Lecture 17-18:
Network Configuration Verification and Analysis
Using BDDs

Ehab Al-Shaer
Cyber Defense and Network Assurability (CyberDNA) Center
School of Computing & Informatics
University of North Carolina, Charlotte, NC

April 5, 9, 2010

Acknowledgment: Will Marrero, Hazem Hamed, and Adel El-Atawy
Lecture 17 & 18 Outline

- **Lecture 17**
  - Network Configuration Challenges
  - The need of abstraction in network configuration
  - Limitation of set-theoretic approach
  - Introduction to BDD Configuration Conflict Analysis (ConfigLego)
  - Policy Hardening and Optimization

- **Lecture 18**
  - ConfigChecker: Global End-to-End Network Security Configuration Verification
  - Examples
  - Future research agenda
Role of Security Policies & Configurations

- Security Interfaces
  - Security Policies
    - Security Configurations (e.g., rules, values)
      - Security Protocols (e.g., SSL/TLS, IKE etc)
      - Security Algorithms (e.g., AES, DES)

- Security Network Devices
"Eighty percent of IT budgets is used to maintain the status quo."

"Most of network outages are caused by operators errors rather than equipment failure."

"It is estimated that configuration errors enable 65% of cyber attacks and cause 62% of infrastructure downtime", Network World, July 2006.

Recent surveys show Configuration errors are a large portion of operator errors which are in turn the largest contributor to failures and repair time [1].

"Management of ACLs was the most critical missing or limited feature, Arbor Networks' Worldwide Infrastructure Security Report, Sept 2007.
Challenges of Network Security Configuration

- **Security Systems** are composed of: Algorithms + Protocols + Configuration
- **Network security devices** are policy-based (ACL) devices
  - A policy $P$ is a set of Rules, s.t. $R: <\text{proto}><\text{srcIP}><\text{srcP}><\text{destIP}><\text{destP}> ... \Rightarrow <\text{action}>$

- **Scale challenge** due to large number of devices and rules
  - Policies might have large number of inter-related rules in a single device (15K rules)
  - Policies are distributed, yet inter-connected forming a global security policy
  - Heterogeneous (multi-vendor) security devices

- **Operational semantic Challenge** due to different device roles
  - Rule-order semantics vs. recursive ACL
  - Single-trigger vs. multi-trigger policies
  - Binary vs. multi-value action

- **Network dynamic challenge** due to failures or traffic engineering
  - Multi-domain administration $\Rightarrow$ conflicts due to uncoordinated policy changes
Intra-Firewall Conflicts

- **Shadowing**

  \[ R_x[\text{order}] < R_y[\text{order}], R_x \mathcal{R}_E M R_y, R_x[\text{action}] \neq R_y[\text{action}] \]

- **Correlation**

  \[ R_x \mathcal{R}_C R_y, R_x[\text{action}] \neq R_y[\text{action}] \]

- **Exception**

  \[ R_x[\text{order}] < R_y[\text{order}], R_y \mathcal{R}_I M R_x, R_x[\text{action}] \neq R_y[\text{action}] \]

- **Redundancy**

  \[ R_x[\text{order}] < R_y[\text{order}], R_x \mathcal{R}_E M R_y, R_x[\text{action}] = R_y[\text{action}] \]

  \[ R_x[\text{order}] < R_y[\text{order}], R_y \mathcal{R}_I M R_x, R_x[\text{action}] = R_y[\text{action}] \]

  \[ R_x[\text{order}] < R_y[\text{order}], R_x \mathcal{R}_I M R_y, R_x[\text{action}] = R_y[\text{action}] \text{ and } \exists R_z \text{ where } R_x \mathcal{R}_{\{I, MC\}} R_z, R_x[\text{order}] < R_z[\text{order}], R_x[\text{action}] \neq R_z[\text{action}] \]

- **Irrelevance**

  The path from \( R_x[\text{src}] \) to \( R_x[\text{dst}] \) is not controlled by the firewall.
Intra-Firewall Conflicts

R1: Allow CS to access Registration server
R2: Block Students from accessing Administration-Domain

- If R1 < R2: Students are **ALLOWED** to access the Registration server
- If R2 < R1: Students are **BLOCKED** to access the Registration server

Similarly: what about CS-faculty accessing the Financial-server?

- **Excess**
  - x: udp, 140.192.33.40, any, 161.121.27.40, 53 accept
  - y: udp, 140.192.33.*, any, 161.121.27.*, 53 deny

- **Redundancy**
  - x: udp, 140.192.33.40, any, 161.121.27.40, 53 accept
  - y: udp, 140.192.33.*, any, 161.121.27.*, 53 deny
  - z: udp, 140.192.33.*, any, 161.121.27.*, 53 deny

- **Irrelevance**

  The path from $R_x[src]$ to $R_x[dst]$ is not controlled by the firewall.
Formalization of Inter-Firewall Conflicts

- **Shadowing**
  \[
  R_d R_u \text{EM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \\
  R_d R_u \text{IM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \\
  R_u R_d \text{IM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \\
  R_u R_d \text{IM} R_u, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept} \\
  \]

- **Spuriousness**
  \[
  R_u R_d \text{EM} R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \\
  R_u R_d \text{IM} R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \\
  R_d R_u \text{IM} R_u, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \\
  R_d R_u \text{IM} R_u, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept} \\
  R_u R_d \text{IM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \\
  \]

- **Redundancy**
  \[
  R_d R_u \text{EM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \\
  R_d R_u \text{IM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \\
  \]

- **Correlation**
  \[
  R_u R_d \text{C} R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept} \\
  R_u R_d \text{C} R_d, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \\
  R_u R_d \text{C} R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \\
  R_u R_d \text{C} R_d, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \\
  \]

Uses binary actions & Pair-wise analysis ➔ Does Not Scale
IPSec Inter-Policy Overlapped-tunnel Misconfiguration

- Overlapping tunnels with shared/common traffic
- Traffic decapsulated in reverse order to traffic flow

\[ R_i^{u}[src\_ip] \subseteq R_j^{d}[src\_ip] \text{ and } R_i^{u}[tunnel\_dst] \subseteq R_j^{d}[dst\_ip] \text{ and } \]
\[ \text{Location}(R_i^{u}[tunnel\_dst]) < \text{Location}(R_j^{d}[tunnel\_dst]) \]

### Example Rules

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
<th>Source</th>
<th>Destination</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>1.1.1.1</td>
<td>any</td>
<td>2.2.2.2</td>
<td>protect</td>
</tr>
<tr>
<td>TCP</td>
<td>1.1.1.1</td>
<td>any</td>
<td>2.2.2.2</td>
<td>ESP Tunnel 6.6.6.6 {3DES}</td>
</tr>
<tr>
<td>TCP</td>
<td>1.1.1.1</td>
<td>any</td>
<td>6.6.6.6</td>
<td>protect</td>
</tr>
<tr>
<td>TCP</td>
<td>1.1.1.1</td>
<td>any</td>
<td>6.6.6.6</td>
<td>ESP Tunnel 2.2.2.2 {3DES}</td>
</tr>
</tbody>
</table>

- **Clear Traffic**: E1(T) → 5.5.5.5 → 6.6.6.6 → 2.2.2.2 → E1(T)
Taxonomy of Conflicts in Firewall and IPSec Policies

Network Security Policy Conflicts

Access-list Conflicts

Intra-policy Conflicts

Inter-policy Conflicts

Map-list Conflicts

Nested-session Conflicts

Multi-transform Conflicts

Shadowing

Redundancy

Correlation

Exception

Partial Shadowing

Complete Shadowing

Partial Spuriousness

Complete Spuriousness

*IEEE Communication Magazine, Ehab Al-Shaer and Hazem Hamed, April 2006

Ehab Al-Shaer, Formal Methods in Networking
Looking for a better abstraction

- Limitation of the Set-Theoretic approach
  - Multi-actions will cause exponential growth in conditions.
  - It requires pair-wise analysis of rules
  - It can not be generalized to other ACL devices such as IPSec where multi-trigger and recursive actions are uses
  - It does not support abstraction and composability

- Objectives:
  - Unified/canonical abstraction for different policy semantic
  - Composability
  - Property-based verification
  - Scalability
Modeling Access Control Configuration as Boolean Formulas

\[ <sIPs, dIP, sP, dP, etc> \rightarrow f \]

- Evaluate
- Compare
- Compose

\[ f_1 \circ f_2 \circ f_3 \circ f_4 \]
Modeling ACL Configuration Using BDDs

- An ACL policy is a sequence of filtering rules that determine the appropriate action to take for any incoming packets: \( P = R_1, R_2, R_3, \ldots, R_n \)
- Each rule can be written in the form:
  \[
  R_i := C_i \sim a_i
  \]
  where \( C_i \) is the constraint on the filtering fields that must be satisfied in order to trigger the action \( a_i \)
- The condition \( C_i \) can be represented as a Boolean expression of the filtering fields \( f_1, f_2, \ldots, f_k \) as follows:
  \[
  C_i = f_{v_1} \land f_{v_2} \land \cdots \land f_{v_k}
  \]
  where each \( f_{v_j} \) expresses a set of matching field values for field \( f_j \) in rule \( R_i \). Thus, we can formally describe a ACL policy as:
  \[
  P_a = (C_1 \land b_1) \lor (\neg C_1 \land C_2 \land b_2) \lor (\neg C_1 \land \neg C_2 \land \cdots \land \neg C_{i-1} \land C_i \land b_i)
  \]
  where \( b_i = \begin{cases} 1 & \text{if } action_i = a \\ 0 & \text{if } action_i \neq a \end{cases} \)
Concise Formalization

- **Single-trigger policy** is an access policy where only one action is triggered for a given packet. $C_i$ is the 1st match leads to action $a$

\[
P_a = \bigvee_{i \in \text{index}(a)} (\neg C_1 \land \neg C_2 \ldots \land \neg C_{i-1} \land C_i)
\]

- **Multiple-trigger policy** is an access policy where multiple different actions may be triggered for the same packet. $C_i$ is any match leads to action $a$

\[
P_a = \bigvee_{i \in \text{index}(a)} \bigwedge_{j=1}^{i-1} \neg C_j \land C_i
\]

where \( \text{index}(a) = \{i \mid R_i = C_i \leadsto a\} \)
Introduction to BDD
Boolean variables and functions:

• A boolean variable $x$ is a variable ranging over the range 0 and 1.

• A boolean function $f$ of $n$ arguments is a function from $\{0,1\}^n$ to $\{0,1\}$, $f(n) : B^n \rightarrow B$.

• There are many ways to represent a boolean function.
Boolean functions representation:

• A boolean function $f$ can be represented by:
  • Truth tables.
  • Propositional formulas.
  • Disjunctive Normal Form (DNF), in which a formula is a disjunctions of conjunctions of literals.
  • Conjunctive Normal Form (CNF), in which a formula is a conjunctions of disjunctions of literals.
  • Binary Decision Diagrams (BDDs) (If-Else Normal Form or INM)
    • $x \rightarrow y_1, y_2 \Leftrightarrow (x \land y_1) \lor (\neg x \land y_2)$
    • E.g., $\neg x$ is $(x \rightarrow 0,1)$
What is a BDD?

- **BDD** is a simpler form of Binary decision trees where:
  - Non-terminal nodes are labeled with boolean variables $x, y, z \ldots$
  - Terminal nodes are labeled with either 0 or 1.
  - Each non-terminal node has two edges, one dashed line and one solid line.
  - Dashed line from node $x$ is called $\text{low}(x)$ while the solid line is called $\text{high}(x)$.

- **Reduced (O)BDD** iff
  - **Uniqueness:** if $\text{var}(u) = \text{var}(v)$, $\text{low}(u) = \text{low}(v)$, $\text{high}(u) = \text{high}(v)$ \(\rightarrow u = v\)
  - **Non-redundant test:** $\text{low}(u) = \text{high}(u)$ (\(u\) and \(v\) are different nodes)
**Boolean functions representation:**

<table>
<thead>
<tr>
<th>Representation of boolean functions</th>
<th>compact?</th>
<th>satisf’ty</th>
<th>validity</th>
<th>boolean operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prop. formulas</td>
<td>often</td>
<td>hard</td>
<td>hard</td>
<td>easy</td>
</tr>
<tr>
<td>Formulas in DNF</td>
<td>sometimes</td>
<td>easy</td>
<td>hard</td>
<td>easy</td>
</tr>
<tr>
<td>Formulas in CNF</td>
<td>sometimes</td>
<td>hard</td>
<td>easy</td>
<td>hard</td>
</tr>
<tr>
<td>Ordered truth tables</td>
<td>never</td>
<td>hard</td>
<td>hard</td>
<td>hard</td>
</tr>
<tr>
<td>Reduced OBDDs</td>
<td>often</td>
<td>easy</td>
<td>easy</td>
<td>[medium]</td>
</tr>
</tbody>
</table>

*Figure: Boolean functions representations [CS]*
Representing Boolean Functions

Formula:

\((a \lor c) \land (b \rightarrow d)\)

Normal forms:

\((a \lor c) \land (\neg b \lor d)\)
\((a \land \neg b) \lor (a \land d) \lor (c \land \neg b) \lor (c \land d)\)

Truth table:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Binary Decision Tree

\[(a \lor c) \land (b \rightarrow d)\]
Ordered Binary Decision Diagram

\[(a \lor c) \land (b \rightarrow d)\]
BDD and truth tables

• The main disadvantage of truth tables is the space needed to maintain it.
• if we have 100 variables, we need $2^{100}$ entries in the table.
• in trees, we still need $2^n$ space to maintain it.
• why are BDDs useful?
  • some reductions can be done.
Reducing Decision Trees

Two ways of simplifying decision trees:

1. Identify and share identical subtrees.
2. Remove nodes whose left and right child nodes are identical.

Results in a Reduced Ordered Binary Decision Diagram (OBDD).
Reduction Rule #1: Merge equivalent leaves

From [DP]
Reduction Rule #2: Merge isomorphic nodes
Reduction Rule #3: Eliminate Redundant Tests

From [DP]
Example OBDD

- Canonical representation of Boolean function
  - For given variable ordering
  - Two functions equivalent if and only if graphs isomorphic
    - Can be tested in linear time
  - Desirable property: simplest form is canonical.

Ehab Al-Shaer, Formal Methods in Networking

From [DP]
Properties of BDD

**Storage Efficiency** (often compact)
Many common Boolean functions have small OBDD representations.

**Canonicity**
If the order in which the variables are tested is fixed, then there exists only one OBDD for each Boolean formula.

- **Lemma 1:** (Canonicity lemma)
  For every function $f : \mathbb{B}^n \rightarrow \mathbb{B}$, there is exactly one ROBDD $u$ with variable ordering $x_1 < x_2 < \ldots < x_n$ such that $fu = f(x_1, x_2, \ldots, x_n)$

**Efficient operations**

- **data structure for propositional logic formulas**
  - BDD operations: Build, Apply, Restrict, Existential quantification. SATCount, anySAT, allSAT

Ehab Al-Shaer, Formal Methods in Networking
The Variable Ordering

On every branch in an OBDD, the variables must be tested in the same order, e.g.,

$$a < b < c < d$$

Different variable orderings yield different OBDDs.
Ordered Binary Decision Diagram

\[(a \lor c) \land (b \rightarrow d)\]

\[a < c < b < d\]
Effect of Variable Ordering

\[(a_1 \land b_1) \lor (a_2 \land b_2) \lor (a_3 \land b_3)\]

**Good Ordering**

**Bad Ordering**

Linear Growth

Exponential Growth

From [DP]
Now, APPLY (1/3)

- Let \( v_1, v_2 \) denote that root nodes of \( f_1, f_2 \), respectively, with \( \text{var}(v_1) = x_1 \) and \( \text{var}(v_2) = x_2 \).

1. If \( v_1 \) and \( v_2 \) are leaves, \( f_1 \circledast f_2 \) is a leaf node with value \( \text{val}(v_1) \circledast \text{val}(v_2) \)

\[
\begin{align*}
0 \lor 1 &= 1 \\
0 \land 1 &= 0
\end{align*}
\]

From [DP]
Now, APPLY (2/3)

2. If \( x_1 = x_2 = x \), apply Shannon expansion:
\[
f_1 \oplus f_2 = (\neg x \land f_1 \mid_{x=0} \oplus f_2 \mid_{x=0} \lor x \land f_1 \mid_{x=1} \oplus f_2 \mid_{x=1})
\]

From [DP]
Now, APPLY (3/3)

3. else, suppose $x_1 < x_2$ in the variable order.

$$f_1 \star f_2 = (\neg x_1 \land f_1|_{x=0} \star f_2 \lor x_1 \land f_1|_{x=1} \star f_2)$$

From [DP]
BDDs from below: example.

\[ f_1 : x_1 \leftrightarrow x_2 \]
\[ f_2 : \neg x_2 \]

\[ f_1 \lor f_2 = x_1 \lor (\neg x_1 \land \neg x_2) \]
BDD Operations

- **Negation**: $\neg B$
- **Apply**
  - OR  $B_1 \lor B_2$
  - And  $B_1 \land B_2$
  - Imply  $B_1 \rightarrow B_2$
  - Equivalence  $B_1 \leftrightarrow B_2$
- **Restrict**
  - $\text{Restrict}(1, x, B) = f[1/x]$
  - $\text{Restrict}(0, x, B) = f[0/x]$
- **Existential quantifier**
  $$\exists x f = f[0/x] \lor f[1/x] = \text{apply}(+, \text{restrict}(0, x, B_f), \text{restrict}(1, x, B_f))$$
- **Universal quantifier**
  $$\forall x f = f[0/x] \land f[1/x] = \text{apply}(., \text{restrict}(0, x, B_f), \text{restrict}(1, x, B_f))$$

Ehab Al-Shaer, Formal Methods in Networking
Hints about Variable Ordering

- May not impact the BDD for some (few) problems
  - E.g., parity check
- But it often matters (see previous examples)
- Finding the optimal variable ordering for minimum BDD size is computationally hard (NP complete)
- Many good heuristic obtains often work (built-in in Buddy)
  - Keep correlated variable close
  - Use interleaving variable (x0y0x1y1 ..)
- Application-Based Heuristics
  - Exploit characteristics of application
  - e.g., Ordering for functions of combinational circuit
  - Traverse circuit graph depth-first from outputs to inputs
BDD operations running time

<table>
<thead>
<tr>
<th>Operation</th>
<th>Worst-case</th>
<th>Average-case</th>
</tr>
</thead>
<tbody>
<tr>
<td>MK((i, u_0, u_1))</td>
<td>(O(1))</td>
<td>(O(2^{</td>
</tr>
<tr>
<td>BUILD((t))</td>
<td>(O(</td>
<td>u_1</td>
</tr>
<tr>
<td>APPLY((op, u_1, u_2))</td>
<td>(O(</td>
<td>u</td>
</tr>
<tr>
<td>RESTRICT((u, j, b))</td>
<td>(O(</td>
<td>u</td>
</tr>
<tr>
<td>SATCOUNT((u))</td>
<td>(O(</td>
<td>p</td>
</tr>
<tr>
<td>ANYSAT((u))</td>
<td>(O(</td>
<td>r</td>
</tr>
<tr>
<td>ALLSAT((u))</td>
<td>(O(</td>
<td>r</td>
</tr>
</tbody>
</table>

Note: These running times only hold if dynamic programming is used.

Table 1: Worst-case running times for the ROBDD operations. The running times are the expected running times since they are all based on a hash-table with expected constant time search and insertion operations.
OBDD Packages

**CUDD**
http://vlsi.colorado.edu/~fabio

**Buddy** *(what we used)*
http://buddy.sourceforge.net

**JDD** *(pure Java)*
http://javaddlib.sourceforge.net
BDD Applications in Network Configuration Analysis

Applications

• Conflict Detection

(2) Configuration Hardening
Policy expression $S_a$ represents a policy that incorporates rule $R_i$, and $S'_a$ is the policy with $R_i$ excluded. $R_i$ may be involved in the following conflicts:

- **Shadowing:**
  \[ [(S'_{a_i} \iff S_{a_i}) = true] \text{ and } [(C_i \Rightarrow S'_{a_i}) = false] \]

- **Redundancy:**
  \[ [(S'_{a_i} \iff S_{a_i}) = true] \text{ and } [(C_i \Rightarrow S'_{a_i}) \neq false] \]

- **Exception:**
  \[ [(S'_{a_i} \iff S_{a_i}) \neq true] \text{ and } [(C_i \Rightarrow S'_{a_i}) = false] \]

- **Correlation:**
  \[ [(S'_{a_i} \iff S_{a_i}) \neq true] \text{ and } [(C_i \Rightarrow S'_{a_i}) \neq false] \]
**IPSec Inter-Policy Conflicts Formalization: Crypto-access Lists**

- **Shadowing**: upstream policy blocks traffic

\[
\left[ (S^u_{\text{discard}} \land \neg S^d_{\text{discard}}) \lor (S^u_{\text{protect}} \land \neg S^d_{\text{protect}}) \right] \neq \text{false}
\]

Traffic dropped

1.1.1.1

TCP 1.1.1.1 2.2.2.2

Traffic dropped

2.2.2.2

TCP 2.2.2.2 1.1.1.1

Ehab Al-Shaer, Formal Methods in Networking
IPSec Inter-Policy Conflicts Formalization: Crypto-access Lists cont.

- **Spurious**: downstream policy blocks traffic

\[
[(S^{u}_{bypass} \land \neg S^{d}_{bypass}) \lor (S^{u}_{protect} \land S^{d}_{discard})] \neq false
\]
Security Policy Advisor Tool for Distributed Firewall & IPsec
Companies and Institutions Using Security Policy Advisor

- **Companies:**
  - Lisle Technology Partners, USA; Phontech, Norway; Naval Surface Warfare Center, Panama City, USA; Cisco Systems, USA; At&T, USA; Gateshead Council, UK; Danet Group, Germany; TNT Express Worldwide, UK Ltd, United Kingdom; Checkpoint, USA; FireWall-1, The Netherlands; DataConsult, Lebanon; Rosebank Consulting, GB; Mayer Consulting, USA; Panduit Corp, USA; UPMC Paris 5 University, France; Royal institute of Science, Sweden; GE, US; Aligo, USA; Motorola, Inc., USA; Landmark communications, inc., us; uekae.tubitak.gov, Turkey; Duke Energy, USA; The Midland Co, USA; NITW,INDIA; Deloitte & Touche LLP, US; National Taiwan University, Taiwan; Eircom.net. Ireland; GE CF, USA; AIT, Thailand; Celestica, Thailand; and Others not listed

- **Universities/Institutions:**
  - ISRC, Queensland University of Technology, Australia; Imperial College and UCL, London, UK; Columbia University, USA; Georgia Institute of Technology ;NCSU, USA; USC, USA; University of Pittsburgh, PA; University of Waterloo, Canada; University Student in Cyprus International University, Cyprus; University of Rochester, US; UQAM, University of Quebec in Montreal, Canada; Saarland University, Germany; Technical University of Berlin, Computer Science Department, Germany; UCSB, US; Edith Cowan University, Australia; Universitat Oberta de Catalunya, Spain; ISG, Tunisia; York U, Toronto, Canada; Universidade Federal do Rio Grande do Sul, Brazil; UCL, Belgium; Kent State University, USA; UFRGS, Brazil; University of Stuttgart, IKR, Germany;
Composable Security Configuration Verification & Analysis

Themes:

- Security Configuration Hardening
- Integrating other device and host configuration
- Property based verification
Modeling Routing Access Control

- We can define the routing policies as follows: let a routing rule be encoded as $R_i := D_i \leadsto n$

  - Where $n$ is an integer representing the forwarding port ID where $D_i$ is the destination and $n_i$ is a unique integer (id) designating the next hop in the network. Thus, the policy of the routing entries (ordered based on longest-common prefix) that forward to next hop $n_k$ can be defined as follows:

$$T_n = \bigvee_{i \in \text{index}(n)} \bigwedge_{j=1}^{i-1} \neg D_j \land D_i \text{ s.t. index}(n) = \{i \mid R_i = D_i \leadsto n\}$$

- We can then represent the entire routing table for a node $j$ as follows:

$$T^j = \bigvee_{\forall n = \text{next hop}} T_n$$
Composability: Path Conflict Analysis for Firewalls

- **Lemma:** If $S_{A}^{u}$, $S_{A}^{d}$ are the upstream and downstream firewalls in a path, then
  (a) $S^{u}$ causes inter-policy shadowing with $S^{d}$ iff $\neg (S_{A}^{u} \land S_{A}^{d}) \neq false$
  (b) $S^{u}$ causes inter-policy spuriousness with $S^{d}$ iff $(S_{A}^{u} \land \neg S_{A}^{d}) \neq false$

- **Lemma:** Shadow-free and spurious-free are transitive relations. Thus, assume $S_{A}^{i}$, $S_{A}^{j}$ and $S_{A}^{k}$ are upstream to downstream firewall polices in a path $a$, the following relation is always true (shadowing-free case):

  $$[(\neg S_{A}^{i} \land S_{A}^{j}) = false] \land [(\neg S_{A}^{j} \land S_{A}^{k}) = false] \Rightarrow [(\neg S_{A}^{i} \land S_{A}^{k}) = false]$$

- **Path Conflict:** Assuming $S_{A}^{1}$ to $S_{A}^{n}$ are the firewall policies from upstream to downstream in the path from $x$ to $y$, a path conflict $(x,y)$ between any two firewalls from $i$ to $n$ path is defined as follows:

  (a) **Path-Shadowing** $(x,y)$:

  $$\bigvee_{i=1,n-1 \text{ and } i \in \text{path}(x,y)} \neg S_{A}^{i} \land S_{A}^{i+1} \neq false$$

  (b) **Path-Spuriousness** $(x,y)$:

  $$\bigvee_{i=1,n-1 \text{ and } i \in \text{path}(x,y)} S_{A}^{i} \land \neg S_{A}^{i+1} \neq false$$
Diagnosing Unreachability Problems between Routers and Firewalls

- **Flow-level Analysis:** Is the flow $C_k$ that is forwarded by routers in path $P$ (each routing tables is represented as \text{BDD} $T^i_j$ for router $i$ and port $j$) but blocked due to conflict between Routing and FW Filtering:

$$[(C_k \Rightarrow \bigwedge_{(i,j) \in P} T^i_j) \land (C_k \Rightarrow \neg S^m_A)] \neq \text{false}$$

- This shows that a traffic $C_i$ is forwarded by the routing policy, $T^i_j$, from node $i$ to $n$ but yet blocked by the filtering policy, $S^m_{\text{discard}}$, of the destination domain.

- **Path-level Analysis:** What are all unreachability Conflicts between Routing and Filtering:

$$\phi_k \leftarrow [\text{SAT}^\ast(\bigwedge_{(i,j) \in \text{path}(P)} T^i_j \land \neg S^m_A \land \neg(\bigwedge_{i=1,k-1} \phi_i))] \neq \text{false}$$

- For $\phi=1$, n misconfiguration examples, and $\phi(0) = \text{ture}$

- **Network or Federated-level Analysis:** Spurious conflict between downstream $d$ and upstream $u$ ISP domains:

$$[(S^u_{\text{bypass}} \land \neg S^d_{\text{bypass}}) \lor (S^u_{\text{limit}} \land S^d_{\text{discard}})] \neq \text{false}$$

- Notice that $S^u_{\text{discard}}$, $S^u_{\text{bypass}}$ and $S^u_{\text{limit}}$ are filtering policies representations related to the filtering actions as described in [POLICY08, ICNP05, CommMag06].

Ehab Al-Shaer, Formal Methods in Networking

*: AnySAT
Automating Hardening of Security Configuration
Security Hardening & Intrusion Response

- Given the Boolean formula $P$ that represents the configuration of the entire network, $k$, with variables $v_1, \ldots, v_n$, what are all configuration changes to block all attack scenarios $a_i$ without violating the requirements $H_i$.
  - Example of $A_i$: ($\ast \rightarrow \text{telnetServer}/23$) and ($\text{ftpServer}/\text{any} \rightarrow \text{SQLServer}/550$)
  - Example of $H_i$: ($\text{SQLServer}/\ast \rightarrow \text{DNS}/51$)

$$\phi_i = SAT_{(i=1,n)} (P_A^k \land \neg(\bigvee_{i=1}^n H_i) \land (\bigvee_{i=1}^n A_i) \land \neg(\bigwedge_{i=1,k-1} \phi_i))$$

- Assume that variables $v_1, \ldots, v_n$ are associated with cost $c_1, \ldots, c_n$, what is the most cost-effective configuration changes to block attack scenarios $A_1, \ldots, A_n$ without violating the requirements $H$.

$$\phi_{\text{minCost}} = \text{MinCostSAT}(H \land \neg(\bigvee_{i=1}^n A_i))$$

$$P_A^k \leftarrow P_A^k \land \phi_{\text{minCost}}$$

- To look for minimum number of config changes, assign the same cost as minCostSAT will minimize.

$$C = \sum_{i=1}^n c_i \times v_i$$

Ehab Al-Shaer, Formal Methods in Networking
ConfigLego

- Reporting Interface
- Visual User Interface
- User (C) Program
- ConfigLego API
- ConfigLego BDD Abstraction and Engine
- Network Device Configuration (Files)

Ehab Al-Shaer, Formal Methods in Networking
ConfigLego Examples
Recap

- BDD can be used as primitives for configuration analysis
- Conflict/Inconsistency Analysis
- Fine-grain configuration optimization
- Configuration debugging and tracing
- Focus/limited configuration invitation and analysis