APETEEt: Secure Enforcement of ABAC Policies Using TEE

Pritkumar Godhani, Rahul Bharadhwaj, Shamik Sural
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

CONTENTS

ATTRIBUTE BASED ACCESS CONTROL

- Modern access control framework, known for flexibility.
- Incorporates:
  - SUBJECT: the user/entity requesting access
  - OBJECT: the resource that is being requested for access
  - ENVIRONMENT: the environment conditions under which the access request is made
  - ACTION: the type of access that is being requested
- Provides:
  - Context-Aware Decisions
  - Dynamic Access Control
  - Fine-Grained Policies

ABAC MECHANISMS

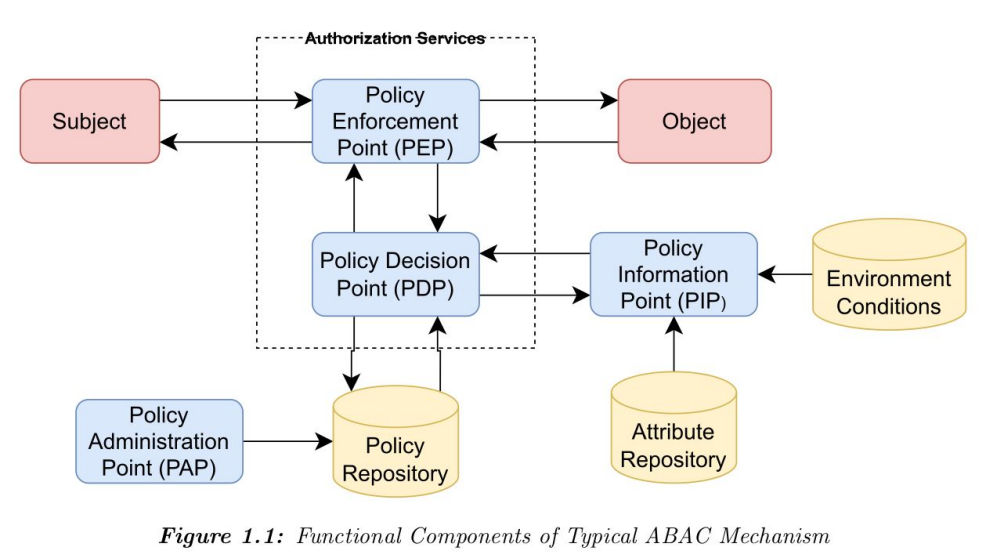
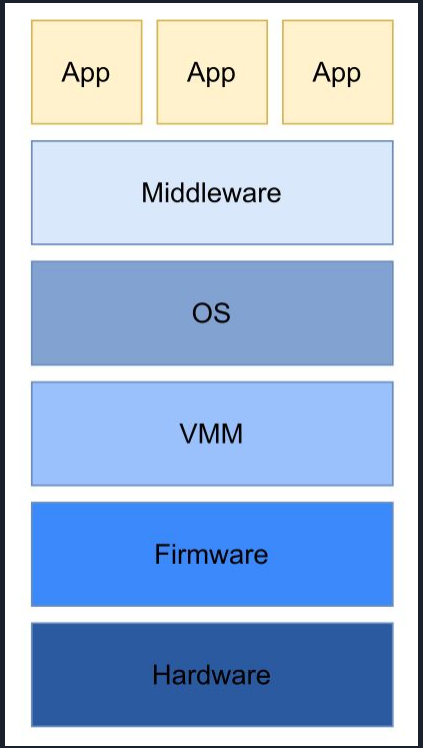


Figure 1.1: *Functional Components of Typical ABAC Mechanism*

- Typically, ABAC mechanisms are subdivided into four functionally separate components.
- Each may run on separate machines that may or may not be co-geolocated.
- Data repositories are used to manage and store data concerning the access control mechanism (ACM):
  - Policy Rules
  - Attributes of Users, Objects
  - Environment Context Detectors

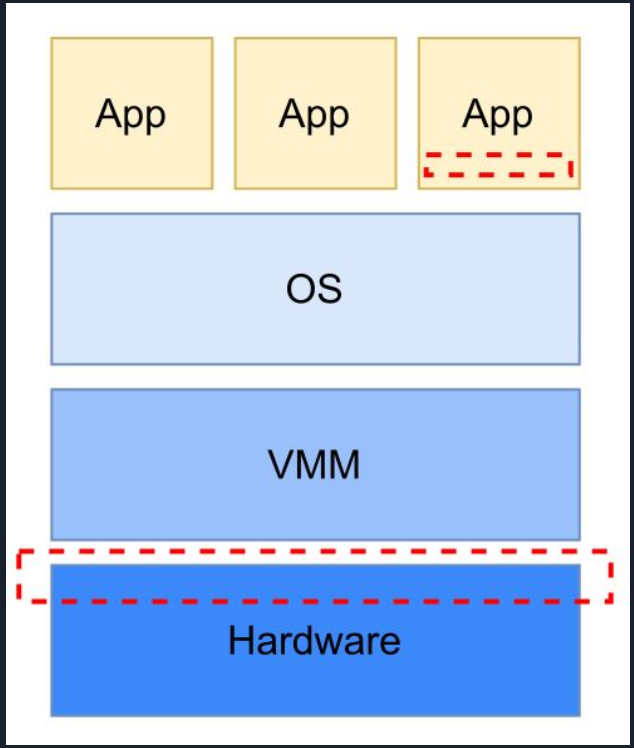
MOTIVATION

- Hosting ACMs on cloud and other remote infrastructures create issues of trust and security.
- In layered computer architectures, the lower, more privileged layers have full control over the resources of the layers higher up.
- Widens the trusted computing base, broadens the attack surface.
- *Spooky action at a distance…*



MOTIVATION (contd.)

- Using hardware security features like trusted execution environments (TEEs) can reduce the trusted computing base.
- TEE exists outside the privilege hierarchy and is supported directly on the hardware.
- TEEs use cryptographic encryption and decryption to communicate with untrusted code.
- Executions done in TEE can be verified by using signed attestation certificates.
- Examples: TPM, Intel SGX



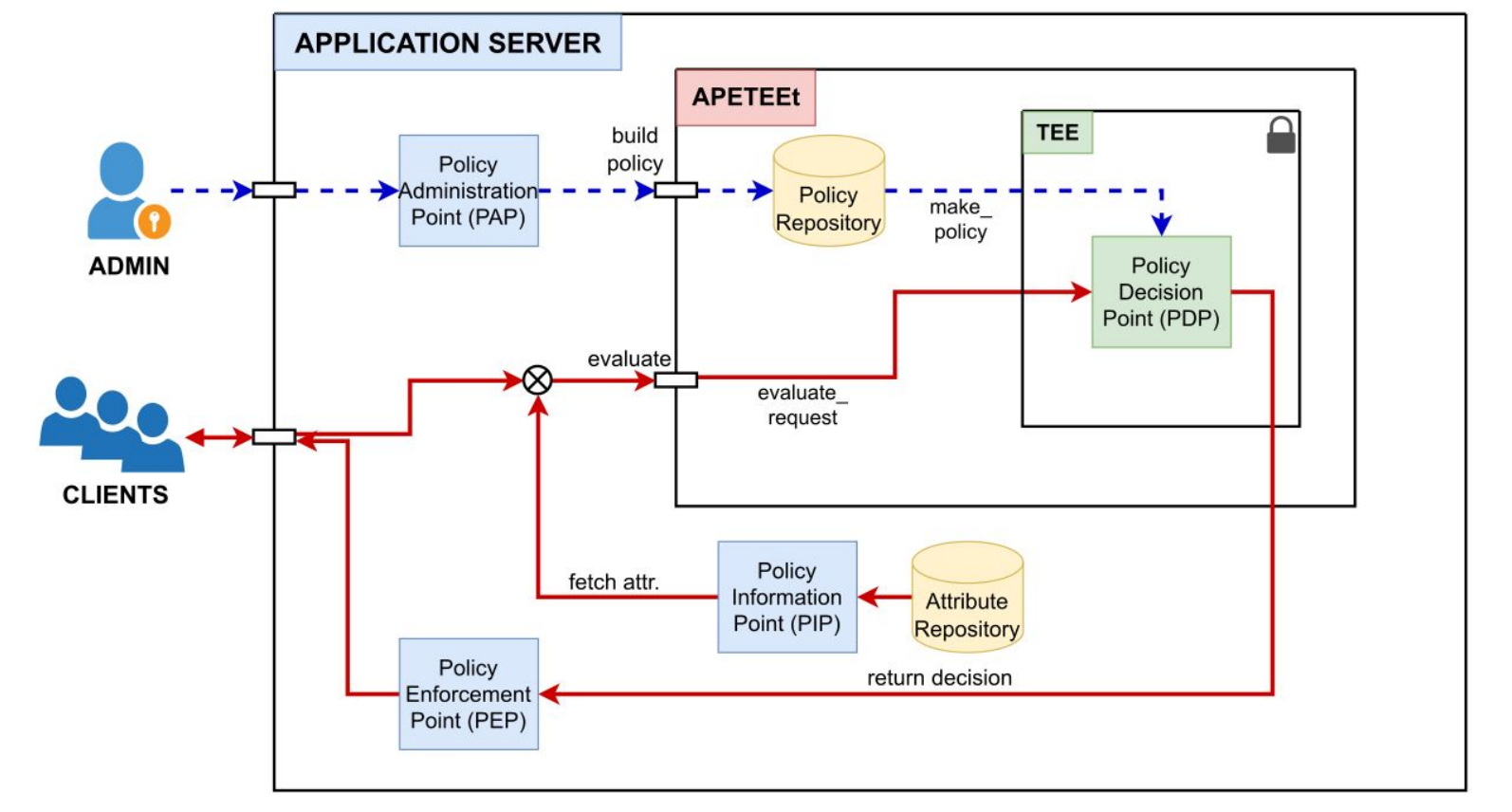
RELATED WORK

- TEE Protected Storage Systems:
  - Block Level : Mose (*Hoang et al, 2020*)
  - File Level : SecureFS (*Kumar et al, 2021*)
- Joplin (*Djoko, 2020*) — uses client side enclaves to ensure security of operations done on a server stored data.

APETEEt:
ABAC Policy Evaluation using Trusted Execution Environment

- Separate access control execution in a secure enclave on the server
- Create trust and security in the access control module
- Securely build policies, and evaluate them inside TEE
  - Ability to generate attestation certificates when required.
- No assumptions on underlying resource
- Servers enabling APETEEt do not need to change file system or disk drivers.
  - No huge configuration change

PROPOSED FRAMEWORK: APETEEt



DESIGN OF APETEEt

- Lightweight, modular and secure design
- Arbitrary policies can be securely built inside SGX enclaves
- Access requests on these policies can be securely evaluated via ecalls to the above defined enclaves
- Designed for infrastructure providers; acts as a utility for application developers to secure their access control mechanism

DESIGN OF APETEEt (contd.)

- Policies are built from the set of rules into an N-ary Policy Tree (PolTree).
- PolTree allow efficient evaluation of access requests.
  - Each non-leaf node acts a decision node depending on a fixed attribute.
  - Each possible value of this attribute is a child.
  - Each leaf node grants an access.
- Evaluation of access requests takes time equal to the depth of the tree, i.e., the number of attributes.
- This PolTree resides in the SGX enclave, once built after a build request.
  - SGX storage sealing is used for persistence across executions.

IMPLEMENTATION

- We release the core APETEEt enclave codes plus C++ wrapper functions;
  - We also release a sample Flask application that support the build and evaluate endpoints.
- Implemented on Ubuntu v20.04 with Intel SGX SDK for Linux.
- ECalls and OCalls only accept string buffers as data.
  - Special data structures wrapping attribute maps
  - Specified using a special Enclave Description Language
- Core APETEEt module consists of the SGX Enclave code and the wrapper functions written in C++; compiled to a object file.
  - Other languages can use linking utilities or libraries (such as pybind11 for Python).

EXPERIMENTS

- A policy generator module is implemented.
  - Given number of attributes and number of values, generates a consistent policy of a required number of rules.
- The number of rules, the number of attributes and the number of cumulative requests are varied pairwise, and the average time to process one request is measured.
- Results shows that APETEEt is scalable and efficient.
- Rules can be expressed in easy format JSON files.
  - Design can be easily modified to support XACML policies as well.

Open

policy_config.txt
~/Downloads/BTP/PolTree

Save

```
@$#SAPE_POLICY_CONFIG_TYPE
2
designation,professor,student
department,cse,ece
###
2
type,journal,textbook
confidentiality,high,low
###
1
day,weekend,weekday
###
2
read,write
```

Open

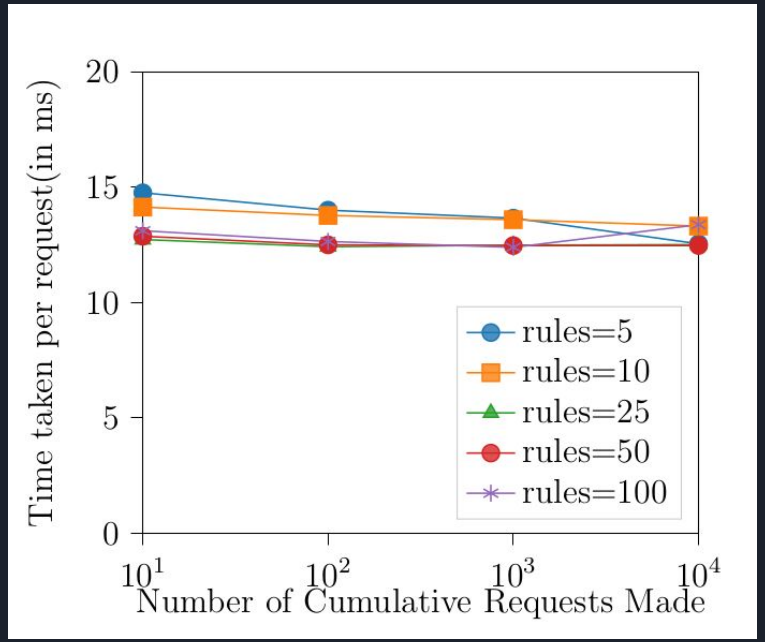
example_policy.json
~/Downloads/BTP/PolTree

Save

```
[
    {
        "name": "r1",
        "ua": { "designation": "professor", "department": "cse" },
        "oa": { "type": "journal", "confidentiality": "high" },
        "ea": { "day": "weekday" },
        "op": "write"
    },
    {
        "name": "r2",
        "ua": { "designation": "professor", "department": "cse" },
        "oa": { "type": "textbook", "confidentiality": "high" },
        "ea": { "day": "weekday" },
        "op": "write"
    },
    {
        "name": "r3",
        "ua": { "designation": "student", "department": "cse" },
        "oa": { "type": "journal", "confidentiality": "high" },
        "ea": { "day": "weekend" },
        "op": "read"
    },
    {
        "name": "r4",
        "ua": { "designation": "professor", "department": "ece" },
        "oa": { "type": "journal", "confidentiality": "low" },
```

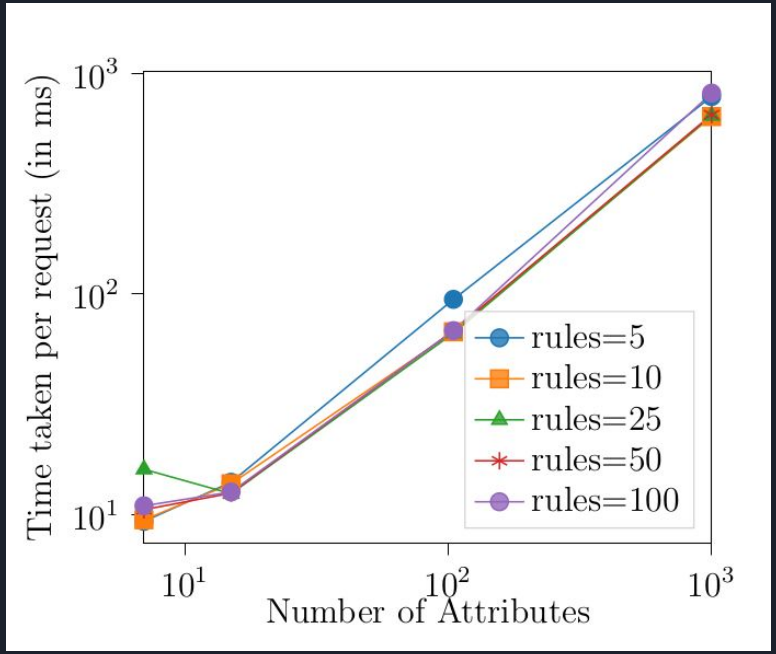
EXPERIMENTS

- For fixed number of attributes, the time taken for different number of cumulative requests for different rule set sizes is plotted.
- Result shows that number of requests made do not affect the time per request and that APETEEt is highly scalable.



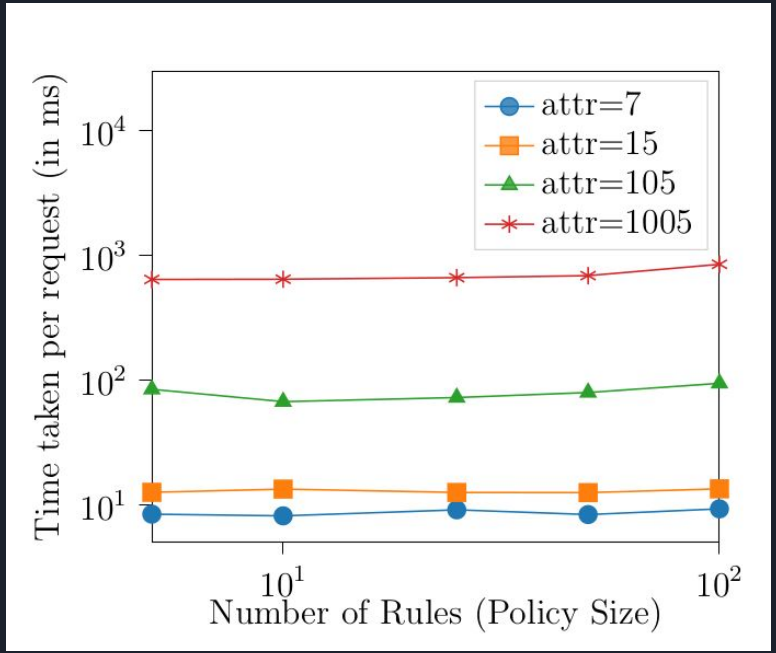
EXPERIMENTS

- Next, the effect of number of attributes on the request evaluation latency for a fixed number of rules is measured.
- Since, PolTree evaluation is linear in the number of attributes, the same variance is observed when number of attributes are increased.



EXPERIMENTS

- Finally, we look at how the number of rules in the policy affect the request latency for a fixed number of attributes.
- Since, PolTree depth is not affected by the number of rules but only by the number of attributes, the request latency remains unaffected by increase in the policy size.



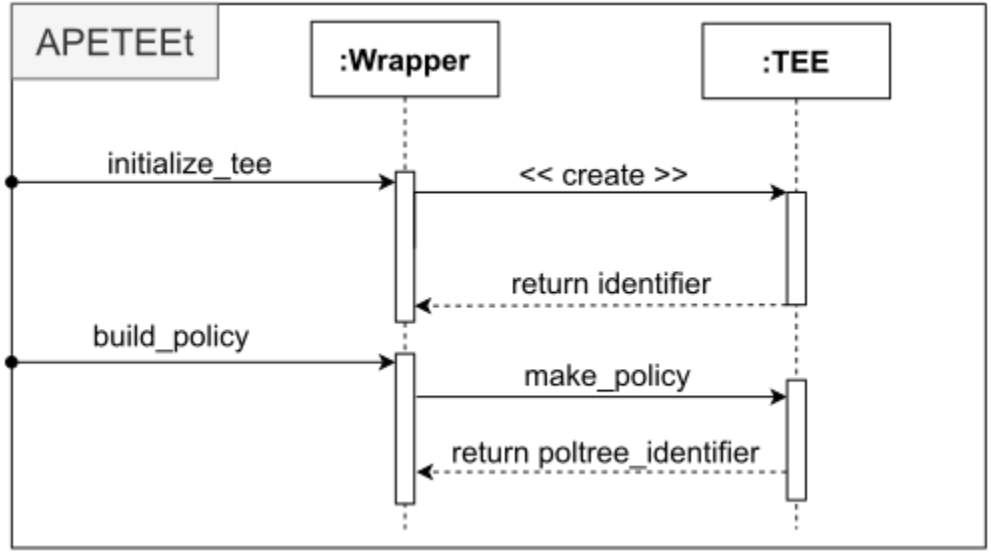
CONCLUSION & FURTHER

- APETEEt secures ABAC policies on the server side in a lightweight fashion.
- It avoids re-configuration changes on servers; works seamlessly as a wrapper on the server-hosted application code.

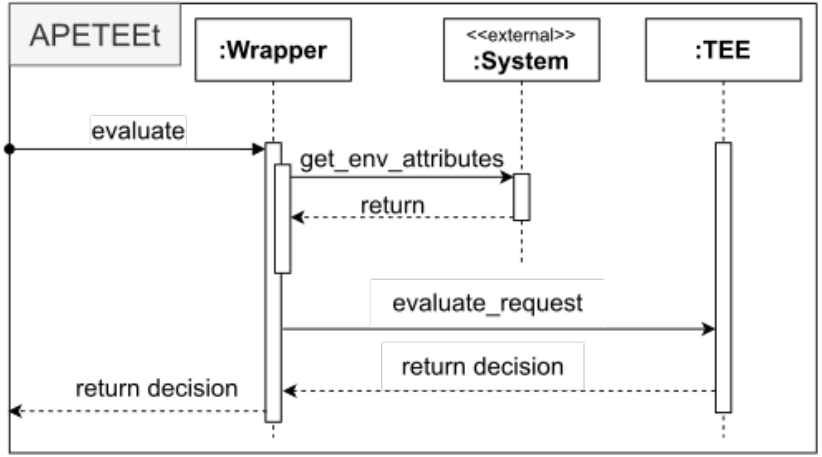
Security of Intel SGX enclaves and communication channels are orthogonally applied to APETEEt. Other parts of ACM into the enclave; customizable design according to developer needs.

Thank You

APPENDIX



(A) Make-Policy Sequence Diagram



(B) Evaluate-Request Sequence Diagram