Técnicas de Perceção de Redes
Network Awareness

DDoS Detection and Counter-Actions

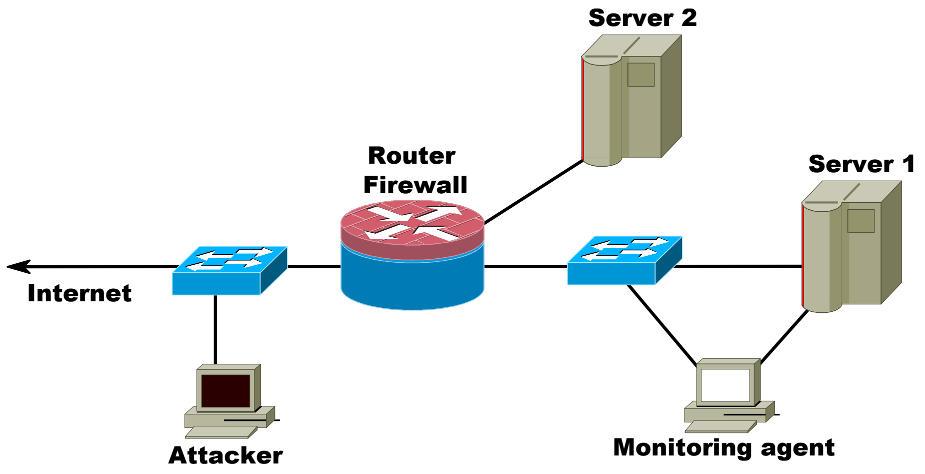
Objectives

- Attack detection.
- Illicit clients detection.
- Blocking of illicit clients.
- Intelligent load balancing (licit users isolation).

DDoS Attack detection and (il)licit clients identification

1. Configure a Linux server (Server 1) with an HTTP server (apache2) and a DNS server (bind9). Add multiple contents to apache2 server (e.g., *login.html, authorized.html, non-authorized.html, page1.html, page2.html*, etc…). Add multiple domain names configurations to bind9 server.

The Router/Firewall can be replaced by a simple router, and Server 2 and the Monitoring Agent PC do not to be implemented yet.



TCP

2. Based on the provided script baseDDoSdet.py, at the server, develop a monitoring rule to identify IP addresses with established or pending establishment TCP sessions above predefined alarm thresholds.

To make a test attack use *hping3* (and *iptables* to block attacker's kernel sent RST TCP packets):

- Use: `iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP`
- Use: `hping3 <ip_address> -S -p 80 -c 10`

Note: use `iptables -L` and `iptables -F`, to list and reset/flush *iptables* rules, respectively.

3. Improve the previously defined identifying rules, by monitoring the access contents at the server by each user to identify licit users.

Note: you may assume that a 200 OK status code for page *authorized.html* only appears for correctly validated users.

Note2: use `wget` or `curl` to perform HTTP queries with the terminal.

4. Based on the provided script baseDDoSdet.py and on the decisions defined before, create blocking rules at Server 1 (using iptables) that block and respond with a TCP RST packet to all TCP connections from an infringing IP address.

```
iptables -I INPUT -s <ip_address> -p tcp -j REJECT --reject-with tcp-reset
```

Retest the attack.

UDP

5. At Server 1, develop a monitoring rule to identify IP addresses with UDP/DNS requests above predefined alarm thresholds.

To make a test attack use *nslookup* and *hping3:*

- Use: `nslookup <domain> - <DNS_ip_address>` to generate multiple DNS queries.
- Use: `hping3 <DNS_ip_address> -2 -p 53 -c 10` to generate multiple (10) connections to port UDP 53.

6. Improve the previously defined identifying rules, by verifying if offending IP addresses are DNS relay servers.

You may assume different alarm thresholds for clients and relay servers.

Note: use NMAP to probe remote IP addresses (look for port UDP 53 open: `nmap -sU <ip_address> -p 53`).

7. Based on the provided script baseDDoSdet.py and on the decisions defined before, create blocking rules at Server 1 (using iptables).

```
iptables -I INPUT -s <ip_address> -p UDP -j DROP
```

Retest the attack.

Centralized Monitoring, Control, and Filtering

8. Create a monitoring agent, connected to servers by an independent connection (VLAN or secondary servers' port). Modify/extend your software to allow the remote monitoring of multiple servers (e.g., via SSH).

9. Deploy and configure a firewall (you may use a Cisco ASA, a Linux machine with iptables or a Linux router/Firewall like VyOS). Configure the monitoring agent to remotely apply all the protection rules (e.g., via SSH).

(Intelligent) Load balancing

10. Deploy and configure Server 2 (with apache2). At the firewall, redirect all licit users to protected Server 2 (assume that authentication is common and known by both servers).
See http://jensd.be/343/linux/forward-a-tcp-port-to-another-ip-or-port-using-nat-with-iptables