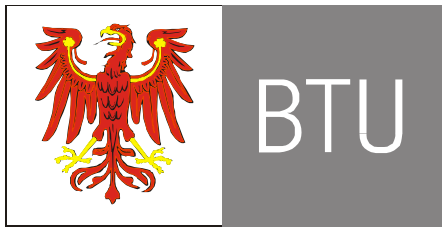
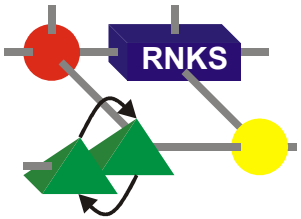



# Improving the Efficiency of Misuse Detection

DIMVA 2005

Michael Meier, Sebastian Schmerl, Hartmut König

Brandenburg University of Technology Cottbus, Germany
Computer Networks and Communication Systems Group
E-Mail: {mm,sbs,koenig}-/at/-informatik.tu-cottbus.de

# Overview

- Motivation
- Modeling Complex Signatures
- Existing Analysis Approaches
- Optimizing Strategies
- Experimental Evaluation
- Conclusion

# Motivation

- increasing performance of networks and end systems
  - ⇨ increase of data volumes to be analyzed
- increasing complexity of networks and systems
  - ⇨ increasing number of attack signatures

⇨ delayed detection of and response to attacks

⇨ IDS drop data in high load situations

⇨ Efficiency of IDS analysis methods becomes more important

# Terms



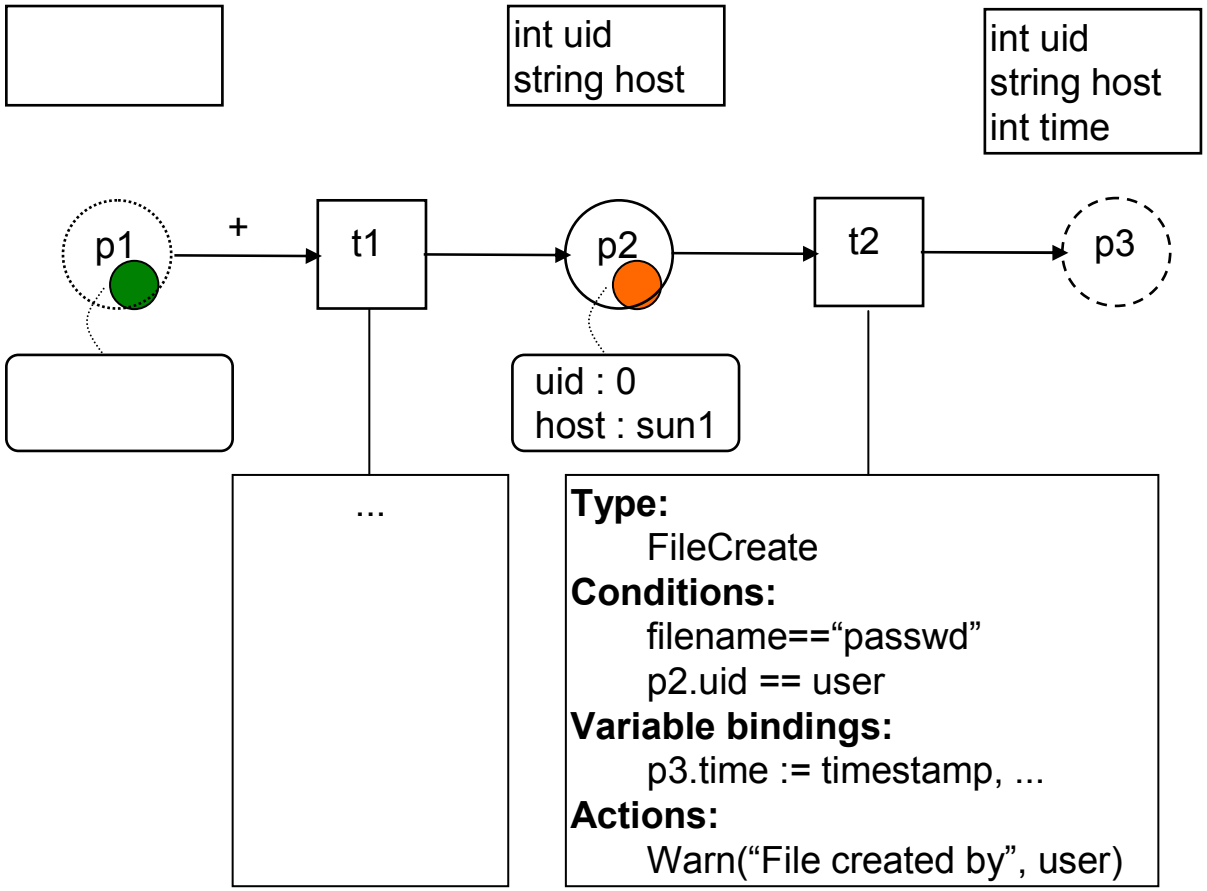
- **Manifestation** of an attack
  - Audit data records generated during this attack (trace)
- **Signature** of an attack
  - Criteria used to identify the manifestation of this attack (type) within the audit data stream (patterns)
- **Signature Instance** of an attack
  - Criteria used to identify the manifestation of a particular instance of this attack (type) within the audit data stream

# Modeling Complex Signatures

- EDL – Event Description Language
- based on high-level Petri Nets
- Basic Concepts
  - Place – system state of an attack
    - defines a set of features
  - Transition – state changes
    - triggered by an event
  - Event – security relevant action
    - associated with a transition
  - Token – instances of a signature
    - contain bound variables for each feature defined by the place it resides on

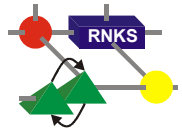
# Modeling Complex Signatures (II)

- Places features
- Token variables
- Transition labels
- Event features




| FileCreate | user:0 | filename:passwd | timestamp:42 | ... |
|---|---|---|---|---|


# Existing Analysis Approaches



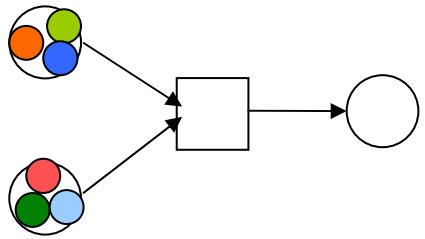

- Program modules for signatures
  - Examples: IDIOT, STAT
  - Signatures are translated into C++ class modules
  - A class instance for each signature instance
  - Run time: events are passed to each class instance
    - independent analysis of each signature instance
    - redundant calculations
    - ⇨ arguable efficiency

# Existing Analysis Approaches (II)

- Expert systems
  - Examples: CMDS (CLIPS), Emerald (P-Best), AID (RT-Works)
  - Translation of signatures into rules and facts
    - tokens and current event represented as facts
    - transitions implemented as rules
  - Optimized match algorithms (e.g. RETE)
    - Avoid redundant calculations (by common sub-expression elimination)
    - Exploit the assumption that fact changes are rare
      - the current event fact is continuously changed
      - validity seems to be doubtful

# Optimizing Signature Analysis Strategies

- Starting point
  - naive analysis procedure for EDL signatures
    - check all transitions of all signatures for each incoming event x
    - For each transition
      - check event type
      - check transition conditions for each combination of tokens on input places and event x
    - ⇨ number of tokens grows during operation
    - ⇨ performance cost increases



- Exploit structural characteristics of signatures to improve performance

# Strategy 1: Type Based Transition Indexing

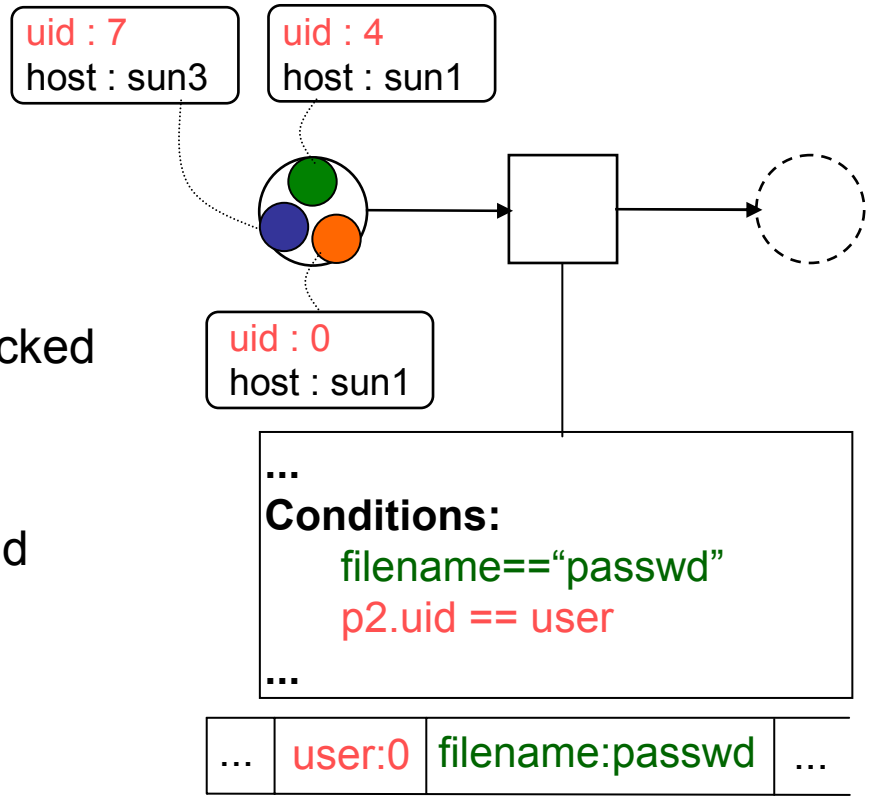
- Problem
  - all transitions are checked for the event type
- Solution
  - create an event type subscription table by static analysis of signatures

| Event Type | Transitions |
|---|---|
| X | t1, t4, t7, t12 |
| Y | t3, t6, t9, t11 |
| Z | t2, t5, t8, t10 |

  - allows to efficiently determine all transitions associated with a given event type
- Additional costs at run time: 0

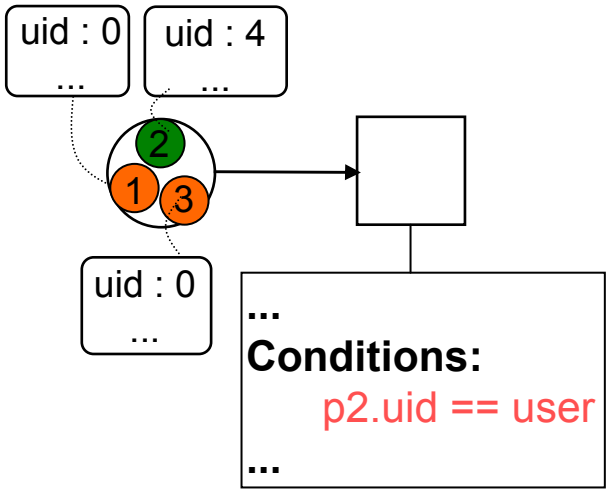
# Strategy 2: Instance Independent Condition Testing

- Problem
  - repeated evaluation of transition conditions
- Solution
  - distinction between Intra- and Inter-Event Conditions
  - Intra-Event Conditions are independent of tokens on input places and need to be checked only once for an event
  - only if Intra-ECs are fulfilled, Inter-ECs are checked for any combinations of tokens and the event
- Additional costs at run time: 0



# Strategy 3: Value Based Indexing of Tokens

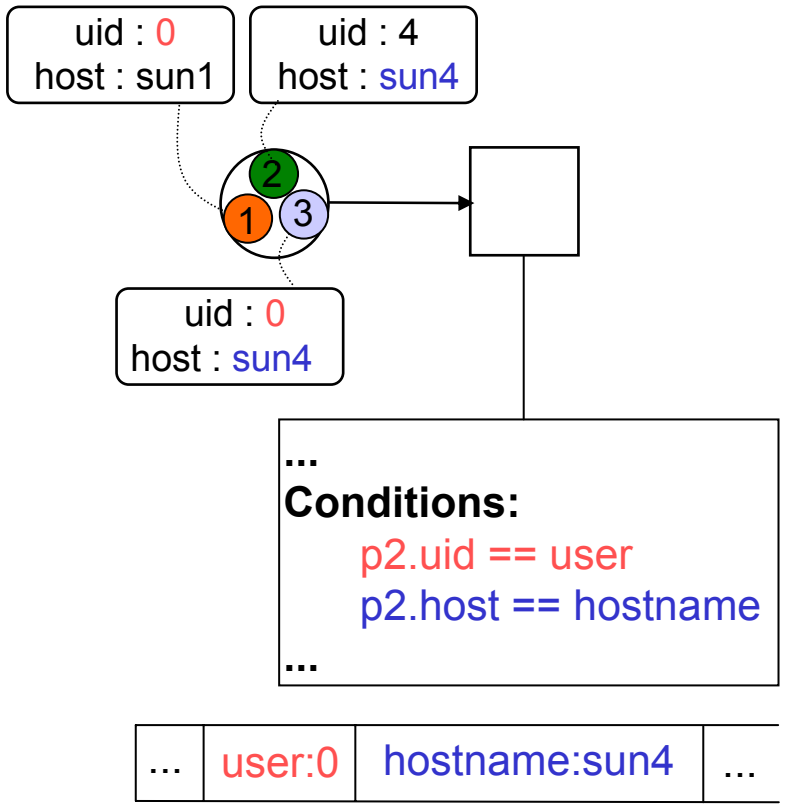
- Problem
  - many tokens or token combinations have to be checked
- Solution
  - analyze comparison operations in Inter-ECs
  - manage value tables for token variables
  - select matching tokens using value tables
  - combine conditions using set operations




- Additional costs at runtime:
  - table updates if tokens move
  - set operations

| p2.uid | |
|---|---|
| **Value** | **Tokens** |
| 0 | 1, 3 |
| 4 | 2 |

# Strategy 3: Value Based Indexing of Tokens (II)




| p2.uid | |
|---|---|
| **Value** | **Tokens** |
| 0 | 1, 3 |
| 4 | 2 |

⇨ {1, 3}

| p2.host | |
|---|---|
| **Value** | **Tokens** |
| sun1 | 1 |
| sun4 | 2, 3 |

⇨ {2, 3}

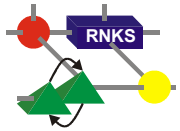
$\{1, 3\} \cap \{2, 3\} = \mathbf{\{3\}}$

- also works for Inter-ECs like p2.uid == p3.owner (see paper)
- can be realized for other comparison operators e.g. <, >

# Strategy 4: Identification of Common Sub-Expressions

- Problem
  - identical (sub-)conditions of different transitions are evaluated repeatedly with same parameters
- Solution
  - cache and reuse results of already evaluated conditions
  - identical conditions are evaluated only once with same parameters
  - parameters of Intra-ECs change only for a new event
    - cached results are valid until the next event
  - parameters of Inter-ECs differ for different token combinations
    - cached results are valid only for one token combination
- Additional costs at run time: cache management


# Strategy 5: Cost-based Prioritization of Conditions




- Problem/Fact
  - different costs and selectivity of conditions
- Solution/Exploitation
  - conditions are often or mostly evaluated false
    - check mostly false conditions first may avoid other conditions checks
  - condition checks require different run-times
    - evaluating cheap conditions first may avoid expensive checks
  - condition prioritization
    - static – based on run-time estimations
    - dynamic – based on regular run-time and selectivity measurements

# Prototype: SAM - Signature Analysis Module

- Six different versions realize different strategy combinations

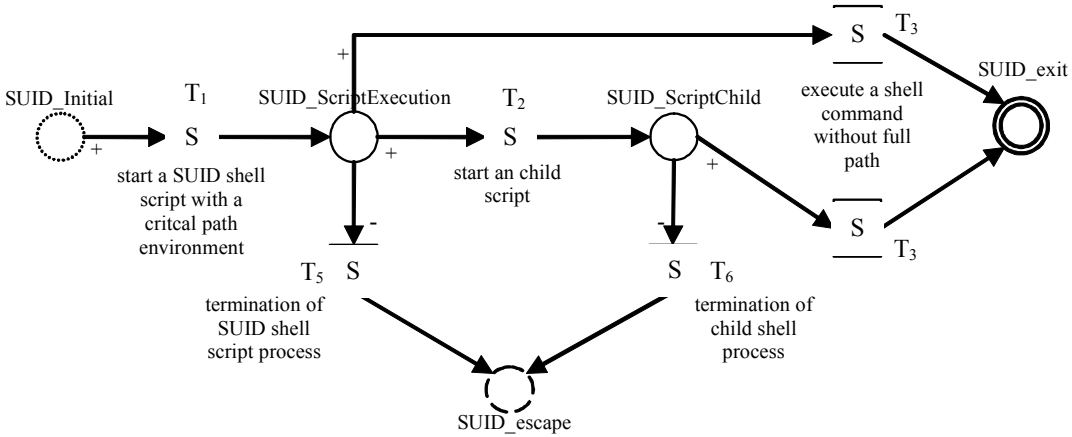
| SAM Version | Realized Strategies |
|---|---|
| SAM_1 | 1 |
| SAM_2 | 1, 2 |
| SAM_3 | 1, 2, 4 |
| SAM_4 | 1, 2, 3, 4 |
| SAM_5 | 1, 2, 3, 4, 5 (static) |
| **SAM_6** | **1, 2, 3, 4, 5 (dynamic)** |

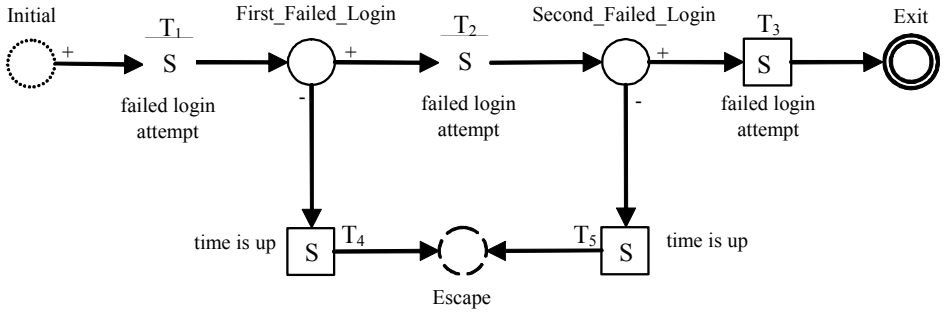
# Experimental Evaluation: Test Candidates

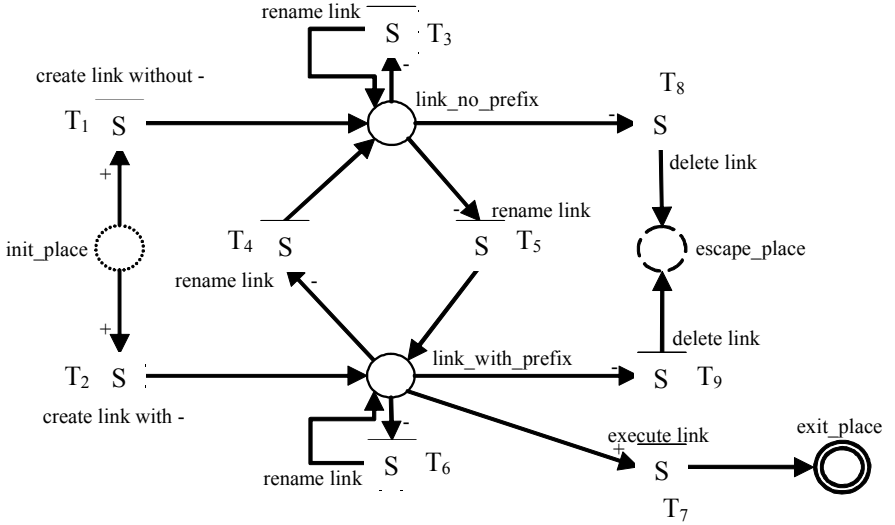
- (For an experimental comparison of SAM versions see paper)
- STAT
  - from UC Santa Barbara
  - realizes the program modules for signatures approach
- CLIPS-IDS
  - prototype of an Expert System based IDS
  - uses the RETE-based Expert System CLIPS
- SAM_6

# Experimental Evaluation: Test Environment

- Three signatures
  - Shell Link Attack
  - SUID Script Attack
  - Login Attack



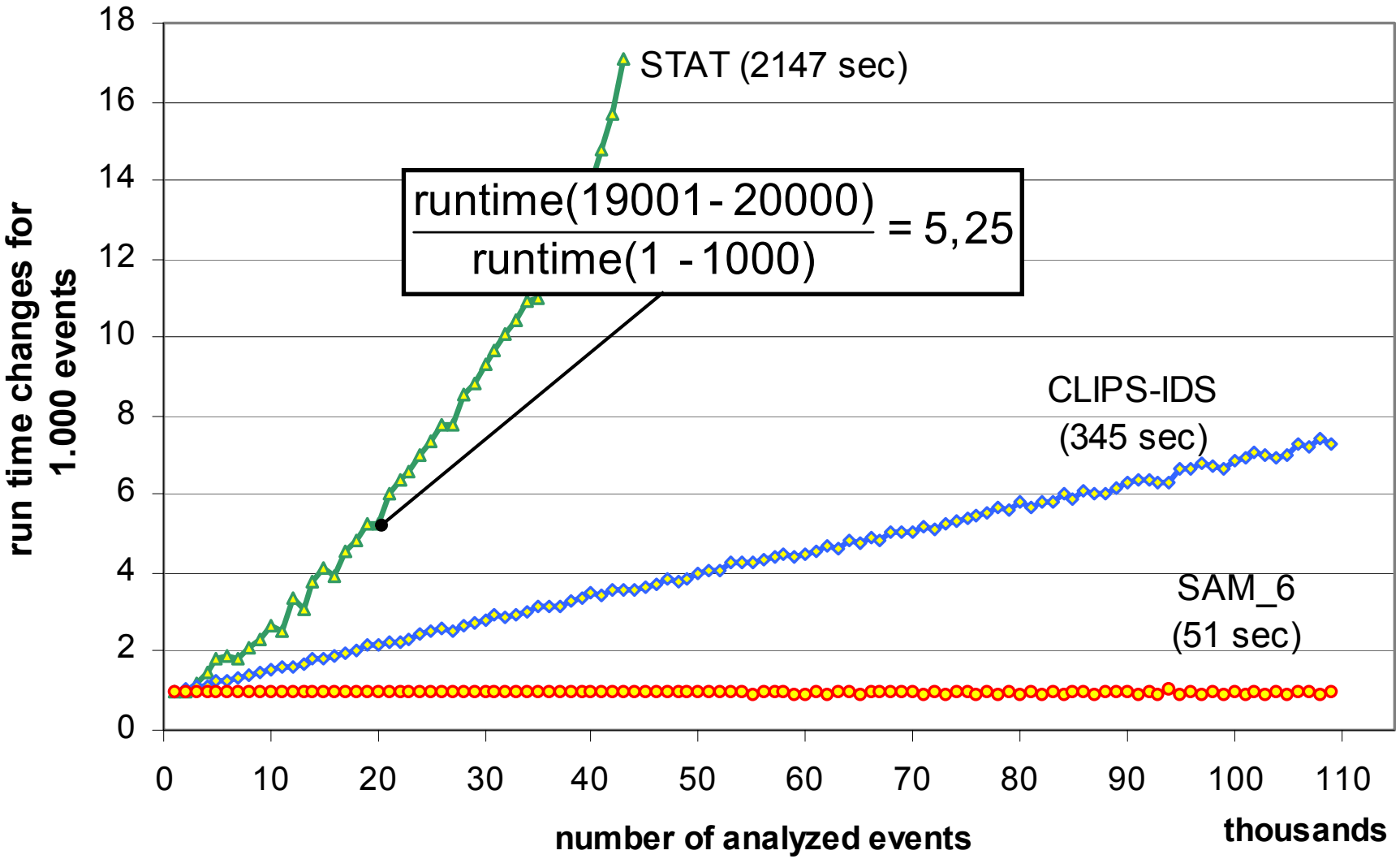



# Experimental Evaluation: Test Environment

- Test data
  - ten concurrent instances of each of the three attacks
  - repeated a 1000 times
  - 110000 audit records (BSM style audit data)
  - ⇨ number of ongoing instances grows with each repetition
- Measurements
  - consumed run times are logged every 1000 events (records)
- Comparison
  - run time changes for growing number of analyzed events

# Experimental Evaluation: Results



# Conclusions

- Use of
  - standard techniques and
  - exploitation of structural characteristics of signatures
  - ⇨ can avoid redundant/useless calculations

⇨ Can significantly improve efficiency

# Thank you!

Contact:

Michael Meier

Brandenburg University of Technology Cottbus

Email: mm-/at/-informatik.tu-cottbus.de

WWW: http://www-rnks.informatik.tu-cottbus.de/~mm

⇨ post-doc opportunities are welcome