# Automated Network-wide Security Analysis (ANSA)

**Petr Matoušek**

`matousp@fit.vutbr.cz`

Brno University of Technology, Czech republic

# Talk Outline

1. **Motivation**
   - **Cisco experience & Network complexity**
   - **Packet Tracer demo**
2. **State-of-the-Art**
   - **Approaches to Analysis of IP Networks – Network Models.**
   - **Packet Classification.**
3. **ANSA project**
   - **Formal Description of the Modelled Network.**
   - **Properties Specification.**
   - **Analysis using Model Checking Approach.**
4. **Conclusion**
5. **References**

# 1. Motivation

❖ **Network Traffic Analysis**

- **Packet flows over network depends on**
  - **dynamically changed routes**
  - **filtering by ACLs**
  - **tranformation by NAT**

❖ **Security Issues**

- **"Can these two hosts communicate?"**
- **"Is this server protected even if some links fail?"**
- **"What-if" failure analysis**

❖ **Testing (ping, traceroute) does not give an answer!**

⇒ **Further security behaviour analysis is needed.**

# 1. Motivation – Research Framework

- **Assumptions**
  - **We have configuration of routers.**
  - **We have a network topology description.**
  - **We know security properties to be checked.**
- **Our goal:**

1. **Automated analysis of network security properties**
   - **Reachability under security policies.**
   - **Analysis of complex communication patterns.**
2. **Automatic generation of secure network configuration under admin requirements.**

# 2. State-of-the-Art

❖ **Unifying Model of the Network [1]**

- **a graph of routers and links:**
  - **vertices – routers**
  - **edges – physical links**
- **a graph of routing processes**
  - **vertices – routing processes**
  - **edges – adjacencies**

❖ **Packet Classification [2]**

- **logical representation**
- **implementation**

# 2.1 The graph of the network

## ❖ The graph of routers and links

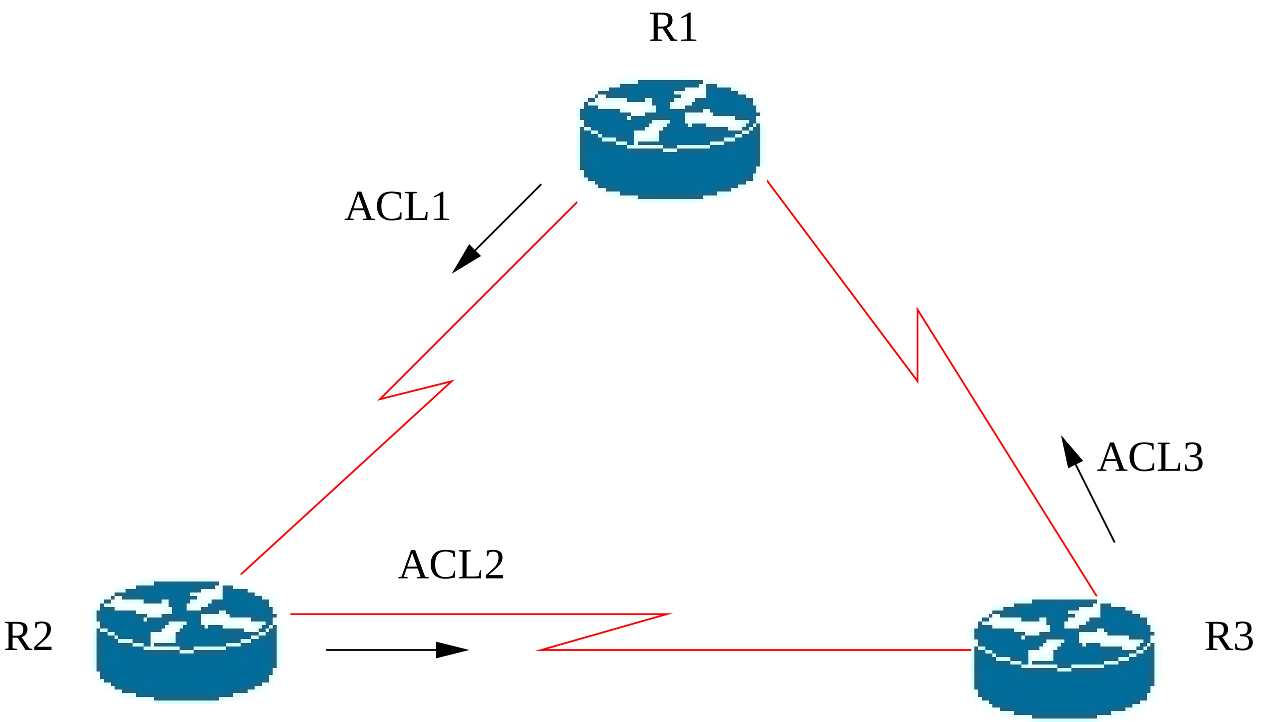
- **a graph:** $G = (V, E, \mathcal{F})$
- **labeling function** $\mathcal{F}$ **(a set of packet filters** $f$**)**
- **an edge:** $< u, v > \in E, F_{u,v} \in \mathcal{F}$
- **filters on the edge from** $u$ **to** $v$**:** $F_{u,v}$
- **packet filter** $f$**: a serie of predicates over packet elements**

## ❖ Example

- $f$ **= (p.src_addr** $\in$ **128.2/16)** $\wedge$ **(p.dest_port** $\neq$ **135)**
- $F_{u,v} = \{p | f(p) = 1\}$

# 2.1 The graph of the network

❖ **Example**




❖ $G = (V, E, F)$

- $V = \{R_1, R_2, R_3\}$
- $E = \{< R_1, R_2 >, < R_2, R_3 >, < R_1, R_3 >\}$
- $F = \{ACL1, ACL2, ACL3\}$**,**
  **where** $F_{R_1,R_2} = \{ACL1\}, F_{R_2,R_3} = \{ACL2\}, F_{R_3,R_1} = \{ACL3\}$

# 2.2 Reachability

❖ **Reachability**

- **a function of the networks's forwarding state $s \in \mathcal{S}$ (it may change)**
- $\mathcal{S}$ **– a set of all possible forwarding states**
- $I_u(s,d)$ **– a set of next hop routers from router $u$ to subnet $d$**
- $F_{u,v}(s) = F_{u,v} \cap \{p | p.\mathbf{dst\_addr} \in \{d | v \in I_u(s,d)\}\}$
- $\mathcal{P}(i,j)$ **– a set of all loop-free path from $i$ to $j$**
- **Reachability from $i$ to $j$ at routing state $s$:**

$$R_{i,j}(s) = \bigcup_{\pi \in \mathcal{P}(i,j)} \bigcap_{<u,v> \in \pi} F_{u,v}(s)$$

# 2.3 Representation of Packet Filters

- **Representation inspired by [3]**
- **Example of filtering rules**

```
access-list 108 permit tcp 192.134.0.0/24 any eq www
access-list 108 deny tcp any any
access-list 108 deny ip any any
```

# 2.3 Representation of Packet Filters

- **Representation inspired by [3]**
- **Example of filtering rules**

```
access-list 108 permit tcp 192.134.0.0/24 any eq www
access-list 108 deny tcp any any
access-list 108 deny ip any any
```

- **Formal Description**
  - $H$ **– the finite set of all possible headers**
  - $\Pi = \{permit, deny\}$ **– the set of policies**
  - **rule** $r = (\eta, \pi)$, **with** $\eta \subseteq H \wedge \pi \in \Pi$**.**

# 2.3 Representation of Packet Filters

- **Representation inspired by [3]**
- **Example of filtering rules**

```
access-list 108 permit tcp 192.134.0.0/24 any eq www
access-list 108 deny tcp any any
access-list 108 deny ip any any
```

- **Formal Description**
  - $H$ **– the finite set of all possible headers**
  - $\Pi = \{permit, deny\}$ **– the set of policies**
  - **rule** $r = (\eta, \pi)$**, with** $\eta \subseteq H \wedge \pi \in \Pi$**.**
  - **Example:**

```
r_1 = ((source_ip = 192.134/24) /\ (proto=www),permit)
r_1 = ((source_ip = 0.0.0.0/0) /\ (proto=tcp),deny)
r_1 = ((source_ip = 0.0.0.0/0) /\ (proto=ip),deny)
```

# 2.3 Representation of Packet Filters

## ❖ Filter as First-Order Logic Formula

- **filter = a set of rules** $H \times \Pi$
- $\varphi = ((\eta_1, \pi_{k_1}), (\eta_2, \pi_{k_2}), \ldots, (\eta_n, \pi_{k_n}))$
  **where** $\pi_{k_i} \in \Pi$ **and** $\eta_i \in H, \forall i \leq n$.

## ❖ Extension of filters as a mapping function

- **filter** $\varphi$ **as a function that maps any header** $h$ **to** $permit$ **and/or** $deny$**.**
- **formally:** $\varphi \rightarrow \{permit, deny, \{permit, deny\}\}$
- $\varphi(h) = \{\pi_{k_i} \in \Pi / h \in \eta_i\}$

## ❖ How to represent formulas?

- **Difference Bound Matrices (DBMs)**
- **Binary Decision Diagrams (BDDs)**
- **Interval Decision Diagrams (IDDs)**

# 2.4 Data Structure IDD

- **Interval Decision Diagrams (IDDs)**
  - **an efficient data structure to store FOL formulas**
  - **efficient in both space and computational time**
  - **allows classification on integer numbers**
- **Structure of an Interval Decision Diagrams, definition see [3]**
  - **IDD is a directed acyclic graph (DAG)**
  - **each node – to test on an integer variable (e.g. x,y,z)**
  - **links**
    - **to another node – associated with an interval within the domain**
    - **to a boolean** *terminal* **(**$True$ **or** $False$**)**

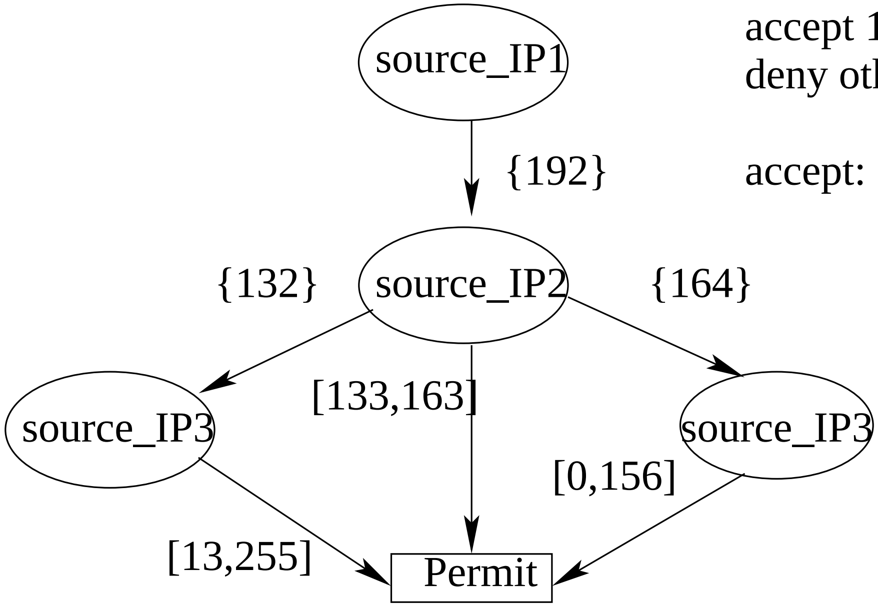
# 2.4 Data Structure IDD

- $\varphi = (x = 0 \wedge y \leq 3) \vee (1 \leq x \leq 6 \wedge z \leq 6) \vee (x = 7 \wedge y = 1)$
- **corresponds to:**

$$\begin{aligned}
t_0 &= x \rightarrow (\{0\}, t_{00})([1,6], t_{000})(\{7\}, t_{01}) \\
t_{00} &= y \rightarrow ([0,3], T)([4,7], F) \\
t_{01} &= y \rightarrow (\{0\}, F)(\{1\}, T)([2,7], F) \\
t_{000} &= x \rightarrow ([0,6], T)(\{7\}, F)
\end{aligned}$$

# 2.4 IDDs and Packet Filters

## ❖ IDD to store packet filters




accept 192.132.13.*−192.164.156.*
deny others

accept: 192.132.13.x/24 − 192.132.255.x/24
192.133.x.x/16 − 192.163.x.x/16
192.164.0.x/24 − 192.164.156.x/24

## ❖ Complexity $\mathcal{O}(m.log\ r)$

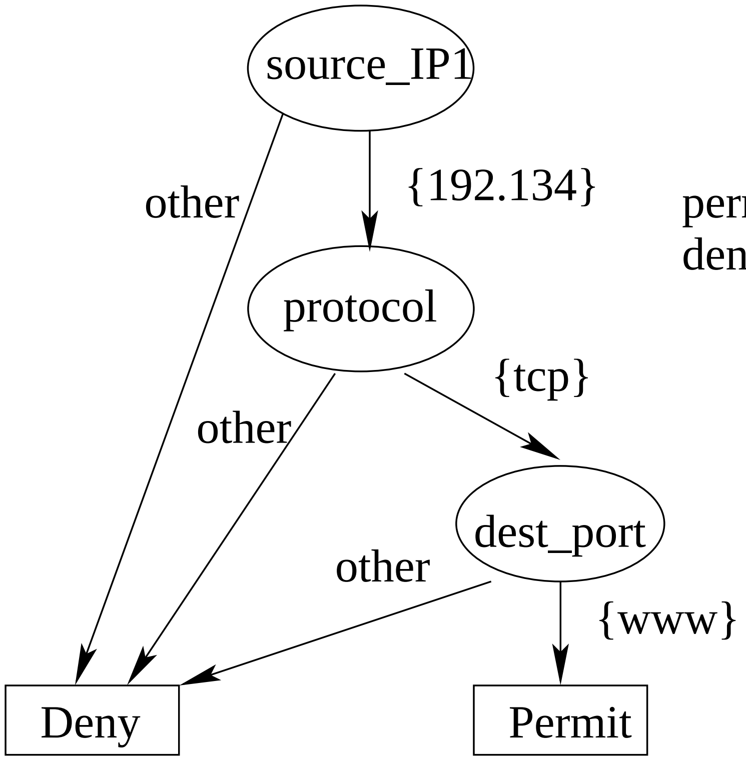
- **m – number of fields, r – number of intervals**
- **worst case: independent rules $\mathcal{O}(m.r)$**

# 2.4 IDDs and Packet Filters

- **Manipulation with IDDs**
  - **operations** $and$**,** $or$**,** $negation$**,** $equivalence$
- **Optimization of IDDs**
  - **Interval Merging**
  - **Node Pruning**
  - **Subtree Merging**
- **Multi-value IDDs – fields of different domains (not only IPs)**
  - **source, destination IP + mask (only IPv4 now)**
  - **source, destination port (1-65636)**
  - **protocol (tcp, udp, ip, icmp)**
  - **ToS (type of service)**

# 2.4 IDDs and Packet Filters

- **Multi-value IDD – Example:**



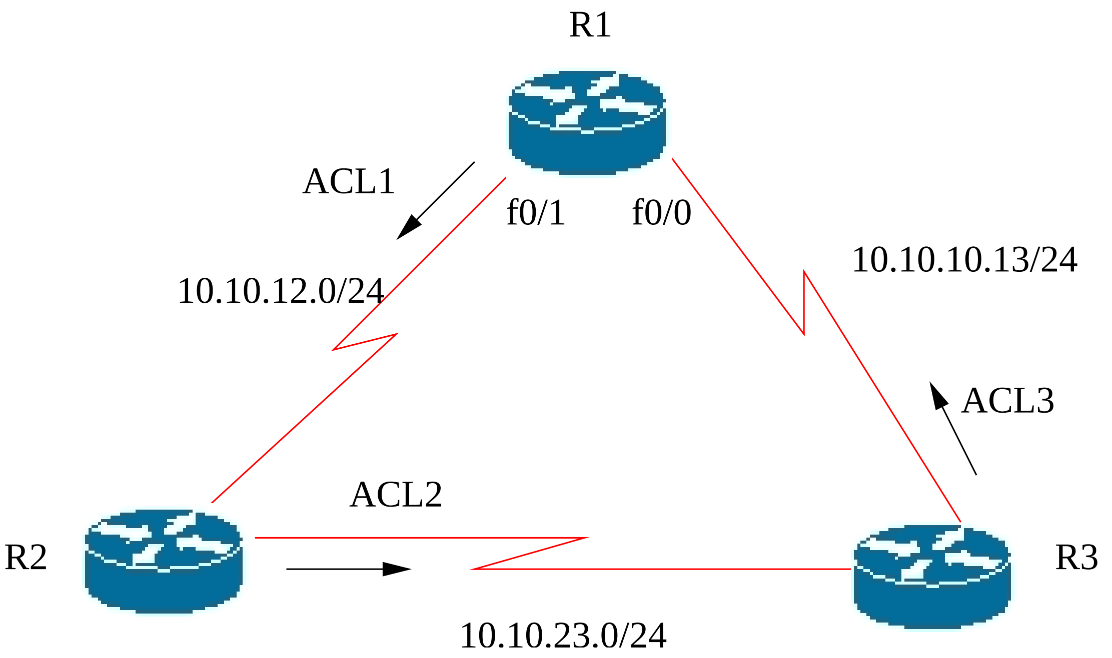

permit 108 tcp 192.134.0.0/24 any eq www
deny ip any any

- **Filtering over multiple fields**

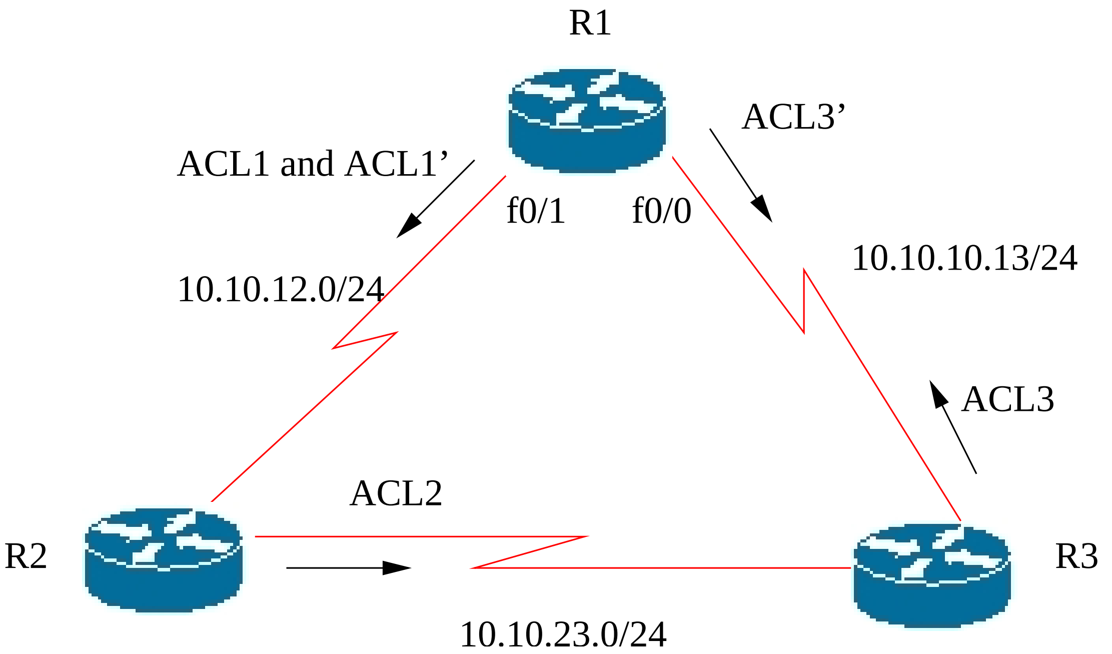
# 2.5 Adding routing information

## ❖ Routing table – example of router R1

| destination network | metric | outgoing interface |
|---|---|---|
| 10.10.12.0/24 | directly connected | f0/1 |
| 10.10.13.0/24 | directly connected | f0/0 |
| 10.10.23.0/24 | 120/1 | f0/1 |
| 10.10.23.0/24 | 120/1 | f0/0 |



# 2.5 Adding routing information




## ❖ Converting routing information into packet filters

- $F_{R_1,R_3} = \{ACL3'\}$
  ```
  permit ip any 10.10.21.0
  permit ip any 10.10.13.0
  deny ip any any
  ```
- $F_{R_1,R_2} = \{ACL1'\}$
  ```
  permit ip any 10.10.12.0
  permit ip any 10.10.23.0
  deny ip any any
  ```

# 2.6 Security Properties

- **Work to do**
- **How to describe them**
  - **Temporal Logic Formulas**
  - **??**

# 2.7 Automated Analysis

❖ **Verification**

- **create a new tool for simple model-checking**

# 3. ANSA project

- **ANSA team**
  - **Petr Matoušek, Jaroslav Ráb, Ondřej Ryšavý**
- **Road map**

1. **Understand current approaches.**
2. **Select/create a formal language to decribe the network configuration.**
3. **Compute "dynamic" configuration from static configuration files.**
4. **Add dynamic information to the formal model.**
5. **Create a transition system (automaton) describing the model behaviour.**
6. **Verify the security properties using model-checking approach or static analysis.**

# 4. Conclusion

❖ **What we have?**

- **A good understanding of packets transmission over networks by Cisco Academy.**
- **The goal to reach – automated analysis of network security.**

❖ **Future steps – near future**

- **Explore different methods how to describe the network model.**
- **Compute routing table from the static configuration on a small network model.**
- **Describe packet filters on router's interfaces.**
- **Perform security analysis on the small model (under dynamic model).**
- **Publish first results.**

# 4. Conclusion

❖ **Future steps – next year(s)**

- **Automatic transformation of Cisco configuration into our model.**
- **Transformation of routing tables (RIP, EIGRP, OSPF, BGP) into the model.**
- **Find/create a model-checker to verify the results (or static analysis).**

❖ **Testing on real devices:**

- **Reachability under security policies.**
- **Generation of secure configuration files for Cisco routers.**

## 5. References

# References

[1] G. Xie, J.Zhan, D.A.Maltz, H.Zhang: "On Static Reachability Analysis of IP Networks", Infocom 2005.

[2] M. Antoš: "Hardware-constrained Packet Classfication", PhD. thesis, 2006.

[3] M. Christiansen, E.Fleury: "An Interval Decision Diagram Based Firewall", IEEE International Conference on Networking (ICN ’04). Colmar, France, 2004.