VOLTAR

Packet Analyzer: criação de Packet Sniffer

Apresentar os recursos para monitorar entrada e saída de dados pela rede através de um analisador tipo sniffer.

NESTE TÓPICO

Marcar tópico

Olá alunos,

Vamos ver agora como farejar (sniffer) ou monitorar a entrada e saída de pacotes de dados pela internet. Hoje existem vários tipos de aplicativos que realizam esta atividade, mas vamos ver como criar um scriot relativamente simples em Python e outros recursos que podem ser utilizados através desta linguagem.

Lembrando que todos os dados que entram na via de transmissão pela rede são empacotados e etiquetados e isto exige certos padrões, por exemplo, se você precisa fazer uma entrega de uma encomenda pelo correio, deverá obedecer regras (padrões) para esta entrega, deverá colocar em um compartimento (depende do tamanho do material) para embalar e colocar em um local visível neste pacote todos os detalhes do destinatário: nome, endereço e o número do CEP.

Analogicamente, ocorre o mesmo na transmissão de dados, os dados são divididos em pacotes, em quantidades de bytes, obedecendo um padrão, o TCP, por exemplo e é colocado uma etiqueta, um endereço, no cabeçalho (header) de cada pacote que entrará na via de transmissão (por cabo ou por wireless).

No destino, estes pacotes serão juntados, para compor a mensagem ou o arquivo, que poderá ser de tipo caractere, de áudio ou de vídeo.

OS SNIFFERs

Os sniffers são aplicativos que verificam este transito de dados, na entrada e saída de um computador especificado, na realidade, estes aplicativos acessam a placa de rede deste computador e verificam os pacotes que estão chegando e saindo pela placa.

Estes aplicativos (sniffers) acessam as informações na rede em um nível mais baixo do que estamos acostumados, por isso, o retorno das informações estão próximas a linguagem Assembler, em valores hexadecimais, daí, necessitaremos de um aplicativo ou recurso para deixar a informação em um nível mais alto.

É necessário utilizar algo que já vimos antes, o **socket** para acessar as informações dos cabeçalhos com os IP e ICMP.

A ideia é fazer um scan (varrer) na rede do computador designado, isto também vai depender sobre qual sistema operacional você fará a ação: Windows ou Linux.

O Windows, por exemplo, exige alguns sinalizadores como IOCTL (Input Output Control), controle de entrada e saída que permite habilitar o modo promiscuo da interface de rede (Windows).

Por questão de segurança do Windows, você precisa habilitar, principalmente, nas versões do Windows, acima do Windows 7, esse modo promiscuo para acessar o socket.

Vamos ver um exemplo de **script** para **sniffer:**

```
1.  # script para sniffer
2.  import socket
3.  import os
4.
5.  host = "192.168.0.14" # este é o IP do meu host, o computador alvo que será escanead
    o
6.
7.  # cria o socket
8.  if os.name == "nt":
9.      socket_protocol = socket.IPPROTO_IP # verifica se o protocolo é endereço IP ou I
    CMP
10. else:
11.     socket_protocol = socket.IPPROTO_ICMP
12.
13. sniffer = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket_protocol)
14.
15. sniffer.bind((host, 0)) # aguarda o host
16.
17. sniffer.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1) # captura os cabeçalhos
    IP dos pacotes
18.
19. if os.name == "nt":
20.     sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON) # para Windows, habilita o mo
    do promiscuo
21.
22. print (sniffer.recvfrom(10000)) # recebe o IP do pacote
23.
24. if os.name == "nt":
25.     sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF) # para Windows, desabilita o
    modo promiscuo
```

No caso do bloco **if** , na linha **8**, verifica se na rede **"nt"** o protocolo é IP ou se é ICMP, se for trabalhar com Windows, não é necessário este bloco if, este bloco é necessário para Linux.

Na linha **17**, captura os endereços com IP, neste exemplo, não será capturado os pacotes com ICMP.

Nas linhas **19 e 20**, habilita o modo promiscuo e nas linhas **24 e 25**, desabilitam o modo promiscuo. O modo promiscuo permite fazer sniffing (varredura) de qualquer pacote que passa pela placa de rede.

Lembre-se que para trabalhar com o modo promiscuo, você precisa ter privilégios de administrador no Windows.

Neste script, capturamos apenas um pacote, mas podemos simplificar mais o script com outro exemplo:

```
1.  import socket
2.
3.  host = "192.168.0.14"
4.
5.  sniffer = socket.socket(socket.AF_INET,socket.SOCK_RAW,socket.IPPROTO_IP) # consider
    ando só o protocolo IP
6.
7.  sniffer.bind((host,0))
8.
9.  sniffer.setsockopt(socket.IPPROTO_IP,socket.IP_HDRINCL,1) # captura o cabeçalho IP d
    os cabeçalhos
10.
11. sniffer.ioctl(socket.SIO_RCVALL,socket.RCVALL_ON) # Para Windows
12.
13. while True:
14.    dados = sniffer.recvfrom(10000) # recebe o IP do pacote
15.    print (dados)
```

O resultado será algo parecido com isto:

```
1.     01 00 5e 7f ff fa fc f1 36 b3 1d 52 08 00 45 00
2.     00 3f 4e 01 40 00 40 11 3b ff c0 a8 00 0b ef ff
3.     ff fa a1 f9 3c f0 00 2b 27 c9 53 45 41 52 43 48
4.     20 42 53 44 50 2f 30 2e 31 0a 44 45 56 49 43 45
5.     3d 30 0a 53 45 52 56 49 43 45 3d 31 0a
```

Como já comentado, o script vai agir diretamente no hardware, em baixo nível e por isso a resposta vai ser também da mesma forma, como no exemplo acima, a resposta foi em hexadecimal.

Para melhorar a resposta teríamos que utilizar outros recursos, no caso, do Python pode-se utilizar uma extensão da biblioteca como wireshark, mas este recurso acabou se tornando independente e você pode fazer download diretamente através do link **http://wireshark.org/ (http://wireshark.org/)** é de licença livre e totalmente gratuito e hoje é um dos sniffer mais utilizados.

A instalação é tranquila pois basta aceitar a licença e de resto é só ir clicando até a conclusão.

No exemplo abaixo, vamos utilizar o wireshark para fazer o mesmo trabalho, e vamos fazer um teste acessando o site da Uninove:

```
1.  Ethernet II, Src: SamsungE_b3:1d:52 (fc:f1:36:b3:1d:52), Dst: IPv4mcast_7f:ff:fa (0
    1:00:5e:7f:ff:fa)
2.  www.uninove.br
3.  Num     time            source          destination     protocol        lenght  Info
4.  881     4.567906        192.168.0.14    143.137.103.67  TCP             54      77 b
    ytes captured (616 bits) on interface
```

O recurso também traz o resultado em hexadecimal e o resultado como acima, verifica os dispositivos que estão na rede, como a placa de rede, a smartTV, o IPv6 e IPv4 (linha 1).

O tempo que levou para acessar, o número do IP da fonte e o número do IP do destino, o tipo de protocolo de transmissão (TCP) e os demais dados capturados (linha **4**).

SAIBA MAIS...

Dê uma olhada nos links abaixo para saber mais sobre a linguagem Python:

https://www.python.org/doc/ (https://www.python.org/doc/)

https://wiki.python.org/moin/PythonBooks (https://wiki.python.org/moin/PythonBooks)

Neste tópico foi visto como criar um simples sniffer e utilizar outro aplicativo (wireshark) para verificar os pacotes de entrada e saída da rede.

Quiz

Exercício Final

Packet Analyzer: criação de Packet Sniffer

INICIAR

Referências

SUMMERFIELD, M. *Programação em Python 3*: Uma introdução completa à linguagem Python. Rio de Janeiro Alta Books, 2012. 495 p.

MENEZES, N. N. C. *Introdução à programação com Python:* algoritmos e lógica de programação para iniciantes. 2. ed. São Paulo: Novatec, 2014. 328 p.

SWEIGART, AL. *Automatize tarefas maçantes com Python:* programação prática para verdadeiros iniciantes. São Paulo: Novatec, 2015. 568 p.

PYTHON, doc. Disponível em: <https://www.python.org/doc/>. Acesso em: Junho/2018.

PYTHON, books. Disponível em: <https://wiki.python.org/moin/PythonBooks>. Acesso em: Junho/2018.

Avalie este tópico

ANTERIOR
Recebendo uma conexão em um servidor Python

Índice

PRÓXIMO
Info Gather

Ajuda?
(https://ava.uninove.br/seu/AVA/topico/topico.php?idCurso=)