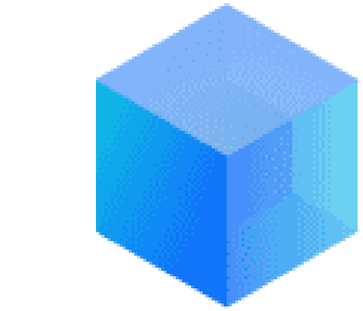
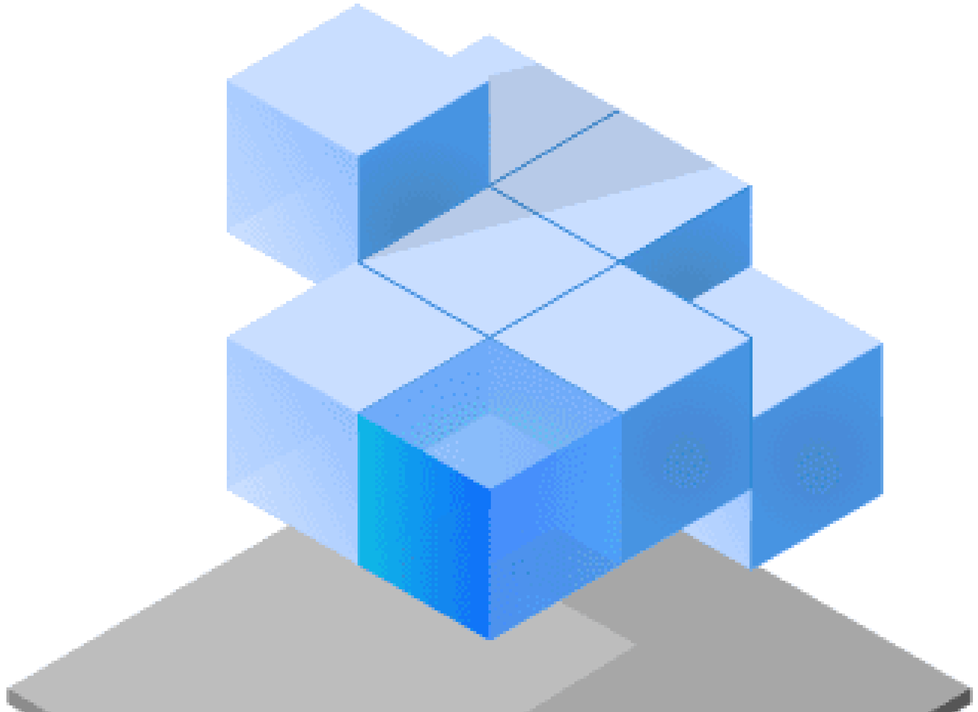
Cyber Security Challenge

Team 15





Improving Audits

Problem Statement

Enhancing the quality of Audits

Deliverables

- Design an application / tool to enhance the quality of Audits.
- Have a multi-faceted approach focusing on enriching the process of Audit for all stakeholders.

Current Auditing Limitations

Traditional methods are fragmented and lag behind in dynamic cyber environments.

Solution

All inclusive web app

User Authentication and Authorization

Dashboard and Communication Hub

Collaborative Planning and Audit Initiation

Technical Integration and Automation

Risk-Based Approach and Security Measures

Automated Report Generation

Personnel Training and Development

Regulator and Auditee Engagement

Solution

User Authentication and Authorization

01.

Multi-factor Authentication (MFA)

02.

Role-Based Access Control (RBAC)

03.

Single Sign-On (SSO)

Solution

Dashboard and Communication Hub

01.

E2E Secure Communication

02.

Document and Information Exchange

03.

Realtime Updates using WebSockets

03.

Track Ongoing Audits

04.

Visualize Key Metrics

05.

Deadline Tracking

Solution

Collaborative Planning & Audit Initiation

01.

Structured Form Submission

Allow auditees to initiate audits through a structured form for consistent information submission.

02.

Customizable Forms

Use technologies like HTML forms or dynamic form builders to provide customizable templates and guidelines.

Collective Definition

To ensure thorough and refined audit scopes.

Comment Sections

To gather feedback and refine audit scopes collaboratively.

Refinement Process

To ensure the completeness and accuracy of audit scopes.

Solution

Technical Integration and Automation

Vulnerability Scanning

Integrate with tools like Nessus for effective vulnerability scanning.

RESTful APIs

Utilize RESTful APIs to ensure smooth connectivity with other tools.

Risk Assessment

Ensure seamless integration for comprehensive risk assessment.

Routine Task Automation

Achieve efficiency by automating tasks like data collection and report generation.

User-Friendly Interface

Ensure an intuitive interface for interacting with automated processes.

Tool Utilization

Leverage Ansible, Puppet, or Chef for effective task automation.

Solution

Risk-based Approach

Integrate Risk Assessment Tools

Prioritize audit activities using tools like OpenVAS and Qualys.

Highlight Critical Assets

Identify high-risk areas within the system for focused protection.

Risk Scoring Algorithms

Implement FAIR framework for comprehensive risk scoring.

Solution

Automated Report Generation

Key Performance Indicators (KPIs)

KPIs such as total time and risk exposure will be included in the automatic reports.

Risk Identification

Reports should include identified risks for proactive mitigation.

Improvement Areas

Find weak points in the systems and suggest ways to protect better.

Solution

Personnel Training and Development

Enrollment in Workshops

- Wide array of workshops.
- Expert trainers with industry experience.
- Practical, interactive training to apply concepts.

Training Materials Repository

- Comprehensive resources from videos to e-books.
- Up-to-date with the latest industry information.
- Easy access to all the needed resources.

E-learning Platforms Integration

- Smooth transition from past usage experiences.
- Implement custom e-learning modules tailored to needs.
- Monitor and assess learning progress easily with integrated SCORM models.

Solution

Ensuring Compliance & Security

Compliance Concerns

Ensure that all compliance concerns are met with secure communication and file transfer protocols.

Safeguarding Data

Implementing secure channels and protocols is critical for safeguarding sensitive information.

Regulatory Requirements

Adhere to regulatory requirements by implementing secure communication and file transfer methods.

Reducing Vulnerabilities

Solution

Security Baseline to secure SDLC

Design	Model all the possible threats and implement mitigations for vulnerability surface area minimization.
Development	Software features are designed and developed with the benefit of aligning the product closely with user needs and requirements.
Testing	Ensures that product meets the quality standards and functions as intended.
Deployment	Make the system operational across an intended environment for real-world usage.
Maintainance	Focus on the ongoing upkeep and improvement of the system, ensuring longevity.

Solution

Phase I: Design

01.

Threat Modelling

Threat Modeling in the application design phase involves modeling potential threats and implementing mitigations to minimize vulnerability surface areas, enhancing overall security.

02.

Design Mitigations based on STRIDE Framework

Enhances application security by addressing potential threats related to Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege.

Solution

STRIDE-based Mitigations

Spoofing	Ensuring authentication integrity by verifying user identities, thereby reducing unauthorized access.
Tampering	Ensuring data accuracy and trustworthiness, reducing the risk of malicious alterations.
Repudiation	Providing accountability and traceability, deterring fraudulent activity
Information Disclosure	Protecting IP and personal data, maintaining privacy & regulatory compliance.
Denial of Service	Enhancing system reliability and uptime, ensuring continuous service delivery.
Elevation of Privilege	Minimizing the impact of breaches by limiting access to only what is necessary for a given role or task.

Solution

Phase II: Development

01.

Secure Coding Practices

02.

Data Security

03.

Web Security

04.

Authorization

05.

Network Security

06.

Third Party Codes

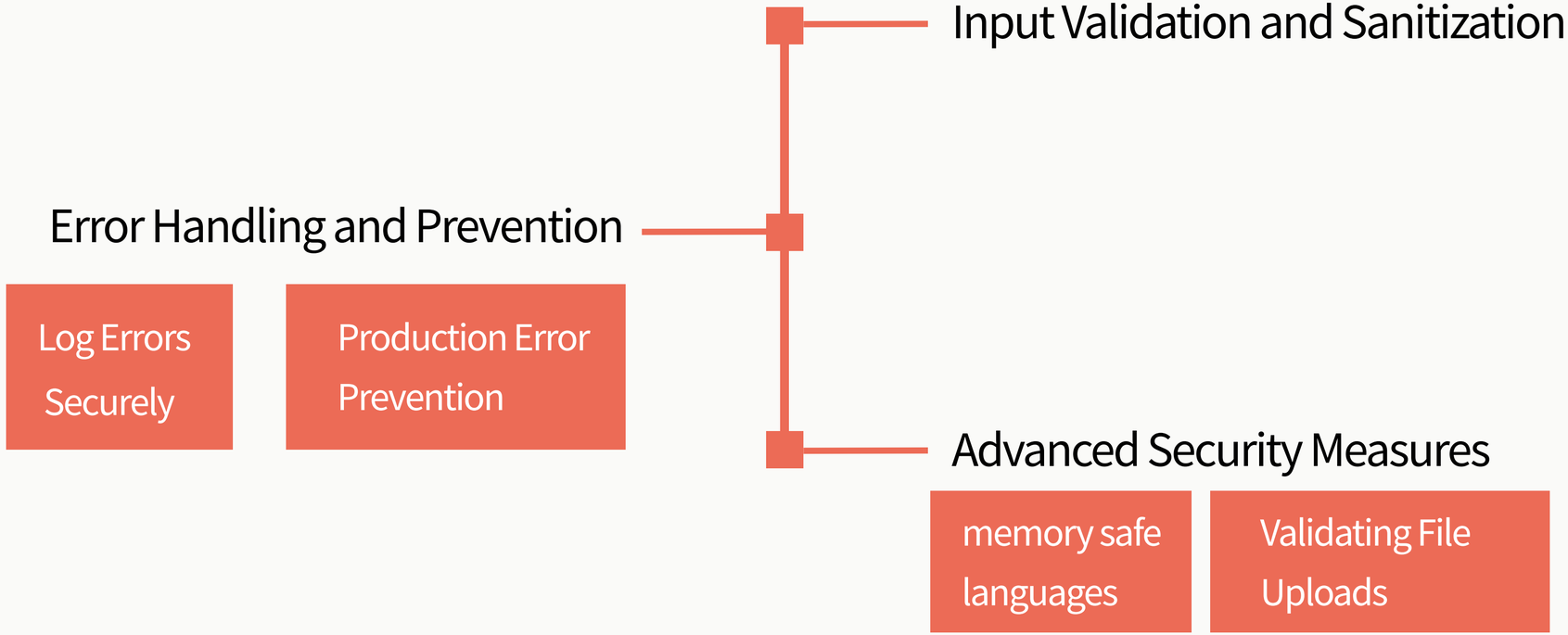
07.

Authentication

Phase - II Development

Secure Coding Practices

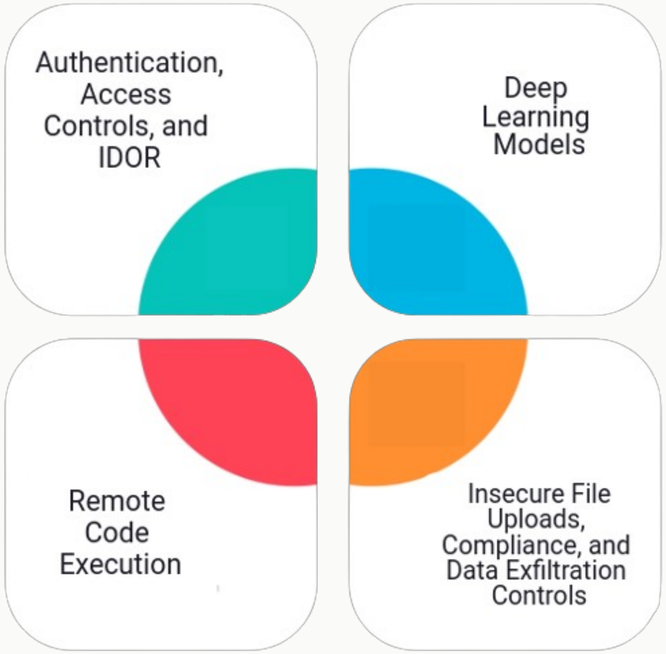
These are guidelines for writing code that is resistant to vulnerabilities, benefiting the project by reducing the risk of security breaches.



Phase - II Development

Data Security

Refers to the handling and management of data and the units that process them, which ensures efficient and secure data operations.



Phase - II Development

Network Security

Isolate Critical Components

Use firewalls and VLANs to segment and isolate critical parts of the network.

Enforce Strong Encryption

Regularly update TLS/SSL standards to enhance resilience against potential breaches.

Centralized Logging

Set up centralized logging to monitor and analyze network activity for potential security issues.

Phase - II Development

Third-Party Codes

Managing risks associated with using external code or libraries to ensure they don't introduce vulnerabilities

Operating System Patches

- Security Updates, Version Tracking
- Integrity Checks

Stay Alert

- Security Tools
- Monitoring Security Alerts

Phase - II Development

Authentication

Verify the identity of a user or process, allowing access by only authorized entities.

01.

Cookie Security

02.

Deep Packet Insepction

03.

Blockchain

Phase - II Development

Authorization

Involves defining and enforcing what an authenticated user is allowed to do.

Access Control Lists

Centralize access control decisions in your codebase and employ methods such as URL checking or inline assertions in the code.

Role-Based Control

Provide role-based access control and ownership-based access control to enhance security measures.

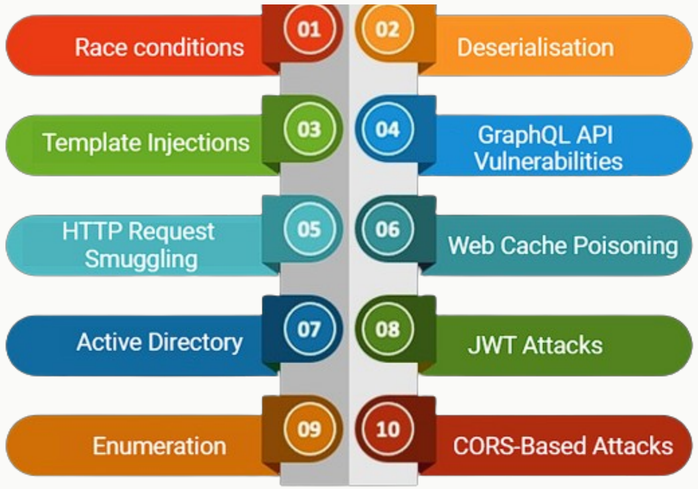
Rate-Limiting and Monitoring

Implement rate-limiting and explore the use of Deep Packet Inspection (DPI) during the authentication process.

Phase - II Development

Web Security

Various measures to protect web applications from internet-based threats, crucial for maintaining the integrity and confidentiality of transmitted data.



Solution

Phase III - Testing

Various measures to protect web applications from internet-based threats, crucial for maintaining the integrity and confidentiality of transmitted data.

Dynamic Analysis

Testing software during execution to find runtime issues, enhancing reliability and performance.

Static Analysis

Examining code without executing it to identify vulnerabilities, improving code quality and security.

Solution

Phase IV - Deployment

Secure Deployment

Create a robust security framework with firewalls, SDN for dynamic segmentation, limited administrative roles, multi-factor authentication, consent restrictions, end-to-end encryption, and controlled traffic through firewalls.

Cloud Security

Utilize AWS Config, Azure Policy, or Google Cloud Security Command Center for automated compliance monitoring, enhance threat detection with EDR solutions, mitigate data exfiltration through egress filtering, secure against web vulnerabilities with a WAF, and employ bastion hosts to minimize the attack surface.

Solution

Phase - V: Maintenance

Host & Network Security

This involves safeguarding computers and networks against unauthorized access or attacks

Secure Container Practices

Implementing security measures in containerized environments

Solution

Phase - V: Maintenance

Vendor Security (For Third-Party Services)

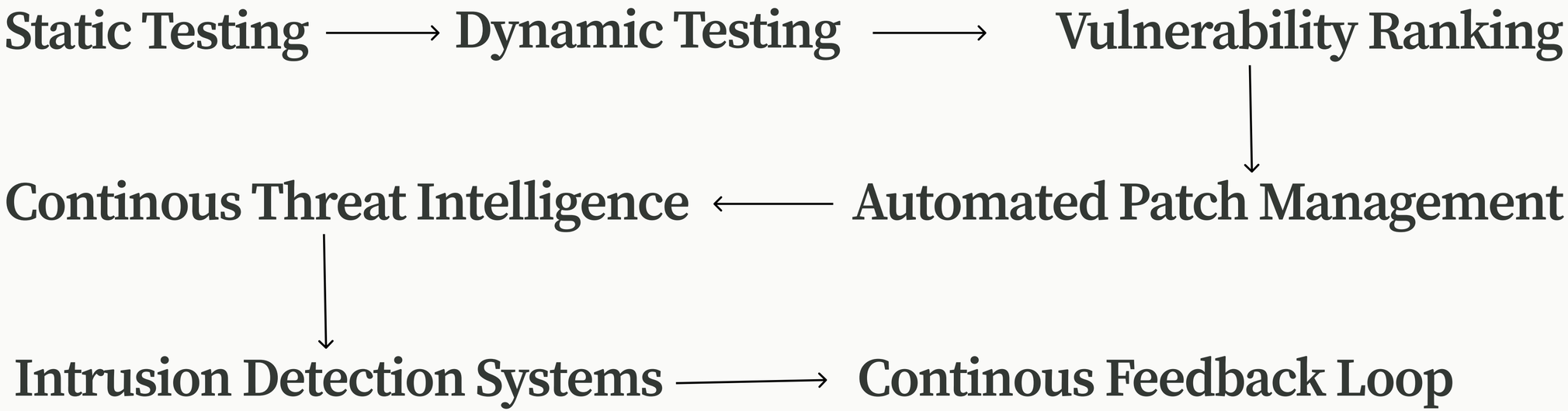
Implementing security measures for third-party services and products

Log Analysis

Systematically examining event logs to detect anomalies, aiding in timely threat detection and system health monitoring.

Solution

Automation workflow



Solution

Impact

Automated framework leads to **early detection** of vulnerabilities which **reduces the cost** needed to mitigate the risk.

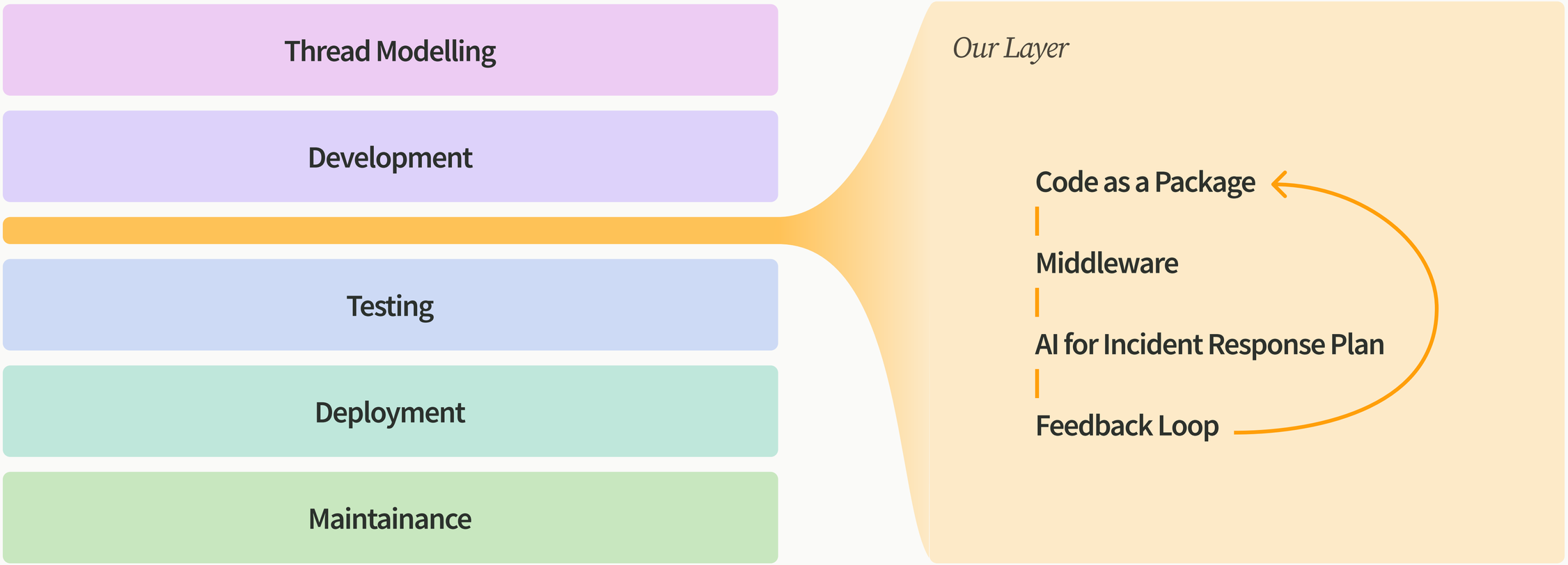
Automation **reduces human errors** and gets **seamlessly integrated** in the SDLC, hence becoming more robust and efficient.

Securing Application Development

Detecting, Reporting, and Responding to Cyber Threats

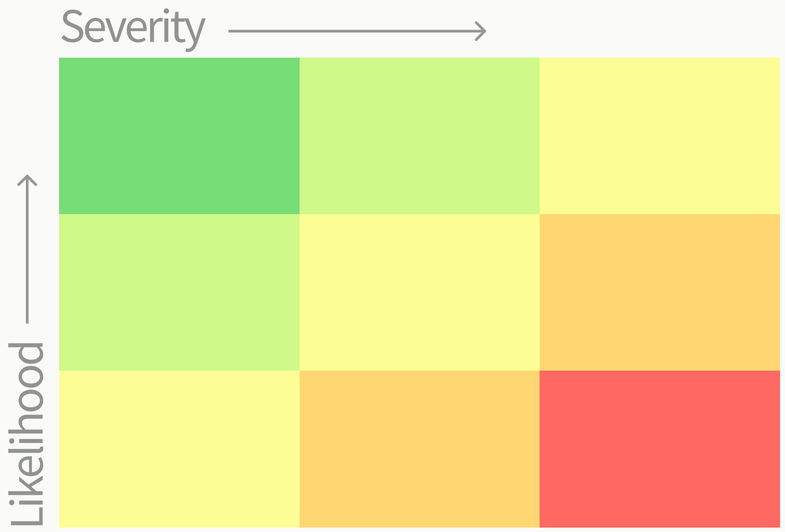
Secure layer addition

Software Development Life Cycle



Secure layer additon

Code as a Package: Risk Matrix




```python
def calculate_risk_matrix(probability, impact):
    """
    Calculate risk matrix score.

    Args:
    probability (float): Probability of the risk (0 to 1).
    impact (float): Impact of the risk (0 to 1).

    Returns:
    float: Risk matrix score.
    """
    return probability * impact

# Example
probability = 0.7
impact = 0.8
risk_score = calculate_risk_matrix(probability, impact)
print(f"Calculated Risk Score: {risk_score:.2f}")

```

Middleware

Technical Implementation of APIs

Opening a Portal ***API:***

`openPortal()`

Calculating Risk Matrix ***API:***

`calculateRiskMatrix (probability, impact)`

Reporting and Responding ***API:***

`reportAndRespond(riskScore)`

```python
@app.route('/openPortal', methods=['POST'])
def open_portal():
    user_credentials = request.json
    session_token = "Generated_Session_Token"
    return jsonify({"message": "Portal Opened Successf
session_token})
```

Middleware

Technical Implementation of APIs

Opening a Portal ***API:***

`openPortal()`

Calculating Risk Matrix ***API:***

`calculateRiskMatrix (probability, impact)`

Reporting and Responding ***API:***

`reportAndRespond(riskScore)`

```python
@app.route('/calculateRiskMatrix', methods=['POST'])
def calculate_risk_matrix_endpoint():
    data = request.json
    probability = data['probability']
    impact = data['impact']
    risk_score = calculate_risk_matrix(probability, impact)
    return jsonify({"riskScore": risk_score})
```

Middleware

Technical Implementation of APIs

Opening a Portal ***API:***

`openPortal()`

Calculating Risk Matrix ***API:***

`calculateRiskMatrix (probability, impact)`

Reporting and Responding ***API:***

`reportAndRespond(riskScore)`

```python
@app.route('/reportAndRespond', methods=['POST'])
def report_and_respond_endpoint():
    risk_score = request.json['riskScore']
    # Log
    # Alert on Portal
    return jsonify({"message": "Response Initiated", "riskSc
risk_score})
```

AI for Incident Response Plan

AI implementation pipeline for IRP

Incident Response Plan

- Preparation
- Identification & Assessment
- Containment & Mitigation
- Recovery & Lessons Learned

Features for analysis

- **Incident Type**
- **Source of Threat**
- **Severity Level**
- **User Behaviour**
- **Affected Assets**
- **Network Traffic Anomalies**

Incident Response Team

Feedback Loop

Continuous Model Enhancement

1

Feedback Loop

2

Continual Retraining

3

Model Updates

Impact

1

Time

Significant reduction in response time with our solution compared to current market solutions

2

Efficiency

The various usecases of our proposed solution

3

Intelligence

Early Threat Detection, Adaptive Nature and Return on Investment (ROI)

Conclusion

1

Redefining cybersec

Intelligent, Efficient, Dynamic

2

Rapid Response & Adaptive Security

3

Continual Retraining

Forward-thinking in design, our approach is not just for today's challenges but also a guard against tomorrow's uncertainties