# Scapy

**Ferramenta de manipulação de pacotes**

Pedro Henrique de Castro Teles Barbosa
Paulo Sérgio da Motta Pires

Universidade Federal do Rio Grande do Norte
Departamento de Computação e Automação

# Agenda

- Definição
  - Aplicações
- Motivação
- Comparações
  - Libnet
  - Ngrep
- Utilização
  - Modos de execução
  - Funções e Métodos
  - Exemplos
- Conclusões

# Definição

- Ferramenta de manipulação de pacotes de rede que provê classes para
  - Forjar ou decodificar pacotes de diversos tipos de protocolos;
  - Enviá-los e capturá-los pela rede;
  - Comparar requisições e respostas;
  - Traçar rotas de verificação do tráfego;

DCA

# Aplicações

- Testes e pesquisas (rápido envio de qualquer tipo de pacote e respostas esperadas)
- Scanear (a rede, portas , protocolos)
- Descobrimentos (traçar rotas, resposta de requisições)
- Ataques simulados
- Relatórios (.text, .html, LaTeX)

DCA

# Motivação

- Facilidade de uso;
- Grande poder de manipulação;
- Modularização;
- Controle total da rede;
- Informação nua e crua;
- Camadas 2 e 3;
- Free;
- Python.

DCA

# Motivação

- Protocolos suportados:
  - Ethernet
  - 802.1Q
  - 802.11
  - 802.3
  - LLC
  - EAPOL
  - EAP
  - BOOTP
  - PPP Link Layer
  - IP
  - TCP
  - ICMP
  - ARP
  - STP
  - UDP
  - DNS

- Protocolos em desenvolvimento
  - IPv6, VRRP, BGP, OSPF

DCA

# Comparações

- Substitui as ferramentas ttlscan, nmap, hping,queso, p0f, xprobe, arping, arp-sk, arpspoof, rewalk, irpas
- “Concorre” com o Libnet e Ngrep, também usando a biblioteca pcap(libpcap).

# Comparações

- Libnet e Scapy
  - Libnet tem utilização bem complexa em relação ao Scapy
    - Depois da instalação das bibiotecas libpcap e do programa libnet, ainda é necessária a compilação do programa desenvolvido em C.
  - Poder de atuação equivalente

***root@pedrohb:~/libnet/libnet#*** *gcc -Wall `libnetcfg --defines` example-1.c -o example-1 `libnet-config --libs`*

# Comparações

## LIBNET

```
libnet_build_ip(LIBNET_TCP_H,
        IPTOS_LOWDELAY,         /* IP tos */
        242,                    /* IP ID */
        0,                      /* frag stuff */
        48,                     /* TTL */
        IPPROTO_TCP,            /* transport protocol */
        src_ip,                 /* source IP */
        dst_ip,                 /* destination IP */
        NULL,                   /* payload (none) */
        0,                      /* payload length */
        packet);                /* packet header memory */

libnet_build_tcp(src_prt,       /* source TCP port */
        dst_prt,                /* destination TCP port */
        0xa1d95,                /* sequence number */
        0x53,                   /* acknowledgement number */
        TH_SYN,                 /* control flags */
        1024,                   /* window size */
        0,                      /* urgent pointer */
        NULL,                   /* payload (none) */
        0,                      /* payload length */
        packet + LIBNET_IP_H);
```

## SCAPY

```
>>> a = IP()

>>> b = TCP()
```

DCA

# Comparações

- Ngrep X Scapy
  - Os dois possuem fácil manuseio;
  - Scapy leva vantagem na gama de opções de utilização;
  - Ngrep lida com maior número de pacotes mais facilmente

```
root@pedrohb:~/libnet/libnet# ngrep dst host 192.168.0.1
interface: eth0 (192.168.0.0/255.255.255.0)
filter: ip and ( host 192.168.0.1 )
match: dst
##################################################################
############################exit
160 received, 0 dropped
```

# Utilização

- Modo Prompt:

*root@pedrohb:~# ./scapy.py -s mysession*

*New session [mysession]*

*Welcome to Scapy (1.0.4.3beta)*

*>>>*

- Modo de execução de módulos:

*root@pedrohb:~# python seguranca.py*

# Utilização

- Funções de baixo nível:
  - sr() - Enviar e receber pacotes na camada 3(rede);
  - sr1() - Enviar pacotes na camada de rede e receber apenas a primeira resposta da rede;
  - srp() - Enviar e receber pacotes na camada de enlace;
  - srp1() - Enviar e receber pacotes na camada de enlace e receber apenas a primeira resposta;

  - srloop() - Enviar pacotes na camada 3 em um loop e imprimir as saídas;
  - srploop() - Enviar pacotes na camada 2 em um loop e imprimir as saídas;
  - sniff() - Capturar pacotes;
  - send() - Enviar pacotes na camada 3;
  - sendp() - Enviar pacotes na camada 2;

  - ls() - Mostra a lista de camadas suportadas pelo Scapy;
  - ls(x) - Mostra as características de uma determinada camada x;
  - lsc() - Mostra todas as funções presentes no Scapy;
  - lsc(x) - Mostra os parâmetros da função x;
  - conf - Mostra todos os parâmetros iniciais predefinidos.

DCA

# Utilização

- Funções de alto nível:
  - p0f() - Função passiva de recebimento de pacotes do SO;
  - arpcachepoison() - Capturar e desviar pacotes de um determinado host para o computador desejado;
  - traceroute() - Traça a rota de IP's até um determinado nó da rede.
  - arping() - Envia um ARP para determinar quais hosts estão funcionando;
  - nmap_fp() - Função que implementa a ferramenta nmap;
  - report_ports() - Scanner de portas que gera uma tabela em Latex como relatório;
  - dyndns_add() - Envia uma mensagem de adição ao DNS para um novo nó;
  - dyndns_del() - Envia uma mensagem para apagar do DNS o nome desejado.
- Funções para criação de pacotes:
  - IP()
  - ICMP()
  - TCP()
  - Ether()
  - NET()

DCA

# Utilização

- Métodos:
  - summary() - mostra a lista de características de cada pacote;
  - nsummary()  - mesma função do anterior, só que informa-se o número do pacote;
  - conversations() - imprime o gráfico da conversação;
  - show() - mostra a representação desejada;
  - filter() - retorna uma lista de pacotes filtrados por uma função lambda;
  - plot() - plota uma função lambda para a lista de pacotes;
  - make_table() - mostra uma tabela de acordo com uma função lambda.

DCA

# Utilização

- Exemplos

```
>>> sr(IP(dst="www.XXXXX.com.br")/TCP(dport=[(20,80)]))
Begin emission:
.*Finished to send 61 packets.
*.....................................................................
Received 82 packets, got 2 answers, remaining 59 packets
(<Results: UDP:0 TCP:2 ICMP:0 Other:0>, <Unanswered: UDP:0
TCP:59 ICMP:0 Other:0>)
>>> ans,unans=_
>>> ans.summary()
IP / TCP 192.168.0.128:ftp-data > 200.176.3.142:ftp S ==> IP / TCP
200.176.3.142:ftp > 192.168.0.128:ftp-data SA / Padding
IP / TCP 192.168.0.128:ftp-data > 200.176.3.142:http S ==> IP /
TCP 200.176.3.142:http > 192.168.0.128:ftp-data SA / Padding
```

DCA

# Utilização

```
>>> traceroute(["www.yahoo.com","www.google.com","www.dca.ufrn.br"],maxttl=20)
Begin emission:
**********Finished to send 60 packets.
**********
Received 58 packets, got 58 answers, remaining 2 packets
   200.19.167.2:tcp80        64.233.161.99:tcp80              68.142.197.77:tcp80

1  192.168.0.1     11        192.168.0.1     11        192.168.0.1     11
2  200.217.50.128  11        200.217.50.128  11        200.217.50.128  11
3  200.164.196.101 11        200.164.196.101 11        200.164.196.101 11
4  200.164.196.89  11        200.164.196.25  11        200.164.196.25  11
5  200.223.131.21  11        200.223.131.13  11        200.223.131.13  11
6  200.223.131.18  11        200.223.131.2   11        200.223.131.66  11
7  200.223.131.74  11        200.223.131.18  11        200.223.131.38  11
8  200.223.254.117 11        200.223.131.74  11        200.223.131.102 11
9  200.222.104.125 11        -                         200.223.128.150 11
10 200.136.34.2    11        212.184.27.229  11        66.110.68.13    11
11 200.143.252.187 11        205.171.1.49    11        216.6.48.9      11
12 200.143.254.118 11        205.171.230.22  11        64.86.8.17      11
13 200.137.0.72    11        205.171.209.114 11        216.6.53.29     11
14 -                         205.171.251.22  11        216.6.53.6      11
15 200.19.167.2    SA              72.165.86.2     11        216.115.104.103 11
16 200.19.167.2    SA              216.239.47.120  11        68.142.193.31   11
17 200.19.167.2    SA              72.14.236.200   11        68.142.197.77   SA
18 200.19.167.2    SA              216.239.49.214  11        68.142.197.77   SA
19 200.19.167.2    SA              64.233.161.99   SA        68.142.197.77   SA
20 200.19.167.2    SA              64.233.161.99   SA        68.142.197.77   SA
(<Traceroute: UDP:0 TCP:12 ICMP:46 Other:0>, <Unanswered: UDP:0 TCP:2 ICMP:0 Other:0>)
```

DCA

# Utilização

```
>>> sniff(filter="tcp and host 192.168.0.128 and port 80", count=10)
    <Sniffed: UDP:0 TCP:10 ICMP:0 Other:0>
    >>> _.show()
    0000 Ether / IP / TCP 192.168.0.128:32770 > 207.46.28.78:http PA / Raw
    0001 Ether / IP / TCP 192.168.0.128:33195 > 207.46.28.78:http PA / Raw
    0002 Ether / IP / TCP 207.46.28.78:http > 192.168.0.128:32770 PA / Raw
    0003 Ether / IP / TCP 192.168.0.128:32770 > 207.46.28.78:http A
    0004 Ether / IP / TCP 207.46.28.78:http > 192.168.0.128:33195 PA / Raw
    0005 Ether / IP / TCP 192.168.0.128:33195 > 207.46.28.78:http A
    0006 Ether / IP / TCP 192.168.0.128:33194 > 207.46.28.78:http PA / Raw
    0007 Ether / IP / TCP 207.46.28.78:http > 192.168.0.128:33194 PA / Raw
    0008 Ether / IP / TCP 192.168.0.128:33194 > 207.46.28.78:http A
    0009 Ether / IP / TCP 207.46.28.78:http > 192.168.0.128:33194 PA / Raw
```

DCA

# Utilização

```
>>>sr1(IP(dst="192.168.0.1")/UDP()/DNS(rd=1,qd=DNSQR(qname="www.
    terra.com.br")))
    Begin emission:
    Finished to send 1 packets.
    *
    Received 1 packets, got 1 answers, remaining 0 packets
    <IP  version=4L ihl=5L tos=0x0 len=78 id=20831 flags= frag=0L ttl=128
    proto=UDP chksum=0x676e src=192.168.0.1 dst=192.168.0.128
    options='' |<UDP  sport=domain dport=domain len=58 chksum=0x13e6
    |<DNS  id=0 qr=1L opcode=QUERY aa=0L tc=0L rd=1L ra=1L z=0L
    rcode=ok qdcount=1 ancount=1 nscount=0 arcount=0 qd=<DNSQR
    qname='www.terra.com.br.' qtype=A qclass=IN |> an=<DNSRR
    rrname='www.terra.com.br.' type=A rclass=IN ttl=7193L
    rdata='200.176.3.142' |> ns=0 ar=0 |>>>
```

# Conclusões

- Scapy ainda tem algumas limitações, como:
  - Manipulação de pequena quantidade de pacotes;
  - Não faz o tratamento da informação recolhida, ou seja, fica a cargo do usuário tal ato;
  - Ainda não pode substituir algumas ferramentas presentes no mercado( Libnet, Ngrep) pela sua principal característica, de criar estímulos e receber respostas a estes.

DCA

# Conclusões

- Scapy é versátil por trabalhar tanto na camada 2 quanto na 3;
- Atravessa firewalls locais;
- Geração rápida de pacotes;
- Valores defaults que funcionam sem problemas;
- Grande poder de combinações;
- Para um simples exame de rede, muito pode se concluir deste;
- Poder ilimitado(além das infinitas combinações, muitos módulos são desenvolvidos atualmente).

DCA

Grato pela atenção de todos!

Perguntas?

Contato:


- pdro.henrique (at) gmail.com