# Primeiro Trabalho de Introdução à Segurança Computacional

## [Recon, port scanner simples para reconhecimento de redes]

Rafael Ravedutti Lucio Machado
Universidade Federal do Paraná
Curitiba, Paraná
rrlm13@inf.ufpr.br

## ABSTRACT

Este artigo provê informações relativas ao primeiro trabalho de Introdução à Segurança Computacional. O objetivo do trabalho é o desenvolvimento de um reconhecedor/escaneador de portas (port scanner) para detectar portas abertas em determinados endereços de rede IPv4.

## Keywords

recon; address; port; scanner; segurança; computacional

## 1. INTRODUÇÃO

Diversas técnicas existem hoje em dia para se escanear portas e endereços específicos de forma a maximizar o desempenho da aplicação (o port scanner) e minimizar os rastros causados pela mesma.

Algumas destas técnicas foram exploradas no desenvolvimento do port scanner em questão. Neste artigo será abordado seu funcionamento e o resultado obtido comparando com os resultados obtidos pelo *Nmap*, um port scanner *estado da arte* vastamente conhecido e utilizado nas plataformas Linux (não discutiremos outras plataformas neste artigo).

## 2. FUNCIONAMENTO

O port scanner desenvolvido tem como técnica padrão a conexão nas portas especificadas, apesar de não se preocupar com os rastros, o desempenho é melhorado estabelecendo um valor de *timeout* para a conexão menor do que o padrão especificado em sistemas Linux.

Essa possibilidade acaba gerando um problema pois surge a necessidade de definir um valor ideal para o *timeout* de forma que: (1) o valor estabelecido não seja muito grande para não permanecer a espera por muito tempo em cada porta escaneada e (2) o valor não pode ser muito pequeno, pois isto pode implicar no *scanner* parar a espera por resposta antes mesmo do servidor conseguir responder, desta forma, distúrbios e instabilidades na rede podem causar a não identificação de determinadas portas.

Outra técnica encontrada é o uso de *Raw Sockets* pela opção *-s*, o que permite a manipulação dos cabeçalhos dos pacotes e contudo não são finalizados todos os estágios de conexão TCP (famoso Three-Way Handshake). Com esta aproximação, a aplicação tende a diminuir o rastro devido à conexão não concluída (ainda é possível observar os pacotes de algumas maneiras como o uso de *sniffers*, como por exemplo o programa *tcpdump*), mas o objetivo maior é a melhora do desempenho, pois no caso de portas sem serviços, é possível detectar a recepção de um pacote com a flag RST sem manter o cliente esperando. Porém, para esta opção ser utilizada, privilégios do usuário *root* são necessários.

Ao efetuar a conexão nas portas específicadas, o port scanner armazena o endereço, porta e o banner em uma tabela de escaneamento, que é uma lista ligada de uma estrutura definida internamente no programa. Após o encerramento do reconhecimento da rede, o programa imprime essa tabela no formato específicado e libera a memória ocupada pela mesma.

Para se obter o banner, foi necessário enviar uma string mágica específica pois alguns serviços não enviam o banner logo de imediato (e.g. HTTP). A escolha da string mágica foi de *"HEAD / HTTP/1.1<enter><enter>"* justamente porque dos protocolos de serviço escaneados, apenas o HTTP necessita do envio de requisição antes do envio do banner (em outros serviços como FTP e SSH não foi necessário enviar uma requisição para conseguir o banner).

O *timeout* de recepção do socket também foi diminuido, isto porque foram identificados serviços que o mantém em modo de espera na função de recepção. Desta forma não espera-se mais do que o *timeout* especificado.

## 3. RESULTADOS

Os resultados mostrados tem como entrada a faixa de endereços 200.238.144.20-29 em todas as portas (0 à 65535). A Tabela 1 mostra o resultado obtido durante diversas execuções do programa.

Em seguida, observa-se os resultados de saída coletados pelo *Nmap* para fins de verificação de corretude do programa desenvolvido:

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

**Table 1: Portas escaneadas pelo programa**

| Endereço | Porta | Banner |
|---|---|---|
| 200.238.144.29 | 5061 | |
| 200.238.144.29 | 5060 | |
| 200.238.144.29 | 3306 | 4 |
| 200.238.144.29 | 1433 | |
| 200.238.144.29 | 445 | |
| 200.238.144.29 | 443 | |
| 200.238.144.29 | 135 | |
| 200.238.144.29 | 80 | HTTP/1.0 200 OK |
| 200.238.144.29 | 42 | |
| 200.238.144.29 | 22 | SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.9 |
| 200.238.144.29 | 21 | 220 Welcome to the ftp service |
| 200.238.144.28 | 8080 | HTTP/1.1 400 Bad Request |
| 200.238.144.28 | 5001 | ? |
| 200.238.144.28 | 22 | SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6 |
| 200.238.144.27 | 32768 | |
| 200.238.144.27 | 111 | |
| 200.238.144.27 | 22 | SSH-1.99-OpenSSH_2.9p2 |
| 200.238.144.27 | 21 | 220 redhood FTP server (Version wu-2.6.1-18) |

```
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
111/tcp   open  rpcbind
32768/tcp open  filenet-tms

Nmap scan report for 200.238.144.28
Host is up (0.010s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
5001/tcp open  commplex-link
8080/tcp open  http-proxy

Nmap scan report for 200.238.144.29
Host is up (0.011s latency).
Not shown: 989 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
42/tcp   open  nameserver
80/tcp   open  http
135/tcp  open  msrpc
443/tcp  open  https
445/tcp  open  microsoft-ds
1433/tcp open  ms-sql-s
3306/tcp open  mysql
5060/tcp open  sip
5061/tcp open  sip-tls
```

Vale ressaltar também que não foi empregada nenhuma técnica de computação paralela para efetuar a otimização do tempo de execução do algoritmo. Devido à isto, o programa levou aproximadamente 15 minutos para concluir a execução, enquanto o tempo de execução do *Nmap* foi de apenas cerca de 2 segundos.

# 4. CONCLUSÃO

Pode-se concluir que a corretude do programa é válida, mesmo podendo ter falhas dependendo do tempo de *timeout* de conexão estabelecido devido à instabilidade de redes, conforme já mencionado anteriormente.

Muitas outras técnicas podem ser adicionadas ao port scanner, tanto para aquisição de informações como os serviços e detecção de portas filtradas por firewall, assim como técnicas para otimização do desempenho do programa. Porém, mantemos o escopo apenas na solução mais simples e funcional do port scanner, sem preocupações com fatores de desempenho neste estágio.