PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

RENATA MOTTIN
SARAH LACERDA DA SILVA

TRABALHO FINAL – REDES DE COMUNICAÇÃO II – RELATÓRIO

A proposta do presente trabalho é de implementar uma aplicação que envia mensagens DHCP para os clientes que solicitam endereços IPv4, para a execução de um ataque do tipo *man-in-the-middle*.

A implementação do trabalho iniciou com a definição de structs para receber os pacotes recebidos através do socket raw. O socket raw recebe um conjunto de bytes, que devem ser mapeados para uma estrutura com sentido, para posterior utilização. Assim, foram utilizados structs para header ethernet, o header IP e o header UDP, que juntos formam um struct que representa o frame ethernet. Esses structs foram baseados na implementação que vimos na aula de Laboratório de Redes no semestre passado.

Em seguida, fizemos a implementação de um struct para receber o pacote DHCP. Esse struct foi construído baseado na estrutura do pacote DHCP, conforme documentação encontrada em referências na internet. O DHCP utiliza a estrutura do pacote BOOTP, com a inclusão de um campo chamado Magic Number (os 4 primeiros bytes da option, na nossa implementação), que recebe um valor específico para indicar que esse pacote é do protocolo DHCP.

Com as structs construídas, partimos para a implementação do algoritmo. A lógica principal do algoritmo é a recepção dos pacotes através do socket raw aberto, buscando por pacotes específicos que permitiram executar o ataque de DHCP Spoofing. Para encontrar os pacotes que queremos, é necessário entender a lógica de como o protocolo DHCP é utilizado por um servidor DHCP para atribuir IP's para clientes. A máquina de estados do protocolo DHCP é complexa, tratando uma série de possíveis cenários, mas a interação principal se dá através da seguinte lógica:

*O cliente que quer ter um IP assinalado envia uma mensagem do tipo DHCPDISCOVER para a rede, em broadcast.

*O servidor DHCP responde a esse pedido com uma DHCPOFFER. Mais de um servidor DHCP pode responder em uma mesma rede, emitindo mais de uma offer para o cliente.

*O cliente seleciona uma das offers, muitas vezes por um critério de ordem de chegada, e envia uma mensagem do tipo DHCPREQUEST endereçada para o servidor que fez a offer, efetivamente fazendo o pedido por um IP.

*O servidor DHCP escolhido pelo cliente responde com uma mensagem DHCPACK, que efetivamente atribui o IP ao cliente.

Essas mensagens não servem apenas para a definição de um IP, mas trazem uma gama de opções, dentre elas a definição do endereço do servidor DNS a ser utilizado pelo cliente e o default gateway a ser utilizado. O ataque realizado pela nossa aplicação trabalha enviando uma offer para o cliente, para determinar um IP, se atribuir como o default gateway e fixar um servidor de DNS infectado (nós não fizemos a implementação do servidor de DNS, utilizamos o fornecido em aula).

Para filtrar os pacotes que buscamos entre todos aqueles que a nossa aplicação está escutando na rede, temos que encontrar alguns códigos específicos nos pacotes recebidos. Inicialmente, o campo do IP header que define o protocolo da mensagem deve ter o valor de 17, que se refere ao protocolo UDP. Em seguida, nas portas definidas no header UDP, buscamos os valores 67 e 68, que são as portas utilizadas pelo protocolo DHCP. Finalmente, para encontrarmos os pacotes DHCP dos tipos específicos referidos na interação acima, buscamos os códigos 1, que se refere a um DHCPDISCOVER, e 3, que se refere a um DHCPREQUEST. Esses códigos se encontram no campo options do pacote DHCP.

Assim, quando a aplicação recebe um pacote do tipo DHCPDISCOVER, ele constrói um pacote de resposta, do tipo DHCPOFFER. O preenchimento da maioria dos campos desse pacote é bastante claro. A maior parte do conteúdo relevante para o nosso propósito está no pacote DHCP. O IP atribuído ao cliente é referido no campo yiaddr (your IP address). As definições do default gateway e do endereço de servidor DNS são feitas nas options, com os códigos 3 e 6. A documentação que encontramos na internet em relação a alguns dos campos do pacote DHCP não foi consistente (em especial os campos que definem os IP’s), sendo que encontramos comportamentos distintos nos pacotes enviados pelos roteadores que testamos. Aparentemente, o campo fundamental é justamente o yiaddr, referido acima, onde é passado o IP que está sendo atribuído ao cliente.

Após o envio do DHCPOFFER, a aplicação espera a resposta do cliente, em um pacote DHCPREQUEST. Quando isso ocorre, construímos um pacote DHCPACK de maneira muito similar ao que é descrito acima, apenas com pequenas mudanças em alguns campos. Quando esse pacote é enviado e recebido pelo cliente, o cliente se atribui o IP passado por nossa aplicação, atribui o endereço da nossa aplicação como default gateway e configura o seu endereço de

DNS para o endereço que passamos, completando assim o ataque *man in the middle*, eis que o tráfego do cliente passará pela nossa aplicação e o DNS acessado será o nosso servidor infectado.

Em relação à implementação, nós tivemos alguma dificuldade na construção dos pacotes, em razão de problemas com a determinação do tamanho exato dos pacotes (e de cada parte dos pacotes) e com a definição do checksum do header IP. O checksum do header IP, na implementação final do trabalho, é feito através do código fornecido em aula. Quando esses problemas foram solucionados, os pacotes começaram a ser reconhecidos pelo cliente.

A aplicação pode ter melhorias. Alguns valores estão passados *hardcoded*, que poderiam ser tratados de maneira dinâmica (a máscara de sub-rede, por exemplo, é atribuída sempre como 255.255.255.0). O tamanho dos pacotes e das estruturas que compõem os pacotes também estão *hardcoded* e poderiam ser calculadas com o tamanho efetivo das estruturas. Essas melhorias não foram realizadas por uma questão de tempo, visto que preferimos focar na implementação da parte essencial do trabalho.

Ao final, a realização desse trabalho trouxe uma compreensão sobre o funcionamento na prática da comunicação por redes. Como exemplo, podemos citar aspectos da estrutura do frame ethernet, com um entendimento melhor sobre qual a finalidade dos campos dos cabeçalhos ethernet, IP, UDP e DHCP. Aspectos de implementação, como a necessidade de uso de funções como htons() também ficaram muito mais claros. Finalmente, pudemos observar que a comunicação por redes se traduz em estruturas pré-determinadas, preenchidas com bytes, e que é plenamente possível trabalhar com essas peças para atingir um objetivo, como, no caso, de fazer um ataque *man in the middle*.