**WEEK 10:**

**Task 1. Essential Eight Mitigation Strategies**

Four of the Essential Eight mitigation strategies:

Application Whitelisting (Malware Delivery and Execution Prevention):

* Implement application whitelisting to restrict the use of unauthorized apps and only allow approved applications to run.
* Project Scenario: Employ application whitelisting on critical servers and workstations to prevent the execution of unauthorized or malicious software that could compromise sensitive project data and systems.

User Application Hardening (Preventing the Delivery and Execution of Malware):

* Develop user applications with minimal vulnerabilities and attack surface.
* Apply security patches, disable unused functionality, and activate security features of the applications.
* Project Scenario: Enhance the security of user applications, such as web browsers and office productivity programs, by regularly updating them with the latest security patches and configuring them to protect against potential attacks or vulnerabilities.

Patching of Software:

* Keep all software and programs up to date with the latest security patches to address known vulnerabilities.
* Project Scenario: Regularly apply updates and upgrades to the project's operating systems, programs, and applications to minimize the risk of vulnerabilities.

Daily Backups with Data Recovery and System Availability:

* Perform regular backups of critical project data to ensure data resilience and enable quick recovery in the event of a cyber disaster.
* Project Scenario: Establish a reliable backup strategy that includes frequent backups of project data, settings, and system images to maintain the availability and integrity of essential project data.
* These paraphrased strategies focus on implementing application whitelisting, hardening user applications, applying software patches, and maintaining regular backups to mitigate cyber risks and enhance the security of the project.

**Reasons for Selection:**

• These strategies were selected considering their suitability for addressing the specific needs and potential threats of the project.

• The adoption of user application hardening and application whitelisting was motivated by the objective of actively preventing malware delivery and execution, thereby minimizing the risk of unauthorized or malicious software compromising the project's systems.

• The inclusion of application patches was driven by the goal of mitigating cybersecurity incidents by addressing known vulnerabilities that attackers could exploit.

• Daily backups were chosen to ensure data and system availability, enabling prompt recovery in case of data loss or disruptions to the system.

These paraphrased statements highlight the rationale behind the selection of these strategies, emphasizing their alignment with the project's requirements and the goal of enhancing security and resilience.

**Task 2. Explore and Select NIST Controls**

AC-2 Account Management: The AC-2 Account Management control involves the creation, maintenance, monitoring, and removal of user accounts in accordance with organizational standards and procedures. It aims to prevent unauthorized access and misuse of personal data by implementing appropriate authentication and authorization processes. Measures such as enforcing strong password rules, implementing multifactor authentication, using role-based access control, setting account expiration and revocation policies, maintaining audit logs, and employing role-based access control can be implemented to ensure effective account management.

AU-6 Audit Review, Analysis, and Reporting: The AU-6 Audit Review, Analysis, and Reporting control ensures that audit records of the information system are regularly examined, analyzed, and reviewed for any signs of improper or irregular behavior. It involves reporting such findings to designated officials or authorities in a timely and relevant manner. By doing so, this control helps in the detection and response to potential security incidents or privacy breaches involving personal data. Routine review of audit logs, identification of anomalies or suspicious patterns, generation of reports or alerts, and escalation of issues as required, either through automated tools or manual methods, can be employed to implement this control effectively.

CM-6 Configuration Settings: The CM-6 Configuration Settings control ensures that the information system is configured in accordance with organizational policies and standards. It involves applying consistent and secure settings to various system components such as operating systems, applications, databases, firewalls, etc., to maintain the security and privacy of personal data. Implementation of this control includes establishing baseline configurations, applying patches and updates, enforcing security hardening measures, and monitoring for any deviations or modifications, which can be done manually or through the use of configuration management software.

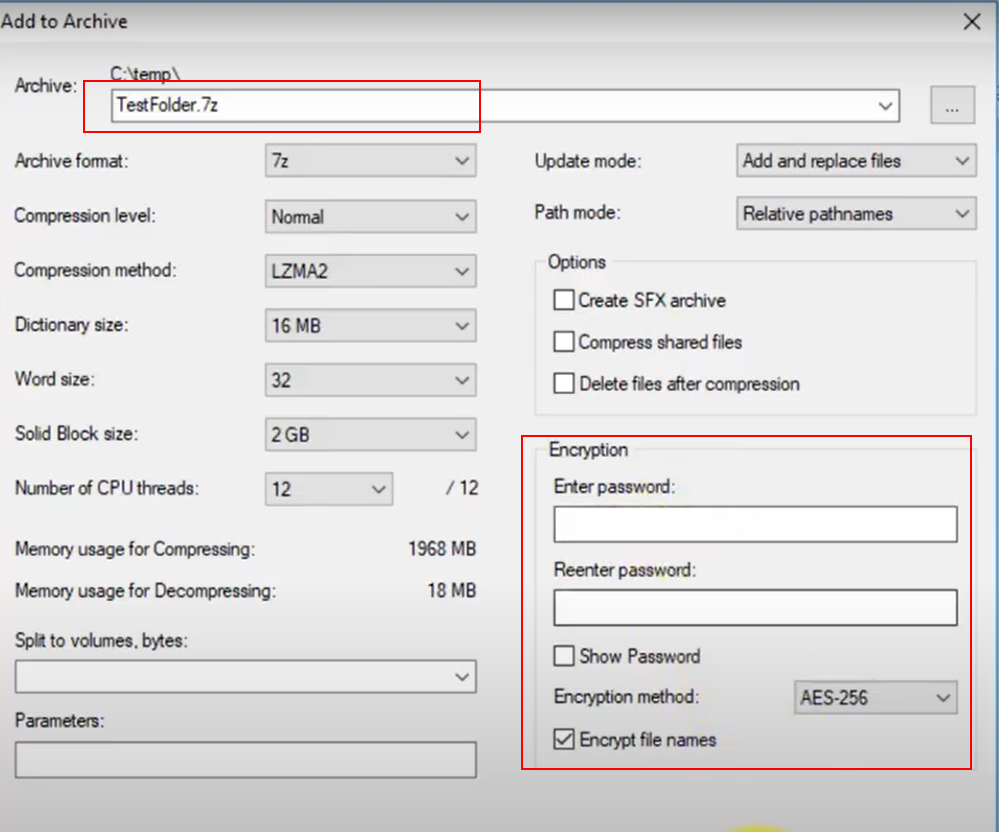
IA-2 Identification and Authentication (Organizational Users): The IA-2 Identification and Authentication control ensures that organizational users or processes operating on their behalf are uniquely identified and verified before accessing the information system. This control helps prevent unauthorized access and exploitation of personal data by confirming the legitimacy of users or processes. Various methods of identification and authentication, such as usernames and passwords, tokens, certificates, biometrics, etc., can be employed to achieve this control.

IP-1 Information Protection Processes and Procedures: The IP-1 Information Protection Processes and Procedures control focuses on the establishment, documentation, sharing, evaluation, updating, and approval of information protection policies and practices. It aims to safeguard the confidentiality, integrity, availability, and quality of personal data. Implementing this control involves developing comprehensive information protection programs that cover all aspects of data lifecycle management, including data collection, processing, storage, transfer, use, disclosure, retention, and destruction. This control also encompasses defining roles and responsibilities, establishing rules and guidelines, implementing standards and best practices, and conducting training and awareness programs.

SA-11 Developer Security Testing and Evaluation: The SA-11 Developer Security Testing and Evaluation control ensures that security testing and evaluation of the information system are conducted during its development phase. It helps identify and rectify systemic security flaws or vulnerabilities that may compromise the confidentiality or security of personal data. Implementation of this control involves employing various security testing and evaluation methodologies, such as code review, static and dynamic analysis tools, penetration testing, vulnerability scanning, etc., to assess the system's security posture.

**Task 3. Encrypt a File**

**Screenshot of the settings used to encrypt the file.**

****

**Q. Discuss how you shared the secret key, the limitations of that approach, and recommendations for more secure ways to share a secret key.**

When sharing a secret key for encryption purposes, it is essential to ensure secure and confidential transmission to maintain the integrity and confidentiality of the encrypted data. One method to accomplish this is through secure, encrypted channels like secure messaging apps, encrypted email, or file-sharing services that employ strong encryption protocols. The secret key can be securely shared by encrypting it with the recipient's public key or by using a pre-shared secret key exchanged through a secure, out-of-band communication channel.

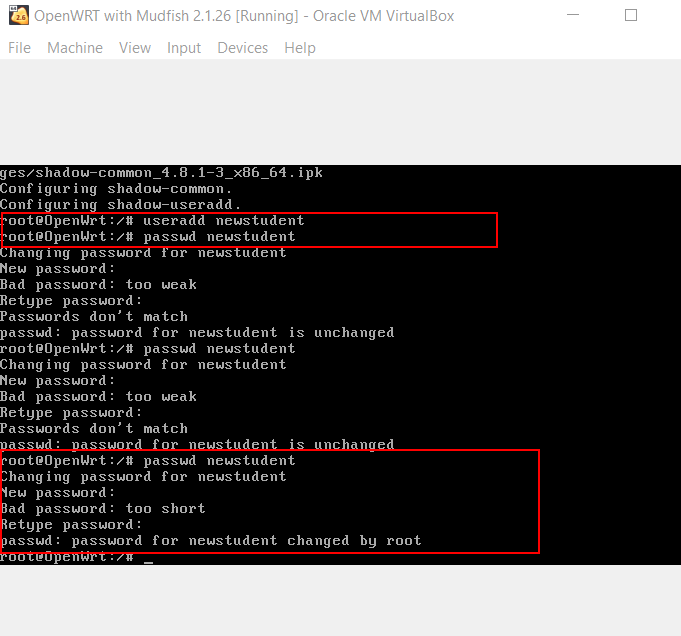
However, there are limitations associated with this approach, including the reliance on the trustworthiness and security of the communication channel used for key sharing. If the channel is compromised or intercepted, the secret key could be exposed, compromising the encryption and the confidentiality of the data. Additionally, the insecure storage or retention of shared keys by either the sender or the recipient can make them vulnerable to unauthorized access or theft. To mitigate these risks, proper key management practices, such as secure storage and limited access, should be implemented.

To ensure more secure ways of sharing a secret key, consider using asymmetric encryption algorithms like RSA or Elliptic Curve Cryptography (ECC), which involve the encryption of the key with the recipient's public key and decryption with their private key, ensuring confidentiality during the key exchange. Implementing Perfect Forward Secrecy (PFS) is also advisable as it generates unique session keys for each communication session, preventing the compromise of a single key from impacting the security of other sessions. Secure key exchange protocols like Diffie-Hellman or Elliptic Curve Diffie-Hellman (ECDH) can be utilized to establish a shared secret key securely over an insecure communication channel.

Adhering to secure key management practices, including secure storage, limited access, and regular key rotation, is crucial. Additional layers of authentication, such as two-factor authentication (2FA) or multi-factor authentication (MFA), can be implemented to strengthen the security of the key exchange process. Utilizing hardware security modules (HSMs) or secure key storage solutions can also provide added protection against unauthorized access to the keys.

**Task 4. View Password Information Stored in Linux**

**Screenshot or copy-and-paste of the /etc/shadow file entries that show your new user and password information.**



Adding new user and password information

A screenshot of a computer

Description automatically generated

/etc/shadow file entries that show password in hash format

**Q. Explanation of the password information stored in /etc/shadow, and why the actual password is not stored.**

In Unix-like operating systems, including Linux, the password information for user accounts is stored in the /etc/shadow file. However, the passwords themselves are not stored in plain text. Instead, the /etc/shadow file contains hashed versions of the passwords. The purpose of this approach is to enhance security and protect user accounts from unauthorized access.

Each line in the /etc/shadow file represents a user account and contains various fields separated by colons (:). The first field identifies the username associated with the account. The second field stores the hashed representation of the password, which is generated using a one-way hashing function. This function transforms the original password into a fixed-length string of characters, making it computationally infeasible to retrieve the original password from the hash.

Additional fields in the /etc/shadow file store information related to password policies and account settings, such as password change time, minimum and maximum password age, and account expiration date.

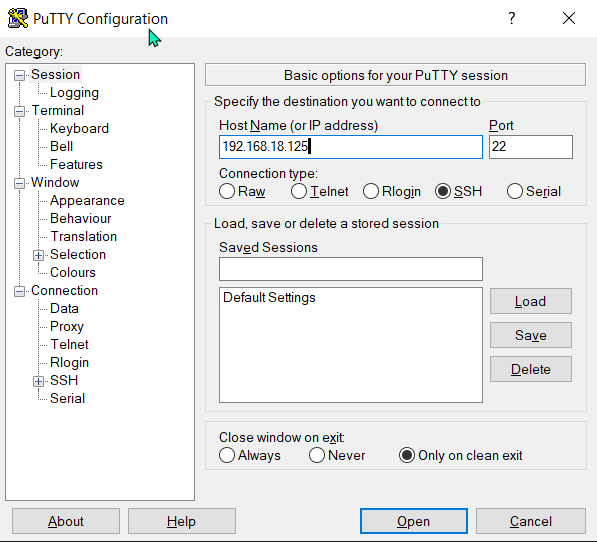
Storing only the hashed passwords instead of the actual passwords enhances security. If the /etc/shadow file is compromised, it would be challenging for an unauthorized user to retrieve the original passwords from the hashes. The use of hashes adds a layer of protection and makes it more difficult for attackers to gain access to user accounts.

During the authentication process, when a user enters their password, the system applies the same hash function used during account creation to the entered password and compares the resulting hash with the stored hash in the /etc/shadow file. If the hashes match, the password is considered correct, and the user is granted access.

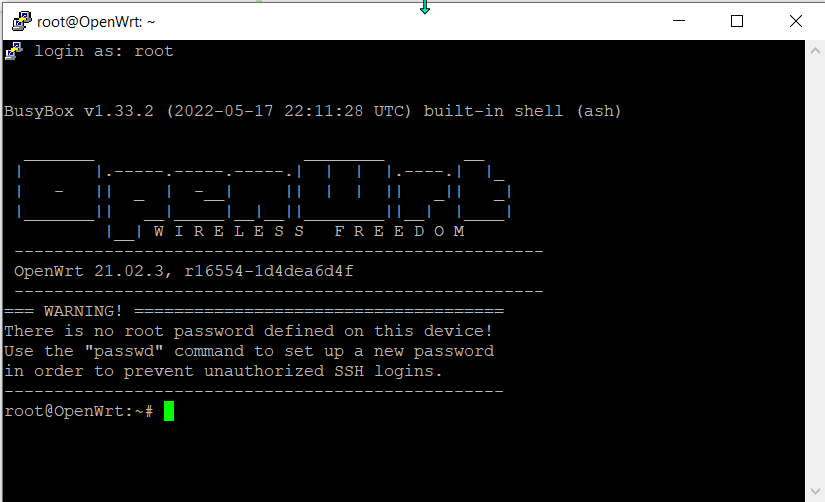
This approach significantly improves security because the actual passwords are not directly accessible even if an attacker gains access to the system's file system. It helps safeguard user accounts and makes the system more resistant to password-related attacks, such as brute-force or dictionary attacks.

**Task 5. Setup Key-Based Authentication**

**Key-based SSH login for OpenWRT Linux VM using PuTTY (see lecture demonstration )**

****

**Putty Screen**



**Q. Explain why key-based authentication can be more secure than password-based authentication when connecting to a SSH server (e.g. on OpenWRT, GitHub or Azure).**

Key-based authentication, also known as public key authentication, can provide enhanced security when connecting to an SSH server compared to password-based authentication for several reasons:

Enhanced encryption: Key-based authentication employs asymmetric encryption, typically utilizing RSA or ECDSA algorithms. The private key, securely stored on the client-side, is never transmitted over the network. Instead, the public key stored on the server is used to encrypt the authentication challenge. This ensures stronger encryption and mitigates the risks associated with intercepting and brute-forcing passwords used in password-based authentication.

Mitigation of password vulnerabilities: Passwords are often weak, reused, or easily guessed, making them susceptible to various attacks like dictionary attacks and credential stuffing. Key-based authentication eliminates the need for passwords entirely, reducing the potential vulnerabilities associated with password-based authentication.

Support for Two-Factor Authentication (2FA): Key-based authentication can be combined with additional security measures such as passphrase-protected private keys or hardware tokens, enabling the implementation of 2FA. This adds an extra layer of security, making it significantly more challenging for unauthorized individuals to gain access to the SSH server.

Centralized key management: Key-based authentication allows for centralized key management, simplifying the control and tracking of access to SSH servers. Server administrators can manage public keys stored on the server, granting or revoking access, instead of dealing with individual user passwords. This streamlines access rights management and reduces the risk of unauthorized access due to compromised passwords.

Improved auditing and accountability: Key-based authentication provides better auditing capabilities and accountability compared to password-based authentication. Each key is associated with a specific user or entity, enabling administrators to track actions performed on the SSH server more effectively. This is crucial for security and compliance purposes, facilitating the tracking of server access and user activities.

Key-based authentication offers a more secure and robust approach to connecting to SSH servers, addressing the vulnerabilities associated with passwords. It leverages stronger encryption, provides advanced authentication options like 2FA, enables centralized key management, and enhances auditing capabilities for improved security and accountability.