Institute for Software and Systems Engineering

TU Clausthal
Clausthal University of Technology

From Data Leak to Secret Misses: The Impact of Data Leakage on Secret Detection Models

Farnaz Soltaniani, Mohammad Ghafari

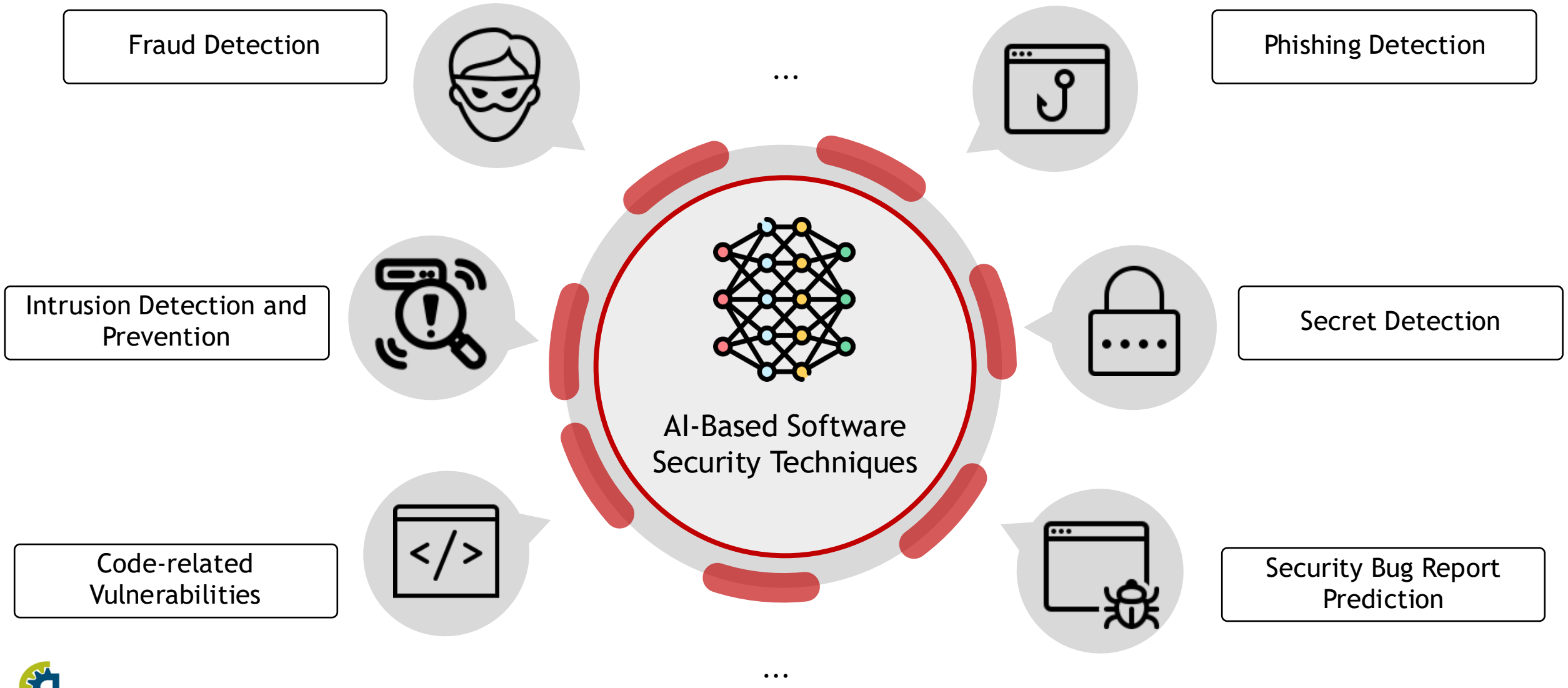
PhD @ Technische Universität Clausthal, Germany

The 33rd IEEE International Conference on Software Analysis, Evolution and Reengineering

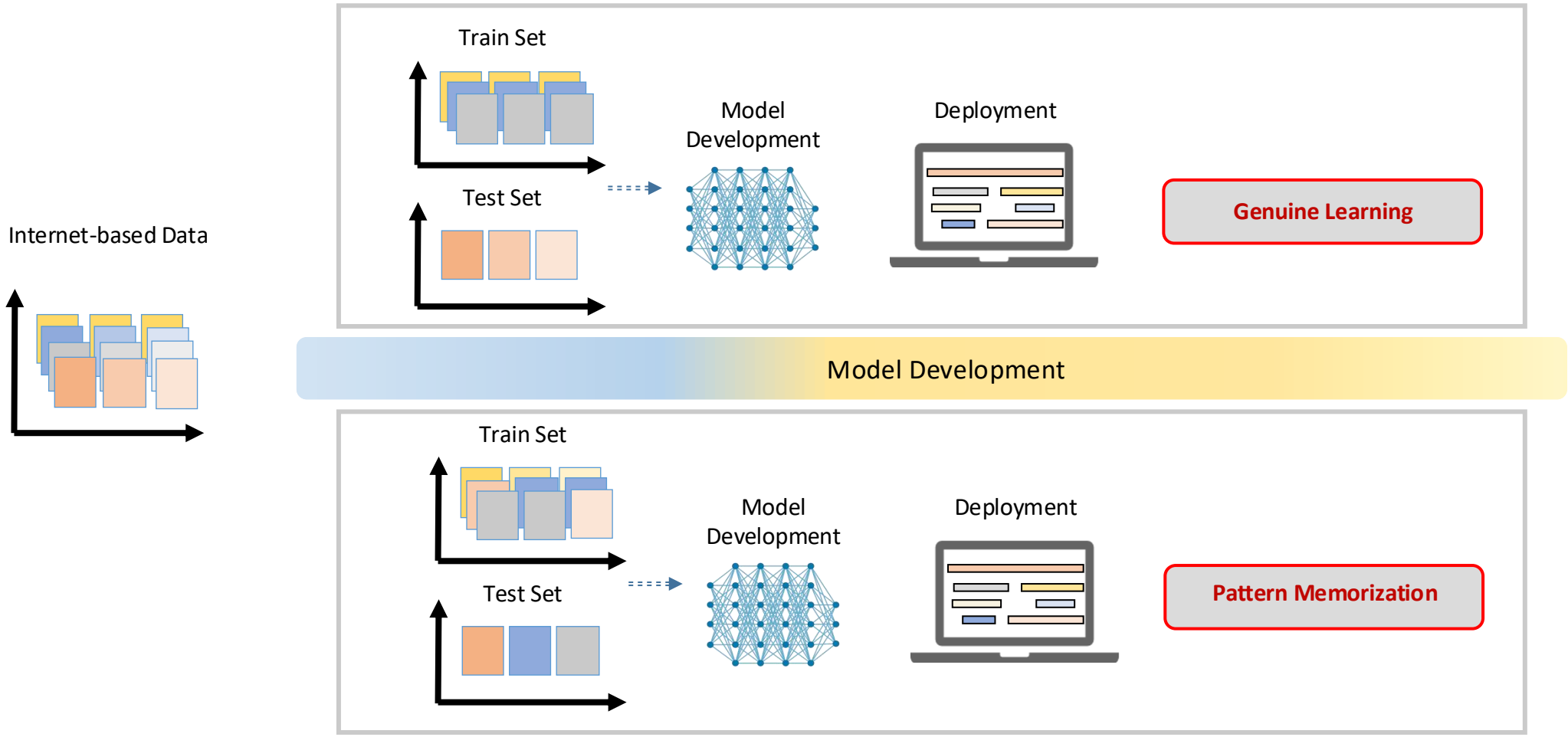
March 2026



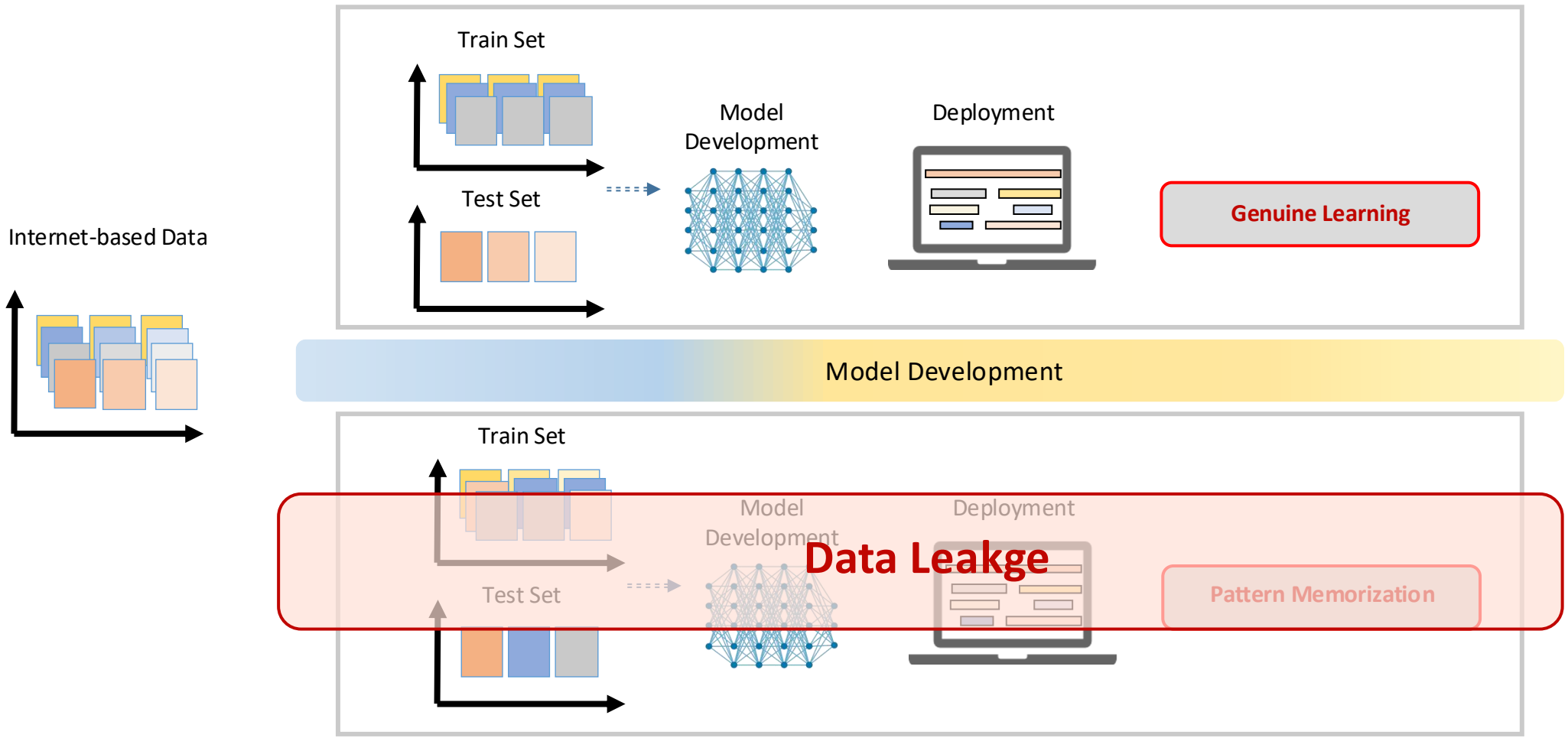
AI-based models are increasingly prevalent in security research



Ideal Generalization vs. Real-World Memorization



Ideal Generalization vs. Real-World Memorization



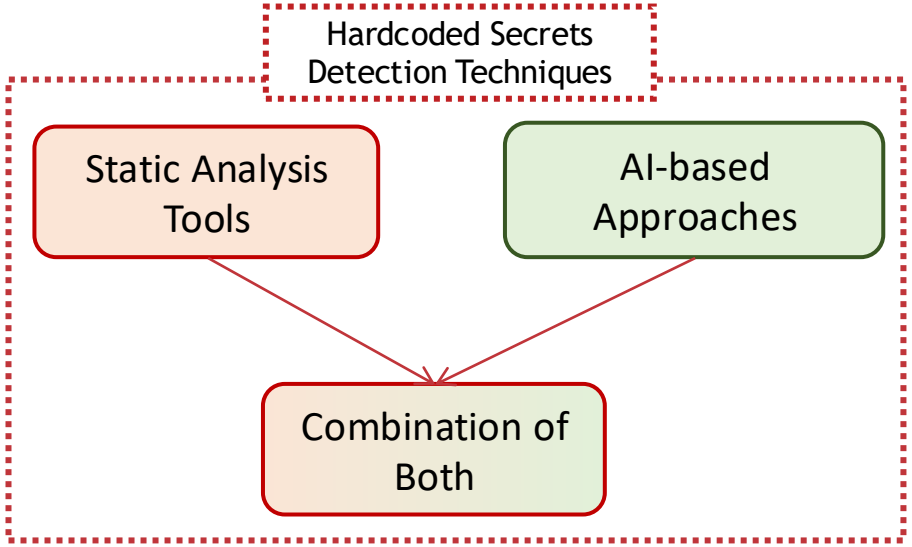
Secret Detection



- Passwords
- API keys
- Private Keys
- Authentication Keys
- Authentication Tokens
- Server URLs
- Generic Secrets
- ...

Secrets

- Developers leverage secrets to access different resources (e.g., databases, APIs, Amazon services).
- While secrets need to be safely stored and loaded from external sources, developers might hard code them plainly in source code. -> **compromise the security of confidential resources**



Secret Detection

Passwords

- Developers leverage secrets to access different resources (e.g., databases, APIs, Amazon services).
- While secrets need to be safely stored and loaded from external sources (e.g., environment variables, configuration files, or secret management tools), developers might hard code them plainly in source code.

Secrets are still exposed

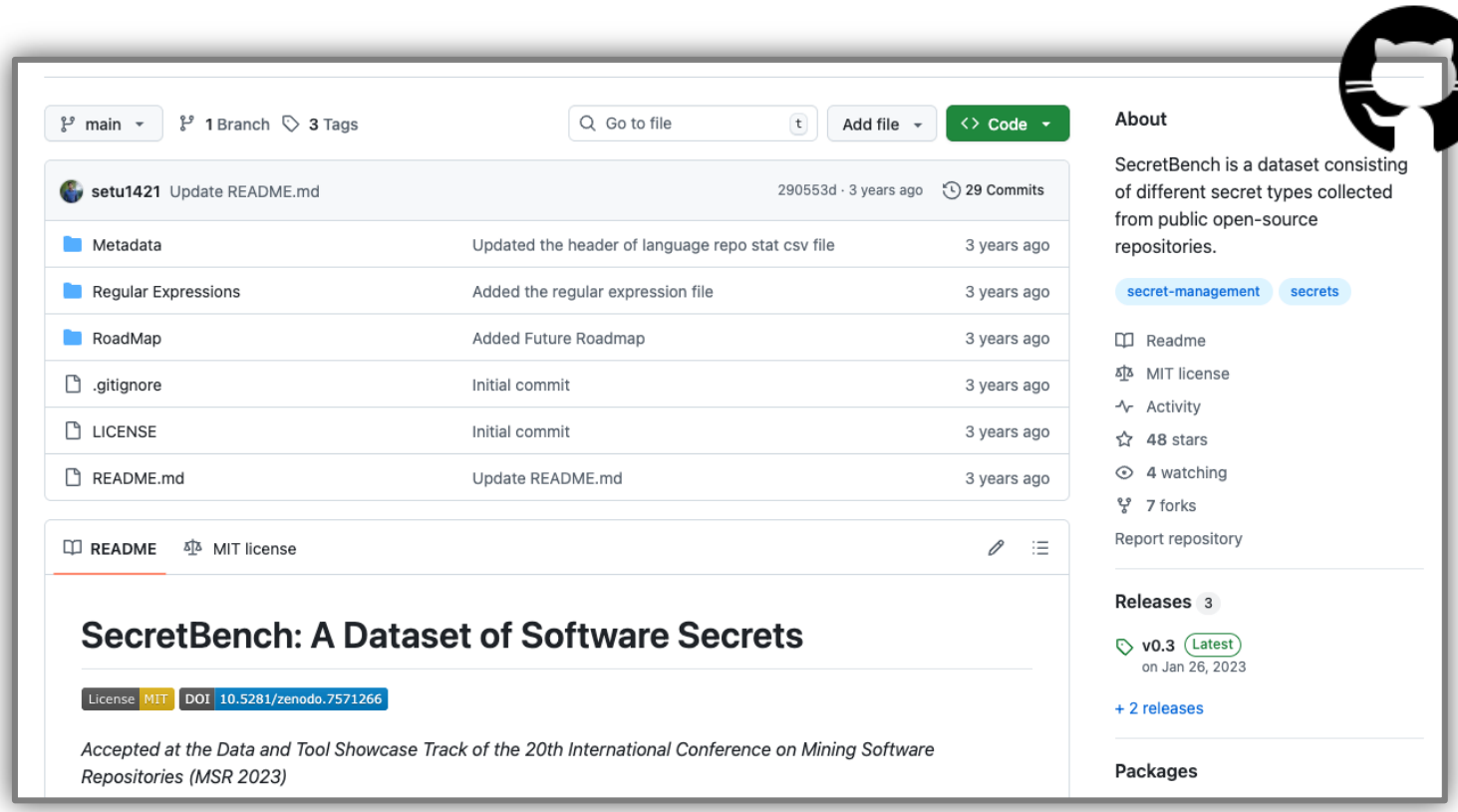
Server URLs

In 2025, the analysis of GitHub repositories associated with startups in the Forbes 50 AI list revealed that 65% of AI companies had exposed secrets that provide access to training data.

Secret Detection

Lack of large-scale, manually labeled datasets necessary for automatic secret detection.

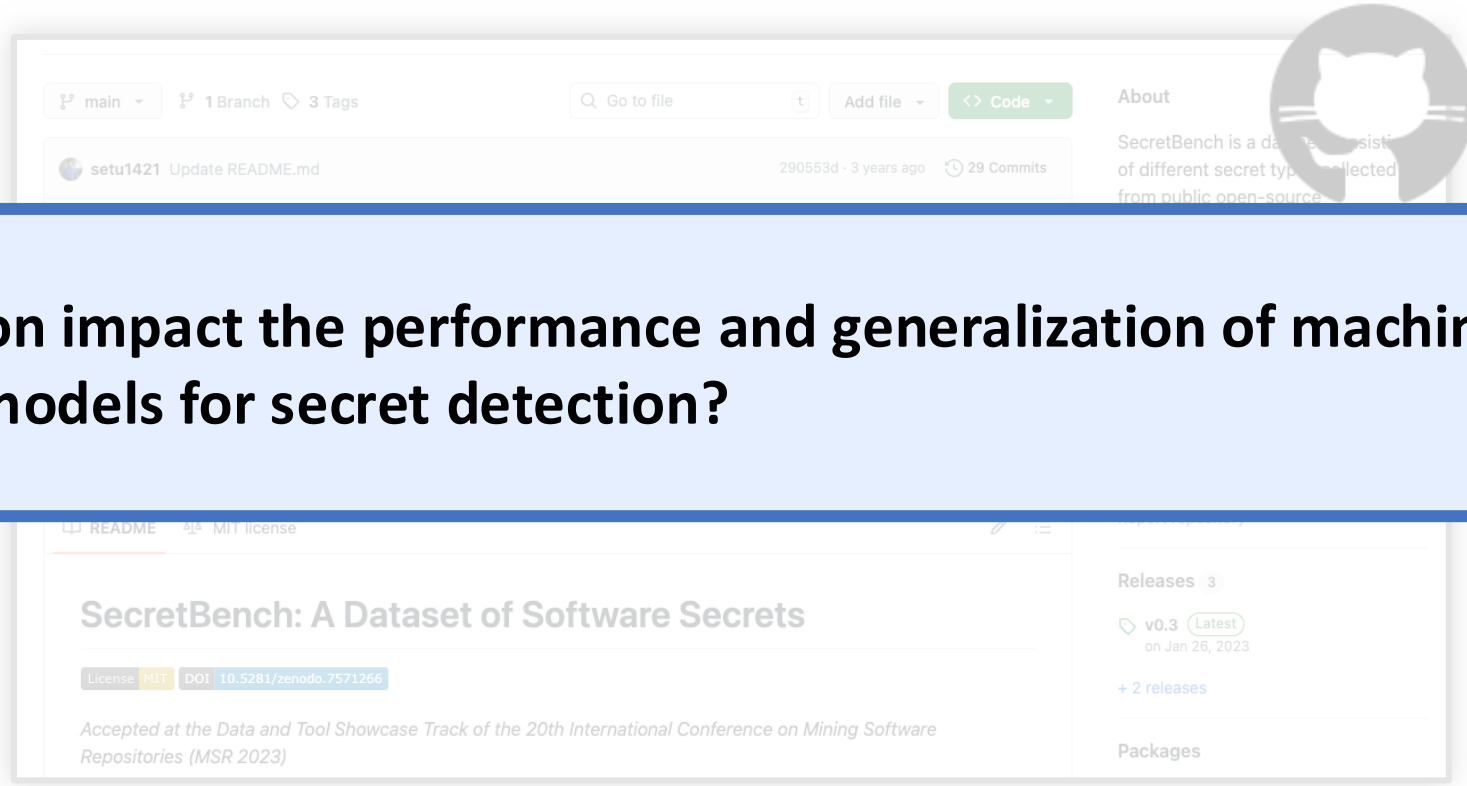
The dependence on GitHub as a data source raises concerns about code duplication within the dataset, which may compromise the validity of secret detection models.



Secret Detection

Lack of large-scale, manually labeled datasets necessary for automatic secret detection.

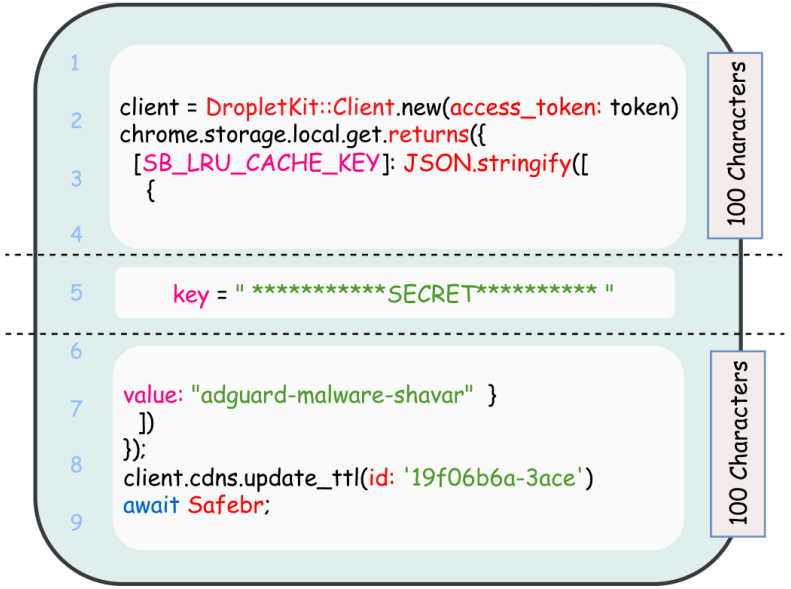
models.



RQ: To what extent does code duplication impact the performance and generalization of machine learning models for secret detection?

Secret Detection

Example of a secret code context

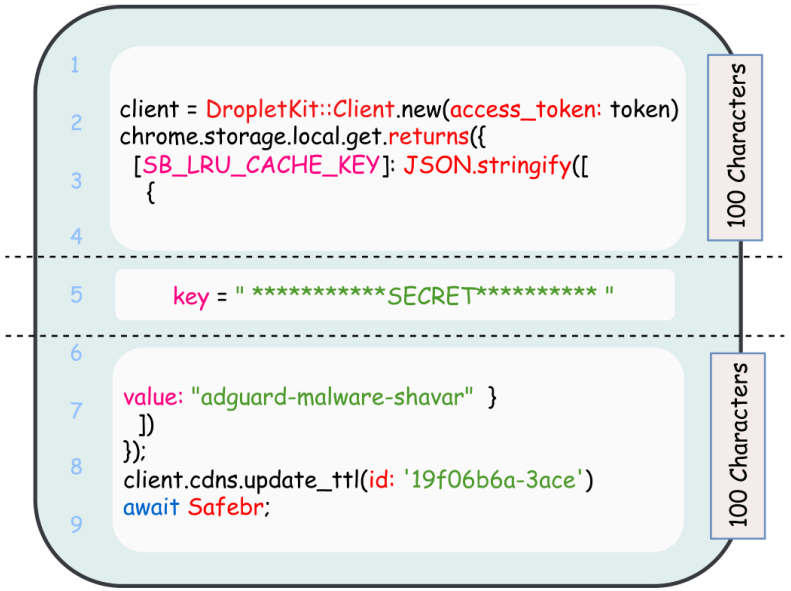


Secret Detection

Let C = {c1, c2, . . . , cn} be the multiset of all n contexts in the dataset.

1. Exact Duplicates (C_{exact}): Contexts that are identical to one or more other contexts in the dataset.
   - We also define a set of representatives (R_{exact}) from the multiset C_{exact} containing one copy of every duplicated context and $R_{exact} \subseteq C_{exact}$.

Example of a secret code context



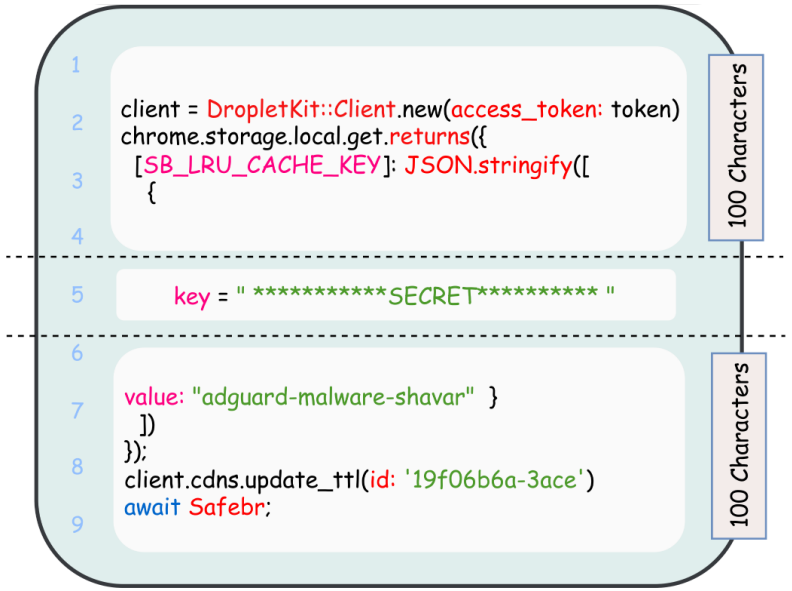
Secret Detection

Let C = {c1, c2, . . . , cn} be the multiset of all n contexts in the dataset.

1. Exact Duplicates (C_{exact}): Contexts that are identical to one or more other contexts in the dataset.
   - We also define a set of representatives (R_{exact}) from the multiset C_{exact} containing one copy of every duplicated context and $R_{exact} \subseteq C_{exact}$.

2. Near Duplicates (C_{near}): Near duplicates are semantically similar contexts that are not identical and differ only in a few aspects.

Example of a secret code context



Secret Detection

Let C = {c1, c2, . . . , cn} be the multiset of all n contexts in the dataset.

1. Exact Duplicates (C_{exact}): Contexts that are identical to one or more oth contexts in the dataset.
   - We also define a set of representatives (R_{exact}) from the multiset C containing one copy of every duplicated context and $R_{exact} \subseteq C_{exact}$

2. Near Duplicates (C_{near}): Near duplicates are semantically similar contex that are not identical and differ only in a few aspects.

```
TestFramework):
    def __init__(self):
        super().__init__(4, 0)
        self.sporkprivkey = "XXX"

    def run_test(self):
        self.sporkAddress =
            self.nodes[0].getaccountaddress("")
```

```
      super().__init__(6, 5,
          extra_args=[['-debug=instantsend']]
          * 6 )
        self.sporkprivkey = "XXX"

    def run_test(self):
        self.sporkAddress =
            self.nodes[0].getaccountaddress("s")
```

1. The values provided during initialization
2. Arguments passed to methods.

Secret Detection

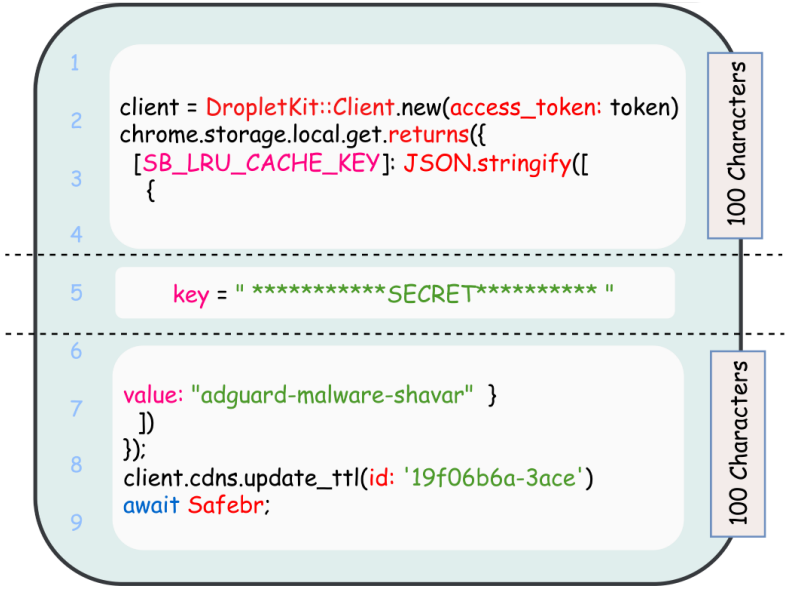
Let C = {c1, c2, . . . , cn} be the multiset of all n contexts in the dataset.

1. Exact Duplicates (C_{exact}): Contexts that are identical to one or more other contexts in the dataset.
   - We also define a set of representatives (R_{exact}) from the multiset C_{exact} containing one copy of every duplicated context and $R_{exact} \subseteq C_{exact}$.

2. Near Duplicates (C_{near}): Near duplicates are semantically similar contexts that are not identical and differ only in a few aspects.

3. Uniques (C_{unique}): A context is unique if it does not exactly match any other secret context. Uniques are sample contexts in $C_{\neg exact}$ that are neither exact duplicates nor near duplicates of any other context in the dataset.

Example of a secret code context



$$C = C_{unique} \cup C_{near} \cup C_{exact}$$

$$C_{dedup} = C_{unique} \cup C_{near} \cup R_{exact}$$

Secret Detection

Duplications in the SecretBench Dataset.

Duplication Types	Count (%)
C_{unique}	21,484 (22.0%)
C_{exact}	67,523 (69.3%)
C_{near}	8,472 (8.7%)
Total	**97,479 (100.0%)**

Duplications in the filtered SecretBench.

Duplication Types	Count (%)
C_{unique}	7,372 (31.6%)
C_{exact}	13,779 (59.0%)
C_{near}	2,201 (9.4%)
Total	**23,352 (100.0%)**

Secret Detection

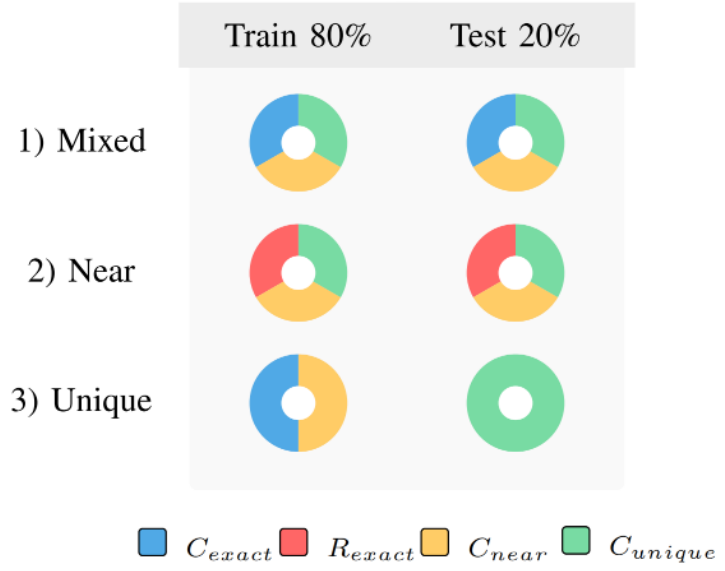
Duplications in the SecretBench Dataset.

Duplication Types	Count (%)
C_{unique}	21,484 (22.0%)
C_{exact}	67,523 (69.3%)
C_{near}	8,472 (8.7%)
Total	**97,479 (100.0%)**

Duplications in the filtered SecretBench.

Duplication Types	Count (%)
C_{unique}	7,372 (31.6%)
C_{exact}	13,779 (59.0%)
C_{near}	2,201 (9.4%)
Total	**23,352 (100.0%)**

1. **Mixed:** measuring model performance in the presence of all duplicates, simulating a naive evaluation on unpreprocessed data.
2. **Near Duplicate:** This scenario establishes the model's baseline performance without exact duplications. Training and Testing on datasets containing different, but semantically similar, instances.
3. **Unique:** measuring the model's ability to generalize from redundant data to entirely novel samples.



Models

Duplications in the SecretBench Dataset.

Duplications in the filtered SecretBench.

Duplication Types	Count
C_{unique}	21,484 (
C_{exact}	67,523 (69.3%)
C_{near}	8,472 (
Total	**97,479 (1**

ation Types	Count (%)
ue	7,372 (31.6%)
C_{exact}	13,779 (59.0%)
	2,201 (9.4%)
	23,352 (100.0%)

Random Forest

LSTM

GraphCodeBERT

GraphCodeBERT + MLP

1. Mixed: measuring model performan
   duplicates, simulating a naive evalua
2. Near Duplicate: This scenario establishes the model's baseline
   performance without exact duplicati
   datasets containing different, but se
3. Unique: measuring the model's abilit
   redundant data to entirely novel samples.

Results

Mixed Scenario Results

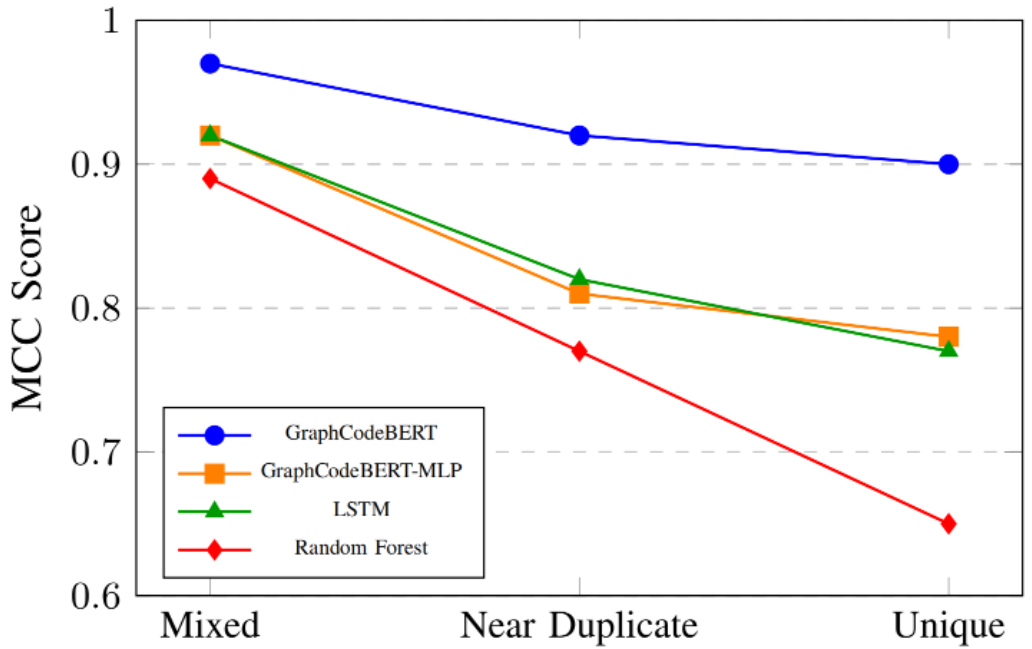
Classifier	Precision	Recall	F1-Score	MCC
GraphCodeBERT	0.97	0.98	0.97	0.97
GraphCodeBERT-MLP	0.92	0.93	0.93	0.92
Random Forest	0.96	0.86	0.91	0.89
LSTM	0.92	0.94	0.93	0.92

Duplicate Scenario Results

Classifier	Precision	Recall	F1-Score	MCC
GraphCodeBERT	0.91	0.96	0.93	0.92
GraphCodeBERT-MLP	0.85	0.83	0.84	0.81
Random Forest	0.90	0.71	0.79	0.77
LSTM	0.84	0.84	0.84	0.82

Unique Scenario Results

Classifier	Precision	Recall	F1-Score	MCC
GraphCodeBERT	0.91	0.92	0.91	0.90
GraphCodeBERT-MLP	0.82	0.80	0.81	0.78
Random Forest	0.85	0.56	0.67	0.65
LSTM	0.82	0.79	0.80	0.77



Conclusion

- Duplications in the SecretBench dataset can significantly inflate the reported performance of AI-based secret detectors.
- Models are not inherently robust to such duplications, and their sensitivity varies across different model families.
- The exceptional performance reported in the literature may reflect memorization rather than true learning.
- We call for the reevaluation of secret detection models to address data leakage and ensure that models are reliable and applicable in real-world settings

Mixed Scenario Results

Classifier	Precision	Recall	F1-Score	MCC
GraphCodeBERT-MLP	0.82	0.80	0.81	0.78
Random Forest	0.85	0.56	0.67	0.65
LSTM	0.82	0.79	0.80	0.77

Farnaz Soltaniani

PhD @ Technische Universität Clausthal, Germany

The 33rd IEEE International Conference on Software Analysis, Evolution and Reengineering

March 2026

