Cybersecurity Masterclass: Frameworks, Risks, and Engineering

- To provide a deep understanding of core cybersecurity frameworks (NIST CSF 2.0, NIST SP 800-160) and the most critical application risks (OWASP Top 10)

- To enable the applications of these concepts through practical, scenario-based problem-solving in the Digital Era.

Strategic Risk Management with NIST Cybersecurity Framework (CSF) 2.0

❑ Overview and The Evolution to CSF 2.0

- The NIST CSF is a voluntary, risk-based framework for managing cybersecurity risk. Its strength lies in its ability to be adopted by any organization—regardless of size, sector, or technical complexity—by focusing on *what* needs to be achieved, not *how* to achieve it.

- **Key Update in CSF 2.0:** The addition of the **Govern (GV)** function, elevating cybersecurity from a technical task to an organizational risk management priority aligned with business strategy.

Strategic Risk Management with NIST Cybersecurity Framework (CSF) 2.0

❑The Six Core Functions: Lifecycle of Security

- The framework describes the cybersecurity risk management lifecycle through six functions. Each function contains categories and subcategories (Specific outcomes).

1. Govern (GV) – The Foundation

- **Purpose:** Establishes the organizational strategy, expectations, and policy to manage cybersecurity risk. It's the "tone at the top."

- **Key Categories:**
  - **Organizational Context:** Understanding the mission, assets, and regulatory landscape.
  - **Risk Management Strategy (GV.RM):** Defining the organization's risk tolerance and how risk decisions are made.
  - **Oversight:** Monitoring and reviewing the effectiveness of the cybersecurity program.

Example (GV.RM): A FinTech startup determines its **risk tolerance** is **low** for data breaches but **moderate** for service downtime. This decision (Govern) mandates that its **Protect (PR)** controls must prioritize data encryption, while **Recover (RC)** plans must focus on rapid system failovers.

2. Identify (ID) – The Understanding

- **Purpose:** Develops an understanding of cybersecurity risk to systems, assets, data, and capabilities. "You can't protect what you don't know you have."

- **Key Categories:** Asset Management, Risk Assessment, Supply Chain Risk Management.

Example (ID.AM): An organization discovers an unpatched server through an automated **Asset Management (ID.AM)** tool. This asset was unknown to the IT team but was running a critical legacy application. The discovery process (Identify) immediately informs a risk assessment and triggers patching (Protect).

3. Protect (PR) – The Safeguards

- **Purpose:** Develops and implements safeguards to ensure the delivery of critical services.

- **Key Categories:** Identity and Access Management (IAM), Data Security, Protective Technology.

Example (PR.AC): To address **Access Control (PR.AC)**, a bank implements **Multi-Factor Authentication (MFA)** for all remote access and uses a **principle of least privilege** policy to ensure tellers only access the specific customer data needed for their daily tasks.

4. Detect (DE) – The Discovery

- **Purpose:** Develops and implements activities to identify the occurrence of a cybersecurity event.

- **Key Categories:** Anomalies and Events, Security Continuous Monitoring.

Example (DE.CM): A **Security Continuous Monitoring (DE.CM)** tool flags a user account accessing 1,000 documents within a minute, which is 10 times the historical average (an anomaly). The system doesn't stop it (that's Protect), but it alerts the security team (Detect).

5. Respond (RS) – The Action

- **Purpose:** Develops and implements activities regarding a detected cybersecurity event.

- **Key Categories:** Incident Management, Mitigation, Communication.

Example (RS.MI): Following the anomaly alert (from Detect), the security team immediately isolates the compromised user's workstation from the network (**Mitigation - RS.MI**) and begins forensic collection, following established **Incident Management (RS.IM)** procedures.

6. **Recover (RC) – The Resilience**

- **Purpose:** Develops and implements activities to maintain plans for resilience and to restore any capabilities or services impaired.

- **Key Categories:** Recovery Planning (RC.RP), Improvements (RC.IP).

Example (RC.RP): After a successful restoration from backup following a disaster, the team holds a "lessons learned" session (**Improvements - RC.IP**) to determine that the backup rotation was too slow, leading to a new **Recovery Planning (RC.RP)** requirement for more frequent snapshots.

Scenario Questions for CSF 2.0

❑ **Govern Function Scenario**:

A company has recently merged with a smaller firm, significantly increasing its IT assets and third-party vendor relationships. Under the new **Govern (GV)** function, what is the *immediate* priority the governance team should address, and what component of the GV category would guide this action? (Focus on **Organizational Context** and integrating the new assets into the existing **Risk Management Strategy**).

Scenario Questions for CSF 2.0 …,

- **Identify and Protect Scenario**:

During an internal **Identify (ID)** function assessment, your team discovers a critical database containing sensitive customer information is accessible to several non-essential employees. Which two core functions are primarily involved in resolving this, and what is the specific control category for the safeguard needed? (Focus on **ID: Risk Assessment** leading to **PR: Access Control (PR.AC)**).

Scenario Questions for CSF 2.0…,

- **Detect and Respond Scenario**:

A new malware variant bypasses your existing antivirus software and is detected by a **Security Continuous Monitoring (DE.CM)** tool. Your team quickly isolates the affected systems. Which function guides the actions taken after isolation, and what specific activities would be performed under that function? (Focus on **Respond (RS): Analysis (RS.AN)** and **Mitigation (RS.MI)**).

OWASP Top 10

- **Overview of the OWASP Top 10**

The OWASP Top 10 is the definitive standard for the most critical security risks facing web applications. It is not an exhaustive list of all vulnerabilities but focuses on the most common and impactful security problems. The 2021 update introduced new categories focusing on design and logging.

OWASP Top 10

A01: Broken Access Control (BAC)

- **The Risk:** Failing to enforce restrictions on what users are allowed to do. Attackers exploit these flaws to bypass authorization, accessing, modifying, or deleting sensitive data.

Example (Insecure Direct Object Reference - IDOR): A banking application uses a parameter like account_id=12345 in the URL. If a user changes this to account_id=98765 and can see the balance of another customer, that is a BAC failure.

- **Mitigation Strategy: "Deny by Default."** Every request must be checked against an authorization matrix. Utilize centralized access control mechanisms and enforce the **Principle of Least Privilege (PoLP)**.

OWASP Top 10

A02: Cryptographic Failures

- **The Risk:** Sensitive data is exposed during transmission or storage due to lack of encryption, poor encryption, or improper key management.

Example: A user registration form transmits passwords over HTTP (unencrypted) instead of HTTPS (encrypted). Even if the password is encrypted in the database (at rest), it is exposed in transit via a Man-in-the-Middle attack.

- **Mitigation Strategy: Enforce HTTPS/TLS 1.2+** for all data in transit. Classify data sensitivity and encrypt all sensitive data **at rest** using strong, validated algorithms (e.g., AES-256) with proper key management. Never store plaintext passwords (use strong, salted, and iterated hashing like Argon2 or bcrypt).

OWASP Top 10

A03: Injection (SQL, NoSQL, OS Command)

- **The Risk:** Unvalidated user input is executed as a command or query by an interpreter (e.g., database, operating system shell, or XML parser).

Example (SQL Injection): An attacker enters ' OR 1=1 -- into a login prompt's password field. If the code is not protected, the resulting query structure changes, allowing the attacker to bypass authentication.

- **Mitigation Strategy: Parametrized Queries (Prepared Statements).** This separates the SQL command structure from the user-supplied data, ensuring the input is always treated as data, not executable code. Use input validation (whitelisting) and output encoding.

OWASP Top 10

A04: Insecure Design

- **The Risk:** A new category focusing on flaws related to missing or ineffective control design. This is about *how* the application is built, not just coding errors.

Example: A developer uses a microservice architecture but fails to implement a **rate-limiting** mechanism on the login service. This design flaw allows an attacker to conduct a high-speed brute-force attack without being blocked.

- **Mitigation Strategy:** Adopt a **secure development life cycle (SDLC)**. Implement **Threat Modeling** at the design stage to proactively identify and mitigate design flaws *before* any code is written.

OWASP Top 10

A05: Security Misconfiguration

- **The Risk:** Leaving default settings, enabling unnecessary features, using verbose error messages, or failing to properly harden servers and cloud environments.

Example: A production server still has the default administrator account enabled, or detailed stack trace error messages are displayed to the user, revealing internal server version numbers and file paths.

- **Mitigation Strategy: Harden all components:** servers, databases, and application containers. Implement automated configuration management and patch management tools. Disable all unnecessary ports, services, and default accounts.

Questions for OWASP Top 10

- **A01: Broken Access Control Scenario**:

A logged-in user changes the numerical ID in the URL path from /user_profile/123 to /user_profile/124 and is able to view another user's private data. Which specific OWASP vulnerability is this, and what "deny by default" principle defense mechanism was likely missing? (Focus on **A01: BAC / IDOR**and the missing server-side check that **User X is authorized to view User Y's data**).

Questions for OWASP Top 10

- **A03: Injection Scenario**: A web application allows users to search for products using a text field. An attacker enters '; DROP TABLE products; -- into the search field, but the query fails because the input was run through a function that escaped single quotes. However, the attacker later discovers they can use an HTML tag in the search field that is reflected on the page, displaying a pop-up. Which two distinct OWASP Injection vulnerabilities are being tested, and what is the difference in their defense mechanisms? (Focus on **A03: SQL Injection** and **Cross-Site Scripting (XSS)**. SQLi uses **Prepared Statements**, XSS uses **Output Encoding**).

Questions for OWASP Top 10

- **A04: Insecure Design Scenario**: A company designs a payment processing system where the server-side code trusts a client-side (browser) total price calculation during checkout to save database lookups. An attacker intercepts the client-side request and changes the calculated price from TZS 100 to TZS 1. Which OWASP risk does this fall under, and what secure design principle was violated? (Focus on **A04: Insecure Design** and the principle of **Never Trust the Client** or **Performing all critical logic on the server**).

Questions for OWASP Top 10

- **A06: Vulnerable Components Scenario**: Your team is preparing to deploy an application that relies on an open-source library. A recent security scan flagged this library as containing a vulnerability from three years ago that was fixed in a newer version. Which OWASP risk is this, and what is the primary mitigation strategy your team should immediately execute? (Focus on **A06: Vulnerable and Outdated Components** and the need for a **Software Composition Analysis (SCA)** tool and immediate patching/upgrading).

Questions for OWASP Top 10

- **A09: Logging and Monitoring Scenario**: A security event is detected: an unauthorized account login. Your security team struggles to determine the source IP, the time of the first failure, and the successful authentication method because the application only logs successful logins without detail. Which OWASP vulnerability is this, and what is the specific improvement required to the application's code and configuration to mitigate this? (Focus on **A09: Security Logging and Monitoring Failures** and the need to log **failures, inputs, and all relevant security context**).

Security Engineering with NIST SP 800-160

❑Overview: Security as a Design Property

The **NIST Special Publication (SP) 800-160 Vol. 1, Systems Security Engineering (SSE)**, emphasizes that security must be an **integral part of the system design and development process**, not an afterthought (a "bolt-on"). It adopts a formal, engineering-based approach to ensure systems are **trustworthy**.

Security Engineering with NIST SP 800-160

❑ **The Systems Security Engineering (SSE) Approach**

Security is an Engineering Discipline

- **The Principle:** Just like performance, reliability, and usability, security must be **quantifiable, verifiable, and designed-in**.

- **Goal:** Moving from simple **compliance checklists** to building truly **resilient and trustworthy systems**.

Security Engineering with NIST SP 800-160

- **The Three Interconnected Contexts**

The framework provides structure by focusing on the purpose, the solution, and the assurance.

- **Problem Context (The WHAT): Define the problem from the stakeholder's perspective.**
  - **Activity:** Define **Stakeholder Protection Needs** (What must be protected?), perform **Threat Analysis** (Who is attacking and how?), and **Vulnerability Assessment**.
  - **Example:** For a new medical device, the Problem Context defines the core need as **"Protecting patient safety and data privacy."** This translates into specific security requirements, such as **integrity controls** over firmware updates and **confidentiality controls** over transmitted health records.

Security Engineering with NIST SP 800-160

Solution Context (The HOW): Design and implement the security solution.

- **Activity:** Develop the **Security Architecture and Design** (system structure), select and tailor **Security Controls** (often from NIST 800-53), and perform **Verification and Validation**.

- **Example:** Based on the requirements from the Problem Context, the Solution Context team designs a hardware security module (HSM) to store cryptographic keys, implementing the **Cryptographic Protection** controls required to meet the data confidentiality need.

Security Engineering with NIST SP 800-160

Trustworthiness Context (The PROOF): Provide evidence and assurance that the system is secure.

- **Activity:** Security **Risk Assessment**, Security **Authorization (A&A)**, and **Continuous Monitoring**.

- **Vivid Example:** Before deployment, the system must undergo a formal **Security Authorization** review. The Trustworthiness Context requires the engineering team to provide extensive documentation and test results (**evidence**) proving that the HSM implementation (Solution) successfully meets the patient data privacy requirements (Problem).

Security Engineering with NIST SP 800-160

Integration into the SDLC

- The key tenet of SSE is that these contexts are applied **early and continuously**. Security must be considered in the **requirements gathering** phase, not just the **testing** phase.

Scenario Questions for NIST SP 800-160

- **Early Integration Scenario**:

An organization is starting a project for a new cloud-based application. The development team wants to begin coding immediately. Based on the NIST SSE framework, what is the *first* critical step that should be performed before development, and under which of the three contexts does this activity fall? (Focus on **Problem Context** - Defining **Stakeholder Protection Needs** and performing **Threat Modeling**).

Scenario Questions for NIST SP 800-160

Multidisciplinary Scenario:

During a design review, the lead developer proposes using a non-standard, custom-built cryptographic algorithm for data encryption to save on licensing costs. The security architect argues against this, citing a lack of trustworthiness assurance. Which core principle of NIST SSE supports the security architect's position, and which context would provide the evidence to resolve this conflict? (Focus on the principle of **Multidisciplinary Collaboration** and the **Trustworthiness Context** for required validation/assurance).

Scenario Questions for NIST SP 800-160

Risk Management Scenario:

A continuous monitoring system (Trustworthiness Context) flags a potential vulnerability in a newly deployed system component. According to the SSE framework, what systematic process must be triggered to address this and ensure the system remains adequately secure? (Focus on the continuous loop of **Risk Assessment -> Control Implementation (Solution) -> Monitoring (Trustworthiness)**).

Scenario Questions for NIST SP 800-160

Traceability and Assurance Scenario:

A stakeholder requests proof that all high-priority security requirements, defined in the Problem Context, have been addressed in the final implemented system. What concept, central to the Solution and Trustworthiness Contexts, must be established and maintained to provide this proof? (Focus on **Traceability** - documenting the link between requirements, design choices, and test results).

Scenario Questions for NIST SP 800-160

Protection Needs Scenario:

An organization is designing a system that handles public, non-sensitive data, but its **Availability** is highly critical to the company's core mission (e.g., a public status page). In the Problem Context, what should the Systems Security Engineer prioritize when defining **Protection Needs**, and how does this contrast with a system handling highly sensitive but less critical data? (Focus on the three goals of security: **Confidentiality, Integrity, and Availability (CIA)**. For the status page, **Availability** is prioritized, mandating **DDoS protection** and **fault tolerance** in the design, while a sensitive system would prioritize **Confidentiality**).

Key Takeaways

- **NIST CSF 2.0:** Provides the **strategic and tactical framework** for managing security risk as a business objective. Govern is the new core.

- **OWASP Top 10:** Provides the **specific, actionable risks** that developers must defend against in web applications. **Insecure Design** requires proactive threat modeling.

- **NIST SP 800-160:** Provides the **engineering methodology** to build security **into** systems from the very beginning (Problem, Solution, Trustworthiness).

Thank You very Much

“Always be vigilant.

Trust nobody—always verify.”

Eng. SHILIBA, G