Real-Time Alert Correlation with Type Graphs

Gianni Tedesco
<gxt@cs.nott.ac.uk>

University of Nottingham
Netfort Technologies Ltd.
Research sponsored by the Smith Institute and EPSRC

http://www.cs.nott.ac.uk/~gxt/

Outline

- Introduction
- Background
- Related Work
- Real-Time Attack Graph Correlation
  - Problem Definition
  - Algorithm
  - Index data structure
- Results
- Conclusions and Future Work

Introduction

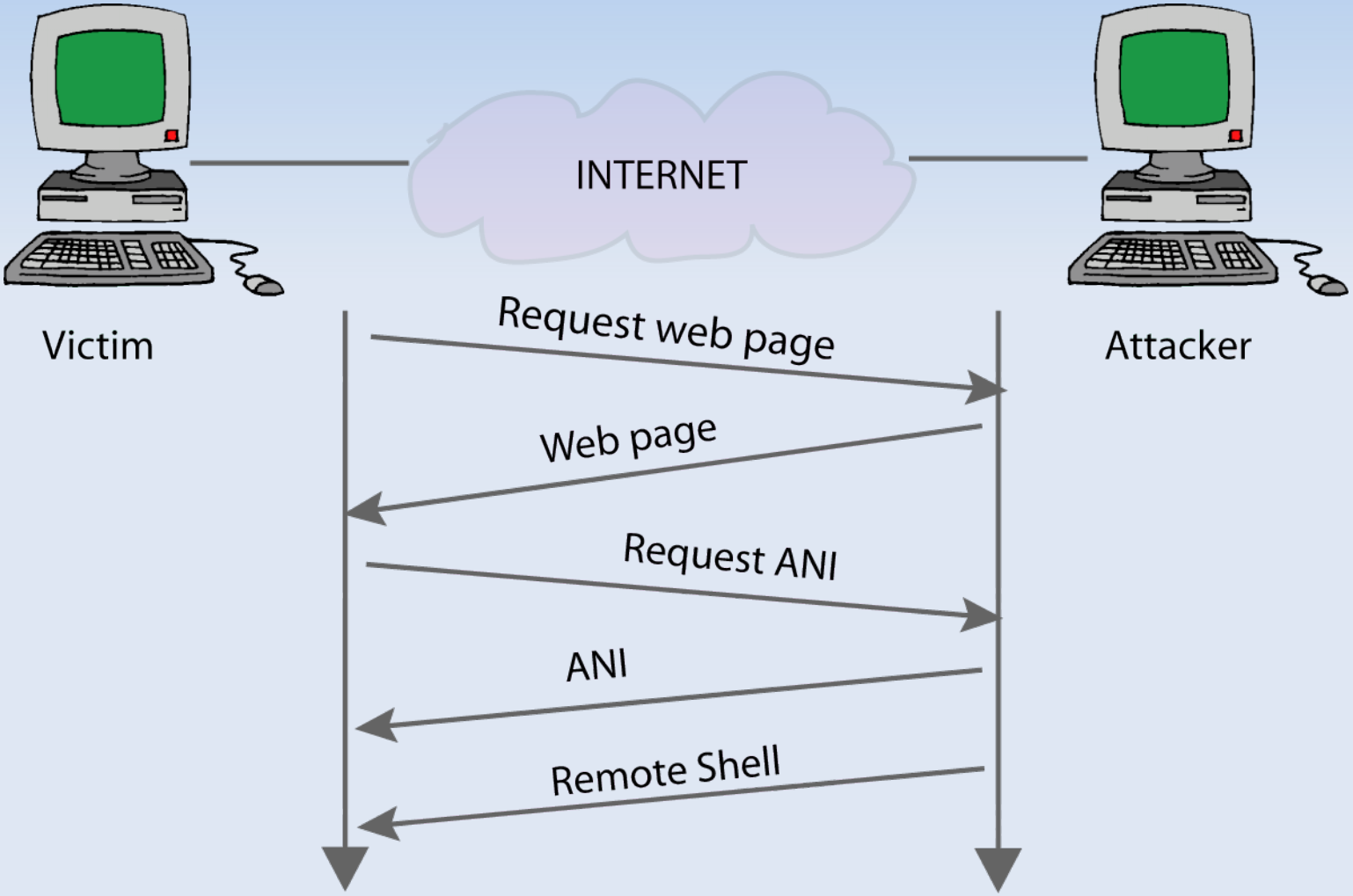
- Our focus is network intrusion detection
- Creating models of attacker behaviour
- Correlating of alerts from NIDS
- Our contribution is
  - A real-time correlation algorithm for interpreting the output of NIDS
- Theoretical and empirical results show the algorithm to be:
  - Effective
  - Performant

What is an Attack?

- Attack: Action or sequence of action conducted by an attacker to achieve goal such as:
  - Denial of service
  - Raising of privileges
  - Unauthorised use of resources
- Vulnerability: Software or hardware fault rendering system amenable to attack
- Exploit: Method for exploiting a vulnerability in order to achieve an attack goal
- Alert: Description of an attack instance

Example Attack

Attacks often proceed in stages.



Earlier stages prepare for later stages.

Network Intrusion Detection

- Network IDS monitors network traffic
- Detects attacks and/or anomalous behaviour
- Produces a time series of ”alerts”
  - Indicate attack or an anomaly
  - Tuple of attribute values usually including source and destination addresses, type of attack, etc.
- Usually there are many false alerts
- Difficult to analyze by hand
- Automatic analysis implies an attack model

Attack Modeling

- Swiler and Phillips introduced attack models in 1998
- Intedended for security problems:
  - Quantifying the security of a network
  - Planning based on cost/benefit analysis, cost of security countermeasures vs. security attained (a hard integer optimisation problem)
- Models intended for symbolic model checkers and production expert systems

Prerequisites and Consequences

- Prereqs and conseqs of attacks are predicate instances
- Output is invariant under various evasion attacks:
  - Mutation
  - Re-sequencing
  - Substitution
  - Distribution
  - Looping
- Generally leads to smaller and simpler models

Alert Correlation

- Attack model used to interpret IDS output:
  - Aggregating alerts implying the same or similar consequences
  - Ignoring extraneous alerts which do not correlate
  - Uncovering missing alerts and inferring their attributes
- Several techniques (off-line)
- Two prior real-time techniques
  - Scenario graphs
  - Type graphs + sliding window

Related Work

- Ning et al. propose scenario-graph correlation.
  - Requires map of vulnerabilities on protected network
  - Very efficient
- Wang et al. propose type-graph correlation.
  - Abstract attack types
  - Too slow for real-time correlation, even with sliding window
- Motivation: real-time correlation, with abstract attack types

Attack Model Definition

- Hyper-alert types. A triple (*fact*, *prereq*, *conseq*)
  - *fact* – set of attributes names for alerts of that type
  - *prereq* – predicates with free vars. bound from fact
  - *conseq* – predicates with free vars. bound from fact
- May prepare for relation
  - Given types A and B, A may prepare for B iff *Conseq(A)* and *Prereq(B)* share one or more predicates
- Type graph. $TG = (V, E, C, T)$
  - T is a bijection of types to vertices
  - Edge $e(v, u)$ exists iff v may prepare for u

Equality Constraints

- Conjunctions of equality comparisons between two types *(A, B)* where *A* may prepare for *B*.
- Let $u_1, u_2, \ldots, u_n$ and $v_1, v_2, \ldots, v_n$ be distinct attributes from *A* and *B* respectively. Each constraint takes the form:

$$u_1 = v_1 \ \wedge \ u_2 = v_2 \ \wedge \ \ldots \ \wedge \ u_n = v_n$$

- Such that there exists $p(u_1, u_2, \ldots u_n)$ in *Conseq(A)* and $p(v_1, v_2, \ldots, v_n)$ in *Prereq(B)* where *p* is the same predicate with possibly different fact assignments.

Example Attack Model

- Logical attack model is translated in to attack graph
- Here is a subset of an example attack model

Name	**Prereq**	**Conseq**
ICMP_Sweep		Pinged(DstIP)
Sadmind_Ping	Pinged(SrcIP)	SadmindPinged(DstIP)
Sadmind_Exp	SadmindPinged(SrcIP)	Exploited(DstIP)
BIND_Exp		Exploited(DstIP)
Mstream_Zombie	Exploited(DstIP)	ReadyForDDOS(DstIP)
Mstream_DOS	ReadyForDDOS(SrcIP)	DDOS(SrcIP)

Example Attack-Type Graph

The Prepare-For Relation

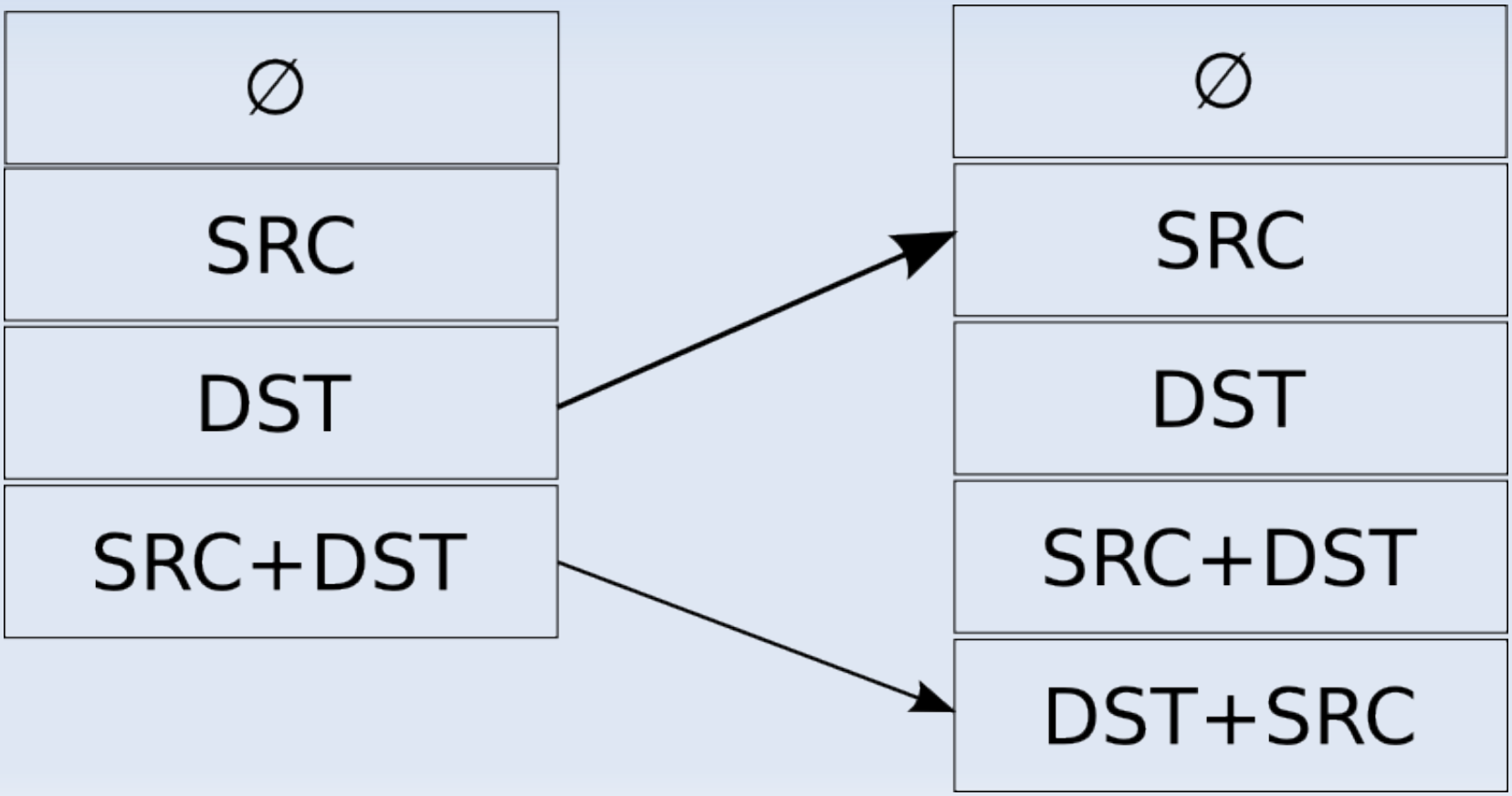
- Hyper-alert. *h* is a tuple of attribute values corresponding to facts in *Type(H).*
- Prepare for relation. A type *A* hyper alert *h* prepares for a Type *B* hyper alert *h'* iff*:*
  - *A* may prepare for *B*
  - At least one equality constraint between *A* and *B* evaluates to true when fact names have been substituted with values
- Output graph: *(V, E, H)*
  - *H* is a bijection between vertices and hyper-alerts
  - Edge *e(u, v)* exists iff *u* prepares for *v*

The Correlation Algorithm

- Algorithm takes one IDS alert at a time and proceeds in two stages:
  1. Implicit correlation check
  2. Marking of consequences
  3. Searching for correlations
- Approach exploits two properties of problem:
  - Monotonicity
  - Number of equality constraints has a fixed upper bound

Structure of Constraints

Constraints may be described as a pair *(c, p)* where *c* is a combination of facts and *p* is a permutation



Index Structure

- Upper bound on number of constraints is exponentially related to number of facts in alerts.
- In practice number of possible constraints is small and may all be indexed.
- A dictionary map is created for each combination of fact attributes referred to in an equality constraint

Example Correlation Graph

Hypothesising Missing Alerts

- Hypothesising missing alerts. What is it?
- Hypothesising can be considered a special case of correlation where:
  - The type graph is recursed backwards
  - Equality constraints are used to infer attribute values of hypothesised alerts
  - Hypothesised alerts may contain only a subset of attribute values
  - Hypothesised alerts with identical attributes are consolidated

Example of Hypothesising

Experimental Protocol

- Exp #1. Performance
  - Algorithm 1a: Correlation
  - Algorithm 1b: Correlation + implicit correlation
  - Algorithm 2a: Hypothesising
  - Algorithm 2b: Hypothesising + merging hyps.
- Exp #2. Hypothesising accuracy
  - Data set with 4 hyper-alert types
  - Data sets with all 2 and 3 combinations of alert types present. (eg. 1 and 2 alert types removed)

Results #1

- Performance: Class B Network

Exp.	Mean Rate (a/s)	Hyper-Alerts	Correlations
1a	126,502	194,817	157,734
1b	176,221	182,727	148,457
2a	151,541	376,786	401,974
2b	154,772	299,395	302,553

- Performance: Class C Network

Exp.	Mean Rate (a/s)	Hyper-Alerts	Correlations
1a	141,088	346,782	888,262
1b	146,675	129,220	641,115
2a	86,380	190,986	691,809
2b	80,440	190,417	691,112

Results #2

- Accuracy of hypothesising algorithm

Input Types	False Negatives	False Positives
ABD	3	12
BCD	32	0
ACD	26	0
ABC	14	0
AC	37	0
BD	35	12
CD	41	0
BC	44	0
AD	35	12
AB	20	0

Analysis

- Roughly 100,000 alerts/sec. Faster than NIDS will realistically produce them
- Real-world data sets: 3,000,000 alerts/sec on a data set consisting of 30,000,000 alerts
- Accuracy of hypothesising algorithm:
  - Robust when 1 or 2 attacks removed from a sequence
  - At least 2 real attacks must be in the full sequence for any hypothesising to take place

Conclusions and Future Work

- Type-graphs can be used for real-time alert correlation
  - Attack models must be carefully designed to avoid combinatorial explosions with hypothesising
  - Memory is unbounded
- Future work (apart from writing up hehe):
  - Memory may be bounded by re-introducing network connectivity matrix in to the attack model
  - Integrate state-based evidence (hard problem)

Thanks!
Please leave a cash donation

Any questions, comments, etc?

???