Assignment 1

Q1. Among windows and linux which one provides security and list them down.

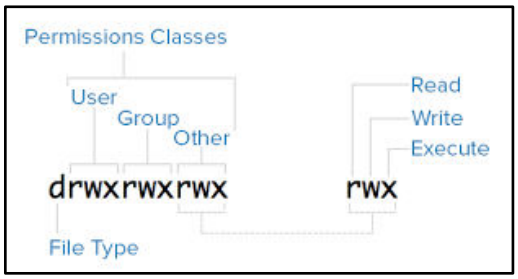
Both Windows and Linux can be secure when configured and managed correctly. The level of security depends on various factors, including how the operating system is used and the security measures implemented. Here are some security aspects of both:

Windows:

- **User Account Control (UAC):** UAC prompts users for permission before making system changes, reducing the risk of unauthorized actions.
- **BitLocker:** Encrypts the entire hard drive to protect data at rest.
- **Windows Defender:** Built-in antivirus and antimalware software.
- **Patch Management:** Regular updates and security patches provided by Microsoft.
- **Active Directory:** Offers centralized user management and access control.

Linux:

- **User Privileges:** Linux follows the principle of least privilege, limiting user rights.



- **Open Source:** Transparency allows for community scrutiny and rapid security fixes.
- **Permissions:** Granular file and directory permissions enhance control.
- **Security-Enhanced Linux (SELinux):** A security feature for access control policies.
- **Firewalls and IPTables:** Robust firewall capabilities.
- **Package Managers:** Simplify software updates and security patching.

It's essential to note that the security of both operating systems heavily depends on the user or administrator's actions and configurations.

Q2. Explain the critical components of cybersecurity governance.

Cybersecurity governance involves establishing policies, procedures, and practices to manage and protect an organization's information assets. Critical components include:

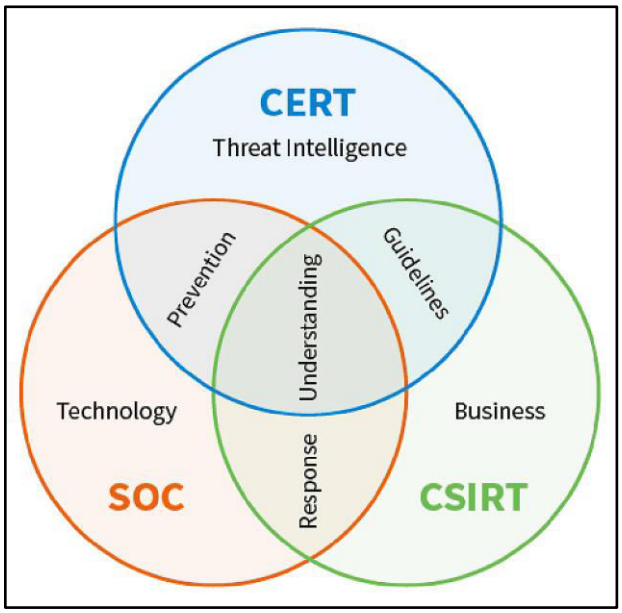
- **Cybersecurity Policies and Procedures:** Define the rules and guidelines for protecting data and systems, including acceptable use policies, incident response plans, and risk management procedures.
- **Risk Management:** Identify, assess, and prioritize cybersecurity risks. Implement risk mitigation strategies and regularly review risk assessments.

- **Compliance:** Ensure compliance with relevant laws, regulations, and industry standards, such as GDPR, HIPAA, or ISO 27001.
- **Security Awareness and Training:** Educate employees and stakeholders about cybersecurity best practices and threats.
- **Incident Response and Management:** Develop a plan for responding to and recovering from security incidents. Define roles and responsibilities.
- **Access Control:** Implement user authentication, authorization, and access control mechanisms to limit unauthorized access.
- **Security Monitoring and Threat Intelligence:** Continuously monitor networks and systems for suspicious activities. Stay updated on emerging threats through threat intelligence.
- **Security Testing and Assessment:** Conduct regular vulnerability assessments, penetration testing, and security audits.
- **Security Architecture:** Design and maintain a secure network and system architecture, including firewalls, intrusion detection systems, and encryption.
- **Third-Party Risk Management:** Assess and manage the security risks associated with third-party vendors and partners.
- **Security Metrics and Reporting:** Measure and report on cybersecurity performance and compliance to stakeholders and senior management.
- **Cybersecurity Culture:** Foster a culture of security throughout the organization, promoting vigilance and responsibility among all employees.

Q3. Explain the role of the computer emergency response team, the emergency response team for data security mechanisms.

- **Computer Emergency Response Team (CERT):** CERTs are responsible for responding to and mitigating cybersecurity incidents within an organization. Their roles include:
    - **Incident Response:** Rapidly assess and respond to security incidents, minimizing damage.
    - **Threat Intelligence:** Collect and analyze information on emerging threats and vulnerabilities.
    - **Security Awareness:** Educate employees about security best practices.

- **Vulnerability Management:** Identify and address software vulnerabilities.
- **Security Monitoring:** Continuously monitor networks for unusual activities.



- **Emergency Response Team for Data Security Mechanisms:** This team focuses specifically on safeguarding sensitive data within an organization. Their responsibilities include:
  - **Data Classification:** Categorize data based on sensitivity and importance.
  - **Encryption:** Implement encryption mechanisms to protect data at rest and in transit.
  - **Access Controls:** Enforce strict access controls and permissions to limit data access.
  - **Data Loss Prevention (DLP):** Deploy DLP solutions to prevent unauthorized data leakage.
  - **Data Backup and Recovery:** Ensure data is regularly backed up and can be recovered in case of loss or breach.
  - **Data Retention Policies:** Define and enforce policies for data retention and disposal.

These teams work collaboratively to secure an organization's digital assets and respond effectively to security incidents.

Q4. What approach can you take to defend the phishing attempts.

Defending against phishing attempts requires a multi-faceted approach:

- **User Training and Awareness:** Educate employees about the dangers of phishing and how to recognize suspicious emails, links, and attachments.
- **Email Filtering:** Implement robust email filtering solutions that can detect and quarantine phishing emails.
- **Multi-Factor Authentication (MFA):** Require MFA for accessing sensitive systems and data to prevent unauthorized access even if credentials are stolen.
- **URL Filtering:** Use URL filtering tools to block access to known phishing websites.
- **Regular Updates:** Keep operating systems, software, and antivirus tools up to date with the latest security patches.
- **Phishing Simulations:** Conduct phishing simulations to test and train employees on their ability to identify phishing attempts.
- **Strong Password Policies:** Enforce strong password policies, including regular password changes.
- **Incident Response Plan:** Develop an incident response plan that outlines steps to take in case of a successful phishing attack.
- **Email Authentication Protocols:** Implement email authentication protocols like DMARC, DKIM, and SPF to verify email sender authenticity.
- **Security Software:** Use endpoint security software to detect and prevent phishing attempts at the user's device.

Q5. Mention the list of challenges for a successful deployment & monitoring the web intrusion detection.

Deploying and monitoring web intrusion detection systems can be challenging due to:

- **Complexity:** Configuring and maintaining intrusion detection systems can be complex, requiring expertise and resources.
- **False Positives**: IDS systems may generate false alarms, leading to wasted time investigating non-existent threats.
- **Tuning:** Fine-tuning the IDS to reduce false positives while not missing real threats is a delicate balance.
- **Traffic Volume:** High volumes of network traffic can overwhelm IDS systems, impacting their effectiveness.
- **Encryption:** Encrypted traffic can evade traditional IDS, making it challenging to detect threats within encrypted payloads.
- **Zero-Day Attacks:** IDS may not detect new, unknown threats until signatures or detection rules are updated.
- **Alert Fatigue:** Frequent alerts can lead to alert fatigue, causing analysts to overlook genuine threats.
- **Resource Constraints:** Smaller organizations may lack the resources to deploy and maintain sophisticated IDS solutions.
- **Integration**: Integrating IDS with other security tools and workflows can be challenging.
- **Evolving Threats:** Attack techniques are constantly evolving, requiring regular updates and adaptations to detection rules.

Assignment No. 02

Q1. Explain data validation and why is it important for a secure software application?

Data validation

It is the practice of checking the integrity, accuracy and structure of data before it is used for a business operation. Data validation operation results can provide data used for data analytics, business intelligence or training a machine learning model. It can also be used to ensure the integrity of data for financial accounting or regulatory compliance. Each type of data validation is designed to make sure the data meets the requirements to be useful.

Data validation is related to data quality. Data validation can be a component to measure data quality, which ensures that a given data set is supplied with information sources that are of the highest quality, authoritative and accurate. Data validation is also used as part of application workflows, including spell checking and rules for strong password creation.

Why validate data?

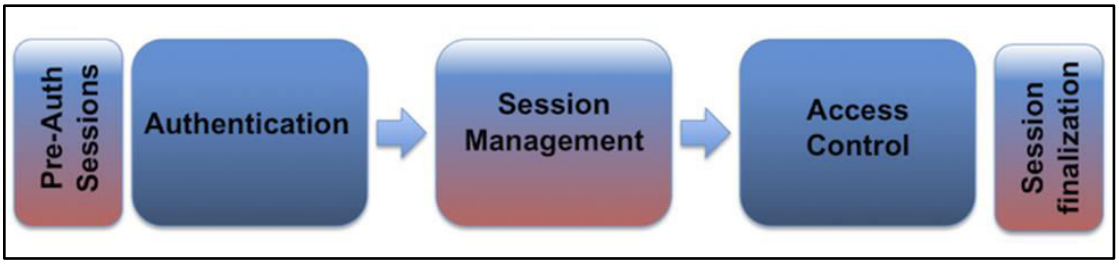
For data scientists, data analysts and others working with data, validating it is very important. The output of any given system can only be as good as the data the operation is based on. Validating the data ensures that the data is accurate, which means all systems relying on a validated given data set will be as well.

Data validation is also important for data to be useful for an organization or for a specific application operation. For example, if data is not in the right format to be consumed by a system, then the data can't be used easily, if at all. As data moves from one location to another, different needs for the data arise based on the context for how the data is being used. Data validation ensures that the data is correct for specific contexts. The right type of data validation makes the data useful.

Q2. Explain session management in web applications.

Session management manages sessions between the web application and the users. The communication between a web browser and a website is usually done over HTTP or HTTPS. When a user visits a website, a session is made containing multiple requests and responses over HTTP.

According to RFC (Request for Comments), HTTP is a stateless protocol. In this process, each request and response is independent of other web processes. Session management capabilities linked to authentication, access, control, and authorization are commonly available in a web application.



Modern web applications require maintaining multiple sessions of different users over a time frame in case of numerous requests. Regarding security, session management relates to securing and managing multiple users' sessions against their request. In most cases, a session is initiated when a user supplies an authentication such as a password. A web application makes use of a session after a user has supplied the authentication key or password. Based on the authentication, the user is then provisioned to access specific resources on the application.

Having many points of attack related to a web session or a large attack surface can compromise web applications and sessions in many different ways. Below are some of the best practices for implementing session management. Implementing these practices will reduce the attack surface and minimize the risk and damage caused by vulnerabilities and attacks resulting from improper session management.

1. Setting secure HTTP flags on cookies: Avoid sending sensitive traffic over unencrypted channels, i.e. HTTP. Setup the secure flag, which will ensure that data is transmitted over encrypted protocols such as HTTPS. The HTTP flag should only be set on session cookies to prevent session hijacking, which can be caused due to client-side javascript execution.

2. Generation of new session cookies: New session token generation should be ensured at every step of the authentication and interaction process, i.e. when a user visits an application or website and when the user gets authenticated. Apart from this, a new session should be created when a user exits from the application. Cookies should have an expiration time. In this way, if an account is inactive for a long time the session will expire.

3. Session cookies configuration: Session tokens should not be easily guessable, they should be long, unique and unpredictable. Doing so will decrease the chances of an attacker being successful in using brute force to figure out the session token. The expiration time of persistent cookies should be no longer than 30 minutes, so that attacks such as session fixation can be prevented.

Q3. Describe buffer overflow attack and explain the mechanisms to handle it.

Buffer overflow is a software coding error or vulnerability that can be exploited by hackers to gain unauthorized access to corporate systems. It is one of the best-known software security vulnerabilities yet remains fairly common. This is partly because buffer overflows can occur in various ways and the techniques used to prevent them are often error-prone.

A buffer overflow attack takes place when an attacker manipulates the coding error to carry out malicious actions and compromise the affected system. The attacker alters the application's execution path and overwrites elements of its memory, which amends the program's execution path to damage existing files or expose data.

A buffer overflow attack typically involves violating programming languages and overwriting the bounds of the buffers they exist on. Most buffer overflows are caused by the combination of manipulating memory and mistaken assumptions around the composition or size of data.

A buffer overflow vulnerability will typically occur when code:

1. Is reliant on external data to control its behavior
2. Is dependent on data properties that are enforced beyond its immediate scope
3. Is so complex that programmers are not able to predict its behavior accurately

There are several types of buffer overflow attacks that attackers use to exploit organizations' systems. The most common are:

1. Stack-based buffer overflows: The stack-based approach occurs when an attacker sends data containing malicious code to an application, which stores the data in a stack buffer.

2. Heap-based buffer overflows: A heap-based attack is more difficult to carry out than the stack-based approach. It involves the attack flooding a program's memory space beyond the memory it uses for current runtime operations.

3. Format string attack: A format string exploit takes place when an application processes input data as a command or does not validate input data effectively. This enables the attacker to execute code, read data in the stack, or cause segmentation faults in the application.

One of the most common methods for preventing buffer overflows is avoiding standard library functions that have not been bounds-checked, which includes gets, scanf, and strcpy. Another common method is to prevent buffer overruns by using bounds-checking that is enforced at runtime. This automatically checks that the data written to a buffer is within the appropriate boundaries.

Modern operating systems now deploy runtime protection that enables additional security against buffer overflows. This includes common protection like:

1. Address space layout randomization (ASLR): Buffer overflow attacks typically need to know where executable code is located. ASLR moves at random around locations of data regions to randomize address spaces, which makes overflow attacks almost impossible.

2. Data execution prevention: This method prevents an attacker from being able to run code in non-executable regions by flagging areas of memory as executable or non-executable.

3. Structured exception handling overwrite protection (SEHOP): Attackers may look to overwrite the structured exception handling (SEH), which is a built-in system that manages hardware and software exceptions. They do this through a stack-based overflow attack to overwrite the exception registration record, which is stored on the program's stack.

Implementing security measures around development code and operating systems is not enough to protect organizations' systems. When a buffer overflow vulnerability is discovered, it is crucial to quickly patch the software and ensure it is made available to all users.

Q4. Explain some tools which can be used to check the vulnerability of an application.

The method of recognizing, categorizing and characterizing the security holes (called as Vulnerabilities) among the network infrastructure, computers, hardware system, and software, etc. is known as Vulnerability Analysis.

Vulnerability scanning tools allow for the detection of vulnerabilities in applications using many ways. Code analysis vulnerability tools analyze coding bugs. Audit vulnerability tools can find well-known rootkits, backdoor, and trojans.

There are many vulnerability scanners available in the market. They can be free, paid, or open-source. Most of the free and open-source tools are available on GitHub. Deciding which tool to use depends on a few factors such as vulnerability type, budget, frequency of how often the tool is updated, etc.

Here are the list of the best vulnerability scanning tools:

1. Invicti: Invicti is a dead accurate automated scanner that will identify vulnerabilities such as SQL Injection and Cross-site Scripting in web applications and web APIs. Invicti uniquely verifies the identified vulnerabilities proving they are real and not false positives.

2. Acunetix: Acunetix is a fully automated web vulnerability scanner that detects and reports on over 4500 web application vulnerabilities including all variants of SQL Injection and XSS. The Acunetix crawler fully supports HTML5 and JavaScript and Single-page applications, allowing auditing of complex, authenticated applications.

3. Intruder: Intruder is a proactive vulnerability scanner that scans you as soon as new vulnerabilities are released. In addition, it has over 10,000 historic security checks, including for WannaCry, Heartbleed, and SQL Injection.

4. SolarWinds: SolarWinds provides Network Vulnerability Detection with its Network Configuration Manager. Its network automation capabilities will rapidly deploy firmware updates to network devices. It has functionalities for monitoring, managing, and protecting network configurations. The tool will simplify and improve network compliance.

Q5. How are hashing algorithms termed as secure for data? Highlight some of the methods to achieve it.

Secure Hash Algorithms, also known as SHA, are a family of cryptographic functions designed to keep data secured. It works by transforming the data using a hash function, an algorithm that consists of bitwise operations, modular additions, and compression functions. The hash function then produces a fixed-size string that looks nothing like the original.

A common application of SHA is to encrypt passwords, as the server side only needs to keep track of a specific user's hash value, rather than the actual password. This is helpful in case an attacker hacks the database, as they will only find the hashed functions and not the actual passwords, so if they were to input the hashed value as a password, the hash function will convert it into another string and subsequently deny access.

SHA-1: SHA-1 works by feeding a message as a bit string of length less than 2^{64} bits, and producing a 160-bit hash value known as a message digest. There are two methods to encrypt messages using SHA-1. Although one of the methods saves the processing of sixty-four 32-bit words, it is more complex and time-consuming to execute. At the end of the execution, the algorithm outputs blocks of 16 words, where each word is made up of 16 bits, for a total of 256 bits.

Although SHA-1 is still widely used, cryptanalysts in 2005 were able to find vulnerabilities on this algorithm that detrimentally compromised its security. These vulnerabilities came in the form of an algorithm that speedily finds collisions with different inputs, meaning that two distinct inputs map to the same digest.

SHA-2: Due to the exposed vulnerabilities of SHA-1, cryptographers modified the algorithm to produce SHA-2, which consists of not one but two hash functions known as SHA-256 and SHA-512, using 32- and 64-bit words, respectively. There are additional truncated versions of these hash functions, known as SHA-224, SHA-384, SHA-512/224, and SHA-512/256, which can be used for either part of the algorithm.

SHA-1 and SHA-2 differ in several ways mainly, SHA-2 produces 224 or 256-sized digests, whereas SHA-1 produces a 160-bit digest. SHA-2 can also have block sizes that contain 1024 bits, or 512 bits, like SHA-1.

Brute force attacks on SHA-2 are not as effective as they are against SHA-1. A brute force search for finding a message that corresponds to a given digest of length LL using brute force would require 2^{L} evaluations, which makes SHA-2 a lot safer against these kinds of attacks.