Various Attack Scenario Generation

If you are using exported ova, remember to change internet and shared folder, otherwise you may encounter confliction.

Folder Structure

```
Handover
├── Result
│   ├── graphs
│   ├── json
│   ├── logs
│   ├── processed.json
│   ├── toSkip.json
│   └── transformError.json
├── Monitor_Host
├── abilities_info.json
├── automation_TTP.py
├── log_filter.py
├── monitor_host.sh
├── spade_collect.sh
├── spade_transform.sh
├── ragdoll.py
├── README.md
└── structure.jpeg
```

`Result`: The experiment results.

`Result/graphs`: Provenance graph in png format.

`Result/json`: Provenance graph in JSON format.

`Result/logs`: The filtered log that can be used to transform into graph and json.

`Result/processed.json`: Successfully generated graphs/json/logs.

`Result/toSkip.json`: TTPs that can't be recorded (e.g., delete file system).

`Result/transformError.json`: Can record logs but encounter errors when automatically transforming logs into graphs/json (can be transformed by manually run the scripts).

`Monitor_Host`: Folder of virtual machine "Monitor_Host".

`Official_Server.ova`: Exported virtual machine "Black Hacker".

`Official_Victim_2.ova`: Exported virtual machine "Benign User".

`abilities_info.json`: The information of TTPs in Caldera, you can obtain this with REST API.

`automation_TTP.py`: The python script that can run the automation system.

`log_filter.py`: The filter to extract log events revelant to specific PID, `python3 log_filter.py -h` for more details.

`monitor_host.sh`: The script for victim1 to execute in snapshot "collect" when power on.

`spade_collect.sh`: The script for spade to execute in snapshot "collect" when power on.

`spade_transform.sh`: The script for spade to execute in snapshot "transform" when power on.

`ragdoll.py`: The script to connect Monitor Host to Caldera server, executed in `monitor_host.sh`.

`structure.jpeg`: Graph used in README.md.

Environment Setup

"./structure.jpeg" could not be found.

Black Hacker (Server)

The virtual machine we build Caldera server on.

Basic Information

Account:

```
User: server
Password: server
```

Network:
two network interface card

```
Adapter 1: Host Only
IP: 192.168.56.200

Adapter 2: Internal Network (IntServer)
IP: 192.168.1.1
```

Plugins

Download necessary plugins

```
sudo apt update
sudo apt install golang vim git python3-pip
```

MITRE Caldera

```
git clone https://github.com/mitre/caldera.git --recursive
cd caldera
pip3 install -r requirements.txt
```

Monitor Host (Victim1)

The agent of Caldera server.

Basic Information

Account:

```
User: root
Password: root

User: spade
Password: spade

User: victim1
Password: victim1

su: root
```

Network:
two network interface card

```
Adapter 1: Internal Network
IP: 192.168.1.2

Adapter 2: Internal Network
IP: 192.168.2.1
```

Use netplan to set static ip.
After you set these two static ip, the NAT would no longer functioning. If you want to use NAT, you may have to move `01-installer-config.yaml` out from `/etc/netplan` and execute `sudo netplan apply`.
`sudo vim /etc/netplan/01-installer-config.yaml`

```
network:
    version: 2
    renderer: networkd
    ethernets:
        enp0s3:
            dhcp4: false
            dhcp6: false
            addresses: [192.168.1.2/24]
            routes:
            - to: "default"
              via: "192.168.1.1"
              nameservers:
                  addresses: [8.8.8.8, 8.8.4.4]
        enp0s8:
            dhcp4: false
            dhcp6: false
            addresses: [192.168.2.1/24]
            routes:
            - to: "default"
              via: "192.168.2.1"
              nameservers:
                  addresses: [8.8.8.8, 8.8.4.4]
```

`sudo netplan apply`

Plugins

Download necessary plugins

```
sudo apt update
sudo apt install vim git wget curl python3 python3-pip auditd
pip3 install beautifulsoup4 requests
sudo pip3 install beautifulsoup4 requests
```

SPADE

This is the tool that can convert the logs recorded by Linux Audit System to provenance graph.
Just strictly follow the document then you can setup SPADE.

Warning

- SPADE won't record events related to the user executing it, so you need to create a user.
- When you want to use reporter and storage, there is a strict ordering of the commands. add storage -> add reporter -> remove reporter -> remove storage

If you add reporter first, the events occur before you add storage would be discarded.

If you remove storage first, nothing would be recorded (I don't know why and this is not mentioned in the document)

Benign User (Victim 2)

The virtual machine that is under same subnet with Monitor Host.

Basic Information

Account:

```
User: victim2
Password: victim2
```

Network:

```
Adapter 1: Internal Network
IP: 192.168.2.2
```

Plugins

Download necessary plugins

```
sudo apt updatev
sudo apt install vim git wget
```

Shared Folder in VirtualBox

1. VirtualBox->Setting->Shared Folders
   Add the folder you want to share, don't need to check automount or read-only.
2. Boot VM
3. Devices->Insert Guest Additions CD image
4. Install guest additions

```
sudo mkdir /media/cdrom
sudo mount -t iso9660 /dev/cdrom /media/cdrom
sudo /media/cdrom/./VBoxLinuxAdditions.run
```

5. Auto mount

`sudo vim /etc/fstab`

```
shared  /home/<username>/shared vboxsf  defaults        0       0
```

`sudo vim /etc/modules`

```
vboxsf
```

`sudo reboot`

TTP Provenance Graph Generation

If you need to manipulate Caldera through command rather than GUI, refer to this document.

You can execute the `automation_TTP.py` to run the system.
Before running the system, you need to:

1. Run the Caldera Server on Black Hacker
   `python3 server.py --insecure`, or `./run.sh`

2. Power on Benign User

When you execute `automation_TTP.py` on HostOS

HostOS-side workflow:

1. Restore snapshot "collect" of Monitor Host
2. Power on it and wait for the configuration complete
3. Issue a TTP attack through terminal and wait
   `curl -H "KEY:ADMIN123" -X POST 192.168.56.200:8888/plugin/access/exploit -d {{"paw":"agentabc","ability_id":"{ability_id}","obfuscator":"plain-text"}}`
   paw is the id of the agent, it is random generated, or you can customize it through modifying ragdoll.py in Monitor Host (`"paw": "agentabc"`).
4. Power off the VM
5. Filter the recorded audit.log
6. Restore snapshot "transform" of Monitor Host
7. Power on it and wait for the configuration complete
8. Wait for the transformation finish
9. Retrieve the filtered.log, JSON, and provenance graph.
10. Back to 1.

Monitor-Host-side workflow:

We use two snapshots of Monitor Host in `automation_TTP.py`, `collect` is for collecting audit.log, and `transform` is for transforming the filtered audit log into JSON and provenance graph.
In the snapshots, scripts would be executed when power on (by crontab), and the scripts for collect and transform are different but with the same name (for implementation simplicity).

Crontab tutorial:
For root, `vim /etc/crontab`.
For normal user, login with the user and `crontab -e`.
`@reboot sleep 3 && bash && command to execute`

Workflow of the script "SPADE for collect"

1. Add reporter (outputLog)
2. Wait for the attack finish
3. Remove reporter

Workflow of the script "victim for collect"

1. connect to Caldera
   `python3 ragdoll.py -W http://192.168.1.1:8888/weather &`

Workflow of the script "SPADE for collect"

1. Add storage (JSON and Graphviz respectively)
2. Add reporter (inputLog)
3. Wait for the transformation
4. Remove reporter
5. Remove storage

Malware Provenance Graph Generation

The procedures are almost the same as TTP provenance graph generation, the only difference is that you may have to add a new ability in Caldera to activate the malwares.