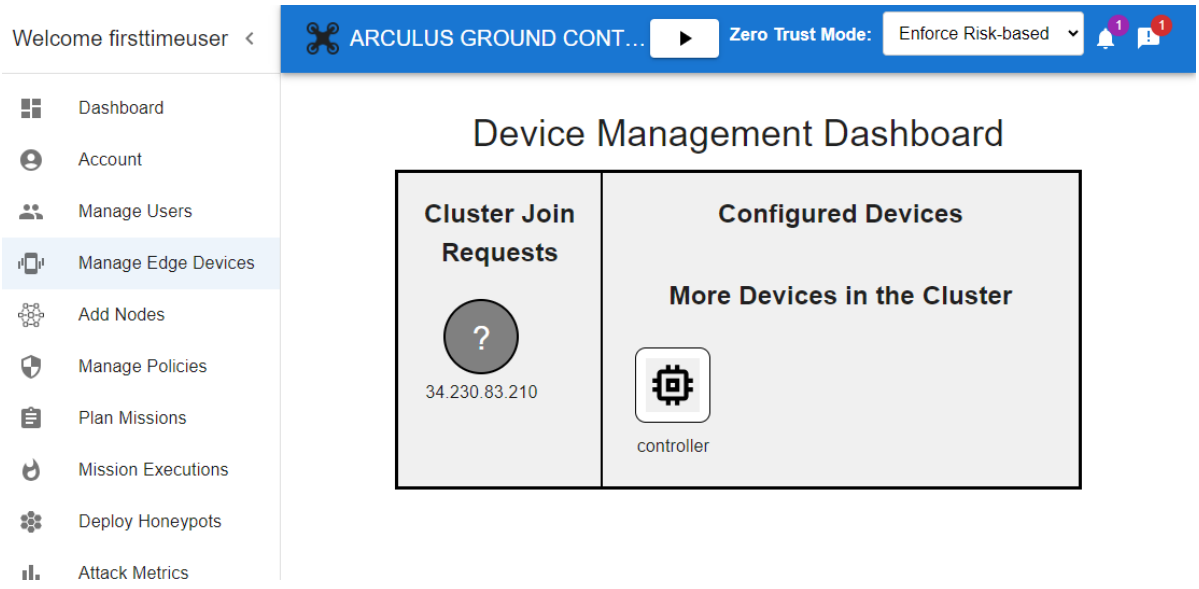
6. The more important functionality of device management begins with the "Add Nodes" tab of the sidebar. This page generates custom scripts for new devices to join the Arculus device cluster. On entering the name we want to give the device that joins the cluster, a script is created with the parameter. This script can be copied to clipboard and be run on the device we want to add to the cluster.



7. The automated script puts in a join request to the Arculus backend. This request pops up on the Device Management Dashboard of the Arculus UI under the “Cluster Join Requests” section.

```
ubuntu@ip-172-31-47-238:~$ sudo su
root@ip-172-31-47-238:/home/ubuntu# curl -o joinClusterWizard.sh http://52.23.225.5
6:3001/tools/downloadJoinWiz &      curl -o honeypotWiz.py http://52.23.225.56:3001/
tools/downloadHoneyWiz &
    wait
    chmod +x joinClusterWizard.sh
    chmod +x honeypotWiz.py
    ./joinClusterWizard.sh 52.23.225.56 172.31.26.207 surveillancedrone
```

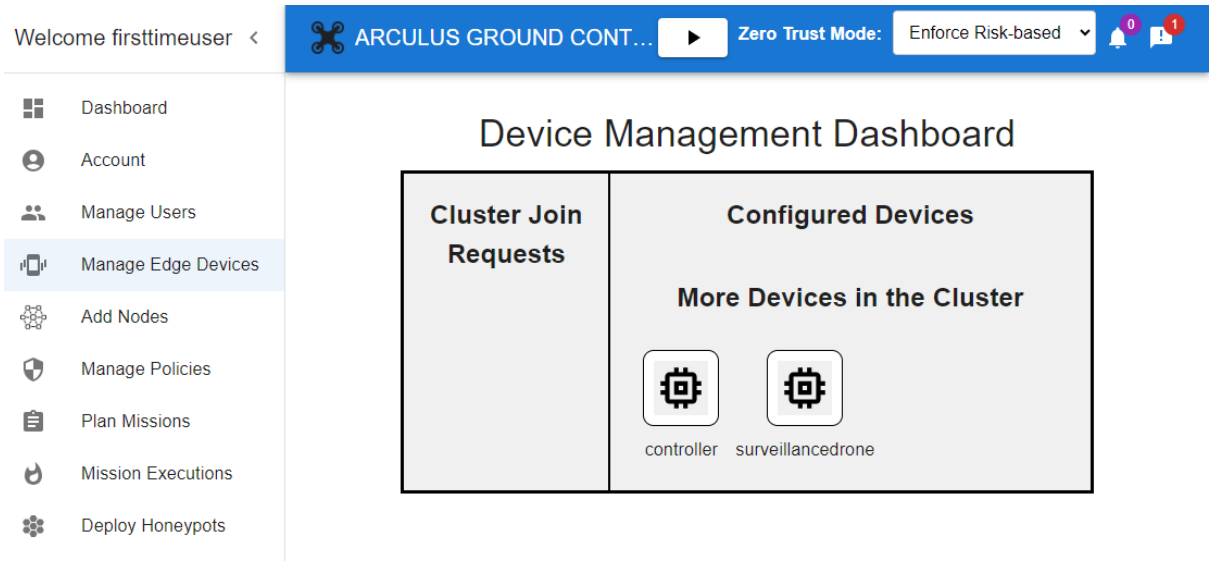
```
0 upgraded, 0 newly installed, 0 to remove and 26 not upgraded.
Cluster join request successful.
Opening websocket to get join status...
ws://52.23.225.56:3002/getJoinStatus?nodeName=surveillancedrone
Received message: Request submitted. Please hang on until an administrator approves your request.
```



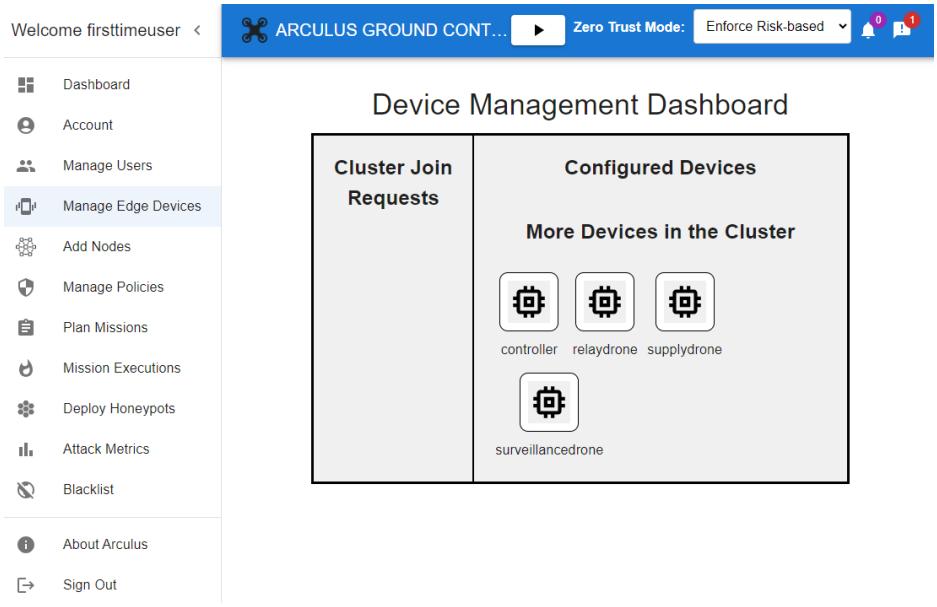
8. Super users (Mission Creators) can view these requests and choose to either approve or decline the request. On approval, the K3s token of the Arculus backend is securely shared with the new device, with which is joins the cluster.


```
Received message: Request submitted. Please hang on until an administrator approves your request.
Join Successful. Proceeding further.
Token received: K102cbcaebf4feecbe1860cf21abe1a5e3d4fba215fa092b3684df4724f8eb56e22::server:12e8262b8
1ea34d0
sudo: /usr/local/bin/k3s-agent-uninstall.sh: command not found
Failed to stop k3s.service: Unit k3s.service not loaded.
Failed to disable unit: Unit file k3s.service does not exist.
[INFO]  Finding release for channel stable
[INFO]  Using v1.29.6+k3s2 as release
[INFO]  Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.29.6+k3s2/sha256sum-amd64
[INFO]  Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.29.6+k3s2/k3s
[INFO]  Verifying binary download
[INFO]  Installing k3s to /usr/local/bin/k3s
[INFO]  Skipping installation of SELinux RPM
[INFO]  Creating /usr/local/bin/kubectl symlink to k3s
[INFO]  Creating /usr/local/bin/crictl symlink to k3s
[INFO]  Creating /usr/local/bin/ctr symlink to k3s
[INFO]  Creating killall script /usr/local/bin/k3s-killall.sh
[INFO]  Creating uninstall script /usr/local/bin/k3s-agent-uninstall.sh
[INFO]  env: Creating environment file /etc/systemd/system/k3s-agent.service.env
[INFO]  systemd: Creating service file /etc/systemd/system/k3s-agent.service
[INFO]  systemd: Enabling k3s-agent unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s-agent.service → /etc/systemd/system/k
.
[INFO]  systemd: Starting k3s-agent
```

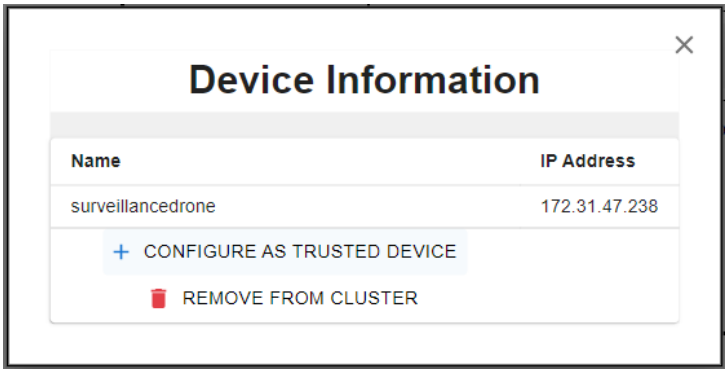
9. Once the new device sets up all the required dependencies on itself and joins the cluster, it shows up on the Device Management Dashboard under the “More Devices in the Cluster” section.



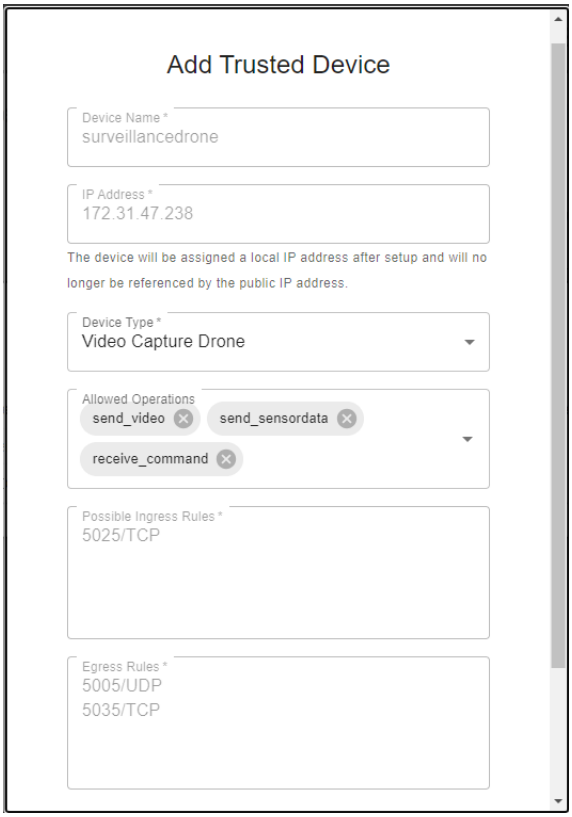
10. As we would need a surveillance drone, a supply drone, a relay drone and a controller, add two more devices to the cluster using the same procedure as above.

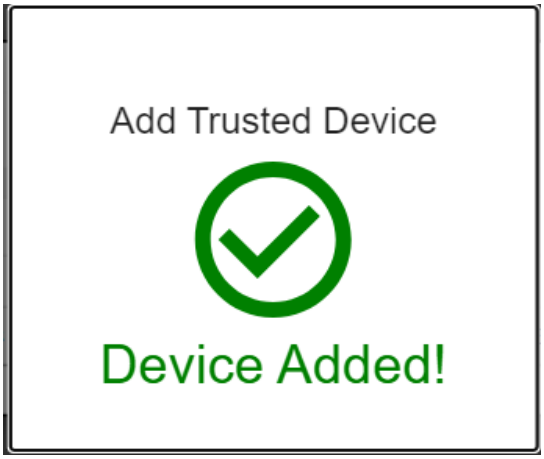


11. On clicking on an device from the “More Devices” section, we can configure them further as trusted devices to attain virtualized behavior by deploying containers on them or remove the devices from the cluster.

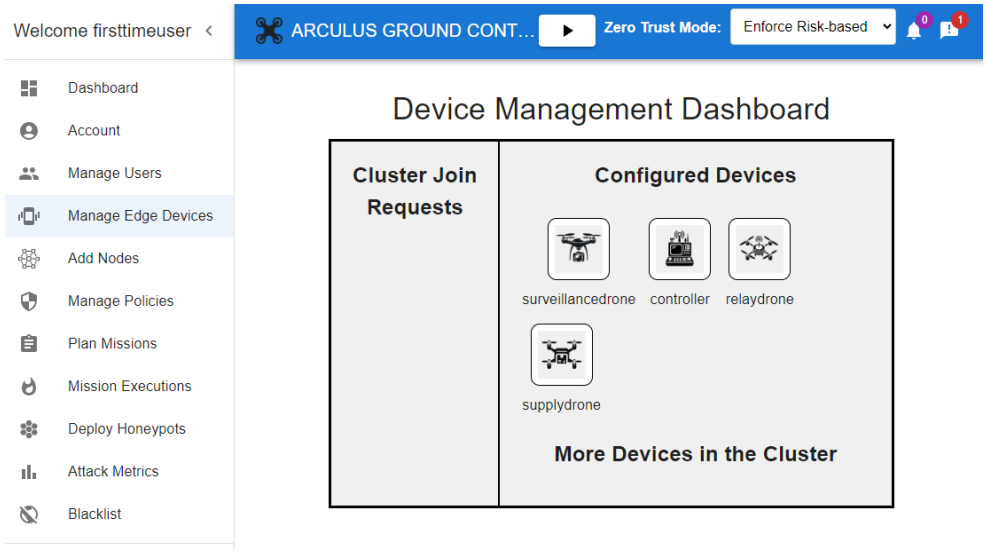


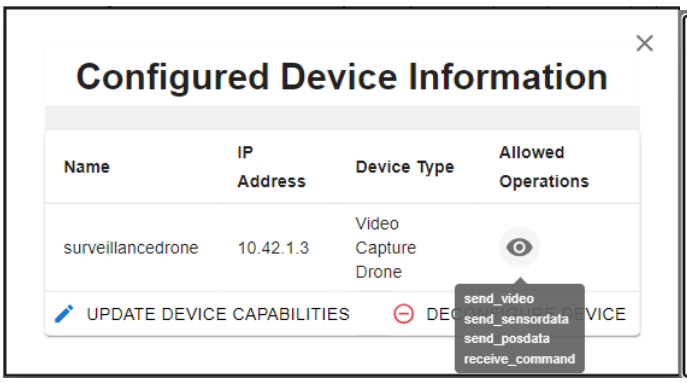
12. By configuring the devices as trusted devices, we can deploy them as certain types of drones/rovers/devices with respective functionalities, and adjust the allowed operations for the devices, which dictate the ingress and egress rules for the devices.

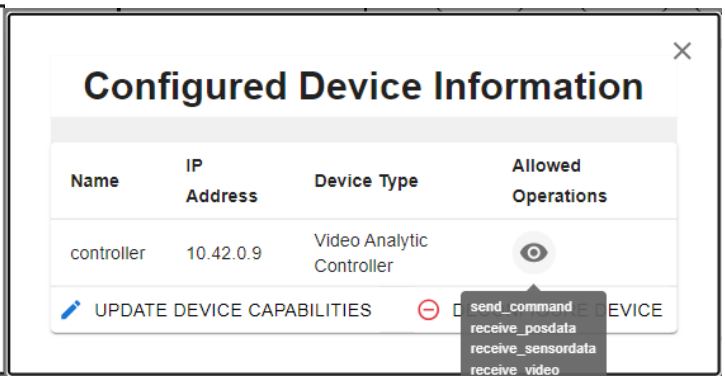


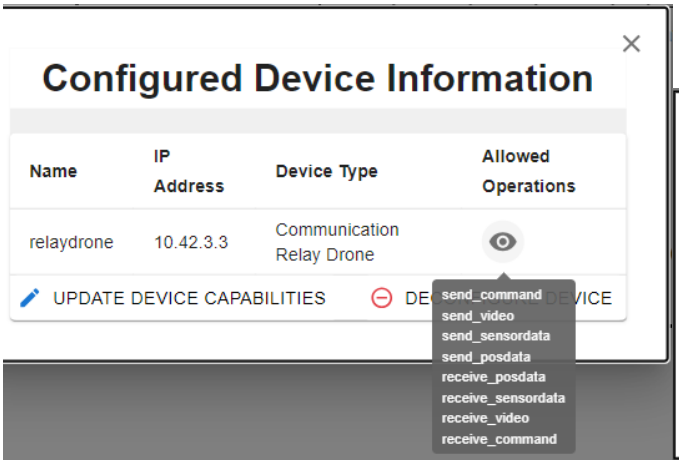


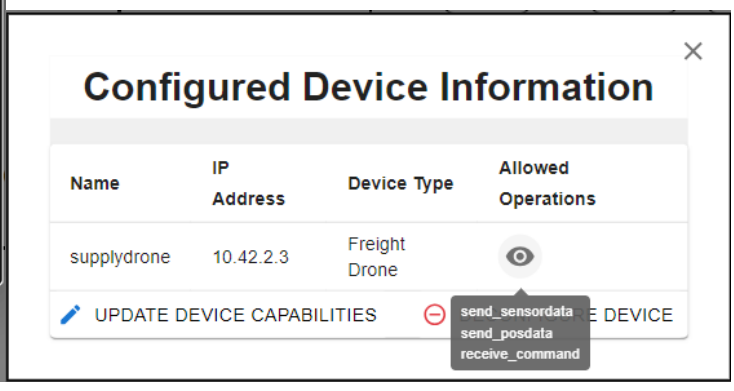
13. Configure all four devices required for the mission execution with their respective drone types and capabilities as shown in the configuration screenshots below.





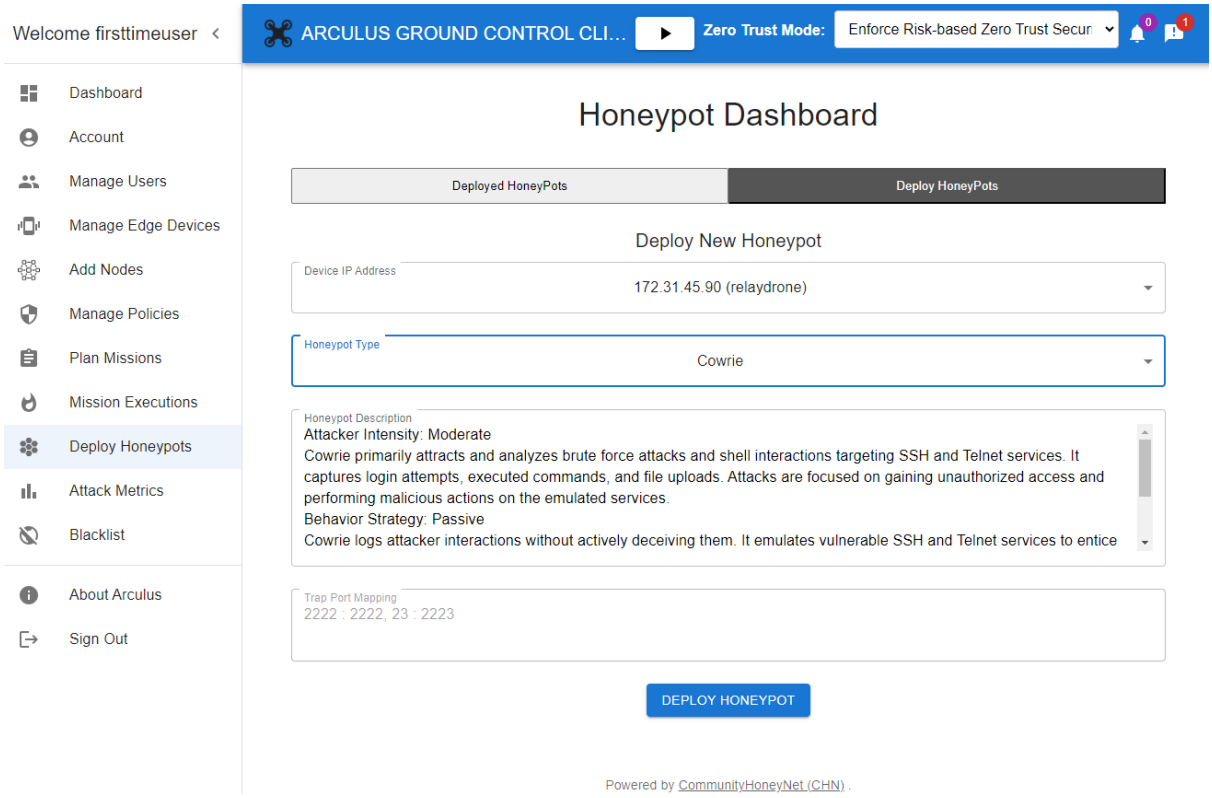




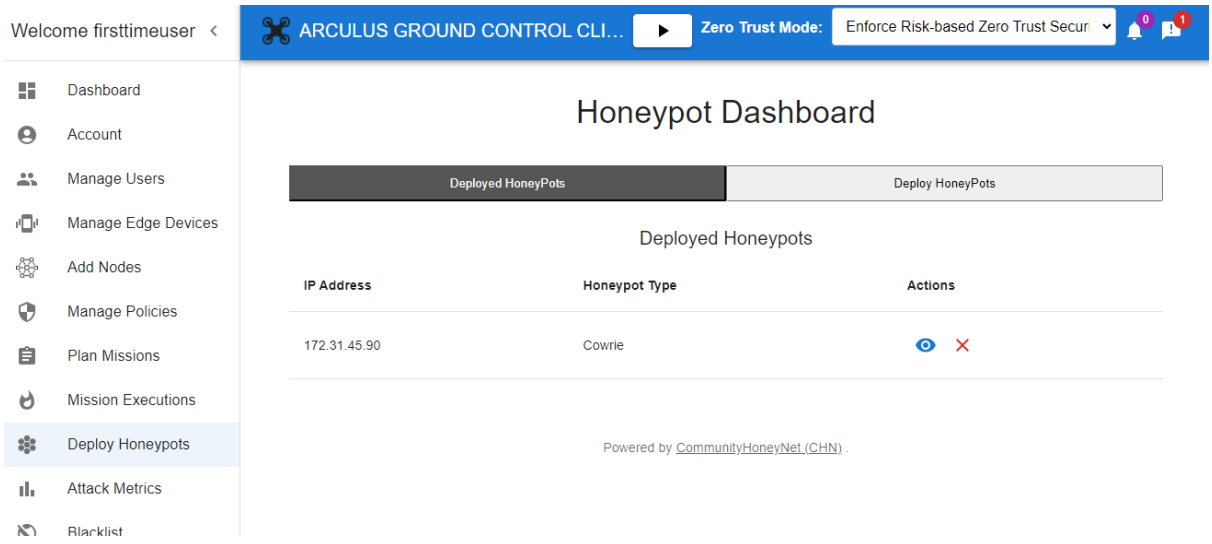


IV. Deploying, Managing and Monitoring Honeypots

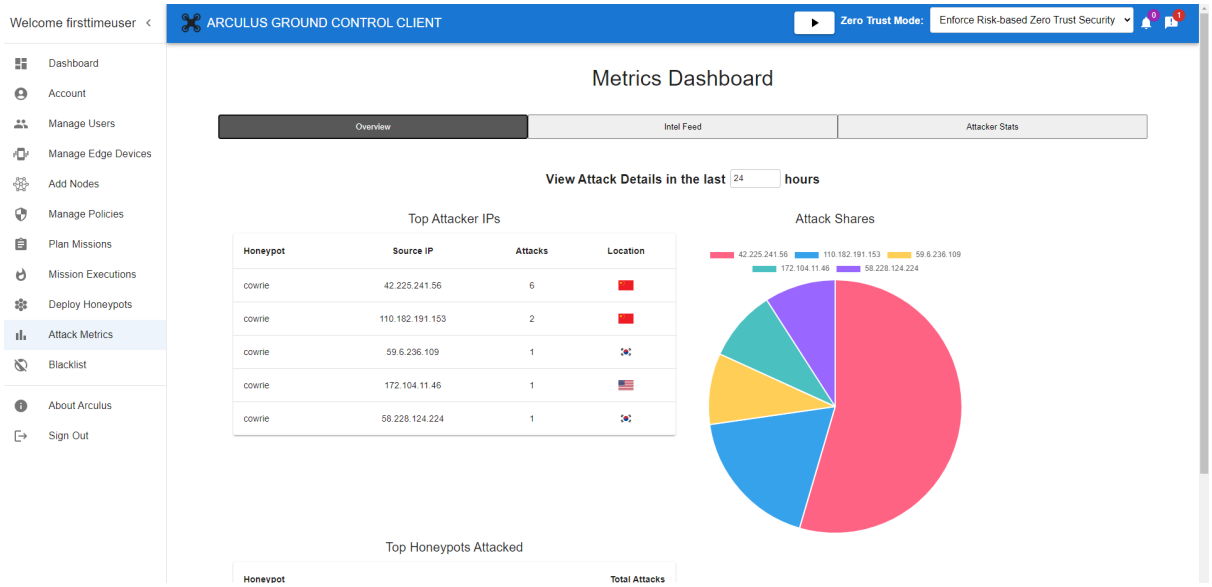
1. The Honeypot dashboard of the Arculus web interface allows for the seamless deployment of multiple types of honeypots. The target device and the honeypot can be selected based on the type of vulnerabilities to be simulated for active defense using the capabilities of the CHN Server.



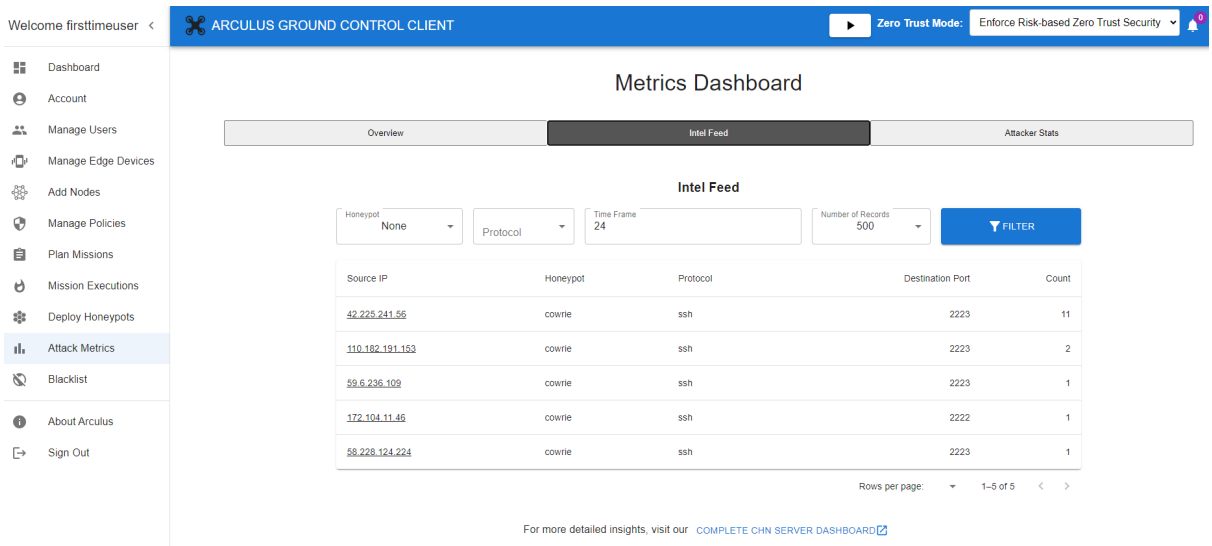
2. The deployed honeypots can be viewed or deleted on the “Deployed Honeypots” pf the Honeypot dashboard.



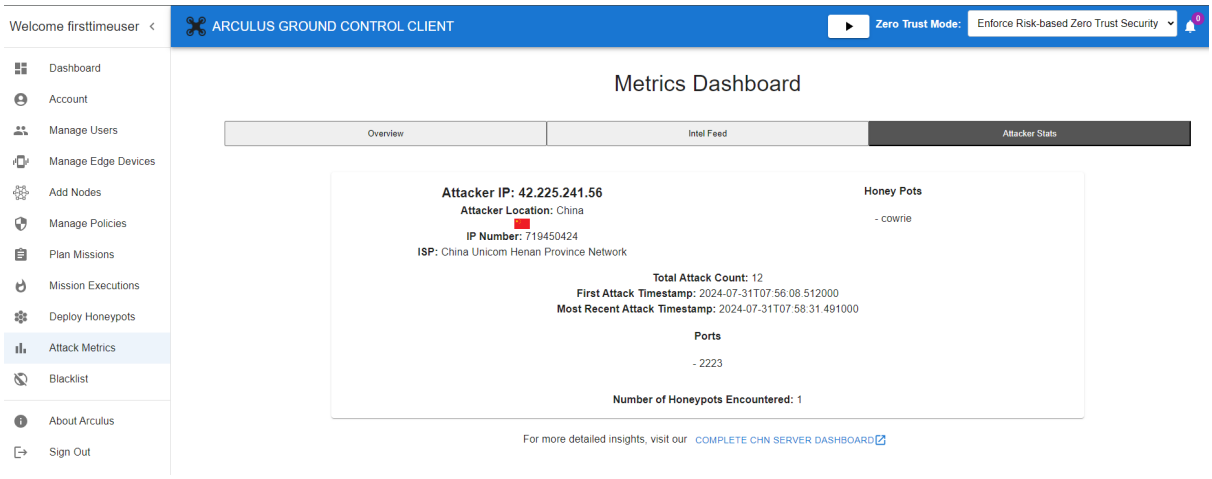
3. On the Metrics Dashboard, we can see the overview of all the attackers and their share of the total malicious activity on our sensors.



4. The Intel Feed gives deep and technical insights into what types of attacks using which protocols and ports were executed on the cluster. This gives us information on the strategies attackers are taking as a pattern to establish attack signatures.

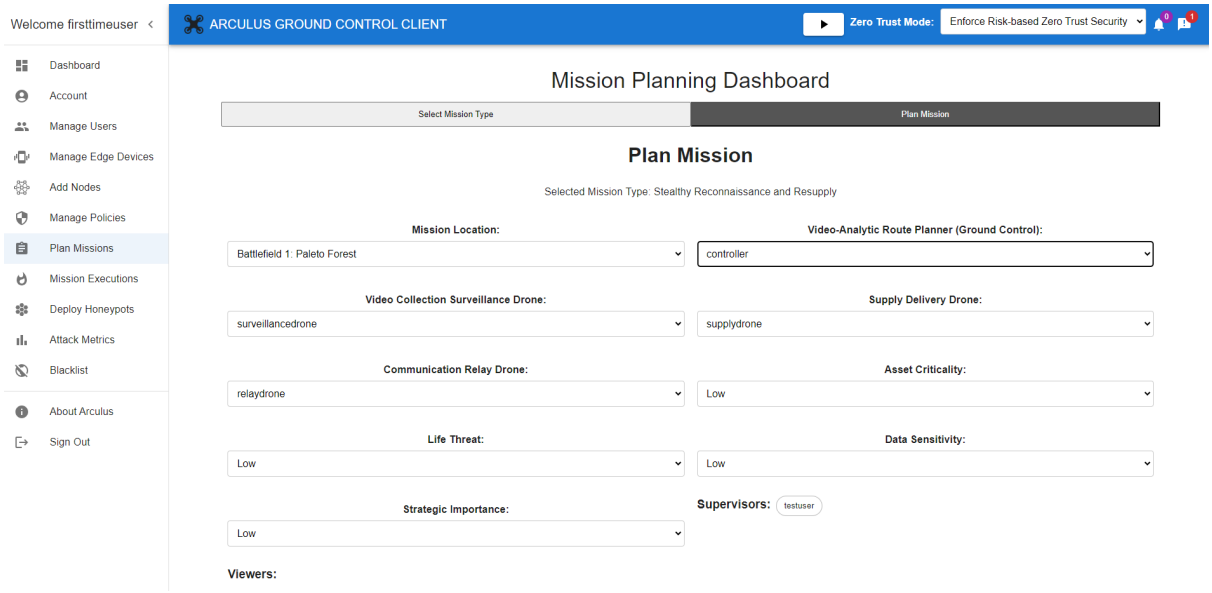


5. We can also view the “Attacker Stats” tab where a detailed information of the source of attacks and the total count of the attack attempts can be seen. These dashboards provide a lot of threat intelligence which is also backed up on the CHN Server for detailed analysis.

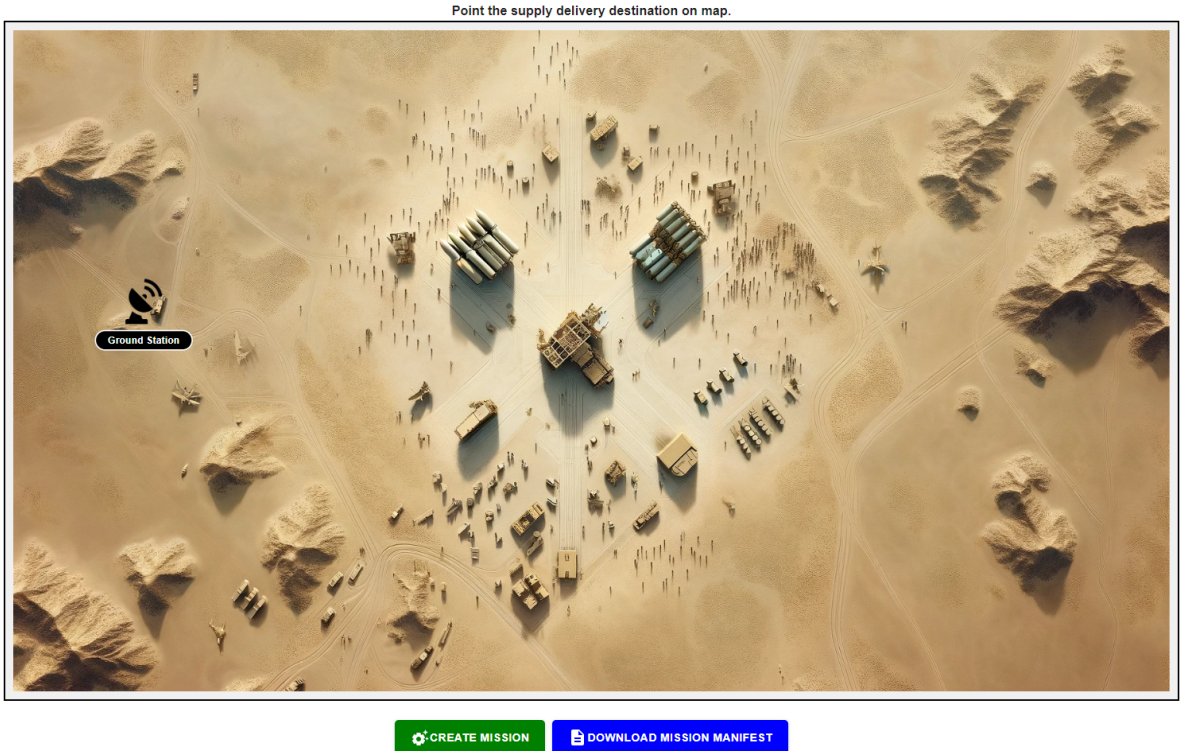


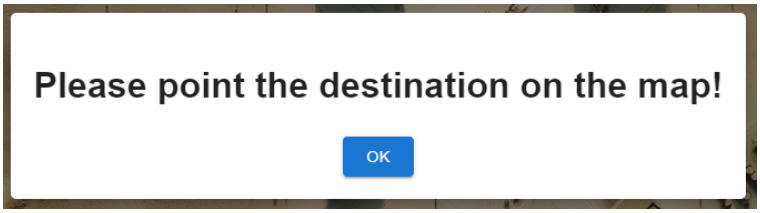
V. Planning and Execution of Drone Missions on the Arculus Testbed

1. The Mission Planning Dashboard allows for configuring drone missions by specifying which devices to use, what the context (criticality) of the mission is, who the supervisors and viewers of the mission are, and logistics information. Configure these options to tune the criticality of the mission dictating the zero trust levels. The device fields are automatically populated based on the availability of the suitable types of devices.



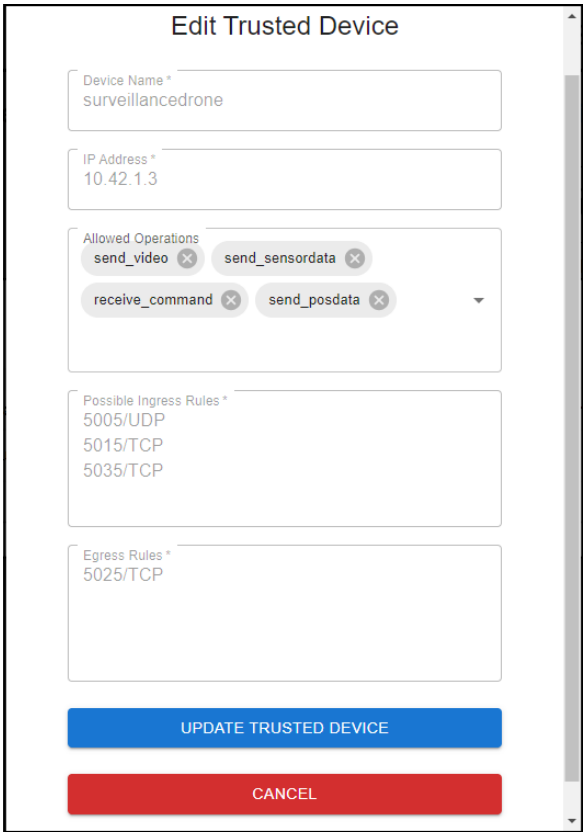
2. The page also provides maps of locations to choose from. On these maps, users can pick a destination point for the delivery of supplies (here, the case of Stealthy Reconnaissance and Resupply mission). Try creating the mission without pointing a destination on the map, and the mission creation will not go ahead asking the user to place a destination on the map.

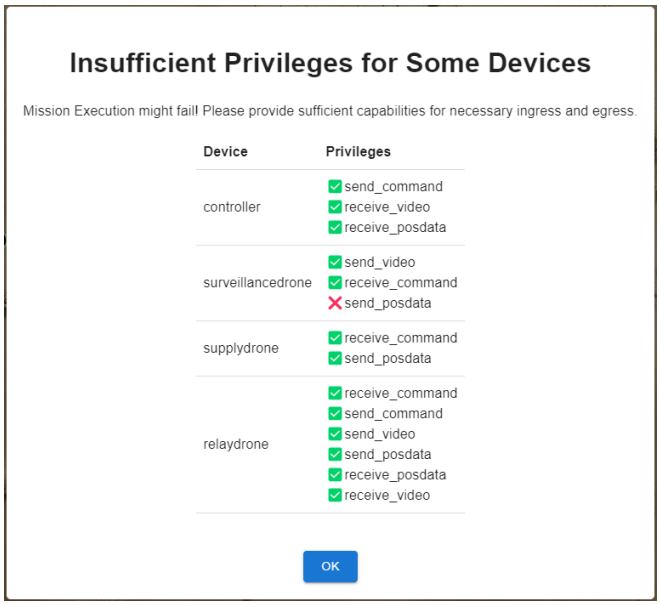




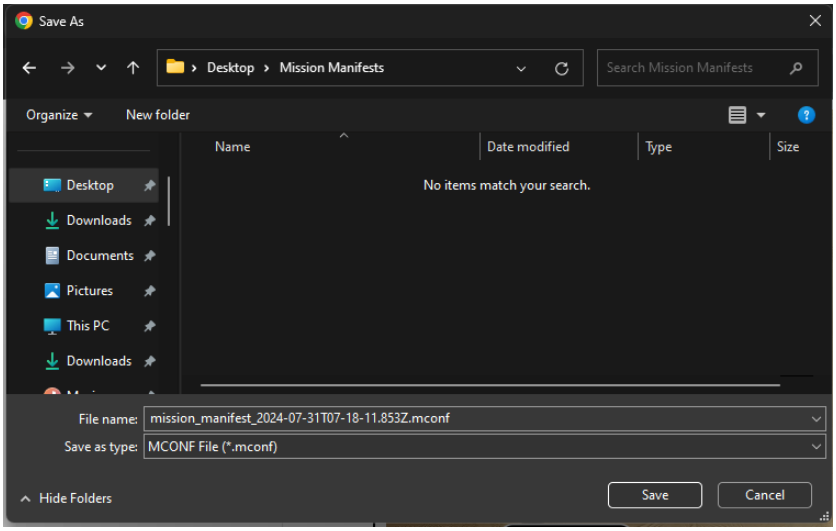
3. Click anywhere on the map to set the supply destination to that location.

4. Try updating the capabilities and remove one of the privileges specified in the device configuration step. The mission creation won't go ahead because, inability of any of the devices to perform their respective functions will cause the mission to fail for sure. This validation mechanism is a proactive approach to missions whose failure cannot be afforded.

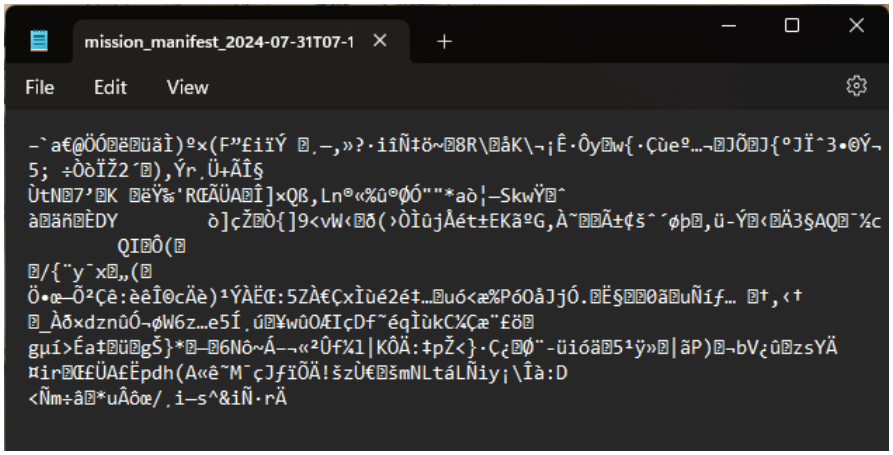


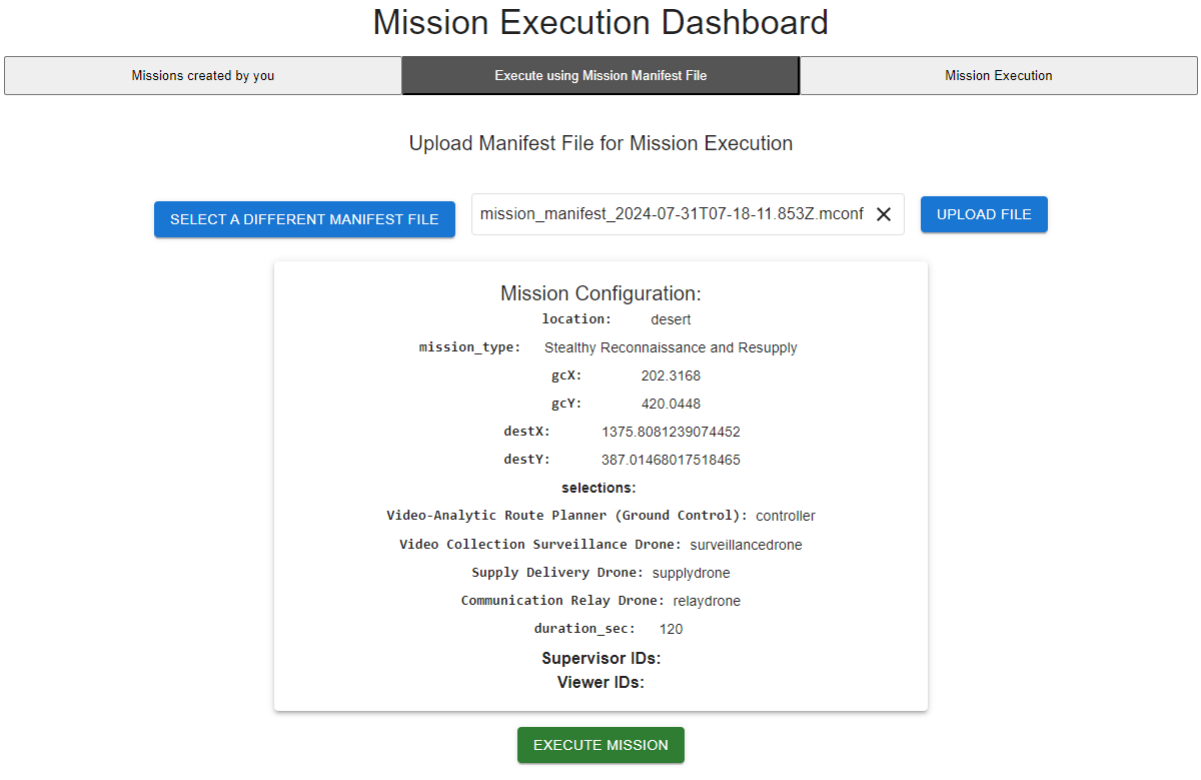


5. Besides normal creation of missions, the “Download Mission Manifest” button allows users to download a repeatable manifest of their mission. This prevents multiple reconfigurations of a mission. The files are encrypted and downloaded with a .mconf extension.

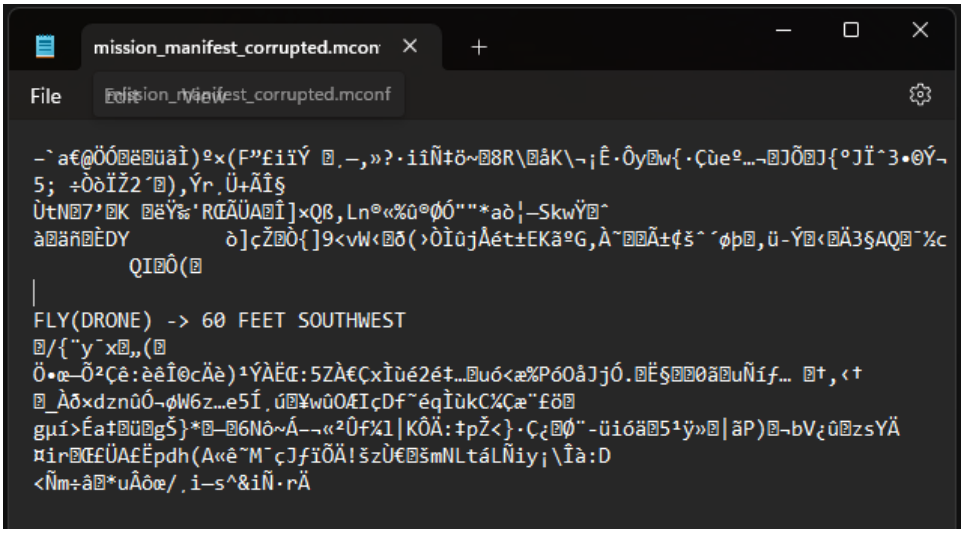


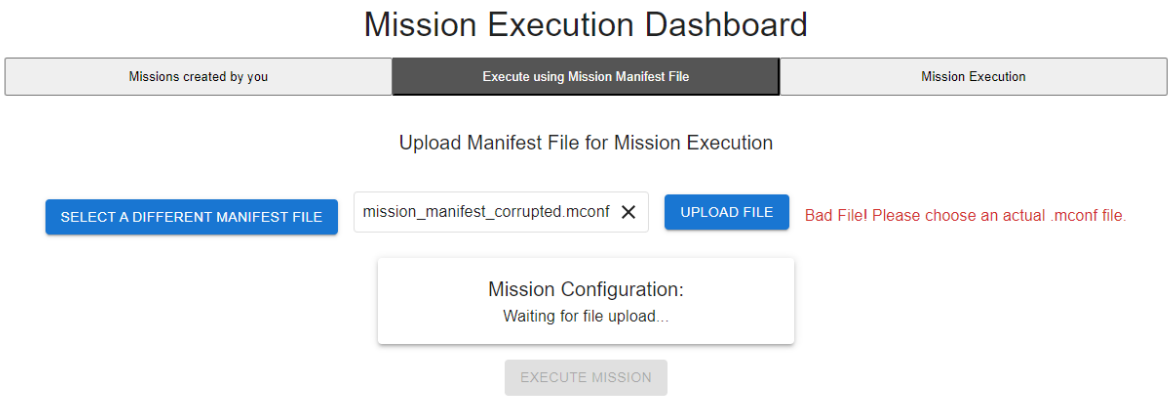
6. From the screenshot below, the manifest files are encrypted and hence, modifications to these files by external actors invalidates the files, leaving no space for deliberate mission adulteration. The next screenshot shows how the Arculus application fetches the mission information from the .mconf file on executing with manifest.



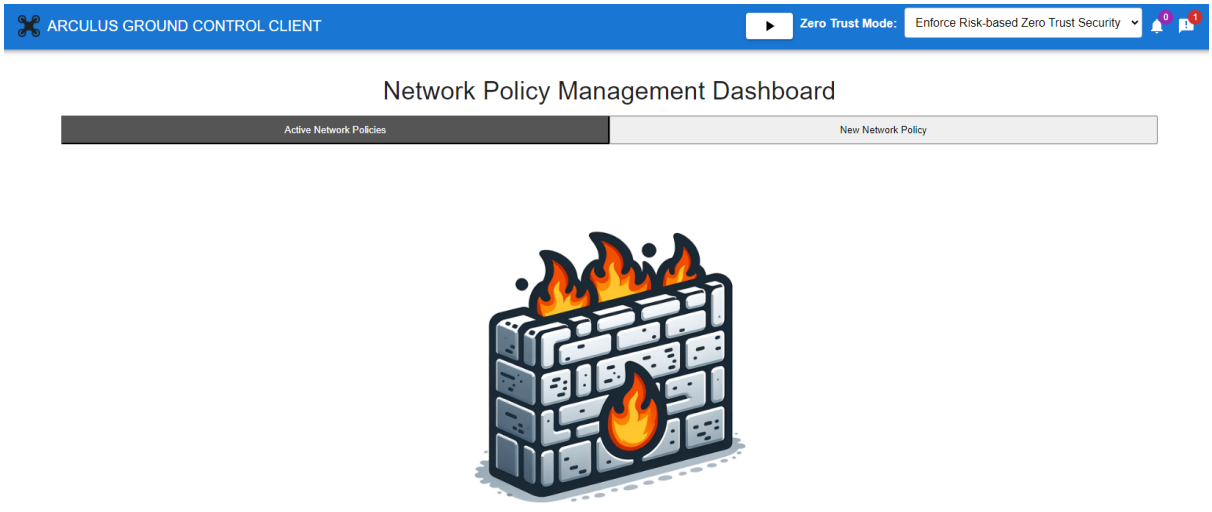


7. However, try adding your own text to the manifest file and try to poison it as shown below. The application throws an error message identifying the manifest file to be bad.

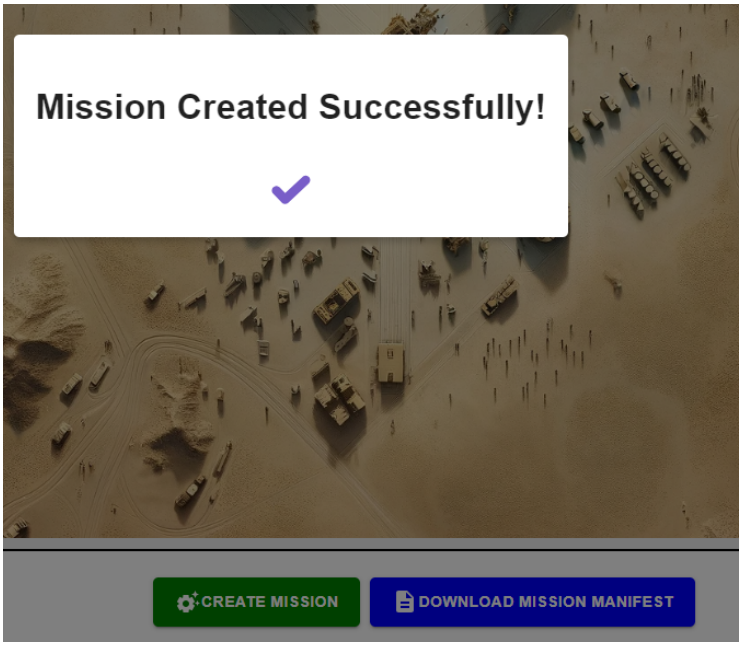


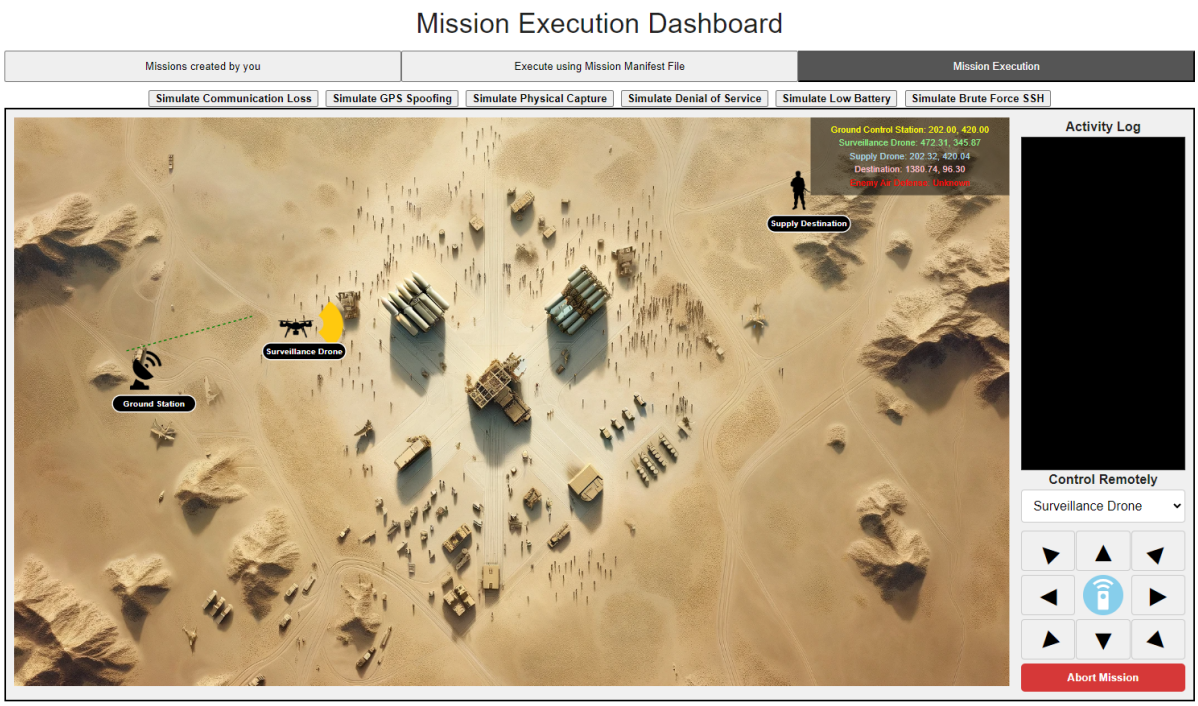


8. Before trying to execute a mission, click the “Manage Policies” tab that takes you to the Network Policy Management Dashboard. It can be seen as in the below screenshot, that there are no active network policies and all traffic is denied by default.

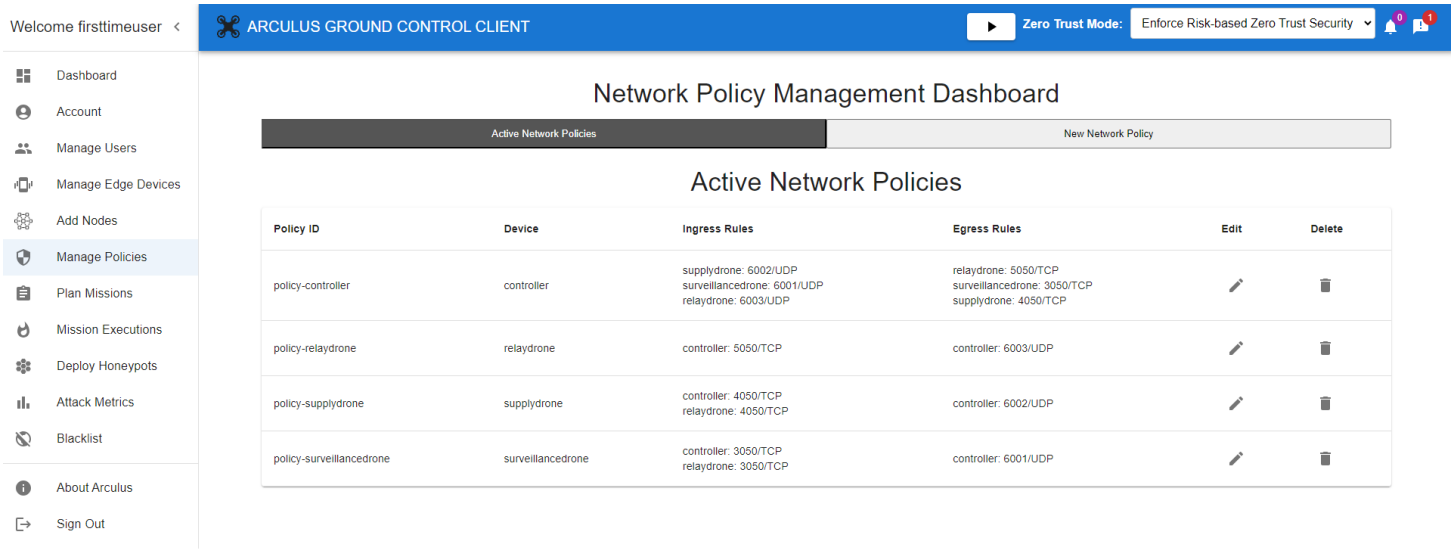


9. Now, let us create a mission on the mission planning dashboard and execute it using the EXECUTE (▷) button on the “Missions created by me” tab. The mission starts executing within seconds and the application takes us to the “Mission Execution” tab where the live mission and its logs can be monitored.





10. It can be observer that, on navigating to the Network Policy Management Dashboard, new policies are created during the mission execution to facilitate communication for respective tasks. However, as soon as the mission execution finishes/terminates, all the network policies are deleted. This demonstrates the extraordinary and dynamic execution of Task-based Access Control (TBAC) for good privilege management for devices on the network.



11. By playing around with the simulation buttons provided on the Mission Execution Dashboard, we can explore the various zero trust capabilities and countermeasures taken by Arculus for secure and successful completion of missions under adverse conditions. These include DDIL scenarios like communication loss, GPS Spoofing attacks, physical hijacking, Denial of Service, Low battery, Brute force attacks, etc.