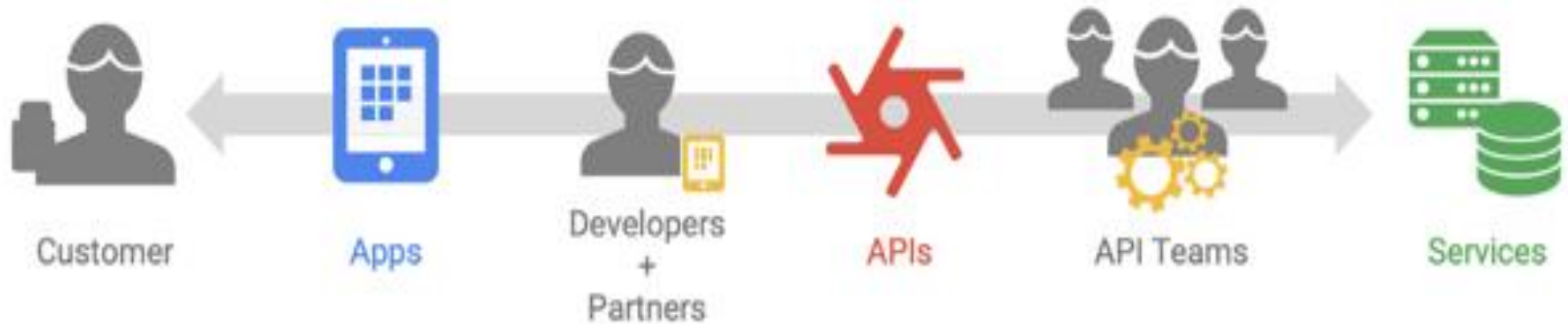




Enterprise Security

Security Landscape



Threat Protection

Behavior Based
Signature Based
Payload Complexity
Spikes
OWASP (SQL Injection, input validation, etc.)

Access Controls

OAuth2
API Keys
Products
Scopes
Quota/Spike Arrest
Logging

Self Service & SSO

IAM Integration
Prov. & DeComm
OpenId Connect
JWT
SAML

Security Governance

Global Policies
RBAC management
Data Masking
Compliance:
ISO, PCI-DSS, HIPAA,
SOC1&2

Data Security

TLS
Two-way TLS
IP Access Control
Encrypted Data Store
and Cache

Application Security Threats



Broken access control -

Impact - Inefficient access controls can lead to unauthorized information disclosure, data modification, destruction of all data, etc.

Fix - Avoid insecure Direct Object Reference, limit CORS, rate-limit APIs, invalidate sessionids after logout or use short-lived JWT tokens for stateless.



Insecure Design

Examples - unprotected storage of credentials, generation of error messages containing sensitive info

Impact - Vulnerable/compromised enterprise

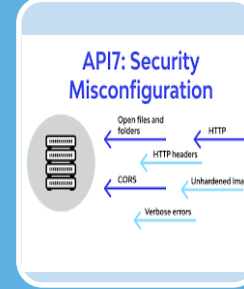
Fix – Use credential vaults, don't display verbose error messages, penetration testing, continuous vulnerability scanning



Cryptographic Failures

Impact - Leads to exposure of sensitive data

Fix - Classify data, apply security controls as per data classification, encryption at rest and in transit for sensitive data using up-to-date strong standard algorithms/protocols, enforce HSTS, disable caching for response that contain sensitive data.

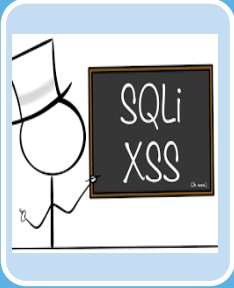


Security Misconfiguration

Impact - This might impact any layer of the application stack, cloud or network, causing costly data breaches.

Example – Files and directories unprotected, unpublished URLs not blocked from ordinary users, directory listing allowed

Fix – Have a repeatable hardening process in place – regular scans (including SAST, DAST), review access controls, hide directory listings

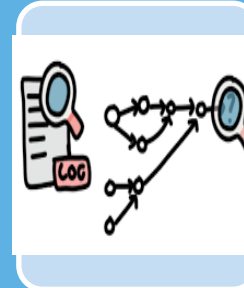


Injection

Examples - CSS, SQL/NoSQL Injection, CSRF

Impact - may result in data loss, data integrity loss, data theft, service denial, and total system compromise

Fix - Automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs is strongly encouraged. SAST, DAST tools should be incorporated in CI/CD pipeline



Security Logging and Monitoring failures


Impact – Due to improper logging, it becomes difficult to dig deeper into actual threats and take precautionary steps for future problems

Fix – Log failed logins, archive logs for a certain period in proper format for log management solutions to look into, ensure monitoring systems alert in real-time

Coding Best Practices

- Code minification and obfuscation : Making your code harder to access, and by extension harder to read
- Avoid shortcuts: hardcoded credentials and security tokens
- Secure coding practices: input validation, parameterized queries, encoding to prevent common attacks such as SQL injection, cross-site scripting (XSS), and command injection.
- Libraries/frameworks up to date
- Strong authentication and authorization mechanisms: Implement strong authentication and authorization mechanisms
- Encryption to protect sensitive data at rest and in transit.
- Implement logging and monitoring.
- Educate developers and users about security best practices, such as password hygiene, social engineering, and phishing attacks.
- Avoid components with known vulnerabilities : open-source components and libraries
- Threat modelling - Design Review
- Traceability Matrix
- Streamlined release management team
- Auditing & logging
- Streamlined CI/CD process

Application Data Security- Best Practices

- **Encryption** is managed through standard encryption mechanisms such as AES-256. Data is automatically encrypted when written to storage and decrypted when read from storage.
- **Access Control** – is managed using role-based access.
- **PII** – Do not store any PII
- **Transport Layer Security** – SSL encryption is used for encrypting data in motion and protects the confidentiality of data. Sensitive data will be transmitted to and from the cloud via a secured VPN.
- **Data at Rest** - Sensitive data at rest will be encrypted and stored.
- The platform uses cloud **API gateway for security**. Key aspects that are taken care of include prevention of DoS (Denial of service) attacks and prevention of unauthorized access using the API gateway access controls.
- All APIs are secured via microservices Gateway using **JWT**
- All the application requests will pass through a **cloud-based firewall** which will protect the application against the common web exploits that may compromise security
- **Rules in the cloud-based firewall address** issues like **OWASP Top 10 vulnerabilities** 
- All the calls through Cloud will be transported through **Secured Sockets Layer which encrypts the data in transit**
- All the calls to the **on-premise systems will be transported through Secured Virtual Private Network** to ensure data protection
- Integrations with on-prem systems through **REST APIs or HTTP URL while ensuring none of the Personally Identifiable Information or PII is stored. It only uses it for the time taken to process the transaction**

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
10. Server-Side Request Forgery

API Security - Best Practices

API Security:

1. Authentication, Authorization:

- Avoid basic authentication, instead consider using API keys, OAuth, or OpenID

2. Request & Response Validation:

- Validate everything that is sent and received - Request Body, Content, Request Headers, Input Parameters, Character length

3. Use HTTPS for APIs, and TLS for application:

- HTTPS traffic can only be decrypted by the owner of the HTTPS certificate, unlike HTTP which can be readable by everyone over internet
- TLS encrypts any data in transit between the client and the server, preventing unauthorized third parties from hijacking or modifying the message along the way, avoiding man-in-the-middle attacks.

4. Follow access-control checklist:

- Implement zero-trust security model
- Implement API Rate Limiting - restrict number of requests a given user or IP address can send over a certain period of time, number of requests your API can process at any given time, API returning error message if limit exceeds or sending alarm to dev team

5. Avoid excessive data exposure:

- Don't leave login credentials in URL, build and version of your server on error screens
- Adopt OpenAPI and RAML standards to limit the exposure of excessive data

6. API Security Testing:

- Conduct regular performance tests, penetration tests.

7. Traffic management:

- Use WAF and API Gateways to guard access to underlying API data. Least-access security rules should be implemented using appropriate whitelisting/blacklisting from specific and narrow IP ranges.

8. Secure Coding practices:

- Use Encoding to handle untrusted data transfer from the API to the web application - HTML encoding, JavaScript encoding, AttributeValue encoding depending on your output
- Maintain mapping between external and internal identifiers to avoid exposing internal IDs like databaseIDs, customerIDs etc. Instead convert them to GUIDs/UUIDs

9. Vulnerability scans for APIs:

- Use static code scanning tools pre-emptively catch and refactor any vulnerable methods in your code before committing to code repos.
- Use dependency scans to scan and make sure that you are not using any compromised tools, libraries, or frameworks in your code that can end up being used as backdoors or attack vectors.
- Always try to use the most current, and long-term-supported version of the dependency

API Security - Best Practices

Key Areas	Description
Avoid Basic Authentication	Using API keys, OAuth, or OpenID as much safer substitutes for a standard combination of a login and password
Implement Two-Factor Authentication	API consumer needs to verify their identity in two distinct ways before they can access their account
Enforce Transport Layer Security	encrypts any data in transit between the client and the server, preventing unauthorized third parties from hijacking or modifying the message along the way
Access Control	Zero-Trust Security Model - should be trusted unless they've been properly authenticated and authorized
	API Rate Limiting -
	The number of requests a given user or IP address can send over a certain period of time
	The number of requests your API can process at any given time
	Any additional fees related to sending new API calls once the limit has been exceeded
	The way an API reacts once any of the rate limits have been reached - from redirecting the user to an error page to triggering an alarm to the development and security teams
	Excessive data exposure is an OWASP vulnerability
	Common Areas :
	Error pages
	URL strings
	API responses
	Data in transit
	Data at rest
Address Excessive Data Exposure Issues	
	OpenAPI and RAML standards to limit the exposure of excessive data
Input & Output	Always Validate User Input - unauthorized access
	Enforce HTTP Methods - API consumers should only be able to use the GET method
	Conduct API Fuzz Input Testing - Feed API large amounts of typically malformed data in an attempt to expose vulnerabilities within your API
	Test for SQL Injections - manipulating database queries to achieve a malicious outcome
	Restrict Parameter Tampering - manipulating URL parameters or form field data to get unauthorized access
API Security Testing	Conduct Functional Tests
	Carry out Performance Tests
	Execute a Penetration Test
	Run a Vulnerability Scan

Access Management Best Practices

- **User Roles & Access Control**

- Permission matrix for user or role-based access.
- Role based access will be implemented. Each role will have access to functionality that is suitable to that role.
- Privileged access users will be provisioned with admin like access only on absolute need basis.
- Shared credentials will not be provisioned, and only specific user-based accounts will be used to access any of the environments from a auditing perspective.
- User access to any of the environments will be provided using the standard industry practices. The user-role and role-permissions relationships make it simple to perform user assignments since users no longer need to be managed individually, but instead have privileges that conform to the permissions assigned to their role(s). This will avoid the common pitfalls associated with shared accounts, privilege user accounts and one user having all the access to the IT infrastructure.

- **Auditing**

- Necessary auditing features will be built in to check and track actions taken by users.
- Standard DB Auditing capability (provided by the database software) will be leveraged to monitor database activities being performed on the production databases.
- The platform maintains audit of user login information/ access logs. The audit logs can be used for further analysis on unauthorized accesses or suspicious behavior and system traffic.
- The API gateway monitoring & logging allows the platform to collect and monitor traffic management and analyze the data metrics for further strengthening the security measures.

Data Storage Vulnerabilities



Lack of encryption

- Many products in the market lack inherent encryption capabilities, requiring the installation of separate encryption software to secure such data.
- Some high-end NAS and SAN devices include automatic encryption - AWS EBS, Azure Storage, NetApp OnTap



Cloud Storage

- While some may argue that cloud storage is more secure than on-premises storage, the adoption of cloud technology introduces a new layer of complexity to storage environments.
- Cloud ops and dev teams should take care of setting up cloud RBACs and appropriate BCDR features.



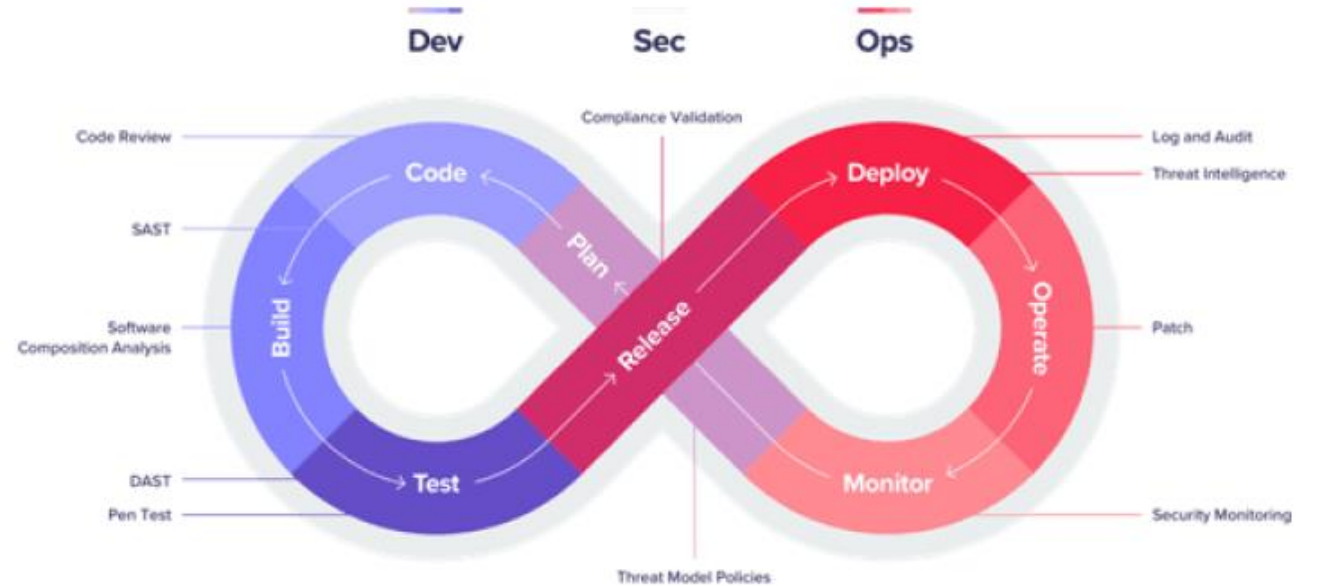
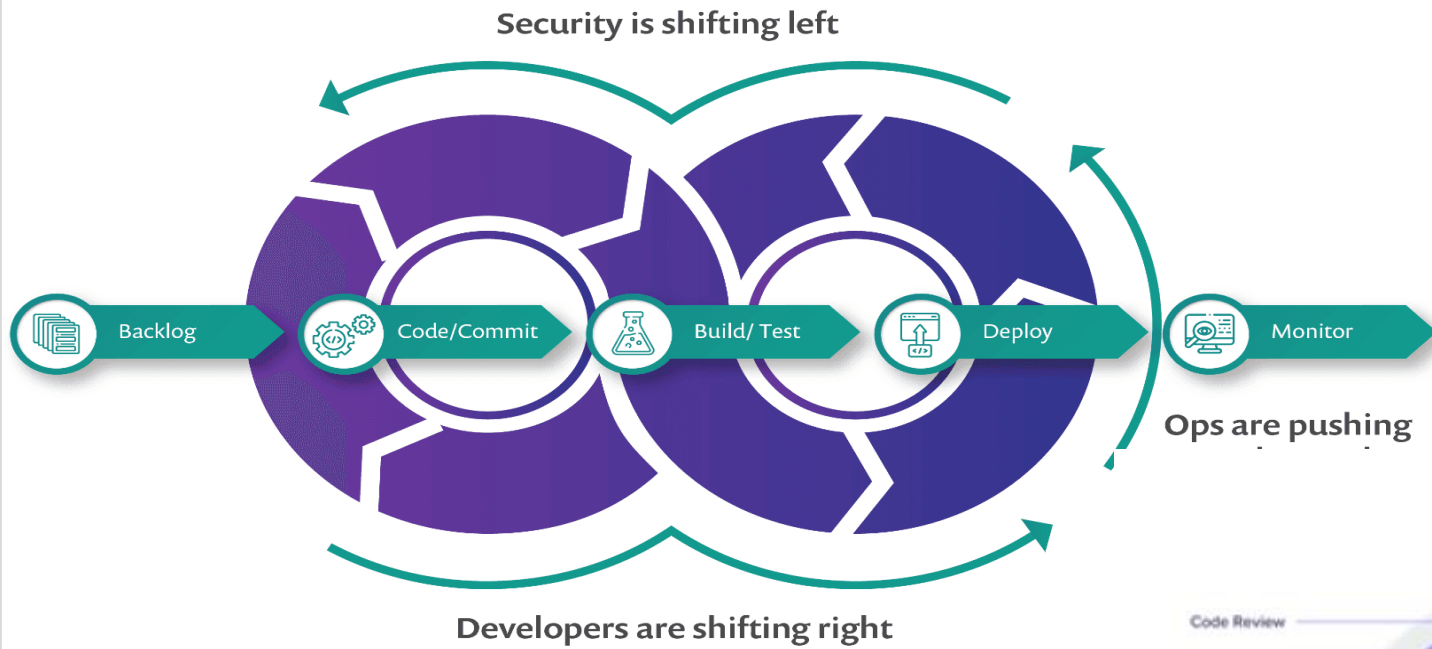
Incomplete Data Deconstruction

- Deleting data from a hard drive or other storage media may not completely erase it, as residual traces can potentially be exploited by unauthorized individuals to recover sensitive information.
- Storage admins should take necessary steps to overwrite the deleted data to make it unrecoverable (with caution).

Storage Security Best Practices

- Develop and implement data storage security policies that cover the appropriate levels of security for different types of data, including access control and encryption measures.
- Use role-based access control (RBAC) to control access to storage systems and data and enforce the use of strong passwords. Administrators should also be sure to change any default passwords on their storage devices and to enforce the use of strong passwords by users.
- Encrypt data both while in transit and at rest in storage systems and use secure key management systems to track encryption keys.
- Deploy data loss prevention (DLP) solutions to identify and prevent attacks in progress.
- Implement strong network security systems, such as firewalls, anti-malware protection, security gateways, intrusion detection systems, and advanced analytics and machine learning-based security solutions.
- Implement appropriate security measures on endpoint devices, including PCs, smartphones, and tablets, to prevent attacks.
- Use redundant storage technologies, such as RAID, to improve availability, performance, and security.
- Implement backup and recovery systems and processes that are adequate for disaster recovery purposes and have the same level of data security in place as primary systems.
- Regularly update and patch storage systems and software to mitigate known vulnerabilities.
- Provide regular security training for storage administrators and users to help them recognize and prevent security threats.

DevSecOps



GitHub User Specific Guidelines

- GitHub ID should be the same as your employee ID and Incture email ID. Please modify your GitHub accounts and include your employee id in the name/profile section so we can identify and validate your accounts
- Cases where Keyword “Incture” in the username and kept our customer’s name keyword in their public repository name, kindly remove any such repository immediately.
- Strong Password guidelines
- Incture repositories should not be accessed with your personal GitHub ID’s. Incture code/data should not be copied to your personal GitHub accounts.
- Do not share your Incture GitHub credential with anyone including your colleagues or friends.
- GitHub repositories will be created only by the authorised team, based on the request raised by project managers.
- Project manager/assigned member will be the administrator to manage, control and access privileges to the respective repository.
- Each project will have only one repository created and administrators can use the branch options to create sub repositories as required
- Employees are strictly advised not to create public repositories in GitHub as it may lead to compromising data security
- Restrict access only to those who require it by assigning permissions to specific users/teams. Use GitHub's built-in access control features, such as teams and collaborators, to manage permissions effectively.
- Always access GIT over HTTPS instead of HTTP to encrypt transit data from your system to GIT's servers
- Never store credentials and sensitive data on GitHub

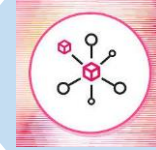
GitHub Admin Specific Guidelines

- Add a Security.md file : Security policy for your repository. The purpose of this file is to officially document processes and procedures relating to security.
- Rotate SSH keys and Personal Access Tokens
- Take regular backups of your code and data to recover later in case of any security breaches
- Enable git branch protection
 - Use branch protection rules to prevent unintended or unauthorized changes to vital branches in your repositories - restrict direct pushes, require code reviews
- Add sensitive files to .gitignore
- Employ a “secrets vault” service
- Manually check for vulnerabilities – GitHub Enterprise gives us this option. Get the information

Network Security Components/Measures



Firewalls monitor incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules.



Network segmentation divides up your network traffic into different classifications, making it easier to enforce different security policies based on endpoint identity.



IDS monitors your network for suspicious or malicious activity by scanning network traffic and sends results directly to admin or to a SIEM.



DLP policies ensure that users don't send sensitive or critical information outside the corporate network and help a network administrator control what data end users can transfer, replicate, etc.



IPS continuously monitors your network, blocks any incoming attacks before they can be executed and captures info about such incidences



The best antimalware programs not only scan for malware upon entry, but also continuously track files afterward to find anomalies, remove malware, and fix damage.



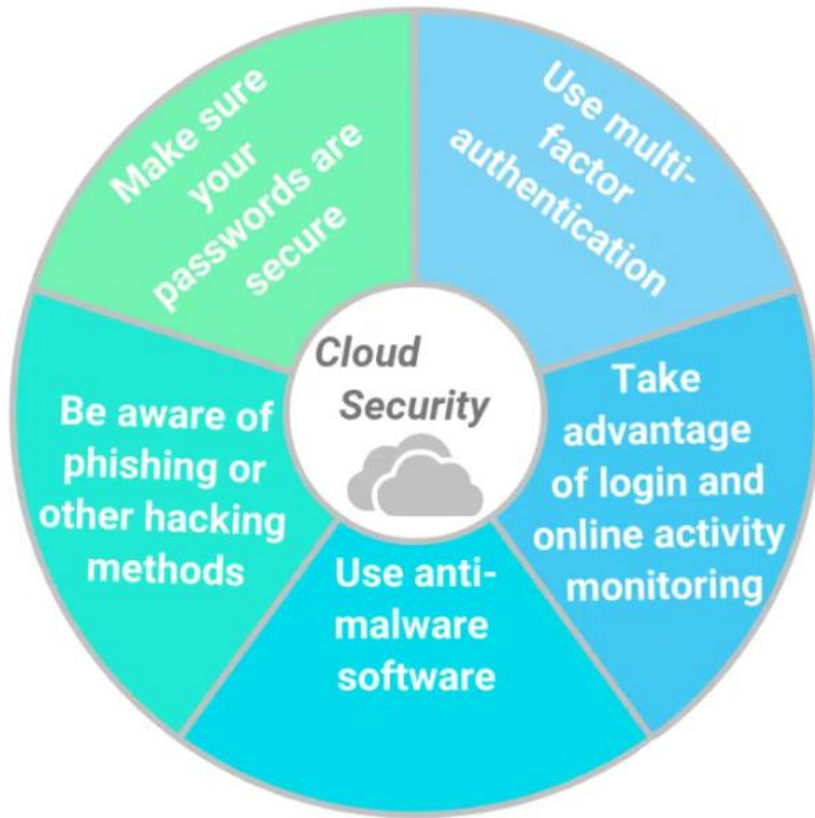
VPN works by encrypting your data and routing your connection through a private server. Consequently, your activity and your privacy is protected online.



Network Access Control enforces who has approved access and block unidentified users or devices from having unrestricted, or any, network access.

Network Security - Cloud

Its not the Cloud itself but the way it is used that causes security issues!!



- **MFA** using OTP to mobile or email, biometrics, answer a secret question
- **Login and online activity monitoring** to ensure security of sensitive data and prevent unauthorized access
- **Anti-malware solutions** - cloud-based endpoint security solution that provides advanced threat protection including anti-malware protection and behavioral analytics to detect and prevent attacks. Eg - MS Defender, AWS GuardDuty
- **Phishing or hacking methods** - Spear phishing, social engineering, URL spoofing, Credential Stuffing techniques, etc
- **Secure passwords with encryption** using HSM-as-a-service

Network Security Best Practices

- Implement a strong password policy - Ensure that all users have strong passwords that are changed periodically, and consider implementing multi-factor authentication for added security.
- Keep software and hardware up-to-date - Regularly apply patches and updates to your network hardware and software to ensure that vulnerabilities are addressed.
- Use firewalls and intrusion prevention systems - These technologies help prevent unauthorized access to your network and can detect and prevent network-based attacks.
- Implement access controls - Restrict access to sensitive data and network resources to only those who need it, and monitor access logs to detect suspicious activity.
- Encrypt sensitive data - Use encryption to protect sensitive data both in transit and at rest.
- Enable two-factor authentication: Two-factor authentication adds an extra layer of security by requiring a second form of authentication, such as a code sent to a mobile device, in addition to a password.
- Use VPNs: Use virtual private networks (VPNs) for remote access to the network. VPNs encrypt traffic, making it more difficult for attackers to intercept data.
- Educate employees - Train employees on best practices for network security, including how to recognize and avoid phishing scams and other social engineering attacks.
- Perform regular security audits - Conduct regular security audits to identify vulnerabilities and ensure that your security measures are up-to-date and effective.

Types of Testing

SAST - Static Application Security Testing is a type of security testing that is performed on the source code of an application or software. SAST tools analyze the code to identify potential security vulnerabilities, such as buffer overflows, SQL injection, cross-site scripting (XSS), and other common security weaknesses. SAST is usually automated, and it is performed during the development phase to identify security flaws before the application is deployed. This helps to prevent security issues that could be exploited by attackers and can potentially save organizations significant time and resources in the long run.

Famous SAST Tools -

Fortify: Developed by Micro Focus, Fortify is a popular SAST tool that can detect security vulnerabilities in a wide range of programming languages, including Java, C++, and .NET.

Checkmarx: Checkmarx is another widely used SAST tool that can scan source code for potential vulnerabilities in multiple languages, including Java, C++, Python, and others.

SonarQube: SonarQube is an open-source SAST tool that can analyze code for security issues, code quality, and other related issues. It supports multiple languages and has a large community of users and contributors.

Veracode: Veracode is a cloud-based SAST tool that can detect security flaws in code written in multiple languages, including Java, .NET, PHP, and others.

Coverity: Coverity is an SAST tool developed by Synopsys that can analyze code for security issues, code quality, and other related issues. It supports multiple languages and integrates with a wide range of development tools.

Types of Testing Contd.

DAST - Dynamic Application Security Testing, is a type of security testing that is performed on a running web application. DAST tools interact with the application to simulate attacks and identify potential security vulnerabilities that may exist in the application's functionality, configuration, or environment. DAST tools typically work by sending requests to the web application and analyzing the responses to identify security flaws such as cross-site scripting (XSS), SQL injection, and other vulnerabilities that could be exploited by attackers. DAST testing can help to identify potential security issues that may not be identified through SAST testing or manual code review.

Famous DAST Tools -

OWASP ZAP: Developed by the Open Web Application Security Project (OWASP), ZAP is a free, open-source DAST tool that can detect security vulnerabilities in web applications. It is platform-independent and can be used on Windows, macOS, and Linux.

Burp Suite: Burp Suite is a commercial DAST tool that is widely used by security professionals. It can detect security vulnerabilities in web applications, including SQL injection, cross-site scripting (XSS), and others. Burp Suite is available in both free and paid versions.

Acunetix: Acunetix is a commercial DAST tool that can detect vulnerabilities in web applications, including SQL injection, cross-site scripting (XSS), and others. It supports multiple programming languages and can be used on Windows, macOS, and Linux.

Qualys Web Application Scanning: Qualys Web Application Scanning is a cloud-based DAST tool that can detect security vulnerabilities in web applications. It is designed to be scalable and can support large-scale web applications

Types of Testing Contd.

Penetration testing (also known as pen testing) is a cybersecurity testing method that involves **simulating a real-world attack** on a computer system, network, or application to identify potential vulnerabilities and weaknesses. The goal of a penetration test is to identify security flaws that could be exploited by an attacker and to provide recommendations for addressing these vulnerabilities. Penetration testing can involve a range of methods, including automated tools and manual testing techniques

Famous Pen Testing Tools -

Metasploit: Metasploit is a widely used penetration testing tool that allows testers to simulate attacks on systems and applications, identify vulnerabilities, and test the effectiveness of security controls.

Nmap: Nmap is a network scanning tool that can be used to identify hosts and services on a network, as well as detect open ports and potential vulnerabilities.

OpenVAS: OpenVAS is an open-source vulnerability scanner that can be used to identify vulnerabilities in systems and applications, as well as provide remediation recommendations.

Types of Testing Contd.

Vulnerability scanning is a cybersecurity testing method that involves scanning a computer system, network, or application for potential security vulnerabilities or weaknesses. The goal of vulnerability scanning is to identify potential security flaws that could be exploited by attackers and to provide recommendations for remediation.

- **Network vulnerability scanning:** This involves scanning a network for potential vulnerabilities, such as open ports, outdated software, or misconfigured settings.
- **Application vulnerability scanning:** This involves scanning web applications for potential vulnerabilities, such as SQL injection or cross-site scripting (XSS) attacks.
- **Host vulnerability scanning:** This involves scanning individual hosts or devices for potential vulnerabilities, such as outdated software or weak passwords.
- **Wireless network vulnerability scanning:** This involves scanning wireless networks for potential vulnerabilities, such as weak encryption or rogue access points

Famous vulnerability Testing Tools -

Nessus: Nessus is a popular vulnerability scanner that can be used to identify potential security vulnerabilities and misconfigurations in systems and applications.

Retina: Retina is a vulnerability scanner that can be used to identify potential security vulnerabilities and misconfigurations in systems and applications

Qualys: Qualys is a cloud-based vulnerability management and assessment tool that can be used to identify vulnerabilities in systems and applications.

Types of Testing Contd.

IAST ,Interactive Application Security Testing, is a type of security testing that combines elements of both SAST and DAST. IAST tools are designed to analyze the running application to identify potential security vulnerabilities by examining how the application behaves in real-time. IAST tools typically work by instrumenting the application at runtime, monitoring its behavior, and identifying potential security vulnerabilities in real-time. This allows IAST tools to identify vulnerabilities such as SQL injection, cross-site scripting (XSS), and others

Famous DAST Tools -

Contrast Security: Contrast Security is a commercial IAST tool that can detect security vulnerabilities in web applications. It supports multiple programming languages and frameworks, including Java, .NET, and Node.js. Contrast Security integrates with a wide range of development and security tools and provides real-time feedback to developers.

R2C: R2C is an open-source IAST tool that can detect security vulnerabilities in web applications. It supports multiple programming languages, including Java, Python, and JavaScript. R2C is designed to be easy to use and can be integrated into the software development lifecycle.

Code Dx: Code Dx is a commercial IAST tool that can detect security vulnerabilities in web applications. It supports multiple programming languages and frameworks, including Java, .NET, and Ruby on Rails. Code Dx provides a centralized view of the security vulnerabilities detected by multiple security tools, including IAST, SAST, and DAST.

Training Material

Link to Enterprise Security presentation. -  [Enterprise Security 2023.pptx](#)(Network, Storage, Application & GIT Hub Security)

Link to the recording for Enterprise Security sessions - <https://people.zoho.com/incture/training#lms-view/course/502642000031304313/module>

Assessment links for Enterprise Security sessions - <https://people.zoho.com/incture/training#lms-view/course/502642000031304313/module>

Link to ISMS Presentation -  [ISMS 2023.pptx](#)

Link to the recording for ISMS - <https://people.zoho.com/incture/training#lms-view/course/502642000015405227/module>

Assessment links for ISMS - <https://people.zoho.com/incture/training#lms-view/course/502642000015405227/module>

About Incture

Incture is one of the leading providers of digital applications, products, and digital engineering solutions on SAP® Business Technology Platform. Cherrywork® is a comprehensive extensions suite for SAP customers delivering packaged business value with agility and at scale to address evolving business requirements.

Incture, Cherrywork and other Cherrywork applications are registered trademarks of Incture Technologies Private Limited.

