Advanced Persistent Threat Lab Framework for Testing and Evaluation

Department of Engineering, Computing, and Cybersecurity

Anthony Jamieson

Bryanna Parkoff

Professor: Dr. Gonzalo D. Parra

Outline

Why this project?

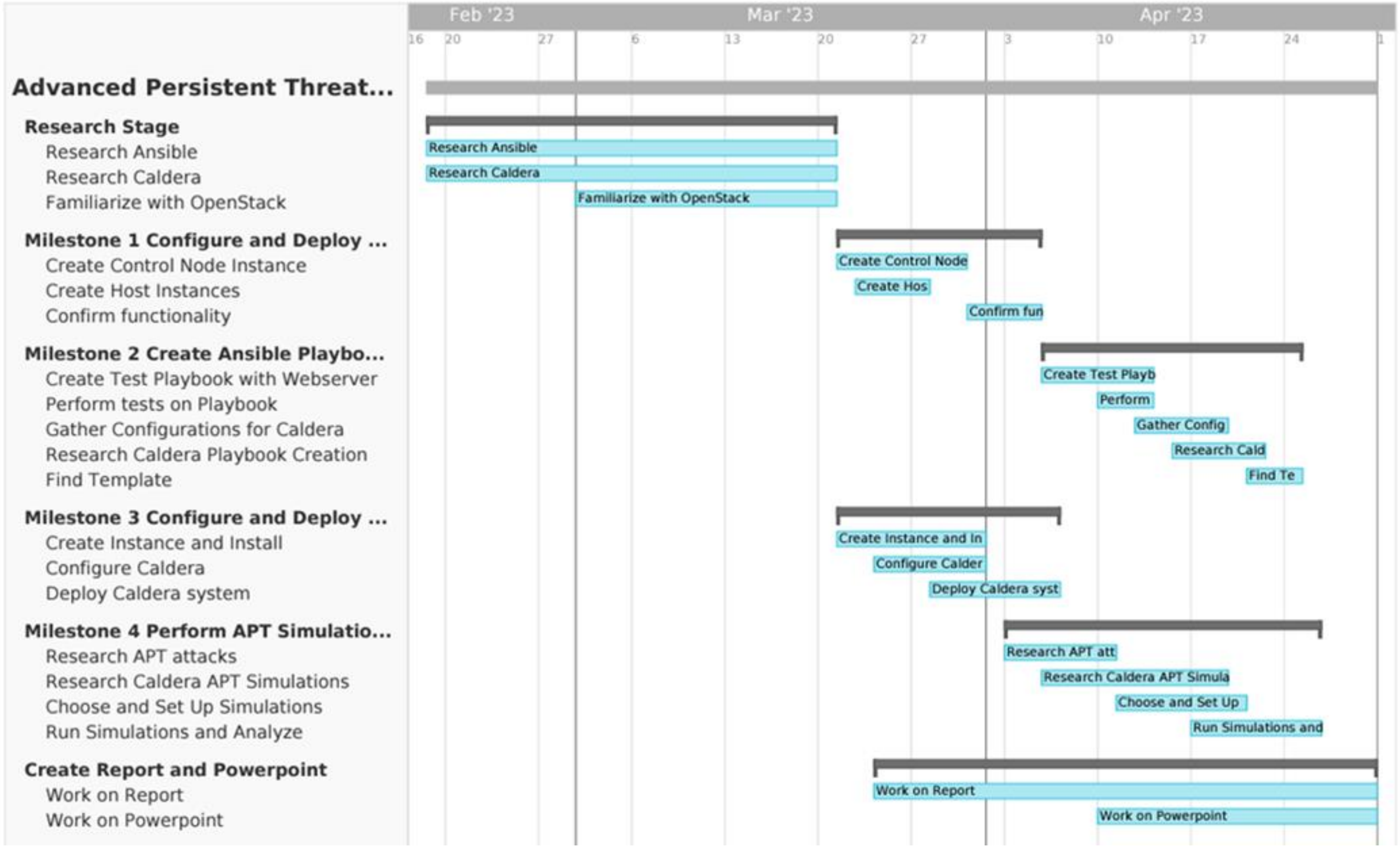
- Advanced Persistent Threats are still difficult to track and not well researched
  - 71% of attacks are malware free according to CrowdStrike's 2023 Global Threat Report
  - Tech industry remains top targeted industry with 21.6% of intrusions
  - Over 150 APT groups are currently known and active
- Automating the creation of cloud-based attack simulation frameworks is not commonly publicly available

Objective

Objective

- **Primary Goals:**
  - Implement and configure a cloud-based attack simulation framework
  - Evaluate and test Advanced Persistent Threat attack simulations
- **Lab framework will utilize:**
  - OpenStack cloud environment
  - Ubuntu 22.04 as standard OS across instances
  - CALDERA by MITRE for the use of simulating attacks
  - Ansible to create and run custom playbooks for critical instances

Timeline



Advanced Persistent Threat...

Research Stage
- Research Ansible
- Research Caldera
- Familiarize with OpenStack

Milestone 1 Configure and Deploy ...
- Create Control Node Instance
- Create Host Instances
- Confirm functionality

Milestone 2 Create Ansible Playbo...
- Create Test Playbook with Webserver
- Perform tests on Playbook
- Gather Configurations for Caldera
- Research Caldera Playbook Creation
- Find Template

Milestone 3 Configure and Deploy ...
- Create Instance and Install
- Configure Caldera
- Deploy Caldera system

Milestone 4 Perform APT Simulatio...
- Research APT attacks
- Research Caldera APT Simulations
- Choose and Set Up Simulations
- Run Simulations and Analyze

Create Report and Powerpoint
- Work on Report
- Work on Powerpoint

Management System

Trello and GitHub Repository

Trello Board

GitHub Repo

https://github.com/awjamieson83/AJ-BP-UIWCSEC-Practicum

Project Milestones

- Milestone 1: Research and Background on Cloud-based Attack Simulation Environments
- Milestone 2: Deployment and Configuration of Cloud-based Attack Simulation Environments
- Milestone 3: Development of Ansible Playbooks for Key Instances
- Milestone 4: Adversary Emulation using Caldera
- Milestone 5: Configure and Deploy Caldera
- Milestone 6: Perform APT Attack Simulations

Milestone 1

Research and Background on Cloud-based Attack Simulation Environments

Background

- Public examples and information on cloud-based attack simulation environments are rarely seen
- Existing full systems are mostly shrouded in secrecy or not available to the public
  - Systems belonging to major corporations or government militaries
- While multiple attack simulation tools exist, very few are ready for cloud-based deployment
  - Locked down to specific cloud providers or limited in options

Researching Tools and Systems

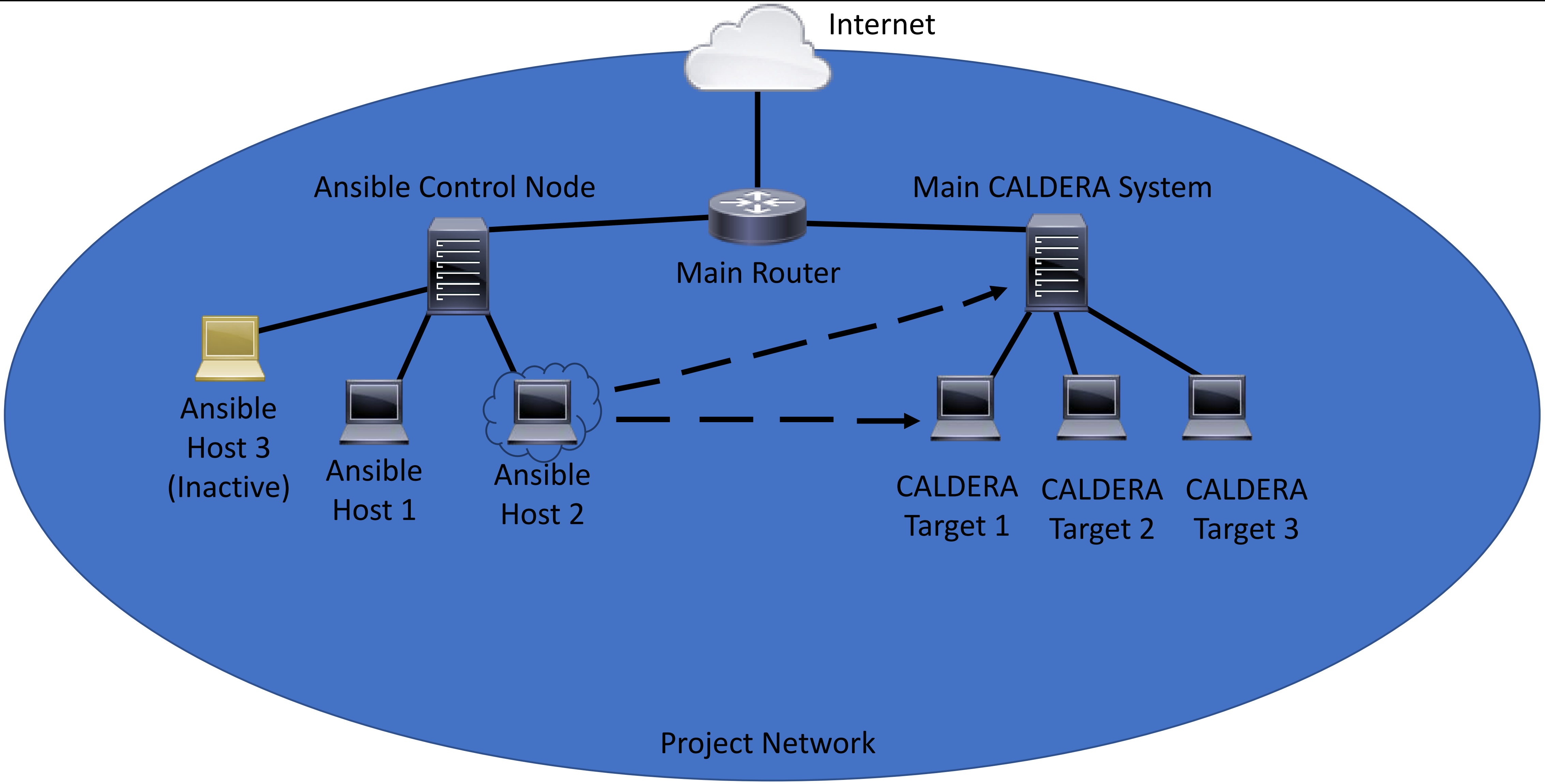
- Our team analyzed offerings from different providers
- Many attack simulation tools are locked behind a paywall
  - The costs for these systems can be easily out of budget for researchers or smaller companies
  - Options considered: AttackIQ, Picus Security, XM Cyber
- Caldera seemed to be the most favorable option for a free, open-source attack simulation tool
  - Created by a reputable corporation (MITRE Corporation)
  - Allows use without licenses
  - Open-source code allows for modification and customization with plugins

Researching Tools and Systems (cont.)

- Many automation tools also had the same issues as the attack simulation tools
- Puppet Enterprise and Chef Automate are major players in the automation industry
  - These programs, while having many additional features, are locked behind a paywall
  - Licensing could be an issue for public distribution and use
- Ansible remains as a growing open-source tool with plentiful community support, documentation and free plugins
  - Though there is an option to purchase through Red Hat, it is mainly for official support and maintenance
  - General project is available through GitHub with active dev community

Milestone 2

Deployment and Configuration of Cloud-based Attack Simulation Environments



Icons courtesy of CISCO. https://www.cisco.com/c/en/us/about/brand-center/network-topology-icons.html

Utilizing Ansible

- Ansible will be utilized for instance deployment automation
- Ansible works by using pre-configured scripts called playbooks
  - The playbooks are run on connected host systems
  - Playbooks run step by step, implementing features and configurations as specified
- Automation reduces chance of user error when deploying new instances
  - Keeps settings consistent for research; limits chances of changing variables in a study environment
- Ansible can help replace broken, compromised, or unhealthy instances needing replacements

Deploying the Main Ansible Instance

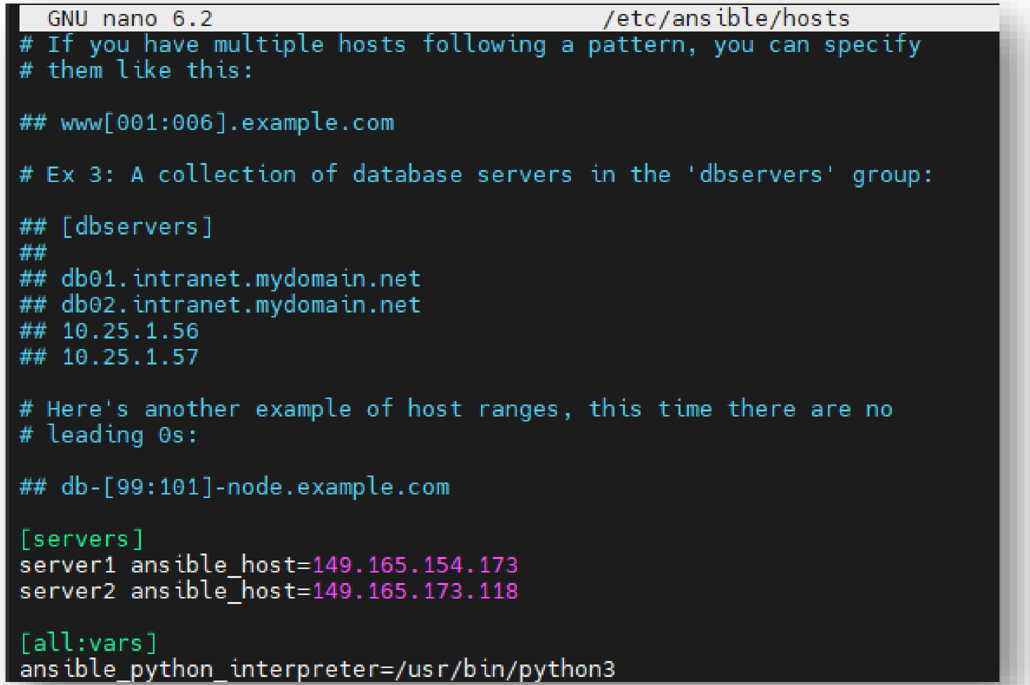
- Ansible works on a master server system
- One control node and multiple hosts connected to that node
- Control node was built on Ubuntu 22.04 instance
- Instance was added to a hardened security group
    - Only allows ingress access to our team's computers
    - Instance security was managed using a pair of private keys
    - Assigned a floating IP for outside SSH access from allowed IP's

Configuring the Control Node

- Accessed node for set up through SSH
- Pulled official Ansible repository package
  - Repository package pulls and installs necessary background utilities, such as Python 3 and related plugins
- Installed main Ansible software from official package
- Inventory file needs to be properly configured before use
  - Our inventory requires hosts to be implemented
- Created two new small instances with assigned IP addresses
  - These instances act as separate hosts that run the control node’s commands

Configuring the Control Node (cont.)

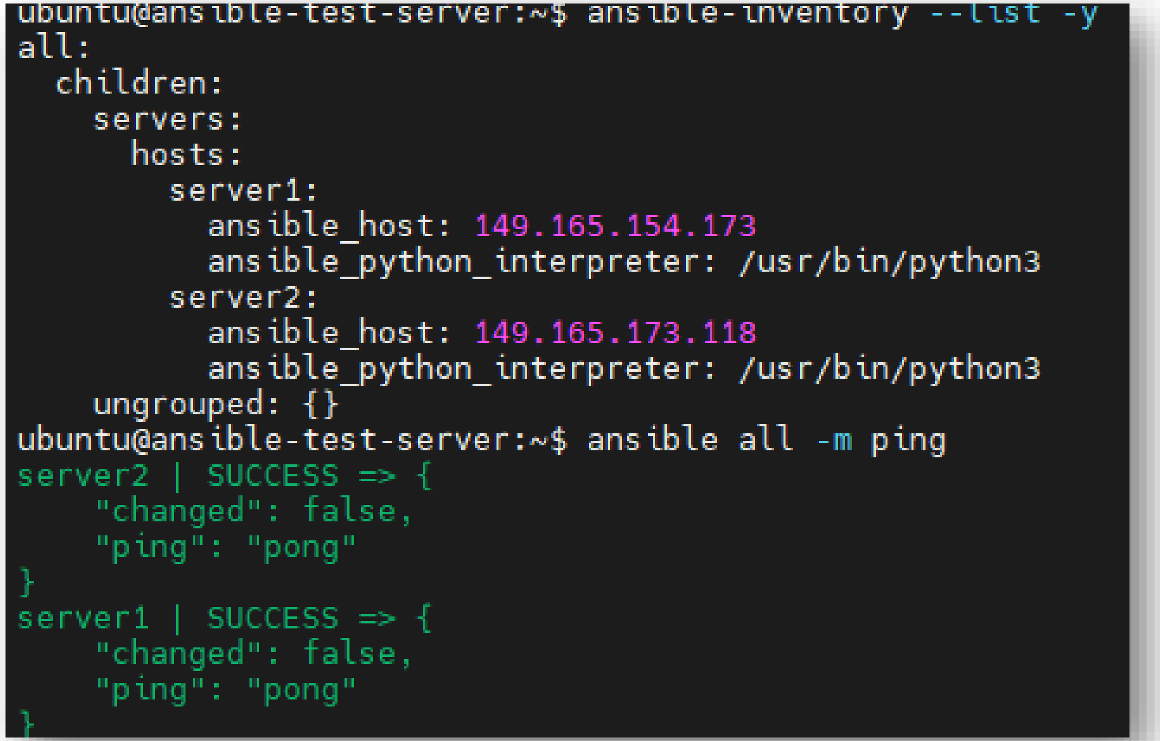
- Opened the ‘hosts’ file in Ansible directory to configure inventory utilizing nano
- Created a new category named ‘servers’
  - Input the two new host instances, named server1 and server2 respectively
  - Created category to direct Ansible to use Python 3 as its default interpreter



```
  GNU nano 6.2                                    /etc/ansible/hosts
# If you have multiple hosts following a pattern, you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group:

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com

[servers]
server1 ansible_host=149.165.154.173
server2 ansible_host=149.165.173.118

[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

Testing and Finding Issues

- Once the inventory file was finished, connection testing was the next step
- Checked Ansible inventory using the ‘ansible-inventory’ command to ensure hosts file was properly read
  - Hosts showed as being detected, allowing us to run ping tests to our two systems
- With our connections in place and showing up inside Ansible, we were ready to move forward with playbook testing



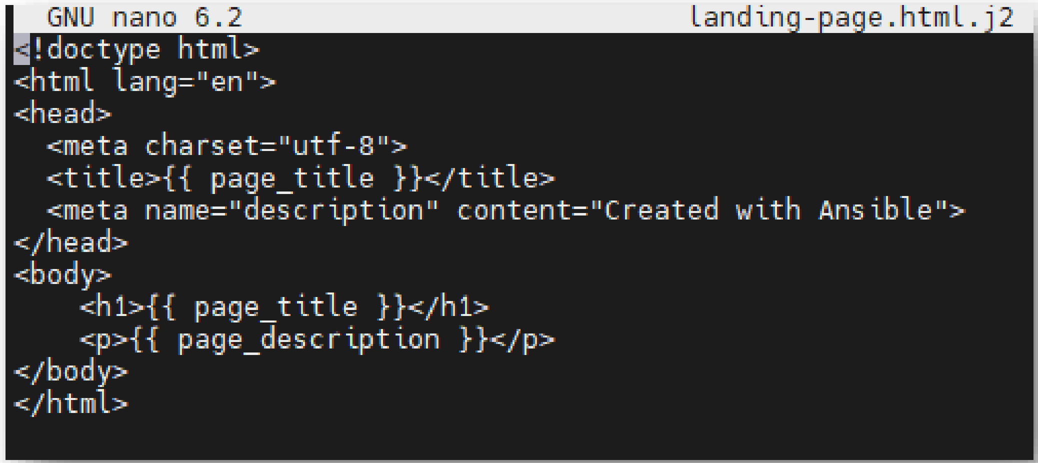
```
ubuntu@ansible-test-server:~$ ansible-inventory --list -y
all:
  children:
    servers:
      hosts:
        server1:
          ansible_host: 149.165.154.173
          ansible_python_interpreter: /usr/bin/python3
        server2:
          ansible_host: 149.165.173.118
          ansible_python_interpreter: /usr/bin/python3
    ungrouped: {}
ubuntu@ansible-test-server:~$ ansible all -m ping
server2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
server1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Milestone 3

Development of Ansible Playbooks for Key Instances

Creating Test Playbook

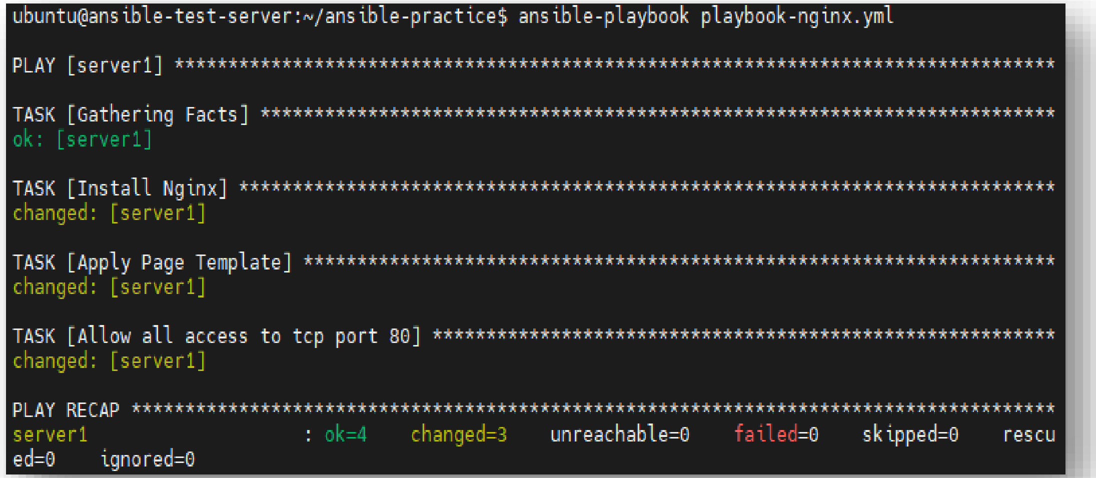
- Now that Ansible had properly installed and connected to hosts, we moved on to testing full functionality
- We decided on creating a simple test playbook that initializes a webserver based on Nginx
- Playbook uses HTML imbedded in a Jinja2 template, allowing playbook to call the file and edit variables inside the playbook itself



```
  GNU nano 6.2                                      landing-page.html.j2
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>{{ page_title }}</title>
  <meta name="description" content="Created with Ansible">
</head>
<body>
    <h1>{{ page_title }}</h1>
    <p>{{ page_description }}</p>
</body>
</html>
```

Creating Test Playbook (cont.)

- Playbook runs with 4 separate tasks in order:
  1. Playbook resolves host connection
  2. Nginx is installed and updated on host
  3. Default configuration is replaced with Jinja page template
  4. Access is allowed on port 80 to the server
- Playbook is run with ‘ansible-playbook’ command, automatically executing the .yml file



```
ubuntu@ansible-test-server:~/ansible-practice$ ansible-playbook playbook-nginx.yml

PLAY [server1] ********************************************************************************

TASK [Gathering Facts] ************************************************************************
ok: [server1]

TASK [Install Nginx] **************************************************************************
changed: [server1]

TASK [Apply Page Template] ********************************************************************
changed: [server1]

TASK [Allow all access to tcp port 80] ********************************************************
changed: [server1]

PLAY RECAP ************************************************************************************
server1                    : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescu
ed=0    ignored=0
```

Configuring Caldera Playbook

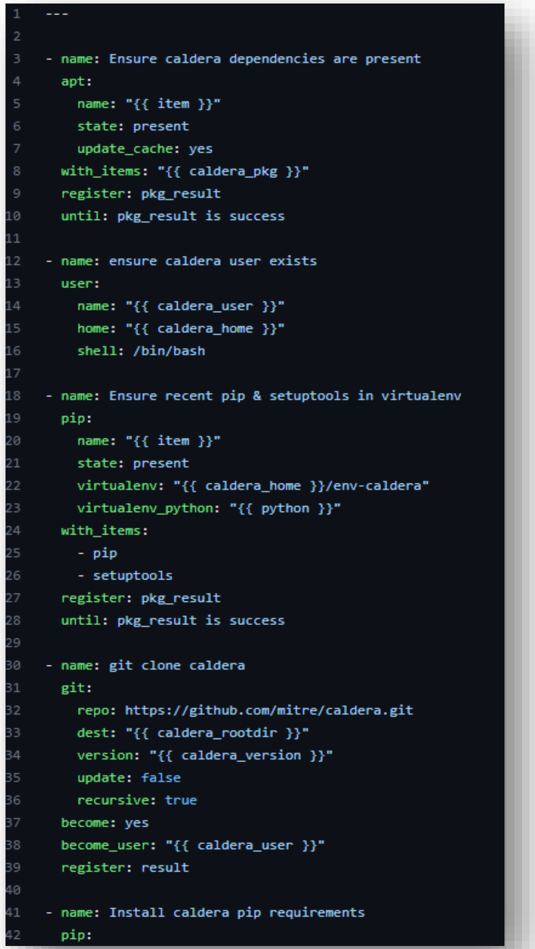
- This section was significantly more challenging
- Caldera is made up of many moving parts with a more rigorous setup process
- Playbook would need to be more complex
  - More possible points of failure
  - More time dedicated to writing and troubleshooting
- Initial attempts at building default playbooks for Caldera had many issues or did not run as intended
- We decided to research this section a bit more in-depth

Configuring Caldera Playbook (cont.)

- There was very little information online about using Caldera with Ansible for automatic deployment
- Much of the data was outdated, using older versions of Ubuntu or Ansible
- Lots of incomplete information and data, just bits and pieces of code and discussions
- We discovered one promising Caldera setup template uploaded to GitHub by NVISOsecurity

Assembling Temporary Caldera Playbook

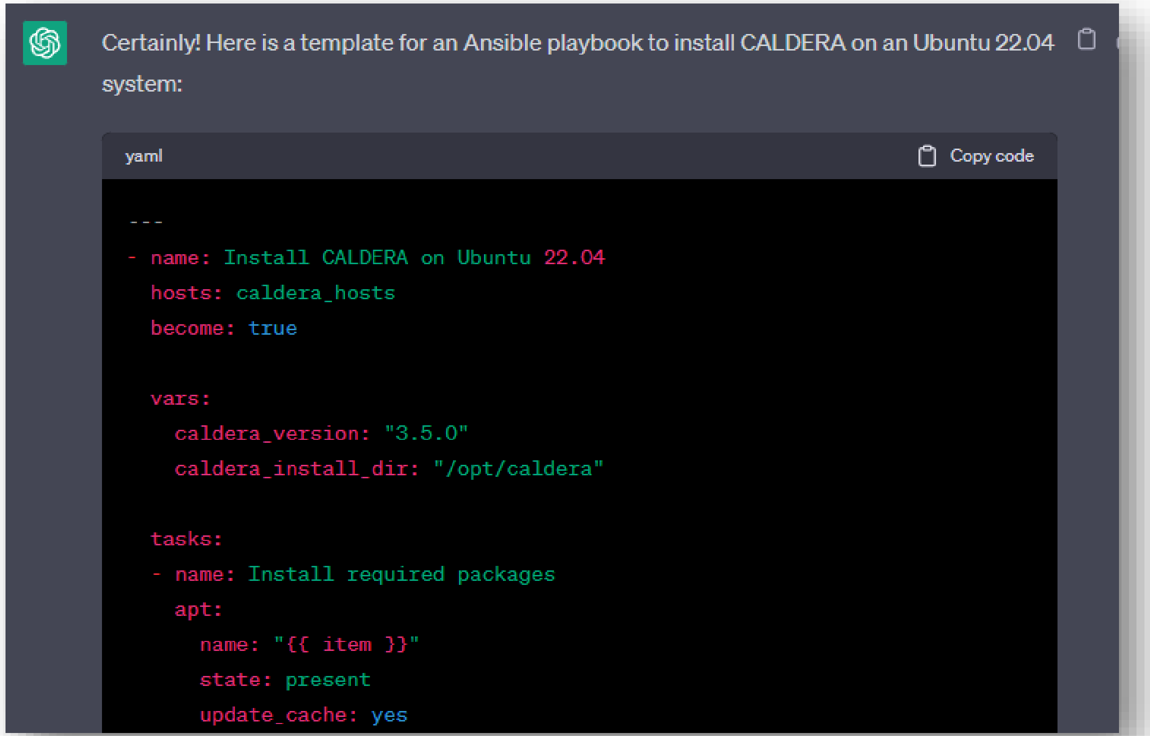
- This temporary playbook was made up of many smaller playbooks that would call in sequence
- As to not disrupt the sequence, the playbook was configured to be shared with a single role
- This would call the entire playbook with one smaller task
  - Reduced footprint
  - Less customizable



```yaml
---

- name: Ensure caldera dependencies are present
  apt:
    name: "{{ item }}"
    state: present
    update_cache: yes
  with_items: "{{ caldera_pkg }}"
  register: pkg_result
  until: pkg_result is success

- name: ensure caldera user exists
  user:
    name: "{{ caldera_user }}"
    home: "{{ caldera_home }}"
    shell: /bin/bash

- name: Ensure recent pip & setuptools in virtualenv
  pip:
    name: "{{ item }}"
    state: present
    virtualenv: "{{ caldera_home }}/env-caldera"
    virtualenv_python: "{{ python }}"
  with_items:
    - pip
    - setuptools
  register: pkg_result
  until: pkg_result is success

- name: git clone caldera
  git:
    repo: https://github.com/mitre/caldera.git
    dest: "{{ caldera_rootdir }}"
    version: "{{ caldera_version }}"
    update: false
    recursive: true
  become: yes
  become_user: "{{ caldera_user }}"
  register: result

- name: Install caldera pip requirements
  pip:
```

AI and Playbook Research

- We continued with our research of Caldera playbooks and found little success
  - We decided to utilize the assistance of ChatGPT as another source of help
- With some prompt engineering, ChatGPT assisted in producing well written and straightforward playbook configurations to manipulate moving forward

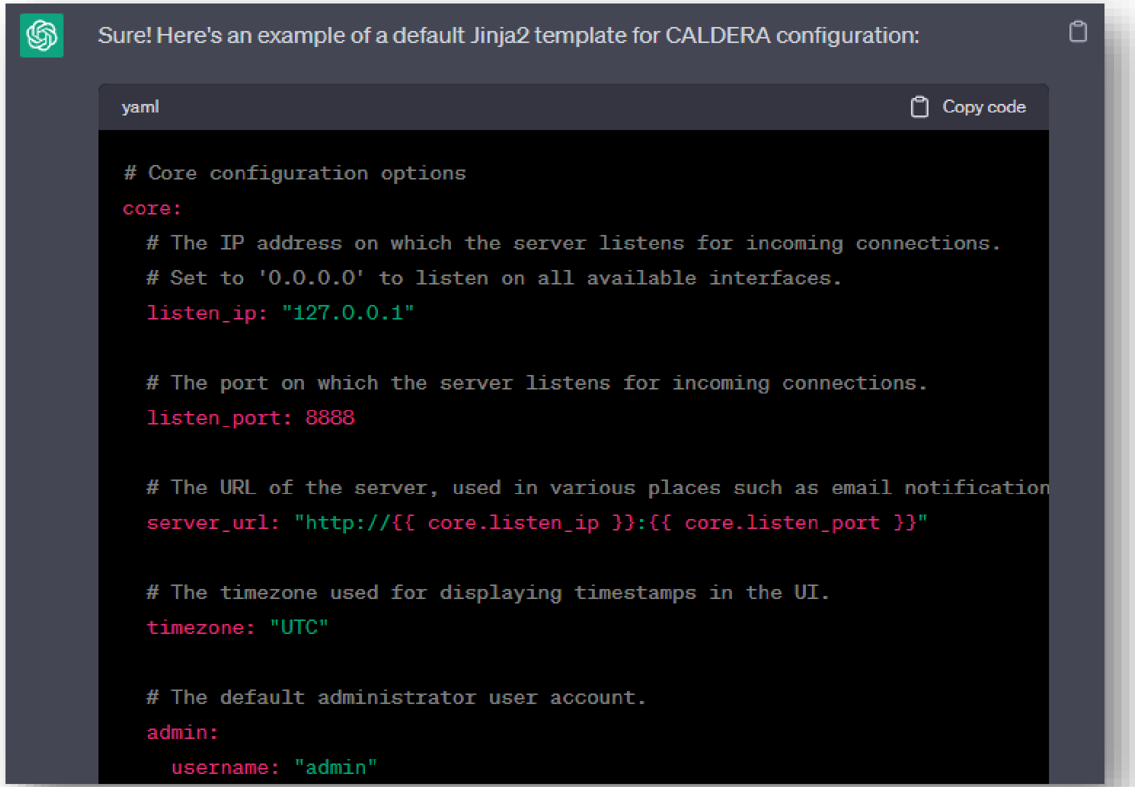


Certainly! Here is a template for an Ansible playbook to install CALDERA on an Ubuntu 22.04 system:

```yaml
---
- name: Install CALDERA on Ubuntu 22.04
  hosts: caldera_hosts
  become: true

  vars:
    caldera_version: "3.5.0"
    caldera_install_dir: "/opt/caldera"

  tasks:
  - name: Install required packages
    apt:
      name: "{{ item }}"
      state: present
      update_cache: yes
```

AI and Playbook Research (cont.)

- ChatGPT was able to deliver a playbook utilizing a Jinja2 template for executing our modified configuration instructions
- We implemented our custom changes for default credentials, IP addresses and other settings in the J2 template
- Allows us more customization of our playbook distribution
  - Easy modification for future changes
  - Keeps playbook neat and easier to read



Sure! Here's an example of a default Jinja2 template for CALDERA configuration:

```yaml
# Core configuration options
core:
  # The IP address on which the server listens for incoming connections.
  # Set to '0.0.0.0' to listen on all available interfaces.
  listen_ip: "127.0.0.1"

  # The port on which the server listens for incoming connections.
  listen_port: 8888

  # The URL of the server, used in various places such as email notification
  server_url: "http://{{ core.listen_ip }}:{{ core.listen_port }}"

  # The timezone used for displaying timestamps in the UI.
  timezone: "UTC"

  # The default administrator user account.
  admin:
    username: "admin"
```

Milestone 4

Automated Adversary Emulation using Caldera

Defining Adversary Emulation

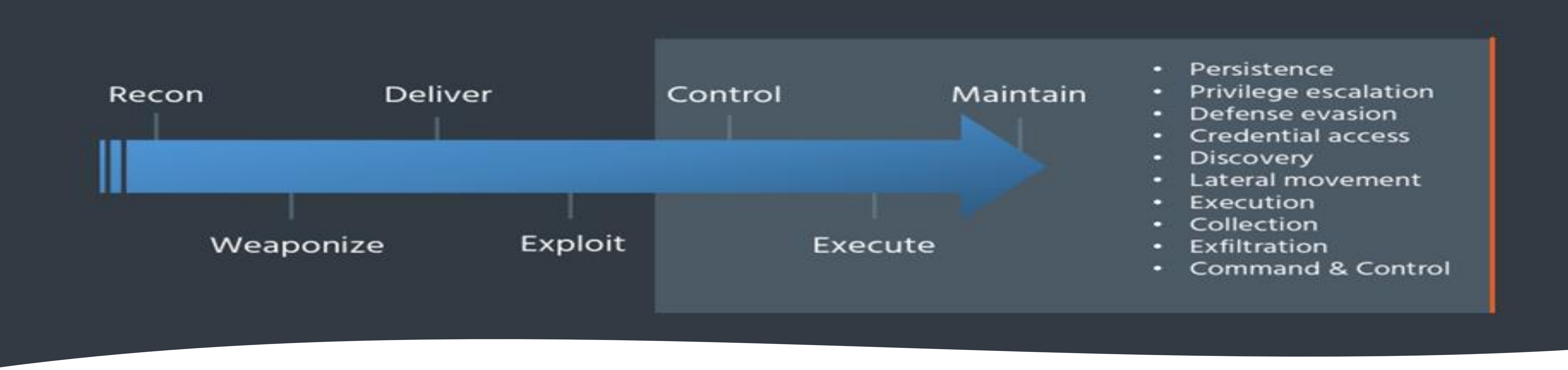
Adversary Emulation: an activity where security experts emulate how an adversary operates. The ultimate goal is to improve how resilient the organization is versus the adversary technique.

TTP

Adversary Activities are described using TTP’s(Tactics, Techniques & Procedures). They describe how the adversary operates at a higher level and should not be mixed with traditional vulnerability scans or penetration testing which isn’t based on TTP’s.

ATT&CK

Adversary Emulation should be performed using a structured approach such as a kill chain or attack flow like MITRE ATT&CK.



Utilizing Caldera

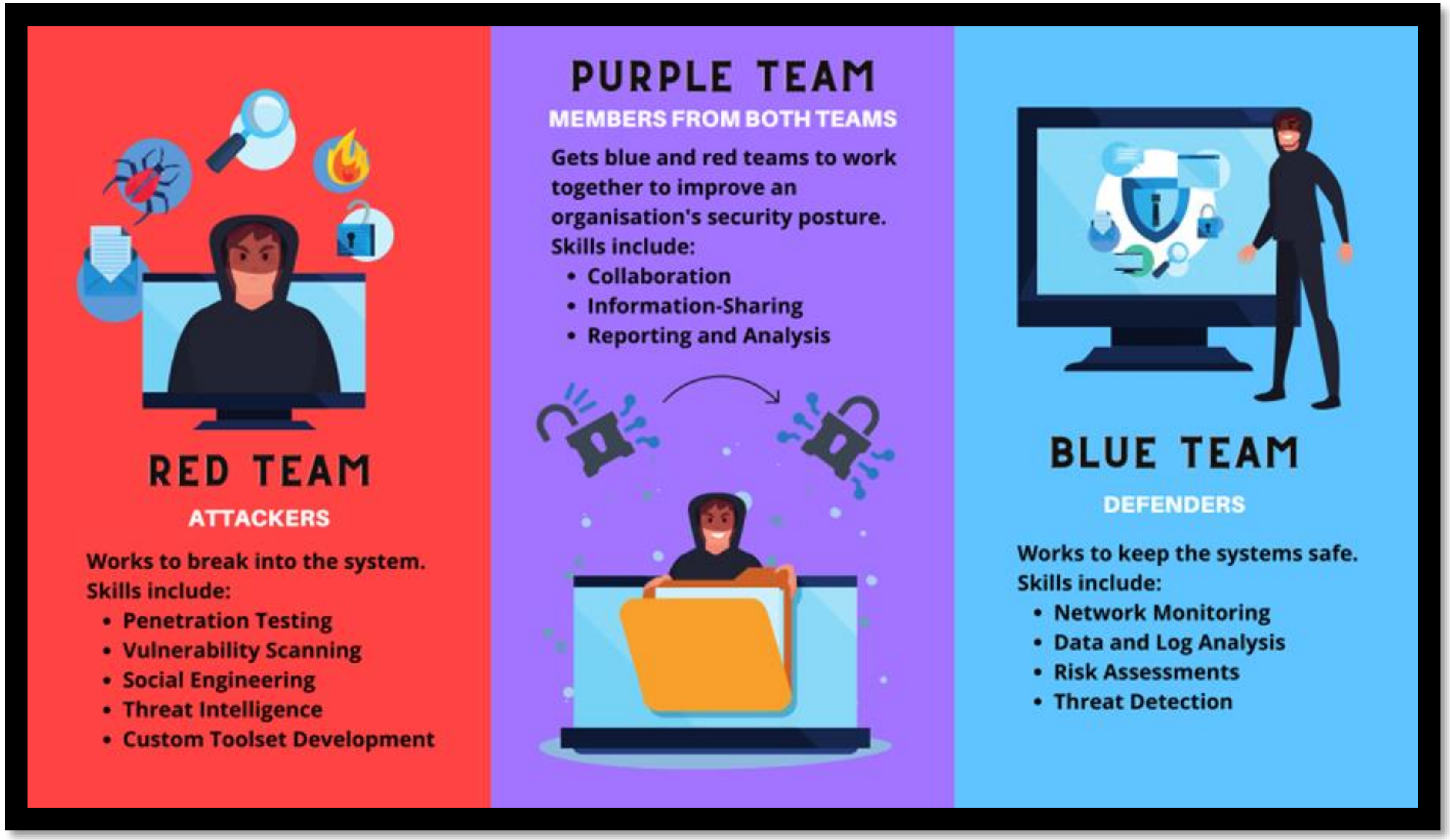
- **Caldera:** a cybersecurity framework tool developed by MITRE with the express purpose of doing adversary emulation. It will actively “attack” target systems by deploying agents and custom backdoors.
- Attack steps work along with MITRE ATT&CK Matrix and Kill-Chain to conduct defensive and offensive security testing.
- This tool will be utilized for setting up agents and initiating APT(Advanced Persistent Threat) operation in a Cloud environment
- These operations will be carried out by two teams: Blue & Red

Caldera Teams

I. **Red:** focuses on offensive security to identify and exploit potential weaknesses within the organization's cyber defenses using sophisticated attack techniques.

II. **Blue:** focuses on defensive security to identify and neutralizes risks and threats before they inflict damage on the organization.

III. **Purple:** red and blue working together as they share information to improve business security.



RED TEAM

ATTACKERS

Works to break into the system.
Skills include:

- Penetration Testing
- Vulnerability Scanning
- Social Engineering
- Threat Intelligence
- Custom Toolset Development

PURPLE TEAM

MEMBERS FROM BOTH TEAMS

Gets blue and red teams to work together to improve an organisation's security posture.
Skills include:

- Collaboration
- Information-Sharing
- Reporting and Analysis

BLUE TEAM

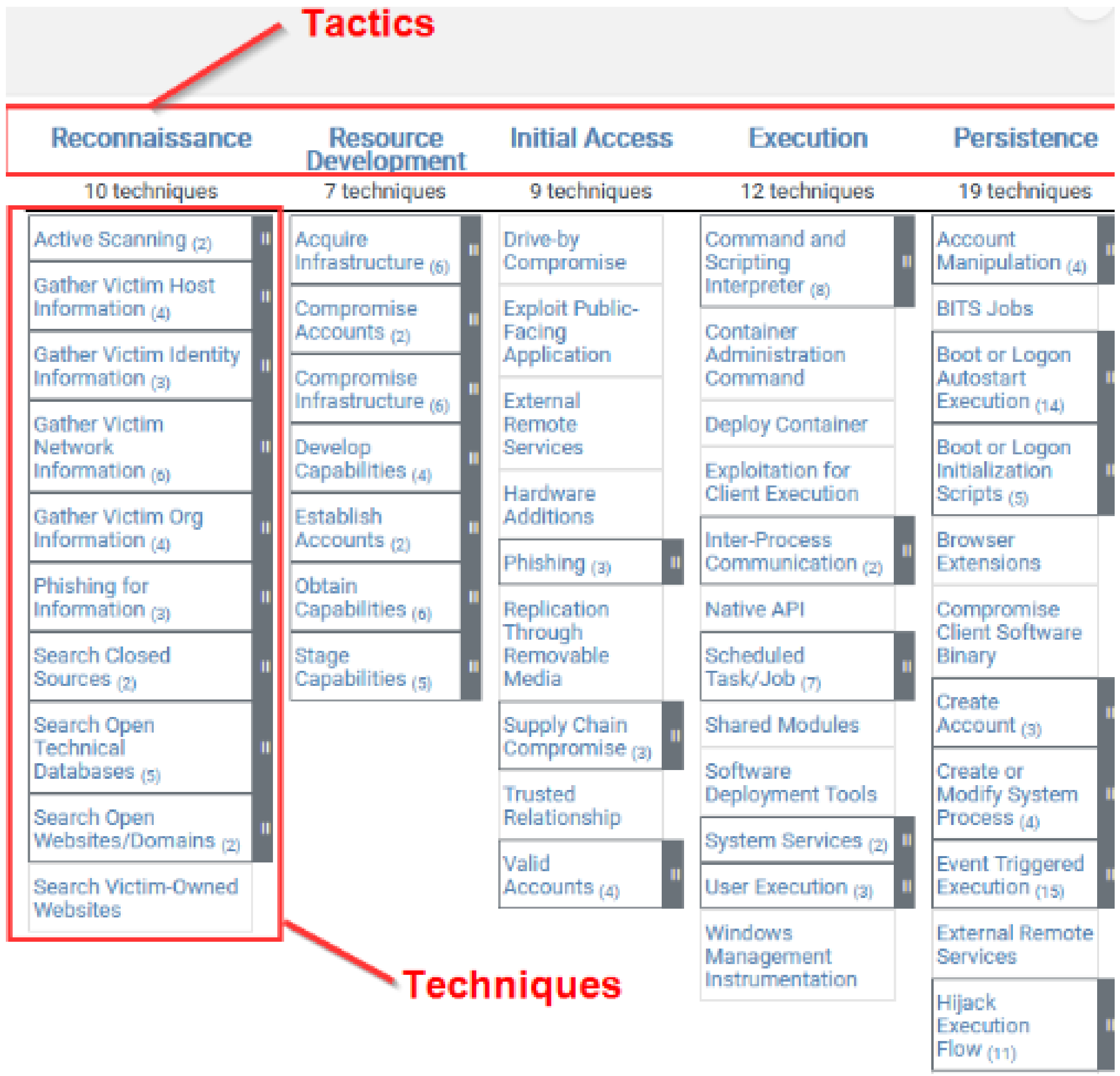
DEFENDERS

Works to keep the systems safe.
Skills include:

- Network Monitoring
- Data and Log Analysis
- Risk Assessments
- Threat Detection

Operationalizing MITRE ATT&CK

- MITRE ATT&CK framework provides us with a structured kill chain that we can utilize for our adversary emulation.
- **Tactics:** specific technical objectives that an adversary intends to achieve, such as lateral movement, defense evasion, or exfiltration.
- **Techniques:** method used by the threat actor to engage in the attack such as skimming, javascript injection attacks or cross-site scripting(XSS).
- **Procedures:** step-by-step orchestration of an attack



Reconnaissance	Resource Development	Initial Access	Execution	Persistence
10 techniques	7 techniques	9 techniques	12 techniques	19 techniques
Active Scanning (2)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (4)
Gather Victim Host Information (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs
Gather Victim Identity Information (3)	Compromise Infrastructure (6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (14)
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)
Gather Victim Org Information (4)	Establish Accounts (2)	Phishing (3)	Inter-Process Communication (2)	Browser Extensions
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary
Search Closed Sources (2)	Stage Capabilities (5)	Supply Chain Compromise (3)	Scheduled Task/Job (7)	Create Account (3)
Search Open Technical Databases (5)		Trusted Relationship	Shared Modules	Create or Modify System Process (4)
Search Open Websites/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)
Search Victim-Owned Websites			System Services (2)	External Remote Services
			User Execution (3)	Hijack Execution Flow (11)
			Windows Management Instrumentation	

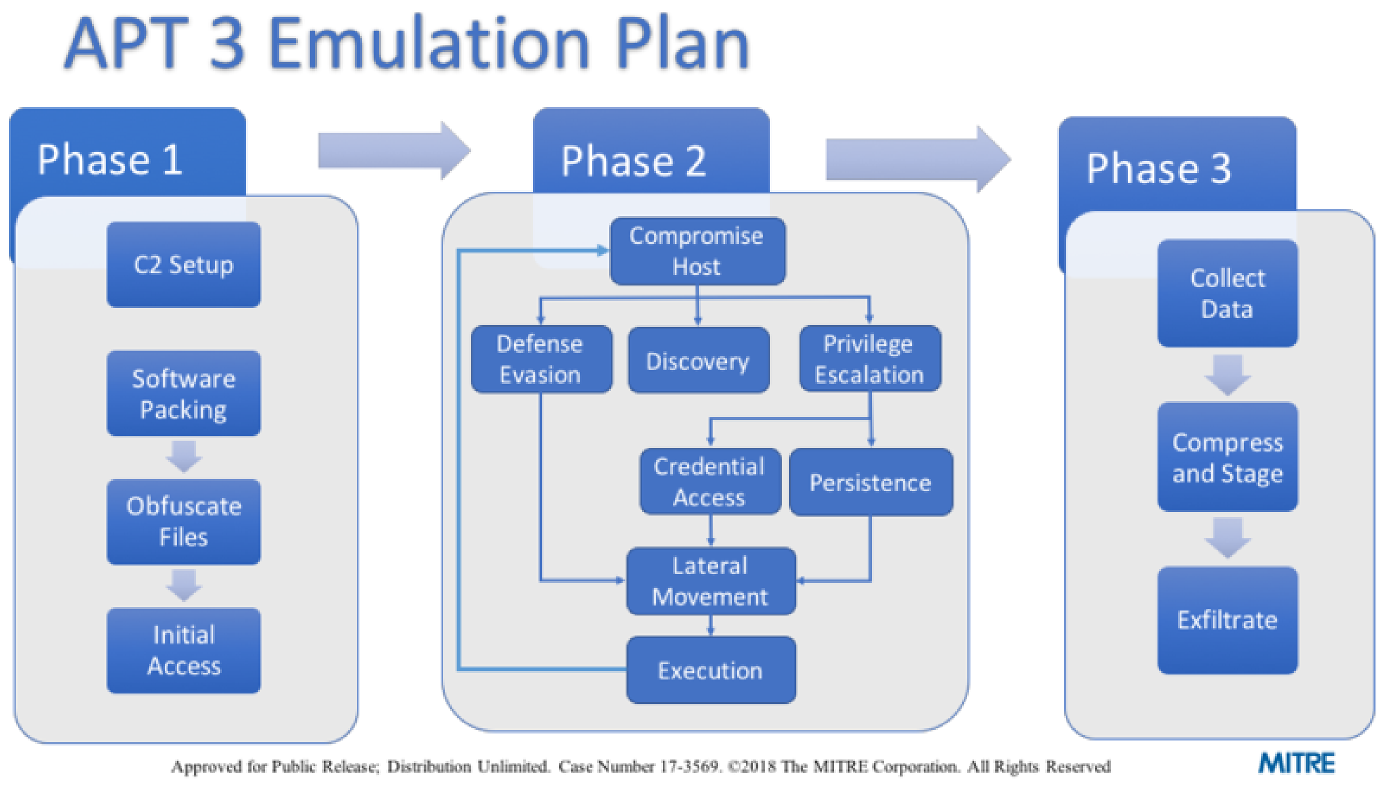
Adversary Team Emulation Plans

- Building a good Adversary Emulation Plan is crucial to success for simulating any APT attacks.
- Plan should mimic an actual adversary and include distinct phases.
- In MITRE APT 3 Emulation Plan the following phases are included:

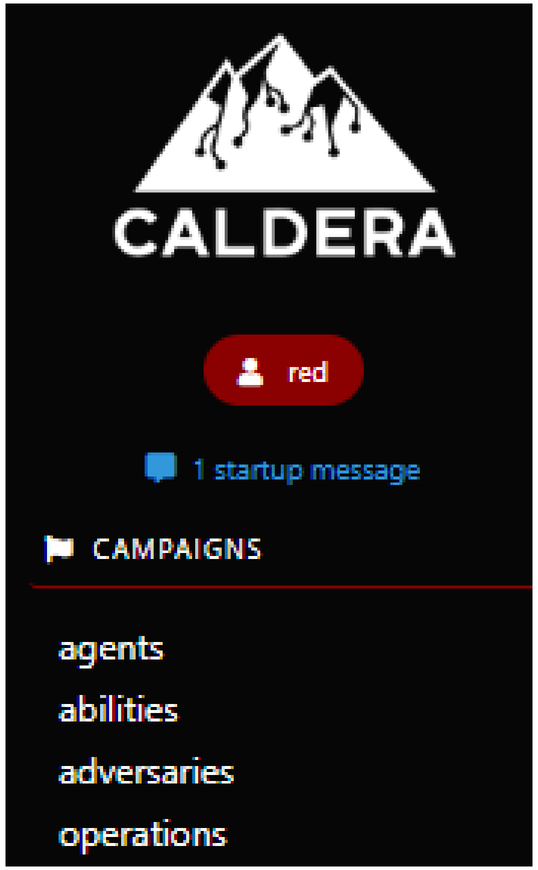
i. Set up Adversary infrastructure & obtain Initial Execution(Initial Access)

ii. Internal Discovery, priviledge escalation, lateral movement(Lateral Movement)

iii. Collection, staging, exfiltration(Action on Objectives)



Caldera Campaigns



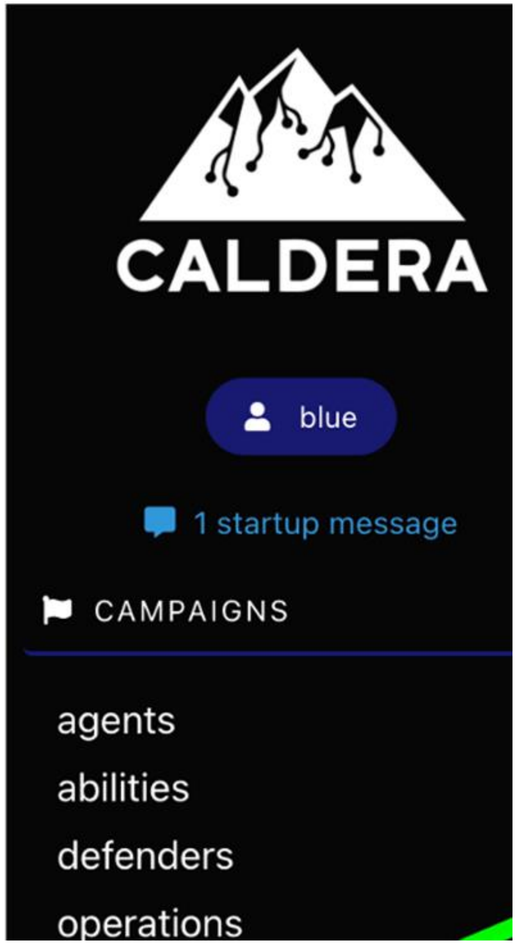
In order to fully realize the potential of the framework, you will need to familiarise yourself with the following terminology:

Agents– Agents are software programs installed on target hosts/clients that connect back to CALDERA at certain intervals to get instructions. Agents communicate with the CALDERA server via a contact method, initially defined at agent.

Groups- collections of agents so hosts can be compromised simultaneously.

Abilities– An ability is a specific ATT&CK tactic/technique implementation that can be executed on running agents. Abilities will include the command(s) to run, the platforms/executors the commands can run on (ex: Windows / PowerShell), payloads to include, and a reference to a module to parse the output on the CALDERA server.

Caldera Campaigns



In order to fully realize the potential of the framework, you will need to familiarize yourself with the following terminology:

Adversaries-collections of ATT&CK TTPs, designed to create specific effects on a host or network. Profiles can be used for offensive or defensive use cases.

Operations-combine agents, abilities, and adversaries to execute attacks against specific targets. This is what is executed against specific targets within the CALDERA platform.

Note: Each one of these terminologies which play a significant role in orchestrating an APT simulation is referred to as a "Campaign" in Caldera.

Milestone 5

Configuring and Deploying Caldera

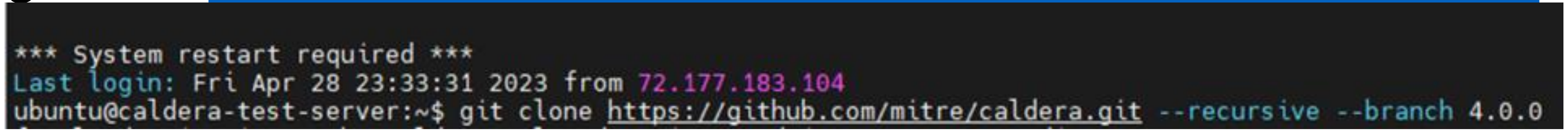
Deploying the Main Caldera Instance

- To run APT simulations in Caldera it is important to create a main-server and target instance
- Main-Server Instance: manages Caldera configuration files
- Manages Security Group rules needed for certain agent deployments and connecting to Caldera’s internal IP Address
- Run Caldera
- Target Instance: where we will be generating the agent attack scripts and overall APT Simulations

Installing Caldera

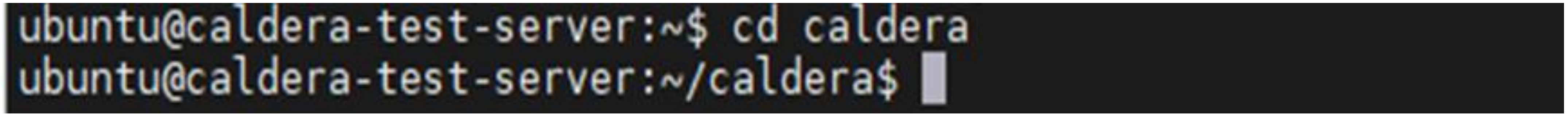
- After, the instances are up and running we installed Caldera following these four commands:

i. git clone https://github.com/mitre/caldera.git --recursive --branch x.x.x



```
*** System restart required ***
Last login: Fri Apr 28 23:33:31 2023 from 72.177.183.104
ubuntu@caldera-test-server:~$ git clone https://github.com/mitre/caldera.git --recursive --branch 4.0.0
```

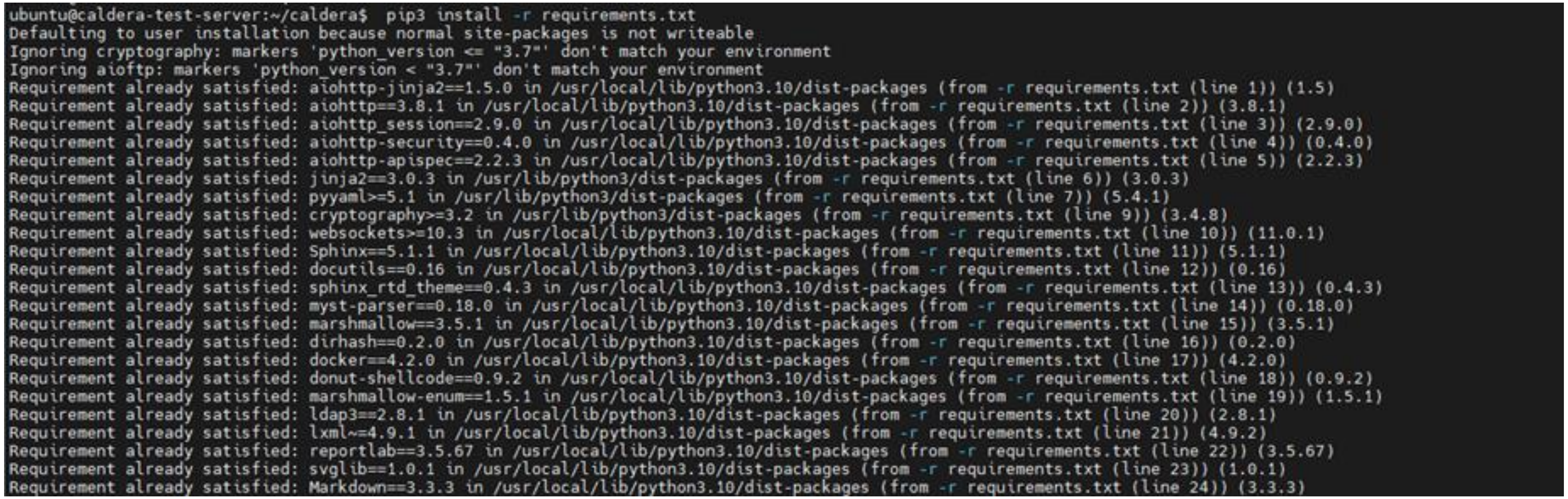
ii. cd caldera



```
ubuntu@caldera-test-server:~$ cd caldera
ubuntu@caldera-test-server:~/caldera$
```

Installing Caldera (Cont.)

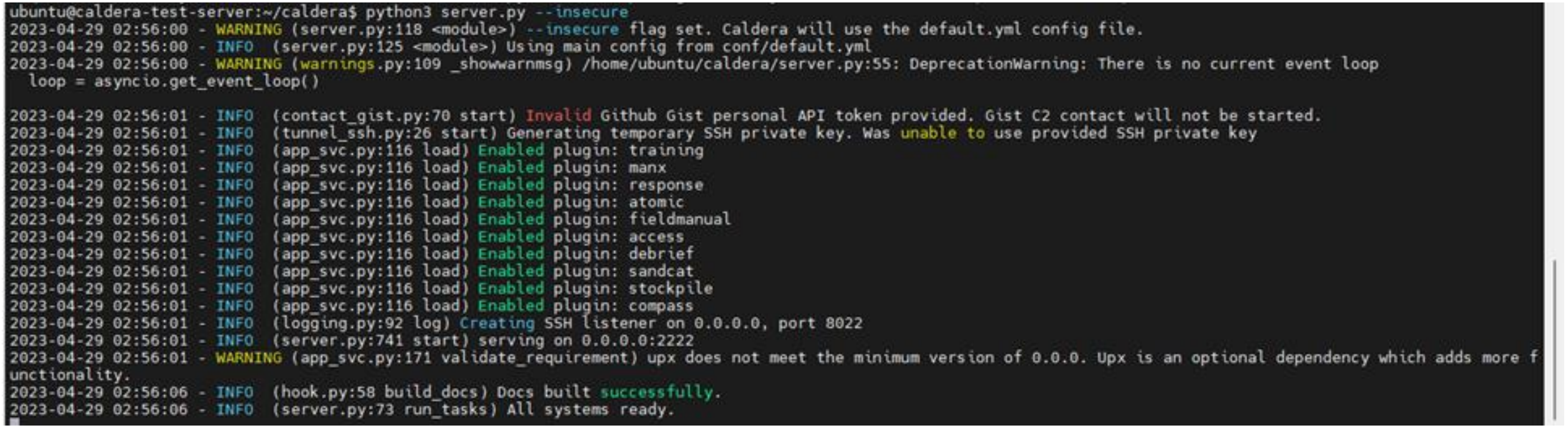
iii. pip3 install -r requirements.txt



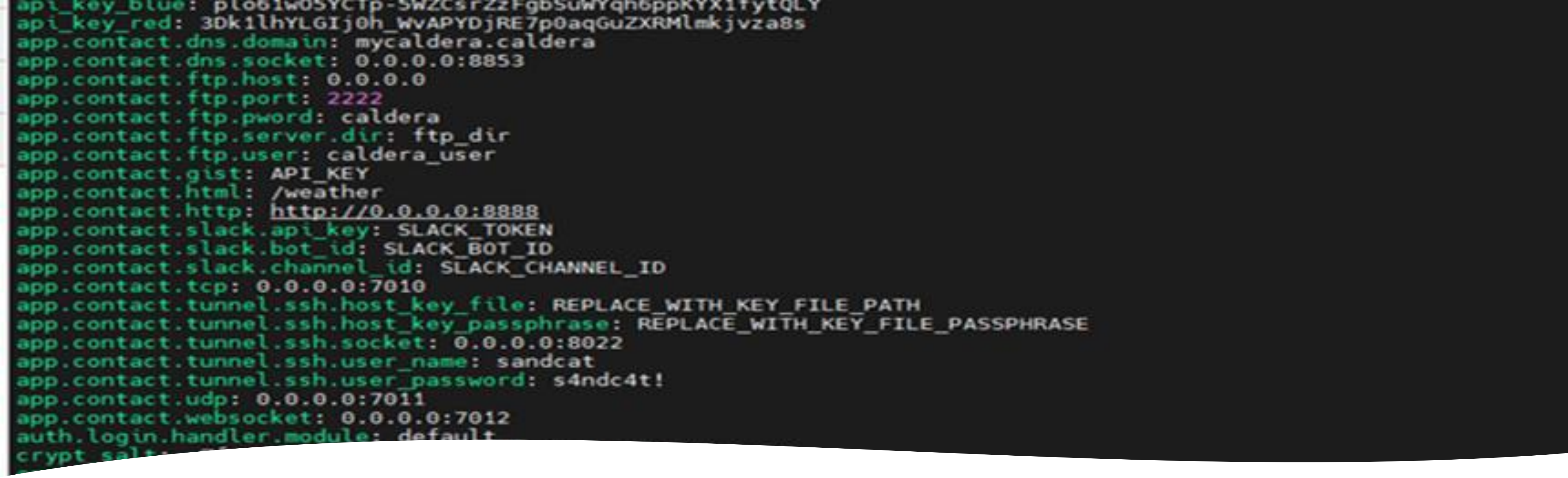
```
ubuntu@caldera-test-server:~/caldera$  pip3 install -r requirements.txt
Defaulting to user installation because normal site-packages is not writeable
Ignoring cryptography: markers 'python_version <= "3.7"' don't match your environment
Ignoring aioftp: markers 'python_version < "3.7"' don't match your environment
Requirement already satisfied: aiohttp-jinja2==1.5.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 1)) (1.5)
Requirement already satisfied: aiohttp==3.8.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 2)) (3.8.1)
Requirement already satisfied: aiohttp_session==2.9.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 3)) (2.9.0)
Requirement already satisfied: aiohttp-security==0.4.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 4)) (0.4.0)
Requirement already satisfied: aiohttp-apispec==2.2.3 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 5)) (2.2.3)
Requirement already satisfied: jinja2==3.0.3 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 6)) (3.0.3)
Requirement already satisfied: pyyaml>=5.1 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 7)) (5.4.1)
Requirement already satisfied: cryptography>=3.2 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 9)) (3.4.8)
Requirement already satisfied: websockets>=10.3 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 10)) (11.0.1)
Requirement already satisfied: Sphinx==5.1.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 11)) (5.1.1)
Requirement already satisfied: docutils==0.16 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 12)) (0.16)
Requirement already satisfied: sphinx_rtd_theme==0.4.3 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 13)) (0.4.3)
Requirement already satisfied: myst-parser==0.18.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 14)) (0.18.0)
Requirement already satisfied: marshmallow==3.5.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 15)) (3.5.1)
Requirement already satisfied: dirhash==0.2.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 16)) (0.2.0)
Requirement already satisfied: docker==4.2.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 17)) (4.2.0)
Requirement already satisfied: donut-shellcode==0.9.2 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 18)) (0.9.2)
Requirement already satisfied: marshmallow-enum==1.5.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 19)) (1.5.1)
Requirement already satisfied: ldap3==2.8.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 20)) (2.8.1)
Requirement already satisfied: lxml~=4.9.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 21)) (4.9.2)
Requirement already satisfied: reportlab==3.5.67 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 22)) (3.5.67)
Requirement already satisfied: svglib==1.0.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 23)) (1.0.1)
Requirement already satisfied: Markdown==3.3.3 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 24)) (3.3.3)
```

Installing Caldera (Cont.)

iv. **python3 server.py –insecure**



```
ubuntu@caldera-test-server:~/caldera$ python3 server.py --insecure
2023-04-29 02:56:00 - WARNING (server.py:118 <module>) --insecure flag set. Caldera will use the default.yml config file.
2023-04-29 02:56:00 - INFO  (server.py:125 <module>) Using main config from conf/default.yml
2023-04-29 02:56:00 - WARNING (warnings.py:109 _showwarnmsg) /home/ubuntu/caldera/server.py:55: DeprecationWarning: There is no current event loop
  loop = asyncio.get_event_loop()

2023-04-29 02:56:01 - INFO  (contact_gist.py:70 start) Invalid Github Gist personal API token provided. Gist C2 contact will not be started.
2023-04-29 02:56:01 - INFO  (tunnel_ssh.py:26 start) Generating temporary SSH private key. Was unable to use provided SSH private key
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: training
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: manx
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: response
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: atomic
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: fieldmanual
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: access
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: debrief
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: sandcat
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: stockpile
2023-04-29 02:56:01 - INFO  (app_svc.py:116 load) Enabled plugin: compass
2023-04-29 02:56:01 - INFO  (logging.py:92 log) Creating SSH listener on 0.0.0.0, port 8022
2023-04-29 02:56:01 - INFO  (server.py:741 start) serving on 0.0.0.0:2222
2023-04-29 02:56:01 - WARNING (app_svc.py:171 validate_requirement) upx does not meet the minimum version of 0.0.0. Upx is an optional dependency which adds more f
unctionality.
2023-04-29 02:56:06 - INFO  (hook.py:58 build_docs) Docs built successfully.
2023-04-29 02:56:06 - INFO  (server.py:73 run_tasks) All systems ready.
```



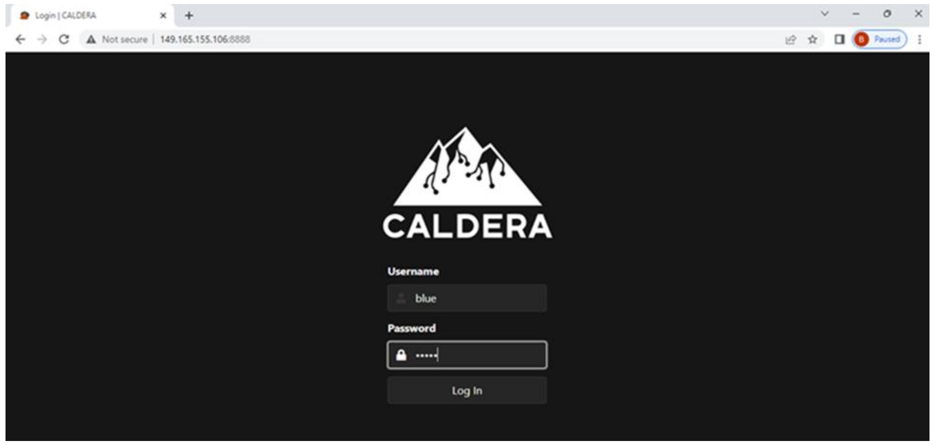
```
api_key_blue: pl06lwOSYCTp-SWZcsrZzFgbSuWYqh6ppKYX1fytQLY
api_key_red: 3Dk1lhYLGIj0h_WvAPYDjRE7p0aqGuZXRMlmkjvza8s
app.contact.dns.domain: mycaldera.caldera
app.contact.dns.socket: 0.0.0.0:8853
app.contact.ftp.host: 0.0.0.0
app.contact.ftp.port: 2222
app.contact.ftp.pword: caldera
app.contact.ftp.server.dir: ftp_dir
app.contact.ftp.user: caldera_user
app.contact.gist: API_KEY
app.contact.html: /weather
app.contact.http: http://0.0.0.0:8888
app.contact.slack.api_key: SLACK_TOKEN
app.contact.slack.bot_id: SLACK_BOT_ID
app.contact.slack.channel_id: SLACK_CHANNEL_ID
app.contact.tcp: 0.0.0.0:7010
app.contact.tunnel.ssh.host_key_file: REPLACE_WITH_KEY_FILE_PATH
app.contact.tunnel.ssh.host_key_passphrase: REPLACE_WITH_KEY_FILE_PASSPHRASE
app.contact.tunnel.ssh.socket: 0.0.0.0:8022
app.contact.tunnel.ssh.user_name: sandcat
app.contact.tunnel.ssh.user_password: s4ndc4t!
app.contact.udp: 0.0.0.0:7011
app.contact.websocket: 0.0.0.0:7012
auth.login.handler.module: default
crypt_salt:
```

Configuring Caldera

- Before, being able to deploy any agent in Caldera we needed to assure that the security groups assigned to our Caldera main instance had a rule for connecting to HTTP server.
- We also needed to nano into the default.yml file to make sure that everything that comes with Caldera such as plugins, encryptions keys, and credentials for blue and red team were configured properly.
- The security group needs to run from internal network not JetStreams

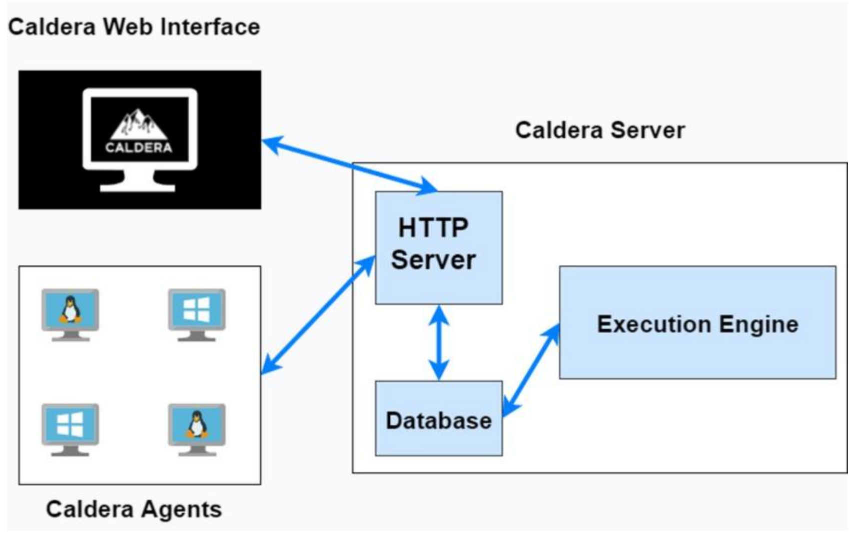
Deploying Caldera Agents

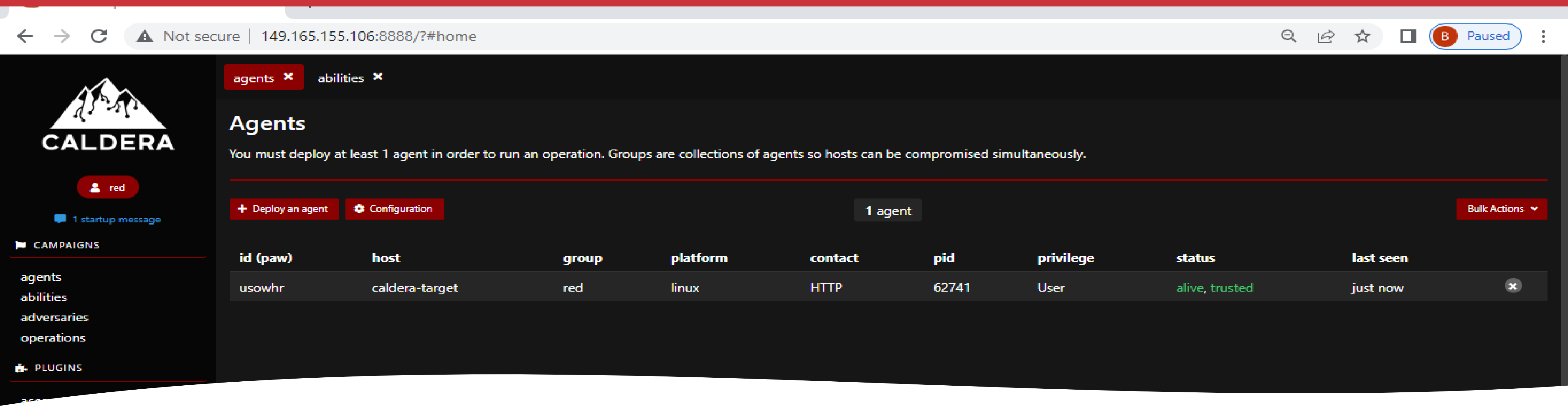
- We log in to http://<caldera-test-server ip>:8888 with the red or blue campaign using the password found in the conf/local.yml file.
- Deploying an agent is the first step for performing any adversary emulation in either red or blue team.
- The agent options offered by Caldera are: Sandcat, Ragdoll & Manx
- Most of these agents can be deployed in target box OS: Linux, Windows, or Darwin but, some have limitations



Username: blue

Password: •••••




Not secure | 149.165.155.106:8888/?#home

Agents

You must deploy at least 1 agent in order to run an operation. Groups are collections of agents so hosts can be compromised simultaneously.

1 agent

id (paw)	host	group	platform	contact	pid	privilege	status	last seen
usowhr	caldera-target	red	linux	HTTP	62741	User	alive, trusted	just now

Deploying Caldera

- To deploy CALDERA you have to deploy an agent.
- When, you click on 'Deploy an agent' it will prompt you to choose an agent.
- **Sandcat** is a good one to start with and a platform (target operating system).
- In our case, the target operating system will be Linux. Here are steps that we follow to deploy this agent:

1. **Check agent options** are correct
2. **app.contact.http** represents the HTTP endpoint, that the C2 server is listening on for agent requests and beacons.
3. **agents.implant_name** represents the base name of the agent binary.

Deploying Caldera (Cont.)

4. **agent.extensions** takes in a comma-separated list of agent extensions to compile with your agent binary. When selecting the associated deployment command, this will instruct the C2 server to compile the agent binary with the requested extensions, if they exist.

5. **Choose a command to execute** on the target machine and paste it there

6. **New agent** should appear with **an PID** in dashboard

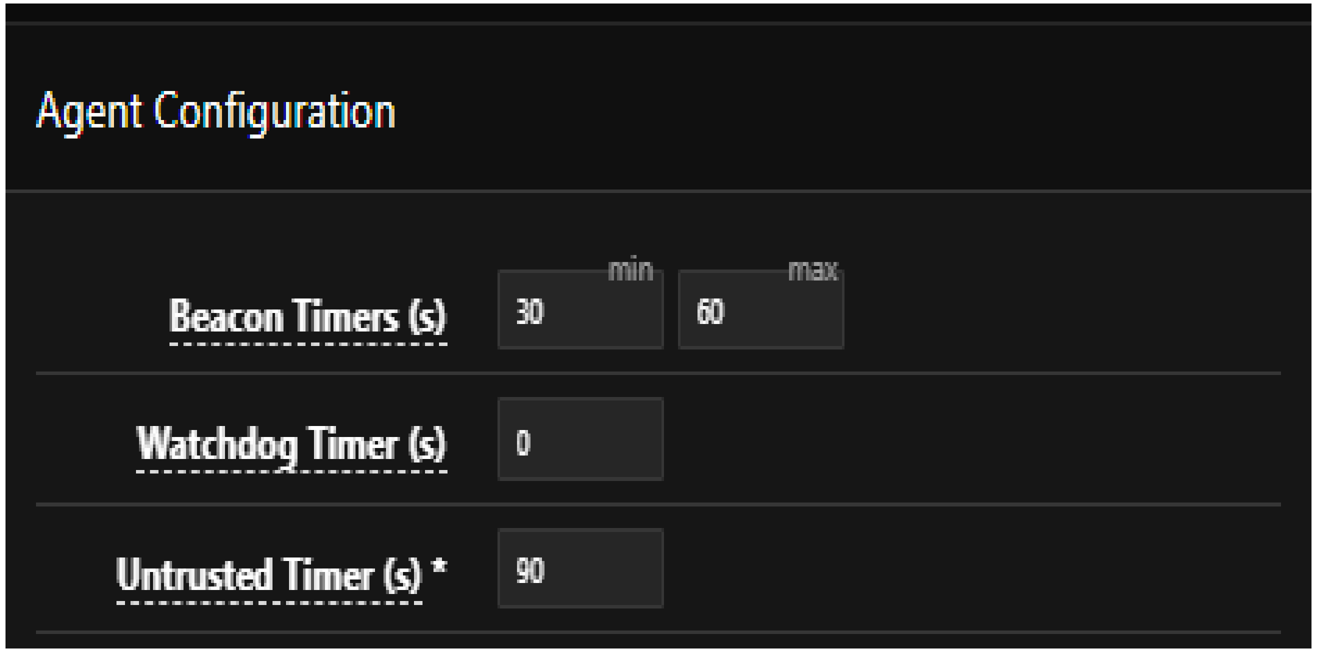
7. **To kill an agent**, use the **"Kill Agent"** button under the agent-specific settings. The agent will terminate on its next beacon.

Agent Settings

- **Beacon Timers:** Set the minimum and maximum seconds the agent will take to beacon home. These timers are applied to all newly-created agents.

- **Watchdog Timer:** Set the number of seconds to wait, once the target agent is unreachable, before killing an agent. This timer is applied to all newly-created agents.

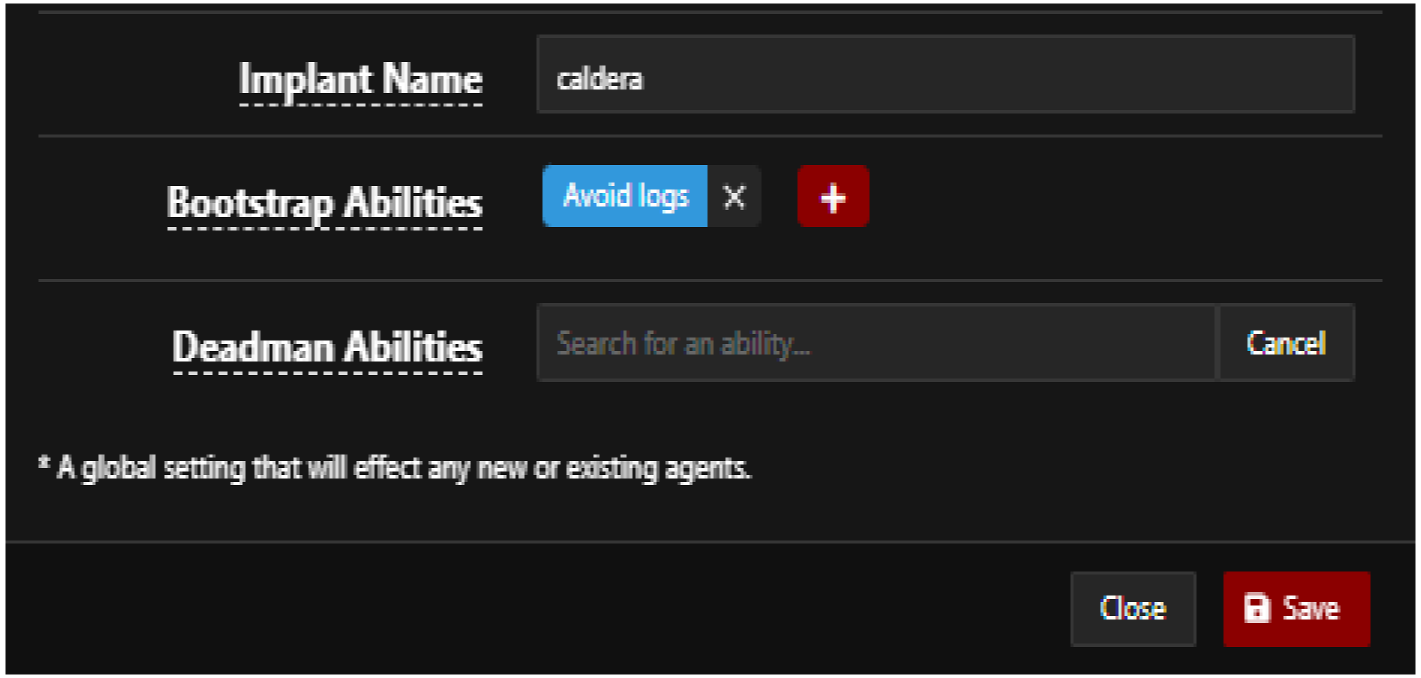
- **Untrusted Timer:** Set the number of seconds to wait before marking a missing agent as untrusted. Operations will not generate new links for untrusted agents. This is a global timer and will affect all running and newly-created agents.


Agent Configuration

	min	max
Beacon Timers (s)	30	60
Watchdog Timer (s)	0	
Untrusted Timer (s) *	90	

Agent Settings (Cont.)

- **Implant Name:** The basename of newly-spawned agents. If necessary, an extension will be added when an agent is created.
- **Bootstrap Abilities:** A comma-separated list of ability IDs to be run on a new agent beacon. By default, this is set to run a command which clears command history.
- **Deadman Abilities:** A comma-separated list of ability IDs to be run immediately prior to agent termination. The agent must support deadman abilities in order for them to run.



Implant Name: caldera

Bootstrap Abilities: Avoid logs ✕ +

Deadman Abilities: Search for an ability... Cancel

* A global setting that will effect any new or existing agents.

Milestone 6

Performing APT Attack Simulations

Red Team APT

Simulation #1

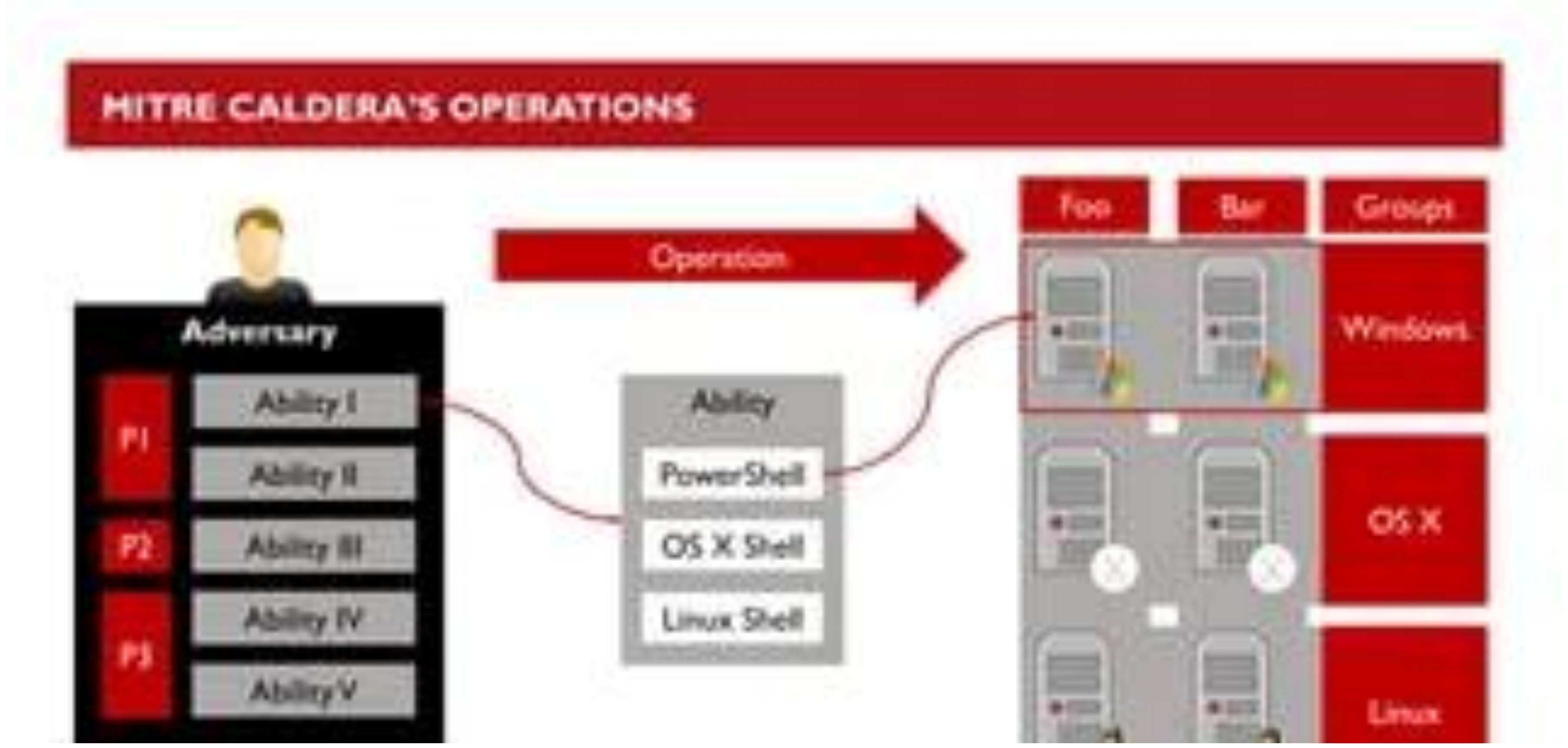
Blue Team APT Simulation

Simulation 1

Deployed Agent

- Sandcat: also referred to as 54ndc47 is a remote access agent written in GoLang for cross platform compatibility, and is the agent we will deploy on the endpoint(s) we want to execute our operations against.
- Script 1: CALDERA's default agent, written in GoLang. Communicates
- Script 2: Download with a random name and start as a background processhrough the HTTP(S) contact by default

Operations Architecture

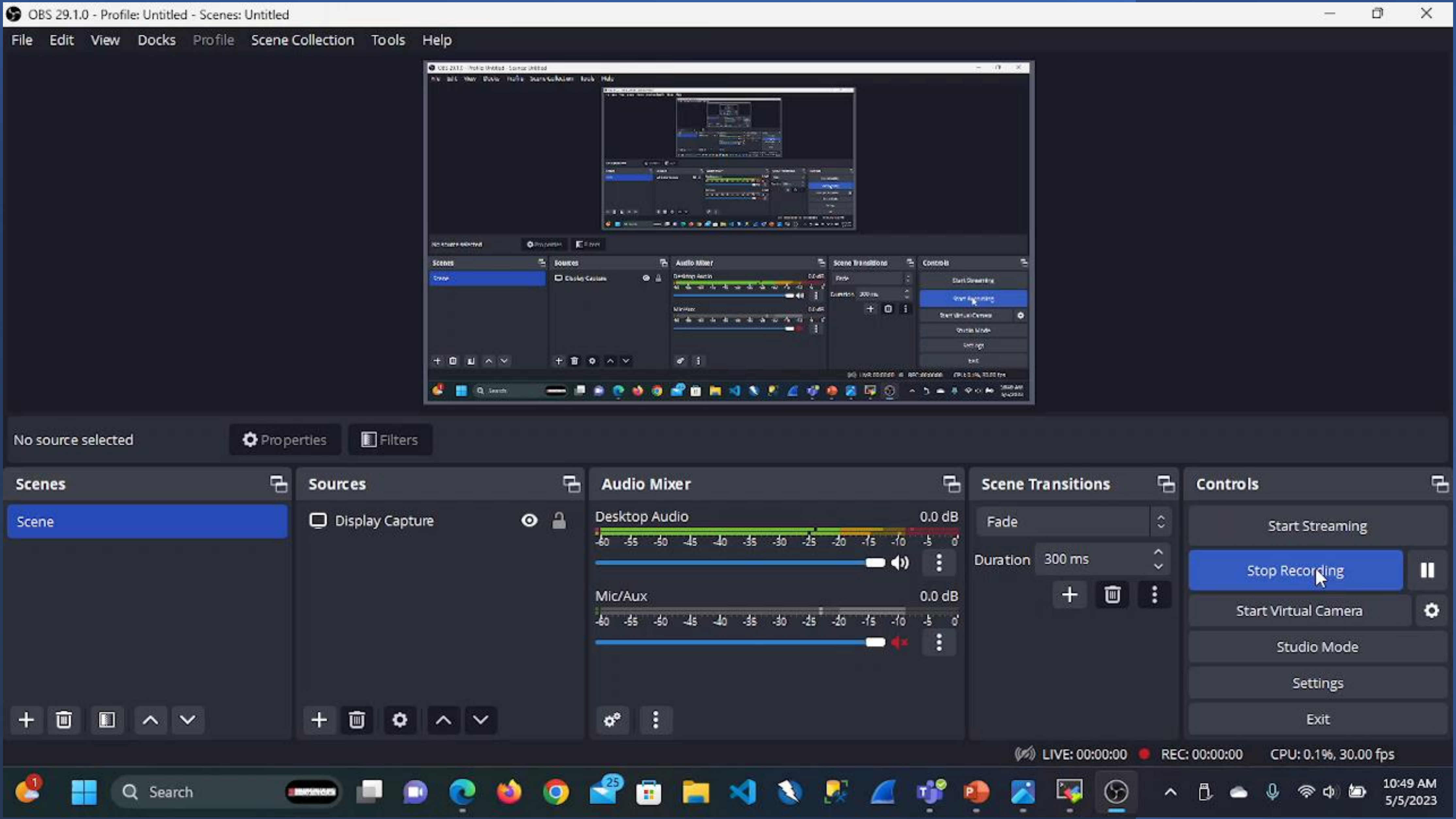


Prototype and Demo

Prototype

Our system utilizes Ansible to generate instances of Caldera and its targets. Caldera runs and delivers agents to its connected targets. From there, analysis is done on the activity on systems infected with the agent payloads.

Our Demo videos will display the process of running APT simulations on target boxes.


OBS 29.1.0 - Profile: Untitled - Scenes: Untitled

File Edit View Docks Profile Scene Collection Tools Help

No source selected Properties Filters

Scenes: Scene

Sources: Display Capture

Audio Mixer: Desktop Audio 0.0 dB; Mic/Aux 0.0 dB

Scene Transitions: Fade; Duration 300 ms

Controls: Start Streaming; Stop Recording; Start Virtual Camera; Studio Mode; Settings; Exit

LIVE: 00:00:00 REC: 00:00:00 CPU: 0.1%, 30.00 fps

Challenges and Limitations

Challenges

- **Creation of Caldera Playbook was increasingly complex**
  - Connected services and functions were difficult to compile together
  - Took extensive troubleshooting and error checking
  - Time taken to configure playbook surpassed estimates
- **Deploying the Caldera Agents was a bit challenging**
  - Took extensive troubleshooting of script errors and cloud issues
  - Troubleshooting Security Groups was a hassle
- **Key pairs kept providing public key issues that slowed progress**
  - Linux key copying utilities did not work; we needed to copy keys manually

Limitations

- **Our time management provided less time than desired**
  - Too much time was spent in research
  - Troubleshooting and error checking was not properly factored into scheduling
- **Related information is hard to find or outdated**
  - Many information sources used outdated versions of systems with deprecated commands and features
  - Added to time needed for troubleshooting
- **Smaller team limited initial project scope**

Future Work

Future Work

- **Expand framework capabilities**
- **Harden security with separate subnets for different project areas**
  - Utilize multiple security groups for least privilege
- **Utilize logging and SIEM systems**
  - Zeek/Bro, Fortinet, SecurityOnion
- **Collect data through Elastic Search to simplify sorting, graphing and charting**
  - Allows us to pull and export data for future reports

Internship Experience

Description (Optional)

Internship Experience - Anthony

- **Interned with the UIW Tier II Support Team**
- Assisted in the setup of different offices and labs around campus
  - Athletics Portable Lab
  - UIW Welcome Center Office
  - Ettling Center Refresh
- Shadowed inventory auditing for IT systems checked out to staff
  - Performed supply room audits for quality checking
- Performed system OS imaging and flashing
  - Prepped new systems for deployment and old systems for donation
- Learned and serviced open tickets on Freshservice Ticket system

Internship Experience - Bryanna

- **Interned with the Sister's of Charity of the Incarnate Word IT Support**
- **Assisted in conducting and managing an IT Asset Inventory of Congregation staff and sisters in SharePoint**
    - Replacing old asset tags with new tags
    - Installing a remote agent and Webroot AV to all assets
    - Having staff and sisters signed a Congregation Property Inventory form
    - Keeping track of all assets' status (new, replaced, decommissioned)
- **Troubleshoot a variety of tickets (low-high) per JIRA ticket system request**
- **Established the first ever Cybersecurity-IT-101 Security Awareness Training Program**
    - Utilized Rise360 for delivery of online instruction(English/Spanish)
    - Progress Meetings with mentor/supervisor
    - Cybersecurity Modules (Password Ethics/Management, Social Engineering, Ransomware, Adware vs Spyware & CyberExtortion)
    - IT Modules(Windows 10 Basics, Cookies & Microsoft Office 365 Application Fundamentals

Conclusion

Conclusion

- Ansible Playbook is a great, powerful and reliable tool for automation across an enterprise. No special coding was necessary to orchestrate an application environment no matter where it's deployed in our case, the Cloud.
- Ansible architecture is so flexible since, it can chain multiple playbooks and avoid the stress of having to execute tasks more than once. The tasks become re-usable once established in playbook.
- Caldera lets business reduce resources they need for assessments so Red Team focuses on sophisticated solutions to hardening problems.
- Caldera tools help the Blue Team test techniques themselves and continuously improve.

Our project's code has been open-sourced in the following repository: https://github.com/awjamieson83/AJ-BP-UIWCSEC-Practicum

Resources

Resources

- CrowdStrike Team. (2023, February 28). *2023 global threat report: CrowdStrike*. crowdstrike.com. Retrieved from https://www.crowdstrike.com/global-threat-report/
- CrowdStrike Team. (2023, April 19). *What is an advanced persistent threat (APT)?* crowdstrike.com. Retrieved from https://www.crowdstrike.com/cybersecurity-101/advanced-persistent-threat-apt/
- Heidi, E., & Camisso, J. (2022, October 8). *How to install and configure Ansible on ubuntu 22.04*. DigitalOcean. Retrieved from https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-22-04
- Heidi, E. (2021, April 15). *How to create and use templates in Ansible Playbooks*. DigitalOcean. Retrieved from https://www.digitalocean.com/community/tutorials/how-to-create-and-use-templates-in-ansible-playbooks
- NVISOsecurity. (2019, July 7). *NVISOSECURITY/Ansible-Caldera: Ansible role for Mitre caldera*. GitHub. Retrieved from https://github.com/NVISOsecurity/ansible-caldera
- MITRE Caldera. (2019, March 23). *Our Impact Intellectual Property: Caldera*. Retrieved April 25, 2023, from https://www.mitre.org/our-impact/intellectual-property/caldera
- MITRE Caldera. (2019, March 23). *Who we are*. Retrieved April 25, 2023, from https://www.mitre.org/who-we-are
- Installing caldera. (2019, April). *Installing CALDERA - caldera documentation*. Retrieved April 28, 2023, from https://caldera.readthedocs.io/en/latest/Installing-CALDERA.html
- Basic Usage. (2019, April). CALDERA Basic Usage. Retrieved April 28, 2023, from https://caldera.readthedocs.io/en/latest/Basic-Usage.html
- Caldera Agents. (2019, April). Agent Settings. Retrieved April 28, 2023, from https://caldera.readthedocs.io/en/latest/Basic-Usage.html#agents