Basic Security Tools Overview

This guide introduces fundamental, widely recognized cybersecurity tools with their main uses, features, and example commands or usage notes.

1 NMAP: Network Scanning Tool

An open-source utility used for network discovery and security auditing.

Purpose

Helps in identifying hosts and services on a network, as well as discovering vulnerabilities.

Common Uses

- Network mapping
- Security auditing
- Vulnerability detection

Key Features

- Port Scanning: Identifies open ports on a target system.
- Service Detection: Determines the services running on open ports.
- OS Detection: Identifies the operating system of the target host.
- Scriptable: Supports scripts for advanced scanning and exploitation.

Example Nmap Commands

```
$ nmap <target-ip>                                  # Scan a Single IP Address

$ nmap <target-ip>/<subnet>                         # Scan a Subnet

$ nmap -p <ports-comma-separated> <target-ip>       # Scan specific ports

$ nmap -sV <target-ip>                              # Service Version Detection

$ nmap -O <target-ip>                               # Scan a Single IP Address
```

Figure 1: Nmap usage

Common Flags:

- `-p`: Specify ports to scan
- `-sV`: Service/version detection
- `-O`: Operating system detection
- `-A`: Aggressive scan (OS, service, script detection)
- `-sn`: Ping scan (host discovery only)
- `-v`: Verbose output
- `-iL <file>`: Scan targets from file

2 Hydra: Brute-force Tool

A powerful, parallelized brute-force tool commonly used in penetration testing.

Purpose

Used to test the strength of passwords on various network services by attempting login with multiple credentials.

Common Uses

- Testing login security on SSH, FTP, HTTP, and other protocols.
- Performing password spraying and dictionary attacks.

Key Features

- Supports multiple protocols for brute-forcing.
- Allows single or multiple usernames and passwords.
- Highly configurable with parallel task support.
- Provides verbose output and logging.

Example Hydra Commands

```
# Password spraying on SSH (list of users, single password)
$ hydra -L users.txt -p <fixed-password> <target-ip> ssh

# Dictionary attack on SSH (user and password lists)
$ hydra -L users.txt -P /usr/share/passwords.txt <target-ip> ssh

# Password guess with hints on SSH
$ hydra -l <fixed-username> -V -x 4:4:aA1 <target-ip> ssh
```

Common Flags:

- `-l`: Single username
- `-L`: Username list
- `-p`: Single password
- `-P`: Password list
- `-o`: Output file
- `-V`: Verbose mode
- `-x MIN:MAX:CHARSET`: Password generation rules

3 Wireshark: Network Protocol Analyzer

An open-source network packet capture and analysis tool widely used for detailed inspection of network traffic.

Purpose

Provides deep visibility into network communication by capturing packets and dissecting protocols, helping troubleshoot, monitor, and secure networks.

Common Uses

- Troubleshooting network connectivity and configuration issues.
- Analyzing protocol behavior and data exchanges.
- Detecting suspicious or malicious network activity.
- Learning and debugging network protocols.

Key Features

- Capture Interfaces Window: List of available network devices.
- Packet List Pane: Summary of captured packets.
- Color Coding: Highlights packets based on filter rules.
- Filter Bar: Where capture and display filters are applied.
- Statistics Menu: Graphs and protocol hierarchies.

Figure 2: Network Interfaces to capture traffic from

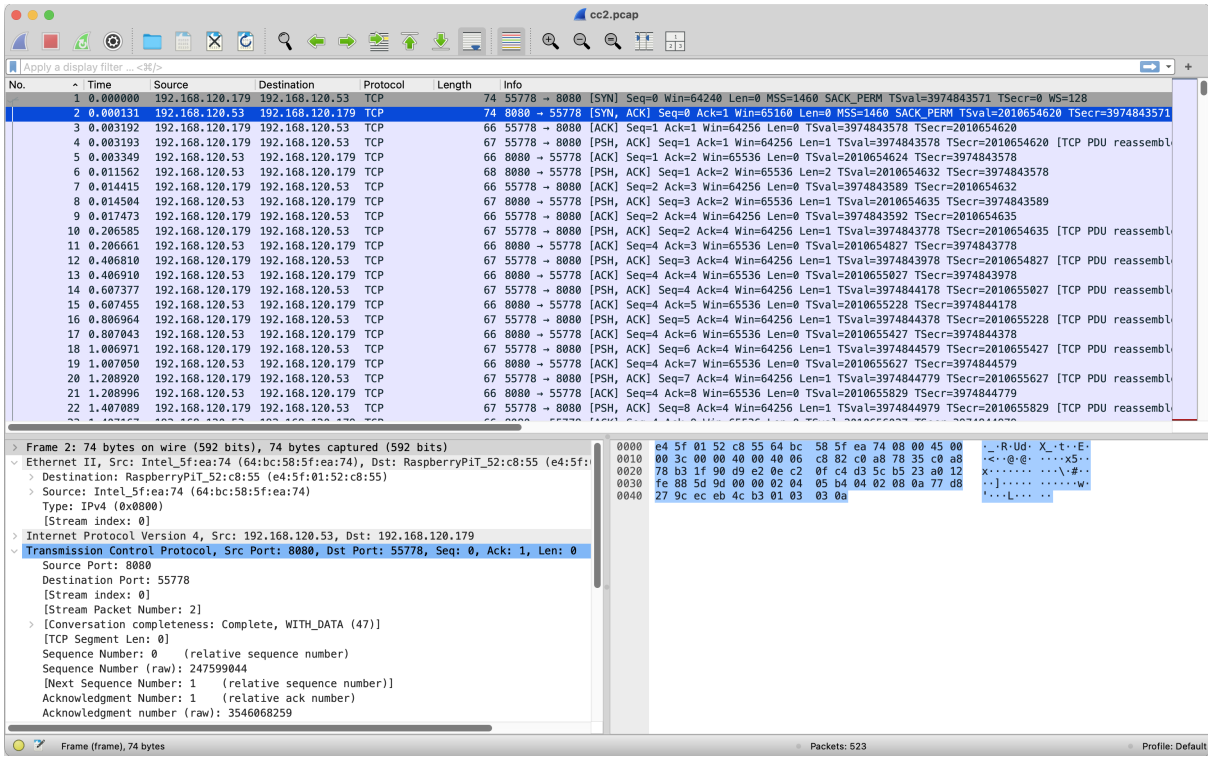

Figure 3: List of packets with details

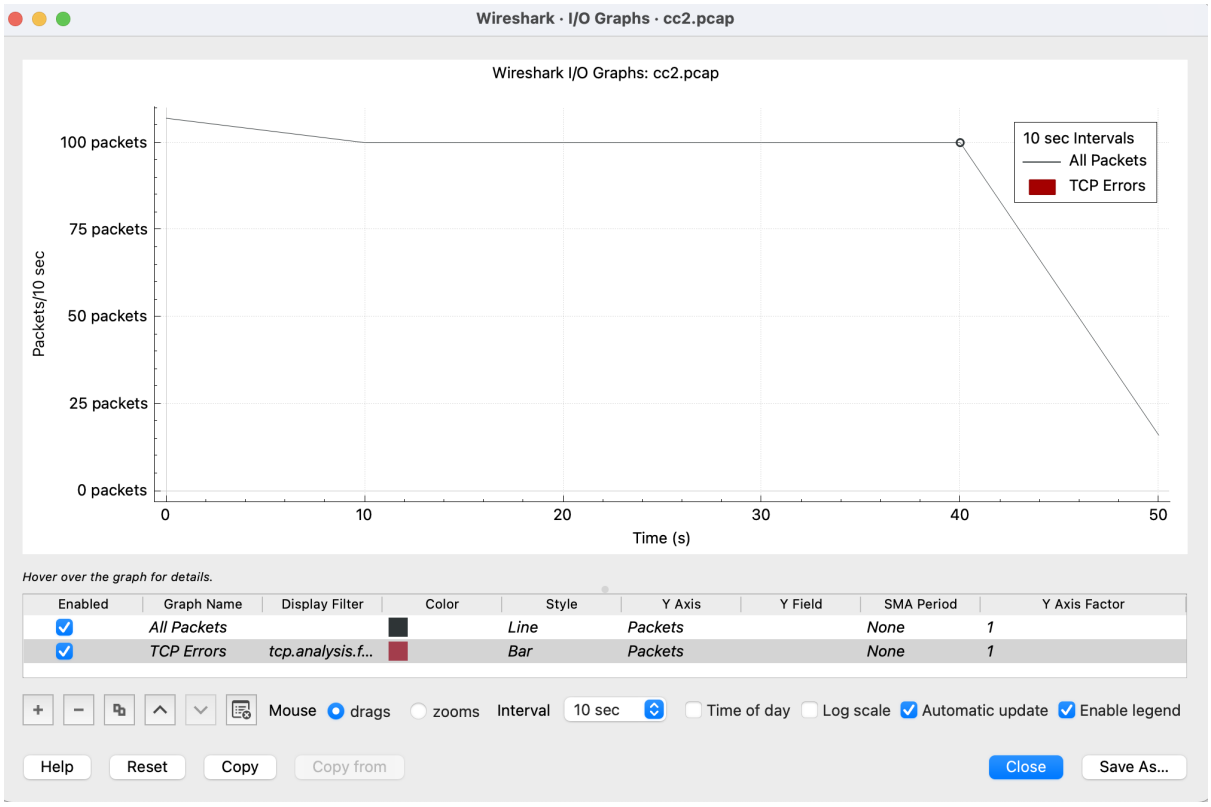

Figure 4: I/O Graph showcasing timing of packets

4 Metasploit Framework: Exploitation Tool

A comprehensive platform for developing, testing, and executing exploits against vulnerable systems.

Purpose

Helps penetration testers and security researchers automate and conduct security assessments by leveraging a vast library of exploits and payloads.

Common Uses

- Developing and running exploit code.
- Managing and executing payloads.
- Performing post-exploitation tasks.
- Generating vulnerability reports.

Key Features

- Extensive exploit and payload database.
- Integration with auxiliary modules for scanning and fuzzing.
- Command-line control using `msfconsole`.
- Cross-platform and multi-protocol support.
- Automation with resource scripts.

Example Metasploit Usage

```
$ msfconsole                                          # Start Metasploit console

msf > search <exploit-name/version/CVE>               # Search for exploits

msf > use <exploit-id>                                # Use an exploit module

msf exploit(<exploit-name>) > show options            # List exploit specific settings

msf exploit(<exploit-name>) > set <OPTION> <value>    # Set target options

msf exploit(<exploit-name>) > exploit                 # Launch the exploit
```

```
  YOU DIDN'T SAY THE MAGIC WORD!
  YOU DIDN'T SAY THE MAGIC WORD!
  YOU DIDN'T SAY THE MAGIC WORD!


       =[ metasploit v6.4.92-dev                                ]
+ -- --=[ 2,563 exploits - 1,315 auxiliary - 1,683 payloads     ]
+ -- --=[ 431 post - 49 encoders - 13 nops - 9 evasion          ]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > search 1.9.17

Matching Modules
================

   #  Name                                          Disclosure Date  Rank    Check  Description
   -  ----                                          ---------------  ----    -----  -----------
   0  exploit/linux/local/sudo_chroot_cve_2025_32463  2025-06-30       normal  Yes    Sudo Chroot 1.9.17 Privilege Escalation


Interact with a module by name or index. For example info 0, use 0 or use exploit/linux/local/sudo_chroot_cve_2025_32463

msf > use 0
[*] No payload configured, defaulting to cmd/linux/http/aarch64/meterpreter/reverse_tcp
msf exploit(linux/local/sudo_chroot_cve_2025_32463) > show options

Module options (exploit/linux/local/sudo_chroot_cve_2025_32463):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   COMPILE   Auto             yes       Compile on target (Accepted: Auto, True, False)
   COMPILER  Auto             yes       Compiler to use on target (Accepted: Auto, gcc, clang)
   SESSION                    yes       The session to run this module on


Payload options (cmd/linux/http/aarch64/meterpreter/reverse_tcp):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   FETCH_COMMAND   CURL             yes       Command to fetch payload (Accepted: CURL, FTP, GET, TFTP, TNFTP, WGET)
   FETCH_DELETE    false            yes       Attempt to delete the binary after execution
   FETCH_FILELESS  none             yes       Attempt to run payload without touching disk by using anonymous handles, requires Linux ≥3.17
                                                (for Python variant also Python ≥3.8 (Accepted: none, bash, python3.8+)
   FETCH_SRVHOST                    no        Local IP to use for serving payload
   FETCH_SRVPORT   8080             yes       Local port to use for serving payload
   FETCH_URIPATH                    no        Local URI to use for serving payload
   LHOST           10.126.92.127    yes       The listen address (an interface may be specified)
   LPORT           4444             yes       The listen port


   When FETCH_COMMAND is one of CURL,GET,WGET:

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   FETCH_PIPE  false            yes       Host both the binary payload and the command so it can be piped directly to the shell.


   When FETCH_FILELESS is none:

   Name                Current Setting  Required  Description
   ----                ---------------  --------  -----------
   FETCH_FILENAME      fYuxPwgYnduf     no        Name to use on remote system when storing payload; cannot contain spaces or slashes
   FETCH_WRITABLE_DIR  ./               yes       Remote writable dir to store payload; cannot contain spaces


Exploit target:

   Id  Name
   --  ----
   0   Auto



View the full module info with the info, or info -d command.

msf exploit(linux/local/sudo_chroot_cve_2025_32463) >
```

Figure 5: Searching for sudo v1.9.17 exploit (CVE-2025-32463) through *msfconsole*

5 SonarCloud: Cloud-based Code Quality and Security Analysis Tool

Purpose

SonarCloud is a cloud-based version of SonarQube designed to automatically analyze source code for bugs, vulnerabilities, and code quality issues directly from online repositories such as GitHub, GitLab, Bitbucket, or Azure DevOps.

Common Uses

- Enforcing code quality and security standards in cloud-based development workflows.
- Automatically analyzing new commits and pull requests.
- Detecting vulnerabilities, code smells, and duplications before merging code.

Key Features

- Cloud Integration: No need for local setup or hosting.
- Seamless CI/CD Integration: Connects with GitHub Actions, Azure Pipelines, and more.
- Automatic Scans: Analyzes code after every commit or pull request.
- Security Hotspots: Highlights potentially vulnerable sections of code for review.
- Dashboards: Provides detailed metrics and trends across branches and projects.

Setting Up SonarCloud for a Repository

1. Go to `https://sonarcloud.io`.
2. Log in using your source control account (e.g., GitHub).
3. Follow the onboarding guide to set up an organization and project.
4. Authorize SonarCloud to access your repositories.
5. The code within the repository will be automatically analyzed based on default security and quality rules.
6. To review vulnerabilities:
   (a) Select the desired branch in the left-hand menu.
   (b) Navigate to Security Hotspots for an overview of potential issues (see Figure 6).
7. Future code pushes will trigger automatic re-analysis, ensuring continuous code quality monitoring.

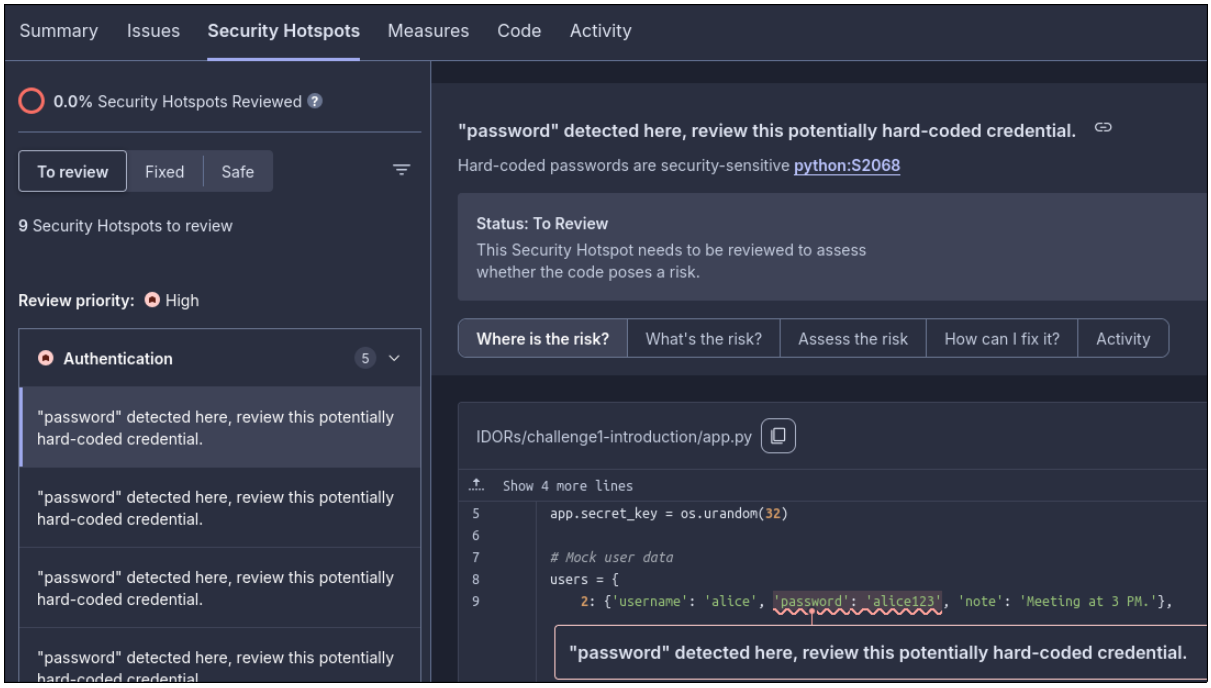

Figure 6: Example view of SonarCloud's Security Hotspots section showing hard-coded credential warnings.

Further Documentation

- **Nmap:** `https://nmap.org/book/man.html`
- **Hydra:** `https://hydra.cc/docs/intro/`
- **Wireshark:** `https://www.wireshark.org/docs/wsug_html_chunked/`
- **Metasploit:** `https://docs.metasploit.com/`
- **SonarCloud:** `https://docs.sonarcloud.io`