Home-based Threat Intelligence

written by Mert SARICA | 1 October 2019

Those of you who read my articles will recall that in my post titled “Escape from Imprisonment“, I enthusiastically discussed the advantages of using a router packed with security features. As I mentioned in the article, I had started using the dnscrypt-proxy tool to encrypt DNS traffic (Dns over HTTPS – DoH).

In today’s world where thermostats are getting smarter, smart TVs are equipped with cameras, and electric water heaters and irons are being turned into spy devices, insecure Internet of Things (IoT) devices connected to our home network pose a great risk to our security and privacy. As I was thinking about how to detect systems in our home network that have been hacked, infected, or contain backdoors, I remembered that thanks to the dnscrypt-proxy tool, I could also record DNS requests made by all systems, devices, and gadgets connected to the home network.

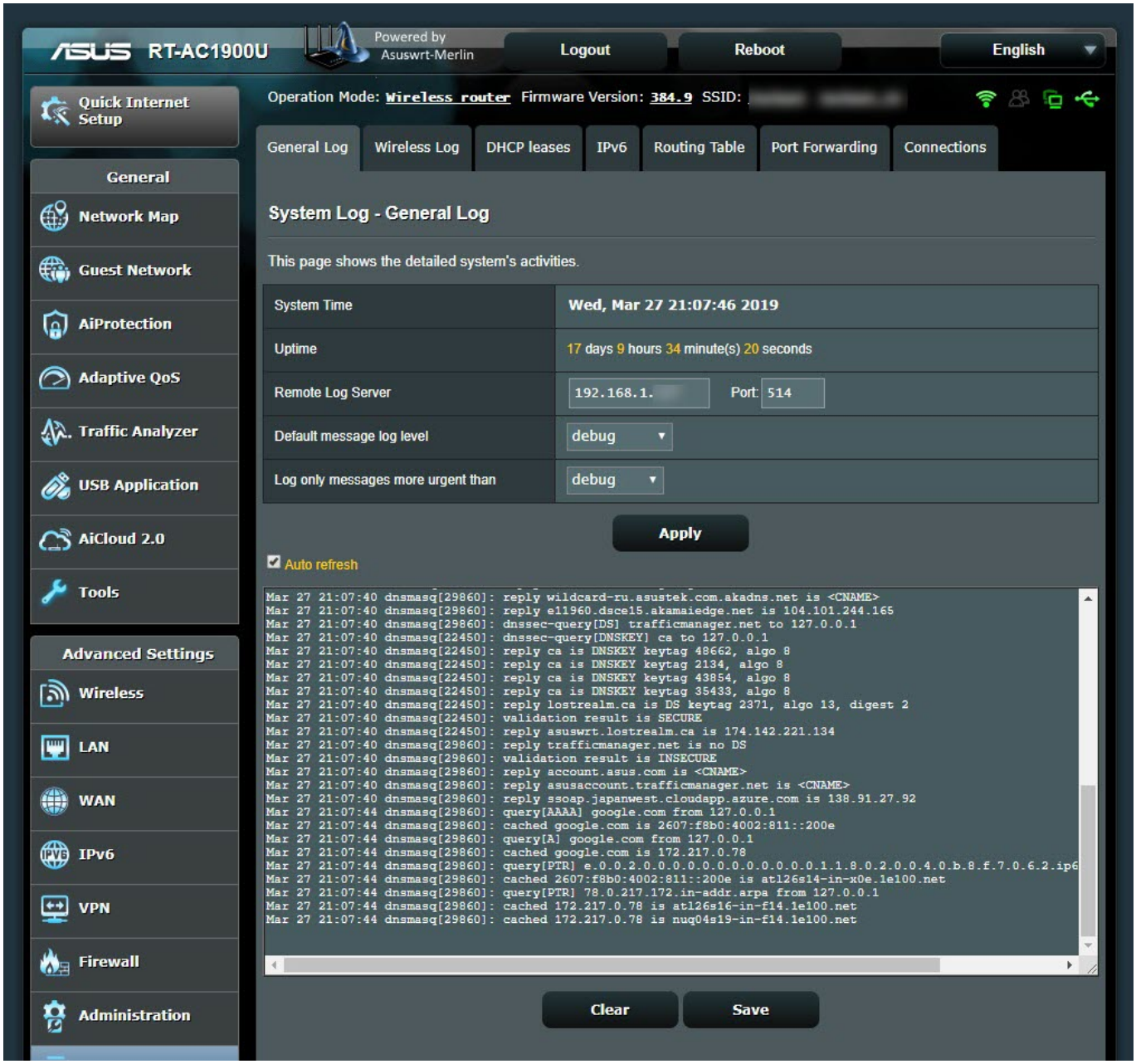
At the point where I could record DNS requests, I realized I could detect malicious systems in my home network by querying the domain names and IP addresses found in these DNS requests through cyber threat intelligence services like Open Threat Exchange (OTX) and Critical Stack. Without wasting time, I started thinking about the list of requirements to bring this idea to life.

First, I decided to install the syslog-ng package on the Ubuntu operating system running on my Mini-PC, which is always at hand and always comes to my aid in such situations. After installing the package, I configured it to record incoming DNS requests in the date.log file under the /var/log/dns-sys/sender’s-ip-address directory and saved this configuration in the /etc/syslog-ng/conf.d/dns-sys.conf file.

```
root@ubuntu:/etc/syslog-ng/conf.d# ls
dns-sys.conf
root@ubuntu:/etc/syslog-ng/conf.d# cat dns-sys.conf
########################################################
options {
        create_dirs(yes);
        perm(0640);
        dir_perm(0750);
};


########################################################
source s_net {
                tcp(ip(0.0.0.0) port(514));
                udp(ip(0.0.0.0) port(514));
};

########################################################
destination d_host-specific {
        file("/var/log/dns-sys/$HOST/$DAY-$MONTH-$YEAR.log");
};

filter f_cached { match("cached"); };                    # Filter regex keyword cached
filter f_query  { match("query");  };                    # Filter regex keyword query
filter f_reply  { match("reply");  };                    # Filter regex keyword reply

log {
        source(s_net);
        filter(f_cached);
        destination(d_host-specific);
};

log {
        source(s_net);
        filter(f_query);
        destination(d_host-specific);
};

log {
        source(s_net);
        filter(f_reply);
        destination(d_host-specific);
};
```

In the next step, to make the dnscrypt-proxy tool log DNS requests to the router's syslog, I added the line 'log-queries' to the /jffs/configs/dnsmasq.conf.add file. Then, to make the router display these requests on its syslog page, I set the 'Default message log level' and 'Log only messages more urgent than' values to 'debug', and to redirect these messages to the syslog-ng application running on Ubuntu, I defined the 'Remote Log Server' value as the IP address of Ubuntu.

```
mert@RT-AC1900U-6610:/jffs/configs# cat dnsmasq.conf.add
no-resolv
log-queries
server=127.0.0.1#65053
mert@RT-AC1900U-6610:/jffs/configs#
```



I started examining the syslog-ng records one by one and looking into which types of records I needed to focus on for threat intelligence. After learning that I could use the query[A], cached, and reply information in the records, I thought I could send these records to Security Onion, which integrates with OTX. After installing and running Security Onion's 16.04.5.6 operating system, I noticed that the logstash service (so-logstash) wasn't working at all. Despite my struggle, I was unsuccessful and started researching alternative methods.

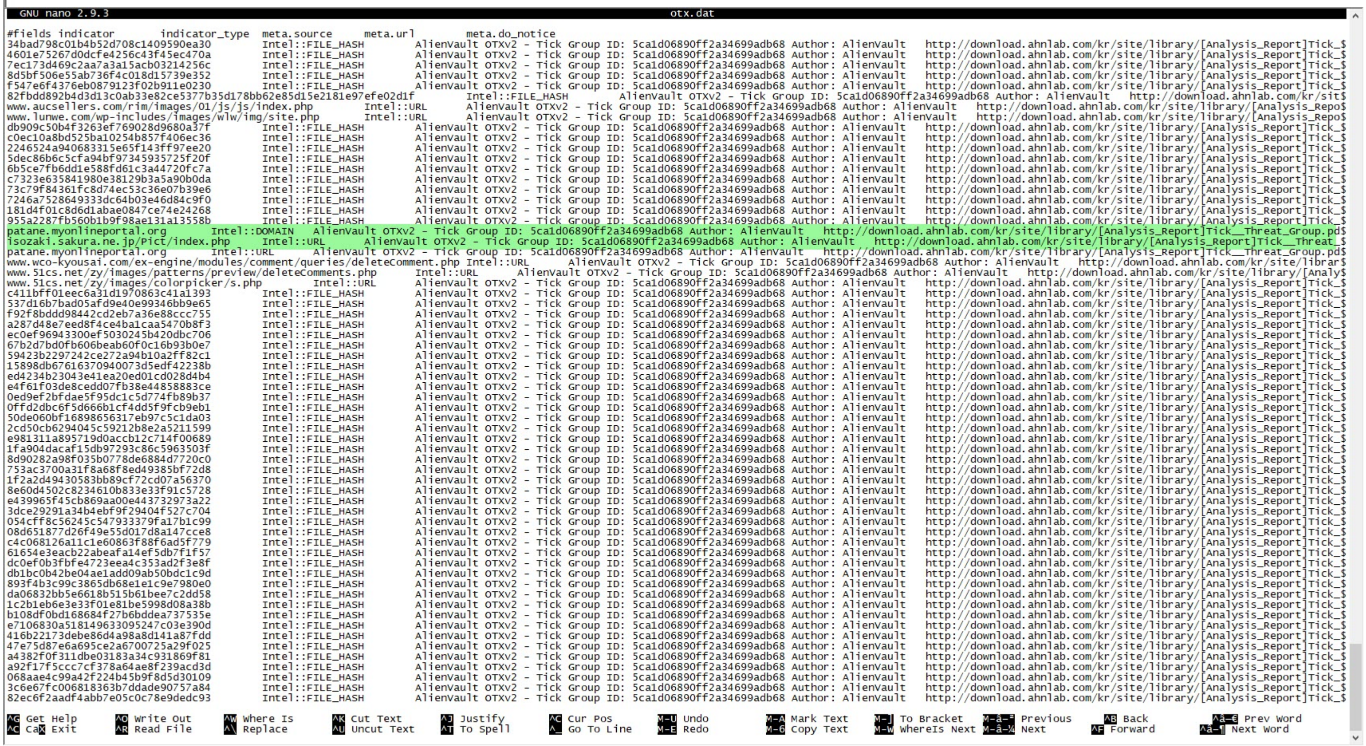
```
root@ubuntu:/etc/syslog-ng/conf.d# tail -n 20 /var/log/dns-sys/192.168.1.1/09-04-2019.log
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply partnerad.l.doubleclick.net is 74.125.21.156
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply partnerad.l.doubleclick.net is 74.125.21.157
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply partnerad.l.doubleclick.net is 74.125.21.154
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply partnerad.l.doubleclick.net is 74.125.21.155
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: query[A] s.w.org from 192.168.1.225
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: query[A] widget.engageya.com from 192.168.1.225
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply s.w.org is 192.0.77.48
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply widget.engageya.com is <CNAME>
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply widget-engageya.edgekey.net is <CNAME>
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply e15247.dscg.akamaiedge.net is 104.96.141.105
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: query[A] www.googletagservices.com from 192.168.1.225
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply www.googletagservices.com is <CNAME>
Apr  9 21:09:25 192.168.1.1 dnsmasq[29860]: reply pagead46.l.doubleclick.net is 172.217.3.226
Apr  9 21:09:27 192.168.1.1 dnsmasq[29860]: query[A] gatr.hit.gemius.pl from 192.168.1.225
Apr  9 21:09:27 192.168.1.1 dnsmasq[29860]: reply gatr.hit.gemius.pl is 5.135.121.144
Apr  9 21:09:27 192.168.1.1 dnsmasq[29860]: reply gatr.hit.gemius.pl is 37.59.195.0
Apr  9 21:09:27 192.168.1.1 dnsmasq[29860]: reply gatr.hit.gemius.pl is 37.187.168.211
Apr  9 21:09:27 192.168.1.1 dnsmasq[29860]: reply gatr.hit.gemius.pl is 149.202.193.219
Apr  9 21:09:27 192.168.1.1 dnsmasq[29860]: reply gatr.hit.gemius.pl is 149.202.204.241
Apr  9 21:09:27 192.168.1.1 dnsmasq[29860]: reply gatr.hit.gemius.pl is 188.165.145.88
root@ubuntu:/etc/syslog-ng/conf.d# cat /var/log/dns-sys/192.168.1.1/08-04-2019.log | cut -d " " -f 7 | sort | uniq -i
cached
dnssec-query[DNSKEY]
dnssec-query[DS]
forwarded
query[A]
query[AAAA]
query[PTR]
query[SRV]
reply
root@ubuntu:/etc/syslog-ng/conf.d#
```

When I shared a message on Twitter about needing to install ELK, I received messages suggesting that I could use cloud and ready-made ELK systems. As I was considering whether to install ELK on Ubuntu or use a cloud system, I learned that Logstash, which has Grok filter and Translate filter plugins, was tailor-made for this job.

I started modifying the securityonion-otx script file, which was developed for Security Onion – OTX integration, according to my needs. I set the bro-otx file to save threat intelligence information from OTX to the /etc/logstash/ls-otx/otx.dat file every hour. I also configured the OTX.py file to extract only domain name information from the malicious URL and DOMAIN entries in the otx.dat file and save it as the

/etc/logstash/translate/OTX.yaml file to be read by the Translate filter at the 5th minute of every hour.

```
root@ubuntu:/etc/cron.d# cat bro-otx
# /etc/cron.d/bro-otx
#
# crontab entry to manage Bro OTX pulse updates

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

0 * * * * root python /etc/logstash/ls-otx/bro-otx.py >> /var/log/bro-otx.log 2>&1
root@ubuntu:/etc/cron.d# cat ls-otx
# /etc/cron.d/bro-otx
#
# crontab entry to create Logstash dictionary from OTX file

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

*/5 */1 * * * root python /etc/logstash/ls-otx/OTX.py >> /var/log/ls-otx.log 2>&1
root@ubuntu:/etc/cron.d#
```

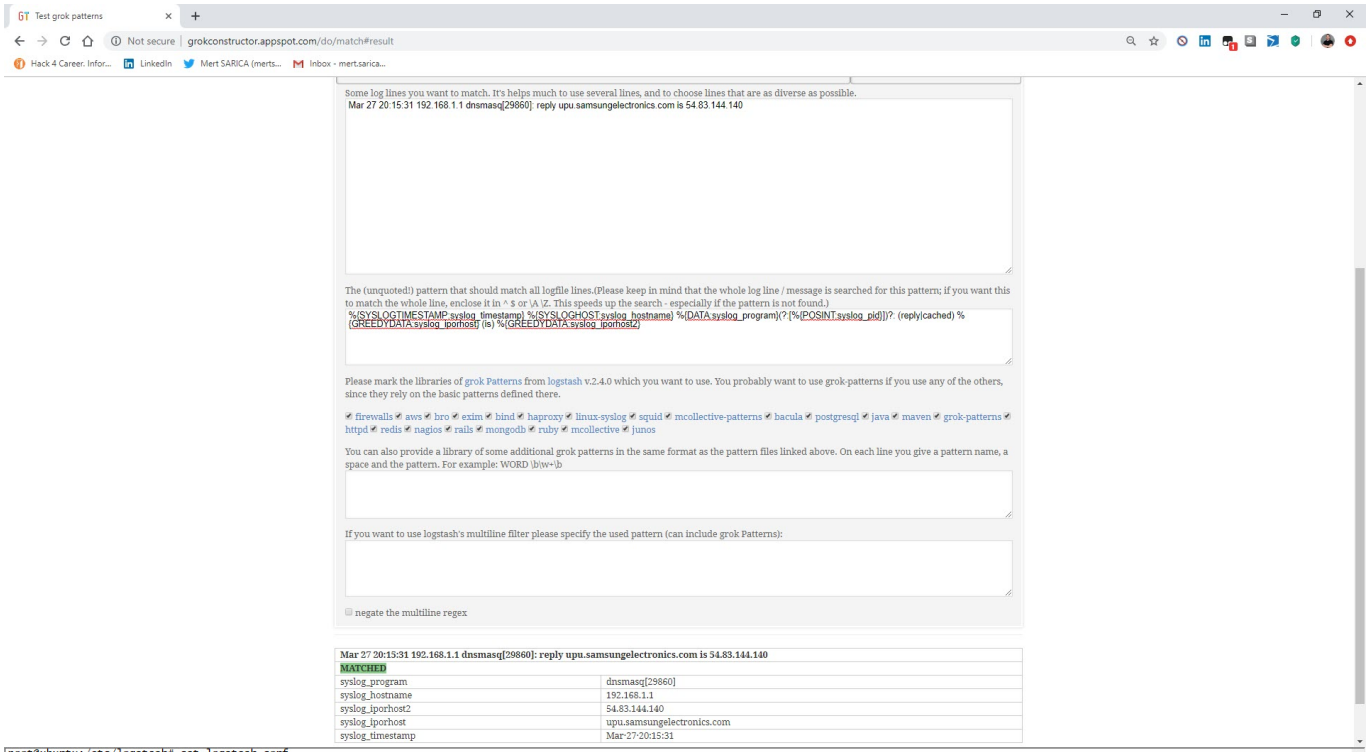


```
root@ubuntu:/etc/logstash/ls-otx# cat OTX.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# OTX to Logstash Dictionary Script
# Author: Mert SARICA
# E-mail: mert [ . ] sarica [ @ ] gmail [ . ] com
# URL: https://www.mertsarica.com
#
# Credit: https://raw.githubusercontent.com/TravisFSmith/MyBroElk/master/maliciousIP.py

import re
debug = 0

def writeYAML():
        fname = "/etc/logstash/ls-otx/otx.dat"
        yamlFile = open('/etc/logstash/translate/OTX.yaml','w')
        with open(fname) as html:
                cti = []
                for line in html.readlines():
                        line = re.sub('\\r|\\n','',line)
                        if line.find("Intel::DOMAIN") >= 0:
                                try:
                                        line = line.split("\t")[0]
                                        if line not in cti:
                                                cti.append(line)
                                                if debug:
                                                        print line.split("\t")[0]
                                                yamlFile.write("\"" + line + "\": \"YES\"" + "\n")
                                except:
                                        continue

                        if line.find("Intel::URL") >= 0:
                                try:
                                        line = line.split("\t")[0]
                                        line = line.split("/")[0]
                                except:
                                        line = line.split("\t")[0]

                                try:
                                        line = line.split(":")[0]
                                        if line not in cti:
                                                cti.append(line)
                                                if debug:
                                                        print line
                                                yamlFile.write("\"" + line + "\": \"YES\"" + "\n")
                                except:
                                        if line not in cti:
                                                cti.append(line)
                                                if debug:
                                                        print line
                                                yamlFile.write("\"" + line + "\": \"YES\"" + "\n")

                yamlFile.close()

if __name__=="__main__":
        writeYAML()
root@ubuntu:/etc/logstash/ls-otx#
```

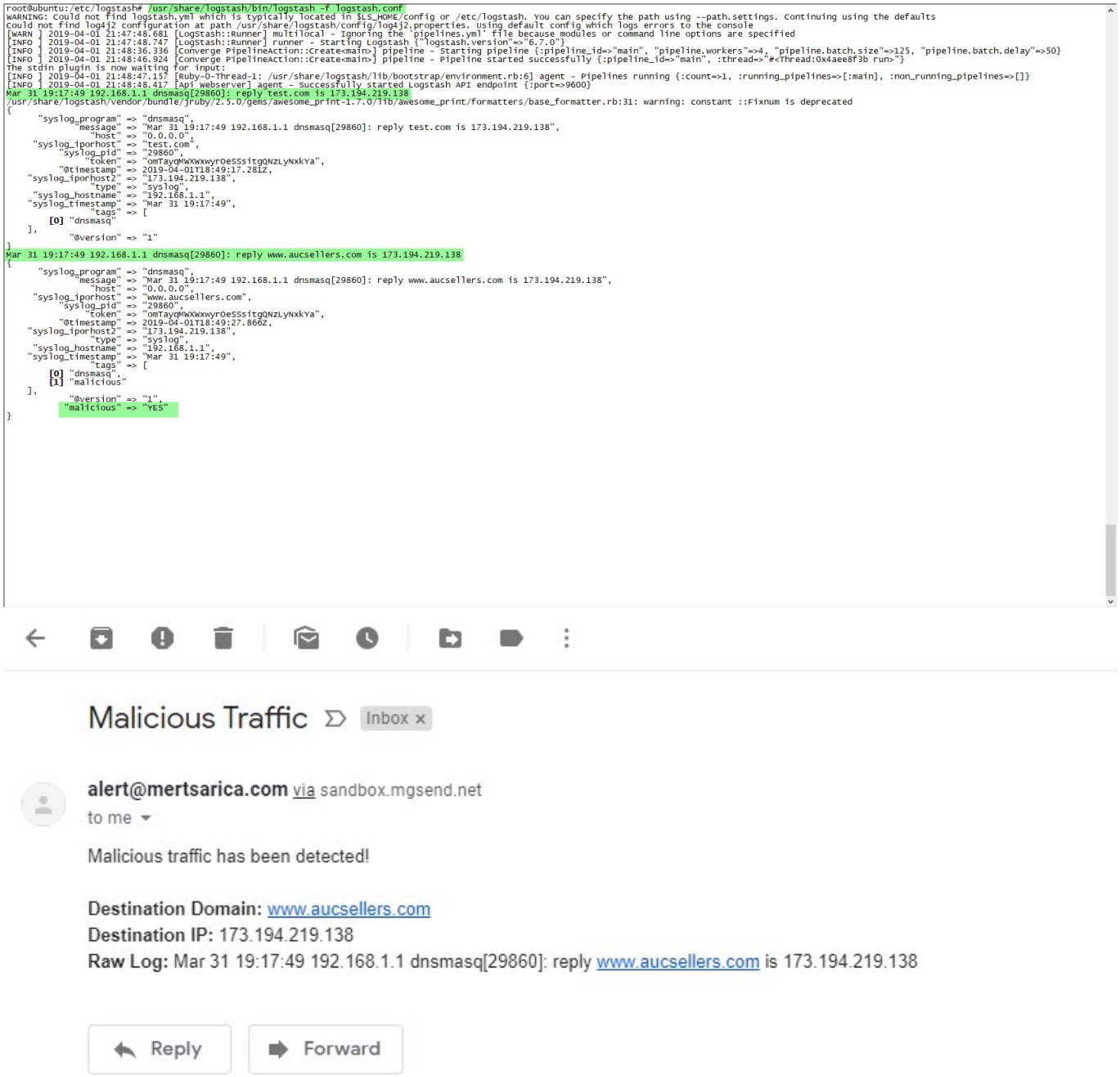
```
root@ubuntu:/etc/logstash/translate# ls
OTX.yaml
root@ubuntu:/etc/logstash/translate# head -n 10 OTX.yaml
"www.aucsellers.com": "YES"
"www.lunwe.com": "YES"
"patane.myonlineportal.org": "YES"
"isozaki.sakura.ne.jp": "YES"
"www.wco-kyousai.com": "YES"
"www.51cs.net": "YES"
"www6.intarnetservice.com": "YES"
"www.webmailerservices.com": "YES"
"go-trust.webmailerservices.com": "YES"
"www.adobeservice.net": "YES"
root@ubuntu:/etc/logstash/translate#
```

In the configuration file of Logstash (logstash.conf), I defined the rules to read DNS records logged by syslog-ng with the Grok filter and to send an alert via email if any of the IP addresses or domain names in these records match with those in the OTX.yaml file using the Translate filter. Then I restarted Logstash and made an nslookup for the address www[.]aucsellers[.]com listed in the OTX.yaml file. With this, the alert was

successfully generated and sent to me by email, and I had successfully implemented the home-based threat intelligence service. :)




```
root@ubuntu:/etc/logstash# cat logstash.conf
input {
  # stdin { type => syslog }
  file {
    path           => "/var/log/dns-sys/192.168.1.1/*.log"
    start_position => "beginning"
  }
}

filter {
  grok {
    match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: (reply|cached) %{GREEDYDATA:syslog_iporhost} (is) %{GREEDYDATA
:syslog_iporhost2}" }
    add_tag => "dnsmasq"
  }
  grok {
    match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: (query\[A\]) %{GREEDYDATA:syslog_iporhost} (from) %{GREEDYDATA
:syslog_queryfrom}" }
    add_tag => "dnsmasq"
  }
  translate {
    field => "syslog_iporhost"
    destination => "malicious"
    dictionary_path => '/etc/logstash/translate/OTX.yaml'
    add_tag => "malicious"
  }
  translate {
    field => "syslog_iporhost2"
    destination => "malicious"
    dictionary_path => '/etc/logstash/translate/OTX.yaml'
    add_tag => "malicious"
  }
  mutate {
    remove_tag => ["_grokparsefailure"]
  }
  if "dnsmasq" not in [tags] {
    drop { }
  }
}

output {
  stdout {
    codec => rubydebug
  }
  if [malicious] == "YES" and [syslog_iporhost2] {
   email {
        address => "127.0.0.1"
        from => "alert@mertsarica.com"
        htmlbody => "Malicious traffic has been detected!<br/><br/>
                     <b>Destination Domain: </b>%{syslog_iporhost}<br/>
                     <b>Destination IP: </b>%{syslog_iporhost2}<br/>
                     <b>Raw Log: </b>%{message}"
        port => 25
        subject => "Malicious Traffic"
        to => "mert.sarica@gmail.com"
        use_tls => false
   }
  } else if [malicious] == "YES" and [syslog_queryfrom] {
   email {
        address => "127.0.0.1"
        from => "alert@mertsarica.com"
        htmlbody => "Malicious traffic has been detected!<br/><br/>
                     <b>Source IP: </b>%{syslog_queryfrom}<br/>
                     <b>Destination IP or Domain: </b>%{syslog_iporhost}<br/>
                     <b>Raw Log: </b>%{message}"
        port => 25
        subject => "Malicious Traffic"
        to => "mert.sarica@gmail.com"
        use_tls => false
   }
```



Hope to see you in the following articles.