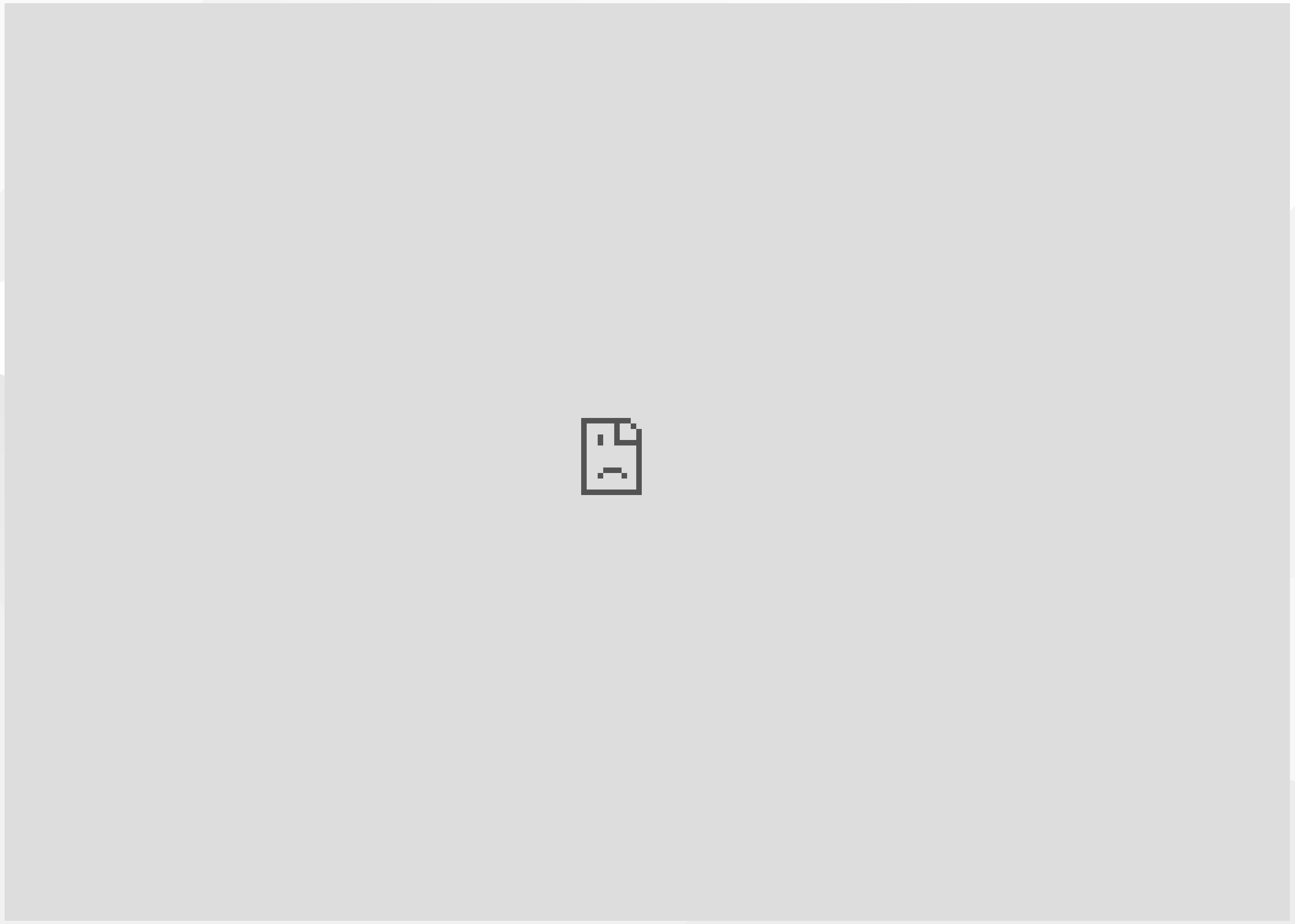
CEN429 Secure Programming

Week-3

Data Security: In Use, In Transit, and At Rest

Download

- PDF
- DOC
- SLIDE
- PPTX



Outline

- Data Security: In Use, In Transit, and At Rest
- Software Development Processes
  - Data-In-Use Security
  - Data-In-Transit Security
  - Data-At-Rest Security
- Protection of Dynamic and Static Assets

Week-3: Data Security - Data In Use, In Transit, and At Rest

Theoretical Topics and Applications

Data-In-Use Security refers to the protection of sensitive information held in memory while an application is running. This security measure is critical for preventing sensitive data from being captured by malicious software during processing.

Applications:

1. **Memory Encryption:** Encrypting sensitive data in memory.
2. **Misuse Detection:** Monitoring suspicious behavior in memory and intervening as needed.
3. **Data Manipulation Tests:** Testing if data is being altered accidentally or maliciously during runtime.
4. **Dynamic Memory Management:** Preventing memory leaks and minimizing data breaches.
5. **Continuous Authentication:** Re-authenticating users periodically during a session.
6. **Data Masking:** Ensuring sensitive data is visible only to authorized processes.
7. **Tamperproof Mechanisms:** Detecting if data in memory is being tampered with and triggering responses accordingly.

protected. Encryption, authentication, and integrity controls are used to ensure secure data transmission.

Applications:

1. **Session Key:** Dynamically generating a session key between client and server and encrypting communication with it.
2. **Device Binding:** Ensuring data can only be decrypted by a specific device to prevent unauthorized access on other devices.
3. **Version Binding:** Allowing only certain versions to communicate, preventing older, insecure versions from receiving data.
4. **Confidential Payload:** Encrypting the data so only authorized parties can read it.
5. **Integrity Control:** Verifying that data has not been altered or corrupted during transmission.
6. **Authenticity Control:** Authenticating the sender and receiver of the data.
7. **Secure Communication Channels:** Using SSL/TLS protocols for secure data

to the server. It is essential to authenticate the server and encrypt transmitted data.

Applications:

1. **Server Authentication Code:** Developing a custom mechanism to authenticate the server.
2. **Secure Server Communication:** Ensuring that data between server and client is encrypted using SSL/TLS.
3. **Session Key Encryption:** Encrypting data using session keys.
4. **Data Monitoring on Server:** Monitoring data traffic to and from the server to detect anomalies.
5. **Data Integrity Verification:** Ensuring that data is transmitted to the server without corruption using integrity checks.
6. **Data Encryption:** Encrypting data on the client-side before sending it to the server.
7. **Response Signing:** Verifying server responses with digital signatures.
8. **Server Backup:** Regularly backing up critical data on the server and storing it

Encryption and integrity controls prevent unauthorized access or attacks on stored data.

Applications:

1. **Whitebox AES:** Securing data using the whitebox method with the AES algorithm in storage.
2. **Whitebox DES:** Encrypting data with the Whitebox DES algorithm and performing security tests.
3. **Security Shell Matrix:** Creating a security shell in the file system to ensure safe storage of data.
4. **Key Management:** Storing encryption keys securely and rotating them regularly.
5. **Encrypted Database:** Encrypting sensitive data in the database, allowing access only to authorized users.
6. **Data Encryption at Rest:** Storing all data in encrypted format to prevent unauthorized access.

Static assets consist of data that remains unchanged in databases or on storage devices. Protecting these assets is crucial to maintain data integrity and prevent unauthorized access.

Applications:

1. **Key Encryption:** Encrypting static keys to ensure their secure storage.
2. **Source Code Protection:** Developing mechanisms to prevent unauthorized copying or modification of source code.
3. **Static File Integrity Checks:** Ensuring the integrity of static files and preventing unauthorized modifications.
4. **Data Signature:** Using digital signatures to verify that stored data has not been altered.
5. **Database Integrity:** Encrypting and protecting critical data in the database.
6. **File Access Control:** Implementing access control mechanisms to protect static files from unauthorized access.
7. **Secret Key Management:** Safely storing and managing static keys.

keys.

Applications:

1. **Dynamic Key Security:** Ensuring dynamic keys are used only during specific sessions and securely rotated.
2. **Session Data Encryption:** Encrypting session data to ensure its confidentiality.
3. **Protection of Device Fingerprints:** Ensuring device fingerprints are verified only by authorized parties.
4. **Session Data Protection:** Securing dynamic session data through encryption.
5. **Dynamic Key Management:** Safely creating and managing dynamic keys during a session.
6. **Session Timeout:** Implementing automatic session timeout to enhance security.
7. **Continuous Monitoring of Data:** Monitoring dynamic data with encryption and detecting security breaches instantly.
8. **Prevention of Data Manipulation:** Establishing security mechanisms to prevent

The properties of an asset include its name, description, location, source, size, creation, and deletion times. It is also essential to determine how to protect an asset according to security needs like confidentiality, integrity, and authentication.

Applications:

1. **Asset Name:** Defining the name of the asset and identifying what it represents.
2. **Description:** Explaining the function and information contained in the asset.
3. **Location:** Defining the physical location of the asset, such as a database, table, or column.
4. **Source:** Identifying the process or data source from which the asset originated.
5. **Size:** Defining the asset's size to optimize storage needs.
6. **Creation Time:** Logging the date and time when the asset was created.
7. **Destruction Time:** Determining when and how the asset should be destroyed.
8. **Default Value:** Defining the default value of the asset and what it should be initially.

Week Summary and Next Week

This Week:

- Data Security In Use, In Transit, and At Rest
- Protection of Static and Dynamic Assets

Next Week:

- Certificates and Encryption Methods
- Authentication and Data Integrity

$End - Of - Week - 3$