Guia Abrangente de Ferramentas de Análise de Rede no Kali Linux: Nmap e Além

Instrutor: Djalma Batista Barbosa Junior

E-mail: djalma.batista@fiemg.com.br

Introdução

A análise de redes é uma pedra angular da cibersegurança moderna, essencial tanto para a defesa de infraestruturas quanto para a avaliação de segurança ofensiva (pentest). Compreender as ferramentas e técnicas para mapear redes, identificar serviços, capturar tráfego e detectar vulnerabilidades é fundamental para qualquer profissional de segurança. O Kali Linux, uma distribuição Linux especializada em testes de penetração e auditoria de segurança, oferece um arsenal robusto de ferramentas projetadas para essas tarefas.

Este documento serve como um guia detalhado sobre os comandos e funcionalidades das principais ferramentas de análise de rede disponíveis no Kali Linux. O foco principal será no Nmap (Network Mapper), a ferramenta padrão da indústria para varredura de portas e descoberta de redes, mas também exploraremos outras ferramentas cruciais como Wireshark/TShark, tcpdump, Aircrack-ng, Bettercap, Hydra e Nikto. O objetivo é fornecer um recurso abrangente que detalhe os comandos essenciais, suas opções e exemplos práticos de uso, permitindo uma análise de rede eficaz e informada.

Visão Geral das Ferramentas de Análise de Rede no Kali Linux

O Kali Linux organiza suas centenas de ferramentas pré-instaladas em categorias que refletem as fases típicas de um teste de penetração ou auditoria de segurança.[3] Para análise de rede, as categorias mais relevantes incluem:

1. **Information Gathering (Coleta de Informações):** Ferramentas para descobrir informações sobre alvos, como mapeamento de rede, enumeração de DNS e OSINT (Open Source Intelligence). Nmap é um pilar aqui.
2. **Vulnerability Analysis (Análise de Vulnerabilidades):** Ferramentas que identificam fraquezas conhecidas em sistemas e serviços. Inclui scanners como Nmap (com NSE), Nikto e OpenVAS.
3. **Web Application Analysis (Análise de Aplicações Web):** Ferramentas focadas em encontrar falhas em aplicações web, como Burp Suite, sqlmap e Nikto.

4. **Password Attacks (Ataques de Senha):** Ferramentas para quebrar ou adivinhar credenciais, como Hydra, John the Ripper e Hashcat.
5. **Wireless Attacks (Ataques Sem Fio):** Ferramentas para analisar e atacar redes Wi-Fi, sendo o Aircrack-ng suite o principal exemplo.
6. **Sniffing & Spoofing (Captura e Falsificação):** Ferramentas para interceptar, analisar e manipular tráfego de rede, como Wireshark, TShark, tcpdump, Bettercap e Ettercap.

Embora muitas ferramentas se encaixem em múltiplas categorias, este guia focará nas mais diretamente aplicáveis à análise fundamental de redes: Nmap, Wireshark/TShark, tcpdump, Aircrack-ng, Bettercap, Hydra e Nikto. O Metasploit Framework, embora primariamente uma ferramenta de exploração, também possui módulos auxiliares valiosos para varredura e enumeração.

Nmap (Network Mapper)

O Nmap ("Network Mapper") é uma ferramenta open-source indispensável para exploração de rede e auditoria de segurança. Projetado para varrer rapidamente grandes redes, ele também funciona perfeitamente contra hosts individuais.[25] O Nmap utiliza pacotes IP brutos de maneiras inovadoras para determinar quais hosts estão disponíveis na rede, quais serviços (nome e versão da aplicação) esses hosts estão oferecendo, quais sistemas operacionais (e versões) estão executando, que tipo de filtros de pacotes/firewalls estão em uso e dezenas de outras características. Embora comumente usado para auditorias de segurança, muitos administradores de sistemas e redes o consideram útil para tarefas rotineiras como inventário de rede, gerenciamento de cronogramas de atualização de serviços e monitoramento de tempo de atividade de hosts ou serviços.

A saída principal do Nmap é uma lista de alvos varridos, com informações suplementares sobre cada um, dependendo das opções usadas. Central para essa informação é a "tabela de portas interessantes", que lista o número da porta, protocolo, nome do serviço e estado. Os estados possíveis são:

- open: Uma aplicação na máquina alvo está escutando conexões/pacotes naquela porta.
- closed: A porta não tem aplicação escutando, mas responde às sondas do Nmap (pode abrir a qualquer momento).
- filtered: Um firewall, filtro ou outro obstáculo de rede está bloqueando a porta, impedindo o Nmap de determinar se está aberta ou fechada.
- unfiltered: A porta responde às sondas do Nmap, mas o Nmap não consegue determinar se está aberta ou fechada (comum em scans ACK -sA).
- open|filtered / closed|filtered: O Nmap não consegue distinguir entre os dois estados.

Quando solicitado (-sV, -A), a tabela de portas também pode incluir detalhes da versão do software em execução.

Tipos de Varredura (Scan Types)

A escolha do tipo de varredura é crucial, pois determina como o Nmap sonda as portas e pode impactar a detecção por firewalls e IDS.

- **TCP SYN Scan (-sS):** O tipo de varredura padrão para usuários privilegiados (root). É rápido e relativamente furtivo, pois não completa a conexão TCP (envia SYN, recebe SYN/ACK, envia RST). É menos provável de ser logado por aplicações. No entanto, requer privilégios de root para criar pacotes SYN brutos.
- A escolha entre -sS e -sT é fundamental: -sS é preferível pela sua furtividade e velocidade quando se tem privilégios de root, enquanto -sT é a alternativa necessária (e mais barulhenta) sem esses privilégios.
- **TCP Connect() Scan (-sT):** O padrão se o SYN scan não estiver disponível (falta de privilégios de root). Usa a chamada de sistema connect() do sistema operacional para tentar estabelecer uma conexão completa com a porta alvo. É mais lento e mais fácil de detectar/logar do que o SYN scan.
- **UDP Scan (-sU):** Varre portas UDP. É inerentemente mais lento e mais difícil do que o TCP scan, pois o UDP é um protocolo sem conexão e não há garantia de resposta, mesmo de portas abertas. Muitas vezes, portas UDP abertas não respondem, enquanto portas fechadas podem responder com um erro ICMP "port unreachable",

que o Nmap usa para inferir o estado.

- **TCP ACK Scan (-sA):** Envia pacotes ACK. Não determina se uma porta está aberta ou fechada, mas é útil para mapear regras de firewall, determinando se as portas são filtered (nenhuma resposta) ou unfiltered (resposta RST).
- **TCP Window Scan (-sW):** Similar ao ACK scan, mas pode ocasionalmente diferenciar portas abertas de fechadas em certos sistemas devido a anomalias na implementação da janela TCP.
- **TCP Maimon Scan (-sM):** Nomeado após seu descobridor, Uriel Maimon. Envia pacotes FIN/ACK. Muitos sistemas BSD respondem com RST para portas abertas e fechadas, enquanto outros descartam o pacote se a porta estiver aberta.[25]
- **TCP NULL (-sN), FIN (-sF), Xmas (-sX) Scans:** Scans "furtivos" que exploram nuances da RFC 793. Sistemas compatentes com a RFC devem responder com RST para portas fechadas e ignorar pacotes para portas abertas. No entanto, nem todos os sistemas seguem a RFC (notavelmente o Windows), e firewalls stateful podem detectá-los.

Tabela 1: Tipos de Varredura Nmap Comun

Opção	Tipo de Varredura	Descrição	Privilégios	Furtividade	Confiabilidade
-sS	TCP SYN	Padrão (root). Rápido, furtivo (half-open).	Root	Alta	Alta
-sT	TCP Connect()	Padrão (não-root). Lento, detectável (conexão completa).	Não	Baixa	Alta
-sU	UDP	Varre portas UDP. Lento, pode perder portas abertas silenciosas.	Root	Média	Média/Baixa
-sA	TCP ACK	Mapeia regras de firewall (filtered vs unfiltered). Não detecta portas abertas.	Root	Média	N/A
-sW	TCP Window	Similar ao ACK, pode detectar portas abertas em alguns sistemas.	Root	Média	Baixa
-sM	TCP Maimon (FIN/ACK)	Explora resposta a FIN/ACK. Eficácia depende do SO.	Root	Média	Baixa
-	TCP	Scans furtivos	Root	Alta	Média/Baixa

sN/sF/sX	NULL/FIN/Xmas	baseados em RFC 793. Nem todos os SOs respondem corretamente. Podem evadir filtros simples.			

Descoberta de Host (Host Discovery / Ping Scanning)

Antes de varrer portas, o Nmap geralmente verifica se os hosts alvo estão online.

- **-sn (No port scan / Ping Scan):** Realiza apenas a descoberta de hosts, sem varrer portas. Útil para um levantamento rápido de quais máquinas estão ativas na rede.
- **-Pn (No ping / Skip host discovery):** Assume que todos os hosts alvo estão online e pula a fase de descoberta. Essencial para varrer hosts que bloqueiam as sondas de descoberta (e.g., ICMP Echo Request), mas pode desperdiçar tempo varrendo IPs inativos.
- **-PS[portlist] (TCP SYN Ping):** Envia pacotes TCP SYN para as portas especificadas (padrão 80). Hosts que respondem (SYN/ACK ou RST) são considerados online. Eficaz contra firewalls que bloqueiam ICMP.
- **-PA[portlist] (TCP ACK Ping):** Envia pacotes TCP ACK (padrão porta 80). Hosts que respondem com RST são considerados online. Útil para passar por firewalls que bloqueiam SYN.
- **-PU[portlist] (UDP Ping):** Envia pacotes UDP para as portas especificadas (padrão 40125). Respostas (UDP ou ICMP port unreachable) indicam que o host está online. Útil quando SYN/ACK são bloqueados.
- **-PE, -PP, -PM (ICMP Ping Types):** Usam ICMP Echo Request (-PE, padrão), Timestamp Request (-PP), ou Address Mask Request (-PM) para descoberta. Podem ser bloqueados por firewalls.
- **-PR (ARP Ping):** Usado automaticamente pelo Nmap para alvos na rede local Ethernet. Envia requisições ARP. É muito rápido e confiável para redes locais, pois firewalls não costumam bloquear ARP.
- **-n / -R (DNS Resolution):** -n desabilita a resolução DNS reversa (mais rápido). -R

força a resolução DNS sempre (pode ser útil, mas mais lento).

- **--dns-servers <server1[,server2],...>:** Usa servidores DNS específicos para resolução.
- **--system-dns:** Usa o resolvedor DNS do sistema operacional.
- **--traceroute:** Realiza um traceroute para cada host descoberto após a varredura.

Especificação de Alvos e Portas

- **Especificação de Alvos:** Nmap aceita nomes de host, endereços IP, notação CIDR (e.g., 192.168.1.0/24), ranges de octetos (e.g., 192.168.1.1-254, 10.0.0-255.1-254), e listas de arquivos.
  - -iL <inputfilename>: Lê alvos de um arquivo.
  - -iR <num hosts>: Escolhe alvos aleatórios na internet.
  - --exclude <host1[,host2],...> / --excludefile <file>: Exclui hosts da varredura.
- **Especificação de Portas (-p):** Controla quais portas são varridas. Se omitido, Nmap varre as 1000 portas TCP mais comuns.
  - -p <port ranges>: Específica portas individuais (e.g., -p 80), separadas por vírgula (e.g., -p 80,443), ranges (e.g., -p 1-1024), ou combinações (e.g., -p U:53,T:21-25,80,135-139,443,445,3389).
  - -p-: Varre todas as 65535 portas.
  - -p <service name>: Varre a porta associada a um nome de serviço (e.g., -p http,https).
  - -F (Fast scan): Varre apenas as 100 portas mais comuns (listadas no arquivo nmap-services). Mais rápido que o padrão, mas menos completo.
  - --top-ports <number>: Varre as <number> portas mais comuns.
  - --port-ratio <ratio>: Varre portas mais comuns que a <ratio> especificada (0 a 1).
- **Ordem de Varredura (-r):** Por padrão, Nmap padroniza a ordem das portas varridas. -r instrui o Nmap a varrer as portas sequencialmente.

Detecção de Serviço e Versão (-sV)

Identificar qual software e versão estão rodando em uma porta aberta é crucial para encontrar vulnerabilidades conhecidas.

- **-sV:** Habilita a detecção de versão. Nmap envia uma série de sondas específicas do protocolo para as portas abertas e analisa as respostas para determinar o serviço e a versão.
- **--version-intensity <level>:** Define a intensidade das sondas (0-9, padrão 7). Níveis mais altos usam mais sondas, aumentando a chance de identificar serviços corretamente, mas também aumentam o tempo de varredura.
- **--version-light:** Equivalente a --version-intensity 2. Mais rápido, mas menos provável de identificar serviços obscuros.
- **--version-all:** Equivalente a --version-intensity 9. Tenta todas as sondas possíveis.
- **--version-trace:** Mostra atividade detalhada da varredura de versão para depuração.
- **Detecção de Sistema Operacional (-O)**

Determinar o sistema operacional do alvo pode ajudar a refinar ataques e prever comportamentos.

- **-O:** Habilita a detecção de SO. Nmap envia uma série de sondas TCP, UDP e ICMP e compara as respostas com um banco de dados de "fingerprints" de sistemas operacionais conhecidos (nmap-os-db). Requer pelo menos uma porta aberta e uma fechada no alvo.
- **--osscan-limit:** Limita a detecção de SO a alvos promissores (que têm portas abertas e fechadas) para acelerar a varredura em grandes redes.
- **--osscan-guess / --fuzzy:** Faz o Nmap tentar adivinhar o SO mesmo quando as condições não são ideais (menos confiável).
- **-A (Aggressive Scan):** Uma opção de conveniência que habilita a detecção de SO (-O), detecção de versão (-sV), varredura com scripts padrão (-sC) e traceroute (--traceroute). É poderosa, mas também mais "barulhenta" e demorada.

Temporização e Desempenho (-T)

Ajustar a temporização é essencial para balancear velocidade, precisão e furtividade.

- **-T<0-5> (Timing Template):** Define um perfil de temporização pré-configurado. Varia de -T0 (paranoid, muito lento, evasivo) a -T5 (insane, muito rápido, agressivo e propenso a perder pacotes). O padrão é -T3 (normal). -T4 (aggressive) é frequentemente usado para varreduras rápidas em redes confiáveis. A escolha do template -T impacta diretamente a detectabilidade da varredura; -T0 e -T1 são projetados para evasão de IDS, enquanto -T4 e -T5 priorizam a velocidade, aumentando o risco de detecção e potencialmente sobrecarregando redes ou hosts mais lentos.
- **Opções Granulares:**
  - --min-rtt-timeout, --max-rtt-timeout, --initial-rtt-timeout <time>: Ajusta timeouts de RTT (Round Trip Time).
  - --max-retries <tries>: Número máximo de retransmissões de sonda.
  - --host-timeout <time>: Abandona um host se a varredura demorar mais que o tempo especificado.
  - --scan-delay, --max-scan-delay <time>: Adiciona um atraso entre as sondas.
  - --min-rate <number>, --max-rate <number>: Define a taxa mínima/máxima de envio de pacotes por segundo.

Tabela 2: Templates de Temporização Nmap (-T)

Template	Nome	Descrição	Velocidade	Furtividade	Risco de Perda de Pacotes/Detecção
-T0	Paranoid	Extremamente lento, para evasão de IDS.	Muito Baixa	Muito Alta	Baixo
-T1	Sneaky	Lento, para evasão de IDS.	Baixa	Alta	Baixo
-T2	Polite	Mais lento que o normal, consome menos banda/recursos.	Média/Baixa	Média	Baixo
-T3	Normal	**Padrão.** Bom equilíbrio entre velocidade e confiabilidade.	Média	Média	Médio
-T4	Aggressive	Rápido, assume rede rápida e confiável. Aumenta risco de detecção/perda.	Alta	Baixa	Médio/Alto
-T5	Insane	Muito rápido, assume rede excelente. Alto risco de detecção/perda/sobrecarga.	Muito Alta	Muito Baixa	Alto

Nmap Scripting Engine (NSE)

O NSE é uma das funcionalidades mais poderosas do Nmap, permitindo automatizar uma vasta gama de tarefas de rede usando scripts Lua.

- **-sC:** Executa os scripts da categoria default. Equivalente a --script default.
- **--script <filename>|<category>|<directory>|<expression>[,...]:** Executa scripts específicos, categorias inteiras, scripts de um diretório ou uma expressão booleana.
- **Categorias:** auth, broadcast, brute, default, discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, vuln.
    - default: Scripts considerados seguros, úteis e não intrusivos.
    - safe: Scripts que provavelmente não travaram o alvo ou consumiram recursos excessivos.
    - intrusive: Scripts com maior risco de serem percebidos como maliciosos ou de causar problemas.
    - vuln: Scripts que verificam vulnerabilidades específicas.
    - exploit: Scripts que tentam explorar vulnerabilidades.
    - auth: Scripts relacionados a autenticação ou bypass.
    - brute: Scripts que realizam ataques de força bruta.
    - discovery: Scripts que coletam mais informações sobre o alvo (e.g., shares SMB, títulos de páginas web).
- **--script-args <n1=v1,[n2=v2,...]>:** Passa argumentos para os scripts. Muitos scripts vuln ou brute requerem argumentos (e.g., credenciais, paths).
- **--script-help <script/category>:** Mostra ajuda para scripts específicos.
- **--script-updatedb:** Atualiza o banco de dados de scripts.
- **Exemplos de Scripts vuln:**
    - ✓ http-enum: Enumera diretórios comuns em servidores web.
    - ✓ smb-vuln-cve-2017-7494: Verifica a vulnerabilidade SambaCry.
    - ✓ http-csrf: Tenta detectar vulnerabilidades CSRF.
    - ✓ http-shellshock: Verifica a vulnerabilidade Shellshock em aplicações web.
    - ✓ dns-update: Tenta realizar atualização dinâmica de DNS sem autenticação.
    - ✓ ssl-heartbleed: Verifica a vulnerabilidade Heartbleed.

- ✓ ftp-anon: Verifica login anônimo em FTP.
- ✓ smb-os-discovery: Tenta obter informações do SO via SMB.

A categoria vuln é particularmente poderosa para varreduras de vulnerabilidade automatizadas. Executar nmap --script vuln <target> realiza uma ampla gama de verificações de segurança conhecidas. Ferramentas como Vulscan e Vulners integram bancos de dados de vulnerabilidades externos com o NSE, expandindo ainda mais essa capacidade.

Evasão de Firewall/IDS e Spoofing

Técnicas para tentar evitar a detecção ou bloqueio por dispositivos de segurança.

- **-f (Fragment packets); --mtu <val>:** Fragmenta os pacotes da varredura em pacotes menores (8 bytes ou múltiplos de 8 com --mtu). Pode enganar filtros de pacotes mais simples que não remontam os fragmentos.
- **-D <decoy1,decoy2[,ME],...> (Decoy scan):** Faz a varredura parecer originar-se de múltiplos endereços IP (decoys), tornando mais difícil para o alvo identificar o IP real do atacante. ME pode ser usado para indicar a posição do IP real na lista.
- **-S <IP_Address> (Spoof source address):** Falsifica o endereço IP de origem. Geralmente útil apenas quando o atacante não precisa ver as respostas (e.g., em scans cegos ou DoS) ou está na mesma sub-rede do alvo.
- **-e <iface> (Interface):** Especifica qual interface de rede usar para enviar/receber pacotes.
- **-g <portnum> / --source-port <portnum>:** Falsifica a porta de origem. Pode ser útil para passar por firewalls que permitem tráfego de portas específicas (e.g., DNS na porta 53).
- **--data-length <num>:** Anexa dados aleatórios aos pacotes enviados. Pode ajudar a evitar detecção baseada em assinaturas de pacotes Nmap padrão.
- **--proxies <proxy_URL1,...>:** Roteia a conexão através de proxies HTTP ou SOCKS4/5.
- **Opções de Temporização:** Usar templates lentos (-T0, -T1) ou ajustar --scan-delay

pode ajudar a evitar detecção por IDS baseados em taxa.

Formatos de Saída

Salvar os resultados da varredura é essencial para análise posterior e relatórios.

- **-oN <filespec> (Normal output):** Salva a saída interativa padrão em um arquivo.
- **-oX <filespec> (XML output):** Salva a saída em formato XML. Ideal para processamento por outras ferramentas ou scripts, ou para visualização com um stylesheet XSL.
- **-oS <filespec> (Script Kiddie output):** Salva em formato "l33t speak". Geralmente não é útil para análise séria.
- **-oG <filespec> (Grepable output):** Salva a saída em um formato fácil de parsear com ferramentas como grep, awk, cut. Cada host ocupa uma linha.
- **-oA <basename> (Output All):** Salva a saída nos três formatos principais (Normal, XML, Grepable) usando o nome base fornecido.
- **-v / -vv (Verbosity):** Aumenta o nível de detalhe da saída na tela. -v mostra mais informações sobre o progresso, -vv adiciona mais detalhes sobre pacotes e respostas.
- **-d[level] (Debugging):** Aumenta o nível de depuração (1-9). Útil para diagnosticar problemas com o Nmap.
- **--reason:** Mostra a razão pela qual o Nmap classificou uma porta em um determinado estado (e.g., syn-ack, rst).
- **--open:** Mostra apenas portas nos estados open ou open|filtered.
- **--packet-trace:** Mostra um resumo de todos os pacotes enviados e recebidos.
- **--iflist:** Lista as interfaces de rede e rotas detectadas pelo Nmap.
- **--append-output:** Anexa a saída aos arquivos especificados em vez de sobrescrevê-los.
- **--resume <filename>:** Continua uma varredura abortada (requer saída Normal -oN ou Grepable -oG da varredura anterior).
- **--stylesheet <path/URL> / --webxml / --no-stylesheet:** Controla a associação de stylesheets XSL com a saída XML (-oX) para formatação HTML.

Exemplos Práticos de Comandos Nmap

- **Scan Básico (SYN Scan, 1000 portas comuns, sem ping):**

  Bash

  ```
  sudo nmap -sS -Pn 192.168.1.1
  ```

- **Scan Rápido (100 portas comuns, sem ping, sem DNS):**

  Bash

  ```
  sudo nmap -F -Pn -n 192.168.1.0/24
  ```

- **Scan Agressivo (Detecção de SO/Versão, Scripts Default, Traceroute, Timing T4):**

  Bash

  ```
  sudo nmap -A -T4 scanme.nmap.org
  ```

- **Scan UDP (Top 50 portas UDP):**

  Bash

  ```
  sudo nmap -sU --top-ports 50 192.168.1.1
  ```

- **Scan Completo de Todas as Portas TCP com Detecção de Versão e SO, Saída em Todos os Formatos:**

  Bash

  ```
  sudo nmap -sS -sV -O -p- -oA full_scan_results 192.168.1.10
  ```

- **Scan de Vulnerabilidades (Scripts da categoria vuln), Verbose:**

  Bash

  ```
  sudo nmap --script vuln -v 192.168.1.15
  ```

- **Scan de Vulnerabilidade Específica (Heartbleed) com Argumento:**

  Bash

  ```
  sudo nmap -p 443 --script ssl-heartbleed --script-args vulns.showall 192.168.1.20
  ```

- **Scan com Evasão (Fragmentação, Decoys):**

  Bash

  ```
  sudo nmap -sS -f -D RND:10,ME 192.168.1.25
  ```

- **Listar Hosts Ativos na Rede Local (ARP Scan):**

  Bash

  ```
  sudo nmap -sn 192.168.1.0/24
  ```

Wireshark e TShark: Captura e Análise Detalhada de Pacotes

Wireshark é o analisador de protocolos de rede mais popular do mundo, amplamente utilizado para análise, solução de problemas, desenvolvimento de software e protocolos, e educação. Ele fornece uma interface gráfica rica (GUI) para capturar tráfego ao vivo de diversas interfaces de rede (Ethernet, Wi-Fi, Loopback, USB, etc.) e para analisar arquivos de captura salvos (como.pcap ou.pcapng). Sua força reside na capacidade de decodificar centenas de protocolos, permitindo uma inspeção profunda dos cabeçalhos e dados de cada pacote.

TShark é a contraparte de linha de comando (CLI) do Wireshark. Ele compartilha o mesmo poderoso motor de dissecação de protocolos e sistema de filtros de exibição do Wireshark, mas opera inteiramente no terminal. Isso o torna ideal para captura e análise de pacotes em servidores remotos, para automação de tarefas de análise através de scripts, e para extração rápida de informações específicas de grandes arquivos de captura.

Funcionalidades Principais (Wireshark/TShark)

- **Captura ao Vivo:** Captura de pacotes em tempo real de interfaces de rede.
- **Leitura de Arquivos:** Análise de dados de arquivos de captura pré-existentes em diversos formatos (pcap, pcapng, etc.).
- **Decodificação de Protocolos:** Inspeção detalhada de centenas de protocolos em diferentes camadas de rede.
- **Filtros de Captura:** Seleção de quais pacotes capturar com base em critérios

como IP, porta ou protocolo (usando a sintaxe BPF - Berkeley Packet Filter).

- **Filtros de Exibição:** Filtragem poderosa de pacotes já capturados (ou lidos de um arquivo) para focar em tráfego específico, usando uma sintaxe rica e específica do Wireshark.
- **Análise Estatística:** Ferramentas para gerar estatísticas sobre o tráfego capturado (Wireshark GUI e opção -z do TShark).
- **Exportação de Dados:** TShark permite exportar campos específicos em formatos como texto, CSV ou JSON.

TShark: Opções Chave da Linha de Comando

O TShark oferece uma vasta gama de opções. As mais relevantes para captura e análise de rede incluem:

- **Interface de Captura:**
  - -i <interface>: Especifica a interface de rede para captura (e.g., eth0, wlan0mon, 1 para a primeira interface listada por -D).
  - -D: Lista as interfaces de captura disponíveis.
- **Filtros de Captura (-f):**
  - -f <capture_filter>: Aplica um filtro BPF durante a captura. Apenas pacotes que correspondem ao filtro são salvos ou processados. Sintaxe similar ao tcpdump.
  - Exemplo: -f "tcp port 80" captura apenas tráfego TCP na porta 80.
- **Leitura/Escrita de Arquivos:**
  - -r <infile>: Lê pacotes de um arquivo de captura.
  - -w <outfile>: Escreve os pacotes brutos capturados em um arquivo (formato pcapng por padrão). Filtros de exibição (-Y) não se aplicam ao escrever em modo de captura ao vivo.
  - -F <format>: Especifica o formato do arquivo de saída (e.g., pcap, pcapng).
  - --compress <type>: Comprime o arquivo de saída ao ler de outro arquivo (-r).
- **Filtros de Exibição (-Y):**
  - -Y <display_filter>: Aplica um filtro de exibição Wireshark aos pacotes (lidos de -r ou capturados). Apenas pacotes que correspondem são exibidos ou processados posteriormente.

- Exemplo: -Y "ip.addr == 192.168.1.1 and tcp.port == 443" mostra tráfego de/para 192.168.1.1 na porta TCP 443.

- **Formatação de Saída (-T, -e, -E, -V, -x):**

  -T <format>: Define o formato da saída decodificada. Opções comuns:
    - text: Padrão, resumo legível por humanos.
    - fields: Extrai campos específicos definidos por -e, formatados por -E.
    - json: Formato JSON detalhado.
    - ek: JSON delimitado por nova linha para Elasticsearch.
    - pdml: XML detalhado (Packet Details Markup Language).
  - -e <field>: Adiciona um campo a ser extraído quando -T fields é usado (e.g., -e ip.src -e tcp.dstport -e http.request.method). Pode ser usado múltiplas vezes.
  - -E <option=value>: Define opções para -T fields, como header=y (imprime cabeçalho), separator=, (usa vírgula como separador - CSV), quote=d (usa aspas duplas).
  - -V: Exibe detalhes completos de todos os protocolos em cada pacote.
  - -x: Inclui um dump hexadecimal e ASCII dos dados do pacote.
- **Outras Opções Úteis:**
  - -c <count>: Para após capturar/ler <count> pacotes.
  - -n: Desabilita a resolução de nomes (rede e transporte).
  - -q: Modo silencioso (quiet), útil com -z.
  - -z <statistic>: Calcula e exibe estatísticas (e.g., -z io,phs para estatísticas de protocolo, -z conv,tcp para conversas TCP). Use tshark -z help para ver todas as opções.

A Diferença Crucial: Filtros de Captura (-f) vs. Filtros de Exibição (-Y)

É vital entender a distinção entre filtros de captura e filtros de exibição:

- **Filtros de Captura (-f <filtro_bpf>):**
  - ✓ Aplicados **durante** o processo de captura.
  - ✓ Usam a sintaxe BPF (Berkeley Packet Filter), a mesma do tcpdump.
  - ✓ Determinam quais pacotes são **descartados** pelo kernel ou pela biblioteca de captura antes mesmo de serem processados pelo Wireshark/TShark.
  - ✓ **Vantagem:** Reduzem a carga de processamento e o tamanho do arquivo de captura, especialmente em redes de alto tráfego.
  - ✓ **Desvantagem:** Pacotes filtrados **não podem ser recuperados** posteriormente. Se você filtrar algo importante por engano, ele se perdeu.
  - ✓ Exemplo: tshark -i eth0 -f "port 53" -w dns_capture.pcapng (Só captura pacotes de/para a porta 53).
- **Filtros de Exibição (-Y <filtro_wireshark>):**
  - ✓ Aplicados **após** a captura (ou ao ler um arquivo).
  - ✓ Usam a sintaxe rica e específica do Wireshark, que permite filtrar por qualquer campo de protocolo dissecado pelo Wireshark.
  - ✓ Determinam quais pacotes são **mostrados** na tela ou processados por opções como -T fields ou -z.
  - ✓ **Vantagem:** Extremamente flexíveis e poderosos para análise. Permitem "esconder" pacotes irrelevantes sem perdê-los; o filtro pode ser alterado a qualquer momento para ver outros pacotes.
  - ✓ **Desvantagem:** Não reduzem o tamanho do arquivo de captura original nem a carga inicial de processamento se aplicados a uma captura ao vivo muito grande.
  - ✓ Exemplo: tshark -r all_traffic.pcapng -Y "http.request.method == POST" (Lê todos os pacotes, mas só exibe requisições HTTP POST).

Recomendação Geral: Capture de forma ampla (talvez com um filtro de captura simples, se necessário, para reduzir o volume) usando -w. Depois, use os filtros de exibição (-Y) e as opções de formatação (-T, -e) do TShark para analisar e extrair os dados relevantes do arquivo salvo.

Exemplos Práticos com TShark

- **Capturar 100 pacotes da interface eth0 e salvar em capture.pcapng:**

  Bash

  ```
  sudo tshark -i eth0 -c 100 -w capture.pcapng
  ```

- **Ler capture.pcapng e exibir apenas tráfego HTTP ou HTTPS de/para o IP 192.168.1.50:**

  Bash

  ```
  tshark -r capture.pcapng -Y "(http or tls) and ip.addr == 192.168.1.50"
  ```

- **Extrair Hora, IP de Origem/Destino, Portas TCP de Origem/Destino e Método de Requisição HTTP de um arquivo, salvando como CSV:**

  - Bash

    ```
    tshark -r web_traffic.pcapng -Y http.request -T fields -e frame.time -e ip.src -e ip.dst -e tcp.srcport -e tcp.dstport -e http.request.method -E header=y -E separator=, -E quote=d > http_requests.csv
    ```

- **Listar as 10 principais conversas TCP em um arquivo de captura:**

  - Bash

    ```
    tshark -r network_traffic.pcapng -q -z conv,tcp | head -n 15
    ```

    (-q suprime a saída por pacote, head -n 15 mostra o cabeçalho e as 10 primeiras linhas de dados)

- **Capturar tráfego DNS ao vivo e exibir em formato JSON:**

  - Bash

    ```
    sudo tshark -i eth0 -f "udp port 53" -T json
    ```

tcpdump: Captura Clássica de Pacotes via Linha de Comando

tcpdump é uma ferramenta fundamental e onipresente para captura de pacotes em sistemas Unix-like. É conhecido por sua leveza, eficiência e pela poderosa sintaxe de filtragem BPF (Berkeley Packet Filter). Embora TShark ofereça capacidades de dissecação mais profundas, tcpdump continua sendo a escolha preferida para captura rápida, monitoramento contínuo em sistemas com recursos limitados, ou quando apenas a funcionalidade básica de captura e filtragem é necessária.

Opções Essenciais do tcpdump

- -i <interface>: Especifica a interface de rede para escutar (e.g., eth0, any para todas).
- -n: Não resolve nomes de host (mostra IPs).
- -nn: Não resolve nomes de host nem nomes de portas/serviços (mostra IPs e números de porta).
- -v, -vv, -vvv: Aumenta a verbosidade da saída, mostrando mais detalhes dos cabeçalhos dos pacotes.
- -c <count>: Sai após capturar <count> pacotes.
- -w <file>: Escreve os pacotes brutos capturados em um arquivo (formato pcap), em vez de exibi-los.
- -r <file>: Lê pacotes de um arquivo pcap previamente salvo com -w.
- -A: Exibe o conteúdo de cada pacote em ASCII (útil para ver dados de texto plano como HTTP).
- -X: Exibe o conteúdo de cada pacote em hexadecimal e ASCII.
- -s <snaplen>: Define o tamanho máximo de captura por pacote (snapshot length) em bytes. -s 0 captura o pacote inteiro (necessário para análise completa do payload).

Filtros (Berkeley Packet Filter - BPF)

A filtragem no tcpdump usa a sintaxe BPF, que é poderosa, mas diferente dos filtros de exibição do Wireshark. Os filtros são aplicados no nível da captura.

- **Tipo:** host, net, port.
- **Direção:** src, dst.
- **Protocolo:** tcp, udp, icmp, arp, ip, ip6, etc.
- **Operadores Lógicos:** and (ou &&), or (ou ||), not (ou !). Parênteses () podem ser usados para agrupar (geralmente precisam ser escapados ou colocados entre aspas na linha de comando para evitar interpretação pelo shell: \(...\) ou '...').

Exemplos Práticos com tcpdump

- **Capturar todo o tráfego na interface eth0:**

  Bash

  ```
  sudo tcpdump -i eth0
  ```

- **Capturar tráfego de/para o host 192.168.1.10:**

  Bash

  ```
  sudo tcpdump -i eth0 host 192.168.1.10
  ```

- **Capturar tráfego TCP na porta 80 (HTTP), mostrando IPs e portas, e conteúdo ASCII:**

  Bash

  ```
  sudo tcpdump -i any -nn -A 'tcp port 80'
  ```

- **Capturar tráfego UDP da rede 10.0.0.0/8 para a porta 53 (DNS):**

  Bash

  ```
  sudo tcpdump -i eth0 -nn 'udp and src net 10.0.0.0/8 and dst port 53'
  ```

- **Capturar 500 pacotes ICMP e salvar em um arquivo:**

  Bash

  ```
  sudo tcpdump -i eth0 -c 500 -w icmp_capture.pcap 'icmp'
  ```

- **Ler um arquivo e mostrar apenas pacotes HTTP GET (procurando "GET " no payload TCP):**

  Bash

  ```
  tcpdump -r web_traffic.pcap -nn -A -s 0 'tcp port 80 and (((ip[2:2] - ((ip&0xf)<<2)) - ((tcp[1]&0xf0)>>2))!= 0) and tcp[((tcp[1]&0xf0)>>2):4] = 0x47455420'
  ```

  (Este filtro complexo calcula o início do payload TCP e verifica os primeiros 4 bytes para "GET ". 0x47455420 é "GET " em hexadecimal ASCII)
- **Capturar tráfego HTTP (portas 80 ou 8080) e salvar em arquivo:**

  Bash

  ```
  sudo tcpdump -i any -s 0 -w http_traffic.pcap 'tcp port 80 or tcp port 8080'
  ```

tcpdump é uma ferramenta eficiente para captura direta. Sua sintaxe de filtro BPF, embora menos intuitiva inicialmente que os filtros de exibição do Wireshark, é extremamente otimizada para filtragem em tempo real no nível do kernel, tornando-a ideal para capturas de alto desempenho ou em ambientes com recursos limitados onde a sobrecarga do TShark pode ser um problema.

Aircrack-ng Suíte: Análise e Auditoria de Redes Wi-Fi

O Aircrack-ng é um conjunto abrangente de ferramentas projetado especificamente para avaliar a segurança de redes Wi-Fi.[4] Ele foca em monitoramento, ataque (injeção de pacotes), teste de capacidade de drivers/placas e cracking de chaves WEP e WPA/WPA2-PSK.

Componentes Principais da Suíte Aircrack-ng

- **airmon-ng:**
  - **Propósito:** Habilita e desabilita o **modo monitor** na interface de rede sem fio.[16] O modo monitor é essencial, pois permite que a placa capture todos os pacotes 802.11 ao alcance, não apenas aqueles destinados ao seu próprio endereço MAC.

Comandos:

- sudo airmon-ng: Lista interfaces sem fio e status.
- sudo airmon-ng check kill: Identifica e termina processos (como NetworkManager, wpa_supplicant) que podem interferir na operação do modo monitor ou injeção de pacotes. É altamente recomendável executar este comando antes de iniciar o modo monitor.
- sudo airmon-ng start <interface> (e.g., sudo airmon-ng start wlan0): Coloca a interface especificada em modo monitor. Frequentemente, isso cria uma interface virtual com um sufixo mon (e.g., wlan0mon, mon0, ou prism0 dependendo do driver) que deve ser usada com as outras ferramentas Aircrack-ng.
- sudo airmon-ng stop <monitor_interface> (e.g., sudo airmon-ng stop wlan0mon): Desabilita o modo monitor e retorna a interface ao modo gerenciado (managed).

airodump-ng:

- **Propósito:** Ferramenta de captura de pacotes 802.11 (sniffer). Descobre Access Points (APs) e clientes sem fio próximos, exibindo informações detalhadas como BSSID (MAC do AP), ESSID (nome da rede), canal, tipo de criptografia (WEP, WPA, WPA2), tipo de autenticação, velocidade e potência do sinal. É usado para coletar dados para cracking (IVs WEP ou handshakes WPA/WPA2).

Comandos:

- sudo airodump-ng <monitor_interface>: Inicia a varredura em todos os canais, mostrando todos os APs e clientes detectados.
- sudo airodump-ng --bssid <AP_MAC> -c <channel> -w <output_prefix> <monitor_interface>: Foca a captura em um AP específico (pelo BSSID) em um canal específico (-c), e salva os pacotes capturados em arquivos com o prefixo especificado (-w, e.g., capture. Gerará capture-01.cap, capture-01.csv, etc.). Este é o comando usado para capturar o handshake WPA/WPA2.

- **Captura de Handshake:** Quando airodump-ng captura com sucesso os pacotes EAPOL que compõem o handshake WPA/WPA2 para um determinado AP, ele exibirá a mensagem `` no canto superior direito da tela.

aireplay-ng:

- **Propósito:** Ferramenta de injeção de pacotes. Usada para realizar vários ataques ativos, sendo o mais comum o ataque de desautenticação (-0) para forçar clientes a se reconectarem ao AP, o que frequentemente desencadeia o handshake WPA/WPA2 necessário para o cracking.Também pode ser usado para ataques de replay ARP (para gerar IVs WEP), ataques de fragmentação, Caffe Latte, etc..

Comando (Deauthentication):

- sudo aireplay-ng -0 <count> -a <AP_MAC> [-c <client_MAC>] <monitor_interface>
  - -0: Especifica o ataque de desautenticação.
  - <count>: Número de deauths a enviar (e.g., 5). Um valor de 0 significa envio contínuo, efetivamente um ataque de negação de serviço (DoS) contra o(s) cliente(s).
  - -a <AP_MAC>: O BSSID do Access Point alvo.
  - -c <client_MAC>: (Opcional) O MAC address do cliente específico a ser desautenticado. Se omitido, aireplay-ng envia pacotes de desautenticação de broadcast, visando todos os clientes conectados ao AP. Atacar um cliente específico costuma ser mais eficaz.

aircrack-ng:

- **Propósito:** A ferramenta principal para quebrar as chaves de criptografia sem fio. Usa ataques estatísticos (como FMS/KoreK e PTW) para quebrar chaves WEP e ataques de dicionário para quebrar chaves WPA/WPA2-PSK usando o handshake capturado.

Comando (WPA/WPA2 Cracking):

- aircrack-ng -w <wordlist_file> -b <AP_MAC> <capture_file.cap>
  - -w <wordlist_file>: Caminho para o arquivo de dicionário contendo senhas potenciais (e.g., /usr/share/wordlists/rockyou.txt). Este é o componente **essencial** para o cracking WPA/WPA2.
  - -b <AP_MAC>: (Opcional, mas recomendado) filtra o ataque para o BSSID do AP alvo, útil se o arquivo .cap contém handshakes de múltiplos APs.
  - <capture_file.cap>: O arquivo .cap (ou múltiplos arquivos usando wildcard, e.g., capture*.cap) contendo o handshake WPA/WPA2 capturado por airodump-ng.

Comando (WEP Cracking):

- aircrack-ng <capture_file.cap>: Para WEP, aircrack-ng usa ataques estatísticos (principalmente PTW) que analisam os IVs capturados. Nenhum dicionário é necessário. A opção -n <bits> (e.g., -n 64) pode ser usada para especificar o tamanho da chave WEP.

Fluxo de Trabalho Básico para Cracking WPA/WPA2-PSK

O processo típico para tentar quebrar uma senha WPA/WPA2 usando Aircrack-ng envolve as seguintes etapas:

1. **Preparação da Interface:**
   - sudo airmon-ng check kill: Mata processos que podem interferir.
   - sudo airmon-ng start wlan0: Coloca a interface wlan0 (ou outra) em modo monitor. Anote o nome da nova interface monitor (e.g., wlan0mon).
2. **Descoberta do Alvo:**
   - sudo airodump-ng wlan0mon: Executa uma varredura para identificar a rede alvo. Anote seu BSSID (MAC address) e o Canal (CH) em que opera. Identifique também o MAC address de pelo menos um cliente conectado (STATION), se possível.

3. **Captura Focada do Handshake:**
    - sudo airodump-ng -c <channel> --bssid <BSSID> -w capture wlan0mon: Inicia a captura focada no AP e canal alvo, salvando os pacotes em arquivos com prefixo capture. Deixe este terminal rodando.
4. **Forçar o Handshake (Ataque de Deauthentication):**
    - ✓ Abra um **novo terminal**.
    - ✓ sudo aireplay-ng -0 5 -a <BSSID> [-c <Client_MAC>] wlan0mon: Envia 5 rajadas de pacotes de desautenticação para o AP (e opcionalmente para um cliente específico) para forçar uma reconexão e a troca do handshake.
5. **Verificação e Parada da Captura:**
    - ✓ Volte ao terminal do airodump-ng. Observe o canto superior direito. Quando a mensagem `` aparecer, o handshake foi capturado com sucesso.
    - ✓ Pressione Ctrl+C no terminal do airodump-ng para parar a captura. Você terá arquivos como capture-01.cap.
6. **Cracking com Dicionário:**
    - ✓ sudo aircrack-ng -w /path/to/wordlist.txt -b <BSSID> capture*.cap: Inicia o processo de cracking. aircrack-ng tentará cada senha do arquivo <wordlist.txt> contra o handshake capturado.
    - ✓ **Importante:** O sucesso depende **exclusivamente** de a senha real estar presente no dicionário fornecido (-w). Se a senha não estiver na lista, aircrack-ng não a encontrará, independentemente do tempo de execução.[16] Ataques de força bruta puros (tentar todas as combinações) são geralmente computacionalmente inviáveis contra WPA/WPA2.
7. **Limpeza:**
    - ✓ sudo airmon-ng stop wlan0mon: Desativa o modo monitor.

Este fluxo de trabalho demonstra a interação entre as principais ferramentas da suíte Aircrack-ng para um objetivo comum. É crucial notar a natureza ativa e potencialmente disruptiva do ataque de desautenticação (aireplay-ng -0), que deve ser usado com responsabilidade e apenas em redes autorizadas.

Bettercap: Framework Modular para MITM e Reconhecimento

Bettercap (versão 2.x e posteriores) é um framework poderoso e modular escrito em Go, projetado como um "canivete suíço" para reconhecimento de rede e ataques Man-in-the-Middle (MITM). Ele opera principalmente através de um shell interativo, onde diferentes módulos podem ser iniciados, parados e configurados dinamicamente. Bettercap suporta ataques em redes IPv4 e IPv6, Wi-Fi, dispositivos Bluetooth Low Energy (BLE) e até mesmo dispositivos HID sem fio (como teclados/mouses).

Visão Geral e Modo Interativo

- **Lançamento:** Para iniciar o Bettercap em modo interativo, geralmente com privilégios de root:

  Bash

  `sudo bettercap [-iface <interface>]`

  A opção -iface permite especificar a interface de rede a ser usada.
- **Comandos Core:** O shell interativo oferece comandos básicos para gerenciamento da sessão e dos módulos :
  - help [nome_modulo]: Mostra ajuda geral ou específica de um módulo.
  - active: Lista os módulos atualmente em execução e seus parâmetros.
  - quit ou q: Encerra a sessão.
  - sleep <segundos>: Pausa a execução.
  - set <parametro> <valor>: Define o valor de um parâmetro de módulo (e.g., set arp.spoof.targets 192.168.1.10).
  - get <parametro>: Obtém o valor atual de um parâmetro (* para todos).
  - clear: Limpa a tela do terminal.
  - include <caplet>: Carrega e executa um arquivo de caplet (script .cap).
  - ! <comando_shell>: Executa um comando do shell do sistema operacional.
- **Gerenciamento de Módulos:** Módulos são ativados com [nome_modulo] on e desativados com [nome_modulo] off.

Módulos Principais para Análise de Rede e MITM

Reconhecimento de Rede (net.recon):

- **Objetivo:** Descobrir hosts ativos na rede local.
- **Comandos:** net.recon on (iniciar), net.recon off (parar), net.show (exibir hosts encontrados).
- **Parâmetros:** set net.recon.targets <range_IP> (e.g., 192.168.1.0/24) para definir o escopo da varredura.

Spoofing ARP (arp.spoof):

- **Objetivo:** Realizar ataques de envenenamento do cache ARP para interceptar tráfego entre alvos na rede local (MITM).
- **Comandos:** arp.spoof on (iniciar), arp.spoof off (parar).
  - **Parâmetros:**
    - set arp.spoof.targets <IP1,IP2,...>: Define os IPs dos alvos. Comumente, inclui-se o IP da vítima e o IP do gateway.
    - set arp.spoof.fullduplex true: Habilita o spoofing bidirecional (vítima <-> atacante <-> gateway), geralmente necessário para uma interceptação completa.

Spoofing DNS (dns.spoof):

- **Objetivo:** Interceptar requisições DNS das vítimas (requer MITM ativo, como ARP spoofing) e responder com endereços IP falsificados, redirecionando-as para servidores controlados pelo atacante.
- **Comandos:** dns.spoof on (iniciar), dns.spoof off (parar).
- **Parâmetros:**
  - set dns.spoof.domains <dominio1,dominio2,...>: Especifica quais domínios interceptar (* para todos).
  - set dns.spoof.address <IP_falso>: O endereço IP para o qual os domínios serão resolvidos (geralmente o IP da máquina do atacante).
  - set dns.spoof.all true: Spoofa todas as requisições DNS.

Proxy HTTP (http.proxy) e HTTPS (https.proxy):

- ✓ **Objetivo:** Interceptar tráfego web das vítimas (requer MITM ativo). O proxy HTTP permite visualizar e modificar tráfego não criptografado. O proxy HTTPS tenta interceptar tráfego SSL/TLS, o que exige a instalação de um certificado CA raiz do Bettercap na máquina da vítima para evitar alertas de segurança no navegador.
- ✓ **Comandos:** http.proxy on/off, https.proxy on/off.
  - ➢ **Parâmetros:**
    - ■ set http.proxy.port <porta> (padrão 8080).
    - ■ set https.proxy.port <porta> (padrão 8081).
    - ■ set http.proxy.script <arquivo.rb>: Carrega um script Ruby para manipulação dinâmica de requisições/respostas HTTP.
    - ■ set https.proxy.script <arquivo.rb>: Similar para HTTPS.

Sniffer de Rede (net.sniff):

- – **Objetivo:** Capturar e analisar passivamente o tráfego de rede, com parsers integrados para extrair credenciais de diversos protocolos (HTTP Auth, FTP, IRC, Mail, etc.).
- – **Comandos:** net.sniff on (iniciar), net.sniff off (parar).
- – **Parâmetros:**
  - ✓ set net.sniff.filter <filtro_bpf>: Aplica um filtro BPF à captura.
  - ✓ set net.sniff.output <arquivo.pcap>: Salva o tráfego capturado em um arquivo pcap.
  - ✓ set net.sniff.local true: Inclui o tráfego da própria máquina na análise.

Modularidade e Interdependência

A força do Bettercap reside na sua arquitetura modular, permitindo combinar diferentes funcionalidades. No entanto, é crucial entender que muitos módulos dependem de outros para funcionar corretamente. Especificamente, os módulos que visam interceptar ou manipular o tráfego de uma vítima (dns.spoof, http.proxy, https.proxy, net.sniff quando focado em outros hosts) geralmente **requerem** que um módulo de spoofing (arp.spoof para IPv4 ou ndp.spoof para IPv6) esteja ativo e

configurado corretamente. Sem o redirecionamento do tráfego via MITM, esses módulos não terão acesso aos pacotes da vítima. Ao planejar um ataque ou análise com Bettercap, deve-se primeiro estabelecer o MITM (e.g., arp.spoof on) e depois ativar os módulos de interceptação desejados.

Exemplos Práticos no Shell Interativo

- **Reconhecimento e ARP Spoofing Completo:**

  Bash

  ```
  net.recon on
  # Esperar alguns segundos para descoberta
  net.show
  # Identificar IPs da rede (e.g., 192.168.1.0/24)
  set arp.spoof.targets 192.168.1.0/24
  set arp.spoof.fullduplex true
  arp.spoof on
  ```

- **ARP Spoofing e DNS Spoofing para um Site Específico:**

  Bash

  ```
  # Assumindo que net.recon on já rodou e identificou vítima (192.168.1.15) e gateway (192.168.1.1)
  set arp.spoof.targets 192.168.1.15, 192.168.1.1
  set arp.spoof.fullduplex true
  arp.spoof on
  set dns.spoof.domains banking.example.com
  set dns.spoof.address 192.168.1.100 # IP do atacante com página falsa
  dns.spoof on
  ```

- **ARP Spoofing e Captura de Credenciais HTTP:**

  Bash

  ```
  # Assumindo ARP spoofing ativo (comandos anteriores)
  net.sniff on
  # Observar a saída para credenciais capturadas (e.g., HTTP Basic Auth)
  (
  ```

Hydra: Teste de Força Bruta Online contra Serviços de Rede

THC-Hydra é uma ferramenta de linha de comando altamente reconhecida, projetada para realizar ataques de força bruta rápidos e paralelos contra serviços de autenticação de rede.[4] Sua flexibilidade permite atacar uma vasta gama de protocolos que exigem login e senha, tornando-a uma ferramenta comum em testes de penetração para verificar a força de senhas e políticas de bloqueio de contas.

Protocolos Suportados (Exemplos)

Hydra suporta dezenas de protocolos, incluindo :

- FTP, FTPS
- SSH (v1 e v2)
- Telnet
- SMTP, SMTP-ENUM
- POP3, POP3S
- IMAP, IMAPS
- HTTP(S) Basic/Digest Auth
- HTTP(S)-GET-FORM, HTTP(S)-POST-FORM
- SMB (NTLM)
- RDP (via módulo externo ou opção específica)
- VNC
- SNMP
- MySQL, PostgreSQL, MS-SQL
- LDAP
- E muitos outros...

Opções Essenciais da Linha de Comando

A sintaxe básica do Hydra é hydra [opções] service://server.

- **Especificação de Login/Senha:**
  - -l LOGIN: Fornece um único nome de usuário para testar.
  - -L FILE: Fornece um arquivo com uma lista de nomes de usuário (um por linha).

- -p PASS: Fornece uma única senha para testar.[11]
- -P FILE: Fornece um arquivo com uma lista de senhas (dicionário, um por linha).
- -C FILE: Fornece um arquivo com combinações login:senha (um par por linha).

Controle de Ataque:

- -t TASKS: Define o número de conexões paralelas (threads) a serem usadas. O padrão é 16. É crucial ajustar este valor, especialmente para serviços como SSH que podem bloquear ou limitar conexões rápidas. Um valor baixo como -t 4 é frequentemente recomendado para SSH. Usar muitas threads pode sobrecarregar o servidor alvo ou acionar mecanismos de defesa.
- -f / -F: -f faz o Hydra parar de testar senhas para um host específico assim que um par válido é encontrado. -F faz o Hydra parar completamente (mesmo ao atacar múltiplos hosts com -M) assim que o primeiro par válido é encontrado em qualquer host.
- -e nsr: Realiza checagens adicionais: n (senha nula/vazia), s (tenta o login como senha), r (tenta o login reverso como senha).

Saída:

- -o FILE: Salva os pares login/senha encontrados no arquivo especificado.
- -b FORMAT: Especifica o formato do arquivo de saída (text, json, jsonv1).
- -vV: Modo verbose/very verbose. Exibe cada tentativa de login e senha.

Alvo e Serviço:

- ✓ service://server: Define o protocolo (ssh, ftp, http-post-form, etc.), o endereço do servidor (IP ou hostname), a porta (opcional se for a padrão) e opções específicas do módulo (OPT).
- ✓ -s PORT: Permite especificar a porta do serviço se ela não for a padrão.
- ✓ -M FILE: Fornece um arquivo com uma lista de servidores para atacar em paralelo.

Sintaxe Específica e Exemplos Práticos

- **SSH:**
  - Atacar o IP 192.168.1.10 com o usuário admin e a lista de senhas pass.txt, usando 4 threads:

Bash

```
hydra -l admin -P pass.txt 192.168.1.10 ssh -t 4
```

- Atacar múltiplos IPs de um arquivo ssh_hosts.txt com a lista de usuários users.txt e a senha password123, saindo após a primeira credencial encontrada em qualquer host:

  Bash

  ```
  hydra -L users.txt -p password123 -M ssh_hosts.txt ssh -F -t 4
  ```

- **FTP:**
  - Atacar ftp.example.com com o usuário guest e a lista ftp_passwords.txt:

    Bash

    ```
    hydra -l guest -P ftp_passwords.txt ftp.example.com ftp
    ```

- **HTTP POST Form:**
  - Atacar um formulário de login em http://10.0.0.20/login.php. O formulário envia os campos log (usuário) e pwd (senha). A resposta em caso de falha contém a string "Username or password invalid". Usar a lista de usuários common_users.txt e a lista de senhas rockyou.txt:

    Bash

    ```
    hydra -L common_users.txt -P /usr/share/wordlists/rockyou.txt 10.0.0.20 http-post-form "/login.php:log=^USER^&pwd=^PASS^:F=Username or password invalid" -V
    ```

- **Explicação da String do Formulário:**

  "/path/do/form:campo_user=^USER^&campo_senha=^PASS^:Indicador_Falha_ou_Sucesso[:Headers_Opcionais]"

  - /login.php: O caminho para o qual o formulário é enviado.
  - log=^USER^&pwd=^PASS^: Os dados do formulário, com ^USER^ e ^PASS^ como placeholders que o Hydra substituirá a cada tentativa. Os nomes dos campos (log, pwd) devem corresponder aos nomes reais no HTML do formulário.

    - F=Username or password invalid: O indicador de falha. F= significa "Falha se encontrar esta string". Alternativamente, S= significaria "Sucesso se encontrar esta string". É crucial inspecionar a resposta do servidor para uma tentativa de login inválida para encontrar uma string confiável para F= ou S=.
    - Headers Opcionais: Pode-se adicionar headers customizados ou especificar cookies usando H=Header: Value ou C=/path/para/obter/cookie.
- **HTTPS POST Form:**
    - Similar ao HTTP, mas usando https-post-form e geralmente a porta 443 (implícita ou com -s 443):

      Bash

      ```
      hydra -l carlos -P /usr/share/wordlists/fasttrack.txt ac311f351ebe430c80453e6300b10013.web-security-academy.net https-post-
      ```

Hydra é uma ferramenta poderosa, mas seu uso deve ser ético e legal. Ataques de força bruta podem sobrecarregar servidores e acionar sistemas de detecção de intrusão ou bloqueio de contas. É essencial ter permissão explícita antes de usar Hydra contra qualquer sistema.

Nikto: Scanner de Vulnerabilidades Web Específico

Nikto é um scanner de vulnerabilidades web open-source, escrito em Perl, que se concentra em identificar problemas de segurança comuns em servidores e aplicações web. Diferente de scanners de rede gerais como Nmap, Nikto é especializado em testes web, verificando milhares de arquivos/CGIs potencialmente perigosos, versões desatualizadas de servidores e problemas específicos de software, além de configurações incorretas do servidor (como métodos HTTP habilitados ou falta de headers de segurança).

Funcionalidades e Opções Essenciais

- **Base de Dados Extensa:** Verifica mais de 6700 arquivos/CGIs perigosos, 1250+ servidores desatualizados e 270+ problemas específicos de versão.
- **Atualizações:** O banco de dados de testes pode ser atualizado via linha de comando (nikto -update).
- **Opções de Alvo:**
  - ✓ -h ou -host <target>: Especifica o alvo. Pode ser um IP, hostname, URL completo (http://example.com:8080) ou um arquivo contendo múltiplos alvos.
  - ✓ -p ou -port <port>: Define a porta a ser escaneada (padrão 80). Aceita portas únicas, listas separadas por vírgula (e.g., 80,8080,443) ou ranges (e.g., 80-90).
- **Opções de SSL:**
  - ✓ -ssl: Força a varredura usando SSL/TLS nas portas especificadas. Isso acelera significativamente a varredura em portas HTTPS, pois evita a tentativa inicial de conexão HTTP.
- **Opções de Saída:**
  - ➢ -o ou -output <file>: Salva os resultados da varredura em um arquivo. A extensão do arquivo geralmente determina o formato.
  - ➢ -Format <format>: Específica explicitamente o formato de saída: txt (texto plano), csv (valores separados por vírgula), htm (HTML) ou xml (XML).
- **Opções de Tuning (-Tuning ou -T):**
  - Permite selecionar categorias específicas de testes a serem executados, tornando a varredura mais focada. Usar qualquer opção de tuning desabilita todos os outros testes, a menos que a opção x (reverse tuning) seja usada para *excluir* os testes especificados.

Tabela 3: Códigos de Tuning Comuns do Nikto

Código	Tipo de Teste	Descrição Breve
0	File Upload	Verifica vulnerabilidades de upload de arquivos.
1	Interesting File / Seen in logs	Procura por arquivos interessantes ou comuns em logs.
2	Misconfiguration / Default File	Verifica configurações incorretas ou arquivos padrão.
3	Information Disclosure	Busca por vazamento de informações sensíveis.
4	Injection (XSS/Script/HTML)	Testa vulnerabilidades de injeção (XSS, etc.).
5	Remote File Retrieval (Web Root)	Tenta buscar arquivos remotamente dentro do web root.
6	Denial of Service	Verifica potenciais vetores de DoS.
7	Remote File Retrieval (Server Wide)	Tenta buscar arquivos remotamente em todo o servidor.

8	Command Execution / Remote Shell	Testa execução remota de comandos.
9	SQL Injection	Verifica vulnerabilidades de injeção SQL.
a	Authentication Bypass	Tenta contornar mecanismos de autenticação.
b	Software Identification	Identifica software específico em execução.
c	Remote Source Inclusion	Testa inclusão remota de código fonte.
x	Reverse Tuning	Exclui os testes especificados em vez de incluí-los.

- **Outras Opções Úteis:**
  - -Plugins <list>: Executa apenas os plugins Nikto especificados (use -list-plugins para ver os nomes).
  - -evasion <techniques>: Aplica técnicas de evasão de IDS/WAF (codificadas numericamente de 1 a 8).
  - -useproxy <proxy_url>: Roteia a varredura através de um proxy HTTP (e.g., http://127.0.0.1:8080).
  - -Pause <seconds>: Adiciona um atraso entre as requisições para reduzir a carga no servidor e a chance de detecção.
  - -id <id:password>: Fornece credenciais de autenticação básica HTTP.
  - -useragent <string>: Define um User-Agent customizado para as requisições.

Exemplos Práticos com Nikto

- **Varredura Básica (HTTP na porta 80):**

  Bash

  ```
  nikto -h example.com
  ```

  ou

  Bash

  ```
  nikto -h 192.168.0.10
  ```

- **Varredura HTTPS na Porta 443 e Salvar em HTML:**

  Bash

  ```
  nikto -h 192.168.0.10 -p 443 -ssl -o scan_report.html -Format htm
  ```

- **Varredura Focada em Misconfiguration (2) e Information Disclosure (3):**

  Bash

  ```
  nikto -h http://test.local -Tuning 23
  ```

- **Varredura Excluindo Testes de Software Identification (b):**

  Bash

  ```
  nikto -h http://app.internal -Tuning xb
  ```

- **Varredura de Múltiplos Hosts de um Arquivo:**

  Bash

  ```
  nikto -h targets.txt -o results.csv -Format csv
  ```

- **Varredura via Proxy Burp Suite:**

  Bash

  ```
  nikto -h http://target.local -useproxy http://127.0.0.1:8080
  ```

Interpretação dos Resultados

Nikto fornece uma saída detalhada listando os problemas encontrados, geralmente com um ID OSVDB (embora o OSVDB esteja desativado, o ID pode ser um ponto de partida para pesquisa) e uma breve descrição. É crucial entender que Nikto, como qualquer scanner automatizado, pode gerar falsos positivos. Por exemplo, ele pode sinalizar um "arquivo interessante" que é, na verdade, benigno no contexto da aplicação específica. Portanto, os resultados do Nikto devem ser sempre **validados manualmente**. Verifique os arquivos reportados, tente explorar as vulnerabilidades de configuração ou injeção sugeridas e use o bom senso para avaliar o risco real de cada item reportado antes de incluí-lo em um relatório final.

Outras Ferramentas Notáveis para Análise de Rede no Kali Linux

Além das ferramentas detalhadas acima, o Kali Linux oferece uma vasta gama de outros utilitários valiosos para diferentes aspectos da análise de rede e segurança:

- **Metasploit Framework:** Embora seja primariamente uma plataforma de exploração, o Metasploit contém numerosos **módulos auxiliares (auxiliary)** dedicados a tarefas de reconhecimento e varredura. Existem módulos para varredura de portas (TCP, UDP), descoberta de hosts (ARP, SNMP, NetBIOS), enumeração de serviços específicos (SMB, SSH, FTP, etc.), verificação de credenciais padrão e até mesmo varredura de vulnerabilidades básicas. A integração com Nmap e outros scanners é comum.
- **sqlmap:** Ferramenta líder para detecção e exploração automatizada de vulnerabilidades de **injeção SQL** em aplicações web. Pode identificar falhas, enumerar bancos de dados, extrair dados e até obter shells em alguns casos.
- **John the Ripper / Hashcat:** Ferramentas poderosas para **cracking de senhas offline**. Elas operam sobre hashes de senhas (obtidos de sistemas comprometidos, dumps de bancos de dados, etc.) e usam ataques de dicionário, força bruta e regras de mutação para descobrir as senhas originais. Hashcat é otimizado para usar GPUs, acelerando drasticamente o processo.
- **Netcat (nc / ncat):** O "canivete suíço" original para interações TCP/UDP básicas,

transferência de arquivos simples, escuta de portas e conexões reversas/bind shells. Ncat (incluído com Nmap) é a versão moderna com mais recursos.

- **Responder:** Especializado em ataques de envenenamento de nomes na rede local (LLMNR, NBT-NS, mDNS). Ele responde a requisições desses protocolos, geralmente capturando hashes de senha NTLMv1/v2 de clientes Windows que podem ser quebrados offline.
- **Enum4linux / enum4linux-ng:** Ferramentas para enumeração detalhada de informações de sistemas Windows/Samba via SMB/NetBIOS, como usuários, grupos, shares, políticas de senha, etc.
- **Gobuster / Dirb / Dirbuster / ffuf:** Ferramentas para **brute-force de diretórios e arquivos** em servidores web. Usam listas de palavras para descobrir conteúdo oculto ou não referenciado que pode conter informações sensíveis ou funcionalidades vulneráveis.
- **Burp Suite:** Proxy de interceptação e suíte de ferramentas para testes de segurança de aplicações web. Permite inspecionar, modificar e repetir requisições HTTP/HTTPS, além de possuir scanners e ferramentas de ataque integradas.A versão Community está incluída no Kali.
- **Outros Scanners:** Ferramentas como arp-scan (descoberta ARP), masscan (scanner TCP assíncrono extremamente rápido), WhatWeb (identificação de tecnologias web), dnsenum / sublist3r / assetfinder (enumeração de subdomínios) também são valiosas para fases específicas da análise.

A escolha da ferramenta depende da tarefa específica, do ambiente alvo e das permissões disponíveis.

Exemplos de Uso Combinado (Fluxos de Trabalho)

Nenhuma ferramenta isolada fornece uma visão completa da segurança de uma rede. A verdadeira eficácia vem da combinação estratégica de diferentes ferramentas em um fluxo de trabalho coeso, onde a saída de uma ferramenta informa a próxima etapa. Abaixo estão alguns cenários comuns ilustrando como essas ferramentas podem ser usadas em conjunto.

Cenário 1: Análise de Rede Externa e Interna (Pentest Típico)

Este fluxo simula um teste de penetração padrão, começando do exterior e movendo-se para o interior após um comprometimento inicial.

1. **Reconhecimento Externo (Nmap):**
    - Objetivo: Identificar hosts vivos e serviços expostos na rede alvo (e.g., servidores web, mail, VPN).
    - Comando Exemplo: sudo nmap -sS -sV -p- --min-rate=1000 -T4 -oA external_scan <IP_Range_Alvo>
    - Análise: Procurar por portas abertas, especialmente aquelas rodando serviços conhecidos ou versões desatualizadas.
2. **Varredura de Aplicações Web (Nikto / Nmap NSE / Burp Suite):**
    - ✓ Objetivo: Aprofundar a análise nos servidores web identificados na etapa 1.
    - ✓ Comando Exemplo (Nikto): nikto -h http://<IP_Alvo>:<Porta> -o nikto_report.txt
    - ✓ Comando Exemplo (Nmap NSE): sudo nmap -p 80,443 --script http-enum, http-vuln-* -oN nmap_web_vuln.txt <IP_Alvo>
    - ✓ Análise: Buscar por vulnerabilidades comuns (XSS, SQLi - embora Nikto/Nmap não sejam os melhores para isso), arquivos de configuração expostos, diretórios padrão, software desatualizado. Usar Burp Suite para análise manual mais profunda.
3. **Teste de Credenciais (Hydra):**
    - ✓ Objetivo: Tentar adivinhar senhas para serviços expostos que requerem autenticação (SSH, FTP, RDP, login web).
    - ✓ Comando Exemplo (SSH): hydra -L known_users.txt -P common_passwords.txt <IP_Alvo> ssh -t 4 -o hydra_ssh.log
    - ✓ Análise: Verificar se alguma credencial padrão ou fraca permite acesso.
4. **Exploração (Metasploit Framework):**
    - ✓ Objetivo: Se vulnerabilidades exploráveis foram encontradas (pelo Nmap NSE, Nikto, ou análise manual), usar Metasploit para tentar obter acesso.
    - ✓ Comando Exemplo: Iniciar msfconsole, procurar (search) por um exploit correspondente à vulnerabilidade/serviço, configurar (set RHOSTS, set

PAYLOAD, etc.) e executar (exploit ou run).

5. **Pós-Exploração e Análise Interna (Nmap, Wireshark/tcpdump):**
    - Objetivo: Uma vez dentro da rede, usar Nmap novamente para mapear a rede interna (sudo nmap -sS -T4 -A 192.168.x.0/24). Usar Wireshark ou tcpdump para capturar tráfego interno em busca de informações sensíveis ou outros alvos.
    - Análise: Identificar sistemas críticos internos, capturar credenciais em texto plano, analisar padrões de tráfego.

Cenário 2: Auditoria de Segurança de Rede Wi-Fi (WPA2-PSK)

Este fluxo foca na avaliação da segurança de uma rede Wi-Fi protegida por WPA2 com chave pré-compartilhada.

1. **Preparar Interface (Airmon-ng):**
    - sudo airmon-ng check kill
    - sudo airmon-ng start wlan0 (Anotar nome da interface monitor, e.g., wlan0mon)
2. **Identificar Alvo (Airodump-ng):**
    - sudo airodump-ng wlan0mon (Identificar BSSID e Canal da rede alvo).
3. **Capturar Handshake (Airodump-ng + Aireplay-ng):**
    - Terminal 1: sudo airodump-ng -c <Canal> --bssid <BSSID_Alvo> -w wifi_capture wlan0mon
    - Terminal 2: sudo aireplay-ng -0 5 -a <BSSID_Alvo> wlan0mon (Enviar deauths para forçar reconexão e handshake).
    - Observar Terminal 1 até `` aparecer. Parar airodump-ng (Ctrl+C).
4. **Tentar Quebrar a Senha (Aircrack-ng):**
    - sudo aircrack-ng -w /path/to/wordlist.txt -b <BSSID_Alvo> wifi_capture*.cap
    - Análise: Se a senha for encontrada, ela está no dicionário. Se não, o dicionário é inadequado ou a senha é forte.

5. **Análise Pós-Acesso (Nmap, Wireshark - Opcional):**
    - Se a senha for quebrada e o acesso à rede for obtido, usar Nmap para escanear a rede interna e Wireshark para analisar o tráfego como no Cenário 1.
6. **Limpeza (Airmon-ng):**
    - sudo airmon-ng stop wlan0mon

Cenário 3: Ataque Man-in-the-Middle (MITM) na Rede Local

Este fluxo demonstra como interceptar tráfego entre uma vítima e o gateway na rede local. **Nota:** Realizar ataques MITM sem autorização é ilegal e antiético.

1. **Descoberta de Alvos (Bettercap):**
    - sudo bettercap -iface eth0 (Iniciar Bettercap na interface correta).
    - net.recon on (Ativar descoberta de hosts).
    - net.show (Listar hosts e identificar IP da vítima e do gateway).
2. **Executar ARP Spoofing (Bettercap):**
    - set arp.spoof.targets <IP_Vítima>,<IP_Gateway>
    - set arp.spoof.fullduplex true
    - arp.spoof on (Iniciar o envenenamento ARP).
3. **Interceptar/Manipular Tráfego (Bettercap / Wireshark):**
    - **Captura de Credenciais:** net.sniff on (Monitorar saída para credenciais capturadas).
    - **DNS Spoofing:** set dns.spoof.domains sitealvo.com; set dns.spoof.address <IP_Atacante>; dns.spoof on (Redirecionar tráfego para um site falso).
    - **Proxy HTTP:** http.proxy on (Interceptar tráfego HTTP. Pode-se usar set http.proxy.script para modificação).
    - **Análise Detalhada:** Executar sudo wireshark ou sudo tcpdump -i eth0 -w mitm_capture.pcap na máquina atacante para capturar todo o tráfego redirecionado para análise offline detalhada.
4. **Parar Ataque (Bettercap):**
    - arp.spoof off (Parar o spoofing ARP é crucial para restaurar a conectividade normal).

- Desativar outros módulos (net.sniff off, dns.spoof off, etc.).
- quit

Estes cenários ilustram como as ferramentas se complementam. O Nmap fornece o mapa inicial, Nikto e Hydra investigam serviços específicos, Aircrack-ng foca no Wi-Fi, Bettercap permite ataques MITM, e Wireshark/tcpdump oferecem a capacidade de inspecionar o tráfego em qualquer fase.

Conclusão

O Kali Linux fornece um ecossistema excepcionalmente rico e poderoso de ferramentas para análise de redes e testes de segurança. Desde a varredura inicial e descoberta de hosts com **Nmap**, passando pela análise profunda de pacotes com **Wireshark/TShark** e **tcpdump**, auditoria de redes sem fio com **Aircrack-ng**, ataques Man-in-the-Middle com **Bettercap**, testes de força bruta com **Hydra**, até a varredura de vulnerabilidades web com **Nikto**, cada ferramenta desempenha um papel vital.

A eficácia dessas ferramentas, no entanto, não reside apenas em suas capacidades individuais, mas na habilidade do analista ou pentester em combiná-las estrategicamente. Um fluxo de trabalho bem planejado, onde os resultados de uma ferramenta informam o uso da próxima, é essencial para uma análise abrangente. Por exemplo, Nmap identifica um servidor web, Nikto detalha suas vulnerabilidades, Hydra testa suas credenciais, e Wireshark monitora a interação.

É fundamental ressaltar que o poder dessas ferramentas exige responsabilidade. A realização de varreduras, ataques ou interceptação de tráfego em redes sem autorização explícita é ilegal e antiética. O conhecimento apresentado neste guia deve ser aplicado apenas em ambientes controlados, laboratórios de teste ou em auditorias de segurança devidamente autorizadas.

Dominar essas ferramentas requer estudo contínuo e prática. A compreensão profunda das opções de linha de comando, da sintaxe de filtros, dos protocolos de rede subjacentes e das implicações de cada ação é o que diferencia um usuário casual de um profissional de segurança eficaz. Este guia serve como um ponto de partida robusto, mas

a exploração das documentações oficiais (man pages, sites dos projetos) e a experimentação prática são indispensáveis para alcançar a proficiência.