Data Driven Probabilistic Graphs for Preemptive Attack Detection

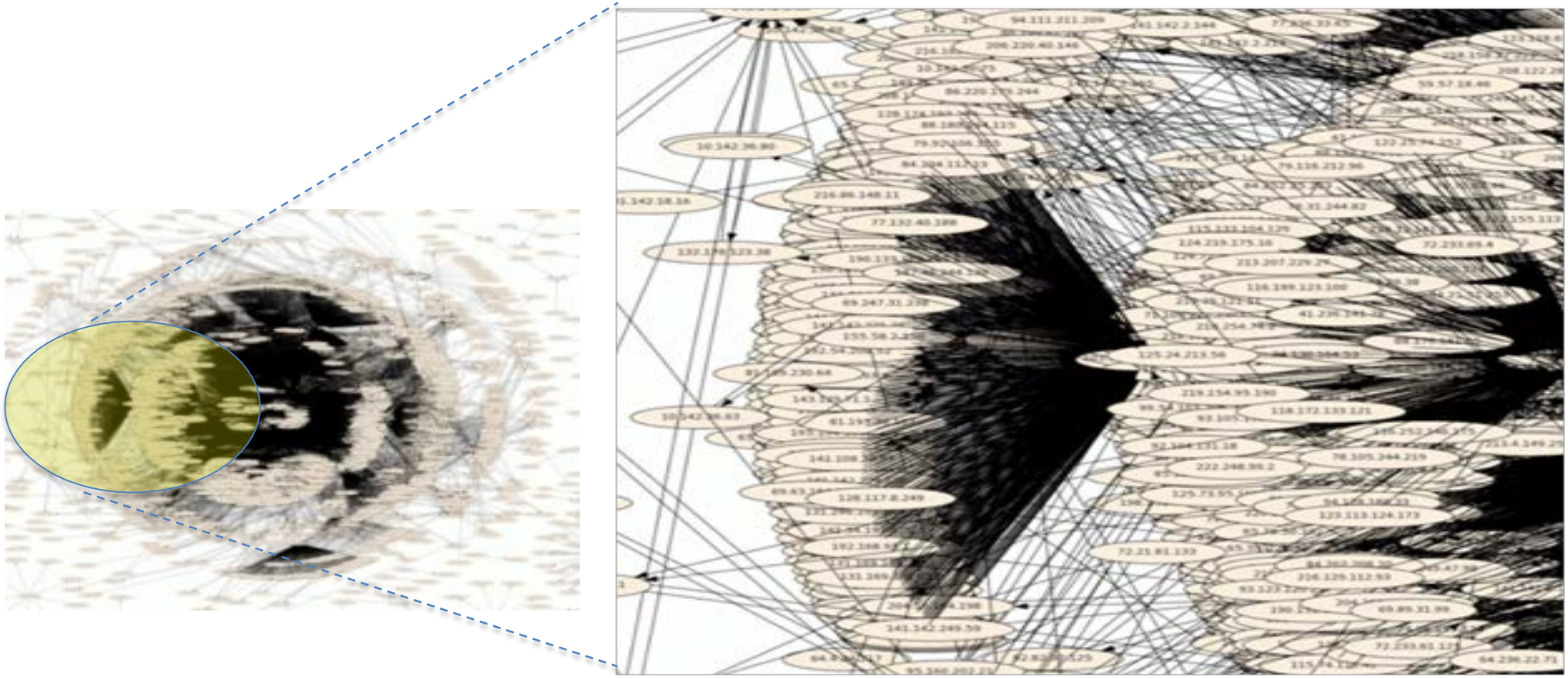
Zbigniew Kalbarczyk

Collaborators: Phuong Cao, Adam Slagel, Ravi K. Iyer

ECE ILLINOIS

ILLINOIS

Magnitude of the Problem



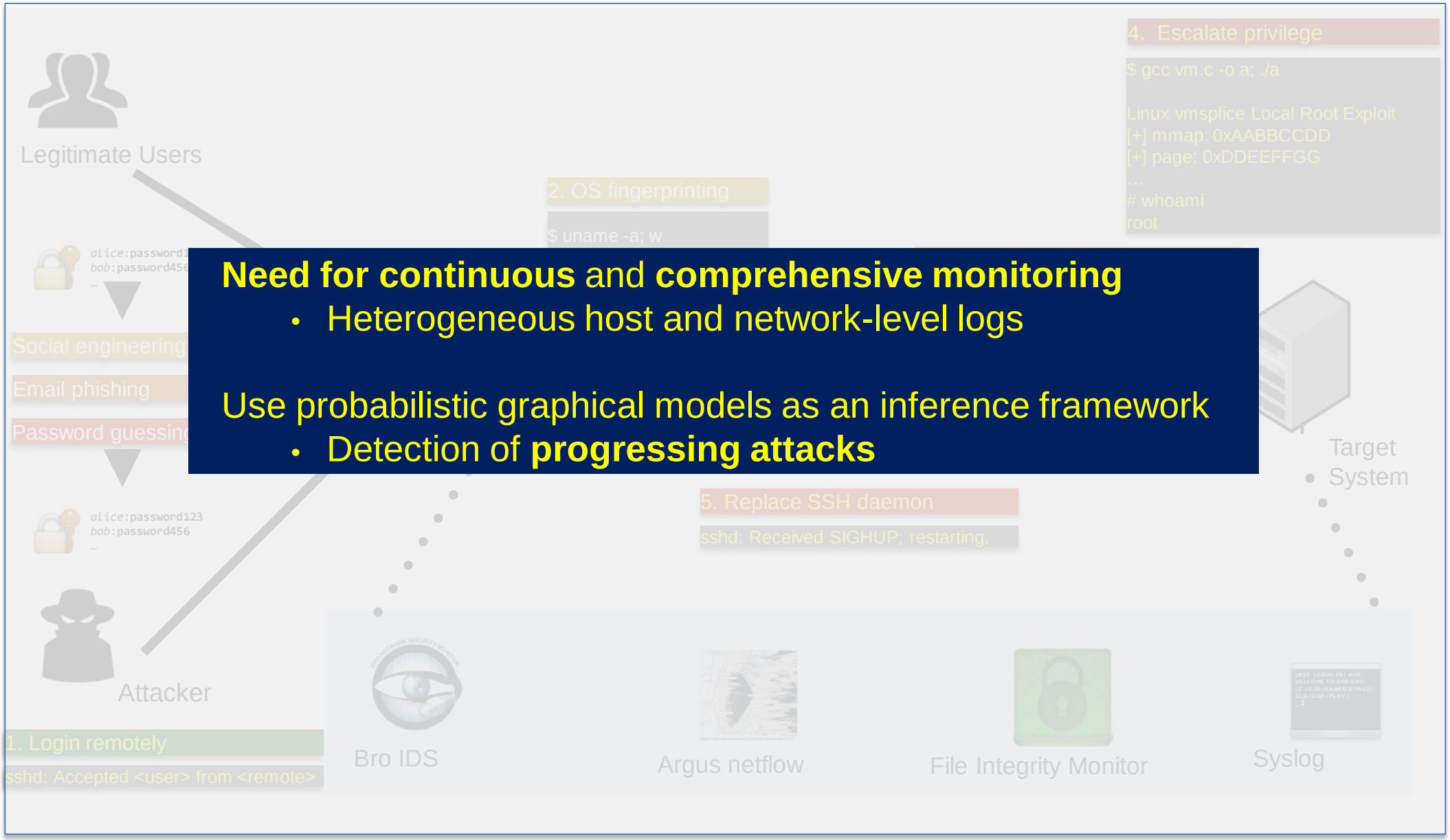
Five-Minute Snapshot of In-and-Out Traffic within NCSA

Challenge

- Leveraging security logs to enable timely attack detection and effective corrective/recovery actions.

- Why is this hard?
  - ✓ huge in-and-out network traffic rates;
  - ✓ format/semantic heterogeneity of detectors;
  - ✓ several GBs/day of data;
  - ✓ false positives;
  - ✓ need to correlate multiple sources to obtain the “big picture”;
  - ✓ analysis is mainly *manual.*

Multi-Stage Attack



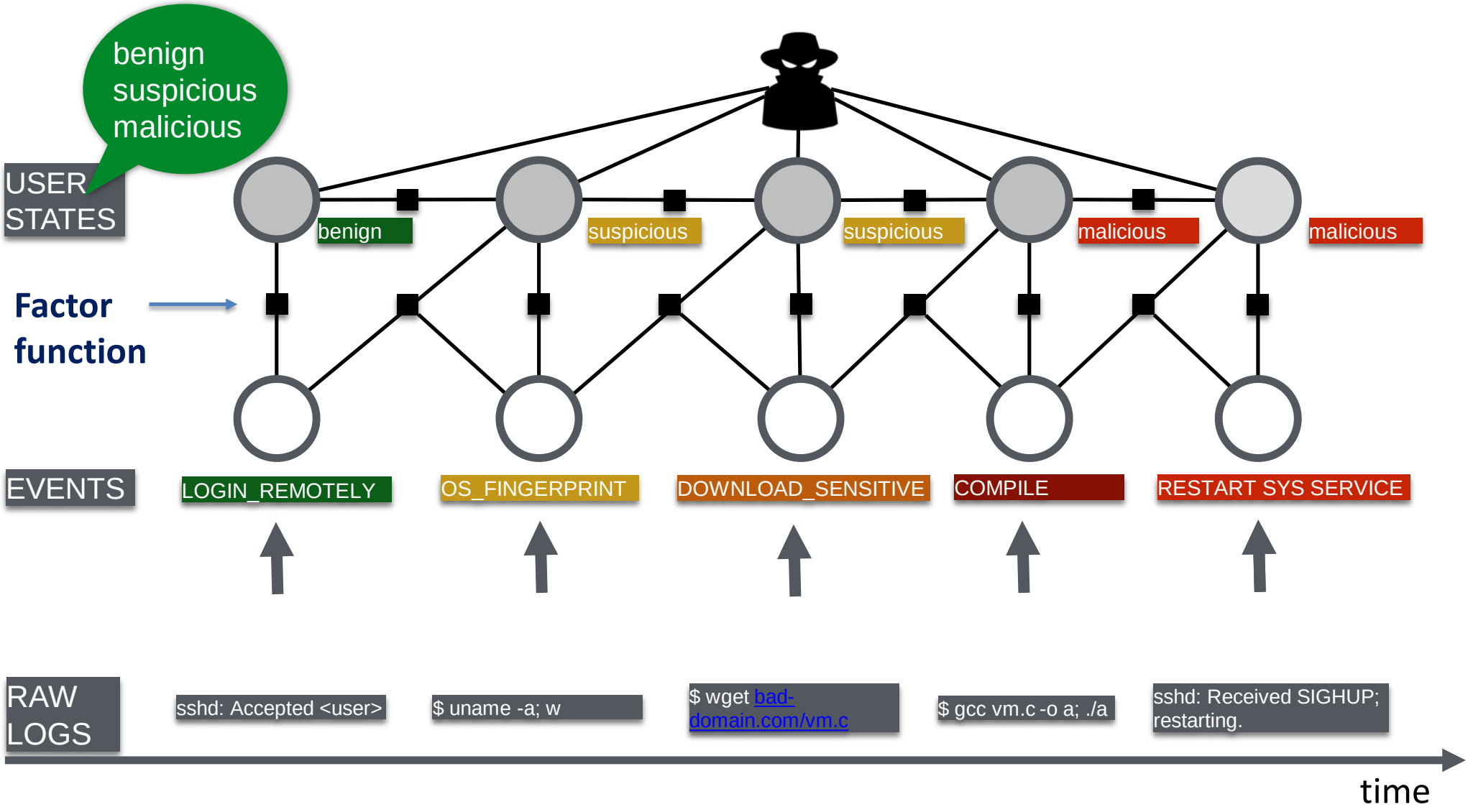
Need for continuous and **comprehensive monitoring**

- Heterogeneous host and network-level logs

Use probabilistic graphical models as an inference framework

- Detection of **progressing attacks**

From Security Logs to Probabilistic Graphical Models: Factor Graphs



Factor Graph Representation of an Example Incident

Known random variables
event e^1 = download sensitive
event e^2 = restart system service
user profile u: past_compromise = true

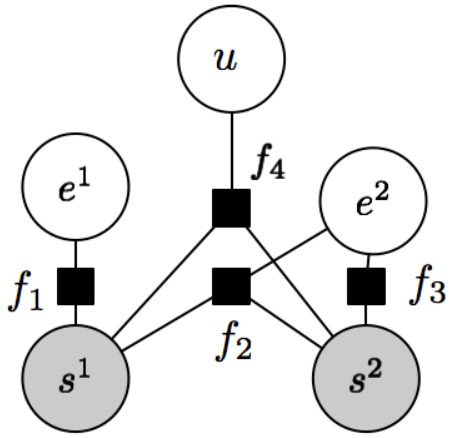
Unknown random variables
state s^1: user state when observing e^1
state s^2: user state when observing e^2

State inference

Enumerate possible s^1, s^2 state sequences

benign, benign
benign, suspicious
benign, malicious,
…
malicious, malicious

An example Factor Graph



Score(s^1, s^2) is the sum of factor functions f_1, f_2, f_3 f_4

$$Score(s^1, s^2) = \sum f(c_f)$$

Most probable s^1, s^2 is suspicious, malicious

Definition of factor functions

$$f_1 = \begin{cases} 1 & \text{if } e^1 = download\ sensitive \\ & \& \ s^1 = suspicious \\ 0 & otherwise \end{cases}$$

$$f_2 = \begin{cases} 1 & \text{if } e^2 = restart\ service \\ & \& \ s^1 = suspicious \\ & \& \ s^2 = malicious \\ 0 & otherwise \end{cases}$$

$$f_3 = \begin{cases} 1 & \text{if } e^2 = restart\ sys\ service \\ & \& \ s^2 = benign \\ 0 & otherwise \end{cases}$$

$$f_4 = \begin{cases} 1 & \text{if } s^{t-1} = suspicious \\ & \& \ s^t = malicious \\ & \& \ u = past\ compromise \\ 0 & otherwise \end{cases}$$

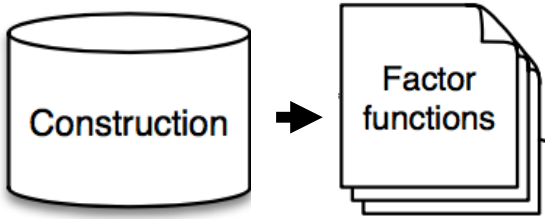
AttackTagger Workflow

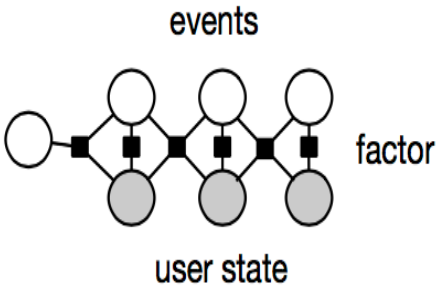
Construct factor functions based on past incidents

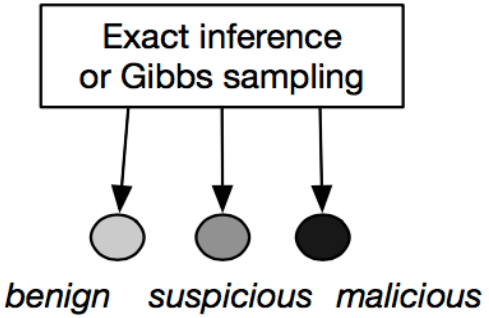
Extract events corresponding to an incident

Construct per-user factor graph

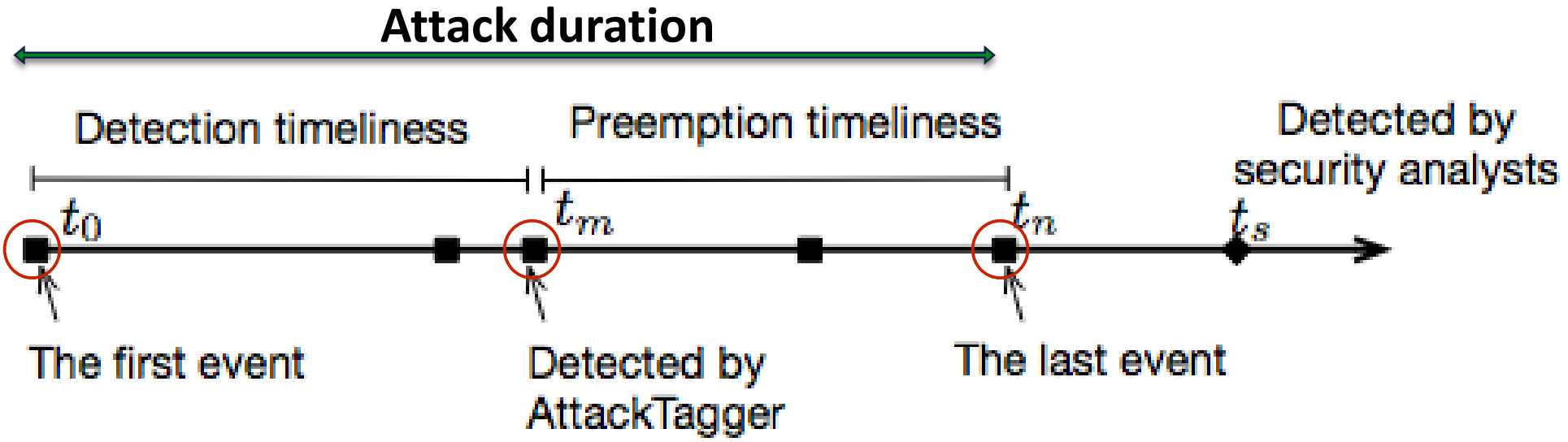
Infer the user states



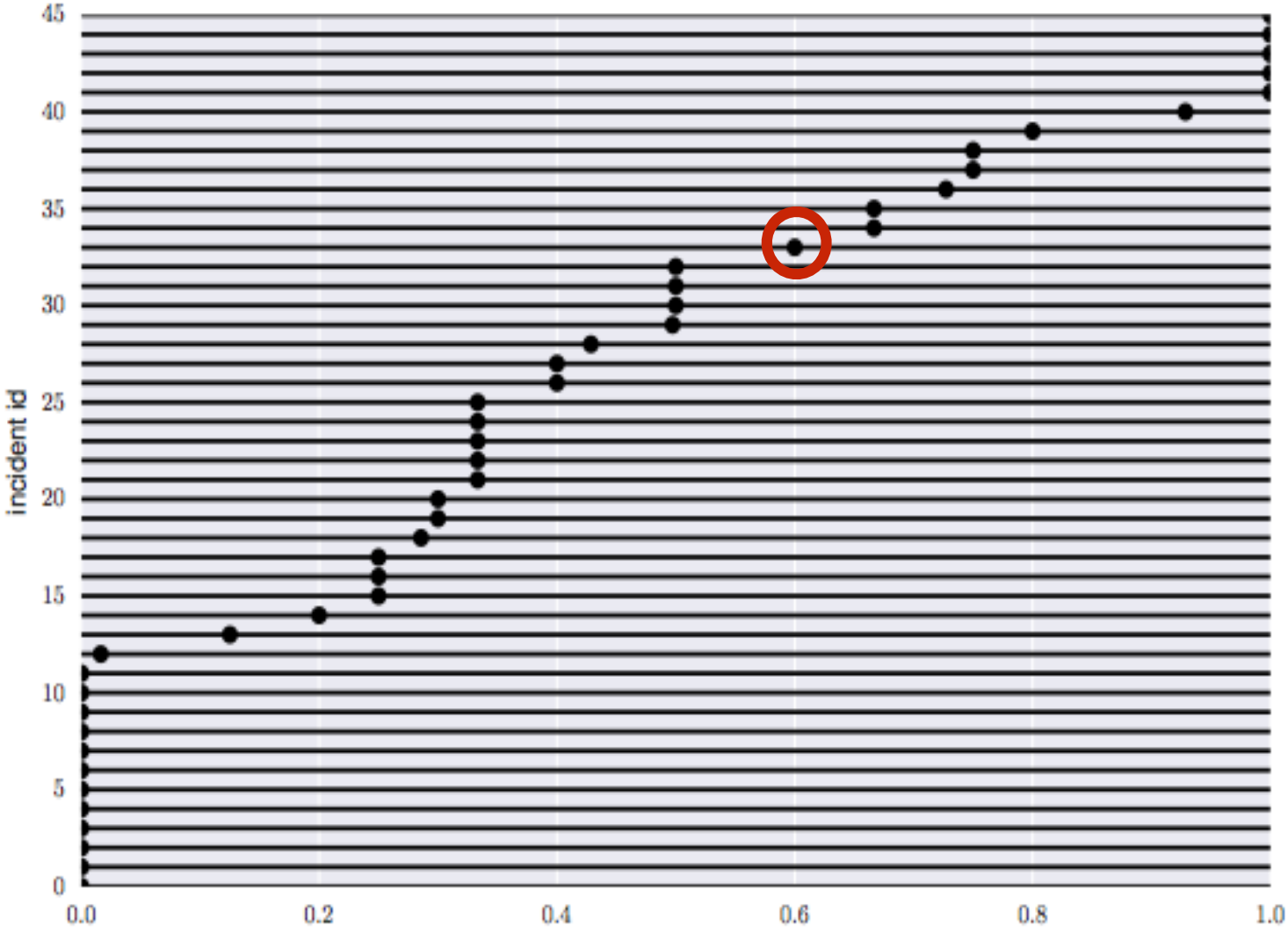




Metrics: Detection timeliness & Preemption timeliness



Detection Timeliness & Preemption Timeliness



46 of 62 malicious users were detected in tested incidents (74%)

41 of 46 identified malicious users were identified before the system misuse

Detection Performance Comparison

Name	*TP*	*TN*	*FP*	*FN*
AttackTagger	*74.2*	*98.5*	*1.5*	*25.8*
Rule Classifier	9.8	96.0	4.0	90.2
Decision Tree	21.0	100.00	0.00	79.0
Support Vector Machine	27.4	100.00	0.00	72.6

Statistical test shows that performance of AttackTagger
is better than Support Vector Machine (SVM) not by chance

- Best detection rate (46 of 62 malicious users)
- Small false detection rate (19 users of 1267 benign users)
- Captures hidden malicious users not identified in incident reports

Conclusions

- **Factor graph is a suitable representation of user/system state transitions in security incidents.**

- **Experimental evaluation of factor graph shows that a majority compromised users (74%) can be detected in advance (minutes to hours before the system misuse)**

- **Our approach can detect a variety of attacks, including hidden attacks that went unidentified by security analysts.**