EECE 655: Assignment 1

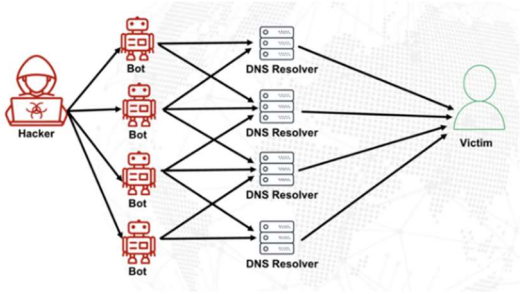
DDoS: DNS Amplification Attack, Detection, & Prevention

By: Reem Arnaout

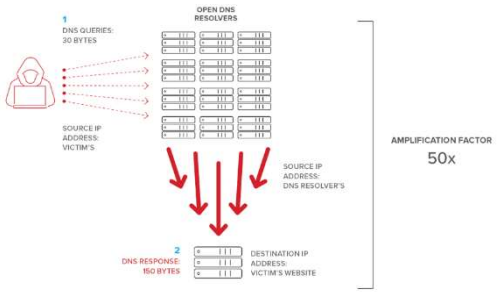
Fall 2024-2025

1. Description:

Definition: DNS amplification is a type of distributed denial of service (DDoS) attack that exploits the open nature of DNS servers to overwhelm a target with traffic. By sending DNS queries with a spoofed IP address (pretending to be the victim's address), attackers can trick DNS servers into sending large responses to the victim. This method is particularly dangerous because it uses amplification; the attacker sends small queries but receives much larger responses, overwhelming the victim's resources. Amplification attacks became prominent in the early 2010s, with major incidents in 2013 targeting banks and governments. Unlike DNS reflection attacks, which simply forward DNS queries to a victim, DNS amplification multiplies the attack's power by exploiting DNS servers' ability to return much larger responses, making it significantly more damaging.



Source: Vercara



Source: CyberHoot

Attack Code: In the provided attack.py code, the script performs DNS amplification by sending spoofed DNS queries to multiple DNS servers using the Scapy library. The code first reads input arguments, including the victim's IP address (spoofed IP), target port (usually 53 for DNS), and a file containing DNS server IPs. It then reads the DNS server IPs from the provided text file and uses a predefined list of domains (such as google.com) to query. The script sends a large number of DNS queries to each DNS server, with each query using the ANY type to maximize response size, causing amplification. The requests are sent in parallel using Python's threading module, allowing multiple DNS servers to be targeted simultaneously. The script also tracks the number of requests sent and the size of the responses, printing updates every 10 seconds. Once a predefined request limit is reached, the script terminates, simulating an attack that ends after a fixed number of requests. This approach is efficient in overwhelming the victim with a high volume of traffic using minimal attacker resources. In normal conditions, the attack code would be run on multiple dummy machines to increase damage until the attacker decides to stop.

Detect+Counter Code: The detect+counter.py script focuses on detecting and mitigating DNS amplification attacks by monitoring network traffic for DNS responses and IP spoofing attempts. The script uses Scapy to capture DNS traffic, tracking the number of DNS requests and responses. It maintains a list of legitimate DNS requests sent by the Ubuntu machine and compares incoming DNS responses to this list. If a DNS response is received but no corresponding request was sent by the Ubuntu machine, the script issues a warning about potential IP spoofing. It also monitors the volume of DNS responses; if the number exceeds a predefined threshold, it raises warnings for a high volume of DNS responses, indicating a potential amplification attack. If the total number of DNS packets exceeds a set limit, the script shuts down the network interface to block further DNS traffic, preventing the attack from continuing. This detection mechanism helps mitigate DNS amplification attacks by monitoring suspicious patterns, such as high response volumes and IP spoofing, while offering an immediate response by blocking traffic once a critical threshold is reached.

2. Instructions:

Requirements: At least one updated linux virtual machine like Kali or Ubuntu.

Prerequisites (if not present):

On the linux machines:

1. sudo apt update && sudo apt upgrade -y
2. sudo apt install python3 python3-scapy

Running the codes:

1. Run VirtualBox/VMware as administrator
2. Move the detection code to the victim machine and move the attack code + text file to the attacker machines or VScode. Navigate to the directory where they're placed.
3. Get the victim's IP using ifconfig and check the network interface. Enter the victim's IP when asked.
4. Run wireshark on victim machine and set the filter to dns: sudo wireshark
5. In another terminal, run the detection code using: sudo python3 detect+counter.py
6. Run the attacker code: On VScode: python attack.py 53 dnsservers.txt
   On Linux: sudo python3 attack.py 53 dnsservers.txt
7. Watch wireshark to see the DNS packets flow in.
8. The code will terminate on its own, but if needed to terminate before: On Linux: ctrl+c On VScode: kill terminal
9. Don't forget to put the network interface back up as the terminal instructs: sudo ifconfig enp0s3 (or eth0) up
10. Stop wireshark using ctrl+c in the first terminal.

3. Prevention Methods:

- Ensure server configurations are secure and up to date.
- Disable recursive query support on public servers and configure firewall rules to block incoming requests from known attack sources.
- Apply rate-limiting policies to limit the number of requests allowed from a single IP address or subnet to prevent overwhelming traffic from one source.
- Implement Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) which can block DNS amplification attacks by identifying specific queries or responses with certain flags set showing spoofed source addresses.

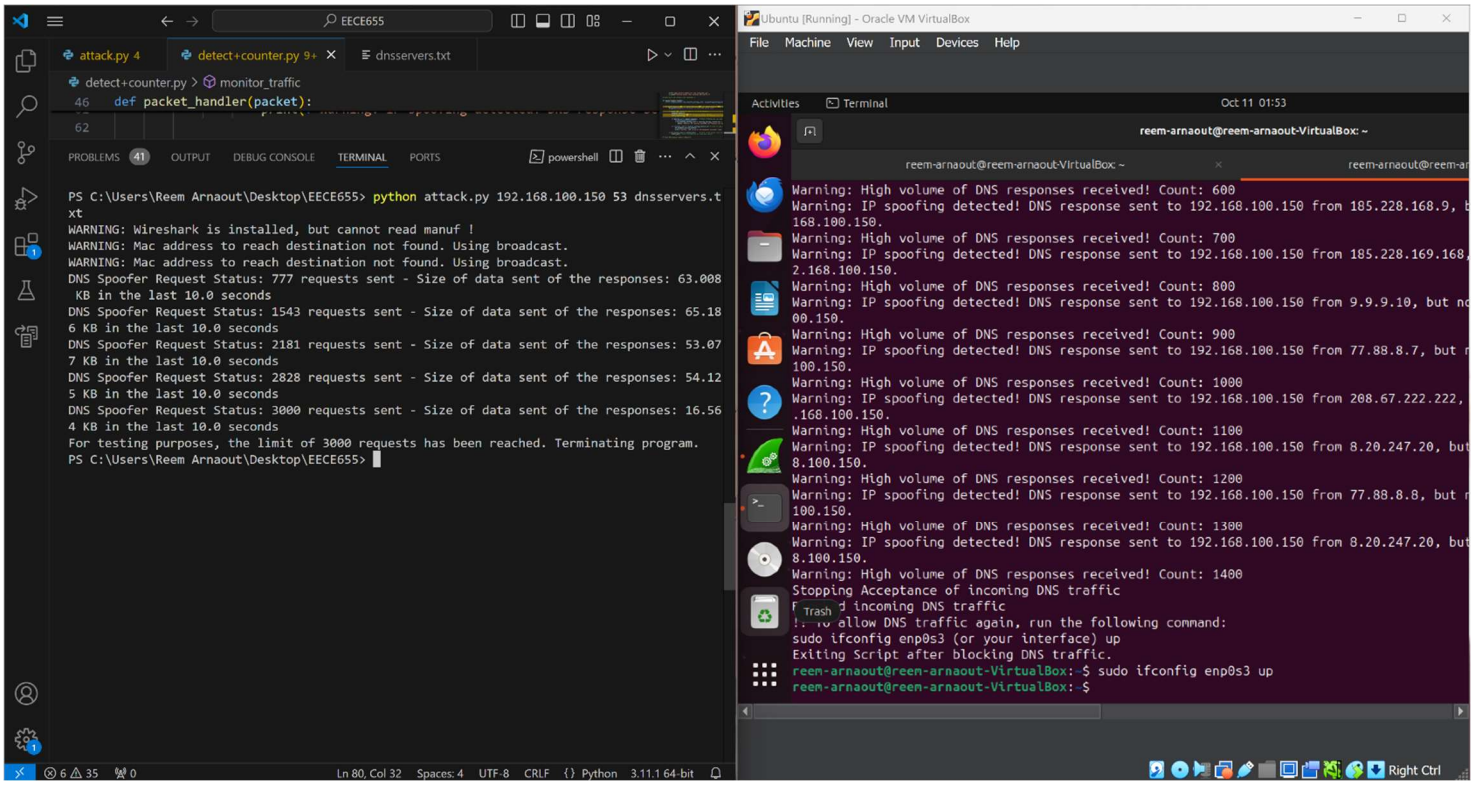
Source: Heimdal®

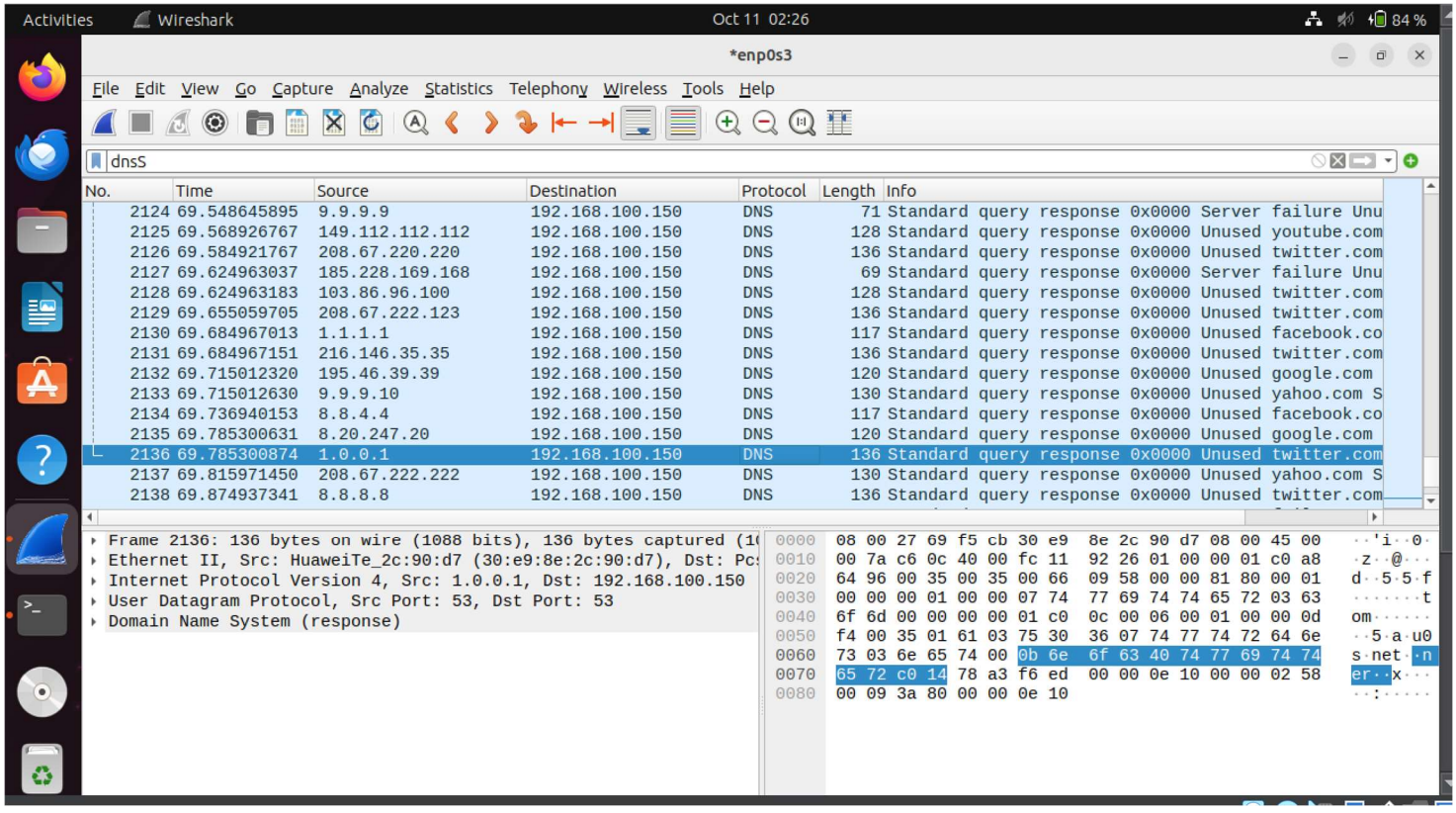
Side note:

The demo shows that when running the detector code it will say "monitoring TTL". That is because monitoring TTL values can help detect IP spoofing because each packet's TTL decreases as it travels through the network. If a spoofed packet has a TTL that doesn't match the expected value based on its origin, it could indicate foul play.

I did implement this in the detector code at first but then removed it because all the responses had the same TTL since the attack code and detect code are being run on the hardware. So, essentially, it was useless in this case.

4. Screenshotted Results:





Note that my assigned partner wasn't able to contribute as he was affected by tough circumstances and couldn't reach me. So, I worked alone.

Thank you.