Global Information Assurance Certification Paper

Demystifying Malware Traffic

GIAC (GCIH) Gold Certification

Author: Sourabh Saxena, sourabhsaxena25@gmail.com
Advisor: Barbara Filkins
Accepted: August 28th, 2016

Abstract

In today's world, adversaries use established techniques, innovative and intricate methods for cyber-crimes and to infiltrate firms or an individual's system. Usage of Malware is one of those approaches. Malware not only creates an inlet for attacks, but it also turns systems into "zombies" and "bots" forcing them to obey commands and perform activities as per the whims and fancies of the adversary. Thus, attacks like data theft, mail relay, access to confidential/restricted area, Distributed Denial-of-Service (DDoS) can easily be launched against not just the infected system but against other systems and environments as well by utilizing these zombies, bots, and botnets. Attackers not only obfuscate the code but can encrypt payloads as well as malware's traffic simultaneously, using approaches like mutation and polymorphism making their detection difficult not just for antiviruses, but even for firewalls, IDS and IPS, Incident Handlers, and Forensic teams. Organizations, having learned from past mistakes, have also shifted their approach from simple defense mechanisms such as antiviruses, IDS and IPS to aggressive strategies like DNS Sinkhole and Live Traffic Analysis. These strategies not only help in the identification and removal of malware but also in understanding the actual impact, blocking of malicious activities and identification of adversaries.

In this paper, the following two preventative approaches are discussed in detail: DNS Sink-Hole and Live Traffic Analysis. The paper explores how to set up example environments for each approach using open source tools. Topics around the demystification of malware traffic using these methods include: automating the updating of blacklisted domains for the DNS sinkhole approach, stopping malware from contacting and receiving commands from its command and control (C2) center, and analyzing encrypted malware traffic.

1. Introduction

With the increase in security incidents and breaches, in addition to various compliance, rules, and standards to be met, organizations are spending funds to incorporate security measures wherever possible (Monteith, 2010). These measures vary from implementing Intrusion Detection System (IDS), Intrusion Prevention Systems (IPS), firewalls, and antivirus (AV) to having various open-source as well as commercial security products and modules in place to handle the traditional attacks and the detection of well-known malware (Akamai, 2016).

Attacks like malicious scans, network exploits, exploitation of loosely configured devices, unpatched services availed by regular users/servers/services or attacks like SQL Injections, and XSS in web applications fall under traditional attacks. These attacks are prevalent because tools and scripts are either already available or can be built for exploiting the security weaknesses. The purpose of these tools/scripts is to send the specifically malformed packets or the request to the target machine to trigger the vulnerability. Though there are several limitations in such traditional attacks, one of them is that the generated packets are sent from the attacker's (or any other) machine (i.e., it will be an inbound traffic for the targeted machine/infrastructure.) Moreover, these packets need to be in the exact format that can trigger the vulnerability when they reach their destination. Thus, whether such packets are encrypted or in any other format during the transition to avoid detection, they have to be transformed back to the original format while hitting the weaknesses. Due to such limitations, these attacks can be detected, controlled and mitigated through perimeter or host-based IDS/IPS, Firewall, security patches, policies, hardening of system, devices, applications and services (Weber, 2012).

These are some of the reasons why attackers are now coming up with non-traditional techniques and have better coordination among themselves for the development and usage of stealth malware using freely and easily available frameworks to achieve the desired results (Grimes, 2012). These malware not only initiate the outbound traffic to bypass Firewall rules but also use innovative or non-traditional

techniques like code obfuscation, encryption, polymorphism, metamorphism, connection to its command-and-control center (also known as C2 and CC) over SSL to go under the radar (Yim, 2010). Moreover, factors like Social Engineering attacks, Zero Day exploits, weak policies and procedures, business exceptions for vulnerability and any changes in the secure environment too are very helpful and are major reasons for the success of malware attacks. Please refer to Appendix – C to understand in detail why such malware are difficult to detect.

Malware attacks are launched against computers, servers, devices, mobile devices and SCADA infrastructure. These attacks convert these resources into zombies, botnets and to achieve attacks like mail relays, access to confidential/restricted areas, Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), data theft, unauthorized scans, online ransom, chain-attacks and other traditional attacks (LaRiza, 2015).

As the malware attacks are unavoidable, teams like IT, Incident Handlers, Forensics, and Malware Analysis should be prepared with methods to stop the malicious activity of malware, to understand the actual impact caused by such malware and the strategies adopted by attackers. In this paper, the following two methods are explored to help analyze and mitigate non-traditional attacks: DNS Sinkholes and Live Traffic Analysis.

The paper starts with what these environments are and how each can be used in a normal malware attack, followed with a demonstration of each in a lab environment and finally, evaluates how effective they can be, given the use of best practices and limitations of the approach.

2. DNS Sinkhole and Traffic Analysis

This section explores what these approaches are, why and when they are needed, and the benefits of these environments.

Sourabh Saxena, sourabhsaxena25@gmail.com

2.1. What Is A DNS Sinkhole

Before understanding the DNS Sinkhole, let's see how the Firewall rules and DNS services are utilized by the malware. It is known that Firewalls (network and host-based) are mostly configured to allow all outbound traffic and only a few specific forms of inbound traffic (Myers, 2013). Malware takes advantage of such policies, apart from other aspects. Therefore, instead of an attacker connecting to the malware or the packets sent by an attacker to the target machine using traditional attacking mechanisms, it is the malware that initiates outbound traffic to connect to its C2 center. This is because any packet sent or any connection initiated by the attacker would be inbound traffic to the infected machine and can easily be detected. Moreover, because users may not have static IPs mapped to their infected systems, outbound traffic from the infected machine is a preferred mechanism in these non-traditional attacks. To initiate the outbound traffic, the malware first queries the IP address of the domain registered for the C2 center using the DNS server and once the IP address is available, it establishes a connection and extracts the commands or (additional) payloads.

A few reasons to configure a malware for domain lookup and not the IP address are as follows:

a. To mimic the human behavior of accessing a site using a URL and not the IP address directly.

b. Use of Dynamic DNS Registration services to avoid IP detection. Dynamic registration helps attackers when they restart their C2 centers if their IPs are blocked and also to avoid attacks on their own server.

c. To obtain the "new" IP address assigned to their C2 center after every specific time period.

Use of domains by malware is the reason a DNS service comes into the picture, which ultimately helps researchers to configure the DNS Sinkhole. This is an environment wherein the DNS service running on a particular DNS server resolves DNS

queries. However, it does not provide the actual IP address of the queried domain if the query is for a blacklisted domain registered for any C2 center. In such a case, DNS is configured to provide the IP address of a server that can be used by a malware analyst or other teams as a sinkhole to isolate the malware from obtaining the IP address of the C2 center, resulting in no connection with the C2 center. Thus, a DNS Sinkhole is used to redirect malware traffic from the infected system to the sinkhole by providing a false IP address to the malware when it tries to retrieve the C2 center's IP address. Once the malware is isolated from contacting its C2 center, all other traffic to and from the system can also be analyzed. For example, when a user is not logged into the system, the traffic is observed to be less; thus, if any system is found to generate more traffic than expected or required, an analyst and other teams can look into the services to analyze the suspicious activity/service (Bambenek, 2015).

A DNS Sinkhole sometimes is also referred to as a DNS Redirection Environment because of its core functionality of redirecting malicious traffic from its original destination to the one specified by the DNS server. This new destination is known as a Sinkhole. All the services, servers and device mechanisms used in a Sinkhole environment can be owned, configured and administered by researcher(s) (Bambenek, 2015). A DNS Sinkhole is also called Blackhole Server, Internet Sinkhole, DNS Redirection Server or Sinkhole Server.

A DNS Sinkhole can be categorized based on the factors like tools used, capabilities, commercial and noncommercial. The DNS Sinkhole can be commercial or configured using a combination of freeware and open-source tools (Virgillito, 2014).

The simplest DNS Sinkhole can provide **127.0.0.1** as a resolved IP address. Advanced sinkholes can provide an IP address of a server that can be configured with other tools, like IDS/IPS and sniffers, for further traffic analysis.

Security best practices should be in place while setting up the DNS server. Please refer to Appendix – A for some of the security best practices applicable to a DNS service.

Sourabh Saxena, sourabhsaxena25@gmail.com

2.2. What Is Live Traffic Analysis And Its Environment

The real-time analysis of the traffic generated by a malware to get the commands from and to send the information back to its Command-and-Control (C2) center is known as Live Traffic Analysis. Analysis can be performed even if the traffic is over Secure-Socket Layer (SSL).

The environment for live traffic analysis is capable of sending malware traffic to the C2 and forwarding commands back to the malware. Thus, this environment works as a proxy or router and do not disrupt a malware's connection to its C2. This gives a strategic way to study the traffic to and from an infected machine without interfering the flow or the functionality of the malware. The environment can be used for all the traffic, whether sent as cleartext or over SSL (Nieto, 2015).

The purpose of analyzing (demystifying) the malware traffic in real time, whether in cleartext or encrypted, is to understand the impact due to an infection, prepare a remediation and removal plan for the infection, identify weaknesses in current infrastructure and plan to overcome these weaknesses by implementing the security best approaches.

Reasons where this environment can be of help include:

- Organizations may want to pinpoint the attacker and the activities.
- Analysts would like to assess the impact due to all the commands issued by attackers.
- To understand if only one machine is infected or if there are other machines as well.
- To confirm if data from other machines are also being accessed by the attacker, i.e., to confirm a chain-attack.
- Security measures like IDS/IPS may not be helpful if the traffic is over SSL. The only way to assess the impact is to have a real-time analysis,

especially important when the activities/traffic or the command is time-based, i.e. after a period of time of time, which can be a few seconds or a few months. Time interval gives less opportunity to perform an offline analysis.

This also covers the situation where a malware may use an IP address rather than domains, overcoming a few limitations of DNS Sinkhole.

2.3. Why Are These Needed

At times, organizations or the teams, like Incident Handler and Malware Analyst, would need to block the communication of malware for various reasons like security policies, to stop further impact that may include information thefts and other attack variations. At other times, organizations/teams may opt for real-time analysis of the traffic. The reasons for real-time analysis have been previously mentioned. Moreover, if the traffic is over SSL, then it becomes very difficult for organizations to perform the analysis.

2.4. When Are These Needed

DNS Sinkhole and Live Traffic analysis are helpful during phases when a system is already infected due to malware, and there is a need to understand the malware behavior in a better way by analyzing the traffic generated to and from the infected machine, assess impact due to malware, and identify adversaries.

2.5. Benefits

Benefits of these environments include configuration and management under the control of users because of the usage of freeware and open sources described in this paper. These environments can be cost effective as well. Traffic can be blocked or can be analyzed in real time even when the traffic is over SSL, helping in the demystification of malware traffic, which includes encrypted traffic. Traffic can be manipulated to send only

false information to attackers. This helps in maintaining the connection of malware with its Command and Control center and, at the same time, teams can continue their analysis.

The next section presents a generic scenario where a system gets infected due to malware, to understand how the traffic flows to and from the infected system and other network components. This is followed by an illustration of how DNS Sinkhole and Live Traffic Analysis can be used to help/aid in these scenarios.

3. What Happens When a System Gets Infected By Malware

Here's the scenario that explains what happens when a system gets infected and how the traffic flows between an infected machine, the DNS server, and the Command and Control (C2) center.

Scenario: Malware, once it is delivered to the victim, infects the system after execution. It sends a request to the DNS server, either internal or external, to obtain the IP address of the C2 center. The DNS server sends a response with the IP address of the C2 center. Then, the malware sends a request to the C2 center, which is now controlled by an attacker. It sends malicious commands to the infected system.

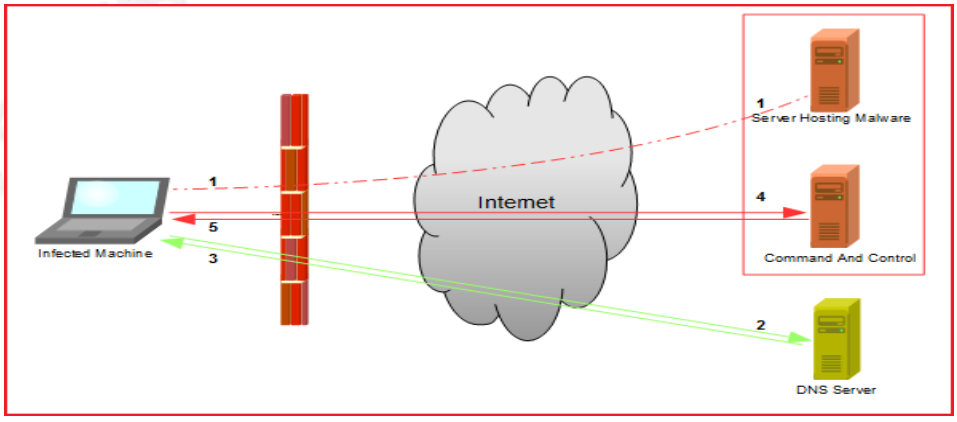


Figure 1: Traffic flow due to malware

The following steps refer to Figure 1 above:

1. Malware gets delivered to the victim, thus infecting the system.
2. DNS query, generated by malware, is sent to a DNS server configured in the system.
3. The IP address is sent as a response to the infected system.
4. Malware establishes a connection with its C2 center after obtaining the IP address. It can be hosted on the C2 center as well.
5. Commands are transferred to the malware in the response.

Appendix – E contains instructions on how to set up the above environment and a detailed analysis of the flow of data to and from the victim's machine, web server (hosting the virus), DNS server, virus connecting to the C2 center, request to and response from the C2 center.

A more complex environment and the flow of malware traffic over SSL will be discussed in the Live Traffic Analysis section. Moreover, in a real scenario, **virus.exe** and **backdoor.exe** can be spread using e-mails, web pages and by other means.

The next two sections will explore these aspects of DNS Sinkhole and Live Traffic Analysis in detail.

4. DNS Sink Hole Analysis

Figure 2 is a screenshot of the above scenario with the phases of downloading, execution and DNS query. **X.X.37.220** is the server from where **virus.exe** is downloaded. **X.X.37.64** is the DNS server with which virus performs the DNS query for its C2. In the following example, **example.com** (93.184.216.34) has been used as C2:

Sourabh Saxena, sourabhsaxena25@gmail.com

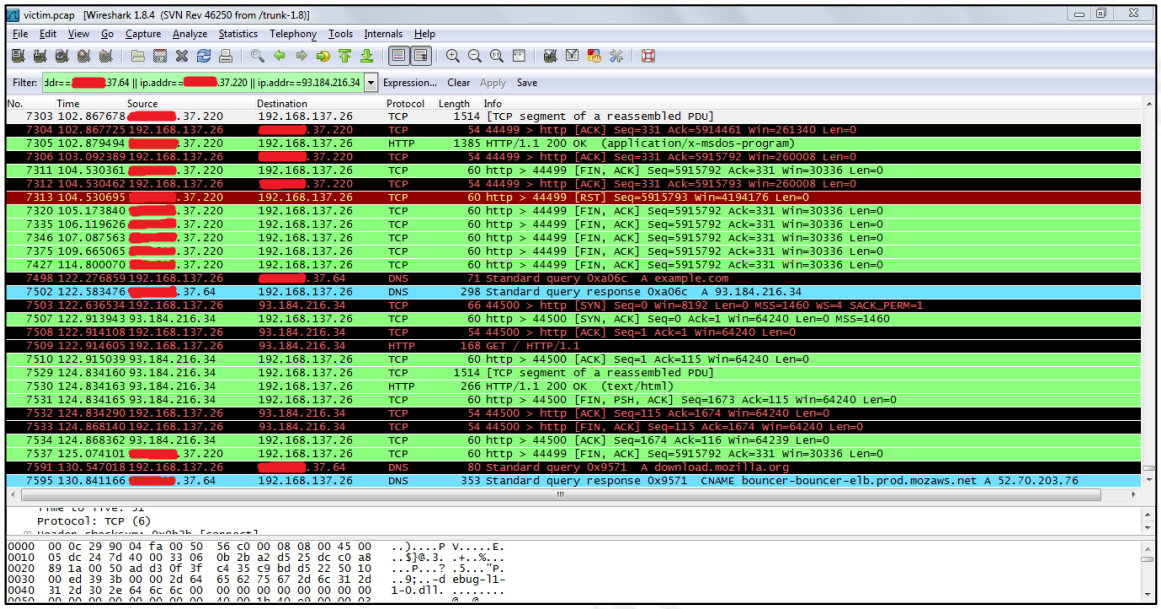


Figure 2: Flow of malware traffic, including DNS Query

Let us consider that researchers can block the DNS query initiated by malware. If so, the malware won't be able to get the IP address of its C2 and thus, won't be able to connect to its C2. Hence, this will help in blocking malware activities that can be achieved using DNS Sinkhole. A detailed flow and setup for the above scenario are noted in Appendix – E.

4.1. Components Used To Set Up A SinkHole Environment

We will now set up the environment for DNS Sinkhole which would be used to achieve the above idea. The automation part would also be looked into for updating the blacklisted domains for DNS Sinkhole.

The components required for building the DNS Sinkhole environment are:

1. Ubuntu: Required for running a BIND service.
2. BIND9: This acts as the DNS Service.
3. Windows 7: This acts like a victim's machine.
4. Kali with Apache2: This hosts **virus.exe**.

4.2. How to Set Up A DNS SinkHole

A simple sinkhole wherein the resolution of a domain will be the local host (**127.0.0.1**) can be set up by the following steps:

1. Boot the Ubuntu system and then install **BIND9** (Albitz & Liu, 2016) using the following command (Refer to Appendix - E, Figure 9):

   ```
   sudo apt-get install bind9
   ```

2. Edit **named.conf** to include the **blockeddomains.conf** file. This file contains the location of the zone file for the blocked domains.

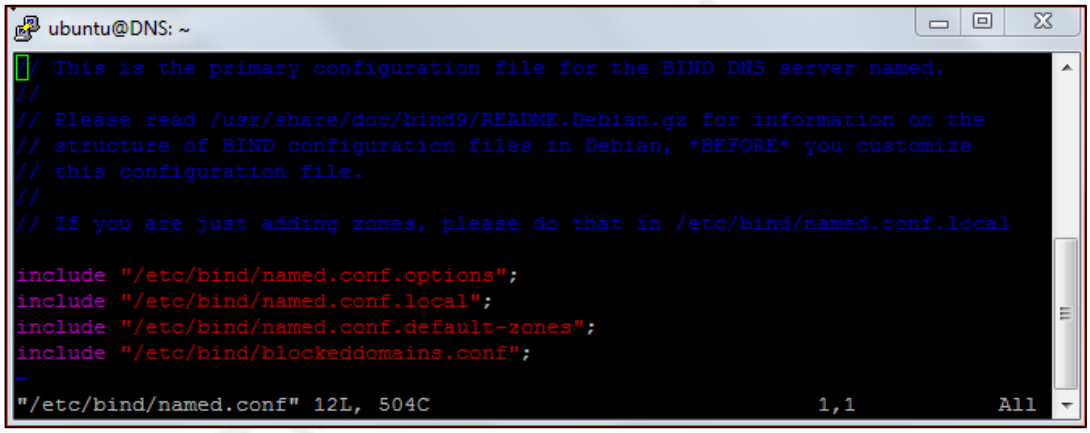


Figure 3: Defining the location of zone file in named.conf

3. The **named.conf.options** file is configured to use BIND as a simple forwarder, as shown in the screenshot below. A DNS zone can also be added here as **RPZ** (commented in the screenshot below for **google.com**).

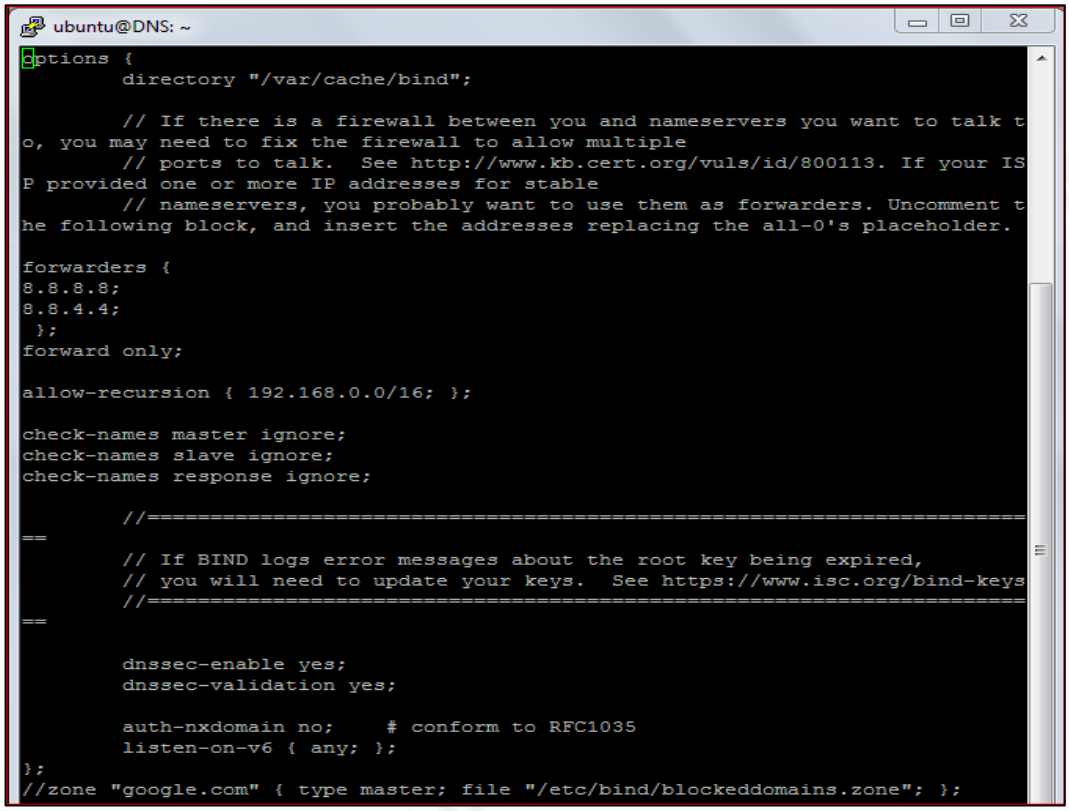


Figure 4: BIND service as a forwarder

4. Create a Config file with the location of the zone file of the blocked domains.

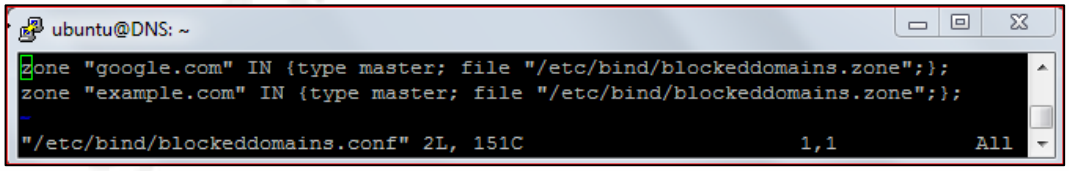


Figure 5: Configuration file with the location of the zone file

5. Create the zone file and add the details, as shown in the screenshot below. These details can be different depending on the requirements of users or organizations.

Note that the last line, i.e. *** IN A 127.0.0.1**, has been added to cover subdomains as well as www for blacklisted domains. **192.168.137.137** is the IP of the server where the BIND service is running.

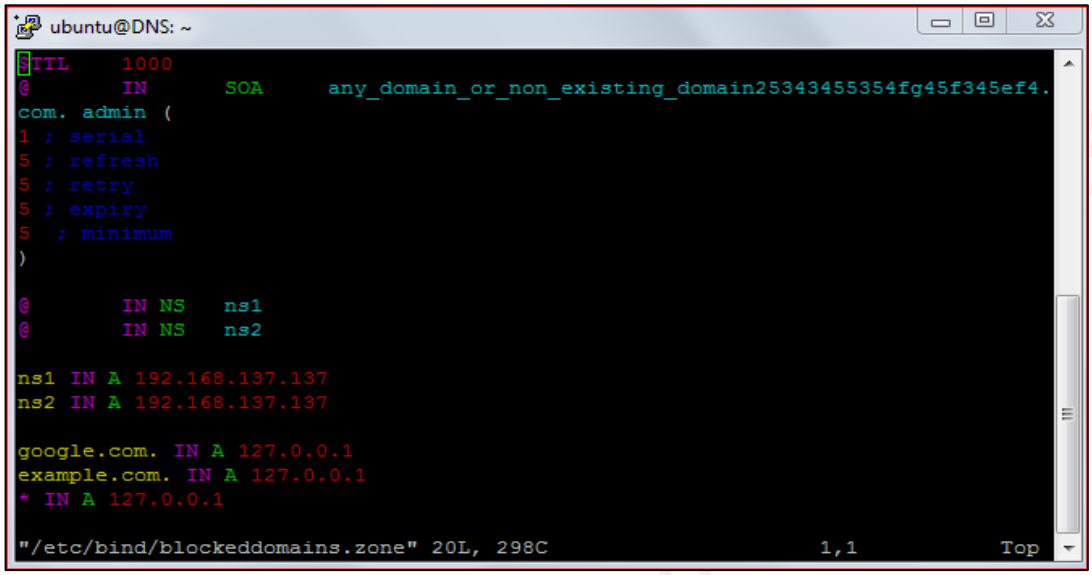


Figure 6: Zone file to resolve specific domains as 127.0.0.1

Now, restart the BIND service using the following command:

service bind9 restart

6. When a resolution occurs for **google.com** by a system using **192.168.137.137** as a DNS Server, the DNS service returns **127.0.0.1** as the IP address as shown in the following screenshot.

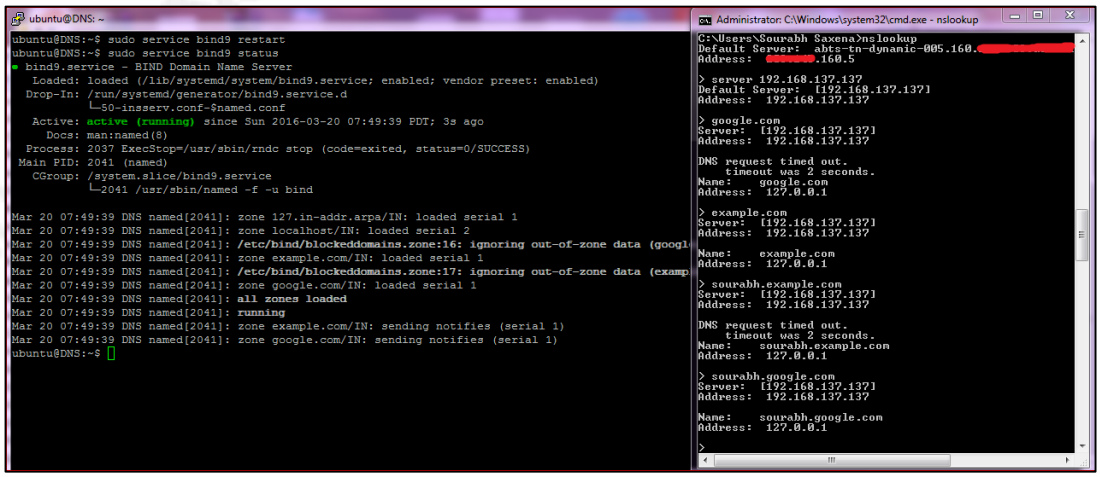


Figure 7: Resolution of specific domain as 127.0.0.1

7. For the advanced version, the zone file can include the IP address of the server, which contains all the software, monitoring and analysis tools for further analysis. The following screenshot (Figure 8) depicts that the requests to **example.com** and **sourabh.example.com** were redirected to a particular server (X.X.37.220), which can later be redirected to the tools (running on X.X.37.220) for malware analysis. Here, observe the IP address in the menu bar of the browser when **example.com** was accessed.

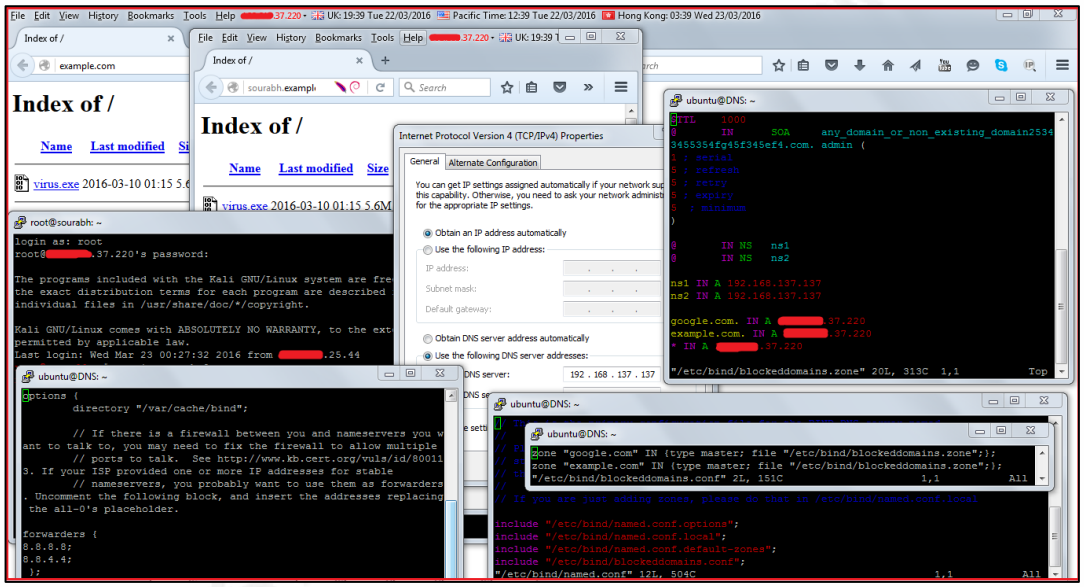


Figure 8: Configurations for X.X.37.220

4.3. How to Automate Blacklisting

The above scenario demonstrates how a DNS Sinkhole can be configured manually. However, the concern here is that there are thousands of blacklisted domains, and such lists frequently change, making it difficult to add these domains manually.

Here, Python (Sweigart, 2015) and other scripting languages can be used as an aid. A sample Python script, documented in Appendix – G, can help in automating the process. This sample script does the following:

1. Downloads the list of blocked domains from **malwaredomains.com**.

2. Creates a **blockeddomains.conf** file for all blocked domains.
3. Creates a **blockeddomains.zone** file for all blocked domains.
4. Saves both these files at the **etc/bind/** location.

The following screenshot shows the execution of the Python script:

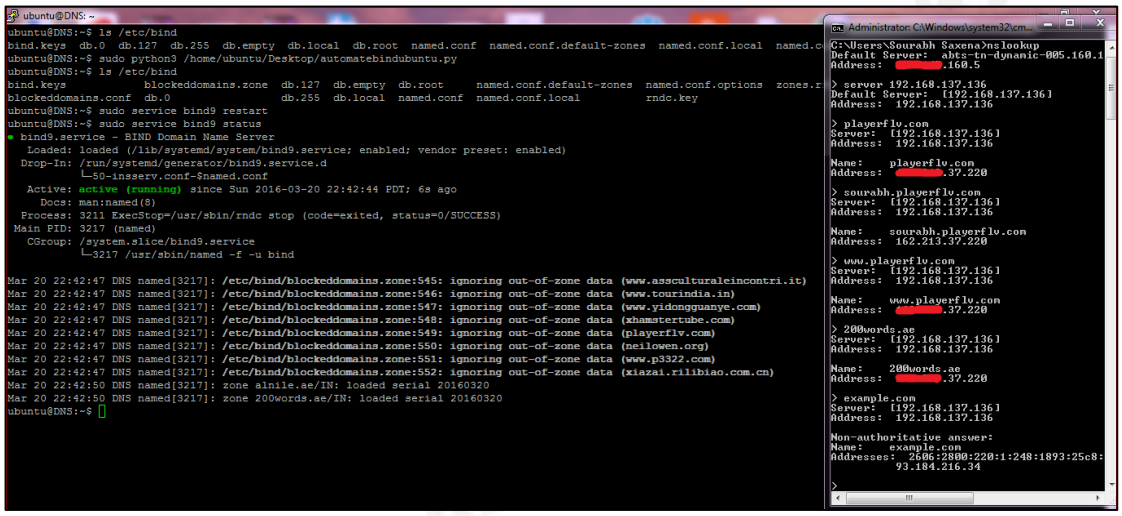


Figure 9: Resolution of specific domains to X.X.37.220

This Python script can be executed using a Shell script as shown below, apart from other commands:

```
#!/bin/sh
python3 /home/ubuntu/Desktop/automatebindubuntu.py
#Please change the path of above python script
#'rndc flush' command too can be used to avoid rndc related error on bind restart
/usr/sbin/service bind9 restart
#ensure the full path for 'service' command using 'which service' command
```

The following screenshot shows **automatebindubuntu.py** and **shell.sh** at the given path (i.e. within the Desktop directory) with the list of all files at **/etc/bind/** before

and after the execution of **shell.sh**. It also reflects the status of the BIND service after the execution of **shell.sh**.

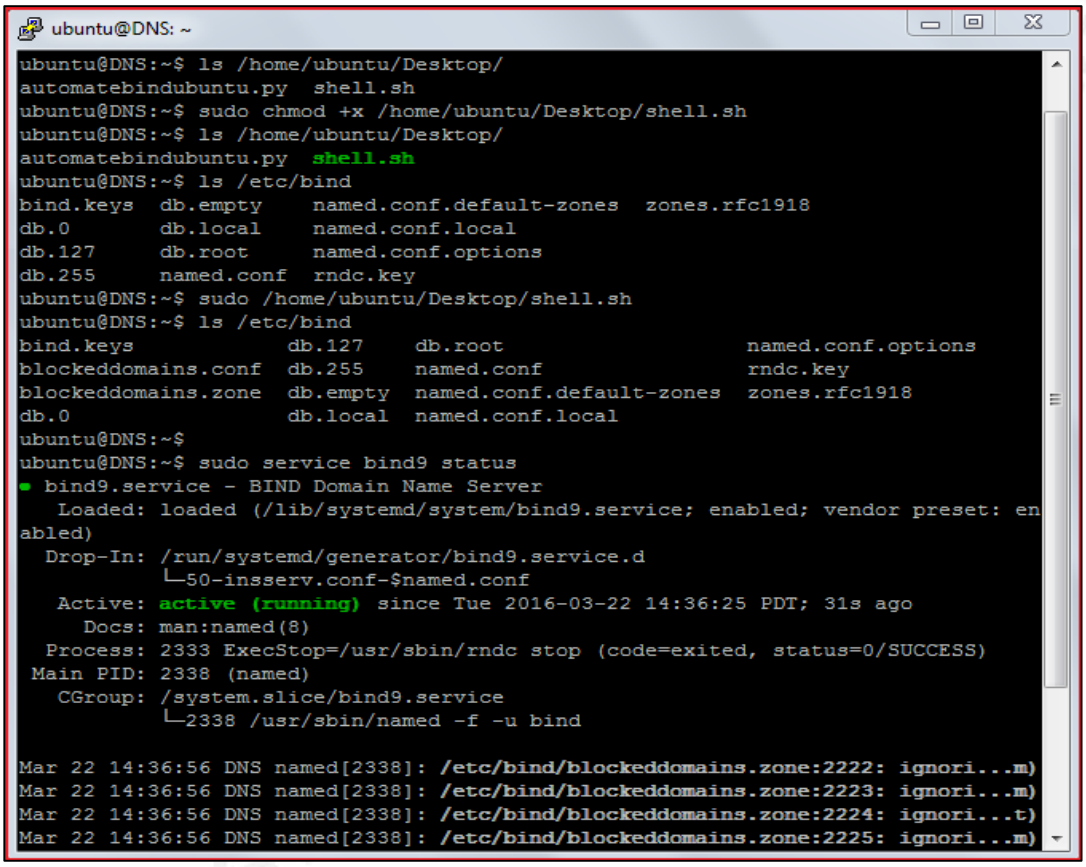


Figure 10: Execution of Shell script and BIND status

The above shell script can be scheduled to run on a monthly basis by first executing the following command: **sudo crontab -e**

Then, enter the following command as a cron job:

30 21 1 * * /home/ubuntu/shell.sh

This command runs the shell script at 09:30 P.M. on the first day of every month.

The following screenshots display when the job was scheduled using crontab to run daily at a given time and also, what the status was for the cron before and after the

specified time the job was scheduled to run for. The status of the BIND service after the execution of **shell.sh** through this scheduled job is shown below:

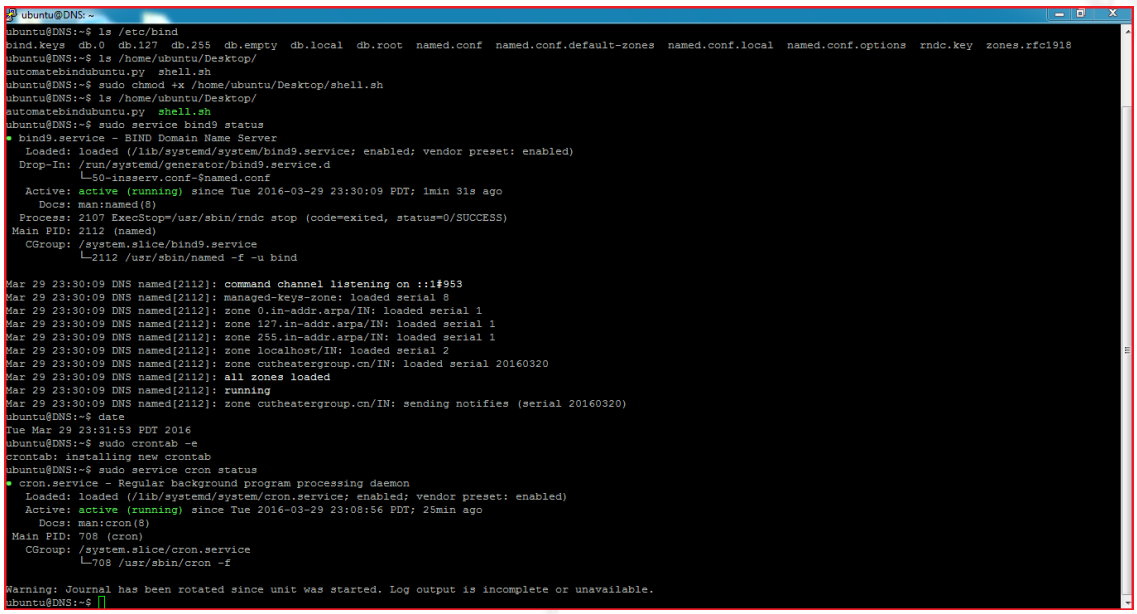


Figure 11: Status of the job

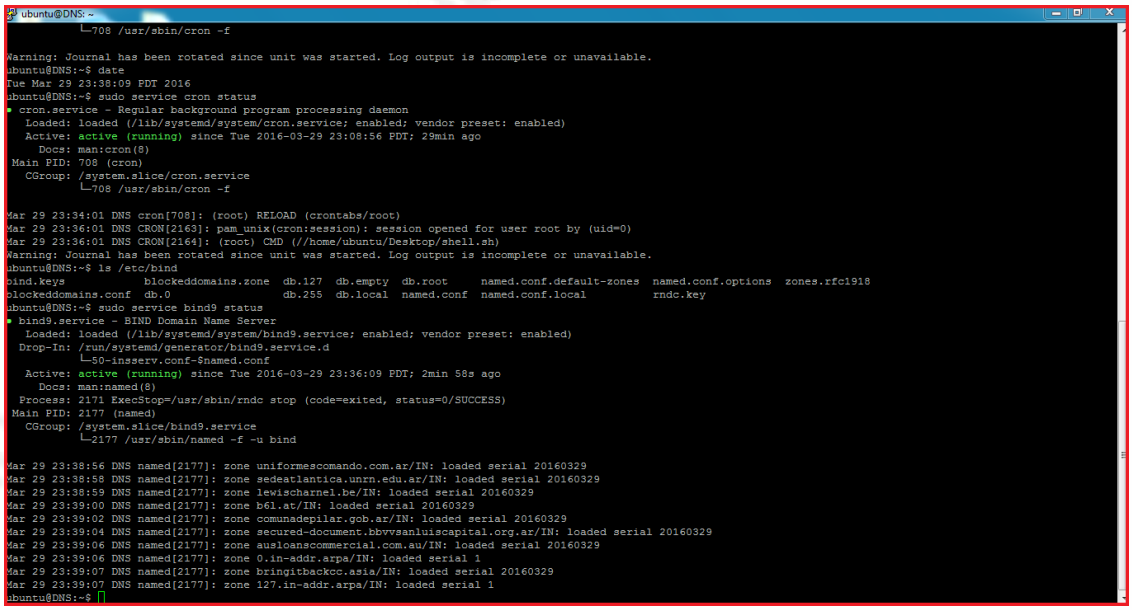


Figure 12: Status of BIND after a specific time specified in the cron job

In the above example, the domain list was fetched from

http://www.malwaredomains.com/. There are different sites that keep track of the

malicious domains and the IP addresses of the C2 center. These are listed in Appendix – D that provides resources from which the blacklisted domain names/IP addresses can be added in DNS.

4.4. Limitations of This Approach

The DNS Sinkhole approach has its limitations:

1. It is mainly used for identification and blocking of the traffic to the blacklisted domains rather than identification or removal of malware. Though, once the traffic and domain are identified and blocked, research teams can take actions like the removal of malware, cleaning of the environment and further investigation of the malware using standard security procedures to overcome this limitation.
2. It cannot identify and block the traffic of all malware that use IP addresses rather than domains. Blocking of such outbound requests at the firewall level can overcome this issue. The environment configured for Live Traffic is going to be used to overcome this.
3. It cannot identify malware that uses its own DNS service. Any DNS traffic that uses the DNS, other than that owned or authorized by the organization, should be monitored or blocked.
4. The administrator of the sinkhole needs to ensure security best practices for DNS servers, to avoid issues and attacks like short TTL, cache poisoning, and other DNS based attacks.

5. Live Traffic Analysis

DNS Sinkhole is good to block the malware activity if malware is using the domains of its C2. However, what if a system gets infected by malware that uses the IP address of C2. But what if it's using SSL as well? These will ultimately pose difficulty in

analyzing the traffic and is where an environment for live traffic analysis would be of help. Other factors highlighted in Section 2.2 also demonstrate the need for live traffic analysis. This environment will cover even few of the limitations highlighted for DNS Sinkhole.

Figure 13 provides a simple overview of environment for live analysis, infected machine, and the C2. Before moving forward let's look into a real scenario where the virus is using SSL, traffic for which has been explained briefly in below section. Please refer to Appendix – F for a detailed flow, screenshots, and directions on how to create the below-highlighted scenario.

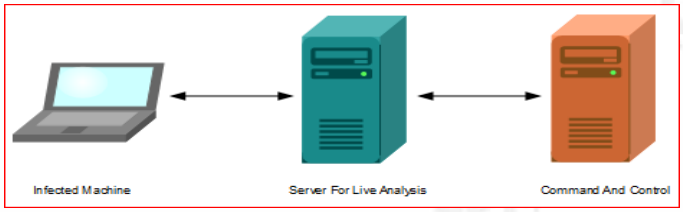


Figure 13: Environment for Live Traffic Analysis

5.1. SSL-Based Malware Traffic: Real Scenario

To demonstrate the encrypted traffic and the difficulty which an analyst faces, a virus.exe was hosted on a server (Kali) to be delivered to infect the victim's machine. This backdoor on execution connects to its C2 center (X.X.37.220) over SSL. A binary payload was created using the following command:

```
msfvenom -p windows/meterpreter/reverse_https LHOST=X.X.37.220 LPORT=443
-f exe > virus.exe
```

Screenshots of these malware interactions after decrypting the traffic in cleartext, using the following mentioned tools and environments, have been included in Appendix – F.

Following are the screenshots of the malware traffic over SSL to and from its CC:

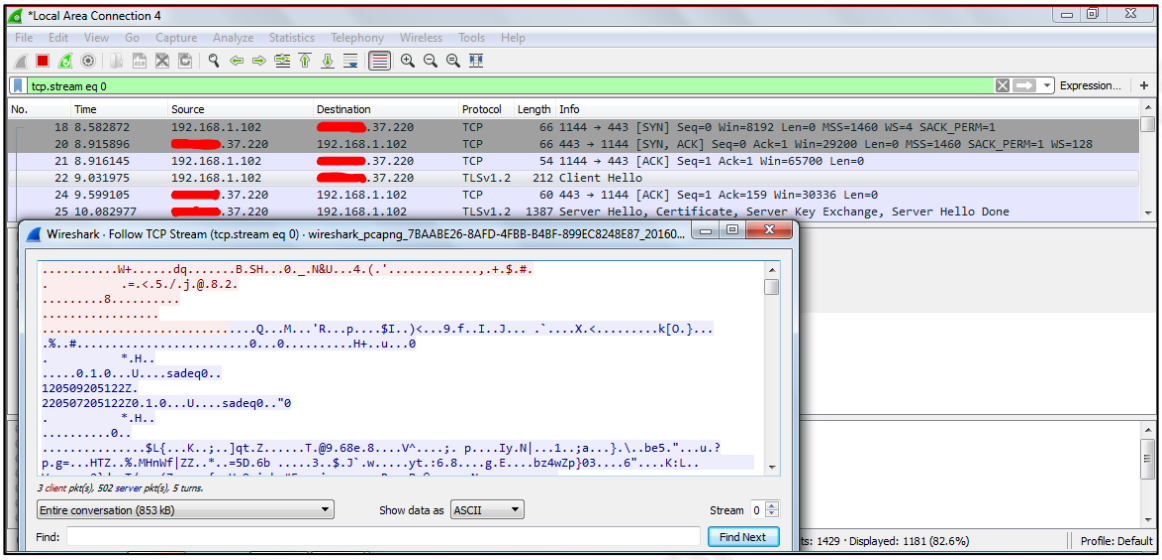


Figure 14: Traffic over SSL

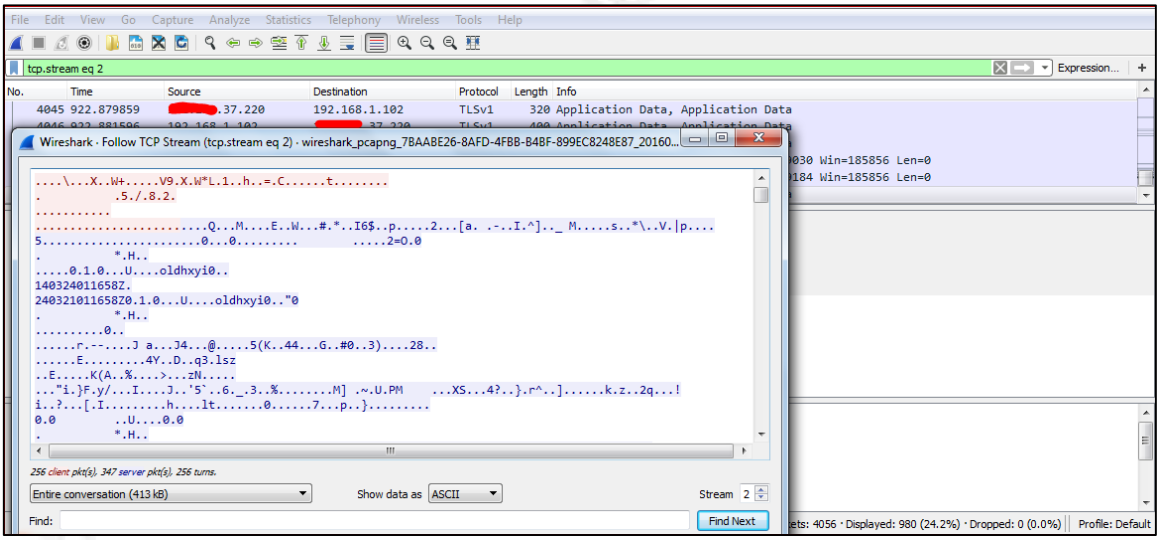


Figure 15: Traffic over SSL

From the encrypted flow above, it's clear that it won't be possible to understand the process involved like, what commands are being sent to the malware or what information is being transferred to C2.

We will now configure different environments which will help in overcoming these limitations. These environments will assist in redirecting the traffic from infected

machines to the server configured with tools for live traffic analysis. This traffic will then be redirected to C2. Responses from C2 will be sent back to the server, which will then be redirected back to the infected machine. Thus, there would be no discontinuity in the connection between the infected machine and its C2, while the researchers would be able to continue their observations and analysis (Refer to Figure 13 in above section.)

Let's first start with the cleartext communication and then look at the malware traffic over SSL.

5.2. Live traffic analysis of malware using cleartext protocol

5.2.1. Components required to set the environment

For analyzing the unencrypted traffic of malware, two components are required: an infected PC and a machine that can work as a router and has the tools, like TCPDump, Cuckoo, Yara, ssdeep, etc. for sniffing and further analysis. This router will be located in the middle of the traffic between the malware and its C2, and the tools will work as a proxy. (Note: Ubuntu, a virtual machine enabled with DHCP, will be used for this paper. Images like REMnux, SIFT, Kali and many more can also be used.)

5.2.2. How to set up an environment

To convert Ubuntu into a router, add two network interface cards to Ubuntu; one (ens34 - 192.168.111.X) for the isolated LAN where the infected machine would be placed and the other (ens33 - 192.168.1.X) for the WAN with internet access.

Enable IP Forwarding by uncommenting **net.ipv4.ip_forward=1** in the **/etc/sysctl.conf** file and having the following rules in the IP tables (Nigel, 2015):

```
sudo iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
sudo iptables -A FORWARD -i ens33 -o ens34 -m state -state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i ens34 -o ens33 -j ACCEPT
```

There are several methods to route the traffic from infected machines through this router. Some of the approaches are ARP poisoning, DNS spoofing, modifying the hosts

file, setting up Ubuntu as the gateway and DNS as the IP address of the WAN gateway. In any case, the traffic has to be routed to any tool running (e.g. on port 8080) in the router, then, rather than **FORWARD**. The following rules should be used:

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
```

The log or output of the tools for the traffic can then be used for further analysis. The same router configuration should be used for SSL traffic as well.

5.3. Live traffic analysis of malware over SSL

For the analysis of any traffic over SSL, decryption and re-encryption are critical. This is not just to obtain cleartext traffic, which can be retrieved by tools, but also to maintain the connection of malware with its C2 center. There are several approaches and tools to obtain cleartext traffic. This paper examines several.

The environment and tools mentioned below can be used not only for Windows machines but also for other infected platforms, devices and operating systems like Android and iOS. The screenshots for all these tools and commands have been compiled in Appendix - F.

Create SSL certificates considering the security best practices in mind (Gigler, Coates, Wichers, & Reguly, 2016).

5.3.1. Components required to set an environment for SSL traffic analysis

The basic components are the same as mentioned for unencrypted traffic. A few tools that can decrypt and re-encrypt traffic for further analysis are needed: (a) SSLsplit, (b) stunnel, (c) socat, (d) ettercap. These tools are capable of terminating the SSL connection while acting as a server for malware or service that require SSL, and then re-initiates the SSL connection with the actual server behaving like a client/malware.

5.3.2. How to set up an environment

The router configuration for SSL based traffic is the same as above, except that the FORWARD rule is not used. This is because the inbound traffic is redirected to the tools and again redirected from the tool to the network interface having internet connectivity. To achieve this, PREROUTING is used, and other rules are used to route the traffic to and from the tools mentioned below.

SSLsplit: This tool supports plain TCP, plain SSL, HTTP, HTTPS over IPv4 and IPv6 (Roethlisberger, 2009). Use the following command to install SSLsplit:

```
sudo apt-get install sslsplit
```

Then, create a self-signed CA certificate using the following commands. This certificate will be used by the tool to interact with the client (SSL-based service):

```
sudo OpenSSL genrsa -out sslsplit.key 2048
sudo OpenSSL req -new -x509 -days 365 -key sslsplit.key -out sslsplit.crt
For pem format: sudo openssl x509 -in sslplsplit.crt -out sslsplit.pem -outform PEM
```

Now, to execute the SSLsplit tool, use the following commands:

```
sudo ./sslspplit -D -l /path/sslconnection.log -j /tmp/sslsplit -S logs/ -k
/path_of_key/sslsplit.key -c /path_of_cert/sslsplit.crt https 0.0.0.0 8443 tcp 0.0.0.0
8080
```

-D is used to display the connections in the console, and all cleartext traffic is logged into files (one file per connection) at the **/tmp/sslsplit/logs** directory created before the execution of the above command. These files can then be used for further analysis and by other tools. The **sslconnection.log** will save all the SSL connections. In case cleartext traffic of other protocols is required, change the protocol from **https** to **ssl** and the port as well, including the rules in IP tables.

STUNNEL: This tool supports protocols like cifs, connect, imap, ACAP, nntp, pgsql, pop3, proxy, smtp, socks, https, etc. It can support any protocol until its TCP does

not use multiple connections and doesn't depend on Out-Of-Band data. Remote sites cannot use application-specific protocols (Trojnara, n.d.).

For this paper, stunnel will be used in the server and the client mode. Stunnel in the server mode will negotiate with clients like the browser and will negotiate with the actual server (for which the traffic has been generated by the browser) as a client. The approach of setting rules in IP tables will be the same as mentioned earlier.

Following are the steps to install the tool and to create the SSL certificate, apart from the rules of iptables mentioned above:

1. Use the following command to generate a PEM certificate:

```
sudo openssl req -new -out sslsplit.pem -keyout sslsplit.pem -nodes -x509 -days 365
```

2. Use the following command to install the stunnel (version 4 was used for this paper:

```
sudo apt-get install stunnel4
```

To enable the stunnel, enter the following command and change the value of **enabled** from 0 to 1:

```
sudo vi /etc/default/stunnel4
```

3. Copy the configuration file to the **/etc/stunnel** directory using the following command:

```
sudo cp /usr/share/doc/stunnel4/examples/stunnel.conf-sample
/etc/stunnel/stunnel.conf
```

4. Open this configuration file and set the server and the client as given below.

a) For the server:

```
[https]
client = no
accept = 8443
connect = 8080
```

```
cert = /path_to_pem_certificate/sslsplit.pem
```

b) For the client:

```
client = no
accept = 8443
connect = CC_IP_OR_DOMAIN:443
```

This initiates the stunnel in the server mode on port 8443 and in the client mode on port 8080. The client then connects to facebook.com. For malware, this can be the domain of malware or the IP address. Moreover, uncomment options like **foreground** and **debug**, if required.

5. Use the following command to launch the stunnel:

```
sudo stunnel4 /etc/stunnel/stunnel.conf
```

6. Finally, run **tcpdump** on the local interface to sniff and analyze the cleartext communication and save the logs in a file:

```
sudo tcpdump -v -i lo -w sniff.pcap
```

Stunnel can be used with a proxy as well by adding **protocol = proxy** in the server section and disabling its client mode, in conjunction with any proxy like SQUID, HAProxy, TCPProx, or socat, which can act as a client. This will help in maintaining the original domain in the request and avoid giving the domain (CC_IP_OR_DOMAIN, in the above case) explicitly in the **Connect** in the client mode.

SOCAT: It is used as a relay between independent channels by establishing bidirectional byte streams and by transferring data between them. The relay can be used for mechanisms like SSL sockets, files, proxy CONNECT connections, pipe, device (serial line or pseudoterminal), socket (UNIX, TCP, UDP, IP6, raw), a file descriptor, a program and much more for a combination of these. Traffic can be dumped in a hex or text format (Rieger, 2000). Follow the steps below for using socat:

1. Use the following command to install socat:

```
sudo apt-get install socat
```

2. Create the certificate as mentioned above for stunnel and then, execute the following two commands to run socat at ports **8443** and **8080**. Relay traffic from 8080 to and from the C2 center of the malware:

```
sudo socat -v openssl-listen:8443,cert=cert.pem,verify=0,reuseaddr,fork
tcp4:localhost:8080
sudo socat -v tcp4-listen:8080,reuseaddr,fork ssl:IP/Domain_of_CC:443,verify=0
```

3. Traffic can be sniffed at the local interface for further analysis.

ETTERCAP: This tool supports many UDP and TCP protocols, including SSL MITM attacks (available in the ARP mode), sniffing and other features like logging (Ornaghi & Valleri, 2004).

Use the following command to install ettercap:

```
sudo apt-get install ettercap
```

ARP-based sniffing: This is a cleaner approach as there is no need to configure the affected machine. A few settings have been provided below, which are required before initiating ARP Poisoning:

1. Set the UID and GID to 0 (zero) in the **/etc/ettercap/etter.conf** file and uncomment the appropriate firewall rules in the **etter.conf** file. As Ubuntu was used for the following, the rules for iptables are uncommented.
2. The infected machine (192.168.1.8) and the machine (192.168.1.101) with ettercap should be in the same LAN with **192.168.1.1** as the gateway.

Once the above settings are configured and the environment is ready, execute the following code:

```
sudo ettercap -TM arp:remote /192.168.1.8// /192.168.1.1//
```

When the above code is executed, all the traffic would pass through **192.168.1.101** and can be seen as cleartext in the console. Traffic can be logged, piped or redirected to the file for further analysis. There is no need to configure iptables or the IP Forwarding in Ubuntu for the ARP mode.

5.3.3. Manipulation of Traffic

Multiple instances of the above tools can be initiated to be used in combinations or with other tools as well. These combinations would help even in situations where there may be a requirement to modify the traffic. In such cases, tools like "Burp" or any other proxy can be used between two stunnels/socat (or other tools for decryption/re-encryption). Tools like Netsed work even for undocumented protocols and are used for manipulating traffic (Michal Zalewski, 2014). Analysis of DNS logs/traffic too would be very important. Thus, having a DNS in the environment helps.

5.3.4. Limitations

It would be difficult to analyze the traffic if the traffic of the protocol(s) used by malware cannot be parsed by tools or if the traffic is encrypted by malware at the application layer as well as the network layer. This includes custom protocol(s) if there are any in use,

6. Conclusion

With the popularity of malware, which can use domain or IP address of its C2 and the SSL for encryption, the two environments mentioned in this paper have become the need of the hour. These can help Forensics teams, Incident Handlers, Malware Analyst and other teams not just to collaborate on results from different logs, alerts, services, devices, but also in the identification of malware, further analysis and actions like understanding the actual impact, blocking of malicious activities by stopping a malware from contacting & receiving commands from its C2 center or to analyze its DNS interaction. Automated updating of the Sinkhole for the blacklisted domains, as

demonstrated, can help teams to keep their DNS Sinkhole environments up to date. These also help in the analysis of even the encrypted malware traffic, thus resulting in the demystification of malware traffic and in the identification of adversaries in real time. The environments explained are even capable of modifying the malware traffic to send only the useless information to C2 while analysis continues, without interrupting the connection between malware and its C2.

All these can be achieved even when these malware are using techniques like encryption, polymorphism, metamorphism, zero-day exploits, frequent change in the IP address or the domain of C2, time-based attacks, and dead code insertion. This is because any C2-based malware will have to connect to its C2 to get the commands or to send the data to its attacker and that's where these two methods can be utilized easily. These environments are also effective when the malware has the capability to detect the virtualization or to kill the antivirus. A user can directly use the infected machine for the analysis without putting the malware sample in any virtual environment for the analysis. And the two methods either don't require any configuration change in the infected system or require the least amount of changes.

As demonstrated, these environments and their components are not only cost effective but also provide a full control to its users/researchers, as all these tools and environments are and can be configured using freeware and open-sources. In closing, there are many tools and Operating Systems (OS) available to analyze the traffic as well as the malware itself. Thus, setting up an environment should not be limited to just the tools mentioned in this paper. Tools/OS like SQUID, Cain and Able, DSniff, Wireshark, Kali, and REMnux among others should be explored further to satisfy specific requirements.

Sourabh Saxena, sourabhsaxena25@gmail.com

7. References

Monteith, K. (2010, February 10). *Survey Results: PCI Standards Helpful, Confusing and Necessary | Practical Ecommerce*. Retrieved from http://www.practicalecommerce.com/articles/1641-Survey-Results-PCI-Standards-Helpful-Confusing-and-Necessary

Weber, D. (2012, September). *TRANSFORMING TRADITIONAL SECURITY STRATEGIES INTO AN EARLY WARNING SYSTEM FOR ADVANCED THREATS*. Retrieved from https://www.emc.com/collateral/software/solution-overview/h11031-transforming-traditional-security-strategies-so.pdf

Albitz, P., & Liu, C. (2016, August 7). *BIND9ServerHowto - Community Help Wiki*. Retrieved August 13, 2016, from https://help.ubuntu.com/community/BIND9ServerHowto

Akamai. (2016). *Security Compliance | Akamai*. Retrieved from https://www.akamai.com/us/en/resources/security-compliance.jsp

Grimes, R. A. (2012, August 13). *9 popular IT security practices that just don't work | InfoWorld*. Retrieved from http://www.infoworld.com/article/2617998/security/9-popular-it-security-practices-that-just-don-t-work.html

LaRiza, V. (2015, April 1). *New Malware Attacks On The Threat Horizon - OpenDNS Umbrella Blog*. Retrieved from https://labs.opendns.com/2015/04/01/new-malware-attacks-on-the-threat-horizon/

Nieto, J. (2015, December 13). *A Network Traffic Analysis Exercise ~ Hacking while you're asleep*. Retrieved from http://www.behindthefirewalls.com/2015/12/a-network-traffic-analysis-exercise.html

Gigler, T., Coates, M., Wichers, D., & Reguly, T. (2016, June 19). *Transport Layer Protection Cheat Sheet - OWASP*. Retrieved from https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

Rouse, M. (2015, July). *What is DNS attack? - Definition from WhatIs.com*. Retrieved from http://searchsecurity.techtarget.com/definition/DNS-attack

Sourabh Saxena, sourabhsaxena25@gmail.com

Bambenek, J. (2015, June 4). *Principles of Malware Sinkholing - Dark Reading*. Retrieved from http://www.darkreading.com/partner-perspectives/general-dynamics-fidelis/principles-of-malware-sinkholing/a/d-id/1319769

Yim, K. (2010). *Malware Obfuscation Techniques: A Brief Survey*. Retrieved from https://pdfs.semanticscholar.org/0401/3804edf3e86218d15868269a355dd3501c0f.pdf

Myers, L. (2013, December). *Why You Need an Outbound Firewall*. Retrieved from https://www.intego.com/mac-security-blog/why-you-need-an-outbound-firewall/

Sweigart, A. (2015). *Automate the Boring Stuff with Python*. Retrieved from https://automatetheboringstuff.com/

Nigel. (2015, August). *IptablesHowTo*. Retrieved from https://help.ubuntu.com/community/IptablesHowTo

Zalewski, M. (2014). *foxsen/netsed: small tool to sed socket contents*. Retrieved from https://github.com/foxsen/netsed

Roethlisberger, D. (2009). *SSLsplit - transparent SSL/TLS interception (SSLsplit)*. Retrieved from http://www.roe.ch/SSLsplit

Trojnara, M. (n.d.). *stunnel TLS Proxy*. Retrieved from https://www.stunnel.org/static/stunnel.html

Rieger, G. (2000). *socat*. Retrieved from http://www.dest-unreach.org/socat/

Virgillito, D. (2014, September). *How a DNS Sinkhole Can Protect Against Malware - InfoSec Resources*. Retrieved from http://resources.infosecinstitute.com/dns-sinkhole-can-protect-malware/

Ornaghi, A., & Valleri, M. (2004). *Ettercap Home Page*. Retrieved from https://ettercap.github.io/ettercap/

Sourabh Saxena, sourabhsaxena25@gmail.com

Appendix - A

Security Best Practices While Setting Up DNS Server For DNS Sinkhole

If the DNS Sinkhole is not configured securely, it may backfire. It does so if any attack is initiated on the sinkhole environment or the weakly configured DNS service. Following are a few security best practices to set a secure DNS Sinkhole:

1. For security reasons, recursion should be disabled on internet-facing DNS servers of any organization to avoid attacks like DoS, Cache Poisoning, DNS Amplification Attacks, etc.
2. Internet-facing DNS Servers should answer to only those queries that they are authoritative for and for child domains. This prevents DNS Cache Snooping attacks.
3. Disable Zone Transfer and limit it only for slaves.
4. DNS Transaction/Query Identifier and the source port should be randomized.
5. Maintain a Split DNS configuration. The internal name server answers to only the internal names and sends external names to the external name server. This external name server does a full recursion for this query and forwards it back to the internal server.
6. Organizations can have a stronger configuration, i.e. split–split DNS configuration, where the internal name server does not contact the external server and is responsible only for internal name resolution. There's a third name server, which receives queries for external names from inside.
7. Implement security measures against attacks like DNS amplification, DoS, Cache Poisoning, Fast Flux attack, etc.

Sourabh Saxena, sourabhsaxena25@gmail.com

Appendix - B

General approach by organizations/users to handle malware

Different organizations use different approaches for handling malware. However, a few common approaches are the usage of AV software, IDS, IPS, and firewalls. The functionalities of these security measures are listed below, before the ways by which these can be bypassed by the malware is comprehended. Approaches used to bypass the detection have been highlighted in Appendix - C.

1. AVs: There are different approaches used by AVs to identify malware.

    a. Signature-Based: AVs use virus definition to identify malware. This definition contains the signature of the malware. The signature can be a series of bytes of malware or the cryptographic hash of the malware or its section.

    b. Heuristic-Based: In this approach, AV checks a few characteristics in the code to identify actions that malware may perform without having an exact signature. For example, the file with a set of instructions to open all the critical and executable files, adding a set of instructions into the executables, and reading and copying the critical files, would be flagged. AV may emulate the running of the file to detect commands and characteristics.

    c. Behavior-Based: AVs check for the malicious behavior of a file for flagging it as malware. Detection with this approach is possible only when the malware is in action.

2. IDS: It can be hardware or software based and is used to analyze network and system activities for any intrusion or malicious activity or violation of rules and policies. Network-based IDS (NIDS) identifies the malicious activities on the network traffic. Host-based IDS (HIDS) identifies unauthorized activities

with the system. IDS has functionalities like gathering and analyzing the logs, generating alerts, etc. However, it is not used for blocking malicious behavior.

3. IPS: This is used just like IDS, but it has the capability to block, i.e. prevent, malicious activities as well.
4. Firewalls: This can be hardware or software based and is used to control the inbound and outbound traffic, as per the rules defined for the traffic, thus preventing unauthorized connections and traffic to or from a network. There are different kinds of firewalls, such as Stateful F/W, Proxy F/W, Application Layer F/W or Packet F/W.

Appendix - C

Why the above security mechanisms may miss detecting malware

There are many ways in which malware can avoid detection. Some of these are as follows:

1. Encrypted Traffic: By encrypting the traffic or its payload, it's possible for the malware to remain undetected.
2. Oligomorphic Malware: This malware has the encrypted payloads and has the list of decryptors, randomly selected in each infection, mutating from one variation to other.
3. Polymorphism: A few malware change their code every time it infects the code. Each time a new decryptor is created, it can have an unlimited number of decryptors. Several techniques can be used, such as dead-code insertion, instruction substitution, etc.
4. Metamorphic Malware: Each time a malware infects a system, a completely new payload is written without changing the action.
5. Zero-Days: Zero-day malware may not get detected due to the lack of signatures.
6. Frequent domain/IP changes: Malware can connect to a dedicated C2 or can get and use a list of domains/IP addresses to connect to, making detection difficult. These techniques are also known as Domain-Shadowing or Fast Flux Technologies.
7. Detection of AV, Virtual Machines: A few malware are capable of changing their behavior or go dormant if they detect any antivirus or environment to be a virtual machine.
8. Concealing the Traffic: Malware may conceal their traffic using mechanisms like TOR, I2P, etc.

Sourabh Saxena, sourabhsaxena25@gmail.com

9. Outdated updates: Malware may work well if the products and antiviruses are not up to date.
10. Time-Based: A few malware remain dormant and work only when there is very less security in the system or only at a specific time.
11. Dead Code Insertion (one of the obfuscation techniques): In this technique, a few ineffective codes are added to change the structure of the code, while maintaining the behavior.
12. Subroutine reordering, Code Transportation, etc. are ways to go undetected.

Appendix - D

Security Analysts and the Research teams need to keep their DNS Sinkholes updated by retrieving the latest blacklisted domains and IP addresses. The blacklisted domains and the IP addresses can be retrieved from the following sites, which maintain such records:

- http://www.malware-domains.com/
- http://www.malwaredomainlist.com/mdl.php
- www.abuse.ch - SpyEye and ZeuS Trackers
- www.malwaredomains.com
- www.threatexpert.com
- https://isc.sans.edu/block.txt
- https://zeltser.com/malicious-ip-blocklists/
- http://www.malwaredomainlist.com/
- http://www.selectrealsecurity.com/public-block-lists
- b.barracudacentral.org
- bl.emailbasura.org
- bl.spamcop.net
- blacklist.woody.ch
- cbl.abuseat.org
- combined.abuse.ch
- db.wpbl.info
- dnsbl-2.uceprotect.net
- dnsbl.cyberlogic.net
- dnsbl.njabl.org
- drone.abuse.ch
- dul.dnsbl.sorbs.net
- dyna.spamrats.com
- http.dnsbl.sorbs.net
- ips.backscatterer.org
- korea.services.net
- multi.surbl.org
- orvedb.aupads.org

Sourabh Saxena, sourabhsaxena25@gmail.com

- phishing.rbl.msrbl.net
- psbl.surriel.com
- recent.spam.dnsbl.sorbs.net
- relays.bl.kundenserver.de
- relays.nether.net
- short.rbl.jp
- socks.dnsbl.sorbs.net
- spam.dnsbl.sorbs.net
- spam.spamrats.com
- spamrbl.imp.ch
- torserver.tor.dnsbl.sectoor.de
- ubl.unsubscore.com
- virus.rbl.jp
- web.dnsbl.sorbs.net
- xbl.spamhaus.org
- zombie.dnsbl.sorbs.net
- bl.deadbeef.com
- bl.spamcannibal.org
- blackholes.five-ten-sg.com
- bogons.cymru.com
- cdl.anti-spam.org.cn
- combined.rbl.msrbl.net
- dnsbl-1.uceprotect.net
- dnsbl-3.uceprotect.net
- dnsbl.inps.de
- dnsbl.sorbs.net
- duinv.aupads.org
- dul.ru dynip.rothen.com
- images.rbl.msrbl.net
- ix.dnsbl.manitu.net
- misc.dnsbl.sorbs.net
- noptr.spamrats.com
- pbl.spamhaus.org
- proxy.bl.gweep.ca
- rbl.interserver.net
- relays.bl.gweep.ca

Sourabh Saxena, sourabhsaxena25@gmail.com

- relays.dnsbl.sorbs.net
- sbl.spamhaus.org
- smtp.dnsbl.sorbs.net
- spam.abuse.ch
- spam.rbl.msrbl.net
- spamlist.or.kr
- tor.dnsbl.sectoor.de
- ubl.lashback.com
- virbl.bit.nl
- virus.rbl.msrbl.net
- wormrbl.imp.ch
- zen.spamhaus.org

Sourabh Saxena, sourabhsaxena25@gmail.com

Appendix - E

A Detailed Flow For "What happens when a system gets infected by a malware"

A closer look into the scenario explained in "*What happens when a system gets infected by a malware*" has been depicted below using the following software/packages/operating-systems: Windows7 (the system that is to be infected, i.e., victim's machine), Kali (X.X.37.220 is to be used to host a virus named as virus.exe using Apache server), BIND9 (X.X.37.64 for DNS daemon, running on Ubuntu – the IP of this server will be used as the Preferred and Alternate DNS Servers in the victim's machine – representing a genuine DNS Cache Server) and TCPDump (to view the traffic to and from the DNS Server).

A harmless program (virus.exe) is used as a malware to show the infection, where this virus runs and sends an HTTP request to **example.com** (consider this as a malicious site for this setup) and receives a response. The response is saved locally and is executed. In a real world, malicious sites send the commands/payloads in the response.

1. The following screenshot shows the Kali server hosting the **virus.exe** and also the TCPDump at work to monitor the traffic of downloading the virus from the victim's machine.

The **apt-get install apache2** command can be used to install the Apache server for Debian Linux and its variants.

Sourabh Saxena, sourabhsaxena25@gmail.com

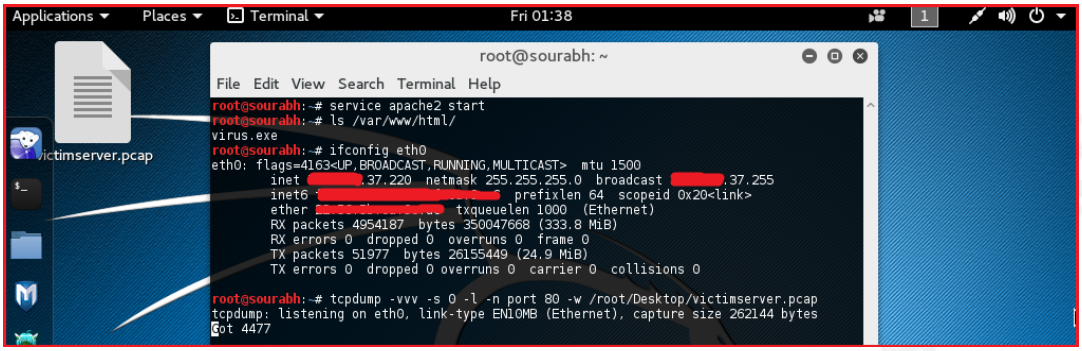


Figure 1: IP configuration of the C2 center

2. The following screenshot shows the victim's machine with the DNS configuration having the IP address of the DNS server, the downloaded .exe file and the response file created as **example.html** after the execution.

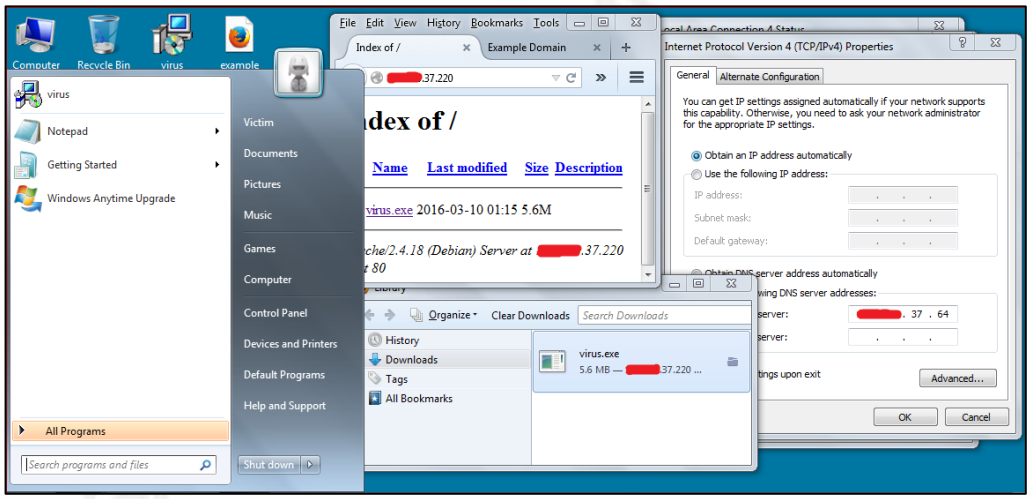


Figure 2: Infected machine with DNS as X.X.37.64

3. The following screenshot shows the DNS server, which was used by the victim's machine for the DNS query when **virus.exe** got executed and the IP address of **example.com** was requested. It also shows how **bind9** and **tcpdump** were initiated.

Sourabh Saxena, sourabhsaxena25@gmail.com

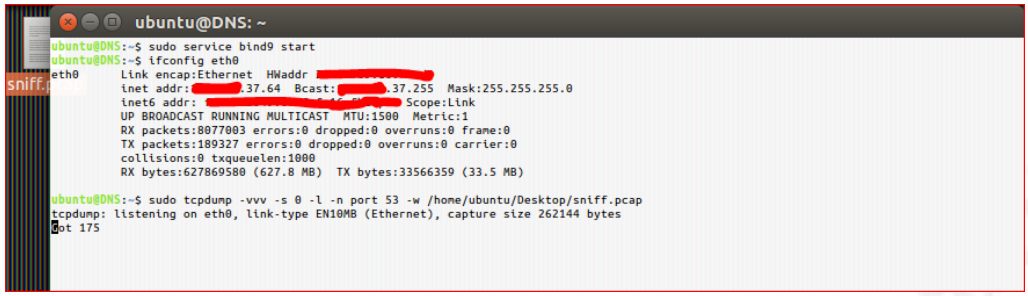


Figure 3: Starting BIND service and TCPDump

4. The following screenshot shows the packets captured in the DNS server for the DNS query from the victim's machine and the response for that query.

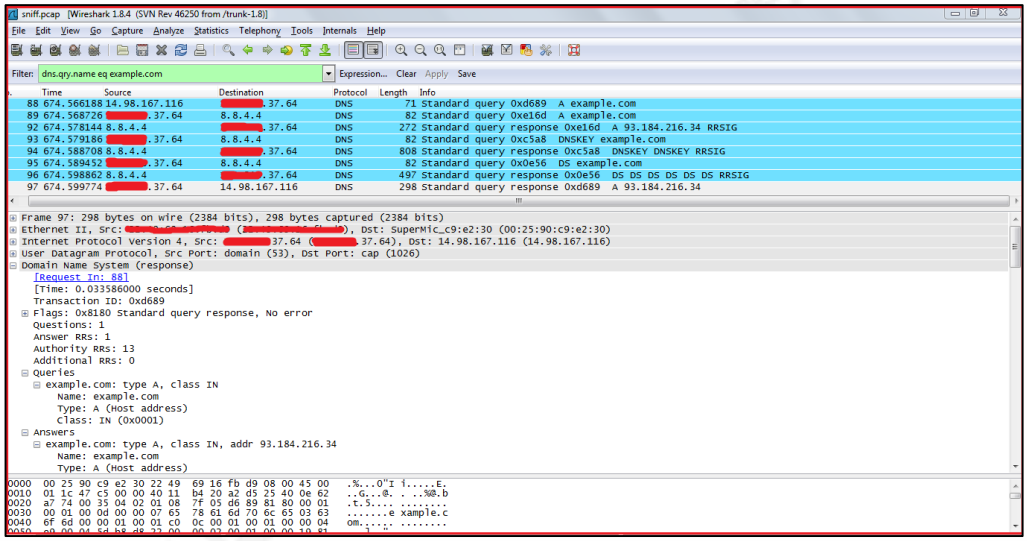


Figure 4: Sniffed data for DNS query

5. The following screenshot shows the packet detail for the downloading of **virus.exe** from the web server.

Sourabh Saxena, sourabhsaxena25@gmail.com

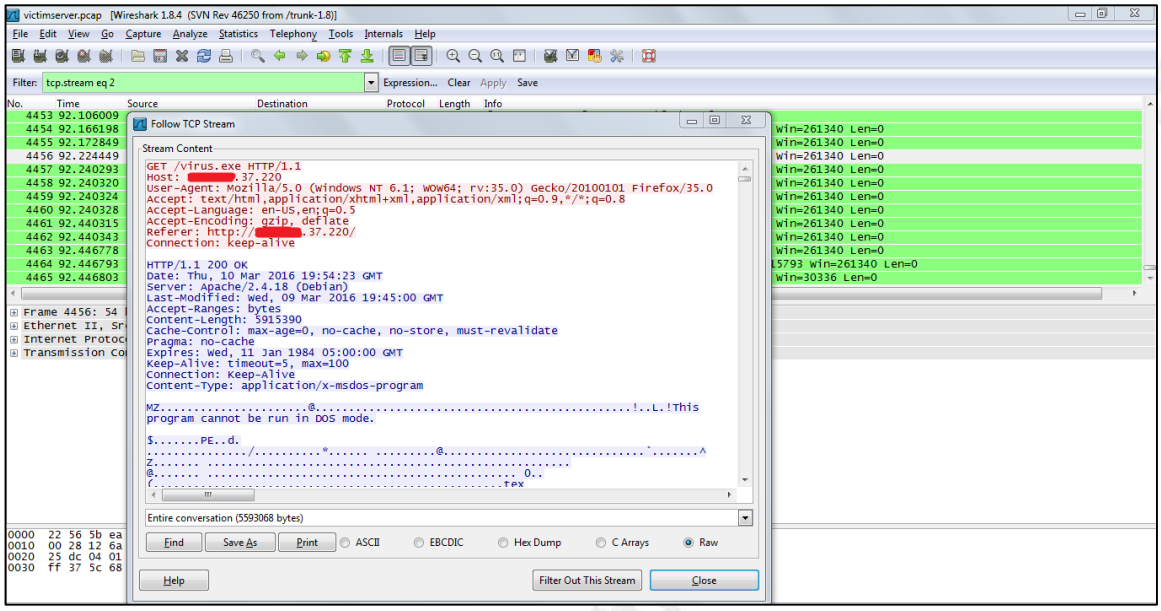


Figure 5: Downloading of malware

6. The following screenshots show the packet details.

   a. The DNS query and response:

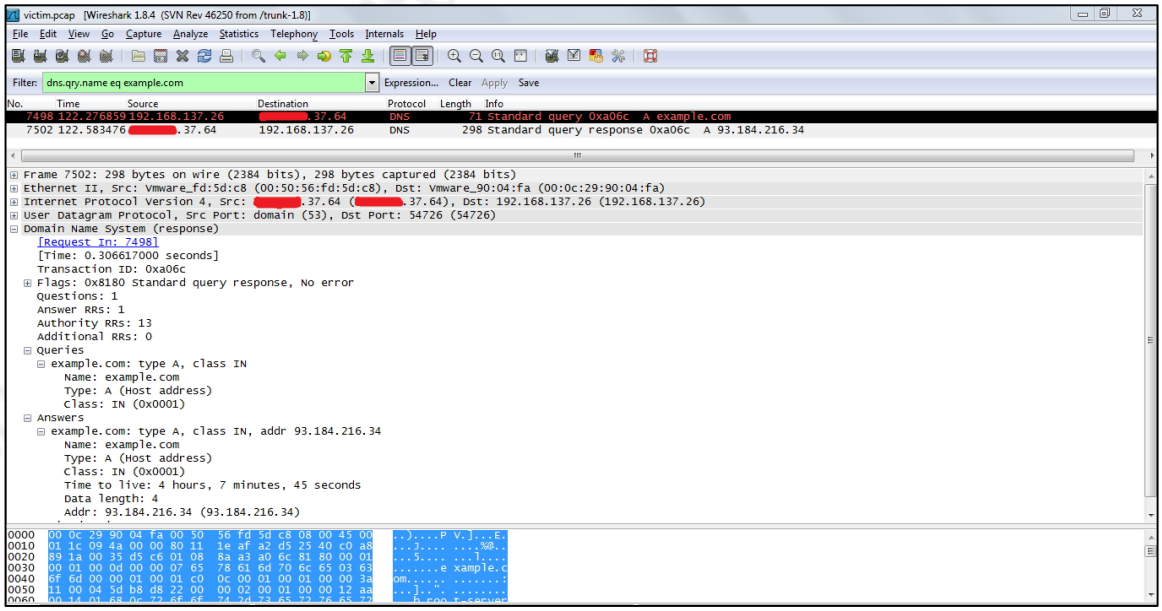


Figure 6: DNS Query and Response

Sourabh Saxena, sourabhsaxena25@gmail.com

b. The request to and the response from C2 when **virus.exe** was executed in the victim's machine.

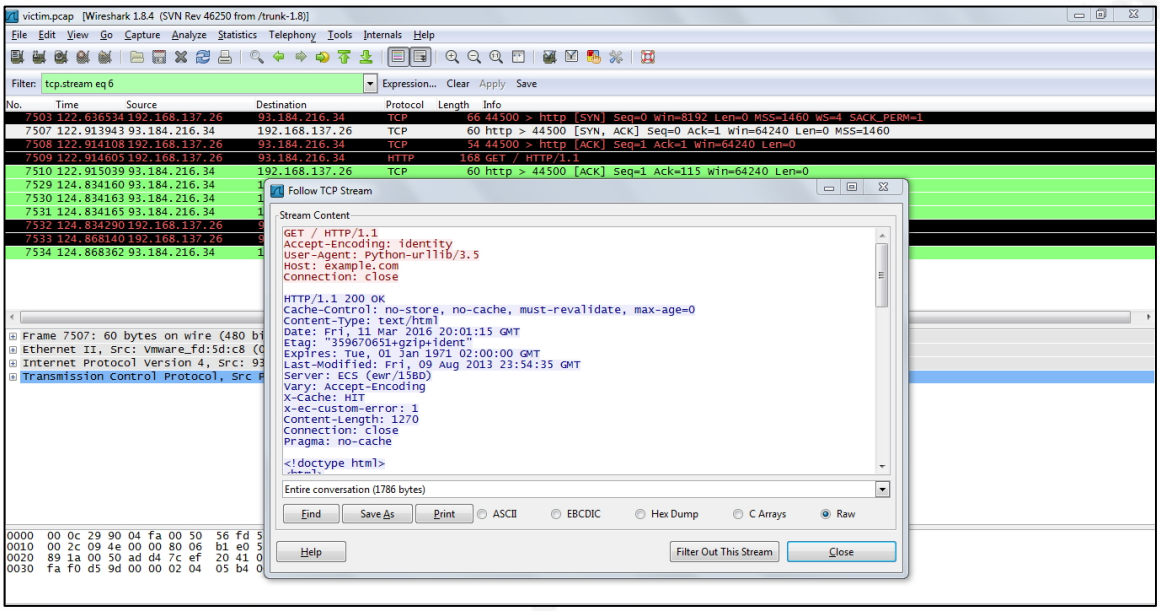


Figure 7: Traffic of malware

c. The following screenshot shows the entire flow of data, starting from the virus download, different stages when the virus was executed (i.e. DNS resolution), connection to C2 and the request to and response from C2.

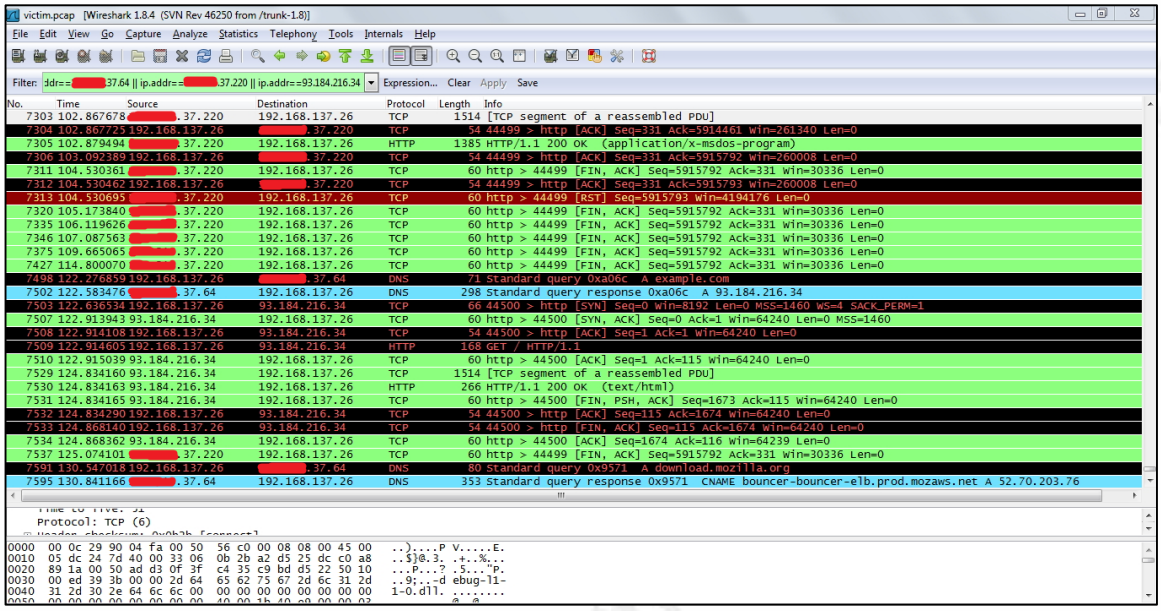


Figure 8: Entire traffic flow

The IP address of all the servers and machines has been suppressed and are shown in the screenshots below.

BIND9 was installed using the following command:

apt-get install bind9

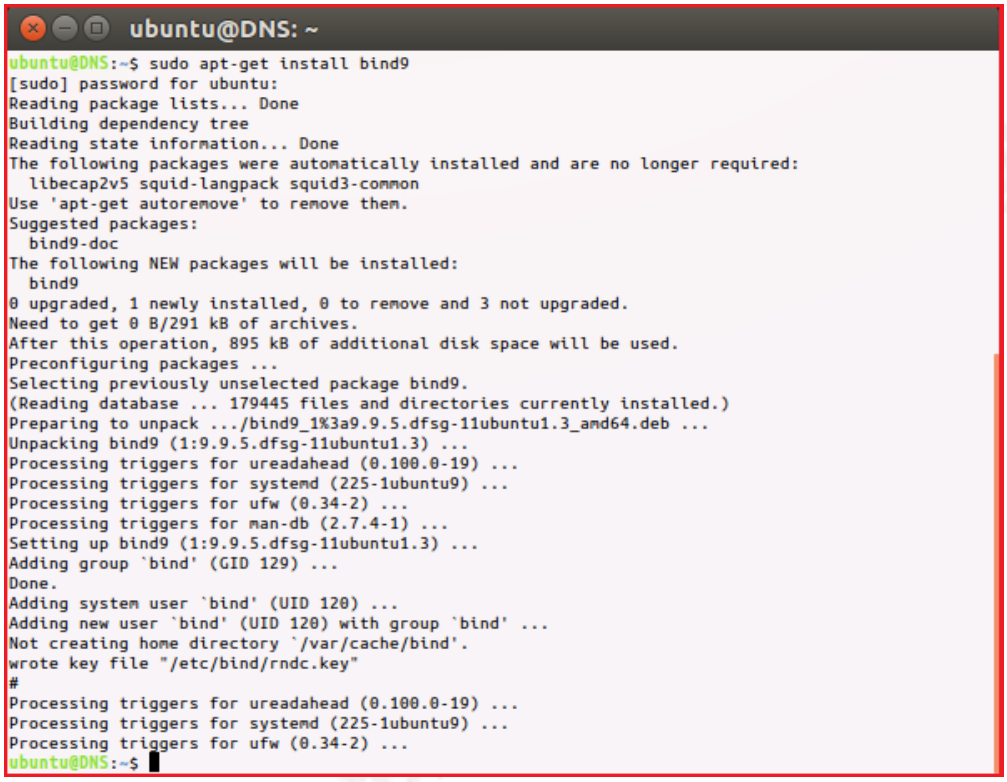


Figure 9: BIND installation

This service was configured in a simple way, by modifying the forwarder in **named.conf.options**:

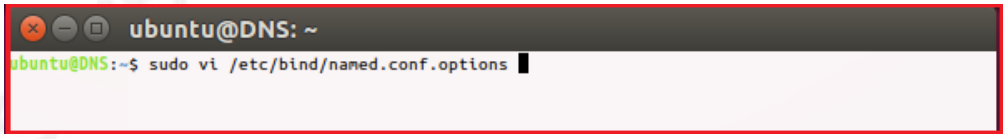


Figure 10: Opening the configuration file

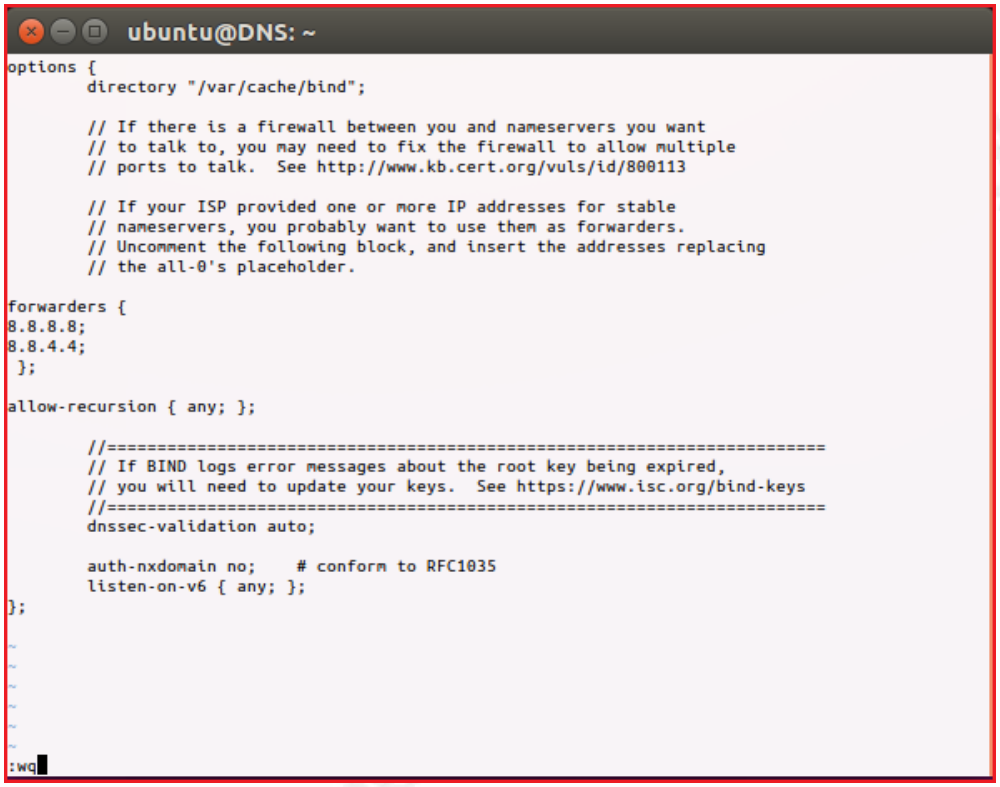


Figure 11: Settings in the config file to act as a forwarder

Appendix - F

How to create a virus for the above environments

1. Create **virus.exe**:

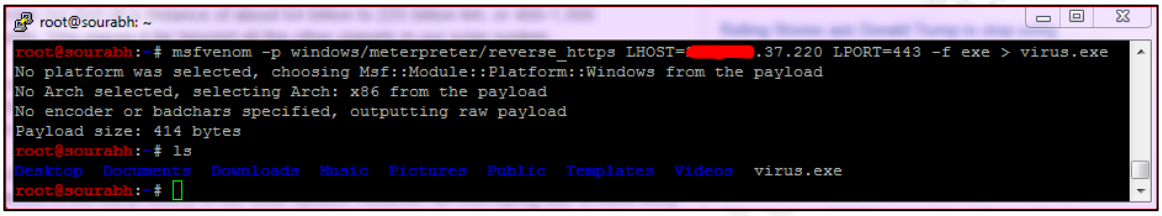


Figure 1: Creating a virus using msfvemon

2. Data over the network when the above virus connects to its C&C over SSL:

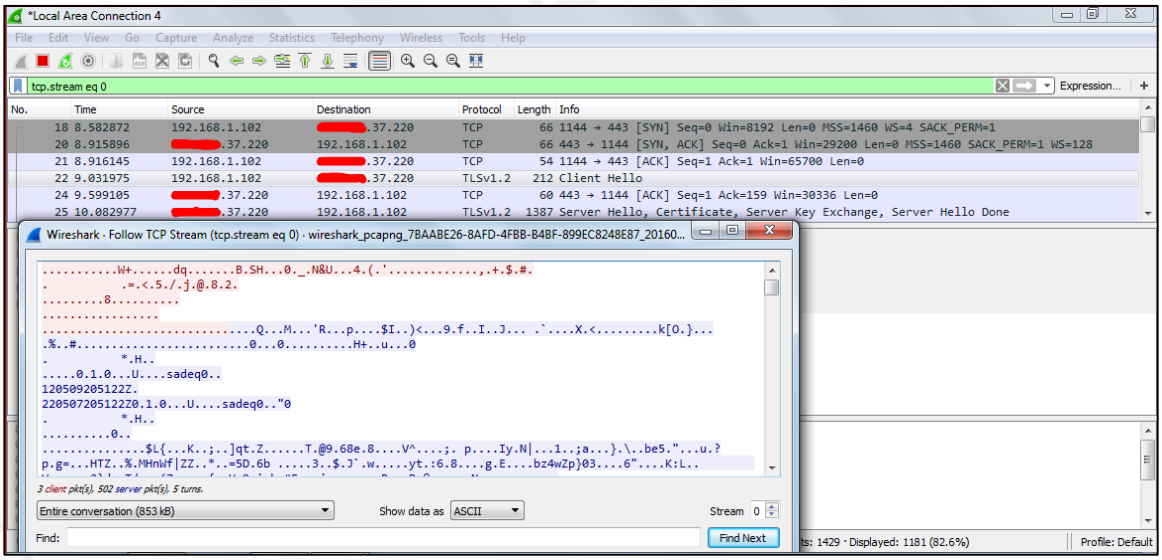


Figure 2: Traffic over SSL

Sourabh Saxena, sourabhsaxena25@gmail.com

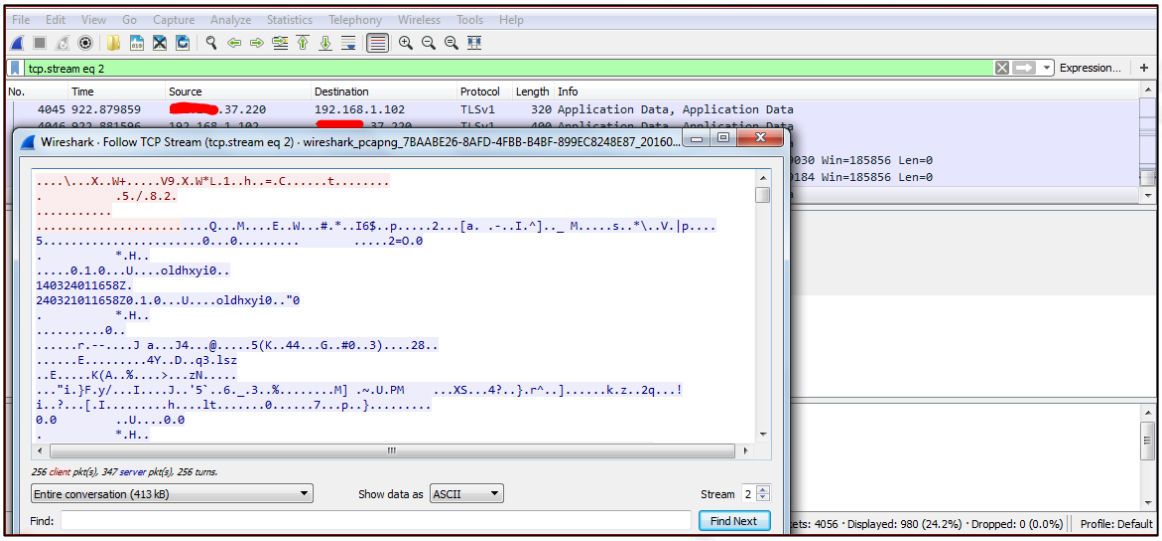


Figure 3: Traffic over SSL

SSL Traffic In ClearText When SSLsplit Is Used

1. SSL connection displayed in the console when SSLsplit is used:

Sourabh Saxena, sourabhsaxena25@gmail.com

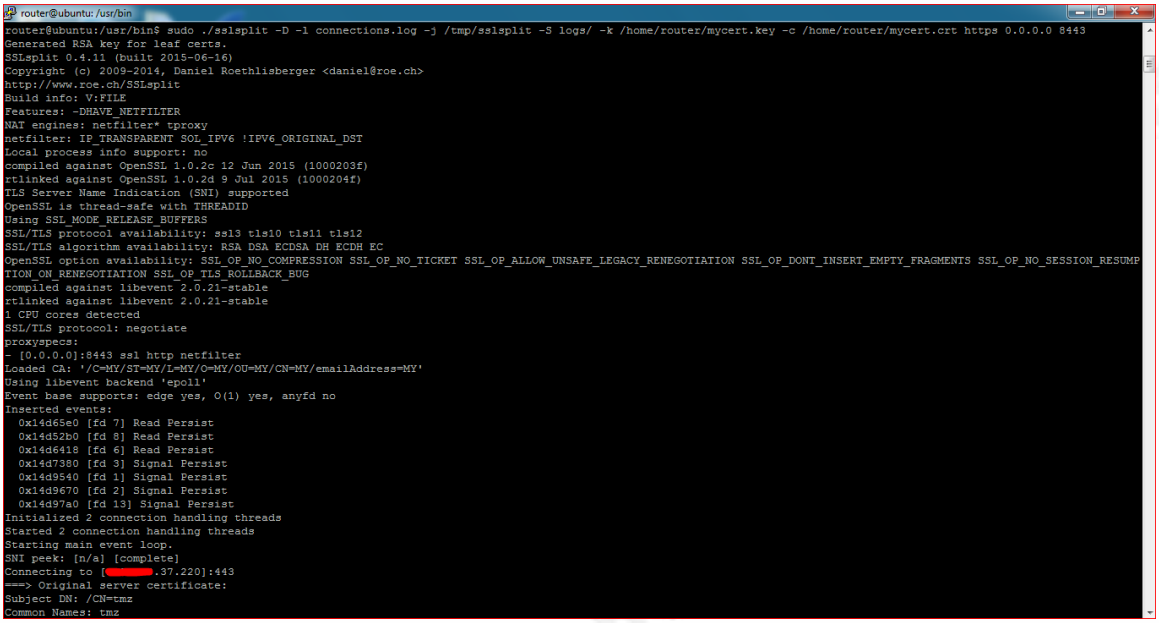
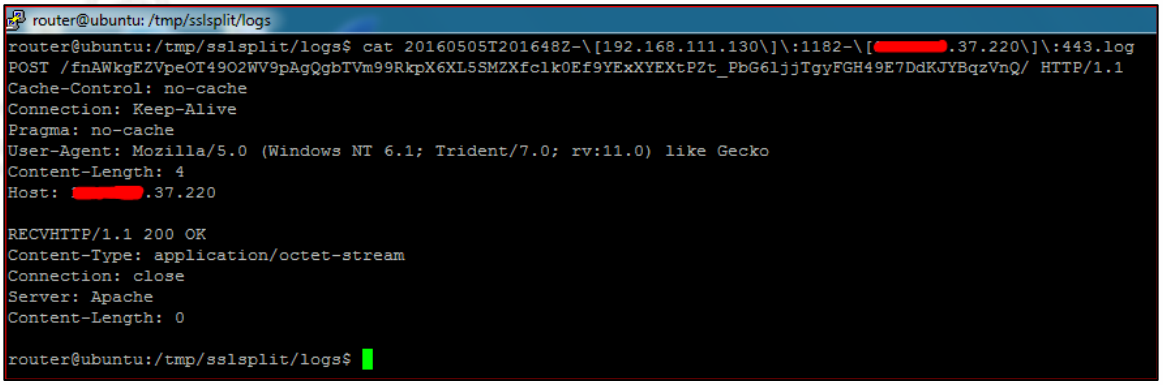


Figure 4: Execution of sslsplit tool and SSL Connection

SSL traffic of the above virus appears as cleartext, captured using the SSLsplit tool – This includes all the Commands received from C2 and the information sent to C2

1. First request and response. As it was a reverse connection, all the requests are from an infected machine and the responses are from C2:



Sourabh Saxena, sourabhsaxena25@gmail.com

Figure 5: Cleartext traffic in the logs of the SSLsplit tool

2. The following screenshot shows a command (getuid) sent from C2 in the response:

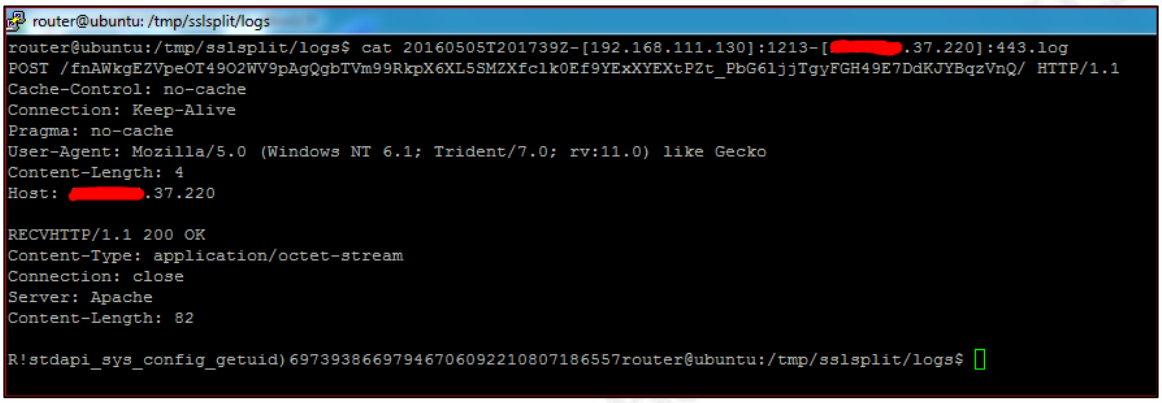


Figure 6: Commands in cleartext

3. The victim's machine sends the output in the request. The suppressed string is the computer's name, and **a** (after the forward slash) is the username by which the user is logged into the machine:

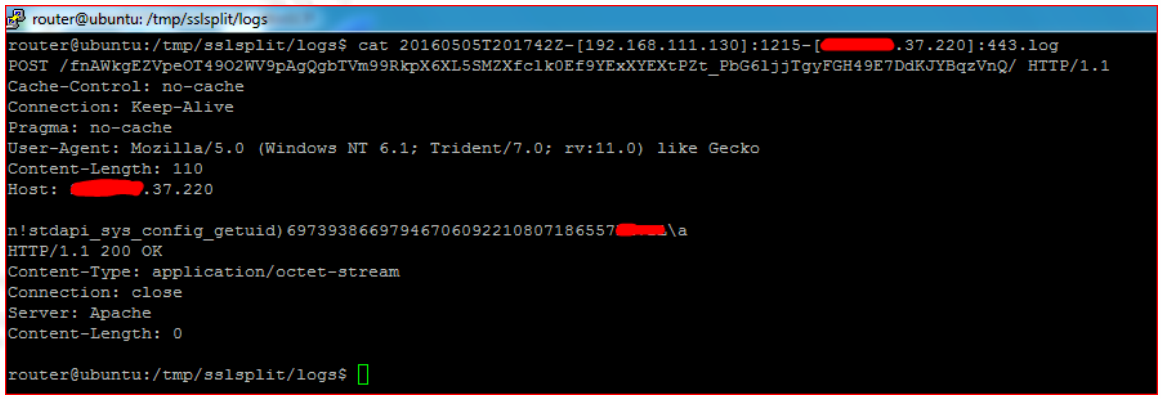


Figure 7: Response in cleartext

Sourabh Saxena, sourabhsaxena25@gmail.com

4. The following screenshot is for another command (ipconfig as get_interfaces) sent by C2:

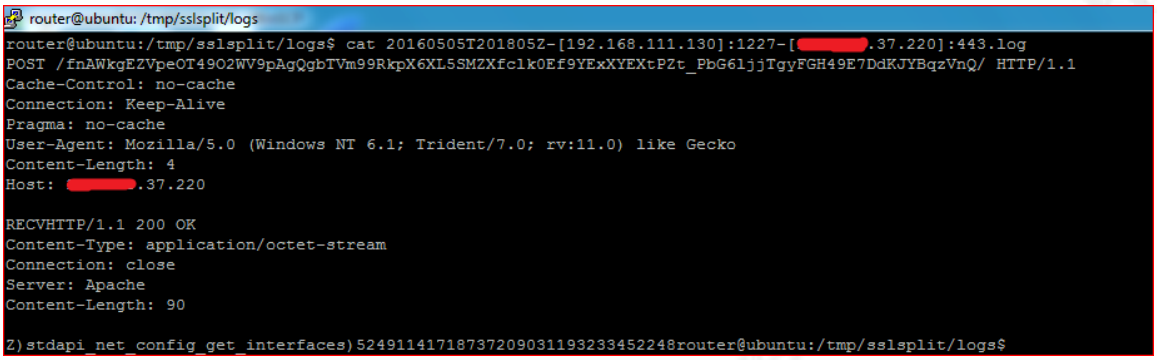


Figure 8: Other commands in cleartext

SSLsplit Setup:

1. Installation of SSLsplit:

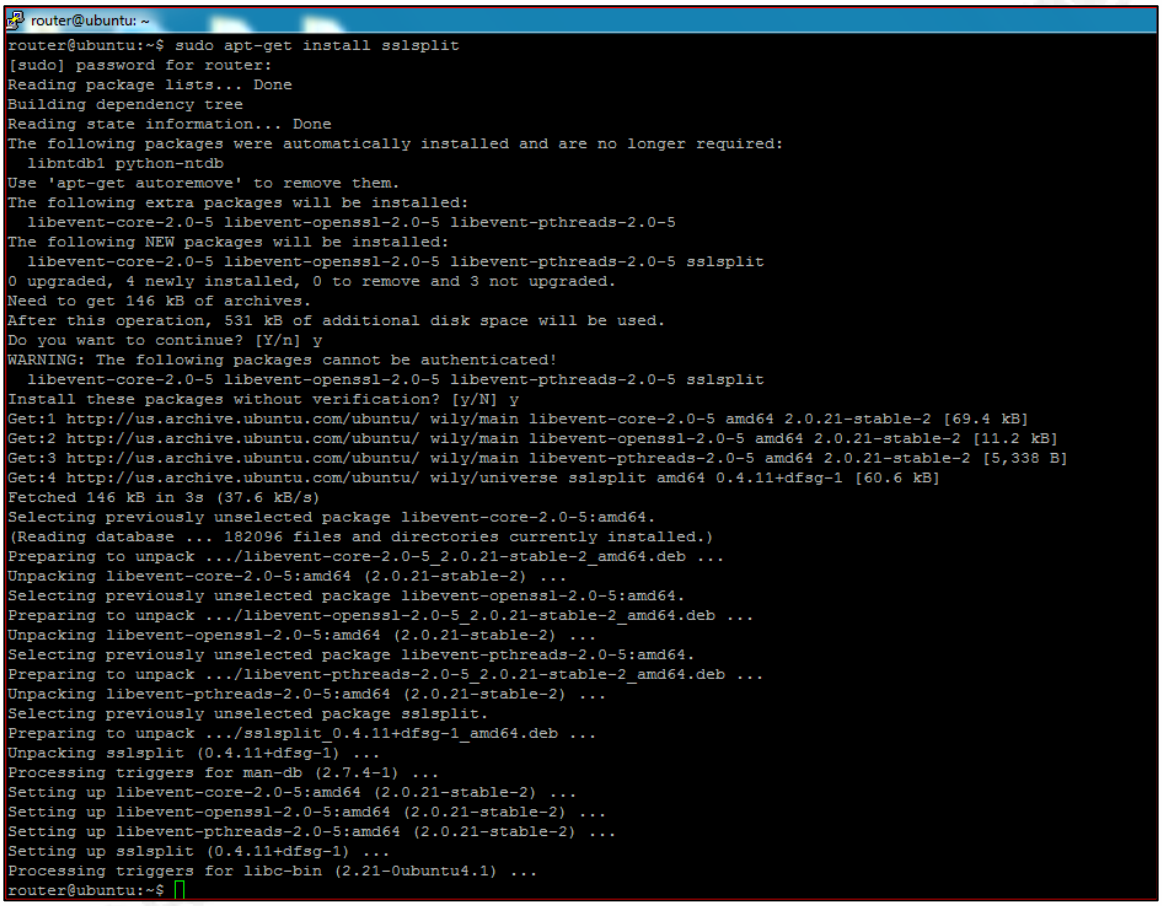


Figure 4: Installation of SSLsplit

2. Certificate Creation:

Sourabh Saxena, sourabhsaxena25@gmail.com

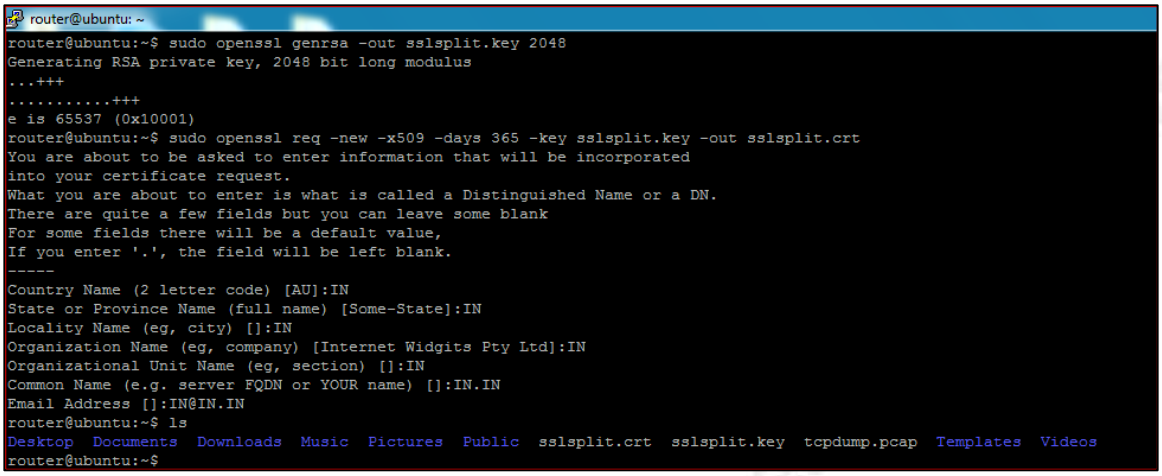


Figure 5: SSL certificate for SSLsplit

3. Execution of SSLsplit:

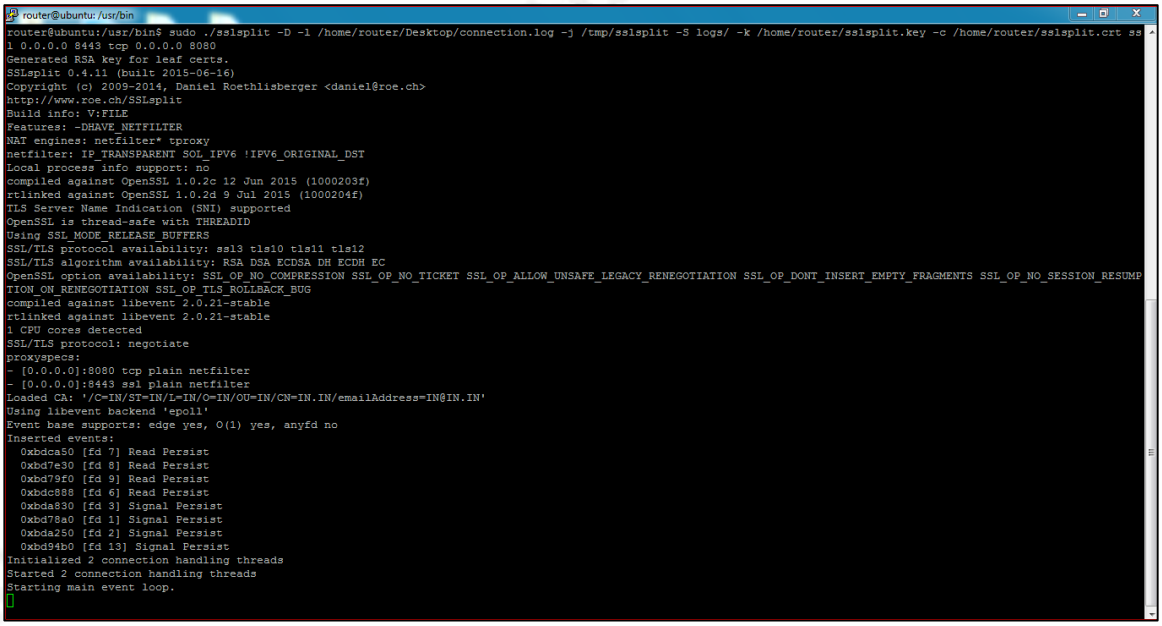


Figure 6: Execution of SSLsplit

4. Certificate warning in the HTTPS browser:

Sourabh Saxena, sourabhsaxena25@gmail.com

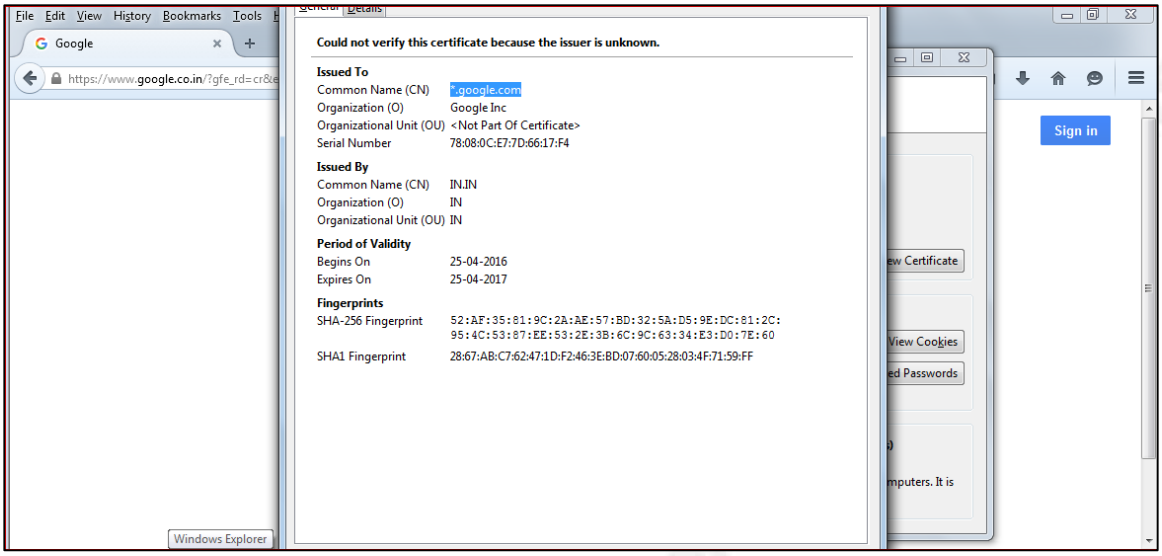


Figure 7: SSL warning in the browser

5. SSL negotiation of SSLsplit with the actual server:

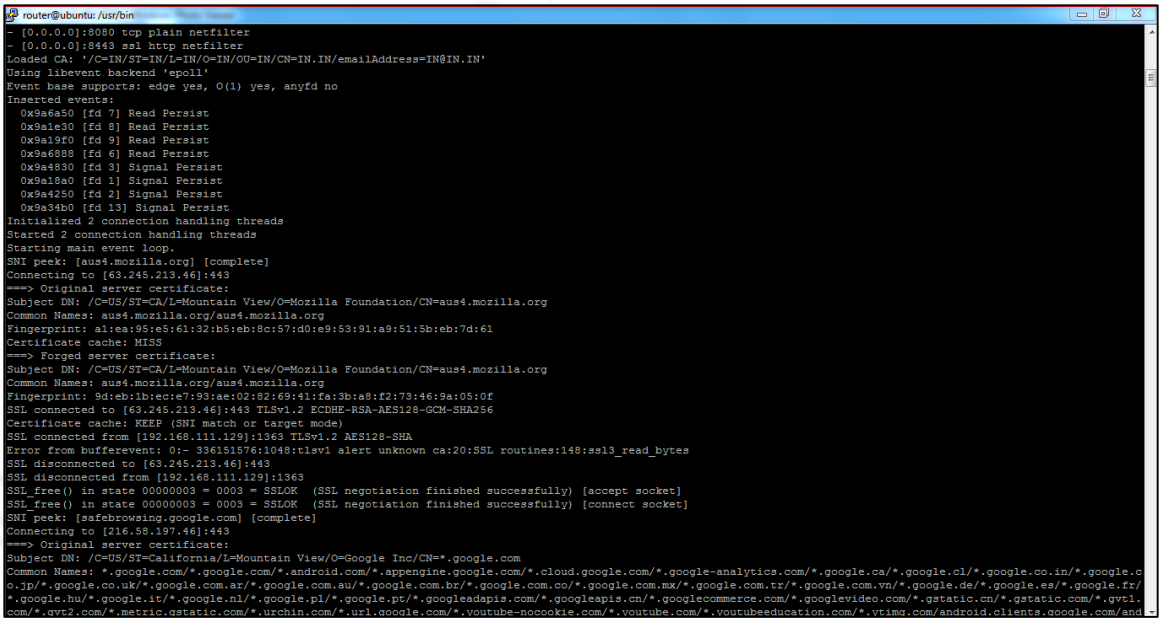


Figure 8: SSL negotiation with server

6. Cleartext traffic:

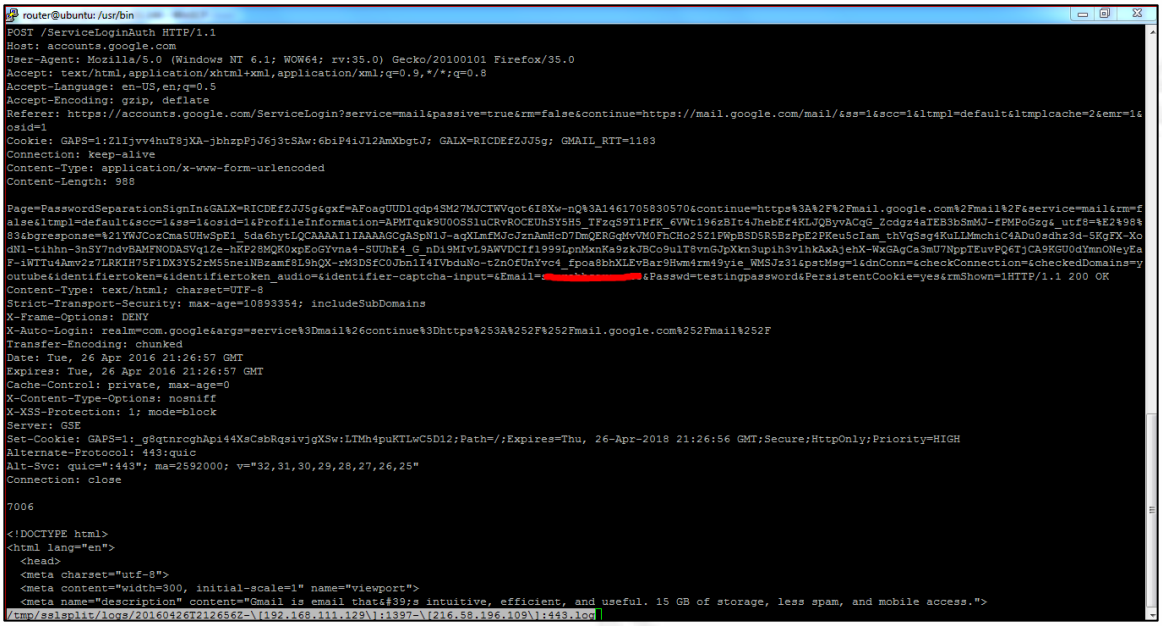


Figure 9: Cleartext traffic in the console

Stunnel Setup:

1. Installation of the stunnel tool:

Sourabh Saxena, sourabhsaxena25@gmail.com

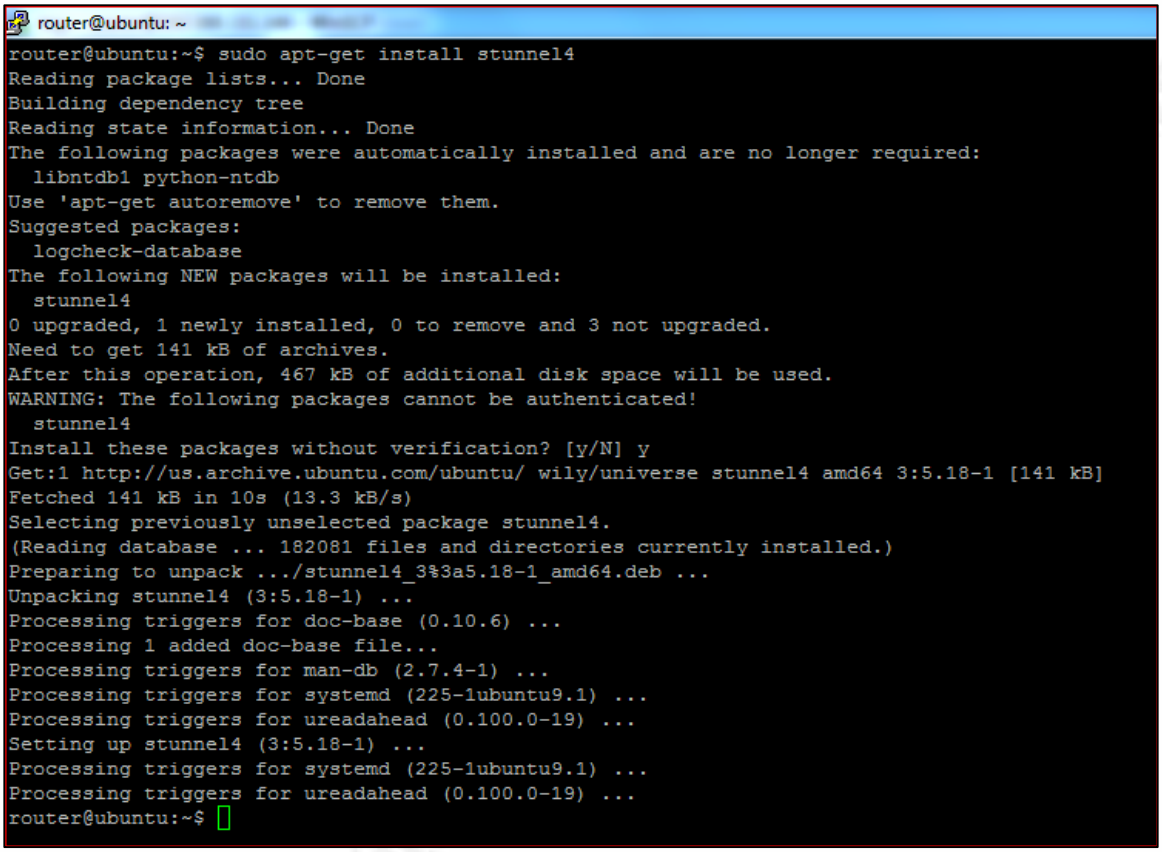


Figure 10: Installation of stunnel

2. Changing the value of the ENABLED parameter:

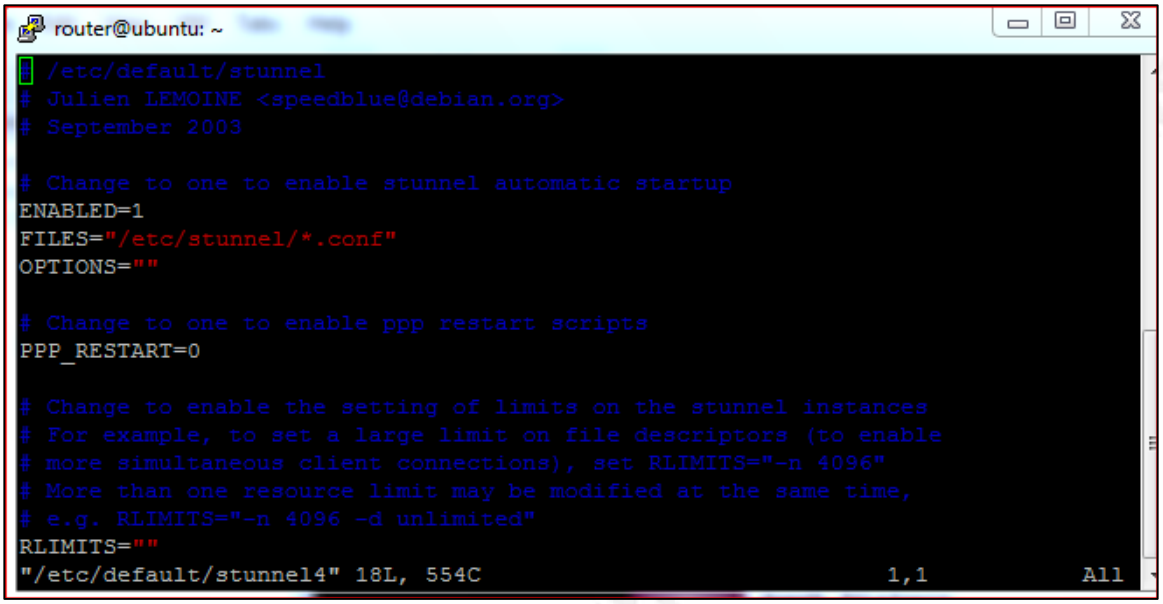


Figure 11: Settings in the stunnel configuration file

3. Certificate for stunnel:

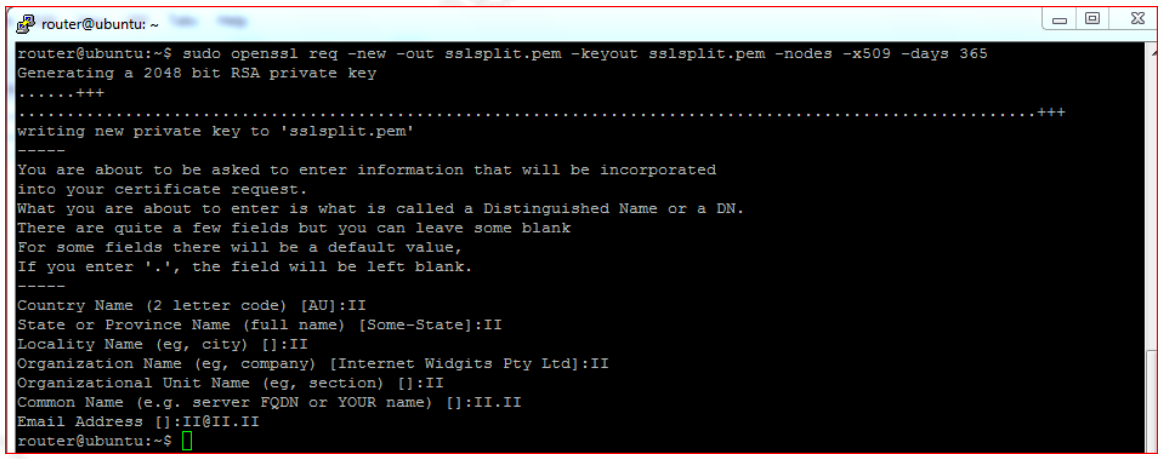


Figure 12: SSL certificate for stunnel

4. SSL connection of stunnel with the client as well as the server:

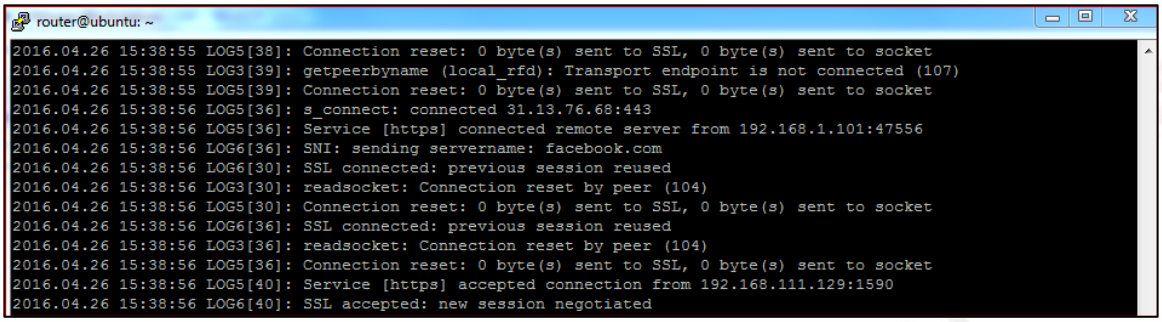


Figure 13: SSL connections with client and server

5. Data sniffed as cleartext:

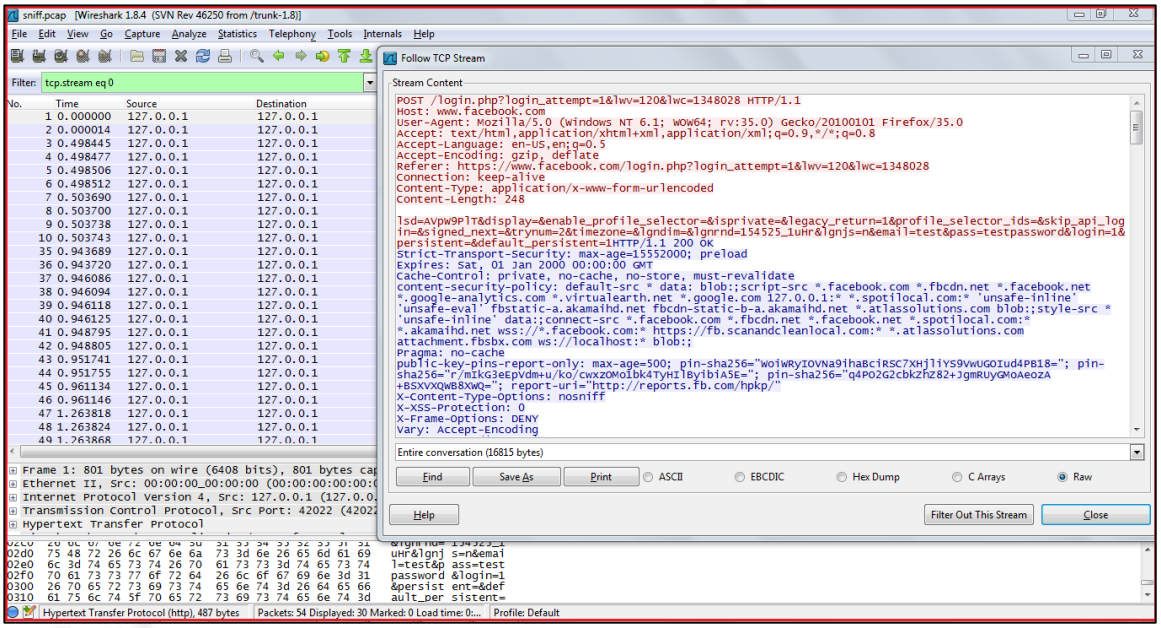


Figure 14: Cleartext traffic

Socat Setup:

1. Installation of the socat tool:

Sourabh Saxena, sourabhsaxena25@gmail.com

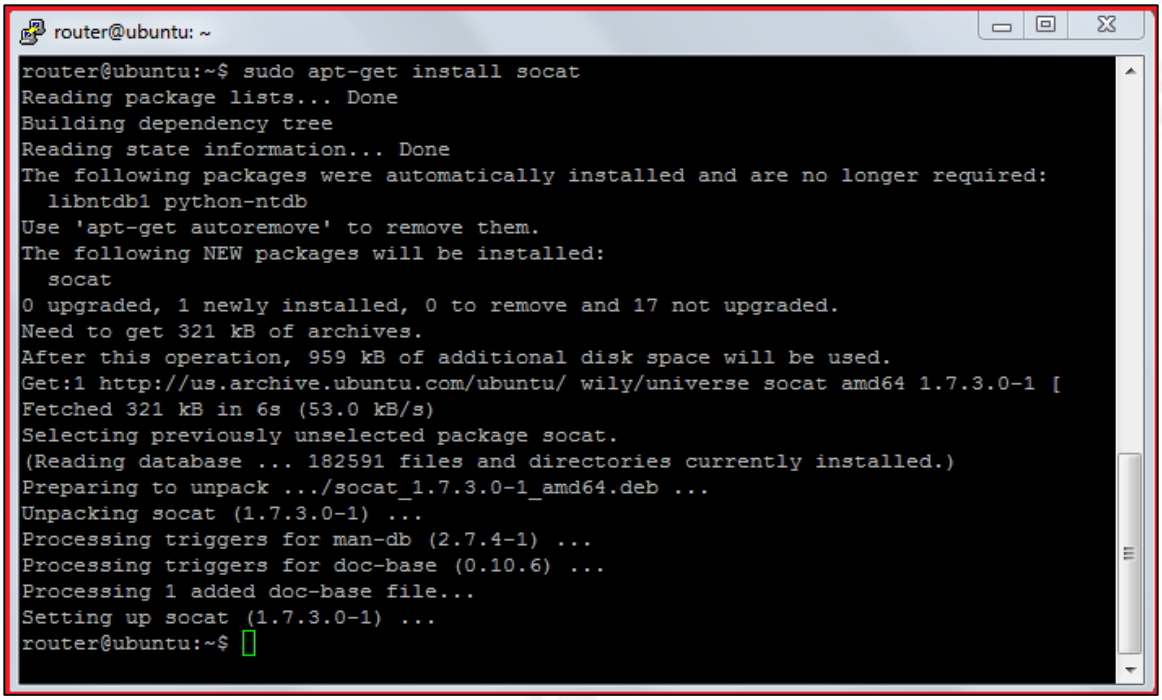


Figure 15: Installation of the socat tool

2. Execution of the socat tool:

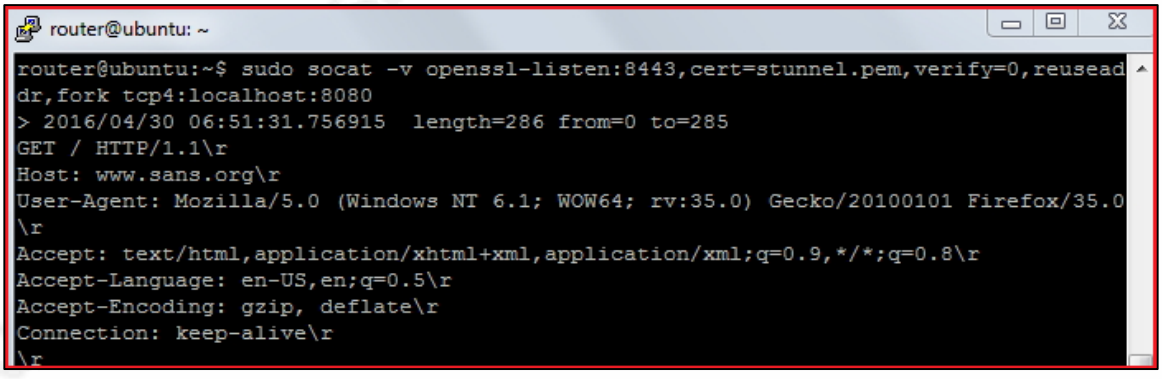


Figure 16: Execution of the socat tool

3. Data sniffed as cleartext:

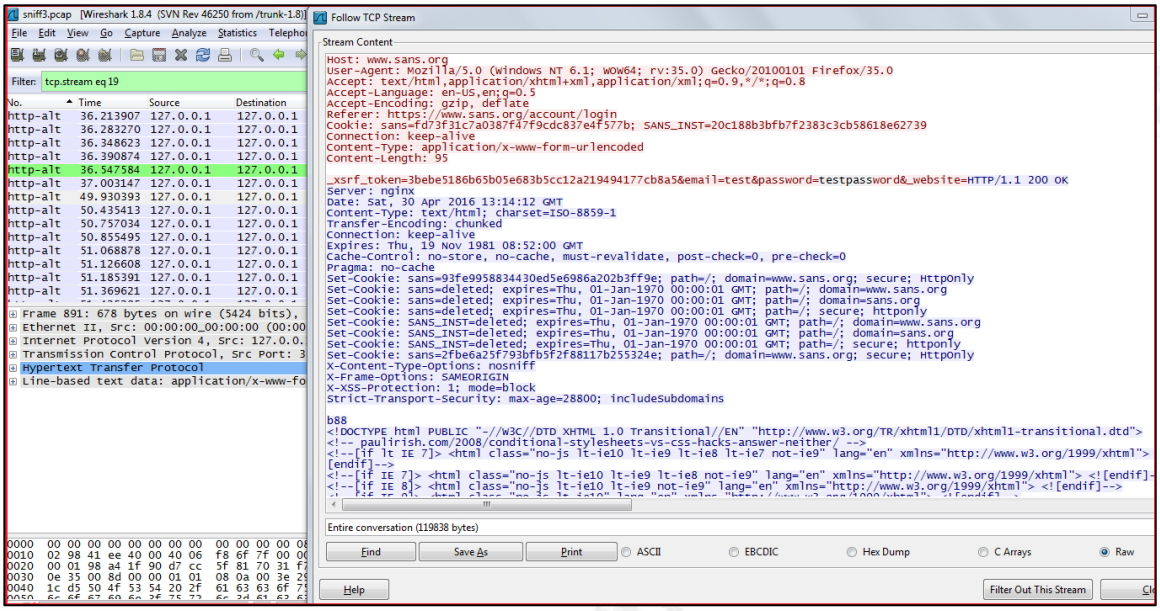


Figure 17: Traffic appears in cleartext

Ettercap Setup:

1. Changes in the Config file:

Sourabh Saxena, sourabhsaxena25@gmail.com

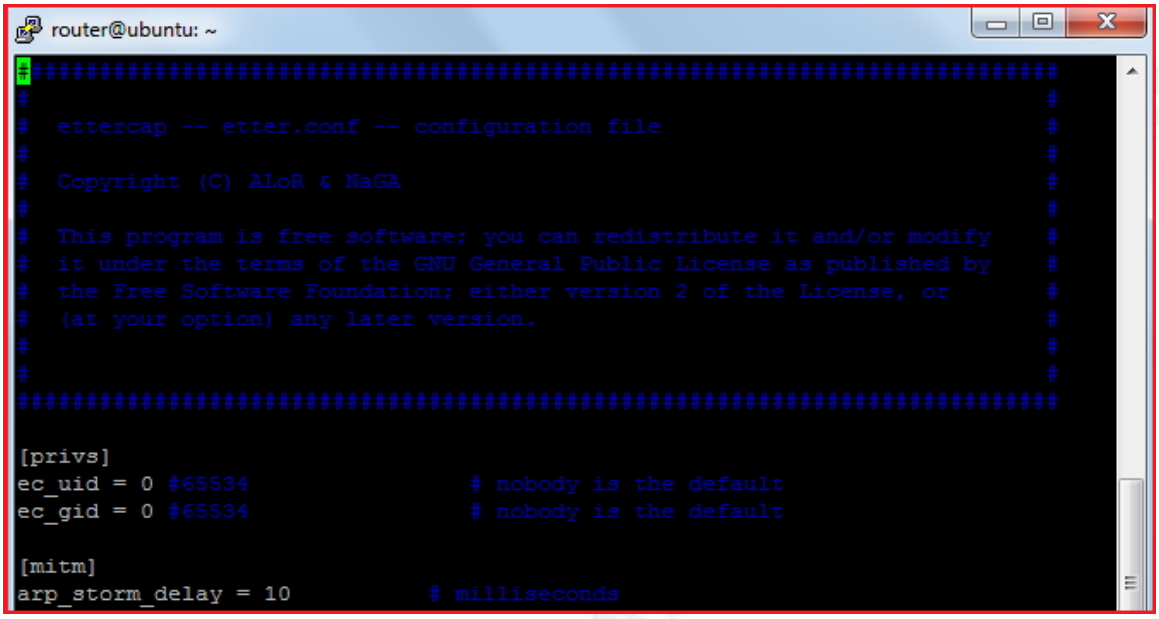


Figure 17: Setting the value of uid and gid to zero

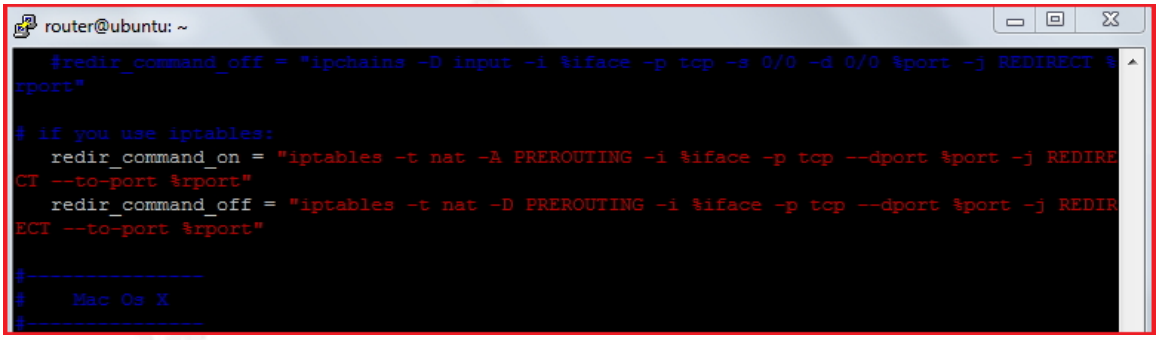


Figure 18: Enabling the firewall rules

2. Execution of Ettercap in ARP mode (192.168.1.8 - Infected Machine):

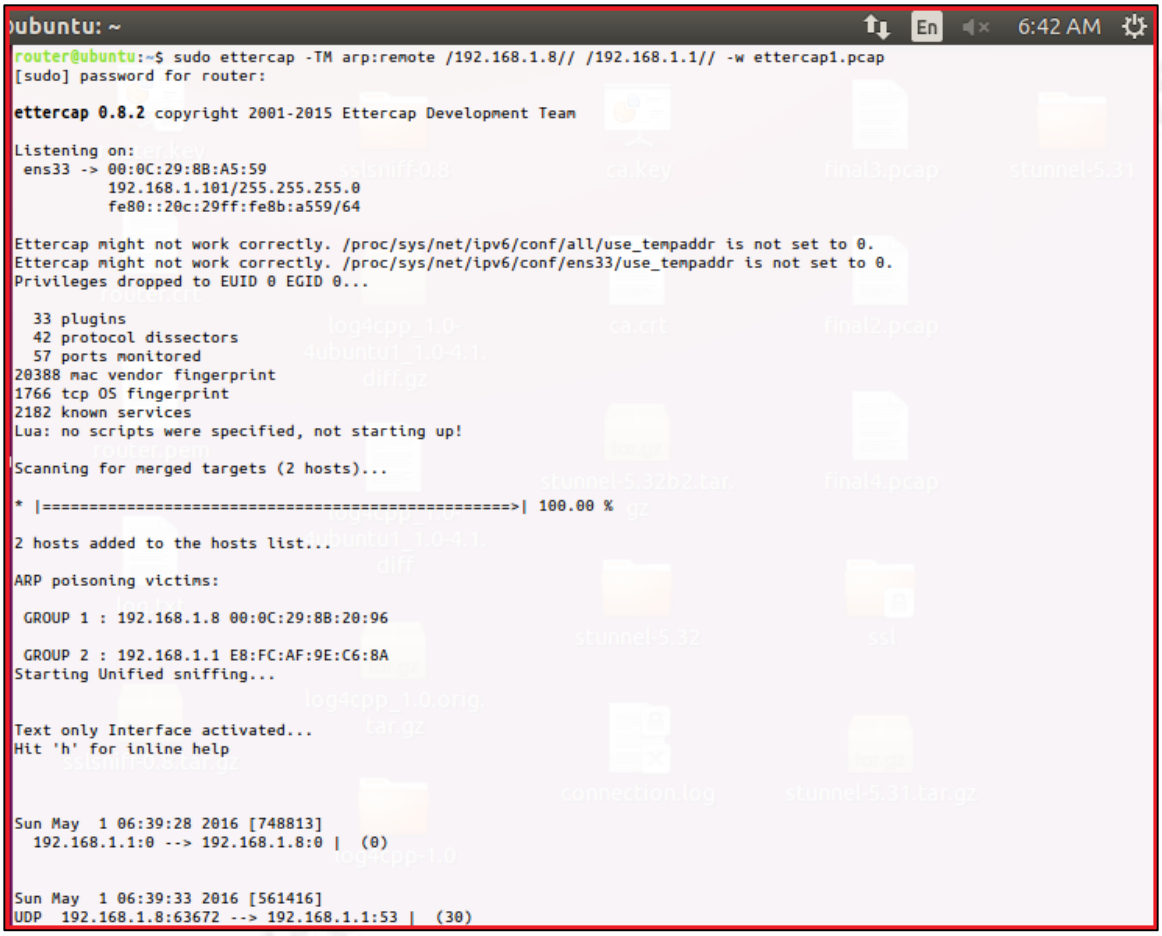


Figure 19: Ettercap in ARP mode

3. SSL traffic in cleartext format and the IP address of the router:

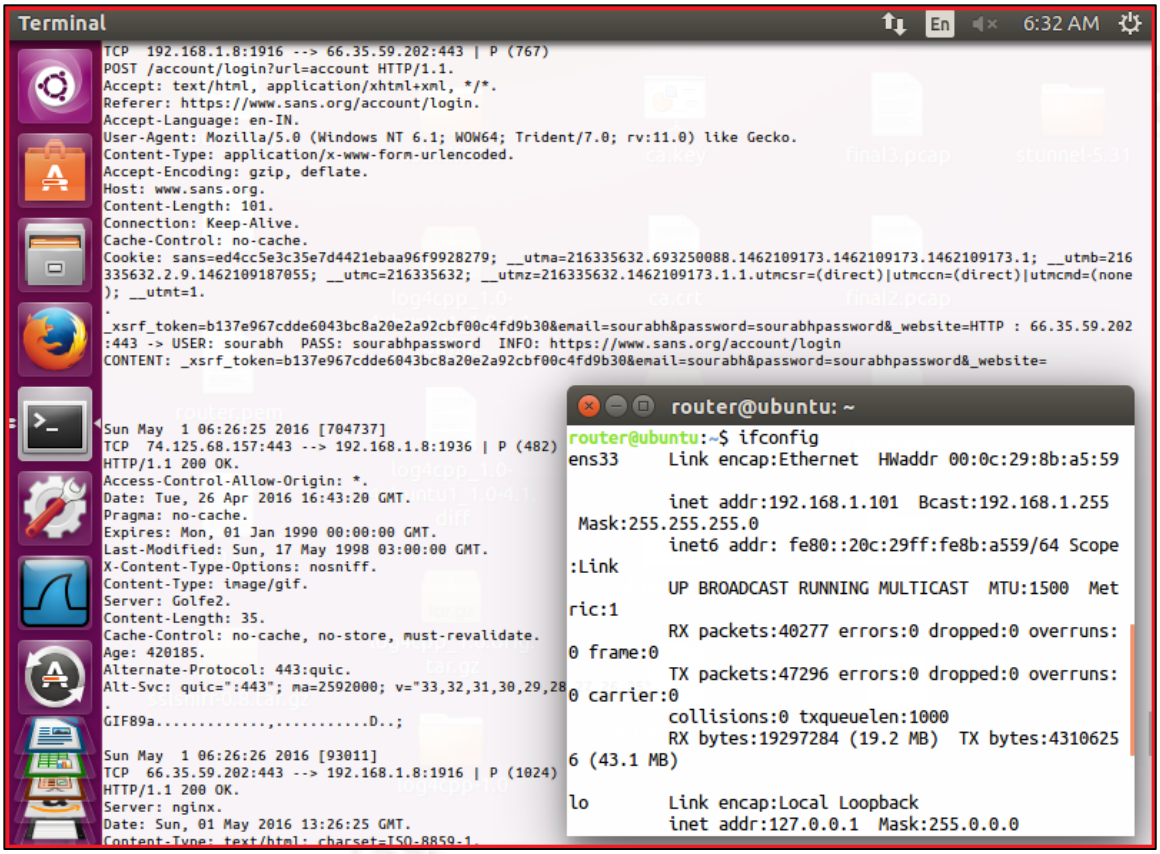


Figure 20: Traffic in cleartext displayed in the console

4. Redirection of the above cleartext traffic to a file:

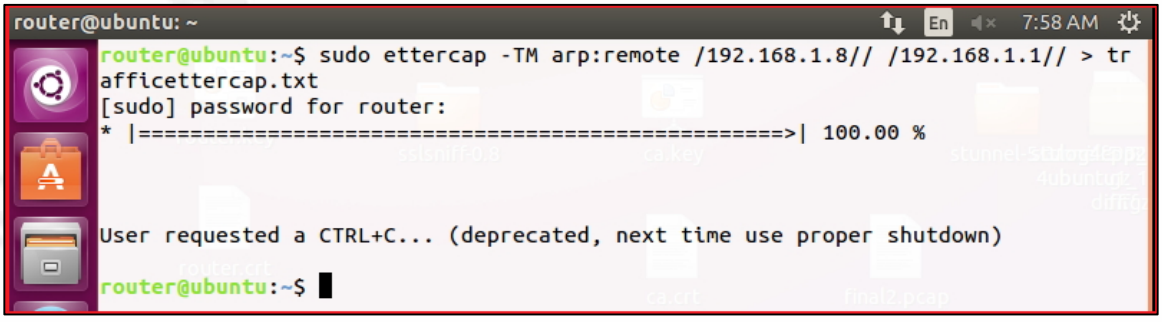


Figure 21: Redirection of the tool's output to a file

Sourabh Saxena, sourabhsaxena25@gmail.com

Appendix - G

```
with open('/etc/bind/blockeddomains.conf', 'r') as file :
  confdata = file.read()

# Removing of unwanted strings in the file
confdata = confdata.replace('zone "', '')
confdata = confdata.replace('" IN {type master; file "/etc/bind/blockeddomains.zone";};', '. IN A 162.213.37.220')

# Write the blockeddomains.zone file and create a proper format of blockeddomains.zone file
stamp=datetime.datetime.now().strftime("%Y%m%d")
with open('/etc/bind/blockeddomains.zone', 'w') as file:
  file.write('$TTL    1000\n@       IN      SOA     yourdnsserveroranynonexistingdomain25343455354fg45f345ef4.com. admin (\n'+
  stamp+' ; serial\n3 ; refresh\n15 ; retry\n1 ; expiry\n1  ; minimum\n)\n@       IN NS   ns1\n@       IN NS   ns2\n\nns1 IN
  A 192.168.137.27\nns2 IN A 192.168.137.27\n\n')
  file.write(confdata)
# below line will help in blocking www as well as subdomains of blocked domains.
  file.write('* IN A 162.213.37.220\n')
```

automatebindubuntu.py

automatebindubuntu.py

```
import urllib.request, sys, os, datetime

#Download the file from malwaredomains.com
urllib.request.urlretrieve("http://mirror2.malwaredomains.com/files/spywaredomains.zones", "/etc/bind/blockeddomains.conf")

#Remove first 11 lines out of the file
lines = open('/etc/bind/blockeddomains.conf').readlines()
open('/etc/bind/blockeddomains.conf', 'w').writelines(lines[11:])

# Read the '/etc/bind/blockeddomains.conf' file
filedata = None
with open('/etc/bind/blockeddomains.conf', 'r') as file :
  filedata = file.read()

# Now, replace the path to '/etc/bind/blockeddomains.zone' which currently is '/etc/namedb/blockeddomain.hosts' within the
blockeddomain.conf file for all the zones
filedata = filedata.replace('"  {type master; file "/etc/namedb/blockeddomain.hosts";}; ', '" IN {type master; file
"/etc/bind/blockeddomains.zone";};')

# Write the file out again
with open('/etc/bind/blockeddomains.conf', 'w') as file:
  file.write(filedata+'\n')

# Read the '/etc/bind/blockeddomains.conf' file to create a 'blockeddomains.zone' file
confdata = None
with open('/etc/bind/blockeddomains.conf', 'r') as file :
  confdata = file.read()

# Removing of unwanted strings in the file
confdata = confdata.replace('zone "', '')
```

Sourabh Saxena, sourabhsaxena25@gmail.com

Appendix - H

Network Interfaces for the Router:

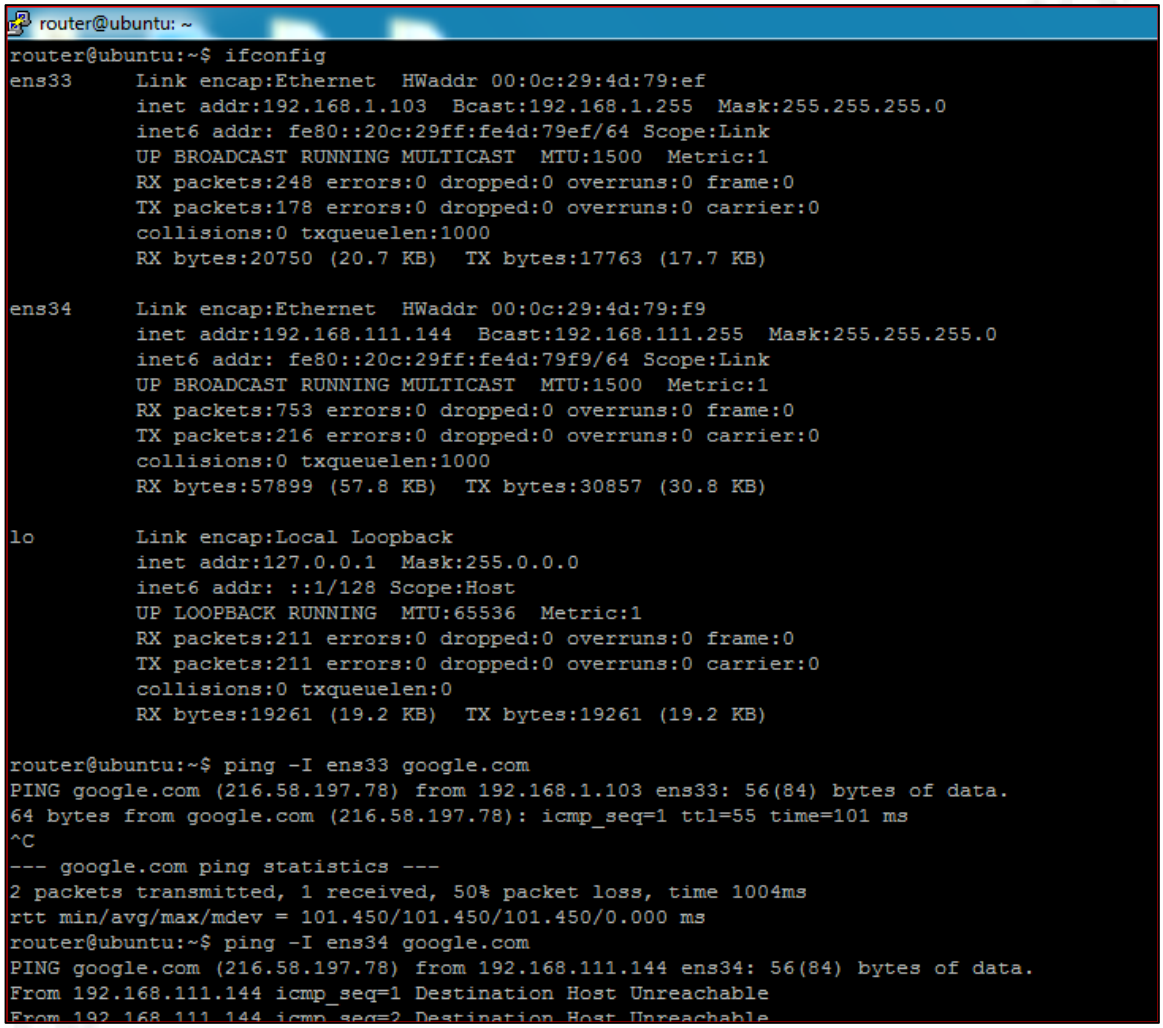


Figure 1: Network interfaces

1. IP forwarding in Router:

Sourabh Saxena, sourabhsaxena25@gmail.com

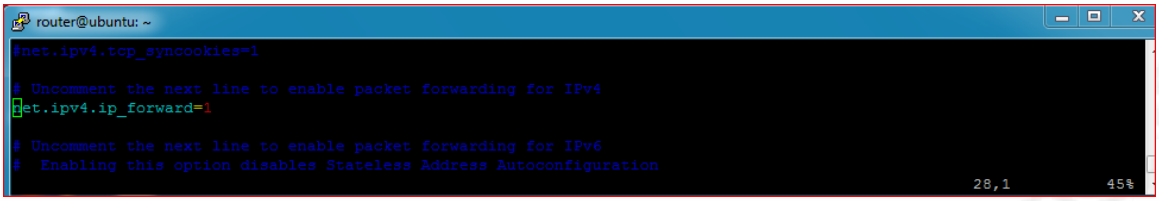


Figure 2: IP Forwarding

2. Firewall Rules:

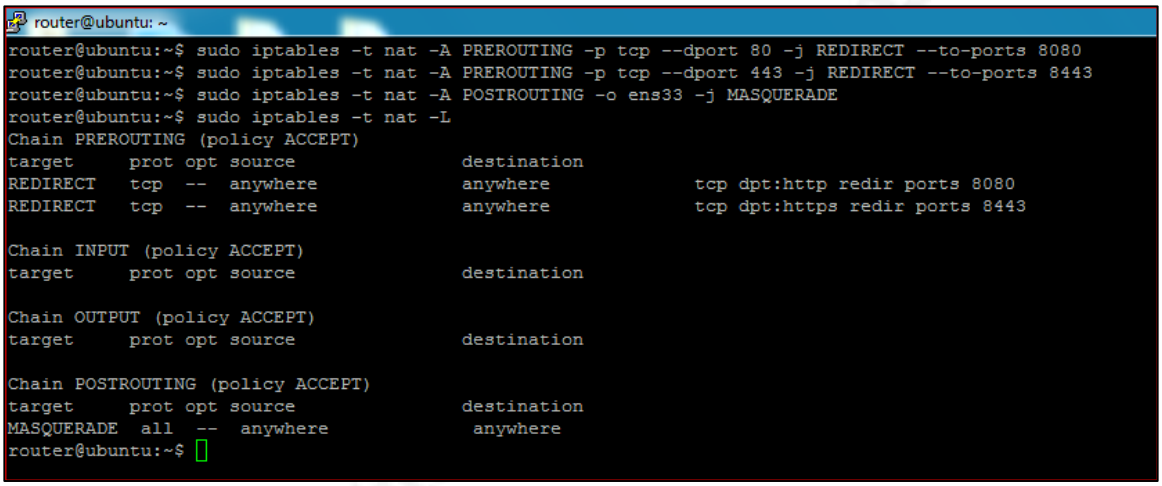


Figure 3: Rules for iptables

Sourabh Saxena, sourabhsaxena25@gmail.com