Reading the tcpdump traffic log

```
14:18:32.192571 IP your.machine.52444 > dns.google.domain: 35084+ A?
yummyrecipesforme.com. (24)

14:18:32.204388 IP dns.google.domain > your.machine.52444: 35084
1/0/0 A 203.0.113.22 (40)
```

The first section of the DNS & HTTP traffic log file shows the source computer (**your.machine.52444**) using port **52444** to send a DNS resolution request to the DNS server (**dns.google.domain**) for the destination URL (**yummyrecipesforme.com**). Then the reply comes back from the DNS server to the source computer with the IP address of the destination URL **(203.0.113.22**).

```
14:18:36.786501 IP your.machine.36086 > yummyrecipesforme.com.http:
Flags [S], seq 2873951608, win 65495, options [mss 65495,sackOK,TS
val 3302576859 ecr 0,nop,wscale 7], length 0

14:18:36.786517 IP yummyrecipesforme.com.http > your.machine.36086:
Flags [S.], seq 3984334959, ack 2873951609, win 65483, options [mss
65495,sackOK,TS val 3302576859 ecr 3302576859,nop,wscale 7], length
0
```

The next section shows the source computer sending a connection request (**Flags [S]**) from the source computer (**your.machine.36086**) using port **36086** directly to the destination (**yummyrecipesforme.com.http**). The **.http** suffix is the port number; **http** is commonly associated with port 80. The reply shows the destination acknowledging it received the connection request (**Flags [S.]**). The communication between the source and the intended destination continues for about 2 minutes, according to the timestamps between this block (**14:18**) and the next DNS resolution request (see below for the **14:20** timestamp).

TCP Flag codes include:

Flags [S] - Connection **S**tart
Flags [F] - Connection **F**inish
Flags [P] - Data **P**ush
Flags [R] - Connection **R**eset
Flags [.] - Acknowledgment

```
14:18:36.786589 IP your.machine.36086 > yummyrecipesforme.com.http:
Flags [P.], seq 1:74, ack 1, win 512, options [nop,nop,TS val
3302576859 ecr 3302576859], length 73: HTTP: GET / HTTP/1.1
```

The log entry with the code **HTTP: GET / HTTP/1.1** shows the browser is requesting data from **yummyrecipesforme.com** with the **HTTP: GET** method using **HTTP** protocol version **1.1**. This could be the download request for the malicious file.

```
14:20:32.192571 IP your.machine.52444 > dns.google.domain: 21899+ A?
greatrecipesforme.com. (24)

14:20:32.204388 IP dns.google.domain > your.machine.52444: 21899
1/0/0 A 192.0.2.172 (40)

14:25:29.576493 IP your.machine.56378 > greatrecipesforme.com.http:
Flags [S], seq 1020702883, win 65495, options [mss 65495,sackOK,TS
val 3302989649 ecr 0,nop,wscale 7], length 0

14:25:29.576510 IP greatrecipesforme.com.http > your.machine.56378:
Flags [S.], seq 1993648018, ack 1020702884, win 65483, options [mss
65495,sackOK,TS val 3302989649 ecr 3302989649,nop,wscale 7], length
0
```

Then, a sudden change happens in the logs. The traffic is routed from the source computer to the DNS server again using port **.52444** (**your.machine.52444 > dns.google.domain**) to make another DNS resolution request. This time, the DNS server routes the traffic to a new IP address **(192.0.2.172)** and its associated URL (**greatrecipesforme.com.http**). The traffic changes to a route between the source computer and the spoofed website (outgoing traffic: **IP your.machine.56378 > greatrecipesforme.com.http** and incoming traffic: **greatrecipesforme.com.http > IP your.machine.56378**). Note that the port number (**.56378**) on the source computer has changed again when redirected to a new website.

Resources for more information

- An introduction to using tcpdump at the Linux command line: Lists several tcpdump commands with example output. The article describes the data in the output and explains why it is useful.
- tcpdump Cheat Sheet: Lists tcpdump commands, packet capturing options, output options, protocol codes, and filter options
- What is a computer port? | Ports in networking: Provides a short list of the most common ports for network traffic and their associated protocols. The article also provides information about ports in general and using firewalls to block ports.
- Service Name and Transport Protocol Port Number Registry: Provides a database of port numbers with their service names, transport protocols, and descriptions
- How to Capture and Analyze Network Traffic with tcpdump?: Provides several tcpdump commands with example output. Then, the article describes each data element in examples of tcpdump output.
- Masterclass – Tcpdump – Interpreting Output: Provides a color-coded reference guide to tcpdump output

A **DNS (Domain Name System) resolution request** is a process initiated by a client system to obtain the IP address associated with a specific domain name. When you enter a human-readable domain name (like www.example.com) into a web browser or any network-aware application, the system needs to translate that domain name into an IP address that can be used to locate the corresponding server on the Internet.

Here is a simplified overview of how DNS resolution works:

1) **User Input**: You type a domain name into your web browser (e.g., www.example.com).

2) **Local DNS Cache Check**: Your computer checks its local DNS cache to see if it already has the IP address for the requested domain. If the information is found and is still valid (not expired), the system uses it without making a new request to DNS servers.

3) **Recursive DNS Server**: If the information is not in the local cache or has expired, your computer contacts a recursive DNS server. This DNS server is responsible for obtaining the information on your behalf.

4) **Root DNS Server**: If the recursive DNS server doesn't have the information, it queries root DNS servers. These servers provide information about the Top-Level Domains (TLDs) like .com, .org, .net, etc.

5) **TLD DNS Server**: The recursive DNS server then queries the DNS server responsible for the specific TLD of the requested domain. For example, if the domain is www.example.com, the TLD server for '.com' is queried.

6) **Authoritative DNS Server**: The TLD server directs the recursive DNS server to the authoritative DNS server for the actual domain (example.com). This server holds the specific IP address associated with the domain.

7) **Response**: The authoritative DNS server responds to the recursive DNS server with the IP address.

8) **Caching**: The recursive DNS server caches the IP address locally for a specified time (TTL - Time to Live) and returns the IP address to the client system.

9) **Final Response**: The client system receives the IP address and uses it to establish a connection with the desired server.

This entire process is known as DNS resolution, and DNS resolution requests play a crucial role in translating human-readable domain names into machine-readable IP addresses on the Internet.