Pentest com Kali Linux

Instrutor:Vitor Mazuco

http://facebook.com/vitormazuco

Email:vitor.mazuco@gmail.com

WebSite:http://vmzsolutions.com.br

Identificando com o Scapy

Há uma grande variedade de técnicas que podem ser utilizados para tentar a impressão digital do sistema operacional de um dispositivo que está na rede. Eles são verdadeiramente utilitários e eficazes na identificação do sistema operacional e empregam um grande número de técnicas para incluir em sua análise. No entanto, o scapy pode ser usada para analisar qualquer um destes fatores separadamente. Esta aula irá demonstrar como realizar a identificação do OS com scapy examinando os valores de TTL. Nessa aula, vamos usar um Metasploitable2 e um Windows

Identificando com o Scapy

Os sistemas operacionais Windows e Linux/Unix têm diferentes valores de TTL que são usados por padrão. Este fator pode ser usado para tentar impressões digitais do tipo de sistema operacional com o qual você está se comunicando. Estes valores encontram-se resumidos na tabela seguinte:

Operating system	Standard TTL value
Microsoft Windows OS	128
Linux/Unix OS	64

Identificando com o Scapy

Alguns sistemas baseados em Unix vai começar com um valor TTL padrão de 255; no entanto, para a simplicidade dessa aula, vamos usar os valores fornecidos como a premissa para as tarefas abordadas dentro deste aula. Para analisar os valores de TTL de uma resposta do sistema remoto, primeiro precisamos fazer um pedido. Neste exemplo, usaremos um pedido de ‘eco’ do Internet Control Message Protocol (ICMP). Para enviar a solicitação de ICMP, é preciso primeiro construir as camadas de esse pedido. A primeira camada teremos de construir é a camada IP:

Identificando com o Scapy

```
root@KaliLinux:~# scapy
Welcome to Scapy (2.3.2)
>>> linux = "192.168.1.196"
>>> windows = "192.168.1.84"
>>> i = IP()
>>> i.display()
>>> i.dst = linux
>>> i.display()
```

Identificando com o Scapy

Para construir a camada IP do nosso pedido, devemos atribuir o objeto IP à variável ’i’. Ao chamar a função de exibição, podemos identificar as configurações de atributos para o objeto. Por padrão, os endereços de envio e recebimento estão definidos para o endereço de auto-retorno ‘127.0.0.1’. Estes valores podem ser modificados, alterando o endereço de destino, definindo ‘i.dst’ igual ao valor da string do endereço que deseja verificar.

Chamando a função de exibição novamente, podemos ver que não só tem o endereço de destino tenha sido atualizado, mas o scapy também irá atualizar automaticamente o endereço IP de origem para o endereço associado com a interface padrão. Agora que temos já construído a camada IP de solicitação, devemos proceder à camada ICMP:

```
>>> ping = ICMP()
>>> ping.display()
```

Identificando com o Scapy

Para construir a camada ICMP do nosso pedido, vamos usar a mesma técnica como fizemos para a camada IP. No exemplo fornecido, o objeto ICMP foi atribuído à variável ping. Como discutido anteriormente, as configurações padrão pode ser identificada ligando para a função ‘display’. Por padrão, o tipo ICMP já está definido para o ‘echo-request’. Agora que nós criamos tanto o IP e as camadas ICMP, precisamos construir o pedido pelo empilhamento essas camadas:

```
>>> request = (i/ping)
>>> request.display()
```

Este mesmo pedido pode ser realizada sem a construção de empilhamento de forma independente e com cada camada. Em vez disso, um único comando pode ser usada ligando diretamente as funções e passando os argumentos apropriados para eles:

```
>>> ans = sr1(IP(dst=linux)/ICMP())
>>> ans
```

Identificando com o Scapy

Note-se que o valor de resposta do TTL do sistema Linux tinha um valor de 64. Este mesmo teste pode ser realizado contra o endereço IP do sistema Windows, e a diferença de valor TTL da resposta devem ser observados:

```
>>> ans = sr1(IP(dst=windows)/ICMP())
>>> ans
```