8.1 Application environments, provisioning, and version control

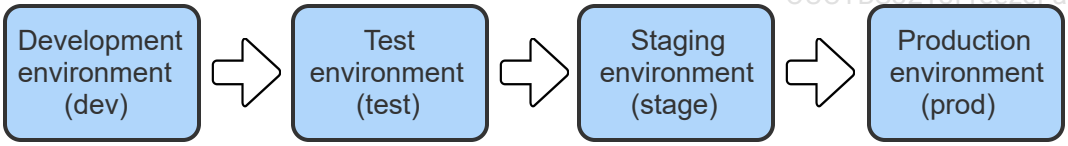
Environment

An ***environment*** is the collection of hardware and software used to build an application. An environment may include hardware resources, such as servers, desktops, and mobile devices, software tools such as an integrated development environment (IDE), and debugging and performance evaluation utilities. Secure application development requires four environments:

- A ***development environment***, or ***dev***, is the environment where an application is created, debugged, modified, and improved.
- A ***test environment***, or ***test***, is the environment where an application is tested against the application's specification requirements.
- A ***staging environment***, or ***stage***, is a pre-production environment that mirrors a production environment and contains the final release version of an application. A staging environment is used for testing the installation, configuration, and migration scripts and procedures.
- A ***production environment***, also known as ***prod*** or ***live environment***, is an environment where the release version of an application is deployed and made available to the application's users.

Quality assurance, or ***QA***, is the process of testing an application's various aspects, including an application's stability, usability, security, functionality, and performance to ensure that the application meets user requirements.

PARTICIPATION ACTIVITY 8.1.1: Development environment.



Animation content:

Static image: A flowchart. Box 1 shows "Development environment (dev)". Box 2 shows "Test environment (test)". Box 3 shows "Staging environment (stage)". Box 4 shows "Production environment (prod)".

Animation captions:

1. A mobile banking application is created, debugged, modified and improved in a development environment (dev).
2. After development of the mobile banking application is complete, the application is moved to a test environment (test) where the application is tested.
3. The final release version of the mobile banking application is moved to a staging environment (stage). A stage mirrors a production environment.
4. The release version of the mobile banking application is moved to a production environment (prod) where the application can be used by the bank's customers.

PARTICIPATION ACTIVITY 8.1.2: Environment.

Select the environment in each scenario.

1) The pre-production environment which contains the final release version of an application that cannot be accessed by the application's users.
   - Development
   - Test
   - Stage
   - Production

2) The environment where an application is used by the application users.
   - Development
   - Test
   - Stage
   - Production

3) The environment where an application is debugged.

- Development
- Test
- Stage
- Production

4) The environment where an application is tested against the application's specification requirements.

- Development
- Test
- Stage
- Production

Provisioning, code signing, and version control

Provisioning is the process of moving an application to a production environment and customizing the application configurations. Application configuration may include creating users, setting permissions, and modifying application appearance. Provisioning may include code signing. ***Code signing*** is the act of digitally signing software by the software's publisher. Code signing confirms the identity of the software publisher and guarantees that the code was not modified or corrupted since the code was signed. ***Deprovisioning*** is the process of removing an application from a production environment. Deprovisioning may also refer to the act of removing user access to applications, systems, and data within a network. Ex: An employee's account is deprovisioned when the employee leaves an organization.

A new software release includes updates which may add functionality, fix bugs, and improve performance. ***Software versioning*** is the process of assigning names or numbers to unique states of released software. Ex: Microsoft Windows 10 operating system versions are labeled YYH1 or YYH2 (YY represents the year, and H1 or H2 the half-year). Windows 10 version 20H1 was released in the first-half of the year 2020. At any given time, different versions of software could be in use. ***Version control***, also known as ***source control***, is the practice of tracking and managing a software's versions.

Example 8.1.1: The Windows 10 Pro version 21H2 installed on a laptop.

Edition	Windows 10 Pro
Version	21H2
Installed on	6/5/2021
OS build	19044.1469
Experience	Windows Feature Experience Pack 120.2212.3920.0

PARTICIPATION ACTIVITY 8.1.3: Provisioning, code signing, and version control.

How to use this tool

Code signing **Software versioning** **Deprovisioning** **Provisioning** **Version control**

	The process of moving an application to a production environment and customizing the application configurations.
	The process of removing an application from a production environment.
	The process of assigning unique names or numbers to unique states of software.
	The practice of tracking and managing a software's versions.
	The act of digitally signing software by the software's publisher.

Reset

PARTICIPATION ACTIVITY 8.1.4: Provisioning, code signing, and version control.

1) Application provisioning is the process of moving an application to what type of environment?

- Test
- Development
- Production

2) Which security properties are guaranteed by code signing?

- Integrity and confidentiality
- Confidentiality and non-repudiation
- Integrity and data origin authentication

3) What is the purpose of software versioning?

- To configure the different released versions of software
- To distinguish between the different released versions of software
- To deprovision the different released versions of software

CHALLENGE ACTIVITY 8.1.1: Application environments.

581480.4344582.qx3zqy7

Start

Select the application environment described in each statement.

Pick — The environment where a web browser application is debugged.

Pick — The environment where a spreadsheet application is tested against the a specification requirements.

Pick — The environment where an application is made available to the applicatic

Pick The environment where a gaming application's migration scripts are test

1 2

Check Next

8.2 Secure coding practices

Secure coding techniques

Secure coding, also known as ***secure programming***, is the practice of developing software in a way that minimizes the risk of creating software vulnerabilities. Ex: Allocating sufficient memory for storing user input prevents buffer overflows. The goal of secure coding is to create software that functions as intended even when the software is subjected to malicious attacks. ***Secure coding techniques*** are methods designed to improve code security. Secure coding techniques include obfuscating code, reusing code, removing dead code, and using third-party software libraries and software development kits (SDKs).

Obfuscation/camouflage

Obfuscation, or ***camouflage***, is a secure coding technique which makes code difficult to read and harder to understand by modifying the code appearance. Obfuscation does not change a program's functionality, but helps hide the program's logic and purpose. Obfuscation can be used to protect intellectual property and to prevent a program from being reverse engineered. Ex: Renaming a program's functions, classes, and methods to use less descriptive names; removing debug information and comments; and adding unused or meaningless code are obfuscation techniques.

PARTICIPATION ACTIVITY 8.2.1: Rename obfuscation.

```
private void CalculateEmployeeTax(sList employeeGroup) {
   while(employeeGroup.HasMember()) {
      employee = employeeGroup.GetNext (true);
      employee.UpdateTaxRate();
   }
}
```

Original code

```
private void a(b c) {
   while (c.d()) {
      f = c.e(true);
      f.g();
   }
}
```

Obfuscated code

Animation content:

Static image: A box labeled "Original code" containing Java code.
Start Java code.
private void CalculateEmployeeTax(sList employeeGroup) {
   while (employeeGroup.HasMember()) {
      employee = employeeGroup.getNext(true);
      employee.UpdateTaxRate();
   }
}
End Java code.
The line "employee.UpdateTaxRate();" is highlighted in blue. A box titled "Obfuscated code" containing Java code.
Start Java code.
private void a(b c) {
   while (c.d()) {
      f = c.e(true);
      f.g();
   }
}
End Java code.

Animation captions:

1. The function CalculateEmployeeTax is renamed to a, type sList to b, and parameter employeeGroup to c.
2. The parameter employeeGroup is renamed to c, and the method HasMember to d.
3. The parameter employee is renamed to f, employeeGroup to c, and method GetNext to e.

4. The parameter employee is renamed to f, and method UpdateTaxRate to g. The obfuscated code is difficult to read and harder to understand compared to the original code.

PARTICIPATION ACTIVITY 8.2.2: Secure coding techniques.

How to use this tool

Obfuscation **Secure coding techniques** **Secure coding**

	A secure coding technique which makes code difficult to read and harder to understand by modifying the code appearance.
	The practice of developing software in a manner that minimizes the risk of creating software vulnerabilities.
	Methods designed to improve code security.

Reset

PARTICIPATION ACTIVITY 8.2.3: Secure coding techniques.

1) What is the goal of secure coding?

- Create software that functions according to the software's specifications
- Create software that functions as intended even when the software is subjected to malicious attacks
- Create software that uses less memory and executes faster

2) How do obfuscation techniques improve code security?

- By adding code that does not slow down code execution time
- By making code more difficult to read and harder understand
- By removing code that is never executed

3) In the above animation, which method is replaced with d?
   - UpdateTaxRate
   - GetNext
   - HasMember

Code reuse and dead code

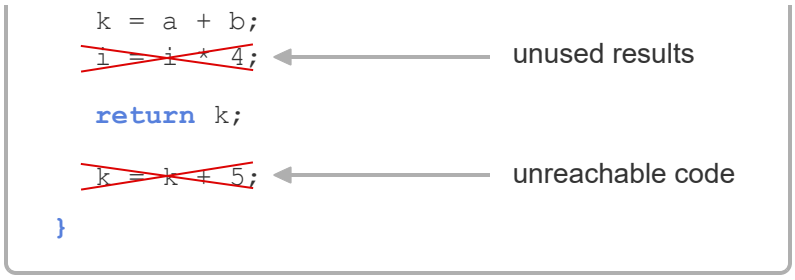
Code reuse, also known as ***software reuse***, is the use of existing software to build new software. The aim of code reuse is to reduce redundancy and development time and save resources by utilizing components that have already been created within a software development process. APIs provide a mechanism to enable code reuse. Ex: A software library is a collection of precompiled functions, routines, and other resources that can be used in the development of a new program.

Dead code is code that can never be executed at run-time, or is executed but whose result is never used in any other computation. Ex: The code in a method that is never called or code that is not executed because of a branch is dead code. The execution of dead code wastes program memory and computation time. Dead code can be detected by static code analysis and data flow analysis. Dead code can be removed by an optimizing compiler.

PARTICIPATION ACTIVITY 8.2.4: Dead code.


```
int function(int a, int b)
{
   int i, j, k;
   i = 3;
   j = 4;
```

unused storage



Animation content:

Static image: Start C code.
int function(int a, int b)
{
   int i, j, k;
   i = 3;
   j = 4;

   k = a + b;
   i = i * 4;
   return k;
   k = k + 5;
}
End C code.
The line "j = 4;" is covered by a red X and has the label "unused storage". The line "i = i * 4;" is covered by a red X and has the label "unused results". The line "k = k + 5;" is covered by a red X and has the label "unreachable code".

Animation captions:

1. Storage is allocated to variable j, but the variable is not used in the function.
2. The result of multiplying variable i by 4 is not used in the function.
3. Any code after return is unreachable and is not executed.
4. Removing dead code frees up memory and speeds up a program's execution time.

PARTICIPATION ACTIVITY 8.2.5: Code reuse and dead code.

1) Code reuse increases software development time.

- True
- False

2) Dead code only refers to code that can never be executed.

- True
- False

3) In the animation above, variable i is not used in the function.

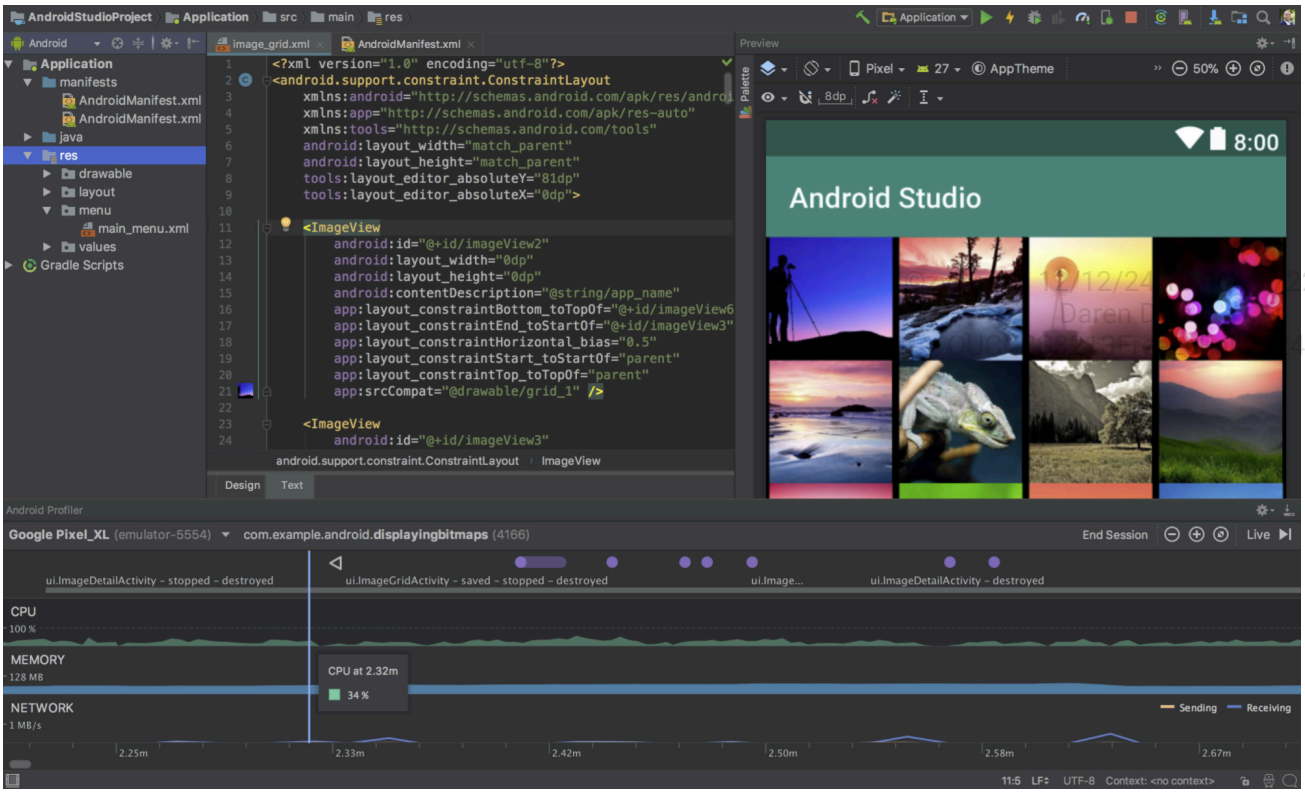
- True
- False

Third-party libraries and SDKs

A ***third-party library*** is a reusable software component developed by an entity other than the original publisher of a software development platform. Third-party software libraries save development time and cost by promoting component-oriented software development. Third-party software libraries enable software developers to integrate pre-tested and reusable code with new software.

A ***software development kit (SDK)*** is a collection of software development tools in one package. An SDK is specific to a hardware platform and operating system combination. Ex: Android SDK includes a debugger, a handset emulator, libraries, an integrated development environment (IDE), documentation, sample code, and tutorials.

Example 8.2.1: Android Studio IDE included in the Android SDK.



Credit: Google LLC[1]

PARTICIPATION ACTIVITY 8.2.6: Third-party libraries and SDKs.

1) What entity develops a third-party software library?

- A software reseller
- The original software publisher
- An entity other than the original software publisher

2) How does a third-party software library save development time?

- By enabling software developers to use existing code without developing new code
- By enabling software developers to modify the code in a third-party software library
- By enabling software developers to use pre-tested and reusable

code

3) A software development kit (SDK) is specific to a(n) _____.

- operating system
- hardware platform
- hardware platform and an operating system

Sandboxing

Sandboxing *is a security practice involving the creation of a separate, isolated environment, referred to as a* ***sandbox****, to test and execute untrusted or unknown code, applications, or files without harming the main system or network. Designed to emulate a production environment, a sandbox imposes additional restrictions and integrates monitoring capabilities.*

A sandbox facilitates the analysis and execution of potentially malicious code or applications in a controlled and secure manner, helping to identify and understand potential threats. A sandbox is typically disposable, meaning that the environment can be easily created, utilized, and discarded as needed, thereby reducing the likelihood of persistent threats.

(*1) Google LLC "Android Studio". https://developer.android.com/studio.

8.3 Elasticity, scalability, and software diversity

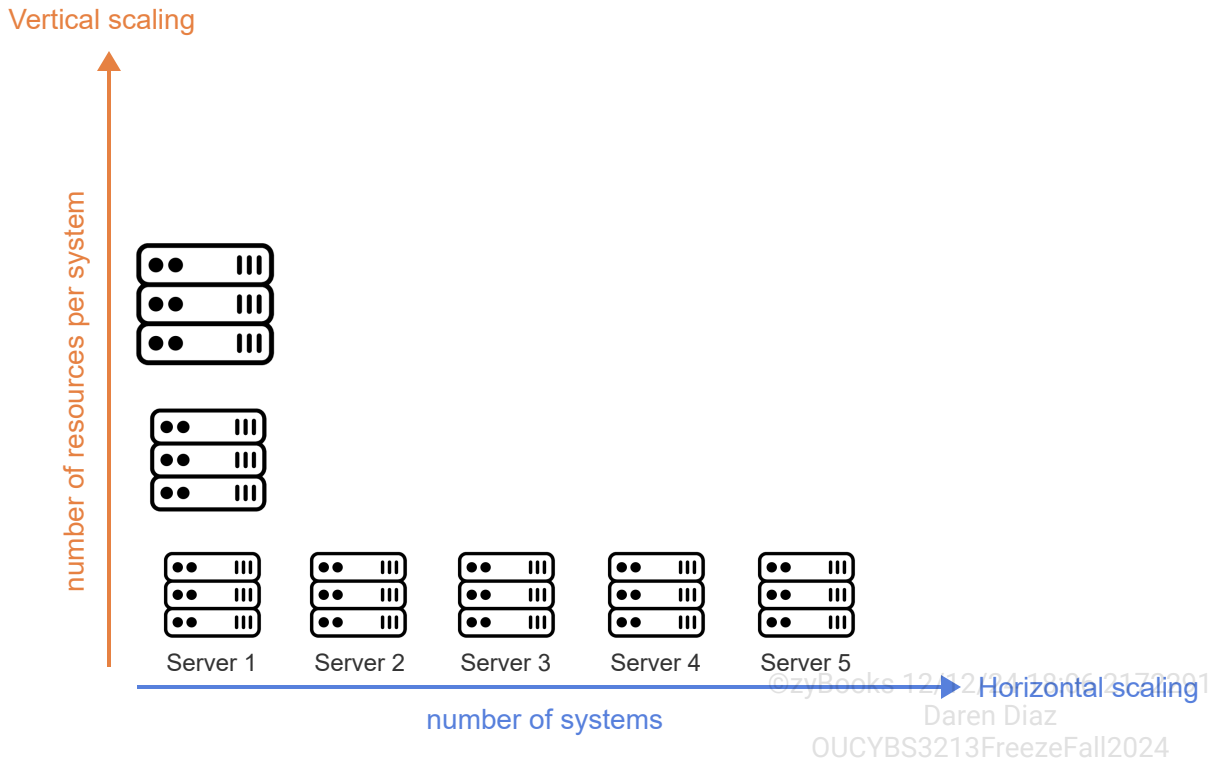
Elasticity and scalability

Elasticity is the degree to which a system is able to adapt to workload changes by automatically provisioning and deprovisioning resources. The goal of elasticity is to match a system's available resources to the system's demands at any given time. Elasticity helps with minimizing resource costs and is commonly used in pay-per-use cloud services. Ex: As the number of visitors to a retailer's web server surges during a sales event, additional web servers are dynamically provisioned to meet the higher demand. Web servers are dynamically deprovisioned as the number of visitors decreases.

Scalability is the measure of a system's ability to handle increased demands. Ex: As a start up company grows, the company provisions additional servers to handle the computing needs of the company's new employees. Two scalability types exist:

- ***Horizontal scalability*** is scalability which entails adding new systems to an existing infrastructure. Ex: Increasing the number of web servers hosting a web site. Horizontal scalability increases the complexity of administrative tasks such as monitoring, patch management, and backups.
- ***Vertical scalability*** is scalability which entails adding resources to existing systems. Ex: Increasing a web server's memory, storage, or CPUs. Vertical scalability is easier to implement than horizontal scalability because software-level configurations for the added resources are often not required.

PARTICIPATION ACTIVITY 8.3.1: Horizontal and vertical scalability.



Animation content:

Static image: An orange arrow labeled "number of resources per system" points from the bottom of the animation upward. The text "Vertical scaling" is above the orange arrow. A blue arrow

labeled "number of systems" starts in the same position as the orange arrow and points to the right. Five servers named "Server 1" through "Server 5" are in a row just above the blue arrow showing horizontal scaling. Two additional servers are above Server 1 next to the orange arrow showing vertical scaling.

Animation captions:

1. In horizontal scaling, new systems are added to an existing infrastructure to handle increased demands.
2. In vertical scaling, new resources such as memory and storage are added to a system to handle increased demands.

PARTICIPATION ACTIVITY 8.3.2: Elasticity and scalability.

How to use this tool

Scalability **Vertical scalability** **Elasticity** **Horizontal scalability**

	The degree to which a system is able to adapt to workload changes by automatically provisioning and deprovisioning resources.
	Scalability which entails adding new systems to an existing infrastructure.
	Scalability which entails adding of resources to an existing system.
	The measure of a system's ability to handle increased demands.

Reset

PARTICIPATION ACTIVITY 8.3.3: Elasticity and scalability.

1) How does elasticity minimize resource costs?

- By enabling resource providers to charge a fixed amount for reserved resources
- By enabling users to manually add resources when resources are needed
- By enabling users to pay for resources only when resources are used

2) What type of scaling is used when a company increases the storage space on a database server?

- Horizontal scaling
- Vertical scaling

3) What type of scaling is used when a new application server hosting a mobile app is provisioned to handle increased demand for the mobile app?

- Horizontal scaling
- Vertical scaling

Software diversity

A software vulnerability can be exploited in all the software's running instances. Ex: A vulnerability in Microsoft Word which allows a user to gain administrative access to system resources can be exploited on all the computers that run Microsoft Word. If every instance of a software is different, a vulnerability in the software cannot be widely exploited. ***Software diversity*** is the practice of creating diversity within the software development process with the aim of reducing the extent of software vulnerability exploits.

Software diversity can be created by a compiler that generates different binary executable files for a program by randomizing the program's implementation aspects. A program's behavior can be changed by modifying the program's functions and processes without changing the program's output. Ex: A compiler with diversification abilities can randomize the allocation of a program's variables into registers or the order in which the program's functions are laid out in memory. The randomness created at the executable code level ensures that each compilation of a program generates a unique binary executable file. Software diversity helps reduce the extent of a software vulnerability exploit because the software's different executable binaries may not contain the same vulnerability.

PARTICIPATION ACTIVITY 8.3.4: Software diversity.

1) A compiler which has diversification abilities changes a program's functionality to create software diversity.
   - True
   - False

2) Software diversity can be achieved by creating identical binary executable files for a program.
   - True
   - False

3) The goal of software diversity is to reduce the possibility of a software vulnerability from being exploited on all computers that run the software.
   - True
   - False

8.4 Database security

Database security

An application may use databases for various reasons, such as storing data, processing transactions, and providing content. A database should be secured to prevent data exposure. ***Data exposure*** is the intentional or unintentional disclosure of information to an unauthorized user or application. Database security can be improved by implementing different measures, including normalization, stored procedures, encryption, salting, and hashing.

Normalization

Database normalization is the process of organizing data in a relational database. Normalization involves creating tables and establishing relationships between those tables in a manner that removes data redundancies and improves data integrity. Ex: Normalization may remove a customer address

field from all but one table to reduce data redundancy. Normalization improves database security by preventing data inconsistencies and update anomalies.

PARTICIPATION ACTIVITY 8.4.1: Database normalization.

Product table

ProductID	Color	Price
1	blue	8.75
2	green, red	2.35
3	black	6.21
4	yellow	7.28
5	white, grey	9.14

Product color table

ProductID	Color
1	blue
2	green
2	red
3	black
4	yellow
5	white
5	grey

Product price table

ProductID	Price
1	8.75
2	2.35
3	6.21
4	7.28
5	9.14

Animation content:

Static image: A table titled "Product table" with columns "ProductID", "Color", and "Price". Row 1, ProductID: 1. Row 1, Color: blue. Row 1, Price: 8.72. Row 2, ProductID: 2. Row 2, Color: green, red. Row 2, Price: 2.35. Row 3, ProductID: 3. Row 3, Color: black. Row 3, Price: 6.21. Row 4, ProductID: 4. Row4, Color: yellow. Row 4, Price: 7.28. Row 5, ProductID: 5. Row 5, Color: white, grey. Row 5, Price: 9.14. Rows 2 and 5 are highlighted blue. An arrow points from the Product table to a table titled "Product color table" with columns "ProductID" and "Color". Row 1, ProductID: 1. Row 1, Color: blue. Row 2, ProductID: 2. Row 2, Color: green. Row 3, ProductID: 2. Row 3, Color: red. Row 4, ProductID: 3. Row 4, Color: black. Row 5, ProductID: 4. Row 5, Color: yellow. Row 6, ProductID: 5. Row 6, Color: white. Row 7, ProductID: 5. Row 7, Color: grey. The Product table has a second arrow pointing to a table titled "Product price table" with columns "ProductID" and "Price". Row 1, ProductID: 1. Row 1,

Price: 8.75. Row 2, ProductID: 2. Row 2, Price: 2.35. Row 3, ProductID: 3. Row 3, Price: 6.21. Row 4, ProductID: 4. Row 4, Price: 7.28. Row 5, ProductID: 5. Row 5, Price: 9/14.

Animation captions:

1. An unnormalized table which stores more than one value in the Color field for two rows. To normalize the table, the Product table can be split into two tables.
2. The Product color table stores ProductID and Color.
3. The Product price table stores ProductID and price. Both tables are normalized because the Color and Price fields store only one value in each row.

PARTICIPATION ACTIVITY 8.4.2: Database security.

1) Data exposure only refers to the intentional disclosure of information.
   - True
   - False

2) Data integrity is improved as a result of database normalization.
   - True
   - False

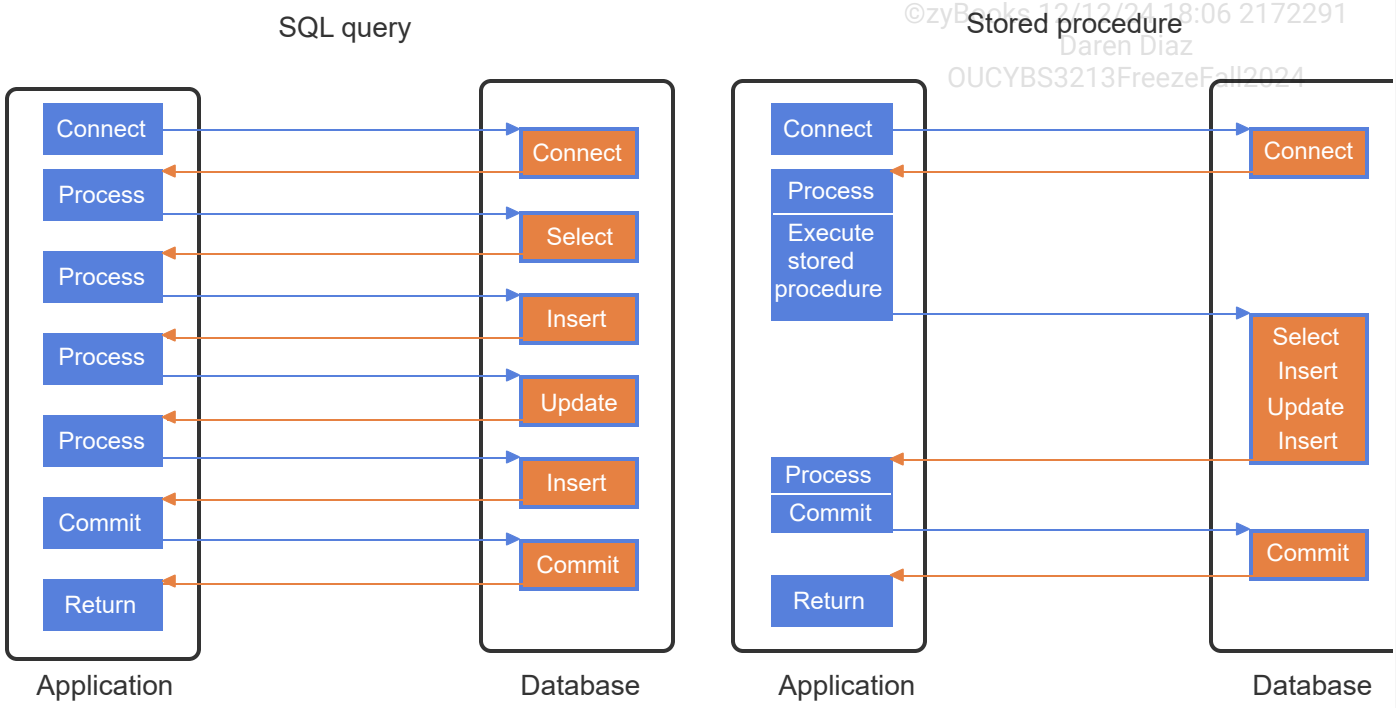
3) Database normalization removes data from a database.
   - True
   - False

Stored procedures

A ***stored procedure*** is a group of SQL statements created to perform a specific task. A stored procedure is compiled and stored in a database and may operate on multiple database tables. A stored procedure improves database security because a user or application is limited to executing database operations implemented in a stored procedure. Access rights are granted to a stored procedure, not to the tables used by the stored procedure.

A stored procedure supports parameterized queries. A ***parameterized query*** is a query in which placeholders are used for parameters and the parameter values are supplied at execution time. Parameterized queries minimize the risk of SQL injection by removing the need to create SQL queries based on user input. Data rules are defined in a database, instead of an application.

PARTICIPATION ACTIVITY 8.4.3: Stored procedure.



Animation content:

Static image: The left side is labeled "SQL query" and has vertical boxes labeled "Application" and "Database". Arrows show communication between the application and the database. An arrow points from "Connect" in the application to "Connect" in the database. An arrow points from "Connect" in the database to "Process" in the application. An arrow points from "Process" in the application to "Select" in the database. An arrow points from "Select" in the database to a second "Process" in the application. An arrow points from the second "Process" in the application to "Insert" in the database. An arrow points from "Insert" in the database to a third "Process" in the application. An arrow points from the third "Process" in the application to "Update" in the database. An arrow points from "Update" in the database to a fourth "Process" in the application. An arrow points from the fourth "Process" in the application to another "Insert" in the database. An arrow points from the second "Insert" in the database to "Commit" in the application. An arrow points from "Commit" in the application to "Commit" in the database. The final arrow points from "Commit" in the database to "Return" in the application. The right side is labeled "Stored procedure" and also has "Application" and "Database" boxes. An arrow points from "Connect" in the application to "Connect" in the database. An arrow points from "Connect" in the database to "Process" in the application. An arrow points from "Execute stored procedure" in the application to "Select, Insert, Update, Insert" in the database. An arrow points from "Select, Insert, Update, Insert" in the database to another "Process" in the

application. An arrow points from "Commit" in the application to "Commit" in the database. The final arrow points from "Commit" in the database to "Return" in the application.

Animation captions:

1. An application sends SQL queries to a database one at a time. After each query execution on a database, the application processes the query results.
2. A stored procedure reduces database interactions and network traffic and latency.

PARTICIPATION ACTIVITY 8.4.4: Stored procedure.

1) Why do stored procedures improve database performance?

- Because stored procedures reduce the number of SQL statements
- Because stored procedures are compiled and stored in a database
- Because stored procedures minimize the risk of SQL injection

2) Why do parameterized queries prevent SQL injection attacks?

- Because the SQL queries in stored procedures do not accept user input
- Because stored procedures do not use SQL
- Because SQL statements are not created based on user input

3) In the animation above, how many messages are exchanged between the application and the database when using a stored procedure?

- 2
- 6

○ 12

PARTICIPATION ACTIVITY | 8.4.5: Database security.

How to use this tool

Data exposure | **Database normalization** | **Stored procedure** | **Parameterized query**

	The intentional or unintentional disclosure of information to an unauthorized user or application.
	The process of organizing data in a database.
	A group of SQL statements created, compiled, and stored in a database.
	A query in which placeholders are used for parameters and the parameter values are supplied at execution time.

Reset

Database encryption

Database encryption protects sensitive data through one or more selective encryption methods. Selective encryption optimizes system performance by focusing encryption efforts on the most sensitive or vulnerable data elements, rather than encrypting the entire dataset. Such an approach reduces computational overhead and storage requirements while ensuring efficient and robust protection of critical information. Selective encryption methods:

- ***Column-level encryption*** , or ***columnar encryption*** is selective encryption configured on a specified database table column or columns. Ex: A database stores customer information, including social security numbers, and column-level encryption encrypts the social security number column and leaves less sensitive columns (names and addresses) unencrypted.
- ***Record-level encryption*** is selective encryption configured on all instances of a specified database table record. Ex: A healthcare database can selectively encrypt each patient's medical history, including diagnoses and treatment plans, using record-level encryption.

Table 8.4.1: Comparison of column-level and record-level database encryption methods.

Feature	Column-level encryption	Record-level encryption
Description	Encrypts specific columns within a table	Encrypts entire records within a table
Performance impact	Lower computational overhead due to targeted encryption	Higher overhead from encrypting full records but more efficient than full database encryption
Security considerations	Ideal for databases with mixed sensitivity levels across columns	Best for records where most or all data is sensitive
Example	Retails databases encrypt only columns with credit card details	Financial databases encrypt entire records to secure all transaction details

PARTICIPATION ACTIVITY 8.4.6: Database security.

1) What is the primary benefit of selective encryption in database encryption techniques?

   - Encrypts entire datasets for maximum security
   - Optimizes system performance by focusing on sensitive data
   - Increases computational overhead and storage requirements

2) What type of encryption targets specific data columns?

   - Record-level encryption
   - Columnar encryption

- Hashing

3) What is the benefit of using record-level encryption?

- Allows for different security levels based on data sensitivity
- Encrypts entire datasets for maximum security
- Increases computational overhead and storage requirements

Secure storage techniques

Securing stored authentication data like passwords is vital for preventing credential exposure if a database is compromised. One technique for improving the security of stored credentials is hashing. Hashing transforms a variable-length string to a fixed-size value, known as a hash. Since hashing is a one-way process and the original string cannot be derived from the hash, databases store password hashes instead of passwords. During authentication, the hash of the user-entered password is computed and compared with the stored password hash. If the hashes match, the password is validated and the user is successfully authenticated.

Salting is implemented to enhance the security provided by hashing. ***Salting*** is the process of adding a unique string of characters, known as ***salt***, to a password before hashing the password. Salting protects against brute-force attacks utilizing rainbow tables. A ***rainbow table*** is a precomputed table of hashes for common passwords. A rainbow table enables an attacker to quickly find plaintext passwords corresponding to precomputed hash values, facilitating rapid decryption of password hashes.

PARTICIPATION ACTIVITY 8.4.7: Secure storage techniques.

1) What is the result of hashing a password?

- A fixed-size value
- A plaintext password
- An encrypted password

2) What is salting used for in password security?

- To store passwords in plaintext

- To protect stored password hashes from brute force attacks
- To transform a variable length string to a fixed-size value

3) Why does salting a password make the password more difficult for a rainbow table attack to succeed?

- Salting makes the hash algorithm reversible
- Salting increases the number of possible hash values, making pre-computation impractical
- Salting encrypts the hash with another layer of security

8.5 Email security

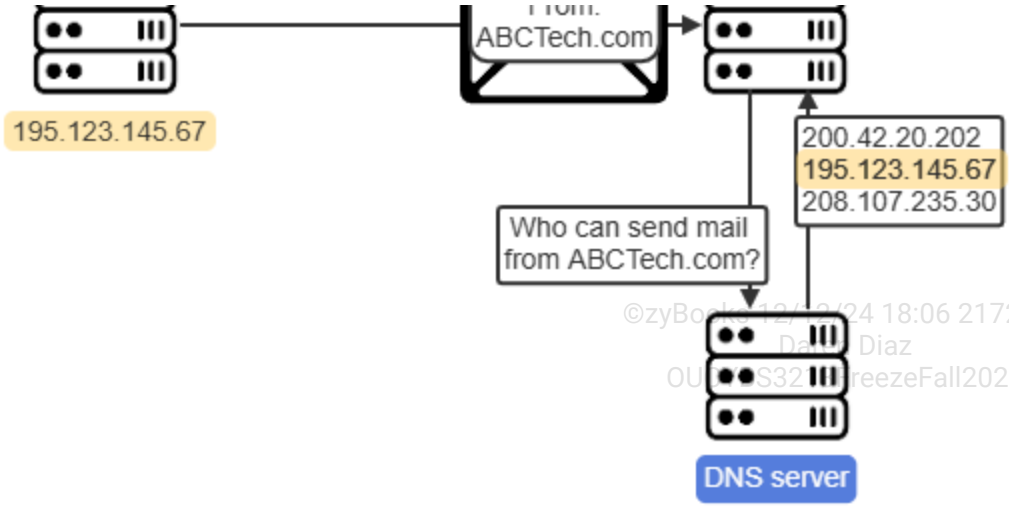
Sender policy framework

An attacker can use email as an attack vector by posing as a legitimate sender. Ex: An attacker sends an email appearing to be from Amazon that asks the recipient to provide credit card information.

Sender policy framework (SPF) is an email authentication method that ensures the sending mail server is authorized to send emails from the sender's domain. SPF is implemented by including an SPF TXT record in a domain's DNS record that specifies authorized email senders for the domain.

SPF improves email security by reducing the risk of phishing attacks where attackers impersonate a trusted domain.

PARTICIPATION ACTIVITY 8.5.1: Sender policy framework (SPF).



Animation content:

Static figure: An email sent from ABCTech.com to a customer. The receiving mail server asks the DNS server "Who can send mail from ABCTech.com?" The DNS server responds with a list of IP addresses. One of the IP addresses matches the sending mail server's IP address.

Animation captions:

1. An ABC Tech employee sends an email to a customer. The email's envelope-from domain is ABCTech.com.
2. To verify that the email was sent from an authorized ABC Tech mail server, the receiving email server queries the DNS for the sending domain's SPF record.
3. The DNS server responds with a list of IP addresses that are authorized to send email with the ABCTech.com envelope-from domain.
4. The receiving mail server verifies that the sending mail server IP address is included in the list of authorized IP addresses.

PARTICIPATION ACTIVITY | 8.5.2: SPF.

1) SPF verifies that an email's ____.

- contents were not changed in transit
- envelope-from and header-from addresses match
- sending mail server is authorized to send from the

given domain

2) SPF information is included in a domain's DNS record as a/an _____ record.

- MX
- TXT
- A

3) The *mycompany.com* DNS record includes the SPF TXT record:

```
v=spf1 ip4:196.219.143.125 ~all
```

The SPF TXT record includes _____ to instruct a receiving mail server on handling email from an unauthorized sending mail server.

- `v=spf1`
- `ip4:196.219.143.125`
- `~all`

4) A mail server receives an email with the envelope-from domain *techcompany.com*. The receiving mail server sends a DNS query, but *techcompany.com*'s DNS record does not include an SPF TXT record.

The SPF check returns a result of _____.

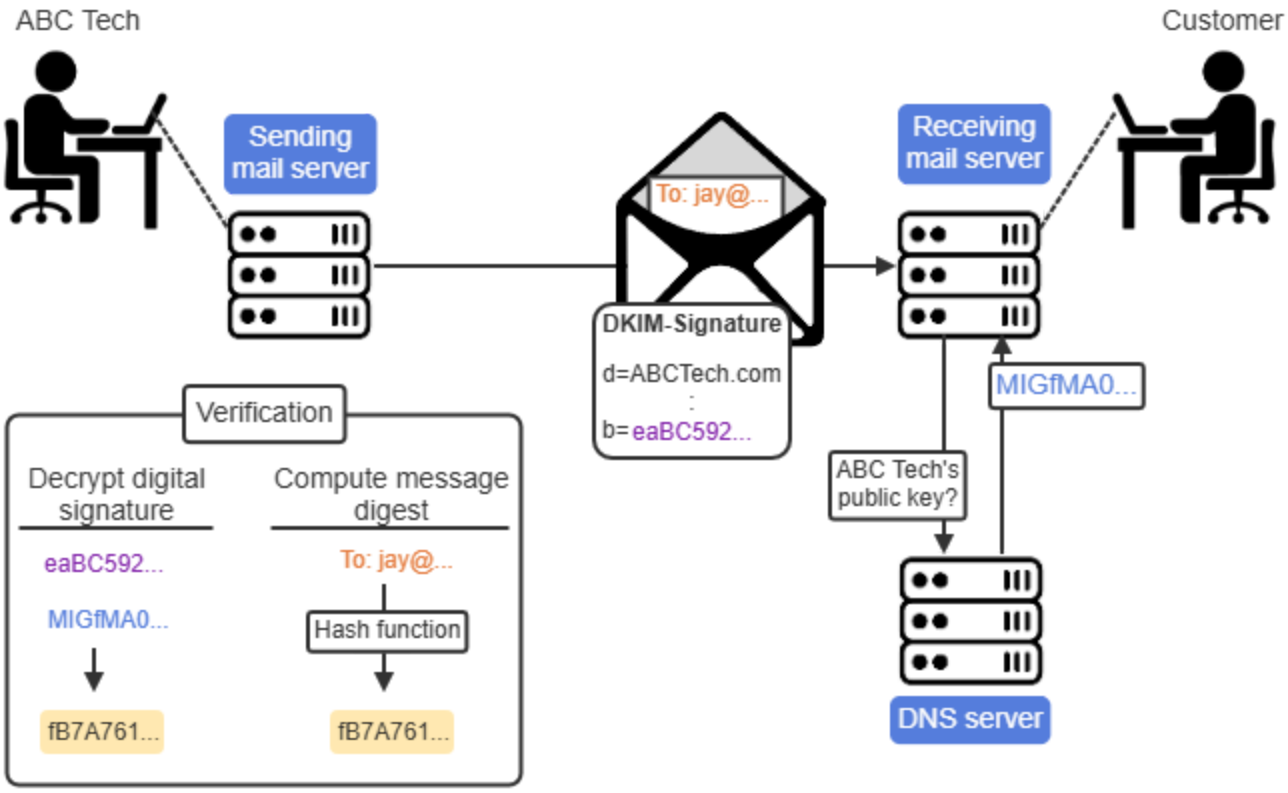
- Fail
- None
- Softfail

DomainKeys identified mail

SPF verifies that email was sent from an authorized mail server but does not confirm the authenticity of the email's content. ***DomainKeys identified mail (DKIM)*** is an email authentication method that uses digital signatures to verify that an email was sent and authorized by the owner of the sending domain. Implementing DKIM requires adding a domain's public key to the domain's DNS records and embedding a DKIM-Signature header in every email sent. The DKIM-Signature header includes a digital signature that is verified by the receiving mail server using the domain's public key.

DKIM improves email security by verifying the sender's identity and the integrity of the message content. DKIM helps mitigate email-based threats such as spoofing, tampering, and phishing attacks that rely on forged sender information to deceive recipients.

PARTICIPATION ACTIVITY 8.5.3: DomainKeys identified mail (DKIM).



Animation content:

Static figure: A sending mail server and a receiving mail server. An email sent from ABCTech.com with a DKIM-Signature.

Step 1: An ABC Tech employee sends an email to a customer. A DKIM-Signature field is added to the email's header.

An email appears near the sending mail server. A "DKIM-Signature" text box appears on top of the email.

Step 2: The DKIM-Signature field specifies the signing domain and includes the digital signature of the email's content.

The line "d=ABCTech.com" appears in the DKIM-Signature box. Two dots appear in the next line, implying skipped text in the DKIM-Signature box. The line "b=eaBC592..." appears as the last line of the DKIM-Signature box. The email and DKIM-Signature box move towards the receiving mail server.

Step 3: The receiving mail server queries the DNS for ABCTech.com's DKIM records and receives the domain's public key in return.
A DNS server appears. The receiving mail server asks for ABC Tech's public key. The DNS server responds with "MIGfMA0..."
Step 4: The receiving mail server decrypts the digital signature using the public key to get the message digest.
A box labeled "Verification" appears with a column labeled "Decrypt digital signature." A copy of the text "eaBC592..." moves from the DKIM-Signature box to the Decrypt digital signature column. A copy of the text "MIGfMA0..." moves from the DNS server response to below "eaBC592..." in the Decrypt digital signature column. An arrow appears below "MIGfMA0...". The text "fB7A761..." appears below the arrow.
Step 5: The receiving mail server then computes the message digest from the email's content. The message digest is compared to the decrypted digital signature for verification.
A column labeled "Compute message digest" appears in the verification box. The email envelope opens, and a text box emerges with the text "To: jay@..." A copy of the text "To: jay@..." moves from the email to the Compute message digest column. An arrow labeled "Hash function" appears below "To: jay@...". The text "fB7a761..." appears below the Hash function arrow. The text "fB7a761..." is highlighted in the Decrypt digital signature column and in the Compute message digest column.

Animation captions:

1. An ABC Tech employee sends an email to a customer. A DKIM-Signature field is added to the email's header.
2. The DKIM-Signature field specifies the signing domain and includes the digital signature of the email's content.
3. The receiving mail server queries the DNS for ABCTech.com's DKIM records and receives the domain's public key in return.
4. The receiving mail server decrypts the digital signature using the public key to get the message digest.
5. The receiving mail server then computes the message digest from the email's content. The message digest is compared to the decrypted digital signature for verification.

PARTICIPATION ACTIVITY 8.5.4: DKIM.

1) DKIM uses a digital signature for _____.
   - ○ increased confidentiality
   - ○ receiver verification
   - ○ sender verification

2) The receiving mail server requests the public key for the domain given in the _____.

- DKIM-Signature `d=` tag
- envelope-from address
- header-from address

3) The *mycompany.com* DNS record includes DKIM information, but a mail server receives an email from *mycompany.com* without a DKIM-Signature header field. The receiving mail server reports the DKIM result as _____.

- Pass
- Fail
- None

4) A mail server receives an email from *newcompany.com* with a DKIM-Signature header field, but the DNS query returns no DKIM information for *newcompany.com*. The receiving mail server reports the DKIM result as _____.

- Pass
- Fail
- None

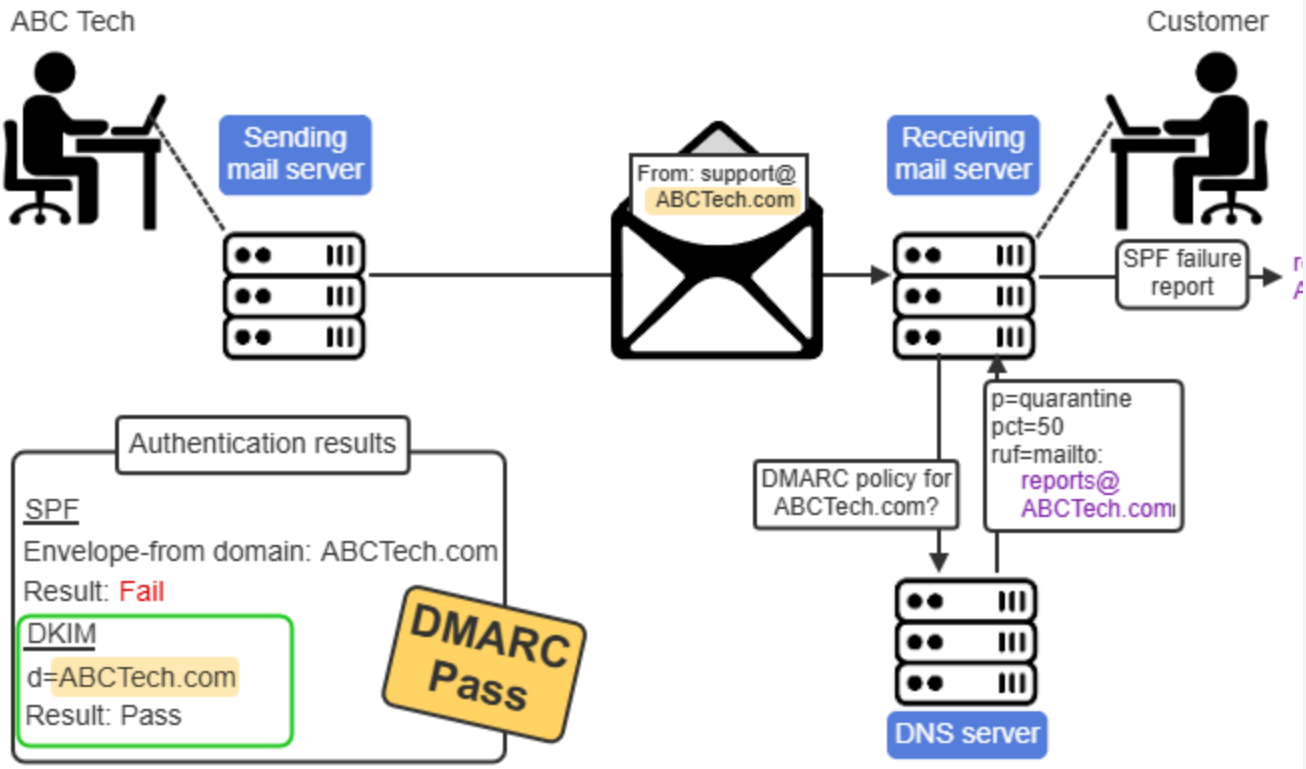
Domain-based message authentication, reporting, and conformance

SPF and DKIM aim to prevent fraudulent emails by verifying an email's sending domain. However, a receiving mail server that rejects all SPF or DKIM failures may unintentionally reject legitimate emails. Further, a domain owner is not notified of authentication failures so is unaware of rejected legitimate emails. Ex: A company adds a new sending mail server without updating the domain's DNS SPF record. Any emails sent from the new mail server fail the SPF verification. Since SPF lacks a mechanism for direct failure notifications, the domain's DNS SPF records are not updated with the new mail server's information.

Domain-based message authentication, reporting, and conformance (DMARC) is an email authentication protocol that secures email communications by verifying sender identities, specifying actions for authentication failures, and providing reports on email delivery and integrity. DMARC utilizes SPF and DKIM verification mechanisms to enforce and enhance email security policies. A DMARC record includes information such as:

- `p` - Policy indicating the requested action for an email that fails the DMARC check: reject, quarantine, or none
- `pct` - Percentage of received emails to which the policy is to be applied
- `rua` - Addresses to which aggregate reports should be sent
- `ruf` - Addresses to which failure reports should be sent

PARTICIPATION ACTIVITY | 8.5.5: Domain-based message authentication, reporting, and conformance (DMARC).



Animation content:

Static figure: Static figure: An email sent from ABCTech.com's sending mail server to a customer's receiving mail server.
Step 1: An ABC Tech employee sends an email to a customer. The receiving mail server requests the DMARC policy for the header-from domain, ABCTech.com, from the DNS server.
An email envelope moves from the sending mail server to the receiving mail server. The envelope opens and a page appears with the text "From: support@ABCTech.com." A DNS server appears. The receiving mail server requests the DMARC policy for ABCTech.com.
Step 2: The policy tells the receiving mail server to quarantine 50% of emails from ABCTech.com that fail DMARC and send failure reports to reports@ABCTech.com.

The text "p=reject; pct=50; ruf=mailto:reports@ABCTech.com" is sent from the DNS server to the receiving mail server.
Step 3: The receiving mail server conducts SPF and DKIM checks and records the domain used for each check. If both SPF and DKIM fail, the DMARC result is "fail."
A box labeled Authentication results appears. The label SPF appears in the Authentication results box. The text "From: ABCTech.com" appears on top of the email. The text "ABCTech.com" moves from the email to the SPF section of the Authentication results box and is labeled Envelope-from domain. The text "Result: Fail" appears in the SPF section. The label DKIM appears below the SPF section. A DKIM-Signature text box appears on top of the email. The text "d=ABCTech.com" moves from the DKIM-Signature text box to the DKIM section of the Authentication results box. The text "Result: Pass" appears in the DKIM section.
Step 4: Since DKIM passed, the DKIM d tag is compared to the email's header-from domain. The domains match, so the DMARC result is "pass."
The DKIM section of the Authentication results box is outlined in green. In the line "d=ABCTech.com", "ABCTech.com" is highlighted. In the line "From: support@ABCTech.com" within the email, the text "ABCTech.com" is highlighted. The text "DMARC Pass" appears on the Authentication results box.
Step 5: Since SPF failed, the receiving mail server sends a failure report to reports@ABCTech.com.

Animation captions:

1. An ABC Tech employee sends an email to a customer. The receiving mail server queries the DNS for ABCTech.com's DMARC policy.
2. The policy instructs the receiving mail server to quarantine 50% of emails from ABCTech.com that fail DMARC and send failure reports to reports@ABCTech.com.
3. The receiving mail server conducts SPF and DKIM checks and records the domain used for each check. If both SPF and DKIM fail, the DMARC result is "Fail."
4. Since DKIM passed, the DKIM d tag is compared to the email's header-from domain. The domains match, so the DMARC result is "Pass."
5. Since SPF failed, the receiving mail server sends a failure report to reports@ABCTech.com.

PARTICIPATION ACTIVITY 8.5.6: DMARC.

1) An email must have a pass result for SPF _____ DKIM in order to pass the DMARC check.
   - ○ and
   - ○ or

2) If an email passes SPF but fails DMARC, the DMARC failure is due to _____.

- a difference between the email's envelope-from and header-from domains
- the email failing DKIM verification
- the receiving mail server's DNS query returning no DMARC policy

3) The DMARC record for *mycompany.com* includes

```
p=reject; pct=30 .
```

mycompany.com requests that receiving mail servers _____.

- perform a DMARC check on 30% of emails and reject all emails that fail
- perform a DMARC check on all emails; of the emails that fail, reject 30% and ignore the rest
- perform a DMARC check on all emails; of the emails that fail, reject 30% and quarantine the rest

Email gateway

*An **email gateway** is a server that processes an organization's incoming and outgoing email to protect the organization's internal servers. An email gateway scans incoming email, rejecting unwanted emails and forwarding the rest to the organization's internal server.*

8.6 Code review, fuzzing, and automation

Code review

A ***code review*** is the process of systematically examining code with the aim of improving code quality. A code review examines the code's correctness, readability, and understandability.

Two code review methods exist:

- ***Dynamic code analysis***, also known as ***dynamic analysis***, is a code review method that examines code during the code execution. Dynamic code analysis is commonly automated and is performed by using test inputs that aim to cover all possible outputs.
- ***Static code analysis***, also known as ***static analysis*** or ***source code analysis***, is a code review method that examines code without executing the code. Static code analysis is performed by analyzing the code against a set of coding rules and can be performed manually or by automated tools. ***Manual code review*** is the process of performing static code analysis by one or more code reviewers.

In addition to code review, a comprehensive approach to codebase security involves proactive management of dependencies and third-party libraries. ***Package monitoring*** is the practice of systematically tracking and managing various elements within a software project. Regular package monitoring involves updating dependencies, validating licenses, and mitigating known vulnerabilities, thereby protecting the codebase and preventing potential threats. By identifying and addressing risks introduced by third-party dependencies, package monitoring maintains the integrity and security of the codebase. Ex: Package monitoring ensures that a web application's MySQL database runs with the latest release, preventing security vulnerabilities and optimizing performance.

Table 8.6.1: Comparison of code review methods.

Aspect	Dynamic code analysis	Static code analysis
Purpose	Identifies runtime issues and bugs	Checks code against coding rules
Inputs	Test inputs to observe runtime behavior	Source code
Methods	Analyzes code behavior during runtime	Checks code structure and syntax
Tools	Code profilers, debuggers	Static analysis tools, code analyzers

Benefits	Improves runtime performance and reliability	Enhances code quality and maintainability

PARTICIPATION ACTIVITY 8.6.1: Code review.

Select the code review method in each scenario.

1) Code review performed during code execution.
   - Dynamic
   - Static

2) Code review performed by one or more code reviewers.
   - Dynamic
   - Static

3) Code review commonly performed by automated tools.
   - Dynamic
   - Static

4) Code review performed manually.
   - Dynamic
   - Static

PARTICIPATION ACTIVITY 8.6.2: Code review and analysis.

How to use this tool

Static code analysis **Code review** **Dynamic code analysis**

The process of systematically examining code with the aim of improving code quality.

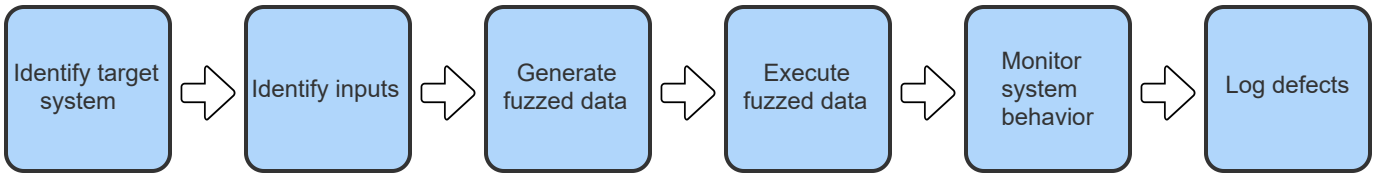
	A code review type that examines code without executing the code.
	A code review type that examines code during the code execution.

Reset

Fuzzing

A dynamic code analysis tool may use fuzzing to find errors in code. ***Fuzzing***, also known as ***fuzz testing***, is an automated software testing technique for finding software bugs and vulnerabilities by feeding invalid and random data into software. The aim of fuzz testing is to ensure that software can handle unexpected inputs. A ***fuzzer*** is a program that automatically generates random data for software testing. By generating random inputs instead of deliberately designed inputs, a fuzzer can potentially find input combinations that are not tested by the software's developers.

PARTICIPATION ACTIVITY 8.6.3: Fuzz testing.



Animation content:

Static image: A flowchart showing fuzz testing. The first box is "Identify target system". The second box is "Identify inputs". The third box is "Generate fuzzed data". The fourth box is "Execute fuzzed data". The fifth box is "Monitor system behavior". The final box is "Log defects".

Animation captions:

1. The software that is to be tested is identified (known as target system).
2. Input data is identified for testing.
3. Input data is converted into fuzzed data (randomized).
4. Testing process initiates using the generated fuzzed data.
5. Software behavior is examined with the fuzzed data.
6. In the final phase, the identified software defects are logged for further analysis and fixing.

PARTICIPATION ACTIVITY 8.6.4: Fuzzing.

1) In fuzz testing, only expected input is entered into software.
   - True
   - False

2) Fuzz testing involves the generation of random but valid data.
   - True
   - False

3) The aim of fuzz testing is to ensure software can handle unexpected input.
   - True
   - False

Automation

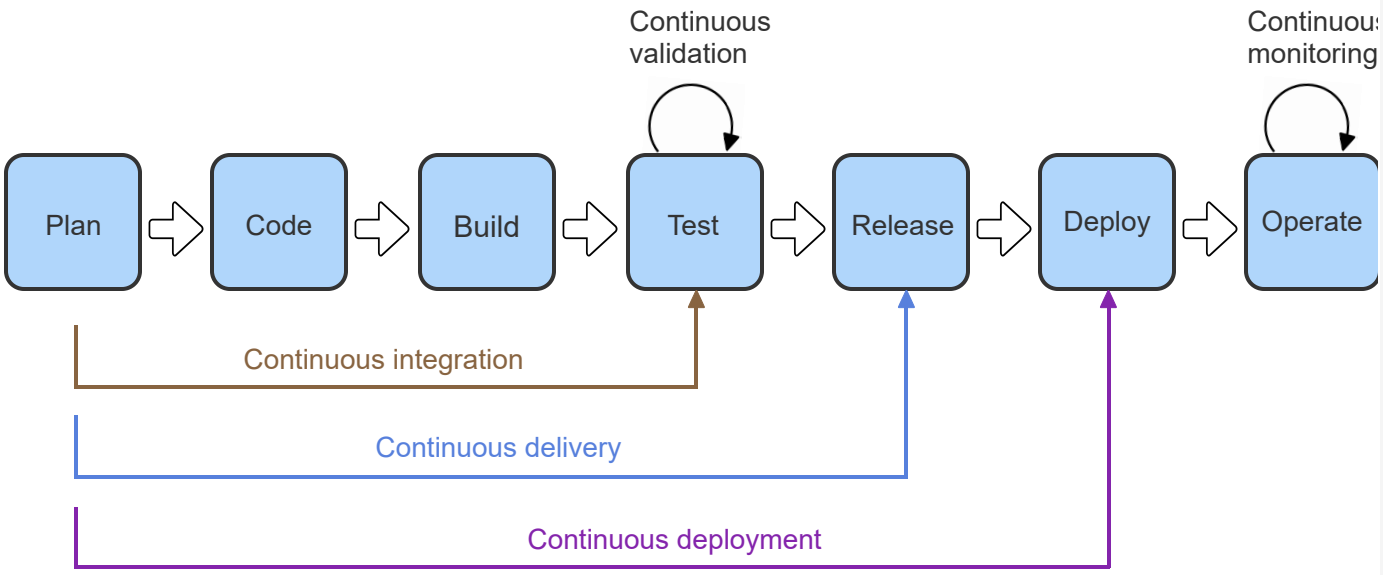
Development operations or ***DevOps*** is a set of practices and tools that automate and integrate software development and IT operations. The goal of DevOps is to improve the coding, building, testing, releasing, deploying, and monitoring elements of a software development life cycle (SDLC). DevOps facilitates faster and more frequent software deployments.

DevOps uses tools to automate manual and repetitive tasks in SDLC. Automation is commonly implemented using scripts written in a language such as Python. Automation is used in the various phases of DevOps:

- ***Continuous integration*** is the practice of frequently merging new and modified code from multiple developers into a single software project.

- ***Continuous delivery*** is the practice of frequently releasing new software that is ready to be deployed into a stage environment.
- ***Continuous deployment*** is the practice of frequently deploying new software into a production environment.
- ***Continuous validation*** is the practice of frequently verifying that new and existing software is tested.
- ***Continuous monitoring*** is the practice of frequently monitoring released software to detect errors, security risks and compliance issues.

PARTICIPATION ACTIVITY 8.6.5: Automation in DevOp.



Animation content:

Static image: A flowchart showing DevOps. The first box is "Plan". The second box is "Code". The third box is "Build". The fourth box is "Test". The fifth box is "Release". The sixth box is "Deploy". The final box is "Operate". A circular arrow labeled "Continuous validation" is above the Test box. A circular arrow above the Operate box is labeled "Continuous monitoring". A gold arrow labeled "Continuous integration" goes from the Plan box to the Test box. A blue arrow labeled "Continuous

delivery" goes from the Plan box to the Release box. A purple arrow labeled "Continuous deployment" goes from the Plan box to the Deploy box.

Animation captions:

1. Continuous integration is the frequent verification that different components of software work together correctly. Continuous validation is the frequent verification that new code is tested with existing code.
2. Continuous delivery is the frequent release of new software that is ready to be deployed.
3. Continuous deployment is the frequent deployment of new software in a production environment.
4. Continuous monitoring is the frequent monitoring of software in production to detect any errors, security risks and compliance issues.

PARTICIPATION ACTIVITY 8.6.6: Automation in DevOps.

1) What is the goal of DevOps?

- Separate the software development process from IT operations
- Deploy software faster, but less frequently
- Improve the elements of a software development life cycle (SDLC)

2) The frequent verification that different components of software work together correctly is referred to as continuous _____.

- integration
- delivery
- validation

3) The frequent release of new software that is ready to be deployed into a stage environment is referred to as continuous _____.

- deployment
- monitoring

delivery

CHALLENGE ACTIVITY | 8.6.1: Code review and automation.

581480.4344582.qx3zqy7

Start

Select the code review method in each scenario.

Pick — Examining a web browser application's code during execution.

Pick — Examining a media player application's code without executing the code.

Pick — Using test inputs with the aim of covering all possible outputs of a text edit application.

1 2

Check Next

8.7 OWASP and input validation

OWASP

The ***open web application security project (OWASP)*** is a non-profit foundation that publishes methodologies, documentation, tools, and technologies for raising awareness and improving web application security. ***Web application security***, also known as ***Web AppSec***, is the set of practices and technologies that enable the creation of secure web applications. Web AppSec addresses security concerns relating to the World Wide Web, HTTP, and web application software.

OWASP Top 10 is a regularly-updated list of the 10 most common web application security risks. The risks are based on the frequency of discovered web application security defects, the severity of the

vulnerabilities, and the magnitude of the vulnerabilities' potential impacts. OWASP Top 10 serves as an important checklist and an internal web application development standard for most organizations.

Table 8.7.1: OWASP Top 10 (2021).

Rank	Vulnerability
1	Broken access control
2	Cryptographic failures
3	Injection
4	Insecure design
5	Security misconfiguration
6	Vulnerable and outdated components
7	Identification and authentication failures
8	Software and data integrity failures
9	Security logging and monitoring failures
10	Server-side request forgery (SSRF)

PARTICIPATION ACTIVITY 8.7.1: OWASP.

1) The web application security risks in OWASP Top 10 are ranked by web application developers.
   - ○ True
   - ○ False

2) The OWASP Top 10 is a software application development standard for most organizations.

- True
- False

3) Web AppSec relates to web application software, HTTP, and World Wide Web.

- True
- False

PARTICIPATION ACTIVITY 8.7.2: OWASP.

How to use this tool

Web application security **OWASP Top 10** **OWASP**

	A foundation that publishes methodologies, documentation, tools, and technologies for raising awareness and improving web application security.
	The set of practices and technologies that enable the creation of secure web applications.
	A list of the 10 most common web application security risks.

Reset

Input validation

Input validation is a secure coding technique that ensures only valid input is entered into software. Software input should be valid both syntactically (valid syntax) and semantically (valid meaning). Input validation can prevent injection attacks such as SQL, XML, and LDAP injection by ensuring that only valid input is entered into a query or application.

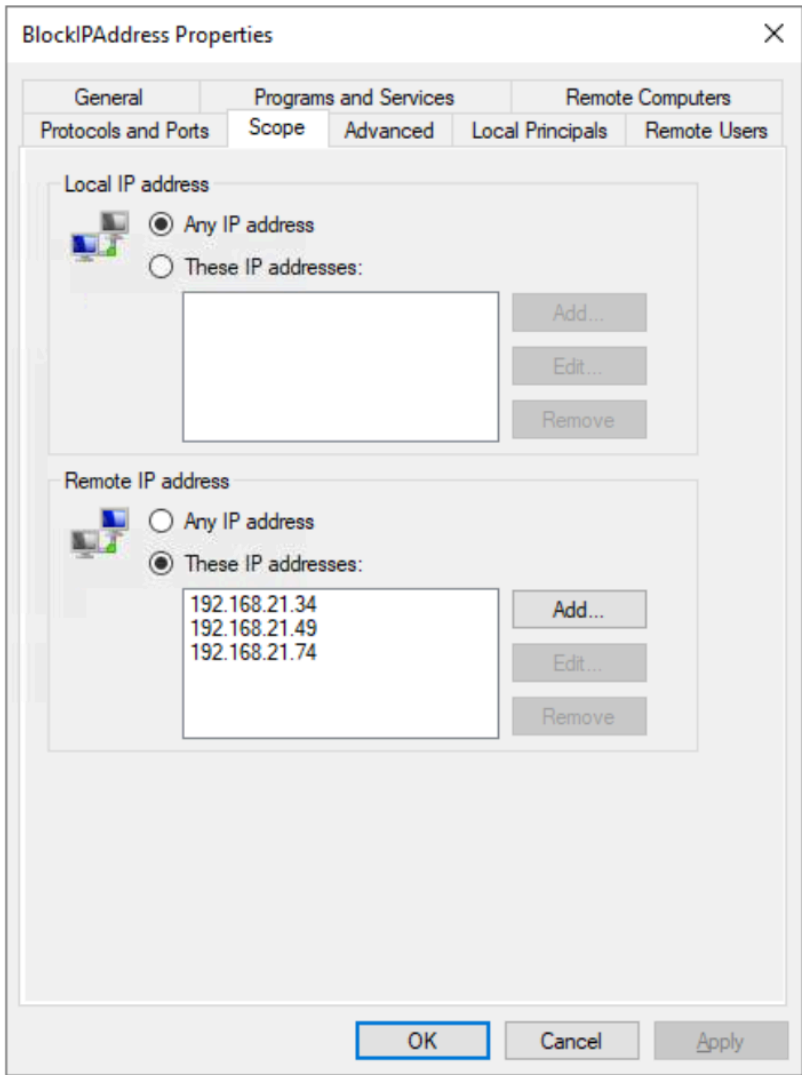
Two input validation methods exist:

- ***Allow list***, also known as ***white list***, is an input validation method that ensures an input matches a set of known good values.

- ***Block list***, also known as ***deny list***, is an input validation method that ensures an input does not contain known bad values.

Block lists are also used by firewalls to prevent network communications from specific IP addresses.

Example 8.7.1: A Windows Defender Firewall rule that blocks three IP addresses.



PARTICIPATION ACTIVITY 8.7.3: Input validation methods.

Select the best input validation method in each scenario.

1) Allow user input that is a number between 1 and 25

- Allow list
- Block list

2) Allow all user input except $

- Allow list
- Block list

3) Allow all user input that is a number less than 99

- Allow list
- Block list

4) Allow all user input except special characters, such as / and *

- Allow list
- Block list

PARTICIPATION ACTIVITY 8.7.4: Input validation.

How to use this tool

Input validation **Block list** **Allow list**

	A secure coding technique that ensures only valid input enters software.
	An input validation method that ensures an input matches a set of known good values.
	An input validation technique that ensures an input does not contain known bad values.

Reset

Input validation in web applications

Input validation in web applications can be performed both on the server and the client. ***Server-side validation*** is input validation that is performed by a server after input has been sent to the server. Server-side validation is used when server resources are required to validate input and is performed using a scripting language such as ASP.Net or PHP. Since input received from a client should not be trusted, server-side validation should be performed on all input.

Client-side validation is input validation that is performed by a client before input is sent to a server. Client-side validation is used if server resources are not required to validate input. Client-side validation is performed using HTML5 attributes or JavaScript. Client-side validation is less secure than server-side validation because input can be modified by the client after the client-side input validation has been completed. Input can also be modified while in transit or at an intermediary proxy. However, client-side validation is faster than server-side validation because the validation response does not have to travel to a server and back.

PARTICIPATION ACTIVITY | 8.7.5: Client-side input validation in HTML forms.

```
<!DOCTYPE html>
<html>
<body>
<form id="form" autocomplete>
   <input type="text" minlength="5" required>
   <input type="email" required>
   <input type="password" pattern="(?=.*[a-z]).{8}" required>
</form>
</body>
</html>
```


Animation content:

Static image: Start XML code.

```
<html>
<body>
<form id="form" autocomplete>
   <input type="text" minlength="5" required>
   <input type="password" pattern="(?=.*[a-x]).{8}" required>
</form>
```

</body>
</html>
End XML code.

Animation captions:

1. The 'required' attribute ensures that an input of type "text" is not empty and has a minimum of length of 5.
2. The 'required' attribute ensures that an input of type "email" is not empty and is in correct email format.
3. The 'required' attribute ensures that an input of type "password" is not empty and conforms to the regular expression specified in the "pattern" attribute.

PARTICIPATION ACTIVITY 8.7.6: Input validation in web applications.

1) Which scripting language is commonly used to perform server-side validation?
   - C++
   - PHP
   - Javascript

2) Why is input validation faster on a client than on a server?
   - Because a server is busy with validating input from many clients
   - Because a client always runs faster than a server
   - Because validation response does not have to travel to a server and back

3) Why is server-side input validation more secure than client-side input validation?
   - Because JavaScript is not a secure scripting language
   - Because input can be modified in transit or at a proxy after

client-side input validation has been completed

- Because server-side input validation is faster than client-side input validation

Secure cookies

A cookie is a data file that stores information about a user's web browsing session. Cookies are used to retain previously entered information, such as authentication verification or shopping cart contents. Intercepting authentication cookies can allow an attacker to gain access to a user's account. A ***secure cookie*** *is a cookie with the Secure attribute, which requires the cookie to be transmitted over secure channels like HTTPS. Sending cookies through encrypted channels increases confidentiality.*

8.8 LAB: Application fuzzing (Walkthrough)

IT-Labs are not printable at this time.

8.9 LAB: SQL injection (Walkthrough)

IT-Labs are not printable at this time.