Relatório Trabalho 2 - Laboratório de redes de computadores

Deivid Santos

1. Topologia

Foi escolhida uma topologia utilizando 6 máquinas conectadas por um switch, sendo uma delas o atacando e uma a vítima. A máquina atacante pode ser qualquer uma das máquinas exceto a vítima. A vítima deve estar rodando o código do trabalho e somente pode ser a máquina com IP 10.0.0.12 pois existe uma validação no código.

Os IP's foram definidos de forma dinâmica, criados diretamente pelo CORE e o IPv6 foi removido.

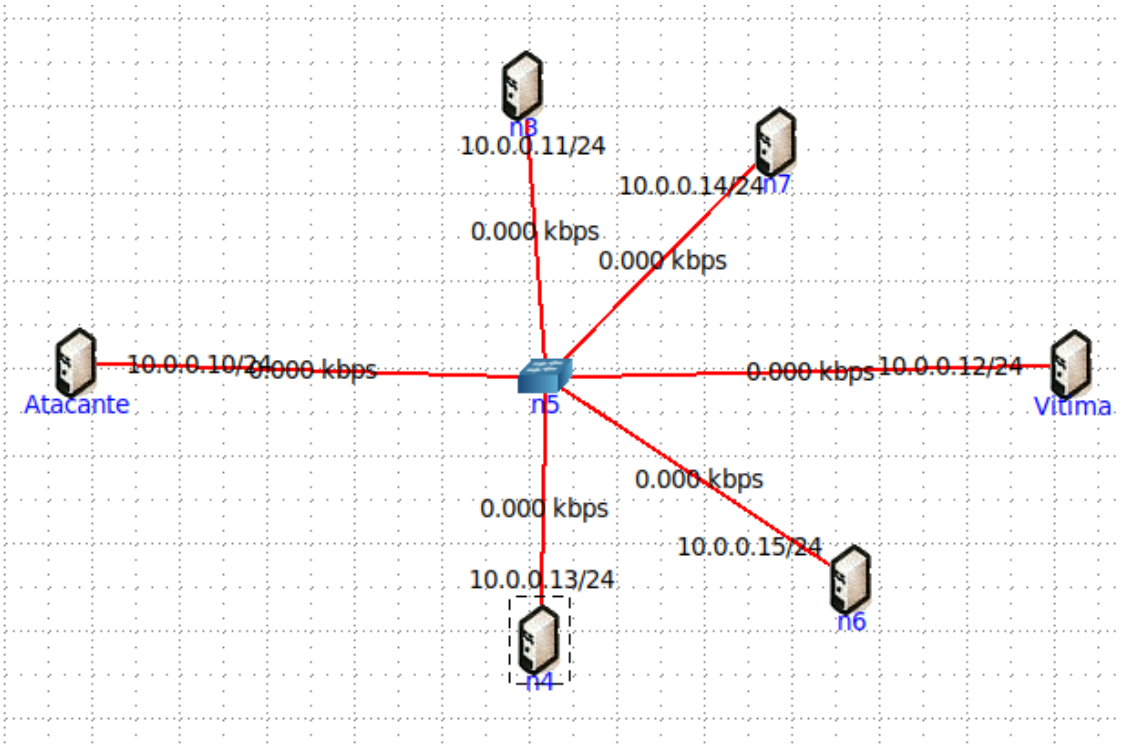

Figura 1 - Topologia

2. Identificação do ataque

O código somente pode rodar máquina vítima que possui IP 10.0.0.12. Esse IP é usado para filtrar os pings recebidos que tenham somente o próprio IP como destino. Antes desse filtro todos os pings estavam sendo contabilizados, inclusive os enviados pela própria máquina vítima e gerando um loop infinito de contra ataques.

```
local_ip = "10.0.0.12"
#Mapa de lista onde primeiro item é o contador de
# Inicia variável com IP local para poder atacar c
ipList = {local_ip: [0, datetime.datetime.now()]}
```

Figura 2 - Estrutura

A estrutura principal definida é um dicionário inicializado com o IP da própria vítima para ser usado no contra ataque e que possui como chave os IP's das máquinas que mandam um ping. O valor é uma lista com dois itens que são usados para saber se a vítima está recebendo um ataque efetivamente, sendo eles:

Posição 0: Contador de Pings recebidos

Posição 1: Horário em que o ping foi recebido

Cada vez que um ping for recebido por algum IP o contador é incrementado em 1 e caso o IP não exista na lista ele será adicionado com contador inicial em 0. O horário em que o ping foi recebido é usado em uma parte da validação de ataque para saber se o ataque está rápido o suficiente para ser considerado um ataque, essa velocidade é definida pelo delta que por padrão é 5 segundos. Caso o ping seja enviado depois de 5 segundos, então o contador é resetado e não é mais considerado ataque.

```
# Necessário saber quanto tempo passou desde o ultimo ping
# Caso maior que o delta (5 segundos) está lento o ping e é desconsiderado, resetando o contador
# Caso menor que o delta (5 segundos) é considerado possivel atacante
if s_addr not in ipList or datetime.datetime.now() > ipList[s_addr][1] + datetime.timedelta(seconds=5):
    # Criando novo IP conhecido ou resetando um IP pois não é atacante
    ipList[s_addr] = [0, datetime.datetime.now()]
else:
    # Contador de ataques caso esteja recebendo ping com diferença de tempo menor que o delta
    ipList[s_addr] = [ipList[s_addr][0] + 1, datetime.datetime.now()]

# Caso o contador seja maior 5 com o intervalo entre os pings menor que o delta, então é um atacante
if ipList[s_addr][0] > 5:
    counter_attack(s_addr)
```

Figura 3 - Identificação do ataque

3. Contra ataque

Ao identificar um ataque iniciamos o processo de contra ataque utilizando outro socket em cada ping recebido do atacante, montando um icmp echo request manualmente utilizando o endereço do ip do atacante como origem e enviando para todos IP's que estão na ipList exceto o IP atacante.

```
ip_saddr = socket.inet_aton(attacker_ip)
ip_daddr = socket.inet_aton(ip)

ip_ihl_ver = (ip_ver << 4) + ip_ihl

# Pack IP header fields
ip_header = struct.pack("!BBHHHBBH4s4s", ip_ihl_ver, ip_tos, ip_tot_len, ip_id, ip_frag_off, ip_ttl,
                        ip_proto, ip_check, ip_saddr, ip_daddr)

#########################

# Destination IP address
dest_ip = ip
dest_addr = socket.gethostbyname(dest_ip)

# Send icmp_packet to address = (host, port)
print("counterattacking with ip: " + ip)
sender_socket.sendto(ip_header + icmp_packet, (dest_addr, 0))
```

Figura 4 - Contra ataque modificando o ip de origem.

O código possui comentários com alguns detalhes da implementação de acordo com o relatório.