The VEREFOO Network Security Automation Approach

Riccardo Sisto
Politecnico di Torino

SoSySec Presentation, January 19th, 2024, Rennes

netgroup

1859

Outline

- VEREFOO Context and Motivation
- The VEREFOO Approach
- Latest VEREFOO Developments
- Conclusions

VEREFOO Context and Motivation

Network Security Configuration

- Traditionally, network security is configured **manually** with **trial-and-error** naive approaches:
  - first, administrators configure security according to their **initial** threat model;
  - if later a cyber attack occurs, they simply **modify** the configuration that could not prevent the attack, to avoid possible repetitions.

- Such an approach works only with **small-sized** networks, where everything is almost **static** and under the direct control of a human user.

Security Misconfiguration

In 2021 82% of breaches involved human errors in security configuration.

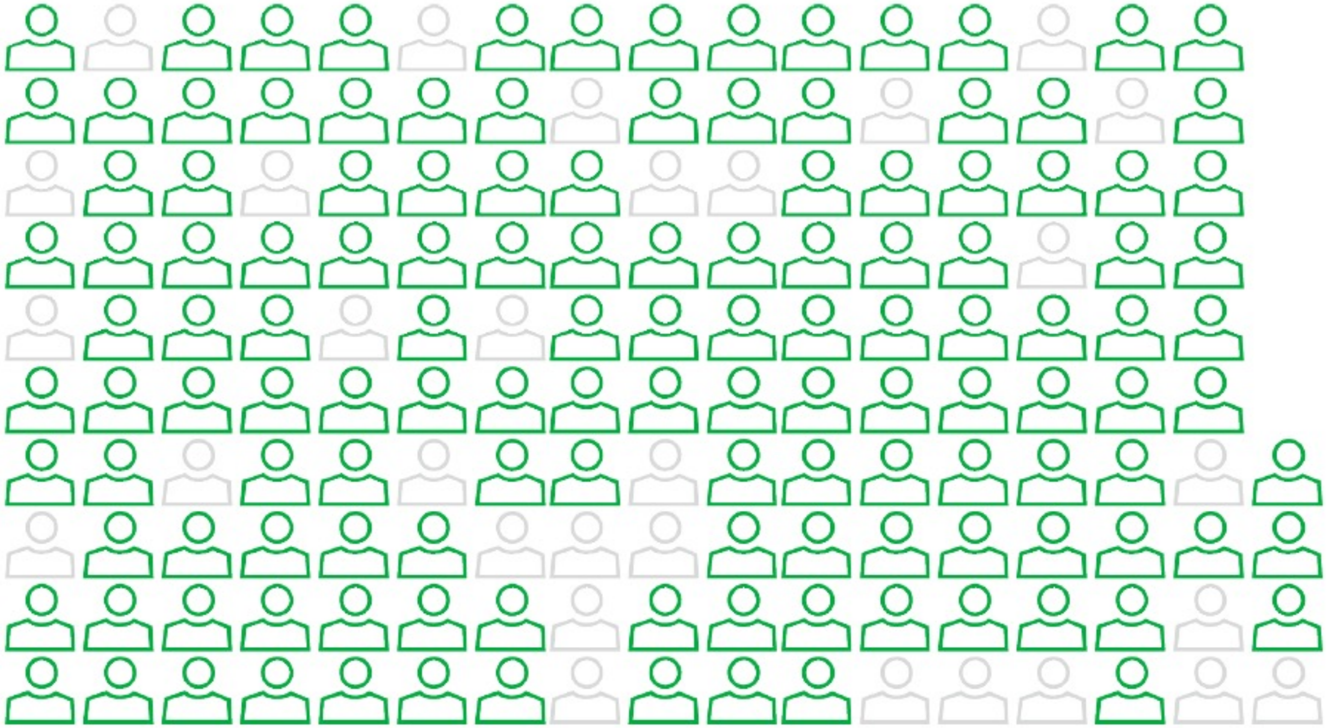
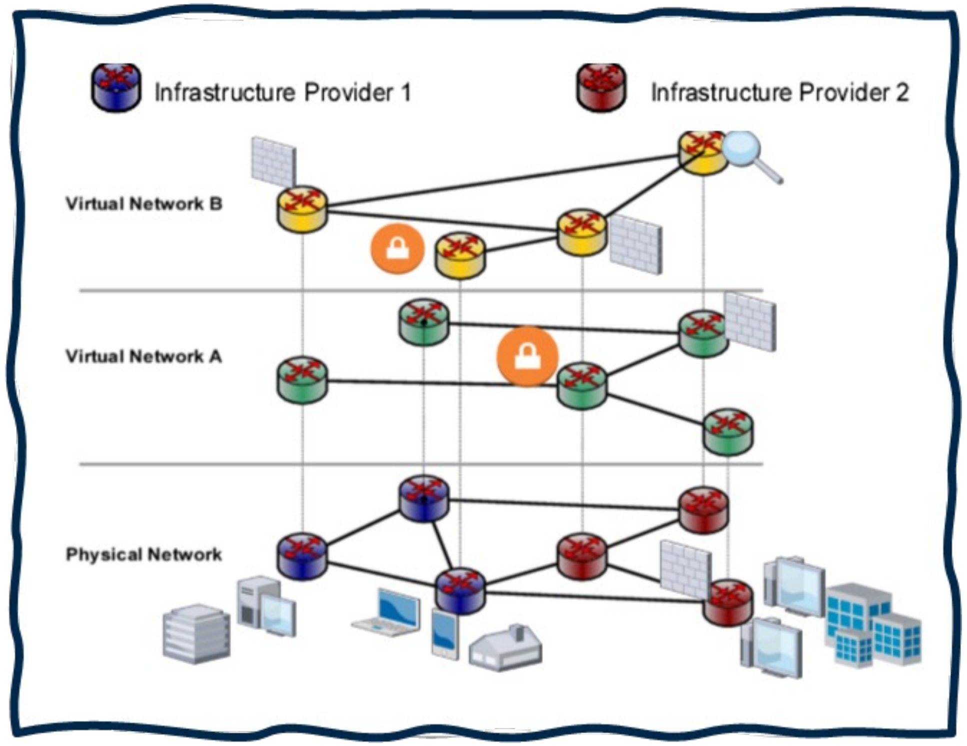


Figure 9. The human element in breaches (n=4,110)
Each glyph represents 25 breaches.

Security Management in Virtualized Networks

SDN, NFV, IaaS, IaC



In virtualized networks, **security management** has become a daunting task because of:

- increasing **dynamism** and agility;
- increasing network **size**;
- increasing **heterogeneity**.

Manual Security management does not work any more.

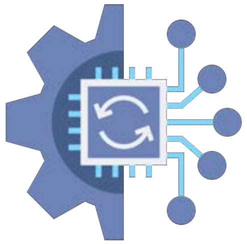
VEREFOO: Problem Statement

Main problem:

Manual management is error-prone, unoptimized, time consuming.

Automation can help security management:

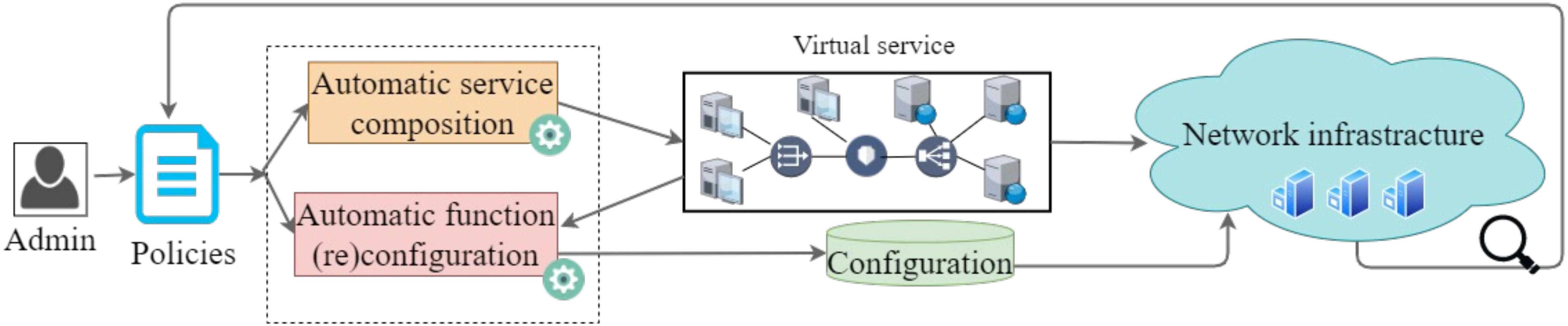
- **avoid** manual trial-and-error configuration;
- exploit the **agility** of network virtualization;
- leverage **formal verification** and **optimization**.



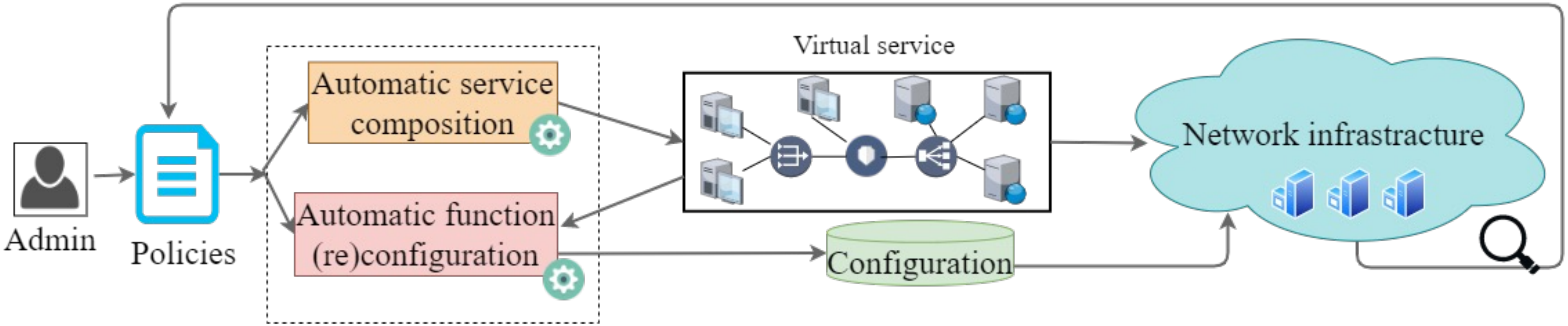
VEREFOO: Wish List

- Manage both **Traditional** and **Virtual** Networks
  - support networks with different types of middleboxes (LB, NAT, etc.)
  - support not only **function chains** but also **network service graphs**
- Full Automation
  - administrator provides only network graph/conf and security requirements
  - automatic network security design from scratch
- Formal Correctness Guarantee
- Optimization
- Vendor-Independence
- Scale to significant network size

VEREFOO: Policy-Based Security Management Workflow



A Policy-Based Workflow for Security Management



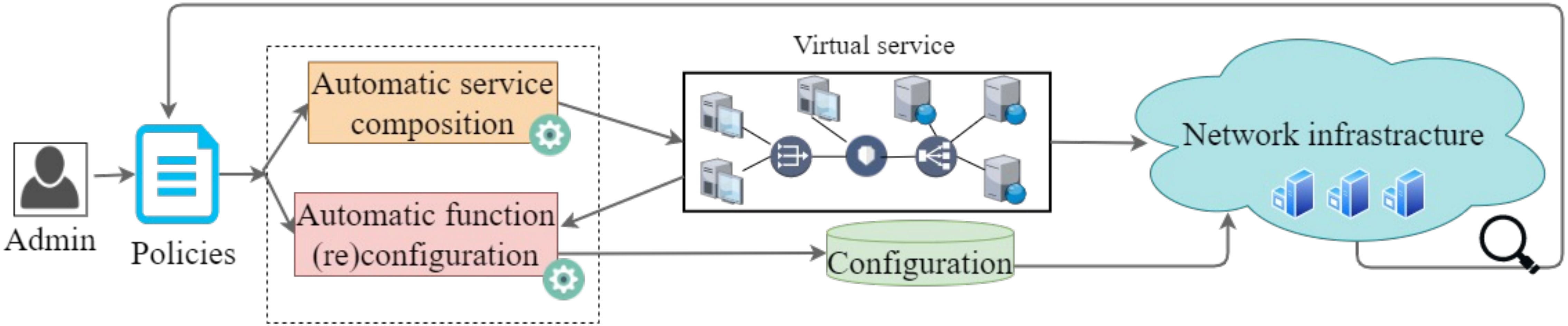
1

Policy Specification

Policy Specification

- A human user specifies the **security requirements** that must be enforced in the computer network by means of **policies**.
  - RFC-3198 definition: “a network security policy is a **set of rules** to administer, manage, and control access to the network resource”.
- The security policies should be specified:
  - with a **user-friendly** language, usable by an administrator with limited security skills;
  - **abstracting** from the vendor-dependent characteristics of NSF implementations.
- Example:
  - “All the traffic between the sub-networks of the IT and Business departments must be encrypted when outside those sub-networks”

A Policy-Based Workflow for Security Management



1

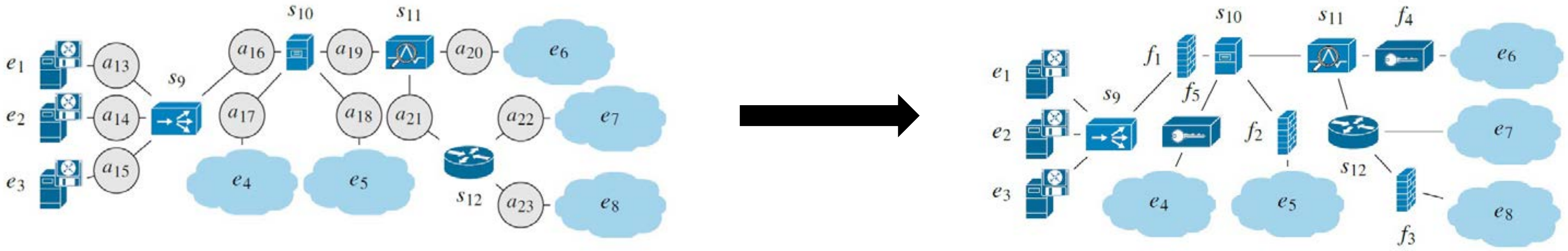
Policy Specification

2

Security Configuration

Security Configuration

- Security configuration involves refining the policies into two elements:
  1) service structure composition;

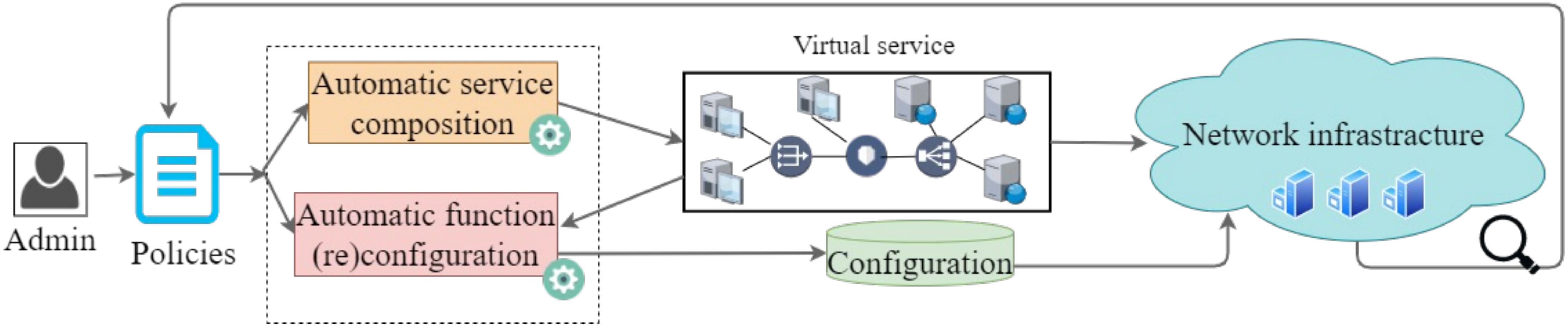


  2) NSF configuration rules (expressed in medium-level language).

#	Action	IPSrc	IPDst	pSrc	pDst	tProto
1	Allow	130.192.225.*	220.226.50.2	*	80	TCP
2	Allow	130.192.120.*	220.226.50.3	*	*	*
3	Allow	220.226.50.*	130.192.*.*	*	*	*
D	Deny	*.*.*.*	*.*.*.*	*	*	*

Action	IPSrc	IPDst	pSrc	pDst	tProto	algorithm
Enforce	192.168.0.2	220.226.50.*	*	*	*	HMAC-SHA2-256
Remove	192.168.0.3	220.226.50.*	*	*	*	AES-GCM-256
Enforce	130.192.225.*	220.226.50.3	*	*	*	HMAC-SHA2-512

A Policy-Based Workflow for Security Management



1 Policy Specification

2 Security Configuration

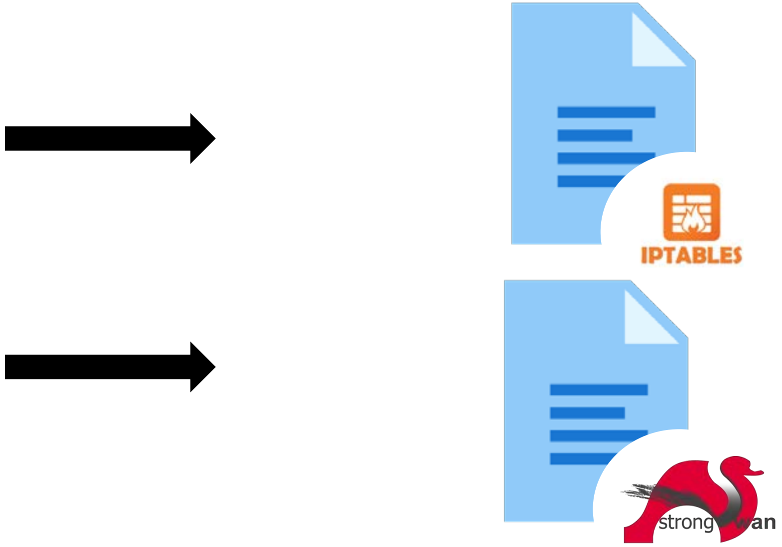
3 Security Deployment

Security Deployment

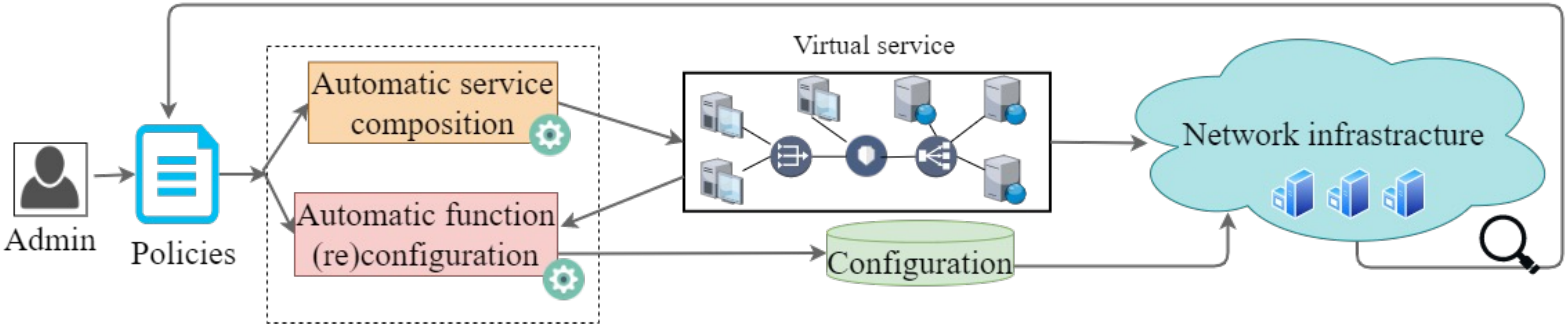
- The NSFs composing the security service are **deployed** in the network.
- The configuration produced in the previous stage is translated into **low-level languages** and installed in the NSF instances.
  - This step is a simple **syntax change**.

#	Action	IPSrc	IPDst	pSrc	pDst	tProto
1	Allow	130.192.225.*	220.226.50.2	*	80	TCP
2	Allow	130.192.120.*	220.226.50.3	*	*	*
3	Allow	220.226.50.*	130.192.*.*	*	*	*
D	Deny	*.*.*.*	*.*.*.*	*	*	*

Action	IPSrc	IPDst	pSrc	pDst	tProto	algorithm
Enforce	192.168.0.2	220.226.50.*	*	*	*	HMAC-SHA2-256
Remove	192.168.0.3	220.226.50.*	*	*	*	AES-GCM-256
Enforce	130.192.225.*	220.226.50.3	*	*	*	HMAC-SHA2-512



A Policy-Based Workflow for Security Management



1 Policy Specification

2 Security Configuration

3 Security Deployment

4 Security Mitigation

Security Mitigation

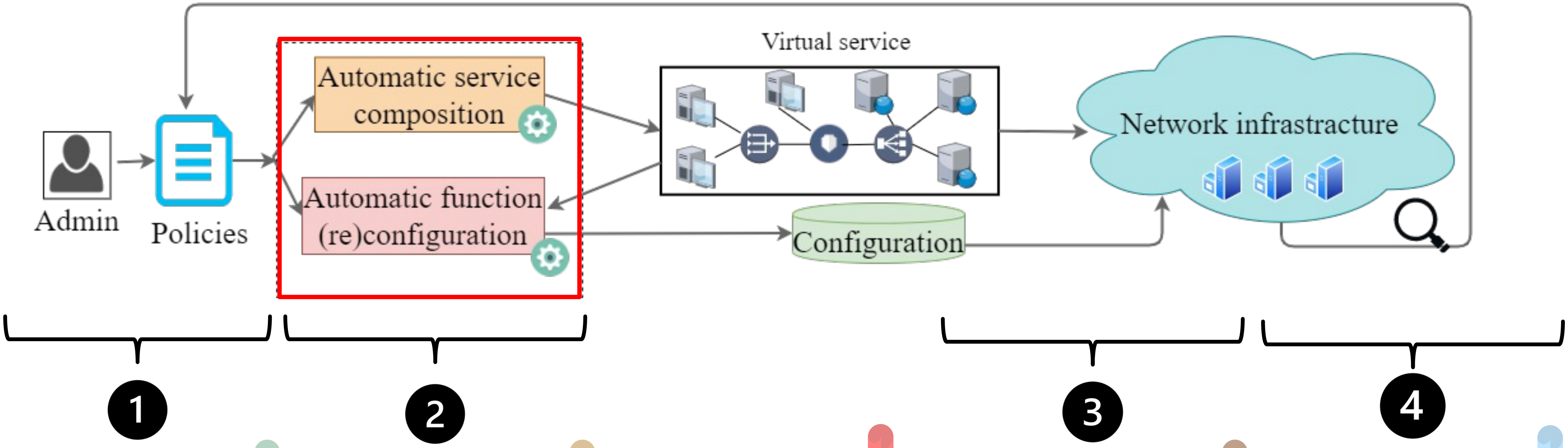
- **Monitoring** agents (e.g., intrusion detection systems) analyze network traffic for the identification of cyber-attacks.
- If an attack is identified, a **mitigation** process is triggered:
  - **new** security policies may be defined to mitigate the unexpected attack;
  - **old** security policies may have to be discarded;
  - a new **security configuration** is computed from the updated policies and deployed.

Security Mitigation

- **Monitoring** agents (e.g., intrusion detection systems) analyze network traffic for the identification of cyber-attacks.
- If an attack is identified, a **mitigation** process is triggered:
  - **new** security policies may be defined to mitigate the unexpected attack;
  - **old** security policies may have to be discarded;
  - a new **security configuration** is computed from the updated policies and deployed.

Security mitigation closes the **loop** of the security management workflow.

A Policy-Based Workflow for Security Management



Policy Specification

Security Configuration

Security Deployment

Security Mitigation

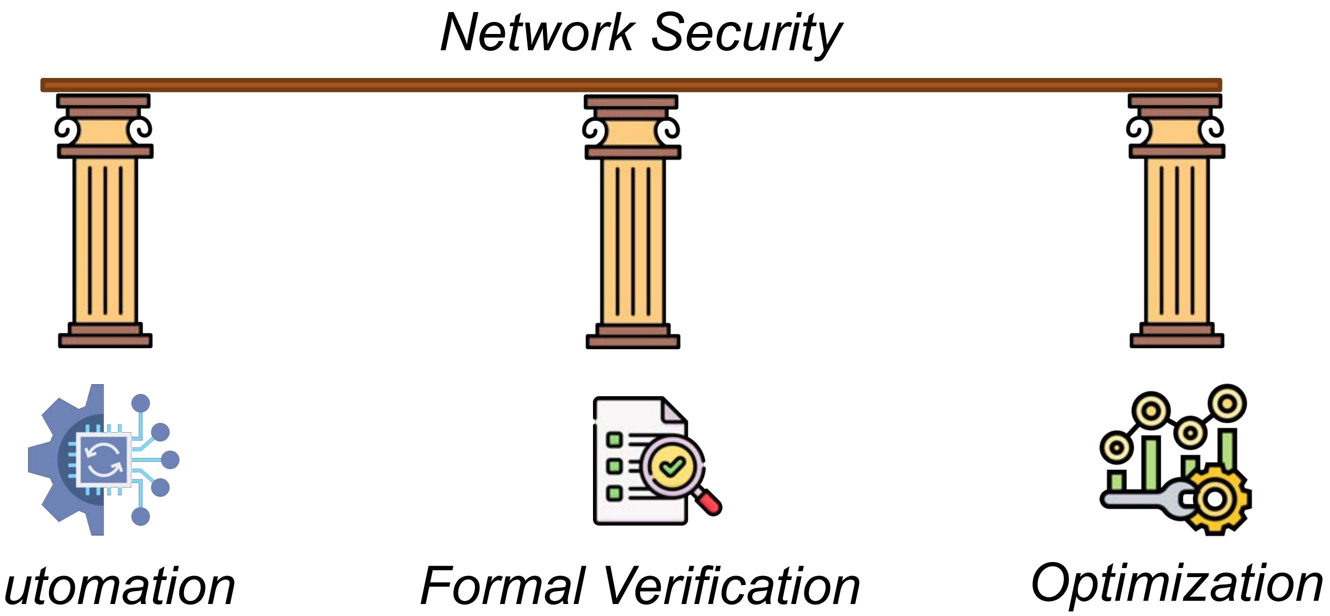
State of the Art of Automatic Security Configuration before VEREFOO

- In literature, there were several **limitations** preventing to satisfy our wish list:
  - service composition and NSF configuration were **rarely addressed together** (e.g., *Schnepf et al.* and *Basile et al.* combine them, but only for simple function chains);
  - most state-of-the-art methodologies **did not** pair automation with security **optimization** and formal **verification** at the same time
    - *García-Alfaro et al.* and *Ocampo et al.* address only networking optimization
    - *Adi et al.* and *Gember-Jacobson et al. address only* formal verification
  - **scalability** was **limited to tens of NSFs**

The VEREFOO Approach

The New Approach: VEREFOO

- The VEREFOO goal is to combine **Security Automation, Formal Verification, Optimization** in a unified approach to **simultaneously**:
  1. define the optimal **allocation** scheme for NSFs;
  2. generate the optimal **configuration** of the allocated NSFs.



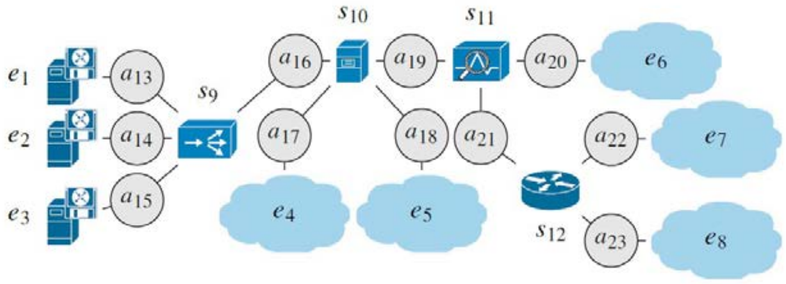
VEREFOO

***VE**rified **REF**inement*
*and **O**ptimized **O**rchestration*

The VEREFOO Approach

Input

Virtual Network Topology

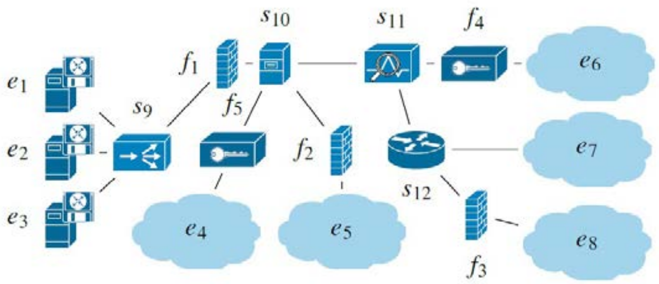


Network Security Policies

VEREFOO

Output

NSFs allocation scheme



NSFs configurations

MaxSMT Problem Formulation

Partial weighted MaxSMT: an optimization-enhanced version of the Satisfiability Modulo Theories (SMT) problem.

- **Partial:** there are **hard** constraints that must be always satisfied and **soft** constraints that should be satisfied *as far as possible*.
- **Weighted:** find an assignment of the variables that maximizes the total weight of the satisfied **soft** clauses.

$H = \{x_1 + x_2 > 5 \land A_1, x_1 < 4 \lor x_2 < 5 \lor \neg A_2\}$

$S = \{(x_1 > 1, 5), (\neg A_2, 10), (x_1 + x_2 > 6, 8)\}$

x_1	x_2	A_1	A_2	sum of weights
2	3	true	false	15
2	4	true	true	13
2	5	true	false	23
…				

MaxSMT Features

The **MaxSMT** formulation allows achieving:

- *Automation*: an automated solver searches for the solution.
- *Formal correctness guarantee*: MaxSMT guarantees "**correctness by construction**", provided the correctness conditions are expressed as hard constraints;
- *Optimization*: soft constraints can express optimality objectives.

MaxSMT Features

The **MaxSMT** formulation allows achieving:

- *Automation*: an automated solver searches for the solution.
- *Formal correctness guarantee*: MaxSMT guarantees "**correctness by construction**", provided the correctness conditions are expressed as hard constraints;
- *Optimization*: soft constraints can express optimality objectives.

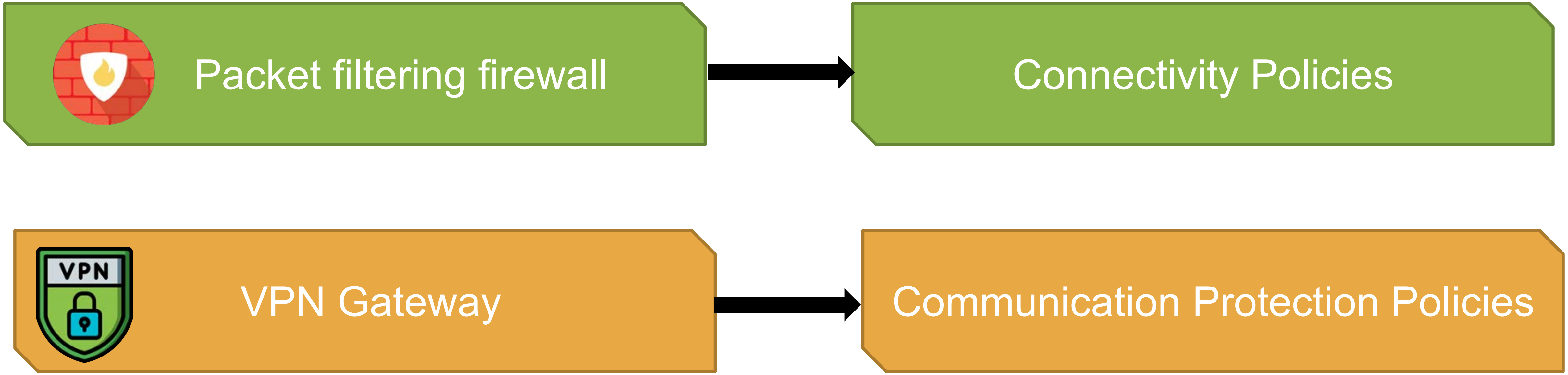
How to use this approach for security configuration?

What about **performance** and **scalability?**

They can be achieved only by careful (lightweight) modeling!

VEREFOO Application to two NSFs

- The VEREFOO approach has been used for the automatic security configuration of two NSFs that are frequently used in computer networks:



VEREFOO Application to two NSFs

- The VEREFOO approach has been used for the automatic security configuration of two NSFs that are frequently used in computer networks:

Packet filtering firewall → Connectivity Policies

VPN Gateway → Communication Protection Policies

Modeling Choices for Packet-filtering Firewall Configuration

- Packet classes (traffic) : predicates over IP 5-tuple fields
- Packet flows : $f = [n_s, t_{s,a}, n_a, t_{a,b}, n_b, \ldots, n_k, t_{k,d}, n_d]$
- Network function behavior:
  - forwarding : $\boldsymbol{deny_i}$(t)
  - transformation : $\boldsymbol{\mathcal{T}_i}$(t)
- Packet filtering configuration decisions modeled as free variables
- Static computation of all (relevant) possible flows before using MaxSMT

Medium-level Connectivity NSRs

- The high-level Policies are translated into medium-level **Network Security Requirements** (NSRs):

Network Security Requirements

Type	IPSrc	IPDst	pSrc	pDst	tProto
Isol	192.168.0.2	220.226.50.*	*	*	*
Isol	192.168.0.3	220.226.50.*	*	*	*
Isol	130.192.225.*	220.226.50.3	*	*	*
Reach	130.192.225.*	220.226.50.2	*	80	TCP
Isol	130.192.225.*	220.226.50.2	*	≠80	TCP
Isol	130.192.225.*	220.226.50.2	*	*	UDP
Reach	220.226.50.2	130.192.225.*	*	*	*
Isol	130.192.120.*	220.226.50.2	*	*	*
Reach	130.192.120.*	220.226.50.3	*	110	TCP
Isol	130.192.120.*	220.226.50.3	*	≠110	TCP
Isol	130.192.120.*	220.226.50.3	*	*	UDP
Reach	220.226.50.3	130.192.120.*	*	*	*
Isol	130.192.120.*	130.192.225.*	*	*	*

- Each NSR $r \in R_s$ is modeled as

  r = (type, IPSrc, IPDst, pSrc, pDst, tProto)

- The *type* of each NSR can be:
  1) *reachability*, if the NSR requires that the packets satisfying the conditions can reach their destination;
  2) *isolation*, if the NSR requires that the packets satisfying the conditions must not reach their destination.

MaxSMT Problem Formulation

MaxSMT Problem

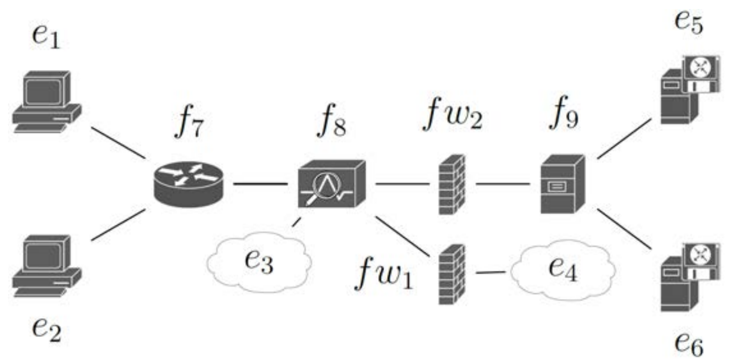
Hard Constraints

- **forwarding behavior** of the functions
- **enforcement** of the **NSRs**

Soft Constraints

- optimal firewall **allocation** scheme
- optimal firewall **configuration** rule set

Outcome of VEREFOO



Firewall fw_1

#	Action	IPSrc	IPDst	pSrc	pDst	tProto
1	Allow	220.226.50.3	130.192.120.*	*	*	*
2	Allow	130.192.120.*	220.226.50.3	*	110	TCP
D	Deny	*.*.*.*	*.*.*.*	*	*	*

Firewall fw_2

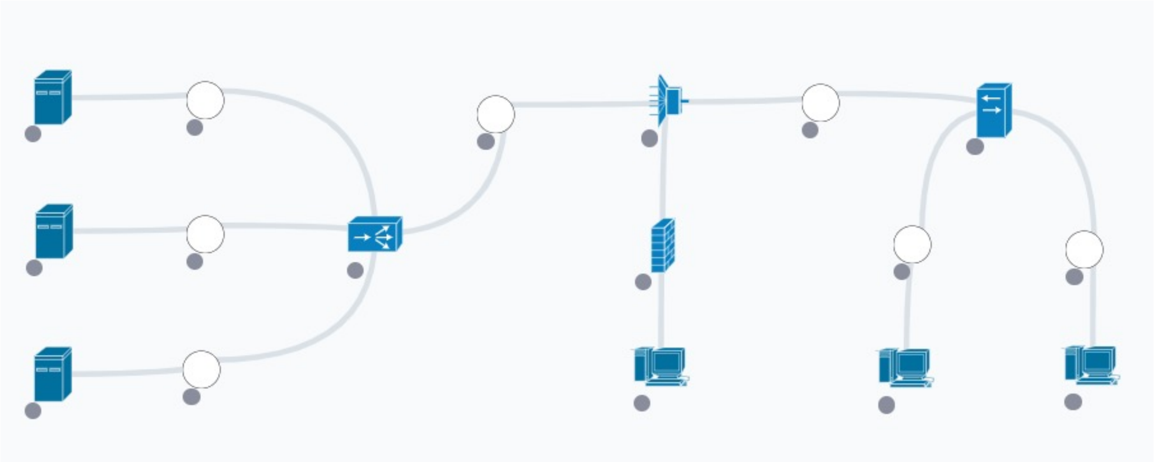
#	Action	IPSrc	IPDst	pSrc	pDst	tProto
1	Allow	130.192.225.*	220.226.50.2	*	80	TCP
2	Allow	130.192.120.*	220.226.50.3	*	*	*
3	Allow	220.226.50.*	130.192.*.*	*	*	*
D	Deny	*.*.*.*	*.*.*.*	*	*	*

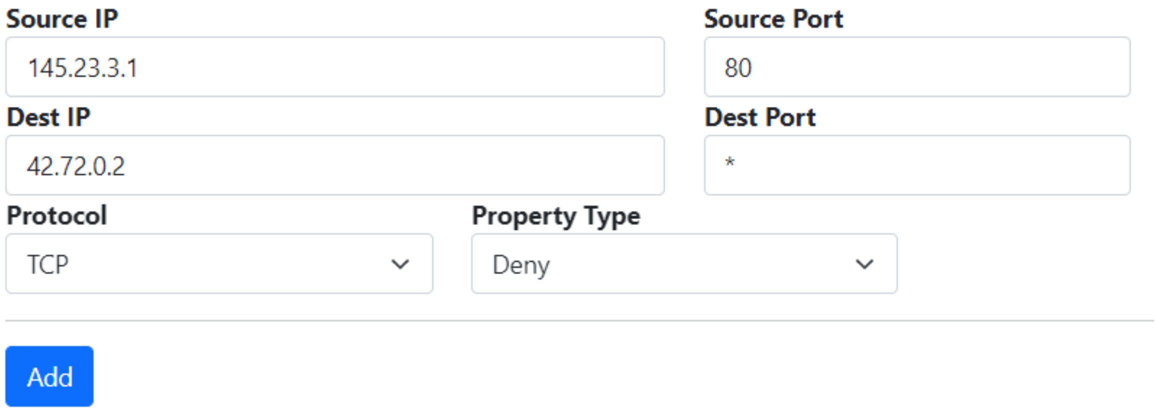
- From the hard and soft constraints, the MaxSMT solver finds a solution with:
  - the minimum number of allocated firewalls
  - the minimum number of rules in each allocated firewall

VEREFOO Implementation

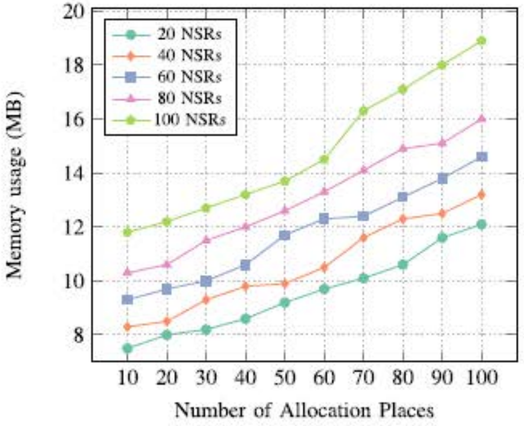
Implementation:

- Java-based framework
- z3 as MaxSMT solver
- REST APIs and GUI for interaction
- GitHub repository: https://github.com/netgroup-polito/verefoo/

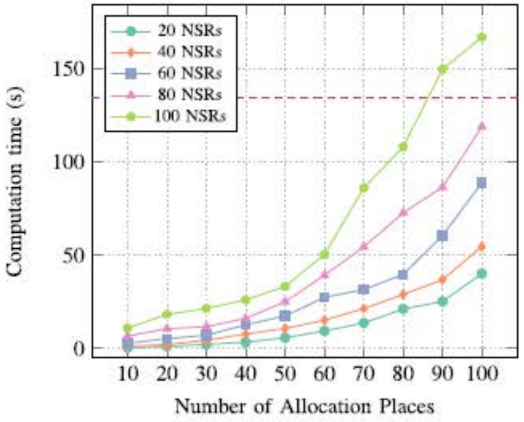




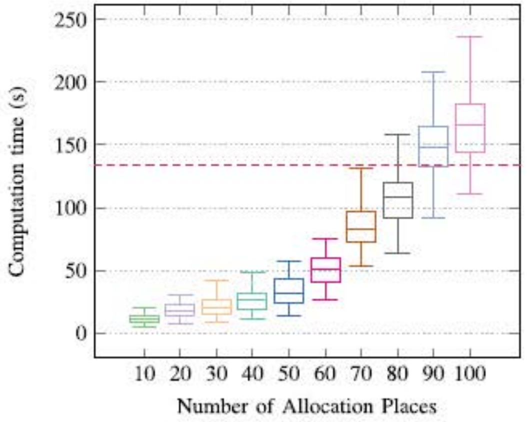
VEREFOO Evaluation (Firewalls)



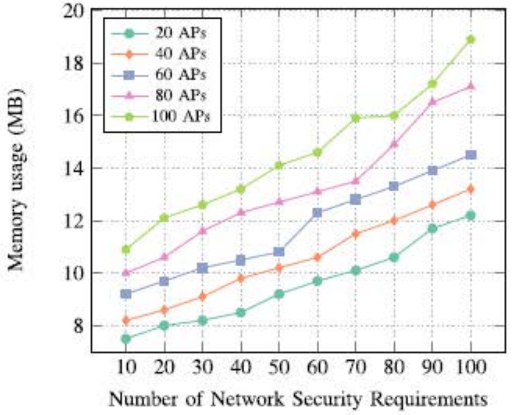
(a) Memory usage chart



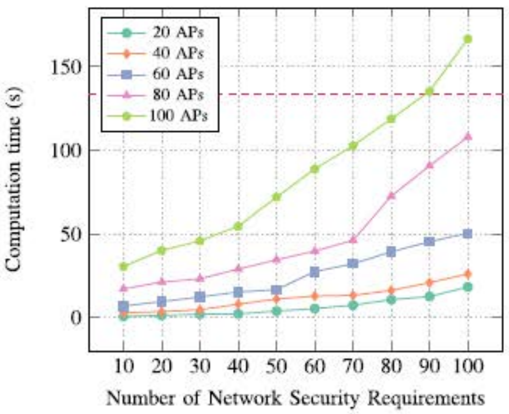
(b) Computation time chart



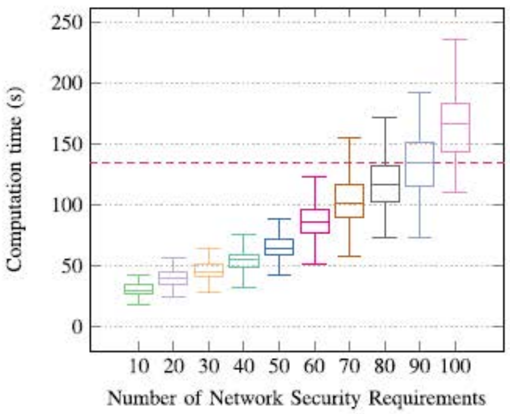
(c) Whisker plot for computation time



(a) Memory usage chart

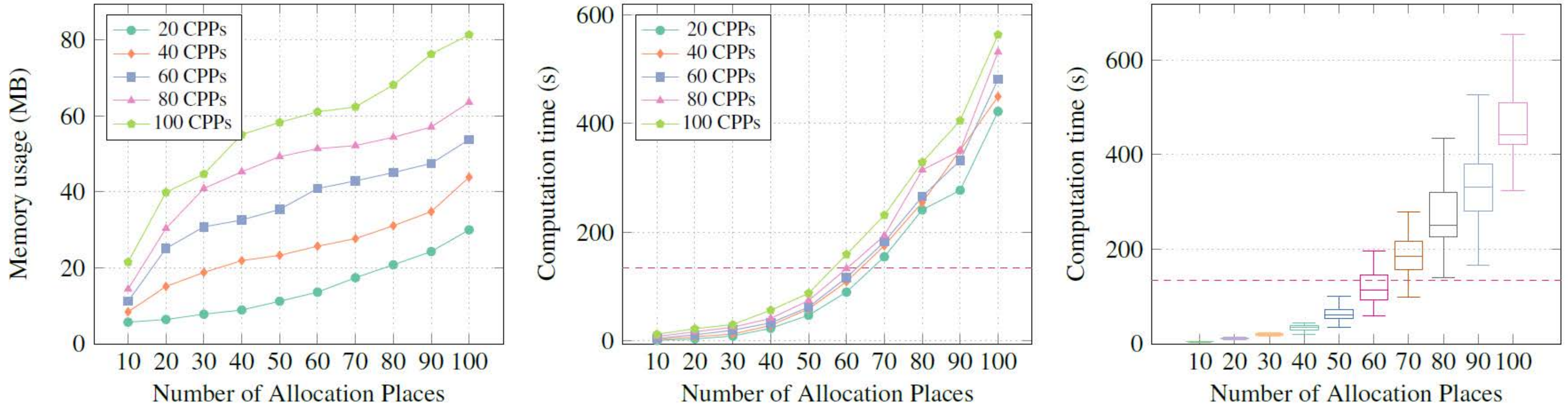


(b) Computation time chart

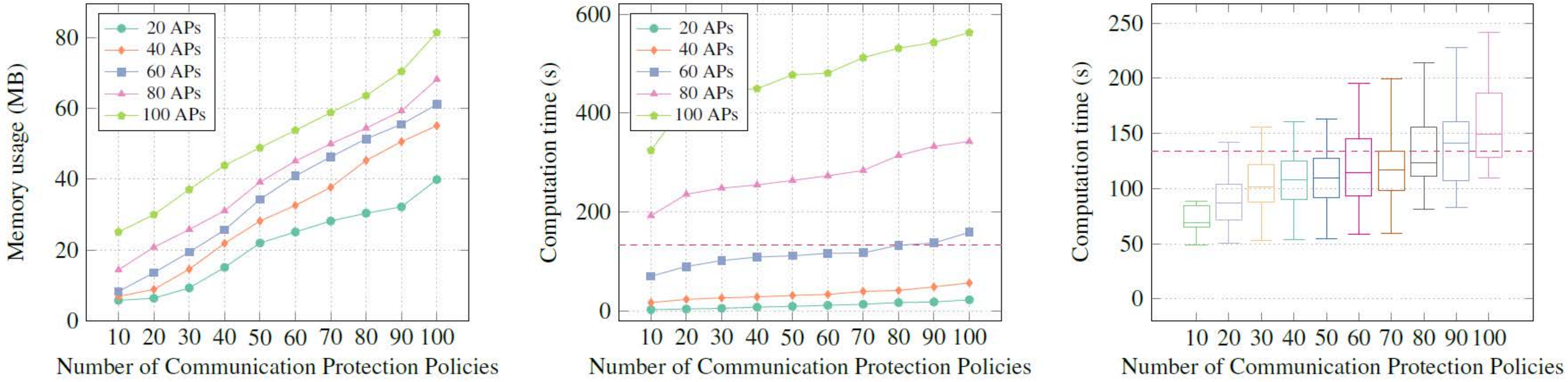


(c) Whisker plot for computation time

VEREFOO Evaluation (VPN Gateways)



(a) Memory usage chart (b) Computation time chart (c) Whisker plot



(a) Memory usage chart (b) Computation time chart (c) Whisker plot

VEREFOO Integration with Docker Compose

- VEREFOO has been integrated with **Docker Compose**:
  - it is an **automated** deployment environment;
  - it allows **maximizing** efficiency, minimizing resource consumption for the NSF deployment, and keeping only the minimal network functionalities;
  - It supports **three** types of packet filtering firewalls (iptables, BPF-based firewalls, Open vSwitch).

- The network topology output by VEREFOO is automatically **translated** into a set of containers.

THE 24TH INTERNATIONAL CONFERENCE ON
INFORMATION AND COMMUNICATIONS SECURITY (ICICS 2022)
University of Kent, Canterbury, UK, 5-8 September, 2022

ICICS Canterbury, UK 2022

BEST DEMO AWARD

VEREFOO: AN AUTOMATED FRAMEWORK FOR VIRTUAL FIREWALL CONFIGURATION

Daniele Bringhenti, Guido Marchetto, Riccardo Sisto and Fulvio Valenza

Shujun Li
General Co-Chair

Sponsored by
University of Kent | Institute of Cyber Security for Society (iCSS)

Keenan Jones
Local Award Judging Chair

VEREFOO Applications in IoT Networks

- VEREFOO has been applied to IoT networks in two scenarios:

1) Demonstration in the EU **CyberSec4Europe** Project:
   - application to a platform for urban management in a **smart city**;
   - to solve the firewall configuration problem under complex **interactions** between human users and heterogeneous services.

2) Integration with the **ONOS** Orchestrator:
   - application to IoT-aware SDN networks;
   - to solve **SDN switch** configuration problem while interacting with probes that may detect cyber-attacks.

Cyber Security for Europe

Open Network Operating System

Papers about the VEREFOO Approach

D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "**Towards a fully automated and optimized network security functions orchestration**", in **IEEE ICCCS 2019**, Rome, Italy, October 10-12, 2019.

D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "**Automated optimal firewall orchestration and configuration in virtualized networks**", in **IEEE/IFIP NOMS 2020**, Budapest, Hungary, April 20-24, 2020.

D. Bringhenti, G. Marchetto, R. Sisto, and F. Valenza, "**Short Paper: Automatic Configuration for an Optimal Channel Protection in Virtualized Networks**", in **CYSARM 2020**, co-located with **ACM CCS 2020**, Orlando, United States, November 9-13, 2020.

D. Bringhenti, G. Marchetto, R. Sisto, S.Spinoso, F. Valenza, and J. Yusupov, "**Improving the formal verification of reachability policies in virtualized networks**", in **IEEE Transactions on Network and Service Management**, March 2021.

D. Bringhenti, J. Yusupov, A.M. Zarca, F. Valenza, R.Sisto, J.B.Bernabe, and A.Skarmeta, "**Autonomic, Verifiable and Optimized Policy-based Security Enforcement for SDN-aware IoT networks**", in **Computer Networks, Elsevier**, August 2022.

D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "**Automated firewall orchestration configuration in virtual networks**", in **IEEE Transactions on Dependable and Secure Computing**, March 2023.

D. Bringhenti, R. Sisto, and F. Valenza, "**A novel abstraction for security configuration in virtual networks**", in **Computer Networks, Elsevier**, June 2023.

Latest VEREFOO Developments

Latest VEREFOO Developments

- Research about the VEREFOO approach is still on-going.
- Main research directions:

1) Improve performance and scalability
    - use of **atomic flows** instead of **maximal flows** for the MaxSMT problem formulation;
    - optimization of **re-configuration**;
    - investigation of **heuristic** approaches.
2) Extend coverage
    - **stateful** NSF configuration;

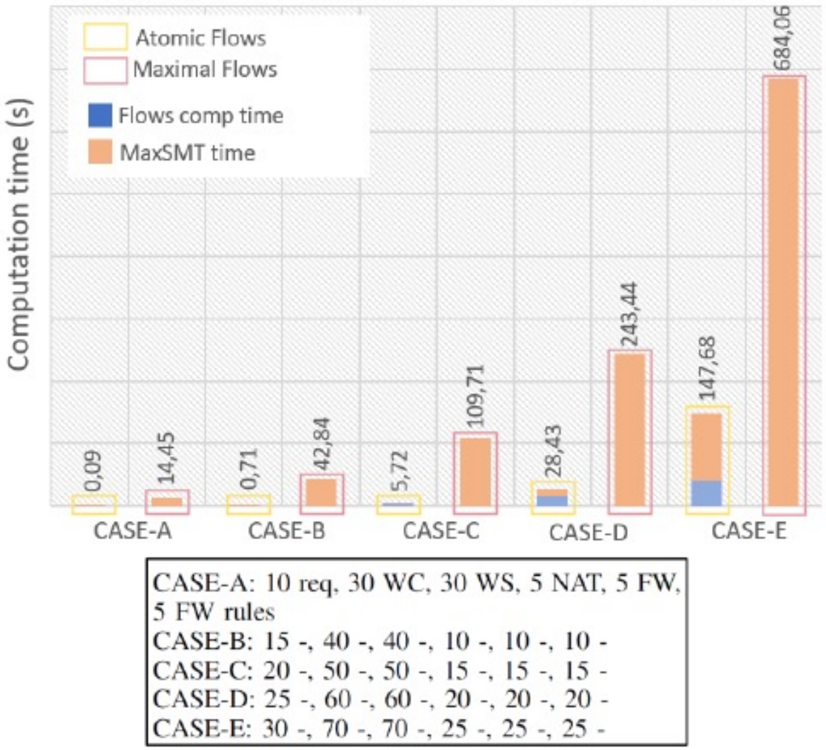
Atomic Flows versus Maximal Flows

- **Maximal Flows : minimize** the number of flows (by **aggregating** together all flows that behave in the same way when crossing the network)
- **Atomic Flows** : use flows corresponding to **disjoint atomic traffic components** (leveraging the theory of atomic predicates by Woo and Lam)

- Maximal Flows: **few** flows but with **complex** predicate representation
- Atomic Flows: **many** flows but with **simple** predicate representation

Comparison between Atomic and Maximal Flows

- Atomic Flows computation is **slower** than Maximal Flows computation (and generates more flows)
- But with Atomic Flows the MaxSMT problem complexity decreases
- the overall computation time decreases
- => better scalability



Optimization of Reconfiguration

- The original VEREFOO algorithm is **not optimized** for NSF **reconfiguration**.
- When policies are updated (e.g. because of attack detection), VEREFOO can only **recompute** the configuration of the whole network **from scratch**, even if the change is just limited to a small portion.
- => reconfiguration from scratch may require **excessive time**, causing a high window of exposure.

The REACT-VEREFOO Idea

REACT-VEREFOO

Fast and efficient reconfiguration approach for distributed firewalls, with the goals of:

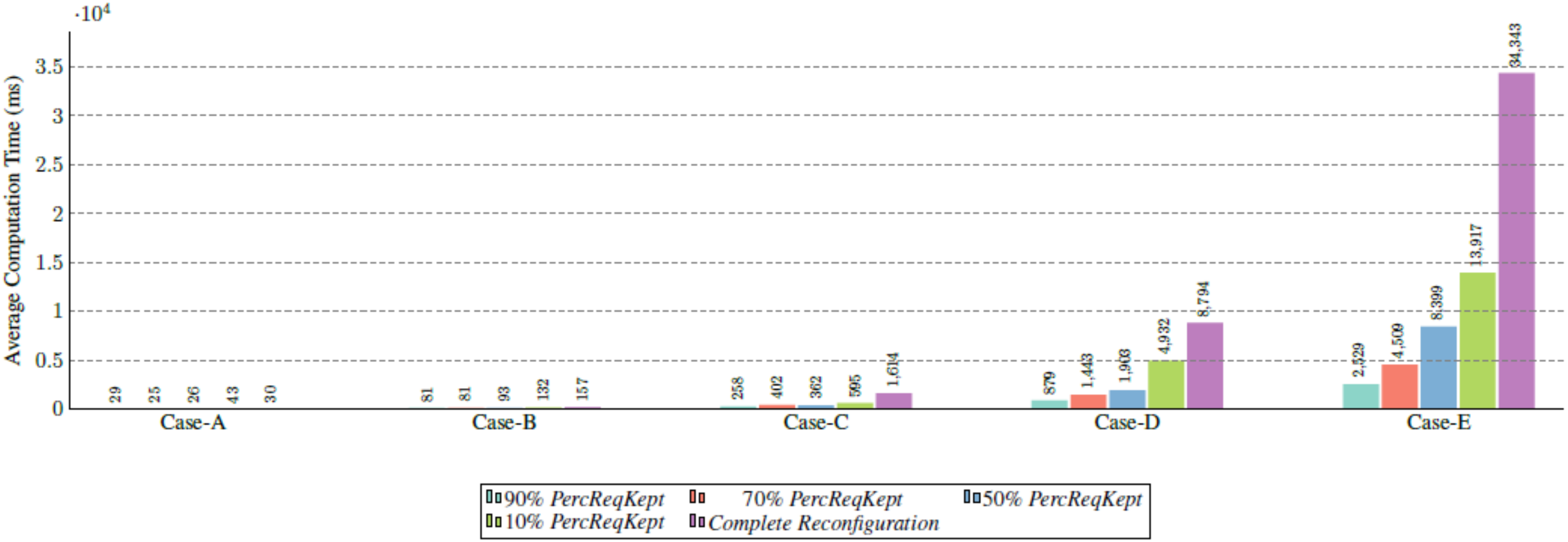
- achieving a new formally assured configuration within a **short computation time**
- satisfying a **new set of Network Security Requirements (NSRs)**

Main changes to the original VEREFOO:

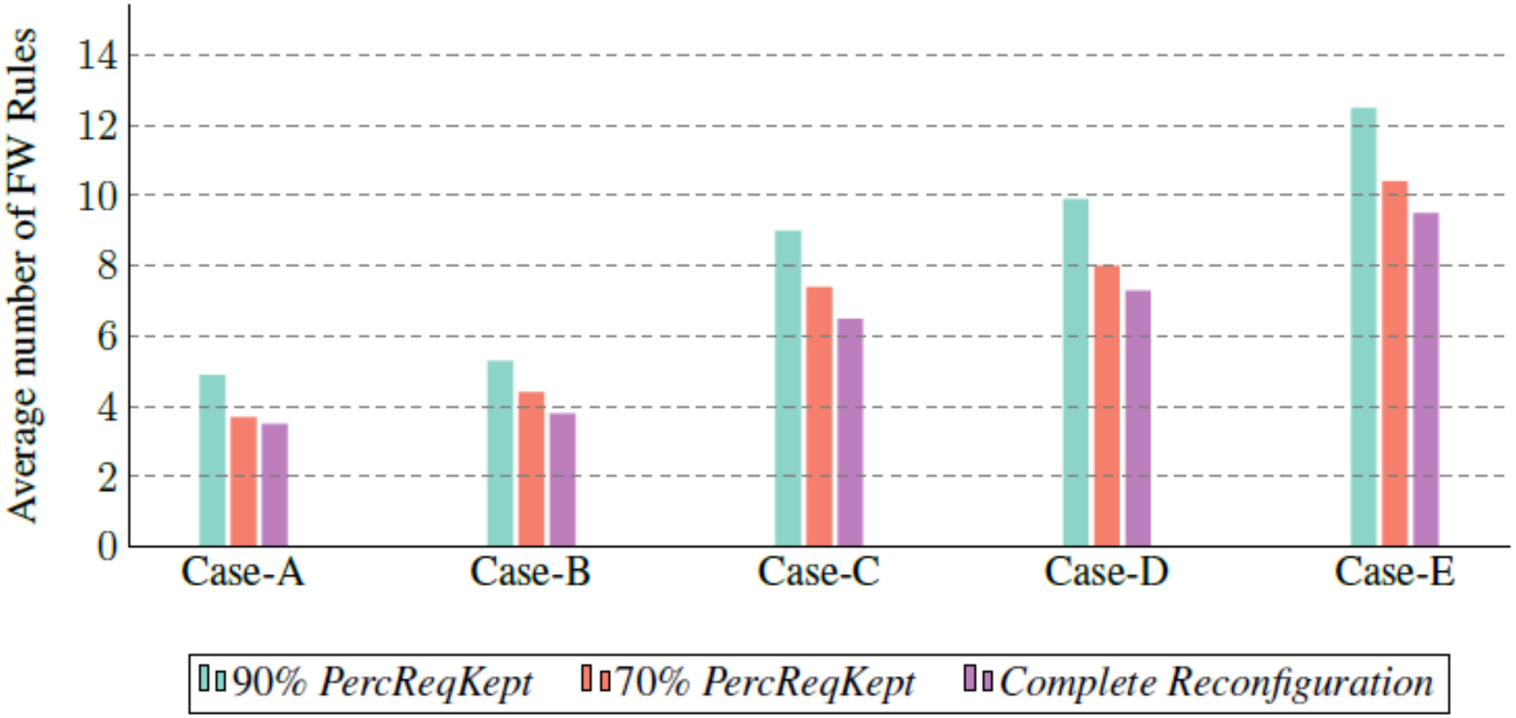
1) Modification of the **inputs**
2) Algorithm to select the **minimum** network area that must be reconfigured,
3) Update of the MaxSMT soft constraints to **minimize the changes** in the configuration and achieve better performance.

F. Pizzato, D. Bringhenti, R. Sisto, and F. Valenza, "**Automatic and optimized firewall reconfiguration**", accepted for publication in **NOMS 2024** - 37th IEEE/IFIP Network Operations and Management Symposium.

REACT-VEREFOO: Preliminary Performance Evaluation



REACT-VEREFOO Preliminary Optimization Evaluation



The sub-optimality of some results is due to two factors:

- the reduction of the solution space for the solver;
- the soft constraints force the preference of reusing old configuration elements even if a completely new configuration would result in a slightly better solution.

Investigation of Heuristic Approaches (Preliminary Results)

- The original VEREFOO algorithm can scale to medium-sized networks…

  …but not to large networks
- We defined an alternative heuristic strategy:
  - formulated as a variant of the **knapsack** problem
  - scales to much larger problems (100x)
  - still providing correctness guarantees (to be proved)

Stateful Functions

- In the original VEREFOO version, only **stateless** functions are modeled.
- However, the most commonly used firewall products (e.g., iptables, ipfw, Open vSwitch, pfsense, nft) are stateful.
- For example, iptables can be configured to save a **local state** when a packet of a certain type is received, and to accept packets related to an **established**/related communication for which there is such state:
  - iptables -A INPUT -p tcp –dport 22 -m conntrack --ctstate NEW -j ACCEPT
  - iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

Conclusions

- We proposed the VEREFOO approach to achieve **automation**, **formal correctness guarantee and optimization** to solve the NSF **allocation and configuration** problem in **service graphs**.
- A main challenge has been the definition of formal models that are sufficiently detailed, but **lightweight** enough to make the automatic resolution algorithm **usable in practice** (extending the state if the art significantly).
- Latest enhancements have made the approach even more scalable.

Riccardo Sisto,
Daniele Bringhenti, Fulvio Valenza

riccardo.sisto@polito.it

daniele.bringhenti@polito.it, fulvio.valenza@polito.it

VEREFOO GitHub Repository:

https://github.com/netgroup-polito/verefoo

Demo: