DIMAQS – Dynamic Identification of Malicious Query Sequences

Master Thesis Presentation

Michael Jobst

Advised by: Prof. Dr.-Ing. Alexandra Dmitrienko

Outline

Outline

What is Database Ransomware?

Two known Ransomware types:

- Crypto Ransomware
- Locker Ransomware

imposed 5 billion USD loss in 2017 predicted to hit 11.5 billion in 2019

What about Database Ransomware?

- first appearance in 2016
- connect to DBMS and deleting (dropping) databases/tables
- attacks against MySQL, MongoDB, ElasticSearch, Cassandra, Hadoop, and CouchDB

Prerequisites & Goals

Database Ransomware

- Dropping of Databases and Tables
- Demanding ransom to get database dump (copy of the data) back
- No evidence for such dumps

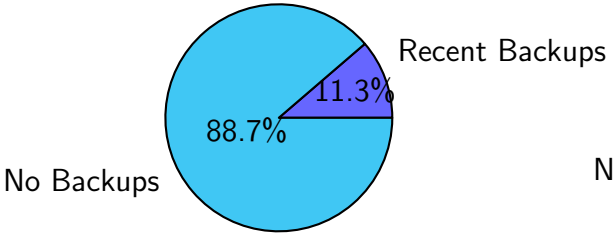
Goals

- Protect from data loss
- Detect malicious sequences, not only single malicious queries (SQLi)
- Determine database ransomware attacks

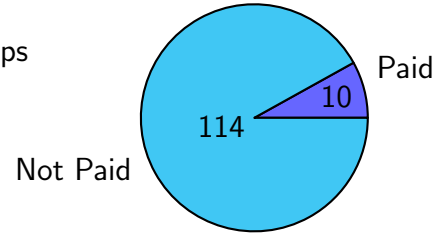
Outline

Attack Analysis
Problem Areas

- More than 45.000 servers compromised in total since 2016
- BinaryEdge: 124 companies and institutions were victims between 3rd January 2017 and 15th January 2017



a) 11.3% of victims had recent database backups, 88.7% had not



b) 114 victims did not pay ransom, 10 did: no response

Attack Analysis
Threat Model

1. Brute Force Password / Connect
2. Execute SQL Statements (Malicious Query Sequence, varying)
   - List Databases
   - Drop Databases
   - Create Database (e.g. 'PLEASE_READ')
   - Create Table (e.g. 'WARNING')
   - Insert Ransom Message
3. Disconnect

Attack Analysis
Threat Model

1. Brute Force Password / Connect
2. Execute SQL Statements (Malicious Query Sequence, varying)
   - List Databases
   - Drop Databases
   - Create Database (e.g. 'PLEASE_READ') $\Leftarrow$ not indicating
   - Create Table (e.g. 'WARNING')
   - Insert Ransom Message
3. Disconnect

Requirement Analysis

Requirements

- System, that tracks the executed queries
- Backup dropped (permanent) Databases/Tables
- Hide backed up database tables and DIMAQS information from unprivileged users
- Allow authentication for privileged mode
- Restore backed up database tables by privileged users
- Notify Administrator about incidents

Outline

Background
Colored Petri net (CPN)

- Query Sequence Analysis
- Colored Petri net (CPN) consists of
  - Places (Circles)
  - Transitions (Bars)
  - Tokens (Dots)
  - Arcs
- ShieldFS
  - Backup data at the right time
  - Copy data to a safe storage space

Single Query Analysis (SQLi)

Benign Query: `SELECT * FROM A WHERE id = 3`

Malicious Query: `SELECT * FROM A WHERE id = 3 OR 1 = 1`

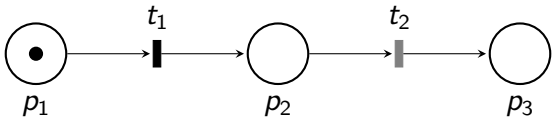
Query Sequence Analysis

1: `SELECT * FROM information_scheme.tables`

2: `CREATE TABLE A ...`

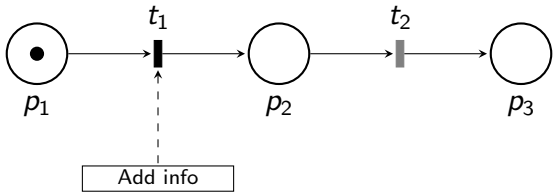
3: `INSERT INTO A ...`

Background
Colored Petri net (CPN)



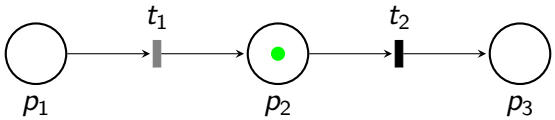
Token at p_1; Transition t_1 active

Background
Colored Petri net (CPN)



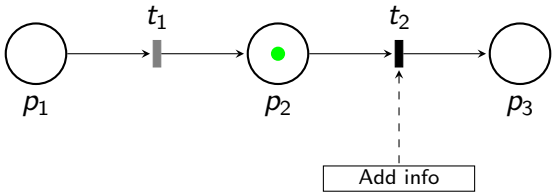
Transition t_1 fires and adds information to token from p_1

Background
Colored Petri net (CPN)



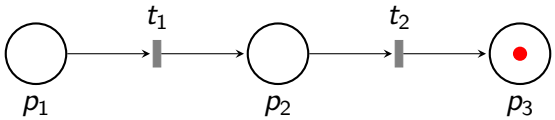
Token at p_2; Transition t_1 is disabled; Transition t_2 is enabled

Background
Colored Petri net (CPN)



Transition t_2 fires and adds information to token from p_2

Background
Colored Petri net (CPN)



one Token at p_3; Transition t_2 is disabled

Background
Colored Petri net (CPN) – Extensions & Addons

- Dynamic Color Creation based on added information
- Token Duplication
- Transition Action & Condition (additional query type and value checks)
- Transition Always Action (fires immediately when active)
- Place Action (e.g. Query Rewriting, Backup Databases)
- Token Merging
- Token Expiration (remove Tokens after a certain period of time)

ShieldFS

- Backup strategy inspired by ShieldFS
- Developed in July 2016 to cope with crypto ransomware
- Copy files on the fly when process is suspicious
- Acts on file system level

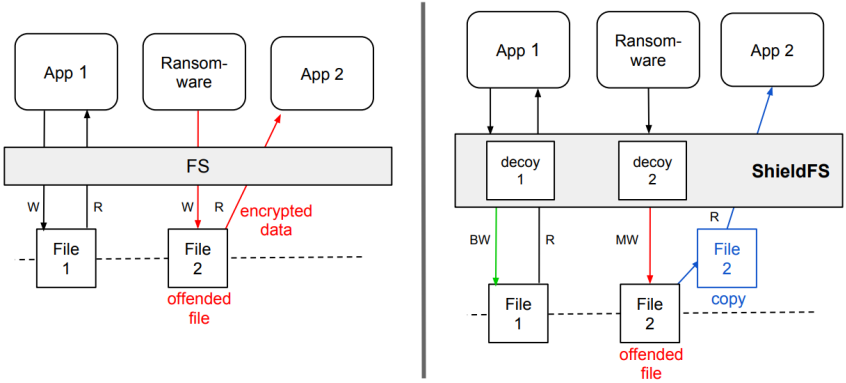


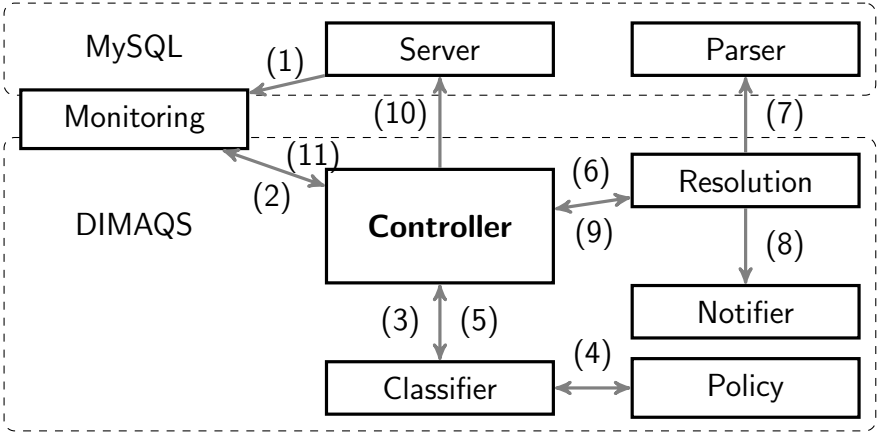
Figure: On the right ShieldFS shadowing a file offended by ransomware malicious write, in comparison to standard file systems (on the left)

Outline

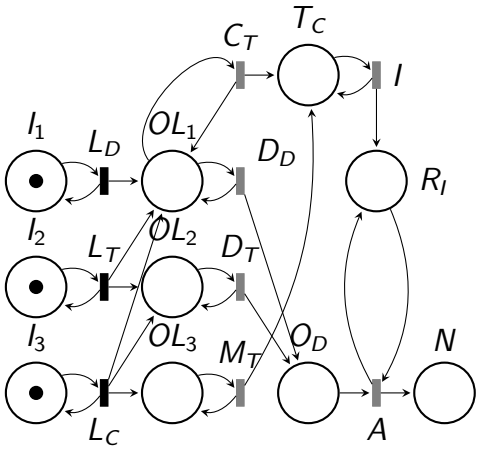
Proposed Solution
Details

- MySQL auditing plugin
- not limited to users and connections (global observation)
- CPN enhancements to reduce complexity and improve performance
- act on certain queries
  - move tables instead of dropping
  - notify administrator when attack detected
  - hide sensitive information (backed up data)
  - create triggers for newly created table

Proposed Solution
System Architecture

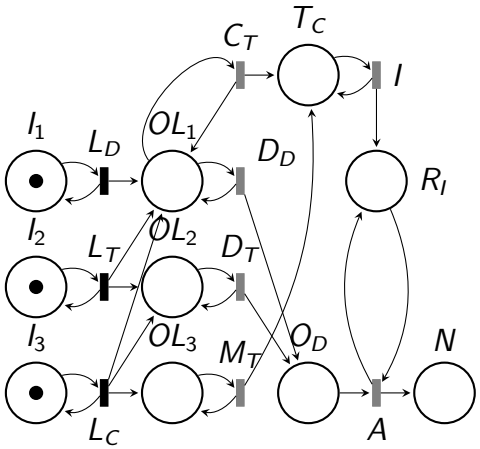


Proposed Solution
Classifier (Policy)



Place	Description
I_{1-3}	Initial places
OL_1	Object "Database" listed
OL_2	Object "Table" listed
OL_3	Object "Column" listed
T_C	Table created
O_D	Object "Database" or "Table" deleted
R_I	Ransom message inserted
N	Admin notification to be sent

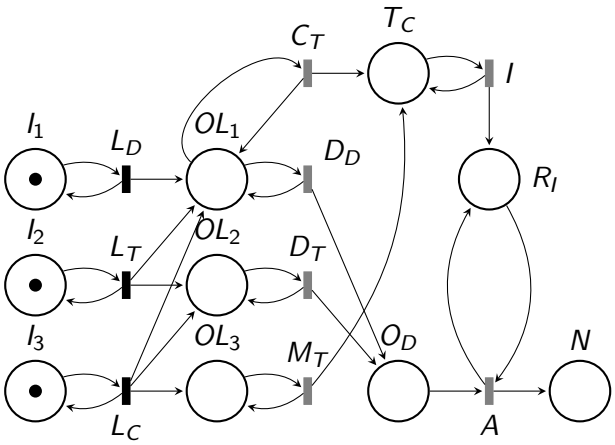
Proposed Solution
Classifier (Policy)



Transition	Description
L_D	List Databases
L_T	List Tables
L_C	List Columns
C_T	Create Table
D_D	Drop Database
D_T	Drop Table
M_T	Modify table
I	Insert ransom message
A	Always

Proposed Solution

Policy – Example



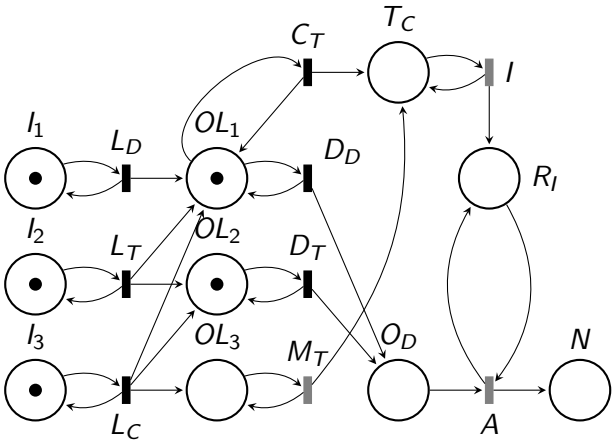
Initial places $\mathbf{I_{1-3}}$ contain one empty token.

Other places do not contain tokens.

Transitions $\mathbf{L_D}$, $\mathbf{L_T}$, and $\mathbf{L_C}$ are **active** and will be triggered on matching queries.

All other transitions are disabled.

Proposed Solution
Policy – Example



After transition $\mathbf{L_T}$ was triggered.

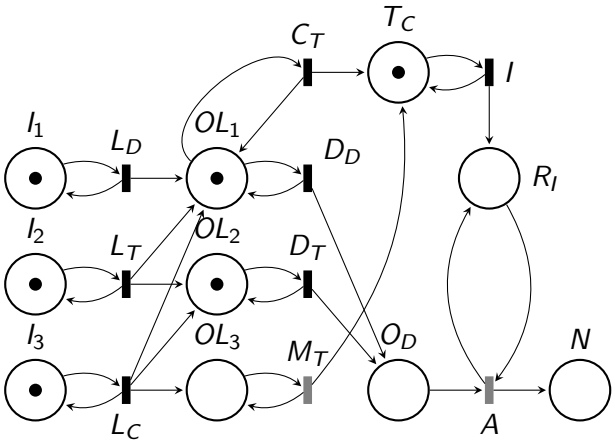
Initial states $\mathbf{I_{1-3}}$ still contain tokens.

Token from $\mathbf{I_2}$ is **transferred to** the places $\mathbf{OL_1}$ and $\mathbf{OL_2}$.

Tokens contain transition information which tables and databases were listed.

The **transitions** $\mathbf{C_T}$, $\mathbf{D_D}$, and $\mathbf{D_T}$ become **active**.

Proposed Solution
Policy – Example

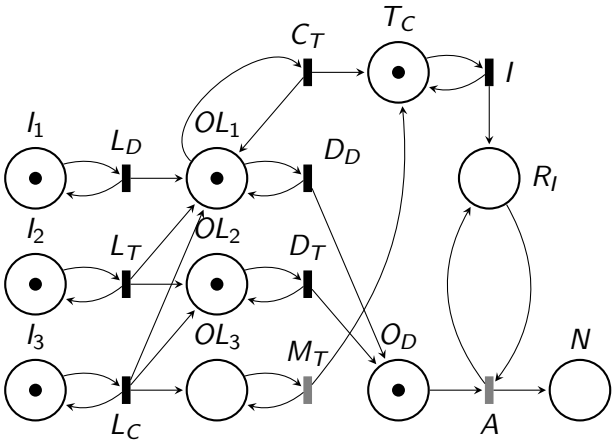


After firing $\mathbf{C_T}$. Tokens from $\mathbf{OL_1}$ are copied to $\mathbf{T_C}$.

$\mathbf{C_T}$ adds information about the created table to the transferred tokens.

Transition **I** becomes active.

Proposed Solution
Policy – Example

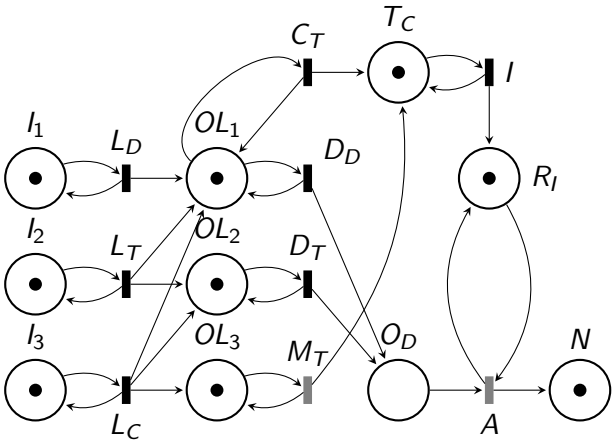


After firing $\mathbf{D_T}$. Tokens from $\mathbf{OL_2}$ are copied to $\mathbf{O_D}$.

Transition $\mathbf{D_T}$ adds information about the dropped Table to the transferred tokens.

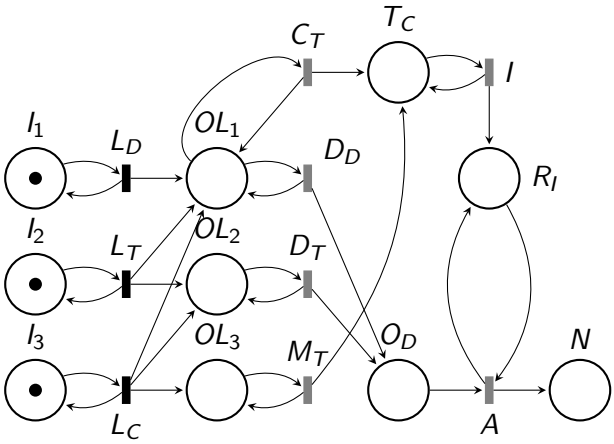
A does not become active because of $\mathbf{R_I}$.

Proposed Solution
Policy – Example



After firing **I**. Tokens from $\mathbf{T_C}$ that match the Table name are transferred to $\mathbf{R_I}$. **I** adds information about inserted message.
A becomes active and **fires** immediately **until** $\mathbf{O_D}$ does not contain tokens anymore. Token values from $\mathbf{R_I}$ are **merged with** the token values from $\mathbf{O_D}$.

Proposed Solution
Policy – Example



After token expiration.

Outline

Self generated: False Negatives

- 13485 tests

False Positives

- Bibspace
  - Query logs from 13th of April 2018 to 22nd of May 2018
  - contains 52085 queries
- MediaWiki
  - Query logs from 3rd of April 2018 to 22nd of May 2018
  - contains 2514764 queries

False Negatives

- no false negatives occurred
- 100% detection rate
- expected, since policy is designed to capture attacks from the malicious data set

False Positives

Query set	I_1	I_2	I_3	OL_1	OL_2	OL_3	T_C	O_D	RI	N
Bibspace	1	1	1	2	2	0	24	0	0	0
MediaWiki	1	1	1	7	5	1	0	0	0	0

No false positives occured

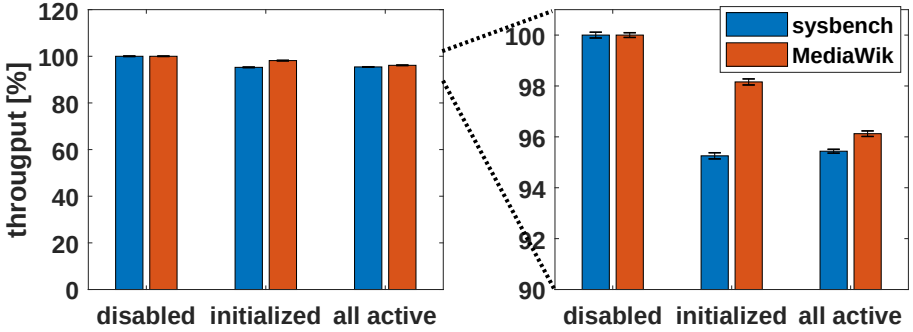


Figure: Performance influence of DIMAQS for sysbench and MediaWiki. Values are normalized to the respective value for the disabled plugin.

Limitations

- Variations of BitCoin values
- Renaming of tables during classification
- Capturing of new attack forms requires adjustments of the policy
- Possibility to fill up secure storage space

Outline

Conclusion & Future Works

Conclusion

- Ransomware is an emerging threat
- DB Ransomware attacks have severe consequences as attackers do not always create dumps.
- DIMAQS uses a colored Petri net–based classifier
- DIMAQS implemented as MySQL plugin
- allows to reduce complexity of system representation
- performance overhead below 5%

Conclusion & Future Works

Future Works

- Token Merging functionality needs enhancements to increase performance and reduce Notifications
- Trace table renaming during classification
- Detect other malicious query sequences (`INTO_OUTFILE`)

Resources

If you want to improve this style

- BinaryEdge Attack analysis
  `https://docs.google.com/spreadsheets/d/1QonE9oeMOQHVh8heFIyeqrjfKEViL0poLnY8mAakKhM/`
- ShieldFS
  `http://shieldfs.necst.it/continella-shieldfs-2016.pdf`

Questions ?

Michael Jobs
michael.jobst@stud-mail.uni-wuerzburg.de