**The Anatomy of IoT Security: Threats, Vulnerabilities, and Case Studies**

**----- Bhavesh M. Dhake**

**Index :**

**What is IOT ………………………………………………………………………………………….2**

**Examples of IOT…………………………………………………………………………………….2**

**Vulnerabilities in IOT ……………………………………………………………………………4**

**Common IOT attack vectors………………………………………………………………….5**

**IOT attack case studies………………………………………………………………………...6**

**Impact …………………………………………………………………………………………………7**

**Motivation …………………………………………………………………………………………..8**

**Technical approach(lite)……………………………………………………………………….9**

**Part 2 (advance)……………………………………………………………………………………16-47**

**WHAT IS IOT ?(source: techtarget.com)**

* The internet of things, or IoT, is a network of interrelated devices that connect and exchange data with other IoT devices and the cloud. [IoT devices](https://www.techtarget.com/iotagenda/definition/IoT-device) are typically embedded with technology such as [sensors](https://www.techtarget.com/whatis/definition/sensor) and software and can include mechanical and digital machines and consumer objects.

These devices encompass everything from everyday household items to complex industrial tools. Increasingly, organizations in a variety of industries are using IoT to operate more efficiently, deliver enhanced customer service, improve decision-making and increase the value of the business.

With IoT, data is transferable over a network without requiring human-to-human or human-to-computer interactions.

**========================================================**

**Examples :- (source: chatgpt.com)**

1. Smart Home Devices

- Amazon Echo (Alexa) and Google Nest Hub: These smart speakers and displays serve as central hubs for controlling other IoT devices through voice commands, from lights to thermostats.

- Smart Thermostats: Devices like Nest Thermostat and Ecobee learn from user habits to optimize home heating and cooling, saving energy.

- Ring Video Doorbell: Provides real-time video monitoring and alerts for home security.

- Philips Hue and LIFX Smart Lighting: Allow users to control lighting, set schedules, and change colors through mobile apps or voice commands.

- Smart Locks: Brands like August and Yale enable keyless entry and remote control over home locks.

2. Wearable Devices

- Apple Watch and Fitbit: Popular fitness trackers and smartwatches monitor health metrics, track activity, and provide alerts.

- Oura Ring: A ring-style wearable tracking sleep, activity, and other health data.

- WHOOP Strap: A health-focused wearable tracking recovery, strain, and sleep, popular with athletes.

3. Healthcare Devices

- Continuous Glucose Monitors (CGMs): Devices like Dexcom and FreeStyle Libre offer real-time glucose monitoring for diabetes management.

- Smart Inhalers: Devices like Propeller connect with inhalers to monitor and improve asthma and COPD management.

- Smart ECG Monitors: KardiaMobile and Apple Watch can provide medical-grade ECG readings and alerts.

4. Industrial IoT (IIoT) Devices

- Smart Sensors for Predictive Maintenance: Sensors from companies like Siemens and Bosch monitor machinery health, predict failures, and reduce downtime.

- Industrial Robots: Collaborative robots (cobots) from Universal Robots assist in manufacturing by working safely alongside humans.

- Smart Asset Tracking: GE Predix and Cisco Kinetic platforms provide real-time asset tracking for supply chains.

5. Agriculture IoT Devices

- Smart Irrigation Systems: Devices like Hydrawise optimize water use based on real-time weather data.

- Livestock Monitoring: Solutions like CowManager use ear tags to monitor cattle health, location, and behavior.

- Soil Sensors: IoT soil monitoring devices from CropX track moisture, temperature, and nutrients.

6. Connected Cars and Automotive IoT

- Tesla Autopilot: An advanced driver assistance system for semi-autonomous driving and real-time updates.

- OBD-II IoT Adapters: Plug-ins like Automatic and Zubie provide vehicle diagnostics, trip data, and location tracking.

- Vehicle-to-Everything (V2X): Sensors from companies like Qualcomm enable vehicles to communicate with each other and infrastructure for safety and traffic efficiency.

7. Smart City Devices

- Smart Street Lights: Lights from Philips CityTouch adjust brightness based on real-time needs, reducing energy costs.

- Traffic Management Systems: IoT-enabled cameras and sensors analyze traffic flow to optimize signals, reduce congestion, and improve safety.

**========================================================**

**Vulnerabilities in IOT : (source : fortinet.com)**

A vulnerable device can risk [IoT security](https://www.fortinet.com/resources/cyberglossary/iot-security) by giving cyber criminals access to connected networks, enabling them to steal critical corporate data and user credentials. Organizations therefore must understand how to secure IoT devices and recognize the top IoT vulnerabilities they face.

Top IOT vulnerabilities :

**1. Weak/hardcoded passwords**

Weak or hardcoded passwords are among the most frequent methods attackers use to compromise IoT devices. Weak and reused passwords, which are short or easy to guess, are simple for attackers to crack, which they then use to compromise devices and launch large-scale attacks.

**2. Insecure networks**

Insecure networks make it easy for cyber criminals to exploit weaknesses in the protocols and services that run on IoT devices. Once they have exploited a network, attackers can breach confidential or sensitive data that travels between user devices and the server. Insecure networks are particularly susceptible to [man-in-the-middle (MITM) attacks](https://www.fortinet.com/resources/cyberglossary/man-in-the-middle-attack), which aim to steal credentials and authenticate devices as part of broader cyberattacks.

**3. Insecure ecosystem interfaces**

Insecure ecosystem interfaces, such as application programming interfaces (APIs) and mobile and web applications, allow attackers to compromise a device. Organizations need to implement [authentication and authorization processes](https://www.fortinet.com/resources/cyberglossary/authentication-vs-authorization) that validate users and protect their cloud and mobile interfaces. Practical identity tools help the server differentiate valid devices from malicious users.

**4. insecure update mechanisms**

Devices with insecure update processes risk installing malicious or unauthorized code, firmware, and software. Corrupt updates can compromise IoT devices, which could be critical for organizations in the energy, healthcare, and industrial sectors. Updates need to be secure and on encrypted channels, while all software must be validated and approved.

**5. Insecure or outdated components**

The IoT ecosystem can be compromised by code and software vulnerabilities and legacy systems. Using insecure or outdated components, such as open-source code or third-party software, can present vulnerabilities that expand an organization’s [attack surface](https://www.fortinet.com/resources/cyberglossary/attack-surface).

**6. Lack of proper privacy protection**

IoT devices often collect personal data that organizations need to securely store and process to comply with various data privacy regulations. Failing to protect this data can lead them open to fines, a loss of reputation, and lost business. Failing to implement sufficient security can lead to [data leaks](https://www.fortinet.com/resources/cyberglossary/data-leak) that jeopardize user privacy.

**7. Insecure data transfer and storage**

Data that IoT devices receive or transmit across networks needs to be secured and restricted from unauthorized users. This is critical to maintaining the integrity and reliability of IoT applications and organizations’ decision-making processes.

**8. Improper device management**

Failing to manage devices properly throughout their lifecycle leaves them open to vulnerability exploitation, even if they are no longer in use. Businesses need to understand which assets or devices are connected to their networks and manage them properly. Unauthorized or inactive devices can provide attackers with access to corporate networks, enabling them to steal or intercept sensitive data. This makes the discovery and identification of IoT devices crucial to monitoring and protecting devices.

**9. Insecure default settings**

IoT devices, like personal devices, ship with default and hardcoded settings that enable simple setup. However, these default settings are highly insecure and easy for attackers to breach. Once compromised, hackers can exploit vulnerabilities in a device’s firmware and launch broader attacks

. This makes them easier for attackers to target and disrupt, manipulate, or sabotage.

against companies.

**10. Lack of physical hardening**

The nature of IoT devices sees them deployed in remote environments instead of controlled situations that are easy to manage

**Common IOT attack vectors : (source : wallarm.com)**

## IOT attack zones :

* **Devices :** Assaults could be sent off principally through gadgets. Memory, firmware, the actual connection point, the web interface, and the organization administrations are for the most part weak parts of a gadget. In addition to other things, aggressors can exploit uncertain default settings, obsolete parts, and unstable update components.
* ‍**Channels of communication :** Assaults against IoT parts can get through the channels that associate them. IoT conventions could have security imperfections that influence the whole framework. Forswearing of administration (DoS) and caricaturing are two notable organization dangers that can influence IoT frameworks.
* **Software and Applications :** Frameworks can be compromised because of imperfections in web applications and related programming for IoT gadgets. Web applications can be utilized to take client qualifications or push noxious firmware refreshes, for instance.

**Common IOT attacks :**

* [**Botnet attack**](https://www.wallarm.com/what/what-is-a-botnet): Digital criminal gatherings can think twice about gadgets associated with the web and use them all at once to complete assaults. By introducing malware on these gadgets, digital lawbreakers can lay hold of them and utilize their aggregate processing ability to take on bigger focuses in IoT DDoS attacks, send spam, take data. If you are wondering which iot devices were used for the ddos attack, the covert operative was done utilizing IoT gadgets with a camera or sound recording capacities. Monstrous botnets comprised of many thousands or even huge number of IoT gadgets have likewise been utilized to do iot botnet attack.
* [**Ransomware**](https://www.wallarm.com/what/explained-ransomware-attack) : Ransomware is a sort of infection that encodes documents or gadgets and holds them prisoner until a payment is paid. IoT attack vectors, then again, seldom have many - if any - documents. Accordingly, a ransomware attack on IoT gadgets is probably not going to deny clients from getting to vital information (which powers the installment of the payment). In view of this, digital crooks undertaking IoT ransomware assaults may rather attempt to lock the actual gadget, which can undoubtedly be scattered by resetting the gadget as well as introducing a fix.
* **Convergence :** Due to the importance IoT plays in the present undertakings, IoT gadgets are intended to be associated with the web. Nonetheless, this association offers an extra assault vector. The predominant procedure of fragmenting savvy frameworks inside their own particular organizations, for instance, just goes such a long ways in modern associations (on the grounds that IoT gadgets are associated with the web). Frameworks that were beforehand air gapped are presently intended to be on the web, regularly over remote organizations, as Internet of Things (IoT) gadgets have acquired in noticeable quality in functional innovation.
* **Unencrypted data :** Due to the capacity centered way to deal with IoT plan, most IoT gadgets come up short on ability to give hearty encryption. In spite of the way that numerous IoT gadgets don't store documents locally, they in all actuality do send vital telemetry information (like video or sound information) back to organizations or to the cloud. That traffic is especially defenseless against listening in, surveillance, and capturing assuming there are no solid encryption norms set up. Aggressors may, for instance, change camera takes care of or keep them from recording, or adjust touchy clinical or customer information.

**IOT attack case studies : (source : cm-alliance.com)**

Hackers have the power to launch assaults and enter thousands or millions of unprotected connected devices, destroying infrastructure, taking down networks, or accessing confidential data. Here are some of the most illustrative cyber attacks demonstrating IoT vulnerabilities:

* The Mirai Botnet : An IoT botnet (a network of computers, each of which runs bots) was used to execute the worst DDoS attack against Internet performance management services provider Dyn back in October 2016. As a result, several websites went offline, including majors like CNN, Netflix, and Twitter.   
    
  After becoming infected with Mirai malware, computers continuously search the web for susceptible IoT devices before infecting them with malware by logging in using well-known default usernames and passwords. These gadgets included digital cameras and DVR players, for example.
* The Verkada hack : [Verkada](https://www.verkada.com/), a cloud-based video surveillance service, was hacked in March 2021. The attackers could access private information belonging to Verkada software clients and access live feeds of over 150,000 cameras mounted in factories, hospitals, schools, prisons, and other sites using legitimate admin account credentials found on the internet.  
    
  Over 100 employees were later found to have "super admin" privileges, enabling them access to thousands of customer cameras, revealing the risks associated with over privileged users.

###  Cold in Finland : In November 2016, cybercriminals turned off the heating in two buildings in the Finnish city of Lappeenranta. After that, another DDoS assault was launched, forcing the heating controllers to reboot the system repeatedly, preventing the heating from ever turning on. This was a severe attack since Finland experiences severely low temperatures at that time of year.

###  The Jeep Hack : In July 2015, a group of researchers tested the security of the Jeep SUV. They managed to take control of the vehicle via the Sprint cellular network by taking advantage of a firmware update vulnerability. They could then control the vehicle’s speed and even steer it off the road.

###  Stuxnet : Stuxnet is probably the most well-known IoT attack. Its target was a uranium enrichment plant in Natanz, Iran. During the attack, the Siemens Step7 software running on Windows was compromised, giving the worm access to the industrial program logic controllers. This allowed the worm's developers to control different machines at the industrial sites and get access to vital industrial information. The first indications of a problem with the nuclear facility's computer system surfaced in 2010. When IAEA inspectors visited the Natanz plant, they saw that a strangely high percentage of uranium enrichment centrifuges were breaking. Multiple malicious files were later found on Iranian computer systems in 2010. It was discovered that the Stuxnet worm was included in these malicious files.  Iran hasn't provided detailed information on the attack's results, but the Stuxnet virus is believed to have damaged 984 uranium-enrichment centrifuges. According to estimates, this resulted in a 30% reduction in enrichment efficiency.

### ==========================================================

### Impact : (source : techradar.com)

Botnet-driven DDoS attacks cause “billions of dollars” in financial losses across industries, around the world, the report claims. Furthermore, by DDoSing OT devices, critical industrial processes can be disrupted, possibly even putting human lives at risk.

More than half of IoT device traffic comes from manufacturing and retail companies (52%), with 3D printers, geolocation trackers, industrial control devices, automotive multimedia systems, data collection terminals, and payment terminals sending the majority of signals over digital networks.

At the same time, the manufacturing sector experiences 6,000 IoT malware attacks every week, on average. Another sector that can’t catch its breath due to a constant barrage of malware attacks is education. This is mostly because the education industry stores vast amounts of sensitive information that cybercriminals can leverage in different ways. IoT malware attacks in the education sector increased by nearly 1000%, the report claims.

Most infections for the year - 46% - happened in Mexico, followed by Brazil and Colombia (in no particular order). Almost all of the IoT malware (96%) is distributed from compromised IoT devices in the United States.

### ========================================================

### Motivation : (source : coretech.us)

#### 1. Financial Gain

The primary motivation of a hacker is money, and getting it can be done with a variety of methods.

They could directly gain entry to a bank or investment account; steal a password to your financial sites and then transfer the assets over to one of their own; swindle an employee into completing a money transfer through a complicated spear phishing technique, or conduct a ransomware attack on your entire organization.

The possibilities are endless, but most hackers are out to make a profit.

#### 2. Recognition & Achievement

Some hackers are motivated by the sense of achievement that comes with cracking open a major system. Some may work in groups or independently, but, on some scale, they would like to be recognized.

This also ties into the fact that cyber criminals are competitive by nature, and they love the challenge their actions bring. In fact, they often drive one another to complete more complicated hacks.

#### 3. Insider Threats

Individuals who have access to critical information or systems can easily choose to misuse that access—to the detriment of their organization.

These threats can come from internal employees, vendors, a contractor or a partner—and are viewed as some of the greatest cyber security threats to organizations.

However, not all insider threats are intentional, according to an [Insider Threat Report](https://crowdresearchpartners.com/wp-content/uploads/2017/07/Insider-Threat-Report-2018.pdf) from Crowd Research Partners. Most (51%) are due to carelessness, negligence, or compromised credentials, but the potential impact is still present even in an unintentional scenario.

#### 4. Political Motivation “Hacktivism”

Some cyber criminal groups use their hacking skills to go after large organizations. They are usually motivated by a cause of some sort, such as highlighting human rights or alerting a large corporation to their system vulnerabilities. Or, they may go up against groups whose ideologies do not align with their own.

These groups can steal information and argue that they are practicing free speech, but more often than not, these groups will employ a DDoS ([Distributed Denial of Service](https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/)) attack to overload a website with too much traffic and cause it to crash.

#### 5. State Actors

State-sponsored actors receive funding and assistance from a nation-state. They are specifically engaged in cyber crime to further their nation’s own interests. Typically, [they steal information](https://www.rand.org/content/dam/rand/pubs/testimonies/CT400/CT490/RAND_CT490.pdf), including “intellectual property, personally identifying information, and money to fund or further espionage and exploitation causes.”

However, some state-sponsored actors do conduct damaging cyberattacks and claim that their cyberespionage actions are legitimate activity on behalf of the state.

#### 6. Corporate Espionage

This is a form of cyber attack used to gain an advantage over a competing organization.

Conducted for commercial or financial purposes, corporate espionage involves:

* Acquiring property like processes or techniques, locations, customer data, pricing, sales, research, bids, or strategies
* Theft of trade secrets, bribery, blackmail, or surveillance.

==================================================================================

**Technical approach : (source : Sciencedirect.com)**

**1. Programming Languages Used in IoT Hacking**

* **Python**: Widely used for developing scripts and tools to automate tasks, exploit vulnerabilities, and create malware targeting IoT devices. Python’s extensive libraries support network analysis and device scanning, making it a popular choice for IoT-focused attackers.
* **C/C++**: Many IoT devices run on firmware built with C or C++. Attackers use these languages to exploit firmware(**program embedded into hardware devices to help them operate effectively)** vulnerabilities, develop malware, and reverse-engineer firmware to understand device functions and find security flaws.
* **JavaScript**: In some web-enabled IoT devices, JavaScript is used to exploit vulnerabilities in web interfaces. Attackers may also use it in cross-site scripting (XSS) attacks on IoT device web dashboards.
* **Assembly Language**: Some IoT hackers delve into assembly language to exploit lower-level vulnerabilities, especially when working with constrained devices or directly manipulating device hardware.

**2. Hacking Methods and Techniques**

* **Reverse Engineering and Firmware Analysis**:

Attackers often download or extract **firmware** from IoT devices to perform reverse engineering and uncover vulnerabilities that could be exploited. Firmware is the software embedded in the hardware of IoT devices, controlling their functions and operations. By analyzing firmware, hackers can identify weak spots, such as hardcoded credentials, outdated libraries, or poorly implemented encryption protocols, which could be leveraged for unauthorized access or attacks.

* **Extraction Techniques**: Firmware can be obtained from **device manufacturers’ websites**, through **public repositories**, or by physically accessing the device and dumping the firmware directly from the device's storage. Tools like **Binwalk** are commonly used for extracting and analyzing firmware files.
* **Vulnerabilities in Firmware**: Once firmware is extracted, attackers search for flaws such as buffer overflows, improper memory handling, or insecure communication protocols. These weaknesses can be exploited to gain access to the device, inject malicious code, or escalate privileges.
* **Reverse Engineering**: By using reverse engineering techniques, attackers can decompile or disassemble the firmware to understand its underlying code and logic. This allows them to find flaws or hidden backdoors that may not be immediately obvious in the device's public interface.
* Tools like **IDA Pro** or **Ghidra** are commonly used for reverse engineering, while **binwalk** is popular for analyzing firmware files.
* **Vulnerability Scanning**:

Hackers use a variety of tools and techniques to scan IoT networks for vulnerabilities such as open ports, weak passwords, and unpatched devices. These vulnerabilities are common entry points for attackers seeking to exploit IoT devices.

* **Open Ports**: IoT devices often have multiple communication ports open for legitimate purposes. However, if these ports are left open without proper security controls, they can become targets for hackers. Tools like **Nmap** and **Shodan** are often used to scan networks for these open ports. Once identified, attackers can attempt to access these ports and gain control over the devices.
* **Weak Passwords**: Many IoT devices come with default or easily guessable passwords (e.g., "admin" or "123456"). Attackers often use **brute-force attacks** or **dictionary attacks** to guess these passwords. Tools such as **Hydra** and **Medusa** can be used to automate password guessing attempts on IoT devices with weak authentication.
* **Unpatched Devices**: Manufacturers release patches and updates to fix vulnerabilities, but many IoT devices are not regularly updated by users or administrators. Attackers take advantage of this by using tools like **Metasploit** or **OpenVAS** to scan for devices with outdated firmware or software that could be vulnerable to known exploits.
* **Nmap** and **Shodan** (a search engine for IoT devices) are popular tools for finding and mapping vulnerable devices connected to the internet.
* **Weak Credential Exploits**:
  + Many IoT devices are shipped with **default usernames and passwords** that users often forget to change. These default credentials are widely known and can be easily found in public databases, online forums, or device manuals. Because many people neglect to update them, these devices become easy targets for attackers.
  + Hackers use tools like **brute-force** and **credential-stuffing attacks** to gain access to these devices. In a **brute-force attack**, the attacker uses software to try a large number of possible passwords until the correct one is found. Tools like **Hydra** or **Medusa** can automate this process, making it much faster and more efficient. For **credential-stuffing**, attackers use databases of known username-password pairs (often obtained from previous data breaches) and try them across many devices, hoping to find a match. This is effective because many users reuse passwords across multiple platforms.
  + Once attackers gain access to IoT devices, they can take control of the devices, steal data, or use the device as part of a larger botnet. This is a major reason why IoT security is so important—if users don't secure their devices, they risk exposing them to these kinds of attacks
  + Tools like **Hydra** and **Medusa** automate brute-force attacks, testing a large number of password combinations on IoT devices.
* **Botnets and Distributed Denial of Service (DDoS)**:

Once **IoT devices** are compromised, they can be hijacked by attackers and incorporated into a **botnet**—a network of infected devices controlled remotely to perform malicious tasks. In many cases, **botnets** are used for **Distributed Denial-of-Service (DDoS)** attacks, where the attacker floods a target with a massive amount of traffic, overwhelming the system and causing it to become slow, unresponsive, or completely offline.

* **How IoT Devices Become Part of a Botnet**: When attackers gain access to IoT devices, often through weak passwords, default credentials, or unpatched vulnerabilities, they install malicious software (malware) that allows them to control the device remotely. These devices can include **cameras**, **routers**, **smart thermostats**, and **voice assistants**, all of which are often connected to the internet and lack robust security features.
* **DDoS Attacks Using Botnets**: After compromising numerous IoT devices, the attacker can command the botnet to send huge volumes of traffic to a specific target, such as a website or server, with the intent to exhaust the target's resources (bandwidth, processing power). For example, the **Mirai Botnet** in 2016 exploited IoT devices like cameras and DVRs, using them to launch one of the largest DDoS attacks ever recorded. This attack, aimed at **Dyn**, a major DNS provider, disrupted major websites including Twitter, Netflix, and Reddit​
* **Tools for Botnet Creation**: Hackers typically use botnet software to control compromised IoT devices. This software can issue commands to each infected device, enabling them to participate in the DDoS attack. Once part of a botnet, IoT devices work together to send synchronized requests, making it difficult for the target to block the attack.
* Attackers often use pre-existing malware (like Mirai) that can automatically scan and infect vulnerable devices to build botnets.
* **Cross-Site Scripting (XSS) and Injection Attacks**:

Some **IoT devices** have **web interfaces** that allow users to configure and control them via a browser. However, these interfaces can be vulnerable to common web application attacks like **Cross-Site Scripting (XSS)** and **SQL Injection**, which can give attackers the ability to manipulate the device or steal sensitive data.

1. **XSS (Cross-Site Scripting)**: In this attack, hackers inject malicious scripts into the web interface of an IoT device. If the device does not properly sanitize user input, the malicious script can be executed in the browser of anyone accessing the device’s web interface. This could allow the attacker to:
   * **Steal session cookies** or other sensitive data from users.
   * **Redirect users** to phishing sites.
   * **Execute unauthorized actions** on the device, such as changing settings or even gaining full control over it.

XSS vulnerabilities are particularly dangerous because they can affect any user interacting with the compromised interface, regardless of their device or location. Attackers often use automated tools to scan IoT devices for XSS vulnerabilities, especially in those with poorly designed or outdated web interfaces​

1. **SQL Injection**: SQL injection occurs when an attacker manipulates a web interface to insert malicious SQL queries into the backend database of the IoT device. These attacks exploit vulnerabilities in the way the device handles user input in database queries. For instance:
   * Attackers can **extract sensitive data** such as usernames, passwords, or even private device information stored in the device's database.
   * Attackers might **alter or delete data** stored on the device, disrupting its normal functioning or causing it to behave unpredictably.
   * In some cases, SQL injection could allow the attacker to **gain administrative access** to the device, potentially taking full control.

Many IoT devices use lightweight web applications with databases to store configuration and usage data. If these applications fail to properly validate or sanitize user inputs before interacting with the database, they become vulnerable to SQL injection

Tools like **Burp Suite** and **OWASP ZAP** are used to find and exploit web application vulnerabilities in IoT dashboards or admin panels.

* **Man-in-the-Middle (MitM) Attacks**:

Many **IoT devices** communicate over **unsecured protocols** like **TCP**, **HTTP**, and **UDP**, which are not encrypted or protected from eavesdropping and manipulation. This makes it easier for attackers to intercept, alter, or inject malicious data into the communication between devices, leading to a range of security risks.

* **Unencrypted Communication**:
  + **TCP** and **HTTP** are commonly used communication protocols in IoT devices, but they often lack built-in encryption or authentication mechanisms. This means that data sent over these protocols can be intercepted easily by attackers using tools like **Wireshark** or **Tcpdump** to capture unencrypted traffic.
  + For example, sensitive data such as **user credentials**, **device configurations**, or even **personal information** could be exposed if transmitted over HTTP without **SSL/TLS encryption**. Hackers can simply listen to the traffic on the network and gain access to this data.
* **Man-in-the-Middle (MITM) Attacks**:
  + If an attacker is able to position themselves between two communicating IoT devices or between a device and a server, they can perform a **MITM attack**. This means they can intercept and modify the data being sent, potentially injecting malicious commands or altering legitimate ones.
  + For instance, in a smart home system, an attacker could manipulate commands sent from a smartphone to a thermostat, altering its temperature settings or even disabling security devices like smart locks.
* **Exploiting Protocol Weaknesses**:
  + Many IoT devices use **UDP** or **non-secure HTTP** for **fast communication**. However, these protocols do not have the robust security measures found in **HTTPS** or **TLS-encrypted communication**, making them particularly vulnerable. Attackers can exploit the lack of encryption to **inject malicious data**, **spoof device identities**, or perform **denial-of-service** attacks.
  + **HTTP** vulnerabilities, such as **session hijacking** or **cookie theft**, can also enable attackers to impersonate authorized users and gain control over IoT devices remotely.
  + **Wireshark** and **Ettercap** are tools commonly used to perform MitM attacks, intercepting and analyzing network traffic to capture sensitive data or inject malicious commands.

* **Malware Development**:

As IoT devices proliferate, they present new attack surfaces that malicious actors can exploit. One common approach is for attackers to create **IoT-specific malware**, such as **botnets** or **spyware**, to gain control over devices or monitor user activity. These types of malware are tailored specifically to the limitations and vulnerabilities inherent in IoT systems, which often lack strong security controls.

1. **Botnets**: A **botnet** is a network of compromised IoT devices that are controlled remotely by an attacker. Once infected, these devices can be used to carry out a variety of malicious activities, such as **distributed denial-of-service (DDoS)** attacks. The most well-known example is the **Mirai botnet**, which exploited weak or default passwords in IoT devices like cameras, routers, and DVRs. Once hijacked, these devices could be used to flood a target with overwhelming amounts of traffic, causing websites or services to crash​. Attackers may modify existing botnet malware or develop new variants to exploit specific vulnerabilities found in different types of IoT devices.
2. **Spyware**: **IoT spyware** is another form of malware specifically designed to monitor users without their knowledge. This type of malware can silently record sensitive information such as user activities, conversations, or even keystrokes. For example, an IoT-enabled **smart camera** or **voice assistant** could be compromised to **record audio** or **transmit video** to an external server, violating the privacy of users.

**3. Networking Protocols Exploited in IoT**

**MQTT and CoAP Protocols** : **MQTT** (Message Queuing Telemetry Transport) and **CoAP** (Constrained Application Protocol) are lightweight protocols used by IoT devices to communicate. While they are efficient for low-power, low-bandwidth environments, they often lack proper **encryption** and **authentication**. This makes the data they exchange vulnerable to **interception** and **injection attacks**, where attackers can modify or steal sensitive information transmitted between devices.

**HTTP and WebSocket Vulnerabilities**: Many IoT devices use **HTTP** and **WebSocket** to send data. Without encryption (like **HTTPS**), these communications are susceptible to **man-in-the-middle** attacks, where attackers can intercept or modify data. This can lead to issues like **data theft** or unauthorized changes to device settings, posing a major security risk.

**Bluetooth and Zigbee Attacks**: **Bluetooth** is widely used in IoT devices, but many implementations have **poor security**. Attackers can exploit weak pairing protocols to **intercept data** or **spoof devices**, allowing them to take control of the devices.

**Zigbee**, used for smart home devices, also has security gaps. Although it uses encryption, attackers can exploit weak default keys or poor **authentication** to **join networks** and control devices, making it easier to compromise smart home systems.

**4. Advanced Techniques in IoT Hacking**

* **Binary Exploitation and Buffer Overflow Attacks**:

In In IoT devices, **memory allocation** is a critical part of how the device stores and manages data. However, many IoT devices have **poorly handled memory allocation**, which means that when data is stored or processed, it may not be properly controlled, leading to issues such as **buffer overflows**. A **buffer overflow** happens when a program writes more data to a block of memory (a "buffer") than it can hold, causing the extra data to overwrite adjacent memory.

This **overflow** can cause unexpected behavior, like crashing the device or allowing an attacker to take control of it. In many cases, attackers exploit this flaw to execute **arbitrary code** on the device, meaning they can run commands or programs of their choice on the compromised system. This is particularly dangerous for IoT devices, as it could allow attackers to gain full control over the device and manipulate it remotely.

**Exploiting Buffer Overflows:**

A **buffer overflow** happens when a program writes more data to a block of memory (a "buffer") than it can hold, causing the extra data to overwrite adjacent memory.This **overflow** can cause unexpected behavior, like crashing the device or allowing an attacker to take control of it. In many cases, attackers exploit this flaw to execute **arbitrary code** on the device, meaning they can run commands or programs of their choice on the compromised system.

Attackers often use **advanced techniques** like **Return-Oriented Programming (ROP)** or **Jump-Oriented Programming (JOP)** to exploit buffer overflows. These techniques are used to bypass security mechanisms that are designed to prevent malicious code from running directly on the device.

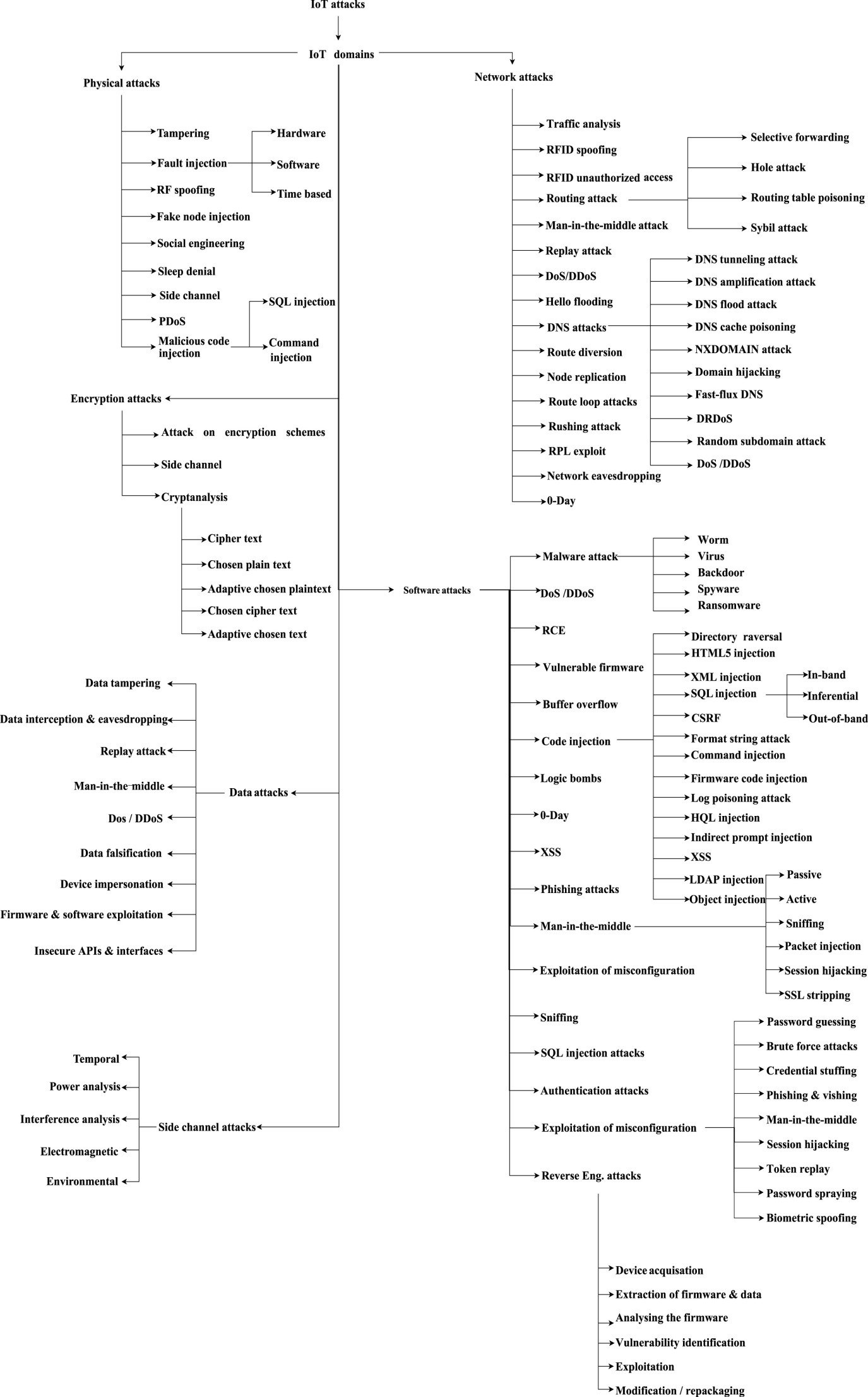
1. **Return-Oriented Programming (ROP)**:
   * In **ROP**, attackers don't inject their own code into the device's memory. Instead, they use small chunks of existing code (called **gadgets**) that are already present in the device’s firmware. By **chaining** these gadgets together, the attacker can execute arbitrary commands without needing to inject new malicious code.
   * This technique is particularly effective because it allows attackers to bypass security mechanisms like **Data Execution Prevention (DEP)**, which normally prevents the execution of code in certain regions of memory.
2. **Jump-Oriented Programming (JOP)**:
   * **JOP** is similar to ROP, but instead of using return instructions to control the flow of execution, it uses **jump instructions** that direct the device to execute code in unintended ways. Attackers use this method to **manipulate program flow**, making it difficult for security mechanisms to detect and stop the attack.
   * Just like ROP, JOP allows attackers to take control of the device by making it run unintended instructions.

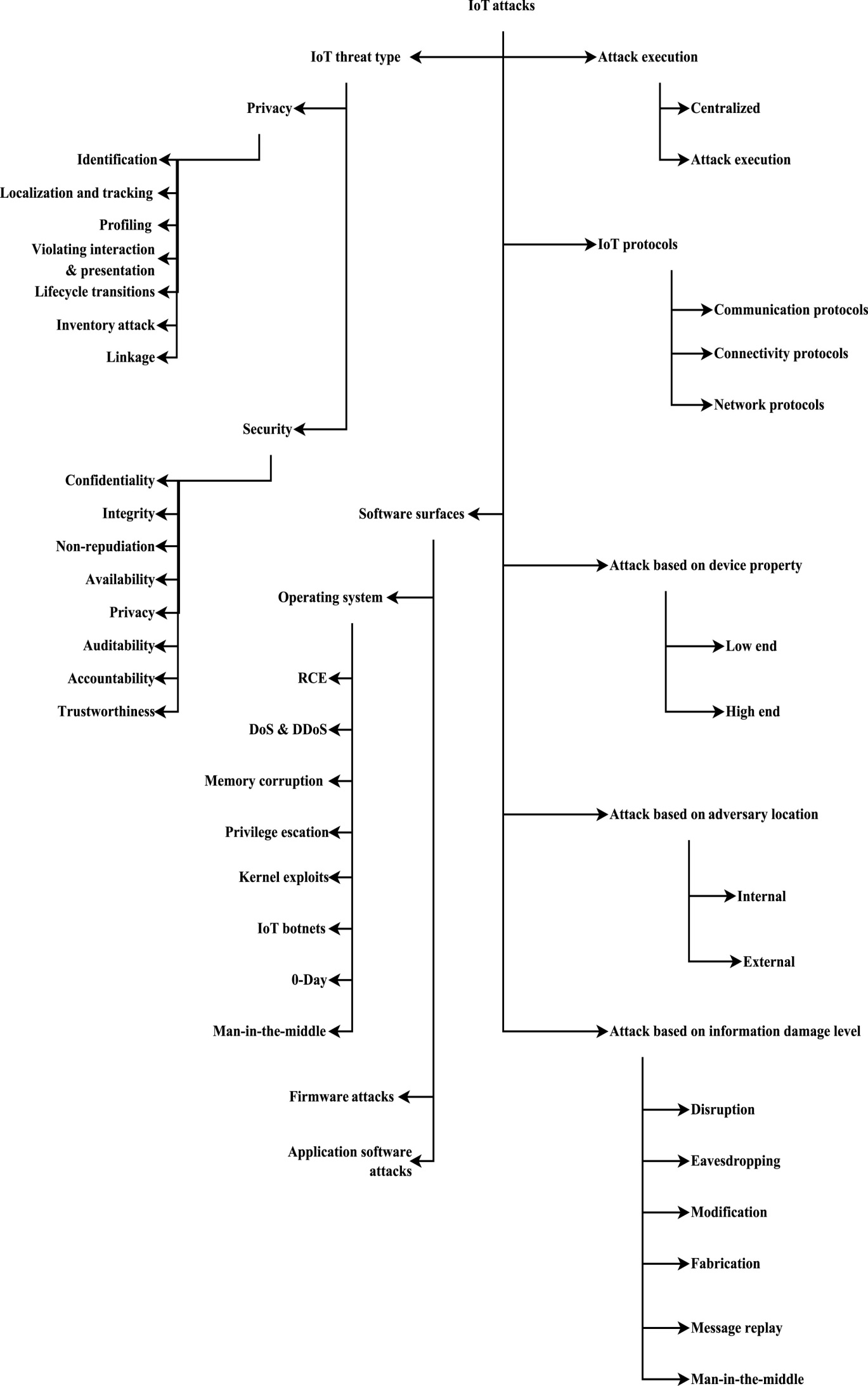
==================================================================

**Part 2 :**

**For Cybersecurity Proffesionals -**

**(Offensive/Red-Team/Penetration-Testers/Ethical-Hackers) :**





#### 7.1.1. Physical attacks

Physical attacks can be launched if the attacker remains physically close to the network or devices of the system [[2](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib2)]. New IoT vulnerabilities are frequently found through physical attacks. The attacker will attempt to physically access the device before launching the attack by purchasing a duplicate of the targeted IoT device from the market. They would then develop a false attack “test” using reverse engineering to determine what kind of outputs might be acquired from it. These physical attacks expose the system's vulnerabilities [[49](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib49)].

Tampering: It describes the process of physically altering a device (such as an RFID) or communication link [[2](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib2)].

[*Fault Injection*](https://www.sciencedirect.com/topics/computer-science/fault-injection): The act of forcing a running device to behave differently to unearth new security details or methods in the system is known as fault injection. It is a technique for evaluating IoT systems and devices to determine their dependability and resilience. It entails purposefully [introducing defects](https://www.sciencedirect.com/topics/computer-science/introducing-defect) or flaws into a system to watch how it responds and find possible weaknesses. Researchers and security experts frequently employ this method to comprehend how a product or system performs under various circumstances and spot vulnerabilities an attacker may exploit [[50](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib50)]. Some of the examples are:

Hardware Fault Injection: It involves inducing faults directly into the physical components of the IoT device, such as memory corruption or voltage manipulation [[50](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib50)].

[*Software Fault*](https://www.sciencedirect.com/topics/computer-science/software-fault) Injection: It involves injecting faults into the software of the IoT device, such as modifying data values, altering control flow, or injecting exceptions [[50](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib50)].

Time-Based Fault Injection: It involves manipulating the timing of events to create faults, such as introducing delays or modifying clock frequencies [[51](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib51)].

RF Spoofing/Jamming: An attacker launches DoS attacks against [RFID](https://www.sciencedirect.com/topics/computer-science/radio-frequency-identification) tags by creating and transmitting [noise signals](https://www.sciencedirect.com/topics/computer-science/noise-signal) instead of radio frequency (RF) signals to obstruct connectivity [[52](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib52)].

Fake Node Injection: A fake node is dropped by an attacker between two actual network nodes to manipulate data flow between them.

Social Engineering: Social engineering obtains sensitive data by deceiving an IoT system's users.

Sleep [*Denial Attack*](https://www.sciencedirect.com/topics/computer-science/denial-attack): By providing the battery-powered devices with incorrect inputs, the attacker keeps them awake. This wears out their batteries, resulting in a shutdown. The attack's goal is to keep the target devices busy by overtaxing the network with routing traffic due to fictitious requests for actual or imagined destination devices, which prevents them from responding to legitimate requests. This attack prevents the targeted device from entering sleep mode to extend battery life [[53](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib53)].

Side Channel Attacks: As mentioned in the Section [7.1.6](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "sec7.1.6).

Permanent [*Denial of Service*](https://www.sciencedirect.com/topics/computer-science/denial-of-service) (PDoS): A PDoS attack, also known as a permanent denial of service attack, is similar to a DoS attack in that hardware is destroyed or sabotaged instead of resources. Rebooting the device won't help recover the IoT device [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib17)]. Permanent DoS attacks were concisely described in work [[54](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib54)], which identifies the different IoT attacks in IoT wireless protocols such as [BLE](https://www.sciencedirect.com/topics/computer-science/bluetooth-low-energy), LoRaWAN, Z-Wave, and [ZigBee](https://www.sciencedirect.com/topics/computer-science/zigbee) along with their vulnerabilities.

[*Malicious Code*](https://www.sciencedirect.com/topics/computer-science/malicious-code) Injection: It is an attack strategy that adversaries use to undermine the security of IoT devices. It involves injecting malicious code or commands into a system to gain unauthorized access, steal data, disrupt operations, or achieve other malicious objectives. [Code injection attacks](https://www.sciencedirect.com/topics/computer-science/code-injection-attack) are typically executed through vulnerabilities in the software or firmware of the IoT devices [[55](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib55)].

* Common types of malicious code injection attacks include:

[*SQL*](https://www.sciencedirect.com/topics/computer-science/structured-query-language) Injection: It exploits vulnerabilities in web applications to insert malicious SQL statements into the database, potentially leading to data theft or unauthorized access [[56](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib56)].

Command Injection: It involves injecting malicious commands into the system's command-line interface, allowing attackers to [execute arbitrary code](https://www.sciencedirect.com/topics/computer-science/execute-arbitrary-code) [[56](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib56)].

#### 7.1.2. Network attacks

IoT devices use the network layer to send data to a server or other devices for processing after receiving it from the [physical layer](https://www.sciencedirect.com/topics/computer-science/physical-layer). To damage IoT network systems, network attacks are carried out by manipulating them. Without being near the network, it may be deployed with ease. Network attacks are a subset of cyberattacks that target the communication infrastructure of IoT devices. The attacks concentrate on taking advantage of flaws in communication channels, network protocols, and device connections to threaten the security, availability, and integrity of the IoT ecosystem. IoT network attacks may result in catastrophic outcomes, including unauthorized access, data breaches, service interruptions, and, in certain circumstances, even physical harm [[2](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib2),[28](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib28)].

Some of the network attacks are:

Traffic Analysis Attack: It is a type of cyberattack that focuses on analyzing the patterns, volume, and behaviour of data traffic between IoT devices and the network infrastructure. Such attacks aim to extract [sensitive information](https://www.sciencedirect.com/topics/computer-science/sensitive-informations) or infer valuable insights from the communication patterns without directly intercepting the content of the data packets. By understanding the data flow, an attacker can deduce sensitive information about the IoT devices, users, or the network [[28](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib28)].

RFID Spoofing: RFID spoofing, also known as RFID cloning or RFID emulation, is a technique used to deceive radio frequency identification (RFID) systems by mimicking the signals of legitimate RFID tags or devices. In an RFID [spoofing attack](https://www.sciencedirect.com/topics/computer-science/spoofing-attack), an adversary replicates or manipulates RFID signals to impersonate a valid RFID tag or device. This allows the attacker to gain unauthorized access, bypass security measures, or perform fraudulent activities. RFID spoofing aims to trick the RFID reader into recognizing the attacker's spoofed signal as a genuine RFID tag, enabling them to exploit the system for malicious purposes [[57](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib57)].

RFID Unauthorized Access: RFID unauthorized access refers to gaining entry to a system, facility, or sensitive information by exploiting vulnerabilities in radio frequency identification (RFID) technology. Unauthorized access occurs when an individual or attacker, without proper authorization or credentials, finds a way to exploit weaknesses in the RFID system to gain entry or access sensitive information. This can have profound security implications, allowing unauthorized parties to bypass physical barriers and security checkpoints or gain control over valuable assets [[55](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib55)].

Routing Attacks/Routing Information Attacks: IoT routing attacks are a subset of attacks that target the routing architecture and protocols of IoT networks. These attacks are designed to prevent data packets from being appropriately routed between IoT devices and other network elements, which might result in data interception, manipulation, or denial of service. Attackers who get access to the routing systems can reroute, delay, or discard data packets, seriously impairing the IoT system's ability to communicate and function [[58](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib58)].

Selective Forwarding: In a selective forwarding attack, the attacker selects some data packets to reject or forward while letting others pass. By controlling the flow of data, the attacker can selectively block critical information or let malicious data through [[58](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib58)].

Hole Attacks: “Hole-196” attacks are a specific kind of cyberattack called “hole attacks”. The vulnerability in the RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) routing protocol, which is used in many IoT networks, gave rise to the name “Hole-196.”

* + The RPL protocol, intended to facilitate effective routing on low-power and lossy networks, such as those frequently present in IoT environments, has a weakness exploited by “Hole-196” attacks. The attack uses the protocol's capacity to form a network “black hole”, seriously disrupting communication [[59](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib59)].

[*Sinkhole Attacks*](https://www.sciencedirect.com/topics/computer-science/sinkhole-attack): In a sinkhole attack, the attacker advertises that it has the fastest or shortest route to the target to draw data flow. IoT devices route their data through the attacker's node, unknowing its malicious purpose, allowing it to intercept, modify, or discard the data [[55](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib55)].

Blackhole Attacks: Like a sinkhole attack, it involves the attacker dropping all incoming data packets rather than forging them. The connectivity between IoT devices and the legitimate network infrastructure is interfered with as a result [[55](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib55)].

Routing Table Poisoning (RTP): One of the most [successful attacks](https://www.sciencedirect.com/topics/computer-science/successful-attack) involves installing a fake router or sending malicious routing table updates to change the routing table. The attacker may launch a [MiTM](https://www.sciencedirect.com/topics/computer-science/man-in-the-middle-attack) attack or just divert the traffic to get through FW or IPS in this situation [[60](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib60)].

[*Sybil Attack*](https://www.sciencedirect.com/topics/computer-science/sybil-attack): In a peer-to-peer network, a Sybil attack occurs when an adversary generates fake or stolen identities to act as numerous nodes. An adversary can get excessive control to lower the network's efficiency. Selfish behaviour inside a network can impact data integrity, resource usage, and overall network performance [[60](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib60)].

Man-in-the-middle Attack (MiTM): A man-in-the-middle attack involves the attacker discreetly putting themselves in the user's and the IoT's path. Whenever someone requests information from the Internet, it passes through several routers before getting to the right place. If the data is not encrypted, the attackers can read anything that travels through them by putting themselves in the midst. The information can be altered if the data is encrypted and the attackers get hold of the [encryption key](https://www.sciencedirect.com/topics/computer-science/encryption-key). Another variation of this attack involves the attacker not modifying any data but instead storing the information they learn about their target. Since there is no method for the user to check the accuracy of the information, these attacks are challenging to spot [[29](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib29)].

Replay Attack: A replay attack involves the attacker capturing and subsequently retransmitting genuine data packets sent back and forth between IoT devices. The attacker can use this method to obtain unauthorized access or carry out [malicious activity](https://www.sciencedirect.com/topics/computer-science/malicious-activity) if the data is not sufficiently safeguarded. Attackers utilize replay attacks when they want to delay information delivery to machines. In this attack, a packet carrying a machine's future instructions is intercepted and transmitted laterally. When an attacker intercepts a packet, they also have the power to alter the instructions and trick the system into carrying out the wrong operations [[29](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib29)].

Denial/Distributed Denial of Service (DoS/DDoS) Attack: Multiple devices are utilized to launch a DDoS attack, a kind of DoS attack. These types of attacks are carried out by bombarding the desired target with an excessive number of access requests until they are overwhelmed. A denial of service (DoS) attack involves a malicious attacker trying to saturate the system with rogue and amplified traffic to consume network resources and target legitimate users' CPU time and/or bandwidth. [Botnets](https://www.sciencedirect.com/topics/computer-science/botnets) are networks of Internet-connected devices that are contaminated or under control and used in effective DDoS attacks [[61](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib61)].

Hello-flooding: To introduce themselves to their neighbours, nodes are required by some [WSN](https://www.sciencedirect.com/topics/computer-science/wireless-sensor-network) routing protocols to broadcast hello messages. When a node receives one of these messages, it may believe that the sender is within radio range of it. This assumption, however, may not always be accurate; on occasion, a laptop-class attacker might deceive every other node in the network into believing the attacker is its neighbour by broadcasting routing or additional information with significant transmission power. For example, if an adversary advertises an extremely high-quality route to the base station, many network nodes may seek to use it. But nodes sufficiently remote from the adversary would send the packets into oblivion. The network is consequently left in a state of chaos. This attack mainly affects protocols that rely on localized information sharing between nearby nodes for topology maintenance or flow management. An attacker can use the hello flood attack without creating actual traffic. It can just broadcast overhead packets again with sufficient strength for all other network nodes to receive them [[62](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib62)].

Clone (Node-replication): An attack on IoT devices called a clone node explicitly targets the [authentication system](https://www.sciencedirect.com/topics/computer-science/authentication-system). Device replication or device cloning attacks are other names for clone node attacks. A cloned node behaves like a genuine node. It engages in malicious network-related activities such as selective forwarding attacks, [wormhole attacks](https://www.sciencedirect.com/topics/computer-science/wormhole-attack), and blackhole attacks since it has access to all [secret information](https://www.sciencedirect.com/topics/computer-science/secret-information) (authentication codes and keys) [[63](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib63)].

* In a clone node attack, the attacker can seize physical devices from the IoT network by obtaining their secret credentials, such as IDs and public and private keys. There are several steps to exploiting this vulnerability, which involve capturing the physical device, obtaining the privileged credentials, modifying its function, and placing it back in the network at some desired location. IoT devices frequently lack updated security software and certifications and are developed and manufactured by unreliable security partners, which might contribute to the clone node attack [[63](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib63)].

Routing Diversion/Misdirection Attacks: In a route diversion attack, the enemy succeeds in diverting some current routes but does not stop the formation of new routes. This indicates that the protocol establishes routes different from those it would construct if the adversary did not interfere with the protocol's execution due to the adversary's presence. Route diversion may give attackers more control over communications between select victim nodes. In this scenario, the adversary seeks to ensure that the redirected routes comprise a node under its control or a connection under its observation. The attacker will thus have a simpler time intercepting or changing data exchanged between the victim nodes [[64](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib64)].

* Establishing a tunnel is a particularly effective approach to rerouting routes through nodes controlled by an adversary. Many pairs of communicating nodes might pick the tunnelled paths because they seem shorter, making it more straightforward for the adversary to access their conversations. Increased resource consumption by some nodes may be another goal of route diversion. For instance, the nodes near the two ends of the tunnel will get a higher volume of transit traffic due to the above-described route diversion, necessitating the employment of extra resources at those nodes to forward that traffic. Alternately, the attacker can direct numerous routes straight via a victim node by altering or falsifying routing messages. Last but not least, route diversion may seek to lengthen the already known routes, increasing specific nodes’ end-to-end delays and perhaps degrading their quality of service [[64](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib64)].

Routing Loop Attacks: The routing loop attack relies on the fact that a router does not know whether there is an endpoint that can be reached via its tunnel with the packet's source or destination address. Routing Loop attacks exploit inconsistencies between the native IPv6 routing state and a tunnel's overlay IPv6 routing state. They mainly use that each endpoint in an autonomous tunnel is blind to the other nodes currently a part of it. The attacker takes advantage of this by creating a packet sent over a tunnel to a node not part of that tunnel. This node sends the packet over the tunnel onto a real IPv6 network. The entrance point in that network sends the packet back into the tunnel before routing it again. As a result, the packet will enter and exit the tunnel repeatedly. The nodes that [forward packets](https://www.sciencedirect.com/topics/computer-science/forward-packet) into and out of the tunnel are called the attack's victims. Only when the hop limit field in the packet's IPv6 header is zeroed out can a loop end. 255 is the highest possible value that may be entered into this field [[65](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib65)].

* It should be noted that the hop limit does not reduce when the packet is tunnelled through IPv4 routers. Each step along the attack packet's path will be travelled through 255/N times, where N is the number of IPv6 routers that make up the loop. The loops can, therefore, be utilized as 255/N traffic amplification methods. The type of attack and the positions of the two victims define how many IPv6 routers are in the loop. The amplification ratio will increase the closer the two victims are to one another [[65](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib65)].

Rushing Attacks: This attack aims to add the malicious device to the [routing path](https://www.sciencedirect.com/topics/computer-science/routing-path). So, by swiftly sending a route discovery packet to the forwarding group, the attacker can abuse the network. The attacker can tamper with the packet by using this technique to include himself in the routing table. A Rushing attack, for instance, can only be successful if the attacker can get its altered packets to other genuine devices before these devices get the genuine ones. To do this, the attacker can speed up [packet transmission](https://www.sciencedirect.com/topics/computer-science/packet-transmission) by reducing delays at the MAC or routing layer [[53](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib53)].

RPL Exploit: It is a simple exploit used in IoT today and does not include all of the capabilities of the routing protocols in use. It was primarily designed to be used with data sinks that have multiple-point communications [[66](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib66)].

[*DNS*](https://www.sciencedirect.com/topics/computer-science/domain-name-system) Attacks: Attacks on the DNS (Domain Name System) in the IoT involve taking advantage of vulnerabilities in the DNS infrastructure or protocols to sabotage connectivity between IoT devices or reroute traffic to malicious destinations. A crucial part of the internet is the DNS, which converts human-readable domain names (such as [www.example.com](http://www.example.com)

* ) into [IP](https://www.sciencedirect.com/topics/computer-science/internet-protocol) addresses that computers and IoT devices can comprehend. DNS plays a crucial role in facilitating communication between devices and servers on the Internet. Some DNS attacks are:

DNS Tunnelling Attack: Encoding data from other programs or protocols within DNS queries and responses is known as DNS tunnelling. Typically, it contains data payloads that can hijack a DNS server and give attackers control over the remote server and its applications. DNS tunnelling frequently relies on a hacked system's external network connectivity as a backdoor into an internal DNS server with network access. Controlling a domain, which serves as an authoritative server and executes server-side tunnelling and data payload [executable programs](https://www.sciencedirect.com/topics/computer-science/executable-program), is also necessary [[67](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib67)].

DNS [*Amplification Attack*](https://www.sciencedirect.com/topics/computer-science/amplification-attack): DNS amplification exploits cause a targeted server to experience distributed denial of service (DDoS). This involves exploiting open DNS servers that are publicly available to overwhelm a target with [DNS response](https://www.sciencedirect.com/topics/computer-science/system-response) traffic. Typically, the threat actor sends a DNS query request to the open DNS server as the first step in an attack, faking the source address to become the target address. Once the DNS server returns the DNS record response, it is passed to the new target, which the attacker controls [[67](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib67)].

DNS Flood Attack: DNS flood attacks employ the DNS protocol to conduct a [user datagram protocol](https://www.sciencedirect.com/topics/computer-science/user-datagram-protocol) (UDP) flood. Threat actors launch valid (but fake) DNS request packets at a very high packet rate, generating a massive group of source IP addresses. The target's DNS servers respond to all queries since they appear genuine. The enormous volume of queries may then cause the DNS server to crash. A DNS attack uses many network resources, wearing down the targeted DNS infrastructure until it is shut down. As a result, the target's ability to access the internet is likewise interrupted [[67](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib67)].

DNS Spoofing or DNS Cache Poisoning: DNS spoofing, also known as DNS cache poisoning, is a method of utilizing updated DNS records to reroute [online traffic](https://www.sciencedirect.com/topics/computer-science/online-traffic) to a malicious website that seems to be the intended destination. Users are prompted to enter their accounts at the fake site. They effectively allow the threat actor to steal [access credentials](https://www.sciencedirect.com/topics/computer-science/access-credential) and any sensitive information entered into the fraudulent login form after they submit the information. Furthermore, these malicious websites are frequently used to download viruses or worms onto end users' computers, giving the threat actor ongoing access to the device and any data it holds [[67](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib67)].

NXDOMAIN Attack: A DNS NXDOMAIN flood DDoS attack uses many requests for incorrect or nonexistent records to overload the DNS server. A DNS proxy server frequently handles such attacks by querying the DNS authoritative server with most (or all) of its resources. Both the DNS authoritative server and the DNS proxy server spend time processing invalid queries. As a result, the response time for valid queries gradually increases until it ceases [[67](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib67)].

Domain Hijacking: This attack may entail alterations to your DNS servers and domain registrar, which might send your traffic to alternative locations instead of the original servers. However, domain hijacking may also occur at the DNS level when attackers seize control of your DNS records [[68](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib68)].

* + Domain hijacking is frequently triggered by various factors connected to exploiting a weakness in the domain name registrar's system. Once the threat actors have taken control of your domain name, they'll likely utilize it to carry out malicious actions like creating a phony website for payment systems like PayPal, Visa, or financial institutions. Attackers will exact copies of the legitimate website that stores crucial personal information like email addresses, usernames, and passwords [[68](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib68)].

Fast-flux DNS: The idea behind a fast-flux network is to provide botnets with the ability to quickly switch from one Internet protocol (IP) address to the next while exploiting a host that has been completely compromised. A fast-flux network employs various IP addresses and swiftly switches between them. One [malicious domain](https://www.sciencedirect.com/topics/computer-science/malicious-domain) name will have all the IP addresses pointing at it, but there will be many ways for users to access it. The IP address changes often, even though the domain remains constant regarding the website each user accesses [[69](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib69)].

* + Botnets will use a range of IP addresses with malicious domains. Each will only be active for a short period before cycling in a new one, and as soon as people connect, the domain will collect their login details and other private data. Because the IP address changes frequently, it is pretty challenging to locate the source and stop it [[69](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib69)].

Distributed Reflection Denial of Service (DRDoS): A DRDoS attack aims to send requests from its servers. The secret is to spoof the source address assigned to the victim so that all machines respond and flood the target. As witnessed when KrebsOnSecurity was struck by DRDoS in 2016, this type of attack frequently includes and is created by botnets that operate hacked systems or services that will ultimately be utilized to generate the amplification effect and attack the victim [[68](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib68)].

Random Subdomain Attack: Random subdomain attacks are frequently categorized as DoS attacks since they have the same objective as standard DoS. In this instance, attackers submit several DNS requests to a legitimate, active domain name. The primary domain name will not be the focus of the requests, but rather a large number of invalid subdomains. To stop all DNS record lookups, the attackers must first generate a DoS that overwhelms the authoritative DNS server that hosts the primary domain name. It's an attack that's hard to detect, as the queries will come from botnets from infected users who don't even know they're sending these types of queries from what are ultimately legitimate computers [[68](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib68)].

DoS and DDoS: A denial of service (DoS) attack occurs when an attacker utilizes their resources to make connection requests to a server that provides services to various users, eventually jamming and shutting down the server that provides services. In a DDoS attack, an attacker (botmaster) requests a server that allows service providers to access various users using bots or computers under the attacker's control that may or may not be known to the computer's owner. As a result, the server is inevitably overwhelmed with requests [[70](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib70)].

* + In February 2020, Amazon's AWS Shield service thwarted the most significant DDoS attack ever (2.3 Tbps), which was carried out using hijacked Connection-less Lightweight [Directory Access Protocol](https://www.sciencedirect.com/topics/computer-science/directory-access-protocol) web servers [[71](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib71)].

Network Eavesdropping or Sniffing: [Cybercriminals](https://www.sciencedirect.com/topics/computer-science/cybercriminals) or attackers listening in on network traffic passing through [PCs](https://www.sciencedirect.com/topics/computer-science/personal-computer), servers, mobile devices, and IoT devices are said to be conducting [eavesdropping attacks](https://www.sciencedirect.com/topics/computer-science/eavesdropping-attack). The act of reading or stealing data as it passes between two devices is known as network eavesdropping, often referred to as network sniffing or snooping, and it happens when malicious parties take advantage of weak or [unsecured networks](https://www.sciencedirect.com/topics/computer-science/unsecured-network). [Wireless communication](https://www.sciencedirect.com/topics/computer-science/wireless-communication) is the most popular kind of eavesdropping [[11](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib11)].

Zero-day: In general, “zero-day” refers to freshly identified security flaws that hackers may exploit to attack systems. Since the vendor or developer has only recently become aware of the problem, they have “zero days” to repair it. This situation is referred to as a “zero-day” defect. When hackers take advantage of the vulnerability before developers can fix it, it is known as a zero-day attack [[72](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib72)].

* Zero-day is sometimes known as 0-day. The terms vulnerability, exploit, and attack are frequently used in conjunction with zero-day, and it's essential to know the distinctions between them. A software vulnerability the vendor has not yet patched is a zero-day vulnerability. Since there is no fix for zero-day vulnerabilities due to the vendor's unawareness, attacks are more likely to succeed. A zero-day exploit is a method hackers use to attack systems with a previously unidentified vulnerability. A zero-day attack is when a system is vulnerable, and a zero-day exploit is used to harm the system or steal data from it [[72](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib72)].

#### 7.1.3. Software/application attacks

IoT (Internet of Things) software attacks involve exploiting vulnerabilities in IoT devices, systems, or network software components to compromise security, steal data, disrupt operations, or gain unwanted access. These attacks are directed at the software layer of IoT devices, which includes their operating systems, applications, firmware, and any software interfaces with which they communicate. Some of the software attacks are:

Malware Attacks: Malware is malicious software designed to exploit or attack devices through their hardware or software. Malware is classified into several types: viruses, Trojans, rootkits, backdoors, etc. During the 1980s, malware was file infectors or boot sectors conveyed via floppy disks placed into the machine. However, as technology and electronic devices became more standardized, malware to target such systems progressed. IoT, a set of devices linked to the Internet without human involvement, is one such new technology being abused by malware. Personal computers began to target IoT devices by increasing their capabilities [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib17)].

Unlike typical malware, IoT malware crawls the Internet for susceptible devices. It hosts their initial payload, which is a stager script, in the devices to download the architecture-specific binary sample. After downloading, the script runs the sample, which talks with the C&C server. Because the malware contains some scanner modules, it infects more devices by sharing the sample. Most malware used to attack personal computers, such as Gamut, Necurs, and Skeeyah, began attacking IoT devices by enhancing their capabilities [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib17)]. Some of the malware categories are:

Worm: This type of IoT malware spreads and propagates automatically in IoT devices. Juniper Threat classifies the worm as annoying malware due to its propagation mode. Mirai, Darlloz, Brickerbot, and Gitpaste-12 are some of the worms in IoT devices [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib17)].

Trojan: A Trojan, often known as a [Trojan horse](https://www.sciencedirect.com/topics/computer-science/trojan-horse) or Trojan virus, is another type of IoT malware that seems innocent to users despite having hidden malicious functionality. Indeed, the functionality of a Virus and a Trojan is completely different because the Trojan cannot replicate itself, but the Virus can. ProxyM is an IoT virus that does email spamming and attacks involving DDoS [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib17)].

Virus: Although the term “virus” is often used in computer science, the term “virus” in IoT devices appears to be confusing. IoT viruses behave similarly to others, except they infect IoT devices via self-replicating malicious code. As a result, the virus is difficult to remove and attacks the device in a complicated way. Silex, for example, is an IoT virus that enters the device and bricks it, commonly known as a permanent DoS attack [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib17)].

Backdoor: Backdoor, an IoT malware category, is a type of malware where manufacturers make several hidden access mechanisms. Although these mechanisms make the customer fulfil requirements, these pave the way for making the device poor in [security aspects](https://www.sciencedirect.com/topics/computer-science/security-aspect). As a result, backdoors are also known as the front doors of attackers. Tsunami and Bashlite are backdoor IoT malware with a few resources that address them as Trojans [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib17)].

Spyware: The IoT malware category spyware allows attackers to listen in on or spy on a target's data via an infected device. The type of IoT malware spyware allows attackers to listen in on or spy on a target's data via an infected device. Spybot, Skeeyah, and HNS are examples of IoT spyware that monitors users. As the use of IoT devices grows, so does the number of attacks perpetrated by this malware [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib17)].

Ransomware: IoT ransomware is a type of malware that encrypts IoT devices and demands a ransom from the victim to exchange the device. Once infected, the attacker encrypts the data and prevents users from viewing it. The attacker delivers the decryption key and releases the device after receiving the ransom. Necurs is an IoT virus that performs ransomware attacks and other forms of digital extortion [[17](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib17)].

Denial of Service (DoS) and [*Distributed Denial of Service*](https://www.sciencedirect.com/topics/computer-science/distributed-denial-of-service) (DDoS) Attacks: As mentioned in Section [7.1.2](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.2).

[*Code Injection*](https://www.sciencedirect.com/topics/computer-science/code-injection): Code injection is a technique threat actors use to enter or inject malicious code that exploits a [software validation](https://www.sciencedirect.com/topics/computer-science/software-verification) weakness. The malicious code is often “injected” into the targeted application and then executed by the server. Applications that employ unvalidated input data are often vulnerable to code injection. Code injection attacks often target flaws in data validation. Data validation concerns may include [data formats](https://www.sciencedirect.com/topics/computer-science/data-type), the amounts of data expected, and the types of characters allowed [[78](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib78)]. Some of the different kinds of code injection are:

[*Directory Traversal*](https://www.sciencedirect.com/topics/computer-science/directory-traversal): Directory traversal is a web vulnerability that allows attackers to access files and directories not meant to be accessible by a UI interface. It occurs due to insufficient validation and sanitization of user input or file paths. In a directory traversal attack, attackers modify the file path using specially crafted input such as “../” or “../../” to access sensitive files or folders. Successful attacks may result in unauthorized access to sensitive information, file modification or deletion, or even execute arbitrary code on a device [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib79)]. Some directory traversal attacks are null byte injection, [local file inclusion](https://www.sciencedirect.com/topics/computer-science/local-file-inclusion) attack, and [remote file inclusion](https://www.sciencedirect.com/topics/computer-science/remote-file-inclusion) attack.

Hypertext [*Markup Language*](https://www.sciencedirect.com/topics/computer-science/markup-language) 5 (HTML5) Injection: HTML5 injection is a web application vulnerability form allowing an attacker to embed malicious code into a web page, especially within the victim's browser environment. The attack has the potential to do harm and provide the attacker with the capacity to launch more attacks. Some of the HTML5 injection attacks are content injection, script injection, attribute injection, template injection, SVG injection, CSS injection, etc. [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

[*Extensible Markup Language*](https://www.sciencedirect.com/topics/computer-science/extensible-markup-language) (XML) Injection: XML injection attacks are a type of [security vulnerability](https://www.sciencedirect.com/topics/computer-science/security-vulnerability) in environments that use XML to transfer and store data. These attacks happen when an attacker modifies the XML data being communicated or saved by altering the elements, attributes, or entities of an XML document, which may then be executed on the victim's application. As a result, this might have a variety of ramifications, allowing the attacker to carry out other harmful acts. Some of the XML injection attacks are XPath injection, XML entity expansion, XML external entity (XXE) injection, extensible stylesheet language transformations (XSLT) injection, etc. [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

SQL Injection: SQL injection is a type of application security vulnerability when an attacker inserts malicious SQL queries into application input fields to modify the underlying database and extract sensitive information. This attack can potentially target various technologies that rely on a database for storage, including online apps, [desktop applications](https://www.sciencedirect.com/topics/computer-science/desktop-application), and mobile applications. An attacker may transmit specially designed information that the application does not properly check, allowing the attacker to execute arbitrary SQL code.

* + Furthermore, the attack may change the parameters of a SQL query to include new instructions or data. Depending on the database type of the vulnerable application, the attacker may additionally utilize particular syntax in the input to manipulate the structure or behaviour of the SQL query. Some of the SQL injection attacks are:
    - ∗

In-band SQL Injection: In-band SQL injection, also known as traditional SQL injection, refers to an injection attack in which the attacker exploits the same communication channel to execute the attack and get the outcomes. In technical terms, the attack entails inserting a malicious SQL query into a vulnerable application. This query is then communicated to the attacker using the same communication channel and the acquired results. There exist two subtypes.

* + - In the context of SQL injection, union-based SQL injection refers to the technique employed by an attacker to insert a UNION clause into a query. This UNION clause merges the outcomes of many SELECT queries, allowing the attacker to obtain unauthorized access to [confidential information](https://www.sciencedirect.com/topics/computer-science/confidential-information).
    - In the context of error-based SQL injection, the attacker deliberately transmits SQL queries that are improperly formatted to the application. This action prompts the database to generate error messages, which inadvertently disclose details about the underlying structure of the database. The displayed mistakes typically serve as feedback to the individual carrying out the attack, allowing them to manipulate the injected SQL query to achieve a successful outcome [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
    - ∗

Inferential SQL Injection: The attack, often known as blind SQL injection, involves the transmission of data payloads to the targeted device to observe and analyze its reactions and behaviour to get knowledge about its underlying architecture. This methodology is sometimes called blind SQL injection (SQLi) as it involves the absence of data transmission from the targeted website's database to the perpetrator. Consequently, the attacker cannot get in-band information about the attack [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

* + - Blind SQL injections may be categorized into two distinct types: The Boolean kind of attack involves the transmission of a SQL query to the database, which then prompts the application to respond. The outcome of this result will be contingent upon the integrity of the enquiry. Consequently, the content of the Hypertext Transfer Protocol (HTTP) response has the potential to either undergo modifications or remain unaltered. Based on the modifications above, the attacker can ascertain if the message generated a correct or [incorrect response](https://www.sciencedirect.com/topics/computer-science/incorrect-response) [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)]. The time-based attack involves submitting a SQL query to the database, resulting in a deliberate delay of the database's response for a predetermined duration of time. The attacker can ascertain a query's veracity by analysing the database's response time. Depending on the result, an HTTP response will be generated promptly or after a certain period. This enables the attacker to ascertain the veracity of their message's outcome without necessitating direct access to the database's data [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
    - ∗

Out-of-band SQL Injection: In this particular form of SQL injection attack, the attacker inserts a malicious SQL query into a vulnerable application. However, rather than passively awaiting the application's response for data recovery, the attacker proactively initiates an independent request to obtain the data over another route. As an illustration, the attacker may use a domain name system (DNS) inquiry to acquire the data or utilize an HTTP request to a device within their jurisdiction to collect the info [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Command Injection: Command injection is a type of vulnerability that enables a threat actor to insert [operating system commands](https://www.sciencedirect.com/topics/computer-science/operating-system-command) directly into a program and subsequently execute them. These instructions are akin to the ones often entered into a Bash or Powershell terminal [[80](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib80)]. Command injection attacks refer to exploiting vulnerabilities present in programs that execute system commands. A potential threat actor can introduce malicious commands into the program, which are later executed by the underlying operating system. This enables the attacker to execute any intended operations on the system being targeted, potentially providing them with complete control.

* + Command injection attacks may be categorized into two main types: standard command injections and blind command injections. In the context of standard command injection, the attacker can view the outcomes of the injected instructions immediately. In contrast, in the context of blind command injection, the attacker cannot directly monitor the outcomes of the executed instructions. Instead, individuals must depend on alternate results, such as alterations in application behaviour, to assess the attack's success. Hence, conducting a blind command injection attack is typically more challenging than the above-mentioned method. The rationale for this is that the perpetrator does not receive prompt feedback on the efficacy or ineffectiveness of their injection attempt.
  + In an alternate scenario, the attacker needs to employ either guesswork or supplementary techniques to ascertain the outcomes of their injection. Some of the command injection attacks are powershell injection, command prompt injection, shell injection, script injection, remote command injection, [dynamic link library](https://www.sciencedirect.com/topics/computer-science/dynamic-link-library) (DLL) injection, library injection in unix-like operating systems, cron injection, process injection, memory injection attack, environment variable injection attacks, registry injection attacks, lightweight directory access protocol (LDAP) injections and Java logging framework injection (Log4j) [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Cross-site Request Forgery (CSRF): Cross-site request forging (CSRF), also known as session riding, cross-site reference forging, and hostile linking, is a security vulnerability that arises when a malicious attacker deceives the target into unintentionally executing an action on a web application while they are currently authorized. The attacker commonly does this by introducing malicious code into a target individual's web browser via a malicious website or email, resulting in the user's browser requesting the vulnerable application without the user's awareness or authorization [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

* + The main concern associated with cross-site request forgery (CSRF) is the web application's incapacity to distinguish between a legitimate request launched by the user and a [malicious request](https://www.sciencedirect.com/topics/computer-science/malicious-request) orchestrated by an attacker. This is because the web application depends exclusively on the [authentication credentials](https://www.sciencedirect.com/topics/computer-science/authentication-credential) saved in the user's browser to recognize the user without further procedures to confirm the user's intention to carry out the requested operation. Consequently, an attacker can exploit cross-site request forgery (CSRF) to execute activities on behalf of the user, including unauthorized financial transfers or altering the user's password, without their awareness or authorization. The subsequent examples illustrate several cross-site request forgery (CSRF) attacks [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + In the context of GET-based cross-site request forgery (CSRF), an attack transpires when an adversary entices a target to click on a hyperlink that incorporates a malicious request within the URL. The execution of the request occurs when the recipient interacts with the hyperlink, prompting their web browser to transmit the request to the vulnerable program. Due to the use of the victim's [login credentials](https://www.sciencedirect.com/topics/computer-science/login-credential), the vulnerable application cannot differentiate between a valid request and a malicious one.
  + The POST-based cross-site request forgery (CSRF) attack resembles the GET-based CSRF attack, but it exploits the victim's act of submitting a form that includes a malicious request. The attacker can potentially deceive the target by employing social engineering strategies, such as camouflaging the form as an authentic login or registration form. The login CSRF attack focuses explicitly on exploiting the login procedure of a vulnerable application. The potential attacker can include malicious code into a login form or request, so enabling unauthorized access to the victim's account without their awareness or permission [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + The logout CSRF attack resembles the login CSRF attack since it focuses on exploiting the logout mechanism of a vulnerable application. The potential offender can include harmful code into a logout request, resulting in the victim being logged out of the application without their awareness or consent. An Ajax-based cross-site request forgery (CSRF) attack transpires when a [malicious actor](https://www.sciencedirect.com/topics/computer-science/malicious-actor) takes advantage of a vulnerable Ajax request present on a webpage. The potential attacker possesses the capability to alter the Ajax request to incorporate a malicious request that is performed upon the victim's visit to the webpage [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Format String Attacks: Format string attacks are a type of security vulnerability that may be exploited in many software programs. Format string attacks occur when an attacker can inject format string input into software that is vulnerable to such exploitations. A format string may be seen as a series of characters that establishes the arrangement and presentation of data when outputted or shown on a [computer screen](https://www.sciencedirect.com/topics/computer-science/computer-screen). The format string input can be exploited by an attacker by manipulating its content, resulting in the extraction of private information from a program's memory or the execution of arbitrary code. The attacker can utilize specific format string conversion specifiers, such as %n, %s, %x, or %p, to get access to memory regions and modify the data stored within those places. The potential consequences of a successful format string attack are substantial since it allows an attacker to gain control over a susceptible machine and execute malicious code [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

* + Some forms of format string attacks are format string vulnerabilities exploited by [information disclosure](https://www.sciencedirect.com/topics/computer-science/information-disclosure) attacks to acquire sensitive information from a program's memory. By utilizing precise format string conversion specifiers, malicious actors can get data from memory locations following the desired input's memory address. This could expose system settings, passwords, or user data. Denial of service (DoS) attacks using formatted strings refer to instances where an attacker deliberately delivers input containing format string conversion specifiers, resulting in software crashes or halts. This may be achieved by using the %n specifier to write data to a memory location that is not valid, resulting in program failure [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + [Arbitrary code execution](https://www.sciencedirect.com/topics/computer-science/arbitrary-code-execution) attacks include the exploitation of format string vulnerabilities to execute arbitrary code on a system susceptible to such attacks. Malicious actors can write specific values to a designated address using format string conversion specifiers. This action can result in executing code loaded into such an address, enabling the attackers to gain control over the system. The abovementioned kind is often regarded as the most severe and dangerous manifestation of format string attacks [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Object Injection: Object injection attacks leverage a security vulnerability present in web applications through serialization. Serialization is converting [data structures](https://www.sciencedirect.com/topics/computer-science/data-structure) or objects into a format that can be stored or sent. The act of object injection involves the intentional manipulation of the serialization process. This manipulation allows an adversary to inject objects that include malicious code [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

* + Once injected, these objects can be run on the [application server](https://www.sciencedirect.com/topics/computer-science/application-server) hosted on a specific IoT device. The consequences of such an attack may be severe, possibly leading to the entire compromise of the system. The vulnerability arises because serialization not only captures the state of an object but also includes information about the object's class and its related methods [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + Object injection can manifest in several forms, including deserialization vulnerabilities, which refer to a specific form of object injection when an adversary manipulates the serialized data to incorporate harmful objects that can be executed during the deserialization process [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + Prototype pollution vulnerabilities refer to a class of security vulnerabilities that arise from manipulating JavaScript prototypes. These vulnerabilities might allow an attacker to modify the behaviour of objects and functions. This form of object injection happens when a malicious entity changes an object's prototype to add arbitrary attributes to global object prototypes. As a result, objects created by users can inherit these newly added characteristics. Expression language (EL) injection refers to a specific sort of security vulnerability when an unauthorized individual inserts arbitrary code or expressions into the data processing expressions of an application. This vulnerability commonly impacts web applications that utilize expression languages such as JavaServer Pages (JSP) expression language or AngularJS expressions, among other similar technologies [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Firmware Code Injection: Firmware refers to the fundamental software that operates at a low level and is tasked with managing the hardware components of a specific device. The storage of data in non-volatile memory, such as flash memory or read-only memory (ROM), is a [common practice](https://www.sciencedirect.com/topics/computer-science/common-practice). This storage is of [utmost importance](https://www.sciencedirect.com/topics/computer-science/utmost-importance) as it plays a crucial role in the proper functioning of a device [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

* + Nevertheless, the [development process](https://www.sciencedirect.com/topics/computer-science/development-process) may occasionally neglect the firmware of IoT devices, resulting in potential security risks. The attack vector known as firmware code injection involves the exploitation of vulnerabilities to implant malicious code into the firmware of a device. The attacks may manifest in several ways, such as inserting a backdoor into the initial firmware or altering the bootloader to initiate the execution of [malicious scripts](https://www.sciencedirect.com/topics/computer-science/malicious-script) during the booting process [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + Furthermore, the attacker can activate unprotected protocols inside the firmware, rendering it vulnerable to exploitation. Moreover, the attacker can alter or append scripts inside the firmware, rendering it susceptible to further forms of code injection attacks, like SQL injection and XSS. Upon executing the inserted code, the attacker acquires unauthorized access to the device and its functionalities [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + The potential outcomes of this situation encompass a wide range of effects, including but not limited to the unauthorized acquisition of data, malfunctioning of the device, and the occurrence of more advanced forms of attacks, such as the utilization of the compromised device as a component of a botnet or the initiation of distributed denial of service (DDoS) operations.
  + Several techniques exist that malicious actors might employ to introduce malicious programming into the firmware of IoT devices. In the case of direct physical access, an adversary can exploit hardware debugging tools or influence the firmware update procedure to introduce malicious code into the firmware directly. Remote exploitation refers to attackers using weaknesses in a device's firmware update process or communication protocols to introduce malicious code remotely. The process of attaining this objective frequently involves the practice of reverse engineering the firmware of the device and doing a thorough analysis to identify any vulnerabilities [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + Supply chain attacks include firmware compromise during the production phase before the device is delivered to the end user, enabling attackers to introduce malicious code. This issue is of particular significance due to its elusive nature, making it challenging to identify, and its potential to impact several devices concurrently. The breach of firmware can occur when attackers manipulate the files accessible for download on the manufacturer's website. Upon acquiring the firmware, the potential attacker may analyze its structure and functionality using reverse engineering techniques. Subsequently, they might embed a malicious code inside the firmware and reprogram the device with the updated, compromised version [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Log Poisoning Attacks: Log poisoning refers to a method employed by malicious actors to inject a code into a log file. This attack exploits [misconfigurations](https://www.sciencedirect.com/topics/computer-science/misconfiguration) in log settings, allowing the injected code to be run. As a result, the attacker can avoid detection and carry out malicious actions on the targeted system. Log files are crucial for system administrators and security experts due to their inherent value in documenting system events, user activity, and potential security vulnerabilities [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

* + Various types of logs can be vulnerable to log poisoning attacks. Web server logs are a type of record that document various aspects of client requests, server answers, and any issues that may arise while processing these requests. The attacker can introduce malicious code into HTTP requests or change URL parameters using code. Some examples of log files often used in web servers include Apache access logs, nginx access logs, and IIS logs. Many software programs, including websites, mobile apps, and desktop software, create application logs. An attack occurs when an attacker inserts malicious code into fields designated for user input, influencing the log entries produced by the program [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + System logs encompass several sorts of logs that offer valuable insights into the events taking place within an operating system and its services and components. These logs serve the purpose of providing information and documentation on the activities mentioned above. Some examples of [log management systems](https://www.sciencedirect.com/topics/computer-science/log-management-system) include Linux Syslog and Windows Event Logs [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + Authentication logs are responsible for recording [user authentication](https://www.sciencedirect.com/topics/computer-science/user-authentication) occurrences, including instances of successful login, unsuccessful login attempts, and instances when an account has been locked. The attack entails the insertion of harmful code into fields designated for usernames or passwords. Examples of log files often used in [computer systems](https://www.sciencedirect.com/topics/computer-science/computer-system) are SSH logs in Linux-based operating systems and Windows Event Logs for login events.
  + The mail server logs store comprehensive data on email transactions, including sent and received messages, encountered issues, and other pertinent occurrences. An attack occurs when an attacker inserts malicious code into the headers or body of emails. Some examples of logs are those generated by the Postfix, Sendmail, and Exim [mail transfer agents](https://www.sciencedirect.com/topics/computer-science/mail-transfer-agent) [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + Database logs are responsible for recording events that are associated with database operations. These events include performed queries, data updates, and failures. Attackers can insert harmful SQL code into queries, using weaknesses in log [parsing](https://www.sciencedirect.com/topics/computer-science/parsing) tools or log management systems. Illustrative instances encompass log files derived from several [database management systems](https://www.sciencedirect.com/topics/computer-science/database-management-system), such as MySQL, [PostgreSQL](https://www.sciencedirect.com/topics/computer-science/postgresql), Oracle, and SQL Server [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Hibernate [*Query Language*](https://www.sciencedirect.com/topics/computer-science/query-language) (HQL) Injection: HQL is an acronym for hibernate query language. Hibernate is an object-relational mapping (ORM) framework that facilitates mapping class definitions inside source code to corresponding SQL databases. The HQL (Hibernate Query Language) is a programming language similar to SQL (Structured Query Language). However, unlike SQL, HQL operates on persistent objects rather than directly interacting with tables and columns. The Hibernate framework facilitates the translation of HQL queries into SQL queries, which are then executed within the database [[81](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib81)].

* + Hibernate query language (HQL) injection is a type of injection attack in which an adversary manipulates the HQL query to execute harmful SQL statements that can manipulate the [database server](https://www.sciencedirect.com/topics/computer-science/database-server) of a web application [[82](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib82)].

Indirect Prompt Injection Attacks: The indirect prompt injection refers to an attack that capitalizes on a vulnerability, allowing an attacker to insert harmful code into an application by manipulating prompts or messages presented to the user. This injection occurs when an application requests input from a user, and the attacker exploits this request to insert harmful code into the program.

* + Recent research [[83](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib83)] has investigated the occurrence of injection attacks on application-integrated [large language models](https://www.sciencedirect.com/topics/computer-science/large-language-model) (LLMs). The authors noted that integrating LLMs, such as [ChatGPT](https://www.sciencedirect.com/topics/computer-science/chatgpt), with other applications might render them vulnerable to the intake of untrusted data, potentially including malicious prompts. The authors demonstrated the potential use of such injections in delivering precise payloads. The methodology potentially enables malicious actors to assume control of LLMs by breaching critical security limits by executing a solitary search query [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Cross-site Scripting (XSS): Cross-site scripting (XSS) is a form of cyberattack wherein the malevolent actor inserts malevolent scripts into websites and web applications. The objective is to ensure the execution of these scripts on the end-point devices of users, enabling threat actors to circumvent safeguards and assume the identity of users. Cross-site scripting (XSS) attacks include exploiting a benign website or online application, transforming it into a means to transmit harmful scripts to the web browsers of those unaware of the impending danger. The objective of the attack is to illicitly acquire cookies, session IDs, names, and passwords [[78](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib78)].

* + Numerous programming environments and languages, including Flash, ActiveX, JavaScript, and VBScript, are susceptible to potential risks. JavaScript is often executed on web pages within web browsers, and cyber attackers often focus their efforts on exploiting vulnerabilities in JavaScript due to its strong integration with most browsers [[78](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib78)].
  + XSS attacks may be categorized into three primary categories. The security vulnerability known as reflected XSS occurs when an attacker can insert malicious code into a webpage. This code is then shown on a user's browser and executed inside the context of the website. This vulnerability stems from the web application's inability to adequately authenticate user input, enabling malicious actors to incorporate malicious code into the application's response. To carry out this attack, the attacker commonly entices the target to interact with a deceitfully constructed hyperlink or to provide input through a specifically tailored form that incorporates the malicious code. When the user interacts with the vulnerable web application, the malicious code is transmitted back to the user's browser and executed. This action grants the attacker the ability to initiate more attacks.
  + In the context of stored XSS, an attacker with [malicious intent](https://www.sciencedirect.com/topics/computer-science/malicious-intent) inserts and retains harmful code within a web application's database or alternative storage medium. The malicious code is sent to all individuals who visit the impacted webpage, possibly jeopardizing their security and enabling the attacker to execute various malicious activities.
  + DOM-based XSS is a form of online application vulnerability that arises when an adversary can insert malicious code into a web page's document object model (DOM). This injected code is then executed inside the context of the victim's browser. DOM-based cross-site scripting (XSS) attacks commonly include the manipulation of URL parameters, form input fields, or other web page elements that contribute to the dynamic generation of content inside the document object model (DOM) [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
  + In contrast to reflected and stored XSS attacks, which arise from inadequate server-side input validation, [DOM](https://www.sciencedirect.com/topics/computer-science/document-object-model) XSS vulnerabilities stem from executing client-side scripts. Identifying and mitigating such attacks would become more complex, as traditional techniques of server-side input validation may not be enough to reduce these risks [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

LDAP Injection: An LDAP injection refers to a type of attack that takes the use of web-based apps that are susceptible to vulnerabilities, wherein these programs create LDAP statements using user input. If software neglects to sanitize user input correctly, it becomes susceptible to potential exploitation by attackers who can manipulate LDAP statements by utilizing a local proxy. This vulnerability enables the execution of arbitrary actions, including the unlawful granting of permissions to queries and the change of material inside the LDAP tree. An LDAP injection attack frequently employs similar exploitation techniques as those utilized in SQL injection attacks [[84](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib84)].

[*Remote Code Execution*](https://www.sciencedirect.com/topics/computer-science/remote-code-execution): Remote code execution (RCE) refers to a security vulnerability that enables unauthorized individuals to execute arbitrary code within an application's programming language, as per the language chosen by the developer. The word “remote” refers to the capability of an attacker to perform actions from a place distinct from the system where the application is being executed. RCE vulnerabilities have the potential to manifest in several forms of computer software across a wide range of programming languages and on diverse platforms. RCE vulnerabilities may be found in several software systems, including standalone Windows apps developed using C#, online applications and [APIs](https://www.sciencedirect.com/topics/computer-science/application-programming-interface) implemented in PHP, mobile applications created using Java, and even within operating systems [[85](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib85)].

* Certain remote code execution (RCE) attacks have the potential to occur after a period of latency. As an illustration, the program can first store the remote code execution (RCE) payload within a [configuration file](https://www.sciencedirect.com/topics/computer-science/configuration-file), deferring its execution until later, even on several occasions. The vulnerability is a stored remote code execution (RCE) vulnerability. It is essential to acknowledge that there is a common misconception between RCE (Remote Code Execution) and OS (Operating System) command injection [[85](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib85)].
* In the context of remote code execution (RCE), the executed code is written in the same programming language as the application and operates within the application's environment. In the context of OS command injection, the perpetrator executes an operating system command. It is essential to acknowledge that although the phrase “code injection” is favoured by OWASP and expressly specified in CWE-94, the term “remote code execution” has more prevalence in usage across many contexts [[85](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib85)].
* A well-known case of remote code execution (RCE) is the CVE-2021-44228 (often referred to as Log4Shell) found in Apache Log4j 2.x. This vulnerability was then accompanied by CVE-2021-45046 and CVE-2021-45105, which introduced a denial of service vulnerability. These cases exemplify instances of RCE occurring within non-web applications. The vulnerability mentioned above in Log4j, which does not need authentication from the attacker, impacts several versions of Log4j and is mainly located inside the JndiLookup class of this widely used open-source logging library [[85](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib85)].
* Several widely used applications and services, such as Steam, Apple iCloud, and Minecraft, were first discovered to have vulnerabilities. CVE-2021-1844, observed in Apple's iOS, macOS, watchOS, and Safari, further illustrates remote code execution (RCE) vulnerability within a module of an operating system. When a victim employs a susceptible device to access a URL under the control of an attacker, the operating system will proceed to execute a harmful payload on said device [[85](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib85)].
* The occurrence of CVE-2020-17051 within the Microsoft Windows NFSv3 framework illustrates a remote code execution (RCE) vulnerability found within a module of an operating system. A potential attacker can connect with a susceptible NFS server and transmit a payload, causing the targeted endpoint to execute such a payload.CVE-2019-8942 within WordPress 5.0.0 serves as an instance of a remote code execution (RCE) vulnerability found in a widely used online application. The potential for an attacker to run arbitrary code within the WordPress platform arises when a specifically manipulated picture file is uploaded, whereby the Exif metadata of said file contains PHP code [[85](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib85)].

[*Buffer Overflow*](https://www.sciencedirect.com/topics/computer-science/buffer-overflow): A buffer overflow is a security vulnerability that arises when a computer program attempts to exceed the capacity of a buffer, a designated space for temporary data storage, by writing excessive data. This phenomenon can potentially result in program failure or, in certain instances, enable an unauthorized individual to execute malevolent instructions on the system.

* Buffer overflows can manifest when the software fails to adequately verify the size or structure of the input it receives, enabling a malicious actor to transmit a substantial volume of data that surpasses the buffer's allotted limit. In such an occurrence, the surplus data can overwrite additional segments of the program's memory, allowing the attacker to execute arbitrary code or gain control over the system [[86](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib86)].
* Buffer overflows are a prevalent security risk, particularly in outdated or inadequately engineered software. Preventing such occurrences might pose challenges due to their inherent nature of including unanticipated or [malicious input](https://www.sciencedirect.com/topics/computer-science/malicious-input), which deviates from the planned scope of the program's functionality. To mitigate the risks associated with buffer overflows, developers must exercise caution in validating information and implementing robust error-handling mechanisms inside their programs. This entails verifying the integrity and appropriateness of incoming data while fortifying the program's resilience against unforeseen inputs to prevent system crashes and potential [security breaches](https://www.sciencedirect.com/topics/computer-science/security-breach) [[86](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib86)].
* Buffer overflow attacks provide a significant security risk because they enable the unauthorized execution of arbitrary code on a system. This unauthorized execution can grant attackers complete control over the system, facilitating the possible theft of sensitive information. Buffer overflow frequently attains a prominent position inside the SANS top 20 most dangerous software errors [[86](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib86)].
* The common weakness enumeration (CWE) is a comprehensive compendium of software security vulnerabilities encompassing a range of problems. Among these issues, there are several that pertain to the occurrence of a buffer overflow. CWE-120, sometimes referred to as “Buffer Copy without Checking Size of Input”, delineates a specific circumstance in which a computer replicates data from one buffer to another without sufficiently verifying the input size, creating the possibility of a buffer overflow vulnerability. Additional vulnerabilities under the common weakness enumeration (CWE) framework that are associated with buffer overflows encompass: CWE-119: “Improper Restriction of Operations within the Bounds of a Memory Buffer”, CWE-121: “Stack-Based Buffer Overflow”, CWE-122: “Heap-based Buffer Overflow”, CWE-125: “Out-of-bounds Read”, CWE-131: “Incorrect Calculation of Buffer Size” [[86](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib86)].
* Several variations exist of buffer overflow attacks: Stack-based buffer overflow refers to the act of overwriting the memory of a program's stack, which is responsible for storing local variables and function calls. This manipulation allows for the execution of arbitrary code or the modification of the program's control flow. A heap-based buffer overflow refers to overwriting the memory allocated in a program's heap, which is responsible for dynamically allocating memory. This can result in the execution of arbitrary code or the modification of the program's behaviour [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
* The [integer overflow](https://www.sciencedirect.com/topics/computer-science/integer-overflow) attack refers to a security breach that arises when a mathematical computation generates a result that exceeds the chosen data type's capacity, leading to an inaccurate representation of the value. It is commonly observed when the outcome of an operation exceeds the upper or lower limit that may be represented by the given data type, resulting in the value “wrapping around” and returning to a lower or higher value [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
* Return-oriented programming (ROP) is a technique that involves utilizing pre-existing code snippets, referred to as gadgets, within a program to run arbitrary code without the need to inject new code. Jump-oriented programming (JOP) refers to utilizing pre-existing code snippets within a program to execute arbitrary code by navigating to various memory regions, all without introducing new code injection [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
* The global offset table (GOT) buffer overflow is a specific buffer overflow attack that focuses on exploiting the GOT data structure located within a program's memory. Notably, software programs commonly utilize the global offset table (GOT) to hold the memory locations of dynamically linked functions and variables. By manipulating the GOT software, an attacker can reroute the program's execution flow towards their malicious code [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].
* The unicode overflow attack is a type of hack that takes advantage of a vulnerability resulting from inadequate processing of unicode-encoded data. Like other forms of buffer overflow, this attack is executed by transmitting malicious unicode input that surpasses the allocated buffer size. As a result, memory corruption occurs, potentially leading to the execution of malicious code. The return-to-libc attack is devised and executed to circumvent security measures, such as a non-executable stack. Instead of directly inserting and running malicious code into the stack, the attacker replaces the return address with the address of a targeted function that already exists in a standard C library, such as printf or scanf, among others [[79](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib79)].

Vulnerable Firmware: The exploitation of vulnerabilities in firmware, which refers to the software integrated into physical devices, can enable malicious actors to modify the behaviour of the device or gain unauthorized control over it.

Man-in-the-middle (MitM) Attacks: Man-in-the-middle (MitM) attacks manifest when an unauthorized individual gains access to the communication channel connecting two end systems. This is achieved by introducing a malicious node between the [legitimate nodes](https://www.sciencedirect.com/topics/computer-science/legitimate-node) or exploiting vulnerabilities in the communication protocols inside an IoT network. The idea of man-in-the-middle (MitM) allows hackers to manipulate the flow of data, modify the configuration of the [network architecture](https://www.sciencedirect.com/topics/computer-science/network-architecture), fabricate counterfeit identities, and generate deceptive and inaccurate information to undermine the security of an IoT system. The many forms of man-in-the-middle (MitM) attacks include eavesdropping, Sybil attacks, Wormhole attacks, Identity replication attacks, Node replication attacks, and others [[87](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib87)]. MitM attacks may be categorized into two basic types: [passive attacks](https://www.sciencedirect.com/topics/computer-science/passive-attack) and active attacks [[88](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib88)].

Passive Man-in-the-middle (MitM) Attack: A passive man-in-the-middle (MitM) attack is characterized by the absence of active alteration of communication by the attacker. In contrast, the assailant clandestinely intercepts the conversation to obtain unauthorized access to confidential data. Despite the absence of communication manipulation by the attacker, they are still capable of acquiring sensitive information, such as usernames, passwords, and other [secret data](https://www.sciencedirect.com/topics/computer-science/secret-data). This is mostly due to inadequate encryption of the majority of traffic. Eavesdropping can be classified as a passive man-in-the-middle (MitM) attack [[88](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib88)].

Active Man-in-the-middle (MitM) attack: An active man-in-the-middle (MitM) attack refers to the deliberate interception and alteration of communication between two entities by an attacker. Once an unauthorized individual has access to the communication channel, they can control the communication process by intercepting, altering, or introducing new messages. The most prevalent forms of active man-in-the-middle (MitM) attacks are impersonation, spoofing attacks, and data manipulation [[88](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib88)].

* + It is essential to acknowledge that MitM represents a multilayered threat, including all layers of the IoT architecture. The veracity of this claim may be demonstrated using a spoofing attack. DNS Spoofing impacts the application layer, IP spoofing affects the network layer, and ARP Spoofing operates at the link layer. Moreover, those who do malicious acts can employ [rogue access point](https://www.sciencedirect.com/topics/computer-science/rogue-access-point) attacks to intercept a network at the physical layer. In addition to the sorts mentioned above of man-in-the-middle attacks, several tactics can be employed to execute these attacks [[88](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib88)].

Sniffing: As mentioned in the Section [7.1.2](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.2).

Packet Injection: Packet injection is a method that enables malicious actors to control the flow of network traffic by introducing malicious packets into the network. Within the domain of the IoT, malicious actors may employ this methodology to disrupt the communication linkages between IoT devices and cloud-based services or to introduce commands that potentially alter those devices' [operational characteristics](https://www.sciencedirect.com/topics/computer-science/operational-characteristic). Scapy and MitM are robust tools for implementing this strategy.

[*Session Hijacking*](https://www.sciencedirect.com/topics/computer-science/session-hijacking): Session hijacking is a technique employed by malicious actors to acquire a user's session information illicitly, granting them the ability to assume the user's identity and gain unauthorized access to their data. Within the domain of the IoT, someone with malicious intent can employ session hijacking as a means to enter the user's IoT devices illicitly and then exert remote control over them. Bettercap is widely recognized as a highly effective tool for implementing this strategy [[88](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib88)].

SSL Stripping: SSL stripping is a method that involves converting an encrypted SSL connection into an unencrypted one. In the IoT context, malicious actors can employ SSL stripping techniques to surreptitiously intercept and modify the data transmitted between IoT devices and cloud-based services. SSLStrip is widely recognized as a tool that may be utilized to execute this particular strategy [[88](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib88)].

Zero-day Exploits: As mentioned in the Section [7.1.2](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.2).

[*Authentication Attacks*](https://www.sciencedirect.com/topics/computer-science/authentication-attack): Authentication attacks in the context of the IoT are activities aimed at circumventing or altering the authentication systems employed by IoT devices, networks, or services. Authentication refers to the procedure through which the identification of individuals, devices, or entities is confirmed before authorizing their access to resources or services. In IoT settings, malicious actors exploit [authentication mechanisms](https://www.sciencedirect.com/topics/computer-science/authentication-mechanism) to attain illegal entry, pilfer confidential data, cause disruptions to operational processes, or execute destructive activities [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib89)].

[*Password Guessing*](https://www.sciencedirect.com/topics/computer-science/password-guessing): Attackers employ trial-and-error techniques to make educated guesses to ascertain usernames and passwords. Passwords that are weak or easily guessable are especially vulnerable to this attack [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib89)].

Brute Force Attacks: Brute force [password attacks](https://www.sciencedirect.com/topics/computer-science/password-attack) employ a systematic approach to systematically test all conceivable combinations to ascertain a password. This approach demonstrates efficacy when used with passwords that possess a limited number of characters and exhibit a relatively low complexity. Even with the most advanced contemporary systems, the feasibility of utilizing a password consisting of eight or more characters can be compromised. If a password only consists of alphabetic characters, encompassing both uppercase and lowercase letters, it is estimated that around 8,031,810,176 attempts would be required to decipher it successfully. This assumes that the potential attacker knows the length and complexity criteria of the password. Additional considerations are [numerical values](https://www.sciencedirect.com/topics/computer-science/numerical-value), the distinction between uppercase and lowercase letters, and special characters within the context of the specific language being localized [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib89)].

* + A brute force attack can finally discover the password given the appropriate settings. The [computational resources](https://www.sciencedirect.com/topics/computer-science/computational-resource) and execution duration frequently render brute force testing inconsequential upon its completion. The duration of attacks is contingent upon the duration required to produce all conceivable permutations of passwords. Subsequently, the consideration of the target system's reaction time is considered. Brute force password attacks are often regarded as the least efficient approach to [password cracking](https://www.sciencedirect.com/topics/computer-science/password-cracking). Therefore, threat actors employ them as a final option [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib89)].

Credential Stuffing: Credential stuffing is an automated hacking technique that uses stolen credentials. The credentials consist of collections of usernames, email addresses, and passwords. The method commonly utilizes automation to initiate login requests targeted at an application and to record successful login attempts for subsequent exploitation [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib89)].

* + Credential stuffing attacks do not engage in the process of systematically attempting to crack or speculate passwords. The threat actor employs specialized tools to automate authentication by leveraging previously obtained credentials. This methodology may include doing a large number of iterations to ascertain the probable instances when a user may have employed the same login credentials on a different website or service. Credential stuffing attacks exploit the practice of password reuse, leveraging the fact that many users use identical credential combinations across different online platforms [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib89)].

Phishing & Vishing: Phishing and vishing, which refers to voice calls, are frequently utilized as means of acquiring information for further attacks, as well as to introduce malicious software onto an endpoint. The malware, as mentioned above, can illicitly extract credentials. It can also be utilized as a faked password reset attack component. A prevalent strategy employed by phishing emails is the inclusion of a hyperlink that prompts the recipient to reset their account password. The attackers may assert that this occurrence is attributable to the probable breach of a previously utilized password. The email may display the emblem and resemblance of a commercial entity, such as a financial institution, vendor, or service provider [[90](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib90)].

* + Nevertheless, the hyperlink inside the electronic mail redirects the target individual to a deceptive interface designed to reset passwords. Subsequently, the attacker procures the authentic password to get unauthorized access to the target individual's valid account. In the context of an employee, it is noteworthy that a deceptive email regarding password reset may manifest as originating from the corporate help desk. These attacks serve as a persistent reminder of the significance of end-user training. Users must exercise constant vigilance to verify the legitimacy of solicited email addresses or phone numbers.

Man-in-the-middle (MitM) Attacks: As mentioned in Section [7.1.3](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.3).

Session Hijacking: As mentioned in Section [7.1.3](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.3).

Token Replay: Attackers use vulnerabilities to acquire [authentication tokens](https://www.sciencedirect.com/topics/computer-science/authentication-token), afterwards employing them to assume the identity of the authorized user and get unlawful access [[91](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib91)].

Password Spraying: Password spraying is an attack that relies on credentials and aims to gain unauthorized access to several accounts by employing a limited set of commonly used passwords. In terms of conceptualization, this phenomenon might be considered the opposite of a brute-force password attack. The brute force method involves systematically and repetitively generating many [password combinations](https://www.sciencedirect.com/topics/computer-science/password-combination) to obtain unauthorized access to a specific user account [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib89)].

* + In the context of a password spray attack, the attacker systematically tests a singular, often employed password (e.g., “12345678” or “Passw0rd”) across several accounts before moving on to test an alternative password [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib89)].
  + The threat actor systematically attempts to authenticate each user account in their roster using a uniform password, resetting the roster and testing the next password. This strategy effectively reduces the likelihood of the threat actor being detected and experiencing lockouts on a single account due to the time intervals between successive attempts. The probability of a threat actor successfully accessing a resource increases when an [individual user](https://www.sciencedirect.com/topics/computer-science/individual-user) does not possess proper password hygiene or on a specific account [[89](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib89)].

[*Biometric*](https://www.sciencedirect.com/topics/computer-science/biometrics) Spoofing: In the context of IoT systems employing [biometric authentication](https://www.sciencedirect.com/topics/computer-science/biometric-authentication) methods such as fingerprint or face recognition, malicious actors have the potential to exploit the system by utilizing counterfeit [biometric data](https://www.sciencedirect.com/topics/computer-science/biometric-data) to circumvent the [authentication process](https://www.sciencedirect.com/topics/computer-science/authentication-process).

* + Biometric spoofing pertains to fraudulent activities in biometric authentication wherein counterfeit samples, such as fabricated fingerprints, face scans, or Iris scans, are presented as a means of executing a “presentation attack”. Although biometric identification technologies are more robust than password-based systems, it is important to note that they are not impervious to vulnerabilities. Modern threats are undergoing a process of adaptation to circumvent the safeguards mentioned above. Therefore, several stringent authentication prerequisites outlined in compliance and regulatory frameworks need liveness detection mechanisms to ascertain the authenticity of the provided credentials [[92](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib92)]. Some of the different biometric spoofing attacks are:
    - ∗

Facial Recognition Spoofing Attacks: Attackers may trick face recognition systems using various methods. Print attack employs a printed image of the target's face to deceive the facial recognition software. This is one of the most basic techniques, and it works well against less complex systems (the majority of which must be used where they would safeguard sensitive data). Replay attack involves hackers filming the target's face on camera and replaying it in front of the camera. This strategy frequently works better than a print attack because it includes motion, which certain facial recognition systems might need [[92](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib92)].

* + - During a 3D mask attack, the attacker fashions a lifelike 3D mask of the target's face and puts it on. While this strategy might be more complicated to detect, it can also be challenging to use effectively without the right tools and abilities. In a deep fake attack, a video of the target's face is made using machine learning or [AI](https://www.sciencedirect.com/topics/computer-science/artificial-intelligence) software. Thanks to [deepfake](https://www.sciencedirect.com/topics/computer-science/deepfakes) technology, some [face recognition algorithms](https://www.sciencedirect.com/topics/computer-science/face-recognition-algorithm) find it challenging to distinguish between actual and synthetic facial movements and expressions [[92](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib92)].
    - ∗

Fingerprint Recognition Spoofing Attacks: Fingerprint [verification systems](https://www.sciencedirect.com/topics/computer-science/verification-system), although often robust, may still be susceptible to spoofing without adequate countermeasures. In fake fingerprints, hackers make synthetic fingerprints that resemble the target user's fingerprint pattern, frequently derived directly from a fingerprint, using materials like gelatin. The attacker's finger or a fake finger can then be put over the fake fingerprint to fool the [fingerprint scanner](https://www.sciencedirect.com/topics/computer-science/fingerprint-scanner) [[92](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib92)].

* + - In latent fingerprints, a perpetrator uses adhesive tape or other means to extract a target user's latent fingerprint from a surface and transfer it to a substance that can deceive the fingerprint scanner. 3D-printed fingerprint describes a complex attack that entails using digital technology to create a 3D model of the target user's fingerprint, which is subsequently printed using materials that have qualities similar to those of human skin. This technique can produce accurate copies that can trick some [fingerprint readers](https://www.sciencedirect.com/topics/computer-science/fingerprint-reader) [[92](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib92)].
    - ∗

[*Iris Recognition*](https://www.sciencedirect.com/topics/computer-science/iris-recognition) Spoofing Attacks: While Iris recognition is typically considered a highly secure [biometric modality](https://www.sciencedirect.com/topics/computer-science/biometric-modality), it can still be subject to spoofing attacks if the [proper safeguards](https://www.sciencedirect.com/topics/computer-science/proper-safeguard) are not taken. Digital Iris images are one type of Iris recognition spoofing attack that involves projecting a digital image or video of the target user's Iris onto a device screen, like a smartphone or tablet, and then presenting it to the Iris scanner. This technique can trick some biometric scanners by altering gadgets' lighting and sharpness settings [[92](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib92)].

* + - Physical eyes, albeit an uncommon and extreme option, can also deceive the Iris [identification system](https://www.sciencedirect.com/topics/computer-science/system-identification) by employing a preserved cadaver eye with the target user's Iris pattern. These can be tougher to detect if the contacts are correctly made. This deterrence causes someone to steal a deceased subject's eye and utilize it quickly to be effective [[92](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib92)].

[*Phishing Attacks*](https://www.sciencedirect.com/topics/computer-science/phishing-attack): This attack is widespread and frequently used to steal sensitive user information. It happens when a hacker poses as a reliable entity and tricks people into downloading an attachment or entering personal information on a phoney website, which leads to the installation of malware or the revealing of sensitive data. Specialized phishers target the absence of specialized active security measures by systems and the lack of knowledge or attentiveness of users with compromised attacks that combine social engineering and sophisticated tactics known as compromised attacks for critical infrastructures. Zero-day malware, link manipulation, filter evasion, obscuring brand logos, website forgery, covert redirect, etc., are some of the tactics used. These are initially targeted at vendor/remote websites, followed by the hacking of [IIoT](https://www.sciencedirect.com/topics/computer-science/industrial-internet-of-things) systems and, generally speaking, the taking over of operational systems that are linked to it [[93](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib93)].

* The IIoT is often accessed or entered at a front-end level by malicious users. He stays there while doing [reconnaissance](https://www.sciencedirect.com/topics/computer-science/reconnaissance) and network mapping until the best opportunity to launch a broad attack is identified. He then pivots between different systems to apply the necessary exploits to hack ICS systems [[93](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib93)].

SQL Injection Attacks: As mentioned in Section [7.1.3](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.3).

Reverse Engineering Attacks: Reverse engineering attacks against IoT (Internet of Things) devices entail disassembling, examining, and comprehending the inner workings of these devices to find vulnerabilities, exploit flaws, and perhaps obtain unauthorized access or control over them. Although reverse engineering might be a difficult task, it is a vital tool for anybody involved in IoT security. Security experts may discover possible attack mechanisms, create new security protocols, and enhance system security by studying the underlying processes of a device or system. Because IoT devices sometimes have low computing capabilities and may not be equipped with sufficient security protections, these attacks can represent substantial security threats [[94](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib94)].

* Although usually based on [static analysis](https://www.sciencedirect.com/topics/computer-science/static-program-analysis), reverse engineering may occasionally be carried out through dynamic analysis while utilizing [debugger](https://www.sciencedirect.com/topics/computer-science/debugger) tools. Reverse engineering attacks on IoT devices can serve various purposes, including extracting sensitive data (passwords, encryption keys, personal information), changing a device's firmware to allow unlawful functionality, creating bespoke firmware for maliciousness and taking control of a device to create a botnet or perform remote control operations [[95](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib95)]. Some of the common steps in reverse engineering attacks are [[96](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib96)]:

Device Acquisition & Inspection of the Device: Acquire the desired IoT device. This might entail buying the gadget, getting one used, or discovering one in a controlled setting. Crack up the gadget to see what's inside. Obtaining access to the electronics and memory can entail removing the shell, soldering, or doing other physical operations.

Extraction of Firmware and Data: Remove the firmware (software that regulates the operation of the device) from the device's memory. To do this, it may be necessary to replicate the firmware from storage components like flash memory using specialist tools and methods.

Analyzing the Firmware: Examine the extracted firmware to learn more about its components, code, and structure. This could entail looking at the code and data structures with tools like disassemblers, decompilers, and debuggers.

Vulnerability Identification: Look for vulnerabilities in the firmware, such as buffer overflows, unsafe communication protocols, inadequate encryption, and hardcoded passwords.

Exploitation: Use the vulnerabilities to your advantage to take over the system or get access without authorization. This can entail creating malicious code, inserting it into the firmware, or taking advantage of software flaws.

Modification or Repackaging: Change the firmware to accomplish certain objectives, including backdoors, changing functionality, or modifying the gadget for malicious purposes.

[*Sniffing Attacks*](https://www.sciencedirect.com/topics/computer-science/sniffing-attack): As mentioned in the Section [7.1.2](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.2).

Logic Bombs: A logic bomb is a sequence of instructions in a program that may attack an operating system, a program, or a network by delivering a [malicious payload](https://www.sciencedirect.com/topics/computer-science/malicious-payload). It doesn't start unless certain criteria are satisfied. These restrictions might be as straightforward as a certain day or hour. An even more complicated illustration is when a company terminates an employee and records the termination in its database. Logic bombs frequently include a [computer worm](https://www.sciencedirect.com/topics/computer-science/computer-worms) or virus. Even though some individuals confuse the two names, they don't refer to the same kinds of malware [[97](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib97),[98](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib98)].

* A logic bomb technically isn't malware, but the code inside is harmful. A logic bomb attack is often a sort of cyber sabotage conducted by an inside attacker, who is typically malicious, in contrast to certain malware-like viruses or worms that break into a safe system on their own. An example of this may be a dissatisfied current or former employee who has access to sensitive data or administrative control over systems, such as a programmer or information technology administrator [[97](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib97),[98](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib98)].
* Logic bombs might be made by workers who fear being dismissed as retaliation against their employers. They may disarm the explosives daily if their firms still employ them. Once free, they can fire attacks whenever they choose to do the most damage. Logic bombs also provide certain external hazards from [malicious programs](https://www.sciencedirect.com/topics/computer-science/malicious-program). WORM\_SOHANAD.FM is one example. Logic bombs can be hazardous due to their secrecy [[97](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib97),[98](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib98)].
* In addition to the fact that they are dormant and ready to explode like a volcano, their payloads pose a mystery hazard. These strikes have the potential to surprise their targets completely. Furthermore, it might be difficult to identify the threat actor who attacked since evidence can be erased when a logic bomb is being finished. Additionally, attackers might utilize the additional time to hide their identities [[97](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib97),[98](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib98)].

Exploitation of a Misconfiguration: Several systems and components must be configured for IoT applications to function successfully. Consequently, each of these components has to be correctly configured for security. They are easily exploitable by a malicious actor if they are not correctly set. Operating systems, servers, frameworks, database management systems, and any other applications must be appropriately set up for a secure IoT environment [[70](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib70)].

Cross-site Scripting: As mentioned as in Section [7.1.3](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.3).

#### 7.1.4. Encryption attacks

IoT security threats may include attacks on [encryption schemes](https://www.sciencedirect.com/topics/computer-science/encryption-scheme). Side-channel attacks target the implementation of cryptographic methods rather than the algorithms themselves. By examining physical measurements made during calculation and the [internal state](https://www.sciencedirect.com/topics/computer-science/internal-state) of the physical device during processing, attackers can deduce the encryption key [[99](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib99)].

[Cryptanalysis](https://www.sciencedirect.com/topics/computer-science/cryptanalysis) attacks look for vulnerabilities in the [cryptographic algorithm](https://www.sciencedirect.com/topics/computer-science/cryptographic-algorithm) to derive encryption keys. Cryptanalysis attacks can be ciphertext-only, chosen-plaintext, adaptive-chosen-plaintext, chosen-ciphertext, or adaptive-chosen-ciphertext, depending on the information the attacker has access to Ref. [[99](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib99)].

A malicious actor can intercept communications between two users and use those communications to decrypt data using keys shared by both users, making [encryption methods](https://www.sciencedirect.com/topics/computer-science/encryption-method) vulnerable to MitM attacks. Users continue to believe they are simply speaking to one another, as they have done in previous MitM attacks [[99](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib99)].

#### 7.1.5. Data attacks

The term “data attack” in the context of the IoT refers to a variety of malicious actions intended to jeopardize the availability, confidentiality, integrity, and general security of data inside IoT systems. The IoT is a network of interconnected smart devices that gather, share, and analyze data, making them vulnerable to various data-related attacks. Some common data attacks in IoT include:

Data Interception and Eavesdropping: Attackers intercept and eavesdrop on the communication between IoT devices and data servers. This can lead to unauthorized access to sensitive information, such as [personal data](https://www.sciencedirect.com/topics/computer-science/personal-data), passwords, and confidential business data.

Data Tampering: The data exchanged between IoT devices and servers are subject to alteration by malicious actors. This may entail the alteration of sensor readings, directives, or any other kind of data transmission. An instance of tampering might be manipulating temperature sensor readings within an industrial environment, resulting in erroneous decision-making processes based on unreliable data.

Replay Attacks: As mentioned in Section [7.1.2](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.2).

Man-in-the-middle (MitM) Attacks: As mentioned in Section [7.1.3](https://www.sciencedirect.com/science/article/pii/S2949715923000793#sec7.1.3).

Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks: As mentioned in Section [7.1.2](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "sec7.1.2).

Data Falsification: A data falsification attack within the context of the IoT pertains to a malicious act in which an attacker deliberately modifies or manipulates data inside an IoT system to deceive, disrupt, or compromises its functioning, integrity, or security. Attackers introduce fabricated information into the system by capitalizing on weaknesses or actively changing device inputs. This phenomenon can potentially result in erroneous decision-making processes founded upon facts of questionable reliability [[100](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib100)].

Device Impersonation: Device spoofing refers to a method of attack when a malicious device alters the IP address, MAC address, or other identifying details of a genuine device, therefore assuming the identity of a valid device [[101](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib101),[102](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib102)].

Firmware and Software Exploitation: As mentioned in the Section [7.1.3](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "sec7.1.3).

Insecure APIs and Interfaces: The presence of vulnerabilities in APIs (Application Programming Interfaces) and interface services or frameworks utilized for communication with IoT devices can result in unauthorized access to or modification of data.

#### 7.1.6. Side channel attacks

Side channel attacks are a class of attacks that exploit “side channel information”, which refers to information that may be obtained from [encryption devices](https://www.sciencedirect.com/topics/computer-science/device-encryption) other than the plaintext or the [ciphertext](https://www.sciencedirect.com/topics/computer-science/ciphertext) produced by the [encryption process](https://www.sciencedirect.com/topics/computer-science/encryption-process). Encryption devices provide quantifiable time information, statistics on power usage, and several other data.

Side channel attacks leverage several types of information to get the cryptographic keys employed by the targeted device. The premise of this statement is rooted in the observation that logic operations possess physical attributes contingent upon the input data they receive. Instances of side channel information include temporal attacks, power analysis attacks, interference analysis attacks, electromagnetic attacks, and environmental attacks [[2](https://www.sciencedirect.com/science/article/pii/S2949715923000793#bib2),[103](https://www.sciencedirect.com/science/article/pii/S2949715923000793" \l "bib103)].