Week 02 - Introduction to Information Security

Information is one of the most valuable assets of individuals, organizations, and governments. Just like physical assets (money, property, or equipment), information must be protected from loss, theft, misuse, or damage. With the rapid growth of digital technologies, information is no longer stored only on paper but also on computers, networks, and the internet. This creates both opportunities and risks.

Definition of Information Security

Information Security (often called InfoSec) is the practice of protecting information and the systems that store, process, and transmit it. The purpose is to prevent unauthorized access, disclosure, alteration, or destruction of data.

Primary Objectives of Information Security

The main goals of information security are often described using the CIA Triad, which stands for Confidentiality, Integrity, and Availability. These three principles work together to ensure that data and systems remain protected and reliable.

1. Confidentiality

Meaning: Confidentiality ensures that information is only accessible to people who have the proper authorization. Unauthorized individuals should not be able to view or steal sensitive data.

How it is achieved: Through methods such as encryption, strong passwords, access controls, and user authentication.

Example: A hospital keeps patient medical records encrypted so that only doctors and authorized staff can access them. If hackers or unauthorized users try to open the files, the data would remain unreadable.

2. Integrity

Meaning: Integrity focuses on maintaining the accuracy, consistency, and completeness of data. It prevents unauthorized changes, whether intentional (like hacking) or accidental (like human error).

How it is achieved: By using error detection tools such as checksums, hashing algorithms, digital signatures, and audit logs to track changes.

Example: In online banking, when you transfer money, integrity ensures that the transaction details are not altered by a hacker during transmission. Hashing and verification methods confirm that the data remains unchanged from sender to receiver.

3. Availability

Meaning: Availability ensures that information and systems are accessible to authorized users whenever they need it. Even if technical failures or attacks occur, services should remain functional.

How it is achieved: By using backups, redundant systems, failover mechanisms, disaster recovery plans, and protection against denial-of-service (DoS) attacks.

Example: An e-commerce website must be online 24/7. If its main server fails, a backup server should automatically take over so that customers can still place orders without interruption.

Threats, Vulnerabilities, and Risks

In information security, it is important to understand how threats, vulnerabilities, and risks are connected. These three terms describe different parts of the security landscape.

Threat

Definition: A threat is any potential cause of an unwanted event that could harm a system, organization, or individual. It represents “what could go wrong.”

Types of threats:

Human-made threats – hackers, malware, phishing, insider attacks.

Natural threats – earthquakes, floods, fires, storms.

Technical threats – hardware failure, software bugs, network outages.

Example: A hacker attempting to steal confidential information is a threat. Similarly, a flood that could damage a data center is also a threat.

Vulnerability

1. Definition: A vulnerability is a weakness or flaw in a system that can be exploited by a threat. It is essentially a “security hole” that makes the system less secure.
2. Common vulnerabilities:
    1. Weak or reused passwords.
    2. Unpatched operating systems or outdated software.
    3. Misconfigured firewalls or servers.
    4. Lack of employee training on phishing attacks.
3. Example: If employees use “123456” as their password, it becomes a vulnerability that hackers can exploit.

1. Risk

1. Definition: A risk is the chance or probability that a threat will successfully exploit a vulnerability, leading to damage, loss, or disruption.
2. Factors influencing risk:
    1. The strength and frequency of the threat.

    2. The severity of the vulnerability
    3. The potential impact or damage if the attack succeeds.
3. Example: If a company has outdated antivirus software (vulnerability), and malware is actively spreading online (threat), the risk of a data breach becomes high.

Simplified Formula:
Risk = Threat × Vulnerability × Impact

1. If there is a threat but no vulnerability, the risk is low.
2. If there is a vulnerability but no threat, the risk is also low.
3. Risk becomes significant when both a threat and a vulnerability exist, especially if the impact is high.

Historical Evolution of Information Security

Information security has evolved alongside the development of technology. As computing and networking advanced, new threats appeared, requiring stronger protection methods.

- Early Era (Pre-1960s)
- Focus: Security was mainly physical.
- Information was stored on paper documents, filing cabinets, and physical media.
- Protection methods included locked rooms, safes, guards, and access restrictions.
- Example: Military or government offices restricted access to sensitive files by using safes and requiring security clearance.

- Mainframe Era (1960s–1970s)

Large centralized mainframe computers were introduced in government, research, and businesses.

Security concerns shifted from purely physical to logical access control.

Password protection became one of the first methods of securing computer systems.

Example: Universities and corporations required login credentials to access mainframe systems.

Networking Era (1980s)

1. Computers became connected through local area networks (LANs) and eventually wider networks.
2. This introduced risks such as unauthorized remote access and spreading of malicious software.
3. First generations of firewalls and antivirus software were developed to prevent external attacks.
4. Example: Worms and early viruses began spreading, showing that security had to go beyond physical locks and simple passwords.

- Internet and Cybercrime Era (1990s–2000s)

- The rapid growth of the internet opened global access but also created global threats.
- Hackers, viruses, worms, and denial-of-service (DoS) attacks became common.
- Cybercrime emerged, with attackers stealing data, spreading malware, and committing fraud.
- Organizations and governments responded by developing laws, standards, and security policies (such as ISO/IEC 27001, HIPAA for healthcare, and early data protection acts).
- Example: The “ILOVEYOU” virus (2000) spread worldwide, infecting millions of computers and causing billions of dollars in damages.

- Modern Era (2010s–Present)

Security challenges became more complex and global.

Rise of Advanced Persistent Threats (APTs), state-sponsored hacking, and organized cybercrime.

Growth of cloud computing, mobile devices, and Internet of Things (IoT) increased the attack surface.

Ransomware attacks became widespread, locking critical systems until a ransom was paid.

AI-driven attacks and automation introduced new risks but also new defense tools.

Data privacy regulations such as GDPR (Europe) and HIPAA (US healthcare) enforced stricter rules for protecting personal information.

Example: Major ransomware attacks like WannaCry (2017) affected hospitals, banks, and businesses worldwide, highlighting the importance of cybersecurity.

Scope of Information Security

Information security is a broad field that extends beyond just installing technical tools or software. It covers the people, processes, and technology within an organization, ensuring that information remains protected in all its forms.

- People
- Human behavior is often the weakest link in security. Even the most advanced systems can be compromised if users are careless.
- Employees must be properly trained to recognize threats (such as phishing emails or social engineering).
- Organizations should enforce security policies, such as strong password requirements, proper handling of sensitive data, and restrictions on unauthorized software.
- Example: An employee trained to detect phishing emails can prevent a major data breach.

Processes

Security also involves well-defined procedures and rules to ensure that information is managed safely.

This includes:

Risk management – identifying and reducing potential risks.

Incident response – how to react when a security breach happens.

Audits and compliance – checking that systems follow laws, standards, and company policies.

Example: A company with an incident response plan can quickly recover from a ransomware attack and restore services.

- Technology

1. Technical tools are essential for protecting systems and data.
2. Common security technologies include:
    1. Firewalls – block unauthorized network traffic.

    2. Encryption – protects data from being read if stolen.
    3. Authentication systems – verify the identity of users (e.g., biometrics, two-factor authentication).
    4. Intrusion detection systems (IDS) – monitor and alert on suspicious activity.
3. Example: Online banking uses encryption and two-factor authentication to secure customer transactions.

- Data in All Forms

1. Information security is not limited to digital files on computers. It protects data in three main forms:
    1. Digital data – stored in databases, hard drives, servers, or cloud platforms.
    2. Physical data – printed documents, ID cards, contracts, or even USB drives.
    3. Data in transmission – information sent through networks, the internet, wireless communication, or even phone lines.
2. Example: A printed patient record (physical), a hospital database (digital), and a telemedicine session (transmission) all require protection.

Week 03 - Importance Of Protecting Data and Systems

In today's digital environment, information is considered one of the most valuable assets. Just like money, property, or equipment, data must be protected because it drives decision-making, operations, and innovation. Losing or exposing it can cause serious financial, personal, and even national consequences.

Data as a critical resource:

Almost every sector relies on information to function.

Banking: Digital records track balances, transfers, and investments.

Healthcare: Patient histories, test results, and prescriptions are stored electronically.

Education: Student records, grades, and research are kept in digital systems.

Government: National IDs, census data, and tax information are managed digitally.

Social Media: Billions of personal posts, photos, and messages are stored online.

The risks without protection:

Sensitive data could be stolen by cybercriminals.

Information could be leaked accidentally or through insider threats.

Files could be destroyed by malware, system crashes, or natural disasters.

Protecting data ensures three key principles (CIA Triad):

Confidentiality: Data is only accessible to those with permission.

Example: Encrypting files so only authorized employees can open them.

Integrity: Data must remain accurate and unaltered, ensuring reliability.

Example: Using digital signatures or hashing to verify that a file has not been tampered with.

Availability: Data and systems should be accessible when needed, without unnecessary downtime.

Example: Backup servers and disaster recovery plans to keep services running.

Example: Online banking is a service millions rely on daily. If a bank does not secure its systems:

Hackers could steal customer login details and empty accounts.

Attackers could alter transaction data, leading to financial chaos.

Customers might be unable to access their accounts during an outage.
Protecting the system with strong security measures ensures customer trust and the stability of financial operations.

Common Types of Attacks and Motivations of Threat Actors

Cyberattacks can take many forms, depending on the methods used and the goals of the attackers. Understanding these helps organizations and individuals prepare defenses.

A. Types of Attacks

Malware (Malicious Software)

Definition: Software designed to disrupt, damage, or gain unauthorized access to systems.

Types:

Viruses – attach to files and spread when files are shared.

Worms – self-replicating programs that spread across networks.

Trojans – disguised as legitimate software but contain harmful code.

Ransomware – locks or encrypts data until a ransom is paid.

Example: The WannaCry ransomware attack (2017) infected hundreds of thousands of computers worldwide.

Phishing

Definition: Fraudulent emails, texts, or websites that trick users into revealing sensitive data (passwords, bank details).

Common technique: Fake login pages that look identical to real ones.

Example: An email pretending to be from a bank asking users to “verify” their accounts.

Denial of Service (DoS) and Distributed Denial of Service (DDoS)

Definition: Overwhelming a system, website, or server with excessive requests so it becomes unavailable to legitimate users.

DDoS uses multiple devices (often part of a botnet) to launch large-scale attacks.

Example: A DDoS attack taking down a shopping site during a major sale.

4. Man-in-the-Middle (MITM) Attack

2. Definition: An attacker secretly intercepts and possibly alters communication between two parties.
3. Often happens on insecure Wi-Fi networks.
4. Example: A hacker intercepting login details when a user logs in to a website using free public Wi-Fi.

4. SQL Injection
4. Definition: Attackers exploit weak or insecure coding in web applications to gain unauthorized access to a database.
5. Impact: Can allow attackers to steal, delete, or alter stored information.
6. Example: A poorly coded login form allows hackers to input malicious SQL commands to bypass authentication.

- Insider Threats
- Definition: Attacks or data leaks carried out by employees, contractors, or anyone with legitimate system access.
- Motivations: Revenge, financial gain, or negligence.
- Example: A disgruntled employee leaking confidential company data to competitors.

B. Motivations of Threat Actors

- Financial Gain
- Cybercriminals seek money through theft, fraud, or extortion.
- Methods: stealing credit card details, deploying ransomware, selling stolen data.
- Example: Ransomware gangs demanding cryptocurrency payments.

Espionage

Purpose: Stealing confidential or classified information for political, military, or competitive advantage.

Actors: Nation-states, corporations, or criminal organizations.

Example: Government-sponsored hackers stealing defense technology secrets.

5. Hacktivism

- Definition: Using hacking to promote political or social causes.
- Actions: Defacing websites, leaking sensitive information, or disrupting services.
- Example: Anonymous hacking government websites to protest policies.

- Revenge or Personal Motives
- Often carried out by insiders or individuals with grudges.
- Example: A fired employee deleting important files before leaving a company.

Nation-State Attacks

- Governments may use cyberattacks to target other nations for strategic, political, or economic reasons.
- Targets: Power grids, communication systems, financial institutions, military databases.
- Example: Alleged state-sponsored cyberattacks on election systems to influence outcomes.

Step-By-Step Guide to Install Oracle VM VirtualBox:

How to Install VirtualBox on Windows

- **Download Installer**
  - Go to: https://www.virtualbox.org/wiki/Downloads
  - Click **Windows hosts** to download the installer (.exe file).
- **Run Installer**
  - Double-click the downloaded file.
  - If prompted by User Account Control, click **Yes**.
  - Follow the setup wizard. Default options are usually fine.
  - Note: The installer may reset your network connection briefly.
- **Finish Installation**
  - Click **Install** → then **Finish**.
  - VirtualBox should now be available in the Start menu.
- **(Optional) Install Extension Pack**
  - On the same downloads page, download the **Extension Pack** (must match your VirtualBox version).
  - Double-click the .vbox-extpack file.
  - VirtualBox will open and prompt to install it. Accept the license.
- **Verify Installation**
  - Open **VirtualBox** from Start menu.
  - Go to **Help** → **About VirtualBox** to confirm the version.

Tips / Common Issues

Make sure virtualization (Intel VT-x / AMD-V) is enabled in BIOS/UEFI.

If VirtualBox won't start a VM, check if Hyper-V is enabled — it may need to be disabled.

Keep VirtualBox updated for best compatibility with guest OSes.

Installing Kali Linux (pre-built VirtualBox VM) on Windows

KALI + VIRTUALBOX (Windows) — STEP-BY-STEP

Prerequisites

- Windows PC with enough disk space (VM ~10–30 GB depending on image) and >=4 GB RAM (8+ GB recommended).
- Oracle VM VirtualBox installed on Windows. Download from the official VirtualBox page if you don't have it. (VirtualBox)
- 7-Zip installed (to extract the Kali .7z archive). (7-Zip official: https://www.7-zip.org). (7-zip.org)
- Ensure CPU virtualization (Intel VT-x / AMD-V) is enabled in BIOS/UEFI and that Hyper-V is disabled if it conflicts with VirtualBox. See Microsoft / VirtualBox documentation if you hit VT-x errors. (Microsoft Learn)

Step 1 — Download the pre-built Kali VirtualBox image - https://www.kali.org/get-kali/#kali-virtual-machines

4. Open Kali's Get Kali page → scroll to **Pre-built Virtual Machines** and click the **VirtualBox** image link (the file is hosted on cdimage.kali.org). This page also lists the SHA256 sums for the VM files. (Kali Linux)
5. Download the **VirtualBox .7z** file (example name: kali-linux-<version>-virtualbox-amd64.7z) into a folder on your PC. Also download the SHA256SUMS and SHA256SUMS.gpg files from the same directory if available (recommended). (Kali Linux)

Step 2 — Verify download (recommended)

- SIMPLE (quick) SHA256 check in PowerShell:
  - Open PowerShell in the download folder and run:
    Get-FileHash -Algorithm SHA256 .\kali-linux--virtualbox-amd64.7z
  - Compare the printed hash to the SHA256 shown on Kali's download page or the SHA256SUMS file. (Kali Linux)
- STRONG (recommended for security): verify SHA256SUMS with Kali's GPG key (requires gpg on Windows — e.g., install Gpg4win or use WSL). Kali's docs show the exact commands and how to import Kali's signing key and verify the signature. If the signature is valid you've got a genuine image. (Kali Linux)

Step 3 — Extract the downloaded .7z file

3. Use 7-Zip: right-click the .7z → 7-Zip → "Extract Here" (or run 7z x kali-linux-<version>-virtualbox-amd64.7z if you have 7z on the PATH). This produces a

folder containing a .vbox file (and VDI/VM files). Kali docs show the same extract step. (Kali Linux)

Step 4 — Import / Add the VM into VirtualBox

Launch **Oracle VM VirtualBox**.

In the VirtualBox window choose **File** → **Add…** (or the Add icon).

Browse to the folder you extracted and select the .vbox file (the VM definition).
Click **Open** / **Add**. The VM entry will appear in VirtualBox. (Kali docs demonstrate adding the .vbox file). (Kali Linux)

(Alternate: if you have an .ova you can use **File** → **Import Appliance…** and follow the wizard.)

Before starting, check and adjust VM settings if needed (Memory, CPUs, Network mode). Recommended: 2+ CPUs, 4096+ MB RAM for comfortable use (adjust to host capacity).

Step 5 — First boot, login, and immediate hardening

- Start the VM.
- Default login for pre-built VM: **user** = kali and **password** = kali. Change this immediately after first login. (Kali documents state pre-built VMs use kali/kali by default.) (Kali Linux)
- Change the password:
  - Open a terminal and run:
    passwd
  - Follow prompts to set a secure password.

Step 6 — Update the system and install guest tools

- In the VM terminal, update and upgrade packages:
  sudo apt update
  sudo apt full-upgrade -y
- Ensure VirtualBox Guest Additions / guest packages are installed (pre-built images often include guest additions; if not, reinstall virtualbox-guest-x11):
  sudo apt install -y --reinstall virtualbox-guest-x11
  sudo reboot
  Kali docs show the guest additions guidance and the virtualbox-guest-x11 package. (Kali Linux)

Step 7 — Optional (recommended) safety & convenience steps

- Take a VirtualBox snapshot before you do major experiments (VirtualBox → select VM → Snapshots → Take). This lets you revert quickly.
- If you need network isolation for training, change the VM's network adapter to **Host-only** or **NAT** depending on your goals.
- If you'll use USB devices in the VM, install the VirtualBox Extension Pack on the host (matching VirtualBox version) and ensure your Windows user is allowed to access USB devices. (VirtualBox)

Troubleshooting tips (common issues)

VT-x / VM wont start / 64-bit guest missing — enable virtualization in BIOS/UEFI and disable Hyper-V / Windows Hypervisor Platform if it blocks VT-x. Microsoft docs show how to disable Hyper-V if needed. (Microsoft Learn)

Extraction error — make sure 7-Zip is up to date and you have enough disk space; re-download the file if the checksum failed. (7-zip.org)

Guest Additions problems — reinstall virtualbox-guest-x11 inside the guest and reboot. (Kali Linux)

Quick checklist (short)

Install VirtualBox on Windows. (VirtualBox)

Download Kali VirtualBox .7z from Kali (cdimage.kali.org). (Kali Linux)

Verify SHA256 (and GPG if possible). (Kali Linux)

Extract .7z with 7-Zip. (7-zip.org)

File → Add → choose .vbox. (Kali Linux)

Boot, login kali/kali, change password, sudo apt update && sudo apt full-upgrade. (Kali Linux)

Sources / further reading (official)

- Kali — Get Kali / Pre-built VMs (VirtualBox link & SHA256 sums). (Kali Linux)
- Kali — Import Pre-Made Kali VirtualBox VM (exact import & extract steps). (Kali Linux)
- Kali — Download images securely (GPG + SHA256 verification instructions). (Kali Linux)

- VirtualBox — official downloads and Extension Pack info. (VirtualBox)
- 7-Zip — official extractor (Windows). (7-zip.org)
- Kali — VirtualBox Guest Additions instructions. (Kali Linux)

What is Metasploitable?

Metasploitable is a purposely **vulnerable virtual machine** designed for learning and practicing penetration testing, vulnerability scanning, and exploit development in a safe, legal environment. It contains many intentionally insecure services, applications, and configurations so students and security professionals can learn to find and exploit real-world issues without harming production systems.

Main things to know

1. **Purpose:** training, labs, testing security tools (Nmap, Metasploit, vulnerability scanners), and practicing exploit techniques.
2. **Not for production:** it's deliberately insecure — don't put it on open or production networks.
3. **Common contents (examples):** outdated services like FTP, SSH, Samba, MySQL, Tomcat, vulnerable web apps, misconfigured daemons and default credentials. These provide lots of hands-on exercises.
4. **Typical versions:**
    1. **Metasploitable 2:** the most widely used — a prebuilt Linux VM (easy to import or attach as a VMDK). Default credentials often used in labs: msfadmin / msfadmin.
    2. **Metasploitable 3:** a newer, more realistic collection (Windows and Linux builds) that is usually **built** from source with tools like Packer and Vagrant rather than provided as a single ready OVA.
5. **Where it's used:** cybersecurity training, Capture The Flag (CTF) practice, tool testing, classroom labs, and demos.

Example beginner commands (safe to run in your lab)

- Discover hosts/services:
  nmap -sV -Pn 192.168.56.101
- Quick Metasploit workflow (from an attacker VM like Kali):
    - msfconsole
    - search vsftpd (find an exploit)
    - use exploit/unix/ftp/vsftpd_234_backdoor
    - set RHOST 192.168.56.101
    - run

(Those are lab examples — always target only machines you own or have permission to test.)

Safety & legal reminders

2. **ONLY** run Metasploitable in isolated lab environments (Host-Only or NAT + Host-Only) you control.
3. Never deploy it on public networks or systems you don’t own — doing so is dangerous and illegal.

Lesson: Websites & Web Security
Learning objectives
By the end of this lesson learners will be able to:
Explain how websites and web applications work (client, server, HTTP/S, APIs).
Describe common web vulnerabilities and why they happen.
Understand the legal and ethical boundaries of security testing.
Use safe, legal methods and labs to practice finding and fixing vulnerabilities.
Apply basic secure-coding and defensive practices to reduce risk.

1 — How websites work (high-level)

Clients & servers: A web browser (client) requests resources from a web server. The server sends HTML/CSS/JS and may provide backend services (databases, APIs).
HTTP & HTTPS: HTTP is the application protocol; HTTPS is HTTP over TLS and provides encryption and authentication.
Frontend vs Backend: Frontend = what users see and run in the browser. Backend = server-side logic, databases, authentication, business rules.
State & sessions: HTTP is stateless. Sessions (cookies, tokens) maintain user state. Improper session handling often causes security issues.
Common tech stack: Web server (Nginx/Apache), application (Node/Python/Ruby/Java/.NET), database (MySQL/Postgres/Mongo), reverse proxies, CDNs, and APIs.

2 — Categories of web vulnerabilities (conceptual)

Important: This section explains *what* vulnerabilities are and *how to defend*. It does **not** provide exploit recipes.
Injection (e.g., SQL, NoSQL, command injection): Unsanitized input becomes part of commands/queries. *Defense:* parameterized queries/ORMs, input validation, least privilege for DB accounts.
Cross-Site Scripting (XSS): Malicious scripts injected into pages seen by other users. *Defense:* output encoding, Content Security Policy (CSP), input validation.
Cross-Site Request Forgery (CSRF): Attacker tricks a browser into making authenticated requests. *Defense:* CSRF tokens, SameSite cookies, require re-authentication for sensitive actions.
Broken Authentication & Session Management: Weak passwords, session fixation, or exposed tokens. *Defense:* secure session cookies (HttpOnly, Secure, SameSite), multi-factor authentication, strict password policies, token expiration.
Broken Access Control: Users access or perform actions they shouldn't. *Defense:* enforce server-side authorization checks, use role-based access control (RBAC), deny-by-default.
Sensitive Data Exposure: Storing/transmitting secrets insecurely. *Defense:* TLS for transit, encryption at rest, proper key management, avoid sensitive data in logs.
Security Misconfiguration: Default credentials, verbose error messages, unnecessary services exposed. *Defense:* hardening guides, removing unused features, secure defaults.
Insecure Direct Object References (IDOR): Predictable identifiers allow access to others' resources. *Defense:* unpredictable identifiers, server-side authorization checks.

Using Components with Known Vulnerabilities: Outdated libraries and frameworks. *Defense:* dependency scanning, timely updates, monitoring advisories.
Insufficient Logging & Monitoring: Attacks go unnoticed. *Defense:* centralized logging, alerting, retention policies, incident response plan.

3 — Legal & ethical rules (non-negotiable)

Only test systems you own or are explicitly authorized to test in writing. Unauthorized testing is illegal and unethical.
Get scope, time windows, safe-word, and data-handling rules in writing before performing any active testing.
Prefer non-destructive testing methods when in doubt. Maintain confidentiality of any discovered data.
Report responsibly: follow coordinated disclosure policies or the target's bug-bounty program rules.

4 — Methodology: Web security testing (high-level process)

Reconnaissance (passive first): gather public info (DNS, subdomains, certificate transparency, public endpoints).
Mapping & discovery: enumerate endpoints, files, parameters, APIs, and inputs.
Analysis & identification: identify potential weaknesses in authentication, input handling, access control, configuration.
Validation (non-destructive): verify vulnerabilities in a safe way (proof-of-concept that does not expose or exfiltrate real data).
Reporting & remediation: document findings with reproducible steps, risk level, and recommended fixes.
Retest after fixes.

Lesson: Information Gathering with Knockpy on Kali Linux

Learning objectives:

Understand what Knockpy does and when to use it.
Install and configure Knockpy on Kali Linux.
Run common Knockpy workflows to enumerate subdomains and analyze results.
Combine Knockpy output with other tools (e.g., curl, dig, nmap) for further reconnaissance.
Understand legal and ethical constraints for reconnaissance activities.

1 — What is Knockpy?

Knockpy is a Python-based subdomain enumeration tool that uses wordlists and DNS queries to discover subdomains for a given domain. It supports DNS zone guessing (brute-force-like with wordlists), wildcard detection, and simple result exporting. Use it as part of the reconnaissance phase to map an organization's external attack surface.
When to use it: early in active reconnaissance to find subdomains that may host services, web apps, or misconfigurations. It's a supplement to other discovery tools (Amass, Sublist3r, assetfinder), not a replacement.

2 — Legal & ethical note (read this first)

Only run Knockpy against domains you own or have explicit permission to test.
Passive/active recon may trigger monitoring or defensive responses — ask for authorization in writing (scope + targets).
Do not scan or exploit systems outside the agreed scope.

3 — Install Knockpy on Kali Linux

Open a terminal on Kali.
Update package list (optional but recommended):

```
sudo apt update && sudo apt upgrade -y
```

Install prerequisites (Python3 and pip are usually installed on Kali):

```
sudo apt install -y python3 python3-pip git
```

Clone the Knockpy repository and install:

```
cd /opt
sudo git clone https://github.com/guelfoweb/knock.git knockpy
cd knockpy
sudo pip3 install -r requirements.txt
# Make the main script executable (if needed)
sudo chmod +x knockpy.py
```

Optional: add a symlink so you can run knockpy from anywhere:

```
sudo ln -s /opt/knockpy/knockpy.py /usr/local/bin/knockpy
# Now you can run: knockpy example.com
```

(If the repo structure differs, adjust paths — the above commands are the common pattern. If a packaged version exists in apt or pip, you may use that instead.)

4 — Basic Knockpy usage

Syntax (typical):

```
knockpy <domain> [options]
```

Common options:
-w PATH : specify a custom wordlist (one subdomain name per line).
-o FILE : output results to a file.
-t N : threads (speed).
--all : try common prefixes and suffixes (tool-dependent).
-e : enable wildcard detection handling (if supported).
Example — quick scan with default wordlist:

```
knockpy example.com
```

Example — use custom wordlist and save output:

```
knockpy example.com -w /usr/share/wordlists/subdomains-top1million-5000.txt -o
knock_results.txt -t 20
```

Interpreting results: Knockpy will list discovered subdomains and resolved IPs. If many results share the same DNS response, check for wildcard DNS (see next section).

5 — Handling wildcard DNS and false positives

Wildcard DNS means many non-existent subdomains resolve to the same IP. To detect this:
Use dig or nslookup to query a deliberately random subdomain:

```
dig asdf1234random.example.com +short
```

If it returns an IP (same IP your tool reported), this suggests wildcard DNS is active — filter or treat these results cautiously.
Knockpy may include wildcard detection flags or heuristics; if not, manually filter results by checking which hostnames resolve uniquely.

6 — Post-processing and enrichment (combine tools)

After you have a list of discovered subdomains, enrich and verify:

5. **Resolve & verify live hosts:**

```
# using massdns / dnsx / host / dig
cat knock_results.txt | awk '{print $1}' > subdomains.txt
for host in $(cat subdomains.txt); do echo $host && dig +short $host; done
```

5. **Passive reconnaissance:** use whois, crt.sh (certificate transparency logs), or online services to find related hosts.
6. **Port scan live hosts** (permission required):

```
nmap -Pn -sV -p 1-1000 -iL subdomains.txt -oA nmap_subs
```

5. **HTTP probing** to see web apps:

```
# curl / httpx / whatweb
cat subdomains.txt | httpx -silent -ports 80,443
```

7. **Screenshot and manual inspection** (e.g., with gowitness or aquatone) for visual reconnaissance.

7 — Example lab walkthrough (step-by-step)

Target (lab-only): lab.example.local (replace with a permitted domain)

- Run Knockpy with a medium-sized list:

```
knockpy lab.example.local -w /usr/share/wordlists/subdomains-top1million-5000.txt -o
lab_knock.txt -t 25
```

- Check for wildcard:

```
dig r4nd0m-l3tters.lab.example.local +short
```

- Extract subdomains and resolve:

```
awk '{print $1}' lab_knock.txt > lab_subs.txt
for h in $(cat lab_subs.txt); do echo -n "$h -> " ; dig +short $h | tr '\n' ' ' ; echo ; done
```

- Scan ports on resolved IPs:

```
nmap -sV -iL lab_subs.txt -oA lab_nmap
```

Probe HTTP endpoints:

```
httpx -l lab_subs.txt -o lab_httpx.txt
```

Document findings: compile subdomain, IP, open ports, HTTP status, and notes about wildcard behavior.

8 — Wordlists & tuning

6. Start with curated subdomain lists (smaller to faster — then scale up). Common locations on Kali: /usr/share/wordlists/ or use SecLists (git clone https://github.com/danielmiessler/SecLists.git).
7. Tune threads (-t) according to your network and DNS provider limits — too many threads risks rate-limiting or detection.
8. Use exponential/backoff or delays when testing public targets to avoid being blocked.

1 — What is DIRB?
DIRB is a simple, command-line web content scanner that brute-forces directories and filenames on web servers using wordlists. It requests many candidate paths (from a supplied wordlist) and reports responses that indicate existing resources (200, 301/302, 403, etc.). It's intended for discovery/footprinting during web reconnaissance and is included or easily installable on Kali Linux.

When to use: early reconnaissance to find hidden panels, backup files, admin pages, or misconfigured content on a web host — *only* for systems you own or are authorized to test.

2 — Legal & ethical rules
Only scan domains/hosts you own or have explicit written authorization to test.
Prefer **isolated lab targets** (Juice Shop, DVWA, local VMs) for practice.
Avoid high-volume noisy scans against production targets unless in scope and scheduled.
Log and document everything; use nondestructive checks for validation.

3 — Install DIRB (Kali)
DIRB is usually preinstalled. If not:
```
sudo apt update
sudo apt install dirb
Wordlists bundled with DIRB commonly live in
/usr/share/dirb/ or /usr/share/wordlists/dirb/. SecLists
(/usr/share/seclists/) is a good additional source.
```

4 — Basic usage & examples
Syntax:
dirb <url> [wordlist] [options]
Examples
```
# Quick default scan (uses DIRB's default wordlist)
dirb http://example.com

# Use a specific wordlist and save output
dirb http://example.com /usr/share/wordlists/dirb/common.txt -o dirb_results.txt

# HTTPS (use -S to force SSL)
dirb https://example.com /path/to/wordlist.txt -S

# Use authentication (basic auth)
dirb http://example.com /usr/share/wordlists/dirb/common.txt -a 'user:pass'
```

Typical output lines show discovered paths and codes, e.g.:
```
+ http://example.com/admin (CODE:200|SIZE:1543)
+ http://example.com/backup.zip (CODE:200|SIZE:204800)
+ http://example.com/login (CODE:301|REDIRECT:/signin)
```

Maltego — Graph-Based OSINT & Recon (Lesson)
1 — What is Maltego?
Maltego is a visual link-analysis and OSINT tool that collects, correlates, and visualizes relationships between entities (people, domains, IPs, email addresses, infrastructure, social accounts, certificates, etc.). It uses *transforms* — modular data-gathering functions — to query sources (public APIs, DNS, certificate transparency, search engines, social platforms) and build interactive graphs that reveal associations and attack surface relationships. Maltego is widely used for reconnaissance, threat intel, and investigative research.

2 — Editions & licensing (short)
Community Edition (CE): free, limited transforms and rate limits — great for learning.
Classic / XL / Enterprise: paid editions with more transforms, higher throughput, collaboration features, and commercial integrations (Shodan, VirusTotal, ZoomInfo, etc.).
Always pick the edition that fits your scope and use-case; for class/lab use CE or an approved license.

3 — Legal & ethical rules
Only gather OSINT and run transforms on targets you have permission to investigate if the transforms access non-public data or could be intrusive.
Respect terms of service and API usage limits.
Document your sources, timestamps, and any credentials used for API-based transforms.
OSINT can reveal sensitive info — handle responsibly and follow disclosure or reporting rules where applicable.

4 — Installing Maltego (short how-to)
Kali Linux: Maltego is often available in Kali repositories as a package (maltego), or you can download installers from the vendor (Paterva / Maltego website).
Windows/macOS: download the installer from Maltego's site and follow the GUI installer.
First run: create or sign in with a Maltego account. For API-based transforms (Shodan, VirusTotal, Censys, etc.) add API keys in *Manage Accounts / Transform Settings*.
Community Edition note: CE limits transforms per run and available data sources — you'll see prompts to configure or purchase integrations.

5 — Maltego fundamentals
Entities: nodes on the graph (Domain, DNS name, IP, Person, Email address, Website, Certificate, Social Network Account, etc.).
Transforms: actions you run from an entity (e.g., Domain → DNS records, Domain → subdomains, Email → associated names). Transforms return new entities and link them to the graph.
Seed entity: the starting point (e.g., example.com).
Graph canvas: interactive area where entities and links are visualized and arranged.

Local transforms & integrations: some transforms run locally (regex parsing), others query external services (Shodan, Censys, Bing, Whois).
Machines: Maltego workflows that run multiple transforms in sequence automatically (useful for repeatable recon patterns).

6 — Typical Maltego workflow (high-level)
Open a new graph and add a seed entity (e.g., a domain name or email).
Run passive transforms first (Whois, DNS, Certificate transparency, web page links, search engine results). These are lower-impact and rely on public data.
Enrich discovered entities (resolve domains to IPs, find ASN info, geo-locate, query Shodan/Censys for services if authorized).
Pivot from one entity to another (e.g., Domain → subdomains → IPs → open ports → associated organizations).
Cluster & analyze the graph visually to identify central nodes, infrastructure overlaps, or potential points of interest.
Export & document findings (CSV, GraphML, images) and save the graph for repeatability.

7 — Example transforms to use (common)
Domain → DNS Names / MX / NS / SOA (find authoritative servers & mail).
Domain → Subdomains (CT logs, passive sources).
Domain → SSL Certificates (certificate transparency / crt.sh entries).
Domain/IP → Shodan / Censys (service discovery — API key required).
Email → Owner / Breach data (where supported — may require API).
Website → Links / Contacts (crawl the site for linked entities).
Person → Social Media Profiles (OSINT-based matching).
Always check which transforms are passive vs active; many Maltego transforms are passive (safe) but some API queries count against quotas.

Kali Linux is a Debian-based Linux distribution built for information-security tasks: penetration testing, security research, forensics, and reverse engineering. It bundles hundreds of security tools and is maintained by the Kali project (offical site and docs). Always download Kali from the official site and verify checksums/signatures. (Kali Linux)

Metasploitable refers to intentionally vulnerable virtual machines created for learning and testing security tools and techniques. Metasploitable2 (older Ubuntu-based image) and Metasploitable3 (built from source via GitHub) are provided as training targets so students and defenders can practice detection and remediation in an isolated lab. These images are intentionally insecure and must be run only in controlled environments.

Why pair them? Kali ships many discovery and testing tools (Nmap, Wireshark, Metasploit, Burp, etc.). Metasploitable provides safe, reproducible vulnerable targets—together they form a contained lab for learning how attacks look and how to defend against them. (Metasploit Framework docs explain the framework used for many lab demos.)

Kali Linux (what it enables)

Host discovery and enumeration (Nmap, netcat).

Traffic capture and analysis (tcpdump, Wireshark).

Web application inspection (Burp Suite, proxies).

Vulnerability scanning and controlled proof-of-concept testing (Metasploit Framework, auxiliary scanners).

Learning platform for defensive tasks: log analysis, IDS tuning, patch verification. (Kali Linux)

Metasploitable (what it enables)

Practice recognizing common misconfigurations and vulnerable services (old FTP, web apps, DB services).

Safe target for tool testing (validate scanners, demonstrate signatures in IDS, practice incident response).

Teaching remediation: patching, hardening, network segmentation.

Learning outcomes from using them together

Students learn how to discover exposed services, interpret results, prioritize remediation, and document findings—without ever touching external, production systems.

This is a high-level, practical checklist you can follow in a classroom or on a personal lab machine. I avoid any exploit/actionable attack steps—only setup, verification, and safe testing guidance.

1) Prepare the host machine

Ensure virtualization support (Intel VT-x / AMD-V enabled in BIOS/UEFI).

Have enough resources: recommended minimum 8–16 GB RAM and 40–100 GB free disk to comfortably run two VMs (Kali + Metasploitable).

Use a recent hypervisor: VirtualBox or VMware Workstation / Player are common choices and supported by Metasploitable documentation.

2) Download official images (important safety step)

Kali Linux: get the official ISO/VM images from Kali's downloads page and verify the SHA256 checksums and signatures. Prefer the *Installer* image for full installs or prebuilt VM images if you want quicker setup. (Kali Linux)

Metasploitable: obtain Metasploitable2 images from Rapid7 or trusted mirrors (and Metasploitable3 from the Rapid7 GitHub if you plan to build it). Note: Metasploitable3 may require building with Packer/Vagrant for certain providers.

3) Create a safe, isolated virtual network

Configure your VMs to use **Host-Only** or **Internal Network** (VirtualBox) / **Host-only network** (VMware). This ensures the Kali ↔ Metasploitable VMs can communicate while being isolated from the Internet and your organization's network

Optionally, put the hypervisor host behind a separate physical network or firewall if more isolation is desired. (The point is to prevent accidental scanning or exposure to third parties.)

4) Install or import the VMs

Kali options:

Import prebuilt VM: download official VM images (OVA) if available and import into VirtualBox/VMware. OR

Install from ISO: attach the Kali ISO to a new VM and perform a standard install following Kali's installation guide (choose disk sizing, locale, user setup). Use the Installer image recommended by Kali docs. (Kali Linux)

Metasploitable options:

Import prebuilt image: for Metasploitable2 you can import the provided VM image/OVA into your hypervisor.

Build Metasploitable3: follow the GitHub instructions if you want Metasploitable3 (it uses Packer/Vagrant and can be more complex to build).

5) Configure VM resources & snapshots

Give Kali and Metasploitable modest CPU and RAM (e.g., 1–2 vCPU, 2–4 GB RAM each for simple labs; scale up if your host allows).

Take snapshots or save a clean state immediately after setup so you can quickly revert after demonstrations. Snapshots are essential for restoring an intentionally vulnerable VM to a known baseline.

6) Network verification & basic checks (safe)

Start both VMs and confirm they are on the same isolated network. Use `ip a` / `ifconfig` (or check the hypervisor UI) to find each VM's IP address.

From Kali, **ping** the Metasploitable IP to verify connectivity. (This is benign connectivity testing.)

From Kali, check that you can reach services on the Metasploitable target *only* within the isolated network—verify the VM cannot reach the Internet (if isolation requires that).

Document the IPs and network config in your lab log.

7) Tool readiness (safe, non-destructive)

Update Kali packages (if permitted by your lab rules) or use the preinstalled tools. Note: do not update Metasploitable (it's intentionally vulnerable—updating defeats the learning objective unless you are practicing patching).

Confirm presence of common tools (Nmap, Wireshark, Metasploit) by launching them and checking versions. Use them only for discovery or benign demonstrations in the lab.

8) Lab safety & housekeeping

Keep an access control list for the lab.

Always use snapshots and reset the target VM after demos that change its state.

Keep copies of the official download checksums and document from where you downloaded images. (Kali Linux)

Weevely is a lightweight PHP “web-shell” framework designed to provide a remote command shell (and many post-exploitation modules) through an HTTP endpoint. It’s distributed as an agent (a small PHP script dropped on a web host) and a client that talks to that agent. Because it’s explicitly designed to allow remote administration via HTTP, attackers have used it as a backdoor — but security educators also use it in isolated labs to teach detection and incident response.

Burp Proxy is the interception proxy component of Burp Suite (PortSwigger). It sits between your browser and the target web application so you can intercept, inspect, and (where appropriate in a lab) modify HTTP(S) requests and responses. Burp Proxy is a core tool for web application testing and for understanding HTTP workflows. PortSwigger documents it as a safe, user-driven tool for learning how web apps behave.

What they *do* (capabilities & legitimate uses)
Weevely
Provides a remote PHP-based shell and many modules for file operations, database queries, command execution, and simple privilege-escalation checks — basically a small, extendable backdoor. Useful in labs to demonstrate how web shells function and the impact of a successful file upload/backdoor. (GitHub)
Legitimate classroom uses: demonstrate post-infection indicators, show how a web shell persists, practice forensic recovery and cleanup, test detection rules (YARA, IDS signatures), and practice patching/hardening.
Burp Proxy
Intercepts and displays HTTP(S) traffic; lets you forward, drop, or modify requests; integrates with other Burp tools (Repeater, Intruder, Scanner). Ideal for understanding request/response cycles, headers, cookies, and how web apps handle input. (PortSwigger)
Legitimate classroom uses: teach web request anatomy, demonstrate how injection inputs propagate, validate secure cookie flags, test WAF rules in a controlled lab, and capture evidence for incident response exercises.

Safe lab exercises (non-actionable)
Use an isolated environment (host-only / internal network) and snapshots. Example safe exercises:
Weevely (defensive focus): deploy a benign, instructor-controlled PHP file that only logs requests (DO NOT teach students how to upload shells on live systems). Students then analyze webserver logs and file system artifacts to identify the presence/behavior of the “agent” and write detection rules (YARA, simple regex).
Burp Proxy (hands-on): configure a lab browser to proxy through Burp and use it to inspect normal traffic, view headers, and practice identifying insecure settings (absent

HSTS, missing secure cookie flags). Use Burp's Intercept & Repeater to replay benign requests to observe server responses. (All actions confined to lab targets.)

Detection signals & forensic indicators (what defenders look for)
For web shells like Weevely and similar backdoors, common indicators include:
Unusual files in web directories (odd filenames, recently modified files, small PHP files with obfuscation).
Odd request patterns: shortly encoded or encrypted POST bodies, repeated small POSTs to a single endpoint, or requests with strange User-Agent/headers.
Web server logs showing requests that trigger shell behavior (commands encoded in parameters) or consistent POSTs to non-standard endpoints.
Network patterns: long-lived HTTP sessions that behave like interactive shells, or requests with suspicious timing patterns.
Filesystem / process artifacts: newly created files, unexpected cron jobs, or changed file permissions.
Enterprise guidance and research (DoD/CSIRT style) emphasize that web shells are frequently obfuscated and hard to detect — combine host and network signals and use multiple detection layers.

Ethical & legal notes (must-read)
Weevely **is dual-use**: it was built for post-exploitation research and administration in controlled contexts but can be abused. Never deploy web shells against systems you do not own or have explicit written permission to test. Teaching must emphasize legal/ethical boundaries and controlled environments (isolated VMs, instructor oversight).