

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
```

```

[red] [yellow] [green] tmux
bencekanyok@Bences-MacBook-Air ~ % nmap 172.20.10.2 [74/78]
Starting Nmap 7.98 ( https://nmap.org ) at 2025-10-25 18:53 +0200
Nmap scan report for 172.20.10.2
Host is up (0.000041s latency).
Not shown: 993 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
2222/tcp  open  EtherNetIP-1
3306/tcp  open  mysql
5000/tcp  open  upnp
5432/tcp  open  postgresql
7000/tcp  open  afs3-fileserver
8080/tcp  open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.57 seconds
bencekanyok@Bences-MacBook-Air ~ % nmap -sV 172.20.10.2
Starting Nmap 7.98 ( https://nmap.org ) at 2025-10-25 18:53 +0200
Nmap scan report for 172.20.10.2
Host is up (0.000040s latency).
Not shown: 993 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 9.9 (protocol 2.0)
2222/tcp  open  ssh          OpenSSH 10.0 (protocol 2.0)
3306/tcp  open  mysql        MySQL 8.0.44
5000/tcp  open  rtsp
5432/tcp  open  postgresql   PostgreSQL DB 9.6.0 or later
7000/tcp  open  rtsp
8080/tcp  open  http         Werkzeug httpd 3.1.3 (Python 3.12.12)
[3] 0:[tmux]* "Bences-MacBook-Air.lo" 18:54 25-Oct-25
```

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.

Capture



Learn

[User's Guide](#) · [Wiki](#) · [Questions and Answers](#) · [Mailing Lists](#) · [SharkFest](#) · [Wireshark Discord](#) · [Donate](#)

You are running Wireshark 4.4.6 (v4.4.6-0-gaebb20483889). You receive automatic updates.

Figure 2: Network Interfaces to capture traffic from

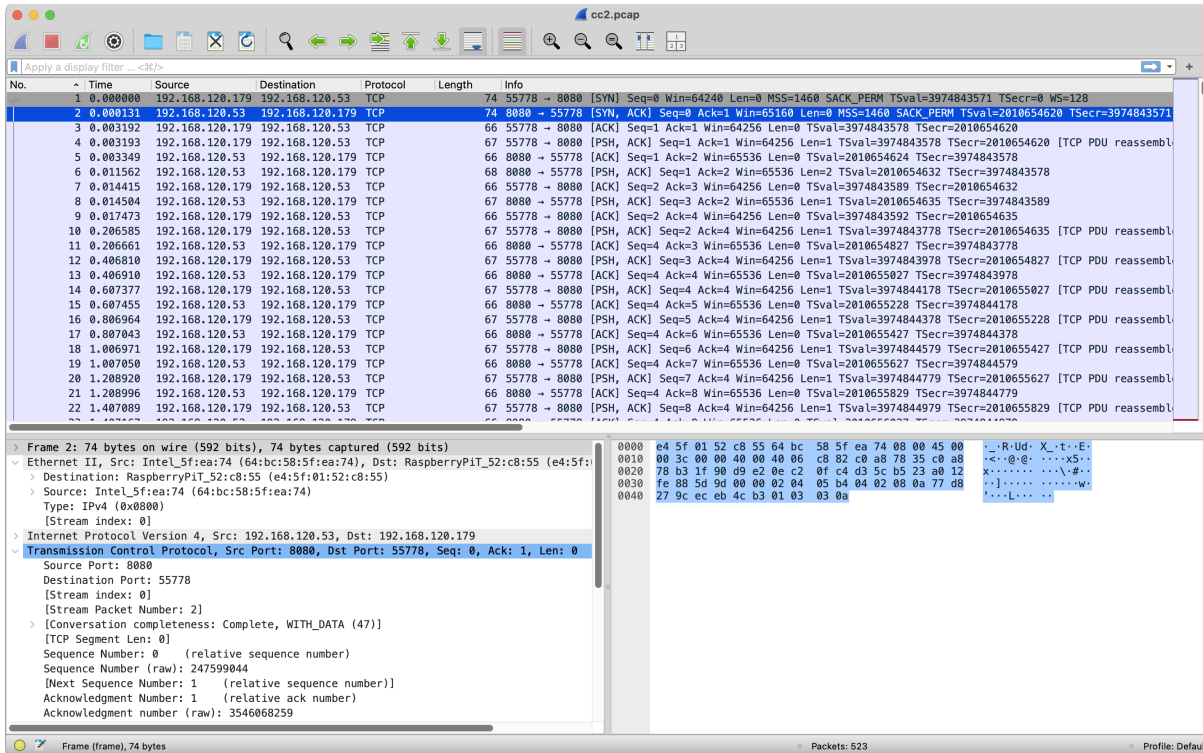


Figure 3: List of packets with details

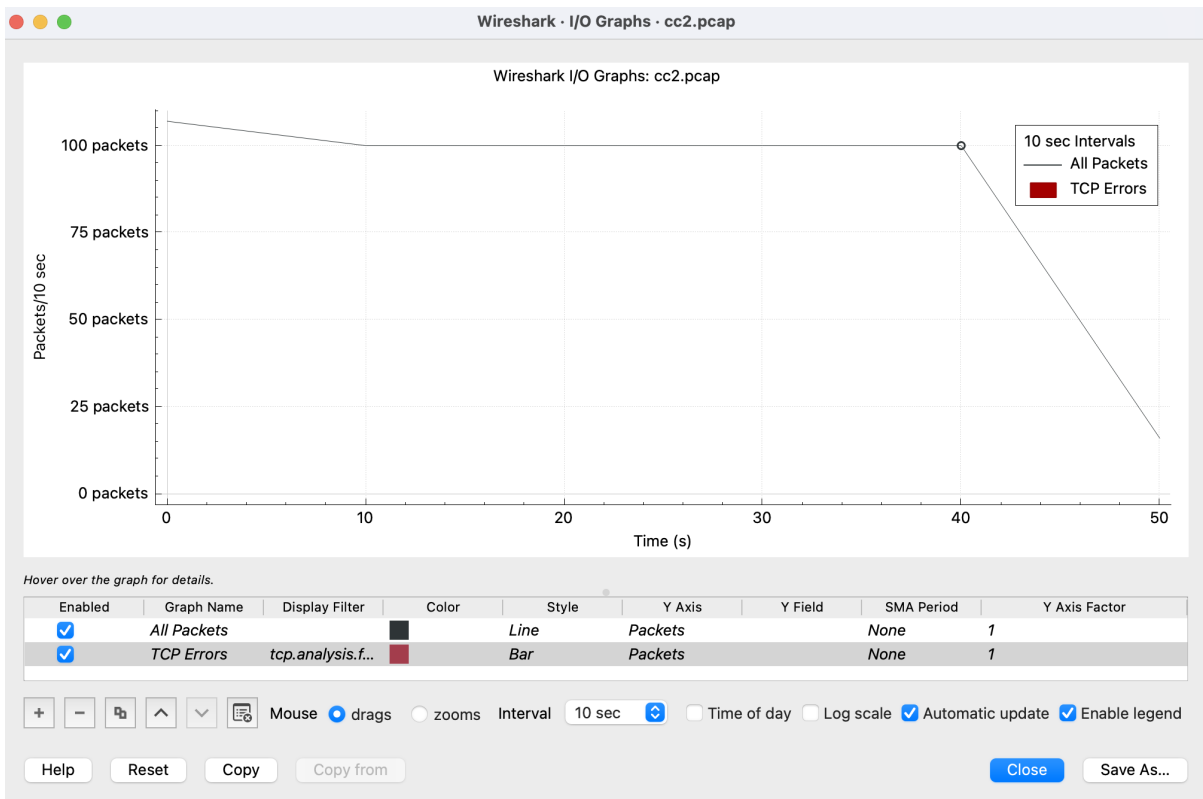


Figure 4: I/O Graph showcasing timing of packets

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
  - - - - -
  COMPILER  Auto             yes       Compile on target (Accepted: Auto, True, False)
  COMPILER  Auto             yes       Compiler to use on target (Accepted: Auto, gcc, clang)
  SESSION   yes              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               no        Local IP to use for serving payload
  FETCH_SRVPORT  8080             yes       Local port to use for serving payload
  FETCH_URI_PATH no                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

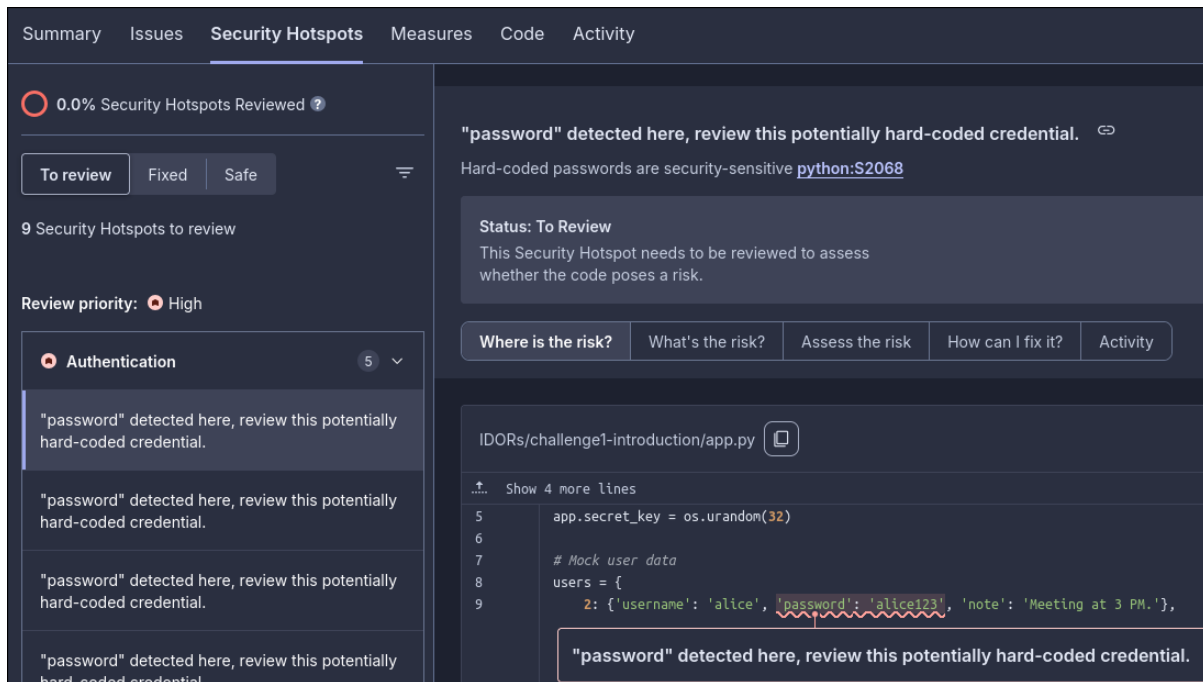
- Enforcing code quality and security standards in cloud-based development workflows.
- Automatically analyzing new commits and pull requests.
- Detecting vulnerabilities, code smells, and duplications before merging code.

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.



The screenshot displays the SonarCloud interface for Security Hotspots. At the top, there are navigation tabs: Summary, Issues, Security Hotspots (selected), Measures, Code, and Activity. A summary bar shows "0.0% Security Hotspots Reviewed" with a red circle icon. Below this, there are filters for "To review", "Fixed", and "Safe", with "To review" selected. It indicates "9 Security Hotspots to review" and a "Review priority: High" setting. A list of hotspots is shown, with the first one selected, displaying the warning: "password" detected here, review this potentially hard-coded credential. The right-hand pane provides details for the selected hotspot, including the message: "password" detected here, review this potentially hard-coded credential. It also notes that hard-coded passwords are security-sensitive and provides a link to the rule (python:S2068). The status is "To Review" with a sub-message: "This Security Hotspot needs to be reviewed to assess whether the code poses a risk." Below the status are several action buttons: "Where is the risk?", "What's the risk?", "Assess the risk?", "How can I fix it?", and "Activity". The code view shows a Python file (IDORs/challenge1-introduction/app.py) with a snippet of code where a password is hard-coded: `2: {'username': 'alice', 'password': 'alice123', 'note': 'Meeting at 3 PM.'},`. A red squiggly line underlines the password value, and a warning message is displayed below the code snippet.

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>