Day 7

Mapping AI Systems to Standards & Frameworks:

NIST CSF 2.0 — Applied to AI Systems

1. Identify → Asset Management of AI models
2. Protect → Model hardening, data protection, access controls
3. Detect → Monitoring model attacks, logs, anomalies
4. Respond → Incident response for AI attacks
5. Recover → Model rollback, integrity checks

Detail understanding of each …

1. IDENTIFY:

The IDENTIFY function is the foundation of NIST CSF. If you can't identify the AI assets, you can't protect them.

Identifying All AI-Related Assets:

If you don't know what assets exist, you can't detect vulnerabilities or prevent misuse. You must identify everything involved in AI operations:

- **Models** (LLMs, classifiers, recommender systems, OCR, etc.)
- **Datasets** (training, validation, inference data)
- **Pipelines** (training pipeline, deployment pipeline, CI/CD)
- **APIs** (AI inference APIs, data APIs)
- **Prompts** (system prompts, templates, prompt libraries)
- **Embeddings** (vector representations stored in vector DBs)

AI Asset Classification and Tagging:

Each AI asset should be labelled using tags such as:

- **Model type** (LLM, vision, NLP)
- **Sensitivity level** (low/medium/high risk)
- **Data type used** (personal data, internal data, public data)
- **Business criticality** (critical, important, optional)

Tagging helps prioritize security, compliance, and monitoring efforts.

AI System Inventory Creation:

This inventory becomes the single source of truth for audits and governance.Create a central inventory (like CMDB) that lists:

- All models
- Version numbers
- Datasets used
- Deployment locations
- Business owners
- Risk level
- Regulatory requirements
- Dependencies (APIs, cloud platforms, vector DBs)

AI Data Lineage (Data Origin Tracking):

Data lineage shows data flow, helping detect privacy risks and poisoning risks. Data lineage answers:

- Where did the training data come from?
- How was it processed?
- What inference data does the model consume?

Model Provenance Tracking:

Provenance is critical for trust, reproducibility, and audits. Provenance = the history of the model. You must track:

- Who trained it (engineers, vendor)
- When it was trained
- On what data
- Using which tools/frameworks
- What hyperparameters or training configs were used
- Which version is currently deployed

Identifying High-Risk AI Assets:

High-risk models require extra governance, monitoring, and approval.

Identifying Third-Party / Vended AI Systems

Many organizations use: OpenAI API, Google Gemini, AWS Bedrock etc. For each external system, identify:

- Data-sharing policies
- Privacy/legal obligations
- Dependency risks
- Security guarantees
- Licensing/copyright restrictions

Third-party AI adds supply chain risk.

AI Dependency Mapping:

AI systems depend on many external components, such as LLM Providers, Vector databases, APIs, Plugins etc. Mapping dependencies helps detect:

- Single points of failure
- Weak security links
- Regulatory exposure (cross-border data flow

Business Context Mapping: Understanding context ensures proper risk prioritization.

Example:
A chatbot is low risk.
An AI credit scoring system is high risk.

Regulatory Mapping for Each Model

Different AI systems fall under different regulations:

- **Privacy laws** → GDPR, DPDP Act
- **AI laws** → EU AI Act
- **Sector rules** → healthcare, banking, insurance
- **Copyright laws** → data sourcing & training data obligations

For each model, document:

- Which regulation applies
- Which compliance controls are required
- What documentation/evidence must be stored

Basically,

- IDENTIFY means knowing all your AI assets: models, data, prompts, APIs, embeddings.
- Track their classification, risk level, and dependencies.
- Maintain an AI inventory with provenance, lineage, and ownership.
- Identify high-risk models, third-party AI, and regulatory requirements.
- Understanding the AI environment is the foundation for secure and compliant AI.

2. PROTECT:

The PROTECT function focuses on putting defenses and safeguards around AI systems. This includes model hardening, data protection, access control, supply-chain security, and secure development practices.

Model Hardening Techniques: Model hardening means making the model difficult to attack or misuse.

a. Adversarial Defense: Protects against attacks where the attacker gives slightly modified inputs to fool the model.
   Techniques:
   - Adversarial training (train model with adversarial samples)
   - Gradient masking
   - Input sanitization
   - Defensive distillation

b. Prompt Injection Defense (For LLMs): Prevents attackers from manipulating prompts to override system instructions.
   Techniques:
   - Strong system prompts
   - Output constraints
   - Input validation
   - Context separation (no mixing user prompt with system rules)
   - Guardrail models / prompt firewalls

c. Input/Output Filtering: Validating input and output before sharing them to the model, check outputs before returning them to user, Block harmful instructions, PII leakage, SQL commands, jailbreak prompts

d. Rate limiting: limits requests per second/user/API key.

Data Protection for AI Pipelines:

AI pipelines deal with training data, evaluation data, and inference data. Each stage needs strong protection. This can be achieved by the following ways:

1. Encryption of Training/Inference Data

- Encrypt data at rest (storage)
- Encrypt data in transit (network)
- Especially important when using cloud-based AI models

2. Secure Dataset Storage

- Store datasets in secure buckets (S3 with IAM policies)
- Logs on every dataset access
- Version control for datasets
- Detect unauthorized modifications (poisoning attempts)

3. Differential Privacy

Adds controlled noise to protect personal data in training sets.

Used to:

- Hide individual user information
- Reduce risk of model memorization
- Support privacy laws (GDPR/DPDP)

4. Secure Feature Engineering

- Validate input features
- Remove PII where unnecessary
- Detect abnormal features (poisoned data)
- Maintain secure feature pipelines

Access Controls for AI Systems: Strong access controls prevent unauthorized usage or model theft. Segmentation reduces blast radius if something goes wrong.

1. RBAC for Models

Define roles and permissions:

- Who can train models?
- Who can fine-tune?
- Who can deploy?
- Who can query the model?

2. Privileged Access Management

Reduce risk of insider misuse by:

- Limiting admin accounts
- Using just-in-time access
- Monitoring privileged actions
- Rotating secrets and credentials

3. Model Access Segmentation

Different models have different risk levels.

- Public-facing chatbots → low privileges
- Internal financial models → restricted
- Highly sensitive models → restricted to a few engineers

Supply Chain Security for AI: AI uses many external sources: datasets, libraries, base models, cloud APIs. These must be checked for security.

1. Verifying Datasets

- Check data source legitimacy
- Ensure no copyrighted/scraped illegal data
- Validate data integrity (hash checks)

2. Verifying Libraries

- Pin dependency versions
- Avoid untrusted packages
- Use SBOM (Software Bill of Materials)

3. Verifying Model Packages

- Check signatures of downloaded models
- Ensure no backdoors or malicious layers
- Scan model metadata

4. Securing Fine-Tuning Workflows

- Validate fine-tuning data quality
- Restrict fine-tuning permissions
- Log all fine-tuning experiments
- Prevent malicious data from altering model behaviour

Secure Development Practices for AI: Security must be integrated into the ML lifecycle.

1. Threat Modeling for AI

Identify threats such as:

- Data poisoning
- Model inversion
- Adversarial attacks
- Prompt injection
- Model theft
- Unauthorized access

Threat modeling helps you design correct defenses early.

2. Secure-by-Design for ML Pipelines

Means building AI pipelines with security from day one:

- Secure data ingestion
- Identity & access controls
- Secure compute environments
- Code scanning
- Continuous monitoring
- Secure deployment (container hardening)

ML pipelines must not be treated as "just data science notebooks". They must be treated as production software.

Basically,

- PROTECT = applying defenses to secure AI models, data, and pipelines.
- Protect models using adversarial defense, prompt injection defense, input/output filters, rate limiting.
- Protect data using encryption, secure storage, differential privacy, and safe feature engineering.
- Use strong access control: RBAC, privileged access, model segmentation.
- Secure the supply chain by verifying datasets, libraries, model packages, and fine-tuning workflows.
- Apply secure development practices: threat modeling + secure-by-design ML pipelines.

--The End--