**Eric Friedrich, Marcelo Holgado**

**Redes de Computadores II Eng. de Computação**

**DHCP Spoofing**

**Attack man-in-the-middle**

**Porto Alegre 2020/1**

**Sumário**

[**Introdução ao método proposto** 3](#_Toc44408527)

[**O protocolo DHCP e servidores** 3](#_Toc44408528)

[**O protocolo DNS e Servidores** 4](#_Toc44408529)

[**Desenvolvimento do software atacante** 4](#_Toc44408530)

[**Estruturas usadas** 4](#_Toc44408531)

[**Logica do software atacante** 5](#_Toc44408532)

[**SendDHCP** 7](#_Toc44408533)

[**Servidor de DNS Falso** 8](#_Toc44408534)

[**Testes para validar o Servidor DNS e DHCP Spoofing** 8](#_Toc44408535)

[**Bibliografia** 9](#_Toc44408536)

[**Referências** 9](#_Toc44408537)

# **Introdução ao método proposto**

Um dos protocolos mais usados em redes é protocolo DCHP (Dynamic Host Configuration Protocol), ele está em quase todos os parelhos que se conectam à internet. O crescimento do número de máquinas em uma rede fez do DHCP um protocolo bastante usado hoje em dia, pois não há necessidade de fixar um ip em uma máquina e evita que ocorra duplicação de IPs em uma rede. O presente trabalho propõe uma tentativa de atacar uma máquina que esteja querendo se comunicara através do protocolo DHCP, assim validado a eficácia do protocolo.

O Spoofing é a tentativa de falsificar um servidor de DHCP e por sua vez passar um IP, Gateway e DNS falsos para a máquina solicitante. Quando ocorre de uma máquina aceitar uma falsificação de um servidor DHCP, ela agora responde na faixa de ip do servidor falso ficando vulnerável a outros tipos de ataques, como proposto no trabalho a resolução de nomes errada servidor de DNS falso**.**

A partir da verificação de que uma máquina não valida a autenticidade de um servidor DHCP que fornece ela um endereço para se comunicar, nos surgem inúmeros problemas. Mesmo que seja necessário estar dentro da rede para forçar este tipo de ataque, fica o questionamento há como garantir que quem acessa a rede não fará o ataque?

# **O protocolo DHCP e servidores**

 É um protocolo de serviço TCP/IP que oferece configuração dinâmica de máquinas, entregando endereços IP de host, máscara, default gateway e servidores DNS. O DHCP surgiu como padrão em outubro de 1993. Um servidor DHCP é onde foi instalado e configurado o serviço DHCP para poder efetuar a distribuição automática dos endereços de IP para diversos sistemas operacionais clientes, tais como: Windows, Linux, Mac e afins.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente O protocolo DHCP ocorre quando uma máquina se conecta há rede, esta máquina então verifica se existe um servidor de DHCP enviando um pacote Broadcast, se existir um servidor ou mais estes respondem com um endereço de rede disponível, então máquina solicitante reponde de volta que vai querer este endereço recebido, o servidor reponde de volta com um ACK

Figura 1: Comunicação do protocolo DHCP

# **O protocolo DNS e Servidores**

O protocolo DNS(Domain Name System) é um servidor que contém um banco de dados com endereço de IP públicos e seus nomes de domínio. Um servidor de DNS é capaz de transformar uma URL em um endereço IP. Ao requisitar um domínio, o pedido é encaminhado ao servidor responsável pelo atendimento de tal, o qual o direcionará para outro servidor que atende a hospedagem do site. No trabalho a máquina que foi conectada irá receber um IP de um servidor DNS que configuramos, o objetivo é que quando máquina fizer uma requisição de um domínio ela será redirecionada para um site que foi pré-escolhido e não o site correto.

# **Desenvolvimento do software atacante**

Para o desenvolvimento do software atacante foi utilizada a linguagem C e algumas bibliotecas para comunicação em rede com raw sockets. Usamos um Buffer (constant char) para receber os pacotes utilizando a função recvfrom(buffer) que fica na biblioteca sys/sockets.h, essa biblioteca já vem nos sistemas do Linux. Para enviar utilizamos a função sendfrom(sdbuffer) da mesma biblioteca sys/sockets.h. Estes buffers que usamos são preenchidos por valores hexadecimais e se imprimíssemos só veríamos um monte de valores que não teriam muito significado. Para resolver isso usamos estruturas (structs) para cada parte do pacote recebido. Por exemplo sabemos que os primeiros campos do pacote contêm o MAC Addrress da máquina destino e o da máquina local. Então apontamos o buffer para essa posição e adicionamos na struct ethernet.

struct ethhdr {

    unsigned char h\_source[6];

    unsigned char h\_dest[6];

    unsigned char h\_proto;

};

struct iphdr {

    uint8\_t tos;

    uint16\_t tot\_len;

    uint16\_t id;

    uint16\_t frag\_off;

    uint8\_t ttl;

    uint8\_t protocol;

    uint16\_t check;

    uint32\_t saddr;

    uint32\_t daddr;

};

struct udphdr {

    u\_short uh\_sport;

    u\_short uh\_dport;

    short   uh\_ulen;

    u\_short uh\_sum;

};

## **Estruturas usadas**

Para o campo ethernet foi usa uma struct chamada ethhdr da biblioteca netinet/ether.h. Para o campo do IPv4 usamos uma struct chamada iphdr que faz parte da biblioteca netinet/ip. E o protocolo UDP tem uma struct udphrd que faz parte da biblioteca netinet/udp.h.

Código 1: Structs usadas como headers

Para o campo DHCP foi utilizada uma struct chamada dhcp\_packet que pertence a biblioteca dhcp.h, essa biblioteca foi criada pela (Internet Systems Consortium), que pode ser encontrada fazendo uma busca na internet. Essa biblioteca permitiu que os campos do protocolo fossem preenchidos o campo DHCP quando se recebe um pacote e quando se envia, facilitando a leitura do pacote.

## **Logica do software atacante**

O primeiro objetivo do software é ficar monitorando a rede por pacotes DHCP Discovery, estes pacotes são broadcast, então quando chegam na aplicação falsificadora ela envia um pacote oferecendo um endereço de IP e informando quem é o servidor de DHCP da rede, neste caso será a máquina que está rodando aplicação falsificadora.

Observando a o trecho de código abaixo Código 2, fica facil entender como fazemos para identificar um pacote DHCP Discovery. Dentro de um while verificamos todos pacotes IPv4 que chegam com protocolo igual a 17 (Protocolo UDP). Se o pacote UDP tem como destino a porta 67, então é um pacote DHCP, executamos a função buscaDHCPType() para verificar qual o tipo de pacote DHCP. Neste caso estamos esperando por um pacote DHCP Discovery.

printf("\nBuscando pacotes DHCP ...\n");

while(1) {

    saddr\_size = sizeof saddr;

    data\_size = recvfrom(sock\_raw , buffer , 65536 , 0 , &saddr , &saddr\_size);

    if(data\_size < 0){

        printf("Error recebendo pacotes\n");

        return 1;

    }

    prepDHCPack();

    if (iph\_recv->protocol == 17) {

        if (ntohs(udph\_recv->dest) == 67) {

            dhpro = buscaDHCPType();

            if (dhpro == 1) {

                printDHCP();

                printf("   |-DHCP Message Type: (Discovery)\n");

                sendDHCP(buffer);

                dhpro = 0;

                break;

            }

        }

    }

}

Código 2: Recebimento de pacotes DHCP Discovery

O segundo objetivo da aplicação é verificar se a máquina que está sendo atacada aceitará o endereço oferecido, para isto a aplicação fica monitorando a rede por pacotes DHCP Requests. Quando estes pacotes chagam esta envia um pacote de DHCP ACK para finalizar a comunicação com a máquina atacada. Após a segunda etapa a máquina atacada estará com endereço que a aplicação enviou DNS e Gateway.

Como podemos notar a parte que recebe pacotes DHCP Discovery e Release são parecidas pois para diferenciar um pacote DHCP Discovery para um Request, temos que olhar para o campo option header. A função responsável por diferenciar os pacotes é buscaDHCPType().

Código 3: Recebimento de pacotes DHCP Request

while(1) {

    saddr\_size = sizeof saddr;

    data\_size = recvfrom(sock\_raw , buffer , 65536 , 0 , &saddr , &saddr\_size);

    if(data\_size < 0){

        printf("Error recebendo pacotes\n");

        return 1;

    }

    prepDHCPack();

    if (iph\_recv->protocol == 17) {

        if (ntohs(udph\_recv->dest) == 67) {

            dhpro = buscaDHCPType();

            if (dhpro == 3) {

                printDHCP();

                printf("   |-DHCP Message Type: (Release)\n");

                sendDHCP(buffer);

                dhpro = 0;

                break;

            }

        }

    }

}

## **SendDHCP**

Nas duas etapas o software faz um sendDHCP(buffer) a função é a mesma, mas dependendo da variável global dhpro o pacote que será enviado vai ser montado como Offer ou ACK. Os pacotes enviados são bem parecidos o que os diferencia é o campo option é o campo mais importante do DHCP. Um trecho da parte que envia um pacote pode ser observado no Código 4. A função sendto() usa sockets para a comunicar pela internet, a variável sockraw é o socket que foi criado, sdbuffer é o buffer que foi preenchido para ser enviado, ele contém todos os valores do pacote. A variável tot\_len é o tamanho do sdbuffer. As structs são usadas contém o MAC destino e local.

Código 4: Envio de um pacote

send\_len = sendto(  sockraw,

                    sdbuffer,

                    tot\_len,0,

                    (struct sockaddr\*)&sdaddr\_ll,

                    sizeof(struct sockaddr\_ll)

                );

if (send\_len < 0) {

    printf("\nErro sending packet\n");

    return -1;

}

Na parte de montagem dos cabeçalhos Ethernet, IP, UDP e DHCP as variáveis dentro das structs recebem valores hexadecimais. Exemplo de como foi montado o pacote UDP no Código 5. Olhando para a função que monta o pacote UDP vemos que dependendo da variável dhpro será construído um pacote Offer ou ACK.

Código 5: Preparação do campo UDP

void constructUDP()

{   // Construct the UDP header

    udp\_send = (struct udphdr \*)(sdbuffer + sizeof(struct iphdr)+sizeof(struct ethhdr));

    udp\_send->source = htons(67);

    udp\_send->dest = htons(68);

    udp\_send->check = 0;

    if (dhpro == 1) { // A Discovery message

        constructDHCPOffer();

    }

    else if (dhpro == 3) { // A Request message

        constructDHCPACK();

    }

    udp\_send->len = ntohs(sizeof(struct dhcp\_packet)- sizeof(struct udphdr)-4);

}

# **Servidor de DNS Falso**

Foi implementado em uma máquina Linux um servidor DNS usando o Bind9, o intuito deste servidor é falsificar nomes. O servidor DHCP entrega o IP do servidor DNS falso, então quando uma máquina for atacada por DHCP Spoofing ela usará o DNS falso para resolver nomes.

Exemplo do servidor DNS um usuário digita no navegador www.google.com, e o servidor foi configurado para redirecionar para o ip de um outro site, www.gov.br. O intuito do servidor de DNS é mostrar como o fato de uma máquina ter aceito um endereço DHCP falso implica em não só ter recebido um endereço falso, mas como mostra que a máquina atacada ficou submissa ao atacante.

Para o servidor com o Bind9 funcionar deve-se configurar alguns arquivos. O primeiro é o named.conf.local localizado na pasta do Bind9. Nele é configurado a zona direta e a inversa. No nosso exemplo a zona direta foi configurada como google.com e a inversa com 192, que é o começo do IP 192.168.0.30 do nosso servidor DNS. A configuração completa pode ser vista na Figura 2:

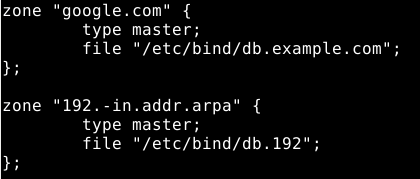


Figura 2: Configuração do arquivo named.conf.local

Na zona se coloca o arquivo onde estará localizada as configurações. No nosso caso estes arquivos são o db.exemplo.com e db.192. Neste arquivo é onde estará localizado o IP que será entregue quando o usuário tentar acessar determinado site, no nosso exemplo usamos o www.google.com. Na Figura 3 pode se observar a configuração da zona direta, onde o www.google.com irá retornar o IP 157.240.22.35.

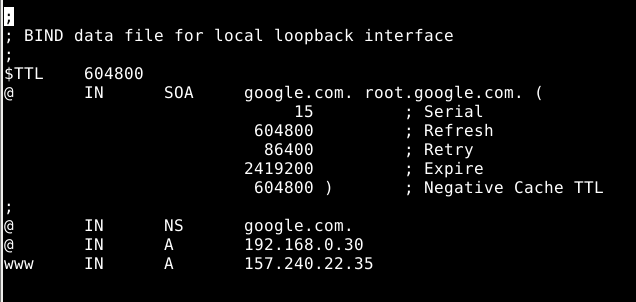


Figura 3: Configuração do arquivo db.exemplo.com

Na zona reversa foi configurado o resto do IP.

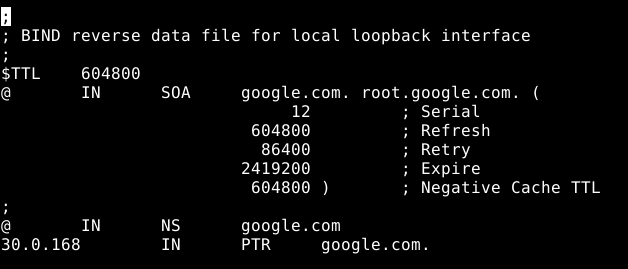


Figura 4: Configuração do arquivo db.192

Após escrever as configurações nos arquivos é necessário resetar o servidor do Bind9.

# **Testes para validar o Servidor DNS e DHCP Spoofing**

Para verificar os testes foi usado duas máquinas virtuais (VMs) com o sistema operacional Linux Debian, uma VM é a máquina que será atacada e outra vai rodar o programa atacante e o servidor de DNS falso. Foi também utilizado uma ferramenta chamada Wireshark (monitora todo o tráfego da rede) nas duas VMs para verificar se os pacotes estão chegando corretamente. Após verificar que a máquina atacada recebeu o endereço falso, utilizamos a ferramenta *nslookup* do Linux para tentar acessar um site previamente configurado no servidor de DNS, assim quando tentamos acessar o site google.com o servidor nos retornava um ip de um site diferente. Também pode ser usado o comando host, como em host www.google.com para verificar o IP retornado. No nosso teste nós alteramos o IP da máquina atacada para 192.168.0.70, o seu servidor DNS para 192.168.0.30 e o endereço de www.google.com para 157.240.22.35, este sendo o endereço para o site facebook.com.

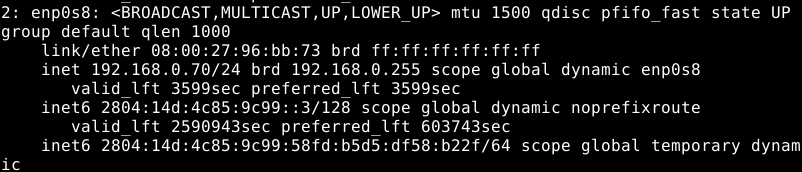


Figura 5: IP da máquina atacada alterado para 192.168.0.70

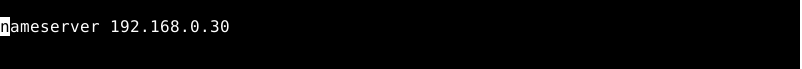


Figura 6: DNS da máquina atacada modificado



Figura 7: Teste para verificar o IP retornado quando www.google.com é acessado

# **Bibliografia**

Redes de Computadores e a Internet: uma abordagem top-down – Kurose, J

Redes de Computadores - Alexandre Fernandes de Moraes

# **Referências**

Academia - Main in The Middle

<https://www.academia.edu/4283121/DHCP_Main_in_The_Middle>

OpenSource - A Guide to Using Raw Sockets

<https://opensourceforu.com/2015/03/a-guide-to-using-raw-sockets/>

BinaryTides - Programming raw udp sockets in C on Linux

<https://www.binarytides.com/raw-udp-sockets-c-linux/>