Injetor de Pacotes em Python

Bruna Almeida Osti
Rafael Cortez Sanches
28 de Junho de 2020

Resumo

Este trabalho tem como objetivo realizar um experimento de injeção de pacotes em uma conexão TCP/IP estabelecida. Primeiramente, são apresentados alguns conceitos do protocolo TCP e alguns ataques conhecidos direcionados a ele. O ambiente de experimentação é constituído por duas máquinas virtuais (cliente/servidor) rodando em um mesmo sistema hospedeiro, sendo todos eles Linux Ubuntu. A injeção de pacotes é feita com um script Python, o qual utiliza a biblioteca Scapy para interceptar pacotes legítimos e com esses forjar pacotes falsos. Por fim, são apresentadas possíveis contramedidas para a injeção de pacotes no protocolo TCP.

1 Introdução

O TCP/IP (TCP - Protocolo de controle de transmissão, IP - Protocolo de Internet) é um protocolo para transmissão de dados na rede, no qual funciona em conjunto com vários outros protocolos responsáveis por etapas distintas do processo e que juntos garantem a integridade dos dados que passam pela rede (1).

Os protocolos são uma espécie de linguagem utilizada para que as máquinas consigam se comunicar, os quais seguem na prática o modelo de arquitetura TCP/IP formado pelas camadas: aplicação, transporte, internet e rede (2).

Entretanto, esse protocolo acaba tendo muitas vulnerabilidades sendo passível de ataques por hackers como: man-in-the-middle, Ip spoofing, Session hijacking e injeção de pacotes (foco do trabalho).

O ataque mais conhecido a respeito de Ip spoofing e Session Hijacking foi performado por Kelvin Mitnick em 1994 ao Tsutomu Shimomura, que era especialista em segurança de redes do Centro Nacional de Supercomputação em San Diego - Califórnia, que montou várias armadilhas para capturá-lo e resultou na prisão do hacker tempos depois (3).

Neste trabalho performaremos parte do que foi feito por Mitnick, injetaremos dados numa conexão já estabelecida do protocolo TCP. Implementaremos a comunicação entre duas máquinas virtuais (cliente e servidor) utilizando o protocolo tcp/ip, analisaremos o tráfego de rede e implementaremos um injetor de pacotes para interceptar a conexão e tentar enviar pacotes falsos no lugar da vítima (cliente).

2 Desenvolvimento

Nesta seção discutiremos a respeito de todos os passos envolvidos no desenvolvimento do trabalho desde os requirimentos iniciais de configuração, passando pela configuração do cliente e servidor que serão vítimas, até o estudo e desenvolvimento do injetor de pacotes.

2.1 Instalação e configurações iniciais

2.1.1 Instalação das máquinas virtuais

Para desenvolvimento do experimento foram utilizadas duas máquinas virtuais (cliente e servidor) ambas com sistema operacional Ubuntu 16.04 e a máquina física (injetor de pacotes) com sistema operacional Ubuntu 19.10, podemos visualizar a estrutura do experimento através da Figura 1.

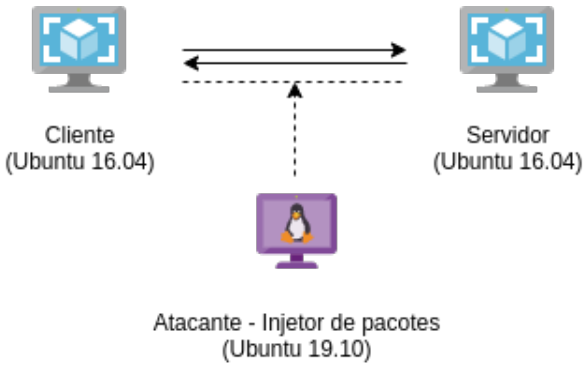

Figura 1: Estrutura do ataque

2.1.2 Configuração de rede

Primeiramente precisamos configurar o modo das placas de rede para que elas consigam detectar umas as outras na rede. Escolhemos colocar as placas todas em modo brigde, para que as máquinas tenham seu próprio endereço de ip mas utilizando a placa de rede do computador.

Após configurado testamos se elas estão conseguindo se detectar através do comando: ping <ip da máquina desejada>.

Após verificar se todas as máquinas conseguiam se enxergar partimos para a próxima etapa, a configuração do servidor e do cliente que serão os alvos do ataque.

2.1.3 Configuração do cliente e servidor

Configuramos as máquinas virtuais como cliente e servidor, para que consigam trocar pacotes utilizando o protocolo tcp. O protocolo tcp especifica três fases durante uma conexão: estabelecimento da ligação, transferência e término de ligação.

Para estabelecer a conexão, o tcp utiliza um aperto de mão de três vias, portanto, quando a conexão é estabelecida é possível injetar pacotes na conexão.

1. *SYN:* A abertura ativa é realizada por meio do envio de um SYN pelo cliente ao servidor. O cliente define o número de sequência de segmento como um valor aleatório A.
2. *SYN+ACK:* Em resposta, o servidor responde com um SYN-ACK. O número de reconhecimento (acknowledgment) é definido como sendo um a mais que o número de sequência recebido, exemplo: A+1, e o número de sequência que o servidor escolhe para o pacote é outro número aleatório B.
3. *ACK:* Finalmente, o cliente envia um ACK de volta ao servidor. O número de sequência é definido ao valor de reconhecimento recebido, i.e. A+1, e o número de reconhecimento é definido como um a mais que o número de sequência recebido, exemplo: B+1.

2.1.4 Configuração da máquina atacante em modo promíscuo

Após a configuração das máquinas que trocariam mensagens percebemos que não conseguiríamos visualizar o tráfego de rede entre as máquinas sem que nossa máquina atacante estivesse em modo promíscuo.

Portanto, configuramos a máquina em modo promíscuo com o comando:

#ifconfig <interface de rede> promisc

E em seguida, tentamos visualizar a rede novamente com o tcpdump filtrando pela interface e pela porta utilizada no servidor, para obter apenas as respostas mais interessantes, da forma:

#tcpdump -i <interface> port <porta do servidor>

2.2 Injetor de pacotes

Para o injetor de pacotes ser implementado são necessários alguns submódulos para o funcionamento, como por exemplo um sniffer de pacotes para verificar o número inicial da sequência de envio do cliente e do servidor, e um gerador de pacotes com base nesses números.

2.2.1 Sniffer de Pacotes

Para a implementação do sniffer de pacotes foi necessário a configuração do modo promíscuo do computador que como visto anteriormente serve para que o computador sem permissões de administrador da rede consiga visualizar todos os pacotes.

Em sequência capturamos os pacotes usando a biblioteca scapy, utilizando o filtro "tcp and port {}" para filtrar as informações mais importantes mais rápido e forjar os pacotes com a numeração.

2.2.2 Ataque

Bom, o ataque se baseia em dessincronizar ambas as máquinas para que não consigam mais trocar dados através do envio de pacotes forjados. Entretanto, é necessário um certo cuidado na montagem desses pacotes. De acordo com o estudo (4), existem padrões na comunicação.

O header do tcp tem sempre o mesmo formato, descrito pela Figura 2, sendo resultado do encapsulamento de todas as camadas do modelo de referência TCP/IP.

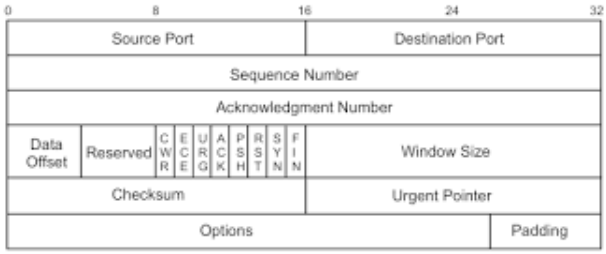

Figura 2: Header TCP

O campo "Número de sequência" é responsável por garantir a confidenciabilidade da comunicação, pois é gerado por ambas as partes no início da comunicação.

Portanto, é necessário sniffar a rede e capturar o número de sequência de ambos os lados, caso contrário fica inviável a adivinhação dos números. Em sequência forjamos os pacotes utilizando a biblioteca Scapy, de modo que:

SEG_SEQ = SEG_SEQ + SVR_ACK

SEG_ACK = SEG_ACK - CLT_ACK

No qual,

SEG_SEQ se refere a sequência de pacotes (header);

ACK_SEG se refere ao número de conhecimento (acknowlegdment);

CLT_ACK se refere ao próximo byte a ser recebido pelo cliente;

SVR_ACK se refere ao próximo byte a ser recebido pelo servidor.

2.2.3 ACK Storm

Depois de encontrado forjado os pacotes é necessário enviar vários pacotes com o número de sequência incrementado, para que o servidor reconheça o pacote forjado antes do pacote legítimo, essa ação é conhecida como "ACK

Storm". O loop é criado a cada vez que o dado é enviado pelo cliente e deve ser mais rápido que o pacote legítimo.

3 Testes

Houve várias etapas de testes para garantir o funcionamento dos módulos separados, entre elas: comunicação das máquinas, reconhecimento dos pacotes, testes de injeção, entre outros.

Primeiramente verificamos se toda a rede conseguia se identificar através do comando ping, nisso verificamos no primeiro momento que as máquinas não estavam se comunicando, portanto configuramos ambas as máquinas em modo bridge. Sendo assim, as máquinas conseguiram verificar a existência de umas as outras.

Na sequência implementamos o cliente e o servidor e os colocamos para funcionar. Tentamos primeiramente visualizar o tráfego através do host, sem sucesso. Então notamos que seria necessário configurar o host para estar em modo promíscuo, permitindo assim a visualização da comunicação entre as duas máquinas virtuais.

Para a interceptação dos pacotes foi necessário a implementação de um sniffer que capturasse o número de sequência de ambos os lados dos pacotes, para que os pacotes fossem forjados e as tentativas de injeção fossem um sucesso. A biblioteca Scapy possui ferramentas tanto para obter pacotes da inteface de rede quanto para forjar pacotes, então ela foi utilizada no script de injeção.

Dado um pacote **legítimo** capturado em trânsito servidor $->$ cliente, é possível prever qual será o próximo número de sequência SEQ de uma mensagem cliente $->$ servidor por meio do valor ACK desse pacote. Entretanto, quando fomos realmente fazer a injeção dos pacotes forjados, percebemos que os pacotes legítimos chegavam muito mais rápido, impossibilitando o ataque. Foi feita uma tentativa de realizar um *flood* no cliente para tentar deixar seu tempo de resposta mais lento, mas essa tentativa não foi bem sucedida.

Por outro lado, tentamos seguir o artigo citado como referência (4), mas não obtivemos sucesso pois o envio das mensagens legítimas ainda estava muito rápido.

Tentamos de tudo mas infelizmente não conseguimos injetar o pacote na comunicação de fato, apenas interceptar os pacotes e montar pacotes forjados. Descrições mais detalhadas sobre a implementação estão nos três scripts anexados a este trabalho: *server.py*, *client.py* e *packet-injector.py*.

4 Tipos de ataques com injeção de pacotes

Nesta seção será abordado os tipos de ataques que utilizam injeção de pacotes e como evitá-los.

4.1 Man-in-the-middle

Bom, o ataque MITM funciona com o invasor se posicionando entre duas partes que tentam se comunicar, intercepta as mensagens enviadas e depois personifica uma das partes envolvidas. Ou seja, um ataque MITM além de utilizar um Sniffer de pacotes (vasculhar a rede), também utiliza o TCP Spoofing (impersonificar uma máquina) (5).

4.2 IP Spoofing

O IP Spoofing baseia-se em mascarar um pacote e enviá-lo na rede usando pacotes falsos, portanto, enviando com o ip de outro usuário. Isso é relativamente fácil pois só demanda a modificação do header TCP (6).

4.3 Session Hijacking

O atacante espera as máquinas iniciarem a comunicação e sequestra a seção, através da visualização do tráfego (sniffer) e da injeção de pacotes/comandos na seção (5).

4.4 Contramedidas

Muitas vezes é difícil discriminar tráfego TCP regular de alguma tentativa de *spoofing* ou injeção de pacotes em uma rede local (LAN), como o ataque que foi conduzido neste experimento. Uma das formas de evitar a interceptação e falsificação de mensagens seria o uso de criptografia. Mesmo que o atacante consiga interceptar tráfego, ele não vai conseguir gerar mensagens autênticas para injetar nas partes comunicantes.

Em aplicações onde a confidencialidade não é tão prioritária quanto o desempenho, as mensagens podem ser enviadas com um *Message Authentication Code* (MAC). Esse código confere autenticidade à mensagem trafegada, de forma que se um atacante tentar modificar uma mensagem em trânsito ou injetar um pacote em uma conexão ativa, esse atacante não conseguirá gerar um MAC válido para a mensagem corrompida, fazendo com que o destinatário perceba a corrupção da mensagem e rejeite seu conteúdo.

Por fim, todo protocolo de aplicação construído em cima da camada de transporte deve observar essas questões de segurança e implementar devidas as contramedidas. O ideal é que esses protocolos sejam exaustivamente testados por equipes de *pentesting* antes de entrarem em produção.

5 Considerações Finais

Embora existam vulnerabilidades nas redes de comunicação, algumas podem ser muito difíceis de serem explorar pois exigem várias áreas de conhecimento, além de poder de processamento.

Durante a disciplina vimos muitos métodos de ataques, alguns mais simples, outros mais complexos, mas que acima de tudo mostravam que nenhum sistema é perfeitamente seguro. O que pode ser considerado seguro agora, pode daqui um tempo ter uma vulnerabilidade descoberta.

Além disso, entendemos que sempre é necessário estar atento a tentativas de ataques pois as tentativas acabam se tornando em ataques reais, e que sempre é necessário melhorar a segurança do sistema.

Neste experimento tivemos a oportunidade de estudar um pouco mais a respeito do funcionamento do protocolo TCP, sniffer de pacotes e o método de injeção de pacotes. Apesar de não termos atingido o resultado esperado, conseguimos entender o funcionamento e as mudanças necessárias ao código e a dificuldade de entender o funcionamento/padrão do número de sequência de pacotes, que é uma das seguranças principais do protocolo TCP. Além disso, podemos perceber que o tempo é um vilão nesse tipo de ataque, pois para o ataque ser um sucesso os nossos pacotes forjados devem chegar primeiro do que os pacotes legítimos.

Referências

[1] TecMundo, "O que é tcp/ip?," jun 2020.

[2] UniãoGeek, "Arquitetura de redes tcp/ip," jun 2020.

[3] F. de São Paulo, "Hacker mais famoso da história volta à rede," jun 2020.

[4] L. Joncheray, "A simple active attack against tcp," *Usenix*, 1995.

[5] netsparker, "What is session hijacking: Your quick guide to session hijacking attacks," jun 2020.

[6] welivesecurity, "Spoofing: entenda a técnica que ganhou destaque nos últimos dias," jun 2020.