CHAPTER 6

Security Processes

Every modern enterprise uses a large amount and variety of data to work on various data-driven initiatives. Having various security processes in place is a must in order to properly and safely use data and create business insights from it.

The previous chapter discussed various patterns for building applications on the cloud. You also took a quick tour of various data security patterns that provide easier and safer access to the data stored in the cloud.

This chapter covers the following topics:

- Complete mediation with threat modeling
- Securing the infrastructure and application deployment
- Security testing
- Key management
- Vulnerability management
- Disaster recovery

Complete Meditation with Threat Modeling

Every company has a dedicated infrastructure and networking team. The application development team focuses more on the development lifecycle and leaves the security assessments and controls to the infrastructure and networking team. The majority of the developers think this way due to the following reasons:

- Security is implemented from Layer 2/Layer 3/Layer 4 of OSI model, as shown in Figure 6-1, and there are various security layers: Physical Security, Perimeter and Network Security, Secure Endpoints, Application Security, Data Security, and Mission-Critical Assets.

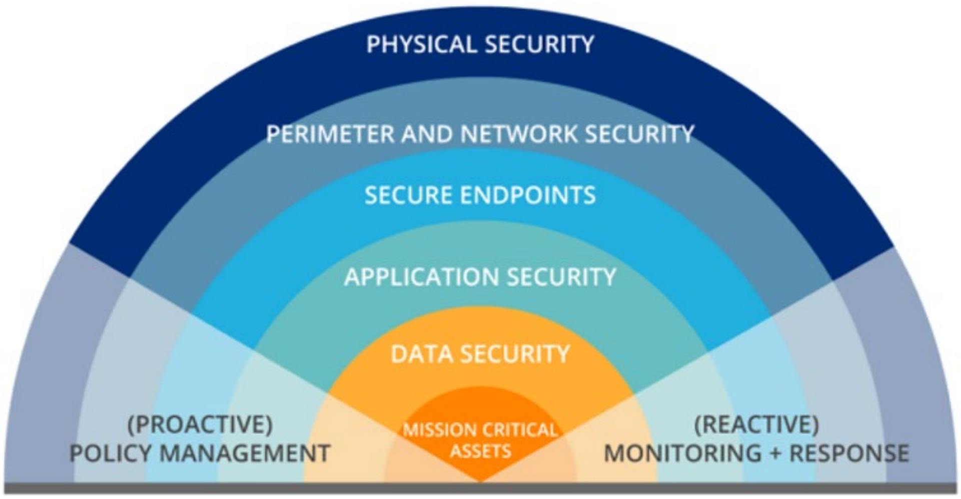

PHYSICAL SECURITY
PERIMETER AND NETWORK SECURITY
SECURE ENDPOINTS
APPLICATION SECURITY
DATA SECURITY
MISSION CRITICAL ASSETS
(PROACTIVE) POLICY MANAGEMENT
(REACTIVE) MONITORING + RESPONSE

Figure 6-1. *Security layers*

- Fault isolation and resiliency weren't considered during the design phase.

As displayed in Figure 6-2, thread modeling is an iterative process that starts by defining the threat, creating the threat modeling diagram, identifying the relevant threats, mitigating the threats, and validate them to see if the relevant threat is resolved.

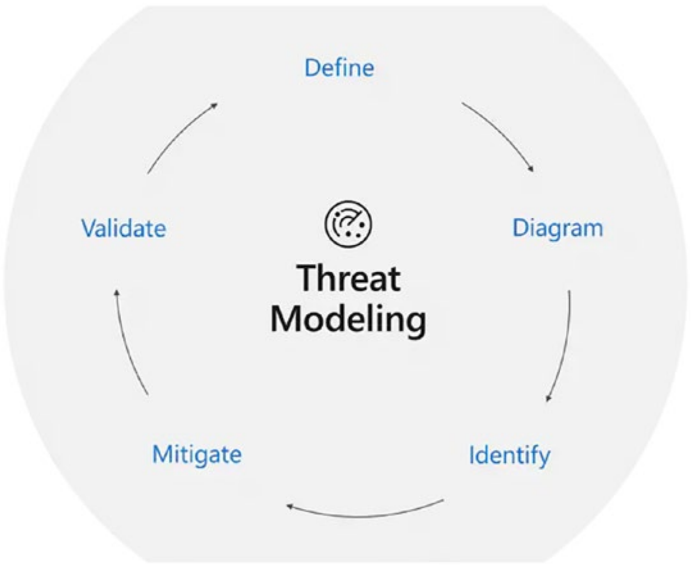


Figure 6-2. *Threat modeling process*

Things are changing fast due to greater public cloud adoption. In the past few years, distributed architecture coupled with DevOps has put forth new requirements like resiliency and fault isolation on applications and services. Application and services need to continue functioning correctly even in the presence of faults in underlying layers. This is also due to the fact that hardware and networks are moving to a commodity model leveraging PaaS models with on premises and public cloud options. Nonfunctional requirements (NFRs) like fault isolation and resiliency are becoming the norm of every application and service. This changing need has also put more focus on "security by design," bringing in the need to implement the security controls from within the application and services.

Enterprises need to determine the security controls which need to be implemented in the application and services. Security controls are not an IT need but a business need and hence the security controls need to be done in the frame of "business risk assessment", modeling threats with a focus on business impact. A beautiful approach to modeling security threats for your application and services is the STRIDE model. With this model, we can answer the questions like "What are the security threats in my application and services and what security controls does my team need to implement for an application or service?".

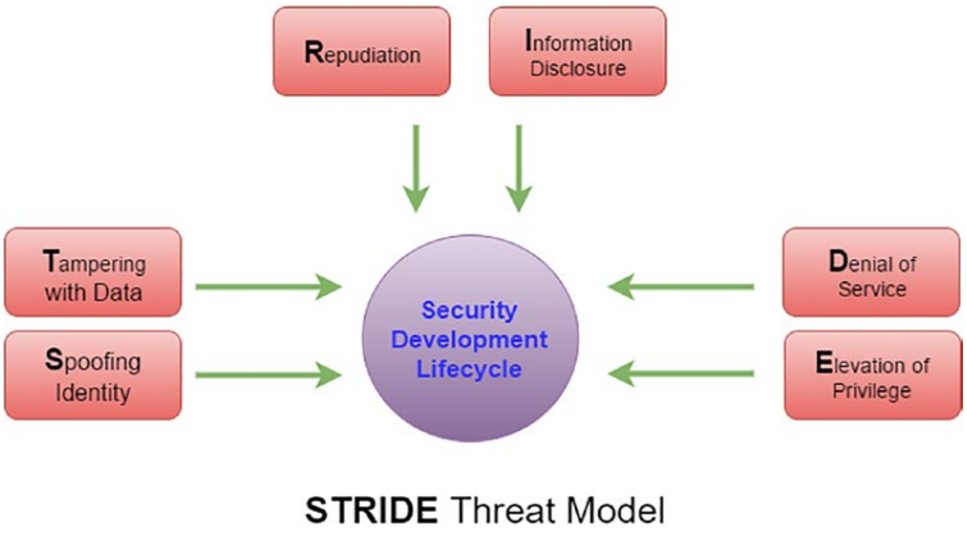


Figure 6-3. *STRIDE threat modeling*

Threat modeling is a process of creating an optimized application by identifying the objectives and security vulnerabilities to prevent the security issues to the application. It helps to identify the security requirements of the system or application which is mission critical and contains sensitive data. It provides a systematic and structured process to identify potential threats and the security vulnerabilities to reduce the risk of the IT resources.

With respect to software security, threat modeling is considered to be a very critical activity of the software design and development. Without evaluating and migrating the threats, it is impossible to build applications and systems which comply with the corporate security policies and regulatory requirements.

We look at the process of threat modeling in the next sections.

Form a Team

The team should include stakeholders, business developers, owners, and security experts. People from diverse backgrounds make the threat modeling process more robust and secure.

Define the Scope

Define the scope of the application to start the threat-modeling process. For example, does the developed application focus on applications, networks, or infrastructures? You can create a catalogue of all components and map them to the architecture diagrams.

Brainstorm and List Potential Security Threats

For all system components, determine if threats exist. With this exercise, you can build the roadmap, pinpoint the expected and unexpected threat scenarios, including the threat trees, and identify possible weakness and vulnerabilities.

Prioritize Threats

Determine the level of the threats and rank them to take appropriate risk-mitigation actions. One common approach to determining priority is to multiply the damage potential of the threat by its likelihood of happening.

Develop and Implement Risk Mitigation

Decide how to mitigate each threat or reduce the risk to an acceptable level to avoid any issues with the security measures. You can transfer the risk, reduce the risk, or accept the risk.

Document the Results

You can document all the findings of the threat-modeling process and related actions so future changes to the application, landscape, and the operating environment can assess and update the threat model. See Figure 6-4.

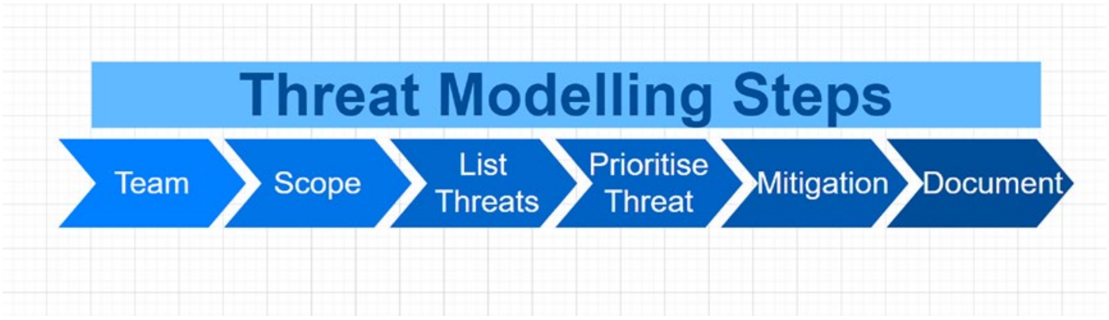


Figure 6-4. *Threat-modeling steps*

The threat modeling (aka the STRIDE) model works on the principle of data flows and trust boundaries. Each security threat in the STRIDE model is a potential violation that needs to be accessed for vulnerabilities in your application and system. If the risks are viable, you must implement the security controls.

Each letter of STRIDE stands for a security threat and is linked to a desired property. Here are the details:

S = Spoofing (linked to the authenticity of your application and services)

T = Tampering (linked to the integrity of the data)

R = Repudiation (linked to the beautiful requirement of non-repudiation)

I = Information disclosure (linked to the confidentiality of your data or parts of application and services)

D = Denial of service (linked to the availability of your application and services)

E = Elevation of privilege (linked to the authorization of your application and services)

STRIDE modeling must be done at the design phase and revisited in the development and testing phases. Security needs to be done from the perspective of capturing all requirements as early as possible. The STRIDE model discusses the potential security risk per threat and marks them against their probability and impact. The risks are seen from inside as well as outside attack vectors. The STRIDE model can reveal threats and, with proper analysis, the trust boundaries can be determined and security controls that need to be implemented can be captured.

It is best to couple STRIDE threat modeling with pattern-based security. Based on the trust boundaries identified from the threat modeling, you can map the security controls to a particular security zone and then implement the controls aligned to this security zone. This keeps security zones and security controls centralized and brings the security team closer to engineers. In addition to this, the model is easily to for any new application or service. The beauty of this approach from an architectural perspective is simplicity while serving the requirements of fault isolation and resiliency for the applications and services across the Development, Test, Acceptance, and Production (DTAP) environments.

After looking at the threat modeling process and its steps in detail, it’s now time to dive into how you can secure the infrastructure and application deployment process.

Securing the Infrastructure and Application Deployment

Every enterprise must have a well-defined software development lifecycle process to deploy applications securely with proper security checks in place during the design, development, testing, and deployment stages. It is best to create a layered system architecture that uses standard frameworks to design an application for the identity, authorization, and access control. Let’s look at a few options to improve security of the infrastructure and application deployment.

Automate Security Releases

Considering the wide range of security vulnerabilities, it is very difficult to deploy, update, and patch application environments to meet security standards without using the automated security tools. In order to automate the security releases, it is best to create continuous integration/continuous deployment pipelines (CI/CD). One advantage of creating automated CI/CD pipelines is that you remove the manual errors, provide a standardized development feedback loop, and increase the speed of deployment for the new features. You can also use automation to scan the security vulnerabilities when the artefacts are created and define policies for different environments (development, test, acceptance and production—DTAP) to verify that the artefacts have been deployed. See Figure 6-5.

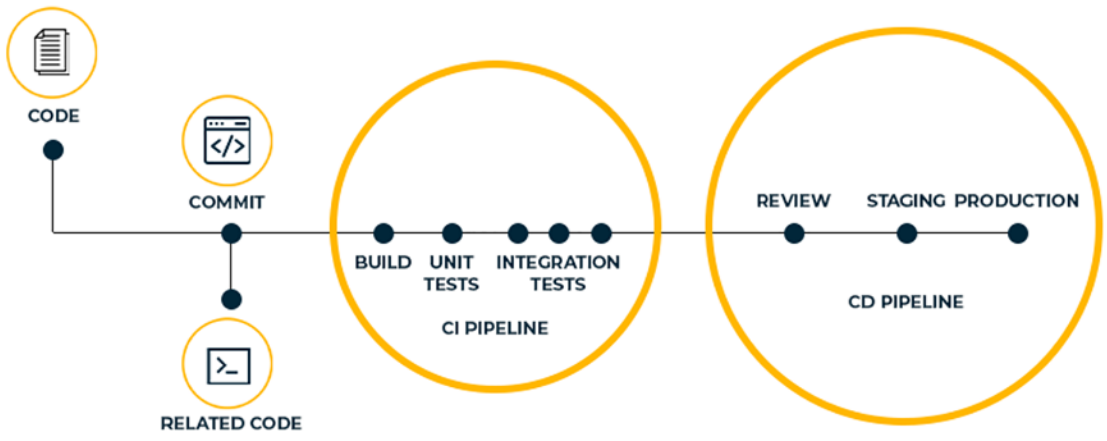


***Figure 6-5.** Automated security with CI/CD pipeline*

There are certain considerations while automating security with CI/CD pipelines. You need to protect the CI/CD pipeline using Access Control.

You need to start by defining who can send code changes to the repository, which will be used as a storage base to ensure the first layer of protection for the automated pipeline. You also need to ensure what is sent, stored, and moved into the CI/CD pipeline. Another point to ensure is that the security of the process occurs with the least possible intervention. See Figure 6-6.

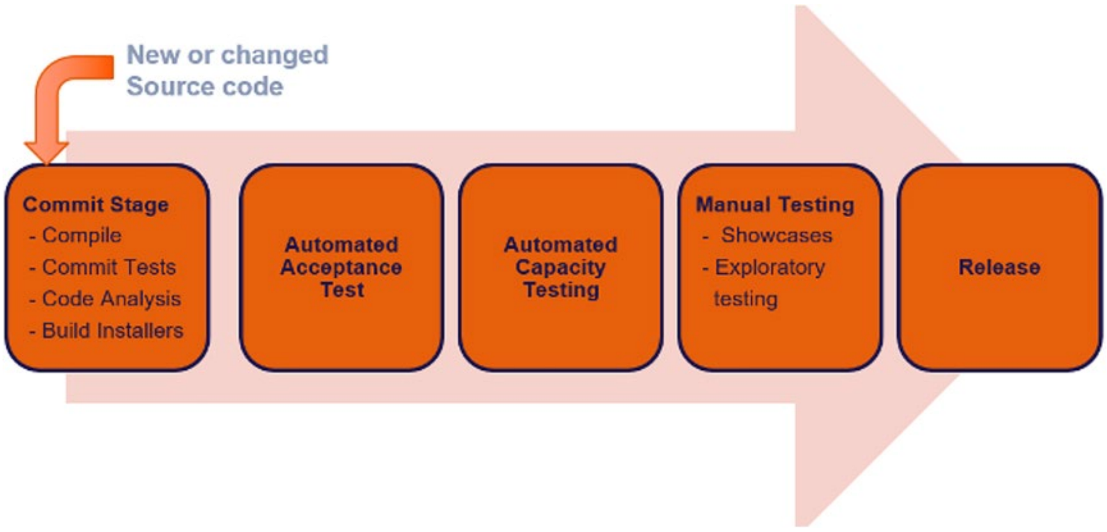


Figure 6-6. *Automated CI/CD pipeline configuration*

Well-Governed Application Deployment

If an attacker compromises the CI/CD pipeline, the entire technology stack can have a major impact. In order to secure the pipeline, you need to enforce the established approval process to deploy the code into production. See Figure 6-7.

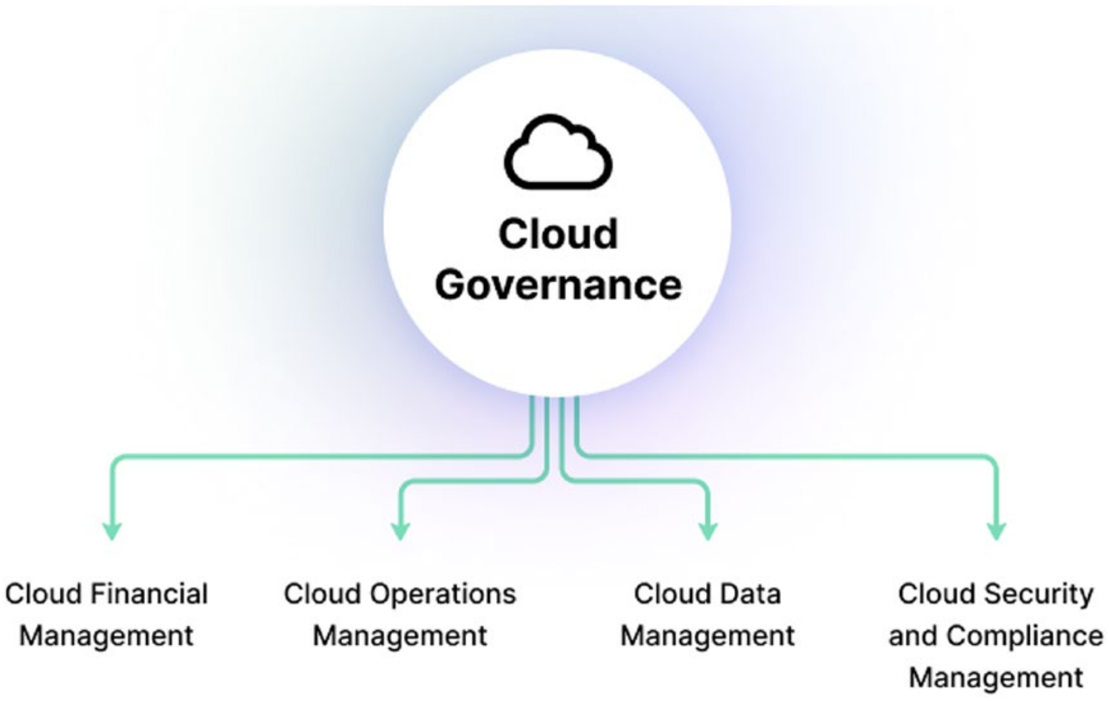

Cloud Governance
Cloud Financial Management
Cloud Operations Management
Cloud Data Management
Cloud Security and Compliance Management

Figure 6-7. *Application deployment governance*

Scan Security Vulnerabilities

It is best to use automated tools to continuously perform a vulnerability scan before packages/containers are deployed to the production environment. See Figure 6-8.

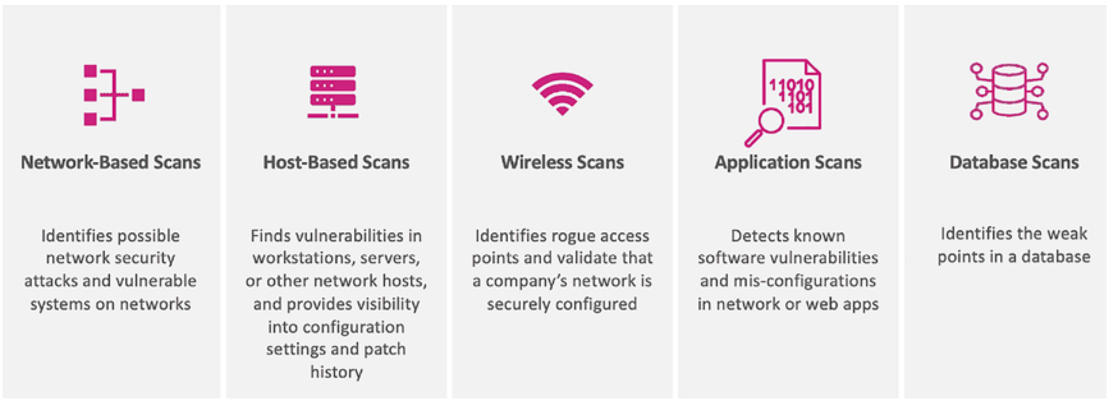

Network-Based Scans
Identifies possible network security attacks and vulnerable systems on networks
Host-Based Scans
Finds vulnerabilities in workstations, servers, or other network hosts, and provides visibility into configuration settings and patch history
Wireless Scans
Identifies rogue access points and validate that a company's network is securely configured
Application Scans
Detects known software vulnerabilities and mis-configurations in network or web apps
Database Scans
Identifies the weak points in a database

Figure 6-8. *Security vulnerability scans*

Monitor Application for Security Vulnerabilities

In order to proactively react to security vulnerabilities, it's best to constantly monitor the application code for vulnerabilities. For example, you can use the web security scanner to identify the security vulnerabilities in the app engine, compute engine, or web applications. This web scanner helps identify vulnerabilities, including XSS (cross-site scripting), Flash injection, mixed content (HTTP in HTTPS for example), and insecure libraries. If you have a monitoring mechanism in place, you can easily detect such vulnerabilities beforehand to tackle those security challenges at the early stage of the application development and deployment process. See Figure 6-9.

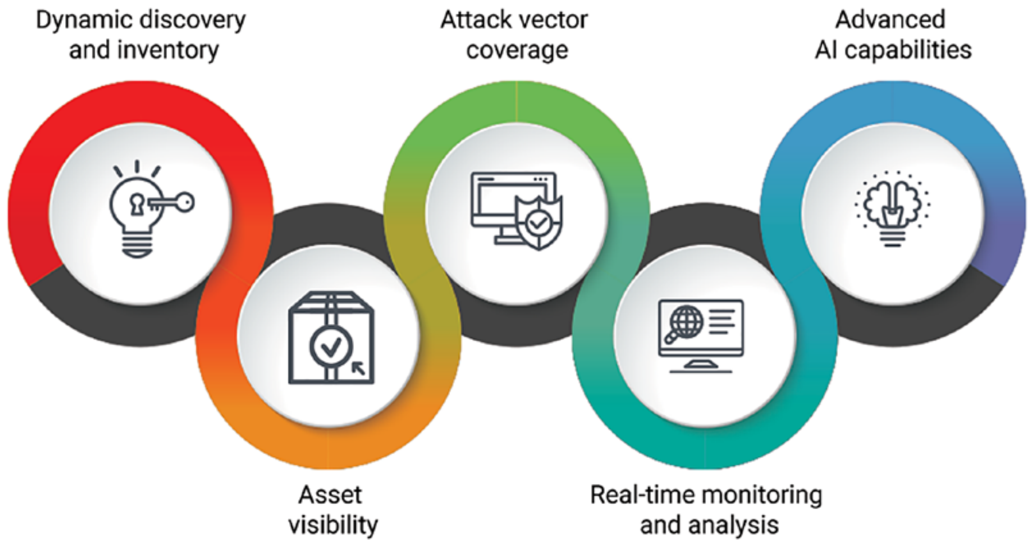

Dynamic discovery and inventory
Attack vector coverage
Advanced AI capabilities
Asset visibility
Real-time monitoring and analysis

Figure 6-9. *Security vulnerability management*

Control the Data Movement

In order to control and secure the data stored in cloud services, you need to configure the cloud resources with correct security configurations.

Infrastructure as a Code (IaC)

All operational changes and modifications should be done through the Infrastructure as a Code pipelines. IaC is a key DevOps practice and it is often used to enable continuous integration and delivery. See Figure 6-10.

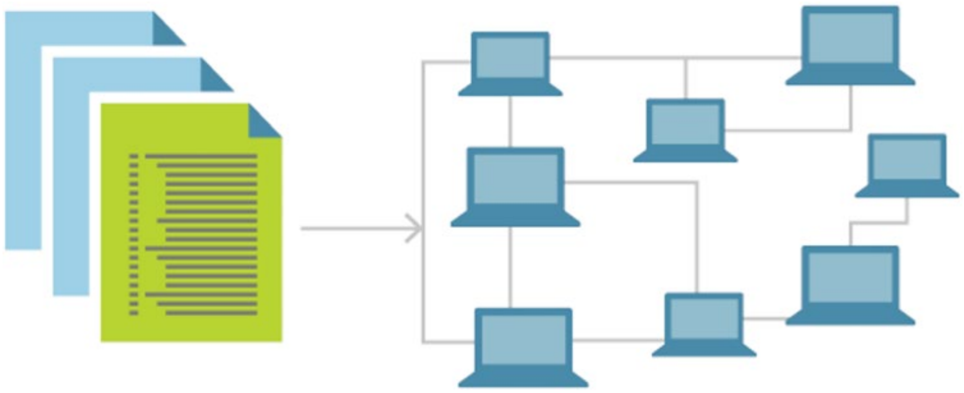


Figure 6-10. *Infrastructure as a Code*

It reduces the manual effort required to do all the configurations and automate the environment deployment using the Infrastructure as a code pipeline. It is a vital practice for the DevOps adoption. It makes the release changes to the production faster and reduces the time to market.

Pipeline Secret Management

Confidential certificates, keys, and secrets used in the deployment pipeline should be stored in the Key Vault (AKV). When you do the deployment of the application infrastructure with the Azure Resource Manager (ARM), Bicep or Terraform, you have various credentials to connect to different services. It is best to use confidential secrets from the secret management tools like Key Vault (AKV) instead of hard coding these values. See Figure 6-11.

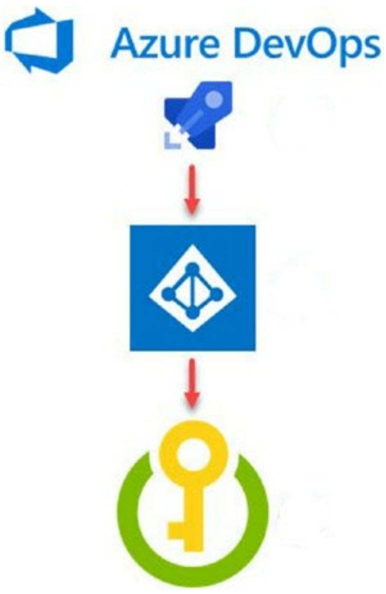


Figure 6-11. *Azure Key Vault for Pipeline secrets*

Adhere to the Principle of Least Privilege

Based on the principle of least privilege, the main goal is to grant only required permissions. If possible, it is also best to grant access to the roles and not to the individual users. Restrict the access as much as possible:

1. Restrict admin access to users as much as possible
2. Users with higher access can change security objects such as roles, users, or permission management
3. Roles that have capability to add, change, or remove security privileges

The principle of least privileged access means that users are only given the privileges that they need to perform their jobs efficiently.

Let's now look at how to perform security testing on an application.

Security Testing

With cloud computing, you can access and use IT resources over the Internet with the pay-as-you-use cost principle. You can access technology and cloud services using the compute power, storage, and databases instead of buying, owning, and maintaining physical data centers and servers. There are many public cloud providers available in the market, including Azure, Google Cloud, and AWS. See Figure 6-12.

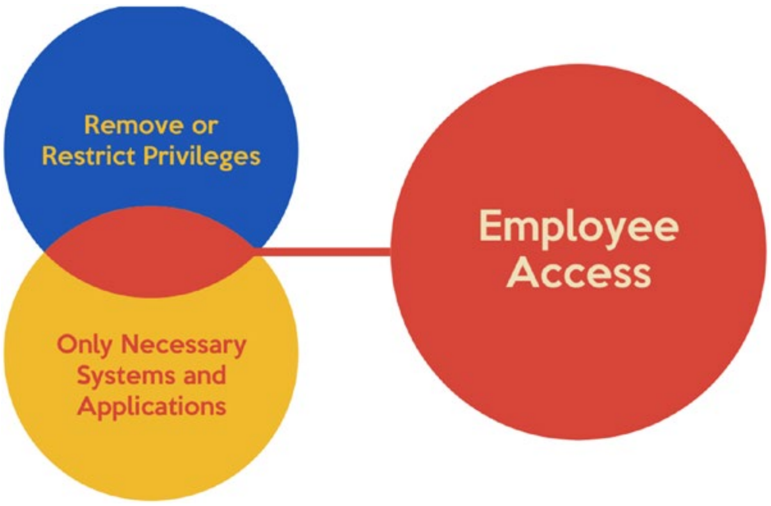

Remove or Restrict Privileges
Only Necessary Systems and Applications
Employee Access

Figure 6-12. *Public cloud providers*

As cloud provider popularity increases day by day, attackers focus more on cloud services and security vulnerabilities. Attackers use security attacks against these managed cloud service providers. Enterprise organizations should focus on securing cloud resources.

In order to avoid cloud security attacks, it is best to enable cloud penetration testing, which is an attack performed to find security vulnerabilities that could cause issues or misconfigurations in a cloud-based system. See Figure 6-13.

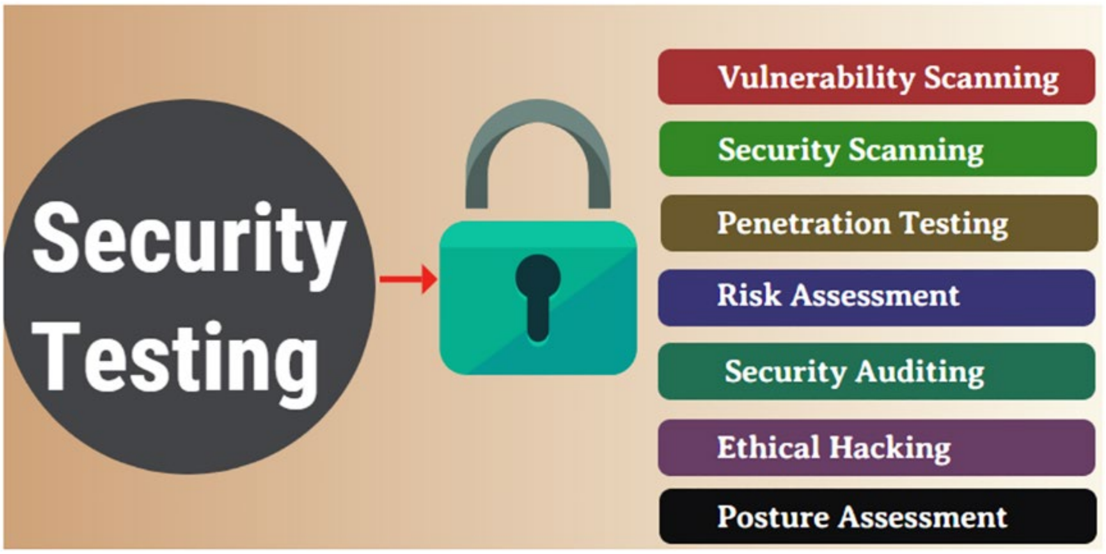


Figure 6-13. *Cloud security testing*

Security testing is an essential step in the software development lifecycle. In security testing, it uncovers security vulnerabilities of the system and ensures that data and system resources are protected from attackers. With security testing, you can also ensure that software systems and applications are free from threats and risks.

The main goal of security testing is as follows:

- Identify the threats in the system
- Measure and detect security vulnerabilities on time
- Proactively detect every possible security risk in the system
- Enable integration with third-party tools for code scanning and vulnerabilities

During security testing, the main goal of the principles are as follows (see Figure 6-14):

- Confidentiality
- Integrity
- Authentication
- Authorization

- Availability

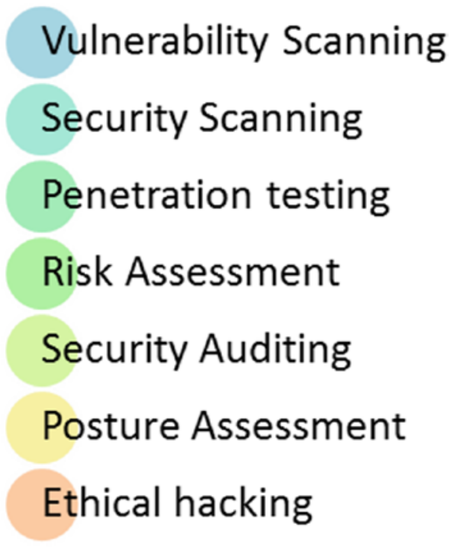


Figure 6-14. *Security testing types*

Let's look at the types of security testing.

Vulnerability Testing

It is best to enable this type of testing performed with the help of an automated software. The main goal is to scan the entire system and application to detect the known vulnerabilities.

Security Scanning

During security scanning, the main goal is to identify network and system weaknesses. This type of scanning provides solutions for reducing the risks or defects.

Penetration Testing

With penetration testing, the main goal is to simulate attacks from attackers to tackle those challenges proactively. This includes an analysis of the system to examine potential vulnerabilities from the hacker to hack the system. See Figure 6-15.

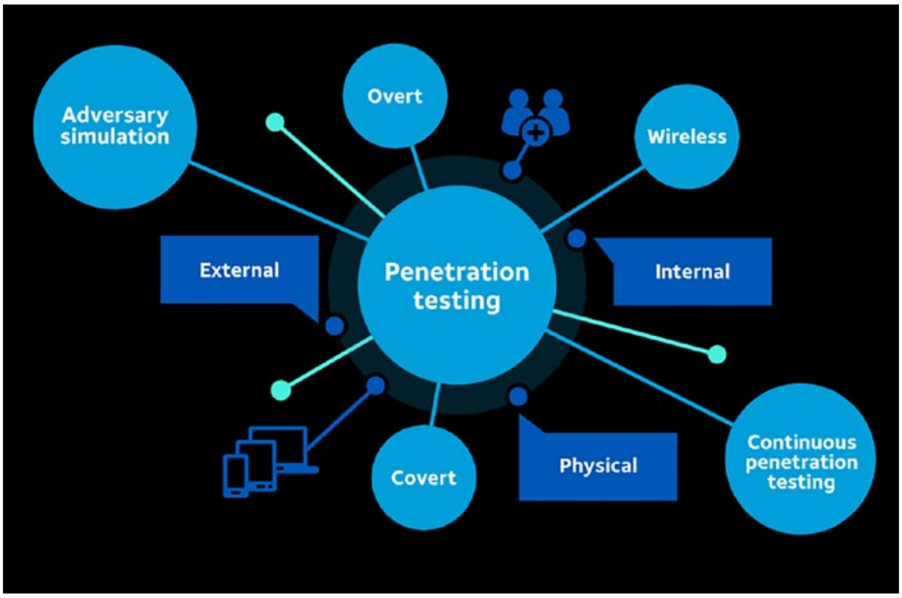
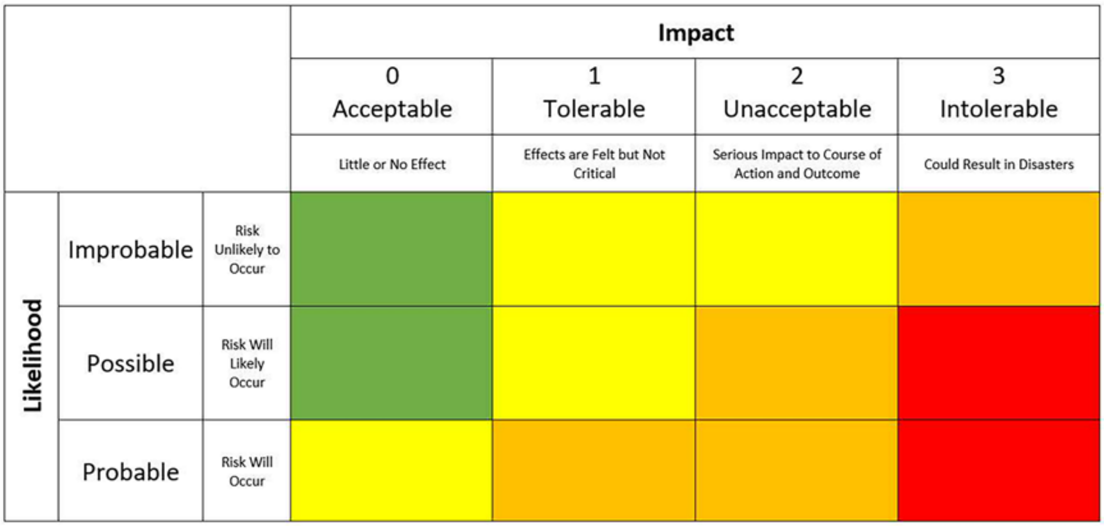


Figure 6-15. *Penetration testing*

Risk Assessment

Risk assessment mainly consists of security risks that were observed in the organization to determine the likelihood and the impact of the risk. Risks are divided into the three types: low, medium, and high. See Figure 6-16.



			Impact			
			0 Acceptable	1 Tolerable	2 Unacceptable	3 Intolerable
			Little or No Effect	Effects are Felt but Not Critical	Serious Impact to Course of Action and Outcome	Could Result in Disasters
Likelihood	Improbable	Risk Unlikely to Occur	🟩	🟨	🟨	🟧
	Possible	Risk Will Likely Occur	🟩	🟨	🟧	🟥
	Probable	Risk Will Occur	🟨	🟧	🟧	🟥

Figure 6-16. *Risk assessment matrix*

Security Auditing

Security auditing is an internal inspection of the software application and OS to identify security defects. Security audits can be performed in various ways by looking at the code line by line. You can start by the planning, scoping, and logistics of the security auditing to be placed across the applications. The next step is to collect data and gather evidence. Once the data is collected, you perform the analysis, interpret the data, and generate the reports. Once that is done, you create an action plan to fix those findings. See Figure 6-17.

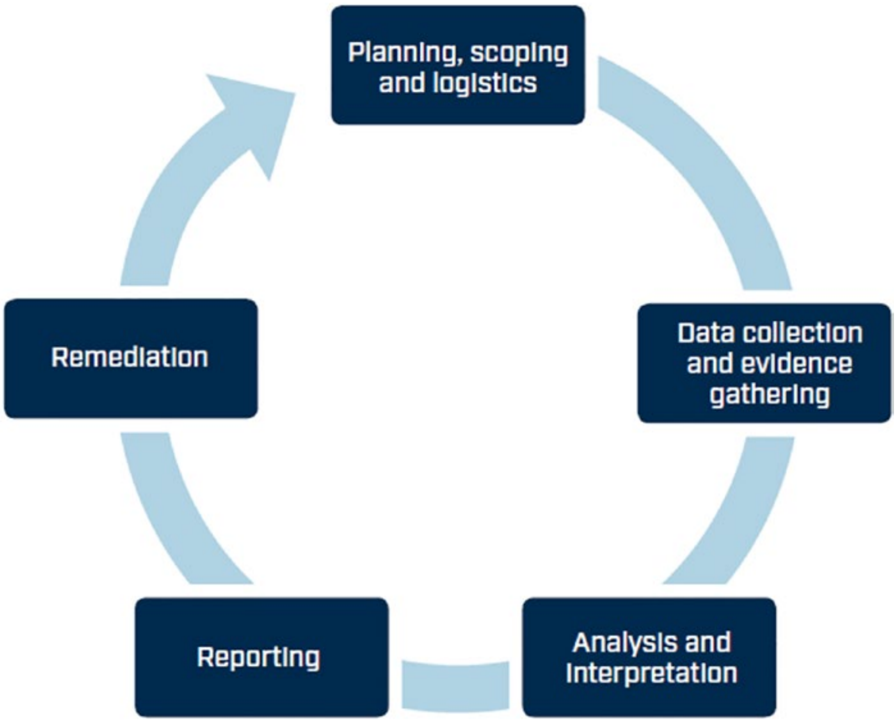

Planning, scoping and logistics
Data collection and evidence gathering
Analysis and interpretation
Reporting
Remediation

Figure 6-17. *Security audit process*

Ethical Hacking

The purpose of ethical hacking is to expose security flaws in the system. It involves an authorized attempt to get unauthorized access to an application, system, or data.

Ethical hacking follows these four concepts:

- Legal: Gather approval to access and perform the security assessment.
- Define scope: Determine the scope to perform the ethical hacking so that hacking activities will remain legal and within the organization's approved boundaries.
- Record vulnerabilities: Once all the vulnerabilities are recorded, notify the organization of all the vulnerabilities and take action to prevent the security issues.

- Posture assessment: This is a combination of ethical hacking, security scanning, and risk assessments to provide an overall security of the organization. This assessment is done to ensure that cybersecurity practices are followed in the organization. Data breaches, cyberattacks, and online threats have become major threats for many organizations. Security posture assessment is calculated based on resources such as people, hardware, and software capabilities whenever a new virus attacks. It shows the security health of the product or the application. There are various levels of the assessment, which indirectly mean posture assessment. See Figure 6-18.

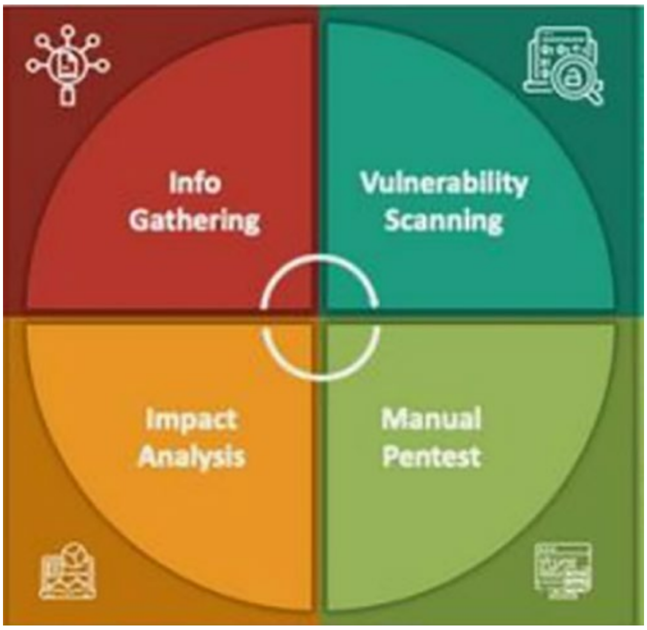


Figure 6-18. *Posture assessment*

After understanding the types of security testing, let's look at the process of security testing. You must include a security testing phase in the software development lifecycle phase. See Figure 6-19.

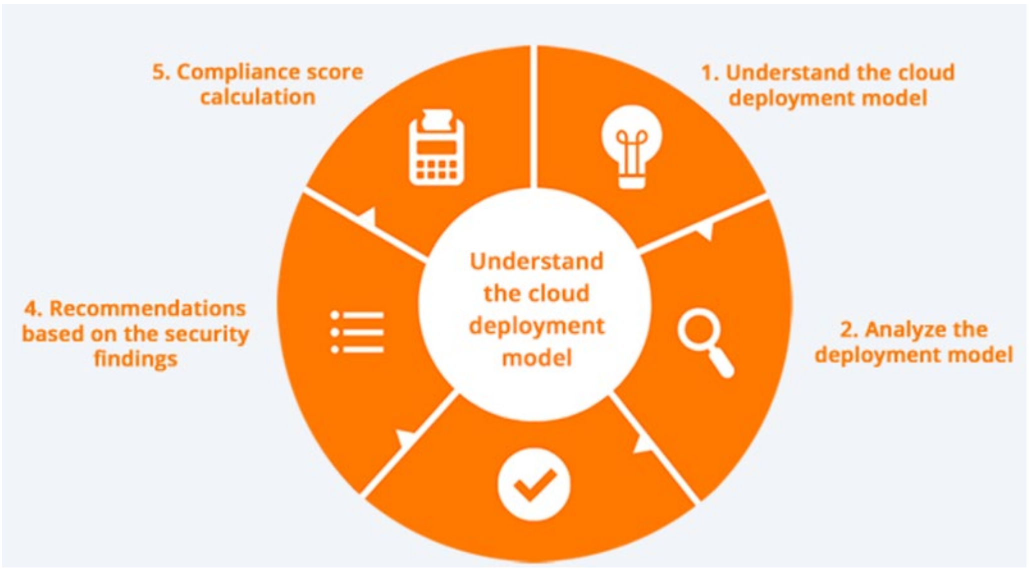

5. Compliance score calculation
1. Understand the cloud deployment model
Understand the cloud deployment model
4. Recommendations based on the security findings
2. Analyze the deployment model

Figure 6-19. Security testing process

Requirements

Gather requirements for the security analysis to check if the application is compatible with the security standards.

Design

Security test plans should be designed and include the development plan for all security tests.

Unit Testing

This is the smallest, testable part of an application that can be individually tested to ensure functionality. See Figure 6-20.

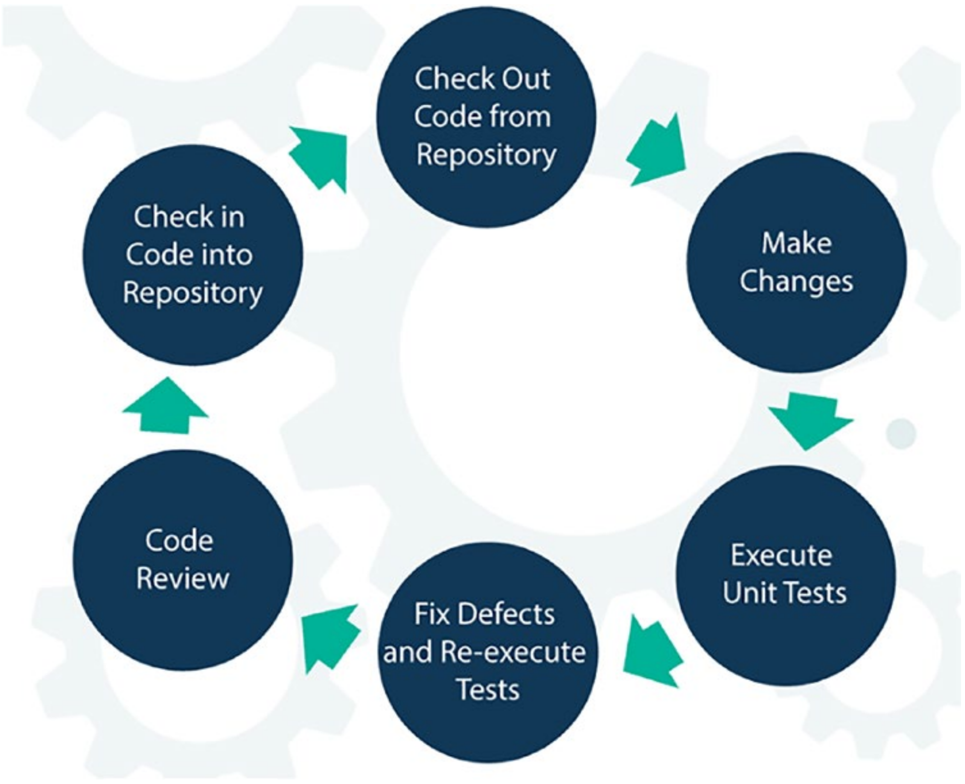

Check Out Code from Repository
Make Changes
Execute Unit Tests
Fix Defects and Re-execute Tests
Code Review
Check in Code into Repository

Figure 6-20. *Unit testing*

Integration Testing

Integration testing is a type of software testing in which different units, modules, or components of the software applications are tested as a combined entity. See Figure 6-21.

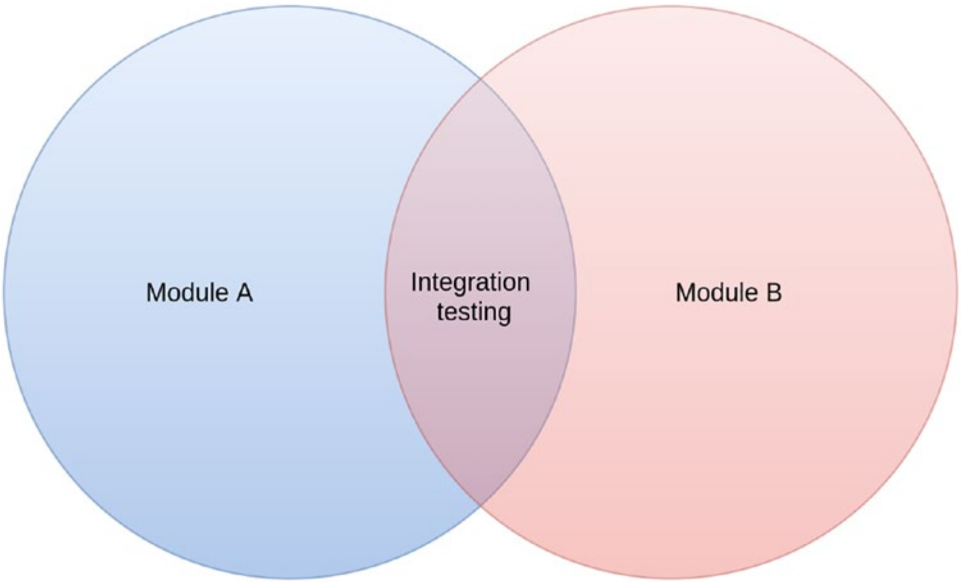


Figure 6-21. *Integration testing*

System Testing

System testing, also referred to as system-level testing, is where the quality assurance team evaluates how various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs its tasks as designed. It mainly focuses on the functionality of the application.

System testing examines every component of the application to ensure that it works as an application as a whole. The QA team typically conducts the system testing after the individual modules with functional or user-story testing is performed.

There are various tools available in the market by which you can perform the system testing. These tools can create, manage, and automate test cases and it also has additional features other than testing.

- Implementation: In the implementation phase, you perform thorough penetration testing and vulnerability scanning to execute and detect any vulnerabilities.
- Support: Once the vulnerabilities are identified and detected, the next step is to mitigate them to make sure the system is free from vulnerabilities. See Figure 6-22.

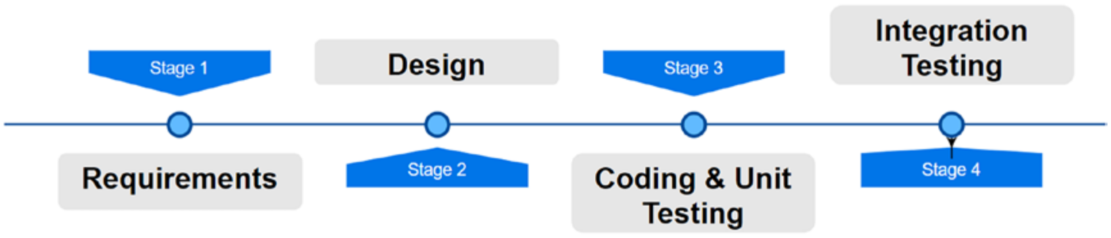


Figure 6-22. *System testing phases*

Now that you understand the security testing process, let's take a quick tour of the key management process.

Key Management

Confidential secrets and keys are an important part of any security system. Using keys, you can perform encryption and decryption of the user authentication. See Figure 6-23.

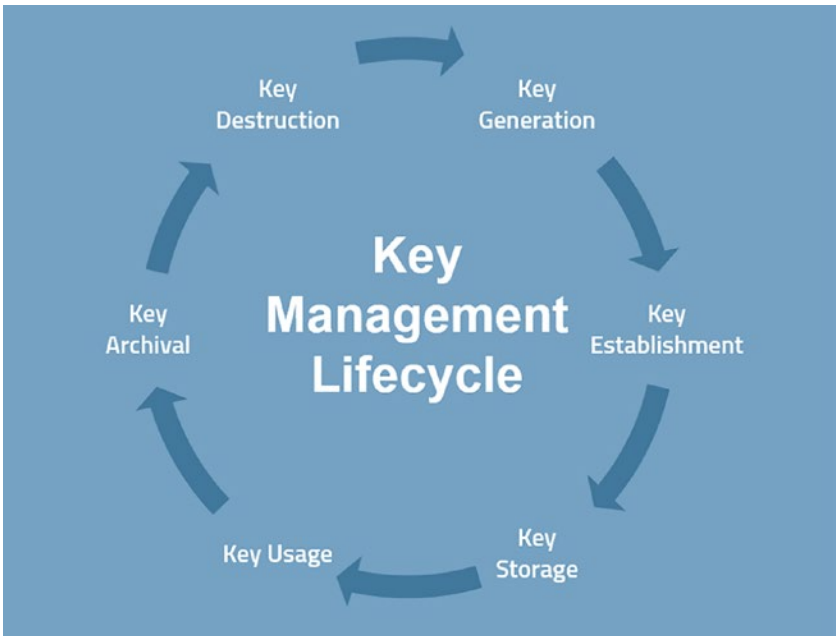


Figure 6-23. *Key management lifecycle*

Proper management of the keys and their related components can ensure the safety of confidential information. Key management is the process of putting standards in place to ensure the security of confidential information. With key management, you can create, exchange, store, delete, and refresh the keys.

Key management is one of the foundational building blocks of all data security. Data is encrypted and decrypted using encryption keys. If the keys are lost, there will be an impact on accessing the data. Keys also ensure the safe transmission of the data across an Internet connection. See Figure 6-24.

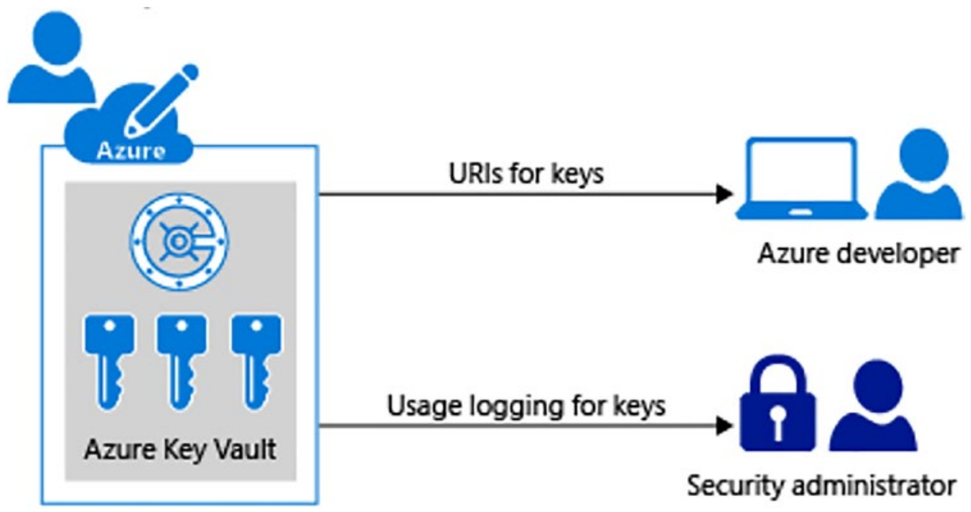


Figure 6-24. *Azure Key Vault: Secret management*

There are two types of keys: symmetric and asymmetric. Symmetric keys deal with the data at rest, which is stored in a static location such as a database. Symmetric encryption uses the same key for encryption and decryption. Encryption using an asymmetric key is more complicated than symmetric encryption. Instead of using the same key, it uses a public key and private key to encrypt and decrypt the data. This public key can be shared with anyone since it encrypts the data but you can't decrypt the data. See Figure 6-25.

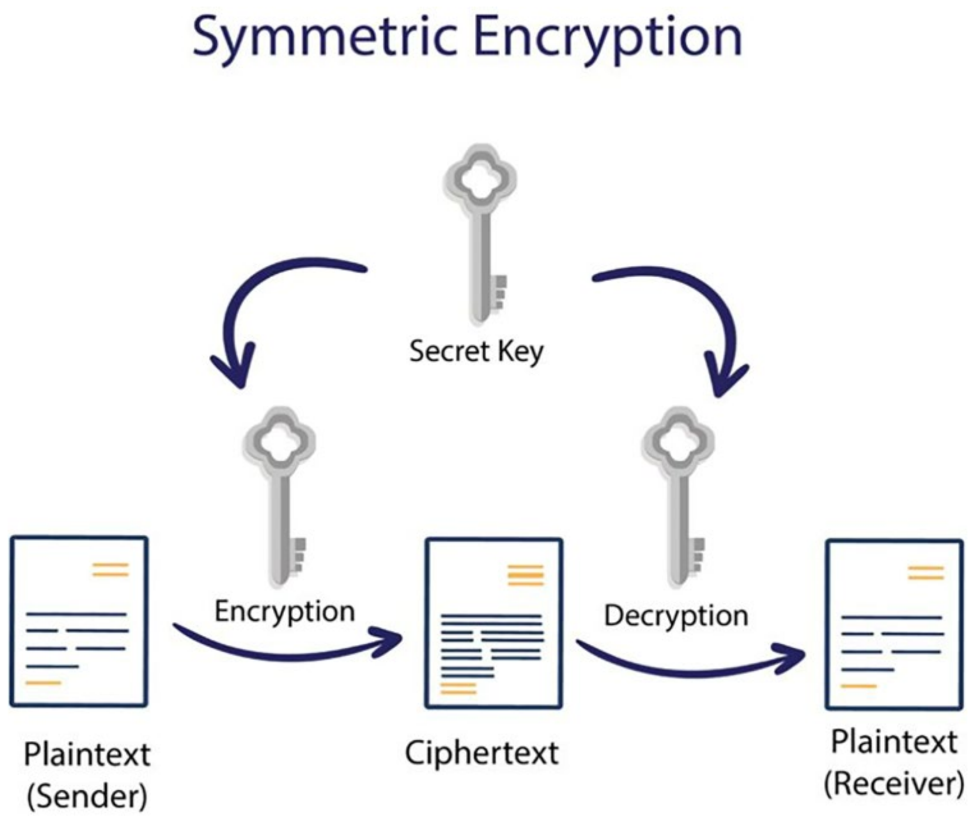


Figure 6-25. *Symmetric encryption*

Asymmetric encryption focuses on encrypting data in motion. Data in motion means the data that is sent across a network connection, which can be a private or public connection. For most data-transmission activities, asymmetric keys are used to protect sensitive data.

Key management follows a lifecycle process for which you need to ensure that the keys are created, stored, used, and rotated securely. Most cryptographic keys follow a lifecycle that involves the following:

- Generation
- Distribution
- Use
- Storage
- Rotation

- Backup
- Destruction

Irrespective of the key management process that organizations follow, a major challenge is to make sure keys are stored securely and are not being misused by unauthorized people. The following are recommended practices to ensure compliance with government regulation and standards.

- Avoid hard-coded keys: It's best not to hard-code values into the source code. Anyone with access to the code can access that confidential information.
- Hardware Security Module (HSM): HSMs are dedicated processors typically designed for the protection of the crypto key lifecycle. They act as a security module that creates trust anchors that protect the cryptographic infrastructure of the organization. HSM is a secure, cryptographic processor designed specifically to protect the lifecycle of the confidential keys. It provides a high level of security in terms of confidentiality, integrity, and availability. See Figure 6-26.

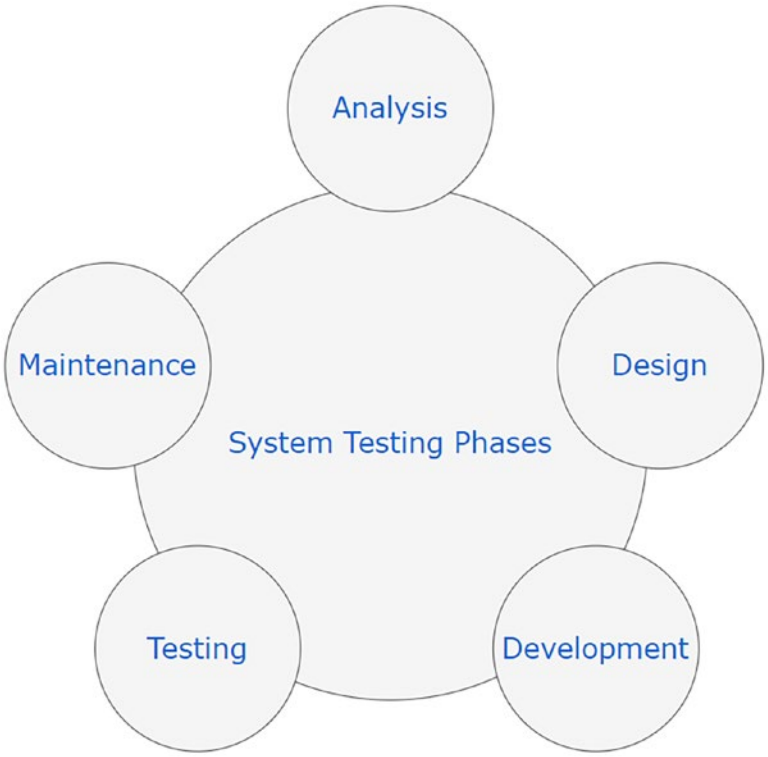


Figure 6-26. *Hardware Security Module (HSM)*

Next, we explore vulnerability management and discuss how you can efficiently detect and handle it.

Vulnerability Management

Patching your application and system with daily, weekly, and monthly updates is a humongous task. Moreover, in the period between patches, depending on the severity of the pending vulnerability, a bug can become a weakness in the application or system.

Keeping virtual machines and devices up-to-date is mandatory to create secure software applications. If the virtual machines or devices are outdated, that leaves the software applications exposed to cyberattacks. Thus, it is very important that vulnerabilities be patched as soon as possible. It is also important to manage patches and vulnerability detection so that severe problems are detected quickly.

A majority of the malware issues are sent via emails. This malicious software can come in multiple forms: worms, viruses, trojans, and ransomware. Worms are a type of malware that spreads from computer to network without user intervention. Computer viruses are malware that affects digital media such as computer memory.

A trojan horse is downloaded on to a system when the user downloads a program or opens an email containing a malicious attachment. Attackers usually use these to steal privileged information, like passwords, from the system. Ransomware is a kind of malicious software that is created to block access to files or directories on an infected computer, after which the attacker demands that the victim pay money to get their files back.

Vulnerability management is a process of identifying, evaluating, and reporting security vulnerabilities in the system and software that runs on them. It refers to the weakness that allows an attacker to get access to the product and the information it holds. It is an ongoing process of identifying, assessing, reporting, and managing security vulnerabilities across workloads, systems, and endpoints.

Security vulnerabilities refer to technological weakness that allow attackers to compromise a product or application. The vulnerability management process can be broken into these four steps:

- Identifying vulnerabilities
- Evaluating vulnerabilities
- Treating vulnerabilities
- Reporting vulnerabilities

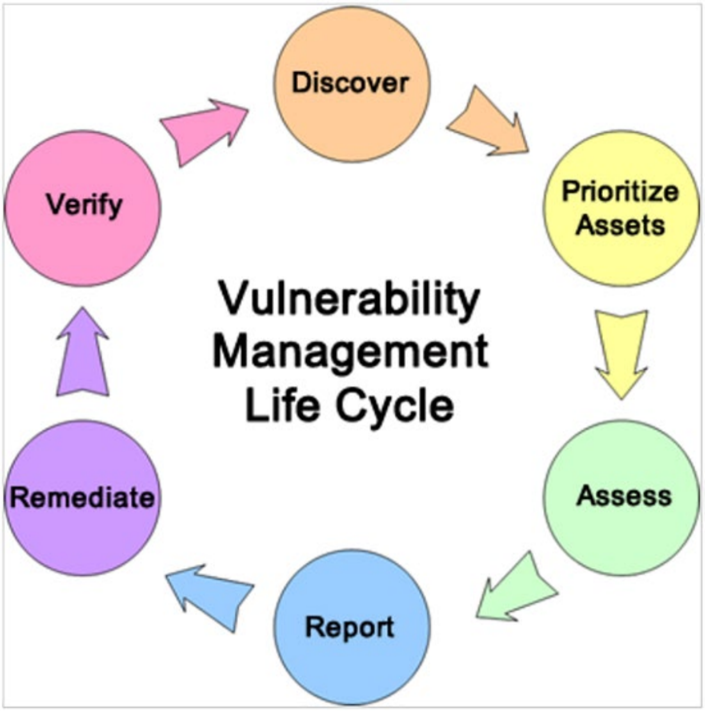


Figure 6-27. *Vulnerability management*

The steps of the vulnerability lifecycle are as follows:

- Discover: The first step is to identify the inventory of all assets across the network and identify host details, including the operating system and open services to identify vulnerabilities. Develop a network baseline and identify security vulnerabilities on a regular automated schedule.
- Prioritize assets: Categorize assets into group or business units and assign the business value to asset groups based on the priority and criticality of the business operation.
- Assess: Determine the baseline risk profile so you can eliminate risk based on criticality, vulnerability, and asset classification. This will help reduce the vulnerability of the developed application.

- Report: Once the vulnerabilities are assessed, measure the risk based on the security policies. Create a security plan, monitor the activities, and describe the known vulnerabilities.
- Remediate: Once the report is ready, prioritize and fix the vulnerabilities according to the business risk. Once that is done, establish the control and show the progress.
- Verification: The last step is to verify that the threats have been eliminated through the audits. See Figure 6-28.

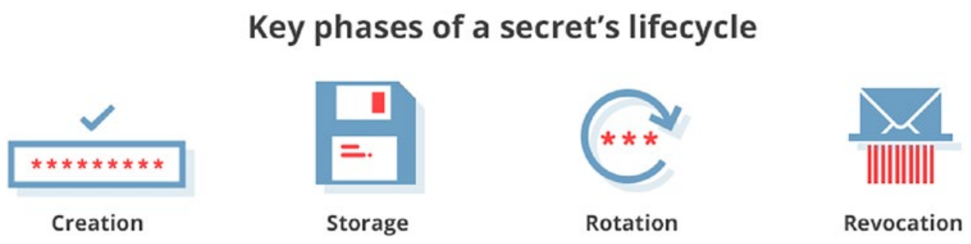


Figure 6-28. *Vulnerability report*

Vulnerability can be defined as a weakness of an asset or group of assets that can be exploited by one or more threats. Any means or act by which an external actor gets unauthorized access to the data or privilege to control an application is considered a vulnerability. Common examples include communication network ports that are open to the Internet and insecure configurations of software and OSs. At a higher level, vulnerabilities can be broken into a few components:

- CVE - Common Vulnerabilities and Exposure: Each CVE defines a specific vulnerability that attackers can attack on the application or system.
- CCE - Common Configuration Enumeration: This is a list of system security configuration issues that can be used to develop configuration guidance.
- CPE - Common Platform Enumeration: This is a standardized method of describing and identifying classes of the application, operating systems, and devices in the environment.
- CVSS - Common Vulnerability Scoring System: This scoring system assigns the scores to each defined vulnerability and is used to prioritize efforts and resources according to the threat.

Disaster Recovery (DR)

Disaster recovery is the process of restoring application functionality during an abnormal situation. Upfront failures can happen at any point in time when you use public clouds like Azure, AWS, or GCP. When such failures occur, you should be able to minimize the effect of failing components. See Figure 6-29.

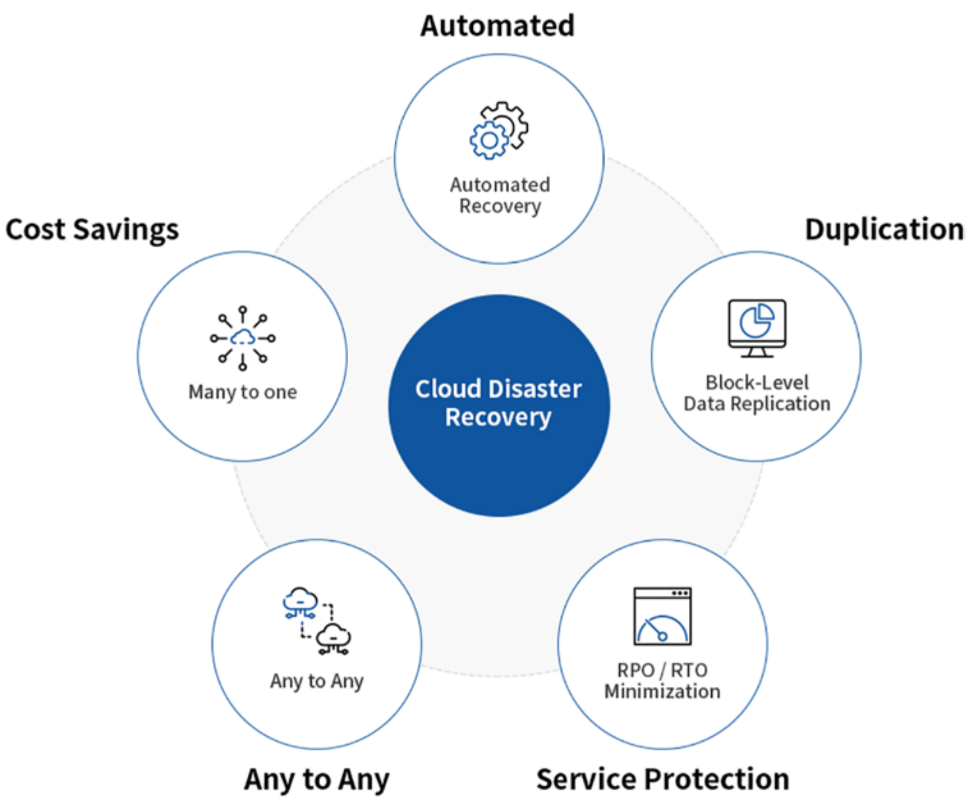


Figure 6-29. *Disaster recovery management*

With automated testing, you can prepare and minimize failures. A standard backup and recovery process is a must in order to handle such failures without an impact. You need to consider the following key points while creating a disaster and recovery plan:

- Create a disaster and recovery plan considering all failure scenarios
- Design a disaster recovery plan to run most applications with reduced functionality
- Design a backup strategy tailored to your business requirements
- Automate processes and runbooks to do the rollback or failover activities

After understanding the basics of backup and disaster recovery, let's look at the disaster recovery process of design and implementation.

- Determine subscription and service requirements: The process of determining the subscription and service requirements consists of various key processes. Certain resources, such as resource groups and storage accounts, are limited in every Azure subscription.
- Azure regional pairs: Multiple regional pairs are accessible in Azure. For example, North China, East China, North Europe, and West Europe, where you can see two regions per continent. Some Azure services can take advantage of cross-region replication to ensure business continuity and protect against data loss. Azure provides several storage solutions that use cross-region replication to ensure data availability. For example, Azure Geo Redundant storage automatically replicates data to a secondary region. See Figure 6-30.

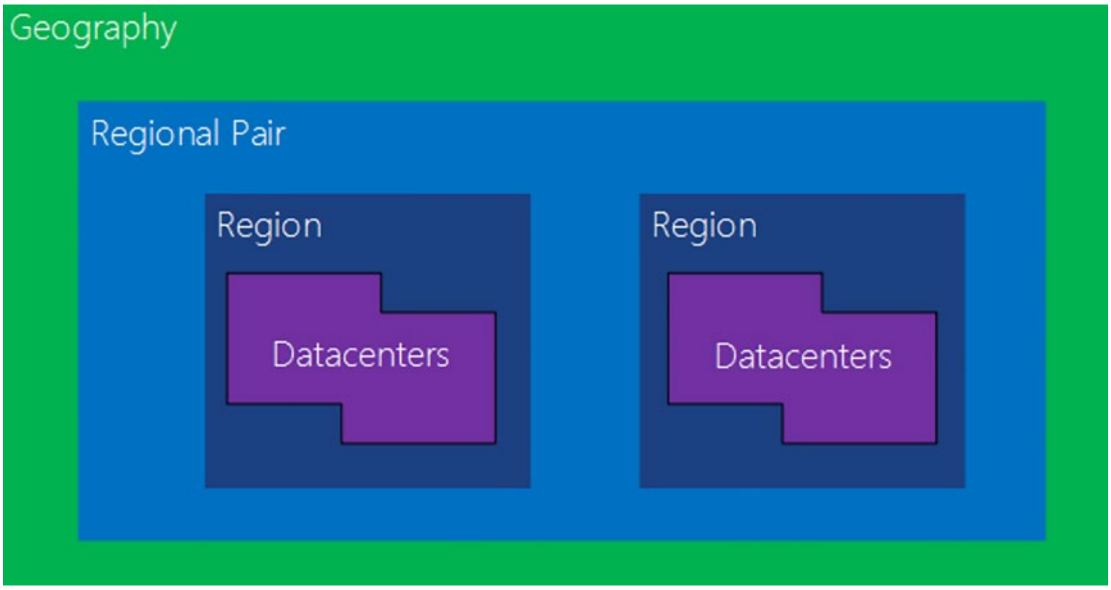


Figure 6-30. *Azure regional pairs*

- Azure availability zones: Availability zones are dedicated physical locations in an Azure region. In order to ensure resiliency, three separate zones are enabled in every region. See Figure 6-31.

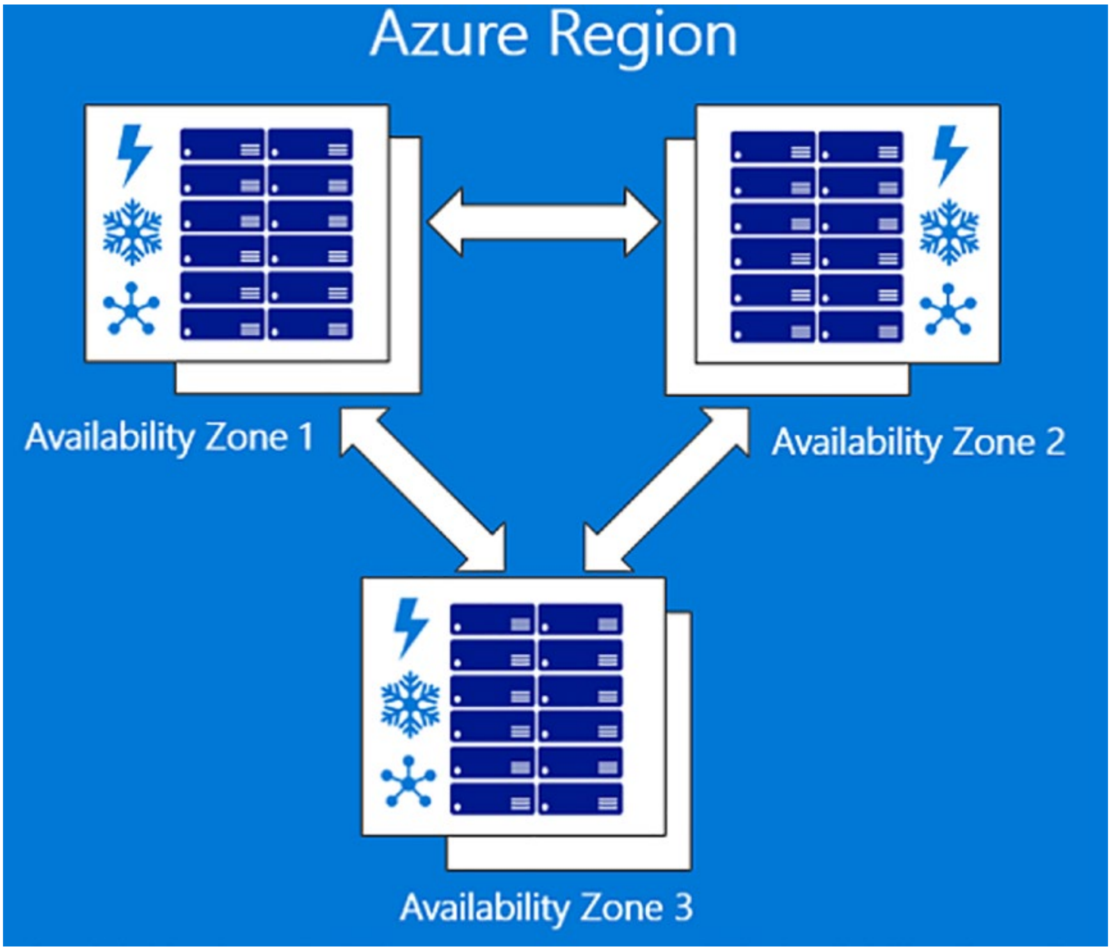


Figure 6-31. *Azure availability zones*

- Azure PaaS components: Azure provides many built-in PaaS services. Each service can be configured to enable backup and georeplication in case there are issues. See Figure 6-32.

Figure 6-32. *Azure PaaS services*

- Determine and document RTO, RPO, and RLO: Recovery Time objective (RTO) is the amount of time and service level within which business processes must be recovered when a disaster happens. See Figure 6-33.

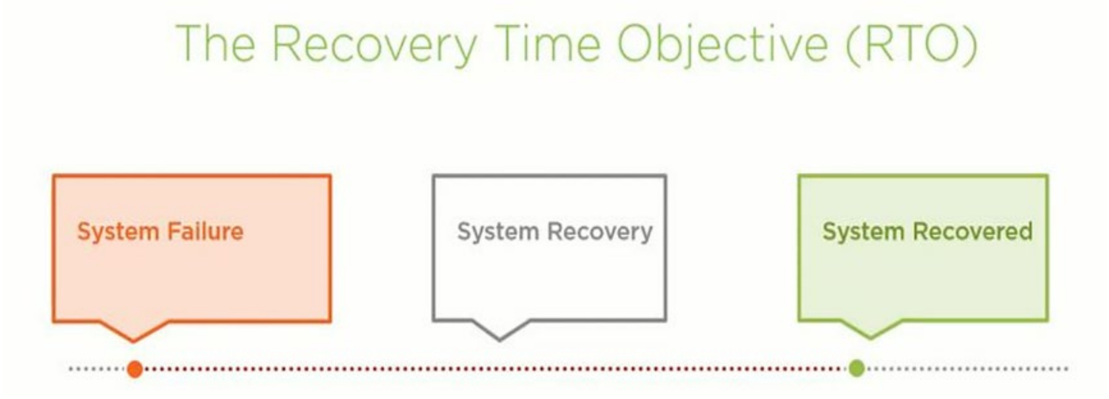


Figure 6-33. *Recovery Time Objective*

Recovery Point Objective (RPO) refers to the amount of time that can be lost before creating damage to the organization. Recovery Level Objective (RLO) specifies the granularity by which data must be recovered.

Conclusion

This chapter explored the threat modeling process and the need to enable it in your organization. You also learned about how to secure your infrastructure and platform deployment with the Infrastructure as a Code (IaC). Then you learned the importance of the security testing process and its related phases. Then you learned about the importance of the key management process and standard process to automate it.

The chapter also discussed vulnerability management and the importance of the disaster recovery process and its best practices.