Visualization-supported Analysis of System Data for Controlled VMI-based Intrusion Detection

Noëlle Rakotondravony, Prof. Hans P. Reiser

Juniorprofessur für Sicherheit in Informationssystemen
Universität Passau

March 2, 2017

Outline

UNIVERSITÄT PASSAU

- Motivation
- Controlled & cost-aware monitoring architecture
- VMI-based system call tracing use-case
- Conclusion & Future work

Motivation

UNIVERSITÄT PASSAU

Visualization-supported analysis

- System level data visualization
- In-depth: Combination of multiple data sources
- Interactive

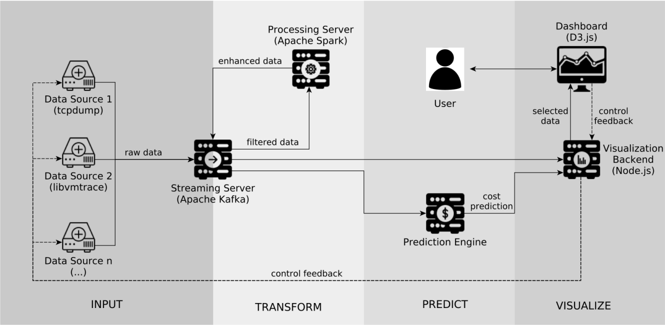
VMI-based security monitoring

- VMI properties: Isolation, Inspection and Interposition
- Performance overhead

Controlled VMI-based intrusion detection

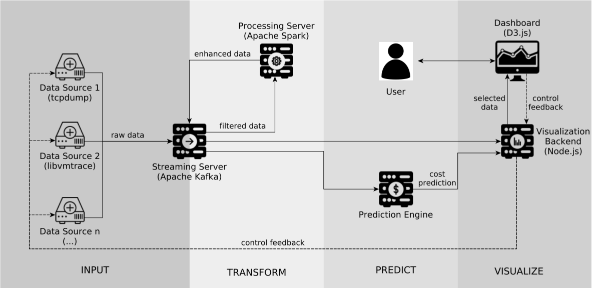
- On-demand monitoring
- Cost-aware
- Trade-off between enriched data and performance overhead

Architecture



Visualization-suppported & cost-aware architecture: components and workflow

Architecture - Input phase

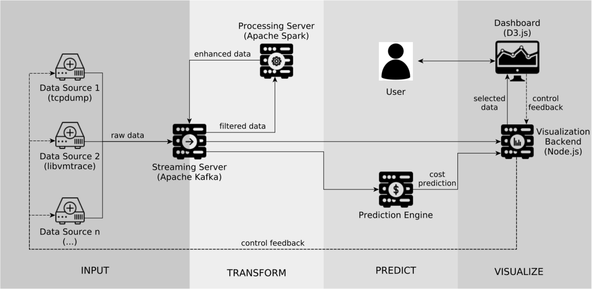


Data sources

- Standard data sources at OS, applications (log files), network levels (traces)
- VMI-based monitoring mechanisms (built on-top of LibVMI)

Streaming server: Apache Kafka

Architecture - Transform phase



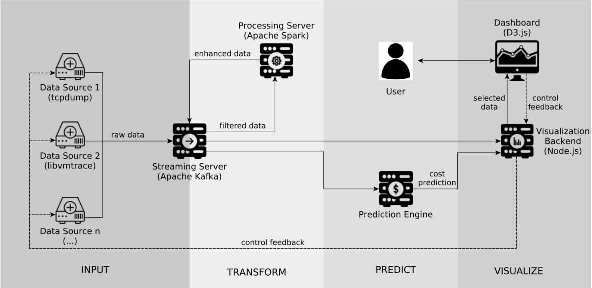
Processing server

- Subscribes to data stream
- Apache Spark

Transformation operations

- Event filtering, coorelation, aggregation & summarization
- Complex analysis like Machine learning

Architecture - Visualize phase



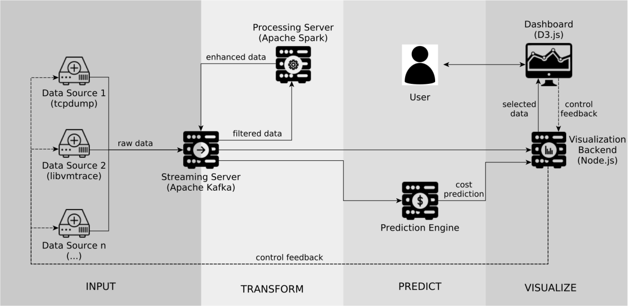
Dashboard

- Interactivity
- on-demand monitoring

Visualization backend

- RESTful web service for data retrieval
- Interface to the monitoring components

Architecture - Predict phase



Prediction engine

- In-depth monitoring → richer data → higher overhead
- Cost prediction for reasonable choice of monitoring mechanism
- Assumptions: impact correlates to frequency of monitored events

UNIVERSITÄT PASSAU

Metrics

- T_m: sampling time to trigger tracing mechanism i ($i \in E$)
- t_i: impact of monitoring a single event of i
- f_i: frequency of events of type i captured during T_m
- $\theta = \sum_{i \in E} f_i \times t_i$: Monitoring work during T_m
- $\lambda = \frac{\theta}{T_m}$: Fraction of total time consumed by monitoring
- $\epsilon = \frac{\theta}{T_m - \theta}$: Indicates increase of runtime due to monitoring
- $T' = (1 + \epsilon)T$: Total execution time

Prediction phase - use-case

UNIVERSITÄT PASSAU

Use case: VMI-based system call monitoring

- Predict impact of tracing system calls
- Evaluate the prediction

Data source: Libvmtrace

- VMI-based library for system call and network monitoring
- Built on top of LibVMI
- Injects breakpoints for system call tracing
- Not yet optimized for perfomance
- Libvmtrace vs. other system call tracing libraries
  - selectively monitor system calls of interest

Metrics

- Set t_i: trace a single system call
- t_i can be determined offline

sys.call	Total(s)	Overhead(s)	Count	$t_i/call$(ms)
sys_read	371.6	313.7	202,950	1.54
sys_write	158.1	100.2	64,850	1.54
sys_open	660.1	602.2	977,870	0.61
sys_mmap	210.5	152.6	261,370	0.51
...				

Table: Determining t_i using a sample application that runs 58 s without monitoring

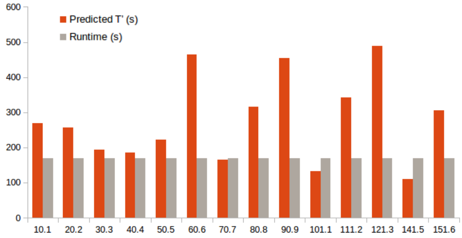
- $t_i \pm$ constant for each system call

Tracing *sys_mmap* in sample program ($t_{sys\_mmap} = 0.51ms$).

- sample program runtimes
  - without monitoring: 100s
  - with monitoring: 170s

Metrics

- $t_i = t_{sys\_mmap}$
- $T_m = 100ms$
- Sampling frequency
  - every 10s
  - (+) constant update of prediction
  - (−) increase in monitoring load λ



Constant update of prediction using $T_m = 100ms$ (every 10 s)

- Sampling frequency
  - every 10s
- Varying sampling size
  - (+) higher accuracy
  - (−) higher monitoring load

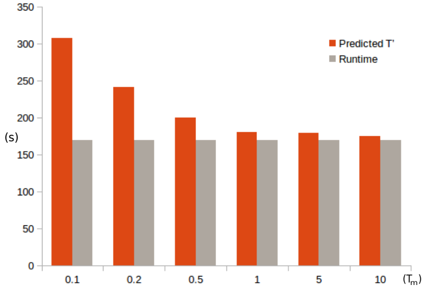
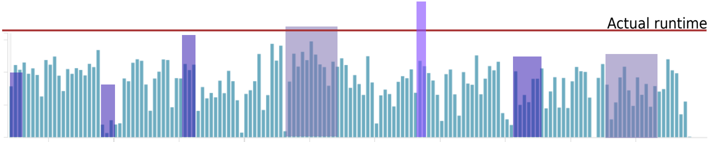


Figure: Average prediction for different T_m (every 10 s)

Prediction accuracy - Challenges



sys_mmap system call distribution during monitoring runtime

The prediction accuracy is function of the distribution of the monitored system call. Challenges in:

- sampling size
- sampling frequency
- sampling timing

Future work

UNIVERSITÄT PASSAU

- Architecture for controlled security monitoring
- Cost-aware aspect for reasonable choice of heavy monitoring
- Interactive monitoring system
  - enable / disable selected monitoring mechanisms
- Machine Learning-based intrusion detection (w. TU München)
  - monitor features of interest
  - with least impact on performance

Thank you.