On Security Policy Migrations (in ACM SACMAT'20)

Jorge Lobo, Elisa Bertino, Alessandra Russo

Wednesday 27th November, 2024

Motivation

Scenario

How can security policies be effectively migrated from a source device to a target device, considering the differing computational capabilities of the devices and the changes in operational context?

Example Scenario

Scenario

Step 1: A database server is accessed by multiple clients, and a policy is in place to prevent SQL injection attacks.

Step 2: If the server is replaced with a less powerful one that cannot enforce the same level of monitoring (e.g., it lacks the ability to track state-action pairs of client behavior), the policy might need to be adapted or approximated, potentially weakening its enforcement.

Step 3: This situation underlines the trade-offs and risks in security policy migration.

Major Contribution i

A Formal Framework for Policy Migration

- Introduces a formal framework to characterize and analyze different types of security policy migrations.
- Highlights that the migration process is influenced by:
  - The computational capabilities of the source and target devices.
  - The contexts in which the policies are deployed.
- Example:
  - **Same Context Migration:** A policy is migrated between devices in the same operational environment (Context-independent migrations).
  - **Different Context Migration:** A policy is migrated between devices in different operational environments.

Major Contribution ii

Analysis of Policy Migration Results

- The paper reviews existing work on policy migration within the same context.
- Identifies methodologies from other areas of security research that can assist in addressing challenges related to: Context-independent migrations and Context-dependent migrations.
- Discusses how to evaluate the quality of the migrated policies using techniques such as policy mining, policy learning, and policy similarity analysis

Same Context Migration

What is it?

Migration of a security policy between devices operating in the same environment or context but with different enforcement mechanisms.

Goal

Ensure the migrated policy on the target device behaves faithfully, preserving the original policy's intent.

Faithful Migration

What is Faithful Migration?

Faithful migration occurs when the source policy can be directly transferred to the target device without modifications or event mapping because the target is either equivalent to or more capable than the source.

Example

- Source: An iptables firewall that inspects packet headers.
- Target: A Snort IDS that inspects both headers and payloads.
- **Why it works:** The Snort IDS can directly implement the iptables rules (as they are a subset of its capabilities) without any changes or event mapping.
- **Result:** Faithful migration is possible because the target has higher computational and observational capabilities.

When Faithful Migration Fails i

When does it fail?

- Faithful migration fails when the target device has **lower computational or observational capabilities** than the source.

Example

- **Source:** Snort IDS inspects both **headers and payloads**.
- **Target:** iptables firewall only inspects **headers**.

Why it fails

The **iptables firewall cannot analyze payloads**, so the policy cannot be directly migrated.

Faithful Implementation i

When is it used

Faithful implementation is used when **faithful migration fails**. The source events are mapped to equivalent events on the target using a mapping function.

Example

- **Source:** Snort IDS inspects payloads for SQL injection patterns.
- **Target:** A basic firewall processes only headers.
- **Event Mapping:** Translate “SQL injection detected in payload” to “block packets from malicious IP.”
- **Result:** The target blocks all packets from flagged IPs, approximating the source’s behavior.

When Faithful Implementation Fails

When does it fail?

Faithful implementation fails when the target device cannot observe, process, or replicate the same events or behaviors as the source device due to limitations in computational capabilities, state representation, or environmental differences (e.g., encrypted traffic).

Example

Source: A Snort IDS inspects plaintext traffic and detects SQL injections in payloads from IP 192.168.1.10.

Target: Another Snort IDS operates in an encrypted network, where it can only see encrypted packets, not the payloads.

Result: Faithful implementation fails because target monitor cannot observe the necessary details (payload data) in the encrypted traffic to replicate the source's behavior.

Safe Implementation

When is it used

- When faithful implementation fails, safe implementation is utilized to ensure policy enforcement.
- It ensures that any execution trace X rejected by the source is also rejected by the target, even if the target rejects more than necessary.

Example

- Source: A Snort IDS in an unencrypted network detects SQL injection patterns in payloads.
- Target: A Snort IDS in an encrypted network cannot inspect payloads but can still analyze packet headers. Approach:
- Instead of inspecting payloads for SQL injection, it blocks all packets from IPs flagged as malicious by the source monitor, even though some benign packets may also be blocked.

Variable Context Migration i

What is it?

- Migration of a security policy between devices operating in the different environment or context.
- A Local Area Network (LAN) within an organization enforces a policy (P_s) and The policy needs to be migrated to another target LAN

Context Dependent Faithful Migration i

What is Context Dependent Faithful Migration?

How a policy (P_s) from a source context (C_s) is migrated faithfully to a target policy (P_t) in a target context (C_t), while maintaining the safety property (P^c).

Example

- **Source Context (C_s):**
  - A Local Area Network (LAN) where IT department devices are assigned IPs in the range 192.168.1.x.
  - **Source Policy (P_s):** Block traffic from 192.168.1.x.
- **Target Context (C_t):**
  - A different LAN where IT department devices are assigned IPs in the range 10.0.0.x.
  - **Target Policy (P_t):** Block traffic from 10.0.0.x.

Context Dependent Faithful Migration ii

Example Cont.

- Safety Property (P^c):
  - General rule: ”Block traffic from devices belonging to the IT department.”
  - In the source context, this means blocking 192.168.1.x.
  - In the target context, this means blocking 10.0.0.x.
- How Faithful Migration Happens:
  - Semantic knowledge is used to map the source context (192.168.1.x) to the target context (10.0.0.x).
  - The safety property (P^c) ensures that the target policy (P_t) aligns with the source policy (P_s) in terms of blocking IT department traffic.

When Context Dependent Faithful Migration Fails i

When does it fail?

- Faithful migration fails due to semantic knowledge gaps about the target context.
- We know the safety property (P^c) but cannot directly map events between the source and target contexts.

Context Dependent Faithful Implementation i

How It Works

- Use the safety property derived from the source (C_s) to enforce policy on the target (C_t)
- Approximate behavior by enforcing generalized safety rules.

Context Dependent Faithful Implementation ii

Example

- Key Points:
  - **Condition:** For every execution X in the set of valid executions (E_t^ω), the policy $P^c(C_t, X)$ is satisfied if and only if $EM_{P_t}^t[X] \neq \perp$ (the monitor does not reject X).
  - **Interpretation:** The implementation ensures that any execution allowed by $EM_{P_t}^t$ complies with the safety property P^c in the target context (C_t).
- Challenges:
  - The definition relies on P^c as the reference point for correctness, not directly on $EM_{P_s}^s$.
  - The exact nature of P^c might not be fully known in the target context, making faithful implementation challenging.

Context Dependent Safe Implementation i

How It Works

It prioritizes security by ensuring no invalid executions occur in the target system.

Context Dependent Safe Implementation ii

Example

- Key Points:
  - **Condition:** For every execution X in E_t^{ω}, if the target monitor does not reject X ($EM_{P_t}^{t}[X] \neq \perp$), then $P^c(C_t, X)$ must hold.
  - **Interpretation:** Safe implementation errs on the side of caution by rejecting any execution that might violate the safety property P^c.
- Challenges:
  - This approach can lead to over-rejection, where valid executions are blocked to ensure no unsafe actions occur.
  - Over-rejection impacts availability, as some valid operations are denied.

Current Approaches to Migration i

Introduces two primary types of policy migrations:

- **Same-context, different monitors:**
    - Source and target environments share the same context, but the monitors (enforcement mechanisms) differ.
- **Different-context, same monitors:**
    - Source and target environments differ in context, but the monitors are the same.

Unexplored Case:

- A scenario where both *context* and *monitors* differ is not directly addressed.

Same contexts, different monitors i

- Focuses on migrations where:
    - The source and target devices operate in the same context.
    - The source and target devices rely on different monitors for enforcing policies.
- It reviews existing solutions.
- It highlights challenges, particularly when:
    - Exact migrations (faithful migrations) are not possible.

Same contexts, different monitors ii

Existing Solutions in Faithful Migration

- Presents methods from the literature for *faithful migrations* in access control models:
    - **MAC/DAC to RBAC:** Mapping mandatory or discretionary rules into roles.
    - **RBAC to ReBAC:** Translating roles into relationship-based policies.
    - **Any Model to ABAC:** Attribute-based access control is highly expressive and supports faithful translations.
- **Network Monitor Migrations:**
    - Example: Translating Snort policies (deep packet inspection) into iptables (header inspection) using tools like *fwsnort*.
    - Highlights that such translations are often *not faithful* due to the limitations of the target monitor.

Same contexts, different monitors iii

Challenges of Non-Faithful Migrations

- Migrations in the reverse direction (e.g., ABAC to RBAC) or across monitors with reduced expressiveness are rarely considered because:
    - Faithful translations may not exist.
    - Loss of policy fidelity can lead to incorrect or incomplete enforcement.

When Faithful Migration Fails

When faithful migration fails, learning approximations are used to derive a target policy that approximates the source policy.

Same contexts, different monitors iv

Learning Techniques

- **Policy Mining:**
  - Extract rules or patterns from the logs of the source monitor.
  - Example: *"Users with admin roles can access all financial files."*
- **Symbolic Learning:**
  - Generalizes patterns from the logs to derive broader rules.
  - Risk: Over-generalization may introduce false positives (unwanted behaviors allowed).
- **Association Rule Learning:**
  - Focuses on identifying specific patterns and relationships in the logs.
  - Risk: Overfitting may lead to false negatives (valid actions denied).

Same contexts, different monitors v

Scoring Mechanisms

- To refine the approximated policy P_t, scoring functions are used:
    - **False Positives:** Incorrectly allow unwanted actions (e.g., allowing unauthorized access).
    - **False Negatives:** Incorrectly deny valid actions (e.g., blocking legitimate users).
    - **Trade-Off:** Balance between safety (minimizing false positives) and availability (minimizing false negatives) based on the migration's goals.
- Scoring Function Examples:
    - Prioritize strict safety for high-security environments.
    - Allow greater availability for environments prioritizing user convenience.

Different Contexts, Same Monitors i

- Explores challenges and potential solutions for policy migration where:
    - The source and target monitors are the same.
    - The contexts differ.
- Contextual differences arise due to:
    - Variations in attributes.
    - Differences in naming conventions.
    - Environmental factors (e.g., IP ranges, network structures).
- The section:
    - Identifies these challenges.
    - Proposes solutions derived from techniques in other areas of research.

Different Contexts, Same Monitors ii

Challenges in Context Dependent Faithful Migration

1. Changes in Attribute Naming and Values
2. Missing Attributes
3. Environmental Configuration Differences

Different Contexts, Same Monitors iii

Changes in Attribute Naming and Values

Attributes used in policies may have different names or values in the target context. **Example:**

- In C_s: Attribute ”Department” with values like ”HR” or ”Finance.”
- In C_t: Attribute ”Team” with equivalent values like ”Human Resources” or ”Accounting.”

Missing Attributes

The target context may lack certain attributes that are critical to the source policy.

Example: A policy in C_s depends on ”User Role,” but C_t has no concept of roles.

Different Contexts, Same Monitors iv

Proposed Solutions

To address these challenges, the authors propose techniques derived from other areas of research:

- Auto-Configuration Systems
- Ontology Mapping
- Learning-Based Approaches

Different Contexts, Same Monitors v

Auto-Configuration Systems

Use partially defined grammars that adapt policies dynamically at runtime based on the target context.

Example: Policies can adapt to local configurations (e.g., automatically map attribute names or IP ranges).

Different Contexts, Same Monitors vi

Ontology Mapping

Use ontologies (hierarchical relationships between terms) to reconcile differences in attribute names and values between source and target contexts.

Example: Map ”Manager” in C_s to ”Supervisor” in C_t using an ontology of organizational roles.

Different Contexts, Same Monitors vii

Learning-Based Approaches

When direct mappings or ontologies are unavailable, use policy mining or machine learning to approximate policies for the target context.

Example: Infer missing attributes by analyzing decisions and generating rules based on observed patterns.

Evaluating Approximations i

Same Context

- Goal:Assess the effectiveness of policy migration by analyzing how well the target policy aligns with the source policy.
- Key Metrices: True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN): Incorrectly blocked valid actions.
- Calculate Precision and Recall
- Criteria for Successful Migration:
  - High Precision: Minimal false positives (security-focused environments).
  - High Recall: Minimal false negatives (availability-focused environments).
  - Balance based on system needs (e.g., security vs. availability).

Evaluating Approximations ii

Different Context

This process emphasizes the need for semantic knowledge, synthetic data generation, and potential human intervention to accurately evaluate policy approximations in context-dependent migrations. By leveraging logs, ontology mappings, and administrator expertise, the migration process can address challenges in aligning source and target policies effectively.

Final Remarks i

- A systematic and precise method is needed to evaluate and compare policy approximations, especially in terms of security and usability.
- Evaluations based on error/confidence matrices are good starting points but there are other aspects that are specific to security that might need to be incorporated.
- Developing frameworks for automatic policy translation.

Final Remarks ii

Cont.

- To carry a migration we need to trust the migration process and the execution monitor. There is a need for the development of trusted migration protocols.
- No prior work addresses scenarios where both contexts and monitors differ simultaneously. A proposed approach is to split the migration into two steps: first, migrate between different monitors assuming the same context, and then migrate between different contexts assuming the same monitor type. However, the effectiveness of this approach remains uncertain at this stage.

THANK YOU :)