Preemptive Intrusion Detection: Theoretical Framework and Real-world Measurements

Phuong Cao, Eric Badger, Zbigniew Kalbarczyk, Ravishankar Iyer, Adam Slagell

University of Illinois at Urbana-Champaign, National Center for Supercomputing Applications

ECE ILLINOIS

ILLINOIS

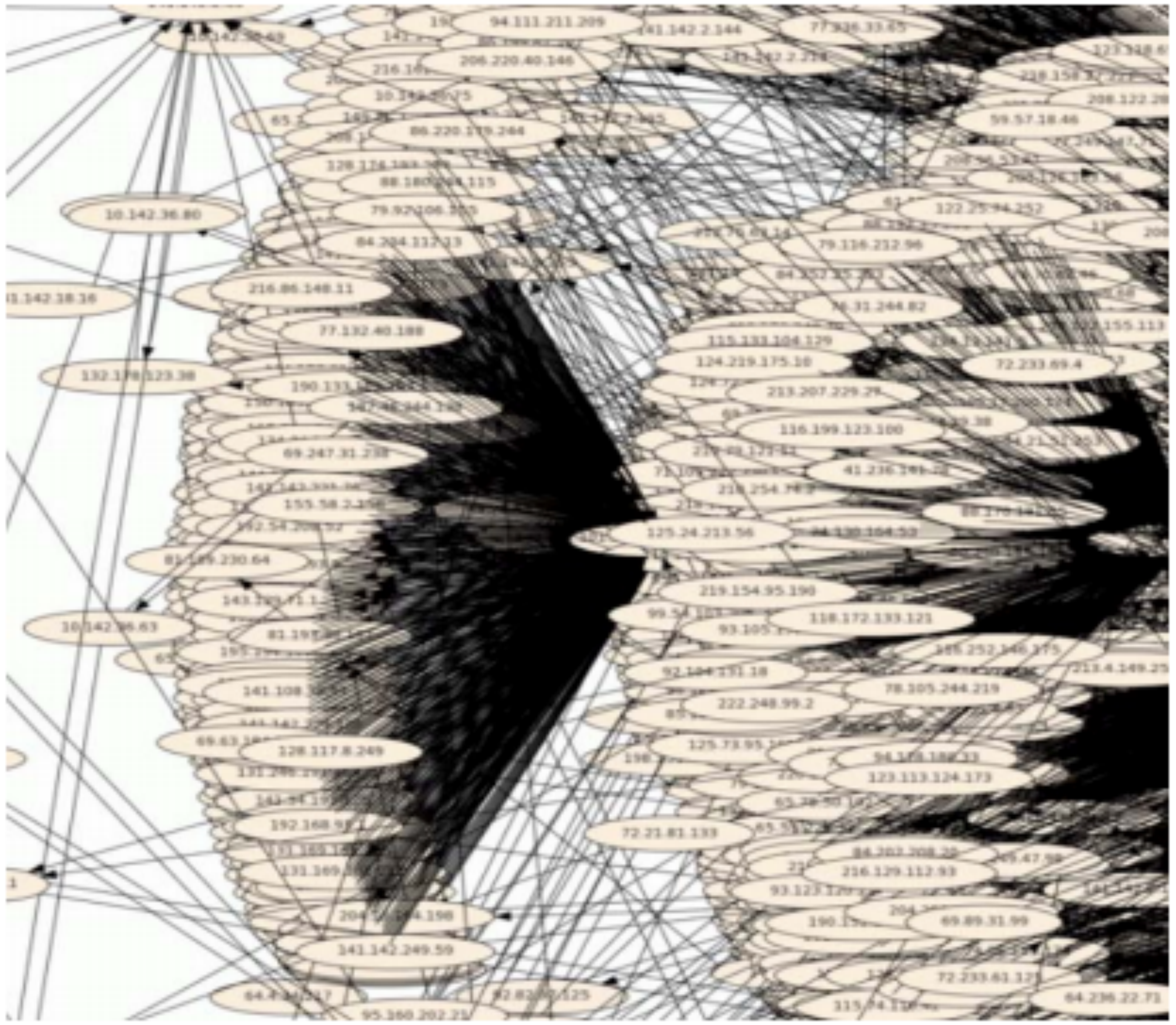
National Center for Supercomputing Applications

6000+
users

5+ millions
connections

34M+
log events

4.5+ GB
compressed log



5-minute snapshot of network traffic in and out of NCSA

Heterogeneous host and network logs

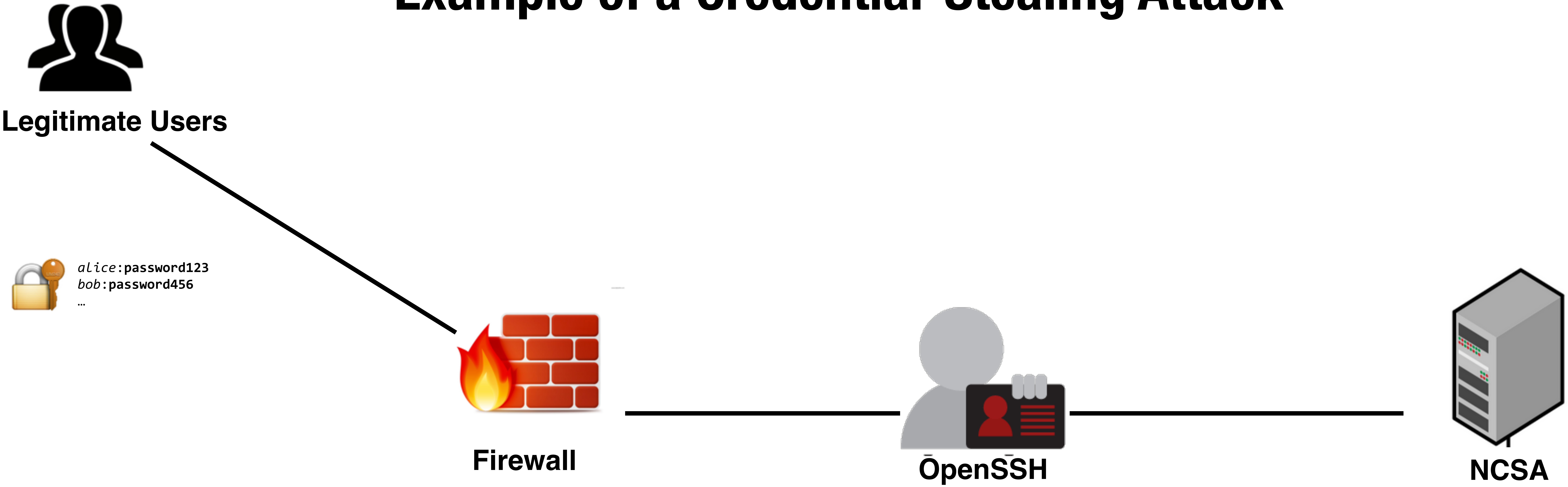
- Syslog
- Netflows
- IDS alerts
- Human-written reports

160 incidents in the past 7 years (2008-2014)

- Brute-force attacks
- Credential compromise
- Abusing computing infrastructure
  - Send spam
  - Launch Denial of Service attacks

NCSA

Example of a Credential-Stealing Attack



Legitimate Users

```
alice:password123
bob:password456
…
```

Firewall

OpenSSH

NCSA

Example of a Stolen Credential Attack

Legitimate Users

4. Escalate privilege

```
$ gcc vm.c -o a; ./a

Linux vmsplice Local Root Exploit
[+] mmap: 0xAABBCCDD
[+] page: 0xDDEEFFGG
…
# whoami
root
```

Continuous and **comprehensive monitoring**

- Heterogeneous host and network-level logs

Probabilistic graphical models as an inference framework

- Detection of **progressing attacks**

```
alice:password123
bob:password456
…
```

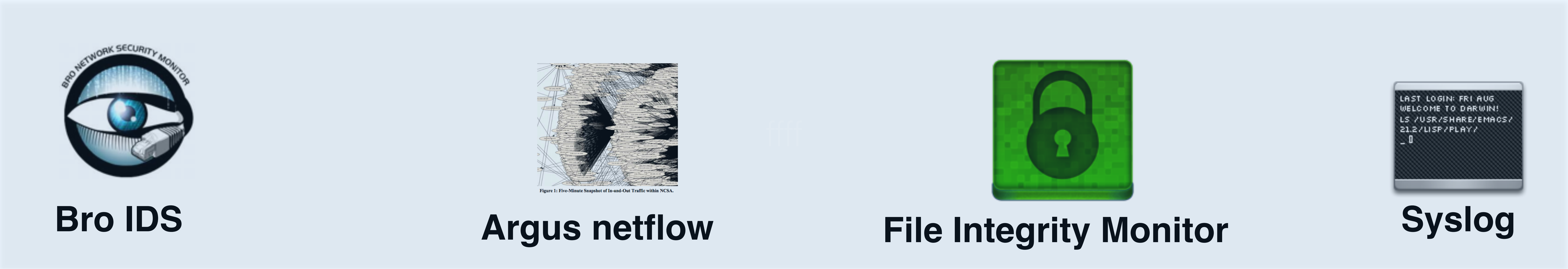
3. Replace SSH daemon

```
sshd: Received SIGHUP; restarting.
```

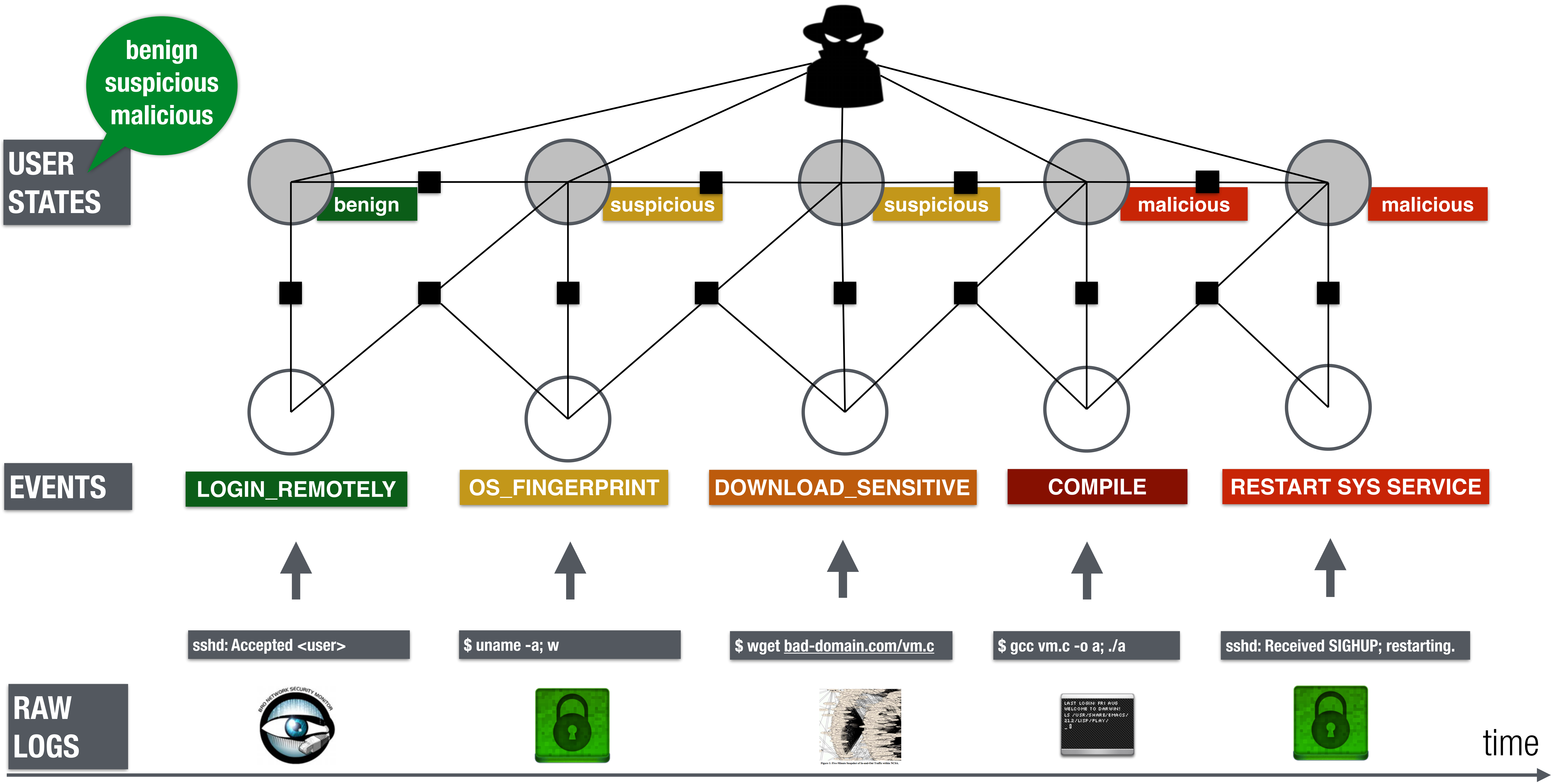
Attacker

1. Login remotely

```
sshd: Accepted <user> from <remote>
```



Integrating Heterogeneous Monitoring Data Using Probabilistic Graphical Models



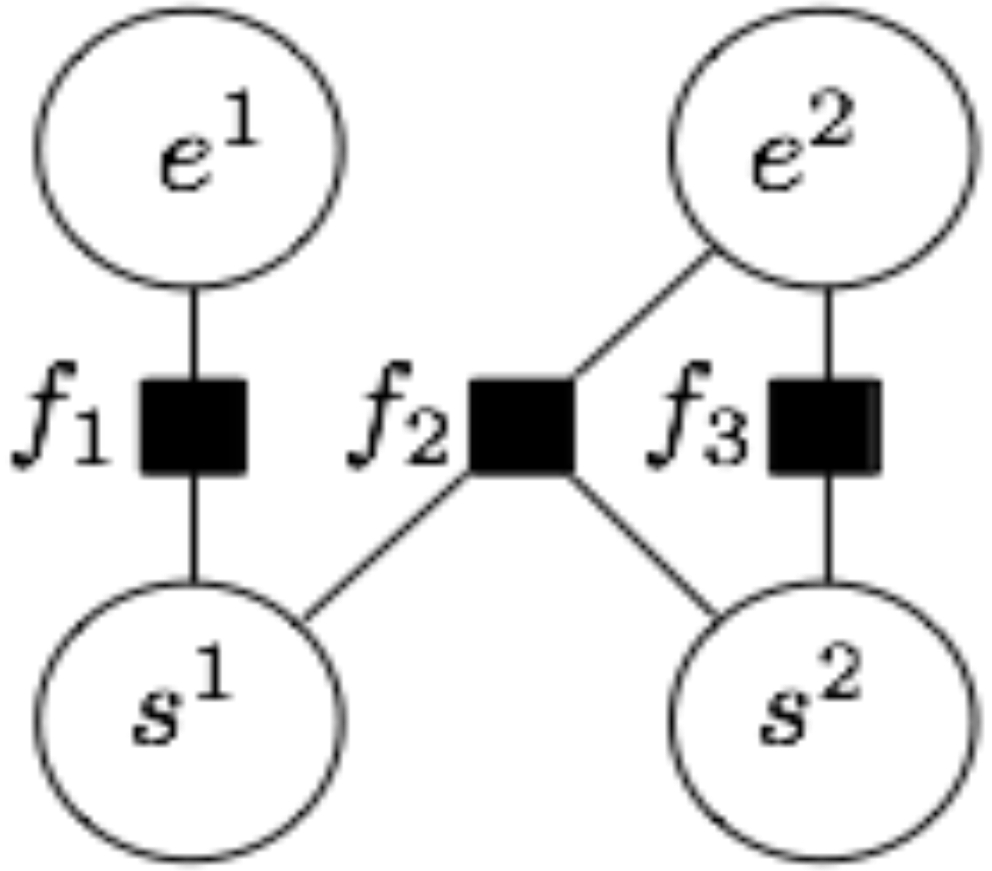
```
sshd: Accepted <user>
$ uname -a; w
$ wget bad-domain.com/vm.c
$ gcc vm.c -o a; ./a
sshd: Received SIGHUP; restarting.
```

Factor Graph Representation and Inference of an Example Incident

Variable nodes are defined using security logs

e^1: download sensitive
e^2: restart system service

s^1: user state when observing e^1
s^2: user state when observing e^2



An example Factor Graph

Factor functions are defined manually

Objectively based on the data from past incidents
Subjectively from security knowledge of the system

Example factor functions

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

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

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

State inference

Enumerate possible s^1, s^2 state sequences

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

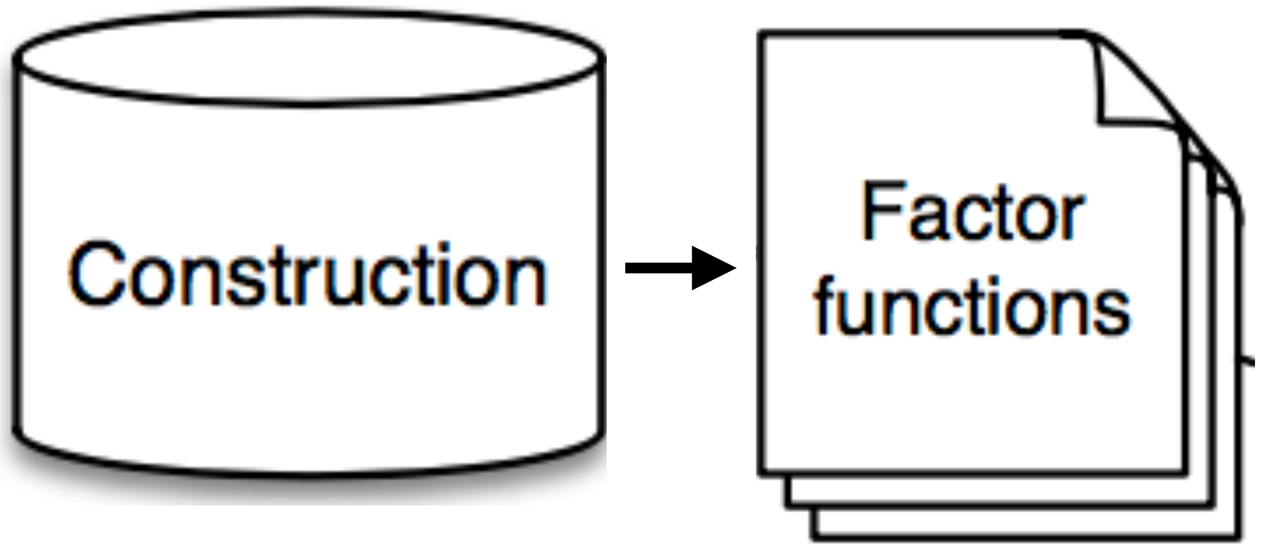
Score(s^1, s^2) is the sum of factor functions f_i

$$argmax_s P(s^1, s^2 | e^1, e^2) = \sum_{s \in S, f \in F} w_f f(e_f, s_f)$$

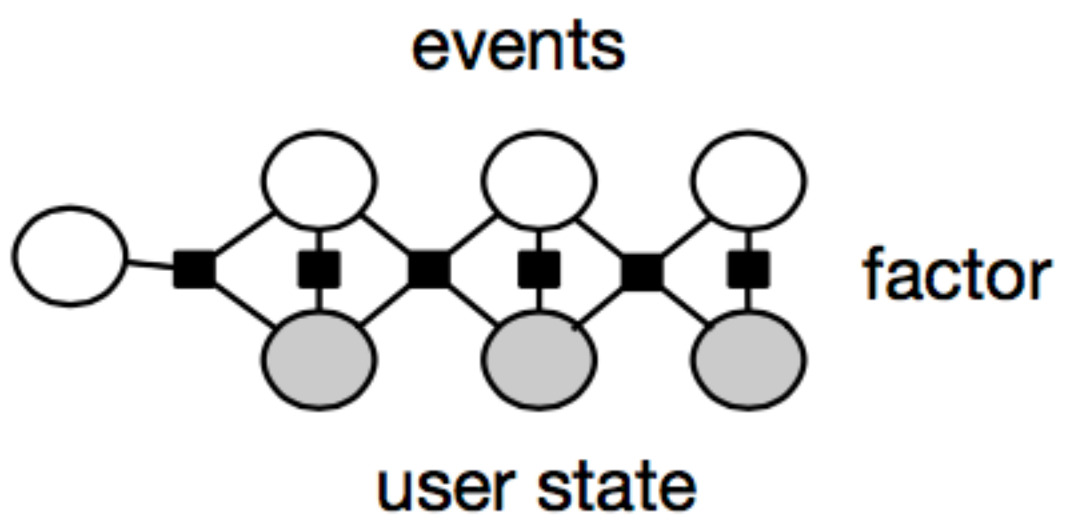
Most probable s^1, s^2 is
suspicious, malicious

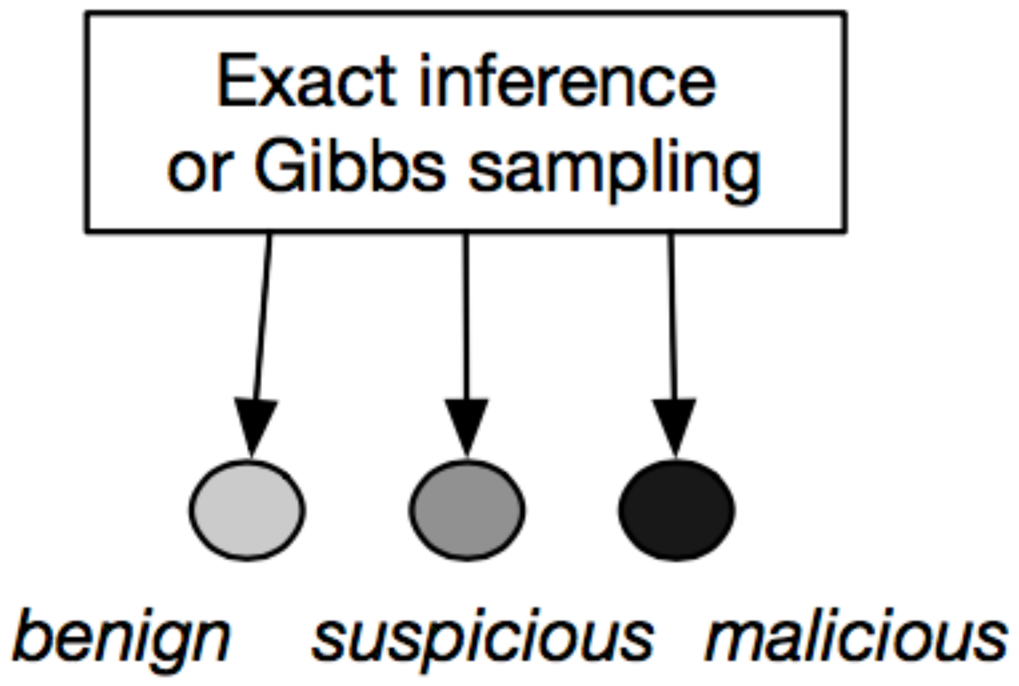
Experimental Workflow of AttackTagger on Real-World Incidents

Construct factor functions from past incidents → Extract events from an incident → Construct per-user factor graph → Infer the user states



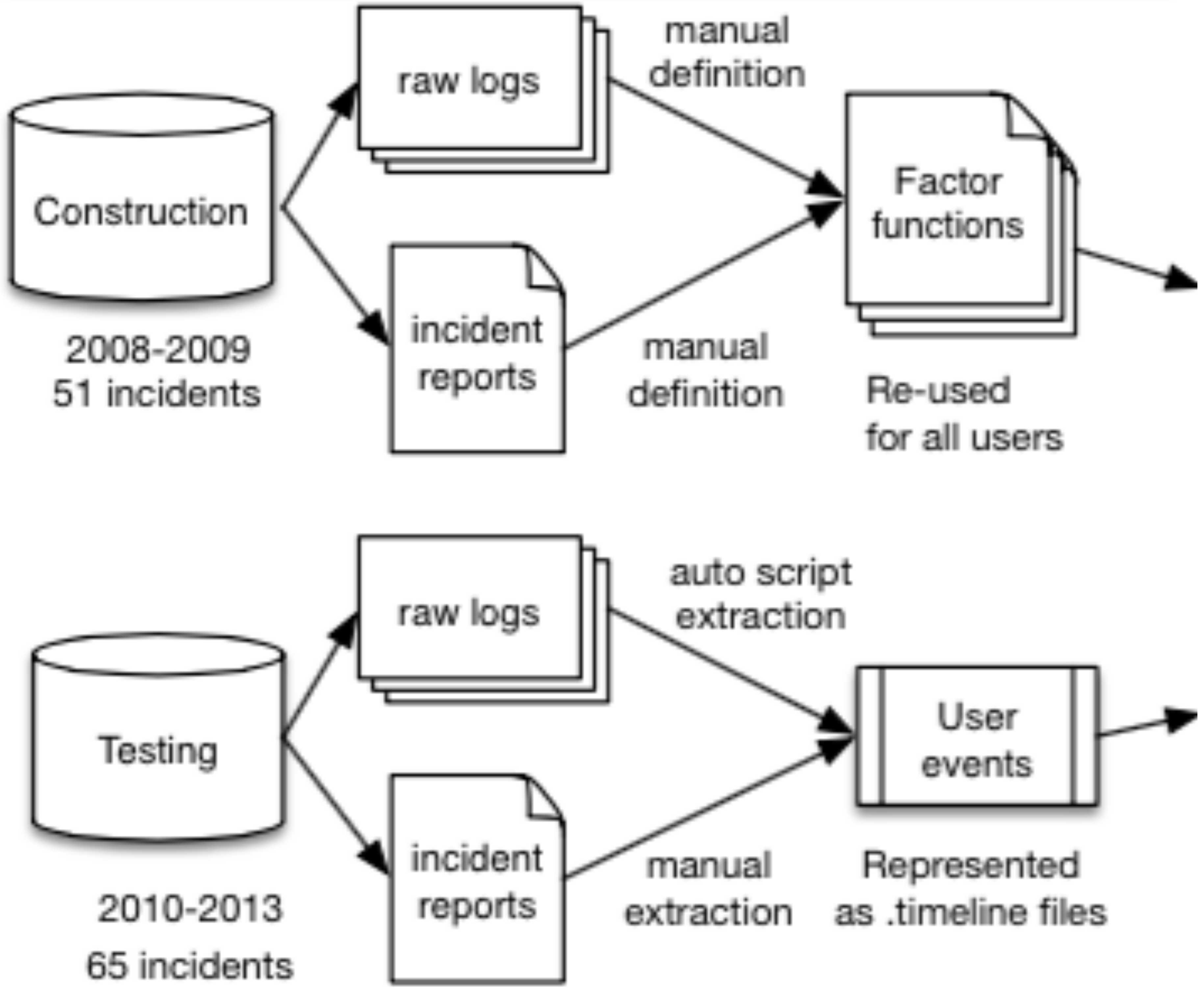
User events





Experimental Workflow of AttackTagger on Real-World Incidents

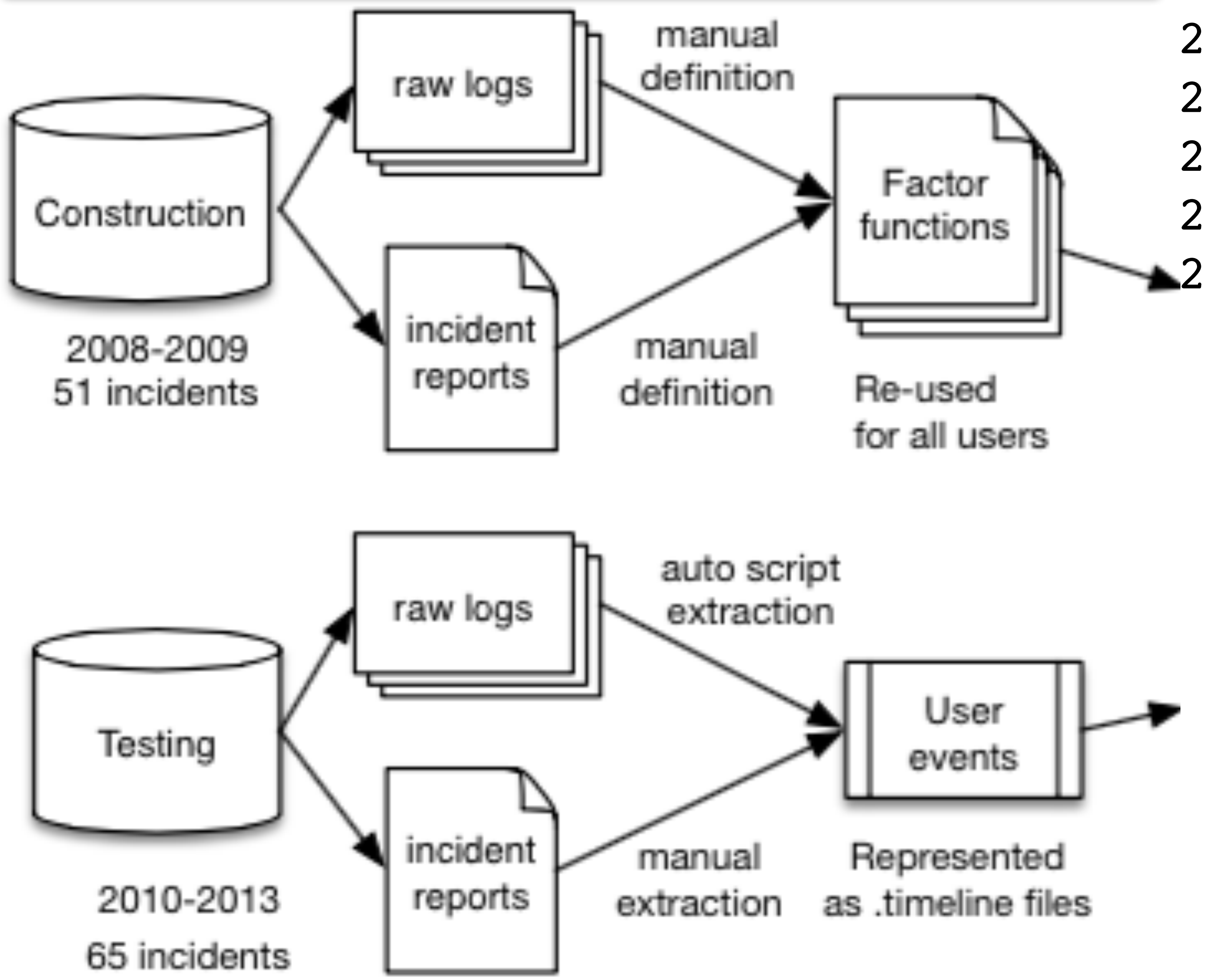
1. Construct factor functions from 51 incidents (2008-2010)



2. Extract events from 65 test incidents (2010-2013)

Experimental Workflow of AttackTagger on Real-World Incidents

1. Construct factor functions from 51 incidents (2008-2010)



2. Extract events from 65 test incidents (2010-2013)

Raw logs

```
11:00:57 sshd: Failed password for root
23:08:26 sshd: Failed password for root
23:08:30 sshd: Failed password for nobody
23:08:38 sshd: Failed password for <user>
23:08:42 sshd: Failed password for root
23:08:57 sshd: Failed password for root
23:09:22 sshd: Failed password for root
```

Absolute Timestamp

Absolute time between the events

Automated

Human-written

The security team received ssh suspicious alerts from <machine> for the user <user>. There were also some Bro alerts from the machine <machine>. From the Bro sshd logs the user ran the following commands

uname -a ..

unset HISTFILE
wget <xx.yy.zz.tt>/abs.c -O a.c;gcc a.c -o a;

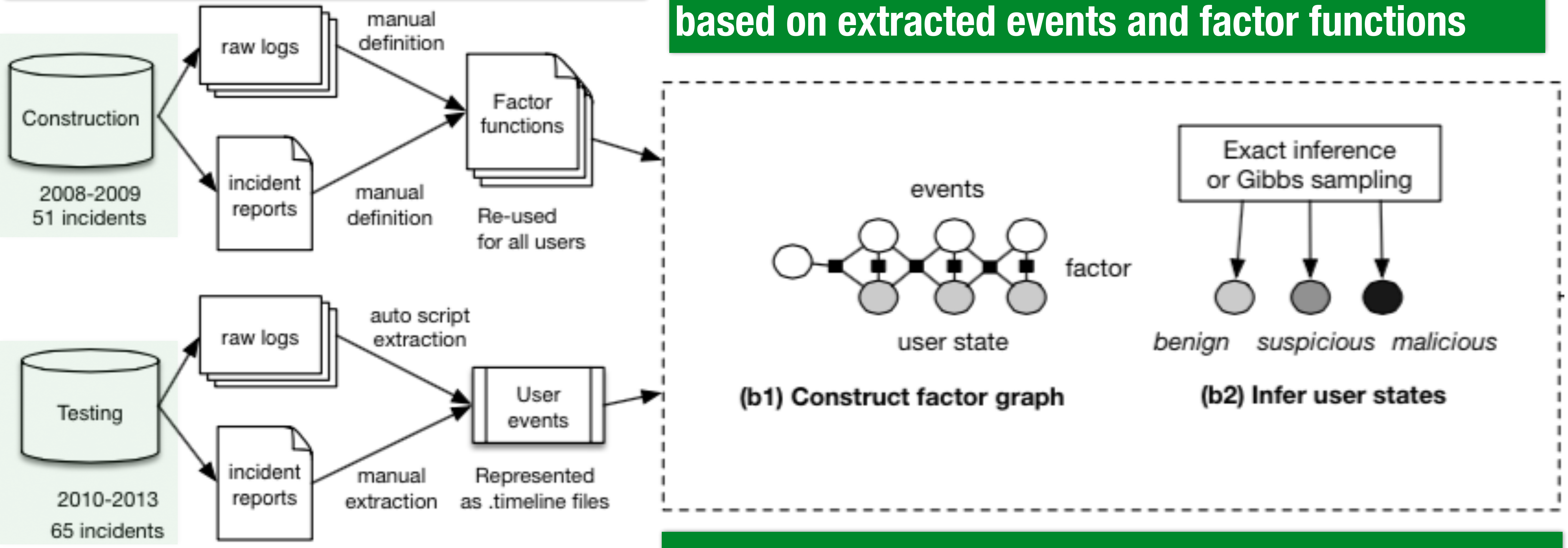
Lamport Timestamp

Relative order of events in an incident

Manual

Experimental Workflow of AttackTagger on Real-World Incidents

1. Construct factor functions from 51 incidents (2008-2010)



2. Extract events from 65 test incidents (2010-2013)

3. For each user, construct a per-user factor graph based on extracted events and factor functions

4. Perform inference on factor graphs using Gibbs sampling or Belief Propagation

Experimental Workflow of AttackTagger on Real-World Incidents

1. Construct factor functions from 51 incidents (2008-2010)

2. Extract events from 65 test incidents (2010-2013)

3. For each user, construct a per-user factor graph based on extracted events and factor functions

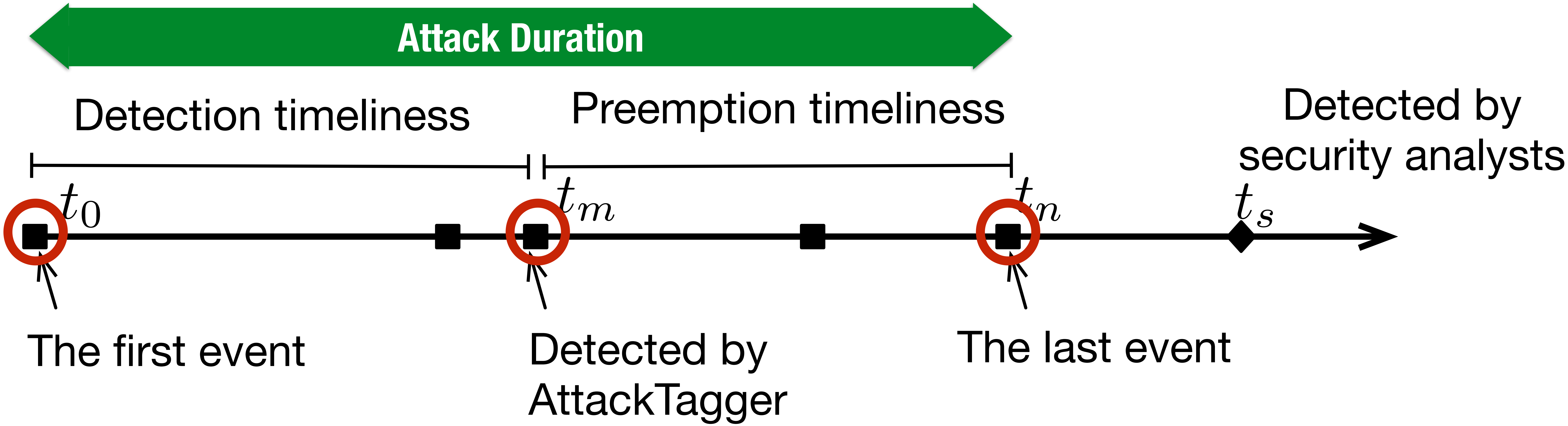
4. Perform inference on factor graphs using Gibbs sampling or Belief Propagation

5. Output user state sequence

6. Take preventive action

Construction
2008-2009
51 incidents
raw logs
incident reports
manual definition
manual definition
Factor functions
Re-used for all users
Testing
2010-2013
65 incidents
raw logs
incident reports
auto script extraction
manual extraction
User events
Represented as .timeline files
events
factor
user state
(b1) Construct factor graph
Exact inference or Gibbs sampling
benign
suspicious
malicious
(b2) Infer user states
Prediction
User u1 is malicious
User u2 is benign

Detection timeliness and Preemption timeliness



Attack Duration

Detection timeliness

Preemption timeliness

Detected by security analysts

t_0

t_m

t_n

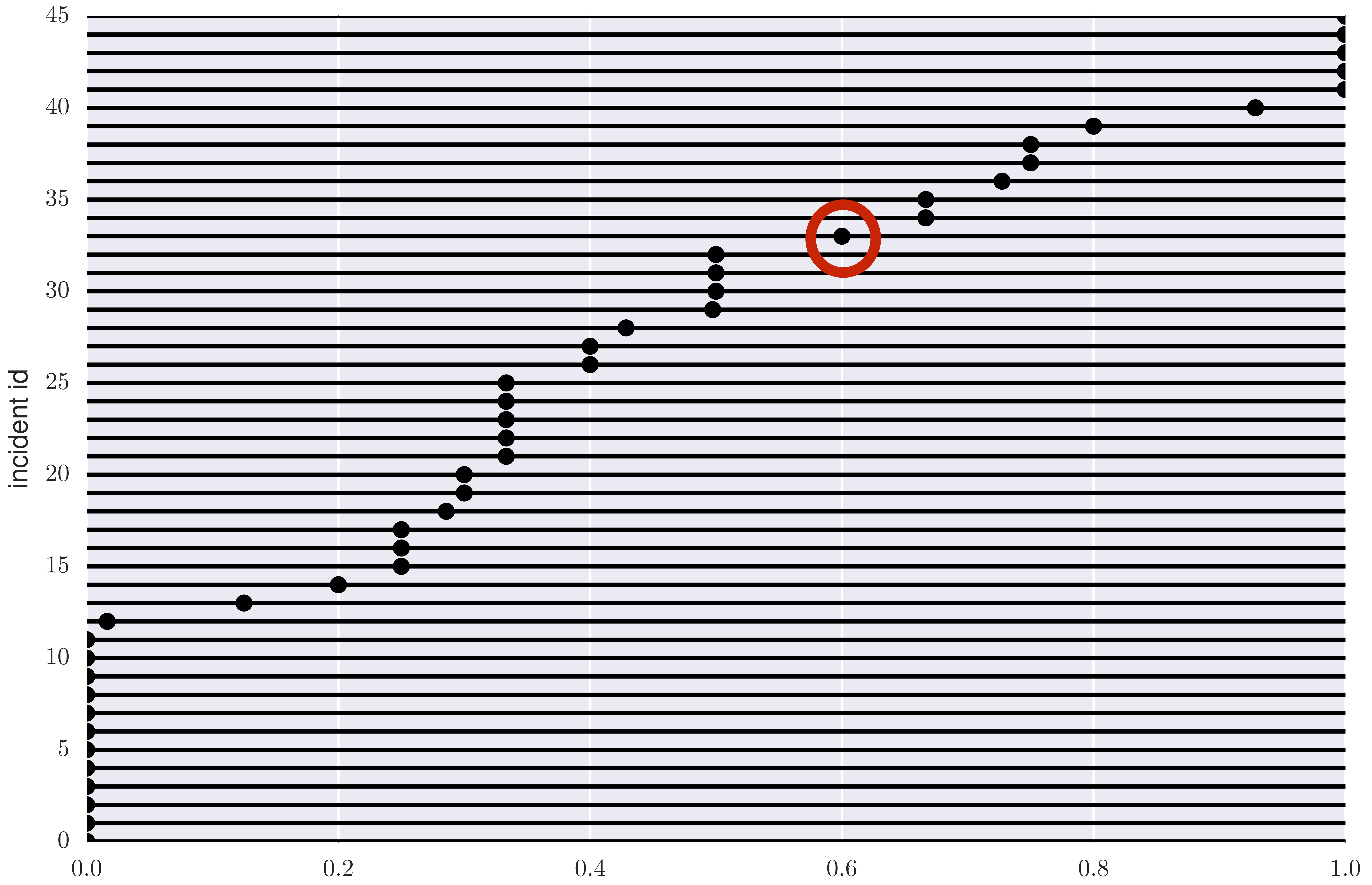
t_s

The first event

Detected by AttackTagger

The last event

Detection timeliness and Preemption Timeliness

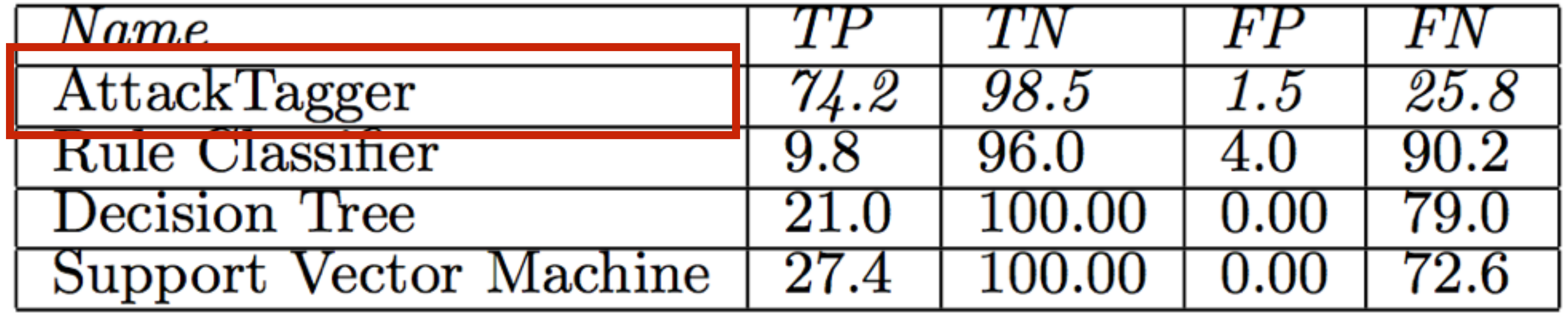


first event Percentage of events observed until attack detection **last event**

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

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

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

Detection performance of the techniques

	AT+	AT-
SVM+	17	0
SVM-	48	1250

McNemar discrepancy matrix

$a=AT^{+}SVM^{+}$, $b=AT^{-}SVM^{+}$, $c=AT^{+}SVM^{-}$, $d=AT^{-}SVM^{-}$

$$\chi^2 = (b+c)^2/(b-c)$$

$$\chi^2 = 48$$

p-value < 0.00001

Our approach has:

- Best detection rate (46 of 62 malicious users)
- Smallest false detection rate (19 users of 1267 benign users).

Show that performance of AttackTagger (AT) is better than Support Vector Machine (SVM) not by chance

- Null hypothesis H_0 : both techniques have the same detection performance.

Measure discrepancy between: AT and SVM

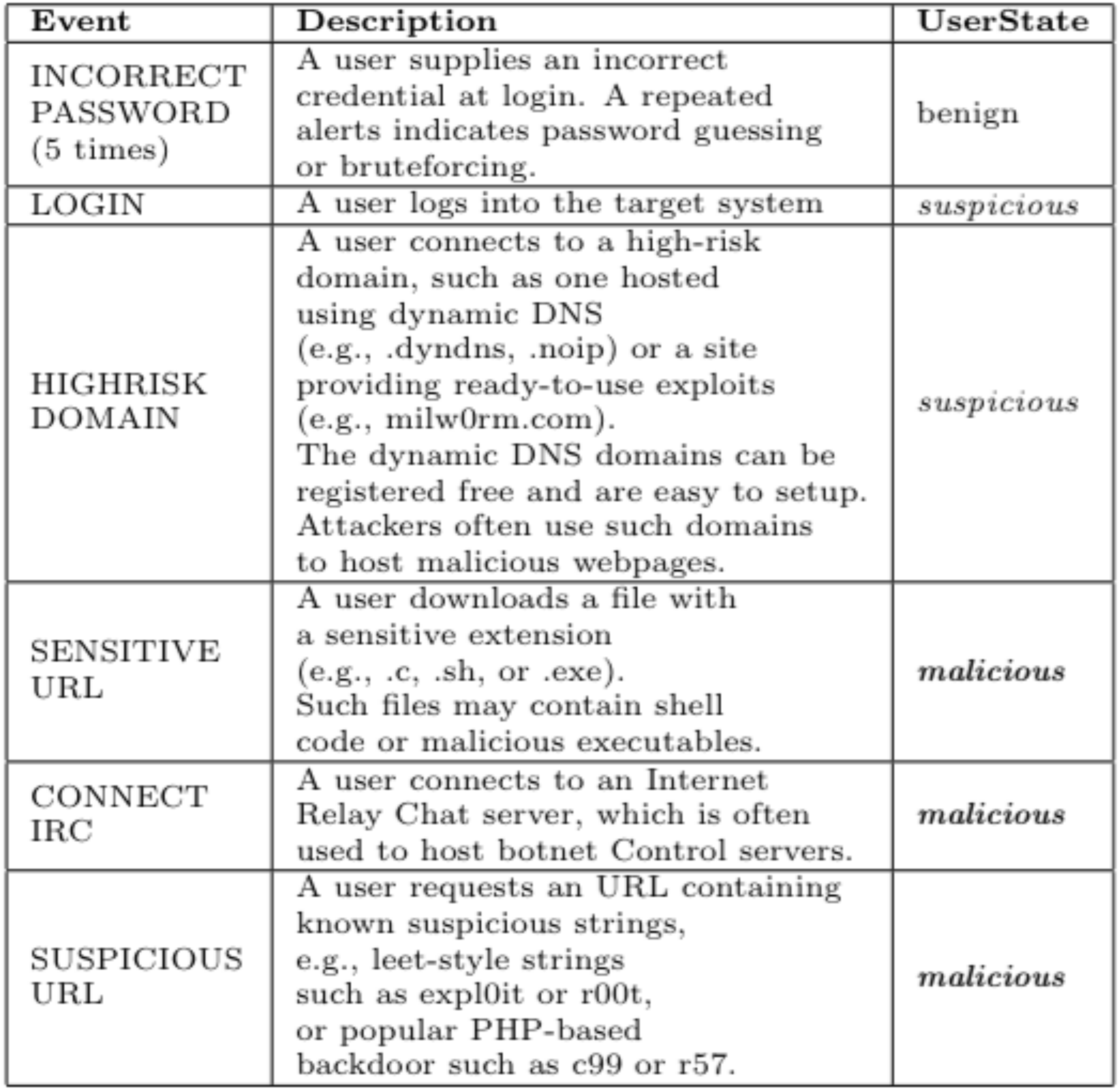
AT detection performance was significantly different than SVM

Detection of unidentified malicious users

Incident ID	Activity
20100416	Illegal activities
20100513	Incorrect credentials (multiple times); Sending spam emails
20101029	Logging in from multiple IP addresses; Illegal activities
20101029	Logging in after a long inactive time; Illegal activities
20101029	Illegal activities

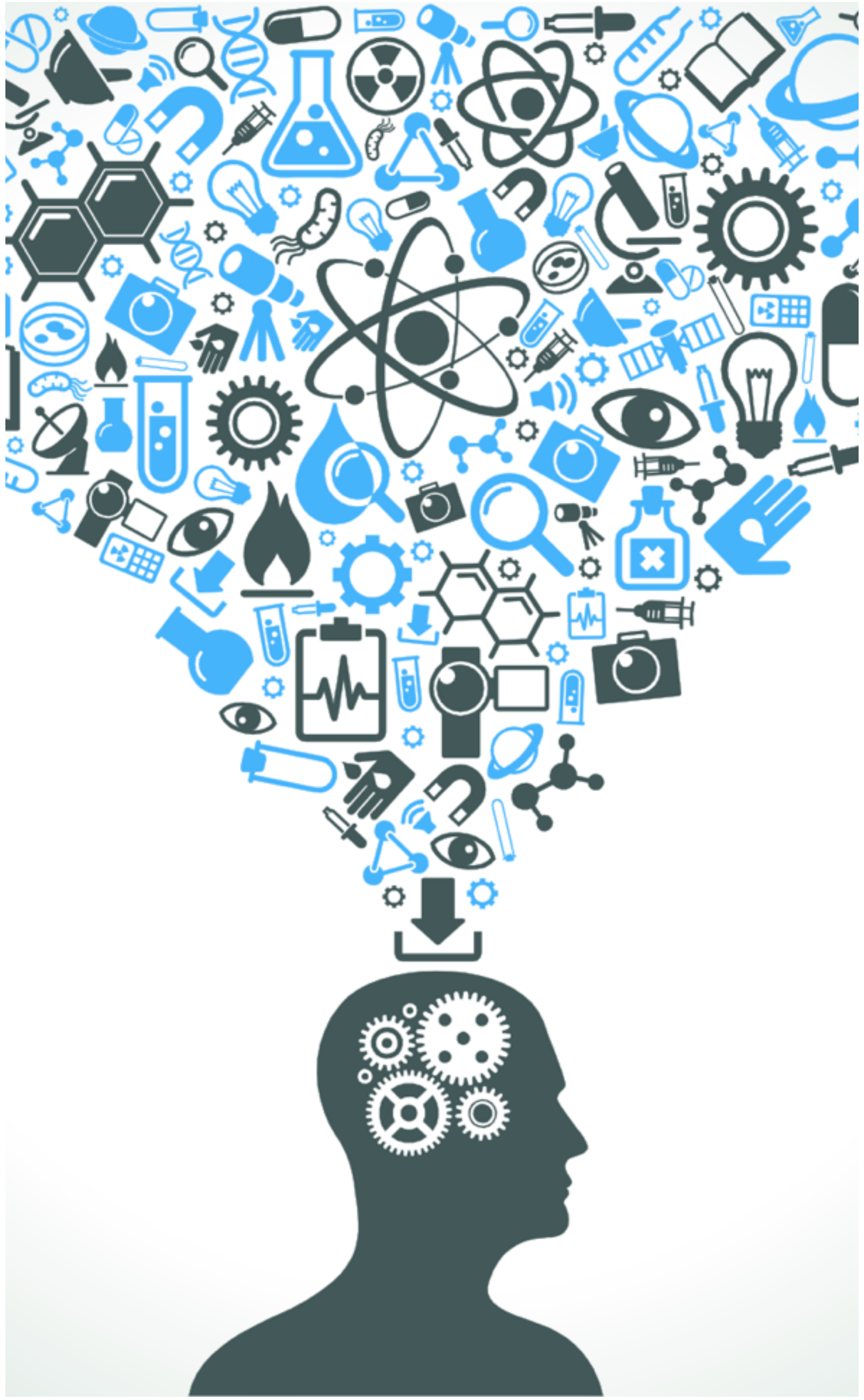
Identified six hidden malicious users who were not identified in the incident reports.

Detection of unidentified malicious users (cont.)



Event	Description	UserState
INCORRECT PASSWORD (5 times)	A user supplies an incorrect credential at login. A repeated alerts indicates password guessing or bruteforcing.	benign
LOGIN	A user logs into the target system	*suspicious*
HIGHRISK DOMAIN	A user connects to a high-risk domain, such as one hosted using dynamic DNS (e.g., .dyndns, .noip) or a site providing ready-to-use exploits (e.g., milw0rm.com). The dynamic DNS domains can be registered free and are easy to setup. Attackers often use such domains to host malicious webpages.	*suspicious*
SENSITIVE URL	A user downloads a file with a sensitive extension (e.g., .c, .sh, or .exe). Such files may contain shell code or malicious executables.	*malicious*
CONNECT IRC	A user connects to an Internet Relay Chat server, which is often used to host botnet Control servers.	*malicious*
SUSPICIOUS URL	A user requests an URL containing known suspicious strings, e.g., leet-style strings such as expl0it or r00t, or popular PHP-based backdoor such as c99 or r57.	*malicious*

Brute-force guess passwords	benign
Login	suspicious
Connect to a high-risk domain to get exploit code	suspicious
Download source code of a root exploit (.c) file	malicious
Connect to a Command & Control server via IRC	malicious
Download PHP backdoor to establish tunnel to the compromised machine	malicious



Conclusion

1. Factor graphs are a suitable representation of user/system state transitions in security incidents.

2. Experimental evaluation of factor graphs show that a majority compromised users (74%) can be detected in advance (minutes to hours before the system misuse)

3. Our approach can detect a variety of attacks, including hidden attacks that went unidentified by in incident reports.

Acknowledgement

DEPEND group after the Fall 2014 retreat!

DEPEND group members

NCSA security team

Dr. Shuo Chen, MSR

Dr. Charles Kamhoua, AFRL

Ms. Jenny Applequist

Questions