Efficient ransomware detection with machine learning in storage systems

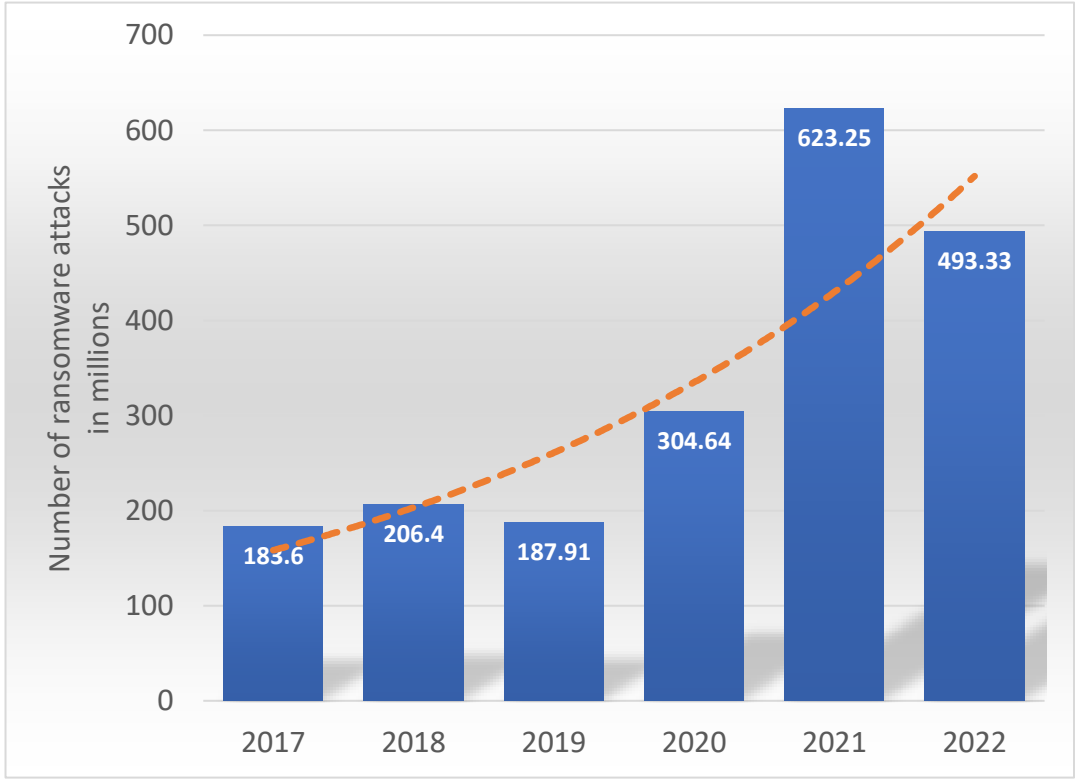
Presenter: Roman Pletka

Contributors: Dionysios Diamantopoulos, Haris Pozidis, Slavisa Sarafijanovic

IBM Research Europe, Zurich

Ransomware cyber threads

- Definition: Crypto-malware preventing access to data until a ransom is paid off
  - Double and triple extorsion (threatens to disclose the victim's data, extort 3rd parties that may be affected by the disclosure, DDoS)
- Ransomware attacks rank among the top threads
  - Top attack type in 2021 (23%) [1]
  - Despite an overall drop in 2022, ransomware began to rise again with highest attack volume ever seen in Q4/2022
  - Approx. 600 million USD ransom payments WW in 2021

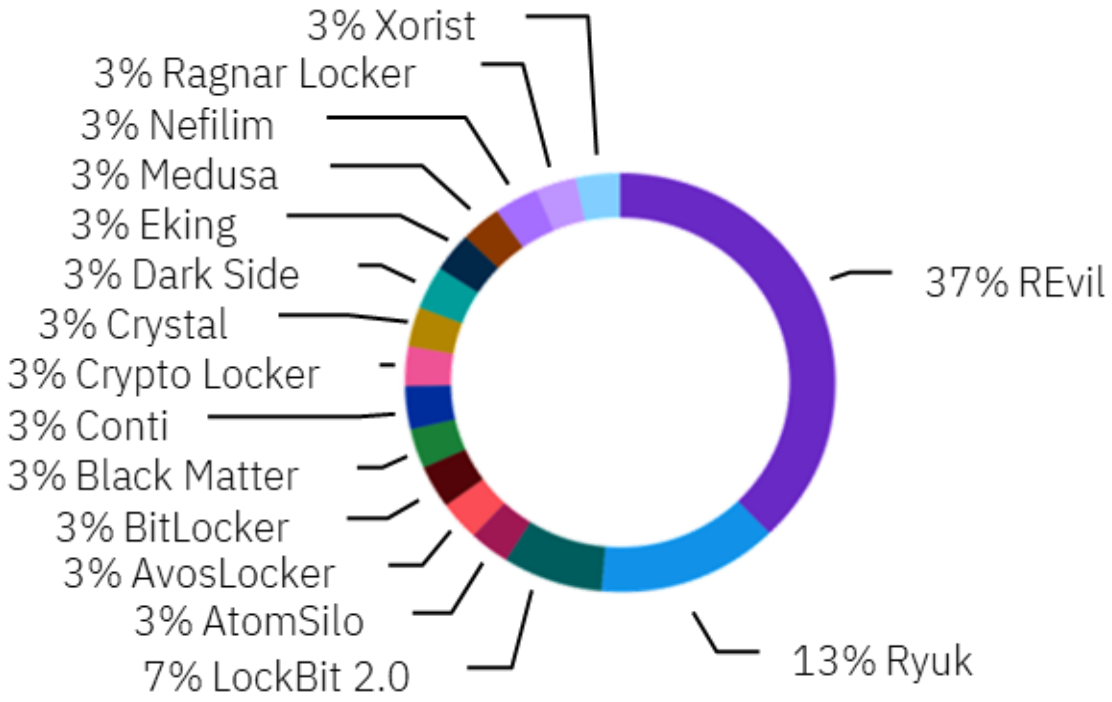


Source: SonicWall, Cyber Threat Report 2023

[1] IBM X-Force Threat Intelligence Index 2022

Ransomware cyber threads overview

- Large number of ransomware families
  - Over 100 new ransomware samples per year
- Ransomware-as-a-Service (RaaS) reduces the entry barrier
- Use of advanced obfuscation techniques (dead code insertion, code integration, intermittent encryption, …)
- Attack surface increase with more applications, tools, interfaces

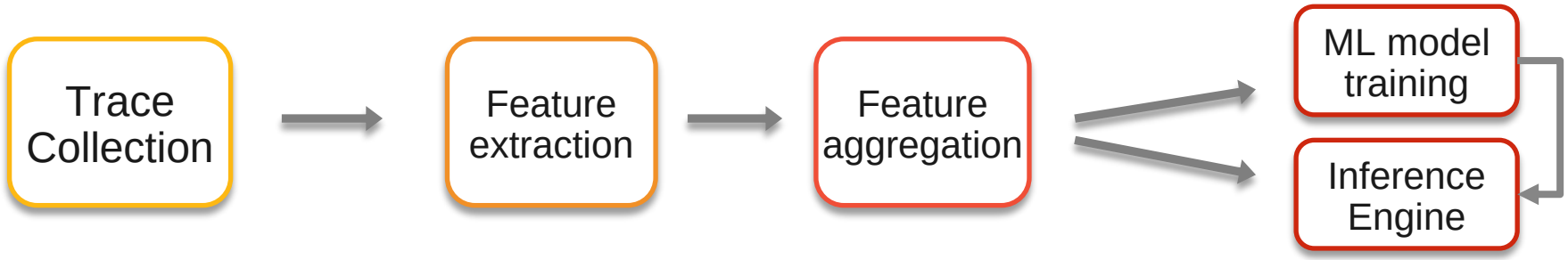


Source: IBM X-Force Threat Intelligence Index 2022

Where to detect malware?

Ransomware Detection	*Granularity*	*Performance Impact*	*Context / Visibility*	*Ease of implementation*
File-system or OS level	👍 Monitors file operations and their metadata and/or process activity. 👎 Can spot ransomware activity based on file access patterns. 👎 May not capture more sophisticated attacks that bypass the file system.	👎 Requires the operating system to intercept and analyse file operations. Higher performance impact. 👎 Can result in overhead and potentially slow down the system, especially when dealing with large volumes of data.	👍 Better visibility w.r.t file operations (e.g., process/user responsible for the actions). Valuable information for detecting ransomware based on behavioural patterns. 👎 Often implemented by blacklisting or whitelisting specific applications or users.	👍 Generally easier to implement, as it relies on the operating system's APIs and file system structures, which are often more accessible and well-documented. 👎 Must be adapted to every OS version.
Block-level	👍 Monitors storage device operations at the granularity of data blocks (sectors). 👍 Can capture changes at a low level, possibly identifying malware activity even if it uses stealth techniques to hide itself. 👎 File information not available	👍 Operates within the storage subsystem, where it can directly monitor and analyse data transfers. 👎 However, may require more advanced techniques to differentiate between malware activity and legitimate operations.	👍 Allows to detect changes that could be hidden from higher-level detection mechanisms. 👎 Harder to detect ransomware activity based on behavioural patterns due to missing file-level context.	👍 Integration into hardware is a competitive advantage 👍 Once implemented it's universal. 👎 Block-level detection may require specialized knowledge of storage subsystems and low-level data structures.

From trace collection to ML Models

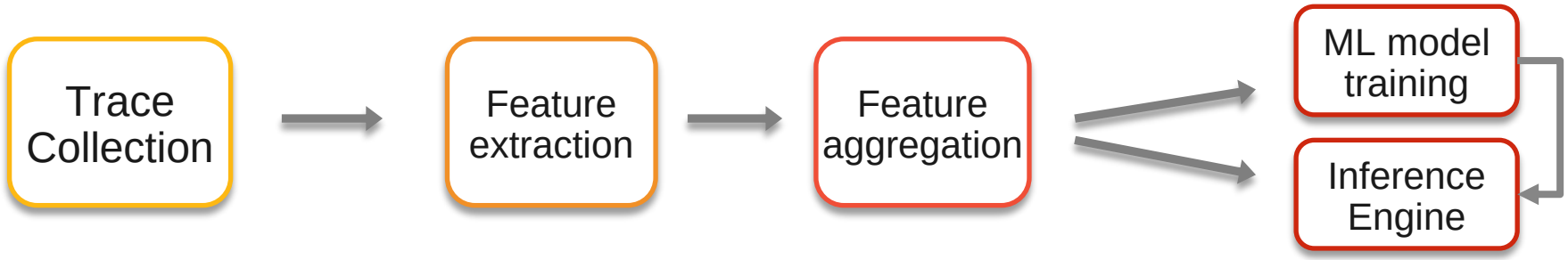


- KVM-based research test environment, real CSDs
- Large number of real ransomware samples with different setups
- Ransomware emulator
- Large real-world data sets (e.g., Govdoc1 with 1M files, …)
- Various benign workloads (doc server, file conversions, …)

- Pattern-preserving feature extraction and aggregation
- Parameters:
  - Feature selection
  - Sampling frequency
  - Aggregation interval

- Analysis of various ML models (decision-tree-ensembles)
  - Detection accuracy, model size, explainability
- Binary and multi-level classification:
  - Detection of malware activity
  - Indication on detected malware family
- Generalizability to unseen ransomware and workload changes
  - Detection when new models must be trained
  - Automate detection of new unknown ransomware

Ransomware detection from storage access patterns



Existing approaches

Hirano 2019, Hirano 2022:

- Detection of 7 ransomware attacks vs benign workloads using Random Forest, Support Vector Machines, and K-nearest neighbor models, accuracy of up to 0.98 (F1 score)
- Hypervisor-based trace generation for feature extraction
- Limited set of features: Shannon entropy of writes, read and write size, variance of LBA of reads and writes
- Offline training and interference, no real time detection
- Unclear generalizability of the model to more realistic environments and mixed workloads

Gagulic 2023:

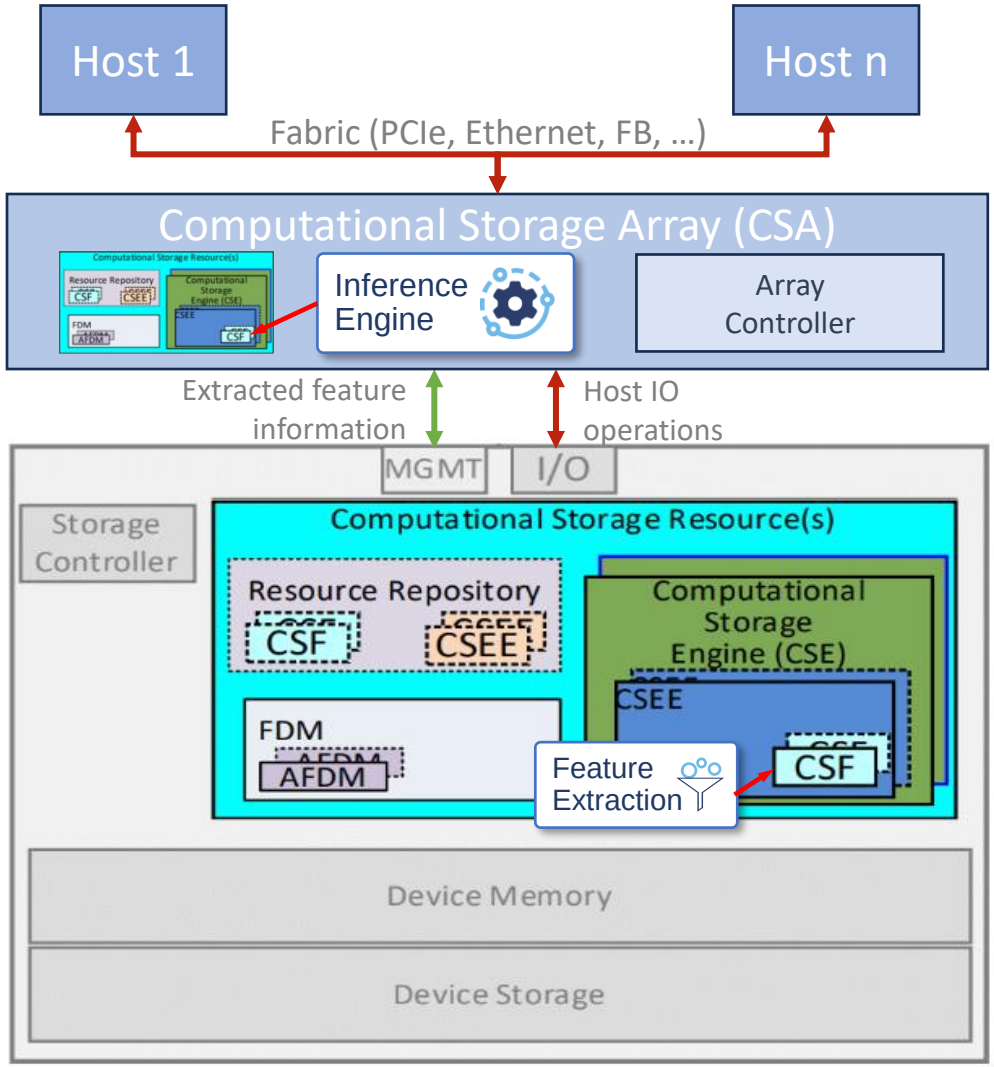
- Detection of 6 ransomware attacks vs benign workloads for different using Random Forest, XGBoost, DNN models
- KVM-based setup with SystemTap and a device mapper module for feature extraction
- By using 7 additional features, accuracy increased by up to 10%
- Offline training and interference, no real time detection
- Generalizability of the model to mixed workloads, different setups, and unseen ransomwares

[Hirano 2019] M. Hirano and R. Kobayashi, Machine Learning Based Ransomware Detection Using Storage Access Patterns Obtained From Live-forensic Hypervisor, IOTSMS 2019.
[Hirano 2022] M. Hirano, R. Hodota, Kobayashi, RanSAP: An open dataset of ransomware storage access patterns for training machine learning models, Digital Investigation, 2022
[Gagulic 2023] D. Gagulic, Lynn Zumtaugwald, Siddhant Sahu, Ransomware Detection with Machine Learning in Storage Systems, 2023

Computational Storage model

NVMe spec TP4091

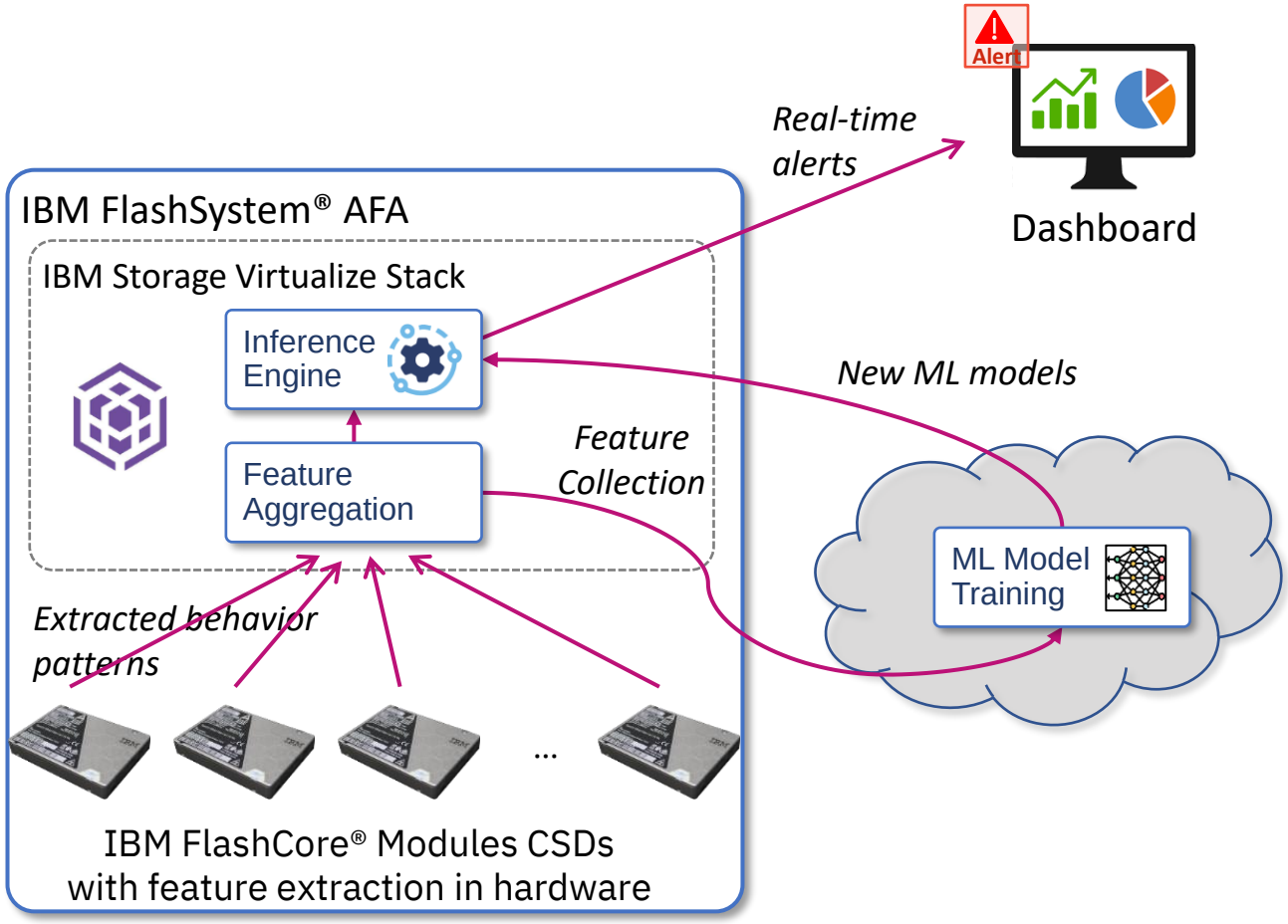
- Computational Storage Function (CSF) in CSD:
  - Feature extraction from I/O operations
  - Periodical aggregation of extracted features in time windows
- CSF in Computational Storage Array (CSA):
  - Collect extracted feature information
  - Perform inference to detect malicious behavior and send alerts
- Advantages
  - Feature extraction implemented in hardware has no impact on host IO operations
  - Aggregation performed in the background using dedicated embedded core(s) in the CSD and CSA



FCM4 Computational Storage Drive (CSD)

Ransomware detection architecture for AFAs

- CSDs collect features extracted from IO operations.
  - Feature extraction done by hardware without delaying host IO operations and summarized by dedicated processors
- Feature Aggregation from each CSD for
  - detecting anomalous behavior in the inference engine using system-specific ML models
  - training new ML-models (outside the storage system)
- Periodic retraining of ML models performed offline in the cloud
- Alerting and mitigation
  - Real-time alerts in FlashSystem UI and IBM Storage Insights®
  - Immutable snapshots

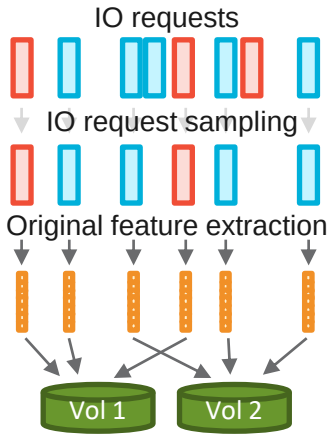


Feature extraction in a CSD

Original features

Extracted from (all/sampled) IO operations

- Shannon Entropy of writes
- Read transfer size
- Write transfer size
- Read LBA
- Write LBA
- NVMe application tag (volume ID)
- …



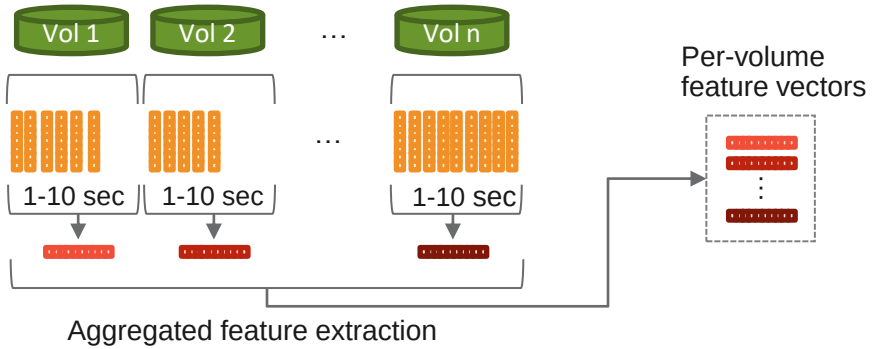
Windowing

Aggregated features are extracted from IO information using a moving window over 1-10 seconds

Aggregated features

Additional features extracted from windows for each volume:

- Mean/variance/Kurtosis of entropy of writes
- Mean/variance read and write transfer size
- Variance/Kurtosis of LBAs read and written
- Read and write IO rate
- …



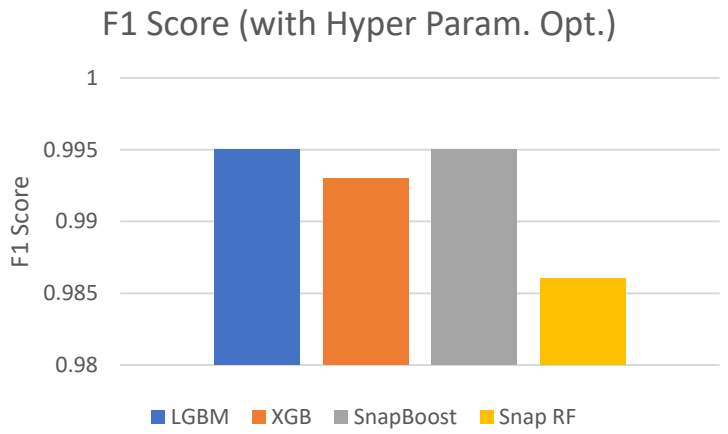
ML model comparison for binary classification

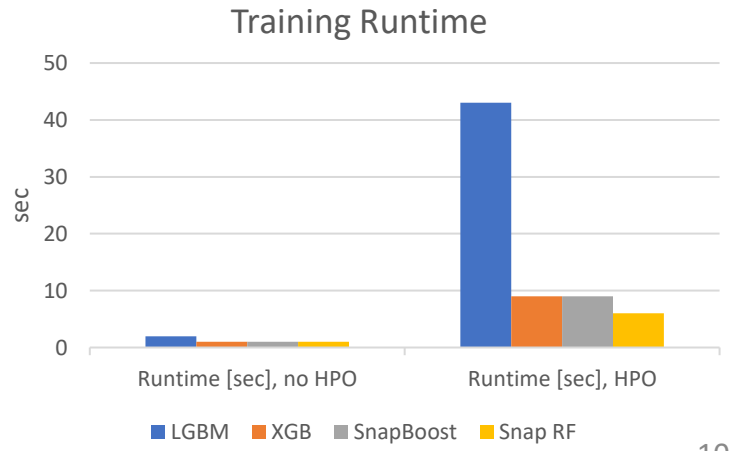
Results from benign and ransomware traces trained with IBM Auto AI using SnapML

- SnapBoost shows the best F1-score and runtime trade-off
- Feature importance depends on model, setup (file system type), and evaluation method (intrinsic, feature permutation, SelectKBest)

Averaged feature importance (top 5 features) from different models using IBM Auto AI:

Average Importance	*XGBoost*	*LGBM*	*Snap RF*	*Snap Boosting*
Feature 1 (86.0%)	Feature 1 (100.0%)	Feature 2 (100.0%)	Feature 4 (100.0%)	Feature 1 (100.0%)
Feature 2 (54.3%)	Feature 3 (19.0%)	Feature 3 (99.0%)	Feature 2 (82.0%)	Feature 2 (16.0%)
Feature 3 (43.3%)	Feature 4 (19.0%)	Feature 1 (98.0%)	Feature 3 (50.0%)	Feature 3 (13.0%)
Feature 4 (31.5%)	Feature 5 (11.0%)	Feature 5 (98.0%)	Feature 1 (46.0%)	Feature 4 (7.0%)
Feature 5 (26.0%)	Feature 11 (7.0%)	Feature 8 (83.0%)	Feature 7 (45.0%)	Feature 11 (5.0%)

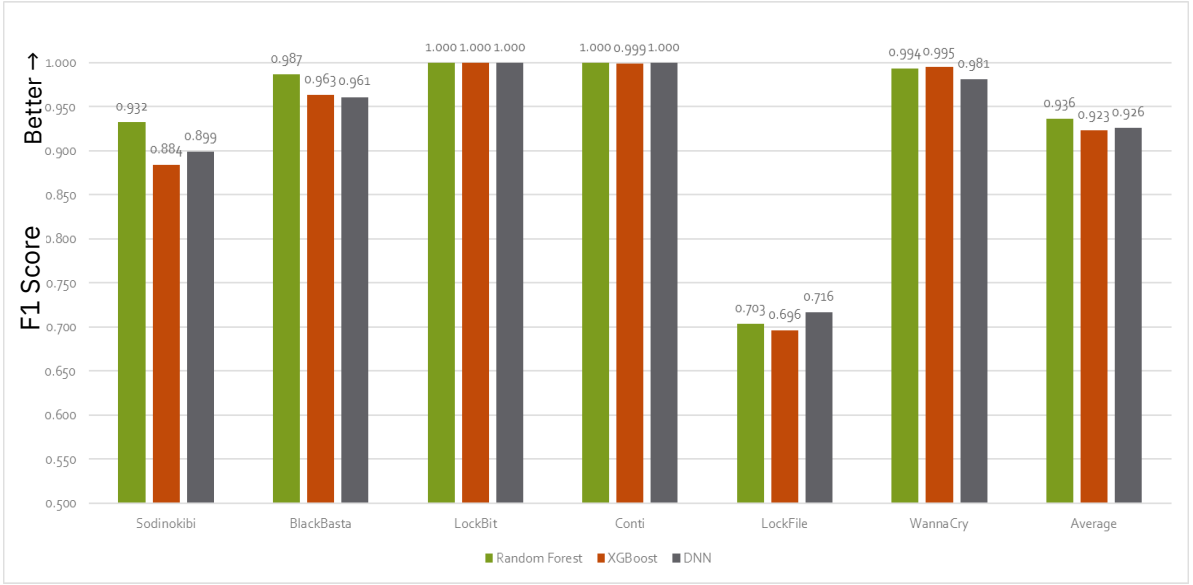




Generalizability

Can an existing ML model detect unseen ransomware?

- Experiment: Train models by excluding the ransomware being detected afterwards
- Overall, most models generalize well to unseen ransomware
  - Well predicted: BlackBasta, LockBit, Conti and WannaCry
  - LockFile is not predicted well due to significant different behavior than other ransomwares (intermittent encryption, in memory encryption with minimal disk IO) => Model retraining advised
- Training and evaluation methodology used:
  - Binary classification using balanced datasets
  - Using 12 extracted features
  - Windowing with window size 10s, offset 1s
  - 5-fold cross validation



Summary

- Protection against ransomware attacks in a multi-dimensional fashion
  - File system, OS-level, block storage, …
- Ransomware can be efficiently detected by observing block IO operations without host impact using CSDs
  - Combination of metrics collected including entropy information
  - Periodical aggregation of metrics
- Clearly defined mitigation strategies
  - Timely alerting with low mis-detection probabilities
  - Maintaining of immutable snapshots in the background

Thank you!

Dr. Roman Pletka

Senior Research Scientist
Master Inventor

rap@zurich.ibm.com

IBM Research Europe – Zurich