BLEKINGE INSTITUTE OF TECHNOLOGY

SSH and Bitrate Monitoring Using DPMI

User Manual 18-01-2018

Version 1.0

Team Members:

1.GURRAM KARTHIK

2.HOSSEN SADDAM

3.JANAGAM ANIRUDH

4.KONDEPATI DIVYA NAGA KRISHNA

5.MAMIDI SAI PRAKASH

6.NAMANA SAI KIRAN KUMAR

7.SATHI SANTHOSH REDDY

8.YALAVARTHI SRI LEKHA

9.KARUMANCHI MAHESH

<u>Summary of contents</u>:

1. USER GUIDE:

---ABOUT SSH MONITOR--

This project describes the SSH (Secure Shell) dictionary attacks detection where we are required to monitor SSH connection attempts to avoid possible dictionary attacks. Dictionary attacks are a type of fraudulent attempts made to compromise a system with SSH connection. This can also involve trying many username and password combinations on the (remote) system to gain access to it. This type of attack can be successfully recognized from analyzing network data. In this subpart, the main objectives will be to configure SSH such that if more attempts than X/Minute, the connection must be blocked for no further attempts for Y Minutes. SSH must be configured in such a way that the blockage time increases every x/minute's attempts made and the increases by 2*Y minutes, 3*Y minutes etc. Suitable IP tables will be configured to block/unblock automatically.

--ABOUT BITRATE--

In DPMI, the measurement points capture traffic between clients, often sent live as measurement streams for consumers. Each MP has set of filters e.g.: only IP traffic. The captured measurement streams are saved as a CAP file. The PCAP streams are extracted from measurement streams using a CAP2PCAP tool from libcap_utlis, in a way readable by us.

We are required to monitor bitrates using measurement streams. The bit rates are to be stored into Influx database and visualized graphically in Grafana with a sample time of one second.

2. INSTALLATION:

A) SYSTEM REQUIREMENTS:

1. Python
2. Pip
3. Curl
4. Consumer-bitrate
5. DPMI Stream
6. Grafana
7. Flask
8. Flask Python Module
9. Bash
10. Shell
11. Influx DB
12. Influx DB Python Module
13. Ruby
14. MySQL
15. Ruby-MYSQL Libraries & Dependencies
16. Ruby-gems

B) INSTALLATION STEPS:

1. Consumer-Bitrate:

   git clone https://github.com/DPMI/consumer-bitrate

   cd consumer-bitrate

   sudo make && sudo make install

2. Libcap-utils:

   Use the below link to install libcap utilities:

   http://libcap-utils.readthedocs.io/en/latest/

3. Influx DB:

   Type the below command in the terminal:

   Sudo apt-get update && Sudo apt-get install influxdb

   Sudo service influxdb start

   influx

   Create database anm

4. MySQL:

   Sudo apt-get update

   Sudo apt-get install mysql-server

5. Grafana:

   Go to below official link to install grafana:

   http://docs.grafana.org/installation/debian/

6. Ruby:

   Go to below official link to install Ruby:

   https://www.ruby-lang.org/en/documentation/installation/

   gem install mysql

7. Rest-server:

   To install Rest and its dependencies type below commands in terminal:

   Pip install flask

   Sudo apt-get install python-influxdb

   To start the Rest-server:

   cd ANMGR4a

   Sudo python restapi.py

8. Other scripts:

   cd ANMGR4a

   type in the terminal:

   sudo sh anm.sh or give permissions to run the shell script

   sudo chmod u+x anm.sh – To capture the Streams and store ips into Mysql

   Run the following in the firewall device:

   Sudo ruby firewall.rb – To retrieve ips from Mysql and add ips to iptables based on the expiry of the ip

3. RESTAPI:

Requirements:

a. Python 2.7 +
b. Python modules – flask

APIs

1.Functionality – To remove a Blocked IP from Iptables

- **URL**

  /removeBlockedIP

  Method:

  `GET`

- **URL Params**

  Required:

  `ip= [ip to provide]`

- **Backend command**

  `iptables -A INPUT -s {} -j DROP`

- **Success Response:**
  - **Code:** 200

    Content: `{output: <output of the command>}`

- **Error Response:**
  - **Code:** 500

    Content: `{error: " Could not find ip. URL format removeBlockedIP? ip=<ipaddress_to_unblock> "}`

- **Sample Call:**

```
Curl -I http:192.168.185.58/removeBlockedIP?ip=”enter_ip_to_be_unblocked”
```

2. Functionality – To add an Ip to Iptables

- **URL**

  /addIP

  Method:

  `GET`
- **URL Params**

  Required:

  `ip= [ip to provide]`
- **Backend command**

  `iptables -A INPUT -s {} -j DROP`
- **Success Response:**
  - **Code:** 200

    Content: `{output: <output of the command>}`
- **Error Response:**
  - **Code:** 500

    Content: `{error: " Could not find ip. URL format addIP? Ip=<ipaddress_to_block> "}`
- **Sample Call:**

```
Curl -I http://192.168.185.58/addIP?ip=”enter_ip_to_be_blocked “
```

3. Functionality – To list Blocked ips from iptables

- **URL**

  /listIP

  Method:

  `GET`

- **URL Params**

  Required:

  `ip= [ip to provide]`

- **Backend command**

  `Sudo iptables -S`

- **Success Response:**
  - **Code:** 200

    Content: `{output: <output of the command>}`

- **Error Response:**
  - **Code:** 500

    Content: `{error: "Could not find ip. URL format listip`

- **Sample Call:**

```
Curl -I http:192.168.185.58/listIP
```

4. Functionality - To calculate bitrates by passing suitable filters and tag

- **URL**

  /addInstance

  Method:

  `POST`

- **URL Params**

  Required:

  `filter= [filters for protocol]`

  `tag= [tag names]`

- **Backend command**

-
  ```
  sudo bitrate -i ens4 01::71 01::72 --ip.proto=<filter> --format=influx
  --influx-user="admin" --influx-pwd="admin" --influx-tag="<tag>" --
  influx-url="http://localhost:8086/write?db=anm" >/dev/null 2>&1 &
  ```

- **Success Response:**
  - **Code:** 200

    Content: `{output: <output of the command>}`

- **Error Response:**
  - **Code:** 500

    Content: `{error: Could not find ip. URL format - /addInstance?filter=<filter_string>&tag=<tag> “}`

- **Sample Call:**

```
Curl -d filter=”—if=d03 –ip.proto=TCP” -d tag=”OUT-tcp” -X POST
http://192.168.185.58/addInstance
```

5. Functionality - To kill a running process using Process id (PID)

- **URL**

  / /killInstance

- **Method:**

  `GET`

- **URL Params**

  Required:

  `id= [process id to kill]`

- **Backend command**
  - `Sudo kill -9 <id>`
- **Success Response:**
  - **Code:** 200

    Content: `{output: <output of the command>}`
- **Error Response:**
  - **Code:** 500

    Content: `{error: 'Could not find id. URL format - /killInstance?id=<id-seen in listInstance output> '}`
- **Sample Call:**

```
Curl -i http://192.168.185.58/killInstance?id=" enter_process_id_to_kill"
```

6. Functionality – To show running Process

- **URL**

  / listInstance

- **Method:**

  `GET`
- **URL Params**

  `No param required`
- **Backend command**
  - `ps -lef | grep bitrate`
- **Success Response:**
  - **Code:** 200

    Content: `{output: <output of the command>}`
- **Error Response:**
  - `No error response. Content null if did not get any output`
- **Sample Call:**

```
Curl -i http://192.168.185.58/listInstance
```

4. ACKNOWLEDGEMENT:

This project couldn’t be completed without the guidance of Patrik Arlos, Professor, Blekinge Institute of Technology. We would like to show our gratefulness in the form of this acknowledgement.