# 14.2.6 IoT Security Challenge Facts

Internet of Things (IoT) technology is growing and affecting more and more aspects of daily life. Manufacturers are competing for the chance to create and release the latest and greatest devices. However, in the process of beating their competitors to the market, most manufacturers are neglecting the security issues connected with data access and management, as well as the security of IoT devices. These circumstances allow hackers to easily exploit IoT. This lesson will explore some of the challenges and attacks currently plaguing the IoT.

This lesson covers the following topics:

- IoT security challenges
- IoT attacks
- OWASP vulnerabilities

## IoT Security Challenges

| Challenge | Description |
|---|---|
| Lack of security and privacy | Smart devices collect important data for many reasons, such as improving efficiency, enhancing experience, improving service, and assisting decision-making. The problem is that most IoT devices and services lack the most basic security and privacy policies required to protect all this data.<br><br>It's imperative to store and process data securely across the network. This means redacting or obfuscating sensitive data before storing it. An alternative is to use data separation to decouple personally identifiable information from data payloads. Data that is not useful or is no longer important should be disposed of in a safe manner as well. |
| Difficult-to-update firmware and OS | Each device should undergo proper testing before being released to the market, and updates should happen regularly. There are a few reasons why updates happen infrequently, if at all, including:<br><br>▪ The update might break the device's functionality.<br>▪ Most manufacturers are more concerned with producing and releasing products faster than their competition, so they don't take time to consider updates or security risks.<br>▪ Manufacturers often stop releasing updates and support for devices once they start a new project.<br>▪ Many IoT devices use unsupported legacy Linux kernels. |
| Vulnerable web interfaces | Most IoT devices are made with embedded server technology. Although it's more practical and user friendly, it makes the devices vulnerable to attacks such as ransomware. Implementing secure authentication for both users and apps can help avoid ransomware attacks. |
| Default, weak, and hardcoded credentials | Many IoT devices allow weak or default passwords, which are easily broken by hackers. One problem is that there's no set regulation for IoT authentication, only guidelines. Some ways to strengthen authentication on IoT devices are two-factor authentication (2FA) and enforcing strong passwords or certificates. |
| Clear text protocols and open ports | Data encryption and decryption is an ongoing process. Most IoT network sensors are not capable of supporting this process, which means that most data is being transferred and received as clear text. In addition, many times, ports listening to the internet are left open continuously. Both of these factors make the data extremely weak against theft, breaches, and other malicious acts. |

## IoT Attacks

The following tables lists common attacks on IoT devices.

| Attack | Description |
|---|---|
| DDoS attack | In this attack, the hacker exploits vulnerabilities to take over and use all the devices in the IoT network as a zombie army to target a server or system. When this happens, services on the device are unavailable. |
| Exploiting HVAC | HVAC stands for heating, ventilation, and air conditioning. HVAC includes different systems, machines, and technologies that provide comfort through environmental regulation in most indoor settings. Hackers exploit HVAC systems to retrieve confidential information from users as well as to take over a network. |
| Ransomware attack | In this malware attack, the hacker utilizes encryption to deny a user access to the device by locking files or even the screen. |

## OWASP vulnerabilities

OWASP stands for Open Web Application Security Project. OWASP is a nonprofit organization made up of software developers, engineers, and freelancers that provides tools and resources for web app security. From time to time, OWASP publishes a report on the 10 most serious web app security risks affecting the cyber world. The following table describes these risks.

| Risk | Description |
|---|---|
| Injection | In this type of attack, a hacker injects malicious code to trick an application into performing actions it wouldn't otherwise perform. Some of the most common injection attacks use Structured Query Language (SQL) and Lightweight Directory Access Protocol (LDAP).<br><br>SQL injection involves a hacker inserting an SQL malicious statement that exposes content being sent to an application. LDAP works in a similar way, but it targets a directory system. For example, the hacker could insert malicious code into a form that accepts usernames. If the form input is unsecure, it will execute the malicious code and expose the usernames.<br><br>The best protection against injection attacks is to validate and sanitize data. Validating means to reject suspicious data. Sanitizing means getting rid of suspicious parts of the data. Additionally, setting controls that limit the amount of information that can be exposed by injection can be helpful. |
| Components with known vulnerabilities | Most web application developers use libraries and frameworks while building their web applications. The libraries and frameworks save time and effort by avoiding redundant work and supplying necessary functionality. However, many hackers look for weaknesses in those libraries and frameworks to perform massive attacks. Most libraries and frameworks are updated and patched regularly to provide better security.<br><br>The problem is that web application developers sometimes don't have the most recently updated version of the components running on their app. The best way to prevent this problem is for web application developers to remove all unused components from their apps. They should also verify that they are using components from a trusted source and the components are the most recently updated version. |
| Broken authentication | Weak authentication systems are vulnerable to attackers who steal genuine user identities and use them to access data or compromise a system. Use strong authentication and session management controls to protect user identities. FIDO, 2-factor authentication (2FA), and Rate Limit are good tools to consider. |
| Sensitive data exposure | Man-in-the-middle attacks are possible when sensitive data is not properly secured by applications. Encrypt all sensitive data stored or in transit and avoid caching it as much as possible. Discard sensitive data as soon as possible. |
| XML external entities (XXE) | Some applications use XML input to load the contents of external files. A hacker could exploit the XML processor by inserting malicious code that makes the processor send the content from the hard drive and other systems to the hacker. One remedy is to turn off the capability of using external entities in the XML processor . It's also a good idea to use Static Application Security Testing (SAST). |
| Broken access control | This vulnerability has to do with users gaining access to other users' data or performing actions they don't have permissions for. Ensure that authorization tokens are used, and set tight controls on them. |
| Security misconfiguration | This is a very common and easily detectable vulnerability. If an app implements a lot of default configurations or displays lengthy, detailed error messages, it can be exploited very easily. Remove all unused features in the code and make error messages very general and vague. |
| Cross-site scripting (XSS) | XSS vulnerabilities allow an attacker to insert malicious code into a URL path, onto a website, or into an application. They use this malicious code to extract sensitive data, spread malware, or perform other malicious acts. Escape from untrusted HTTP requests and validate or sanitize all content generated by the users. Frameworks such as ReactJS and Ruby on Rails are also helpful against cross-site scripting. |
| Unsecure deserialization | Unsecure deserialization affects applications that serialize and deserialize data. This weakness allows a hacker to execute code in an application remotely, change or erase serialized objects, launch injection attacks, and elevate privileges. To avoid these problems, you can restrict the types of objects that are deserialized by the app and never allow the app to deserialize unstructured objects. |
| Insufficient logging and monitoring | Many applications don't log or monitor code activity enough. Therefore, it can take weeks or months before a breach is detected. This allows hackers to execute persistent attacks and cause a lot of damage.<br>Implement detailed logging and monitoring. Having an incident response plan is also very helpful. |