Fixing Privilege Escalations in Cloud Access Control with MaxSAT and Graph Neural Networks

Yang Hu*, Wenxi Wang*, Sarfraz Khurshid, Ken McMillan, Mohit Tiwari

The University of Texas at Austin

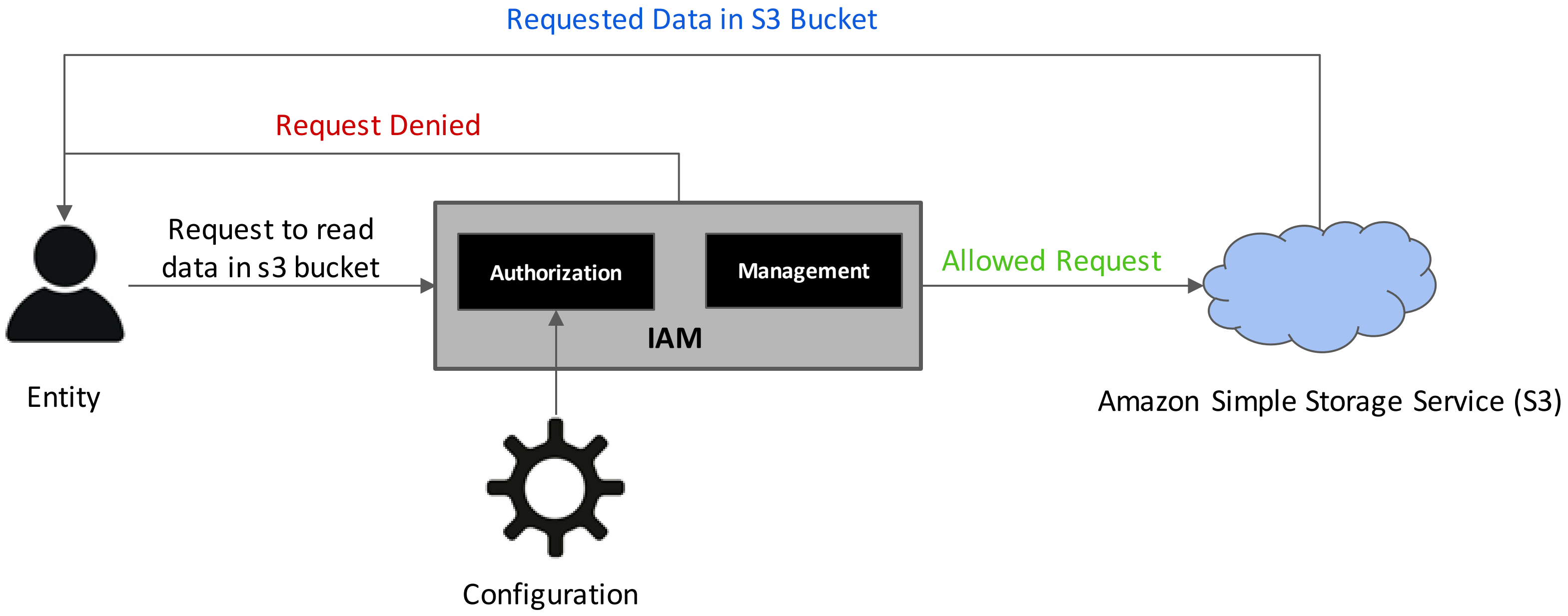
ASE, Luxembourg, 12 Sep 2023

* denotes that these authors contributed equally to the paper.

Background: Identity and Access Management (IAM)

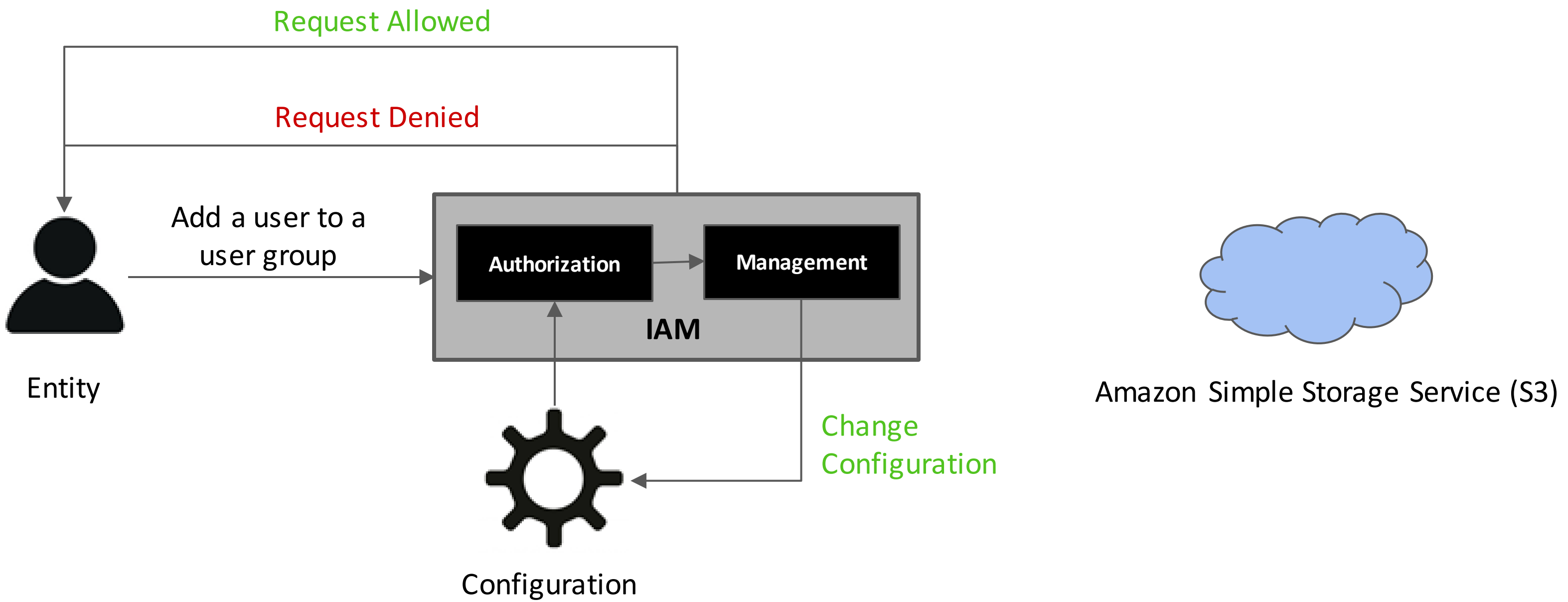
IAM is an access control service in cloud platforms

Example use case:

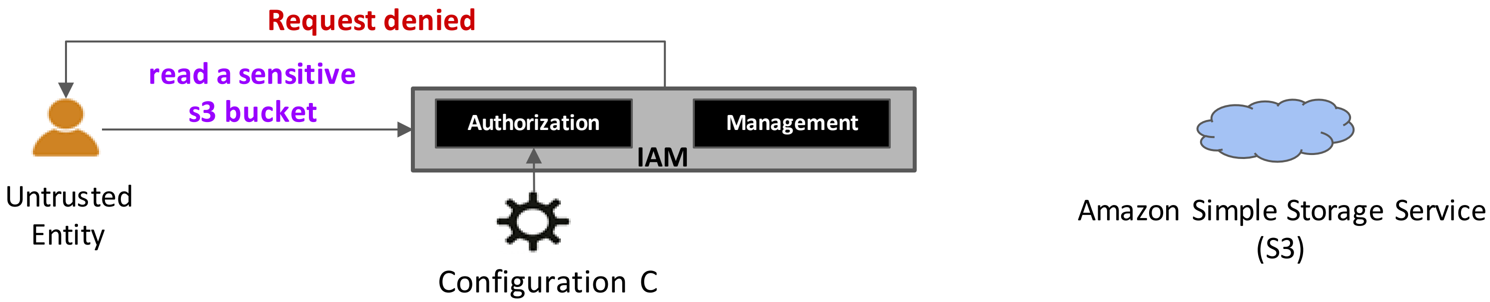


Background: Identity and Access Management (IAM)

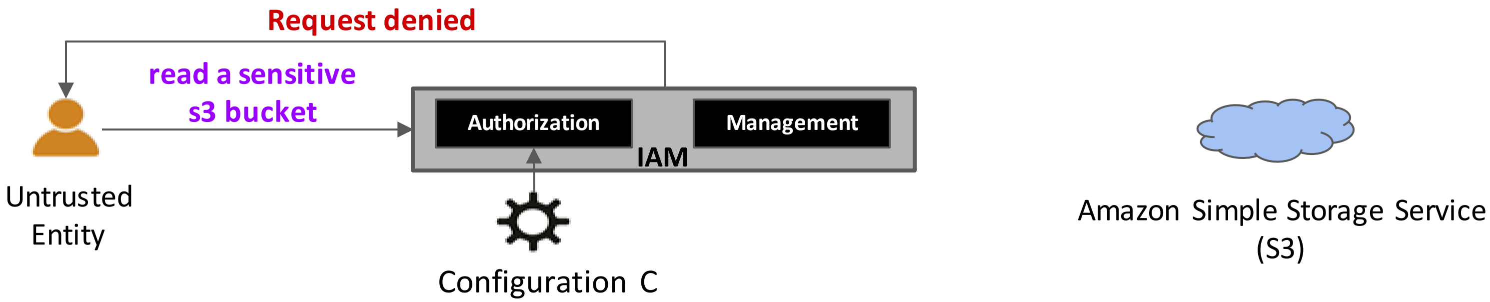
Another example use case: Configuration change



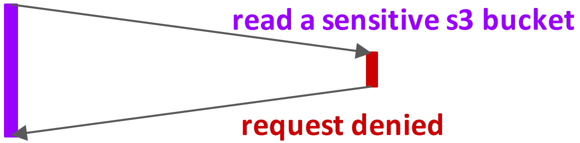
Background: Privilege Escalation (PE) in IAM Misconfiguration



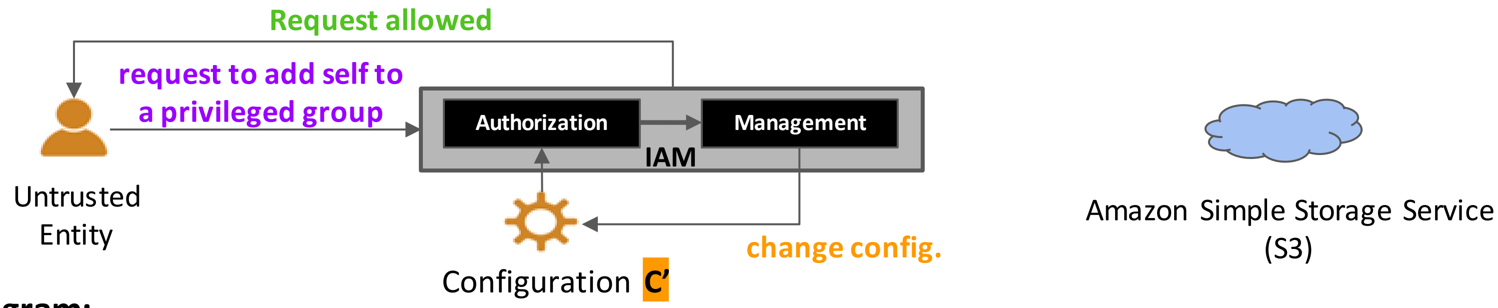
Background: Privilege Escalation (PE) in IAM Misconfiguration



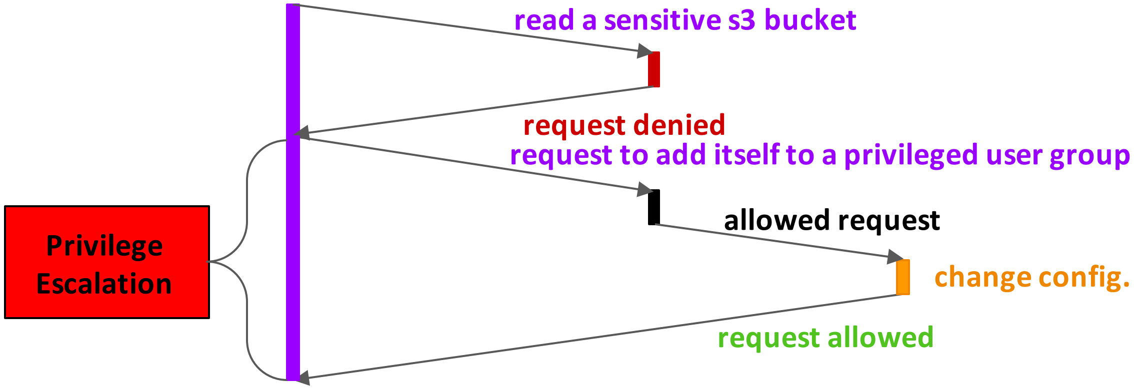
Sequence diagram:



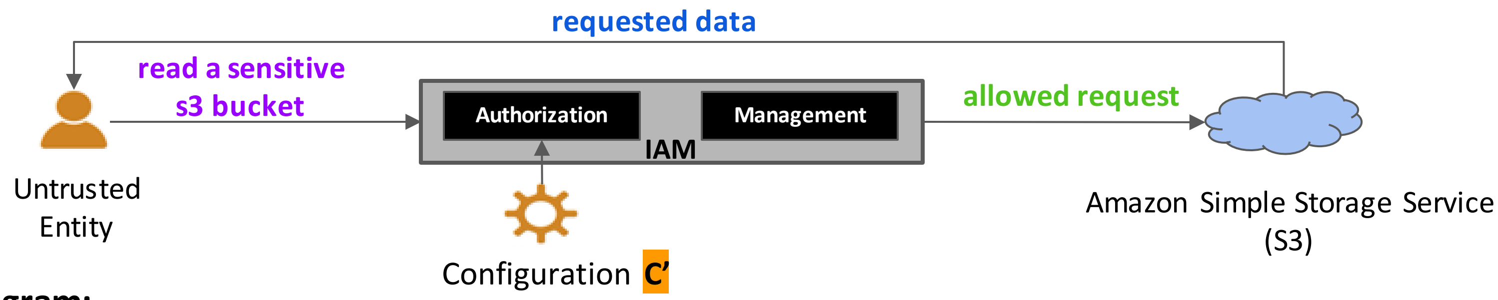
Background: Privilege Escalation (PE) in IAM Misconfiguration



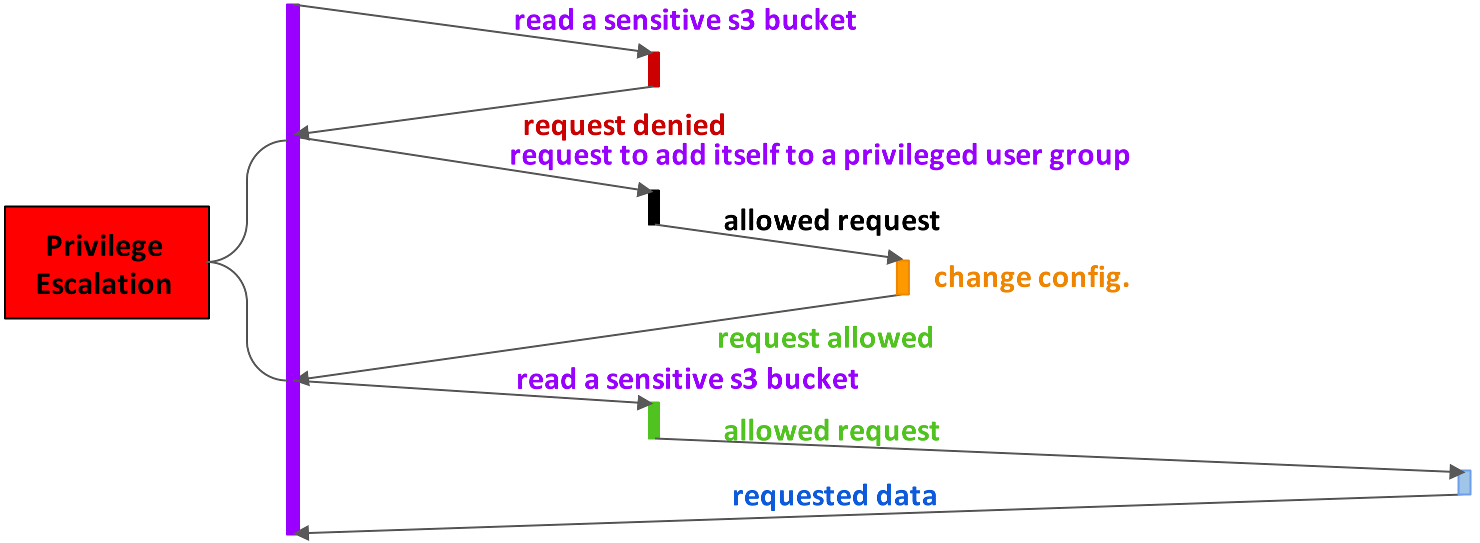
Sequence diagram:



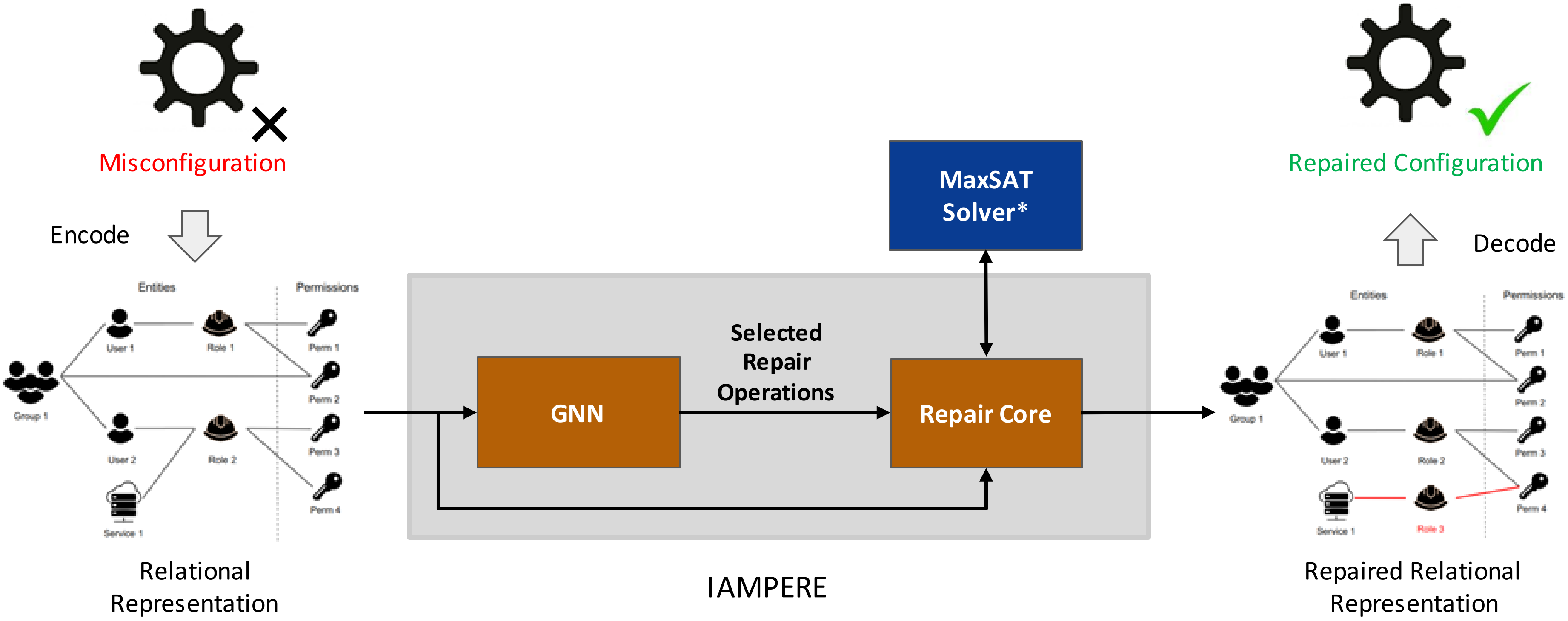
Background: Privilege Escalation (PE) in IAM Misconfiguration



Sequence diagram:

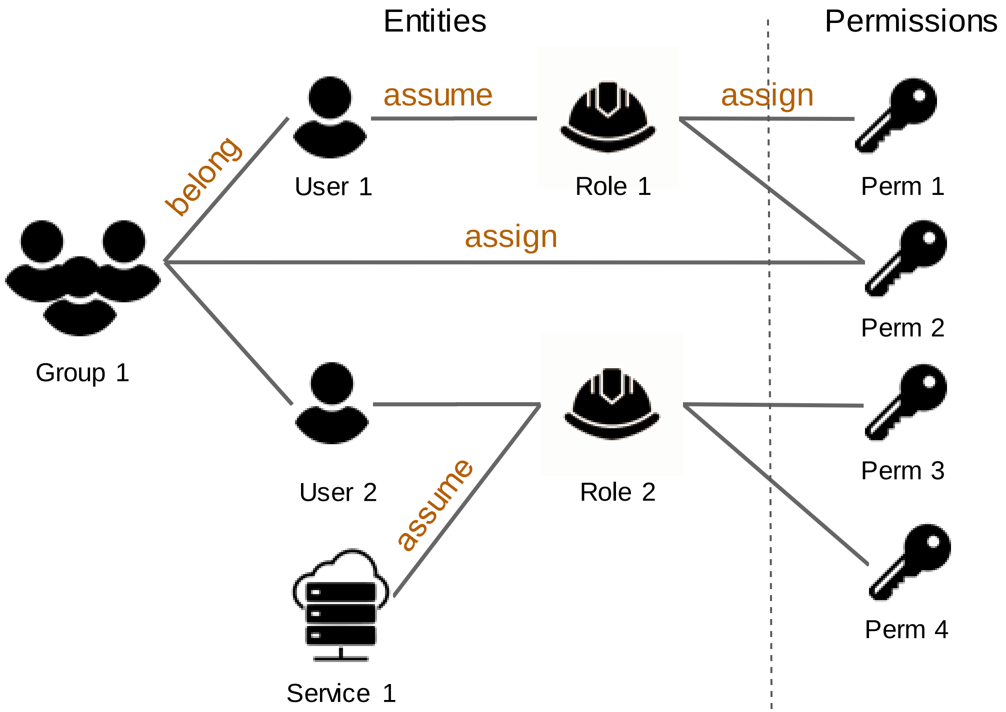


Overview: IAMPERE



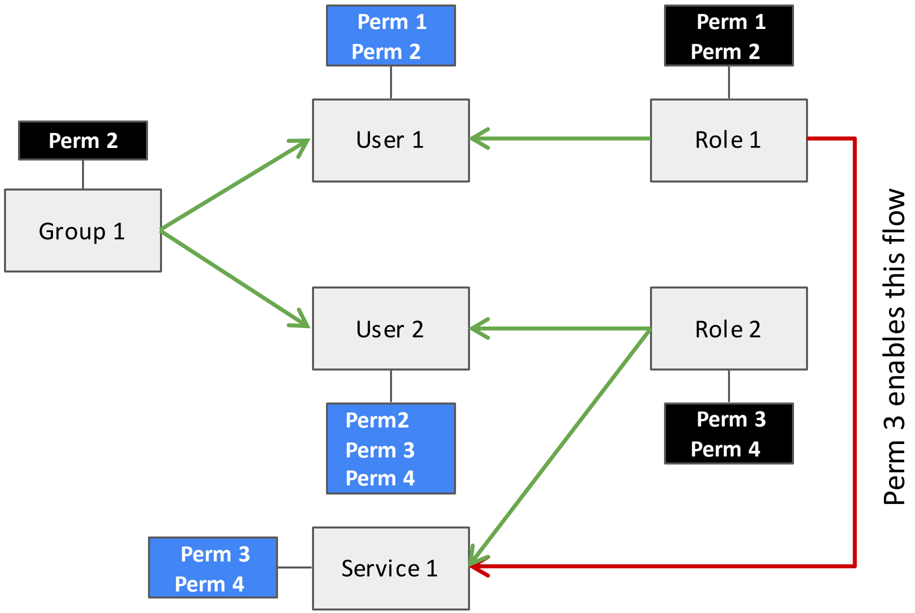
* we use CASHWMaxSAT-CorePlus solver, the winner of MaxSAT Evaluation 2022

Modeling: Relational Model of IAM Configuration [Hu+Arvix'23]



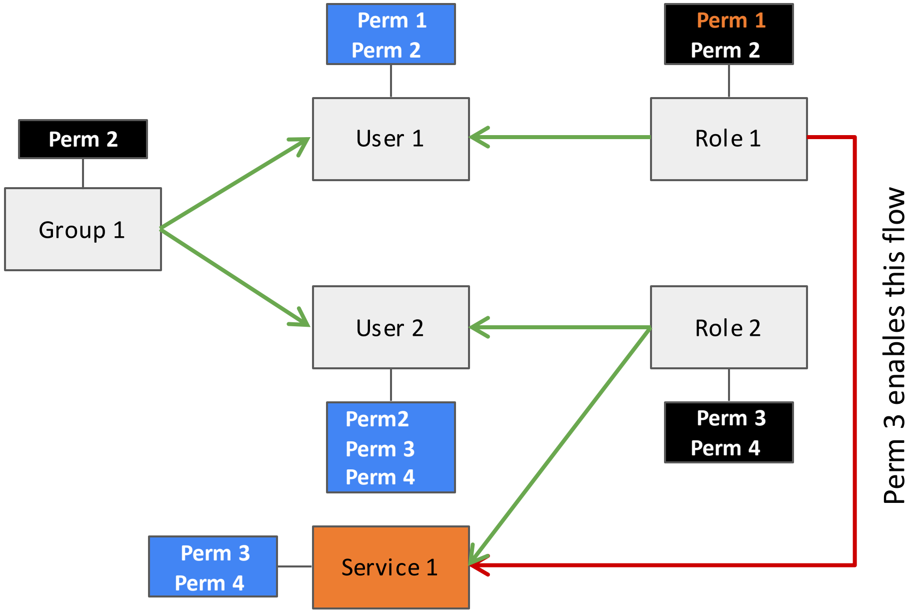
Modeling: Semantic Representation of IAM Configuration

Permission propagation via enabled permission flows



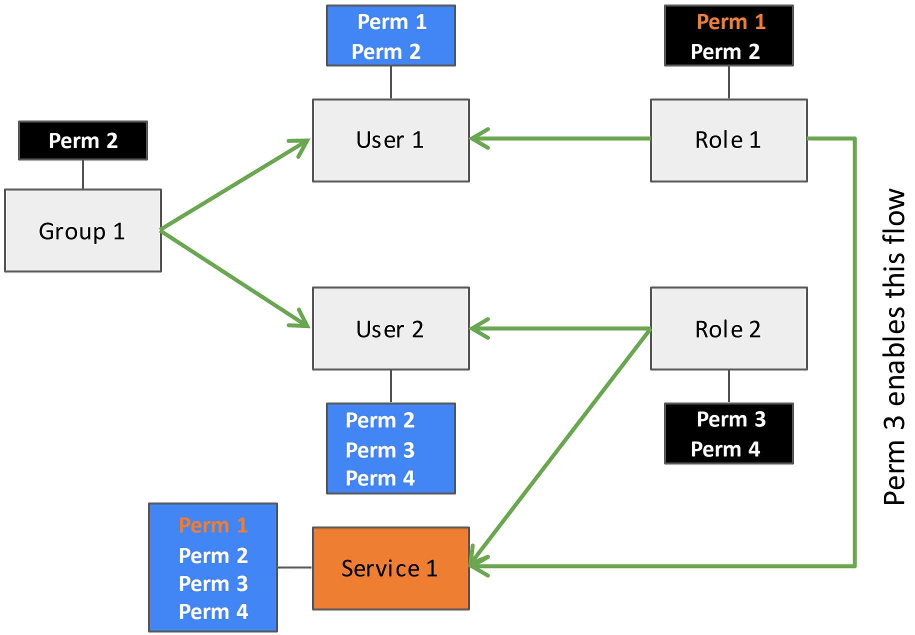
Modeling: Semantic Representation of PE

Service 1 is the untrusted entity who wants Perm 1 (target permission)



Modeling: Semantic Representation of PE

Service 1 applies Perm 3 to enable the permission flow: Role 1 -> Service 1



Our Definition: PE in IAM Misconfiguration

Definition Given a set ***E*** of untrusted entities and a set ***T*** of target permissions in an IAM configuration **C**, PE exists iff

$\exists$ ***e*** $\in$ **E**. $\exists$ ***t*** $\in$ **T**. ***e*** obtains ***t*** by applying a sequence of configuration changes.

Repair Algorithm on Semantic Representation

- Use GNN to sort repair operations based on their likelihood of being in the true minimum patch
- Generate an intermediate patch by iteratively selecting top-k repair operations
- Find a minimum patch with respect to the intermediate patch
  - Use Fixed Point Iteration (FPI) based Model Checking to compute the bound
  - Use Bounded Model Checking (BMC) based MaxSAT to generate repair for the bounded safety property

Input: an IAM misconfiguration *s*, untrusted entities *U*, target permissions *L*
Output: likely minimal repaired configuration r_{min}

```
function repair(s, U, L)
    α = gnn(s, U, L)            /*α is a list of ranked repair operations*/
    r_itm = itm_patch_gen(s, U, L, α)

    safe, bound = fpi_verify(s, U, L)
    while ¬safe do
        r_min = bmc_maxsat_repair(s, U, L, r_itm, bound)
        safe, bound = fpi_verify(r_min, U, L)
    return r_min
```

Formulation: BMC based MaxSAT Repair

SAT encoding for state transitions

$$\frac{G' = G}{m' = \neg m}\ \texttt{T1} \qquad \frac{p \in \mathcal{A}(e)}{p \in \mathcal{A}'(e)}\ \texttt{T2} \qquad \frac{\mathcal{W}(e_1, e_2)}{\mathcal{W}'(e_1, e_2)}\ \texttt{T3}$$

$$\frac{m \quad u \in U \quad p \in \mathcal{A}(u) \quad [\![p]\!] = (e_1, e_2)}{\mathcal{W}'(e_1, e_2)}\ \texttt{PT1}$$

$$\frac{m \quad \neg\mathcal{W}(e_1, e_2) \quad \forall u \in U, p \in \mathcal{A}(u).[\![p]\!] \neq (e_1, e_2)}{\neg\mathcal{W}'(e_1, e_2)}\ \texttt{PT2}$$

$$\frac{m \quad u \in U \quad p_1 \in \mathcal{A}(u) \quad [\![p_1]\!] = (e, p_2)}{p_2 \in \mathcal{A}'(e)}\ \texttt{PT3}$$

$$\frac{m \quad p_2 \notin \mathcal{A}(e) \quad \forall u \in U, p_1 \in \mathcal{A}(u).[\![p_1]\!] \neq (e, p_2)}{p_2 \notin \mathcal{A}'(e)}\ \texttt{PT4}$$

$$\frac{\neg m \quad p \in \mathcal{A}(e_1) \quad \mathcal{W}(e_1, e_2)}{p \in \mathcal{A}'(e_2)}\ \texttt{FT1}$$

$$\frac{\neg m \quad p \notin \mathcal{A}(e_2) \quad \forall e_1 \in \mathcal{N}(e_2).\mathcal{W}(e_1, e_2) \rightarrow p \notin \mathcal{A}(e_1)}{p \notin \mathcal{A}'(e_2)}\ \texttt{FT2}$$

$$\frac{\neg m \quad \neg\mathcal{W}(e_1, e_2)}{\neg\mathcal{W}'(e_1, e_2)}\ \texttt{FT3}$$

Related Work: PE Detection and Repair for IAM Config.

- PE Detection
  - Academic work
    - Reasoning based PE detector [Ilia and Oded, Usenix Security'23]
    - Greybox penetration testing for PE with reinforcement learning [Hu et al., arxiv'23]
  - Open-source tools by cloud security companies
    - Pattern based detectors: Pacu, Cloudsplaining
    - Graph based detectors: PMapper, AWSPX
- PE Repair
  - IAM-Deescalate (by Palo Alto Networks): the only existing PE repair tool
    - **Limitations**
      - Incomplete graph model: no modeling for PEs via non-authentication strategies;
      - Weak threat model: overlook transitive PEs from non-admin entities;
      - Limited repair operations: only support revoking permissions from user or roles;
      - Non-minimal patches: may remove permission assignments irrelevant to PEs;

Related Work: SE

- MaxSAT for SE
    - Typical Related Work
        - Software Fault Localization: BugAssist [Jose and Majumdar, CAV’11]
        - Program Repair: DirectFix [Mechtaev et al., ICSE’15]
        - Software Models: AlloyMax [Zhang et al., ESEC/FSE’21]
    - **Limitation:** completely dependent on MaxSAT solving ability
- Repair in SE
    - Active research area with large body of work

Evaluation

Experimental Setup:

- Benchmarks:
  - ✓ Two real-world IAM misconfigurations with PE, owned by cloud customers from a security startup
  - ✓ 31 publicly available IAM misconfigurations with PE
  - ✓ 1,000 randomly synthesized IAM misconfigurations with PE using IAMVulGen [Hu+Arxiv'23]
- Our Tool:
  - ✓ IAMPERE: using both GNN and the MaxSAT solver to generate close to minimum patch
    - MaxSAT solver: CASHWMaxSAT-CorePlus solver, the winner of MaxSAT Evaluation 2022
- Baselines:
  - ✓ IAM-Deescalate: existing PE repair tool
  - ✓ IAMPERE-GO: only using GNN with iterative deepening to generate a repair
  - ✓ IAMPERE-MO: only using the MaxSAT solver to generate a repair
- Metrics:
  - ✓ Effectiveness: relative patch size = patch size / max patch size
  - ✓ Efficiency: time cost
  - ✓ Validity: relative patch size < 1

Evaluation on Two Real-World Misconfigurations

Time cost and relative patch size (in brackets)
Timeout: 7,200 seconds

Config.	#entities	# perms	IAM-Deescalate	IAMPERE-GO	IAMPERE-MO	IAMPERE
Real-1	251	2,826	**T.O.**	**5,147s (0.889)**	**T.O.**	**--**
Real-2	158	882	**T.O.**	2,107s (0.741)	3,963s **(0.0048)**	**1,190s (0.0048)**

Evaluation on 31 Publicly Available Misconfigurations

- Statistics:
  - # entities: ≤ 3
  - # perms: ≤ 6
- Timeout: 10 seconds
- Repair rate: all 31 misconfigurations are repaired by both IAMPERE and its variants, while 24 misconfigurations are repaired by IAM-Deescalate.
- Patch size: all repairs are minimal
- Time cost: less than 5 seconds per repair.

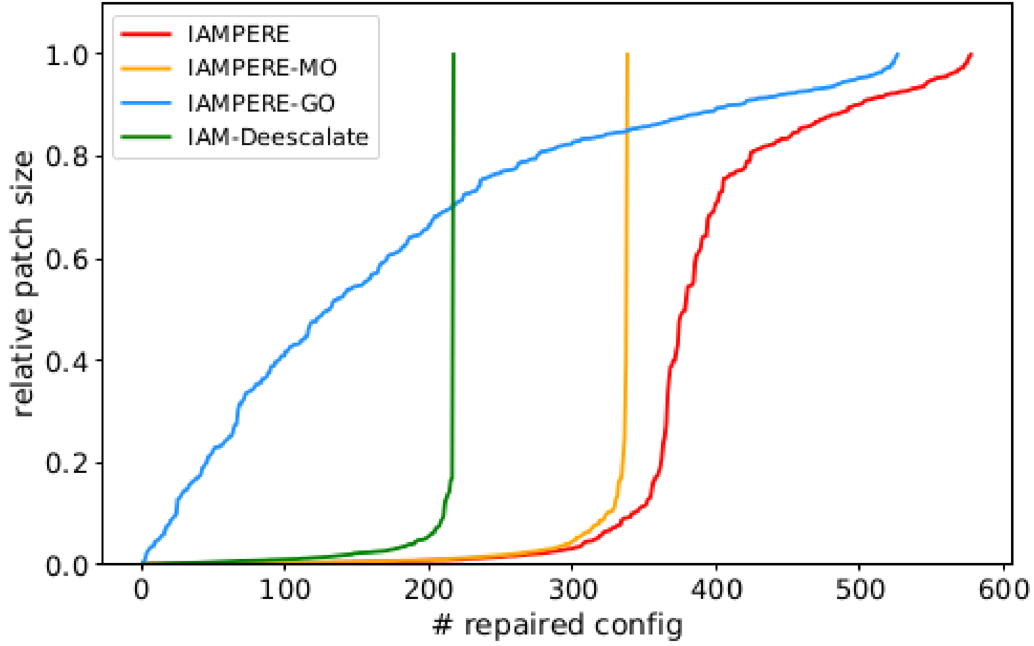
Evaluation on 1,000 Synthesized Configurations

Statistics

- # entities: 11 - 315
- # perms: 42 - 11,737

Evaluation on 1,000 Synthesized Configurations

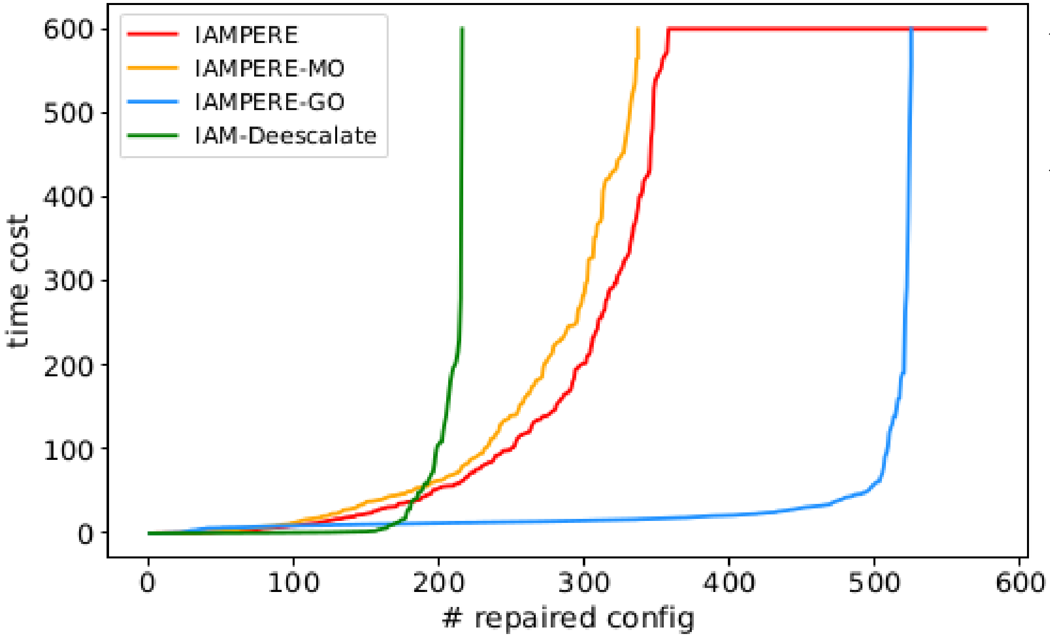
Effectiveness:
the number of IAM configurations repaired by each tool within a specific relative patch size.
Timeout: 600s



✓ IAMPERE not only repairs more configurations but also produces more small patches.

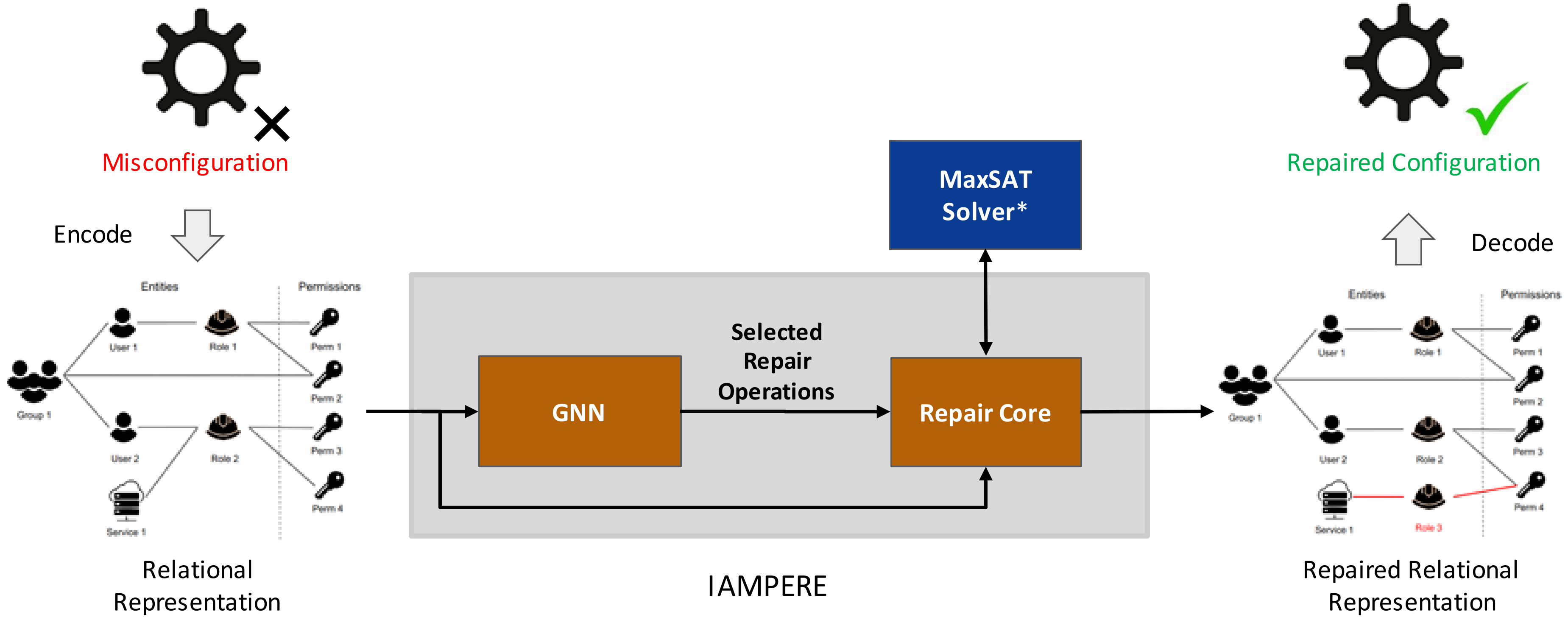
Evaluation on 1,000 Synthesized Configurations

Efficiency:
the number of IAM configurations repaired by each tool within a specific time cost.
Timeout: 600s



- ✓ IAM-Deescalate is significantly outperformed by IAMPERE and its variants.
- ✓ IAMPERE is consistently more efficient than IAMPERE-MO, fixing 220 more misconfigurations

Summary



{huyang,wenxiw,khurshid}@utexas.edu, kenmcm@cs.utexas.edu, tiwari@ece.utexas.edu

* we use CASHWMaxSAT-CorePlus solver, the winner of MaxSAT Evaluation 2022