Engineering Behind Application Network Security Solution

The Application Network Security solution built on Ubuntu Linux Server has the below components,

1) Host Firewall
2) IDS (Intrusion Detection System)
3) Python Agents

In this case we use ConfigServer Security & Firewall (CSF) as the Host Firewall. Suricata as the IDS.

Both Host Firewall and IDS have their own sensors, listening to virtual NICs or Physical NICs.
Note : NIC (Network Interface Card)

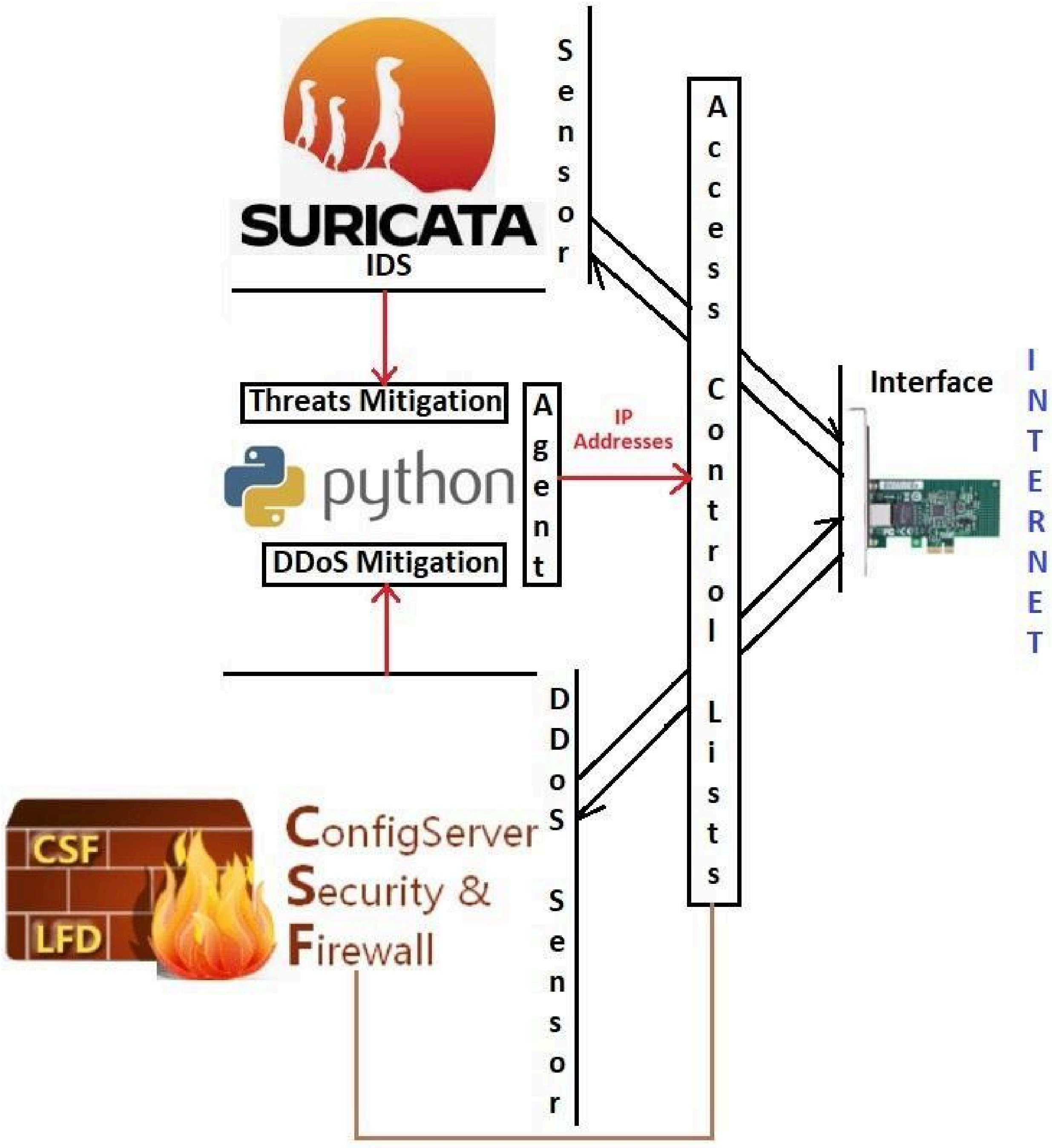
We use two Python agents in between Host Firewall and IDS to bridge both functionalities , thereby achieving effective Application Network Security.

One Python agent listens to the Host Firewall Sensor , another listens to the IDS sensor.

In common to both Python agents , the result is capture of affected vulnerable source IP addresses.

Both the Python agents apply the captured vulnerable source IP addresses to Access Control Lists (csf.deny file) of ConfigServer Security & Firewall (CSF).

Once the vulnerable source IP addresses are applied to Access Control Lists (csf.deny file) of ConfigServer Security & Firewall (CSF), the host blocks the connection at the same instant.


SURICATA
IDS
Sensor
Access Control Lists
Interface
INTERNET
Threats Mitigation
python
Agent
IP Addresses
DDoS Mitigation
DDoS Sensor
CSF
LFD
ConfigServer Security & Firewall

The HOST Firewall (ConfigServer Security & Firewall (CSF))

The host firewall has the following important components,

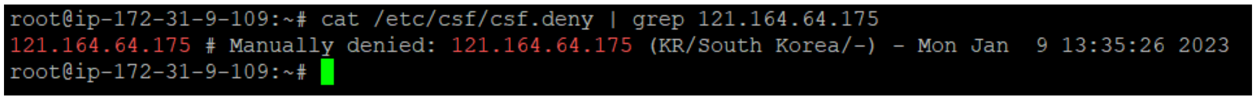
1) Sensor
2) Access to threat intelligence
3) IP Addresses Exception File (csf.ignore)
4) Access Control Lists File (csf.deny)

Let us consider an example of DDoS mitigation, the CSF sensor log output looks like ,

```
Jan  9 13:35:23 ip-172-31-9-109 kernel: [9427409.852981] Firewall: *TCP_IN Blocked* IN=ens5
OUT= MAC=0a:93:9a:c2:ea:5a:0a:15:c0:b5:41:aa:08:00 SRC=121.164.64.175
DST=172.31.9.109 LEN=60 TOS=0x00 PREC=0x00 TTL=41 ID=16869 DF PROTO=TCP
SPT=56442 DPT=5555 WINDOW=65535 RES=0x00 SYN URGP=0
```

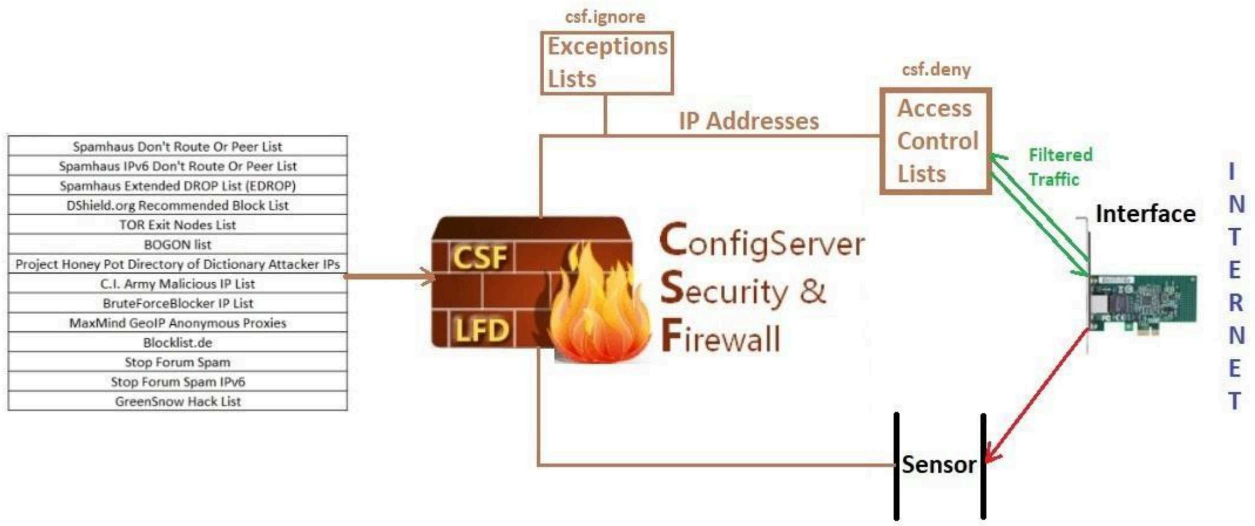
The flag **SYN** refers to DDoS attacks.

The python agent looks at the SRC=121.164.64.175 in the same log line and instantly applies this IP address to csf.deny file (Access Control Lists).



```
root@ip-172-31-9-109:~# cat /etc/csf/csf.deny | grep 121.164.64.175
121.164.64.175 # Manually denied: 121.164.64.175 (KR/South Korea/-) - Mon Jan  9 13:35:26 2023
root@ip-172-31-9-109:~#
```

Threat Intelligence has multiple IP address sources getting downloaded to IP address tables periodically, please refer the diagram below,


Spamhaus Don't Route Or Peer List
Spamhaus IPv6 Don't Route Or Peer List
Spamhaus Extended DROP List (EDROP)
DShield.org Recommended Block List
TOR Exit Nodes List
BOGON list
Project Honey Pot Directory of Dictionary Attacker IPs
C.I. Army Malicious IP List
BruteForceBlocker IP List
MaxMind GeoIP Anonymous Proxies
Blocklist.de
Stop Forum Spam
Stop Forum Spam IPv6
GreenSnow Hack List
csf.ignore
Exceptions Lists
IP Addresses
csf.deny
Access Control Lists
Filtered Traffic
Interface
CSF
LFD
ConfigServer Security & Firewall
INTERNET
Sensor

Due to the presence of Threat Intelligence from various sources and sensor results , we can expect false positives in getting legitimate IP addresses blocked, to override the situation we use csf.ignore file (/etc/csf/csf.ignore) for adding IP address exclusions.

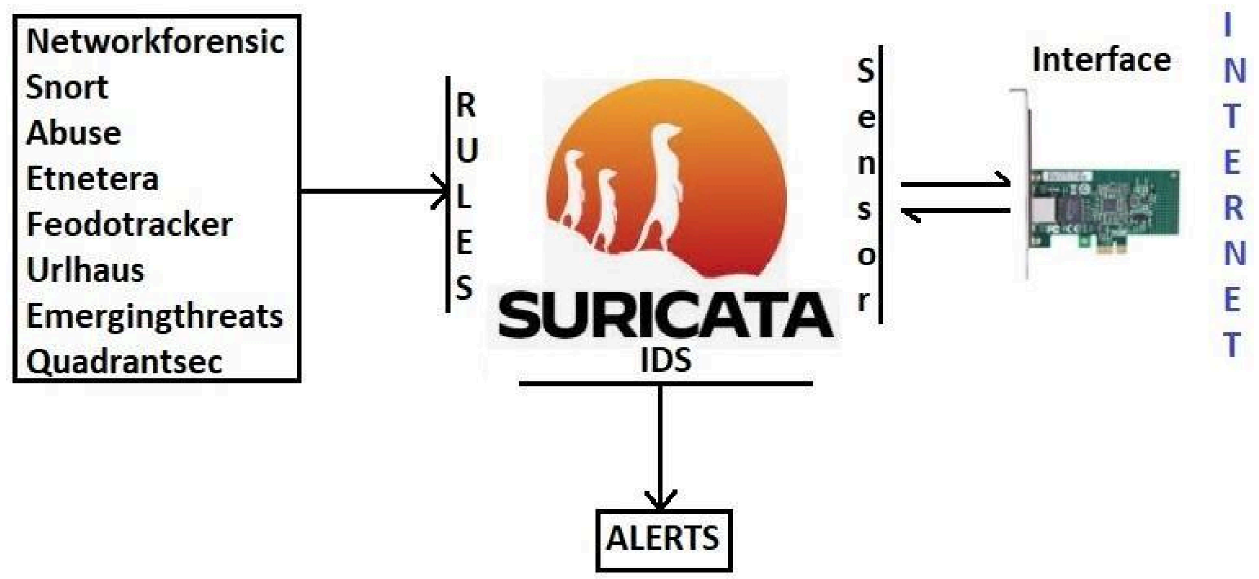
Access Control Lists is the csf.deny (/etc/csf/csf.deny) file that acts as an IP address blocker.An IP address added to this file gets blocked at that instant.

Suricata IDS (Intrusion Detection System)

The IDS has three important components,

1) Sensor
2) Rules
3) Alerts

The Sensor will be listening to the virtual/physical network interface cards.The traffic that flows inbound/outbound of the virtual/physical network interface card will be parsed to the rules engine of the IDS. The rules engine has plenty of rules from various sources.Refer the diagram.



If the traffic pattern matches any of the rules, alerts are generated like ,

```
01/09/2023-14:27:22.892704  [**] [1:2001219:20] ET SCAN Potential SSH Scan [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP} 171.225.184.165:19114 ->
172.31.9.109:22
```

The Python agent picks up the IP address **171.225.184.165** and applies it to the csf.deny (etc/csf/csf.deny) file (Access Control Lists).An IP address added to this file gets blocked at that instant.

Understanding the automated script files involved to build the Network Security Solution

install.sh
installfirewallids.sh
uninstall.sh

If you run the install.sh file , installfirewallids.sh and uninstall.sh will get downloaded.

The actual Host Firewall (CSF) and IDS (Suricata) install file is installfirewallids.sh

When installfirewallids.sh runs, the complete Network security solution gets built.

Files created after running installfirewallids.sh are,

AutoBlock.py
IDSAutoBlock.py
AutoBlocker.log
IDSAutoBlocker.log
BlockedIPs
IDSBlockedIPs

IPinput.sh
clearcsfdeny.sh
startfirewallids.sh
stopfirewallids.sh
restartfirewallids.sh
startids.sh
statusfirewallids.sh

afterchanges.sh
beforechanges.sh

suricatarulesupdate.sh

uninstall.sh

uninstallfirewallids.sh

The python agent files AutoBlock.py and IDSAutoBlock.py run as backend systemd service files.

AutoBlock.py as ddosblock.service (DDoS Mitigation - CSF)
IDSAutoBlock.py as idsblock.service (Threats Mitigation - IDS)

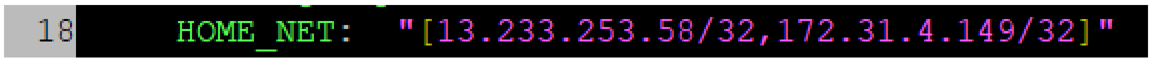
Respective agents log files,

AutoBlock.py - AutoBlocker.log (DDoS Mitigation - CSF)
IDSAutoBlock.py - IDSAutoBlocker.log (Threats Mitigation - IDS)

Respective agents Blocked IP files,

AutoBlock.py - BlockedIPs (DDoS Mitigation - CSF)
IDSAutoBlock.py - IDSBlockedIPs (Threats Mitigation - IDS)

The purpose of IPinput.sh file is to get information from users about the virtual machines public and private IP address respectively with CIDR format (A.B.C.D/32).This script will run during the course of installation and prompt the user to enter the details.This configuration will lead to the configuration change at /etc/suricata/suricata.yaml file like the below,

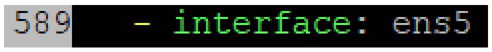


```
    HOME_NET:  "[13.233.253.58/32,172.31.4.149/32]"
```

```
read -p 'Enter First IP :' ip1
read -p 'Enter Second IP (Enter to skip):' ip2

set homeNet = ''

if [[ ! -z "$ip1" ]] &&  [[ ! -z "$ip2" ]];
then
   homeNet="[$ip1,$ip2]"
elif [[ ! -z "$ip1"  ]]; then
   homeNet="[$ip1]"
elif [[ ! -z "$ip2" ]]; then
   homeNet="[$ip2]"
fi

echo $homeNet
```

In /etc/suricata/suricata.yaml line no.589 is a variable from ifconfig output to apply the correct interface ID.We do the below to achieve the same, available within installfirewallids.sh

```
ifconfig | grep flags=4163 | colrm 5 > 1.txt
sed -i "589s/ens5/$(cat 1.txt)/" /etc/suricata/suricata.yaml
```



```
589   - interface: ens5
```

The purpose of clearcsfdeny.sh file is to remove AutoBlocker.log,BlockedIPs,IDSAutoBlocker.log,IDSBlockedIPs, clear iptables,refresh iptables,clear /var/log/syslog,clear /var/log/suricata/fast.log and restart csf,lfd,suricata,ddosblock.service and idsblock.service.

```
systemctl stop ddosblock.service
systemctl stop idsblock.service
systemctl stop csf ; systemctl stop lfd
systemctl stop suricata
rm -f AutoBlocker.log  BlockedIPs IDSAutoBlocker.log  IDSBlockedIPs
rm -f /etc/csf/csf.deny
wget -O /etc/csf/csf.deny http://15.207.104.57/csf.deny
:> /var/log/syslog
:> /var/log/suricata/fast.log
csf -r
systemctl restart csf ; systemctl restart lfd
systemctl restart suricata
systemctl restart ddosblock.service
systemctl restart idsblock.service
```

Suppose if we are doing any modifications in CSF and IDS configurations or any changes in application or maintenance , we can run beforechanges.sh (before maintenance) and afterchanges.sh (after maintenance) script.

beforechanges.sh has stopfirewallids.sh script
startfirewallids.sh has startfirewallids.sh script

The stopfirewallids.sh script does stop services by removing AutoBlocker.log,BlockedIPs,IDSAutoBlocker.log,IDSBlockedIPs, clear iptables, /var/log/syslog,clear /var/log/suricata/fast.log and stop csf,lfd,suricata,ddosblock.service and idsblock.service.

```
systemctl stop csf ; systemctl stop lfd
systemctl stop suricata
systemctl stop ddosblock.service
systemctl stop idsblock.service
rm -f /etc/csf/csf.deny
wget -O /etc/csf/csf.deny http://15.207.104.57/csf.deny
systemctl restart csf ; systemctl restart lfd
systemctl stop suricata
systemctl stop csf
systemctl stop lfd
systemctl stop ddosblock.service
systemctl stop idsblock.service
```

```
:> /var/log/syslog
:> /var/log/suricata/fast.log
rm -f AutoBlocker.log  BlockedIPs IDSAutoBlocker.log  IDSBlockedIPs
```

The startfirewallids.sh script does start services by starting csf,lfd,suricata,ddosblock.service and idsblock.service.

```
systemctl start suricata
systemctl start csf
systemctl start lfd
systemctl start ddosblock.service
systemctl start idsblock.service
```

The restartfirewallids.sh script does a very similar function equal to clearcsfdeny.sh.

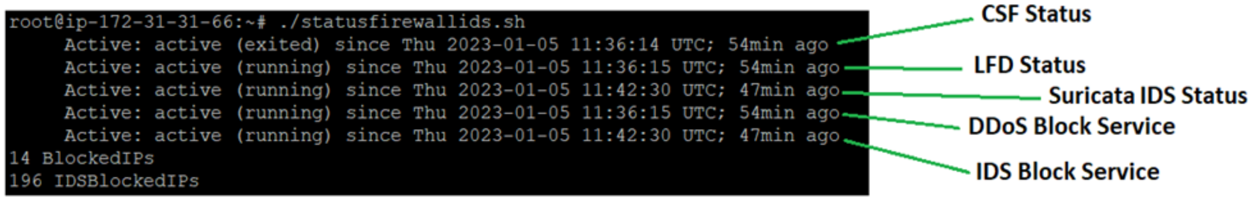
```
systemctl stop csf ; systemctl stop lfd
systemctl stop suricata
systemctl stop ddosblock.service
systemctl stop idsblock.service
:> /var/log/syslog
:> /var/log/suricata/fast.log
rm -f AutoBlocker.log  BlockedIPs IDSAutoBlocker.log  IDSBlockedIPs
rm -f /etc/csf/csf.deny
wget -O /etc/csf/csf.deny http://15.207.104.57/csf.deny
systemctl restart csf ; systemctl restart lfd
systemctl restart suricata
systemctl restart ddosblock.service
systemctl restart idsblock.service
csf -r
```

The startids.sh script does starting/enabling service of suricata and idsblock.service.This script runs automatically during the installation process.

```
systemctl start suricata
systemctl enable suricata
systemctl status suricata
systemctl restart suricata
systemctl start idsblock.service
systemctl enable idsblock.service
systemctl status idsblock.service
```

The statusfirewallids.sh script shows the status of csf,lfd, suricata, idsblock.service and ddosblock.service also number of IPs blocked under the python agents of CSF and IDS like the below,

```
systemctl status csf | grep Active
systemctl status lfd | grep Active
systemctl status suricata | grep Active
systemctl status ddosblock.service | grep Active
systemctl status idsblock.service | grep Active
wc -l BlockedIPs
wc -l IDSBlockedIPs
```



```
root@ip-172-31-31-66:~# ./statusfirewallids.sh
     Active: active (exited) since Thu 2023-01-05 11:36:14 UTC; 54min ago
     Active: active (running) since Thu 2023-01-05 11:36:15 UTC; 54min ago
     Active: active (running) since Thu 2023-01-05 11:42:30 UTC; 47min ago
     Active: active (running) since Thu 2023-01-05 11:36:15 UTC; 54min ago
     Active: active (running) since Thu 2023-01-05 11:42:30 UTC; 47min ago
14 BlockedIPs
196 IDSBlockedIPs
```

The suricatarulesupdate.sh script runs at 5 AM daily.It updates IDS engine rules from the below mentioned threat intelligence,it is updated in **crontab**

Networkforensic
Snort
Abuse
Etnetera
Feodotracker
Urlhaus
Emergingthreats
Quadrantsec

```
rm -f /var/lib/suricata/rules/NF-local.rules
rm -f /var/lib/suricata/rules/NF-SCADA.rules
rm -f /var/lib/suricata/rules/NF-Scanners.rules
rm -f /var/lib/suricata/rules/snort3-community.rules
rm -f /var/lib/suricata/rules/etn_aggressive.rules
rm -f /var/lib/suricata/rules/sslblacklist_tls_cert.rules
rm -f /var/lib/suricata/rules/feodotracker_aggressive.rules
rm -f /var/lib/suricata/rules/urlhausabusech.rules
wget -O /var/lib/suricata/rules/NF-local.zip https://networkforensic.dk/SNORT/NF-local.zip
wget -O /var/lib/suricata/rules/NF-SCADA.zip https://networkforensic.dk/SNORT/NF-SCADA.zip
wget -O /var/lib/suricata/rules/NF-Scanners.zip
https://networkforensic.dk/SNORT/NF-Scanners.zip
wget -O /var/lib/suricata/rules/etn_aggressive.rules
https://security.etnetera.cz/feeds/etn_aggressive.rules
```

```
wget -O /var/lib/suricata/rules/sslblacklist_tls_cert.rules
https://sslbl.abuse.ch/blacklist/sslblacklist_tls_cert.rules
wget -O /var/lib/suricata/rules/snort3-community-rules.tar.gz
https://www.snort.org/downloads/community/snort3-community-rules.tar.gz
wget -O /var/lib/suricata/rules/feodotracker_aggressive.rules
https://feodotracker.abuse.ch/downloads/feodotracker_aggressive.rules
wget -O /var/lib/suricata/rules/urlhausabusech.rules
https://urlhaus.abuse.ch/downloads/suricata-ids/
cd /var/lib/suricata/rules/
unzip NF-SCADA.zip
unzip NF-Scanners.zip
unzip NF-local.zip
tar -xf snort3-community-rules.tar.gz
mv /var/lib/suricata/rules/snort3-community-rules/snort3-community.rules /var/lib/suricata/rules/
rm -f NF-SCADA.zip NF-Scanners.zip NF-local.zip
rm -f /var/lib/suricata/rules/snort3-community-rules.tar.gz
rm -rf /var/lib/suricata/rules/snort3-community-rules
sudo suricata-update
systemctl restart suricata
```

```
0 5 * * * /usr/bin/bash /root/suricatarulesupdate.sh
```

The clearcsfdeny.sh script runs daily at 6 AM, thereby optimizing the **iptables** for seamless service,it is updated in **crontab**

```
0 6 * * * /usr/bin/bash /root/clearcsfdeny.sh
```

The uninstall.sh script has uninstallfirewallids.sh. The uninstallfirewallids.sh does the uninstall process of CSF, IDS, ddosblock.service and idsblock.service.

Deployment engineering

We have a cloud hosted NGINX server 15.207.104.57 with Ubuntu.

All the deployment shell/python scripts are hosted on this server.

AutoBlock.py
IDSAutoBlock.py

IPinput.sh
clearcsfdeny.sh
startfirewallids.sh
stopfirewallids.sh
restartfirewallids.sh
startids.sh
statusfirewallids.sh

afterchanges.sh
beforechanges.sh

suricatarulesupdate.sh

uninstall.sh

uninstallfirewallids.sh

For testing purposes on a local server with Ubuntu 18 and above , you can issue the below command for automated deployment.

```
wget -O /root/install.sh http://15.207.104.57/install.sh
```

You make it fully executable with chmod +x install.sh

When you run ./install.sh , it will download installfirewallids.sh and uninstall.sh.
Both files will be made fully executable.It will run ./installfirewallids.sh

When ./installfirewallids.sh runs it will download csf.tgz and install csf.

Then the following will be **downloaded** from the cloud server 15.207.104.57,

CSF Config files :
```
wget -O /etc/csf/csf.conf http://15.207.104.57/csf.conf
wget -O /etc/csf/csf.deny http://15.207.104.57/csf.deny
wget -O /etc/csf/csf.blocklists http://15.207.104.57/csf.blocklists
wget -O /etc/csf/csf.dirwatch http://15.207.104.57/csf.dirwatch
```

Python Agents Scripts
```
wget -O /root/AutoBlock.py http://15.207.104.57/AutoBlock.py
wget -O /root/IDSAutoBlock.py http://15.207.104.57/IDSAutoBlock.py
```

Python Agents as Backend Linux Services
```
wget -O /etc/systemd/system/ddosblock.service http://15.207.104.57/ddosblock.service
wget -O /etc/systemd/system/idsblock.service http://15.207.104.57/idsblock.service
```

Configuration and Maintenance Shell scripts
```
wget -O /root/IPinput.sh http://15.207.104.57/IPinput.sh
wget -O /root/startids.sh http://15.207.104.57/startids.sh
wget -O /root/stopfirewallids.sh http://15.207.104.57/stopfirewallids.sh
wget -O /root/startfirewallids.sh http://15.207.104.57/startfirewallids.sh
wget -O /root/restartfirewallids.sh http://15.207.104.57/restartfirewallids.sh
wget -O /root/statusfirewallids.sh http://15.207.104.57/statusfirewallids.sh
wget -O /root/clearcsfdeny.sh http://15.207.104.57/clearcsfdeny.sh
wget -O /root/beforechanges.sh http://15.207.104.57/beforechanges.sh
wget -O /root/afterchanges.sh http://15.207.104.57/afterchanges.sh
wget -O /root/suricatarulesupdate.sh http://15.207.104.57/suricatarulesupdate.sh
```

Uninstall Shell Script
```
wget -O /root/uninstallfirewallids.sh http://15.207.104.57/uninstallfirewallids.sh
```

All the script files / service files become fully executable

```
chmod +x /root/AutoBlock.py
chmod +x /root/IDSAutoBlock.py
chmod +x /root/startids.sh
chmod +x /etc/systemd/system/ddosblock.service
chmod +x /etc/systemd/system/idsblock.service
chmod +x /root/IPinput.sh
chmod +x /root/stopfirewallids.sh
chmod +x /root/startfirewallids.sh
chmod +x /root/restartfirewallids.sh
chmod +x /root/statusfirewallids.sh
chmod +x /root/clearcsfdeny.sh
chmod +x /root/beforechanges.sh
chmod +x /root/afterchanges.sh
chmod +x /root/suricatarulesupdate.sh
chmod +x /root/uninstallfirewallids.sh
```

Cronjob will be added for the below shell scripts,

```
echo "0 6 * * * /usr/bin/bash /root/clearcsfdeny.sh" >> /var/spool/cron/crontabs/root
echo "0 5 * * * /usr/bin/bash /root/suricatarulesupdate.sh" >> /var/spool/cron/crontabs/root
```

CSF,LFD,ddosblock.service gets started.

```
systemctl start csf ; systemctl start lfd
systemctl enable csf ; systemctl enable lfd
systemctl status --no-pager csf ; systemctl status --no-pager lfd
systemctl start ddosblock.service
systemctl enable ddosblock.service
systemctl status ddosblock.service
```

Suricata IDS Installs , listening interface gets auto configured , and IDS rules get downloaded and applied to the IDS engine for threat detection.

Suricata Installs

```
sudo apt install unzip -y
sudo apt install net-tools -y
sudo add-apt-repository ppa:oisf/suricata-stable -y
sudo apt update -y
sudo apt install suricata -y
sudo suricata-update
```

Suricata Listening interface capture

```
ifconfig | grep flags=4163 | colrm 5 > 1.txt
```

Suricata rules get downloaded

```
wget -O /var/lib/suricata/rules/Rules.zip http://15.207.104.57/Rules.zip
```

Suricata Listening interface applies to Suricata Configuration

```
sed -i "589s/ens5/$(cat 1.txt)/" /etc/suricata/suricata.yaml
```

Suricata rules getting loaded into IDS engine

```
cd /var/lib/suricata/rules/
unzip Rules.zip
rm -f Rules.zip
```

Get Virtual machines public and private IPs to Suricata Configuration

```
./IPinput.sh > 2.txt
printf '"%s"\n' $(cat 2.txt) > 2.txt
sed -i "1 a \HOME_NET:  $(cat 2.txt)" 2.txt
sed -i '1d' 2.txt
sed -i '18d' /etc/suricata/suricata.yaml
sed -i "17 a \    $(cat 2.txt)" /etc/suricata/suricata.yaml
```

Start Suricata and its block service idsblock.service

./startids.sh

```
systemctl start suricata
systemctl enable suricata
systemctl status suricata
systemctl restart suricata
systemctl start idsblock.service
systemctl enable idsblock.service
systemctl status idsblock.service
```

Finally Suricata and idsblock.service gets started.

The Deployment is done!!!