23CY601 CYBERSECURITY NR(R23)

Mr. K. SRINIVASA RAO,
Assistant Professor,
Computer Science and Engineering
(CYBER SECURITY)

UNIT-I

CYBER SECURITY FUNDAMENTALS

Network and Security Concepts

- Information Assurance Fundamentals,
- Basic Cryptography,
- Symmetric Encryption,
- Public Key Encryption,
- The Domain Name System (DNS),
- Firewalls,
- Virtualization,
- Radio- Frequency Identification

Microsoft Windows Security Principles:

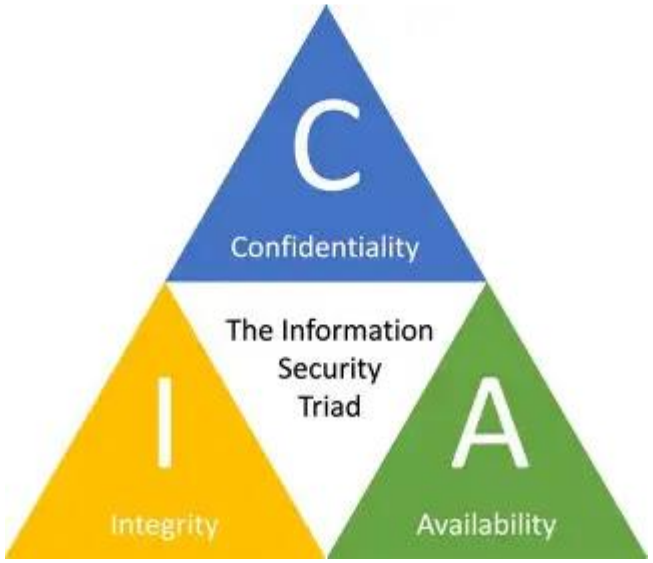
- Windows Tokens,
- Window Messaging,
- Windows Program,
- The Windows firewalls.

Information Assurance Fundamentals

- Confidentiality
- Integrity
- Availability
- Authentication
- Non-repudiation
- Security models

Network & Security Concepts

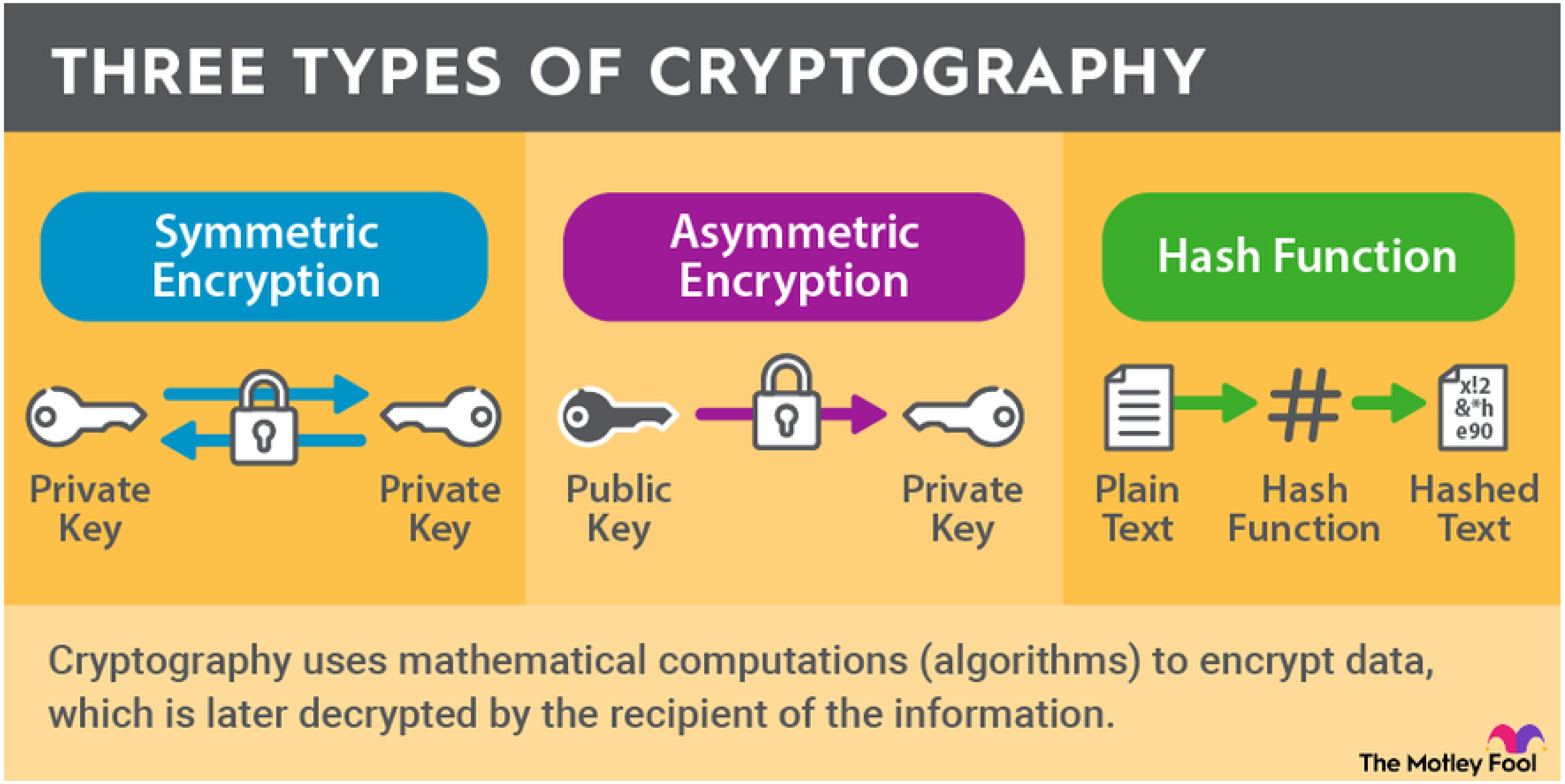
- CIA Triad
- Threats & Vulnerabilities
- Attacks (MITM, DoS, Spoofing)
- Security Controls



Basic Cryptography

- Plaintext, Ciphertext
- Encryption Algorithms
- Hashing Functions
- Cryptographic Keys

Basic Cryptography

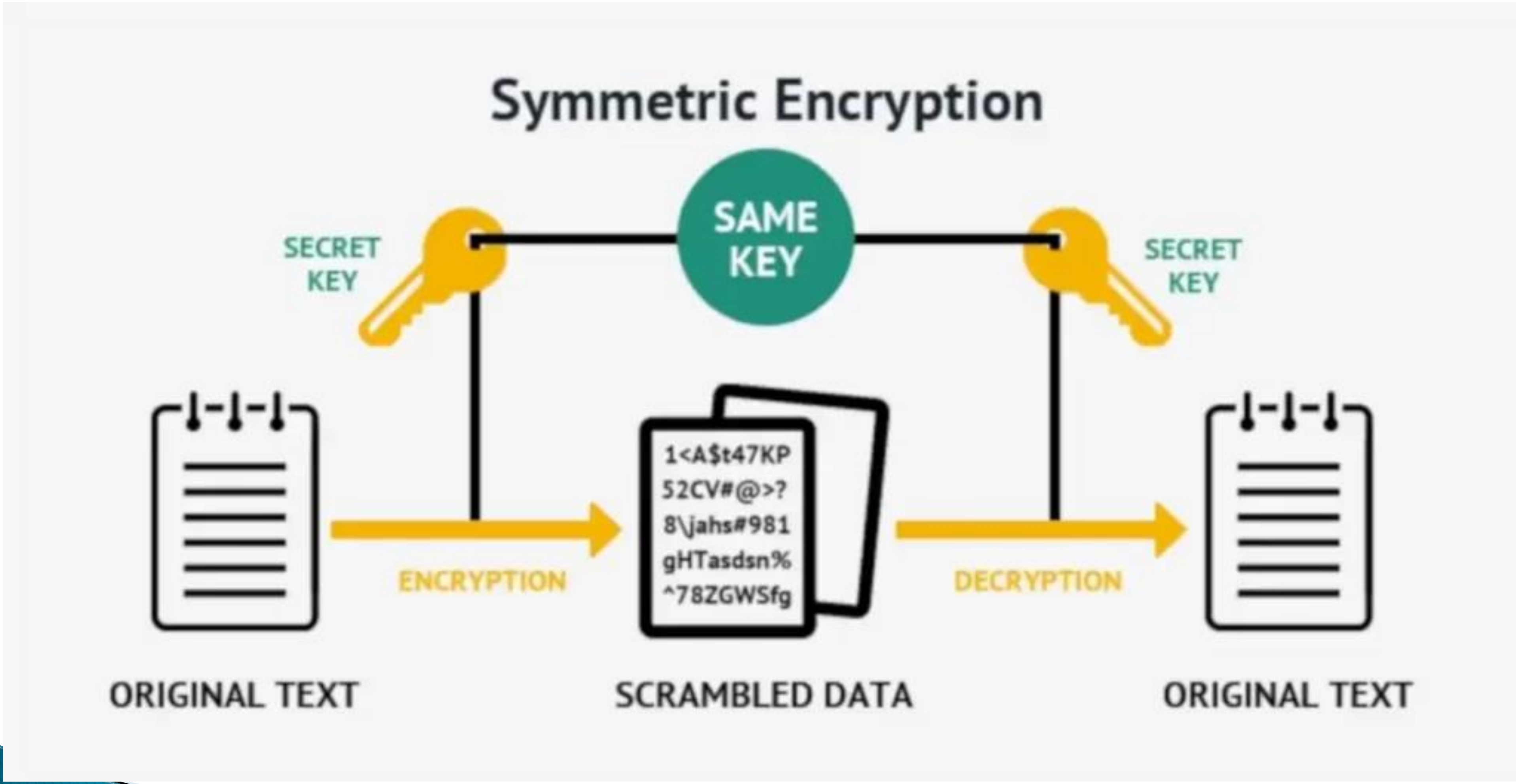


Cryptography uses mathematical computations (algorithms) to encrypt data, which is later decrypted by the recipient of the information.

Symmetric Encryption

- Single-key encryption
- AES, DES, 3DES
- Stream vs Block Ciphers
- Strengths & Weaknesses

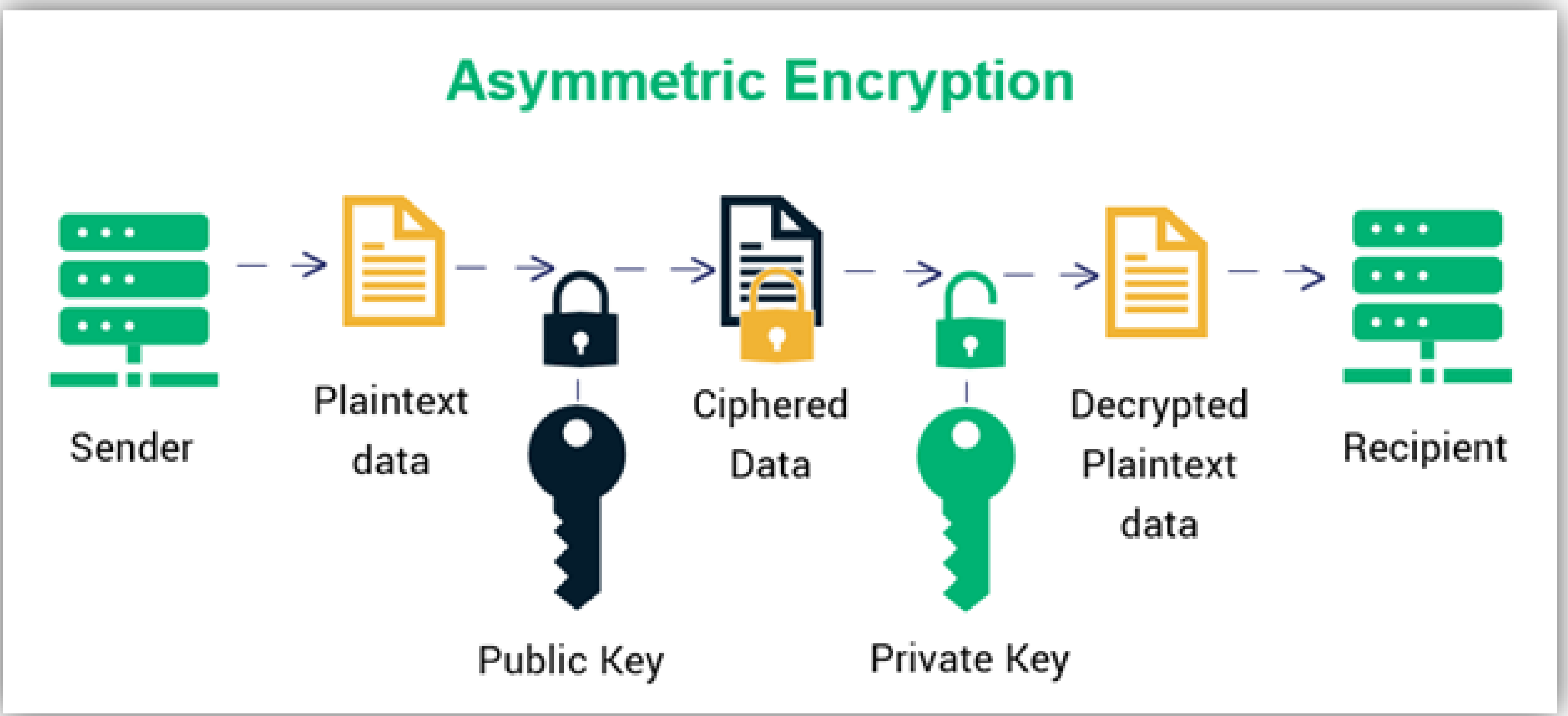
Symmetric Encryption



Public Key Encryption (Asymmetric)

- Public & Private Keys
- RSA, ECC
- Digital Certificates
- Digital Signatures

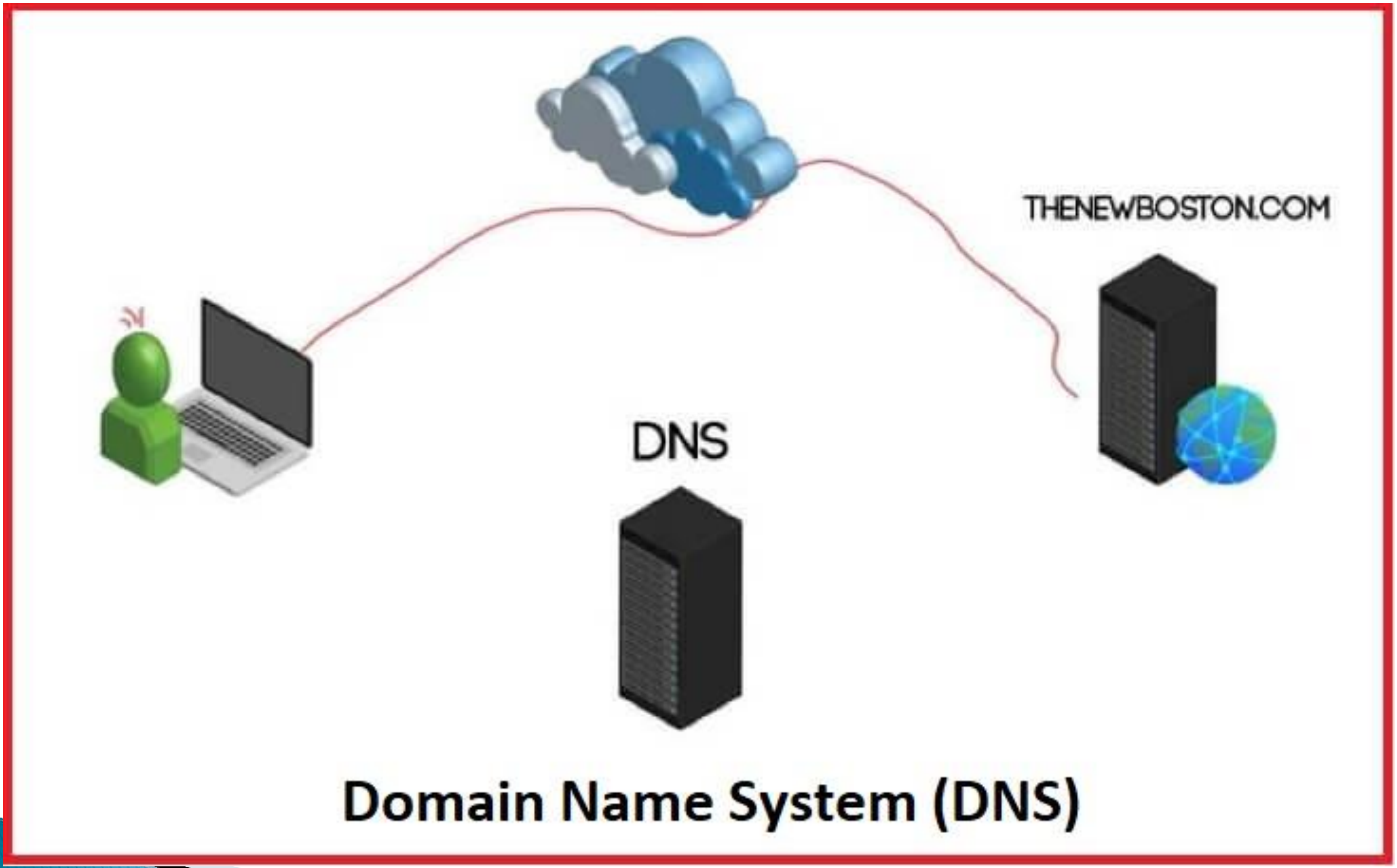
Public Key Encryption (Asymmetric)



Domain Name System (DNS)

- DNS Working
- Record Types (A, AAAA, CNAME, MX)
- DNS Resolver
- DNS Security Issues

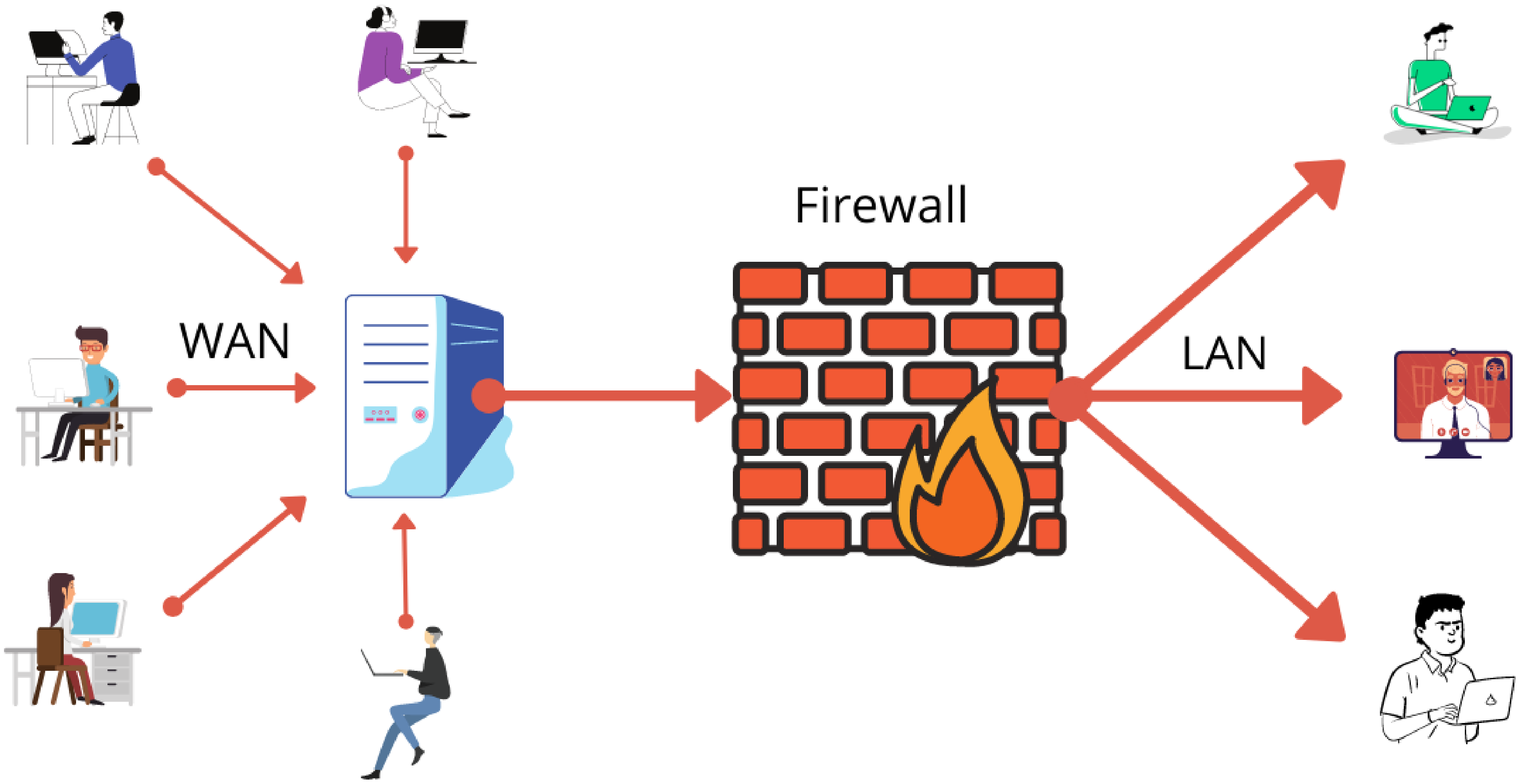
Domain Name System (DNS)



Firewalls

- Packet Filtering
- Stateful Firewall
- Application Layer Firewall
- Firewall Policies

Firewalls



Virtualization

- Hypervisors (Type 1 & Type 2)
- Virtual Machines
- Virtualization Security
- VM Escape

Virtualization

Virtualization is a technology that enables the creation of virtual environments from a single physical machine, allowing for more efficient use of resources by distributing them across computing environments.

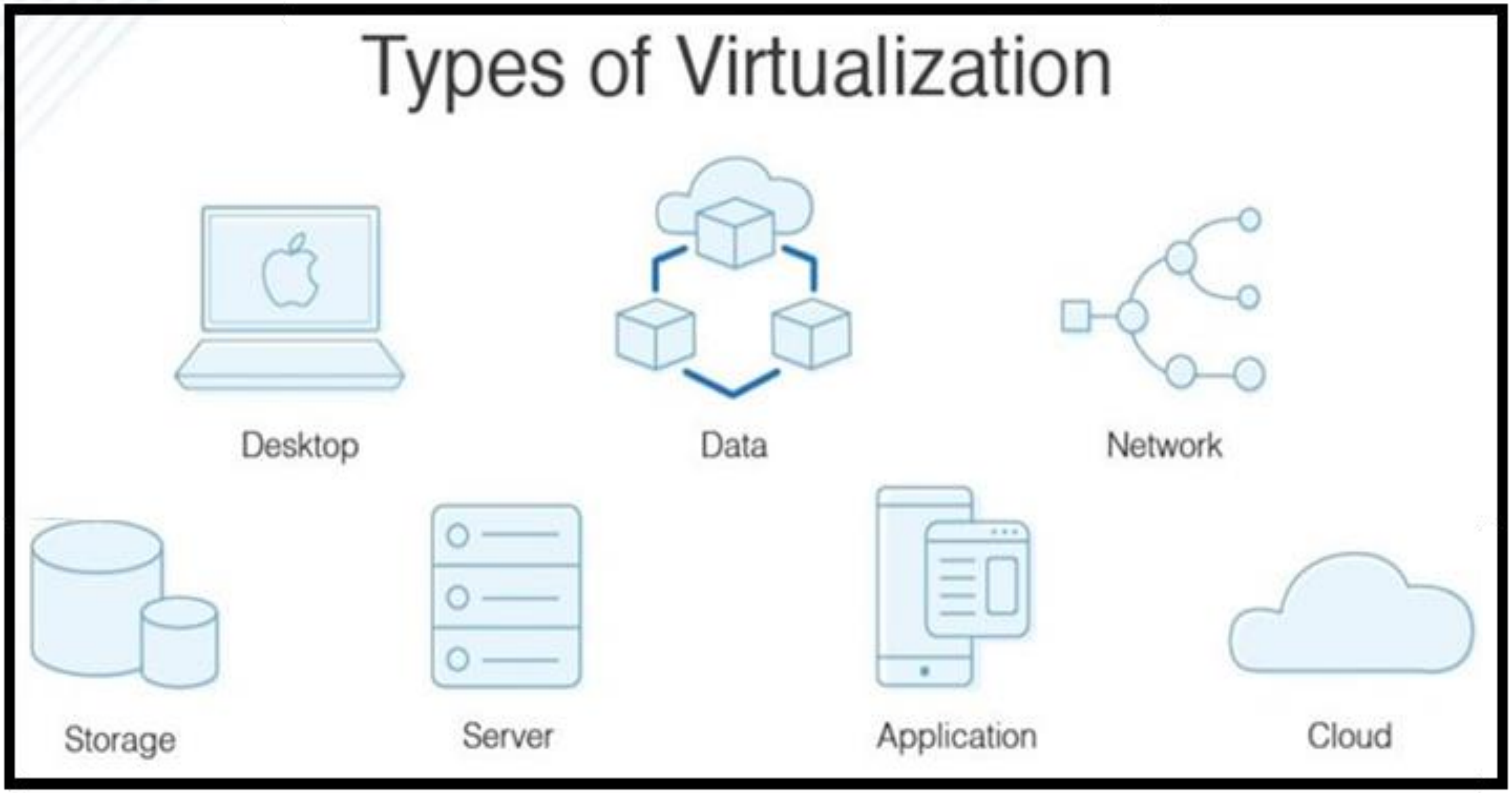
Using software, virtualization creates an abstraction layer over computer hardware, dividing a single system's components such as processors, memory, networks and storage into multiple virtual machines (VMs). Each VM runs its own operating system (OS) and behaves like a separate physical computer, despite sharing the same underlying hardware

Virtualization

Today, virtualization is a fundamental practice in enterprise IT architecture and a key enabler of cloud computing.

It allows **cloud service providers (CSPs)** such as IBM Cloud®, **Microsoft Azure (MSA)**, **Google Cloud (GC)** and **Amazon Web Services (AWS)**, to optimally utilize their

Virtualization

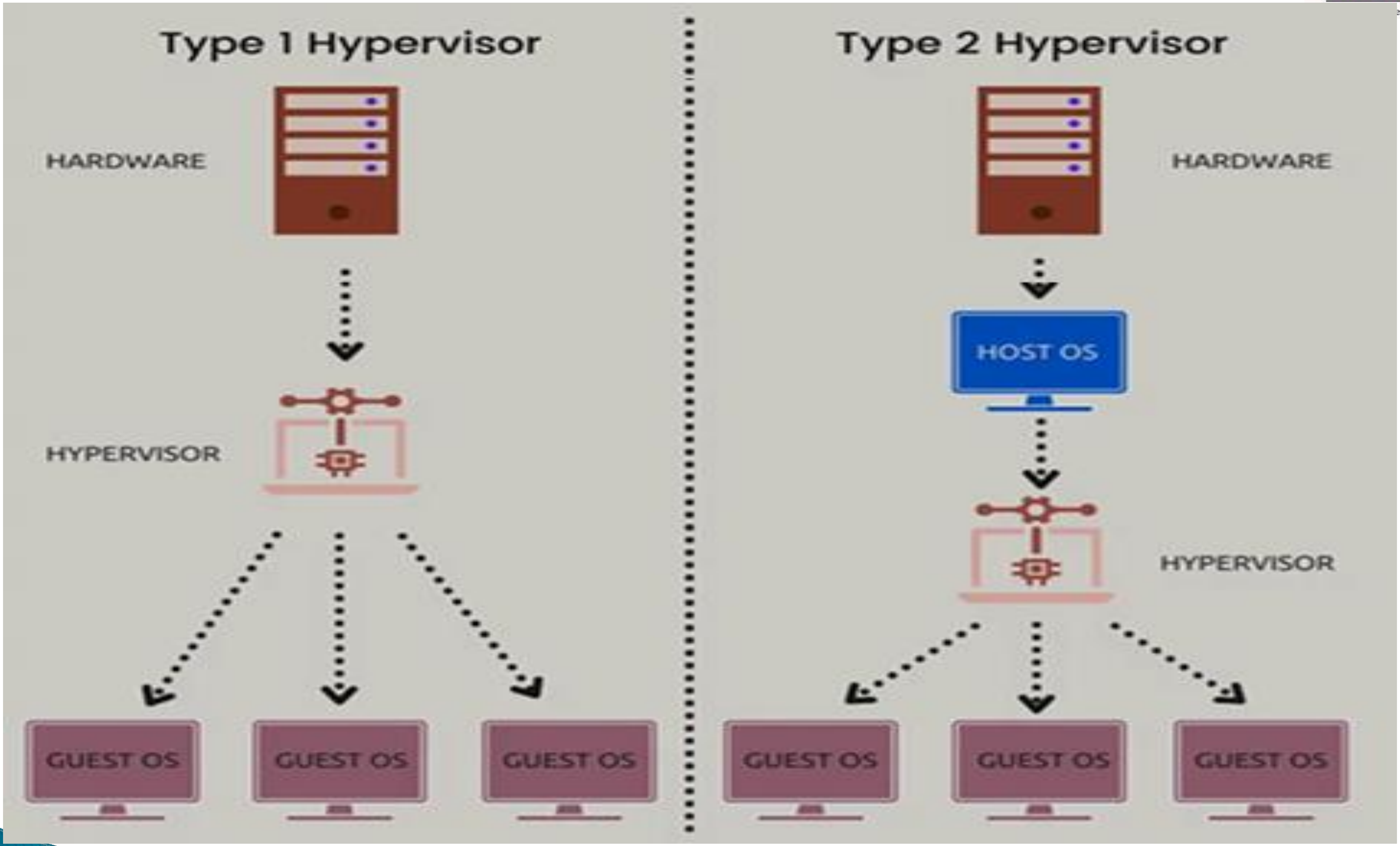


Hypervisors (Type 1 & Type 2)

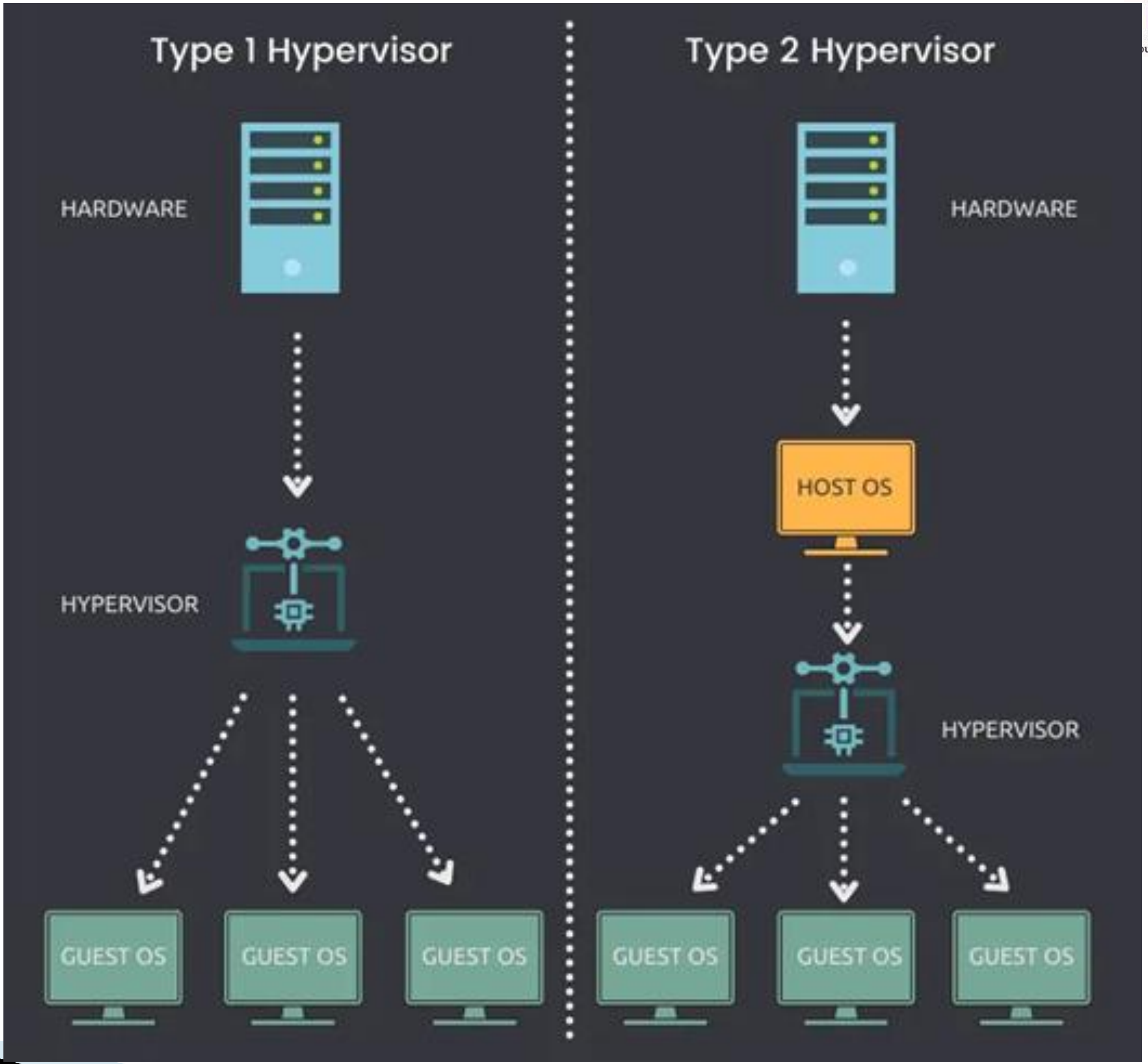
Type 1 and **type 2** **Hypervisors** are software you use to run one or more **virtual machines (VMs)** on a **single physical machine**. A virtual machine is a digital replica of a physical machine. It's an **isolated computing environment** that your users experience as completely independent of the underlying hardware. The **hypervisor** is the technology that makes this possible. It manages and allocates physical resources to VMs and communicates with the underlying hardware in the background.

The **type 1 hypervisor sits** on top of the bare metal server and has direct access to the **hardware resources**. Because of this, the **type 1 hypervisor** is also known as a **bare metal hypervisor**. In contrast, the **type 2 hypervisor** is an application installed on the **host operating system**. It's also known as a ***hosted or embedded hypervisor***.

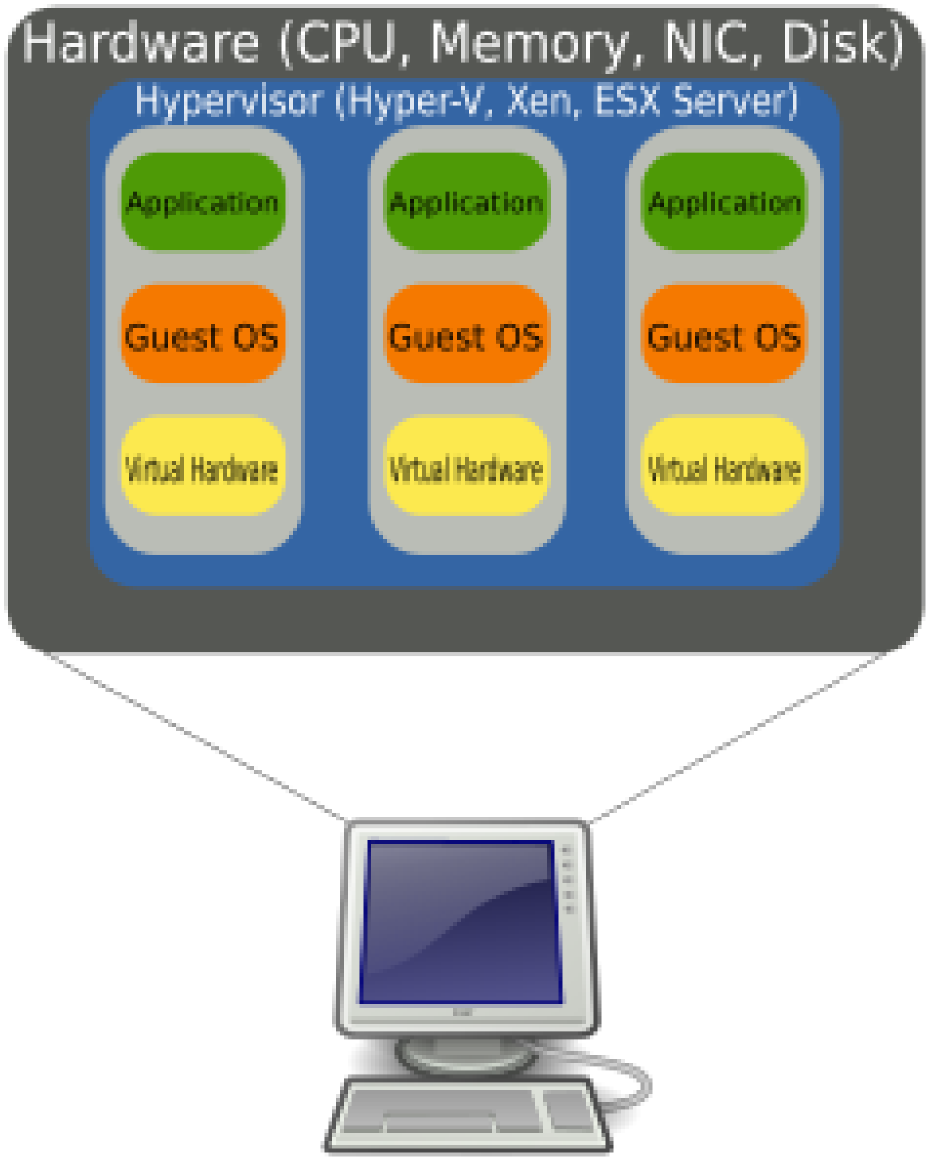
Virtualization



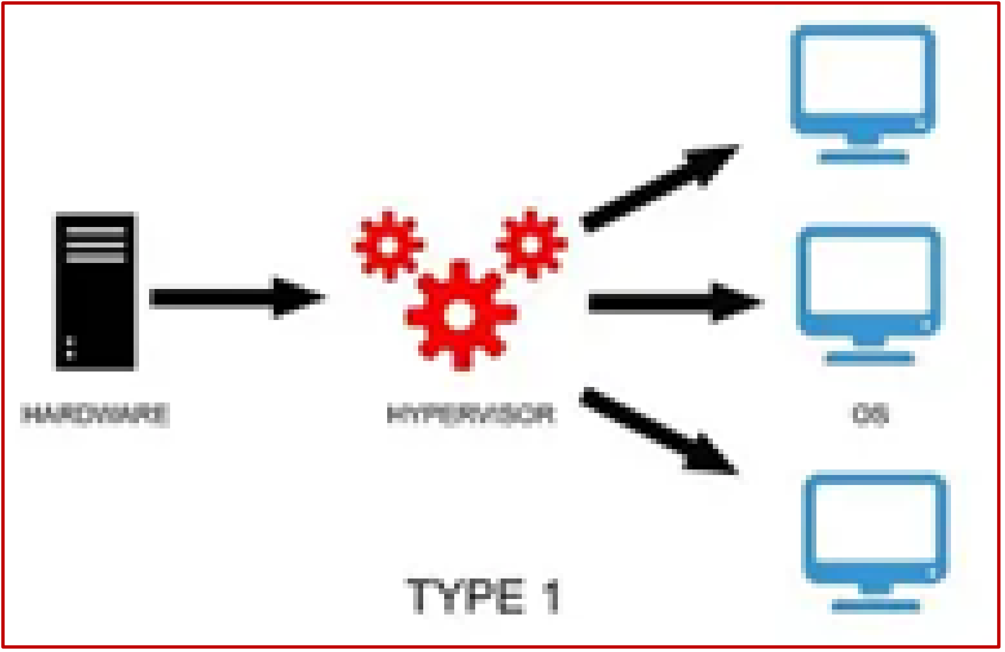
Virtualization

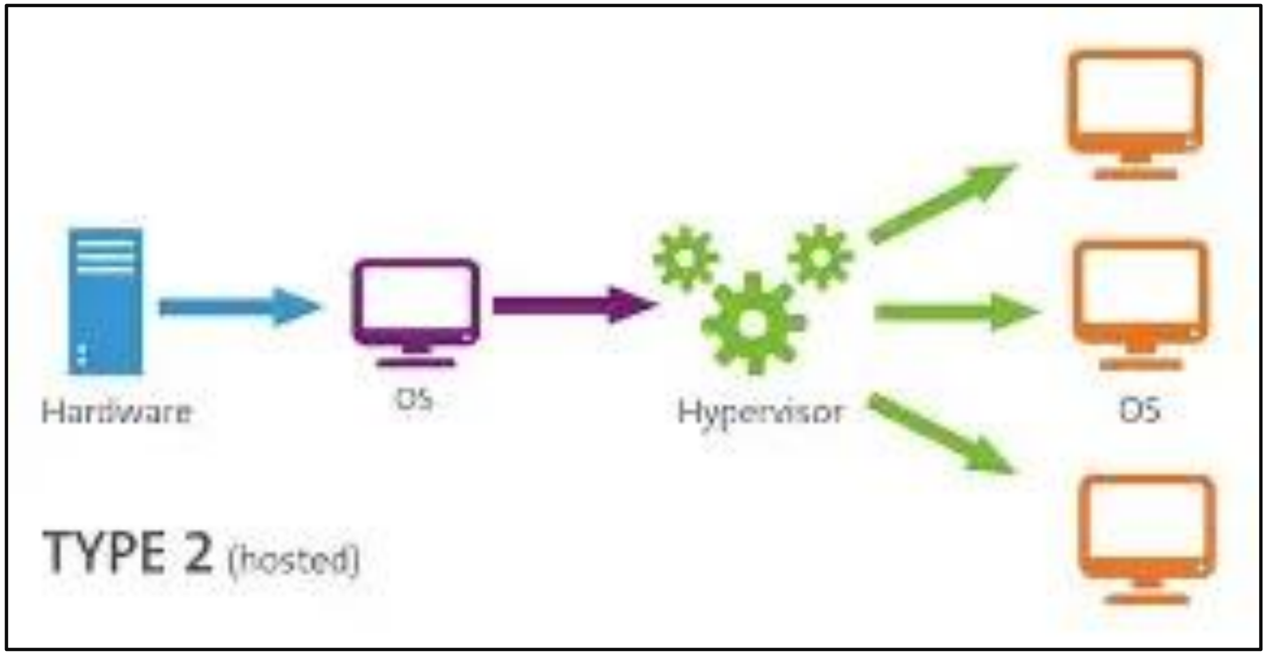


Virtualization



Logical diagram of full virtualization


HARDWARE
HYPERVISOR
OS
TYPE 1


Hardware
OS
Hypervisor
OS
TYPE 2 (hosted)

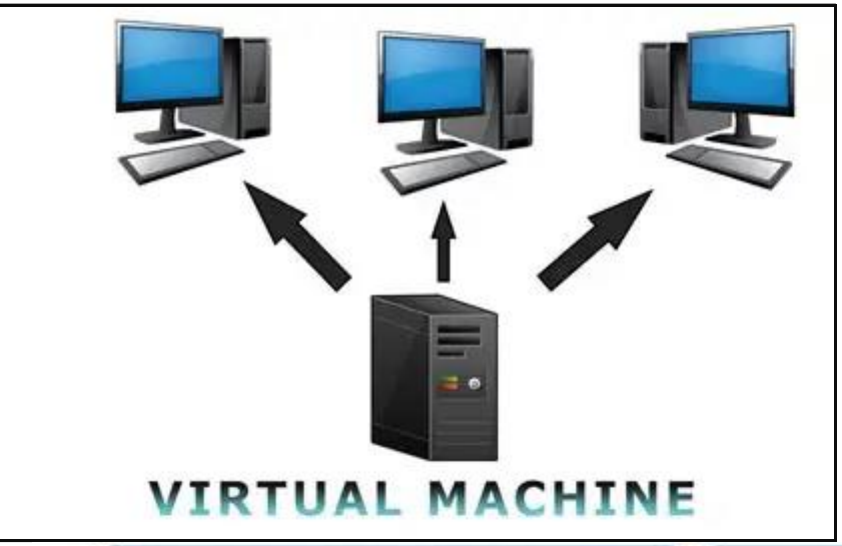
Virtual Machines

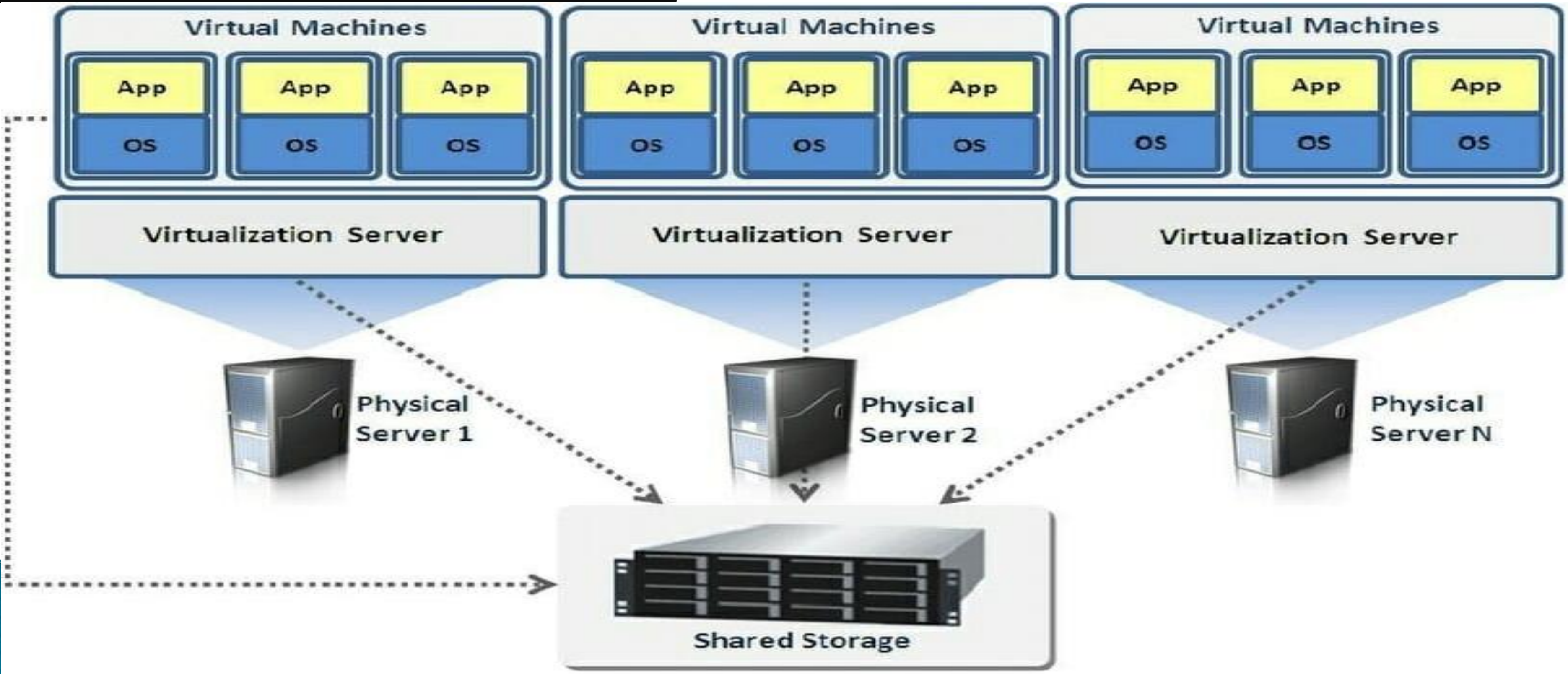
In computing, a **virtual machine** (**VM**) is the virtualization or emulation of a computer system. Virtual machines are based on computer architectures and provide the functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination of the two. Virtual machines differ and are organized by their function, shown here:

System virtual machines (also called full virtualization VMs, or SysVMs provide a substitute for a real machine.

Process virtual machines are designed to execute computer programs in a platform-independent environment.

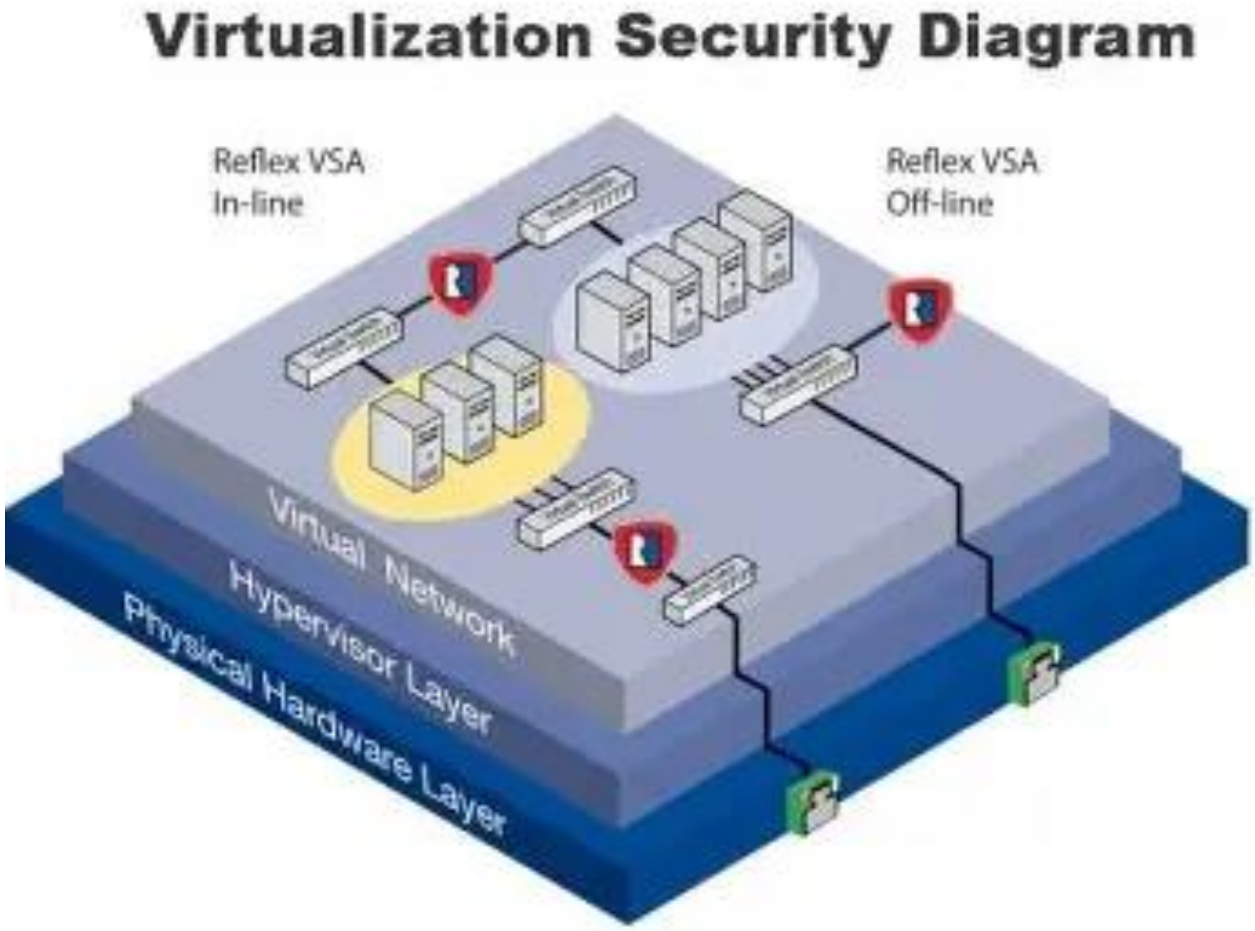
Virtual Machines





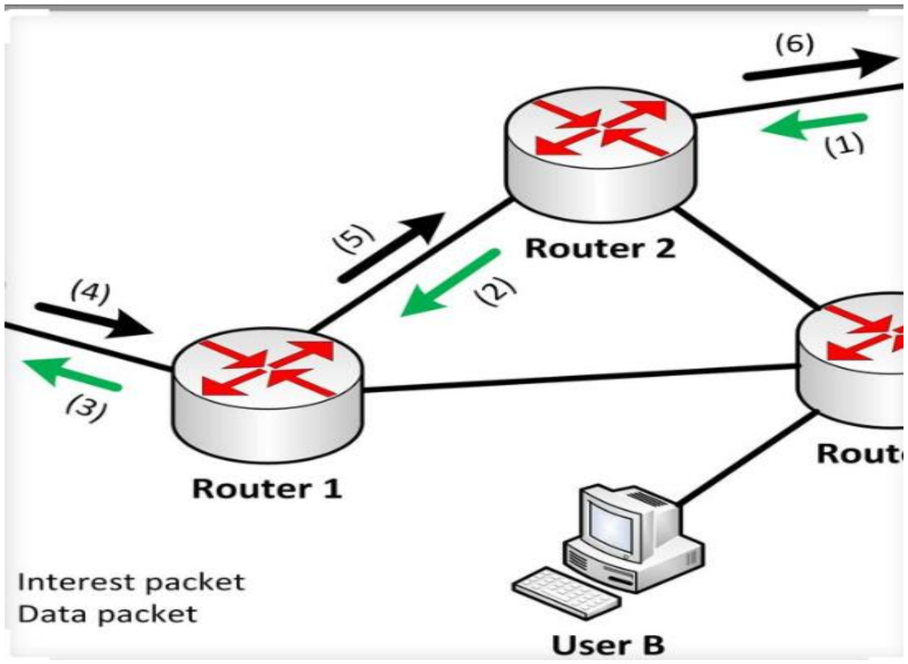
Virtualization Security

Virtualization security refers to the measures and practices employed to protect virtualized IT environments from securit y threats and vulnerabilities. It encompasses various strategi es, including:



• VM Escape

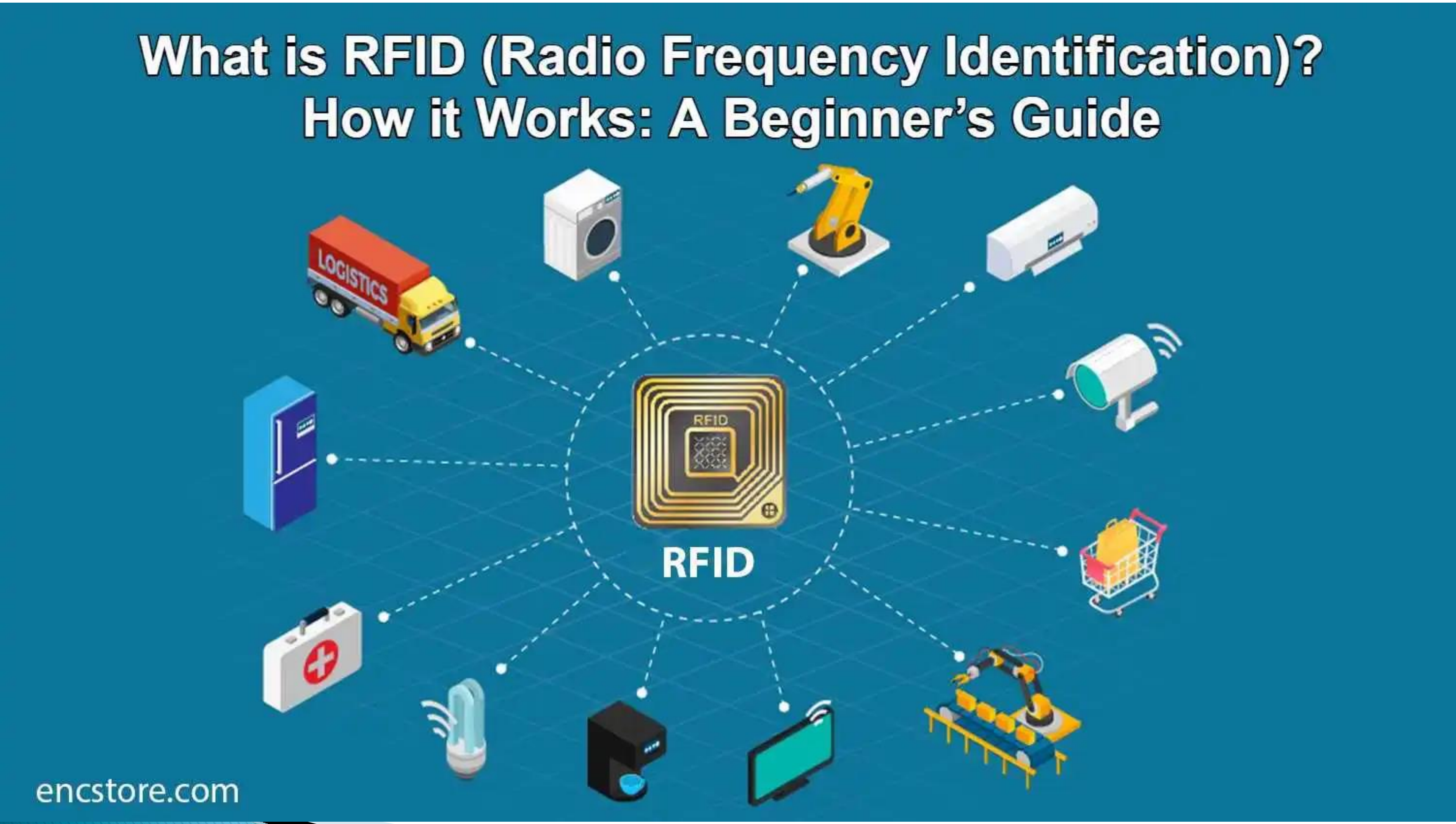
Virtual Machine (VM) escape is a significant security vulnerability in computer systems where a program running inside a virtual machine breaks out of its isolated environment and interacts with the host operating system. This breach can lead to severe security risks, as the attacker can potentially gain control over the host system and other virtual machines running on it.



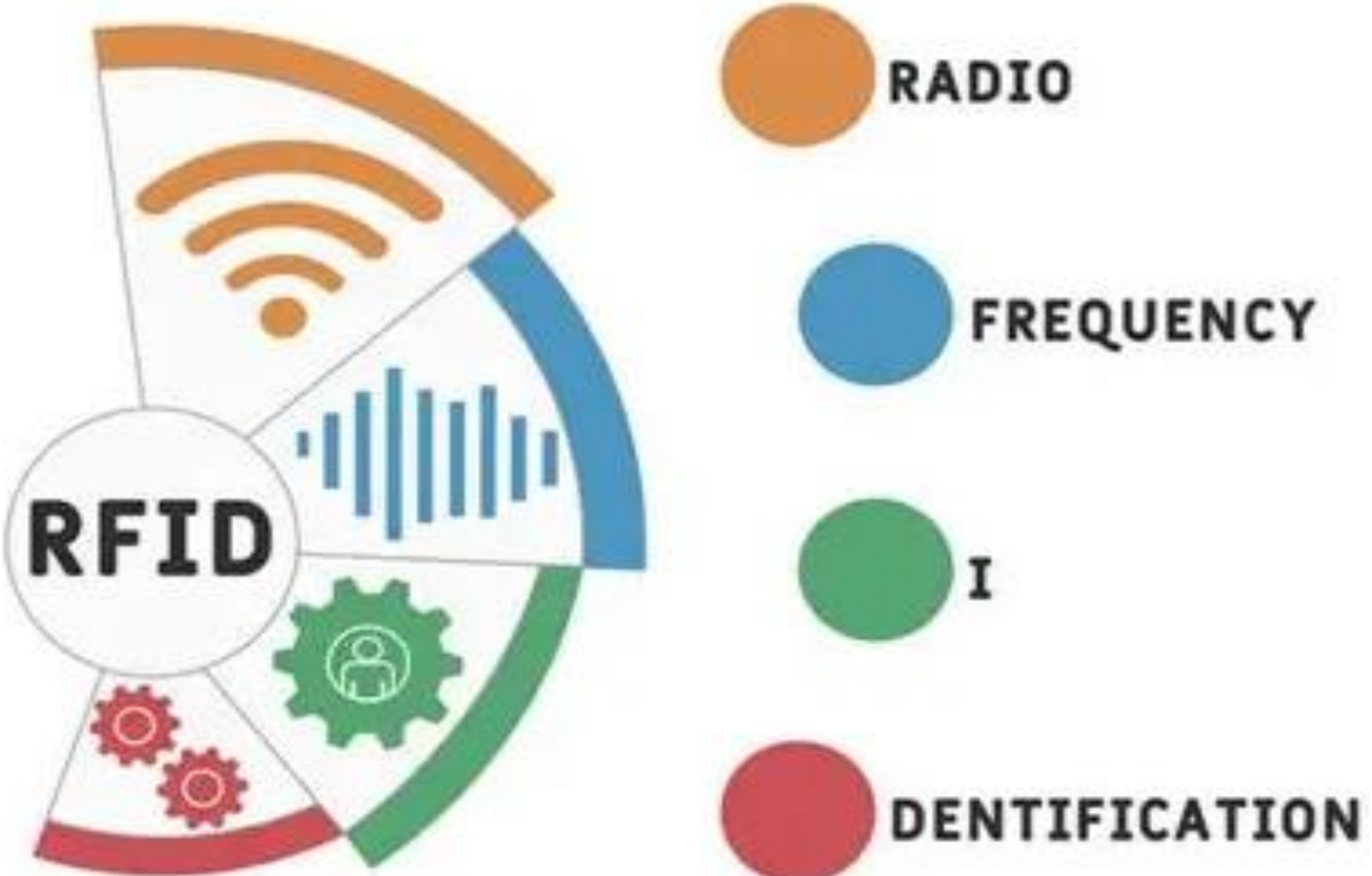
RFID (Radio-Frequency Identification)

- RFID Tags & Readers
- RFID Communication
- Applications
- Security Threats

RFID (Radio-Frequency Identification)


What is RFID (Radio Frequency Identification)?
How it Works: A Beginner's Guide
LOGISTICS
RFID
RFID
encstore.com

RFID (Radio-Frequency Identification)



RFID Tags & Readers

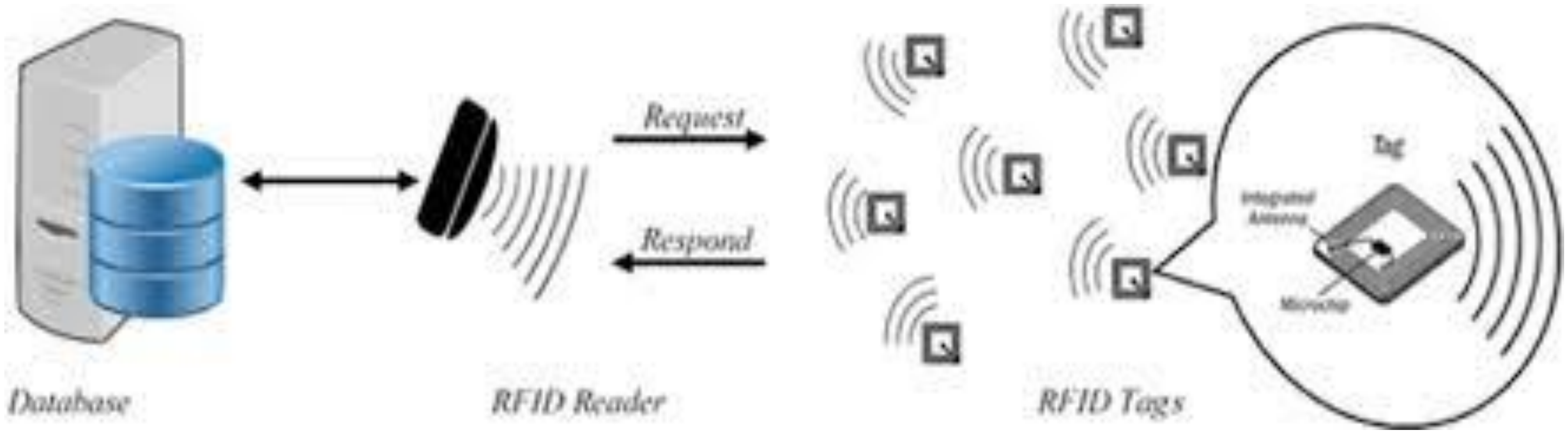
Radio Frequency Identification (RFID) is a form of wireless communication that incorporates the use of electromagnetic or electrostatic coupling in the radio frequency portion of the electromagnetic spectrum to uniquely identify an object or person. It uses radio frequency to search, identify, track, and communicate with items and people.

RFID Tags & Readers

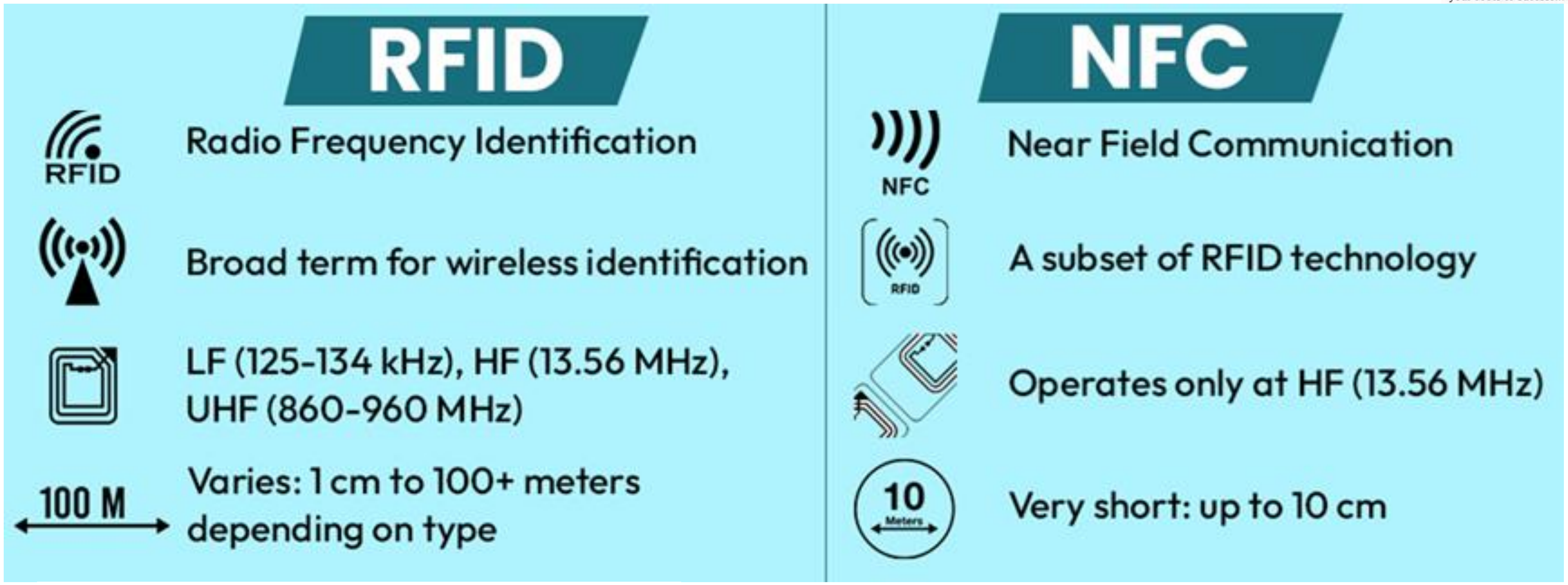
RFID Communication

RFID communication (Radio Frequency Identification) is a contactless technology using radio waves to wirelessly exchange data between a tag and a reader, identifying and tracking objects without a line-of-sight, unlike barcodes, enabling automated data capture for inventory, access control, and supply chain management. A system involves an RFID tag (microchip + antenna), a reader (transmits signals and receives data), and often a database, with tags being passive (reader-powered) or active (battery-powered) for different ranges and capabilities.

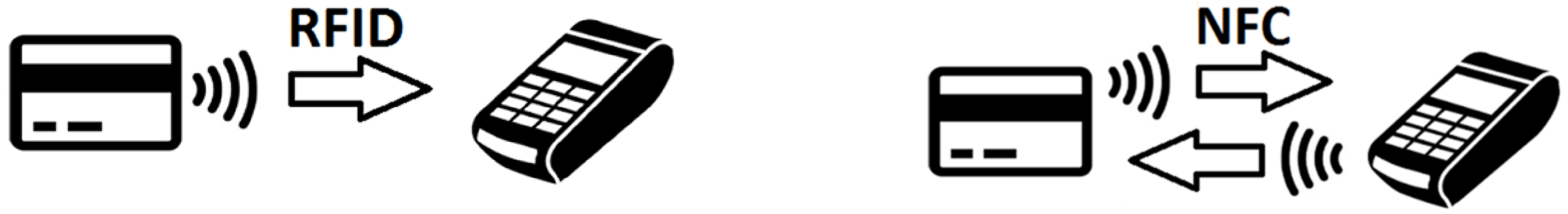
RFID Communication



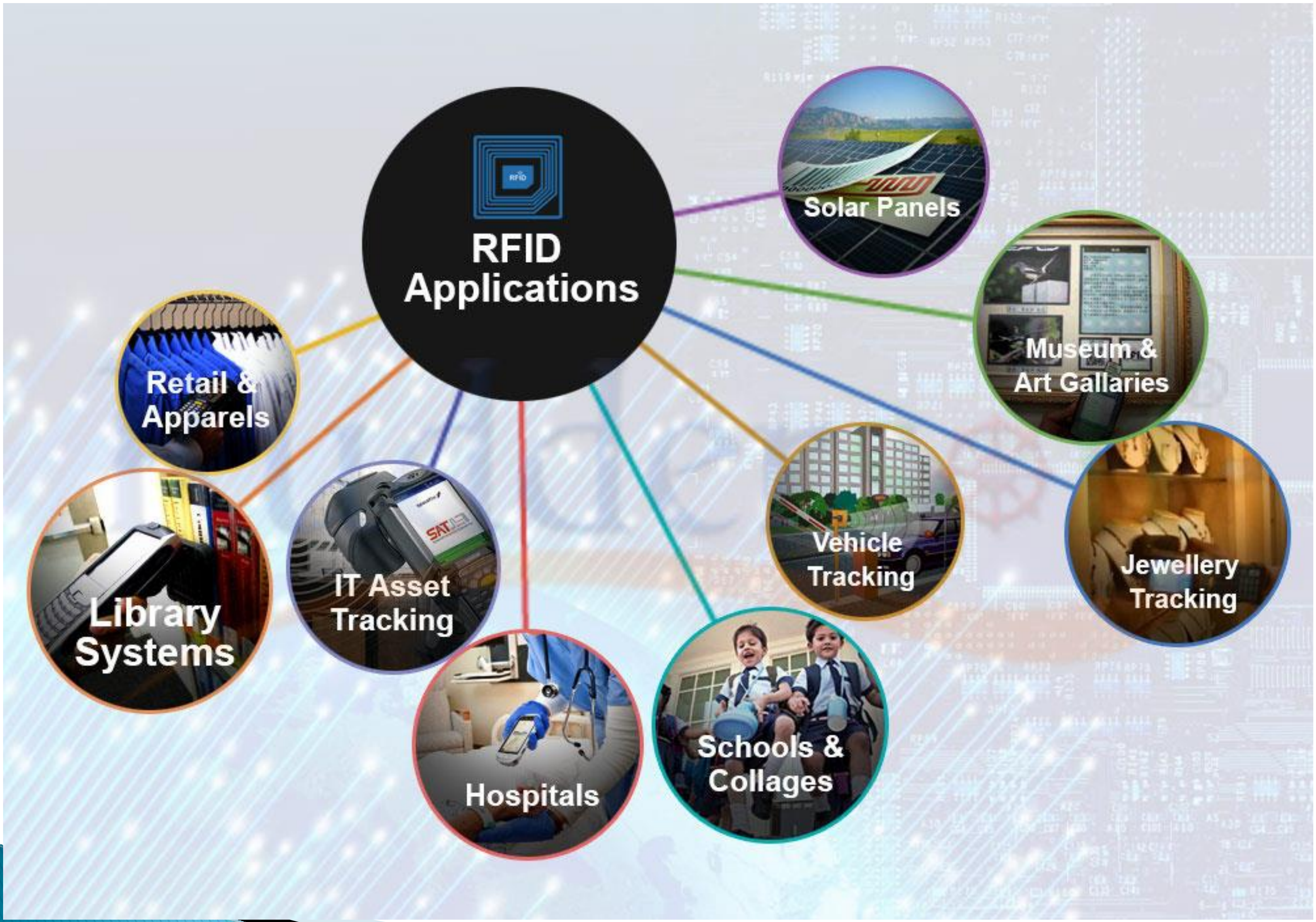
RFID Communication



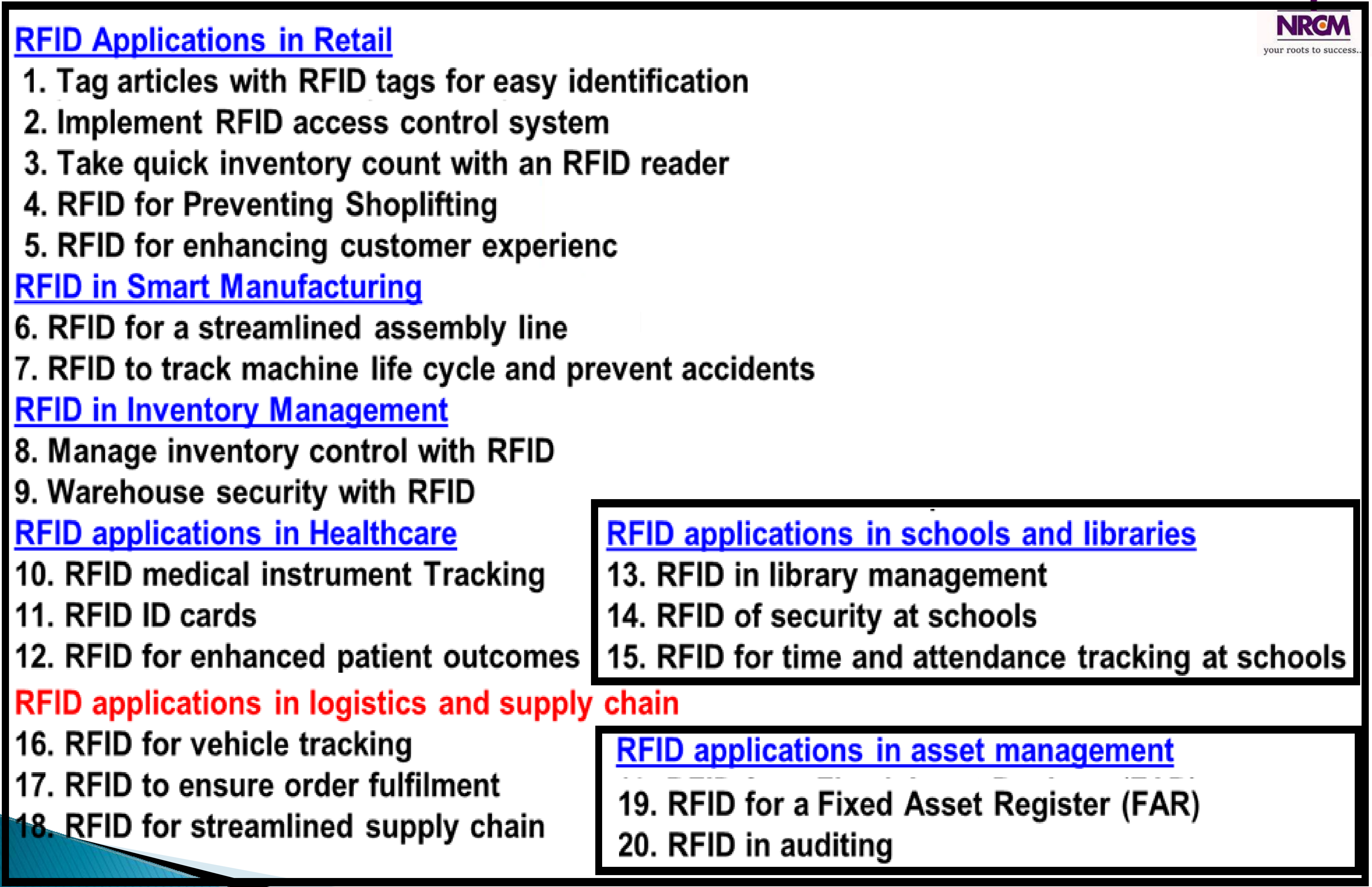
RFID	NFC
Radio Frequency Identification	Near Field Communication
Broad term for wireless identification	A subset of RFID technology
LF (125-134 kHz), HF (13.56 MHz), UHF (860-960 MHz)	Operates only at HF (13.56 MHz)
Varies: 1 cm to 100+ meters depending on type	Very short: up to 10 cm



RFID Applications



RFID Applications



RFID Applications in Retail

1. Tag articles with RFID tags for easy identification
2. Implement RFID access control system
3. Take quick inventory count with an RFID reader
4. RFID for Preventing Shoplifting
5. RFID for enhancing customer experienc

RFID in Smart Manufacturing

6. RFID for a streamlined assembly line
7. RFID to track machine life cycle and prevent accidents

RFID in Inventory Management

8. Manage inventory control with RFID
9. Warehouse security with RFID

RFID applications in Healthcare

10. RFID medical instrument Tracking
11. RFID ID cards
12. RFID for enhanced patient outcomes

RFID applications in schools and libraries

13. RFID in library management
14. RFID of security at schools
15. RFID for time and attendance tracking at schools

RFID applications in logistics and supply chain

16. RFID for vehicle tracking
17. RFID to ensure order fulfilment
18. RFID for streamlined supply chain

RFID applications in asset management

19. RFID for a Fixed Asset Register (FAR)
20. RFID in auditing

RFID Applications

RFID (RadioFrequency Identification) technology has various applications across multiple industries, including:

- Access Control
- Inventory Management
- Supply Chain Management
- Asset Tracking
- Healthcare
- Retail
- Animal Tracking

RFID Applications

LOGISTICS
RFID
RFID Tag

RFID Security Threats

RFID technology, while beneficial for tracking and identif ication, faces significant security threats including unaut horized access, cloning, and denial-of-service attacks

Radio Frequency Identification (RFID) uses **radio waves** to identify and track objects. It consists of RFID tags, readers, and a backend system that processes the data. RFID systems are widely used in various industries for applications such as inventory management, access control, and asset tracking. However, the convenience of RFID comes with vulnerabilities that can be exploited by malicious actors.

Common Security Threats

1). Unauthorized Access and Cloning, 2). Counterfeiting 3). Sniffing

4). Denial-of-Service (DoS) Attacks 5). Unauthorized Tracking

RFID Security Threats

RFID Security

Materials from the FIRB SAT lecture slides by Massimo Rimondini included with permission.

Windows Security Principles

Windows Security Principles

Windows Security Principles are fundamental to protecting IT systems and data. They are based on the security triad, which includes confidentiality, integrity, and availability. These principles guide the protection of information technology systems and data, ensuring that they are safeguarded against loss, unauthorized access, and misuse. Windows Security integrates advanced hardware and software protection, offering robust security from chip to cloud, which is essential for organizations in hybrid work environments.

- **Windows Authentication**
- **User Accounts & Privileges**
- **Processes & Threads**

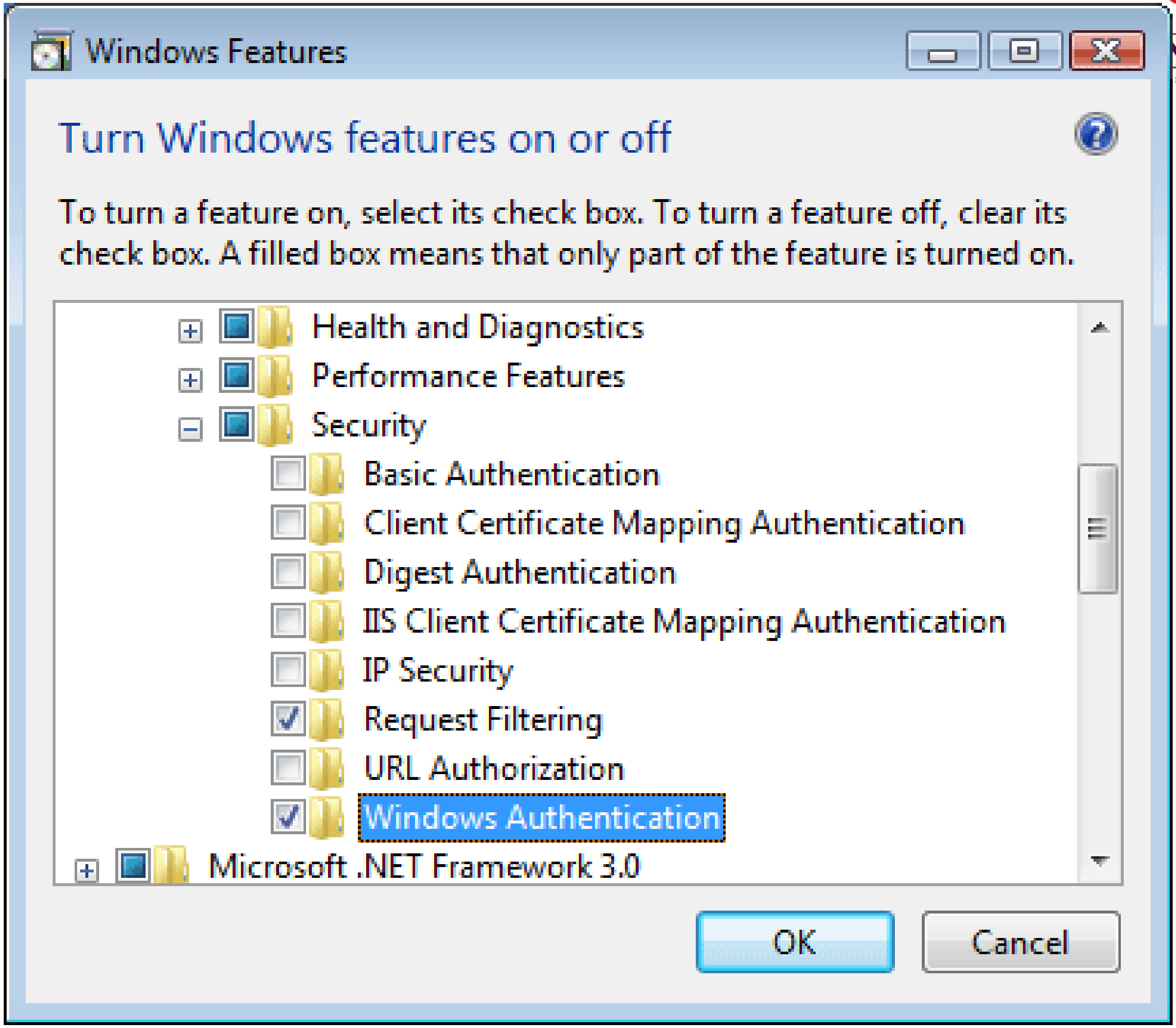
• Windows Authentication

This navigation topic for the IT professional lists documentation resources for Windows authentication and logon technologies that include product evaluation, getting started guides, procedures, design and deployment guides, technical references, and command references.

The server side of the authentication exchange compares the signed data with a known cryptographic key to validate the authentication attempt.

For more information about Windows Authentication including

- Windows Authentication Concepts
- Windows Logon Scenarios
- Windows Authentication Architecture
- Security Support Provider Interface Architecture
- Credentials Processes in Windows Authentication
- Group Policy Settings Used in Windows Authentication



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ▨ Health and Diagnostics
- ▨ Performance Features
- ▨ Security
  - ☐ Basic Authentication
  - ☐ Client Certificate Mapping Authentication
  - ☐ Digest Authentication
  - ☐ IIS Client Certificate Mapping Authentication
  - ☐ IP Security
  - ✓ Request Filtering
  - ☐ URL Authorization
  - ✓ Windows Authentication
- ▨ Microsoft .NET Framework 3.0

• User Accounts & Privileges

User account privileges determine what actions a user can perform on a computer, with common types being Administrator and Standard Use r accounts.

Types of User Accounts:
1). **Administrator Account**
2). **Standard User Account**
3). **Guest Account**

Managing User Account Privileges:
1). **Creating and Modifying Accounts**

You can create new user accounts through the Settings app in Windows. Navigate to **Settings > Accounts > Family & other users** to add a ne w user.

2). **Changing Account Types**
Settings app, **Control Panel**,or using the **Command Prompt**. For example, in the Settings app, select the user account and choose **Chan ge account type** to adjust privileges.

3). **Elevated Privileges**

• Processes & Threads

A **process** is an instance of a program that is executed independently. It includes the program code, its current activity, and the resources it requires, such as memory space and file descriptors. Each process has its own memory space, which provides isolation and protection from other processes.

A **thread** is a lightweight unit of execution within a process. Multiple threads can exist within a single process, sharing the same memory space and resources, which allows for faster communication and resource sharing.

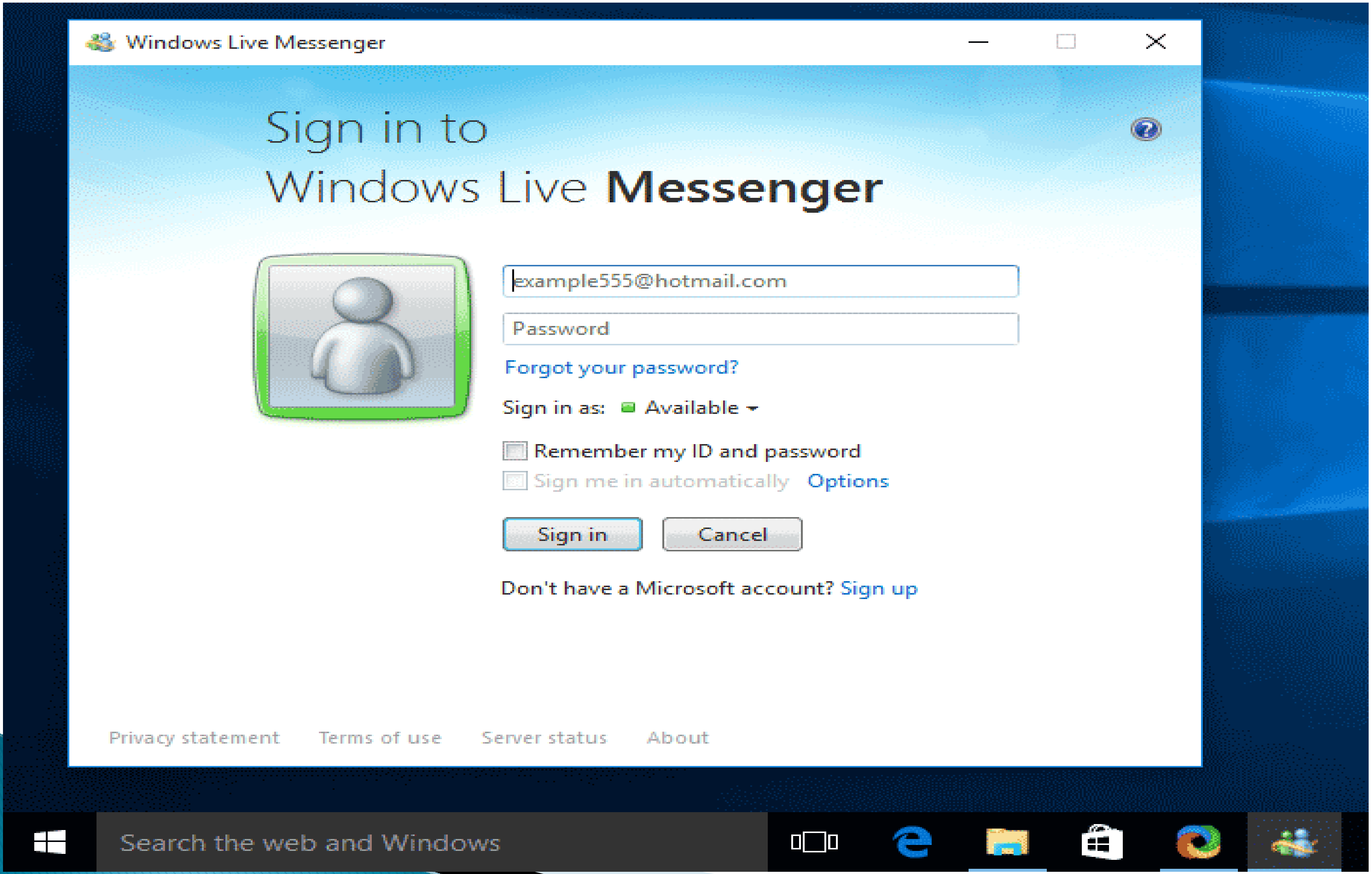
Windows Tokens

- Access Tokens
- Primary Tokens
- Impersonation Tokens
- Token Privileges

Windows Messaging

- Message Queues
- Event Handling
- Security Risks

Windows Messaging



Windows Live Messenger

Sign in to
Windows Live **Messenger**

example555@hotmail.com

Password

Forgot your password?

Sign in as: Available

Remember my ID and password

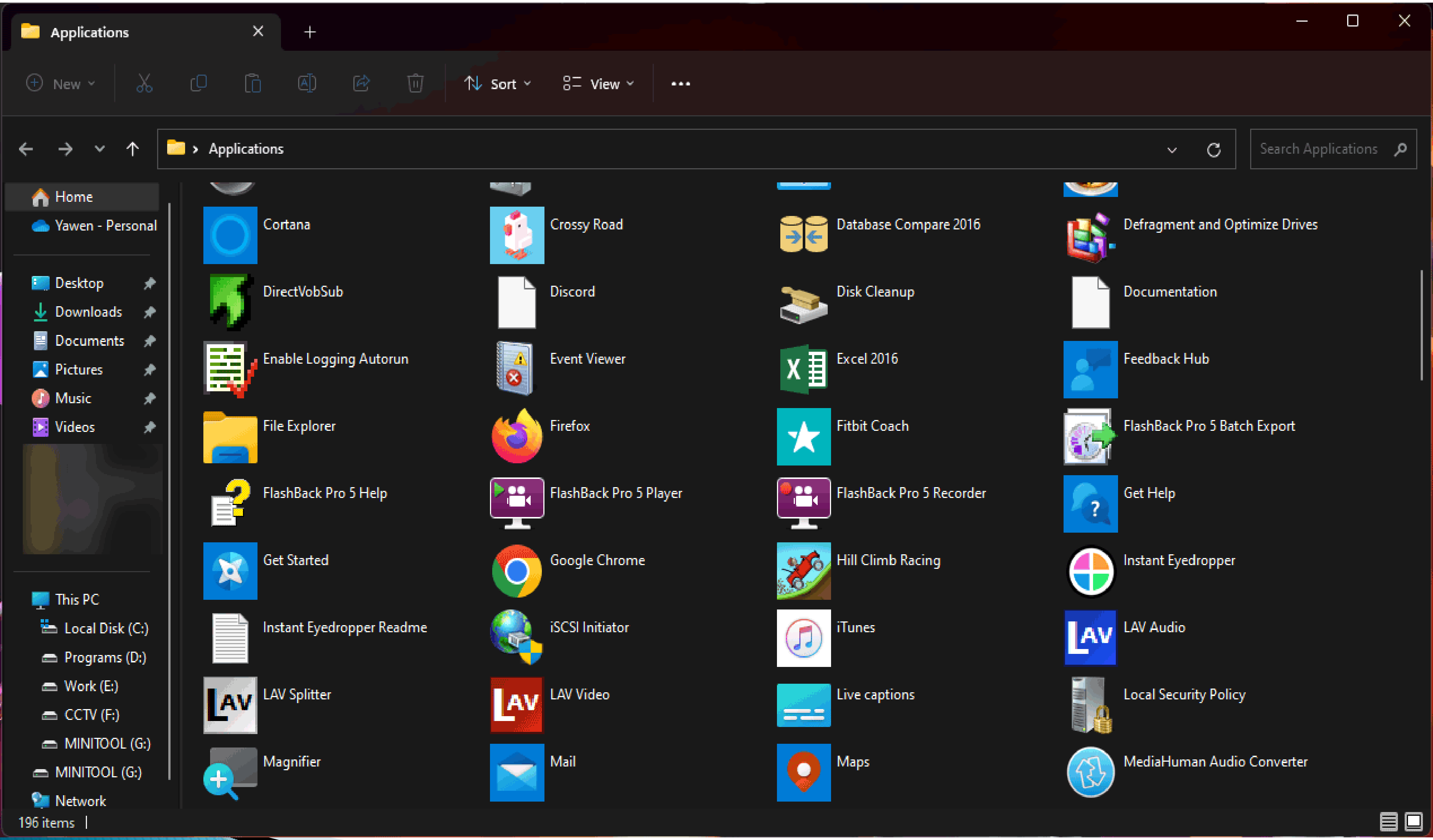
Sign me in automatically   Options

Don't have a Microsoft account? Sign up

Windows Programs

- Process Architecture
- Memory Model
- Application Security

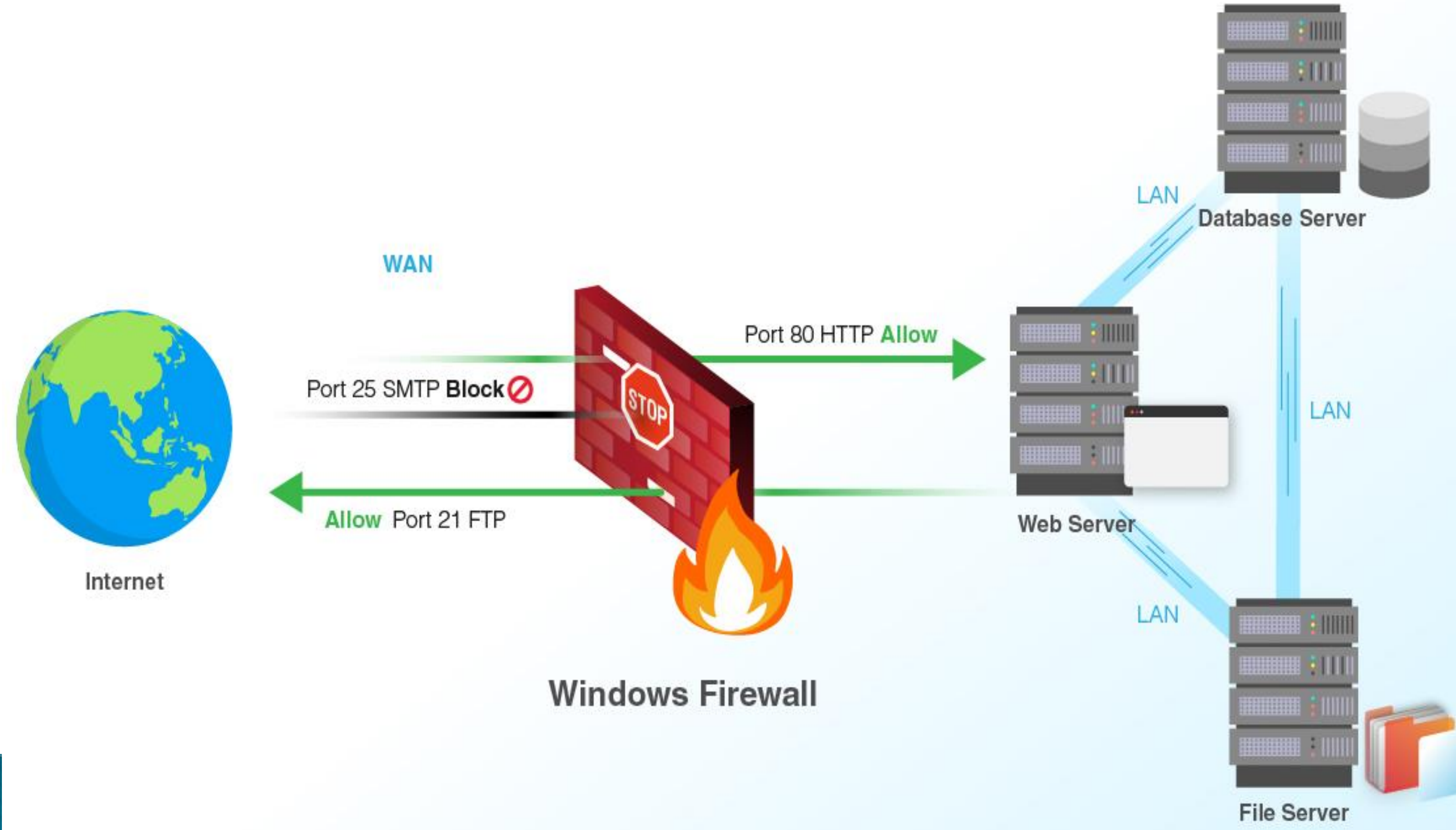
Windows Programs



Windows Firewall

- Profiles (Domain/Private/Public)
- Rules & Filtering
- Integration with Defender

Windows Firewall



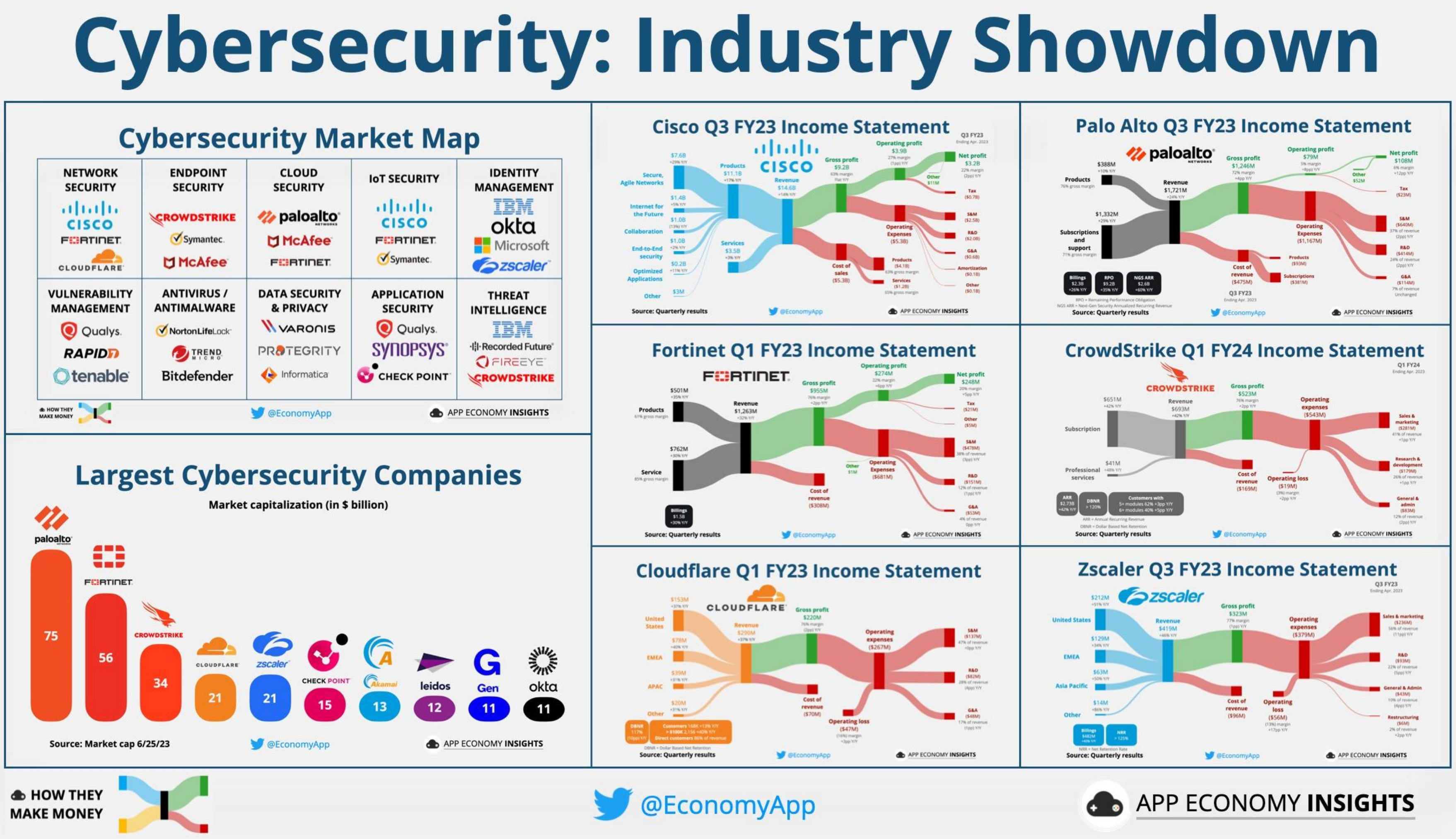
Cyber Security in Industry

- Modern Cyber Threats
- Malware Trends
- Zero Trust Architecture

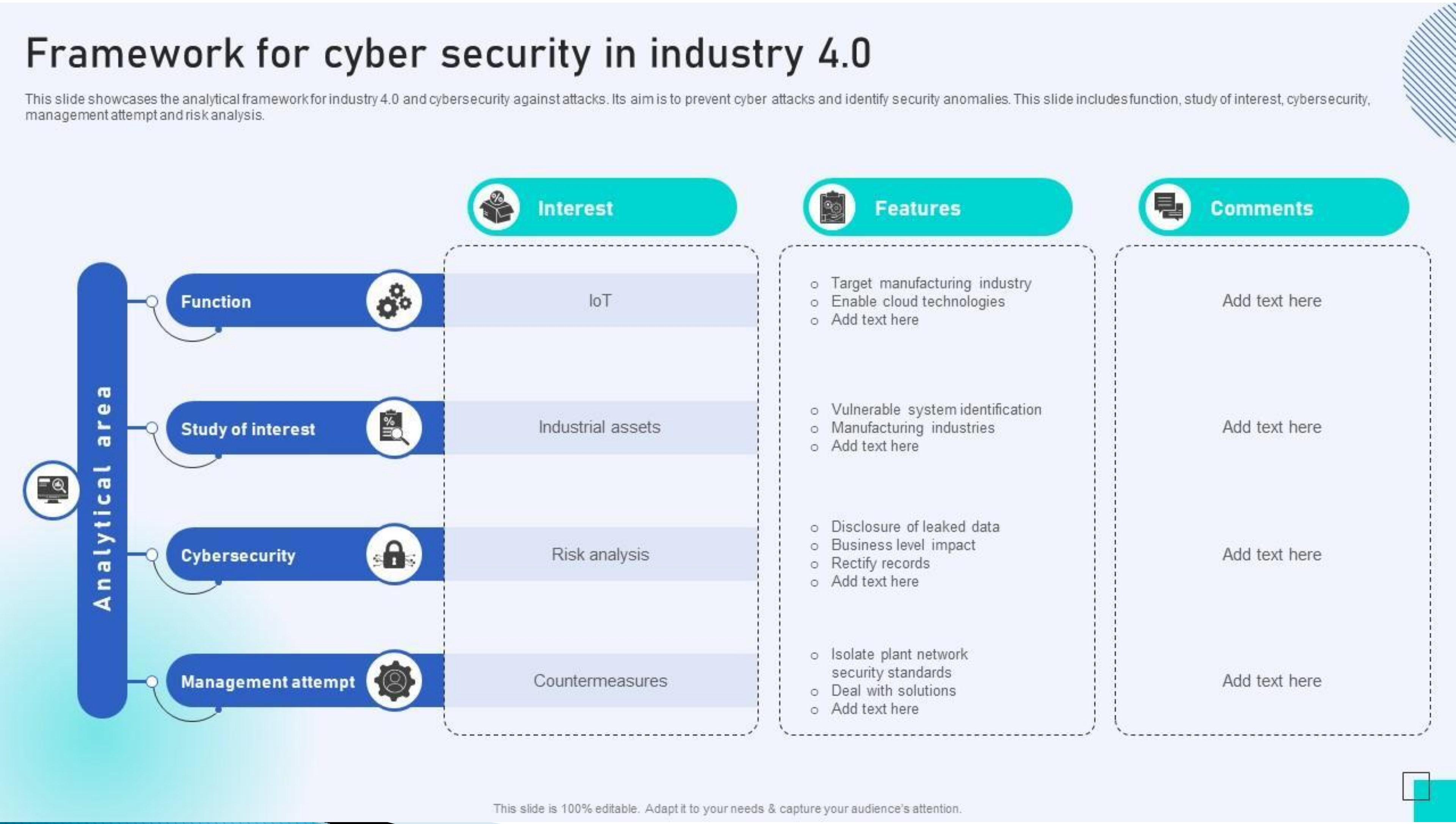
Cyber Security in Industry

CYBER SECURIT
Global Cyber Security Market Size, 2018-2026 (USD Billion)
$131.3 Billion by 2018
$289.8 Billion by 2026
Global Cyber Security Market Share, By Sol
22.
End-point Security
Application Security
Cloud Sec
Others
North America Cyber Security Market Size, 2018
USD 53.3 Billion

Cyber Security in Industry



Cyber Security in Industry



Framework for cyber security in industry 4.0

This slide showcases the analytical framework for industry 4.0 and cybersecurity against attacks. Its aim is to prevent cyber attacks and identify security anomalies. This slide includes function, study of interest, cybersecurity, management attempt and risk analysis.

Analytical area	Interest	Features	Comments
Function	IoT	○ Target manufacturing industry ○ Enable cloud technologies ○ Add text here	Add text here
Study of interest	Industrial assets	○ Vulnerable system identification ○ Manufacturing industries ○ Add text here	Add text here
Cybersecurity	Risk analysis	○ Disclosure of leaked data ○ Business level impact ○ Rectify records ○ Add text here	Add text here
Management attempt	Countermeasures	○ Isolate plant network security standards ○ Deal with solutions ○ Add text here	Add text here

This slide is 100% editable. Adapt it to your needs & capture your audience's attention.

Summary

- Fundamentals of InfoSec
- Cryptography Principles
- Network Defense
- Windows Security

UNIT-II

UNIT–II

Attacker Techniques and Motivations:

1). How Hackers Cover Their Tracks (Antiforensics)

● How and Why Attackers Use Proxies,
● Detecting the Use of Proxies
● Conclusion

2). Tunneling Techniques,

● HTTP
● DNS (Domain Name System)
● ICMP(Internet Control Message Protocol):
● Intermediaries, Steganography, and Other Concepts
● Detection and Prevention

3). Fraud Techniques,

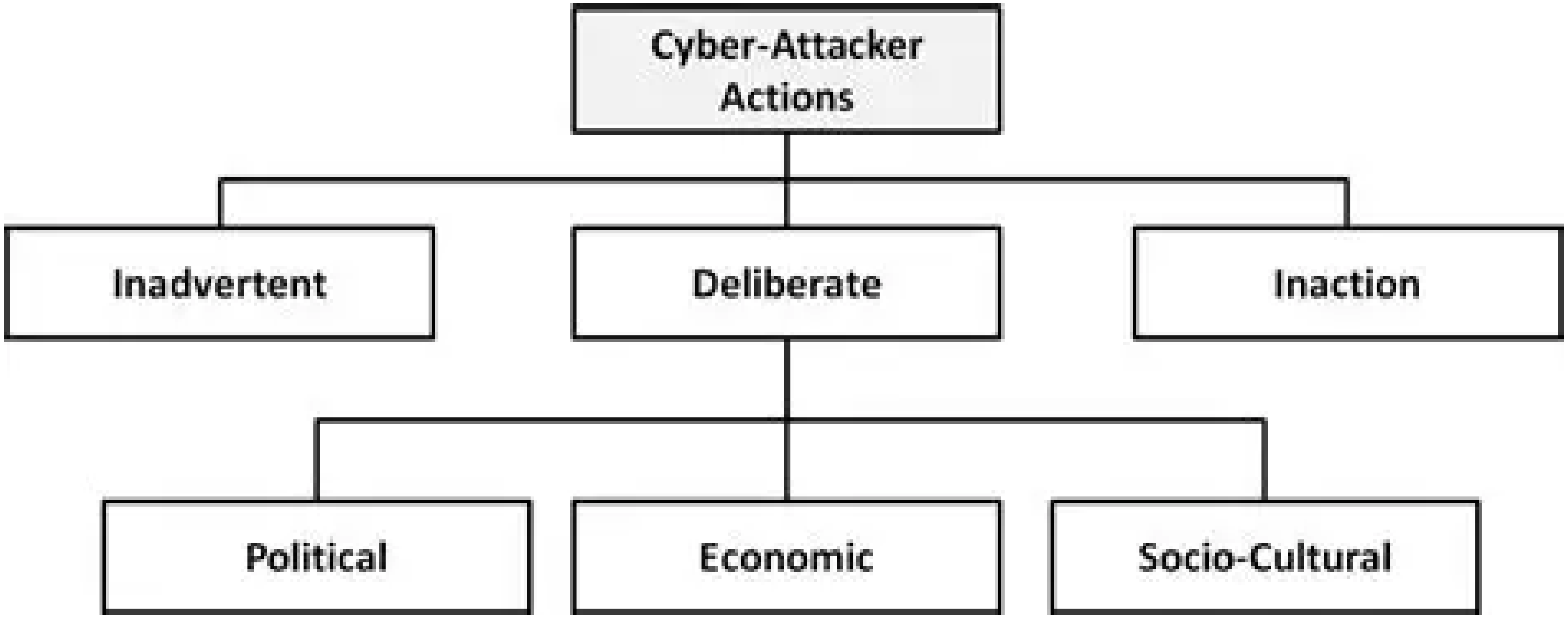
● Phishing, Smishing, Vishing, and Mobile Malicious Code
a). Mobile Malicious Code b). Phishing against Mobile Devices
c). Conclusions
● Rogue Antivirus
a). Following the Money: Payments b). Conclusion
● Click Fraud: a). Pay-per-Click b). Click Fraud Motivations c). Click Fraud Tactics & Detection
d). Conclusions.

4). Threat Infrastructure:

● Botnets ● Fast-Flux ● Advanced Fast-Flux

Attacker Techniques and Motivations

- Attackers (hackers) use various techniques to achieve objectives such as:
  - Financial gain
  - Data theft / espionage
  - Cyber warfare / political motives
  - Reputation / challenge
  - Disruption (DDoS, sabotage)
- Their techniques depend on skill level, resources,


Cyber-Attacker Actions
Inadvertent
Deliberate
Inaction
Political
Economic
Socio-Cultural

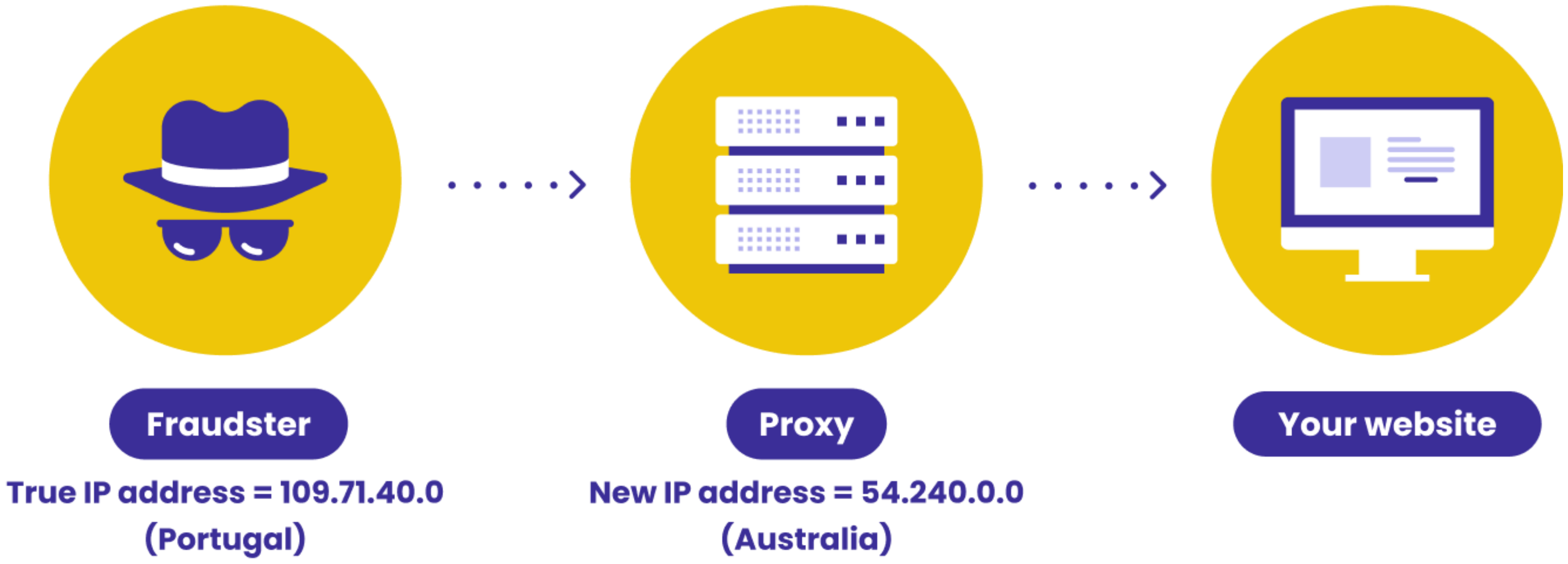
HOW DO
HACKERS COVER
THEIR TRACKS?
SUBSCRIBE

Attacker Techniques and Motivations:

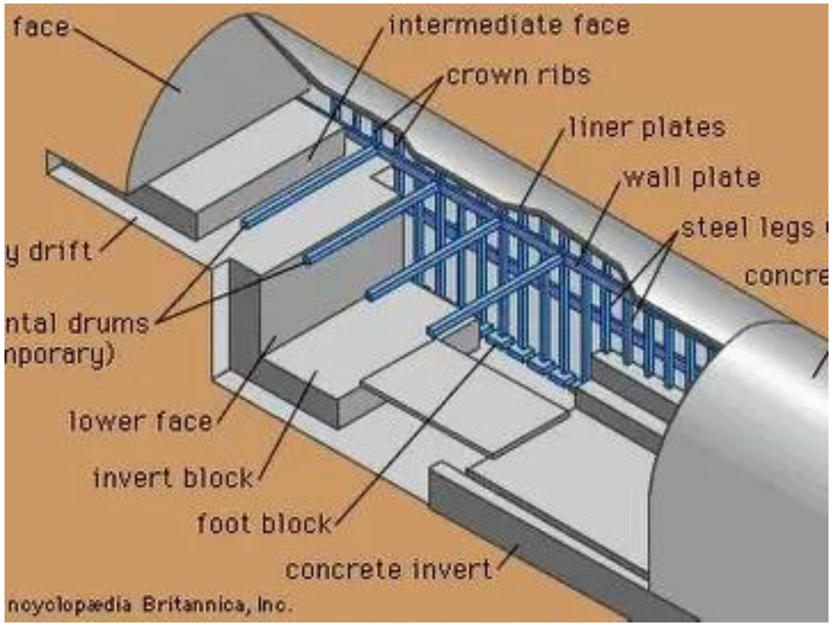
1). How Hackers Cover Their Tracks (Antiforensics)

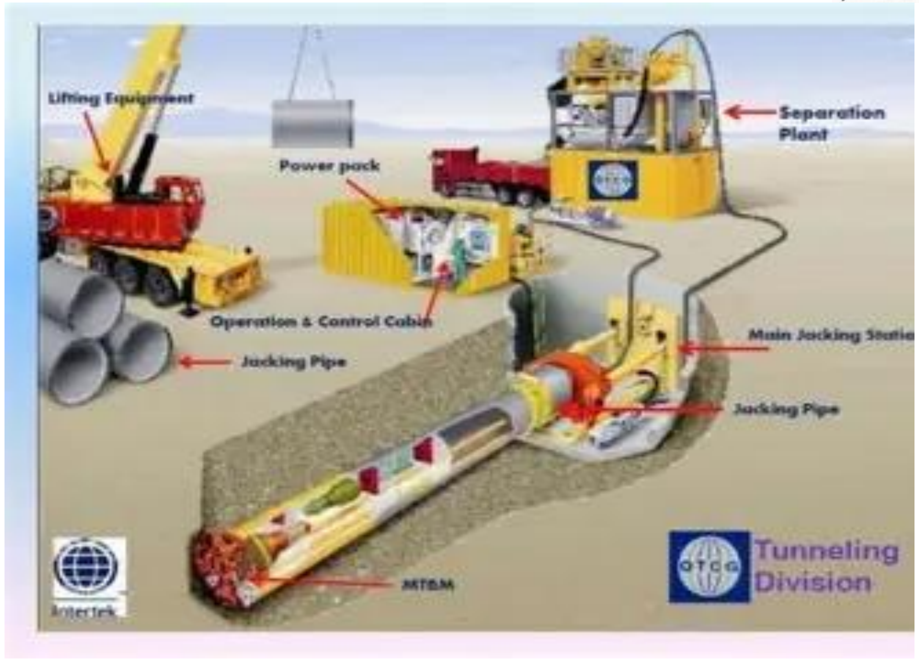
- How and Why Attackers Use Proxies,
- Detecting the Use of Proxies
- Conclusion

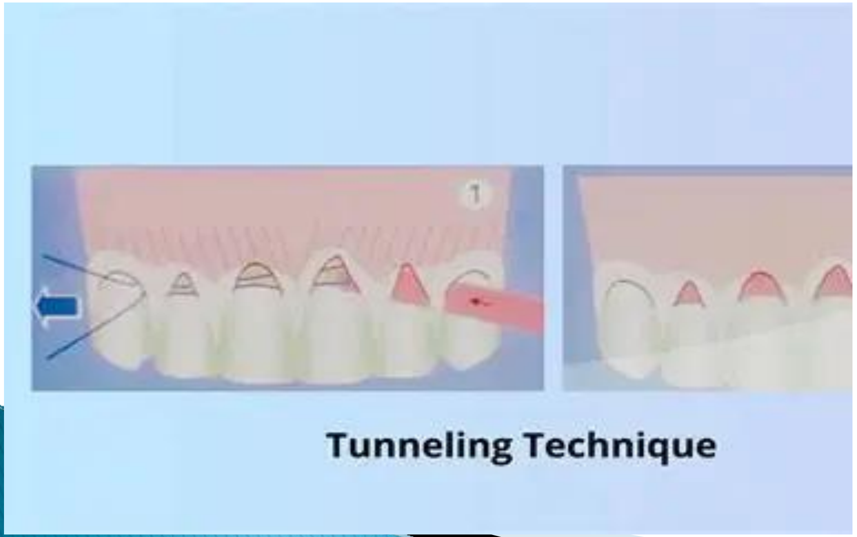
Detecting the Use of Proxies



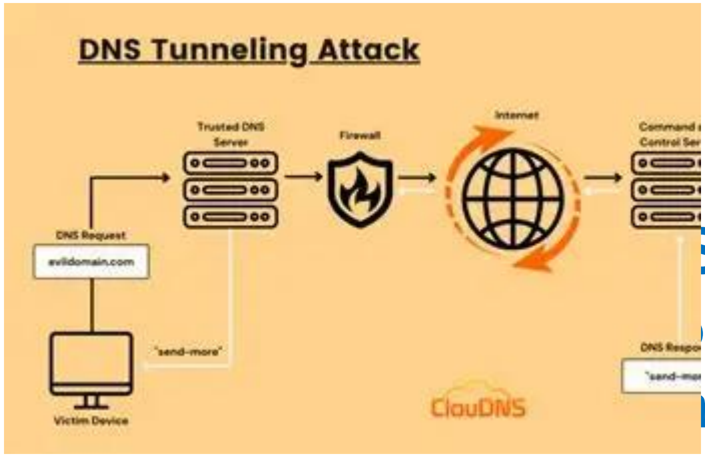
2). Tunneling Techniques,

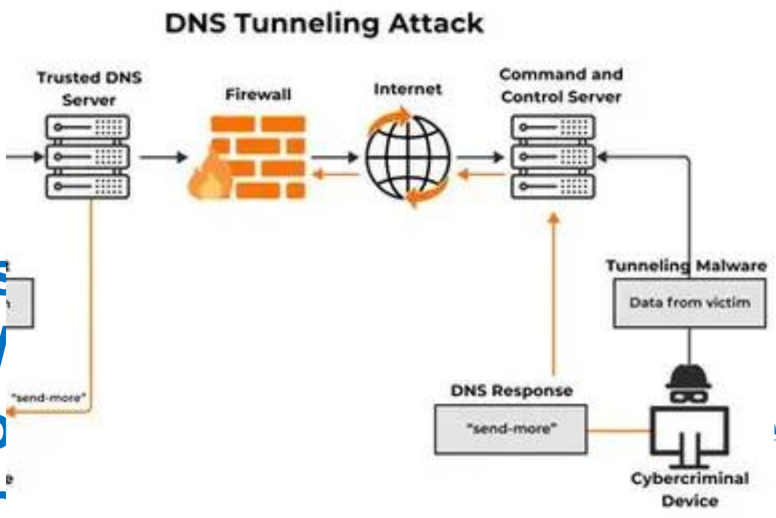






2). Tunneling Techniques,





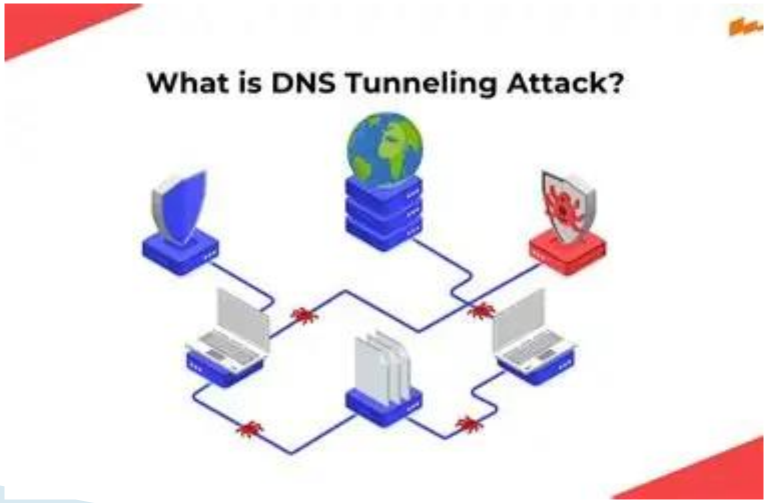
Sys

l M

nd

pts

- Detection and Preventic…



3). Fraud Techniques

- **Phishing, Smishing, Vishing, and Mobile Malicious Code**
  - a). Mobile Malicious Code
  - b). Phishing against Mobile Devices
  - c). Conclusions
- **Rogue Antivirus**
  - a). Following the Money: Payments b). Conclusion
- **Click Fraud:**
  - a). Pay-per-Click
  - b). Click Fraud Motivations
  - c). Click Fraud Tactics & Detection
  - d). Conclusions.

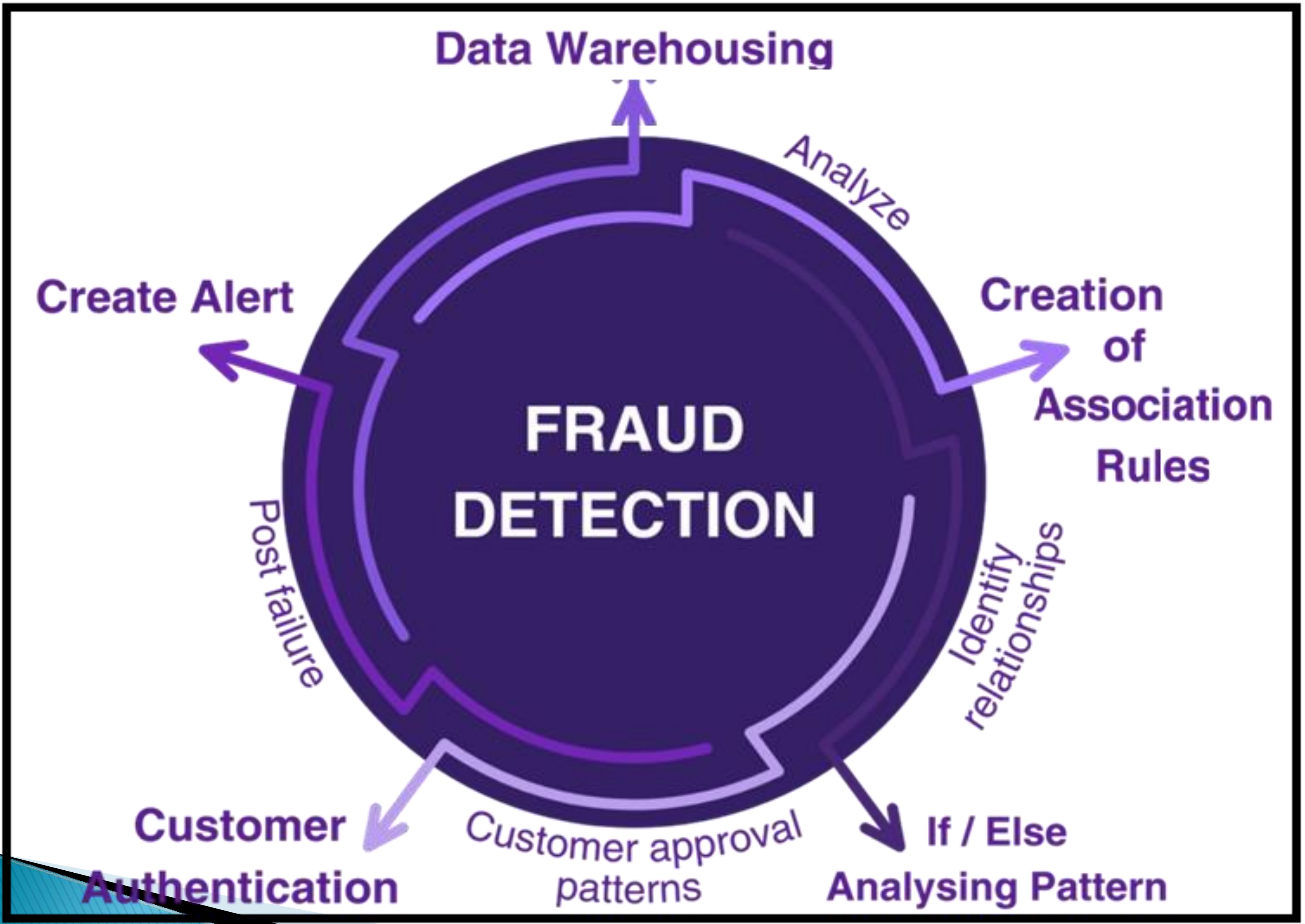
3). Fraud Techniques

Fraud techniques include methods like phishing and account takeover, often using malware or social engineering, as well as physical tactics like dumpster diving. Companies combat these threats using advanced techniques such as machine learning, behavioral analytics, and real-time transaction monitoring. These methods are crucial for identifying and preventing fraudulent activities in various sectors.

Common Fraud Techniques

1). Identity Theft
2). Phishing Scams
3). Asset Misappropriation
4). Cyber Fraud
5). Financial Statement Fraud

3). Fraud Techniques



● Phishing, Smishing, Vishing, and Mobile Malicious Code

Phishing, Smishing, Vishing, and Mobile Malicious Code are all forms of cyber_attacks that exploit trust, urgency, and communication platforms to steal sensitive data. These attacks can take various forms, including emails, SMS, phone calls, and QR codes, and they often involve social engineering techniques to deceive victims into revealing personal information or clicking on malicious links.

● Phishing, Smishing, Vishing, and Mobile Malicious Code

PHISHING
VISHING
No Caller ID
Decline
Accept
SMISHING

● Phishing, Smishing, Vishing, and Mobile Malicious Code

Phishing

Email messages claiming to be from reputable sources urge the recipient to reveal personal information online or download malware.

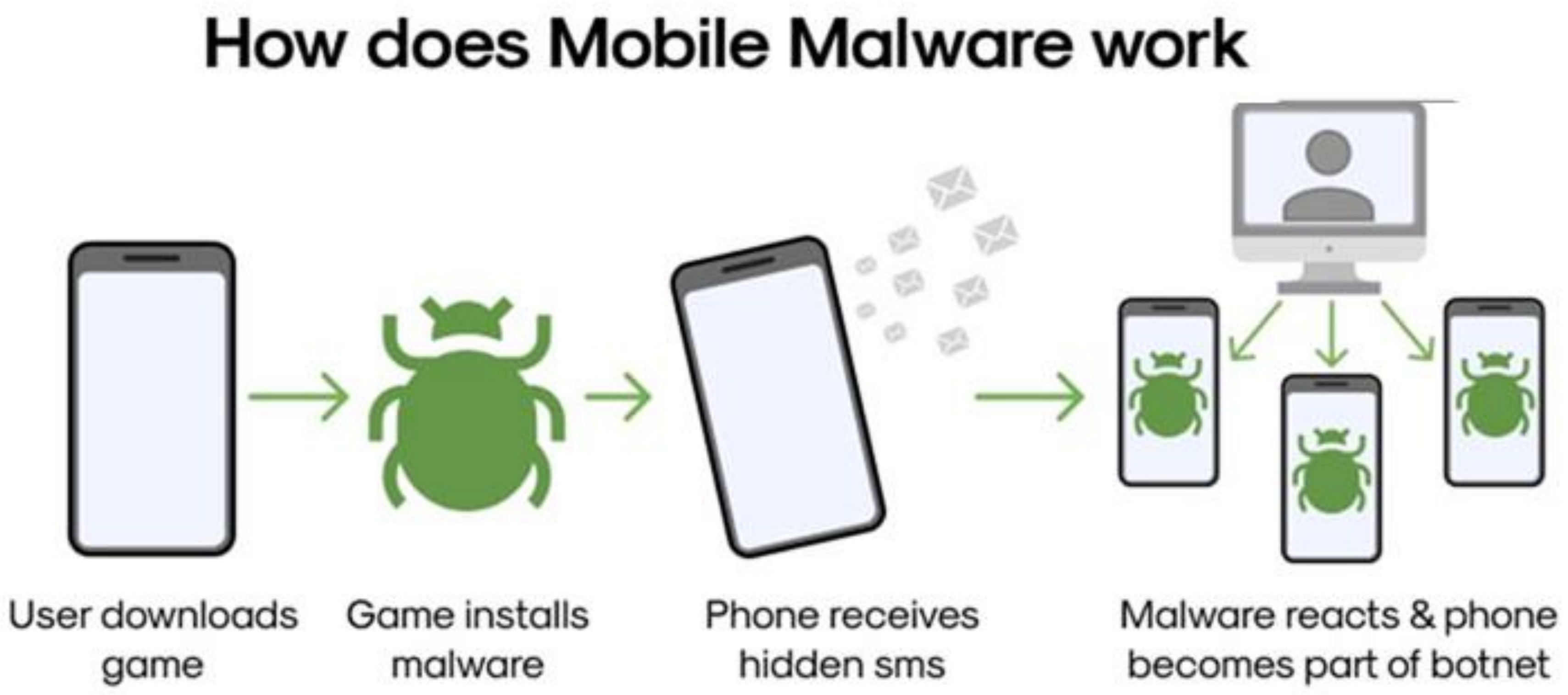
Vishing

Fraudulent phone calls urge the recipient to share sensitive information that will be used for identity theft.

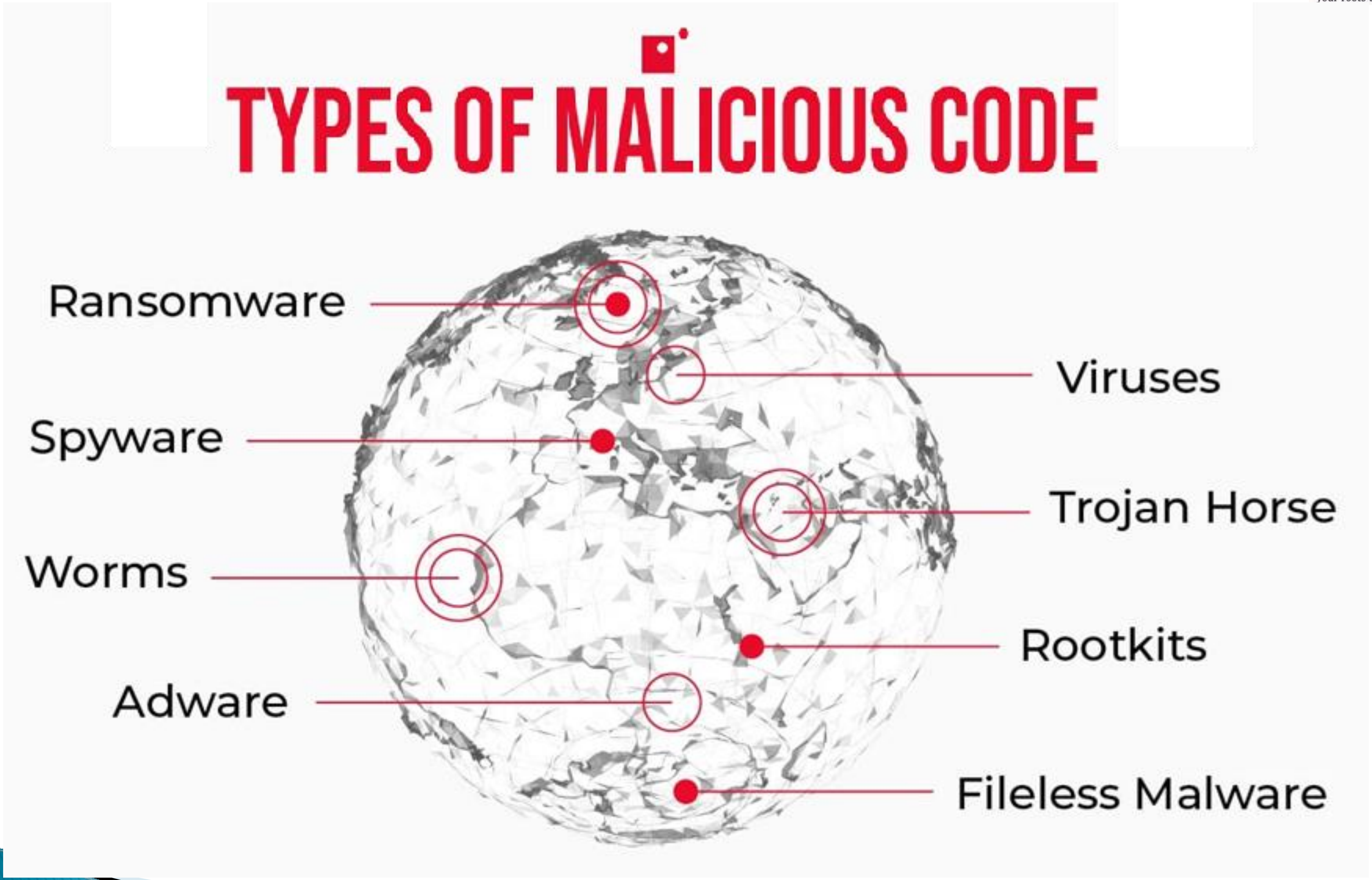
Smishing

Text messages urge the recipient to click a link that downloads malware onto their device or links to a spoofed website.

● Phishing, Smishing, Vishing, and Mobile Malicious Code


How does Mobile Malware work
User downloads game
Game installs malware
Phone receives hidden sms
Malware reacts & phone becomes part of botnet

● Phishing, Smishing, Vishing, and Mobile Malicious Code



a). Mobile Malicious Code

Mobile Malware

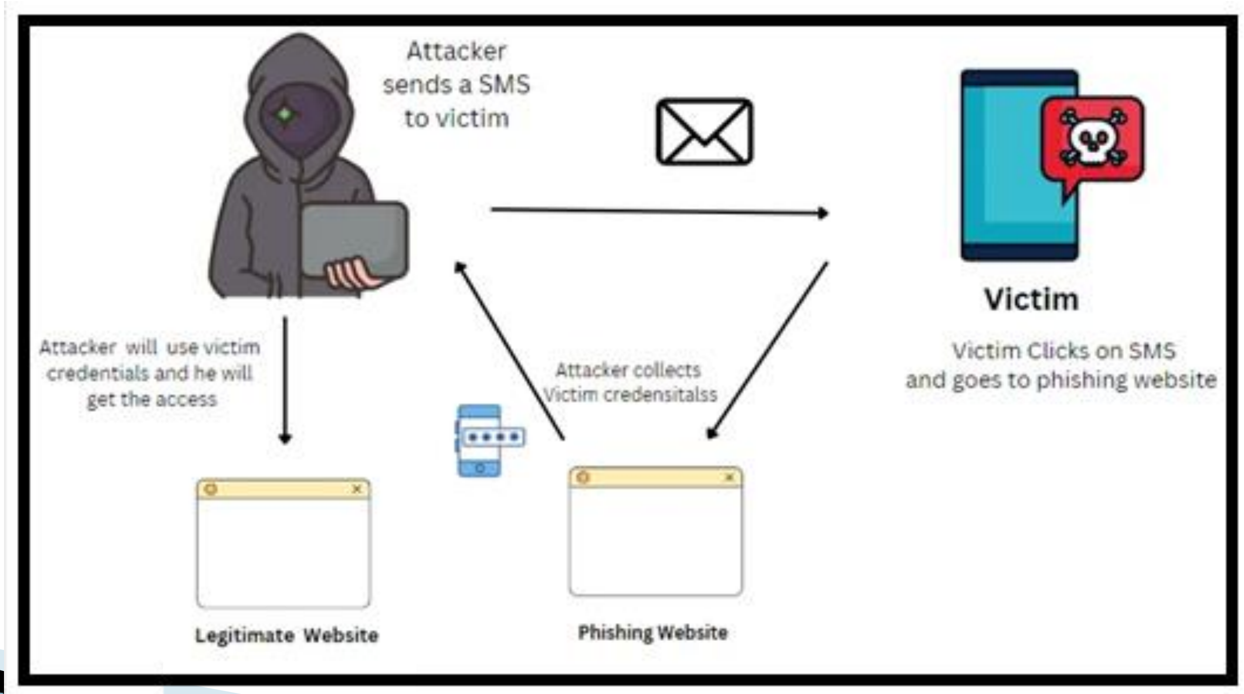
In today’s hyper-connected world, mobile devices are prime targets for cyber threats, with mobile malware leading the charge. From sneaky apps that steal your data to malicious code that locks your phone for ransom, mobile malware poses a growing risk to personal security and privacy. This guide dives deep into understanding what mobile malware is, how it infiltrates your device, and the steps you can take to stay protected in an ever-evolving digital landscape.

- **Phishing, Smishing, Vishing, and Mobile Malicious Code**

b). Phishing against Mobile Devices

Mobile phishing is a growing threat that exploits the convenience of smar tphones, using tactics like SMS phishing (smishing), voice phishing (vishi ng), and malicious apps to deceive users into revealing sensitive informat ion or downloading malware.

Unmasking Mobile Phishing Strategies, Tactics, and Defence


Attacker sends a SMS to victim
Victim
Victim Clicks on SMS and goes to phishing website
Attacker will use victim credentials and he will get the access
Attacker collects Victim credensitalss
Legitimate Website
Phishing Website

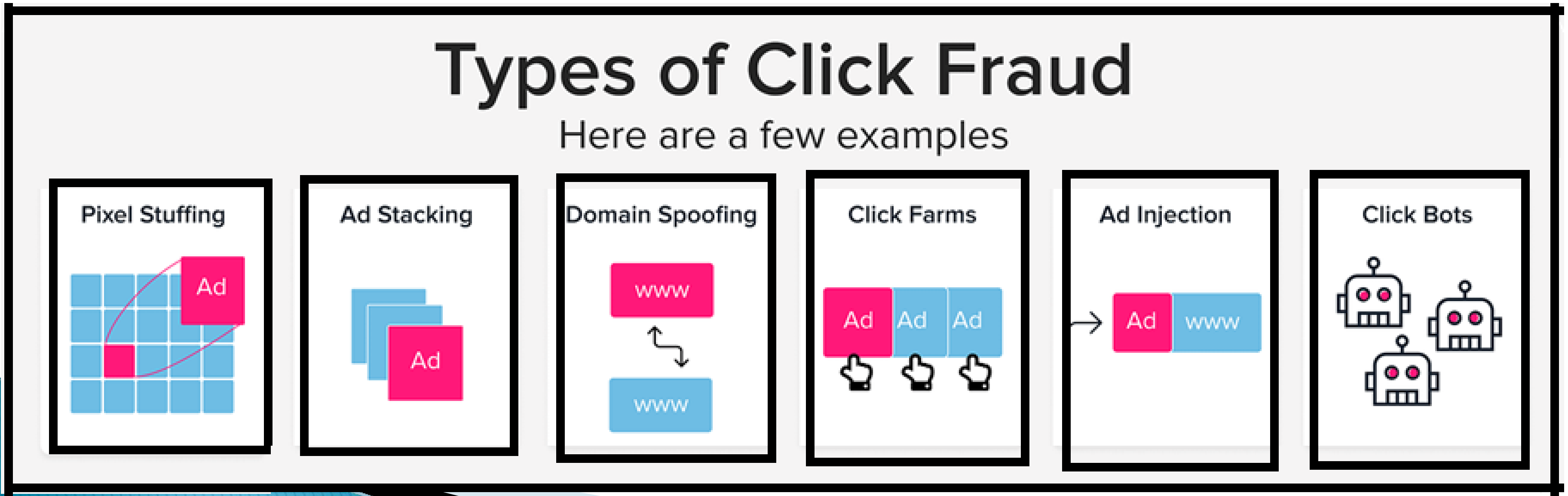
Rogue Antivirus

a). Following the Money: Payments   b). Conclusion

Rogue antivirus, or fake antivirus software, is perhaps the most insidious and harmful type of malware available today. It claims to guard your computer, but actually does the opposite—by making you think your system is infected, it compels you to pay for phony remedies that don't deliver anything. These tactics of intimidation are intended to instill fear so you'll be more inclined to click, download, or even provide payment information.

● Click Fraud:

Click fraud is a deceptive practice where fake clicks artificially inflate online ad engagement, draining budgets and skewing data, often done via automated bots or human click farms to boost publisher revenue or sabotage competitors' campaigns. It wastes advertiser money, reduces ROI, damages trust, and can lead to legal issues, impacting legitimate businesses using pay-per-click (PPC) models.



● Click Fraud:

a). Pay-per-Click
b). Click Fraud Motivations
c). Click Fraud Tactics & Detection
d). Conclusions.

Common types

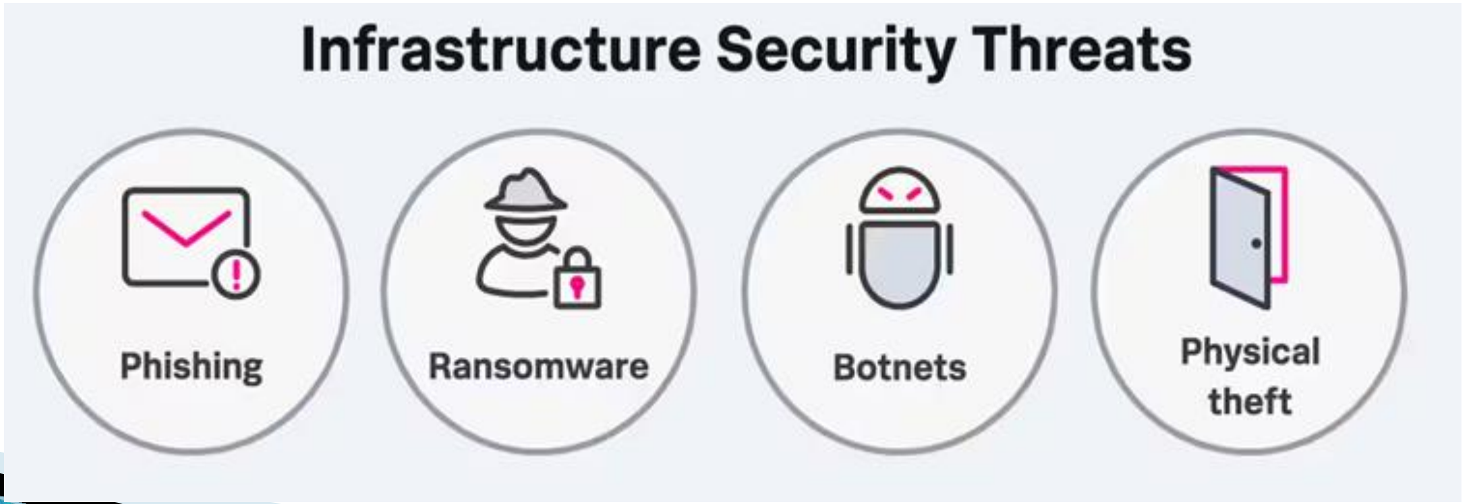
- Bot Clicks:
- Click Injection/Flooding:
- Incentivized Clicks:

4). Threat Infrastructure:

- Botnets
- Fast-Flux
- Advanced Fast-Flux

4). Threat Infrastructure:

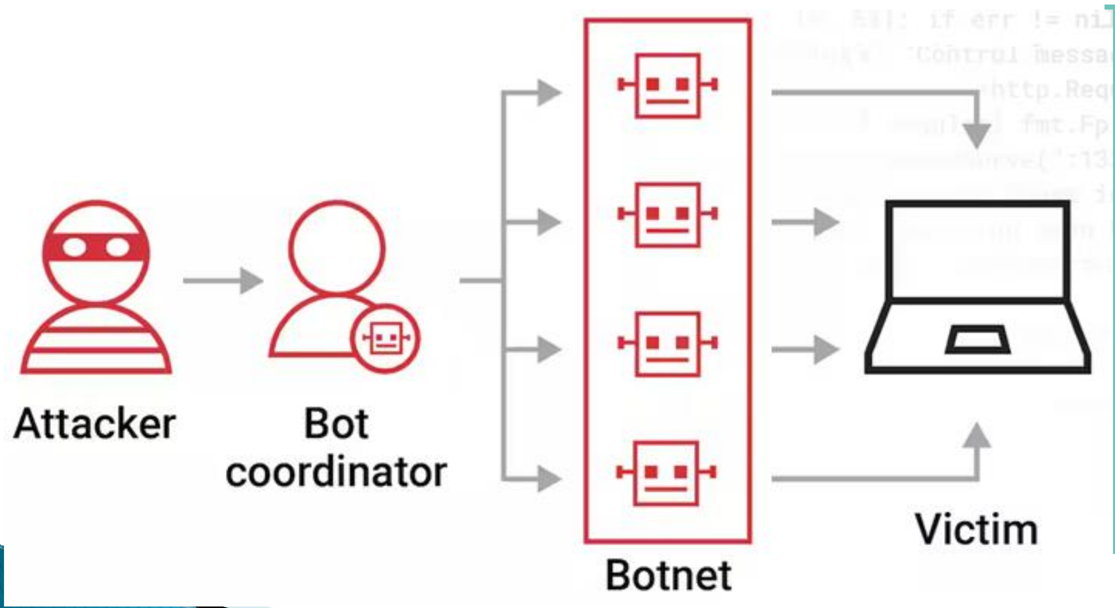
Threat infrastructure refers to the underlying systems (physical and digital) supporting society, like power grids, finance, and communications, facing risks from cyberattacks (DDoS, malware, hacking), physical attacks, natural disasters, or insider threats, all aiming to disrupt or control these essential services, necessitating strong cybersecurity, resilience planning, and threat intelligence.

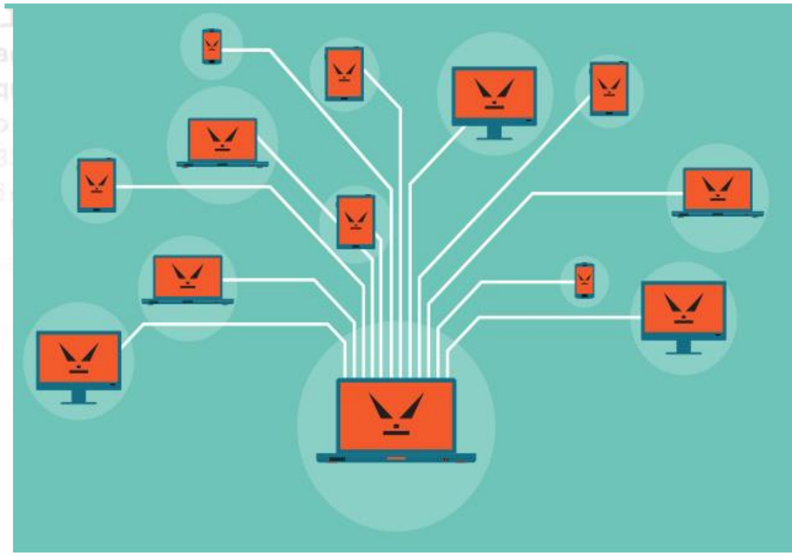


● Botnets

Botnets are large networks of PCs infected with a specific kind of malware. Learn more on how to protect your devices from this type of malware.

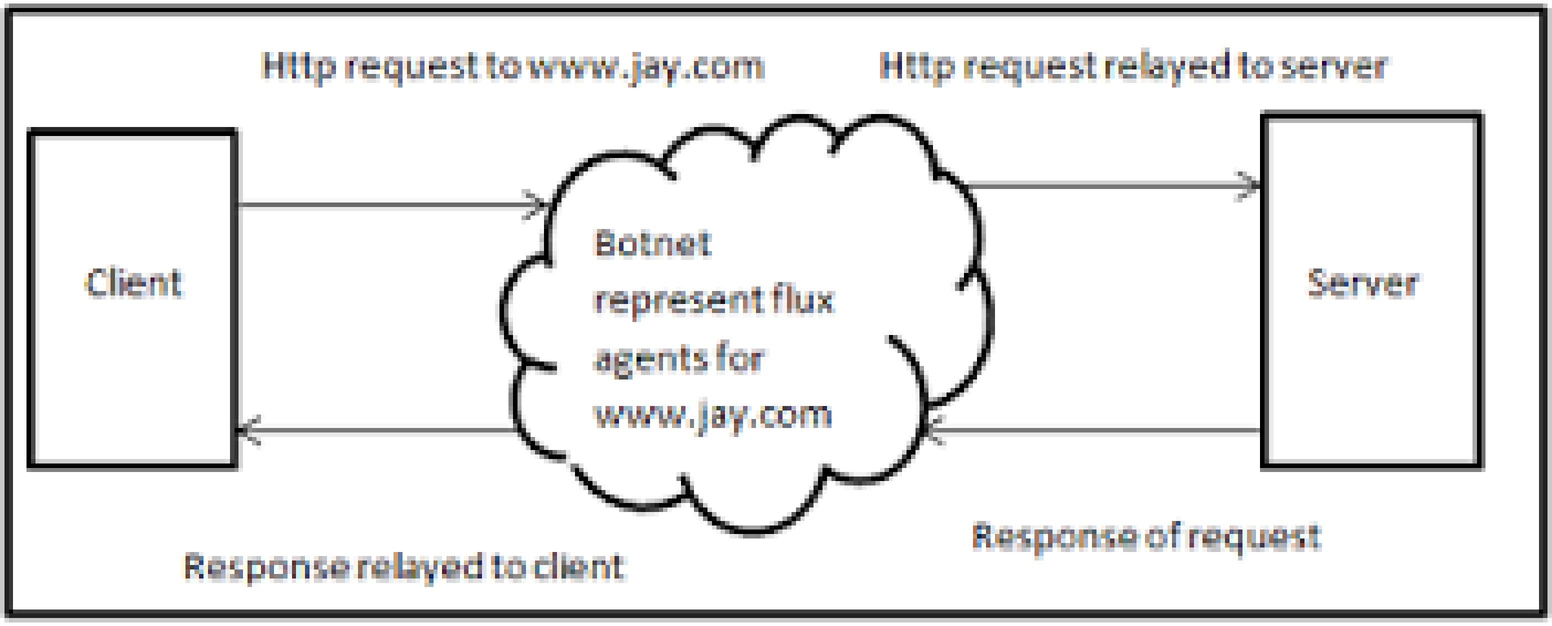
Botnet is the generic name given to any collection of compromised PCs controlled by an attacker remotely — think "virtual robot army." The individual PCs that are part of a botnet are known as "bots" or "zombies," and their owners may not even know they're being used.





● Fast-Flux

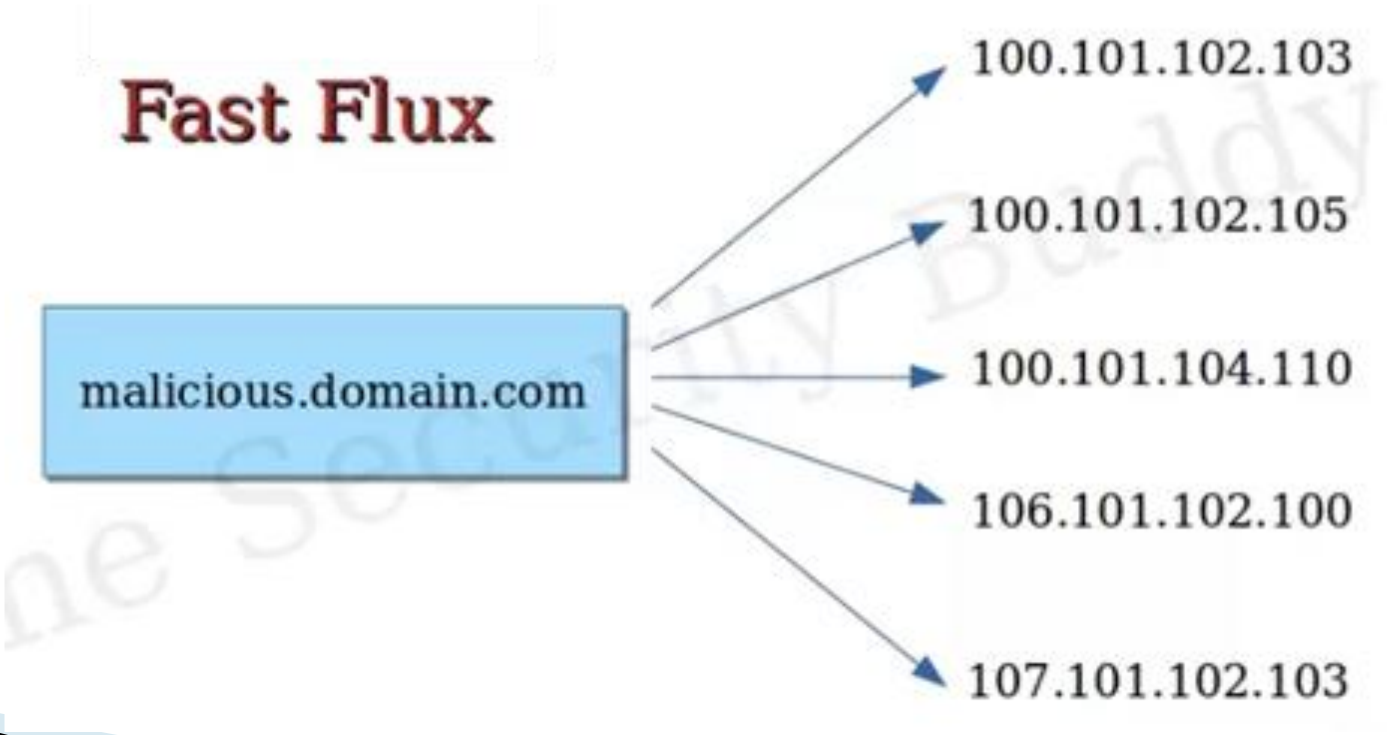
Fast flux is a DNS-based cyber evasion technique where malicious domains rapidly switch between numerous IP addresses (often hundreds or thousands) hosted on compromised computers (botnet nodes), making phishing sites, malware delivery, and command-and-control servers extremely difficult to block, track, and take down by hiding the true origin of attacks. It turns a malicious domain into a "moving target," allowing attackers to maintain high availability for their illegal services and evade security measures.


Http request to www.jay.com
Http request relayed to server
Client
Botnet represent flux agents for www.jay.com
Server
Response of request
Response relayed to client

Fast Flux &Fast Flux Detection

Advanced Fast-Flux

Fast flux is a DNS technique used by botnets to phishing and malware delivery sites behind an ever-changing network of compromised hosts acting as proxies. It can also be referred to as peer-to-peer networking, distributed command and control, web-based load, and balancing proxy redirection used to make malware networks more resistant to delivery and countermeasures.



UNIT–III

UNIT–III

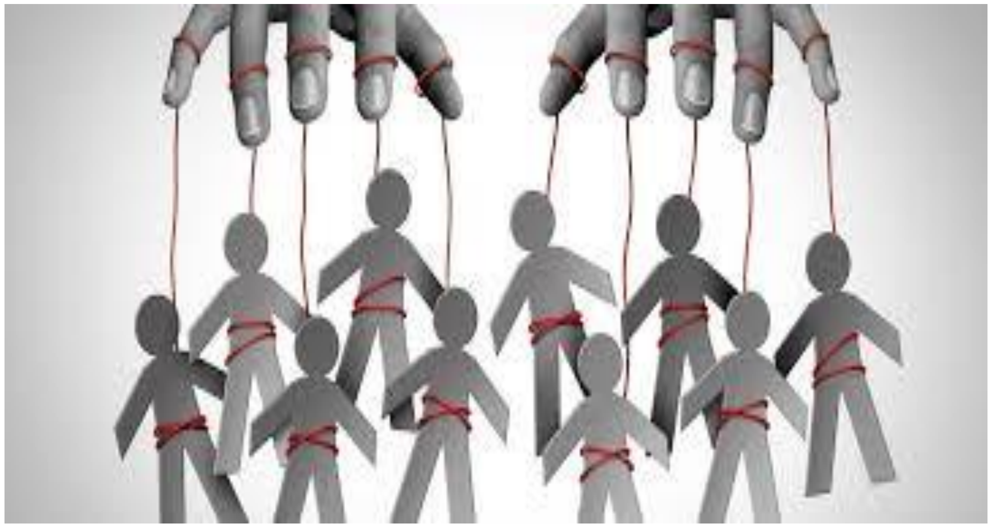
Exploitation:

- Techniques to Gain a Foothold,
- Misdirection- Shell code,
- Integer Over flow Vulnerabilities,
- Stack-Based Buffer Overflows,
- Format String Vulnerabilities,
- SQL Injection,
- Malicious PDF Files,
- Race Conditions,
- Web Exploit Tools,
- DoS Conditions, Brute Force and Dictionary Attacks,
- Re-connaissance, and
- Disruption Methods-Cross-Site Scripting(XSS),
- Social Engineering,
- War Xing,
- DNS Amplification Attacks..

Exploitation:

Exploitation means treating someone unfairly to benefit yourself, often by using their work, ideas, or vulnerability for profit or advantage, like exploiting workers or children; it also means the productive use of resources (like land/minerals), but most commonly carries a negative connotation of unfair manipulation and abuse of power for selfish gain.

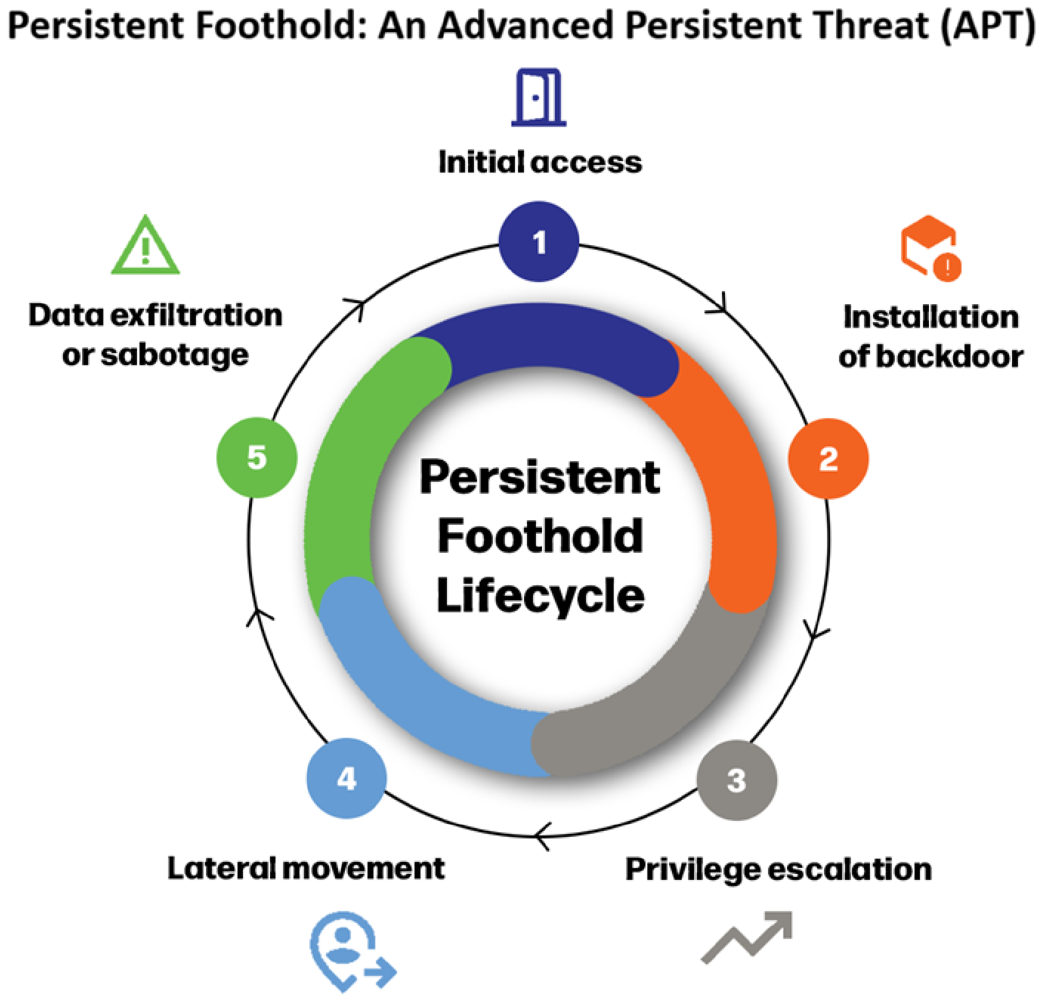
Exploitation - Safer Cornwall



Criminal exploitation - Smart Thinking

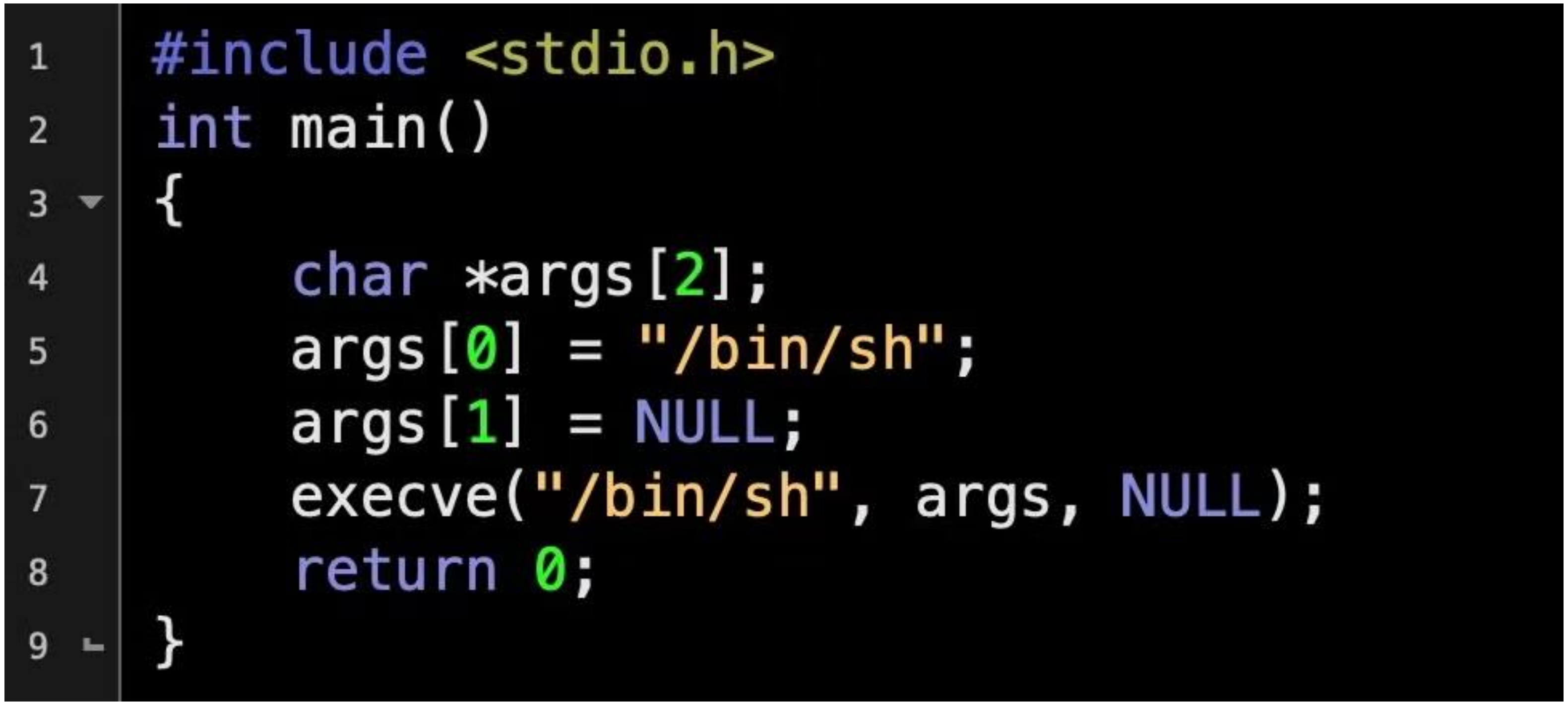
• Techniques to Gain a Foothold

The techniques used to gain the initial foothold include target attacks such as social engineering schemas, phishing and spear-phishing and exploitation.


Persistent Foothold: An Advanced Persistent Threat (APT)
Initial access
1
Installation of backdoor
2
Privilege escalation
3
Lateral movement
4
Data exfiltration or sabotage
5
Persistent Foothold Lifecycle

- **Misdirection- Shell code,**

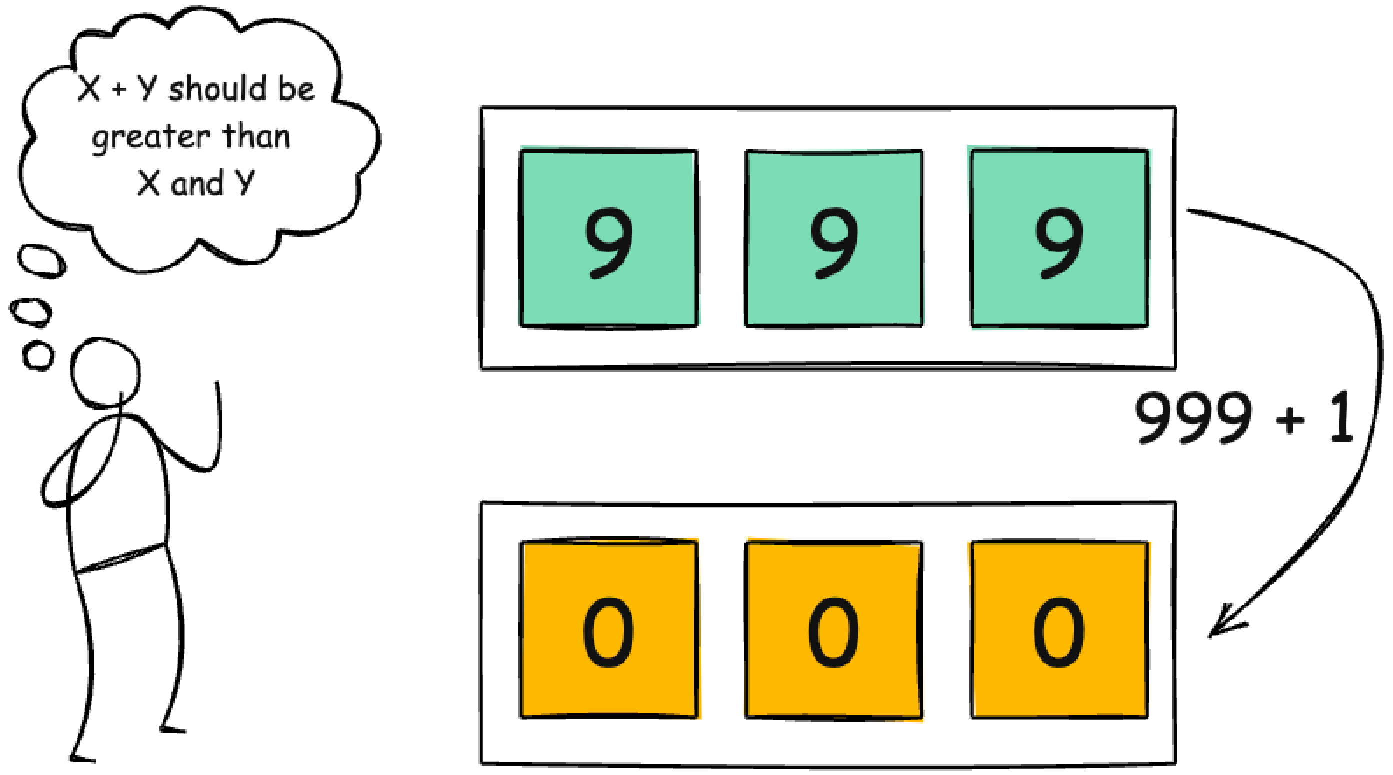
The term "misdirection" in the context of shellcode refers to **evasion and obfuscation techniques** used by malware to hide its true purpose and execution flow from security analysts and detection systems. ,



```c
#include <stdio.h>
int main()
{
      char *args[2];
      args[0] = "/bin/sh";
      args[1] = NULL;
      execve("/bin/sh", args, NULL);
      return 0;
}
```

Integer Over flow Vulnerabilities

- An **integer overflow vulnerability** occurs when a program attempts to store a numeric value in an integer variable that exceeds the maximum value the variable can hold, causing it to wrap around to an unexpected value, which can then be exploited by an attacker. These vulnerabilities often lead to other critical security flaws like buffer overflows or denial-of-service conditions.



Stack-Based Buffer Overflows

stack-based buffer overflow occurs when a program writes more data to a buffer located on the call stack than it can hold, overwriting adjacent memory locations, which can lead to system crashes or arbitrary code execution by an attacker. This is one of the oldest and most well-known software vulnerabilities.

Memory Layout of C program.

Text: Contains program code to be executed.

Data: Contains global information for program.

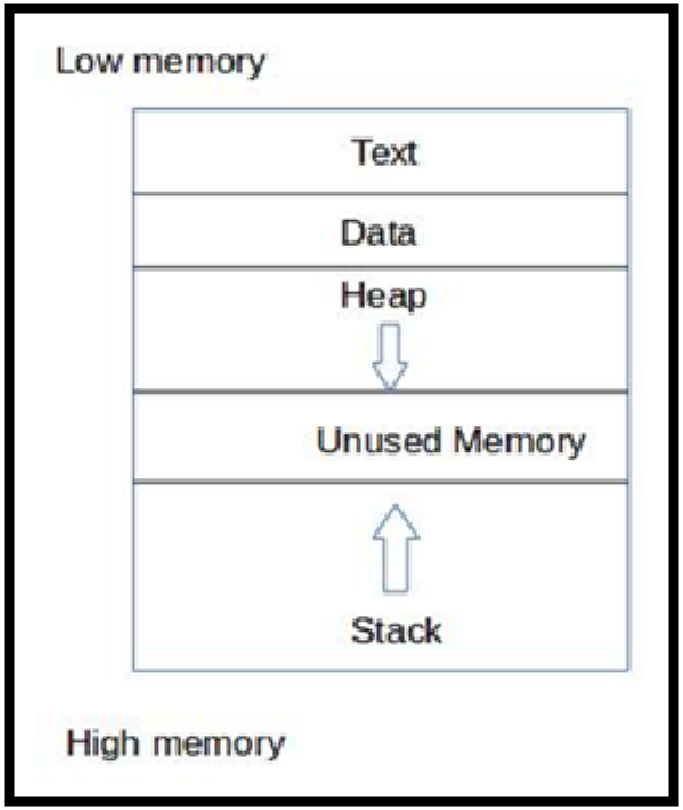
Stack: Contains function parameters,

Heap: Holds all the dynamically allocated memory.

Now its time to look into intel based CPU registers.

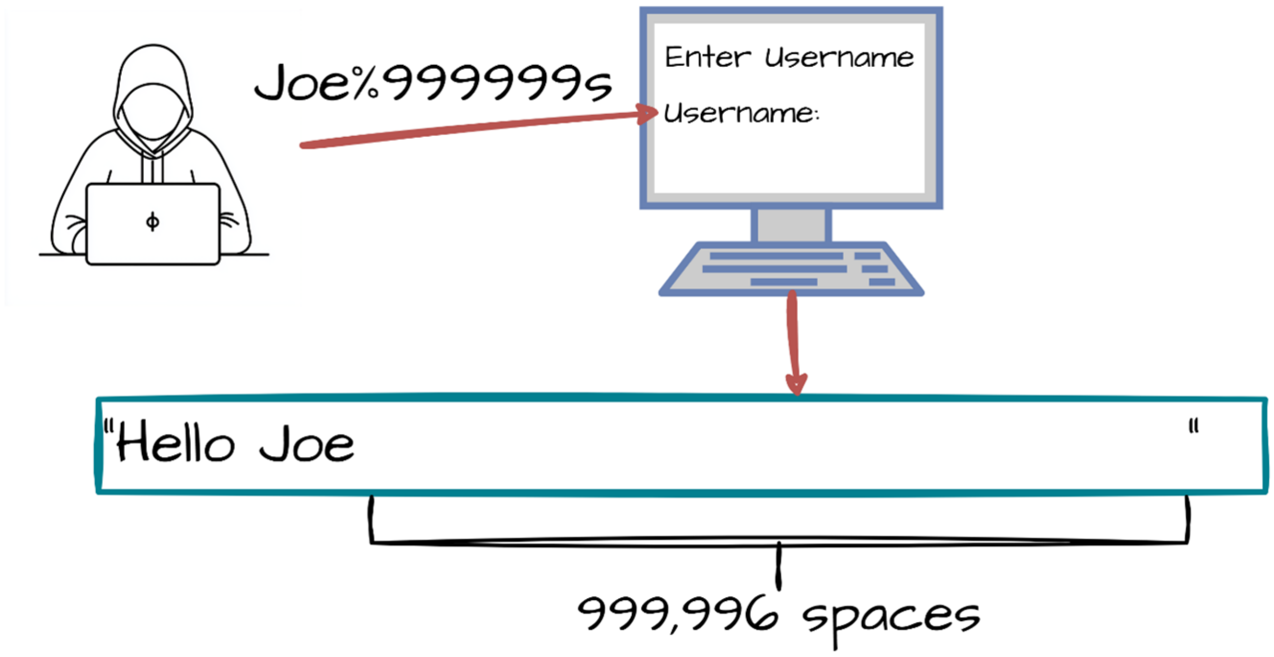
- **EIP** instruction pointer
- **ESP** stack pointer
- **EBP** base pointer
- **ESI** source index
- **EDI** destination index

- **EAX** accumulator
- **EBX** base
- **ECX** counter
- **EDX** data



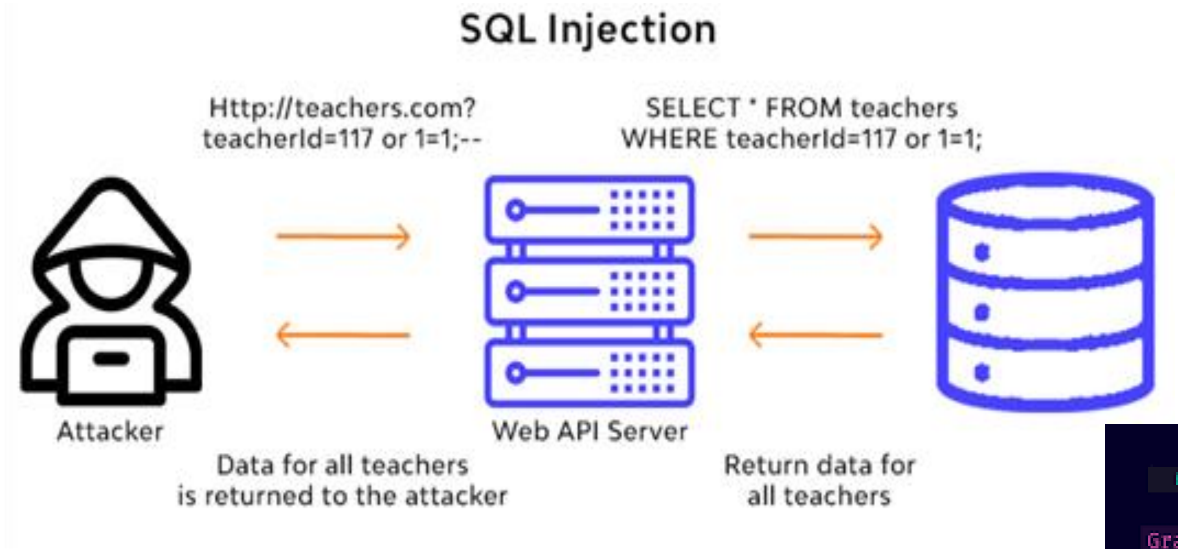
Format String Vulnerabilities

Format string vulnerabilities occur when user-supplied input is directly used as a format string in functions like C/C++'s `printf()`, allowing attackers to inject format specifiers (e.g., `%s`, `%x`, `%n`) to read/write arbitrary memory, crash the program (Denial of Service), or even execute malicious code by manipulating the application's stack and memory. These attacks exploit the function's inability to verify the number of arguments, leading to information disclosure or memory corruption, especially with the %n specifier, which writes data.



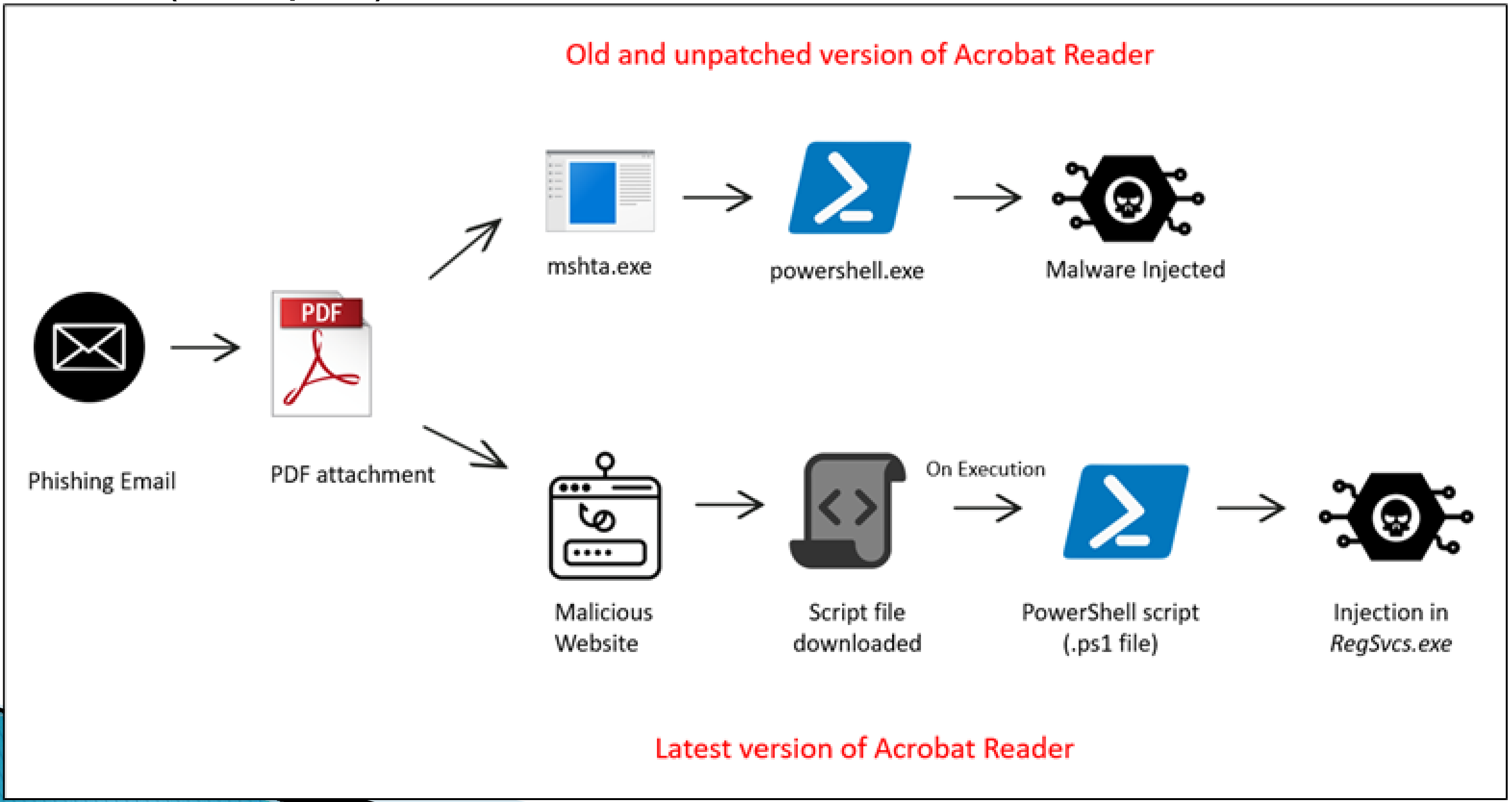
SQL Injection

- **SQL injection (SQLi)** is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It typically occurs when a developer insecurely incorporates user-supplied input directly into an SQL statement, allowing the input to be interpreted as code rather than data.


SQL Injection
Http://teachers.com?teacherId=117 or 1=1;--
SELECT * FROM teachers WHERE teacherId=117 or 1=1;
Attacker
Web API Server
Data for all teachers is returned to the attacker
Return data for all teachers

• Malicious PDF Files

o check if a PDF is corrupted, first try opening it in different viewers (like Adobe Reader, browser) for errors; then use dedicated tools like Adobe Acrobat Pro's Preflight or online validators (PDF Association) for detailed analysis, or command-line tools like qpdf for batch checks; signs of corruption include crashes, garbled text, or missing content, while simple checks involve opening in a text editor (Notepad) to see if it starts with `%PDF-'.

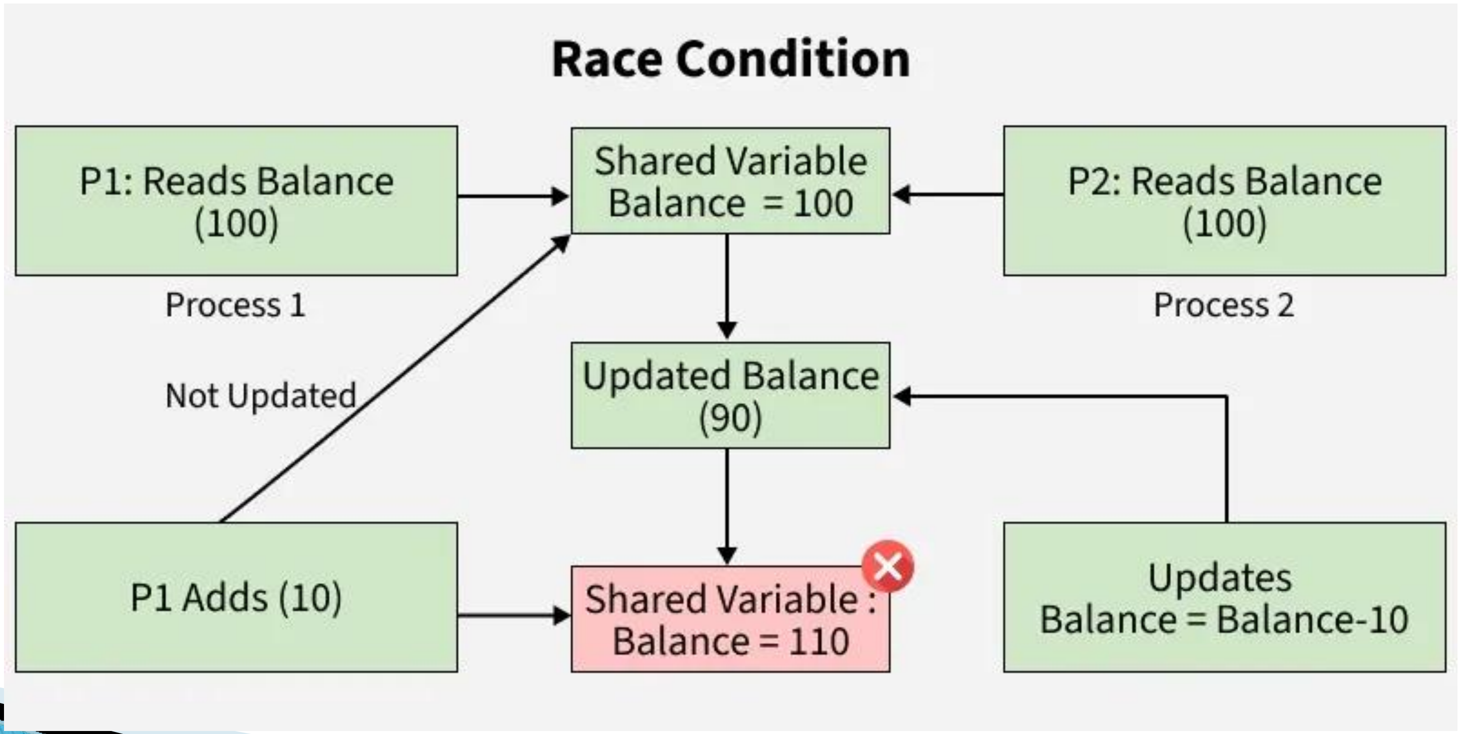


Race Conditions

- A race condition is a software/system flaw where behavior depends on unpredictable timing, usually when multiple threads/processes access shared resources concurrently without coordination, leading to data corruption, crashes, or security vulnerabilities like privilege escalation

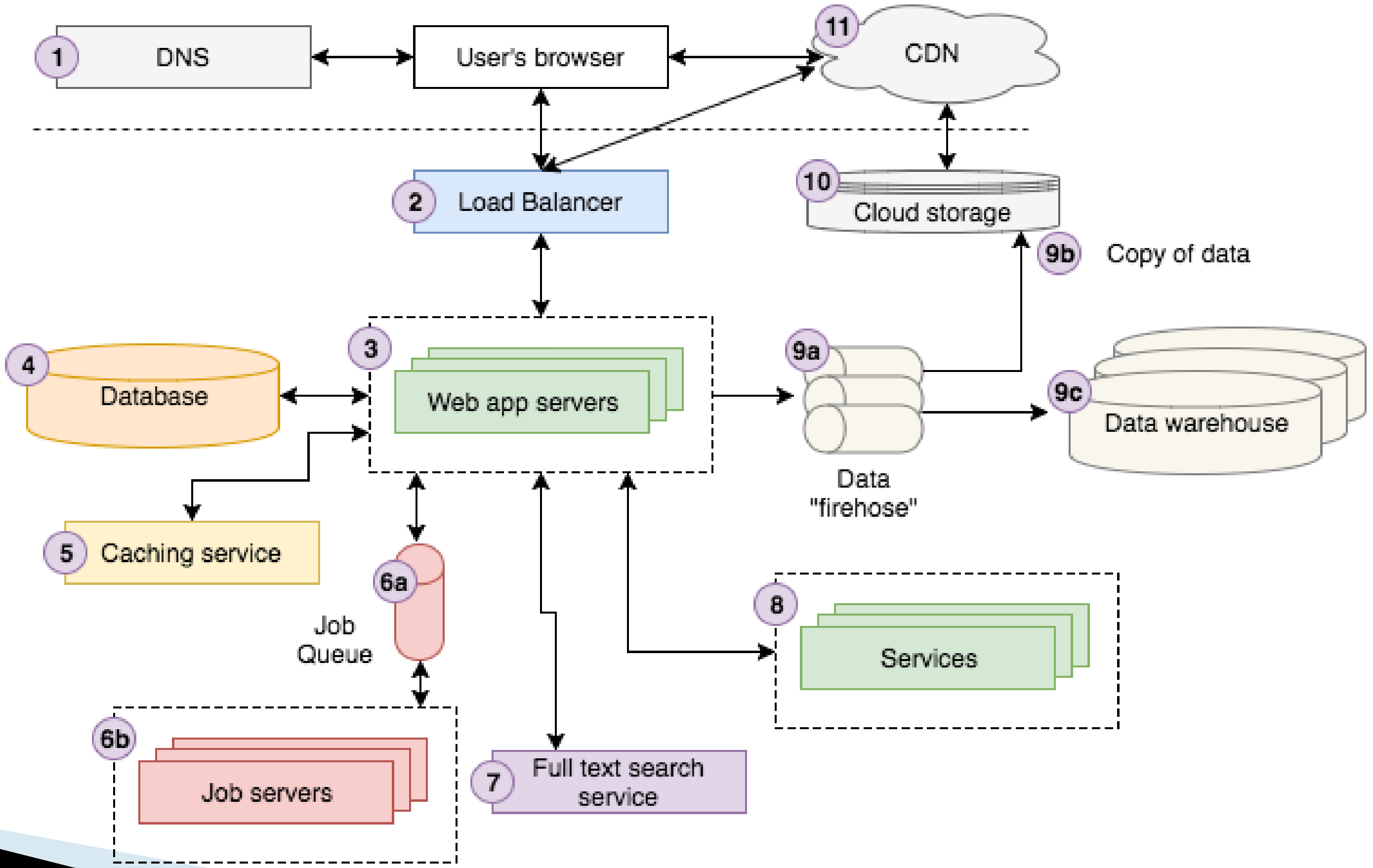
Common Types & Examples

- **Read-Modify-Write.**
- **Time-of-Check to Time-of-Use (TOCTOU)**
- **Data Race**



Web Exploit Tools

- Web exploitation tools are used by cybersecurity professionals to identify, assess, and leverage vulnerabilities in web applications and associated infrastructure as part of authorized security testing (penetration testing). These tools cover various aspects, from reconnaissance to post-exploitation.



DoS Conditions, Brute Force and Dictionary Attacks

DoS (Denial of Service) aims to crash systems by overwhelming them, while Brute Force and Dictionary attacks target user credentials, with **Brute Force** trying *every* possibility (slow, exhaustive) and **Dictionary Attacks** using wordlists (faster, relies on common words). Key defenses involve strong passwords, Multi-Factor Authentication (MFA), account lockouts, and traffic monitoring to stop these credential-guessing methods, and firewalls/DDoS mitigation for DoS.

1. Denial of Service (DoS) Attacks

Goal: Make a service (website, server) unavailable to legitimate users.

How it Works: Floods the target with overwhelming traffic or requests, (bandwidth, CPU).

Types: DoS (single source) vs. DDoS (Distributed DoS, multiple sources/botnets).

Example: Sending massive amounts of data to a server until it crashes.

2. Brute Force Attacks

Goal: Guess passwords or encryption keys by trying all possible combinations.

How it Works: Exhaustively tests every single character combination (e.g., 'a', 'b', 'aa', 'ab', 'ac').

Characteristics: Very slow, computationally intensive,

Mitigation: Account lockout policies, strong password requirements, MFA.

3. Dictionary Attacks

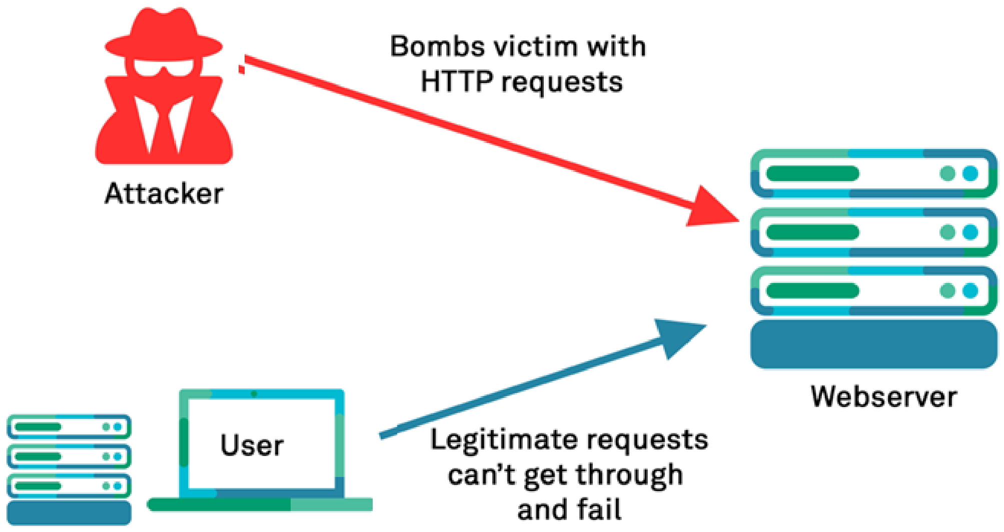
Goal: Guess passwords using pre-compiled lists of common words, phrases, or patterns.

How it Works: Tries words like "password," "123456," or variations of personal info (names, birthdays)

Characteristics: Much faster than brute force but limited by the dictionary's quality; weak passwords.

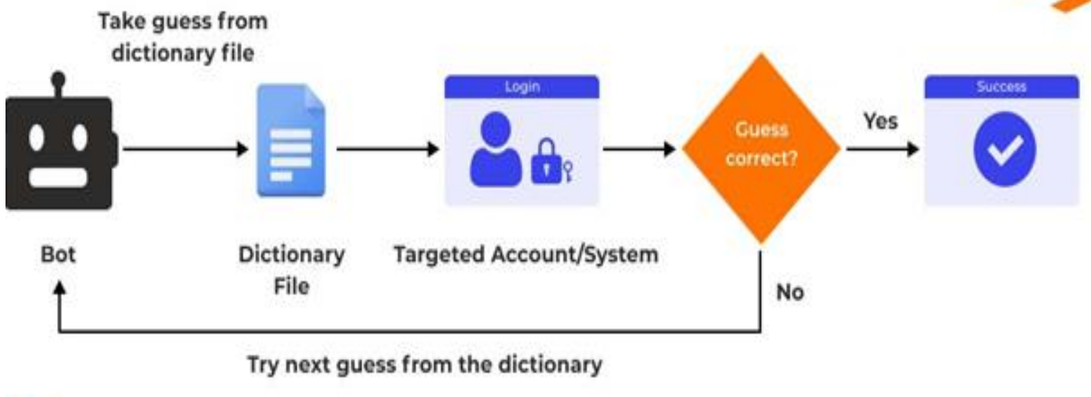
Mitigation: Similar to brute force (strong passwords, MFA, account lockouts).

DoS Conditions, Brute Force and Dictionary Attacks



Attacker
Bombs victim with HTTP requests
Webserver
User
Legitimate requests can't get through and fail

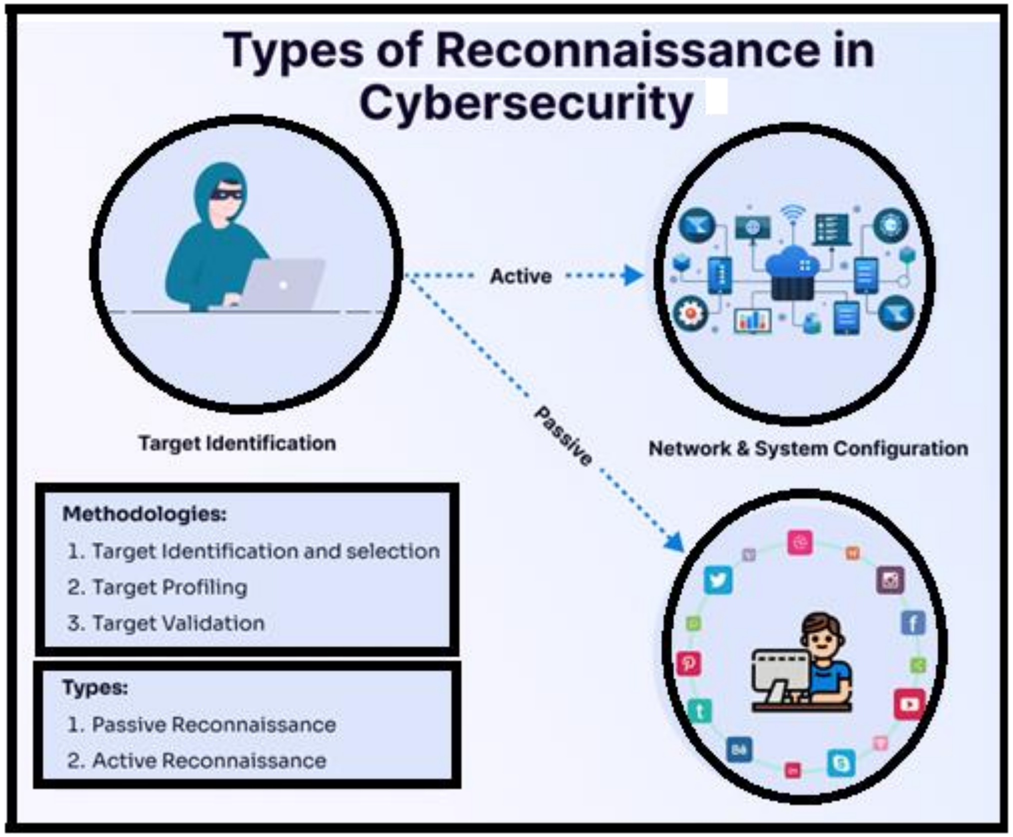
Brute Force Method/Attack



Dictionary Attack

Re-connaissance

- Reconnaissance is a mission to obtain information by visual observation or other detection methods, about the activities and resources of an enemy or potential enemy, or about the meteorologic, hydrographic, or geographic characteristics of a particular area.


Types of Reconnaissance in Cybersecurity
Active
Passive
Target Identification
Network & System Configuration

Methodologies:
1. Target Identification and selection
2. Target Profiling
3. Target Validation

Types:
1. Passive Reconnaissance
2. Active Reconnaissance

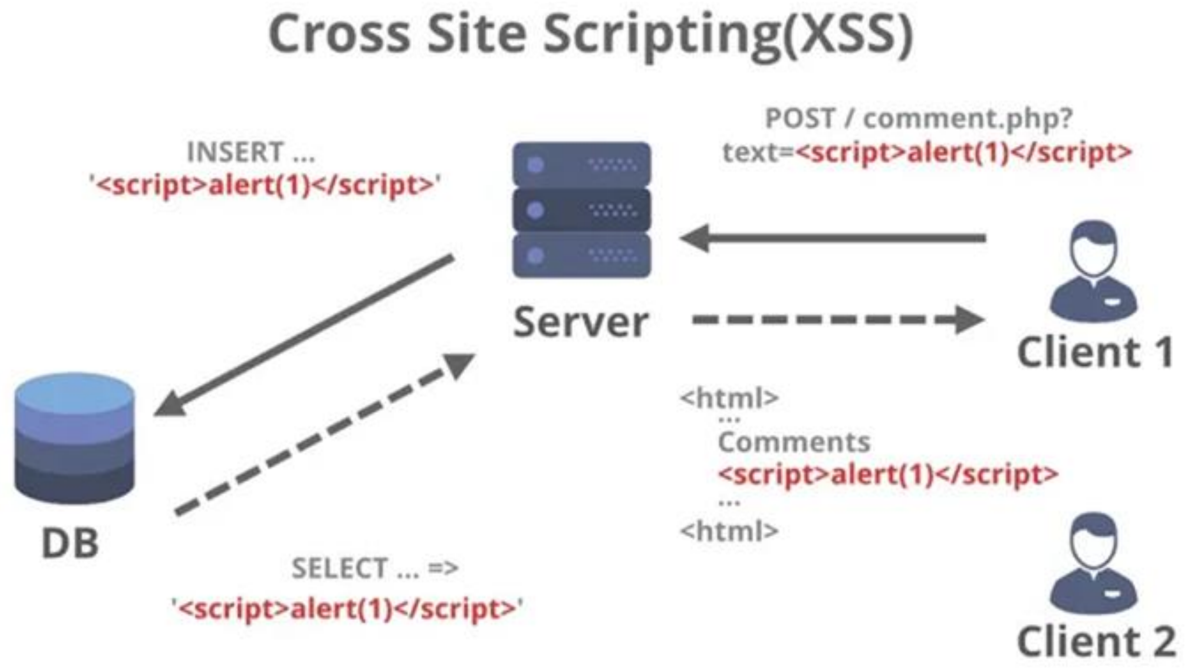
• Disruption Methods-Cross-Site Scripting(XSS)

Cross-Site Scripting (XSS) is a web security vulnerability where attackers inject malicious client-side scripts (typically JavaScript) into legitimate, trusted websites. The user's browser, trusting the website as a legitimate source, then executes the malicious code, allowing the attacker to compromise the user's interaction with the site.

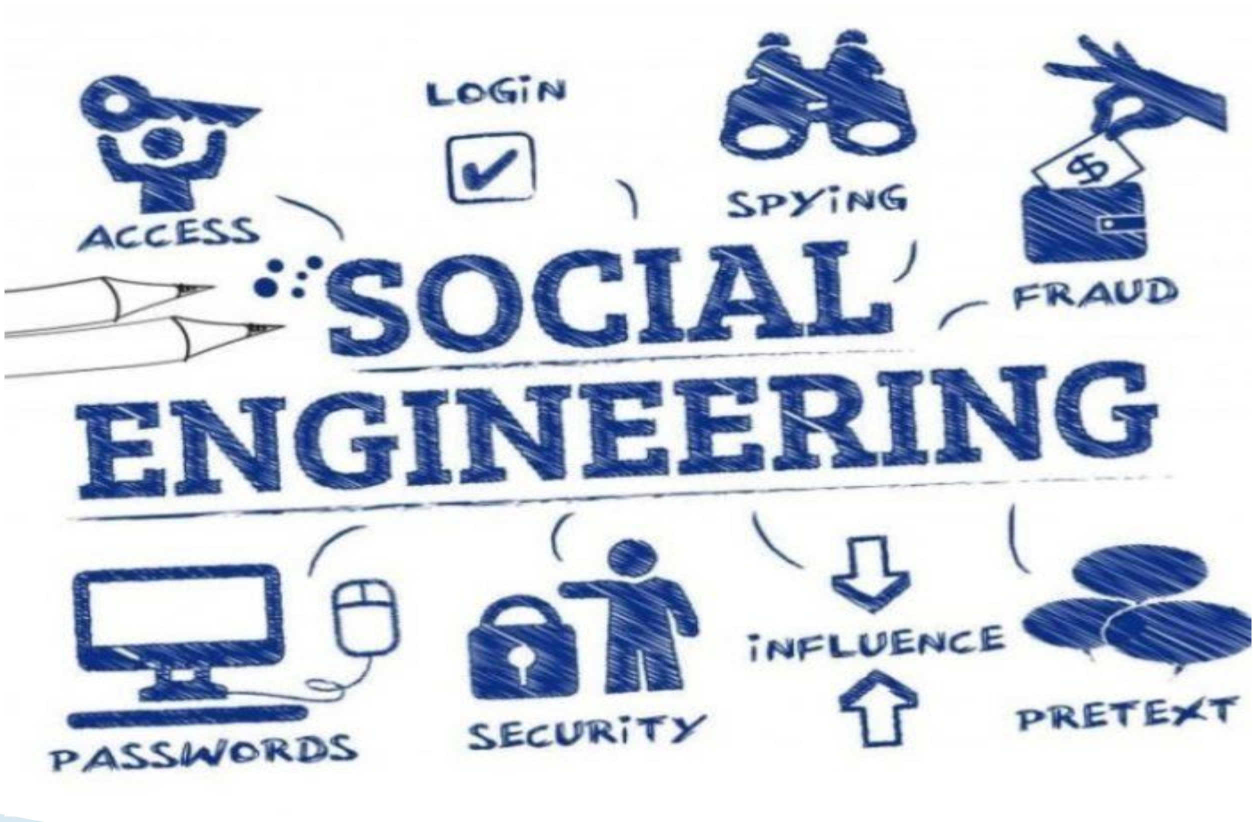
There are **three primary types of XSS attacks**:

1). Reflected XSS (Non-Persistent) **2). Stored XSS (Persistent)**

3). DOM-based XSS


Cross Site Scripting(XSS)
INSERT ...
'<script>alert(1)</script>'
POST / comment.php?
text=<script>alert(1)</script>
Server
Client 1
DB
<html>
...
Comments
<script>alert(1)</script>
...
<html>
SELECT ... =>
'<script>alert(1)</script>'
Client 2

- **Social Engineering,**
- Social engineering is a manipulation technique that exploits human psychology and error, rather than technical vulnerabilities, to trick individuals into divulging sensitive information, granting access to systems, or performing actions that compromise security.

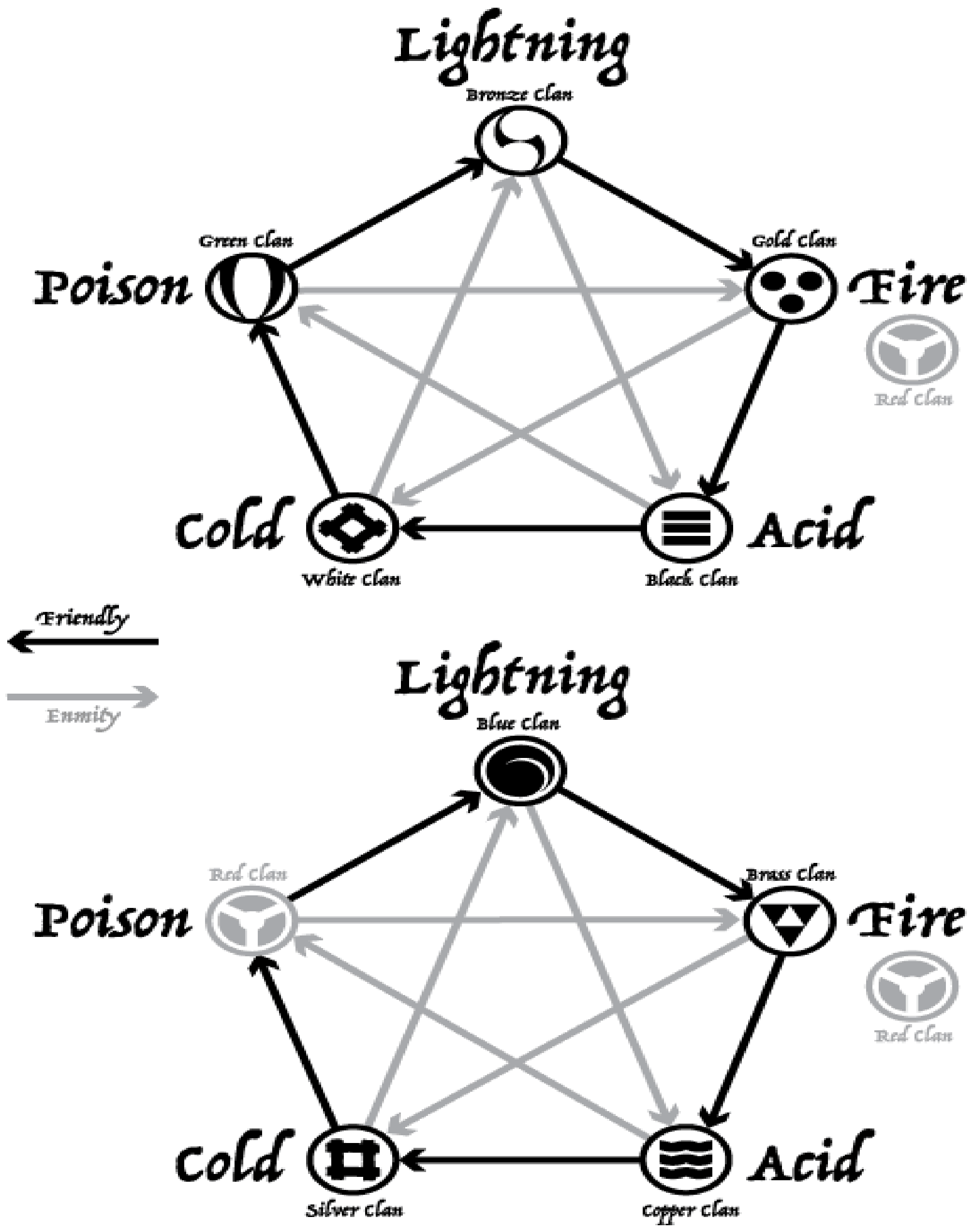


- **War Xing**

What is WarXing in cyber security? WarXing in cybersecurity involves finding and exploiting wireless network vulnerabilities. It comes from "WarDriving," the act of driving with a laptop or mobile device to find open or insecure Wi-Fi networks. WarXing searches for Bluetooth, RFID, and NFC networks as well as Wi-Fi networks.

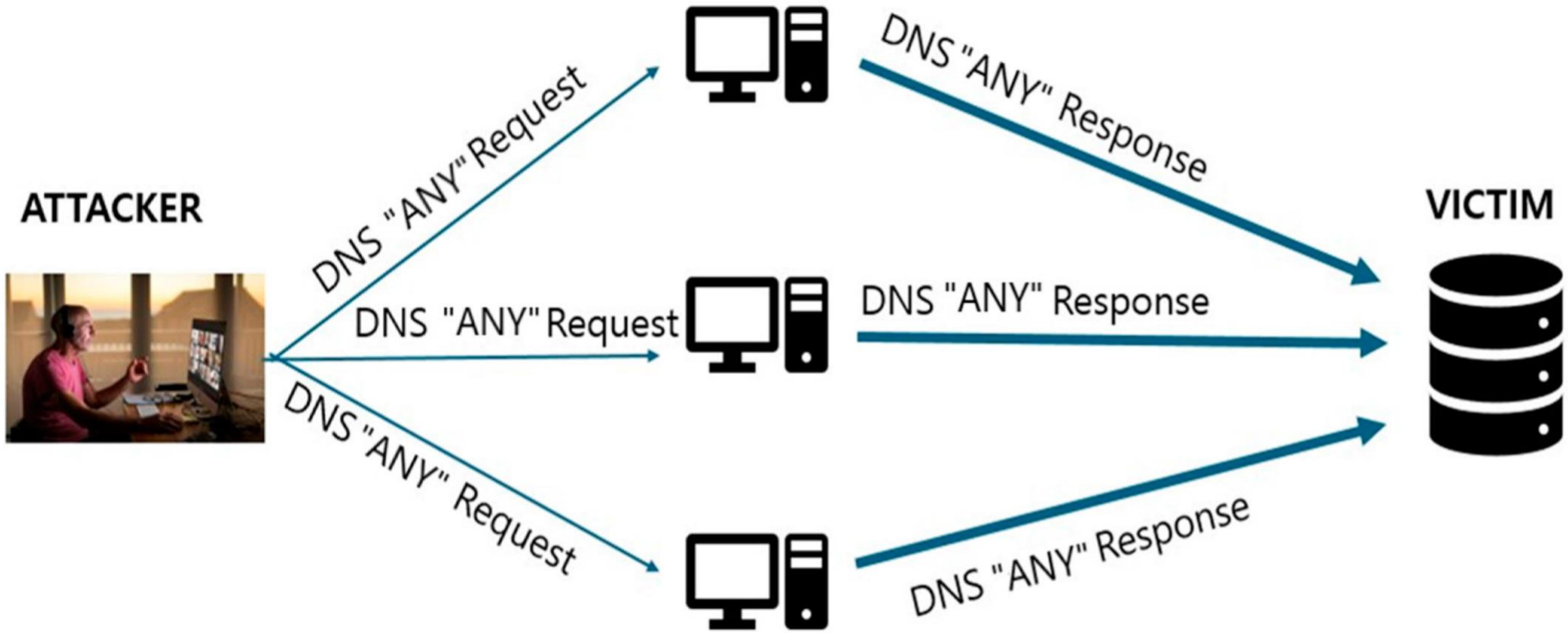
Here you will get some variations of WarXing:
Warchalking: The name itself is a practice that is a reference to a common method of warming.
Community: Here they breadth regarding Wi-Fi arrangement and the sidewalk with a piece of chalk.
Warcycling: Users can search Wi-Fi networks by doing the cycling.
War dialing: In this practice, include phone numbers so that the user can find the modems.
Wardriving: In this process, you can seek Wi-Fi networks in the car.
Warflying: In this, by using an aircraft or drone, the PDA user needs to complete his work.

- **War Xing**


Lightning
Bronze clan
Green clan
Poison
Gold clan
Fire
Red clan
Cold
White clan
Acid
Black clan
Friendly
Enmity
Lightning
Blue clan
Red clan
Poison
Brass clan
Fire
Red clan
Cold
Silver clan
Acid
Copper clan

• DNS Amplification Attacks..

- A DNS amplification attack is a DDoS attack that uses open DNS servers to flood a target with massive amounts of traffic, making its services unavailable. Attackers send small DNS requests to vulnerable servers (resolvers) with the victim's spoofed IP address as the source, tricking the server into sending much larger DNS responses directly to the victim, amplifying the attack's effect and hiding the attacker's identity.



UNIT-IV

UNIT–IV

Malicious Code:

- Self-Replicating Malicious Code- Worms,
- Viruses.
- Evading Detection and Elevating Privileges- Obfuscation,
- Virtual Machine Obfuscation,
- Persistent Software Techniques,
- Root kits,
- Spyware,
- Attacks against Privileged User Accounts and
- Escalation of Privileges,
- Token Kidnapping,
- Virtual Machine Detection.
- Stealing Information and Exploitation- Form Grabbing,
- Man-in-the- Middle Attacks,
- DLL Injection,
- Browser Helper Objects.

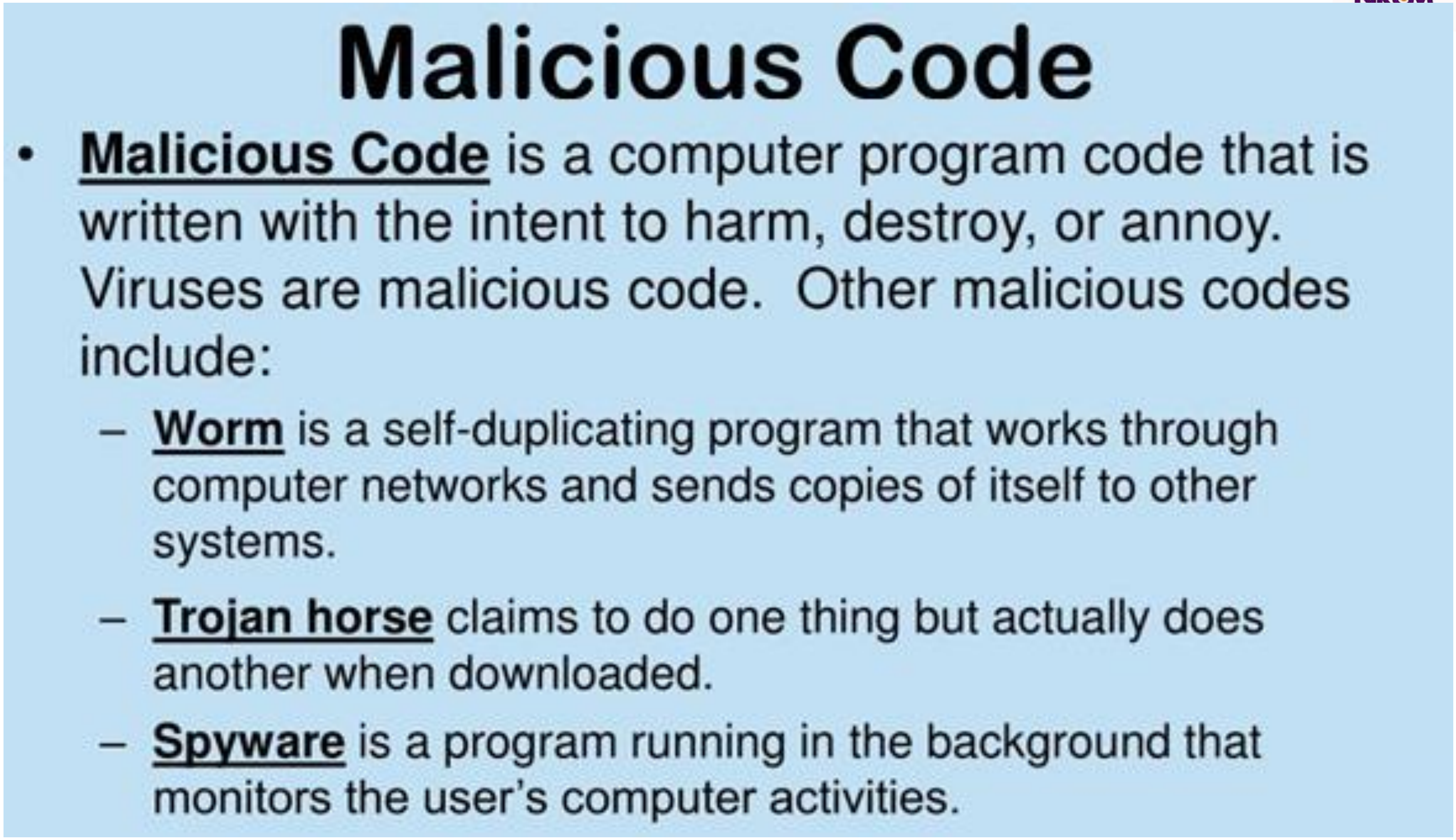
Malicious Code:

Malicious code, often referred to as malware, is any software or script designed to harm, exploit, or com promise the functionality of computer systems, netw orks, or devices.

Definition

<u>Malicious code encompasses a variety of harmful progra ms, scripts, or software that disrupt normal operations, st eal sensitive information, or gain unauthorized access to systems. It is intentionally created to cause damage, unli ke accidental bugs or software flaws</u>

Malicious Code:



Malicious Code

- **Malicious Code** is a computer program code that is written with the intent to harm, destroy, or annoy. Viruses are malicious code. Other malicious codes include:
  - **Worm** is a self-duplicating program that works through computer networks and sends copies of itself to other systems.
  - **Trojan horse** claims to do one thing but actually does another when downloaded.
  - **Spyware** is a program running in the background that monitors the user's computer activities.

Malicious Code:

• Self-Replicating Malicious Code-Worms, Viruses.

Self-Replicating Malicious Code-Worms, Viruses

Selfreplicating malicious code, known as worms and viruses, are two distinct forms of malware that can cause significant damage to systems and networks.

Worms are selfreplicating pieces of malware that spread across networks without requiring user interaction, exploiting vulnerabilities to infect more systems. They can cause widespread disruptions and are often more dangerous than viruses due to their speed and reach.

Viruses, on the other hand, are selfreplicating executable programs that attach to other files or applications, causing damage or altering data. They require user action to propagate and are typically less dangerous than worms. Understanding the differences and characteristics of these malicious codes is crucial for effective cybersecurity measures.

• Viruses

A computer virus is a type of malicious software (malware) designed to attach itself to a legitimate file or program in order to spread from one system to another. Much like a biological virus, it needs a host to survive and replicate.

Types of Computer Viruses:

- **File Infector Virus:** Attaches to executable files (.exe or .dll).
- **Macro Virus:** Infects documents like Word or Excel by exploiting macros.
- **Boot Sector Virus:** Targets the boot sector of hard drives or USBs.
- **Polymorphic Virus:** Changes its code each time it spreads, making it hard to detect.
- **Resident Virus**: Hides in the system memory and infects files as they are opened.

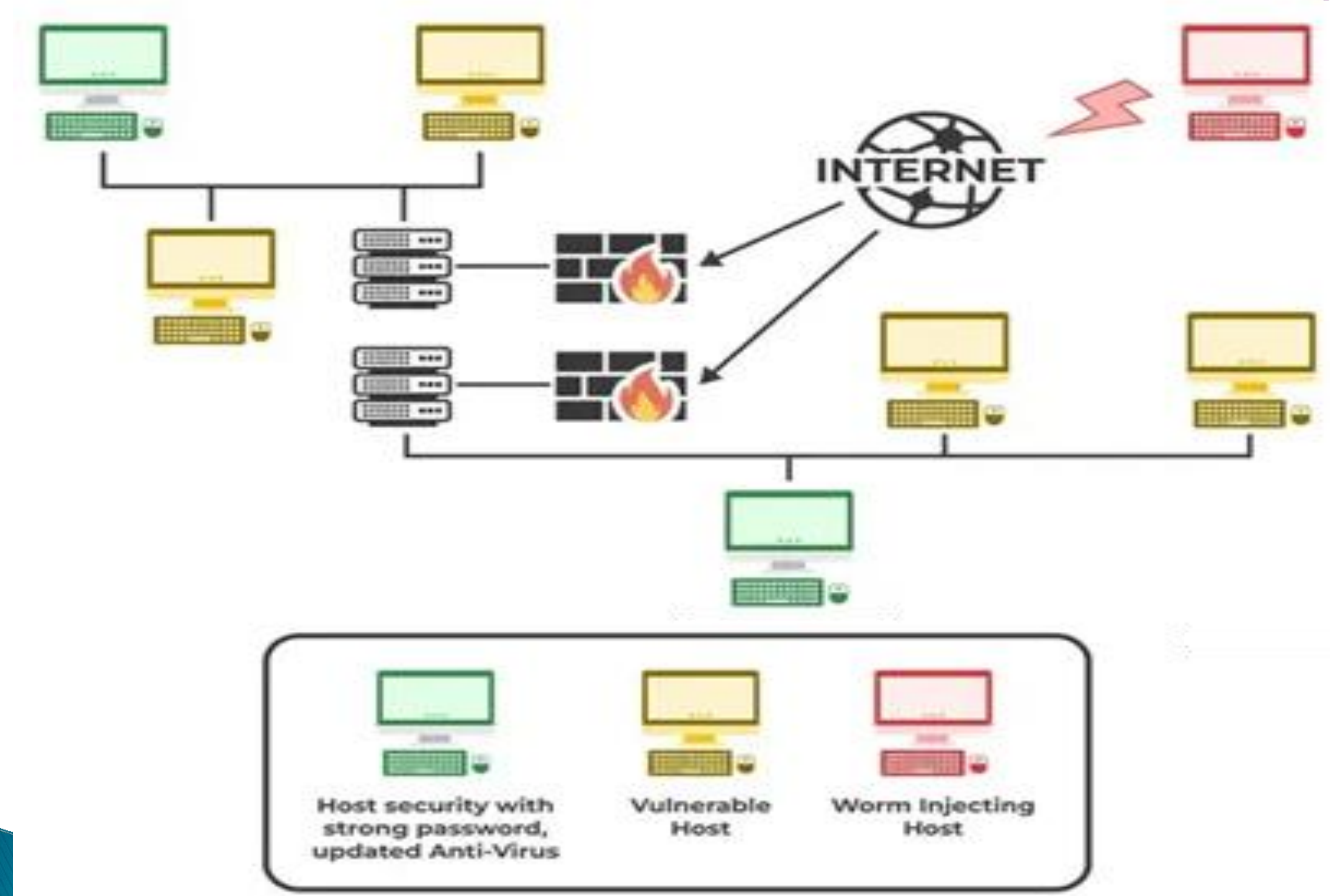
Worms

Worms are similar to a virus but it does not modify the program. A computer worm is a type of malware that can replicate itself and spread across computers and networks without needing a host file or any user interaction. It is self-contained, meaning it doesn’t need to attach to other programs or files to function. Worms can be controlled by remote. The main objective of worms is to eat the system's resources .

Difference Between Worms and Viruses

Basis of Comparison	Worms	Viruses
Definition	A Worm is a form of malware that replicates itself and can spread to different computers via a Network.	A Virus is a malicious executable code attached to another executable file that can be harmless or can modify or delete data.
Objective	The main objective of worms is to eat the system's resources. It consumes system resources such as memory and bandwidth and makes the system slow in speed to such an extent that it stops responding.	The main objective of viruses is to modify the information.
Host	It doesn't need a host to replicate from one computer to another.	It requires a host is needed for spreading.
Harmful	It is less harmful as compared.	It is more harmful.
Detection and Protection	They can be detected and removed by the Antivirus and firewall.	Antivirus software is used for protection against viruses.
Controlled by	They can be controlled by remote.	They can't be controlled by remote.
Execution	They are executed via weaknesses in the system.	They are executed via executable files.
Comes from	Worms generally come from the downloaded files or through a network connection.	They generally come from shared or downloaded files.
Symptoms	1. Hampering computer performance by slowing down it 2. Automatic opening and running of programs 3. Sending of emails without your knowledge	1. Pop-up windows linking to malicious websites 2. Hampering computer performance by slowing down it 3. After booting, starting of unknown programs.
Examples	Examples of worms include Morris worm, storm worm, etc.	Examples of viruses include Creeper, Blaster, Slammer, etc.
Interface	It does not need human action to replicate.	It needs human action to replicate.
Speed	Its spreading speed is faster.	Its spreading speed is slower as compared to worms.

Difference Between Worms and Viruses



• Evading Detection and Elevating Privileges- Obfuscation,

Obfuscation techniques are used to evade detection in cybersecurity. Here are some k ey methods:

String Obfuscation: Breaking recognizable text or commands into fragmented parts to make them unreadable.

Encryption: Scrambling the payload to make it unreadable without a specific key.

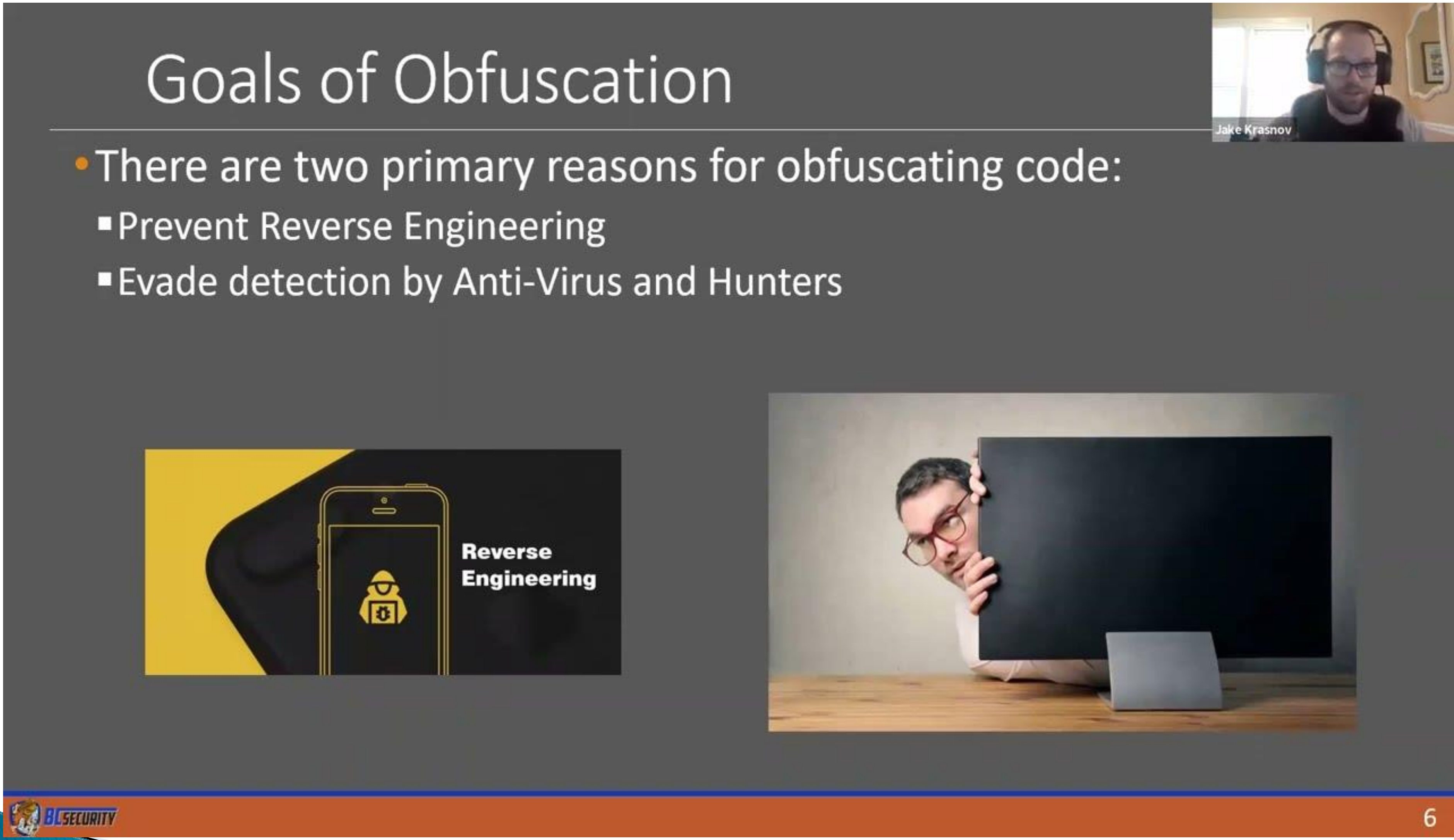
Compression: Reducing file size and disguising malicious code, making detection harde r

Character Substitution: Replacing clear commands or URLs with unusual or encoded c haracters to evade simple scans.

Traffic Obfuscation: Hiding malicious network activity by masking, encrypting, or alterin g data flows to evade detection by security controls.

These techniques are crucial for cybersecurity professionals to develop effective counter measures against advanced malware and evasion tactics.

• Evading Detection and Elevating Privileges- Obfuscation,



Goals of Obfuscation

- There are two primary reasons for obfuscating code:
  - Prevent Reverse Engineering
  - Evade detection by Anti-Virus and Hunters

• Evading Detection and Elevating Privileges- Obfuscation,

Obfuscation techniques are used to evade detection in cybersecurity. Here are some key methods:

understanding Privilege Escalation

Privilege escalation is a cybersecurity tactic where attackers exploit vulnerabilities or bypass security measures to gain elevated access to systems, data, and resources. This can occur in two main forms:

Vertical Privilege Escalation: This occurs when an attacker elevates their access rights from a regular user account to one with higher privileges, such as an administrator or root account. Attackers may exploit unpatched vulnerabilities or misconfigured services to achieve this.

2.Horizontal Privilege Escalation: In this scenario, attackers gain access to accounts with the same privilege level by impersonating another user or using stolen credentials. For example, using stolen session cookies to impersonate another user's account in a web application

• Evading Detection and Elevating Privileges- Obfuscation,

Techniques for Evading Detection
Attackers employ various methods to evade detection while attempting to elevate their p rivileges, including:

-

•Covering Tracks: Deleting logs, masking IP addresses, and using encryption to h ide their activities from security monitoring systems.

•Exploiting System Weaknesses: Taking advantage of software bugs, misconfigur ations, or vulnerabilities to gain unauthorized access without triggering alarms.

•Social Engineering: Using tactics like phishing to trick legitimate users into provi ding their credentials, which can then be used to escalate privileges.

• Virtual Machine Obfuscation

• Virtual Machine Obfuscation is a sophisticated technique used in cybersecurity to protect software systems from attacks.
It involves altering the code to make it difficult for attackers to understand or reverse_engineer.

This is achieved by using a virtual machine as a platform for compl ex and convoluted code execution, which complicates the task of in filtrating systems.

• Persistent Software Techniques

- Persistence mechanisms are techniques used by attackers to maintain lon_term access to a compromised system, even after reboots, software updates, or security interventions.

- These mechanisms ensure that an adversary can regain control of the system without needing to reinfect it.

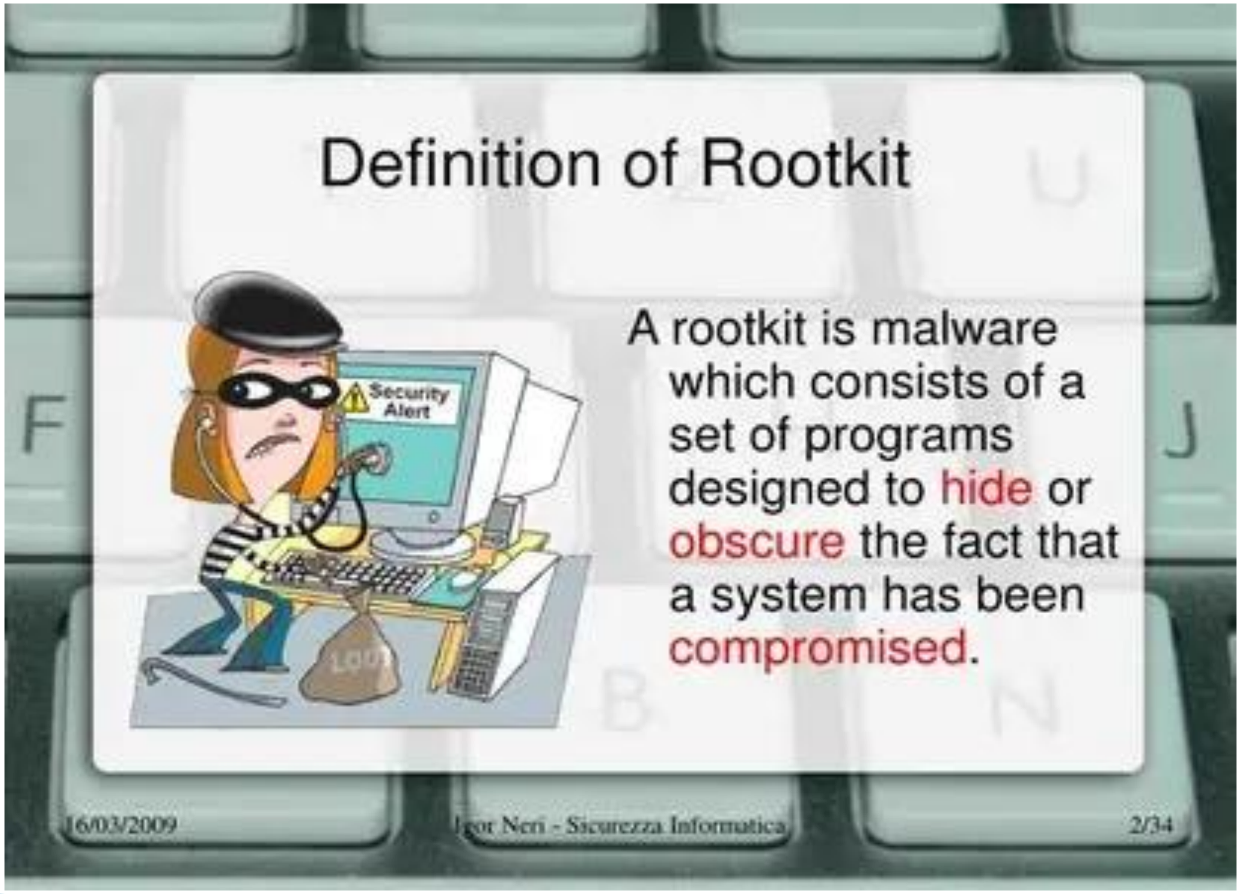
- Attackers often achieve persistence by modifying system configurations, installing rootkits, or leveraging legitimate tools like Windows Task Scheduler, registry modifications, or startup scripts to execute malicious code automatically

• Root kits

Definition, types of rootkit, detecting and removing rootkit, & how to defend against them. A common rootkit definition is a type of malware program that enables cyber criminals to gain access to and infiltrate data from machines without being detected.

Types Of Rootkits:

1. **Firmware rootkits**
2. **Bootloader rootkits**
3. **Memory rootkit**
4. **Application rootkit**
5. **Kernel mode rootkits**



Definition of Rootkit

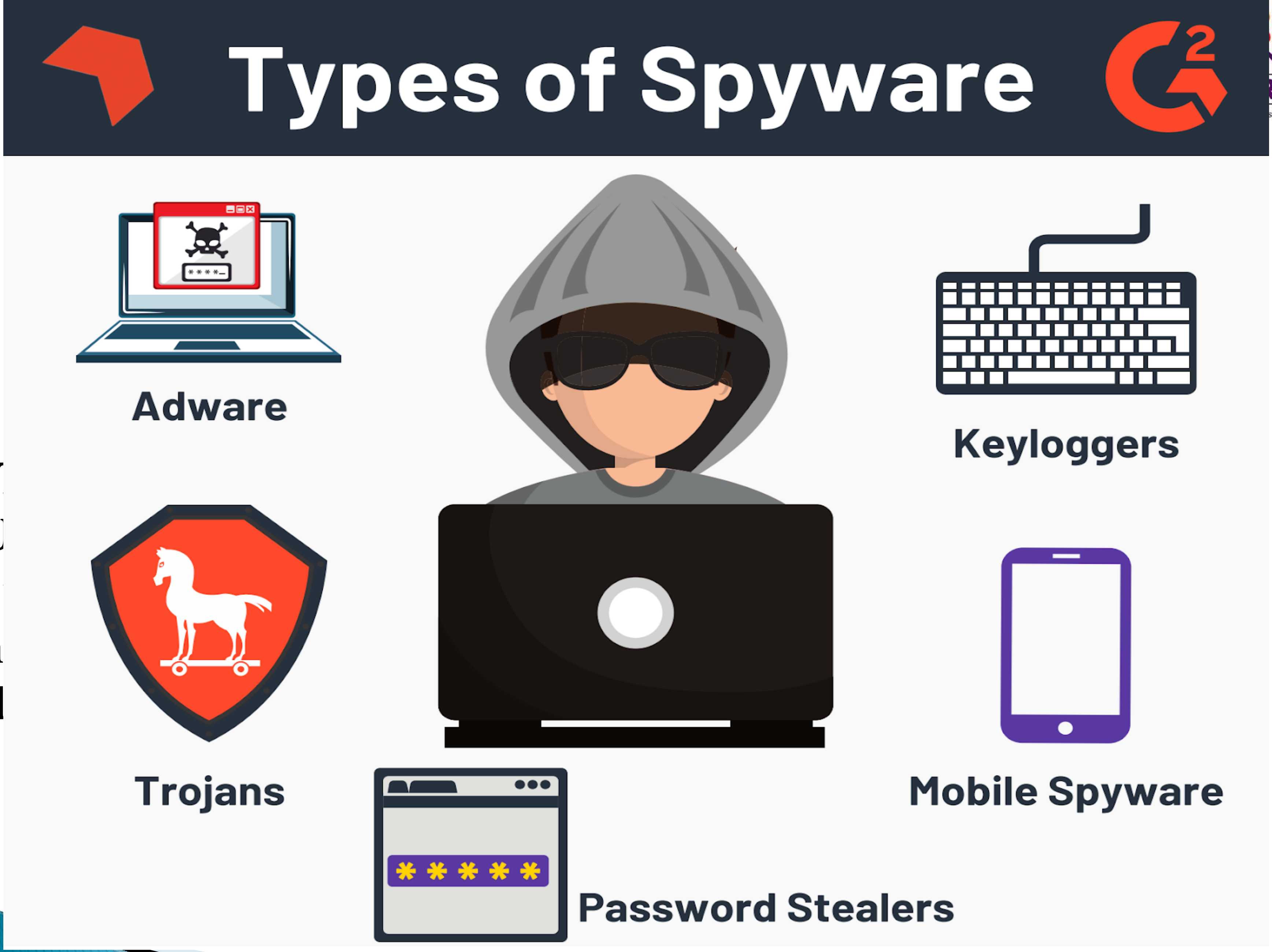
A rootkit is malware which consists of a set of programs designed to hide or obscure the fact that a system has been compromised.

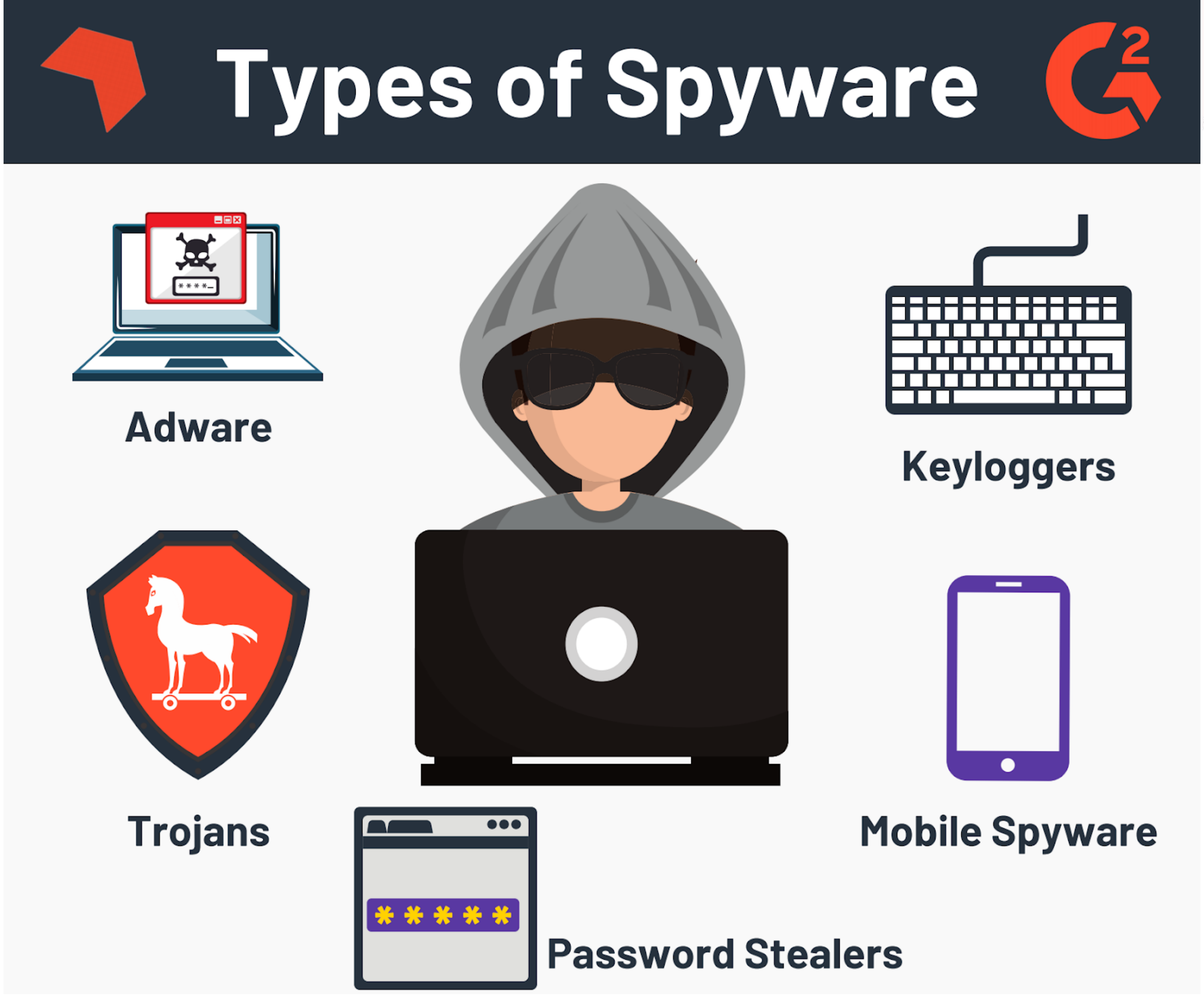
Security Alert
16/03/2009
Igor Neri - Sicurezza Informatica
2/34

•

•

U
s
a
d
s
s


Types of Spyware
Adware
Keyloggers
Trojans
Mobile Spyware
Password Stealers


Types of Spyware
Adware
Keyloggers
Trojans
Mobile Spyware
Password Stealers

• Attacks against Privileged User Accounts and

Privileged user accounts are often the first targets for cybercriminals due to their access to sensitive data and critical systems. Common attacks against these accounts include phishing, insider threats, malware, and brute force attacks. Organizations must implement robust security measures to protect these accounts from being compromised. Privilege escalation attacks, which involve gaining access to higher privileges, are a significant challenge for businesses today. Attackers may use social engineering tactics, such as phishing, to trick users into revealing their login credentials. Organizations should implement **privileged access management (PAM)** and follow the **principle of least privilege (PoLP)** to minimize the risk of privilege escalation attacks..

• Escalation of Privileges

Escalation of privilege refers to a cyberattack where an attacker gains unauthorized access to Higher_level privileges within a system, allowing them to perform actions beyond their intended permissions.

Privilege escalation is a technique used by attackers to gain elevated access to resources that are normally protected from a user or application. This can occur through exploiting bugs, design flaws, or configuration oversights in operating systems or software applications. The result is that an attacker can perform unauthorized actions, such as accessing sensitive data or modifying system settings.

Types of Privilege Escalation:
1). Vertical Privilege Escalation
2). Horizontal Privilege Escalation:

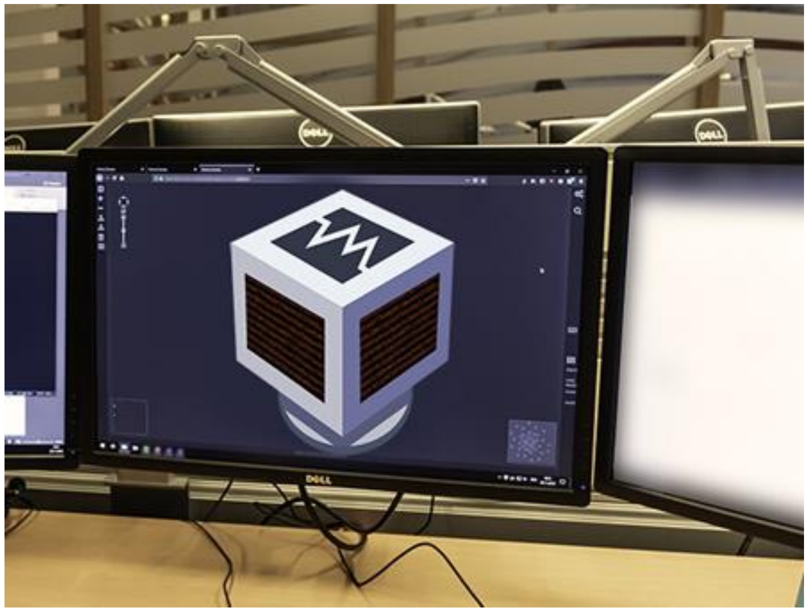
• Token Kidnapping

Token Kidnapping, also known as token impersonation, refers to the misuse of access tokens to gain unauthorized access or control within a system. Attackers can exploit this by obtaining or mimicking tokens, allowing them to impersonate legitimate users and escalate privileges within the system. This technique can bypass security measures and lead to severe threats such as data theft and system disruptions. Common methods include token theft and the use of compromised tokens to access privileged actions across the network.

• Virtual Machine Detection

Virtual machine (VM) detection involves identifying whether a system is running within a virtualized environment. This is often used by malware, security tools, and system administrators to understand the underlying environment. Below are the key techniques and methods used for VM detection:

- **CPUID Instruction**
- **BIOS/UEFI and Hardware Anomalies**
- **Timing and Performance Analysis**
- **VM-Specific Drivers and Services**
- **Hardware Fingerprinting**
- **Windows-Specific Detection**
- **Tools for VM Detection**

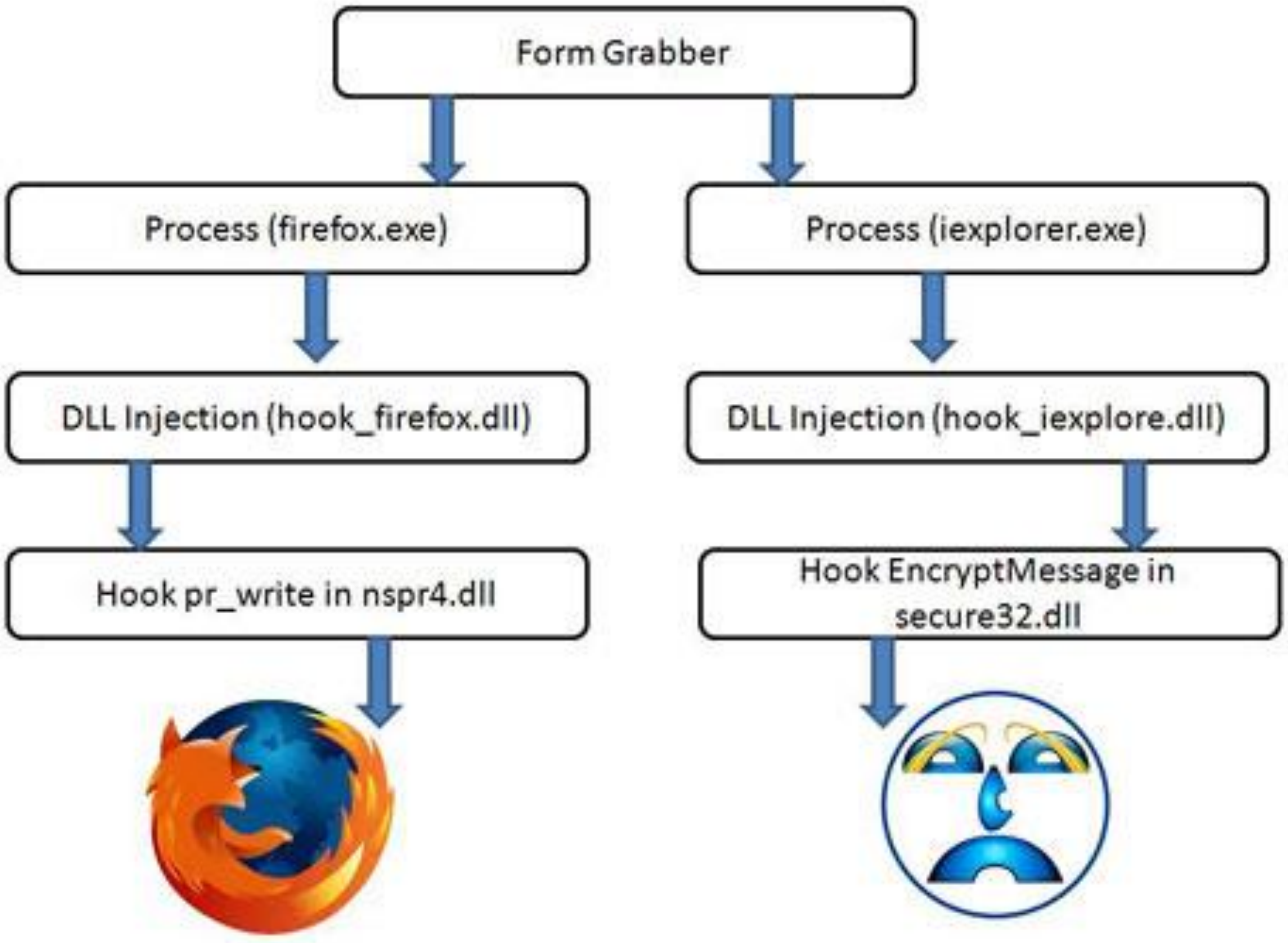


Stealing Information and Exploitation- Form Grabbing

Form grabbing is a form of malware that works by retrieving author ization and login credentials from a web data form before it is pass ed over the Internet to a secure server. This allows the malware to avoid HTTPS encryption, making it more effective than keylogger software. The method was invented in 2003 and became populariz ed by the banking trojan Zeus in 2007. Form grabbing malware can steal sensitive information even if the user inputs the information using virtual keyboards, auto-fill, or copy and paste.

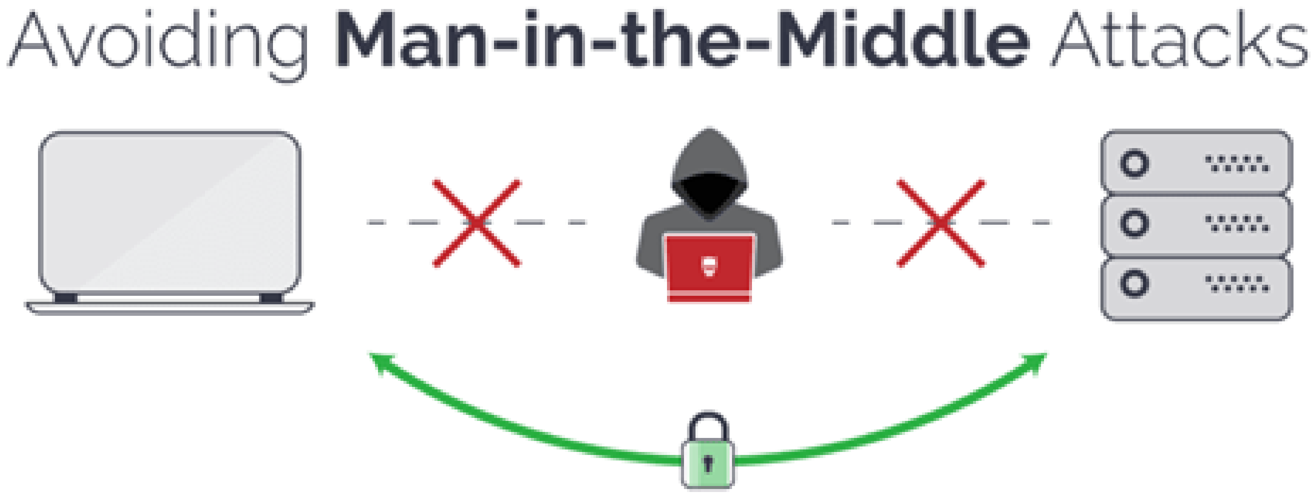
To protect against form grabbing attacks, it is crucial to use antivirus software with anti-form grabbing features, use strong and unique passwords, avoid suspicious websites, and keep software up to date.

Stealing Information and Exploitation- Form Grabbing


Form Grabber
Process (firefox.exe)
Process (iexplorer.exe)
DLL Injection (hook_firefox.dll)
DLL Injection (hook_iexplore.dll)
Hook pr_write in nspr4.dll
Hook EncryptMessage in secure32.dll

• Man-in-the- Middle Attacks

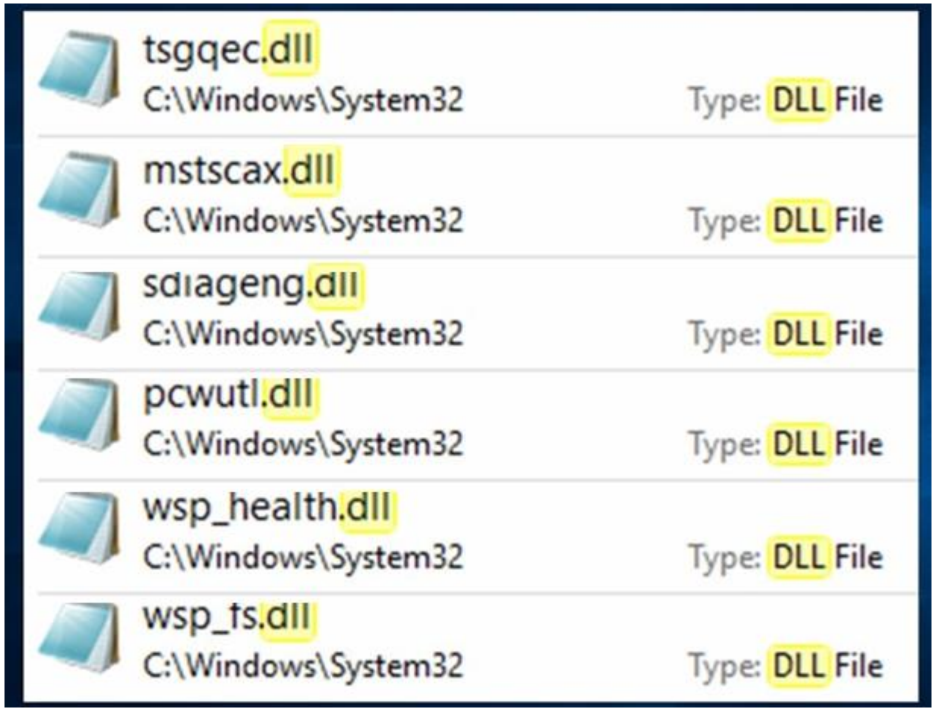
A man-in-the-middle (MITM) attack is a type of cyberattack where an attacker secretly intercepts and relays communications between two parties who believe they are directly communicating with each other. This can involve stealing sensitive information, such as login credentials or personal data, by eavesdropping on the communication channel.



• DLL Injection

DLL injection is a technique used in computer programming to run code within the address space of another process by forcing it to load a Dynamic-Link Library (DLL). This method is often used by external programs to influence the behavior of another program in ways that its authors did not anticipate or intend.

Approaches on Microsoft Windows:
1). AppInit_DLLs
2). AppCertDLLs
3). Process Manipulation Functions
4). Windows Hooking Calls
5). Suspend and Modify Thread Context:



```
tsgqec.dll
C:\Windows\System32                 Type: DLL File
mstscax.dll
C:\Windows\System32                 Type: DLL File
sdiageng.dll
C:\Windows\System32                 Type: DLL File
pcwutl.dll
C:\Windows\System32                 Type: DLL File
wsp_health.dll
C:\Windows\System32                 Type: DLL File
wsp_fs.dll
C:\Windows\System32                 Type: DLL File
```

• Browser Helper Objects

A Browser Helper Object (BHO) is a plugin designed for Internet Explorer that enhances browser functionality, but it can also pose security risks.

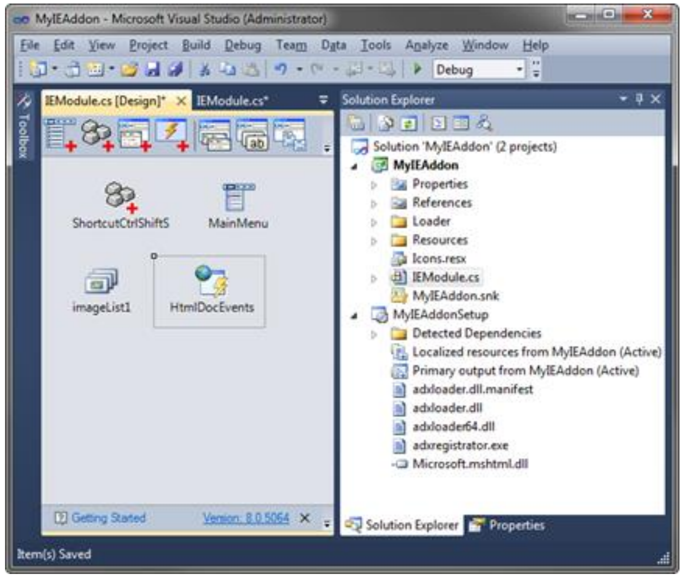
A Browser Helper Object (BHO) is a type of Component Object Model (COM) component that Internet Explorer loads whenever it starts. Introduced in 1997 with Internet Explorer 4, BHOs are designed to extend the capabilities of the browser by allowing additional functionalities, such as adding toolbars, modifying web pages, or tracking user activity

Functionality of BHOs:

1). Enhancements
2). Content Modification
3). Tracking

Security Risks:

1). Install Malware
2). Monitor Activity
3). Introduce Unwanted Ads



Internet Explorer browser helper objects

UNIT–V

UNIT–V

Pattern Matching and Tries:

- Pattern matching algorithms – Brute force,
- The Boyer– Moore algorithm,
- The Knuth-Morris-Pratt algorithm,
- Standard Tries,
- Compressed Tries,
- Suffix tries.

Pattern Matching and Tries

Definition:

given a text string T and a pattern string P, find the pattern inside the text

T: “the rain in spain stays mainly on the plain”

P: “n th”

Applications:

text editors, Web search engines (e.g. Google), image analysis

Pattern matching finds patterns in data, and tries (prefix trees) are efficient tree-based data structures ideal for this, especially with strings, organizing them by characters to enable fast searches, insertions, and prefix lookups, crucial for applications like autocomplete, spell-check, and bioinformatics. A trie stores strings character by character, allowing you to quickly check if a word or prefix exists by simply traversing the tree, making it superior to basic string comparisons for large datasets.

pattern Matching

- **Definition**: The process of searching for specific sequences (patterns) within larger data, often strings.
- **Goal**: Locate occurrences of a pattern in a text efficiently.
- **Applications**: Text editors (find/replace), plagiarism detection, bioinformatics (DNA matching), fraud detection, spell-checking, network security.

Tries (Prefix Trees)

- **Structure**: A tree where each node represents a character, and paths from the root to nodes form prefixes or complete words.
- **How it Works**:
  - **Insertion**: Each character of a word creates a new node (or follows an existing path) from the root.
  - **Search**: Follow the path for the pattern; if you reach the end of the pattern at a valid node, it's found.
  - **Prefix Search**: Navigate to the end of the prefix and explore all subsequent nodes.

Tries (Prefix Trees)

- **Benefits for Pattern Matching**:
- **Speed**: Very fast for prefix searches (autocomplete) and dictionary lookups.
- **Efficiency**: Avoids redundant comparisons, especially useful when searching for many patterns.
- **Storage**: Can be memory-efficient, especially with compression (Compressed Tries)

String Concepts

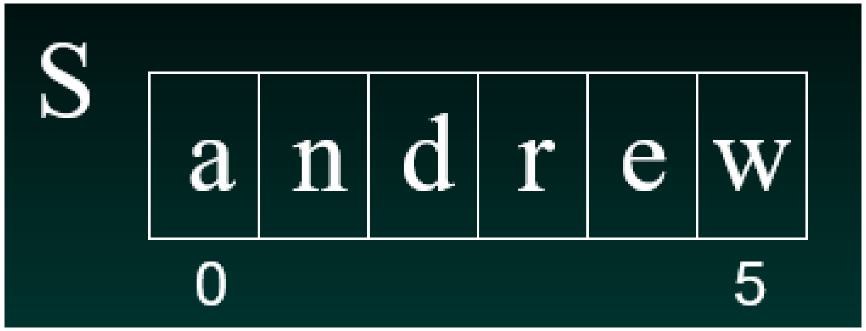
Assume S is a string of size m.

$S = x_1x_2 \ldots x_m$

A *prefix* of S is a substring S[1 .. *k*-1]
A *suffix* of S is a substring S[*k*-1 .. m]

- k is any index between 1 and m
- S[0] is null character

Examples



- All possible prefixes of S:

“∅”, “a", "an", "and", "andr”, "andre“,

- All possible suffixes of S:

“∅”, “w", “ew", “rew", “drew", “ndrew”

Examples

- All possible prefixes of S:
“∅”, “a", "an", "and", "andr”, "andre“,

- All possible suffixes of S:
“∅”, “w", “ew", “rew", “drew", “ndrew”

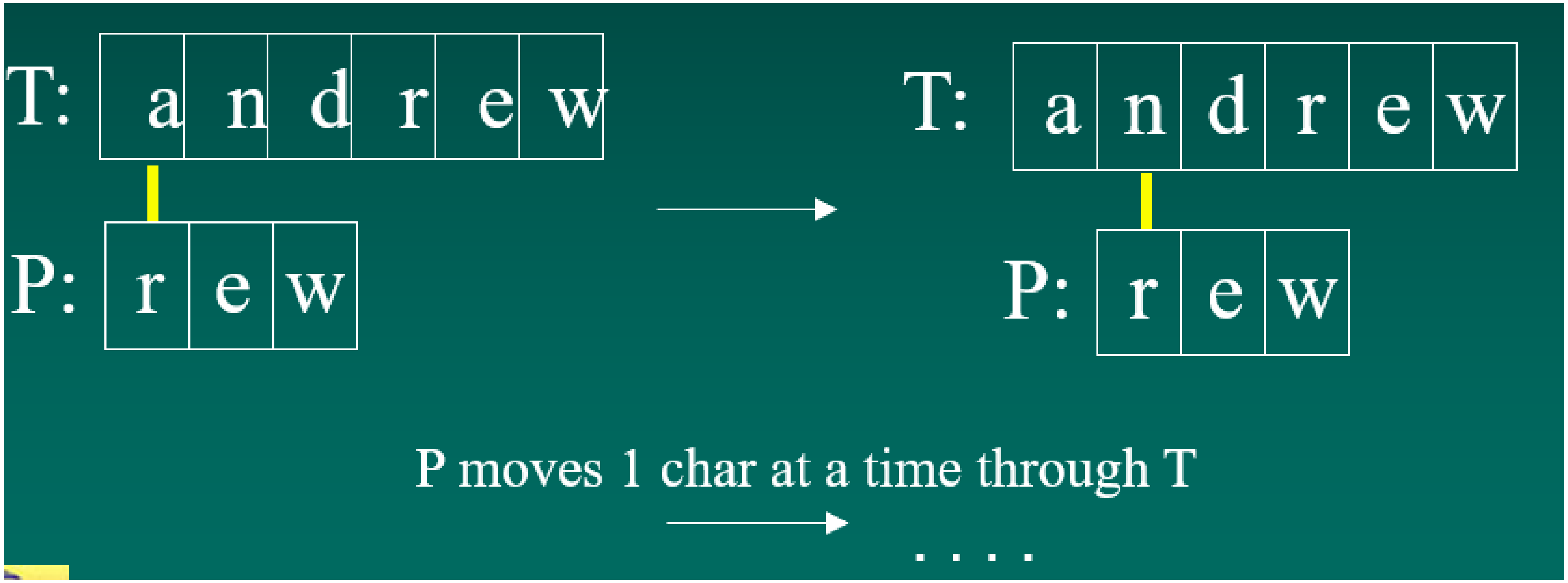
Examples

- All possible prefixes of S:
“∅”, “a", "an", "and", "andr”, "andre“,

- All possible suffixes of S:
“∅”, “w", “ew", “rew", “drew", “ndrew”

The Brute Force Algorithm

Check each position in the text T to see if the pattern P starts in that position



Brute Force in Java

Return index where pattern starts, or -1

```
      public static int brute(String text,String pattern)
{ int n = text.length();    // n is length of text
  int m = pattern.length(); // m is length of pattern
  int j;
  for(int i=0; i <= (n-m); i++) {
    j = 0;
    while ((j < m) &&
          (text.charAt(i+j) == pattern.charAt(j)) )
      j++;
    if (j == m)
      return i;   // match at i
  }
  return -1;   // no match
} // end of brute()
```

Usage

```
     public static void main(String args[])
{ if (args.length != 2) {
    System.out.println("Usage: java BruteSearch
                                          <text> <pattern>");
    System.exit(0);
  }
  System.out.println("Text: " + args[0]);
  System.out.println("Pattern: " + args[1]);

  int posn = brute(args[0], args[1]);
  if (posn == -1)
    System.out.println("Pattern not found");
  else
    System.out.println("Pattern starts at posn "
                                          + posn);
}
```

Analysis

- Brute force pattern matching runs in time O(mn) in the **worst case**.
- But most searches of ordinary text take O(m+n), which is **very quick**.

The brute force algorithm is fast when the alphabet of the text is large

- e.g. A..Z, a..z, 1..9, etc.

It is slower when the alphabet is small

- e.g. 0, 1 (as in binary files, image files, etc.)

Example of a worst case:

T: "aaaaaaaaaaaaaaaaaaaaaaaaaah"

P: "aaah"

Example of a more average case:

T: "a string searching example is standard"

P: "store"

The brute force algorithm is fast when the alphabet of the text is large

- e.g. A..Z, a..z, 1..9, etc.

It is slower when the alphabet is small

- e.g. 0, 1 (as in binary files, image files, etc.)

- **The Boyer– Moore algorithm**

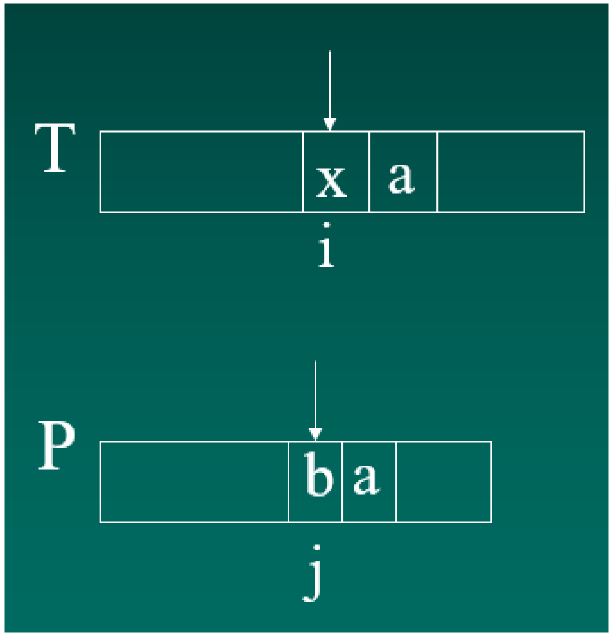
The Boyer–Moore pattern matching algorithm is based on two techniques.

1. The *looking–glass* technique
   - find P in T by moving *backwards* through P, starting at its end

2. The *character-jump* technique

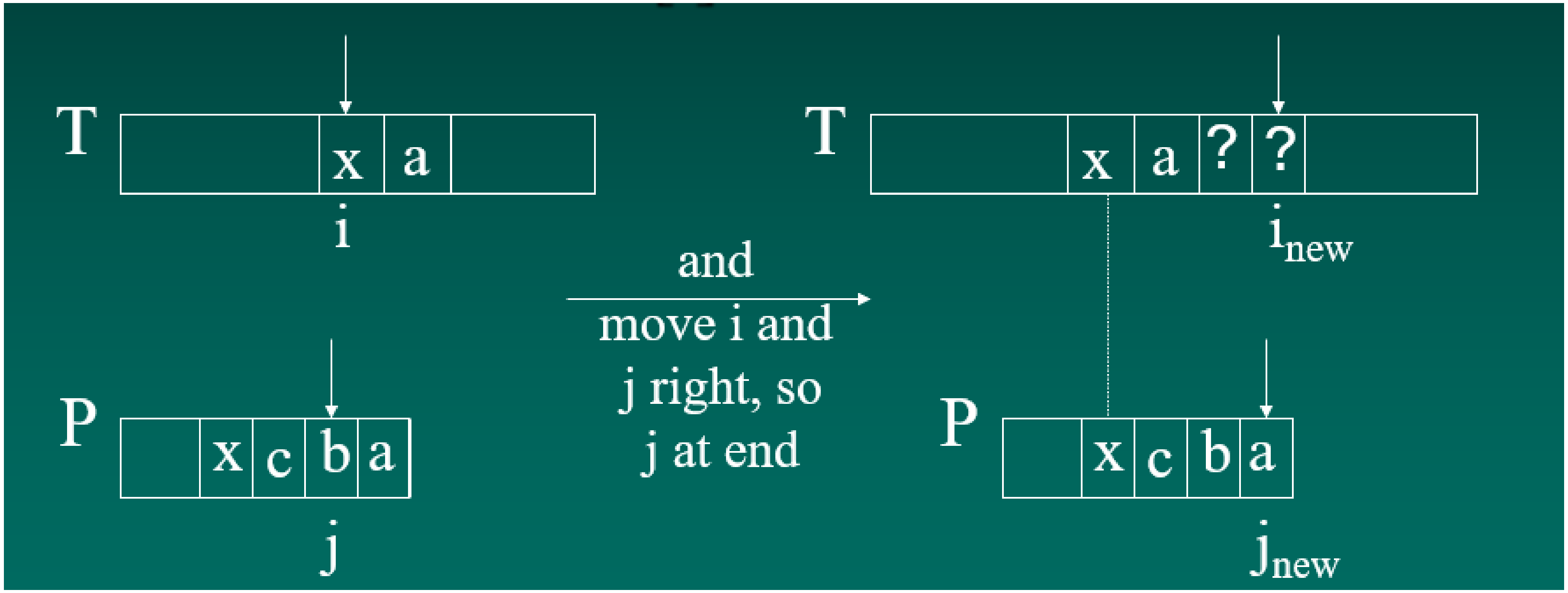
when a mismatch occurs at T[i] == x
the character in pattern P[j] is not the same as T[i]

There are 3 possible cases, tried in order.



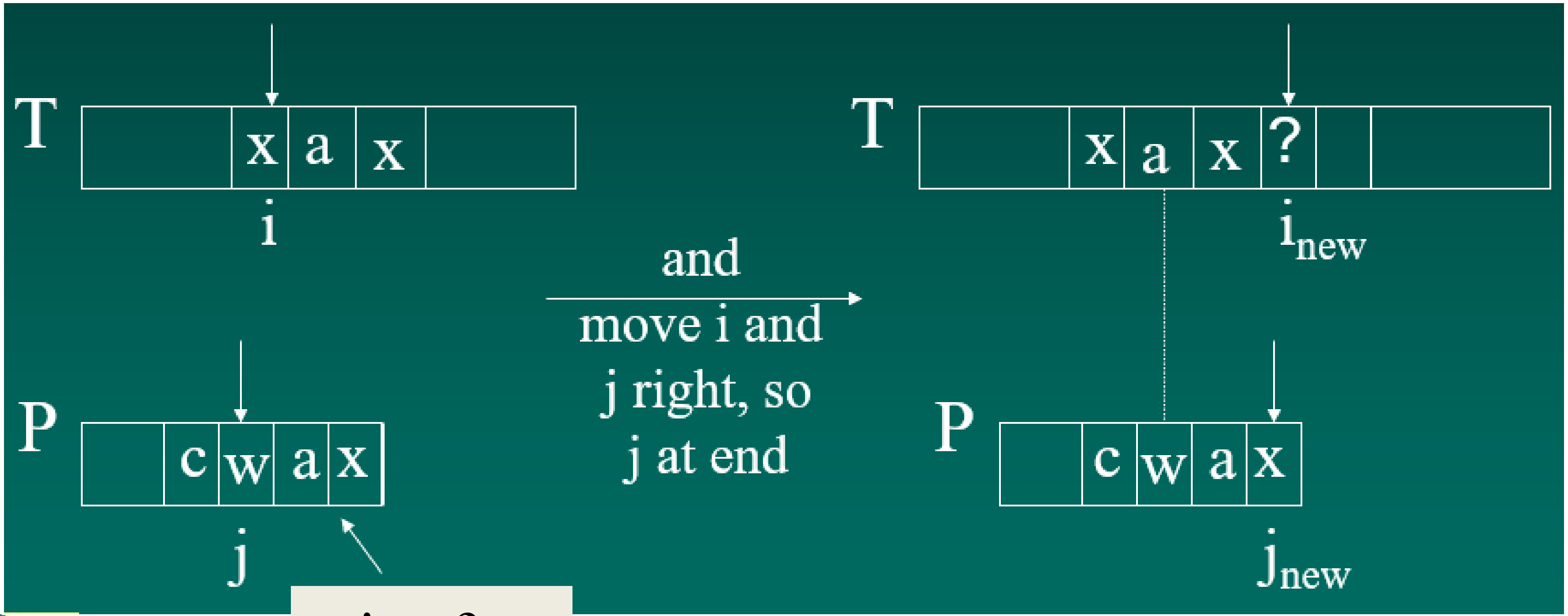
Case 1

If P contains x somewhere, then try to *shift P* right to align the last occurrence of x in P with T[i].



Case 2

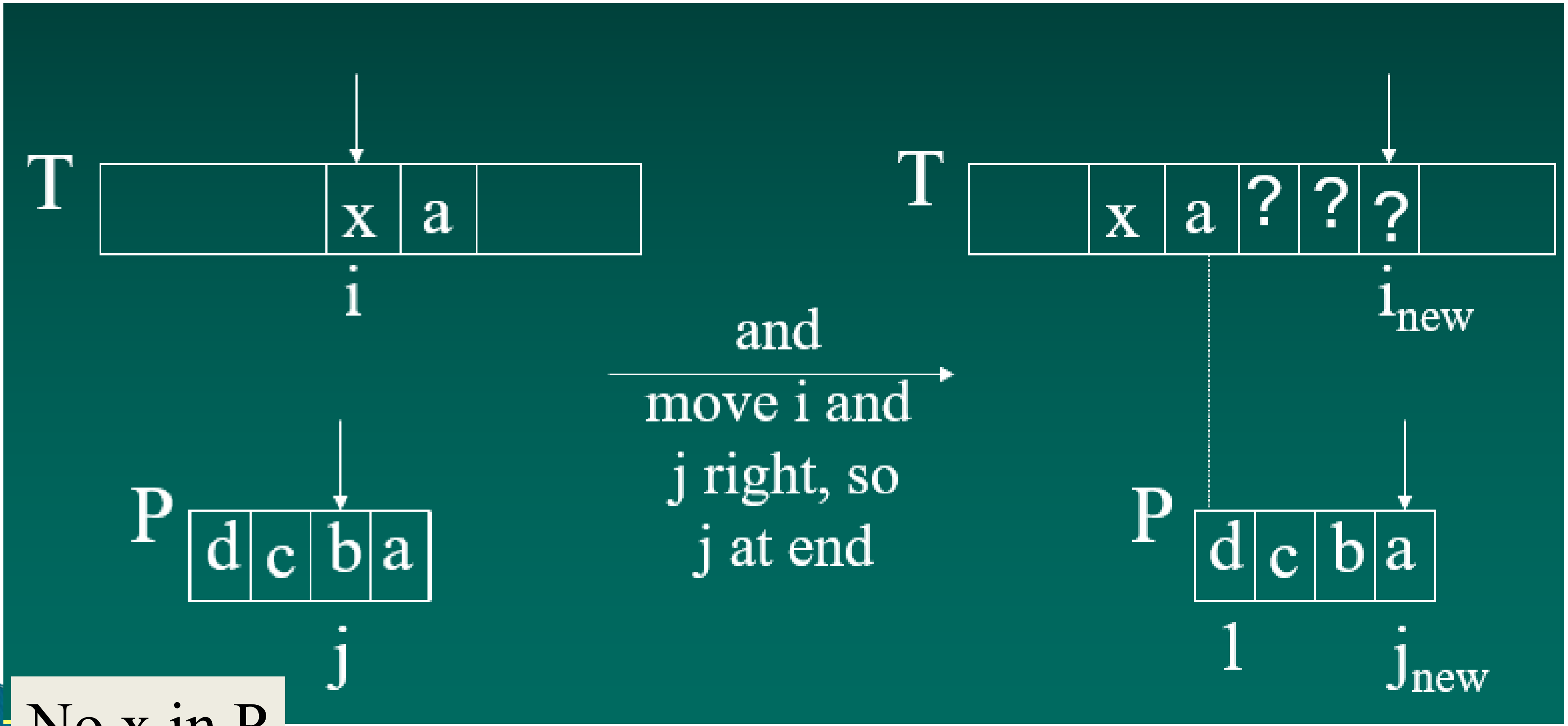
If P contains x somewhere, but a shift right to the last occurrence is *not* possible, then *shift P* right by 1 character to T[i+1].



x is after j position

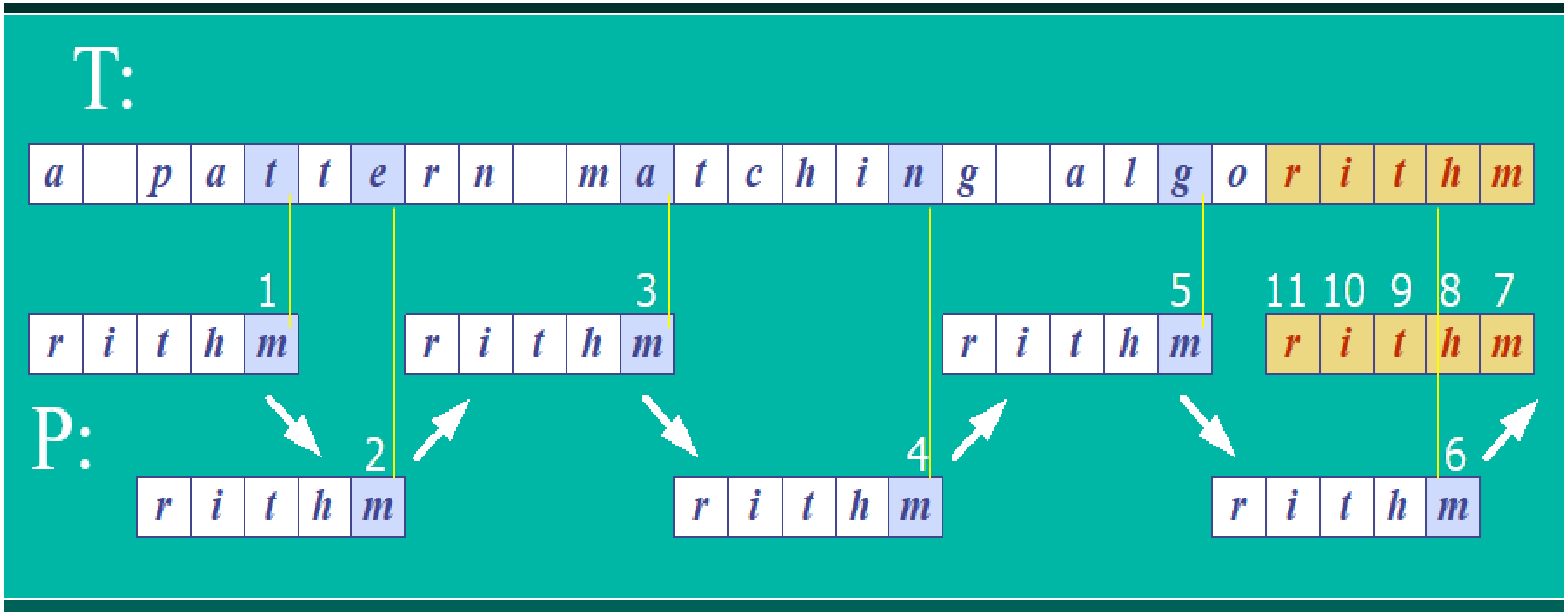
Case 3

If cases 1 and 2 do not apply, then *shift* P to align P[1] with T[i+1].



No x in P

Boyer-Moore Example (1)



Last Occurrence Function

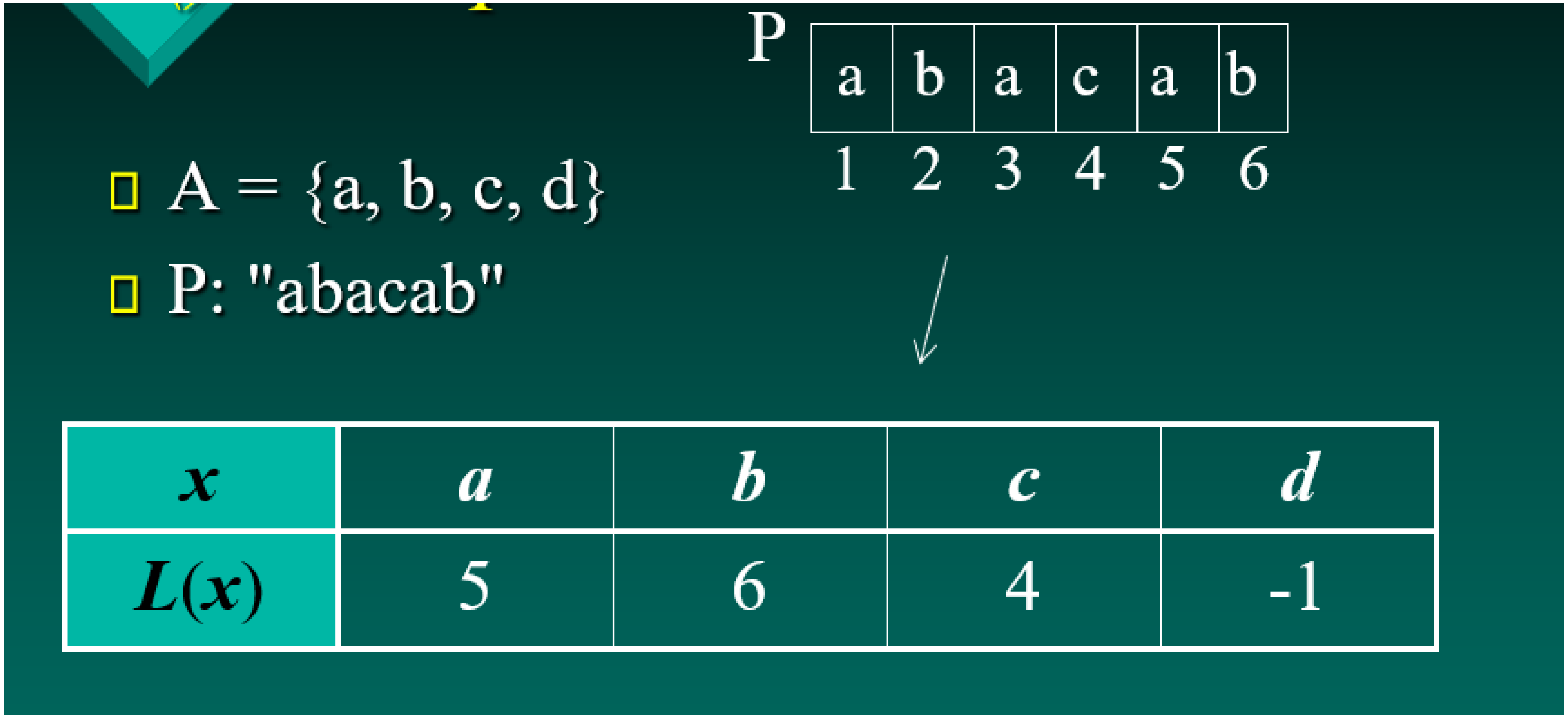
Boyer–Moore's algorithm preprocesses the pattern P and the alphabet A to build a last occurrence function L()

- L() maps all the letters in A to integers

L(x) is defined as: // x is a letter in A

- The largest index i such that P[i] == x, or
-1 if no such index exists

L() Example



P

a	b	a	c	a	b
1	2	3	4	5	6

- A = {a, b, c, d}
- P: "abacab"

x	*a*	*b*	*c*	*d*
L(*x*)	5	6	4	-1

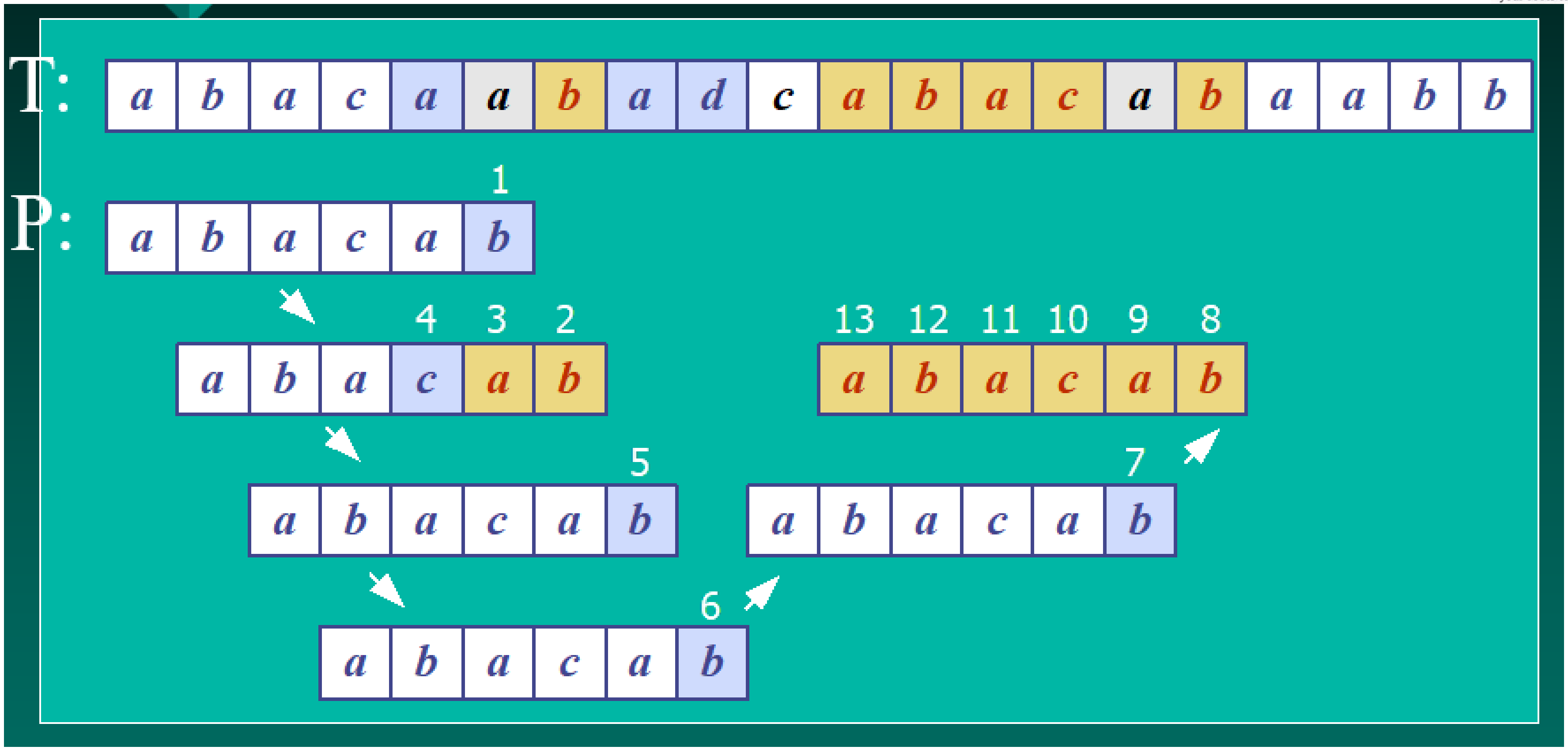
L() stores indexes into P[]

L() stores indexes into P[]

In Boyer–Moore code, L() is calculated when the pattern P is read in.

Usually L() is stored as an array
- something like the table in the previous slide

Boyer-Moore Example (2)



x	a	b	c	d
$L(x)$	5	6	4	–1

Boyer-Moore in Java

Return index where pattern starts, or -1

```
      public static int bmMatch(String
text,
                          String pattern)
  {
    int last[] = buildLast(pattern);
    int n = text.length();
    int m = pattern.length();
    int i = m-1;

    if (i > n-1)
      return -1; // no match if pattern is
                 // longer than text
               :
```

```
          int j = m-1;
    do {
      if (pattern.charAt(j) == text.charAt(i))
        if (j == 0)
          return i; // match
        else { // looking-glass technique
          i--;
          j--;
        }
      else { // character jump technique
         int lo = last[text.charAt(i)];  //last occ
         i = i + m - Math.min(j, 1+lo);
         j = m - 1;
      }
    } while (i <= n-1);

    return -1; // no match
  } // end of bmMatch()
```

```
    public static int[] buildLast(String pattern)
/* Return array storing index of last
  occurrence of each ASCII char in pattern. */
{
  int last[] = new int[128]; // ASCII char set

  for(int i=0; i < 128; i++)
    last[i] = -1; // initialize array

  for (int i = 0; i < pattern.length(); i++)
    last[pattern.charAt(i)] = i;

  return last;
} // end of buildLast()
```

```
      public static void main(String args[])
  { if (args.length != 2) {
      System.out.println("Usage: java BmSearch
                                <text> <pattern>");
      System.exit(0);
    }
    System.out.println("Text: " + args[0]);
    System.out.println("Pattern: " + args[1]);

    int posn = bmMatch(args[0], args[1]);
    if (posn == -1)
      System.out.println("Pattern not found");
    else
      System.out.println("Pattern starts at posn "
                                          + posn);
}
```

Usage

```
        public static void main(String args[])
  { if (args.length != 2) {
      System.out.println("Usage: java BmSearch
                                 <text> <pattern>");
      System.exit(0);
    }
    System.out.println("Text: " + args[0]);
    System.out.println("Pattern: " + args[1]);

    int posn = bmMatch(args[0], args[1]);
    if (posn == -1)
      System.out.println("Pattern not found");
    else
      System.out.println("Pattern starts at posn "
                                        + posn);
}
```

Analysis

Boyer–Moore worst case running time is O(nm + A)

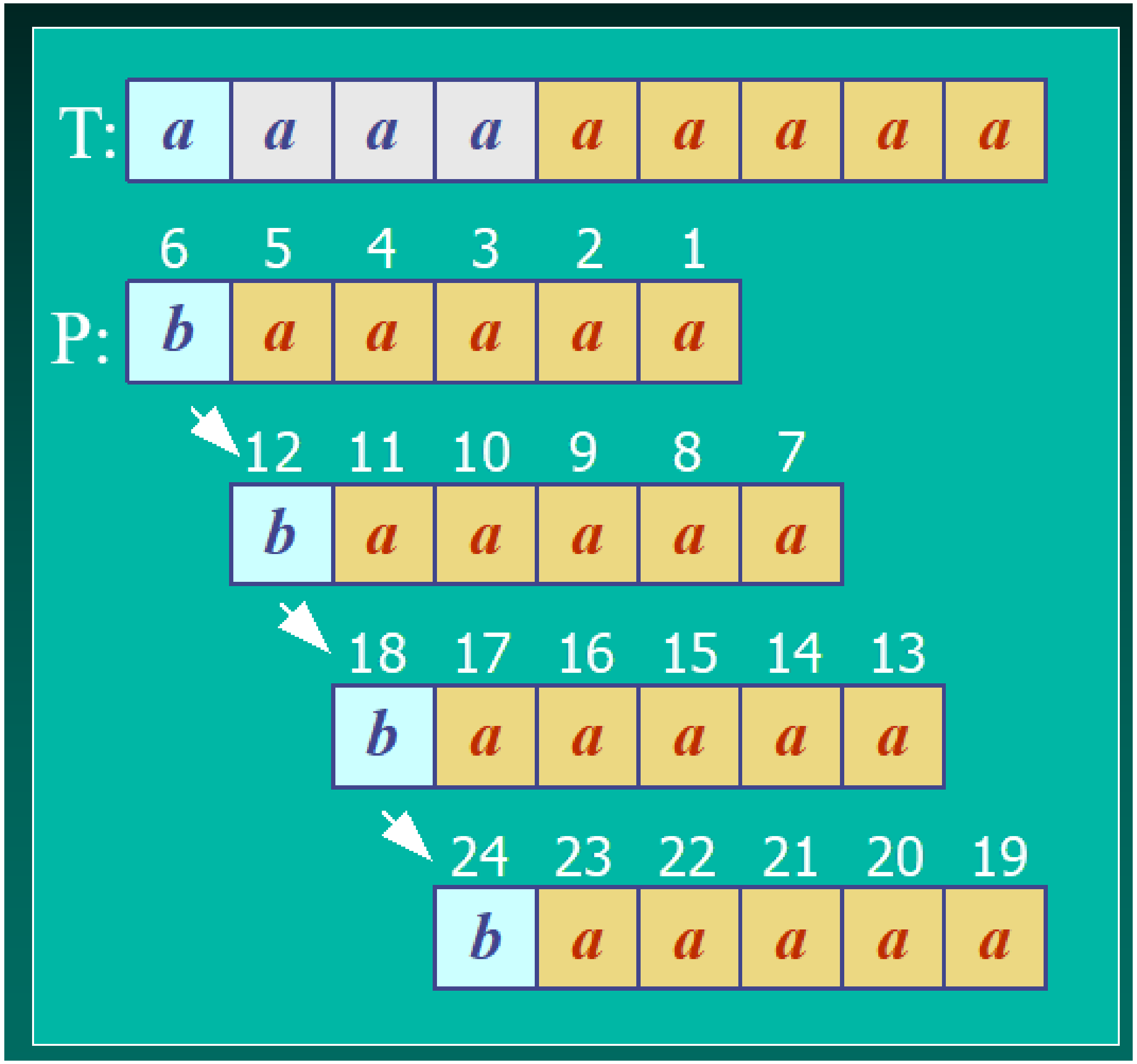
But, Boyer–Moore is fast when the alphabet (A) is large, slow when the alphabet is small.

e.g. good for English text, poor for binary

Boyer–Moore is *significantly faster than brute force* for searching English text.

Worst Case Example

T: "aaaaa…a"
P: "baaaaa"



5. More Information

Algorithms in C++
Robert Sedgewick
Addison-Wesley, 1992
chapter 19, String Searching

Online Animated Algorithms:
http://www.ics.uci.edu/~goodrich/dsa/11strings/demos/pattern/
http://www-sr.informatik.uni-tuebingen.de/~buehler/BM/BM1.html
http://www-igm.univ-mlv.fr/~lecroq/string/

The Knuth-Morris-Pratt(KMP) algorithm

The KMP Algorithm

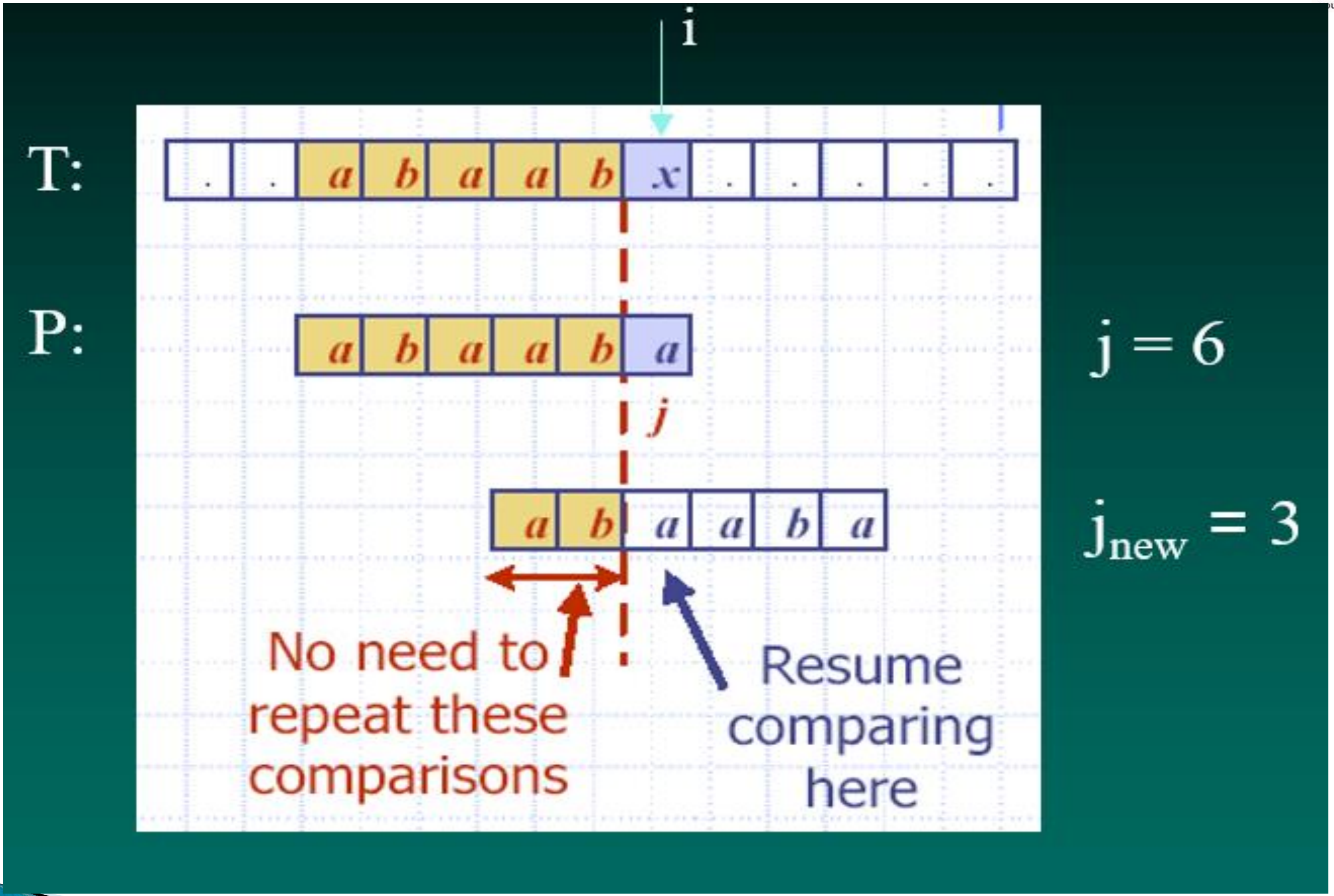
The Knuth-Morris-Pratt (KMP) algorithm looks for the pattern in the text in a *left-to-right* order (like the brute force algorithm).

But it shifts the pattern more intelligently than the brute force algorithm.

If a mismatch occurs between the text and pattern P at P[j], what is the **most** we can shift the pattern to avoid wasteful comparisons?

Answer: the largest prefix of P[1 .. j-1] that is a suffix of P[1 .. j-1]

Example


i
T:
. . a b a a b x . . . . .
P:
a b a a b a
j
j = 6
a b a a b a
j_{new} = 3
No need to repeat these comparisons
Resume comparing here

Why

j==5

Find largest prefix (start) of:

"a b a a b" (P[1..j-1])

which is suffix (end) of:

“a b a a b" (p[1 .. j-1])

Answer: "a b"

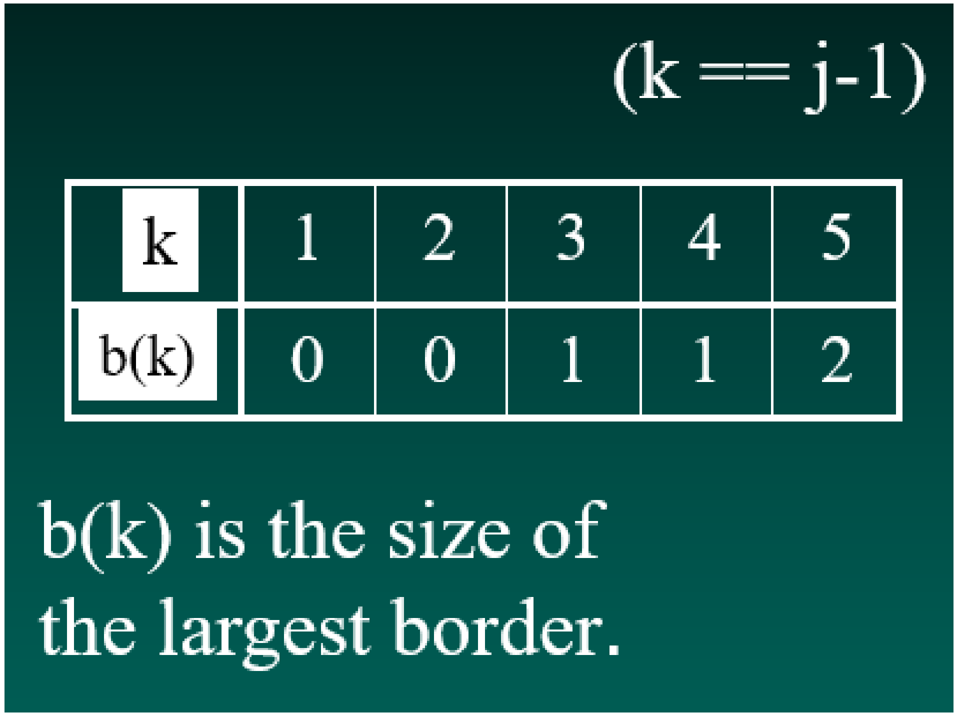
Set j = 3 // the new j value

KMP Border Function

- KMP preprocesses the pattern to find matches of prefixes of the pattern with the pattern itself.
- j = mismatch position in P[ ]
- k = position before the mismatch (k = j-1).
- The *border function* b(k) is defined as the *size* of the largest prefix of P[1..k] that is also a suffix of P[1..k].

Border Function Example

P: "abaaba"
j: 123456



(k == j-1)

k	1	2	3	4	5
b(k)	0	0	1	1	2

b(k) is the size of the largest border.

In code, b() is represented by an array, like the table.

Why is b(5) == 2?

P: "abaaba"

b(5) means
- find the size of the largest prefix of P[1..5] that is also a suffix of P[1..5]
= find the size largest prefix of "abaab" that is also a suffix of "baab"
= find the size of "ab"
= 2

Using the Failure Function

Knuth-Morris-Pratt's algorithm modifies the brute-force algorithm.
if a mismatch occurs at P[j]
(i.e. P[j] != T[i]), then

```
   k = j-1;
   j = b(k) + 1;     // obtain the new j
```

KMP in Java

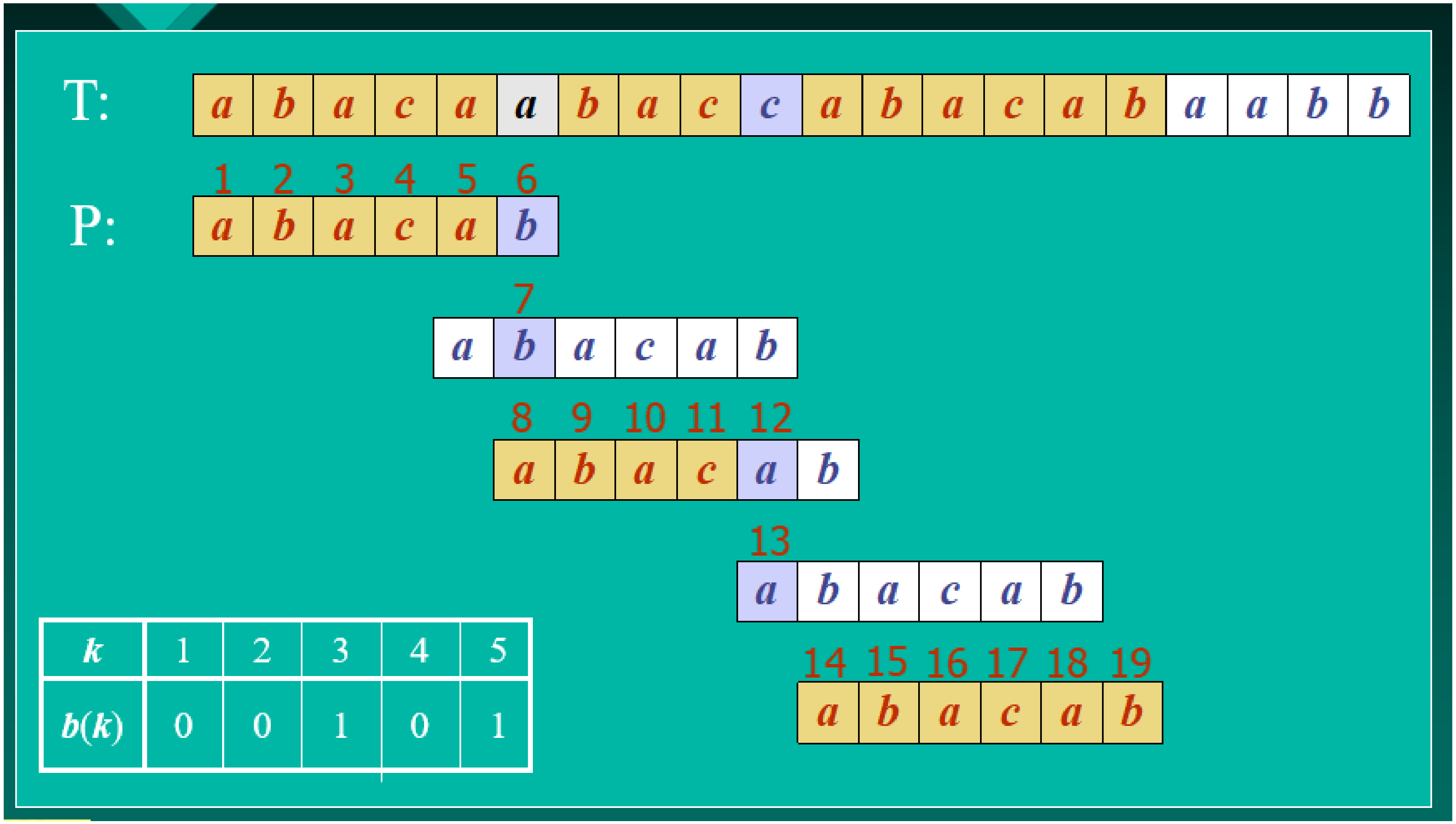
```
public static int kmpMatch(String text,
                               String pattern)
 {
   int n = text.length();
   int m = pattern.length();

   int fail[] = computeFail(pattern);

   int i=0;
   int j=0;
while (i < n) {
      if (pattern.charAt(j) == text.charAt(i))
{
        if (j == m - 1)
          return i - m + 1; // match
        i++;
        j++;
      }
      else if (j > 0)
  j = fail[j-1];
      else
        i++;
    }
    return -1; // no match
  } // end of kmpMatch()
```

```
public static int[] computeFail(
                                   String pattern)
 {
   int fail[] = new int[pattern.length()];
   fail[0] = 0;

   int m = pattern.length();
   int j = 0;
   int i = 1;
while (i < m) {
     if  (pattern.charAt(j) ==
          pattern.charAt(i)) {   //j+1 chars match
       fail[i] = j + 1;
       i++;
       j++;
     }
     else if (j > 0) // j follows matching prefix
       j = fail[j-1];
     else {      // no match
       fail[i] = 0;
       i++;
     }
   }
   return fail;
 } // end of computeFail()
```

Usage

```
public static void main(String args[])
 { if (args.length != 2) {
      System.out.println("Usage: java KmpSearch
                                <text> <pattern>");
      System.exit(0);
   }
   System.out.println("Text: " + args[0]);
   System.out.println("Pattern: " + args[1]);

   int posn = kmpMatch(args[0], args[1]);
   if (posn == -1)
      System.out.println("Pattern not found");
   else
      System.out.println("Pattern starts at posn "
                                         + posn);
```

Example



k	1	2	3	4	5
b(k)	0	0	1	0	1

Why is b(4) == 1?

P: "abacab"

b(4) means
- find the size of the largest prefix of P[1..5]
that is also a suffix of P[1..5]
= find the size largest prefix of "abaca" that
is also a suffix of "baca"
= find the size of "a"
= 1

KMP Advantages

- KMP runs in optimal time: O(m+n)
  - very fast

- The algorithm never needs to move backwards in the input text,
  - Tthis makes the algorithm good for processing very large files that are read in from external devices or through a network stream

KMP Disadvantages

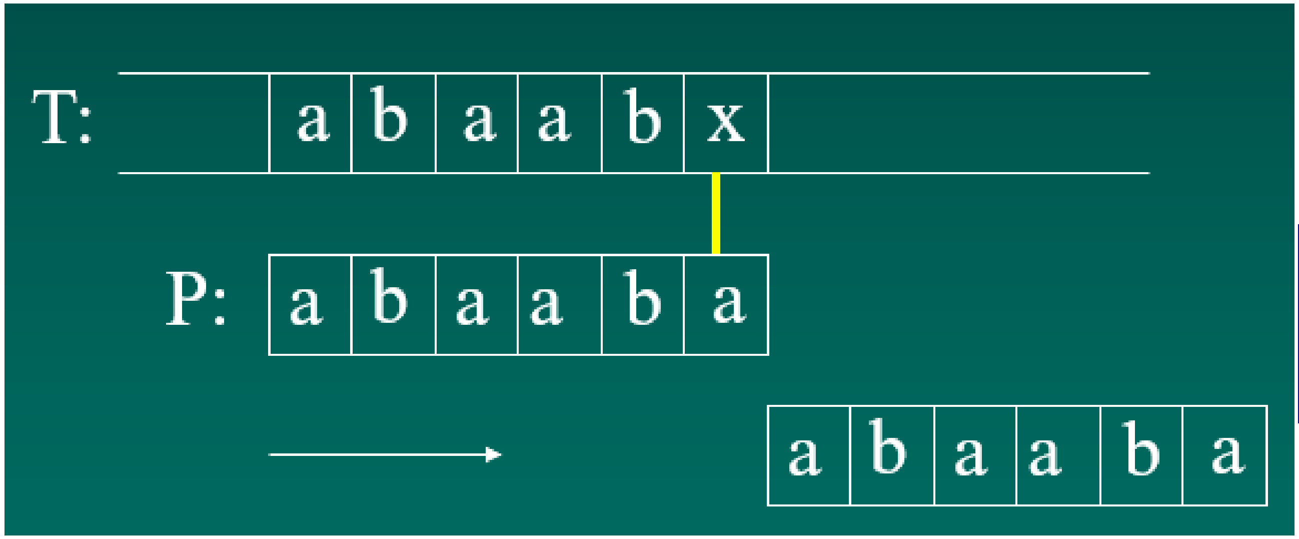
KMP doesn’t work so well as the size of the alphabet increases

- more chance of a mismatch (more possible mismatches)

- mismatches tend to occur early in the pattern, but KMP is faster when the mismatches occur later

KMP Extensions

- The basic algorithm doesn't take into account the letter in the text that caused the mismatch.



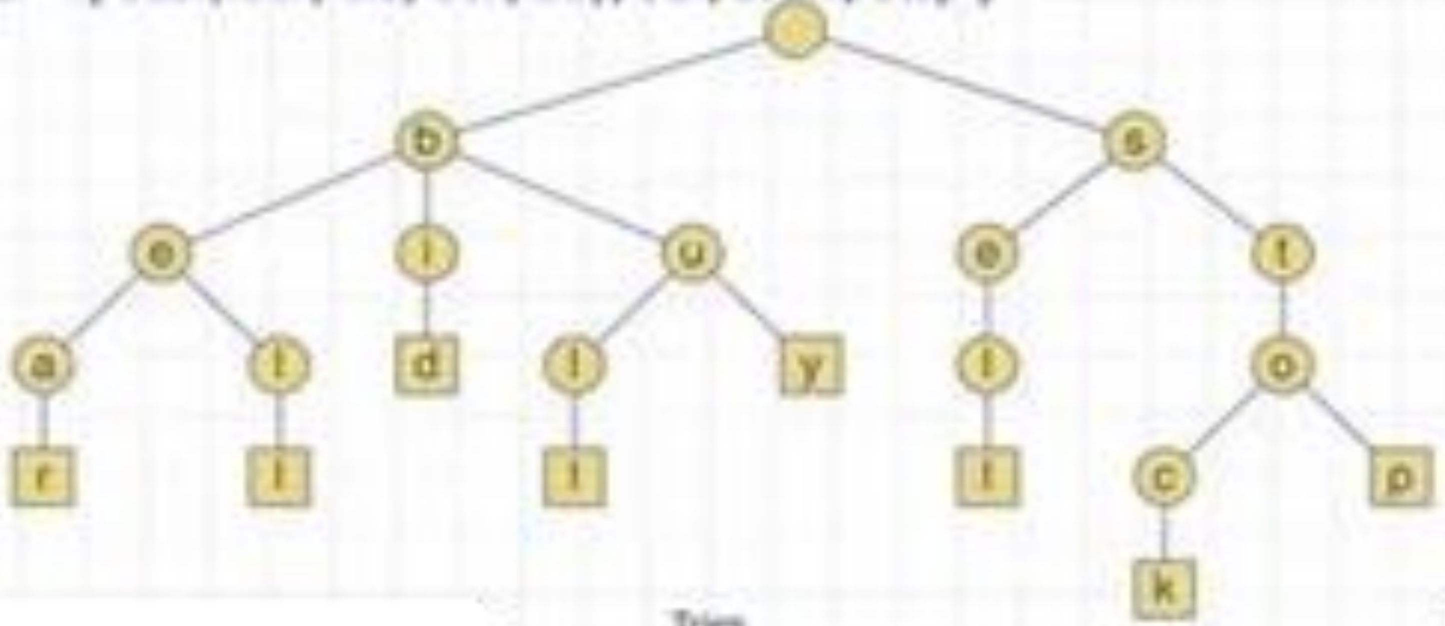
Basic KMP does **not** do this.

Standard Tries

A standard trie, also known as a prefix tree, is a hierarchical data structure that stores a dynamic set of strings. In a standard trie, each node represents a single character in a key or string, with the root node associated with the empty string. The path from the root to a node represents a key, and typical operations such as find, insert, and remove take time proportional to the length of the string being processed.

Standard Tries

- The standard trie for a set of strings S is an ordered tree such that:
  - Each node but the root is labeled with a character
  - The children of a node are alphabetically ordered
  - The paths from the external nodes to the root yield the strings of S
- Example: standard trie for the set of strings
  S = { bear, bell, bid, bull, buy, sell, stock, stop }

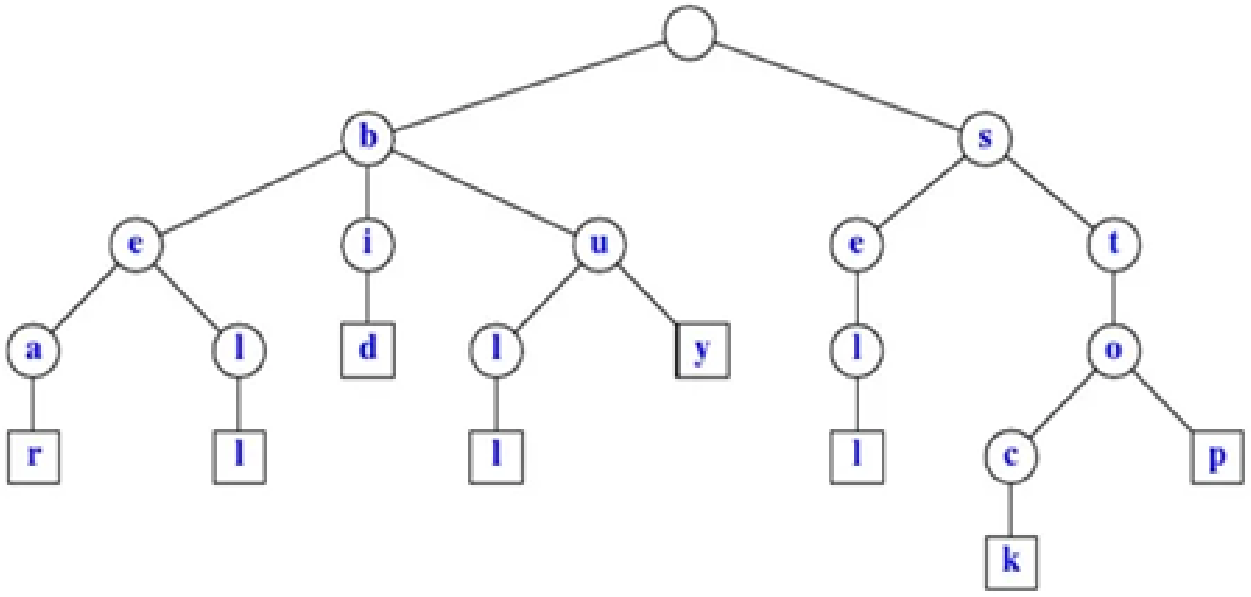


Tries

Standard Tries

TRIES

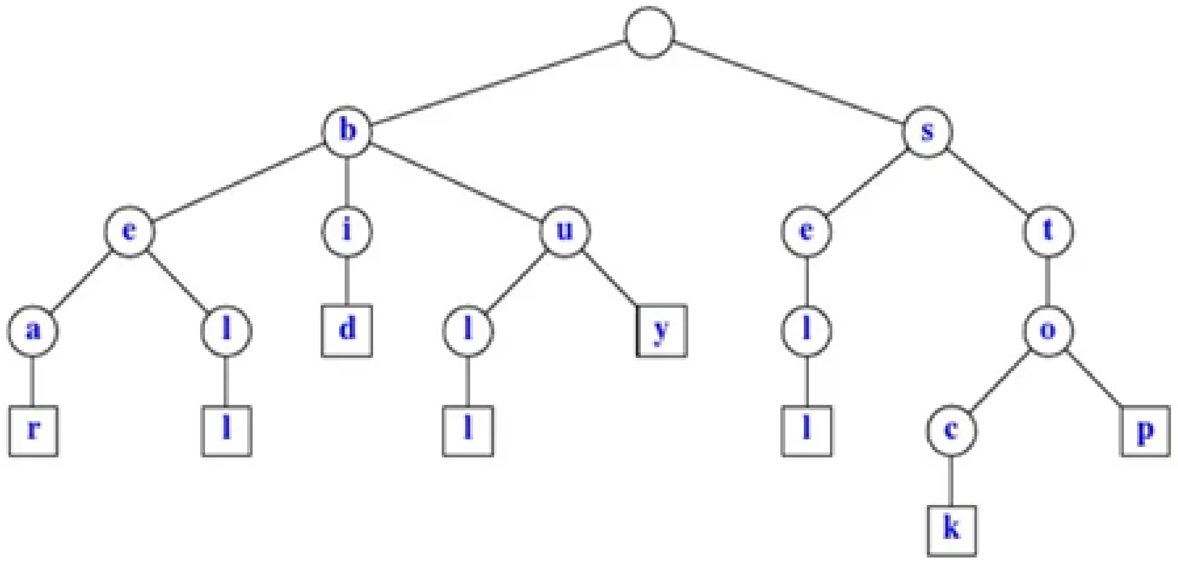
- Standard Tries
- Compressed Tries
- Suffix Tries



Standard Tries

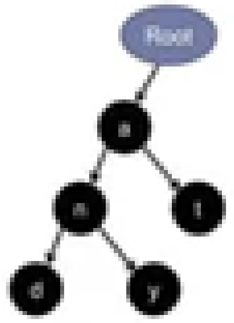
TRIES

- Standard Tries
- Compressed Tries
- Suffix Tries



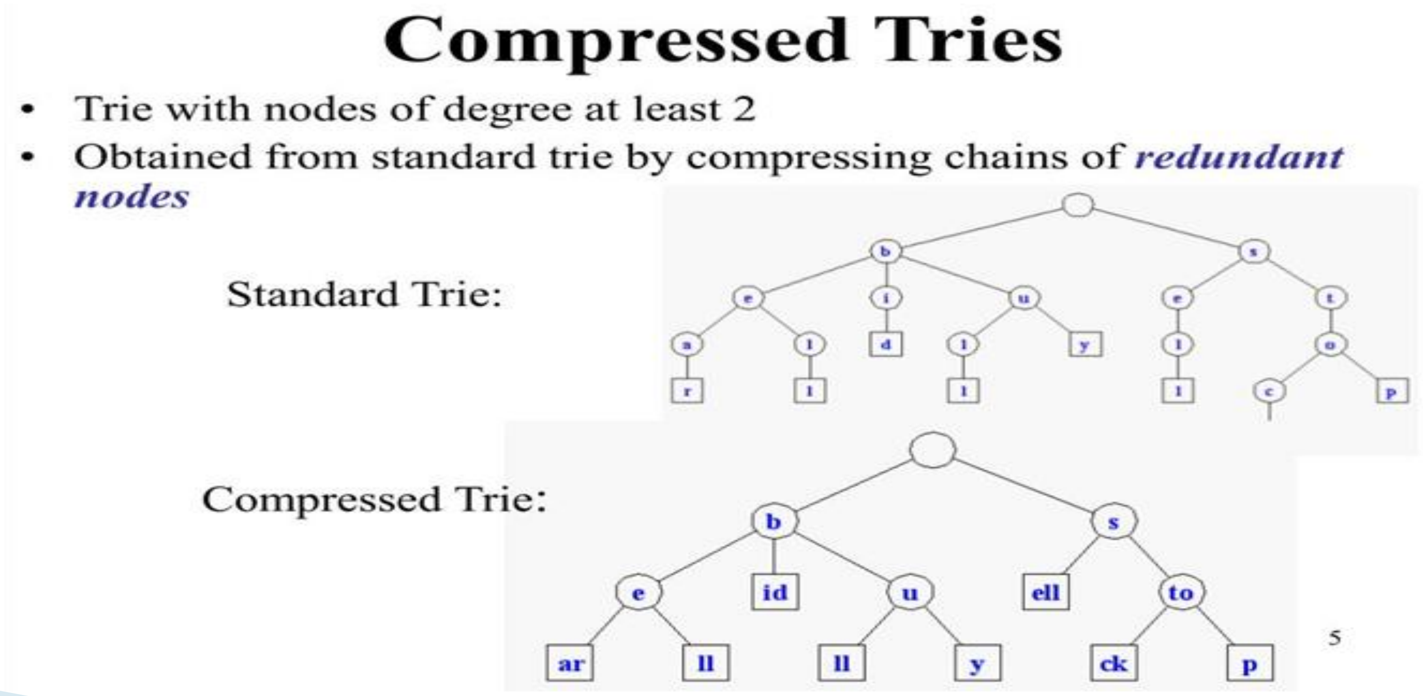
Standard Tries - Insertion

- Strings ={an, and, any, at}



• Compressed Tries

A compressed trie is a type of trie data structure used for storing and retrieving keys, particularly strings. In a compressed trie, nodes with only one child are merged with their parent nodes, which optimizes space and improves efficiency. This structure is also known as a radix tree or compact prefix tree, and it offers significant memory advantages, especially for long strings with common prefixes.



Compressed Tries

- Trie with nodes of degree at least 2
- Obtained from standard trie by compressing chains of ***redundant nodes***

Standard Trie:

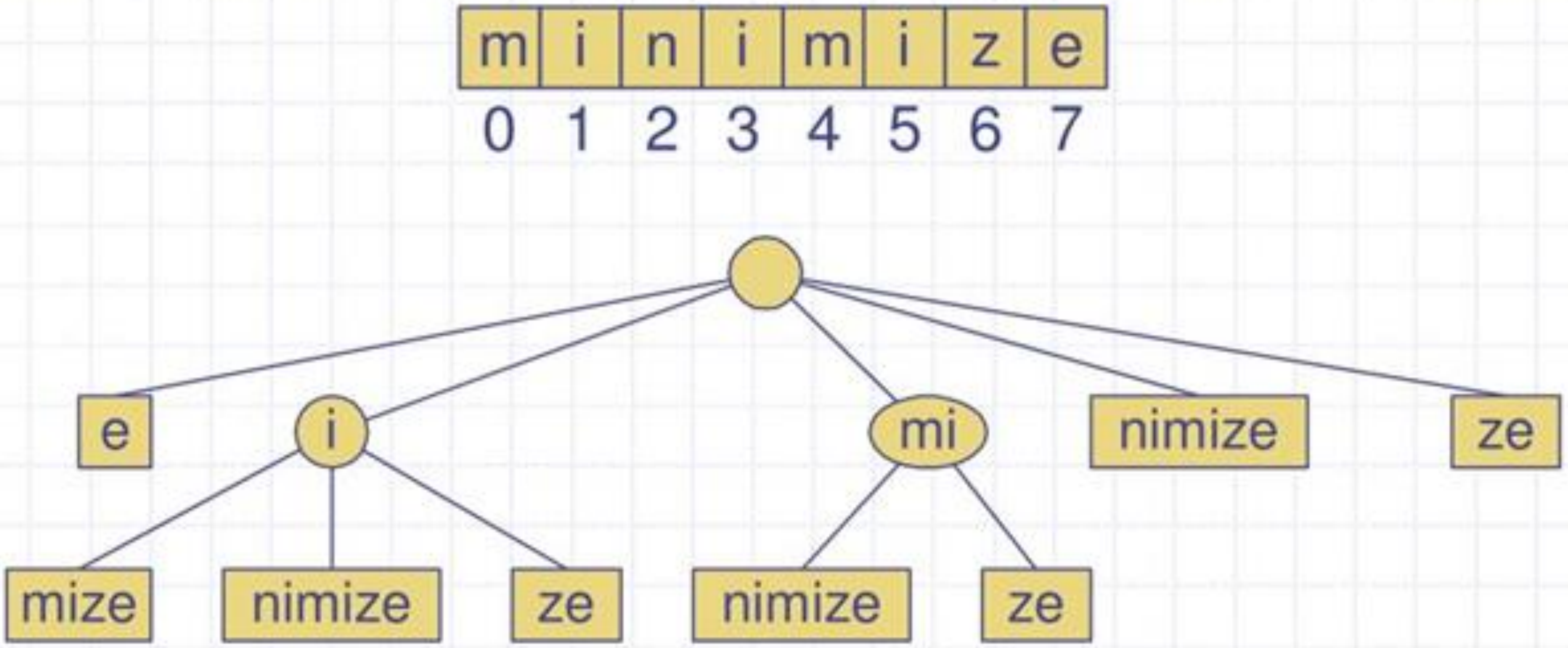
Compressed Trie:

Suffix Tries

A suffix trie is a specialized data structure used to represent all the suffixes of a given string. It is structured as a tree where each node corresponds to a suffix of the string, with the root node representing the entire string and each child node representing a suffix that is one character shorter than its parent. Suffix tries are commonly used in applications such as pattern matching and word matching, allowing for efficient searching and retrieval of substrings. They store both the original word and its suffixes during insertion, making them useful for various string-related problems

Suffix Trie (1)

- The suffix trie of a string X is the trie of all the suffixes of X



your roots to success...

THANK YOU