NEUROBACK: IMPROVING CDCL SAT SOLVING USING GRAPH NEURAL NETWORKS

WENXI WANG, YANG HU, MOHIT TIWARI, SARFRAZ KHURSHID, KEN MCMILLAN, RISTO MIIKKULAINEN

PRESENTED BY

RISHOV PAUL

BACKGROUND: SAT SOLVING

SAT solving refers to the process of determining whether there exists an assignment of values (true/false) to variables such that a given Boolean formula in *Conjunctive Normal Form (CNF)* evaluates to true

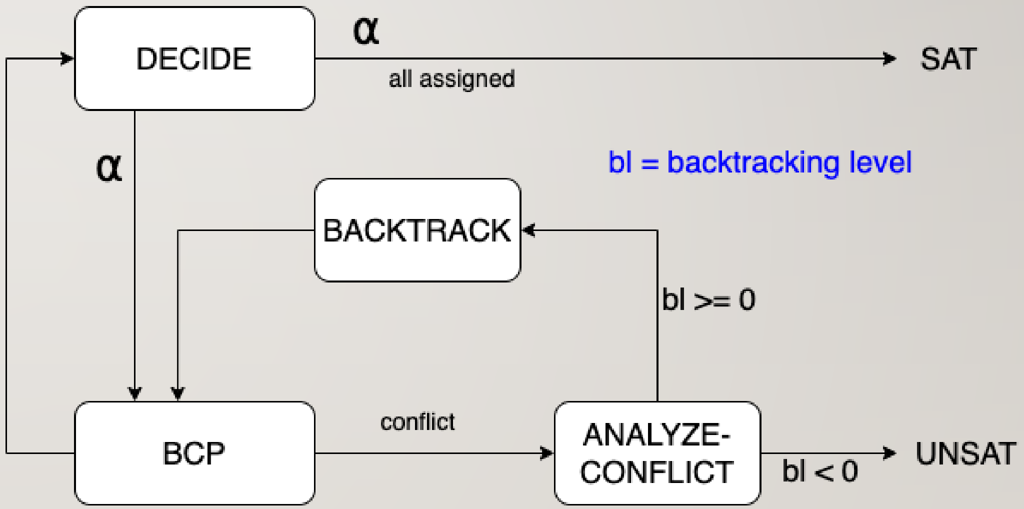
$\varphi = (x1 \lor \neg\, x2\ ) \land (x2 \lor x3) \land x2$

Clauses: $c1 = x1 \lor \neg\, x2$, $c2 = x2 \lor x3$, $c3 = x2$

x1=True x2=True x3=False

BACKGROUND: CDCL SAT SOLVING

- Mainly relies on two kinds of variable related heuristics
  - Variable branching heuristic
  - Phase selection heuristic



BACKGROUND: BACKBONE VARIABLE

- Backbone variables are the variables whose phases remain consistent across all satisfying assignments.

$\varphi = (x1 \lor \neg\, x2\,) \land (x2 \lor x3) \land x2$

- Possible satisfying assignments:
  - x1=True, x2=True, x3=False
  - x1=True, x2=True, x3=True

  two backbone variables, x1 and x2

BACKGROUND: GNN

- GNN is a family of neural network architecture that operate on graphs
- GNN Components

  1. **Node Embeddings**: These are feature vectors that represent the properties of each node
  2. **Message Passing**: At each layer, each node gathers information from its neighbors
  3. **Aggregation Function**: A function(such as mean or sum) that aggregates the messages from neighbours
  4. **Update Function**: Once the node aggregates messages, it updates its feature vector based on this new information

BACKGROUND: GRAPH TRANSFORMER

- Transformers process sequential data (text, images) using self-attention to capture long-range dependencies.
- Combining Transformers with GNN forms the Graph Transformer architecture, excelling in graph and node classification tasks
- GraphTrans model uses multiple GNN layers for local structure encoding, Transformer layers for global self-attention, and an FFN for classification.

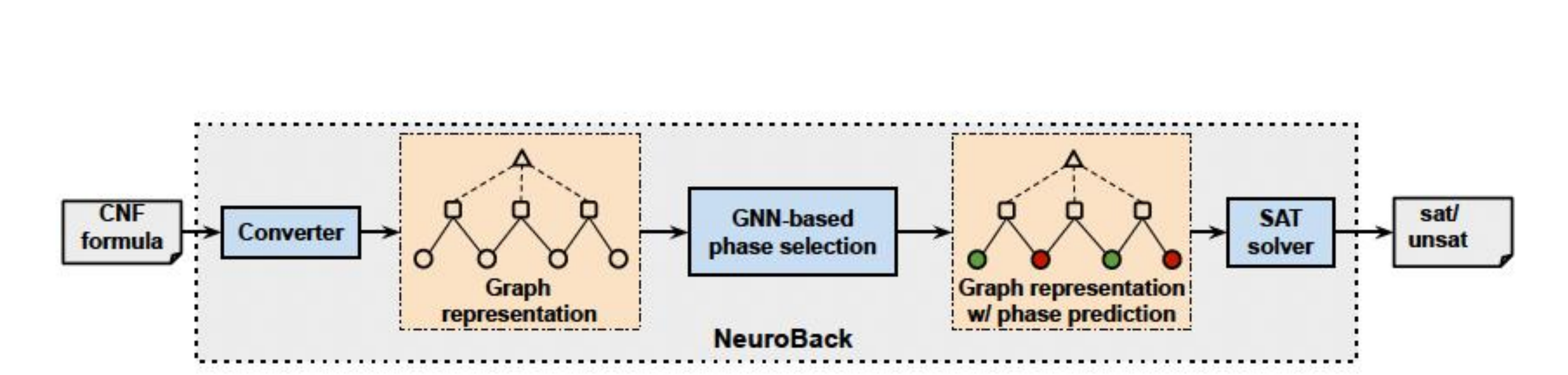
RELATED WORKS

- Wu (2017) applied logistic regression to predict backbone phases but did not improve MiniSat's solving time
- Recent works (Biere et al., 2021; Al-Yahya et al., 2022) focus on using heuristic search to partially compute the backbone during CDCL solving
- NeuroSAT (Selsam et al., 2018) introduced neural models for SAT solving, but with limited effectiveness for large-scale problems
- NeuroCore (Selsam & Bjørner, 2019) enhances CDCL branching heuristics via online inference

INTRODUCTION TO NEUROBACK

- NeuroBack employs offline model predictions on variable phases
- It executes solely on CPU
- Independent of GPU resources
- It enhances the phase selection heuristics in CDCL solvers
- Applies a GNN model, trained solely on predicting the phases of backbone variables, to predict the phases of all variables

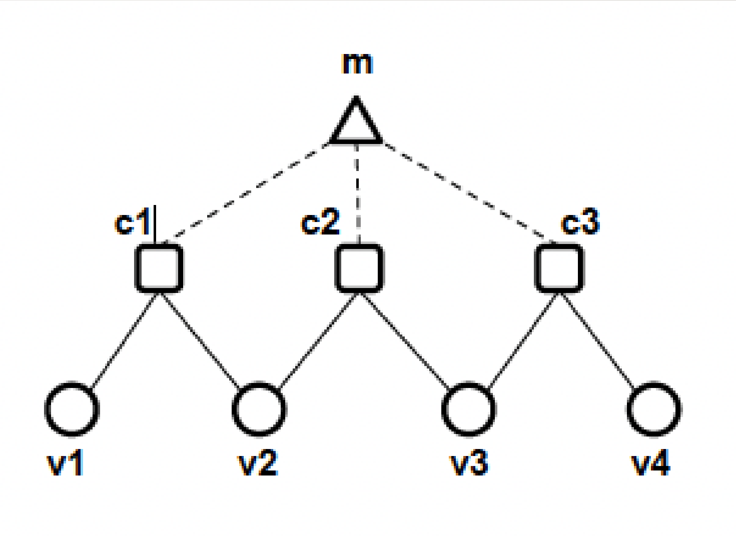
OVERVIEW OF NEUROBACK WORKFLOW



GRAPH REPRESENTATION OF CNF FORMULA

$\varphi = (v1 \lor v2) \land (v2 \lor v3) \land (v3 \lor v4)$

- Two node types represent the variables and clauses
- Each edge connects a variable node to a clause node
- Meta node(m) for each connected component in the graph, with meta edges connecting the meta node to all clause nodes in the component

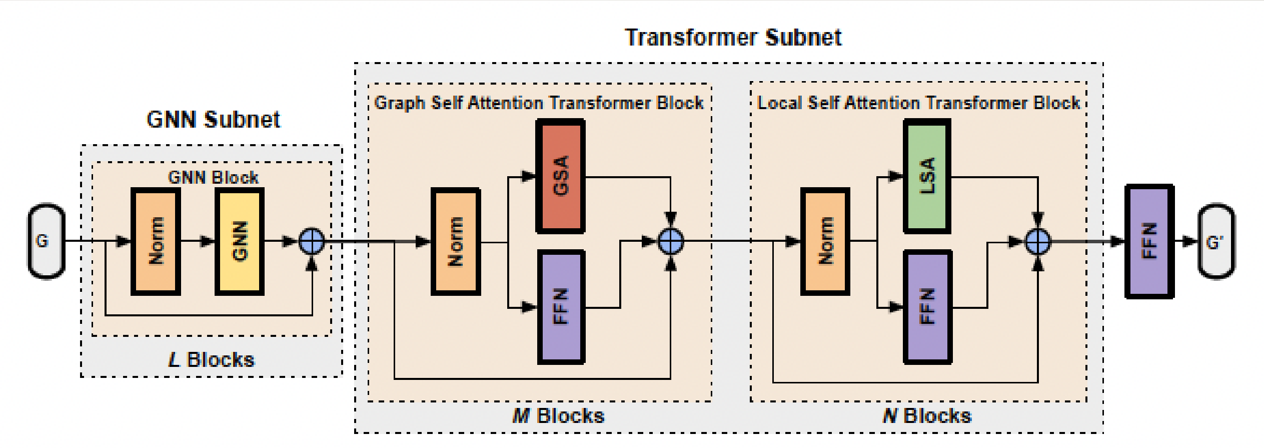


GNN MODEL DESIGN

- Inspired by graph transformer architecture, GraphTrans
- GraphTrans has two limitations
  - It does not explicitly integrate topological graph structure when calculating attention scores
  - Global self-attention mechanism computes attention scores for all possible node pairs, leading to quadratic memory complexity
- Our novel transformer combines GSA and LSA replacing the global self attention of original transformer
- GSA calculates attention scores solely for directly connected node pairs, leveraging information of edges and edge weights
- LSA segments each node embedding into multiple node patches and computes attention scores for each pair of node patches
- Linear memory complexity in terms of the number of edges and nodes in the graph

GNN MODEL ARCHITECTURE

- Three main components
- Each transformer block has a normalization layer, followed by FFN and GSA/LSA layers to enhance training efficiency



DATABACK DATASET DESCRIPTION

- DataBack is a dataset of SAT CNF formulas labelled with backbone variable phases, for pre-training and fine-tuning the NeuroBack model
- Two sets:
  - DataBack-PT
  - DataBack-FT
- Cadiback is used to extract the backbone label
  - Label collection timeout for PT: 1000 seconds
  - Label collection timeout for FT: 5000 seconds
- Significant label imbalance in both DataBack-PT and DataBack-FT

DATA AUGMENTATION STRATEGY

- Original formula: $\varphi = (x1 \lor \neg x2) \land (x2 \lor x3) \land x2$
  - Backbone variable {x1, x2} → {True}
- Create a dual formula by negating all backbone variables in the original formula
- Dual formula: $\varphi' = (\neg x1 \lor x2) \land (\neg x2 \lor x3) \land \neg x2$
  - Backbone variable {x1, x2} → {False} opposite phase

DataBack-PT	CNFgen	SATLIB	MCCOMP	SATCOMP (random)	Overall
# CNF	21,718	80,306	11,560	4,876	118,460
# Var	779	114	16,299	25,310	2,852
# Cla	281,888	529	63,501	88,978	61,898
# BackboneVar	280 (36%)	58 (51%)	8,587 (53%)	1,960 (8%)	1,009 (35%)

DataBack-FT	SATCOMP (main)
# CNF	1,826
# Var	206,470
# Cla	1,218,519
# BackboneVar	48,266 (23%)

MODEL PRE-TRAINING AND FINE-TUNING

- **Loss function:** Binary cross entropy (BCE)
- **Optimizer:** AdamW optimizer
- **Learning rate:** 10^-4
- **Number of epoch**
  - Pre-training – 40
  - Fine-tuning – 60

APPLICATION OF PREDICTIONS

- Leverage phase predictions from GNN model to improve phase selection heuristics
- Use Kissat solver (Biere & Fleury, 2022) for phase initialization with NeuroBack predictions
- Resulting implementation is called NeuroBack-Kissat.

NEUROBACK MODEL PERFORMANCE

Model/metrics	Precision	Recall	F1	Accuracy
pre-trained model	0.903	0.766	0.829	0.751
fine-tuned model	0.941	0.914	0.928	0.887

- NeuroBack model pre-trained on the entire DataBack-PT dataset.
- Fine-tuned on 90% of DataBack-FT samples, with 10% used as validation set.
- Results indicate pre-training helps the model extract generalized knowledge about backbone phase prediction.
- Fine-tuning improved performance by 4% to 15% across all metrics.
- Final precision, recall, and F1 score all exceeded 90%.
- NeuroBack effectively learns to predict backbone phases through pre-training and fine-tuning.

NEUROBACK SOLVING EFFECTIVENESS

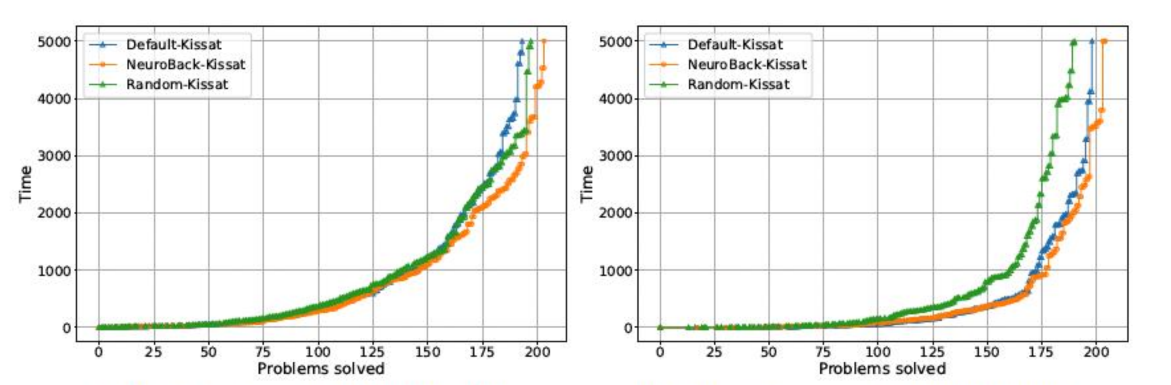
- **Testing dataset:** 800 CNF formulas from the main track of SATCOMP-2022 and SATCOMP-2023
- Baseline solvers
  - **Default-Kissat:** Sets the initial phase of each variable to true
  - **Random-Kissat:** randomly assigns the initial phase of each variable as either true or false
- **Solving time limit:** 5000 seconds

- Model inference for each NeuroBack solver was conducted solely on the CPU, with a memory limit of 10GB
- 308 problems from SATCOMP-2022 and 353 problems from SATCOMP-2023 were successfully inferred
- The CPU inference time for each of these problems ranged from 0.3 to 16.5 seconds, averaging at 1.7 seconds

Model	Inferred Problems (2022)	Solved Problems (2022)	Inferred Problems (2023)	Solved Problems (2023)
Default-Kissat	308	193	353	198
Random-Kissat	308	197	353	190
NeuroBack-Kissat	308	203	353	204

NEUROBACK PERFORMANCE ON TESTING TEST

- NeuroBack-Kissat consistently outperforms both Default-Kissat and Random-Kissat
- Outperforms Default-Kissat on 43 and 40 more problems in SATCOMP-2022 and 2023, reducing solving time by 117 and 36 seconds per problem
- Outperforms Random-Kissat on 22 and 29 more problems in SATCOMP-2022 and 2023, reducing solving time by 98 and 246 seconds per problem
- NeuroBack phase initialization outperforms both default and random in Kissat, showing improved performance in solving SATCOMP-2022 and 2023 problems



(a) Solving progress in SATCOMP-2022

(b) Solving progress in SATCOMP-2023

SUMMARY

- NeuroBack improves CDCL SAT solvers without needing GPU resources during application
- Uses offline model inference on variable phases from satisfying assignments to enhance phase selection heuristics
- Integrated with the Kissat solver, it significantly reduces solving time and increases the number of solved instances in SATCOMP-2022 and 2023
- NeuroBack demonstrates potential in boosting SAT solvers with machine learning

FIXING PRIVILEGE ESCALATIONS IN CLOUD ACCESS CONTROL WITH MAXSAT AND GRAPH NEURAL NETWORKS

YANG HU , WENXI WANG , SARFRAZ KHURSHID, KENNETH L. MCMILLAN, MOHIT TIWARI

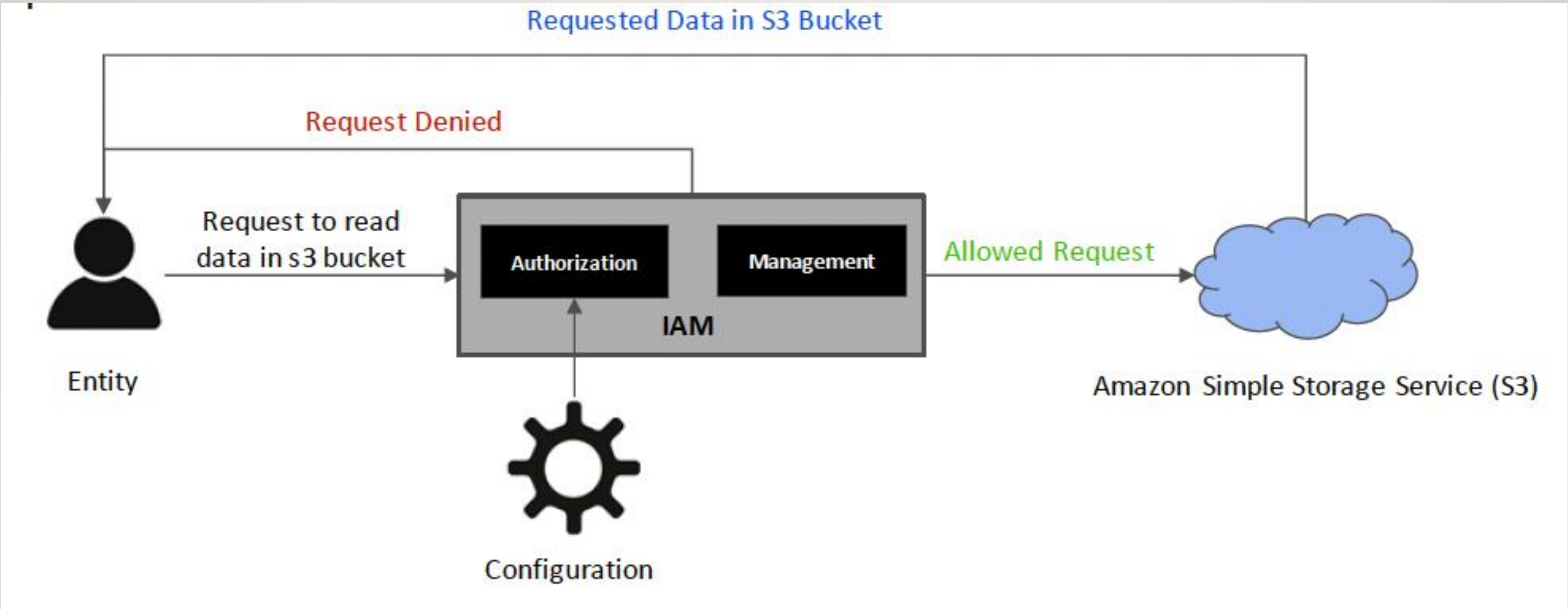
PRESENTED BY

RISHOV PAUL

BACKGROUND: IAM

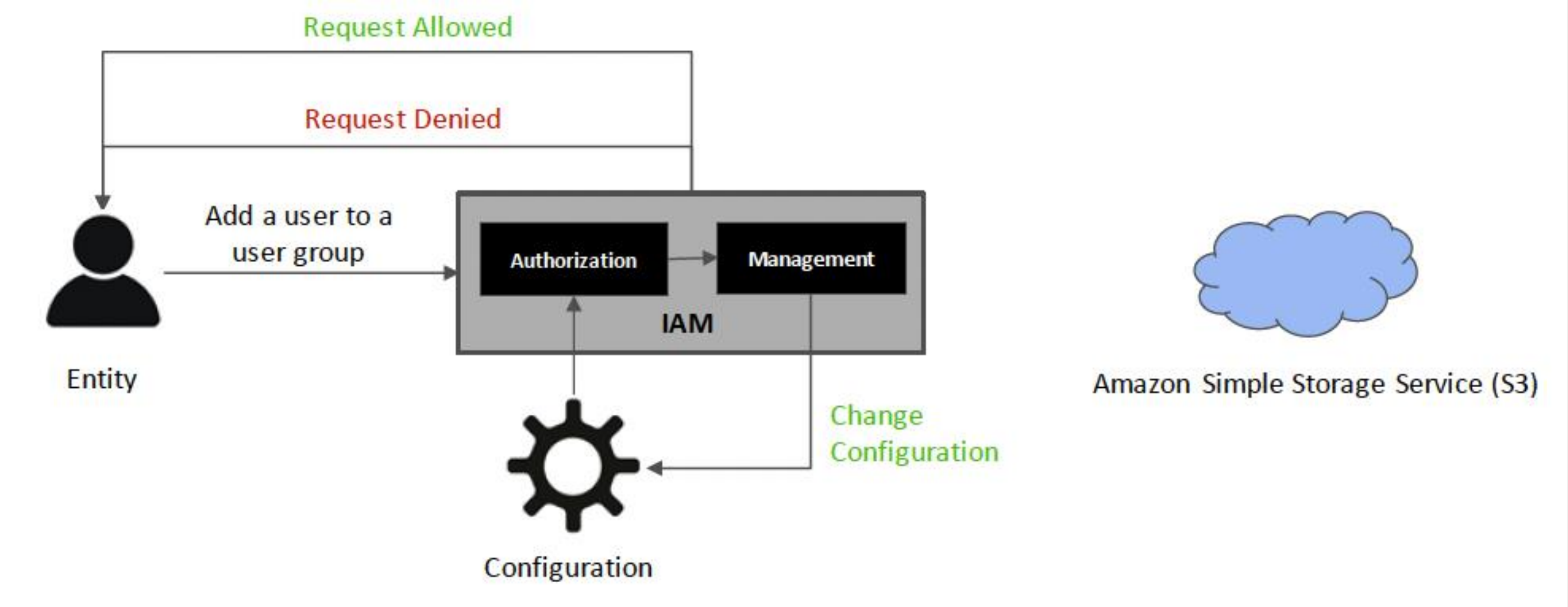
IAM (Identity and Access Management) is an access control service in cloud platform

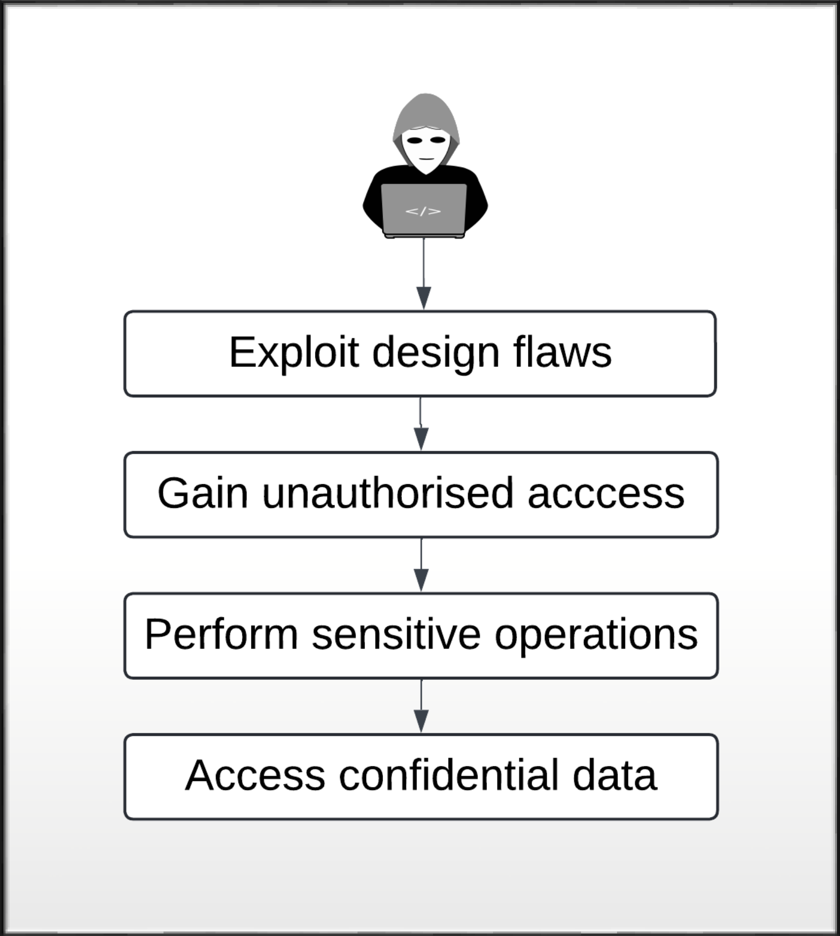
IAM EXAMPLE



Requested Data in S3 Bucket
Request Denied
Request to read
data in s3 bucket
Authorization
Management
IAM
Allowed Request
Entity
Amazon Simple Storage Service (S3)
Configuration

IAM ANOTHER EXAMPLE: CONFIGURATION CHANGE





BACKGROUND: IAM MISCONFIGURATION

CAN LEAD TO PRIVILEGE ESCALATIONS!

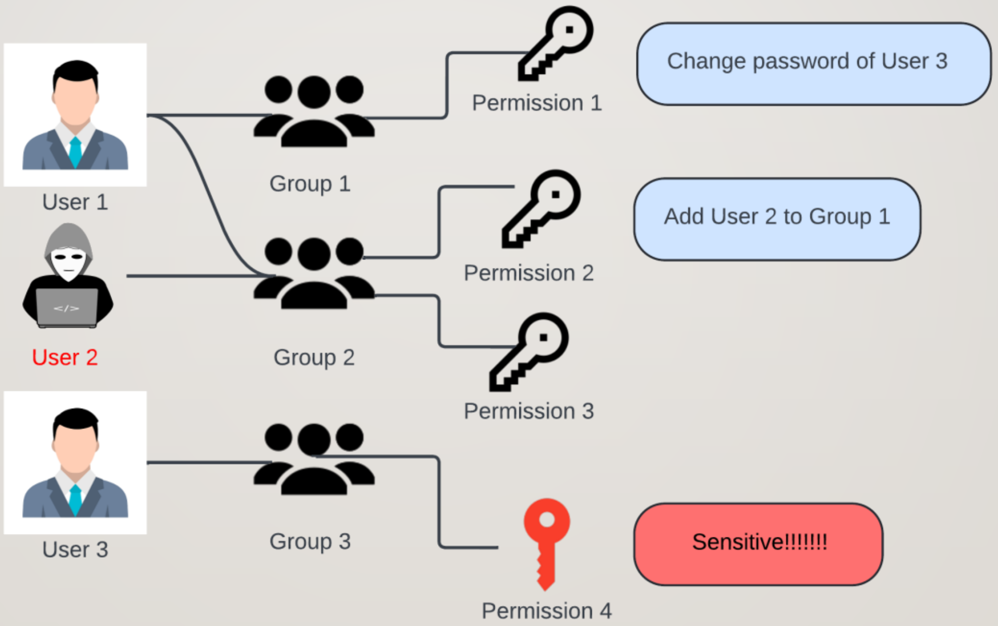
PROBLEM DEFINITION: PRIVILEGE ESCALATION

Given a set E of untrusted entities and a set P of sensitive permissions in an IAM configuration C, PE exists iff

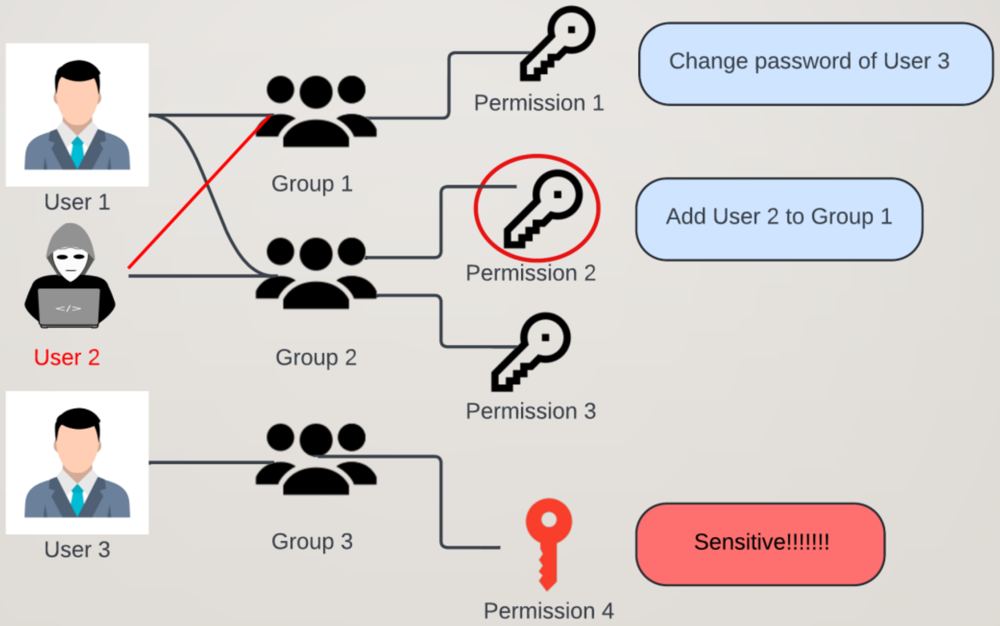
$\exists\, e \in E.\ \exists\, p \in P.$ e obtains p by applying a sequence of permissions

An IAM configuration is safe if no PE exists

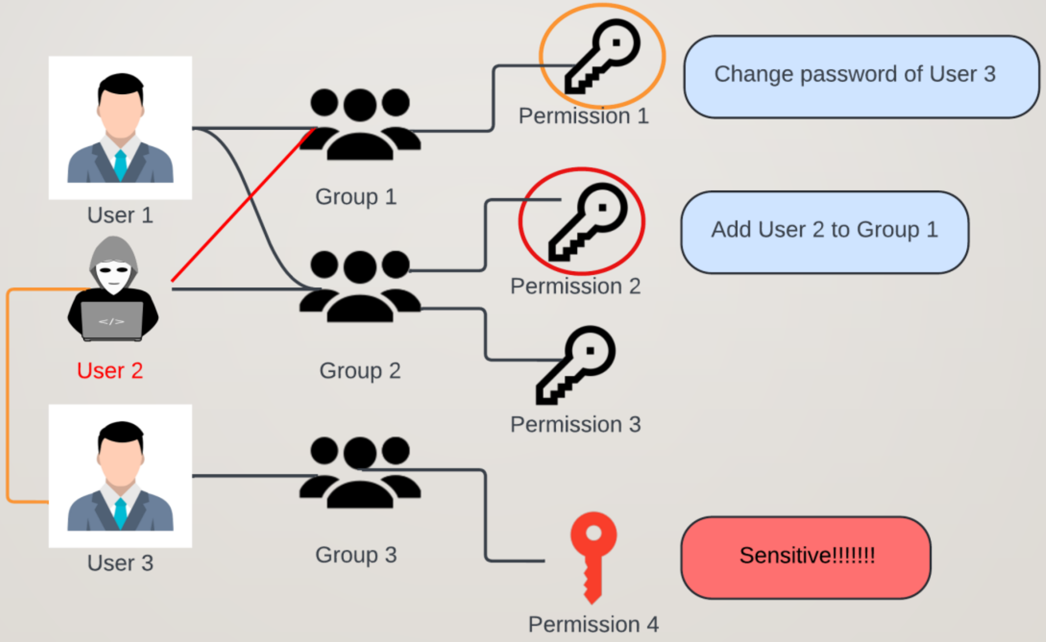
A SIMPLE EXAMPLE



A SIMPLE EXAMPLE CONTD



A SIMPLE EXAMPLE CONTD



GOAL

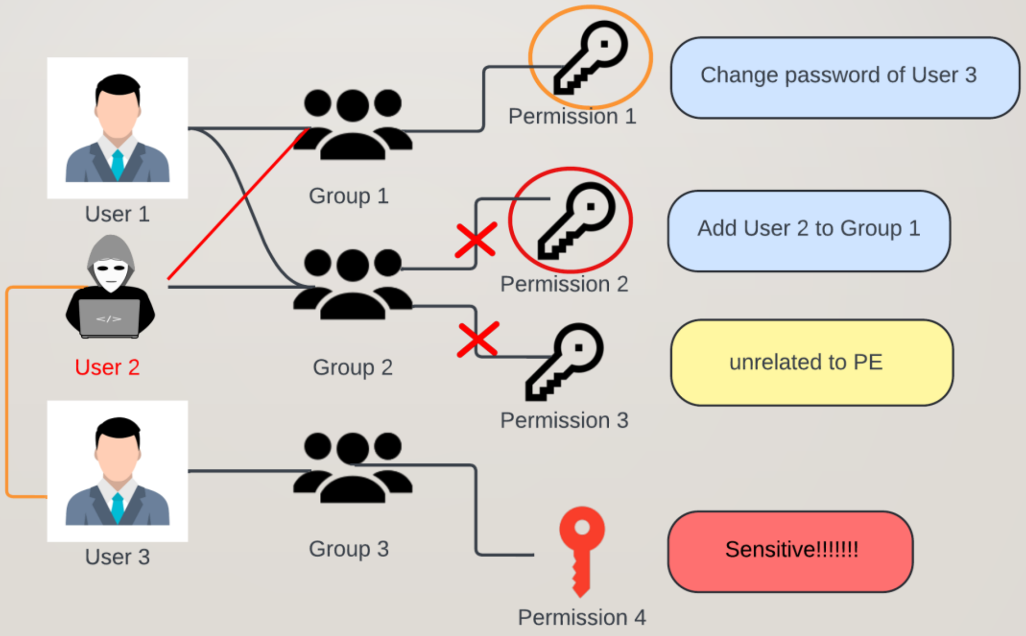
Repair IAM misconfigurations to prevent PEs

PROBLEM DEFINITION: IAM REPAIR GOAL

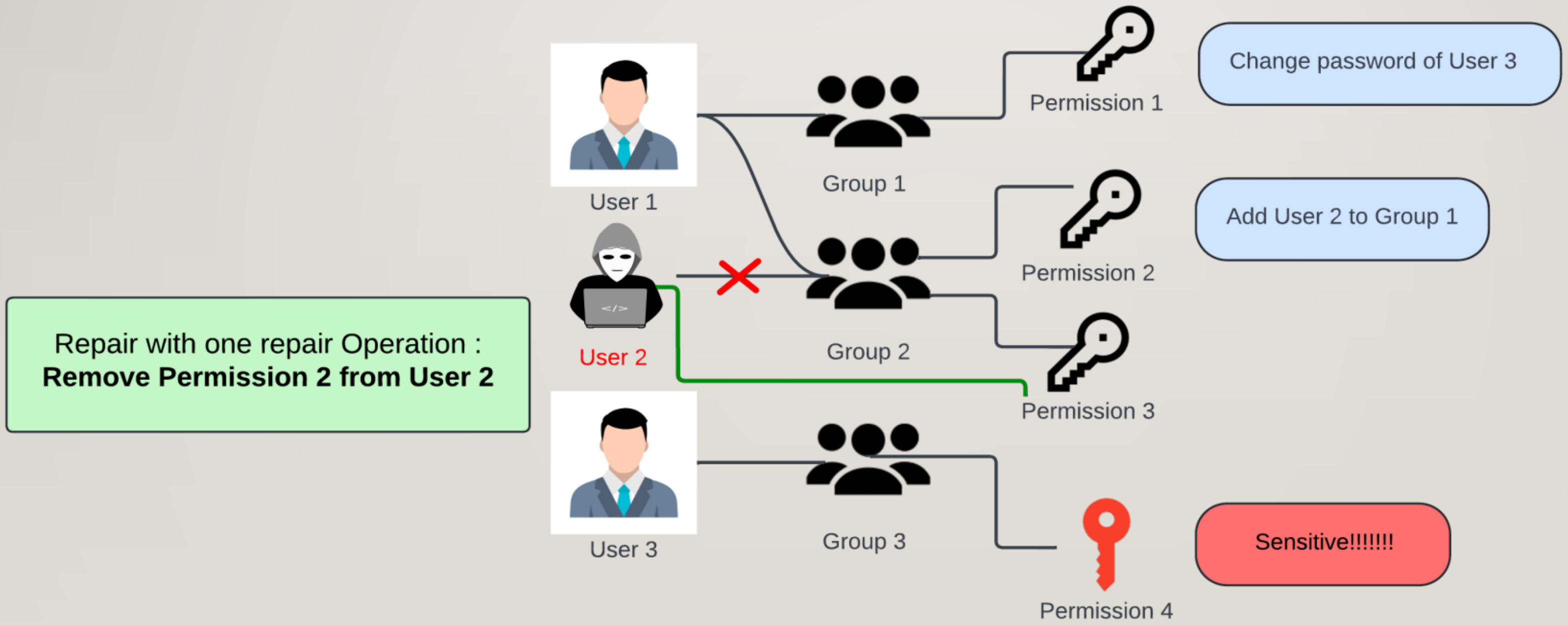
- r_{min} = argmin_r $|r|$ s.t. R(c, r) is safe
- c: IAM misconfiguration
- r: a list repair operations
- |r|: patch size
- R(c, r): apply r on c

VALID BUT NON-MINIMAL IAM REPAIR

Repair with two repair Operations :
Remove Permission 2 from User 2
Remove Permission 3 from User 2



MINIMAL IAM REPAIR



Maintains maximum permission assignments

CHALLENGE

- Real-world IAM configurations are very complicated
- Lots of entities, permissions and connections among them

MAIN GOAL

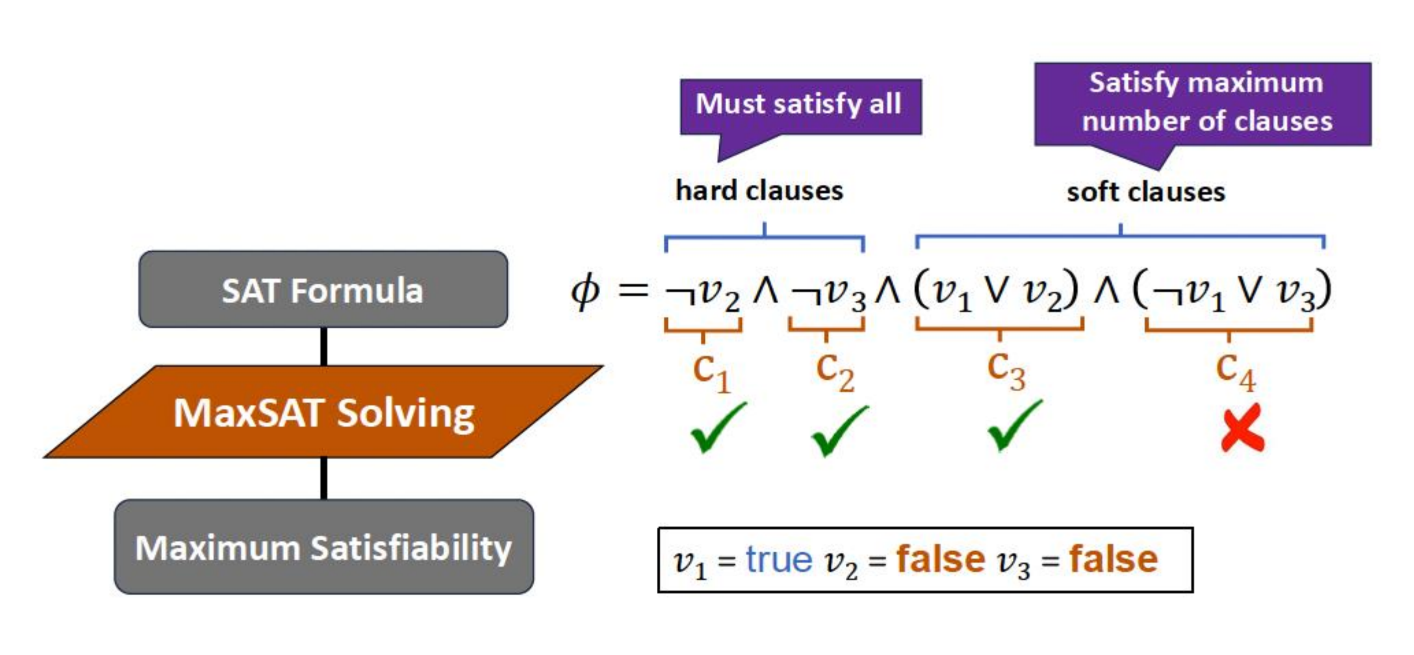
Automatically repair IAM misconfigurations to prevent PE

INSIGHT

Using MaxSAT to automatically produce the

minimal IAM repair

BACKGROUND: MAXSAT



REPAIR WAY - MAXSAT

- Model the IAM repair problem into MaxSAT problem
  - **Hard clauses**
    - Safety verification
    - Untrusted entities can never gain sensitive permissions
  - **Soft clauses**
    - Repair operations
    - Maintain maximum permission assignments

MAXSAT ISSUE

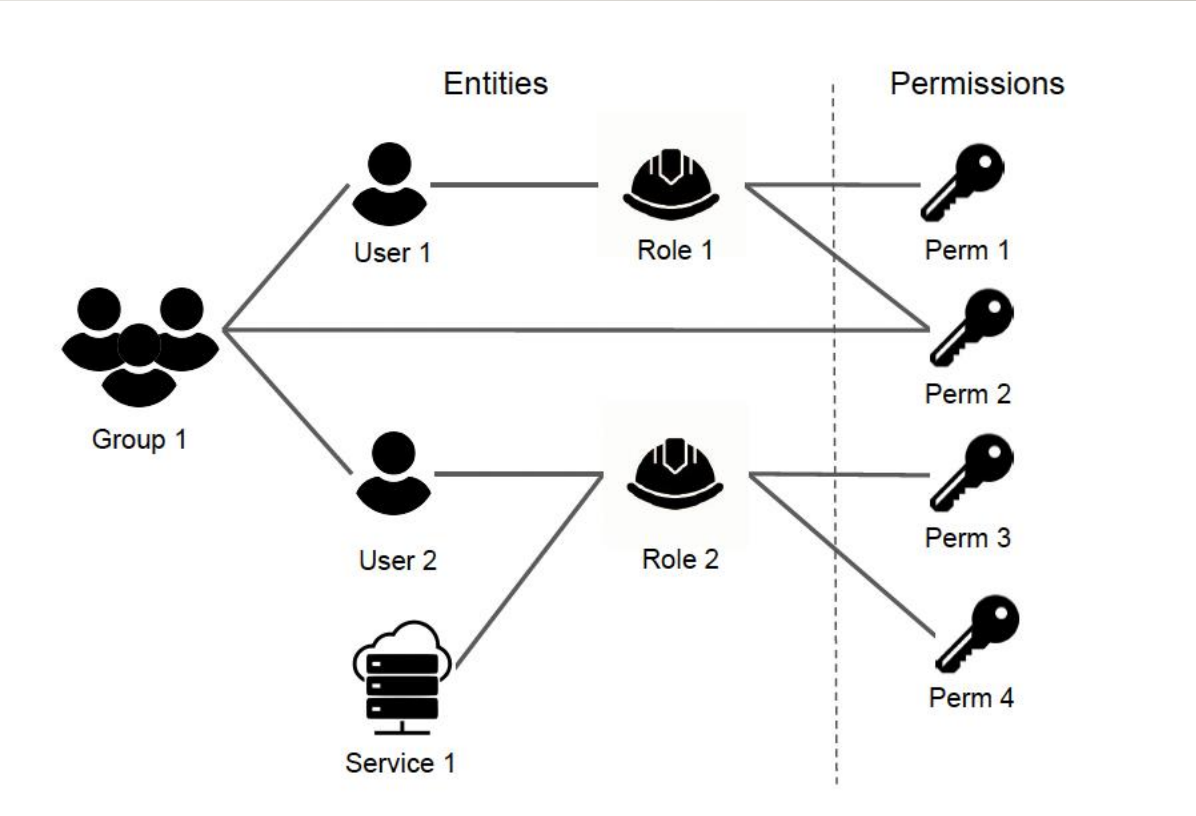
- **But there is scalability issue here**
  - Directly encoding a finite state to cover all possible states can result in a **massive number of propositional constraints.**
  - This approach poses significant challenges when verifying repairs at scale.

OPTIMIZED APPROACH: IAMPERE

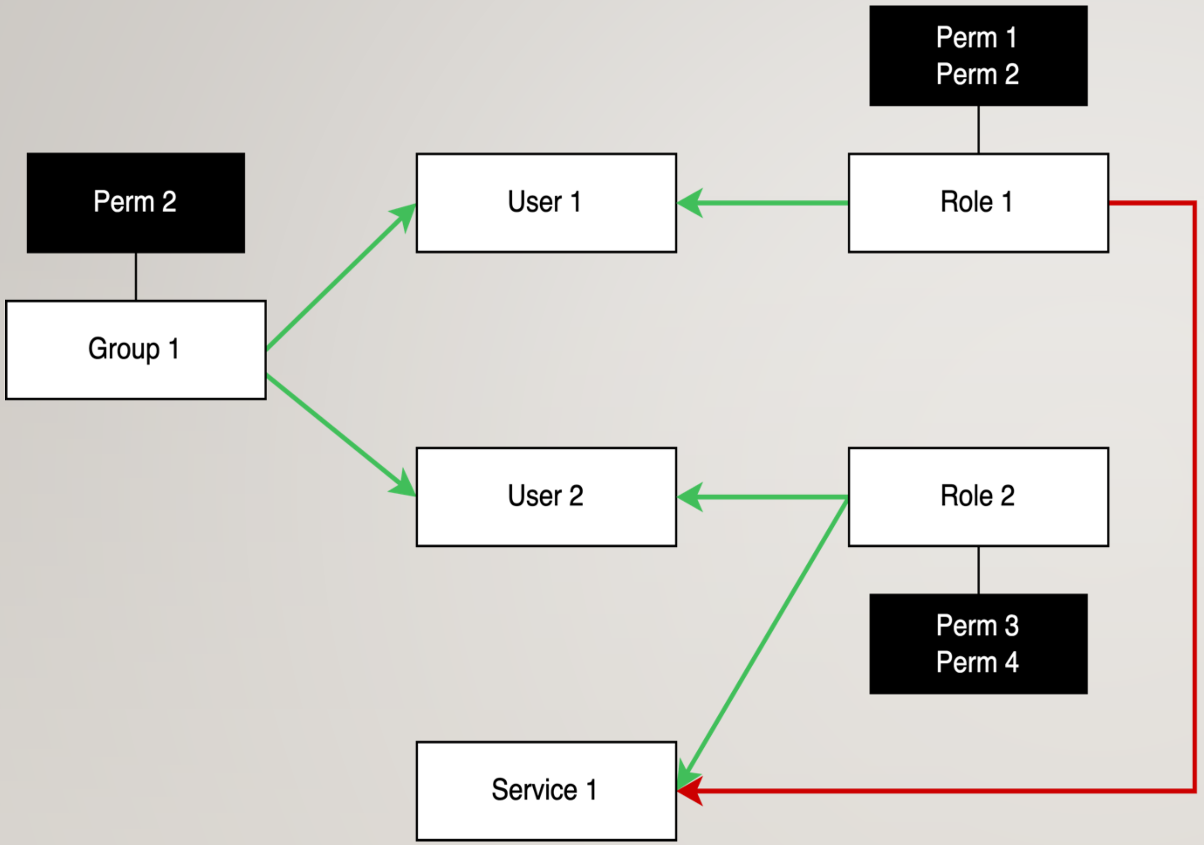
- Improves MaxSAT scalability with GNN
- Prune search space for the MaxSAT solver by employing deep learning
- IAM configurations are made graph structured
- Consists of two phases
  - Training phase
  - Testing phase

RELATIONAL MODEL OF IAM MISCONFIGURATION

Real world IAM PE in 2019



PERMISSION FLOW GRAPH



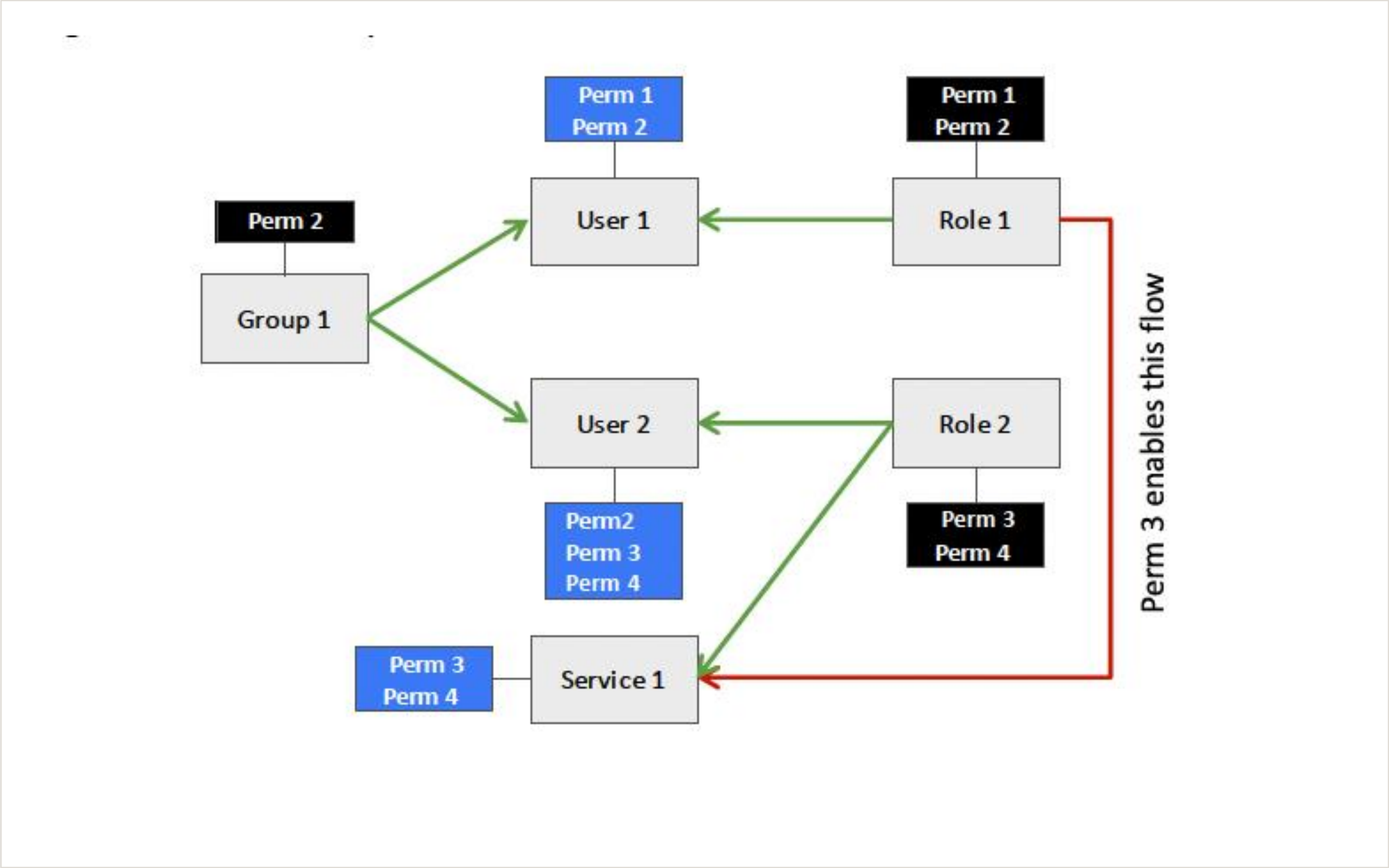
A PFG is proposed to represent

- How permissions are directly or indirectly assigned to entities
- It includes entities as its nodes (annotated with permissions assigned to the entities)
- permission flows as its edges.

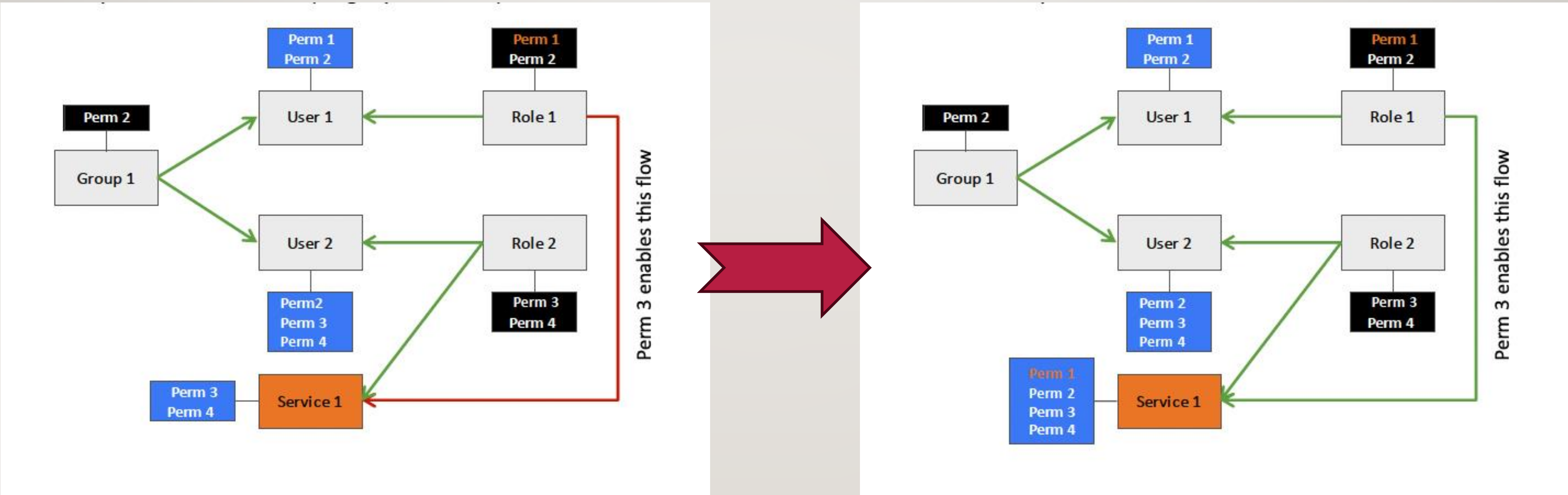
- Perm 3 allows Service 1 to assume Role 1
- A disabled permission flow edge is added from Role 1 to Service 1
- The disabled edge can be enabled in the future if the compromised entity applies Perm 3 to assume Role 1

MODELING: SEMANTIC REPRESENTATION OF IAM CONFIGURATION

directly assigned permissions are indirectly assigned to the corresponding entities according to the enabled permission flows



MODELING: SEMANTIC REPRESENTATION OF PE

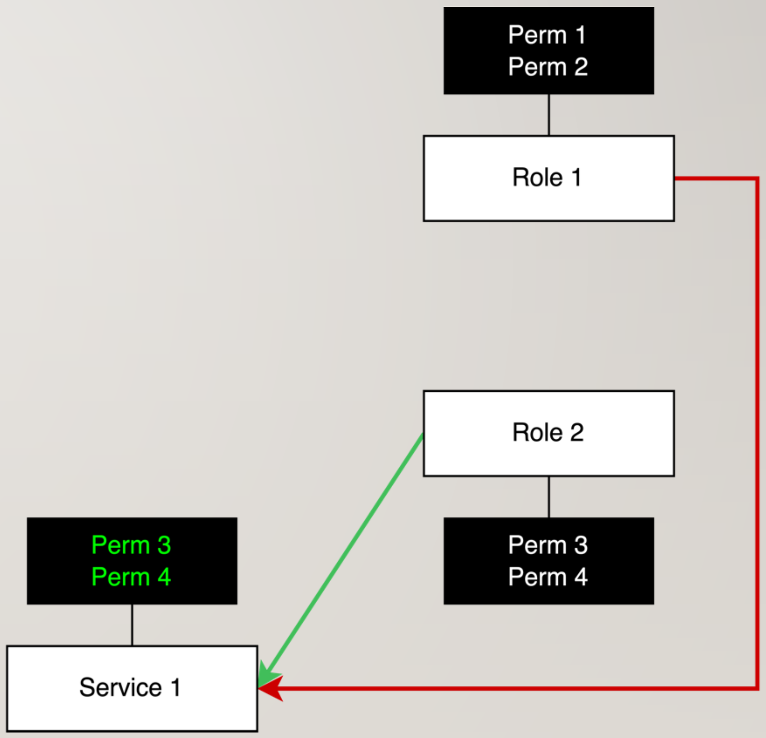


Service 1 is the untrusted entity who wants Perm 1 (target permission)

Perm 3 allows Service 1 to assume Role 1

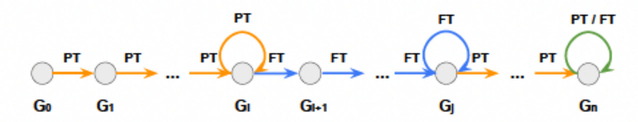
REPAIR ON IAM SEMANTIC REPRESENTATIONS

- The Semantic Representation Reduction
  - Include only the entities and permission flows that are relevant to the IAM PEs



REPAIR ON IAM SEMANTIC REPRESENTATIONS CONTD

- Safety Verification of IAM Configurations
  - Initially model the problem as a Model Checking problem
  - Given the finite-state machine, we check the safety property, asserting that no error states can be reached from the initial state
  - Apply Fixed Point Iteration (FPI) based approach to solve the problem.



(a) The finite state machine in FPI based verification.

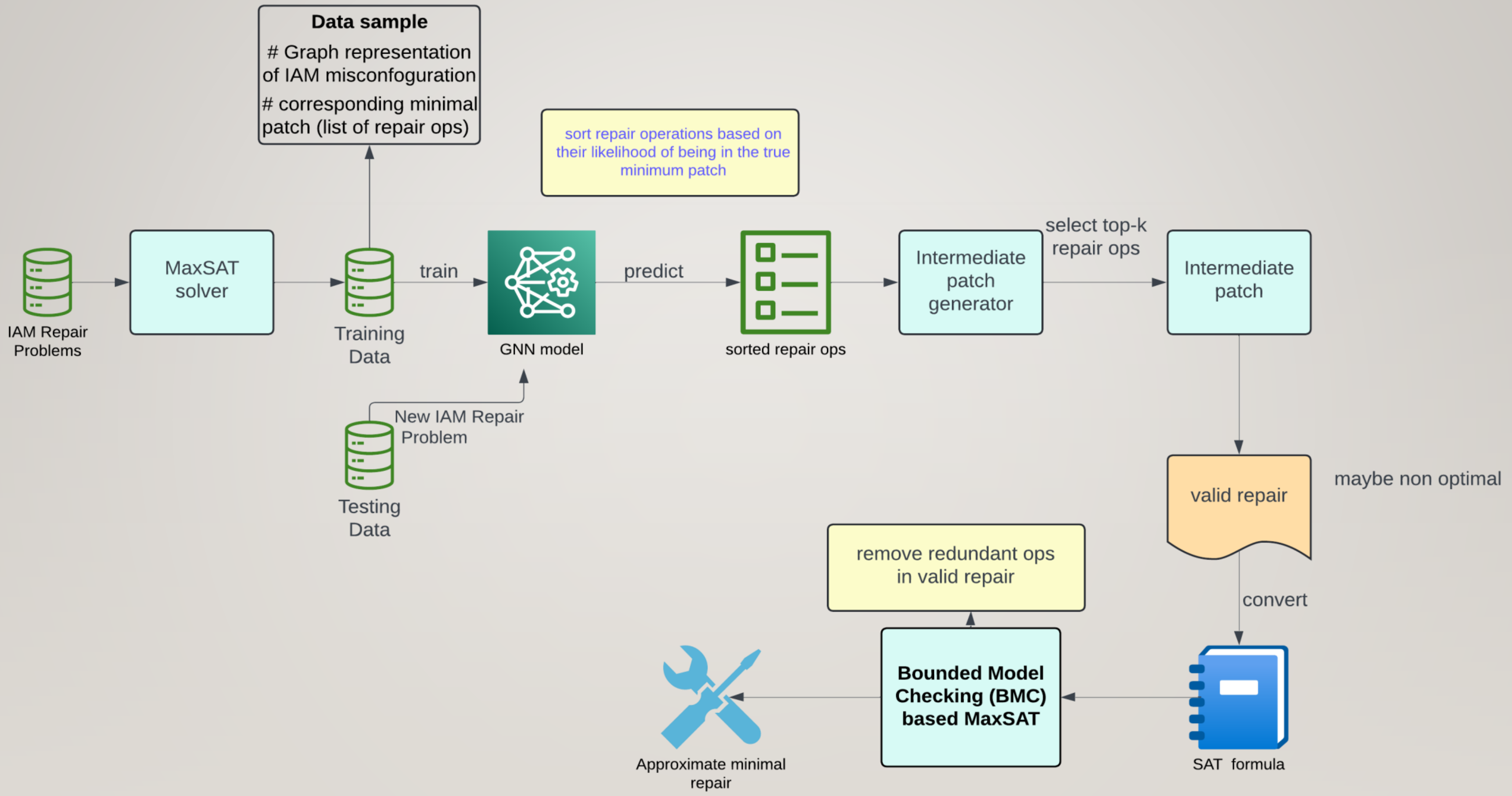
REPAIR ALGORITHM ON SEMANTIC REPRESENTATION

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}

- Use GNN to rank repair operations based on likelihood of being in the true minimum patch
- Iteratively select top-k repair operations to form an intermediate patch
- Find the minimum patch from the intermediate patch
  - Use FPI-based Model Checking to compute the bound
  - Apply BMC-based MaxSAT to generate repairs for the bounded safety property

Training and testing for GNN assisted MaxSAT repair



RELATED WORK OF REPAIRING PE

- IAM-Deescalate(by Palo Alto Networks)
  - The only existing PE repair tool
  - Limitations:
    - **Graph Modeling Focus:** Limited to authentication-based repairs, unable to handle non-authentication strategies (e.g., default IAM policy version changes)
    - **Transitive Privilege Escalations:** Ignores non-admin entities like services that can lead to privilege escalations
    - **Limited Repair Operations:** Only supports revoking permissions, not broader actions like removing users from groups
    - **Patch Objective:** Focuses on identifying patches without reducing or optimizing for the minimum patch size.

EXPERIMENTAL SETUP

- CASHWMaxSAT-CorePlus as MaxSAT solver
  - Winner in the Main track of MaxSAT Evaluation 2022
- Baselines
  - IAM-Deescalate: only existing IAM PE repair tool
  - IAMPERE-MO: exclusively employs the MaxSAT solver to generate repair
  - IAMPERE-GO: relies solely on GNN to generate the intermediate patch

DATASETS COLLECTION

- Total 4 sets
  - Training set, validation set, Test-A and Test-B
- Utilized IAM PE task generator, IAMVulGen by Hu et al
- for Training and Validation set
  - randomly generate 40,000 IAM misconfigurations with PEs for training and validation
  - apply IAMPERE-MO to obtain minimum patches
  - acquire 11,933 IAM misconfigurations with minimum patches within the time limit
    - Each IAM misconfiguration contains between
      - 8 and 336 entities, 24 and 15,525 permissions, and 7 and 15,263 permission flows.
  - Training-validation ratio 90-10

For testing set Test-A

Randomly generate 1,000 IAM misconfigurations

Each IAM misconfiguration contains between

- 11 and 315 entities
- 42 and 11,737 permissions
- 12 and 11,543 permission flows

For testing set Test-B

Collect five real-world IAM configurations

Owned by cloud customers from a US-based security startup

Two of them are misconfigurations

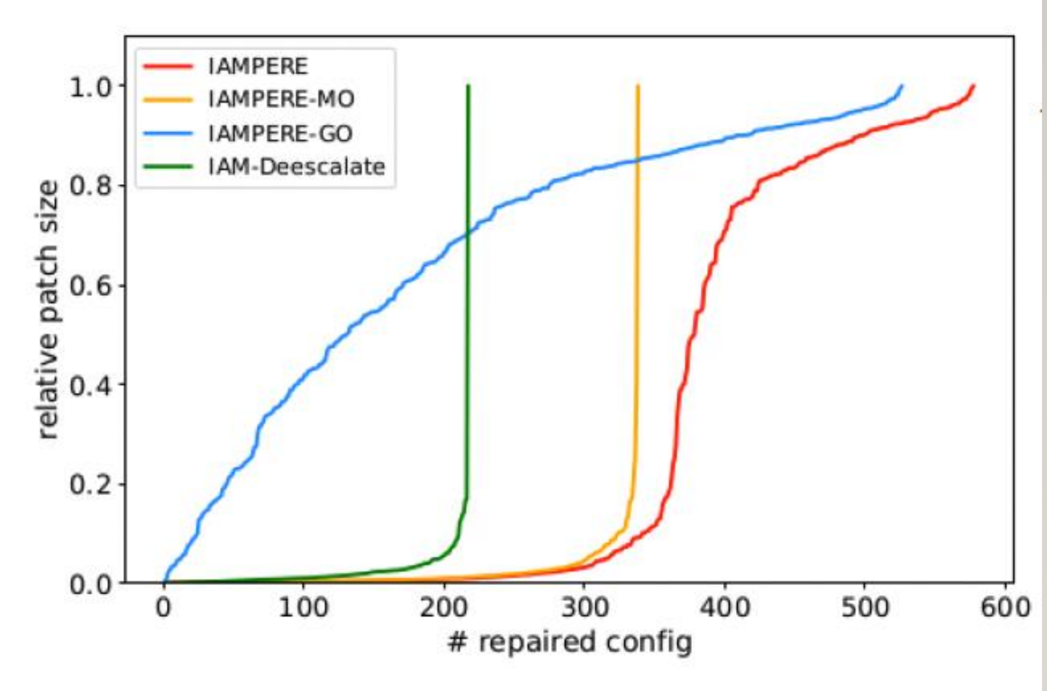
- **Real-1:** 251 entities, 2,826 permissions, and 27,939 permission flows
- **Real-2:** 158 entities, 882 permissions, and 8,704 permission flows
- both misconfigurations comprise over 10 PEs
- transitive PE and have PE path lengths of at least 5

METRICS AND TIME LIMIT

- Evaluation metrics
  - Effectiveness: relative patch size = patch size / max patch size
  - Efficiency: time cost
- Solving time for each repair
  - 600 seconds for training, validation and Test-A
  - 2 hours for Test-B

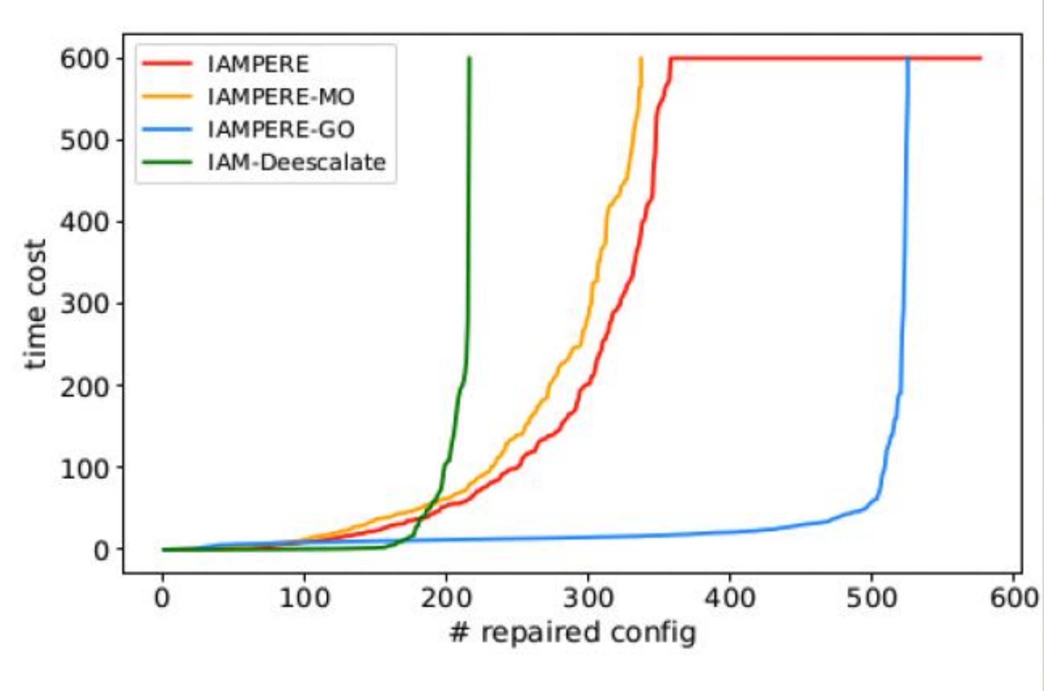
EFFECTIVENESS EVALUATION ON TEST-A

- Both IAMPERE and IAMPERE-MO significantly improve upon IAM-Deescalate in terms of patch size
- IAMPERE not only repairs more configs but also produces small patches



EFFICIENCY EVALUATION ON TEST-A

- IAM-Deescalate is significantly outperformed by IAMPERE and its variants.
- IAMPERE is consistently more efficient than IAMPERE-MO repairing 220 more IAM misconfigurations exactly at the time limit
- The high performance of IAMPERE-GO in terms of repair time cost demonstrates that GNN model inference for intermediate repairs is highly efficient



PERFORMANCE ON TEST-B

IAM Misconfiguration	Model	Repair time(seconds)	Relative Patch Size
Real-1	IAM-Deescalate	Failed	Failed
Real-1	IAMPERE-MO	Failed	Failed
Real-1	IAMPERE-GO	5147	0.889
Real-1	IAMPERE	7200	0.889
Real-2	IAM-Deescalate	Failed	Failed
Real-2	IAMPERE-MO	3963	0.0048
Real-2	IAMPERE-GO	2107	0.741
Real-2	IAMPERE	1190	0.0048

SUMMARY

- **MaxSAT Limitation:** Sole reliance on MaxSAT can exceed time limits for fixing real-world misconfigurations
- **GNN + MaxSAT Combination:** Reduces time and helps generate smaller, potentially minimal patches
- **IAMPERE Limitation:** Aims to generate an approximately minimum patch by leveraging GNN, it does not guarantee the absolute minimum, which makes the approach incomplete
- **Broader Impact:** GNN's effectiveness in aiding MaxSAT shows promise beyond just IAM PE
  - planning and scheduling
  - Verification and validation
  - bioinformatics

THANK YOU

QUESTIONS?