Incremental Security Enforcement of Cyber-Physical Systems

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

TEAM P1-07
Aaditya Kumar Muktavarapu HU21CSEN0100580
Karthik Neriyanuri HU21CSEN0100607
Vijay Anirudh Vallabhu HU21CSEN0100611
Lalit Sai Srivatsa Narayanam HU21CSEN0101246

Under the esteemed guidance of
Dr. Sudeep K.S
Associate Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING GITAM (Deemed to be University)

HYDERABAD

October - 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM

(Deemed to be University)

DECLARATION

We, hereby, declare that the project report entitled "**Incremental Security Enforcement of Cyber-Physical Systems**" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University), submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:26th October 2024

Registration No(s).	Name(s)	Signature(s)
HU21CSEN0100580	Aaditya Kumar Muktavarapu	
HU21CSEN0100607	Karthik Neriyanuri	
HU21CSEN0100611	Vijay Anirudh Vallabhu	
HU21CSEN0101246	Lalit Sai Srivatsa Narayanam	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "**Incremental Security Enforcement of Cyber-Physical Systems**" is a bonafide record of work carried out by **Aaditya Kumar Muktavarapu HU21CSEN0100580, Karthik Neriyanuri HU21CSEN0100607, Vijay Anirudh Vallabhu HU21CSEN0100611, Lalit Sai Srivatsa Narayanam HU21CSEN0101246**
Students submitted in partial fulfillment of the requirement for the award of the Bachelor of Technology in Computer Science and Engineering degree.

Project Guide	**Project Coordinator**	**Head of the Department**
Dr. Sudeep K.S	**Dr. A B Pradeep Kumar**	**Dr.Shaik Mahaboob Basha**
Associate Professor Dept. of CSE	**Assistant Professor Dept. of CSE**	**Professor & HOD Dept. of CSE**

ACKNOWLEDGEMENT

Our project report would not have been successful without the help of several people. We would like to thank the personalities who were part of our seminar in numerous ways, those who gave us outstanding support from the birth of the seminar.

We are very much obliged to our beloved **Prof. Sheik Mahboob Basha**, Head of the Department of Computer Science & Engineering, for providing the opportunity to undertake this seminar and encouragement in the completion of this seminar.

We hereby wish to express our deep sense of gratitude to **Dr. A B Pradeep Kumar,** Project Coordinator, Department of Computer Science and Engineering, School of Technology, and to our guide, **Dr. Sudeep K.S** Associate Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the project report.

We are also thankful to all the Computer Science and Engineering department staff members who have cooperated in making our seminar a success. We would like to thank all our parents and friends who extended their help, encouragement, and moral support directly or indirectly in our seminar work.

Sincerely,

Aaditya Kumar Muktavarapu,

Karthik Neriyanuri,

Vijay Anirudh Vallabhu,

Lalit Sai Srivatsa Narayanam

TABLE OF CONTENTS

Chapter 1 Introduction

Cyber-Physical Systems (CPS) are integrated systems where computational processes control physical processes. They are widely used in sectors like industrial automation, smart grids, autonomous vehicles, and healthcare.
With increased complexity and interconnectivity, CPS have become vulnerable to cyber-physical attacks (CP-attacks), which exploit both cyber and physical components to cause disruption.
Traditional security patches are ineffective for these systems, which require real-time security updates.
This project introduces an **Incremental Security Enforcement Framework** for CPS that allows for the dynamic addition of new security policies to mitigate evolving threats.

1.1 Problem Definition

Current CPS security solutions are inadequate for managing incremental policy updates as new threats emerge. Traditional monolithic approaches to runtime enforcement (RE) suffer from state space explosion and require re-certification each time a new policy is added. There is a need for a system that allows the **incremental composition of enforcers** to adapt to new policies without compromising existing security measures.

1.2 Objective

To develop a **compositional runtime enforcement (RE) framework** that incrementally applies new security policies to CPS. This framework must prevent state space explosion, ensure modularity, and support real-time enforcement, enabling CPS to adapt to emerging threats without a full system overhaul.

1.3 Limitations

- **Initial Set-Up Complexity**: Setting up initial policies requires defining the enforcers with precision to allow smooth incremental updates.
- **Non-Compatible Policies**: Some policies may not be compatible with others in a compositional enforcement structure, requiring careful management of interdependencies.

1.4 Applications

- **Industrial Control Systems**: Protects critical systems from cyber-attacks by allowing dynamic addition of safety policies.
- **Autonomous Vehicles**: Ensures real-time compliance with safety policies that protect passengers and pedestrians.
- **Healthcare Devices**: Enhances the security of medical devices by incrementally adding policies for new health threats or operational requirements

Chapter 2: Literature Review

Our literature review on incremental security enforcement[1] for Cyber-Physical Systems (CPS) introduced us to key approaches for adapting security in response to evolving threats. We analyzed how a compositional runtime enforcement system can be structured to allow the dynamic addition of security policies without necessitating a full system re-certification—a significant advantage for maintaining continuity in critical systems. This approach highlights the importance of modularity, as it prevents disruptions to system performance even when policies evolve over time. We also learned how modular enforcers could enable more granular, flexible responses to specific security events, which is essential for CPS operating in dynamic, high-stakes environments like autonomous vehicles or medical devices. By leveraging these concepts, we gained a deeper understanding of how to balance flexibility with robust security, a central consideration in our project design.

In our review of scalability in security enforcement mechanisms[2] for Cyber-Physical Systems (CPS), we examined strategies for overcoming the limitations of monolithic approaches. The analysis emphasized how compositional methods offer a pathway to efficient, scalable security enforcement by avoiding state space explosion—a common issue when incrementally adding new policies. This insight into scalability is vital for building adaptable systems that can uphold security compliance even as policies increase in both number and complexity. Through this study, we recognized the importance of structuring security frameworks to handle growth seamlessly, ensuring that system performance remains unaffected. This understanding is crucial for designing our project, as it underscores how a modular, compositional approach can support continuous, scalable security adaptations without overwhelming the system.

Our literature review on monitoring and defense mechanisms[3] for industrial Cyber-Physical Systems (CPS) provided valuable insights into addressing unique security challenges specific to this domain. This research highlighted real-time monitoring and alert systems as crucial elements for effective CPS security, particularly in swiftly detecting and responding to common attacks. While the focus was primarily on monitoring rather than enforcement, the paper underscored the importance of integrating detection mechanisms with runtime enforcement to create a comprehensive CPS defense strategy. From this, we gained a deeper understanding of how to blend monitoring with enforcement for enhanced resilience, helping us refine our approach to developing a more responsive, layered security solution for CPS.

Chapter 3: Problem Analysis

Cyber-physical systems are exposed to sophisticated Cyber Physical attacks that require proactive, flexible, and scalable security mechanisms. The research identified that traditional enforcement models could not scale with the continuous addition of policies, creating bottlenecks and inefficiencies.

3.1 Problem Statement

To address the security challenges in CPS, this project seeks to develop a Runtime Enforcement framework that can **incrementally compose enforcers** for new policies, enhancing security adaptability without re-certifying the entire system.

3.2 Existing Methodologies

The conventional CPS security system uses **monolithic enforcers**, which combine all policies into a single-state machine. While effective for enforcing policies, this method suffers from significant limitations in scalability, adaptability, and efficiency.

3.2.1. Monolithic Enforcement

Monolithic enforcement consolidates all security policies into a single, unified state machine, aiming to handle multiple policies simultaneously within one overarching enforcer structure.

In this approach, each policy does not operate independently; instead, all policies are merged into a single enforcement mechanism that must evaluate and apply each policy in tandem.

While this unification allows for centralized control, it can lead to increased complexity as each policy's conditions and requirements must be accounted for within a single system.

As policies grow in number and intricacy, the monolithic enforcer's state machine must expand to represent every possible interaction, which often leads to what's known as "state space explosion"—a dramatic increase in the number of states the system must manage.

This not only strains memory and processing resources but also makes the system harder to modify, as any addition or adjustment to a policy can affect the overall structure.

Consequently, while monolithic enforcement provides a singular control point for policy application, it often sacrifices scalability and flexibility, making it less suitable for dynamic environments like Cyber-Physical Systems (CPS) that frequently need to adapt to evolving security demands.

- **Limitations**:
    - **State Space Explosion**: Each additional policy exponentially increases the state space, requiring more memory and computational resources, which impacts performance.
    - **Re-certification Requirement**: Every time a new policy is added, the enforcer must be re-synthesized, which is a time-consuming and inefficient process. This re-certification disrupts the system, making it unsuitable for dynamic environments.
    - **Loss of Modularity**: Policies are interdependent, making it difficult to isolate or modify them individually. Any change to one policy can inadvertently impact others, reducing

flexibility and increasing the likelihood of conflicts.

3.2.2 Runtime Verification (RV) Techniques

Runtime Verification (RV) techniques are centered on the passive observation of system behavior to ensure policy compliance, focusing on detecting deviations from expected norms without directly intervening in system operations.

In this approach, monitors are designed to observe the sequence of events and states that occur within a system in real-time, checking against predefined security or operational policies. Rather than actively enforcing rules, these monitors serve as a "watchdog" by tracking adherence and detecting violations as they happen.

Observation-based verification is the primary strategy within RV, where monitors operate in the background and continuously assess system actions for compliance. This passive approach allows RV to function with minimal disruption to system performance, as it does not alter the execution flow but instead raises alerts or logs incidents when non-compliant behaviors are detected.

This can be particularly valuable in high-stakes environments, such as industrial automation or autonomous systems, where it's critical to maintain system integrity while minimizing intervention.

- **Limitations**:
    - **No Dynamic Enforcement**: RV methods are passive; they do not correct or enforce compliance dynamically but only report violations. Manual intervention is needed, which can delay response times.
    - **Inadequate for Reactive CPS**: RV techniques lack the real-time intervention capability required in reactive CPS (e.g., autonomous vehicles or medical devices), where immediate correction is essential for safety.
- **Applications**:
    - **Fault Detection**: Identifying when the system deviates from expected behavior.
    - **Anomaly Detection**: Recognizing unusual or potentially harmful patterns in system behavior.
    - **Policy Compliance Reporting**: Logging and reporting compliance status for auditing and analysis purposes.

3.2.3. Synchronous Programming Approaches

Synchronous programming is particularly well-suited for defining deterministic safety automata in Cyber-Physical Systems (CPS), as it emphasizes predictable, time-driven behavior where the system reacts in a tightly controlled, deterministic manner to each cycle or external event.

In a synchronous model, time is divided into discrete steps or cycles, with each cycle representing a moment where the system assesses inputs, updates its internal state, and produces outputs. This cycle-based approach is foundational for safety-critical applications, as it ensures the system's behavior remains consistent and easily predictable.

One of the primary benefits of synchronous programming in CPS is its real-time responsiveness, which is essential for systems where safety policies must be enforced without delay. Each event or input triggers an immediate response, enabling the system to continuously monitor and enforce security policies within each cycle.

For example, if a safety policy detects a potentially hazardous condition, synchronous programming allows for a swift response in the very next cycle, minimizing the risk of damage or security breaches.

- **Usage**:
    - **Deterministic Safety Automata**: Synchronous programming allows for the creation of predictable, cycle-based automata that enforce strict policies.
    - **Predictable Behavior**: Each response is consistent and crucial for CPS applications that demand reliability.
    - **Real-Time Responsiveness**: Ensures the system enforces policies immediately within each cycle, making it responsive to real-time demands.
- **Limitations**:
    - **No Incremental Policy Support**: Traditional synchronous approaches are static, meaning policies are defined at setup and cannot be easily modified or added later.
    - **Challenging Scalability**: It is costly and time-consuming to reconfigure or update policies, limiting the system's adaptability to new threats or requirements, which is a drawback for dynamically evolving CPS security needs.

3.3 Flaws and Disadvantages

- **State Space Explosion**: As policies increase, the state space grows exponentially.
- **Re-certification Needs**: Each policy addition requires re-validation.
- **Loss of Modularity**: Policies are tightly coupled, making modifications challenging.

3.4 Proposed System

The proposed system is a **compositional RE framework** that allows each policy to be enforced by an individual enforcer, added incrementally to the system. By serially composing enforcers, the system supports scalable and modular security enforcement for CPS.

3.5 Functional Requirements

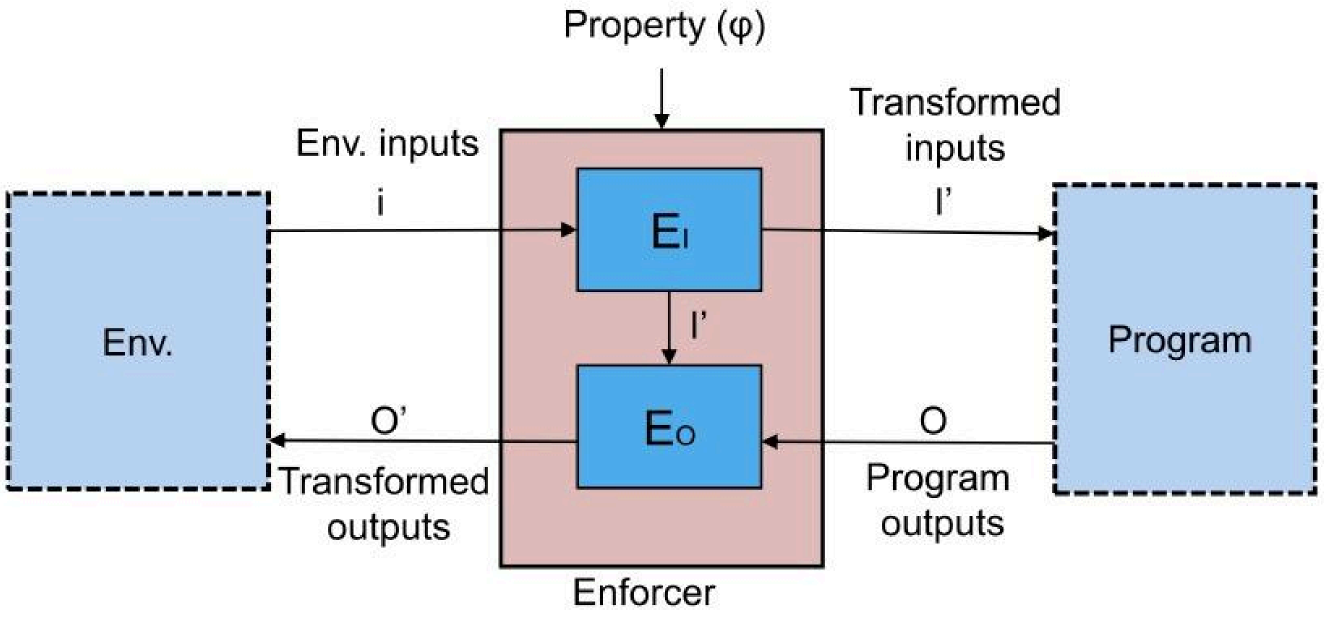
- **Dynamic Policy Addition**: Ability to add new policies without re-certifying the entire system.
- **Real-Time Monitoring**: Immediate enforcement of policies during system operation.
- **Modular Structure**: Independent enforcers for each policy, allowing isolated modifications.

3.6 Non-Functional Requirements

- **Efficiency**: Minimize resource consumption while maintaining performance.
- **Reliability**: Ensure system adherence to policies under all conditions.
- **Scalability**: Support the addition of numerous policies without performance degradation.

Chapter 4 System Analysis

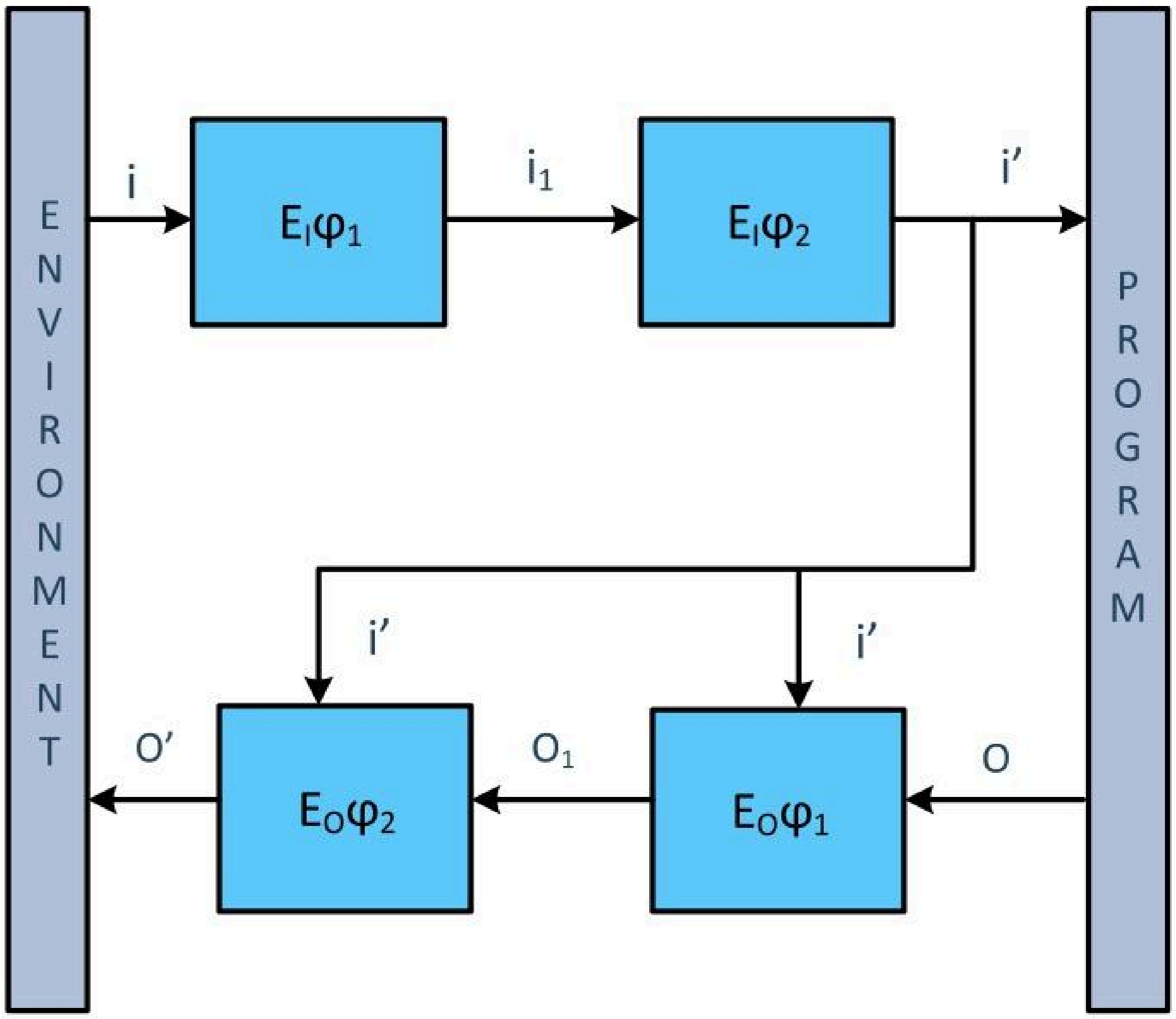
The design consists of a **bi-directional enforcement model** where input and output enforcers verify and adjust data flowing between the CPS and the environment. By intercepting and validating both inputs and outputs, this design prevents unsafe actions from reaching the CPS or the environment.



4.1 Proposed System Architecture

The methodology for this incremental security enforcement project begins with the development of a bi-directional runtime enforcement (RE) framework tailored to monitor and control both inputs and outputs in cyber-physical systems (CPS). This framework addresses the complex requirements of CPS, where new security policies must be incrementally added without triggering a complete overhaul of the system's enforcement structure. The project leverages Python for implementation, with custom tools such as the easy-rte-composition compiler, which extends the capabilities of the original easy-rte runtime enforcement engine to support serial policy composition.

To manage the complexity associated with enforcing multiple security policies, the project applies two approaches: a monolithic enforcement model, which combines all policies into one, and a serial composition model, which enforces policies individually in sequence. The serial composition approach allows the addition of new policies without affecting the existing ones, thus preventing state-space explosion and maintaining efficient operation as the number of policies grows.


ENVIRONMENT
PROGRAM
i
$E_I\varphi_1$
i_1
$E_I\varphi_2$
i'
i'
i'
O'
$E_O\varphi_2$
O_1
$E_O\varphi_1$
O

Chapter 5 Algorithms

The project introduces several key algorithms to implement the **compositional runtime enforcement (RE) framework** for Cyber-Physical Systems (CPS). These algorithms focus on enforcing policies on inputs and outputs, managing multiple policies using serial composition, and ensuring compliance by selecting compliant alternatives when violations occur. Below is a detailed overview of each algorithm.

5.1 Input and Output Enforcement Functions (EI and EO)

These functions enforce security policies on inputs (from the environment to the system) and outputs (from the system to the environment). They are essential for maintaining compliance with policies in real time by verifying and modifying events as needed.

Input Enforcement Function (EI)

- **Purpose**: To monitor and, if necessary, modify inputs from the environment before they are processed by the system.
- **Algorithm Steps**:
    1. **Receive Input**: For each new input event xxx from the environment, the function evaluates if it complies with a specified policy ϕ.
    2. **Compliance Check**: If xxx does not satisfy ϕ, it's deemed non-compliant.
    3. **Modification (Edit)**: The input is modified to a compliant version x′x'x′ by applying the **edit function**, which finds the closest alternative input that meets policy ϕ.
    4. **Pass to System**: The compliant input x′x'x′ is then passed to the system for processing.

Output Enforcement Function (EO)

- **Purpose**: To monitor and potentially modify outputs generated by the system before they are sent back to the environment.
- **Algorithm Steps**:
    1. **Receive Output**: Each output yyy generated by the system is evaluated against policy ϕ\ϕϕ.
    2. **Compliance Check**: If yyy violates ϕ, it is flagged as non-compliant.
    3. **Modification (Edit)**: The output is transformed into a compliant version y′y'y′ using the **edit function**, ensuring it meets policy ϕ.
    4. **Emit to Environment**: The compliant output y′y'y′ is sent to the environment, preserving system security and policy adherence.

5.2 Serial Composition of Enforcers

This algorithm allows the framework to enforce multiple policies by composing individual enforcers in series. Serial composition is crucial for handling incremental policy updates, as it prevents state space explosion and allows for modular policy management.

- **Purpose**: To enforce multiple policies incrementally by chaining enforcers, where each enforcer is responsible for a specific policy.
- **Algorithm Steps**:
    1. **Initialize Enforcers**: Define separate enforcers Eϕ1,Eϕ2,…,Eϕn ϕ1, ϕ_2,...ϕn

Eϕ1,Eϕ2,…,Eϕn for each policy ϕ1,ϕ2,…,ϕn ϕ_1, ϕ_2, … ϕ_nϕ1,ϕ2,…,ϕn.

2. **Compose Serially**: Arrange the enforcers in series. For any input or output event:
    - The event first passes through Eϕ1, which checks it against policy ϕ1.
    - If Eϕ1 modifies the event, the modified version is passed to Eϕ2, and so on, through Eϕn.
3. **Final Output**: The final modified or validated event is output from the last enforcer in the chain.
4. **Compliance Guarantee**: Each policy enforcer works independently, which ensures that each policy is enforced without causing interdependencies or exponential state growth.

5.3 Edit and Select Functions

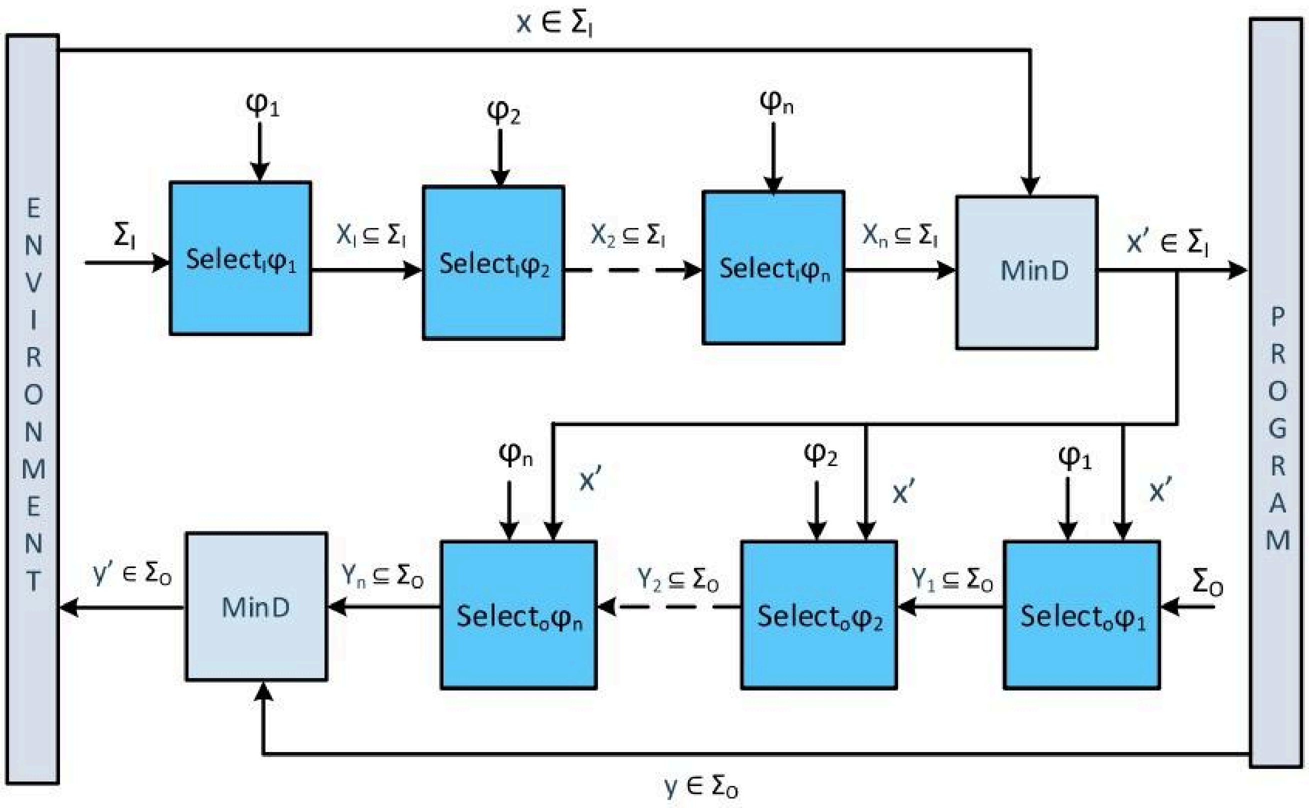
These functions are responsible for finding and selecting policy-compliant alternatives when violations are detected. They minimize disruption to system behavior by selecting alternatives that closely match the original input or output.

Edit Functions

- **editIϕI**: Handles inputs that violate policy, finding a compliant alternative that minimally deviates from the original input.
- **editOϕ**: Handles outputs that do not satisfy policy, selecting an alternative output that aligns with policy requirements.
- **Algorithm Steps for Edit Functions**:
    1. **Identify Violation**: When an input xxx or output yyy fails to comply with policy ϕ.
    2. **Generate Alternatives**: Compute a set of possible compliant alternatives X′X'X′ (for inputs) or Y′Y'Y′ (for outputs).
    3. **Select Closest Alternative**: Use the **Select Function** to choose the alternative x′x'x′ (or y′y'y′) that is closest to the original xxx (or yyy) while meeting policy ϕ.
    4. **Output Compliant Event**: The modified, policy-compliant event is then used as the new input/output for the system.

Select Functions

- **SelectIϕI**: Given a set of possible inputs X′X'X′, chooses a policy-compliant input that closely resembles the original input xxx.
- **SelectOϕ**: Given a set of possible outputs Y′Y'Y′, select a compliant output closest to the original output yyy.
- **Algorithm Steps for Select Functions**:
    1. **Input/Output Alternatives**: The function identifies all compliant alternatives for an input/output event.
    2. **Minimize Deviation**: Among the alternatives, choose the one with the minimal deviation from the original event, preserving system consistency.
    3. **Return Selected Event**: Return the closest compliant event for processing or emission.



5.4 Incremental Enforcement Function

This function defines the overall enforcement strategy by combining input and output enforcement functions to maintain compliance across both incoming and outgoing events in real-time.

- **Purpose**: To enforce policies incrementally by applying input and output enforcers sequentially, ensuring that all events comply without disrupting the entire system.
- **Algorithm Steps**:
  1. **Event Processing**: For each input/output event (x,y)(x, y)(x,y):
     - First, apply **Input Enforcement (EI)** to validate the input event.
     - After the system processes the input and produces an output, apply **Output Enforcement (EO)**.
  2. **Modification for Compliance**: If either the input or output fails to comply, use the **edit function** to select the closest compliant alternative.
  3. **Causal Order Maintenance**: Ensure that each input is enforced before it affects output, maintaining causality and consistency in policy enforcement.
  4. **Emit Final Output**: The compliant input/output events are processed and emitted, ensuring system compliance with security policies in real-time.

5.5 Key Benefits of Using These Algorithms

The proposed compositional RE framework brings several notable advantages, addressing key challenges in securing Cyber-Physical Systems (CPS). This approach combines modularity, scalability, and real-time compliance, making it well-suited for dynamic, high-stakes environments where security policies need to be updated or added without disrupting system performance. Here's an in-depth look at each benefit:

5.5.1 Modularity

- **Independent Policy Management**: In traditional monolithic enforcement, all policies are bundled into a single enforcement structure, which makes modifying or isolating individual policies challenging. In contrast, the compositional RE framework uses **serial composition** to assign each policy its own enforcer. This design allows each policy to operate independently, without interdependencies or conflicts with other policies.
- **Flexible Policy Updates**: The modularity of the framework means new policies can be added, modified, or removed without needing to revalidate or redesign the entire enforcement system. This flexibility is essential in CPS environments where security policies may need regular updates in response to evolving threats.
- **Easier Debugging and Maintenance**: With each policy managed separately, debugging and maintenance are simplified. If an issue arises with a specific policy, engineers can isolate the corresponding enforcer without impacting others, reducing downtime and making troubleshooting more efficient.

5.5.2 Scalability

- **Avoids State Space Explosion**: A major limitation of monolithic approaches is the state space explosion problem, where the addition of new policies exponentially increases the state space, resulting in high memory and processing demands. By independently handling each policy enforcer, the compositional RE framework limits the state space to manageable levels. Each enforcer's state machine remains small and efficient, regardless of the number of policies, enabling the system to scale with minimal impact on performance.
- **Efficient Resource Usage**: As the framework does not merge policies into a single state machine, it uses computational resources more efficiently. This is particularly valuable for CPS with constrained resources, such as embedded systems in IoT devices, where memory and processing power are limited.
- **Linear Performance Scaling**: Testing has shown that the framework maintains a **linear relationship** between performance metrics (e.g., compile time, execution time) and the number of policies. This means that as new policies are added, system performance remains predictable and manageable, allowing CPS to scale securely.

5.5.3 Real-Time Compliance

- **Immediate Policy Enforcement**: The framework's **Input (EI) and Output (EO) enforcement functions** to ensure that policies are enforced on every incoming and outgoing event, maintaining continuous compliance with security policies in real-time. This capability is vital in CPS applications (e.g., autonomous vehicles and healthcare devices), where a delay in enforcement could lead to unsafe or unauthorized actions.
- **Dynamic Response to Threats**: By intercepting and verifying each input and output, the framework can dynamically adjust to security threats as they arise. If a policy violation is detected, the enforcer can modify the event to a compliant alternative in real time, preventing the propagation of unsafe behaviors throughout the system.
- **Consistent System Behavior**: The **edit and select functions** ensure that non-compliant inputs and outputs are replaced with the closest compliant alternatives, minimizing the deviation from intended system behavior. This preserves the system's functionality while ensuring security

compliance, which is crucial for applications where reliability and predictability are essential.

5.5.4 Adaptability to Evolving Threats

- **Incremental Policy Integration**: With the compositional RE framework, policies can be added incrementally as new threats emerge without needing to overhaul or re-certify the entire system. This adaptability allows CPS to respond to a constantly evolving threat landscape, maintaining security compliance without compromising system uptime or performance.
- **Reduced Re-Certification Needs**: In safety-critical CPS applications, adding new policies typically requires re-certifying the entire system, which can be time-consuming and costly. The modular nature of this framework allows for seamless policy updates with minimal re-certification, ensuring compliance while reducing operational disruptions.

5.5.5 Improved Reliability and Security

- **Enhanced Fault Isolation**: The independent enforcers isolate policy functions, meaning that any issue with one policy enforcer does not compromise the operation of others. This isolation enhances reliability by reducing the risk of cascading errors that could compromise the entire system.
- **Predictable and Transparent Operation**: Each enforcer operates in a deterministic, cycle-based fashion, producing predictable behavior. This predictability simplifies system auditing, as each policy enforcer behaves consistently, making it easier to verify that security policies are being enforced reliably.

Conclusion

The compositional runtime enforcement framework we are developing aims to address key security challenges in Cyber-Physical Systems (CPS) as they face increasingly complex threats.

Throughout this project, we are learning how to build a robust, scalable solution that departs from traditional monolithic and runtime verification approaches by implementing modular enforcers, serial composition, and real-time compliance capabilities.

We are exploring how this approach supports incremental policy additions without re-certification, helps avoid state space explosion, and provides consistent, real-time responses to security issues.

By designing a framework adaptable to dynamic environments, we are gaining insight into how to make CPS more resilient, especially for high-security applications like autonomous vehicles, industrial automation, and medical devices.

This project is enhancing our understanding of integrating security with flexibility and efficiency in today’s interconnected systems.

References

1.**Incremental Security Enforcement for Cyber-Physical Systems** by Abhinandan Panda, Alex Baird, Srinivas Pinisetty, and Partha Roop

2.**Scalable Security Enforcement for Cyber-Physical Systems** by Alex Baird, Abhinandan Panda, Hammond Pearce, Srinivas Pinisetty, and Partha Roop

3.**Monitoring and Defense of Industrial Cyber-Physical Systems Under Typical Attacks** by Yuchen Jiang, Shimeng Wu, Renjie Ma, Ming Liu, Hao Luo, and Okyay Kaynak