Analyzing Intensive Intrusion Alerts via Correlation[†]

Peng Ning Yun Cui Douglas S. Reeves

Departments of Computer Science and
Electrical and Computer Engineering

North Carolina State University

RAID 2002

[†] Supported by the National Science Foundation and the Army Research Office

Outline

1. Motivation
2. Previous work on intrusion alert correlation
3. Three utilities for interactive analysis
4. Experimental validation
5. Conclusions and future work

Motivation (cont'd)

- CERT's overview of attack trends (04-18-02)
  - Increasing automation
  - Increasing sophistication of attack tools
- "Traditional" intrusion detection systems (IDS)
  - Focus on low-level attacks or anomalies
  - Mix actual alerts with false alerts
  - Generate an unmanageable number of alerts
    - ID practitioners: "Encountering 10,000 to 20,000 alerts per day per sensor is common"
- Conclusion: we need automated tools to…

A. construct attack scenarios
B. facilitate intrusion analysis

Related Work on Alert Correlation

Method 1: Exploit similarities between alert attributes

- Ex.: Valdes and Skinner (2001), Staniford et al. (2000)
- Cannot fully discover the causal relationships between alerts

Method 2: Exploit known attack scenarios

- Ex.: Cuppens and Ortalo (2000), Dain and Cunningham (2001), Debar and Wespi (2001)
- Restricted to known attack scenarios

Method 3: Use pre- and post-conditions of attacks

- Templeton and Levitt (2000), Cuppens and Miege (2002), Ning et al (2002)

A Model for Alert Correlation

- Represent our knowledge about individual types of attacks
  - *Prerequisite*: necessary condition or system state for an intrusion to be successful
  - *Consequence*: possible outcome or system state of an intrusion
- Correlate alerts (i.e., detected attacks) by reasoning about the consequences of earlier attacks and the prerequisites of later ones
- Ex.: **if** attack A learns a vulnerable service exists, **and** attack B exploits the same vulnerable service, **then** correlate A and B

A Model (cont'd)

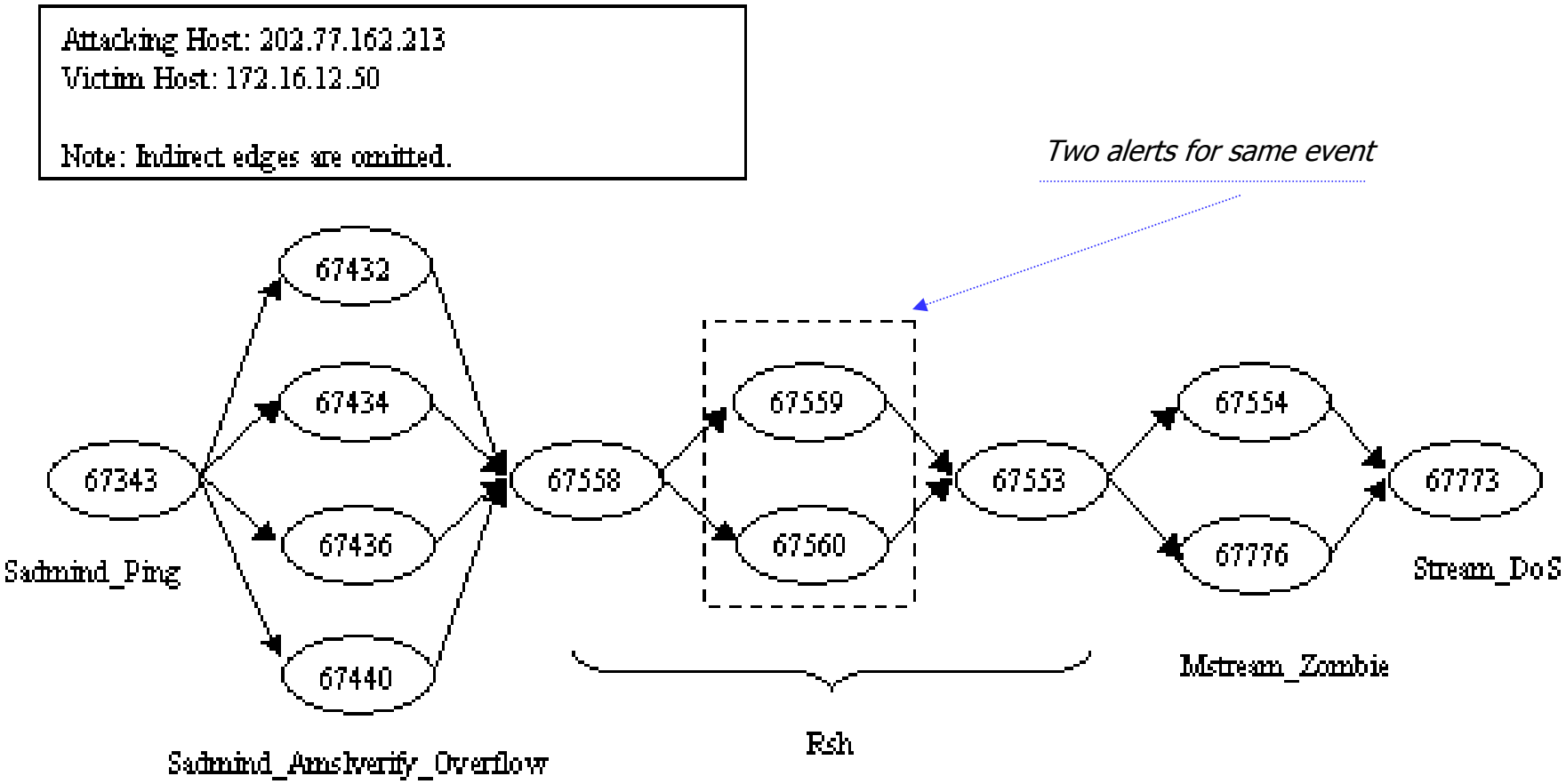
- *Hyper-alert type*: definition of a potential attack, including its prerequisites and consequences
- *Hyper-alert*: a set of occurrences of a hyper-alert type, and the times at which they occurred

A Model (cont'd)

- Given hyper-alerts h_1 and h_2, h_1 *prepares for* h_2 if…
  - h_1 occurred before h_2 and
  - the prerequisite of h_2 implies the consequence of h_1
- For a *hyper-alert correlation graph* CG = (V, E)…
  - V represents a set of hyper-alerts
  - for all h_1, $h_2 \in V$, there is a directed edge from h_1 to h_2 if and only if h_1 prepares for h_2
  - *Note:* We usually omit transitive edges for the sake of readability

A Hyper-Alert Correlation Graph

- Example encountered in an early experiment…



Contributions of This Work

- We propose three utilities to facilitate interactive intrusion analysis
  1. Adjustable graph reduction
  2. Focused analysis
  3. Graph decomposition
- A case study of the effectiveness of the three utilities
  - Test dataset = DEFCON8 "Capture The Flag"

#1. *Adjustable Graph Reduction*

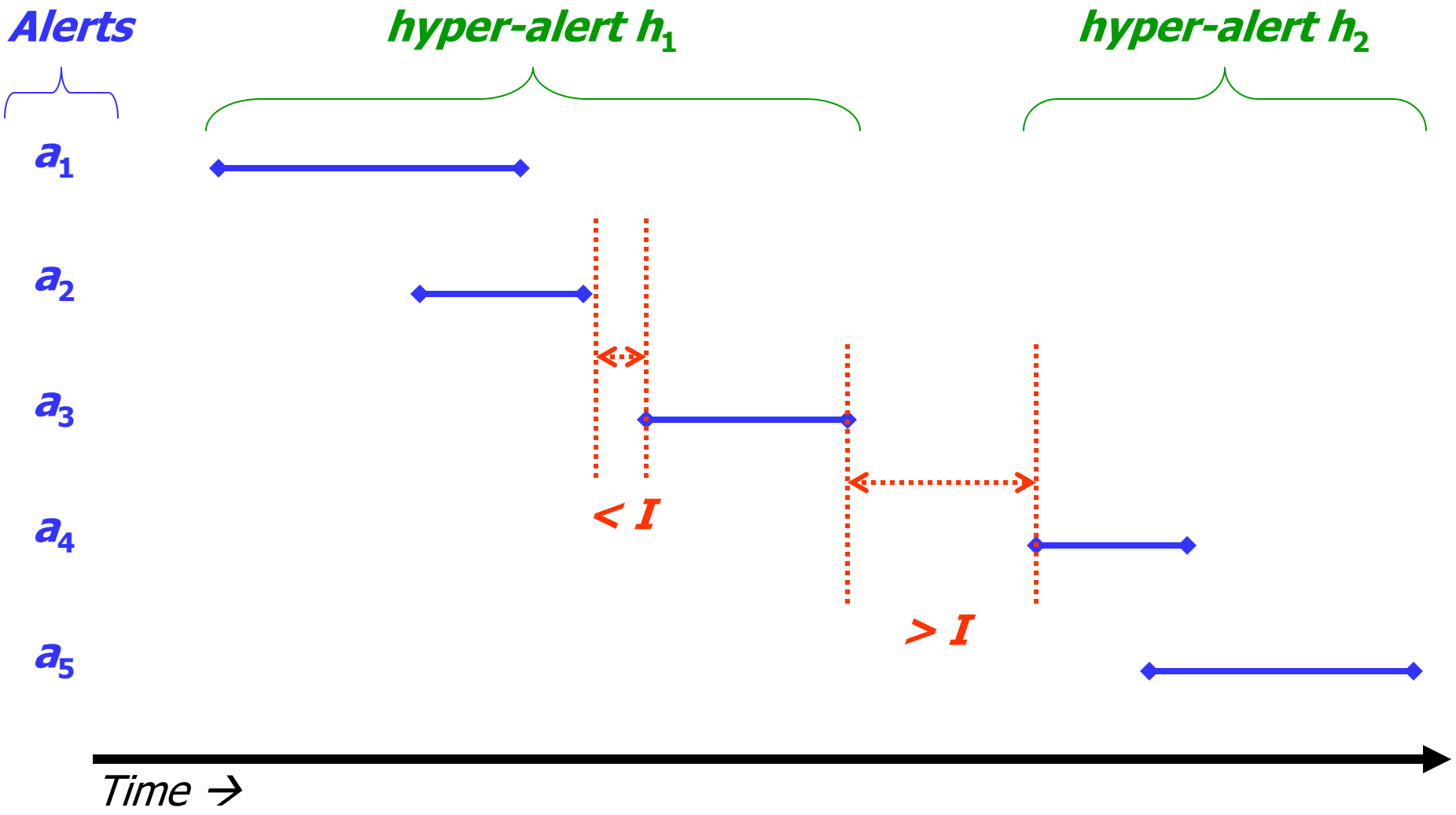
- Basic idea: Aggregate (combine) hyper-alerts of the same type only when they occur close to each other in time

- *Interval constraint*: given a time interval *I* (e.g., 3 seconds), a hyper-alert *h* satisfies interval constraint *I* if

  1) *h* has only one alert, or

  2) for every alert a_i in *h*, there is at least one other alert a_j in *h* which overlaps a_i in time, or which is separated from a_i by no more than *I* units of time

Adjustable Graph Reduction (cont'd)



#2. *Focused Analysis*

- Basic idea: Focus on the hyper-alerts of interest according to user's specification
- *Focusing constraints*
  - A *focusing constraint* is a logical combination of comparisons between attribute names and constants.
  - Example:
    srcIP = 129.174.142.2 $\vee$ *destIP* = 129.174.142.2
  - Only correlate hyper-alerts that evaluate to true w.r.t. the focusing constraint, i.e., filter out irrelevant hyper-alerts

#3. *Graph Decomposition*

- Basic idea: cluster the hyper-alerts based on common features shared by them, and decompose a large graph into smaller, more meaningful graphs (clusters)
- *Clustering constraints*
  - Given sets of attribute names A_1 and A_2 for two hyper-alerts h_1 and h_2, a *clustering constraint* $C_c(A_1, A_2)$ is a logical combination of comparisons between constants and attribute names in A_1 and A_2

Graph Decomposition (cont'd)

- Example
  - $A_1 = A_2 = \{srcIP, destIP\}$
  - $C_c(A_1, A_2) =$
    $(A_1.srcIP = A_2.srcIP) \wedge (A_1.destIP = A_2.destIP)$

  → i.e., two hyper-alerts are clustered if they have the same source and destination IP addresses

Case Study of DEFCON8 Dataset

- Before applying utilities

Category	Count
Total hyper-alert types	115
Correlated hyper-alert types	95 (83%)
Uncorrelated hyper-alert types	20
Total hyper-alert instances	65,054
Correlated hyper-alert instances	9,744 (15%)
Uncorrelated hyper-alert instances	55,310

DEFCON8 Before Applying Utilities

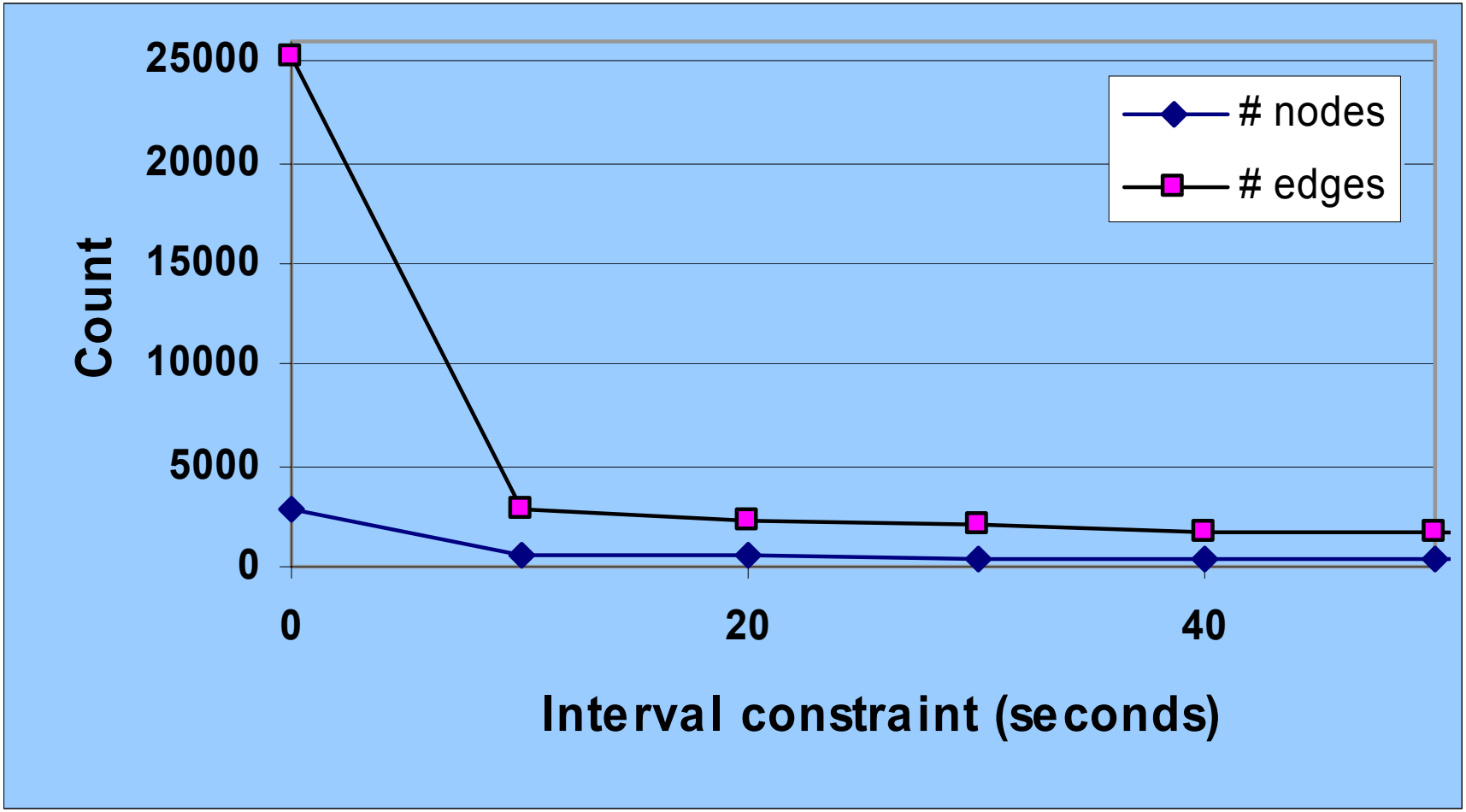
- Some hyper-alert types have many uncorrelated instances

Type	# Correlated	# Uncorrelated
IPHalfScan	958	33,745
Windows_Access_Error	0	11,657
HTTP_Cookie	0	2,119

- 450 hyper-alert correlation graphs
- Largest graph: 2,940 nodes and 25,321 edges
- Average graph: 22 nodes and 311 edges

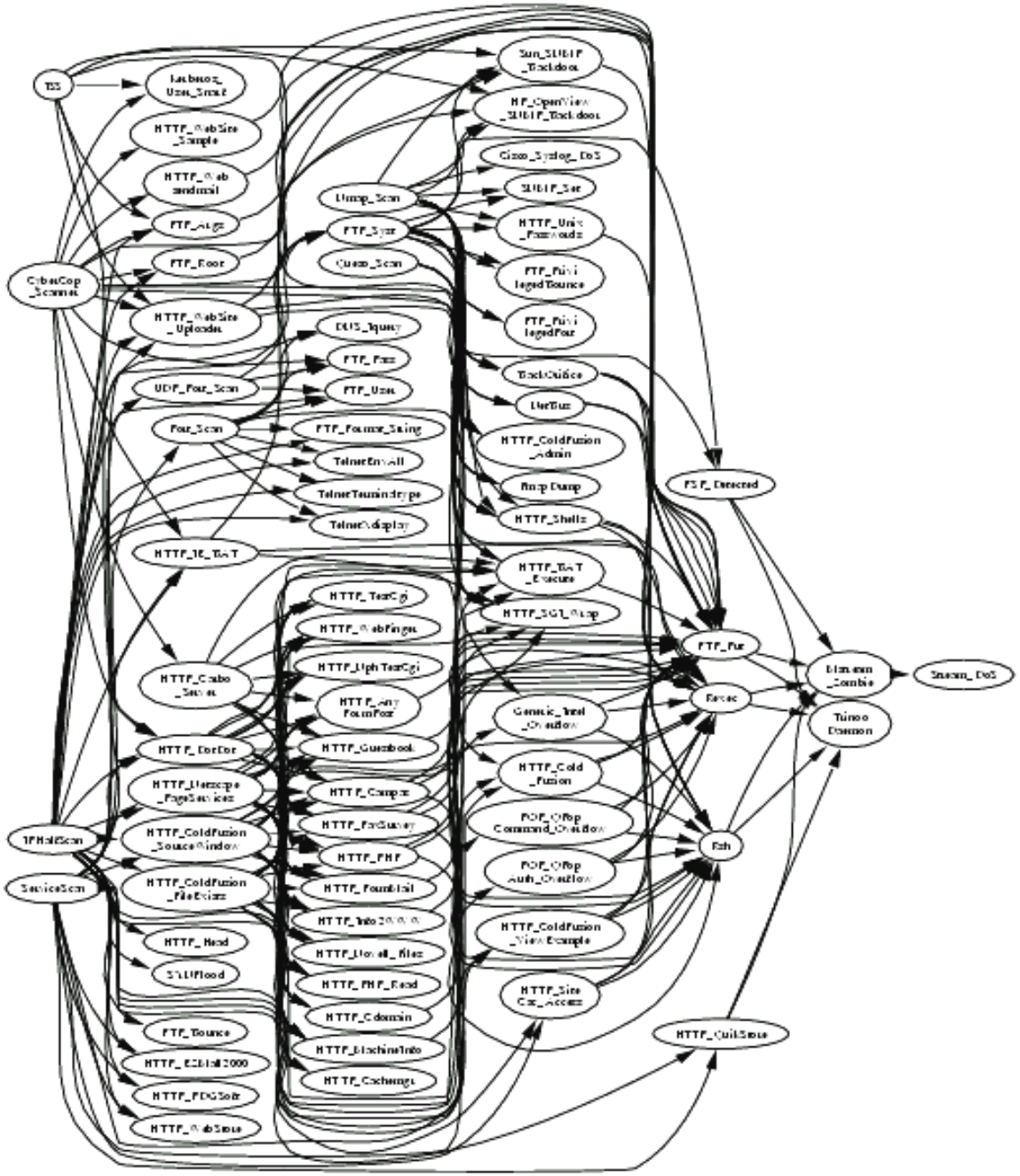
Using Adjustable Graph Reduction

- Most hyper-alerts of the same type are close to each other in time in the DEFCON8 dataset



Largest Correlation Graph after Maximum Graph Reduction

- Nodes: 2,940→77 (97% reduction)
- Edges: 25,321→347 (99% reduction)



Using Graph Decomposition

- $A_1 = A_2 = \{srcIP, destIP\}$

best choice

- $C_{c1}(A_1, A_2) :=$
  $A_1.destIP = A_2.destIP$

  →Generated 12 clusters

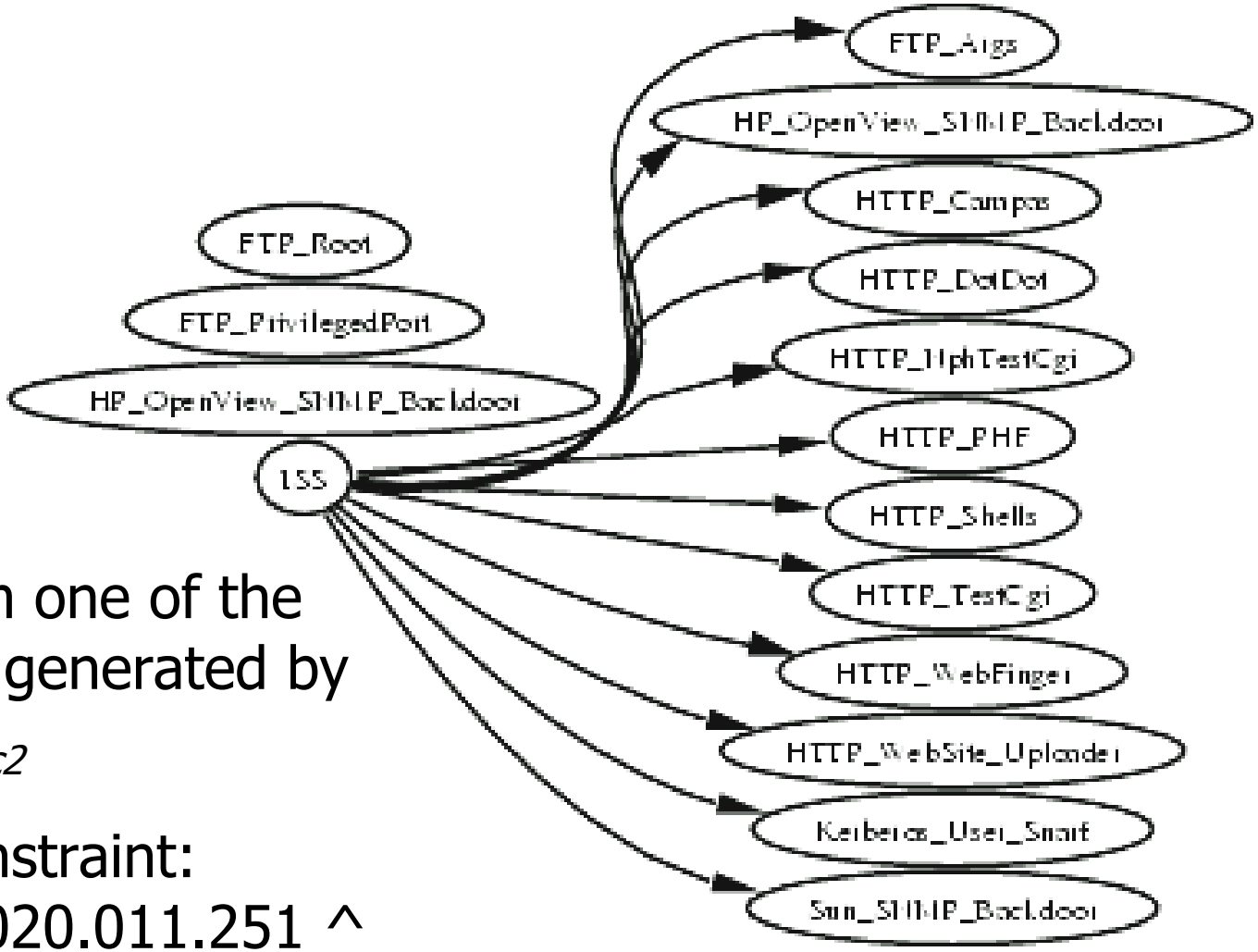
- $C_{c2}(A_1, A_2) :=$
  $(A_1.srcIP = A_2.srcIP) \wedge (A_1.destIP = A_2.destIP)$

  →Generated 185 clusters

- $C_{c3}(A_1, A_2) :=$
  $(A_1.srcIP = A_2.srcIP) \vee (A_1.destIP = A_2.destIP) \vee$
  $(A_1.srcIP = A_2.destIP) \vee (A_1.destIP = A_2.srcIP)$

  →Generated 2 clusters

Using Focused Analysis



- Starting from one of the 185 clusters generated by constraint C_{c2}
- Focusing constraint: *srcIP* =010.020.011.251 ^ *destIP* = 010.020.001.010

Additional Results

- Some common attack strategies were easily identified
  - e.g., Nmap_Scan → PmapDump → ToolTalk_Overflow
  - e.g., HTTP-based attack from 010.020.011.074 to 010.020.001.014, 010.020.001.015, 010.020.001.019…
- Another observation
  - There were many BackOrifice and NetBus alerts
  - i.e., attackers were coordinating multiple machines during their attacks
  - Makes correlation and attack identification more difficult!

Conclusions

- Developed three utilities for interactive intrusion alert analysis
  1. Adjustable graph reduction
  2. Focused analysis
  3. Graph decomposition
- Studied the effectiveness of these utilities through a case study
  - Utilities simplified the analysis of huge number of alerts
- Caution! The results produced by these utilities are distillations of what really happened

Future Work

- Automate analysis process
  - Currently, skill is required to choose the right interval, focusing, and clustering constraints
- Integrate with other complementary correlation methods
- Identify attacks missed by most IDSs
- Use for attack prediction

Software Available

http://discovery.csc.ncsu.edu/software/correlator/ver0.2/iac.html

Some Previous Results

- We implemented a DBMS based intrusion alert correlator
  - http://discovery.csc.ncsu.edu/software.html.
- Validated it with the 2000 DARPA intrusion detection scenario datasets
  - Completeness: At least 93.18% related alerts are correlated in LLDOS 1.0, at least 62.5% related alerts are correlated in LLDOS 2.0.2
  - Soundness: At least 92.3% correlated alerts are indeed related