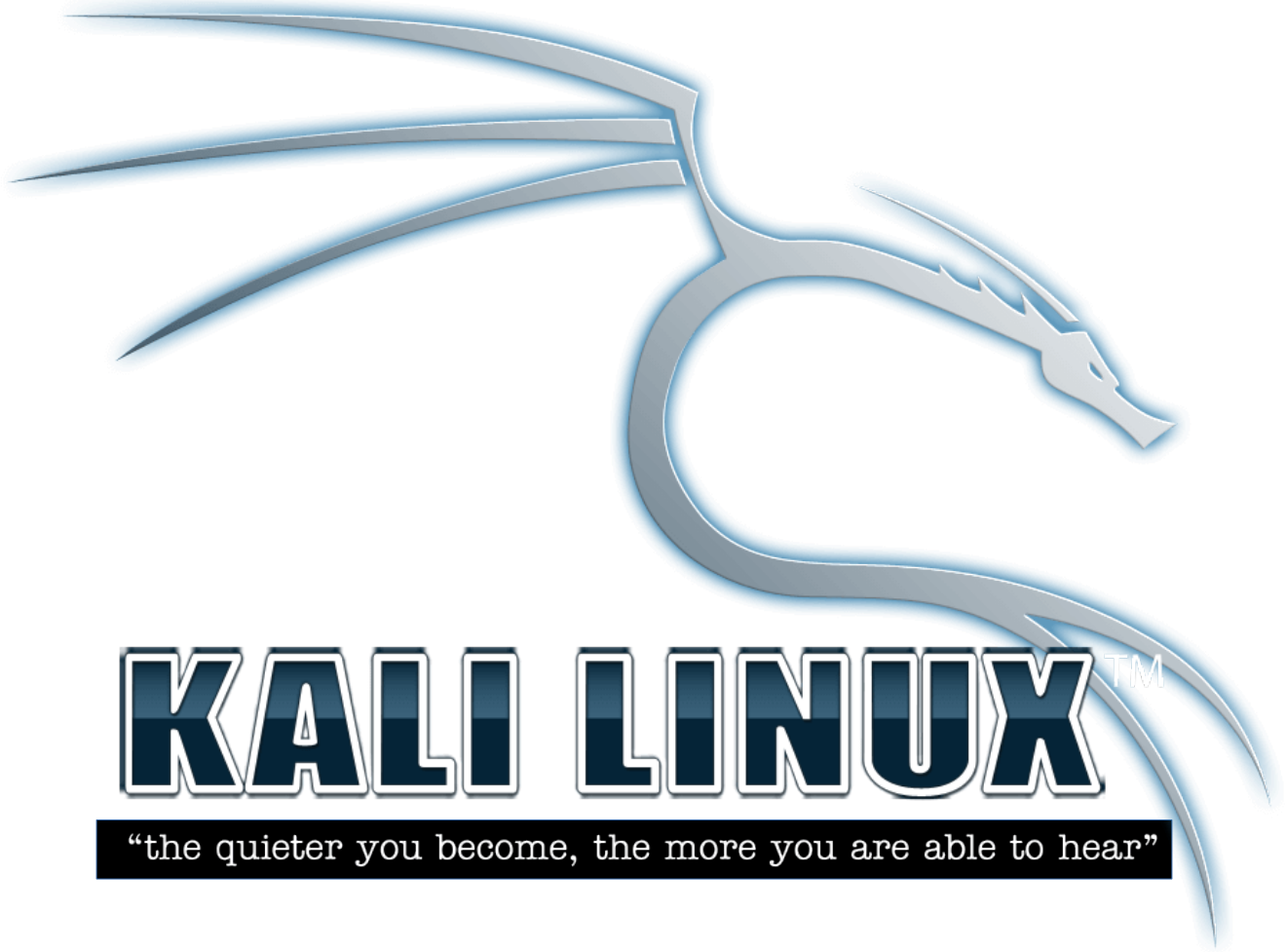
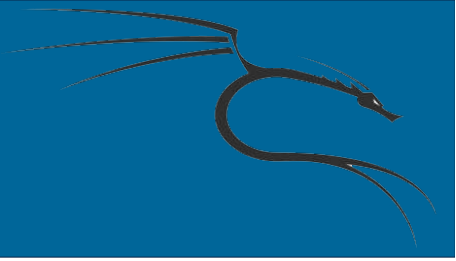
Pentest com Kali Linux





Instrutor:Vitor Mazuco

http://facebook.com/vitormazuco

Email:vitor.mazuco@gmail.com

WebSite:http://vmzsolutions.com.br

Teste de Fuzzing

Uma das técnicas mais eficazes para identificar as vulnerabilidades de *buffer overflow* é o teste fuzz. Fuzzing é a prática de testar os resultados associados com várias entradas, passando de dados criado ou aleatórios para uma função. Em certas circunstâncias, é possível que os dados de entrada pode escapar seu buffer designado e fluir em registros adjacentes ou segmentos de memória. Este processo irá interromper o fluxo de execução e resultar em aplicação ou sistema pode falhar.

Teste de Fuzzing

Em determinadas circunstâncias, as vulnerabilidades de buffer overflow também pode ser aproveitado para executar um código não autorizado. Nesta aula, vamos testar as vulnerabilidades de buffer overflow por desenvolvimento de ferramentas de fuzzing em Python. Nosso alvo, precisa ter um servidor rodando um FTP na porta 21. Pode ser um Windows ou Linux.

Teste de Fuzzing

Ao avaliar os serviços TCP, a função socket pode ser útil para simplificar o processo de realizar a sequência de se conectar a uma porta de serviço. O principal objetivo de qualquer script de fuzzing é enviar dados para qualquer função dada como entrada e avaliar o seu resultado. Pegue o script *ftp_fuzz.py* e vamos executar.

A primeira parte do script define a localização do interpretador Python e importa as bibliotecas necessárias. A segunda parte avalia o número de argumentos fornecidos para garantir que ele é compatível com o uso adequado do script. A terceira parte do script define as variáveis que serão utilizados ao longo da execução do script. Várias destas variáveis recebem seus valores a partir dos argumentos do sistema que são passados para o script após a execução.

As demais variáveis são definidas por aceitar a entrada do usuário do script. Finalmente, o restante do script define o processo de difusão. Nós executamos o arquivo *ftp_fuzz.py* da seguinte forma:

```
root@kali:~# python ftp_fuzz.py 192.168.1.196 21 A 100 1000
```

A carga vai definir o caractere ou sequência de caracteres a serem passados em massa para o serviço. O argumento de intervalo define o número de instâncias do *payload* definido que serão passados para o serviço de FTP na primeira iteração. O argumento também será o número com que o número de casos de carga será incrementado com cada iteração sucessiva, até ao valor máximo. Este valor máximo é definida pelo valor do último argumento.

Depois que o script é executado com esses argumentos do sistema, ele irá solicitar credenciais de autenticação para o serviço de FTP. No exemplo fornecido, o fuzzing foi realizada contra o serviço de FTP que é executado na porta TCP 21 do host.

As credenciais de login anônimos foram passados ao serviço FTP com um endereço de e-mail falso. Além disso, uma série de como foi passado para a função autenticação foi o MKD, começando com 100 casos e incrementando por 100, até ao máximo de 1000 casos foi atingido. O mesmo script também poderia ser usado para transmitir uma série de caracteres:

```
# python ftp_fuzz.py 192.168.1.196 21 ABCD 100 500
```