POLITECNICO
DI TORINO

Master of Science in Computer Engineering

Master Degree Thesis

Automatic Configuration of Secure Communications in Virtualized Networks

Supervisors
prof. Riccardo Sisto
prof. Guido Marchetto
prof. Fulvio Valenza
dott. Daniele Bringhenti

Candidate
Felice Savino

Academic Year 2019-2020

Summary

The *Network Functions Virtualization* (NFV) paradigm is a novel networking technology which decouples the software implementation of network functions from the underlying hardware exploiting virtualization technologies and programmable hardware, such as general-purpose servers and switches. This principle allows the creation of a *Service Graph*, which is a generalization of the *Service Function Chain* (SFC) concept where multiple paths could exist between every pair of end points, with better flexibility and performance.

A problem which, however, arises in the creation of a Service Graph and in the configuration of the virtual network functions composing it is that these operations are preformed manually. These are very sensitive tasks, especially when the network functions to be allocated and configured are security functions, hence functions with the intent of providing some security services such as communication protection or deny/allow certain network traffic. A wrong configuration of these type of functions can easily lead to severe and dangerous incidents such as security breaches. Moreover, the fact that these operations are performed by human beings entails an intrinsic slowness in the reaction speed whenever the security defences need to be updated according to different or additional security requirements.

In view of these considerations, *Network Automation* represents a valid alternative due to the fact that automatizing Network and Security Management the configuration changes can be managed automatically with a slower latency and without the risk of human misconfigurations. This principle is based on the capability of react both when a new requirement is introduced in the system and when an event arises from the network in order to change the configuration of the network itself.

An example of framework which proposes the aforementioned approach to overcome the intrinsic limit of a traditional Service Function Chain is represented by VEREFOO. It stands for VErified REfinement and Optimized Orchestration and it is a framework whose main goals are the optimal automatic allocation and configuration of Network Security Functions on a Service Graph in order to satisfy a set of network security requirements expressed by the security administrator by means of an high-level security language through a refinement process.

The major contribution provided by this thesis work has been the extension of the functionality of the ADP module of VEREFOO to make it capable of automatically allocating and/or automatically configuring Network Security functions called Channel Protection Systems (CPSs), respecting some optimality criteria, in order to satisfy a set of Network Security Requirements related to the enforcement

of secure communications. This has been obtained through the resolution of a MaxSMT problem, whose objective is to maximize the sum of the weights assigned to the satisfied soft clauses, with respect to hard constraints that always require to be satisfied ; the MaxSMT problem is formulated so as to provide also a formal verification that the achieved solution is formally correct. The secure communication are enforced through the creation of end-to-end, site-to-site and secure gateway Virtual Private Networks. Furthermore the reachability assurance of the packets belonging to a secure communication flowing from the source to the destination when a packet filter is present on the traversed path has been granted. This situation is very common and represents a not negligible problem because even if a secure communication is correctly enforced a packet filter on the path, depending on its configuration, could drop the packets belonging to that secure communication making impossible for the destination receiving the related packets and generating the so called filtered channel anomaly.

After all these works have been concluded, a series of performance tests have been carried out to evaluate the impact the number of communication security requirements and the number of allocation places have on the system scalability; the result which has been achieved is that the framework is capable of handle with a large set of requirements and with Allocation Graphs having a high number of allocation places in a reasonable time.

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Thesis objective

In the recent years two novel networking technologies have emerged: Network Functions Virtualization (NFV) and Software-Defined Networks (SDN). Network Function Virtualization can be defined as the capability to run any network function on a standard hardware, possibly with the help of computing virtualization to achieve an efficient use of resources, allowing a flexible deployment of network functions composing a Service Function Chain; the second technology allows the creation of the paths which the packets must follow inside the physical network by means of a software process. Before these paradigms, a service graph, which is a generalization of a service function chain where multiple paths could exist between every pair of end points, was essentially composed by specific hardware devices build ah hoc to implement specific services such as NAT or proxy server; nowadays,instead, is always more frequent virtualize the network functions composing the graph because this allows the use of hardware resources in a very efficient way due to the fact that more than one service function can be installed on the same server sharing the physical resources, moreover to add a new function is no more necessary buy expensive dedicate hardware, but it can simply installed on the already available general purpose servers. These concepts can be applied not only to network service functions such as the Load Balancer and the Network Address Translator (NAT), but also to Network Security Functions (NSFs) like the packet filter firewall and functions able to create secure communication like the VPN Gateway.

A problem which, however, arises in the creation of a Service Graph and in the configuration of the virtual network functions composing it is that these operations are preformed manually. These are very sensitive tasks, especially when the network functions to be allocated and configured are security functions, hence functions with the intent of providing some security services such as communication protection or deny/allow certain network traffic. A wrong configuration of these type of functions can easily lead to severe and dangerous incidents such as security breaches. Moreover, the fact that these operations are performed by human beings entails an intrinsic slowness in the reaction speed whenever the security defences need to be updated according to different or additional security requirements. In view of these considerations, *Network Automation* represents a valid alternative due

to the fact that automatizing Network and Security Management the configuration changes can be managed automatically with a slower latency and without the risk of human misconfigurations.

The thesis objective has been to contribute to the design and implementation of VEREFOO which stands for VErified REfinement and Optimized Orchestration and it is a framework whose main goals are the optimal automatic allocation and configuration of Network Security Functions on a Service Graph in order to satisfy a set of network security requirements expressed by the security administrator by means of an high-level security language through a refinement process. In particular this thesis has dealt with the extension of the functionality of the ADP module of VEREFOO enabling it to automatically allocate and/or automatically configure Network Security functions called Channel Protection Systems (CPSs), respecting some optimality criteria, in order to satisfy a set of Network Security Requirements related to the enforcement of secure communications.

1.2 Thesis description

The rest of the thesis is organized in the following way:

- **Chapter 2** is opened introducing the Service Function Chain concept i.e. a chain made up by several network functions aimed at providing services to a certain communication. Then, two important novelties in the ambit of computer networks are introduced: Software Defined Network and Network Function Virtualization highlighting the advantages that these novelties bring to a SFC. The last section is dedicated to the concept of automatizing the networks and security management thanks to the novelties described above and a new improved SFC, reporting all the advantages in enforcing automatic configurations and management.

- **Chapter 3** is entirely dedicated to the problem of configuring Secure communications. Firstly the policy based approach is introduced describing how does it works and the importance of configuring network security functions through a policy refinement process instead of entrusting it completely to the security administrator. Then the focus is moved on the issues related to the manual configuration of the network security functions in charge of enforcing secure communications. Finally the works already present in literature related to the automatic configuration of security functions able to enforce secure communications are analyzed highlighting their limitations.

- In the **Chapter 4** is described the approach hold to develop this work. In the first section the objective of the thesis is presented and explained dividing it into a set of smaller goals to accomplish in order to satisfy the overall purpose. In the second section more details about the ADP module are provided since it is the module of VEREFOO this thesis gave a contribution in the implementation. Finally, the last section deals about how the approach is based on a formulation of the methodology through a partial weighted MaxSMT problem, describing the principles which it is based on, emphasizing

the difference between hard and soft constraints and how these have been exploited to model different aspects of the ADP module.

- The first part of **Chapter 5** deals with the modeling of the network nodes and links composing a Service Graph or an Allocation Graph. Essentially 4 different category are defined: untrusted nodes, inspector nodes, trusted nodes, and untrusted links illustrating, for each one of them, the advantages of creating these categories. The second part focuses on the model of the communication protection requirements, i.e. those requirements that consent to specify the enforcement of a secure communication, explaining the meaning of each field belonging to it and presenting the formulations which consent the system to verify that the security requirements inserted in input are satisfied by the solution provided by the solver.

- The first 2 sections of **Chapter 6** centers on the channel protection functions, i.e. those network security functions which consent to create a secure communication, describing them in a detailed way starting from their behavior to their configuration exploiting a medium level policy language. The third section introduces the concepts of technology and capability, after defining them, the relation between these two concepts and the Channel Protection Systems is explained. The fourth section deals with the possibility of specifying different profiles to express the preference, from the security administrator, about which of the two type of CPSs should be used to establish secure channels and others details related to their configurations in order to reach an optimal solution. The last section focuses on the reachability assurance of the packets when a Packet Filter is positioned on the path between the source and the destination of a secure channel and as consequence, depending on its configuration, could generate a filtered channel anomaly.

- In the first section of **Chapter 7** are provided details about the implementation of the ADP module, and the solver used to resolve the maxSMT problem. The section two is characterized by two use Cases with the goal of showing through practical examples all the features and characteristics of the ADP module presented in the previous chapters. Finally, In the last part a series of performance tests are presented to evaluate the scalability of the implemented solution.

- **Chapter 8** is dedicated to the conclusions and the future works.

Chapter 2

Network Automation

This chapter is opened introducing the Service Function Chain concept i.e. a chain made up by several network functions aimed at providing services to a certain communication. After describing how it works the focus is moved on its limits highlighting, in particular, the lack of flexibility and agility due to the network architecture principles which this concept lays on.

Therefore two important novelties in the ambit of computer networks, which are able to overcome these limits, are introduced: Software Defined Network and Network Function Virtualization. The first one allows to dynamically decide the path a packet must follow thanks to the use of the software in the forwarding process, making the SFC agiler and simpler to manage. The second is based on the concept of using standard hardware to run network function images in order to make the SFC much more flexible.

The last section is dedicated to the concept of automatizing the networks and security management thanks to the novelties described above and a new improved SFC reporting all the advantages in enforcing automatic configurations and management, in particular in the security filed where the reaction speed is essential when there is the need to cope a cyberattack.; then an example of framework which implement the presented approach is introduced describing its workflow.

2.1 Service Function Chain

The service designer in order to implement an end-to-end service need to select a proper set of function, and ensure that the traffic flows through them in a certain predetermined order to achieve the pre-established user requests. A formal definition of both Service Function (SF) and Service Function Chain (SFC) has been presented in the RFC 7665 [2] :

> **Service Function** A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). As a logical component, a service function can be realized as a virtual element or be embedded in a physical network element. One or more Service Functions can be embedded

in the same network element. Multiple occurrences of the service function can exist in the same administrative domain.

Service Function Chain A service function chain defines an ordered set of abstract service functions and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification.

In the following an example of end-to-end service implemented using a service function chain is presented. The picture shows a communication between a Web Client and a Web Server where three different functions are positioned on the path: a Firewall which is able to drop network traffic according to its configuration, an Intrusion Detection System which is able to detect attacks by monitoring the network and recognizing some known pattern or comparing it with a previous monitoring phase and finally a Reverse Proxy whose main purposes are hiding of the existence and characteristics of origin servers, web caching of static contents and spoon-feeding. It is evident that in order to guarantee a correct working of the end-to-end service the position of the network functions in the chain must be fixed, so that the packets flow through them in a specific and predetermined order.

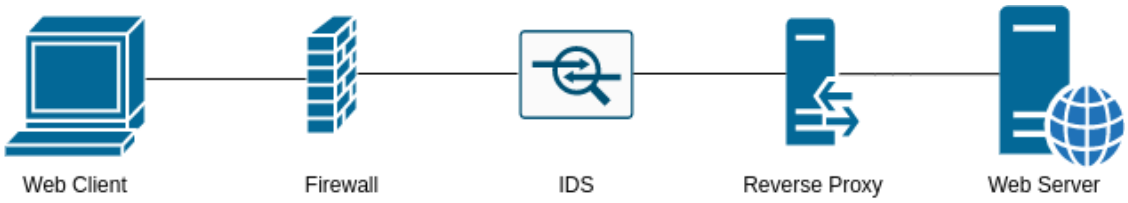


Figure 2.1. Example of Service Function Chain

In the past the network functions were essentially physical box dedicated to a specific purpose, hence, designed and built to implement the corresponding function; as consequence the network function chains made up by this kind of functions suffer of strong limitations:

- Hardware resources are not used at best; some appliances may sustain an heavy load, while other may be almost unloaded and it is not possible to share the available hardware resources between different services, since each one of them is a different physical box with his dedicated hardware;

- Service Disruption when modifying the service chain. Each time a function in the chain need to be added or removed the service is disrupted, because it is necessary to unplug the physical cables, add/remove the function, re-plugin the cables;

- Every time new functions need to be added to a service function chain dedicated hardware must be purchased for each one of them and their installation is, usually, a very time consuming task;

- Difficult to differentiate services among users. Usually different users want to different services, but a SFC does not have the necessary flexibility to handle with this situation. Considering the example shown in the figure 1, what if an user would like to benefit of the complete end-to-end service and another one would like to exploit the same service without the firewall ? the firewall, for instance, should support explicit configuration of the user privilege i.e. per application configuration; however it is not trivial manage this situations and not always is possible.

In the recent years new paradigms have been exploited in order to overcome these critical issues and implement a service function chain with improved flexibility and performance.

2.2 Software Defined Network and Network Function Virtualization

From the architectural/infrastructural point of view the network world has been always the same for over 30 years, novelties like Network Function Virtualization and Software Defined Network allowed the networks to change deeply both architecturally and behaviorally, opening to new solutions for a lot of network and security issues.

2.2.1 Definition and characteristics of Software-Defined Networks

Software defined Networks are networks i.e. a set of network devices, physical or virtual, whose packets forwarding on the network are decided by software, hence the software is able to influence the paths on the network. The software reacts in real time making this a dynamic process which in the face of certain events, depending on the setting, could change the packets forwarding.

The SDN are based on three concepts:

1 decoupling between the data plane, and the control plane. Control plane refers to the all functions and processes that determine which path to use to send the packet or frame while Data plane refers to all the functions and processes that forward packets/frames from one interface to another based on control plane logic.

2 Centralized control plane which is the place where all the intelligence of this technology is situated. Could be a control plane logically centralized or a control plane physically centralized. It is worth to notice that the firs option is better in order to avoid the presence of a single point of failure and scalability issues. Considering a set of routers thanks to this principles would be possible to turn on the OSPF algorithm on all the routers at the same time acting only on the centralized entity.

3 Definition of southbound and northbound interfaces. The first interface allows the SDN controller to communicate with the forwarding infrastructure while the second interface allow the SDN controller to interact with the user-level application or with higher level controllers.

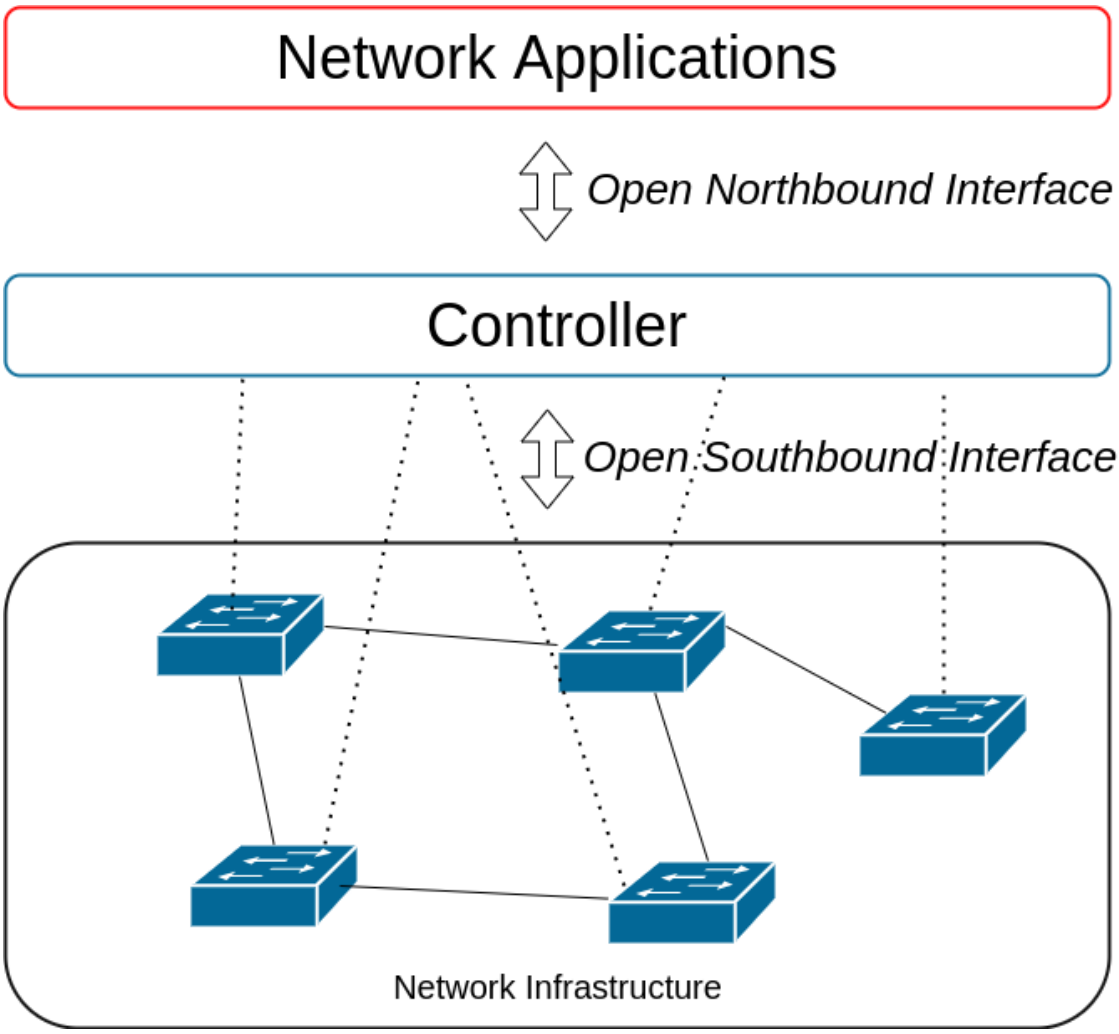


Figure 2.2. SDN Architecture

The figure depicted above present a typical SDN architecture which fully reflects the principles described above. In fact at the lowest level it is possible to notice that the network infrastructure is made up by data forwarding elements which no more require to be complex routers or specific vendor devices, but they can be white-label switches; they are elementary devices because their only task is to forward the input packets, as fast as possible since efficient forwarding is one of the main goal in the SDN approach, to the correct output ports in order to allow them to reach their final destinations. The forwarding table is the element that characterizes each switch, it contains a set of rules whose fields can be divided in two categories: the match fields, which consent to identify a sub set of packet which coincide with them and the action fields, in which are contained the operations the switch has to

perform on those packets (e.g. forward to a specific port, forward to the controller platform, forward at a specific rate, drop the packet, push or pop a field, overwrite or modify a header field). It has been possible to drastically reduce the complexity of the network devices thanks to the decoupling of the data plane and the control plane. Hence all the complexity, such as the computing of the forwarding tables, has been moved in the controller. The controller is able to interact with the network infrastructure thanks to an open south bound interface exploiting different network protocol. The first proposed protocol was Open Flow which is able to perform two principal operation: the injection of the forwarding rules, computed in the upper level, in the network devices and gather the data related to the network infrastructure and push them on the upper level for monitoring and statistics reasons. In order to use this protocol a particular network device called open flow switch need to be used and this, without any doubts, represents the strong limitation that has prevented this protocol to be widely used, because would have meant to substitute all the existing devices with open flow switches. As consequence other protocols have been utilized already supported by the existing network architecture such as Simple Network Management Protocol (SNMP), Network Configuration Protocol (NETCONF) or RESTCONF. Finally, a set of application can be exploited thanks to an open northbound interface, typically REST-based, which consents the controller to communicate with different software modules providing different functionality. The upper layer is fundamental to avoid to extend the operating system of the SDN controller each time a new functionality is required, moreover must be considered that the crash of an application inside the the controller cloud put down the controller itself.

2.2.2 Application of the SDN technology to a SFC

Considering the service function chain concept described in the first section of this chapter, an improvement in his architecture in order to overcome some of the described limitations, supposing that each service function of a Service Function Chain corresponds to a hardware appliance could be link them using a SDN approach in such way that whose forwarding behavior can be completely governed by the SDN controller.

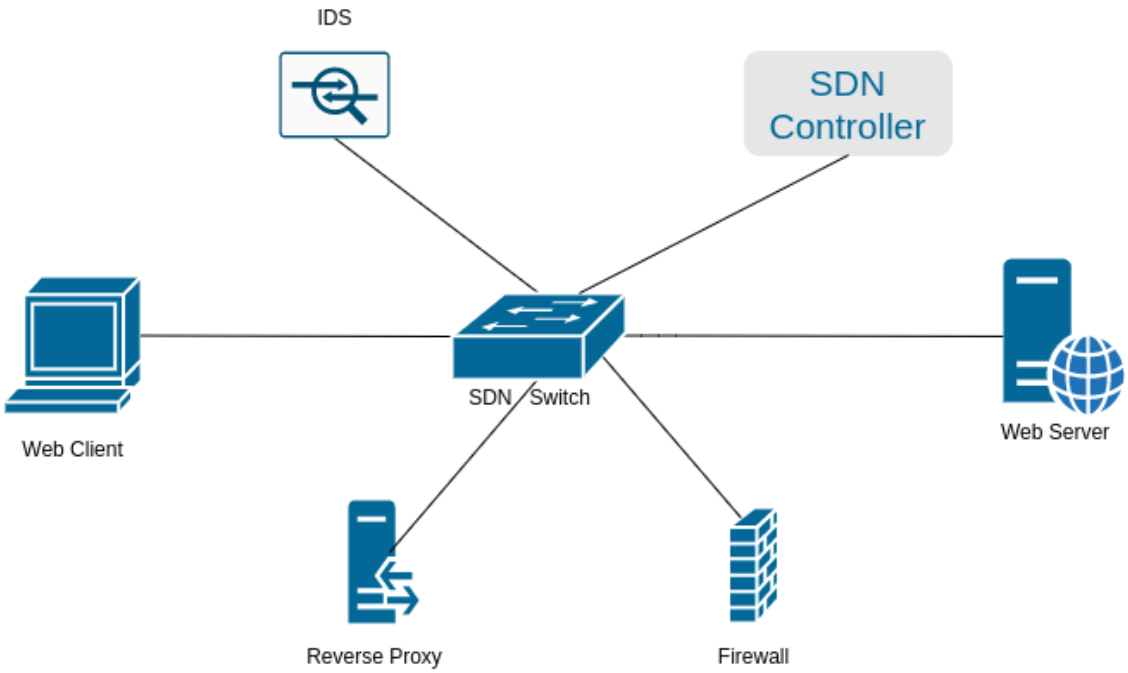


Figure 2.3. Example of Service Function Chain using SDN

Figure 2.3 shows how the Service Function Chain represented in Figure 2.1 can be modeled in an SDN architecture; as explained above the SDN controller is in charge of determining the path which the network functions composing the service graph, connected to the SDN Switch, must follow.

The main advantages of a SFC implemented following this approach are:

- Agility in provisioning new services. After the installation of the box, the routing is done via software instead of connecting the box to the other with physical wires.
- Maintenance and reliability; the cabling is done just one time.
- Different users can have different service chains. Since the routing is done via software it is possible to change its decisions based on other parameters e.g. application layer content.

However it is still difficult partition physical appliance among different tenant and since the appliances are still hardware based, it is not possible to share computational resources among them and installing a new function is a slow operation.

2.2.3 Definition and characteristics of Network Function Virtualization

Network Function Virtualization (NFV) can be defined as the capability to run any network function on a standard hardware, possibly with the help of computing

virtualization to achieve an efficient use of resources. It is characterized by 4 main component:

- Fast standard hardware;
- Software based network function i.e. software image running on standard server;
- Computing virtualization in order to allow a server to host more than one network function;
- Standard API.

An important features related to the NFV is the capacity of auto-scaling of the Virtualized Network Function when extra resources are needed and two different strategies can be adopted:

1 **Scale Up**. Associate more resources to a VNF when it need them (more CPU, more memory, more disk space). Two important aspects must be taken into account when implementing this strategy of scaling. First one it is not always feasible due to the fact that the physical server in which this VNF is running could terminate his resources. Second one assigning more resources does not always improve the situation; for instance if the software is not multi-thread adding n CPUs does not resolve the problem.

2 **Scale Out**. Duplicate the VNF n times in order to obtain more instances of the same VNF granting parallelization and flexibility. The work load of each instances is often supervised by a Load Balancer.

Figure 2.4. Scale up and Scale out examples

Virtual Machines have been the first used technologies to implement virtualized network functions. They provide a fully isolation among the various functions, each one of them running on a different VM, sharing the same server resources. The hyper-visor is the component in charge of coordinating and managing them. The main advantage of this solution is the provided level of security due to the fully isolation, therefore, can be considered the approach which suits best for situation where privacy is fundamental such as multi-tenant data center. The disadvantage is that the amount of resources necessary to create each VM in term of memory and disk space are very high; moreover, the live migration of a Virtual Machine is not a trivial task. Another employed technology to implement VNFs is represented by the

Linux Containers (LXC) whose main goal is lower consume of resources compared to VMs; it is based on the concept of partitioning the kernel of the hypervisor using the namespaces of the Linux world, in order to give the possibility to the created LCX of sharing some kernel parts with the hyper-visor depending on the created namespaces, while all the other parts of the kernel are duplicated and separated. Even if the basic idea is very interesting this technology has not been very used because it is not portable this means that when a LXC is created on a machine it cannot be migrated to another. The main concepts of LXC are exploited by another lightweight virtualization technology called Dockers which nowadays is widespread due to its portability, anyway could not be the best solution for the situations in which security and privacy are mandatory properties due to its loosely isolation.

2.2.4 Application of the NFV technology to a SFC

Although a SFC modeled using SDN is much more flexible because it is able to give agility in provisioning new services modifying the routing via software instead of connecting the boxes to each other with physical cables and modifying it dynamically at any moment, it still suffer of some of the limitations mentioned in the section 1. The Network Function virtualization paradigm helps to overcame these limitations; in fact implementing a NFC using the NFV approach entails :

- Use of hardware resources in a very efficient way due to the fact that more than one service function can be installed on the same server sharing the physical resources.

- To add a new function is no more necessary buy expensive dedicate hardware, but it can simply installed on the already available general purpose servers;

- It is simple partition a service among different tenant creating different instances of a VNF for each one of them.

The figure depicted below represents a service function chain designed using both SDN and NFV principles. Each hardware block has been transformed in a software module capable of being executed on standard servers; moreover it is possible to notice that more network functions can be deployed on the same server thanks to computing virtualization, in fact each VM block represents a possible service of the chain.

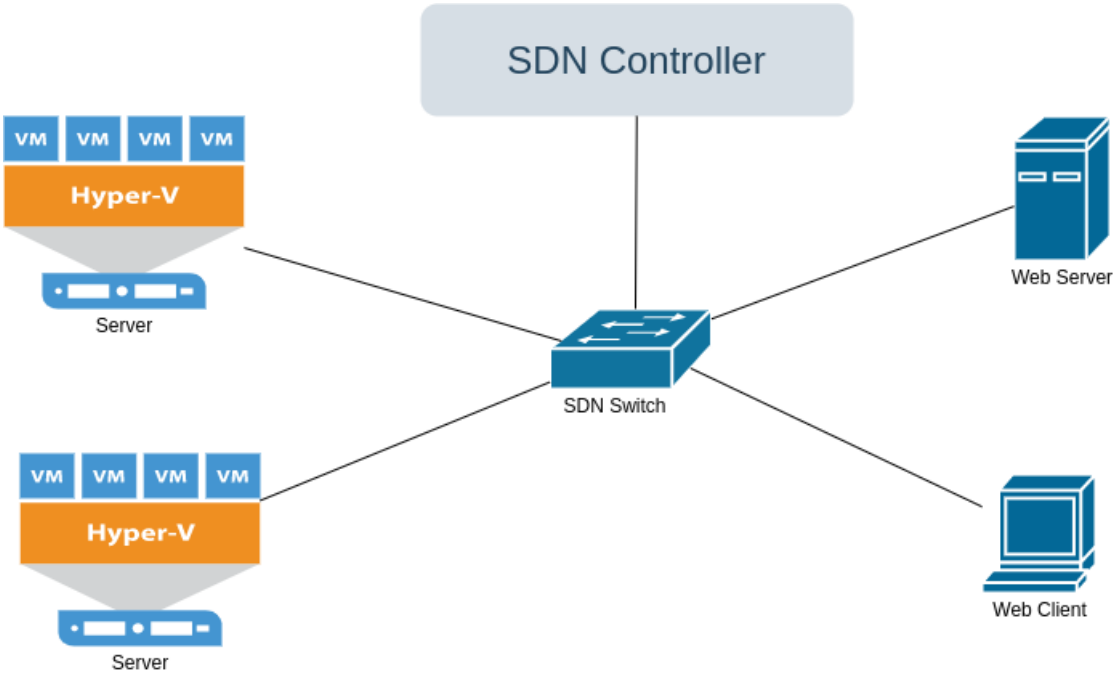


Figure 2.5. Example of Service Function Chain modelled using both NFV and SDN

2.3 Automatizing Network and Security Management

2.3.1 Characteristics and advantages

The novel paradigms presented so far allowed to put a strong emphasis on the principle of Network Automation, in fact it is almost always used in conjunction with network function virtualization and software defined network principles. It can be defined as the process of using software to automate network and security provisioning and management in order to continuously maximize network efficiency and functionality. It is based on the capability of react both when a new requirement is introduced in the system and when an event arises from the network in order to change the configurationo of the network itself; in addition, the current state and behavior of the underlying physical level should be taken into consideration by the decision system.

The IT security area has been traditionally largely hardware-based and required manual provisioning and management; this has been definitely one of the biggest limitation to cope cyberattacks. The reaction speed in the cyber security context

is fundamental to face an increasing variety of attacks which cannot always be predicted in advance, is almost impossible reconfigure manually a set o security devices in a suitable time without make errors. Instead automatizing Network and Security Management the configuration changes can be managed automatically with a slower latency.

The goal is to increase the network resiliency by avoiding manual interruption caused by a human being re-configuration in case of an event, so that the software can close a loop of reactions on its own, retrieving information by the network and promptly use them to allow the network itself to converge on the normal behavior. In order to have an automatic system which work properly and which is able to re-configure the network by itself upon the occurrence of different scenarios it is very important that the system has a complete overview on the entire infrastructure, with sensors which allow to monitor the traffic and establish which is the normal behavior.

2.3.2 VEREFOO

An example of framework which proposes the aforementioned approach to overcome the intrinsic limit of a traditional Service Function Chain is represented by VEREFOO [3] It stands for VErified REfinement and Optimized Orchestration. It is a framework whose main goals are the optimal automatic allocation and configuration of Network Security Functions on a Service Graph in order to satisfy a set of network security requirements expressed by the security administrator by means of an high-level security language through a refinement process; finally, the placement of the network security functions needed to satisfy the security constraints on the servers of a physical underlying network.

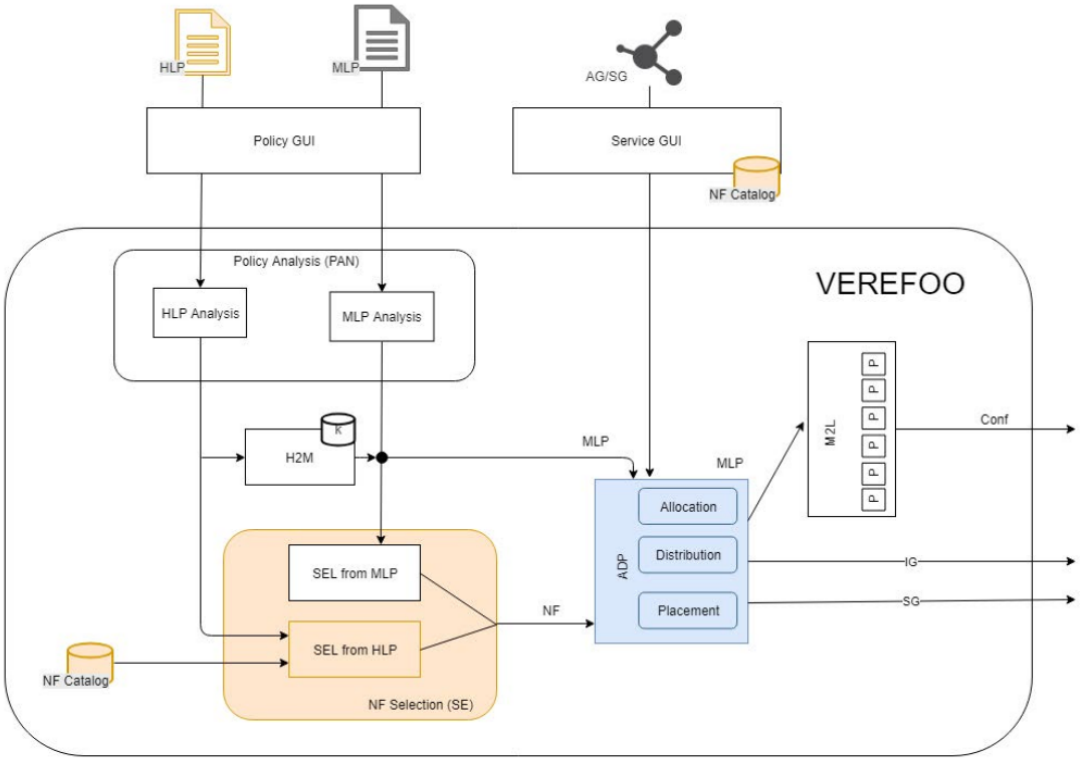


Figure 2.6. VEREFOO Workflow

In the figure the main components of VEREFOO are presented in order to describe the complete workflow of the framework. The user who wants to use VEREFOO has to give two different input:

1 A set of Network Security Requirements representing the security constraints which must be satisfied. The NSR can be specified by means of an high-level or a medium-level language depending on the experience level of the security administrator, through a Policy Graphic User Interface which allows a simpler creation of the requirements;

2 A Service Graph. It is a logical topology composed of a set of network functions like web cache, NAT, forwarder to build a complete end-to-end service; in a Service Graph multiple paths could exist between every pair of end points and also some loops could be present. In the context of VEREFOO a service graph is devoid of network security function like VPN Gateway or Packet Filters, which could be added afterword in the allocation graph in order to enforce some security requirement.

   In alternative, directly an Allocation Graph which is a logical topology, created starting from a Service Graph. An allocation Graph, therefore, is characterized by the same set of network functions of the Service Graph, but also by additional nodes. These additional elements are called Allocation Places and represent the places where a Network Security Function could be allocated.

Hence the security administrator could decide to provide in input directly an allocation graph if he already knows which are the position of the Allocation Places where the Network Security Functions can be allocated. Moreover he can also select the specific network security functions to be allocated, choosing it from a catalog available in VEREFOO accessible by means of a graphic user interface.

A first step performed by the framework is the conflict analysis of the Network Security Requirements received as input in order to detect some conflicts among requirements and to create the minimal set of constraints which must be respected in the network. The Policy ANalysis (PAN) module is the module of VEREFOO in charge of performing this analysis and whether it detects some conflicts which cannot be resolved automatically it generates a non enforceability report.

If the security requirements introduced in input are expressed by means of an a high-level language, the High-to-Medium (H2M) module implements a refinement process to obtain a corresponding set of medium-level Network Security Requirements, which contains all the necessary information for the future creation of the policies of the Network Security Functions automatically allocated on the graph and the low-level configuration of the VNFs placed on the underlying network.

The NF Selection module is able to select the proper Network Security Functions in order to satisfy the input Security Requirements choosing them from a pre-built catalog, which contains the same list of functions available to the user through graphic user interface. This phases could need to an optimization process thanks to which the framework is able to select an optimized set of NSFs, even though a possible limitation could be represented by the fact that it does not have any knowledge about the topology of the Allocation Graph.

The Allocation, Distribution and Placement (ADP) module is the core element of the VEREFOO architecture, taking in input a set of medium-language network security requirements, the list of the selected Network Security Functions and a Service Graph or directly an Allocation Graph is responsible for generating in output a new Service Graph containing the new network security functions automatically allocated in addition to the ones already present in the input graph or a Physical Graph; Furthermore this module provides a configuration,vendor independent, for each allocated network security function exploiting a medium level policy language.

Finally, the low-level configuration which is vendor dependent is generated,through a translation process, starting from the medium level configuration.

Chapter 3

Channel Protection Configuration

This chapter is entirely dedicated to the problem of configuring Secure communications. In the first section the policy based approach is introduced describing how does it works and the importance of configuring network security functions through a policy refinement process instead of entrusting it completely to the security administrator.

The section 2 focuses on the issues related to the manual configuration of the network security functions in charge of enforcing secure communications, explaining how this very complex task often is completely entrusted to users which have not always the adequate skills. In particular a study of the anomalies introduced by the security administrators during the manual configurations is analyzed, presenting 17 of them divided in five different categories depending on their effect; moreover a very useful empirical assessment is showed in this study which emphasizes the importance of the problem of misconfiguration.

Finally the works already present in literature related to the automatic configuration of security functions able to enforce secure communications are analyzed highlighting the fact that these works were not designed for virtualized networks and , as consequence, about how the proposed solutions lack in the flexibility and agility.

3.1 Policy Based Channel Protection Configuration

Without using a policy-based system for the configuration of Channel Protection Systems to enforce a secure communication or more in general for the configuration of security functions which are used to provide a service, this delicate task is completely entrusted to the security administrator, as consequence the probability that some errors will be generated is very high due to the fact that it is very difficult consider at same time a lot of aspect in configuring these functions such as efficiency, performance and of course the security aspects. Using a policy-based approach, instead, it is possible to define abstract rules which describe the behavior of a system and can be exploited by a security administrator, or also by some

other user without specific skills in this ambit, to verify that the network functions configuration satisfies the security requirements.

3.1.1 Policy Definition

A policy rule is a statement which consists of two distinct parts: the condition and the action.

- A policy condition is composed by one or more attributes assuming a specific value. A policy condition is matched if and only if all the values of the attributes are evaluated to true.
- A policy action represents the set of operations which must be performed if and only if the condition, it is bound to, is matched.

Therefore if the condition is met, then the action will be taken.

$$P = C \rightarrow A$$

3.1.2 Policy Refinement

An user can exploit an high-level language to express the network requirement, or if he has the adequate skills a medium level language. The first one allows to express all the requirements in a user friendly language; using an action-object-attribute language, example of high-level policies are "protect all the traffic generated by the website x" and "deny all the traffic to social networks". The second one is a language addressed to end users with good technical knowledge. It is an abstract language, vendor-independent, but differently from the HLPL allows expressing specific and technical requirements in a generic format such as "protect all the traffic from the node 192.168.1.1 to the node 192.168.1.254 using the encryption algorithm AES_128_CBC".

Following the study conducted by Valenza and Lioy [4],the main characteristics of the HSPL are:

- simplicity: the user must be placed in the position to easily specify a security policy, therefore he must be helped by providing him predefined statements and auto completion techniques;
- flexibility: it must be possible to specify any kind of security policy also supporting specific conditions for every security application;
- extensibility: the language must support future extensions, for instance the introduction of new kind of security policies;

while as regards the MSPL:

- Abstraction: the language must contain abstract security-related configurations, independent from a given vendor or product specific representation and storage in order to represent and enforce the same configuration on different security controls.
- Diversity: it must support the description of configurations for a variety of security functions (confidentiality, filtering, etc.).
- Flexibility and extensibility: it must be flexible and extensible enough to support the introduction of new security controls.
- Continuity: it must ensure the continuity of the policy chain, starting from HSPL down to the security control settings. This is useful to tracking which policy is actually enforced and which user is associated to it.

If the requirement are introduced by means of an high-level policy language they are subject to a refinement process to obtain the corresponding set in a medium-level policy language.

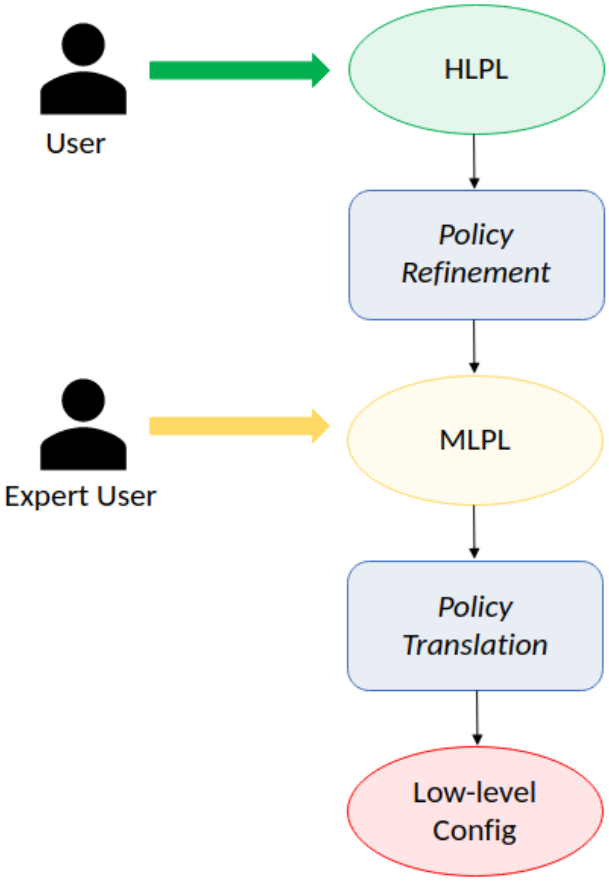


Figure 3.1. Policy Refinement Process

Finally the set of medium-level policies is translated in a set of low-level policies,

hence dependent from the specificity of the network functions implementation which will be used to provide the service.

3.2 Manual Configuration Problems

The manual configuration of network security functions to enforce a secure communication is a very complex and sensitive task. The security administrators in charge of enforcing network and communication security need to have an high level of competence and very specific skills. Nevertheless, in most cases this is not enough to avoid misconfigurations. The main causes, this manual process is so error prone, are:

- The networks are increasingly complex and the network security functions to configure are strictly related each others; this means that when a security administrator wants to configure a NSF need to taking into consideration all the configurations already present on the other ones on the network and is no trivial at all;
- Often the person in charge of configuring the security network functions not has the adequate competence to enforce a secure communication;
- It must be take into account that a human been can make an error at any moment.

Errors in the configuration of a network security function which should granting communication protection lead to several and very dangerous incidents like security breach; which is an incident that results in information being accessed without authorization. Typically, it occurs when an intruder is able to bypass security mechanisms exploiting for example security misconfigurations.

This situation could be improved by providing to the security admin tool support to debug the security controls' configurations and check if the enforced policy is compliant with the high level security requirements.

An important study developed on this relevant topic is [5]. It proposed a list of 19 anomalies which could arise during the implementation of communication protection policies, and designed a formal model based on FOL able to detect them. The anomalies are divided in five macro categories according to a effect-based classification, in the following a description of these anomalies and how they can occur is provided.

3.2.1 Insecure communication

Insecure communication occurs when the security level of the communication is less than the expected one; for instance when a channel does not satisfy the minimum

security level defined in the security requirement an *inadequacy anomaly* is arisen. When a secure communication is made up by more than one channel, the nodes at the junctions of the channels are able to see the traffic without protection, this situation could generate a *monitorability* anomaly. A *skewed channel* is an anomaly which can occur with a wrong tunnel overlapping that removes protection in a part of the communication. Finally, could be the cases that the request channel has a different security level from the from the replay one producing an *asymmetric channel anomaly*.

3.2.2 Unfeasible communication

Unfeasible communication happen when a communication cannot be established due to hard misconfigurations; for instance when the security administrator implement a security policy with a technology not supported by one of the endpoint a *non-enforceability anomaly* is arisen. When the node containing a policy implementation is the wrong node an *out-of-place anomaly* occurs. Sometime happens that the packets of a channel are dropped by a firewall on the path between the source and the destination generating a *filtered channel anomaly*. Performing communication security al L2 of the OSI stack could produce an *L2 anomaly* if there are technological incompatibilities between wired and wireless protocols.

3.2.3 Potential Errors

Potential Errors originate when the original intent of the security administrator is unclear; when two implementation policies have the same selector but different security settings, for instance the first one has an integrity property and the second one a confidentiality property the one with the highest priority will be selected to enforce the channel protection: if it is the first one a *shadowing anomaly* is generated, otherwise *exception anomaly* is produced. A *correlation anomaly* is caused by having two policy implementation with the source and the destination on the same node using the same security technology. Analogously an *affinity anomaly* is obtained with the same condition of the first one but using different security technologies. Finally a *contradiction anomaly* is arisen when exist two implementation policies which specify that a certain communication should be protect and not protect.

3.2.4 Sub-optimal Implementations

Sub-optimal Implementations occur when one or more policy implementations can decrease the network throughput by producing some overhead in the nodes. Their existence is usually not problematic, but their resolution can be beneficial since it improves the network performance and makes the PIs less vulnerable to DoS attacks; for instance when exists a policy implementation which includes another one using a different security technology a *redundancy anomaly* is produced. Analogously an *inclusion anomaly* is obtained with the same condition of the first one but using also the same security technologies. When a tunnel encapsulates other

tunnels with a higher security level a *superfluous anomaly* occur. Finally if there are channels in the network which can be removed without altering the network semantic an *internal loop anomaly* is present.

3.2.5 Sub-optimal Walks

Sub-optimal Walks take place when the path taken by the data is unnecessarily long. This group of anomalies not necessary lead to a misconfiguration. An *alternative path* anomaly occurs in the networks when there are multiple path between the source and the destination of a secure communication. If some packets cross a node more than one time to arrive at destination a *cyclic anomaly* is present.

3.2.6 Statistics

Furthermore a very useful empirical assessment is presented in this study which emphasizes the importance of the problem of misconfiguration. The experiment basically aim to realize whether the anomalies presented in these papers were actually introduced by the security administrators when they try to configure a set of network security functions to enforce some communication protection requirements and the impact that the various levels of expertise of the administrators have on the number of the introduced anomalies. The following figure shows the experiment results:

TABLE I

PERCENTAGE OF ADMINISTRATORS THAT CREATED AT LEAST ONE ANOMALY IN A MACRO-CATEGORY

expertise	insecure communications	unfeasible communications	potential errors	suboptimal implementations	at least one type
low	70.00%	60.00%	60.00%	70.00%	100.00%
medium	60.00%	30.00%	50.00%	40.00%	90.00%
high	30.00%	20.00%	20.00%	70.00%	90.00%
average	53.33%	36.67%	43.33%	60%	93.33%

TABLE II

PERCENTAGE OF ADMINISTRATORS THAT CREATED AT LEAST ONE ANOMALY

expertise	internal loop	non-enforceability	inadequacy	inclusion	affinity	monitorability	superfluous	filtered	contradiction
low	20.00%	30.00%	40.00%	30.00%	50.00%	30.00%	30.00%	30.00%	30.00%
medium	10.00%	20.00%	40.00%	20.00%	40.00%	20.00%	30.00%	10.00%	10.00%
high	10.00%	10.00%	10.00%	20.00%	20.00%	20.00%	50.00%	10.00%	0.00%
average	13.33%	20.00%	30.00%	23.33%	36.67%	33.33%	30.00%	16.33%	13.33%

TABLE III

PERCENTAGES OF ANOMALIES INTRODUCED BY THE ADMINISTRATORS GROUPED IN MACRO-CATEGORIES

expertise	insecure communications	unfeasible communications	potential errors	suboptimal implementations	total
low	18.41%	22.39%	15.92%	7.46%	64.18%
medium	16.54%	12.78%	9.77%	9.77%	48.87%
high	12.90%	4.03%	2.42%	12.90%	32.26%
average	16.38%	14.63%	10.48%	9.61%	51.09%

TABLE IV

PERCENTAGES OF ANOMALIES INTRODUCED BY THE ADMINISTRATORS

expertise	internal loop	non-enforceability	inadequacy	inclusion	affinity	monitorability	superfluous	filtered	contradiction
low	1.49%	4.48%	10.95%	2.99%	6.97%	2.99%	7.46%	17.91%	8.96%
medium	1.50%	3.76%	13.53%	4.51%	5.26%	3.01%	3.76%	9.02%	4.51%
high	1.61%	0.81%	4.03%	3.23%	2.24%	8.87%	8.06%	3.23%	0.00%
average	1.53%	3.28%	9.83%	3.49%	5.24%	6.55%	4.59%	11.35%	5.24%

Figure 3.2. experiment results

The collected data gave very interesting information:

- The 93% of administrators introduced at least one anomaly, regardless of the expertise;
- All the new anomalies have been introduced by at least one administrator;
- All the anomaly types except contradictions were also introduced by expert administrators;
- the more the expertise of administrators the less the number of anomalies.

As can be noticed by the gathered data the anomalies presented in this paper can appear in real world scenario, hence it is useful to look for them; Furthermore, it demonstrates the importance of the creation of a model able to detect these anomalies when a security administrator tries to perform a manual configuration of security function.

3.2.7 Considerations

In the ambit of this thesis, the aforementioned work has been very useful to become aware of all the possible anomalies that a misconfiguration could cause. It is worth to note that the presence of these anomalies is not only a concern of a manual configuration, but also of a tool which performs that automatically, in fact the formal verification is a fundamental phase in the tool execution necessary to verify that all the configurations provided as output satisfy the input security requirements and that non of these anomalies is generated. Therefore in the design phase of the model, this thesis is based on, all these anomalies has been taken into consideration producing an automatic tool for the configuration of the protection policies starting from a set of security requirements able to recognize these anomalies and avoid them. These process depending of the kind of anomaly influenced various parts of the design phase like the characterization of the nodes that will be presented in the next chapter, the drafting of some hard constraints which will be discussed in the cap 8 and the modification of the pre-processing task to make it able to detect most of them.

3.3 Automatic Channel Protection Configuration

Automation for the configuration of network security functions has been in the recent years a widely investigated topic, especially thanks to the diffusion of new technologies like software defined network (SDN) and network function virtualization (NFV), which gave to the networks the necessary flexibility and, as consequence, new ideas and concepts to exploit.

Nevertheless, the automatic configuration of some network security functions has been analyzed more in depth than others, so it is possible to say that nowadays there are some network security functions whose automatic configuration has made great strides, while other ones lack in a good solution.

The reasons of this situations are various:

- some network security functions are more utilized than other ones, so it is normal that the most part of the researches effort were focused on those functions;
- some network security functions are more complex to handle with and configure than other ones;

To better understand the current situation two different category of network security functions can be taken in consideration: the firewall and the Channel Protection System, which are fundamental to enforce a secure communication. The firewalls are one of the most utilized network security functions and also one of the simplest to configure, for these two reasons have been the most treated security functions as regards the automatic configuration issue. On the other side Channel Protection Systems, although implement secure communications is nowadays essential in order grant the integrity, the authentication and the encryption of the

packets flowing from a source to a destination, is something much more difficult to handle with, for instance because a CPS modifies the packet flowing through it or because to establish a secure communication there is the need of more than one CPS, for this reason the progress about automatic configuration has been quite more limited.

In the following the most relevant works on the automatic configuration regarding Channel Protection are analyzed, focusing on what has been done, what is missing and has been implemented in this thesis work.

3.3.1 Related Works

The milestone in this area is represented by [6] whose goal is the automatic generation of security policies for a set of requirements given as input and the formal verification that the produced polices are conflicts free. A security policy set is considered correct if and only if it satisfies all the requirements.

The authors proposed three different algorithms to implement what mentioned above:

1 The **bundle approach** is the first algorithm for solving the IPsec policy-conflict problem. It is characterized by two different steps. The first one regards the separation of the traffic into several disjoint traffic flow sets, called bundles, each of which is subject to a unique set of security requirements. The second one regards the generation of the security policies for each bundle starting from the set of requirement actions each bundle is associated with.

2 The **direct approach** create tunnels directly assuring that new tunnels don't overlap with the existing ones, in fact when this situation occurs, it is able to handle it by creating two new consecutive tunnels which don't overlap each other.

3 The **combined approach** as can be easily guessed by the name is a combination of the previous presented approaches. It starts using the direct approach and if for some reason a solution can not be found then it utilizes the bundle approach.

The same group of researchers proposed in [7] the **ordered-split algorithm** which not only is able to compute a correct solution but is also able to select the optimal one with regarding to minimizing the number of required VPN tunnels. It is able to manage the encryption requirements and the authentication requirements in a separate way. It first converts original requirements into tie-free requirement sets and then generates minimal size canonical solutions for the new requirements. A canonical solution can be defined as a solution in which there are no tunnels starting at the same position or ending at the same position. A Tie-free requirement set can

be defined as a requirement set in which there are no requirements sharing the same from values and the same to values. The from and to values of one requirement determine the network area which needs to be protected. This algorithm produces a lower number of tunnels compared to the bundle and direct approaches.

Another relevant approach is presented by [8] where a heuristic algorithm is used to generate the tunnels in a iterative way starting from the longest. He proposed an algorithm to automatically generate conflict-free policies and satisfy all the security requirements generating fewer tunnels and saving more computational power than the previous algorithms.

In this three works, however, the environment that has been taken into consideration is intra-domain. An extention described in [9] presents a distributed architecture and a collaborative negotiation protocol for inter-domain security policy management. Under the BANDS framework, the low-level policies are automatically generated with high-level requirements defined in advance. The overall protocol includes "AS route path discovery" phase and "collaborative negotiation protocol", trying to keep the requirement information shared as minimal as possible, so that only requirements that are pertinent to the flow are revealed to others. However this work has been tested only for static routing on some given end-to-end connections. Other study where done to improve some aspects for the generation of conflict-free policies such as [10] whose solution improves scalability, agility and robustness.

3.3.2 Considerations

From the analysis of the related works, it emerges that none of these perform an automatic allocation of the CPS, in fact the starting point of their studies is a network topology where the network security functions have already been placed; furthermore they are not variable of the system i.e. their position is fixed and will not be moved by the algorithms. The main reason of this behavior, apart the complexity to consider an automatic allocation of network security functions, is that these works were not designed for virtualized networks but for physical network where each security function represents a different physical box lacking in the flexibility introduced by a virtualized or hybrid network. Another aspect that is almost totally missing in these studies is the research of an optimum solution by introducing some optimum criteria, apart from [yang] with try to do that through an heuristic algorithm. Finally the last issue worth to be mentioned is the scalability of the presented algorithms some of them have been tested with a maximum number of security requirements that is a not coherent considering the dimension of the current networks.

Consequently, it is possible to state that the combination of an automatic allocation of the Channel Protection Systems and an automatic conflict-free configuration

of these functions considering some optimal criteria represents a central novelty in literature.

Chapter 4

Approach

In this chapter is described the approach hold to develop this work. In the first section the objective of the thesis is presented and explained dividing it into a set of smaller goals to accomplish in order to satisfy the overall purpose. Essentially they deal with the modeling and implementation of the ADP module of VEREFOO to make it capable of performing the automatic allocation and configuration of CPSs.

In the second section more details about the ADP module are provided since it is the module of VEREFOO this thesis gave a contribution in the implementation; moreover all the possible scenarios that could occur when using the ADP module for the allocation and configuration of the security function able to enforce secure communications are showed, detailing the input and the output of the this module and the peculiarities of each scenario.

The last section deals about how the approach is based on a formulation of the methodology through a partial weighted MaxSMT problem, describing the principles which it is based on, emphasizing the difference between hard and soft constraints and how these have been exploited to model different aspects of the ADP module.

4.1 Thesis Objective

In the previous chapters it has been provided a clear and detailed description of the actual situation about the creation, configuration and management of secure communications, highlighting the main issues related to this ambit, furthermore it has been explained how the network automation, exploiting novelties such as Network Function Virtualization and Software Defined Network represents the right approach to pursue because it is able to confer flexibility, agility and network resiliency to the solutions applying this approach. Therefore **the objective of this thesis work is the modeling and the implementation of a solution able to automatically allocate and/or automatically configure Network Security functions called Channel Protection Systems (CPSs), respecting some optimality criteria, in order to satisfy a set of Network Security Requirements related to the enforcement of secure communications.** The secure communication are enforced through the creation of end-to-end, site-to-site

and secure gateway Virtual Private Networks. To achieve this goal an existing tool described in the Chapter 1 has been exploited: VEREFOO; in particular the goal has been the extension of the functionality of the ADP module of VEREFOO in order to make it capable of performing the aforementioned objective. The overall purpose of this thesis work can be seen as a sub set of goals whose accomplishment has been essential to the achievement of the overall purpose:

- The first goal has been the modeling and the implementation of a new type of network security requirement, which represents the first input of the ADP module, in such a way that consents to express the intention from the security administrator to create a secure communication exploiting either a HSPL which will be subsequently refined to obtain the corresponding requirement in the MSPL or directly exploiting a MSPL. In the same way have been modeled and implemented the policy implementations which the ADP module is able to give in output for each allocated security function exploiting a MSPL in order to make them technology independent. The achievement of this goal has been characterized by the analysis of the most used frameworks for the manual creation of secure communications; in fact analyzing and experiencing with these tools has been a fundamental step to understand in a detailed manner how secure communications work in practice and which is the set of information that can be considered crucial for the establishment of a secure communication. These notions played a key role in the selection of both the information which a security requirement, related to the enforcement of a secure communication must necessary contains and the information that are essential for each implementation policy in order to consent to the allocated network security functions to work properly.

- The second goal has concerned the design, modeling and implementation of the network security functions able to create a secure communication i.e. the functions that could be automatically allocated and/or automatically configured to satisfy a communication protection requirement. More in details this goal has dealt with : the extension of the capabilities of already existing network functions belonging to the category of the End-Host, i.e. those functions whose position is fixed at the edge of the network, such as WebClients and WebServers, enabling them to protect its own traffic; the creation from scratch of a network security function, the VPN Gateway, in order to protect the traffic generated by other nodes.

- The core goal has been to make the ADP module capable of performing an automatic allocation and/or an automatic configuration of the necessary CPSs following some optimality criteria through the resolution of a Max SMT problem, receiving in input the set of communication protection requirements, the CPSs and the Service/Allocation graph described in the section 3 of the first chapter; the MaxSMT problem is formulated so as to provide also a formal verification that the achieved solution is formally correct. About the optimality criteria an important features has been introduced in this work, that is the faculty from the security administrator to change the priority associated to them having the possibility to choose among some available

profiles, as a consequence of this a different solution could be proposed by the solver which best suits to the admin needs according to the specified priorities.

- Another important goal has been the reachability assurance of the packets belonging to a secure communication flowing from the source to the destination when a packet filter is present on the traversed path. This situation is very common and represents a not negligible problem because even if a secure communication is correctly enforced a packet filter on the path, depending on its configuration, could drop the packets belonging to that secure communication making impossible for the destination receiving the related packets and generating the so called filtered channel anomaly.

4.2 ADP Module

In the Chapter 1 the general model and workflow of VEREFOO has been presented, i.e. a model able to consider different types of network security requirements, depending on the kind of security the administrator would like to introduce in the network and able to select the proper security functions to allocate and to configure in order to satisfy those requirements. Since this thesis work deals with the automatic allocation and configuration of CPSs in order to enforce communication protection requirements some limitations have been introduced in the model:

- The security administrator can specify only communication protection requirements;
- The unique available kind of security function which can be automatically allocated and/or automatically configured are the Channel Protection Systems.

The Allocation Deployment Placement (ADP) is the module of VEREFOO for which this thesis gave a contribution in the implementation. It is in charge of performing two important task for the correctly working of VEREFOO:

1 The Automatic Orchestration and Configuration task. It is in charge of allocating the NSF on the allocation graph received by input or generated from the Service Graph, moreover on each allocated network security function the policy rules are computed, which represent the configurations necessary to satisfy the input security requirements.

2 the Automatic VNFs Placement task, executed after the AOC task, which is appointed to place the VNFs present in the output Service Graph (original Service Graph plus the added NSFs) of the AOC task on the Physical Graph which represents the physical infrastructure.

Considering the aforementioned limitation in the context of VEREFOO four different scenarios can verify in the execution of the AOC task depending on the provided inputs.

In the first scenario the ADP module takes in input:

- A set of communication protection requirements specified by means a medium level policy language which need to be enforced;
- A Service Graph without any type of network security function allocated in it, but with some possible indications about the positions where the Network Security Functions must or must not be placed, if the security administrator has the necessary competences to determine it.
- The functions able to satisfy the communication protection requirements i.e. the CPSs.

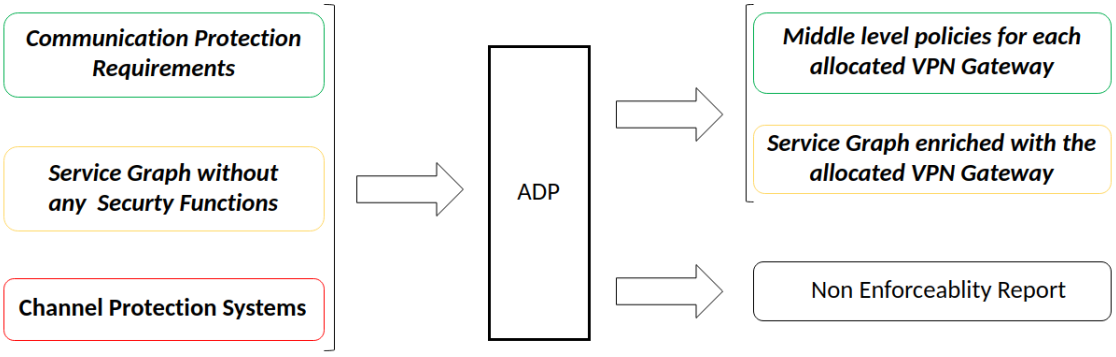


Figure 4.1. Scenario 1

This scenario represents the circumstance in which no security network functions were allocated yet, for instance due to a previous execution of the tool or by a manual configuration of the security administrator. The output varies depending upon whether the solver finds or not a satisfiable solution. If a solution able to satisfy all the input security requirement has not been found a non enforceability report is given in output containing the description of the problem that make impossible to compute the Service Graph. Possible reasons of a non enforcability output could be:

- insufficient number of allocation places;
- the position of the virtual functions on the graph provided in input;

Otherwise, a Service Graph composed by all the network functions present also in the input Graph plus the CPSs allocated in the Allocation Graph, following some optimality criteria, automatically generated from the input logical topology is computed and for each CPS a configuration is automatically created and given in output to the security administrator exploiting a medium-level policy language to satisfy the input communication protection requirements;

In the second scenario the ADP module takes in input:

- A set of communication protection requirements specified by means a medium level policy language which need to be enforced;
- A Service Graph with some network security functions allocated in it. Also in this case some possible indications about the positions where the Network Security Functions must or must not be placed, if the security administrator has the necessary competence to determine it.
- The functions able to satisfy the communication protection requirements i.e. the CPSs.

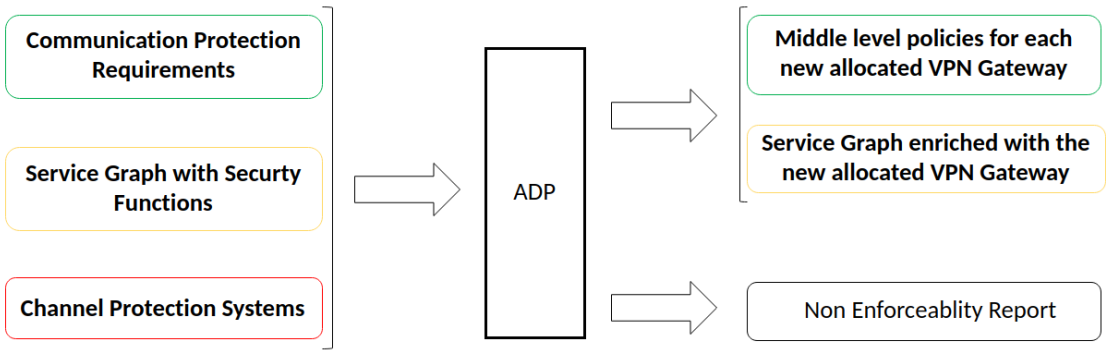


Figure 4.2. Scenario 2

This scenario represents the situation in which the automatic allocation and configuration has already been performed on the considered network or the security administrator allocated and configured some security functions manually. The output varies depending upon whether the solver finds or not a satisfiable solution. If a solution able to satisfy all the input security requirement has not been found a non enforceability report is given in output containing the description of the problem that make impossible to compute the Service Graph. If a solution able to satisfy all the input security requirement has not been found a non enforceability report is given in output containing the description of the problem that make impossible to compute the Service Graph. Possible reasons of a non enforcability output could be:

- insufficient number of allocation places;
- the position of the virtual functions on the graph provided in input;

Otherwise, a Service Graph composed by all the network functions and network security functions present also in the input Graph plus the CPSs allocated in the Allocation Graph, following some optimality criteria, automatically generated from the input logical topology is computed and for each CPS a configuration is automatically created and given in output to the security administrator exploiting a medium-level policy language to satisfy the input communication protection requirements;

It is worth to mention that whether in the input Service graph some Channel Protection Systems have been already allocated, but without any configuration it will be computed by the APD module and provided in output.

In the third scenario the ADP module takes in input:

- A set of communication protection requirements specified by means a medium level policy language which need to be enforced;
- An Allocation Graph with or without network security functions allocated in it. In this case the security administrator can specify also all the possible Allocation Places where the Channel Protection Systems can be allocated; furthermore he can manually allocate a CPS on the graph, configure it manually or leave it without configuration in order to be configured by the ADP module. The tool cannot modify the position of these CPSs and their configuration (if any), unless something different is specified by the secure administrator. Finally other network security functions such as packet filters could be manually allocated in the graph but a configuration must be already be present for these functions.
- The functions able to satisfy the communication protection requirements i.e. the CPSs.

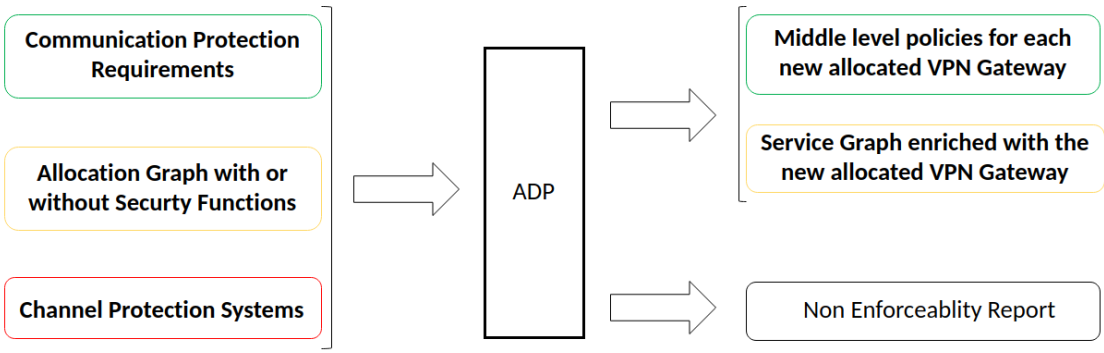


Figure 4.3. Scenario 3

This scenario represents the situation in which the security administrator is very confident with his skills and does not want to perform a complete automatic allocation and configuration but he prefers to force some decision, like the presence of a CPS in a certain position of the network. The output varies depending on the fact that the solver finds or not a satisfiable solution; If a solution able to enforce all the input security requirement has not been found a non enforceability report is given in output containing the description of the problem that make impossible to compute the Service Graph. Possible reasons of a non enforcability output could be:

- the position of the already present Network Security Functions;
- the configuration of the already present Network Security Functions;
- the way the security administrator positioned the allocation places ;

Otherwise, a new Service Graph is computed where additional Channel Protection Systems have been optimally allocated, while keeping the already existing CPSs, and their configuration is automatically created to satisfy the input medium-level communication protection requirements. It is worth to mention that whether in the input Service graph some Channel Protection Systems have been already allocated, but without any configuration it will be computed by the APD module and provided in output.

The goal of the last scenario is verify that the current allocated security functions and their configurations are enough to enforce a set of security requirements, in fact it takes in input:

- A set of communication protection requirements specified by means a medium level policy language which need to be enforced;
- A Service Graph, where the CPSs are already allocated on the Graph. They could be configured or not. If a configuration is present could have been manually provided by the security administrator or automatically generated due to a previous execution of the framework; If a configuration is not present, it should be filled by VEREFOO.

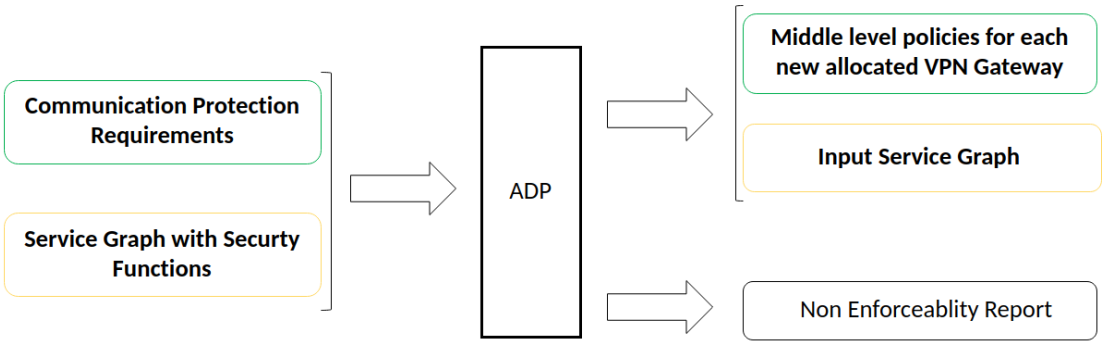


Figure 4.4. Scenario 4

This scenario is usually utilized when the security administrator wants to realize whether a previous computation of the ADP module is suitable for satisfying a new/modified set of communication protection requirements, without allocating and configuring new CPSs; or when the security administrator uses the framework to automatically compute the configuration of the CPSs.

The output varies depending on the fact that the solver finds or not a satisfiable solution; If a solution able to enforce all the input security requirement has not been found a non enforceability report is given in output containing the description of the problem that make impossible to compute the Service Graph. Possible reasons of a non enforcability output could be:

- the position of the already present Network Security Functions;
- the configuration of the already present Network Security Functions;

Otherwise, the input Service Graph is given in output where, if requested the configurations to satisfy the input communication protection requirements is automatically computed and provided in output.

4.3 Method formulation

This approach is based on a formulation of the methodology through a the Maximum Satisfiability Modulo Theories problem (MaxSMT) which is an extension of the SMT problem in the optimization context, where given a set of predicate clauses containing predicate variables, the goal is to find the optimal values of these variables which achieve the maximum satisfiability of the clauses.

Like the SMT problem, the MaxSMT problem is NP-complete in terms of worst case computational complexity. The main difference is, however, that each clause is assigned a weight, unitary in the standard version; consequently, it is not sufficient to find a solution which satisfies the predicate clauses, but among all the possible solutions the chosen one must maximize the number of satisfied clauses.

The most common MaxSMT variants are:

- the weighted MaxSMT, where a different weight can be assigned to each clause and consequently the research for the best solution prioritizes the satisfiability of the most valued clauses;
- the partial MaxSMT, where some constraints are not relaxable because they must be satisfied, while other clauses do not require to be necessarily satisfied for the achievement of a solution;
- the weighted partial MaxSMT, which is a combination of the two previous instances.

In the context of this thesis the weighted partial MaxSMT is adopted, hence two different category of constraints have been defined:

1 **Hard Constraints**. Are the not relaxable constraints, therefore, their fulfillment is mandatory in order to obtain a satisfiable solution. For this reason this kind of constraint has been used to implement the behavior of the components of the ADP module in order to guarantee the proper working of the whole module. For instance the statement "the end-host can protect only its own traffic" must be implemented through the usage of an hard constraint because it is something that must be always guaranteed.

2 **Soft Constraints**. Are the relaxable constraints, hence their fulfillment is not mandatory to obtain a satisfiable solution. Since it has been adopted a weighted MaxSMT problem each soft constraint has a specific weight that represents the priority assigned to it, therefore, the solver among all the satisfiable solutions selects the one whose sum of all satisfied soft constraints, considering their weights, is the highest. Thanks to this kind of constraint it has been possible handle with all those statements which represent preferences to satisfy in order to reach an optimal solution, for instance the statement: "do not use end-systems to enforce secure communications" implemented as soft-constraints means that the solver tries to find a solution without using end-systems to enforce a secure communication, but if a solution is not found it will try to find a satisfiable solution using them.

A formal definition of the the weighted partial MaxSMT problem given a set H of hard constraints and a set S of soft constraints can be :

$$\begin{aligned} &max \sum_{i=1}^{S} w_i * s_i \\ &subject\ to\ h_j,\ \forall j \in [1, H] \end{aligned} \tag{4.1}$$

In order to formulate this methodology as a MaxSMT problem it has been necessary to define different formal models for the different components involved in the execution of the ADP Module to enforce a secure communication. These models are described in the next chapters, in particular:

- Chapter 5 deals with the modeling of the network nodes and links composing a Service Graph or an Allocation Graph which represent the first input of VEREFOO. Essentially each node between a source and a destination can be associated to a one of the predetermined categories, affecting the decision process of the solver. The second part of this chapter focus on the model of the communication protection requirements i.e. those requirements that consent to specify the enforcement of a secure communication.

- Chapter 6 is dedicated to the Communication Protection Systems , i.e. those network security functions which consent to create a secure communication, describing them in a detailed way starting from their behavior to their configuration exploiting a medium level policy language. Then, the concepts of technology and capability are introduced, after defining them, the relation between these two concepts and the Channel Protection Systems is explained. Furthermore, different profiles are presented to express the preference, from the security administrator, about which of the two type of CPSs should be used to establish secure channels and others details related to their configurations in order to reach an optimal solution. Finally, the way to grant reachability assurance of the packets belonging to a secure communication , flowing from the source to the destination when a packet filter is present on the traversed path is illustrated.

Chapter 5

Network and Security Requirements Model

The first part of this chapter deals with the modeling of the network nodes and links composing a Service Graph or an Allocation Graph. Essentially 4 different category are defined: untrusted nodes, inspector nodes, trusted nodes, and untrusted links illustrating, for each one of them, the advantages of creating these categories.

The second part of this chapter focuses on the model of the communication protection requirements, i.e. those requirements that consent to specify the enforcement of a secure communication, explaining the meaning of each field belonging to it and presenting the formulations which consent the system to verify that the security requirements inserted in input are satisfied by the solution provided by the solver.

5.1 Nodes Characterization

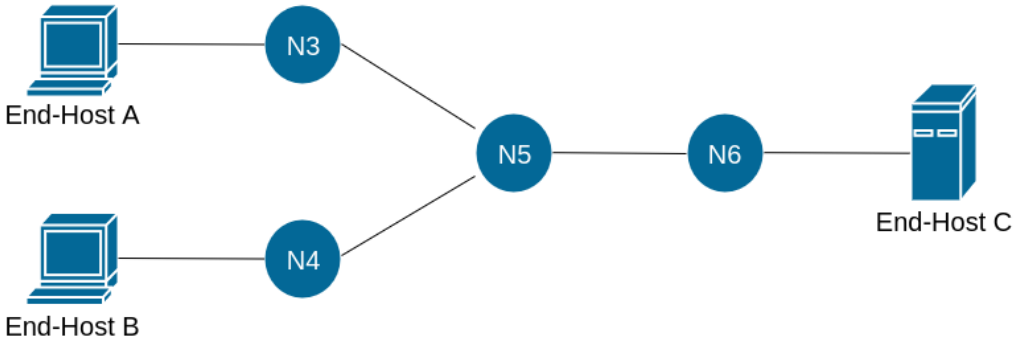
This section deals with the nodes characterization i.e. the study of how categorize the nodes between a source and a destination of a secure communication for instance depending on which type of function is installed on or if it is consider reliable or not. Works such as [6] and [5] were very helpful to understand all the possible roles the intermediate nodes could have, so the nodes present on a service graph has been divided in three different categories.

5.1.1 Untrusted Nodes

The Untrusted nodes are those boxes on which the enforcement of the security requirements is mandatory. Usually when a user try to implement a secure communication does not take care about defining a set of untrusted nodes, but simply by default all the nodes between the source and the destination are considered untrusted. This strategy, although produces a valid secure communication because there is not the risk that some middle-nodes see the traffic without protection, it is an ordinary and not very flexible solution. For this reason this way of proceeding

should be applied only if the person in charge of define the security requirements have no knowledge of the network and of the nodes that compose it. Instead, considering a set of untrusted nodes makes the solution space wider and in most of the cases it allows a savings in term of allocated resources; of course if there are doubts about the reliability of a node it is safer consider that node untrusted.

A clarification example about the advantages of considering a set of untrusted nodes is provided in the following:



Considering the network depicted above let us suppose that the network administrator wants to protect the traffic flowing from the host A and the host B to the host C and that the Node 6 is untrusted.

Without the possibility of specifying a set of untrusted middleboxes in the requirements all the intermediate nodes must be considered untrusted, so the only available configuration to satisfy the requirements is the allocation of 3 VPN Gateway: the first one immediately after the host A, the second one immediately after the host B and the third one between the host c and the node N6.

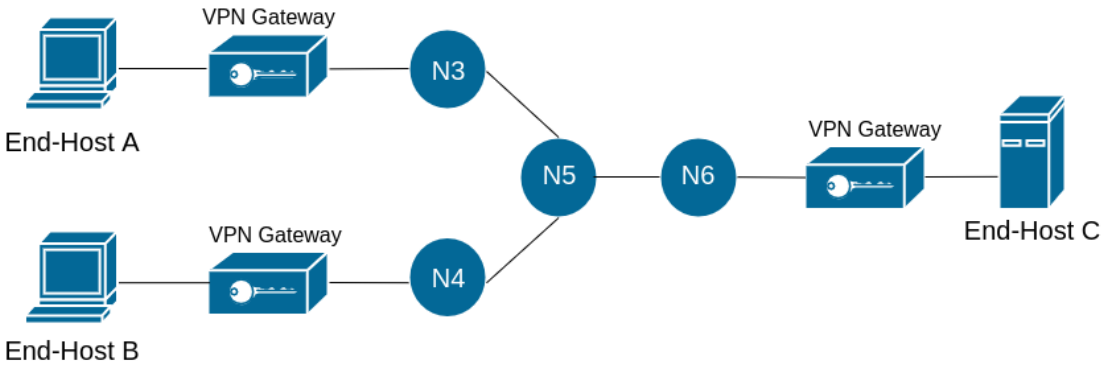


Figure 5.1. solution 1

Instead, with the possibility of specifying a set of untrusted middleboxes, aside from the solution 1, other two possible scenarios are available to satisfy the requirements.

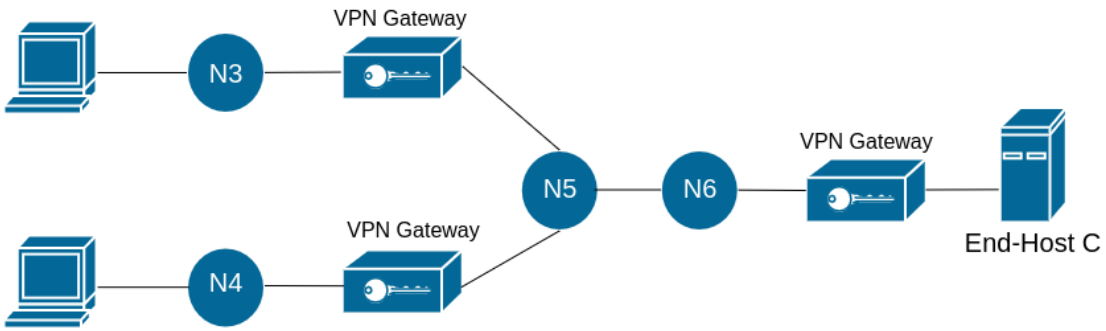


Figure 5.2. solution 2

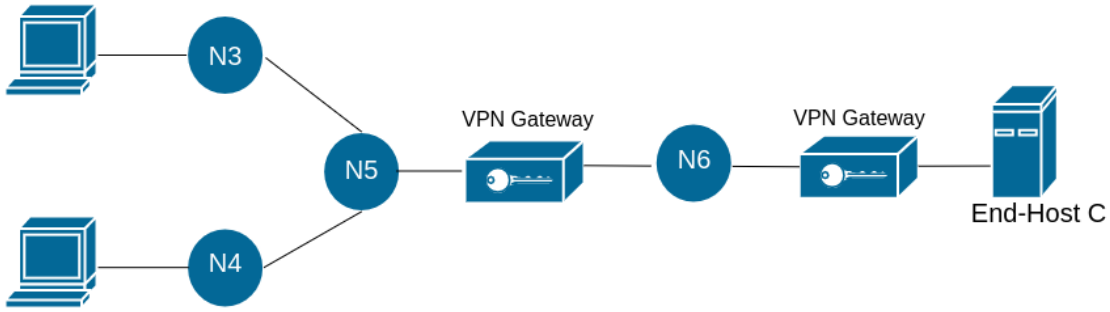


Figure 5.3. solution 3

It is worth to note that the solution three, placing a VPN Gateway between the node N5 and the node N6 and a VPN Gateway between the node N6 and the end-host C, satisfies the security requirements reducing the number of allocated VPN Gateway and all the resources necessary to deploy it compared to the previous solutions.

In some cases can be useful specify the untrusted link instead of the untrusted nodes, rightly when the nodes, a secure communication flows through, are considered reliable but there are doubts about the links that connect them, because for instance could be sniffed. This happens in hybrid network i.e. not fully virtualized network, because can be made the assumption that the links internal to the data center are always trusted. Hence the untrusted links are those links on which, analogously to the untrusted nodes, the enforcement of the security requirements is mandatory.

5.1.2 Inspector Nodes

Inspector nodes are those boxes on which the traffic must flow without protection in order to be analyzed. Often the implementation of a secure communication goes in contrast with other security functions present on the path between the source and the destination of the secure communication itself, because usually the network security functions (for instance IDS) want to see the network traffic in clear in order to analyze it and evaluate if something malicious is contained in them. However, without the possibility of defining the set of inspector nodes, what happen is that these functions are forced to make decisions on the basis that the packets arriving at them are encrypted or not and most of the time are set to drop them or raise some alarm. This category has been thought to avoid this difficult situation facilitating interoperability between security functions, though could lead to an addition delay in the communication due to the fact that the traffic may be encrypted/decrypted more than one time and also the number of apparatus necessary to enforce a secure communication could increase.

Considering the network formerly presented let us suppose that the network administrator wants to protect the traffic flowing from the host A and the host B to the host C and that the node N5 is untrusted and node N6 is inspector. A feasible solution could be :

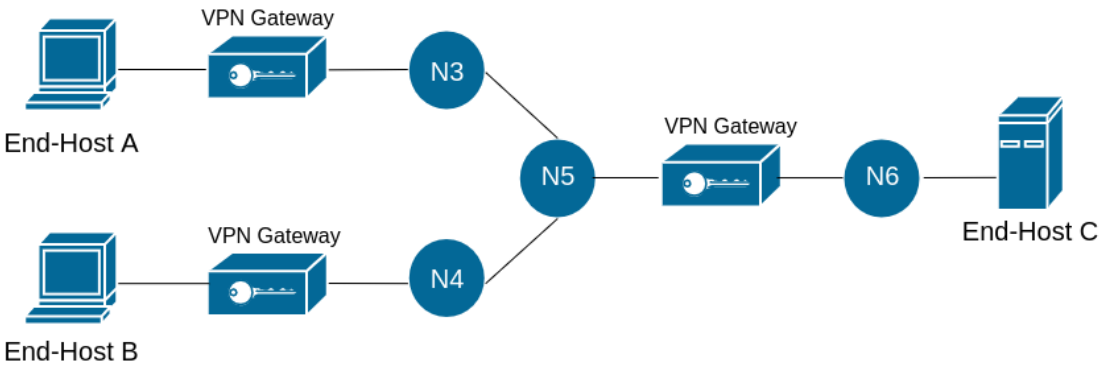


Figure 5.4. inspector node solution

5.1.3 Trusted Nodes

Trusted nodes are those boxes on which the security requirements could be enforced or not, the decision is left to the solver. Let us consider A, U and I as the sets of All, untrusted and Inspector middleboxes respectively, then the set of trusted middleboxes is:

$$T = A - U - I$$

All the middleboxes that are neither untrusted nor inspector will be set as trusted by the solver. This means that the security administrator does not need to specify the middleboxes belonging to this category.

5.2 Network Security Requirements

5.2.1 Description of the Network Security Requirements

The Network Security Requirements represent one of the two input of VEREFOO. The security administrator can insert them using both an HLPL and MLPL depending on his skills in the security field:

1 A high-level representation can be exploited by the security administrator if he has not deep knowledge about the configuration of the network the security functions to be deployed, hence he can't specify detailed information in the requirements. This is an user-friendly approach, very simple to understand and to use where each end point of the service is characterized by a unique identifier, that is a label which can then be mapped to an IP address. To cite an example, it is legitimate to request that the MongoDB database must not be reachable by a certain Web Server, even though their current IP addresses (and even ports) are not known, because these parameters can be established by the framework automatically, exploiting the identifiers of each end point.

2 Instead if the security administrator has high competences in the configuration of network security functions and he is confident in his capabilities, he can decide to introduce the network security requirements using directly a medium-level representation which includes more details such as ip addresses and ports of the involved nodes. The additional operation of refinement will be avoided using this approach; moreover, these representations are formulated so that they do not regard the set-up of the virtual functions, which is dependent on different vendors and can be easily computed afterwards.

In VEREFOO, the H2M module, described in the Chapter 1, performs a translation from the high-level requirements to the medium-level representations; consequently, the ADP module receives exclusively the constraints expressed by means of a medium-level language and which do not need any further modification. The thesis work considered this aspect as an assumption in defining a the model for representing the security requirements. In more details, the Network Security Requirements which this thesis analyzed are the protection requirements between a pair of end points; i.e. the creation of secure communications characterized by a set of security properties depending on what is specified in the requirements.

5.2.2 Communication Protection Network Security Requirements

The Communication Protection security requirement is the network security requirement developed entirely in this thesis work. It is able to establish a secure communication between two end-point, creating one or more channels from the source to the destination, depending on where the untrusted and inspectors nodes are positioned alongside the path. The security channel characteristics must be compliant with which is requested in the network security requirement.

Let R be the set of the communication protection requirements which a security administrator can specify and must be satisfied on an input Service Graph. Each requirement in this set, i.e. each security requirement, is formulated exploiting the following components of a medium-level representation:

$$[ruleType,\ filteringConditions,\ protectionInfo,\ topologyInfo]$$

- **RuleType** states which kind of security requirement should be satisfied. In the context of this thesis can assume only the value *ProtectionProperty*, but it is useful for a further integration with other kinds of requirements.
- **FilteringConditions** in order to select the traffic which must be protected:
  - *IPSrc* is the source of the traffic flow which requires to be protected;
  - *IPDst* is the destination of the traffic flow which requires to be protected;
  - *portSrc* is the transport-level source port of the traffic flow which requires to be protected;
  - *portDst* is the transport-level destination port of the traffic flow which requires to be protected;
  - *transportProto* is the transport-level protocol of the traffic flow which requires to be protected.

  For the representation of the IP addresses, i.e. IPSrc and IPDst, the traditional dot-decimal notation has been exploited:

  $$ip1.ip2.ip3.ip4$$

  where ip_i , $\forall i \in \{1,2,3,4\}$, can be an integer in the interval from 0 to 255, extremes included, or alternatively a wildcard element, identified with $*$. This symbol allows to have a unique statement for both a network address and the corresponding netmask, instead of two separate elements: for example, the 20.6.8.$*$ representation can be used to express the end points that are present in the network 20.6.8.0/24, while the 30.2. $*$.$*$ representation characterizes the network 30.2.0.0/16.

  The IP addresses which are specified as source and destination in the Network Security Requirements do not necessarily coincide with the IP addresses of the end points of the Service Graph; for instance, if the service designer provides a reachability property between the source 10.0.0.$*$ and the destination 30.0.0.1, then if in the Service Graph the two nodes 10.0.0.1 and 10.0.0.2 are present two kinds of traffic should be actually allowed. For this reason, a pre-processing task already existent in the framework was exploited so that, when the formulas of protection properties are built in the model of the MaxSMT problem, each property refers to IP addresses effectively assigned to nodes in the logical topology. In the aforementioned examples, the pre-processing task, from those input security requirements, would easily create two separate requirements, where the first has 10.0.0.1 as source, while the second has 10.0.0.2 as source. The pre-processing task furthermore, assures that there is

no redundancy between requirements, hence the set of security requirements provided in input to the ADP module can be considered redundancy free.

The source and destination transport-level ports portSrc and portDst can be formulated by means of a single number or an interval of numbers, considering a range from 0 to 65535. To clarify this concept, if it is required that the source 10.5.8.4 must not be able to reach the destination 20.3.6.1 is the port numbers are included in the interval [1000, 2000], the traffic flow between the two end points must be blocked if the packets are characterized by a source port number included in this interval, so that the isolation requirement is satisfied. Finally, the transportProto element represents the layer-4 protocol which is used above the IP layer; the formulation can have TCP and UDP as possible values or, also for this component, the wildcard $*$. In this case, $*$ requires that the satisfiability of the property must be guaranteed considering the possibility that the source can send both TCP and UDP packets.

- **ProtectionInfo** are the set of security properties that a communication must satisfy:
  - *securityTechnology* is the technology to adopt in order to enforce the security requirement;
  - *authenticationAlgorithm* is the algorithm which must be used to enforce packet integrity and packet authentication;
  - *encryptionAlgorithm* is the algorithm which must be used to enforce packet encipherment.
- **TopologyInfo** nodes and links characterization:
  - *untrustedNodes* are the nodes in the graph on which the enforcement of the security requirements is mandatory;
  - *inspectorNodes* are the nodes in the graph on which the enforcement of the security requirements must be disabled;
  - *untrustedLinks* are the nodes in the graph on which the enforcement of the security requirements is mandatory.

5.2.3 Formulation of the Communication Protection Network Security Requirements

Let be **R** the set of the security requirements which the service designer gives in input to the system. For each requirement $r \in R$ zero or more untrusted nodes, inspector nodes or untrusted links can be specified; Furthermore it has to be considered that more than one network path could be available between the source and the destination hosts of a network security requirement, so could be necessary to operate on more than one flow in order to satisfy the requirement. So let be:

- $\upsilon(r)$ the set of untrusted nodes related to the requirement r;
- $\iota(r)$ the set of inspector nodes related to the requirement r;

- $\lambda(r)$ the set of untrusted links related to the requirement r;

- $\varphi(r)$ the set of flows related to the requirement r;

- $\pi(f)$ the set of nodes crossed by the flow f;

For the formulation of the security protection requirement two predicates has been defined:

- $PROTECT\ (n, f) \Longrightarrow Boolean \qquad with\ n \in N,\ f \in F$
- $UNPROTECT\ (n, f) \Longrightarrow Boolean \qquad with\ n \in N,\ f \in F$

where:

- N is the set of all network nodes;

- F is the set of all network flows.

The first function definition expresses that given a pair(node,flow) the PROTECT predicate could be active or not. The second function definition expresses that given a pair(node,flow) the UNPROTECT predicate could be active or not.

A set of Formulas exploiting these two function has been deployed:

$$\sum_{n_i \in \pi(f) | n_i < n} PROTECT(n_i, f) \quad > \sum_{n_i \in \pi(f) | n_i < n} UNPROTECT(n_i, f) \qquad (5.1)$$
$$\forall r \in R,\ \forall f \in \varphi(r), \forall n \in \pi(f) \wedge \in \upsilon(r)$$

$$\sum_{n_i \in \pi(f) | n_i < n} PROTECT(n_i, f) \quad = \sum_{n_i \in \pi(f) | n_i < n} UNPROTECT(n_i, f) \qquad (5.2)$$
$$\forall r \in R,\ \forall f \in \varphi(r), \forall n \in \pi(f) \wedge (n \in \iota(r) \vee n.dstAddr = r.dstAddr)$$

$$\sum_{n_i \in \pi(f) | n_i < l.dstNode} PROTECT(n_i, f) \quad > \sum_{n_i \in \pi(f) | n_i < l.dstNode} UNPROTECT(n_i, f) \qquad (5.3)$$
$$\forall r \in R,\ \forall f \in \varphi(r), \forall l \in \pi(f) \wedge \in \lambda(r)$$

Formula 1 states that given a protection security requirement, for each flow generated by the security requirement, for each nodes belonging to that path, if the node is untrusted: the sum of the predecessor nodes that perform a protection action must be higher than the sum of the predecessor nodes that perform a unprotection action. This constraint is necessary to assure that every time the traffic flows through an untrusted node the communication protection is active.

Formula 2 states that given a protection security requirement, for each flow generated by the security requirement, for each nodes belonging to that path, if the node is an inspector node or is the destination node: the sum of the predecessor nodes that perform a protection action must be equal to the sum of the predecessor nodes that perform a unprotection action. This is an indispensable constraint which allows the destination node and the inspector nodes to analyze the traffic without protection.

Formula 3 states that given a protection security requirement, for each flow generated by the security requirement, for each link belonging to that path, if the link is untrusted: considering the predecessors of the destination node of the link, the sum of nodes that perform a protection action must be higher than the sum of the nodes that perform a unprotection action. This constraint is required to guarantee that that every time the traffic passes through an untrusted link the communication protection is active.

5.2.4 Technologies and Capabilities Formulation

Comparing the information contained in the communication protection requirement model provided in the previous section with the information handled in the aforementioned formulas, It is possible to note that the three formulas written and explained above are not enough to satisfy a protection security requirement since that nothing is specified about the technologies and algorithms. A complete and detailed explanation about virtual private network technologies and the related capabilities is provided in the next chapter. In this section is presented the constraint which must be satisfied in order to guarantee that the nodes which start and terminate the secure channels support at least one of the requested technologies and the necessary capabilities.

For the formulation of the security protection requirement a function has been defined:

$$SUPPORT\_CAP\ (n, f, t, c) \Longrightarrow Boolean \qquad with\ n \in N,\ f \in F,\ t \in T,\ c \in C$$

where:

- N is the set of all network nodes;
- F is the set of all network flows;
- T is the set of all virtual private network technologies;
- C is the set of all virtual private network capabilities.

For each requirement $r \in R$ a list of virtual private network capabilities is specified, for example the authentication and encryption algorithms are two capabilities. Furthermore information about which technologies should be used to establish the secure communication could be included as well; So let be:

- $\tau(r)$ the set of virtual private network technologies specified in the requirement r;

- $\kappa(r)$ the set of virtual private network capabilities specified in the requirement r.

$$\begin{aligned} &\forall r \in R,\ \forall f \in \varphi(r), \\ &\forall n \in \pi(f) \wedge (PROTECT(n,f) \vee UNPROTECT(n,f)) \Longrightarrow \\ &\forall t \in \tau(r) \vee (\forall c \in \kappa(r) SUPPORT\_CAP(n,f,t,c)) \end{aligned} \tag{5.4}$$

Formula 4 states that given a protection security requirement, for each flow generated by the security requirement, for each nodes belonging to that path, if the node sets the PROTECT or the UNPROTECT function to TRUE It must support at least one security technology that exposes all the capabilities necessary to satisfy the security requirements.

Chapter 6

Channel Protection Systems Model

The first part of this chapter centers on the channel protection functions, i.e. those network security functions which consent to create a secure communication, describing them in a detailed way starting from their behavior to their configuration exploiting a medium level policy language.

The third section introduces the concepts of technology and capability, after defining them, the relation between these two concepts and the Channel Protection Systems is explained.

The fourth section deals with the possibility of specifying different profiles to express the preference, from the security administrator, about which of the two type of CPSs should be used to establish secure channels and others details related to their configurations in order to reach an optimal solution.

The last section focuses on the reachability assurance of the packets when a Packet Filter is positioned on the path between the source and the destination of a secure channel and as consequence, depending on its configuration, could generate a filtered channel anomaly.

6.1 Virtual Private Network Gateway

Since the secure communications are enforced through the cration of Virtual Private Networks the principal network security function modeled to enforce a secure communication is the Virtual Private Network Gateway. A VPN Gateway is a type of networking device which is able to provide or remove protection to the received traffic and forward it to the next node toward the final destination.

6.1.1 VPN Gateway Behavior

A VPN Gateway is able to operate in three different mode:

- ACCESS. Whether it receives traffic which needs to be protected;

- EXIT. Whether it receives traffic which has already been protected and need to be without protection from this point onward;
- FORWARD. Whether the received traffic need simply to be forwarded to the next node without providing or removing protection to it.

Let be:

- g a node which a VPN Gateway network function is installed on;
- f a flow passing through the network function g.

In order to model these behaviors the boolean predicates PROTECT and UNPROTECT defined in the previous chapter are utilized:

- $PROTECT(g, f) \wedge \neg(UNPROTECT(g, f))$

(ACCESS)

- $\neg(PROTECT(g, f)) \wedge (UNPROTECT(g, f))$

(EXIT)

- $\neg(PROTECT(g, f)) \wedge \neg(UNPROTECT(g, f))$

(FORWARD)

in other words it is possible to understand in which way the VPN Gateway operates for a certain flow by verifying the value of the PROTECT and UNPROTECT functions on the peer (g,f), where g is the considered VPN Gateway and f is the considered flow.

Another consideration that need to be done is that a VPN Gateway can choose only one of the three available behaviors for a certain pair (node,flow):

$$(ACCESS) \vee (EXIT) \vee (FORWARD)$$

6.1.2 Security Policy of a VPN Gateway

Each VPN Gateway is characterized by a Security Policy representing the configuration according to which this Network Security Function decides whether, for a received packet, protection must be added and which algorithms should be used or removed. The Security Policy can be established by the security administrator for a VPN Gateway which he decides to allocate in a specific position of the input Service Graph, if he has the necessary skills in the security field to determine it or if this policy comes as a result of a previous execution of the framework; otherwise, by default for each VPN Gateway which is automatically allocated on the Allocation Graph the ADP module provides a complete configuration. Let be P_A the set of the Allocation Places of an Allocation Graph, given $p_k \in P_A$ an Allocation Place where

a VPN Gateway can be allocated, the related Security Policy is characterized by a set of policy rules Ψ_k, which establish that a packet which matches their conditions should be managed with the corresponding behavior. The assumption is that all the policy rules are not redundant between themselves. When a packet is received by a VPN Gateway, firstly the conditions of each policy rule in the Ψ_k set are applied to the field of the packets; if the matching between a rule and the packet is positive, then the corresponding behavior, which can be ACCESS or EXIT, decides if protection must be added or removed from the packet. If no rule matches the fields of the packets, the packet will be simply forwarded without modify it.

Each rule in Ψ_k is characterized by the following model:

$$[behavior - startChannel - endChannel - Conditions - authAlg - encAlg]$$

where:

- **behavior** can have the values ACCESS or EXIT and specifies which behavior must be performed on each packet which satisfies the conditions of the rule;
- **startChannel** is the starting point of the secure channel;
- **endChannel** is the ending point of the secure channel;
- **Conditions** $[IPSrc - IPDst - portSrc - portDst - transportProto]$
  - *IPSrc* is the source IP address of the channel for which the action is applied;
  - *IPDst* is the destination IP address of the channel for which the action is applied;
  - *portSrc* is the transport-level source port of the channel for which the action is applied;
  - *portDst* is the transport-level destination port of the channel for which the action is applied;
  - *transportProto* is the transport-level protocol of the channel for which the action is applied;
- **authAlg** is the authentication algorithm with which authenticate each packet which satisfies the conditions of the rule;
- **encAlg** is the encryption algorithm with which encipher each packet which satisfies the conditions of the rule.

6.1.3 Tunneling

The network functions, which a network traffic could cross, can be classified in two categories based on the fact that the packets passing through them are being modified or not.

An example of network function which does not modify the packets passing through it, is the Packet Filter. This function analyzes the incoming packets and for each one of them decides whether drop or forward it to the next node, based on a set of internal rules. The fact to remark is that independently of which is his action on the packet (DROP or FORWARD), the packet filter will never modify it.

VNP Gateways instead belong to the second category, since an external header is added or removed to each packet before forward them to the next node.

When a VPN Gateway is the starting point of a channel, it adds an external header in which:

- the source address is its own ip address;
- the destination address is the address of the node that ends the channel.

When a VPN Gateway is the ending point of a channel, it removes the external header previously added and the more external addresses became the original ones again.

A clarification example is introduced below:

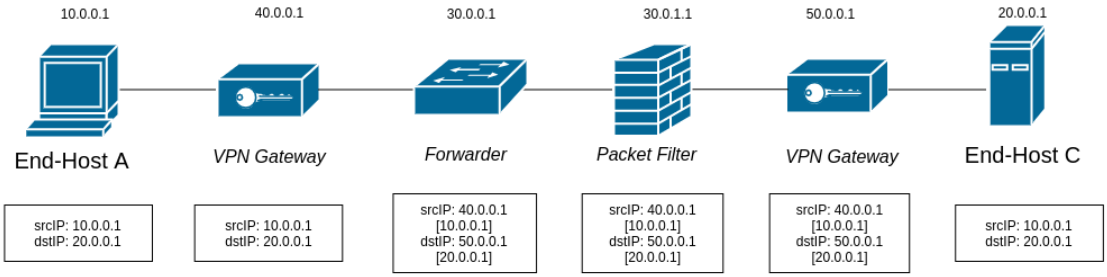


Figure 6.1. tunneling example

The figure 6.1 shows a chain of network devices a packet needs to cross in order to reach the destination. Under each box, representing a network device, a rectangle contains the addresses of the incoming packet.

The packet is generated by the end host on the left side and directed to the end host on the right side, it is possible to notice the addresses variation while flowing through the different network devices present in the path from the source to destination. Particularly when the packet comes in the VPN Gateway ACCESS the original addresses are still exploited for the forward operation, but before it exits and goes into the forwarder two new addresses are attached to the packet. They are the addresses of the external header added by the VPN Gateway ACCESS in order to provide protection. The addresses will became the original ones again after going out from the VPN Gateway EXIT in order to reach the destination without protection.

A relevant thing to take in consideration is that VEREFOO performs the automatic allocation of the VPN Gateways. This means that it is not possible to know a priori on which nodes of the allocation graph a VPN Gateway will be allocated, as consequence, the source tunnel address and the destination tunnel address become variables of the system dependent on which will be the nodes on which the VPN Gateways will be instanced.

Let be:

- Γ_{src} the tunnel source address;

- Γ_{dst} the tunnel destination address.

which a VPN Gateway g apply or remove from a packet belonging to a certain flow. Some hard constraints have been defined to correctly set the value of these address variables depending on the VPN Gateway behavior.

- ACCESS

  - the tunnel source address must be equal to the current VPN Gateway address:

    $$\Gamma_{src} = g.addr$$

  - Let be $\Phi(g)$ the set of the VPN Gateway after the considered VPN Gateway g in the flow path, the tunnel destination address must be equal to the address of the VPN Gateway contained in this set that behavies as VPN Gateway EXIT and the source tunnel address of the current VPN Gateway is equal to the source tunnel address of the selected VPN Gateway

    $$\begin{aligned}&(\Gamma_{dst} = VpnGw.addr) \wedge (VpnGw \in \Phi(g)) \wedge \\ &(VpnGw.\Gamma_{src} = g.\Gamma_{src}) \wedge (VpnGw.Behaviour = Exit)\end{aligned} \tag{6.1}$$

- EXIT

  - the tunnel source address must be equal to the original traffic source address:

    $$\Gamma_{src} = flow.SrcAddr$$

  - the tunnel destination address must be equal to the original traffic destination address:

    $$\Gamma_{dst} = flow.DstAddr$$

- FORWARD

  the packet is forwarded without modify the addresses.

6.2 End-Systems

In networking, end-systems are the devices which sit at the edge of the network and provides information or services. End-systems are usually connected to each other using various network devices rather than using a single communication link. The path that transmitted information takes from the sending end-system, through a series of communications links and network devices, to the receiving end-system is known as a route or path through the network.

Nowadays it is always more frequent the use of VPN technologies from end-hosts respect to the past. That is principally thanks to the hardware improvement in these devices. It is always worth to remark that the algorithms used to protect the packets in a secure communication are very CPU consuming, hence in the past years, where the resources were much more limited than today, was very rare implement such technology in the end-system.

An important work done in this thesis is the extension of the functionality of the end systems, enabling them to protect its own network traffic. A different model need to be introduced compared with the one used for the VPN Gateway since the end hosts are always positioned at the edge of the network.

6.2.1 End Host Behavior

Like a VPN Gateway an end host is able to operate in three different mode:

- ACCESS. Whether it generates traffic which needs to be protected;
- EXIT. Whether it receives traffic which has already been protected and need to remove the protection in order to see it in clear;
- FORWARD. Whether it generate traffic which need simply to be forwarded to the next node without providing protection.

Let be E a node which a end-host function is installed on, and f a flow $\in$ F generated by the rquirement r $\in R$. In order to model these behaviors, also in this case, the two functions PROTECT and UNPROTECT are necessary, but with the addition of some constraints:

ACCESS

$$PROTECT(E, f) \ \wedge \neg(UNPROTECT(E, f)) \wedge \\ E.hasVPNCapability \ \wedge E.srcAddr = r.srcAddr \tag{6.2}$$

EXIT

$$\neg(PROTECT(E, f)) \ \wedge UNPROTECT(E, f) \wedge \\ E.hasVPNCapability \ \wedge E.dstAddr = r.dstAddr \tag{6.3}$$

FORWARD

$$\neg(PROTECT(E, f)) \ \wedge \neg(UNPROTECT(E, f)) \tag{6.4}$$

- Formula 6 states that an End-Host can protect only the traffic which itself generates and in order to do this must support VPN Capabilities;
- Formula 7 states that an End-Host can remove the protection only to the traffic destinated to it and in order to do this must support VPN Capabilities;
- Formula 8 states that no VPN Capabilities are needed if neither protection nor unprotection is performed.

Because of what explained before about the resources available in the end-host, the predicate hasVPNCapability has been introduced. It allows to specify if a certain end-host has or not the capacity of performing channel protection.

6.3 Channel Protection Systems Technologies and Capabilities

6.3.1 Virtual Private Network Technologies

The term VPN Technology refers to the standard utilized in order to implement a secure communication. Nowaday it exists many technologies which can be exploited in order to implement a virtual private network.

A network administrator, as explained in the previous chapter, could prefer a certain technology rather than another to protect the traffic, because each one of them has its own characteristics. For this purpose a catalog has been created in which are present all the security technologies supported by the framework, thus for each network security requirement one or more VPN technologies present in the catalog can be specified. The solver will try to find a solution using one of those technologies; whether no VPN technologies are specified for a certain requirement the solver considers all the technologies present in the catalog to satisfy the requirement.

In the following are listed the most common parameters used to evaluate the VPN technologies:

- *performance*: some VPN technologies perform, although having the same settings, better than others;
- *security*: some technologies suffer from security doubts. For example IPsec suffers to replay attack, whose risk depends on how large is the replay window; anyway this situation could lead to the network administrator to select another VPN technology.
- *network layer*: a virtual private network can be created at various level of the TCP/IP protocol stack and different technologies works at different layers. The network administrator has to select a VPN technologies according to the network layer he wants to create a virtual private network. For example IPsec works at Nwtwork (L3) layer while L2TP at Data Link (L2) layer.

- *capability*: this is perhaps the most influential parameter, i.e. the capabilities owned by a VPN technology, because whether a network administrator needs to create a secure communication with a certain set of capabilities he will be incline to select a VPN technology which exposes all the requested capabilities.

For this thesis work the following VPN Technologies has been considered:

- IPSEC
- TLS

because are the technologies which:

- offer the maximum level of security;
- have great performance;
- are the most popular VPN Technologies;
- have a very large set of capabilities.

6.3.2 Virtual Private Network Capabilities

A capability is a feature, which an entity hold and is able to utilize when requested. Entering in the field of the virtual private network the most relevant capabilities are the encryption and authentication algorithms, i.e. be able to authenticate the data with a certain algorithm is a capability.

Capabilities and Technologies

Each technology exposes a set of capabilities, i.e. supports a number of features related to this ambit.

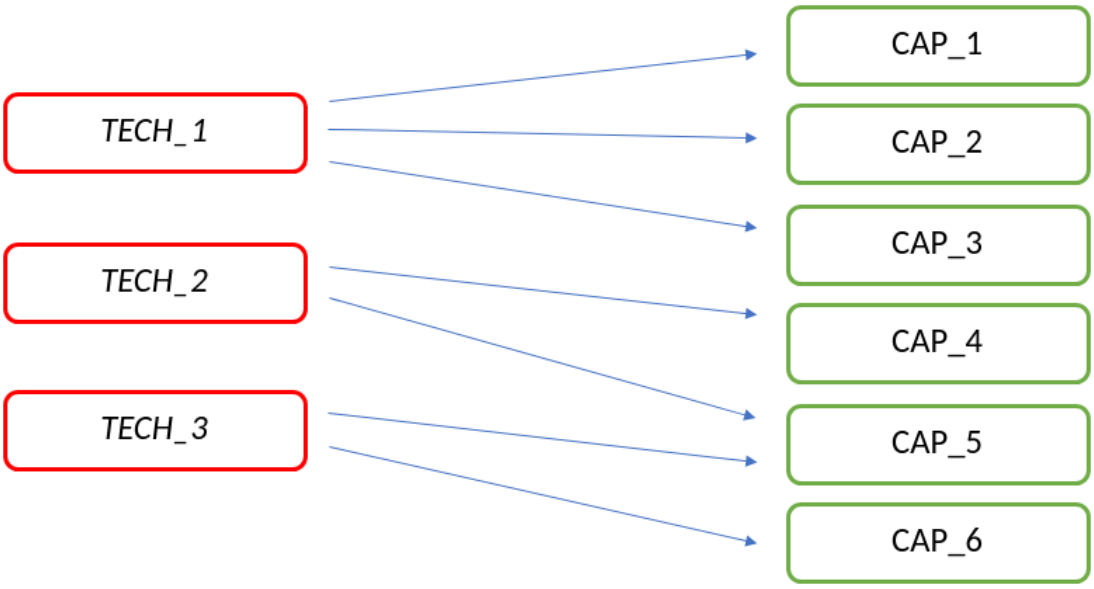


Figure 6.2. virtual private network technologies and capabilities example

The figure shows three VPN technology:

- TECH_1 which exposes the capabilities CAP_1, CAP_2 and CAP_3;
- TECH_2 which exposes the capabilities CAP_4 and CAP_5;
- TECH_3 which exposes the capabilities CAP_5 and CAP_6.

A node could supports zero or more VPN technologies and inherits all the capabilities exposed by the supported technologies, this means that the set of technologies, a node is able to implement, affects the decision process of the solver.

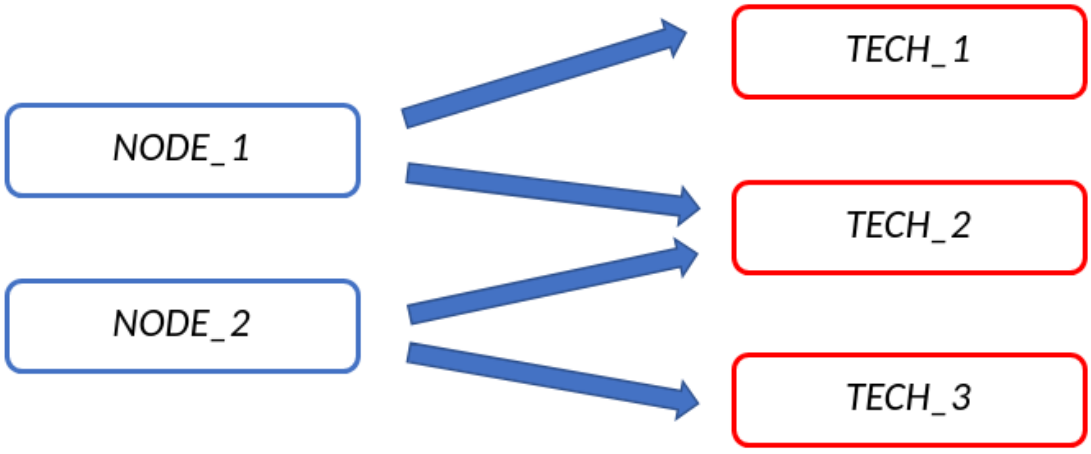


Figure 6.3. virtual private network technologies and capabilities example 2

The figure shows two nodes:

- NODE_1 which supports the technologies TECH_1 and TECH_2;
- NODE_2 which supports the technologies TECH_2 and TECH_3.

Referencing to the figures depicted above, two clarification examples are presented in the following:

1 Let suppose that that the network administrator specifies a security requirement where the capability CAP_4 is needed; by observing the figures it is possible to note that the only technology which exposes the CAP_4 is the TECH_2, this leads the solver to force NODE_1 and NODE_2 to utilize the TECH_2 in order to satisfy the requirement.

2 Let suppose that that the network administrator specifies a security requirement where the capability CAP_6 is needed; by observing the figures it is possible to note that the only technology which exposes the CAP_6 is the TECH_3 and that the NODE_2 does not support this technology, hence NODE_2 can not be taken into consideration by the solver to satisfy this security requirement.

Capabilities Model and End Host

Each end host can support zero or more VPN Technologies present in the VEREFOO VPN Technology catalog and, as explained in the previous section, it inherits

all the capabilities exposed by the supported technologies. Whether no security technologies are specified but the node has VPN capabilities the system considers all the technologies contained in the catalog to satisfy the security requirements. So let be:

- $\varphi(E)$ the set of flows passing through the end host E;
- C the Catalog which contains all the VPN technologies supported by VEREFOO ;
- $\varepsilon(E)$ the set of VPN technologies supported by the end host E;
- $\varsigma(st)$ the set of capabilities exposed by the VPN technology st.

Following the principles previously explained the next two formulas state that if a node supports a VPN Technology st supports all the capabilities c exposed by it, else it is not able to use those capabilities.

$$\begin{aligned} &\forall f \in \varphi(E), \forall st \in C \\ &E.support(st) \implies \forall c \in \varsigma(st) \quad SUPPORT\_CAP(E, f, st, c) \end{aligned} \tag{6.5}$$

$$\begin{aligned} &\forall f \in \varphi(E), \forall st \in C \\ &\neg(E.support(st)) \implies \forall c \in \varsigma(st) \quad \neg(SUPPORT\_CAP(E, f, st, c)) \end{aligned} \tag{6.6}$$

Capabilities Model and VPN Gateway

The approach followed in the modeling of the VPN Gateway Capabilities is different from the one presented above for the End Host, since the VPN Gateway functions are created and deployed exclusively with the purpose of create secure communication. For this reason and also in order to keep the model complete but simple it can be made the assumption that each VPN Gateway supports all the technologies contained in the Catalog of VEREEFOO and, as consequence, supports all the VPN capabilities exposed by those technologies

So let be:

- $\varphi(G)$ the set of flows passing through the VPN Gateway G;
- C the Catalog which contains all the VPN technologies supported by VEREFOO ;
- $\varsigma(st)$ the set of capabilities exposed by the VPN technology st.

$$\begin{aligned} &\forall f \in \varphi(G), \forall st \in C \\ &\forall c \in \varsigma(st) SUPPORT\_CAP(E, f, st, c) \end{aligned} \tag{6.7}$$

6.4 Profiles

Since this project deals with a Max SMT problem, another important step of this thesis work has been the detection of the parameter to consider in order to reach the optimal solution; these parameters are strictly related to the Channel Protection Systems and are presented in the following:

1] the minimization of the number of Channel Protection Systems. This first objective, due to the fact that there are 2 types of CPSs, can be divided in two sub-cases:

   a) the minimization of the number of VPN Gateway which are allocated in the Allocation Graph, so that the resources consumption are minimized when the VNFs implementing the VPN Gateways are deployed on a substrate network, independently from the actual requirements of each virtual instance (e.g. RAM, CPU) which are not considered in the defined approach because already managed by another existent part of the ADP module;

   b) the minimization of number of End-Host which uses VPN Capabilities, also in this case the minimization lead to less consumption in terms of resources, though in a minor key because the allocation of the End-Host will be performed independently of this.

2] the minimization of the traffic which is protected by a CPS, i.e. protect only the traffic which needs to be protected according to the security requirements. The encipher operations performed by a VPN Gateway or by an End-Host with VPN capabilities are not trivial in terms of resources consumption, CPU in particularly, thus reduce the number of packets enciphered to a minimum will decrease consequently the CPU overhead and will improve the system scalability.

From a theoretical point of view the objectives **a** and **b** are strictly related, because in order to satisfy a security requirement, the solver, tries to allocate a VPN Gateway or to use the capabilities of an End-Host; while both are independent from the objective **2**.

An important issue is represented by the assignation of the priority to each objective, because as explained in the chapter 4 each soft constraint has a priority and the solver tries to reach the optimal solution by giving the precedence to the soft constraint with the highest priority. The highest priority is assigned to the the objective 1, i.e. the minimization of the number of Channel Protection Systems, since deploying an additional function is much more expensive than an increase in the traffic to be encrypted. By referring to what written before the objective 1 has been divided into two sub-objective and giving the highest priority to **a** or **b** cause the generation of 2 different solutions. To better understand let consider the simple Allocation Graph depicted below:

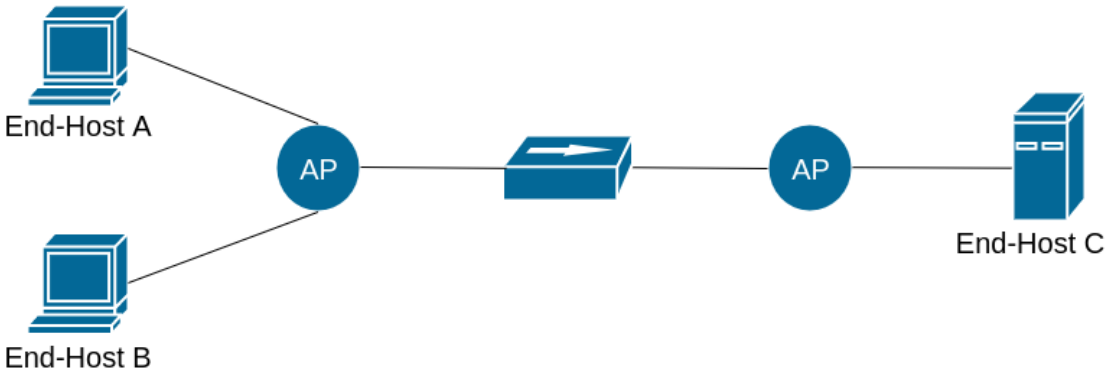


Figure 6.4. Allocation Graph

Supposing that two communication protection requirements are specified stating that the traffic flowing from the end-host A and the end-host B to the end-host C must be protected, giving the highest priority to the objective **b** produces the following solution:

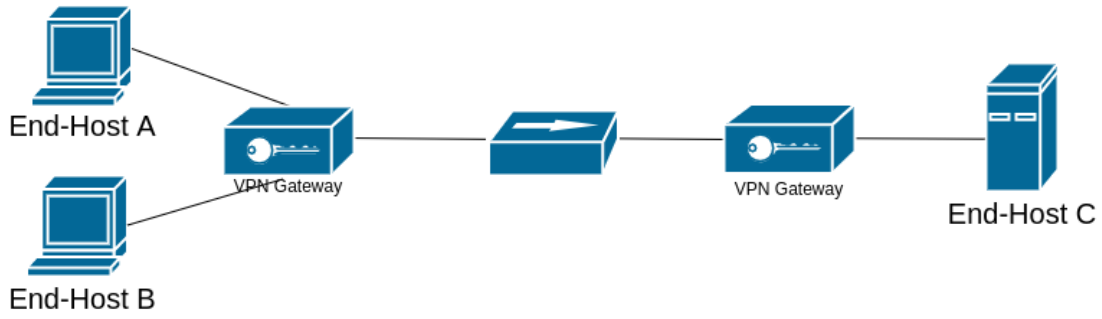


Figure 6.5. solution 1

here two VPN Gateway are allocated in order to satisfy the communication protection requirements and the end-hosts do not provide any protection. While giving the highest priority to the objective **a**:

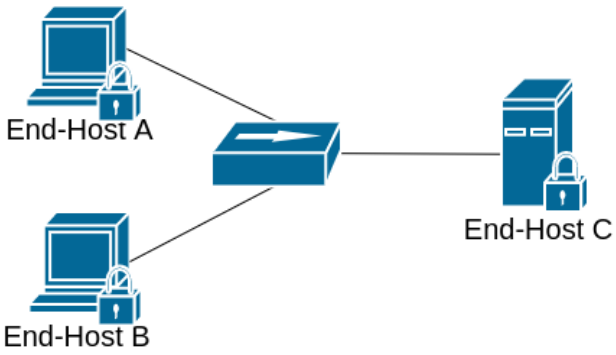


Figure 6.6. solution 2

in this solution no VPN gateway are allocated and the requirements are satisfied through the creation of two end-to-end channels.

Both the solution are valid and could be preferred by the security administrator depending on his needs, for this reason it has been taken the decision to not fix the priority values of these soft constraints, but to give the possibility to the security administrator to choose form 2 different profiles. It is worth to highlight that the proposed profiles depend on the metric selected to find on optimal solution and by the priority given to each metric. So different profiles could be implemented considering different optimality criteria and/or different priority.

The profiles proposed in this work are:

- **Green**
- **Stable**

6.4.1 Green Profile

Nowadays the concept of green computing or green ICT is increasingly widespread, one of the most relevant objective of green computing is maximize energy efficiency during the product's lifetime. Data center facilities are heavy consumers of energy, and the optimized use of Information technology (IT) systems belonging to it can lead to a significantly reduction of the energy consumption. The objectives of the green profile fit perfectly with the green ICT paradigm, in fact the main purpose is use the minimum amount of resources in terms of both allocation and computation resources. These goals are achieved by:

- using the End-Host VPN capabilities when possible and allocating a VPN Gateway if and only if it is strictly necessary to the satisfaction of at least one security requirement, in order to reduce the allocated functions, and the consumation of cpu and ram necessary for the deployment of these functions.

- use the minimum of CPU power, the solver will protect only the traffic that need to be protected creating ad hoc Security Associations for each flow in order to reduce at minimum the protected traffic.

Considering the example showed above the solution 2 would be produced.

.

6.4.2 Stable Profile

The network administrator, although green ICT is a noble concept, could prefer a more stable solution at the expense of resource optimization. This kind of solution tries to demand the creation of secure communication to specialized functions created exclusively for this purpose avoiding end host doing that. The objectives of the stable profile are:

- Apply the minimum number of CPSs necessary to satisfy the communication protection requirements, but prefer the allocation of VPN Gateways to the use of end-hosts with VPN capabilities to satisfy the requirements.
- Use the minimum of CPU power: the solver will protect only the traffic that need to be protected creating ad hoc Security Associations for each flow in order to reduce at minimum the protected traffic.

Considering the example showed above the solution 1 would be produced and on each VPN Gateway 2 different SA would be established one for each security requirement.

6.5 Channel Protection Systems and Packet Filter

The packet filter is a technology of firewall which is able to make a decision about forwarding each incoming packet basing it exclusively on IP addresses and ports of both source and destination, information collocated respectively in the layer three (network) and four (transport) of the ISO/OSI stack. Its main advantages are that it is easy to implement, it is application independent and overall its performance is very good because it is able to avoid incoming Denial-Of-Service attacks processing the packets in the first levels of the networking stack of the Operating System. A packet filter is one of the most important and used Network Security Function inside a network, because the screening router on which it is installed can easily drop most of the packet belonging to prohibited communication, unburdening more complex appliances like an Application-Level Gateway or a Deep Packet Inspector.

Each firewall is characterized by a Filtering Policy representing the configuration according to which this Network Security Function decides if a received packet must be discarded or should be forwarded to the out-ports which allow to reach its destination.

The Filtering Policy is characterized by two components:

- a default action Ω ;
- a set of policy rules Ψ , which establish that a packet which matches their conditions should be managed with the corresponding actions.

When a packet is received by a packet filter, firstly the conditions of each policy rule in the Ψ set are applied to the field of the packets; if the matching between a rule and the packet is positive, then the corresponding action – which can be ALLOW or DENY – decides if the packet can be forwarded or must be discarded. If no rule matches the fields of the packets, it is managed according to the default action Ω , which has less priority then all the other rules. The default action Ω can be assigned two different values:

- Ω = DENY, if the firewall configuration is whitelisting;
- Ω = ALLOW, if the firewall configuration is blacklisting.

The work of a Packet Filter could in same cases make useless the secure channels created by some VPN Gateways or End-Hosts with VPN Capabilities. Entering into details whether a Packet Filter is positioned on the path between the source and the destination of a secure channel and it is configured in the way to drop the packets belonging to that communication, the packets even if protected will not be able to reach the destination!

So another ability which has been given to the solver, through the addition of some hard constraints, is the capacity to find a solution that not only guaranties the protection of the packets flowing from the source to the destination, according to the security requirements, but also guaranties that those packets will reach the destination when a Packet Filter is placed somewhere in the path from the source to the destination. Formalizing what explained above let be:

- $\varphi(pf)$ the set of flows passing through the Packet Filter pf;
- $\Phi(pf)$ the set of rules configured on the Packet Filter pf;
- $\pi(f)$the security requirement associated to the flow f.

$$\begin{aligned} &\Omega = ALLOW \implies \forall f \in \varphi(pf), \forall r \in \Phi(pf) \\ &r.srcAddr \neq T.srcAddr \vee r.dsrAddr \neq T.dstAddr \end{aligned} \tag{6.8}$$

The formula 11 states that if the default action configured on the Packet Filter is $ALLOW$ and consequently the default packet filter behavior is *blacklisting* on the considered packet filter can not exist a rule which drops the traffic related to a security requirement.

$$\begin{aligned} &\Omega = DENY \implies \forall f \in \varphi(pf), \forall r \in \Phi(pf) \\ &\neg(r.srcAddr \neq T.srcAddr \vee r.dsrAddr \neq T.dstAddr \end{aligned} \tag{6.9}$$

The formula 12 states that if the default action configured on the Packet Filter is *DENY*, hence the default behavior is *whitelisting*, on the considered packet filter must exist at least one rule which let the traffic pass in order to reach the destination.

A clarification example on how the solver finds an alternative solution depending on where the packet filter is allocated and which is his configuration is provided below. Let consider the following allocation Graph:

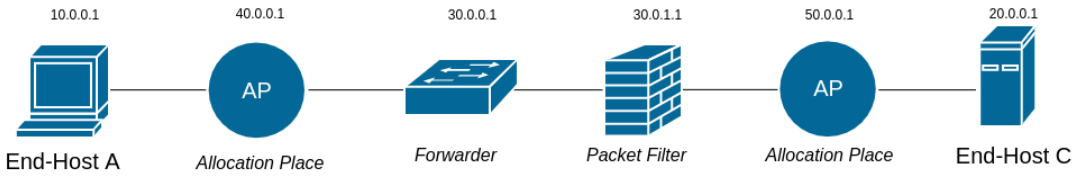


Figure 6.7. Allocation Graph

with the packet filter manually configured according to the figure depicted below:

Packet Filter Configuration

Default behavior: *blacklisting*

Action	***From***	***To***	***SrcPort***	***DstPort***	***L4Protocol***
Deny	*40.0.0.1*	*50.0.0.1*	*any*	*any*	*any*

Figure 6.8. packet filter manual configuration

and the following network security requirement:

Security Requirement

Protect the traffic from 10.0.0.1 to 20.0.0.1 with the encryption algorithm AES_128_CBC and the authentication algorithm SHA2_256.
Untrusted node {30.0.0.1}

Figure 6.9. network security requirement

Furthermore let suppose that:

- the End-Hosts are able to use VPN capabilities;
- the profile chosen by the network administrator is the stable profile, this means that the solver tries to find a solution in which VPN Gateway are allocated in order to satisfy the network security requirements rather than to use End-Host VPN capabilities.

Without the addition of the two formulas explained in the previous page, the solution would be:

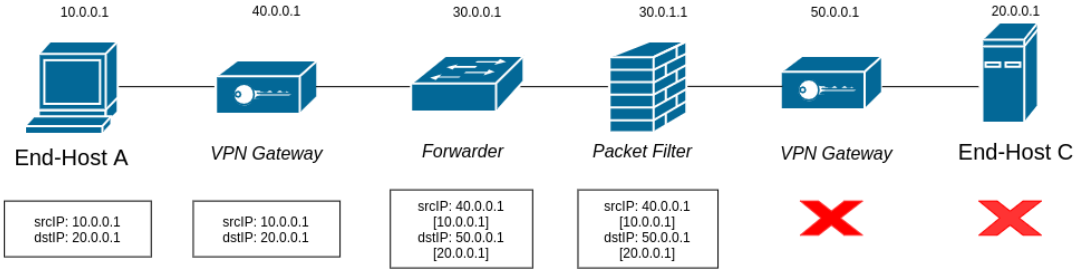


Figure 6.10. solution 1

In this case the solver gives as output a solution without taking into consideration the presence of a Packet Filter on the path and his configuration; as a matter of fact, unfortunately, the packets generated by the End-Host 10.0.0.1 will never reach the node 20.0.0.1 because of the Packet Filter.

By considering the 2 formula one of the possible solution could be:

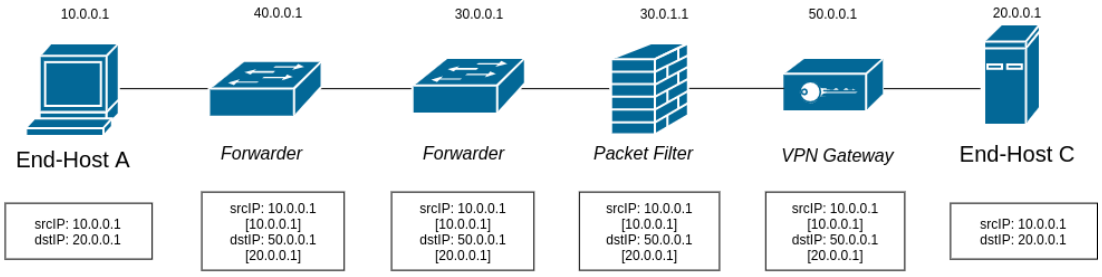


Figure 6.11. solution 2

In this case the solver gives as output a solution which takes into consideration the presence of a Packet Filter on the path and the fact that the rule configured on it denies the traffic from 40.0.0.1 to 50.0.0.1; thus in order to guarantee reachability, as well as security, it is forced to propose a solution which crates a channel whose addresses pair (source,destination) is different from (40.0.0.1, 50.0.0.1).

In the next chapter details about the implementation of the presented models are provided; moreover various use cases, to better understand the overall working of the ADP module and all the explained features, are showed.

Chapter 7

Implementation and Validation

In the first section of this chapter are provided details about the implementation of the ADP module, and the solver used to resolve the maxSMT problem. The section two is characterized by two use Cases with the goal of showing through practical examples all the features and characteristics of the ADP module presented in the previous chapters. Finally, In the last part a series of performance tests are presented to evaluate the scalability of the implemented solution.

7.1 Implementation

This work has been entirely developed using Java 11, the Z3 Java library to implement the partial weighted MaxSMT problem and the XML markup language to implement the structure of the communication protection requirements, the VPN Gateway and the CPSs configurations.

7.1.1 XML schema of the Communication Protection Network Security Requirements

In the XML schema the section regarding the network security requirements is named *propertyDefinition* and can contain from 1 to n *Property* elements, each one of them representing a single security requirement specified by the security administrator. The property element contains a set of attributes necessary to specify the ruleType and the FilteringCondition:

- **name** it is the kind of requirement (i.e. protection);
- **src** it is the identifier (e.g. IP address) of the source node of the requirement;
- **dst** it is the identifier (e.g. IP address) of the destination node of the requirement;
- **src_port** it is the number or interval of numbers for the source port;
- **dst_port** it is the number or interval of numbers for the destination port;

- **l4_proto** it is the type of layer 4 protocol (i.e. TCP or UDP).

Furthermore includes a nested element which carries all the protection and topology information:

- **untrustedNode** contains all the untrusted nodes of the requirement;
- **inspectorNode** contains all the inspector nodes of the requirement;
- **untrustedLink** contains all the untrusted links of the requirement;
- **securityTechnology** contains the security technologies which can be used to enforce the requirement;
- **authenticationAlgorithm** it is the authentication algorithm;
- **encryptionAlgorithm** it is the encryption algorithm.

Two examples are presented in the following:

1] The first example shows a single security requirement in which all the traffic between the node 10.0.0.1 and the node 20.0.0.1 must be enciphered with the encryption algorithm AES_128_CBC and authenticated with the authentication algorithm SHA_256 when passing through the nodes 30.0.0.1 and 30.0.0.2, the solver in order to satisfy this requirement can exploit both TLS and IPSEC technologies;

```
<PropertyDefinition>
      <Property graph="0" name="ProtectionProperty" src="10.0.0.1"
                                                    dst="20.0.0.1">
          <protectionInfo encryptionAlgorithm="AES_128_CBC"
                          authenticationAlgorithm="SHA2_256">
              <untrustedNode node="30.0.0.1"></untrustedNode>
              <untrustedNode node="30.0.0.2"></untrustedNode>
              <securityTechnology>TLS</securityTechnology>
              <securityTechnology>IPSEC</securityTechnology>
          </protectionInfo>
      </Property>
</PropertyDefinition>
```

Figure 7.1. network security requirement example 1

2] The second example shows two security requirements. The first one states that all the traffic between the node 10.0.0.1 and the node 20.0.0.1 must be enciphered with the Encryption default algorithm and authenticated with the authentication default algorithm when passing through the node 30.0.0.2, the solver in order to satisfy this requirement can exploit both TLS and IPSEC technologies; The second one states that all the traffic between the node 10.0.0.2 and the node 20.0.0.1 must be enciphered with the Encryption default algorithm AES_256_CBC and authenticated with the authentication

```
<PropertyDefinition>
      <Property graph="0" name="ProtectionProperty" src="10.0.0.1"
                                                    dst="20.0.0.1">
          <protectionInfo >
              <untrustedNode node="30.0.0.2"></untrustedNode>
              <securityTechnology>TLS</securityTechnology>
              <securityTechnology>IPSEC</securityTechnology>
          </protectionInfo>
      </Property>
      <Property graph="0" name="ProtectionProperty" src="10.0.0.2"
                                                    dst="20.0.0.1">
          <protectionInfo encryptionAlgorithm="AES_256_CBC"
                          authenticationAlgorithm="SHA2_512">
              <untrustedNode node="30.0.0.1"></untrustedNode>
              <inspectorNode node="30.0.0.2"/>
          </protectionInfo>
      </Property>
</PropertyDefinition>
```

Figure 7.2. network security requirement example 2

algorithm SHA_512 when passing through the node 30.0.0.1, but must be without protection when passing through the node 30.0.0.2.

It is worth noting that :

- In the first requirement the encryption algorithm and the authentication algorithm are missing. When this situation occurs the solver uses the algorithms selected as default in order to assure the traffic protection also in this case; To disable the data encryption or the data authentication the keyword is NULL;
- the node 30.0.0.2 is *Untrusted* for the first requirement and *Inspector* for the second one. This is to underline the independence among security requirements;
- In the second security requirement there are no security technologies specified, this means that the solver tries to satisfy the requirement with all the VPN technologies contained in the catalog.

7.1.2 XML Schema of CPSs

A VPN Gateway is represented in the XML schema by a node element whose functional type attribute has VPN_GATEWAY as value. In addition to the traditional internal elements of a node, such as the set of neighbours elements, a VPN Gateway is also characterized by a specific configuration which contains a *vpnGateway* element. A *vpnGateway* element has a set of *SecurityAssociations*, representing the Security Policy rules and expressed by means of a medium-level representation. Each security association is characterized by the elements contained in the figure depicted below:

```
<xsd:complexType name="SecurityAssociationType">
    <xsd:sequence>
    <xsd:element name="behavior" type="BehaviorTypes"
                minOccurs="0" />
        <xsd:element name="startChannel" type="xsd:string"
            minOccurs="0" />
        <xsd:element name="endChannel" type="xsd:string"
            minOccurs="0" />
        <xsd:element name="source" type="xsd:string" />
        <xsd:element name="destination" type="xsd:string" />
        <xsd:element name="protocol" type="L4ProtocolTypes"
            minOccurs="0" default="ANY" />
        <xsd:element name="src_port" type="xsd:string"
            minOccurs="0" />
        <xsd:element name="dst_port" type="xsd:string"
            minOccurs="0" />
        <xsd:element name="authenticationAlgorithm"
            type="AuthenticationAlgorithmType" minOccurs="0" />
        <xsd:element name="encryptionAlgorithm"
            type="EncryptionAlgorithmType" minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>
```

Figure 7.3. security association xml element

An example of a configured VPN Gateway is presented in the following:

```
<node functional_type="VPNGateway" name="40.0.0.1">
    <neighbour name="10.0.0.1" />
    <neighbour name="30.0.0.1" />
    <configuration description="vpn gw access"
        name="conf1">
        <vpnGateway>
            <securityAssociation>
                <behavior>ACCESS</behavior>
                <startChannel>40.0.0.1</startChannel>
                <source>10.0.0.1</source>
                <destination>20.0.0.1</destination>
                <protocol>ANY</protocol>
                <src_port>*</src_port>
                <dst_port>80</dst_port>
                <authenticationAlgorithm>SHA2_256</authenticationAlgorithm>
                <encryptionAlgorithm>AES_128_CBC</encryptionAlgorithm>
            </securityAssociation>
        </vpnGateway>
    </configuration>
</node>
```

Figure 7.4. VPN Gateway configuration example

If the packets coming in the VPN Gateway 40.0.0.1 have:

- the source address equal to 10.0.0.1;
- the destination address equal to 20.0.0.1;
- any source port;
- the destination port equal 80;
- any level 4 protocol;

since the behavior is ACCESS, they will be protected with the authentication algorithm SHA2_256 and the encryption algorithm AES_128_CBC and forward to the next node.

Regarding the End-Host with VPN Capabilities, as for the VPN Gateways, a set of *SecurityAssociations*, representing the Security Policy rules and expressed by means of a medium-level representation can be specified, where each SA follow the schema presented in the figure 7.3.

7.1.3 XML Schema of VPN Technologies and Capabilities

Two main catalog has been implemented using the XML language:

- the first one containing all the supported technologies

```
<xsd:simpleType name="SecurityTechnologyType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="IPSEC" />
        <xsd:enumeration value="TLS" />
    </xsd:restriction>
</xsd:simpleType>
```

- the second one containing all the known capabilities;

```
<xsd:simpleType name="Capability">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NULL" />
        <xsd:enumeration value="TriploDES" />
        <xsd:enumeration value="AES_128_CBC" />
        <xsd:enumeration value="AES_192_CBC" />
        <xsd:enumeration value="AES_256_CBC" />
            :
        <xsd:enumeration value="CAMELIA_128_CCM" />
        <xsd:enumeration value="CAMELIA_192_CCM" />
        <xsd:enumeration value="CAMELIA_256_CCM" />
        <xsd:enumeration value="chacha20poly1305" />
        <xsd:enumeration value="SHA1" />
            :
        <xsd:enumeration value="SHA2_512" />
        <xsd:enumeration value="AES_XCBC" />
        <xsd:enumeration value="AES_CMAC" />
    </xsd:restriction>
</xsd:simpleType>
```

Furthermore for each supported technology a catalog with the all supported technologies has been created, for instance for the IPSEC technology:

```
<xsd:simpleType name="IPSECcapability">
    <xsd:restriction base="Capability">
        <xsd:enumeration value="SHA1" />
        <xsd:enumeration value="SHA2_256" />
        <xsd:enumeration value="SHA2_512" />
            :
        <xsd:enumeration value="AES_128_CBC" />
        <xsd:enumeration value="AES_256_CBC" />
    </xsd:restriction>
 <xsd:simpleType>
```

This approach allows the system to support new VPN technologies and capabilities in a very simple way, in fact to update the framework it is enough to add the new features to the proper catalogs. It is worth to note that it is very frequent that a certain technology has a new capability, for instance a new authentication algorithm, when a new version of the considered technology is available and thanks to this approach it is easy maintain the system updated.

7.1.4 MaxSMT Problem

Z3 is a Satisfiability Modulo Theories (SMT) solver from Microsoft Research which is targeted at solving problems that arise in software verification and software analysis contexts. Z3 became open source under an MIT license. It supports Windows, OSX, Linux and FreeBSD .It is possible also call Z3 procedurally by using a variety of APIs available in C, C++, Java, .Net, OCaml and Python. In this thesis work Java APIs have been used in the version 4.8.8 for 64-bit machines.

The workflow of the framework is presented in the following figure.

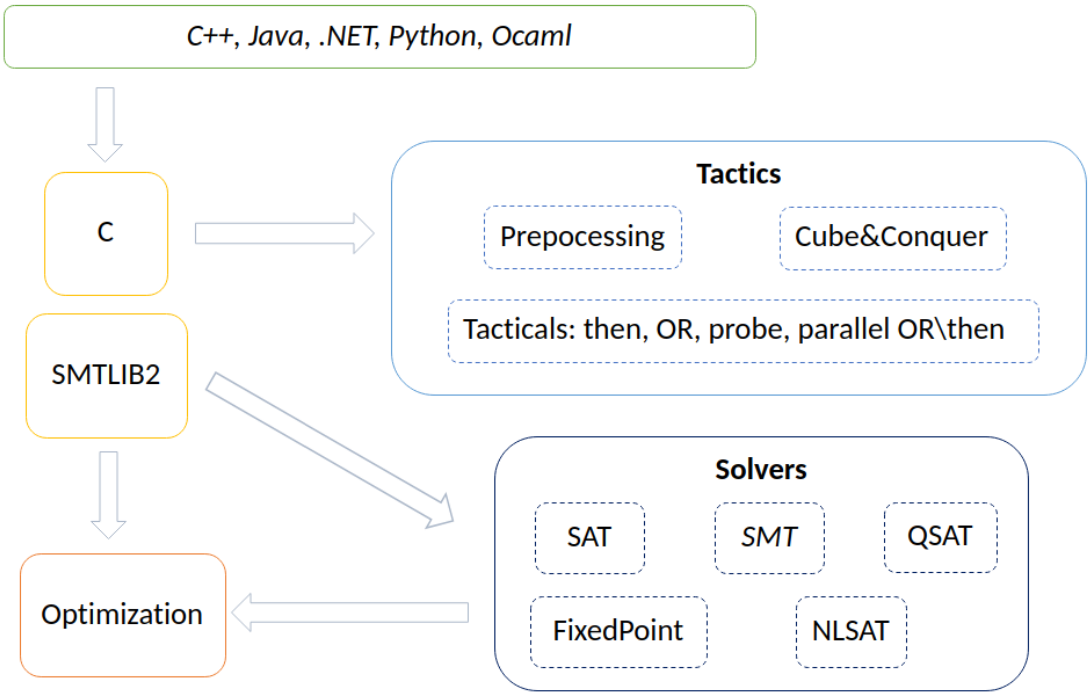


Figure 7.5. Z3 workflow

- A set of formulas are provided in input to the framework Using one of the programming language supported by Z3.

- A one-to-one mapping from the data domain of the program being tested to the data domain of Z3 is performed, namely all the these formulas are translated in a SMTLIB2 file.

- Z3 tries to apply tactics like pre-processing in order to limit the total computation time or heuristics to get a sub-optimal solution.

- Finally, it selects a specific solver (e.g. SMT) to compute a solution or whether an optimization phase is needed, to compute the optimal solution; for the optimization phase, in particular, z3 exploits an optimizer engine, called z3Opt.

In the following is presented an example of weighted partial MaxSMT modelled in Z3 language and the corresponding solution:

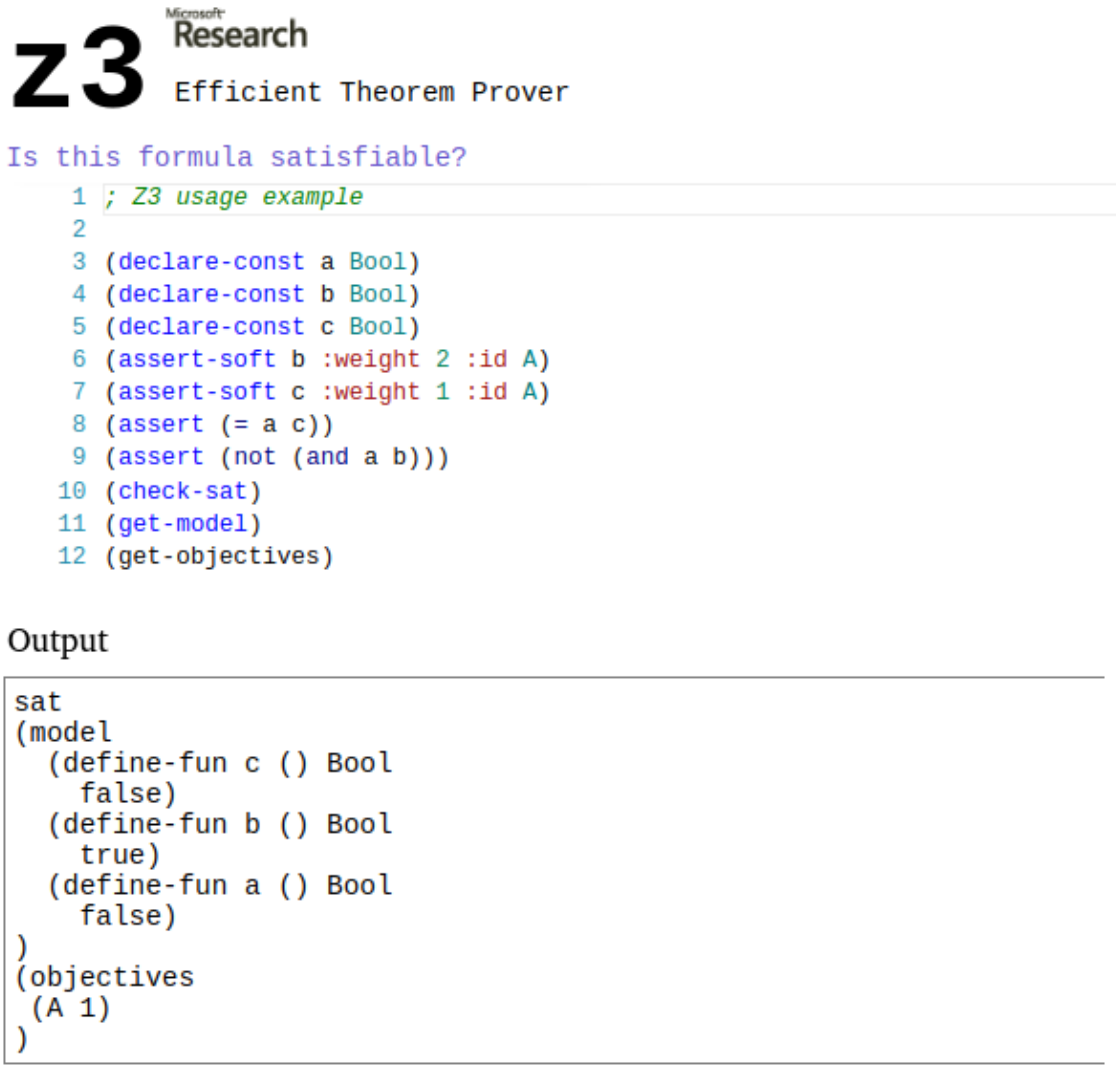


Figure 7.6. Z3 example

In the first half of the picture is shown the formula provided in input to Z3. It assert that the boolean variable a must be equal to the boolean variable c and that a and the boolean variable b cannot be true at the same time. Considering these hard constraints 2 solutions exist: the first one with a and c equal to true and the b equal to false and the second one with a and c equal to false and b equal to true. Since also 2 soft constraints are present which declare that the variable b and c, if possible, should be true but with different weight, Z3 gives in output the second one and it is represented in the second half of the figure.

In the following some examples are provided in order to show how the Z3 java library has been exploited to set up the MaxSMT problem:

the first image shows the declaration of the protect function:

```
public FuncDecl protect;

protect = ctx.mkFuncDecl("protect", new Sort[]{ nodeType, ctx.mkIntSort()},ctx.mkIntSort());
```

Figure 7.7. Function declaration example

the code specifies that a predicate named protect is declared and can assume a different value for each pair [nodeType,Int] where the nodeType represents a node in the considered topology and the int type represents the FlowId of the considered flow; moreover it can be notice that, differently than expected, it is declared as int and not as boolean, this is just because in the formulations about the satisfaction of the security requirements the values of this predicate need to be summed, anyway it never assumes a value different from 0 or 1.

The second image shows the setting up of the hard constraints related to the VPN Gateway Behaviors:

```
VPN GATEWAY ACCESS BEHAVIOR
BoolExpr opt1 = ctx.mkAnd( ctx.mkEq(nctx.protect.apply(source.getZ3Name(), ctx.mkInt(flow.getIdFlow())), ctx.mkInt(1)),
        ctx.mkEq(nctx.disProtect.apply(source.getZ3Name(), ctx.mkInt(flow.getIdFlow())), ctx.mkInt(0)) );
VPN GATEWAY EXIT BEHAVIOR
BoolExpr opt2 = ctx.mkAnd( ctx.mkEq(nctx.protect.apply(source.getZ3Name(), ctx.mkInt(flow.getIdFlow())), ctx.mkInt(0)),
        ctx.mkEq(nctx.disProtect.apply(source.getZ3Name(), ctx.mkInt(flow.getIdFlow())), ctx.mkInt(1)) );
VPN GATEWAY NOT USED BEHAVIOR
BoolExpr opt3 = ctx.mkAnd( ctx.mkEq(nctx.protect.apply(source.getZ3Name(), ctx.mkInt(flow.getIdFlow())), ctx.mkInt(0)),
        ctx.mkEq(nctx.disProtect.apply(source.getZ3Name(), ctx.mkInt(flow.getIdFlow())), ctx.mkInt(0)) );

BoolExpr behaviour = ctx.mkOr(opt1,opt2,opt3);

constraints.add(behaviour);
```

Figure 7.8. Hard Constraints example

each BoolExpr represents a different behavior, in fact for each one of them different values of the predicates protect and unprotect are set; then another boolExpr named behaviour puts in OR the previous three BoolExpr, because like explained in the chapter 7 the VPN Gateway for a given pair (Node,flow) can act just in one way; finally the resulting expression is added to the set of the hard constraints.

Finally an example of soft constraint is presented in the picture depicted below:

```
if(autoplace) {
    // VPN gw should not be used if possible
    nctx.softConstrVPNGWAutoPlace.add(new Tuple<BoolExpr, String>(ctx.mkNot(used), "VPN_auto_conf"));
}else {
    used = ctx.mkTrue();
    constraints.add(ctx.mkEq(used, ctx.mkTrue()));
}
```

Figure 7.9. Soft Constraint example

in this piece of code if the allocation of the VPN Gateway is left to the ADP module then a soft constraint is specified stating that if possible the solver has to

find a SAT solution without using the VPN Gateway; else, i.e. if the VPN Gateway is allocated manually by the security administrator, the used variable is set to true and no soft constraints are specified.

7.1.5 Automatic Allocation and Configuration in Z3

7.2 Validation

Two use cases are presented in this section in order to show all the explained features and see how actually the ADP module works.

7.2.1 Use Case 1

Let consider the network topology depicted below:

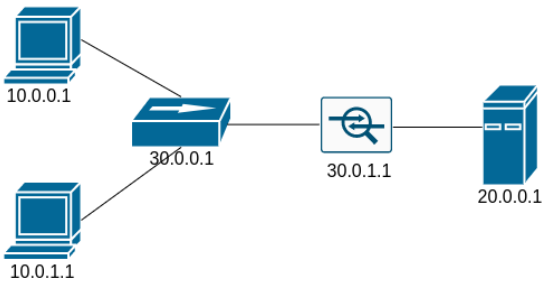


Figure 7.10. Input Service Graph

where:

- the End-Hosts have no VPN Capabilities;
- There are no constraints specified about the allocation of the VPN Gateways;

and the following communication protection requirements:

Security Requirements

1] Protect the traffic from 10.0.0.1 to 20.0.0.1. Untrusted node {30.0.0.1,30.0.1.1}.

2] Protect the traffic from 10.0.1.1 to 20.0.0.1. UntrustedNodes{30.0.0.1}. InspectorNodes{30.0.1.1}.

Figure 7.11. Communication protection requirements

The Use Case 1 focuses on:

- Network Model. Understating how the solver takes decisions depending on the nodes characterization. It is worth to note that the node 30.0.1.1 is untrusted for the requirement 1 and is inspector for the requirement 2.

- Allocation and configuration of VPN Gateways on the allocation places, highlighting the reasons of the solver solution to satisfy the set of network security requirements provided in input.

The first step is characterized by the generation of the Allocation Graph starting from the Service Graph provided in input:

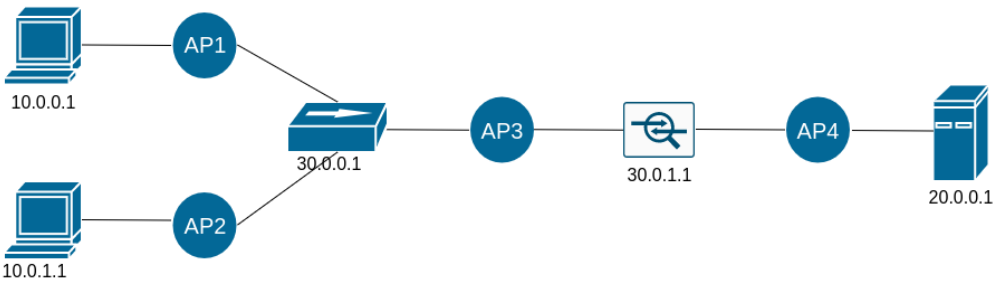


Figure 7.12. Allocation Graph

As can be noticed by the new topology an Allocation Place is positioned between each network function and can be exploited by the solver to allocate a VPN Gateway in order to satisfy the requirements.

Referring to the scenario 1 of the chapter 4 if the ADP module, solving the MaxSMT problem, finds a satisfiable solution a new Service Graph is computed with the allocated and/or configured channel protection functions:

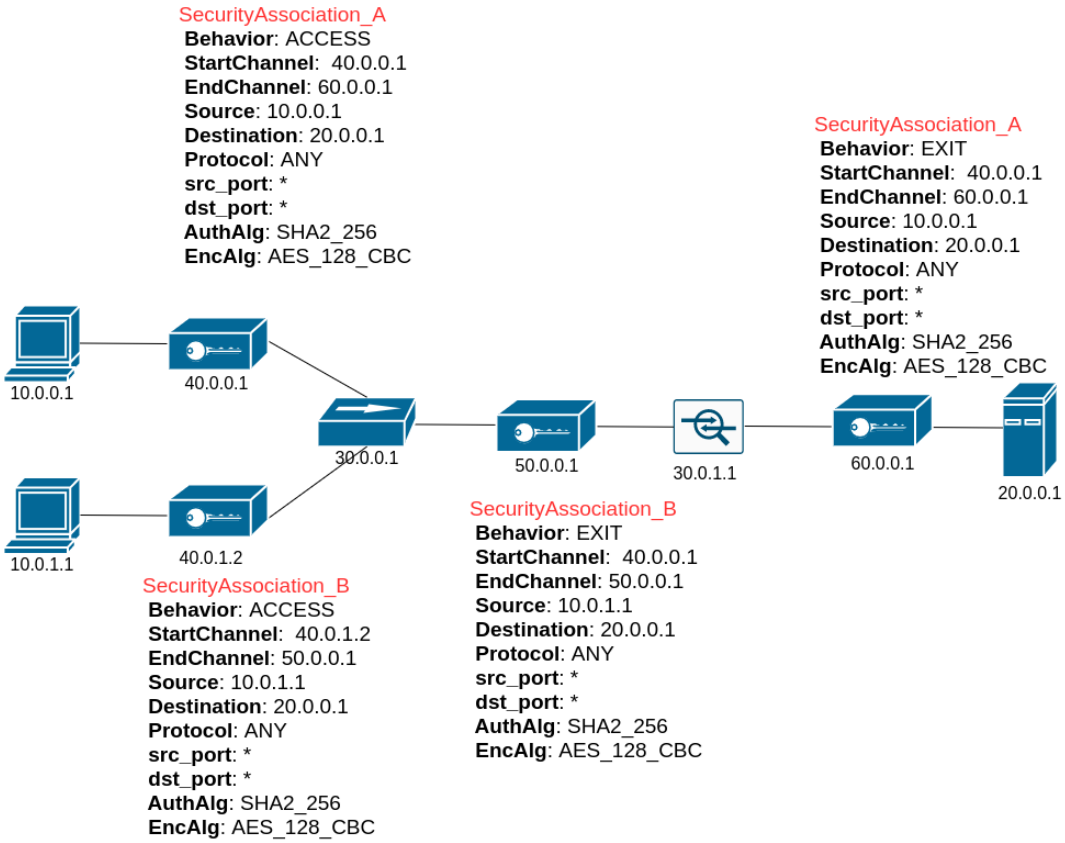


Figure 7.13. Output Service Graph

The solution shows the allocation and the configuration on the Graph of four VPN Gateways, two for the satisfaction of the requirement A enforcing a site-to-site tunnel from the 40.0.0.1 which is the ACCESS gateway i.e. the CPS in charge of protect the traffic and the 60.0.0.1 which is the EXIT Gateway i.e. the CPS in charge of remove the protection on the packet and forward it to the destination node. Two for the satisfaction of the requirement B creating another site-to site channel form the node 40.0.1.2 to the node 50.0.0.1 since the IDS must see the traffic without protection. In the requirements has not been specified anything about the algorithms to use, as consequence, the default ones have been adopted in the configurations. For the same reason the profile is Stable i.e. the default profile, anyway, given that the end-hosts have not VPN Capabilities selecting a different profile would not have produced a different solution.

7.2.2 Use Case 2

Let consider the same Input Service Graph of the Use Case 1, but:

- The End-Hosts have VPN Capabilities, in particular they support IPSEC technology, hence they inherit all the capabilities exposed by IPSEC;

- The security administrator set the following constraints: "no allocation place must be positioned between the node 10.0.0.1 and the node 30.0.0.1";
- The security administrator select the Stable profile;

and the following communication protection requirements:

Security Requirements

1] Protect the traffic from 10.0.0.1 to 20.0.0.1 with authAlg: SHA2_512 and encAlg: AES_256_CBC
UntrustedNodes{30.0.0.1, 30.0.1.1}

2] Protect the traffic from 10.0.1.1 to 20.0.0.1 with authAlg: SHA2_256 and encAlg: AES_128_CBC
UntrustedNodes{30.0.0.1}

Figure 7.14. Communication protection requirements

The Use Case 2 focuses on:

- End-Hosts with VPN Capabilities;
- Minimum allocation;
- Profile selection;

The first step is characterized by the generation of the Allocation Graph starting from the Service Graph provided in input:

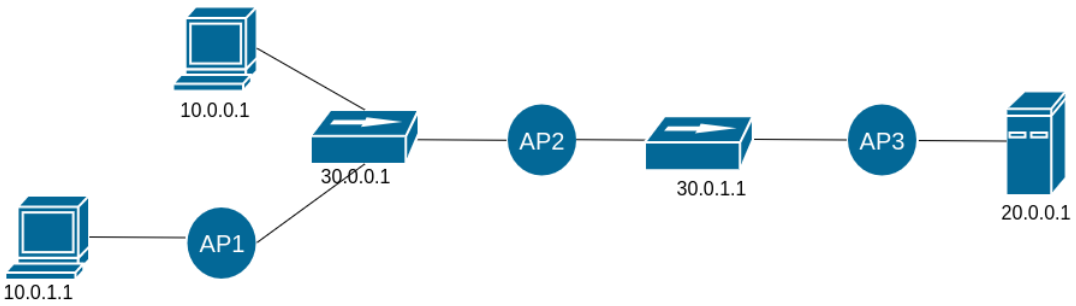


Figure 7.15. Allocation Graph

As can be noticed by the new topology differently from the Use Case 1 an Allocation Place, which can be exploited by the solver to allocate a VPN Gateway in order to satisfy the requirements, is positioned between each network function but between the node 10.0.0.1 and the node 30.0.0.1 due to the constraint specified by the security administrator;

Referring to the scenario 1 of the chapter 4 if the ADP module, solving the MaxSMT problem, finds a satisfiable solution a new Service Graph is computed with the allocated and/or configured channel protection functions:

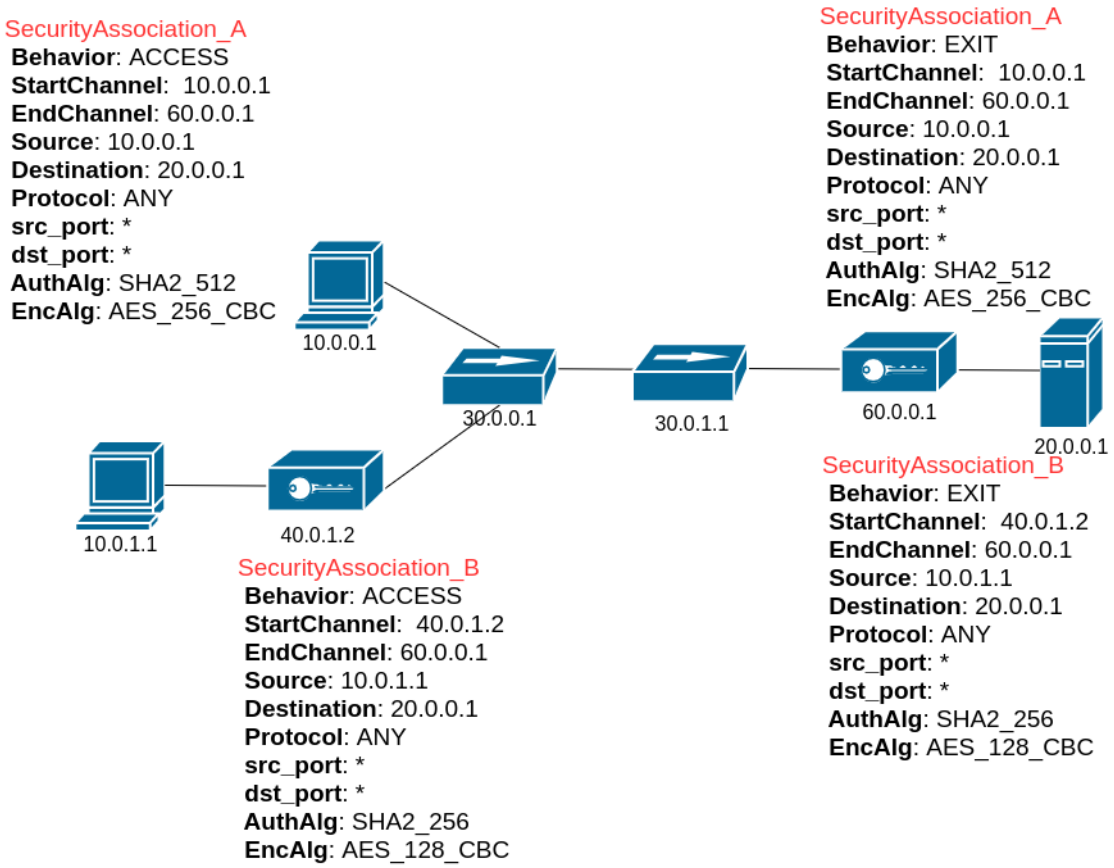


Figure 7.16. Output Service Graph of solution 1

The solution shows the allocation of only two VPN Gateways to satisfy both the communication protection requirements. For the requirement 1, despite the selected profile is the Stable one, since no allocation places was available between the node 10.0.0.1 and the node 30.0.0.1, the solver used the VPN Host Capability of the node 10.0.0.1 to satisfy the considered requirement. Regarding the communication protection requirement 2 the site-to-site channel between the VPN Gateway 40.0.1.2 and the VPN Gateway 60.0.0.1 is created to satisfy it. It is worth to note that the end host 10.0.0.1 is able to protect the traffic because it supports the IPsec technology which exposes, among its capabilities, the SHA2_512 and AES_256_CBC and as explained in the previous chapter the end-hosts inherit all the capabilities exposed by the supported technologies. Furthermore looking at the available allocation places in the Allocation Graph the network security requirement 2 cloud have been satisfied whether allocating a VPN Gateway between the node 30.0.0.1 and the node 30.0.1.1 or between the node 30.0.1.1. and the node 20.0.1, but since the solver tries to allocate the minimum number of CPSs considering all the requirements it chose to allocate a single VPN Gateway between the node 30.0.0.1 and the node 30.0.1.1.

In the following another solution is proposed which is the solution the ADP module would have produced if the security administrator had selected the Green profile.

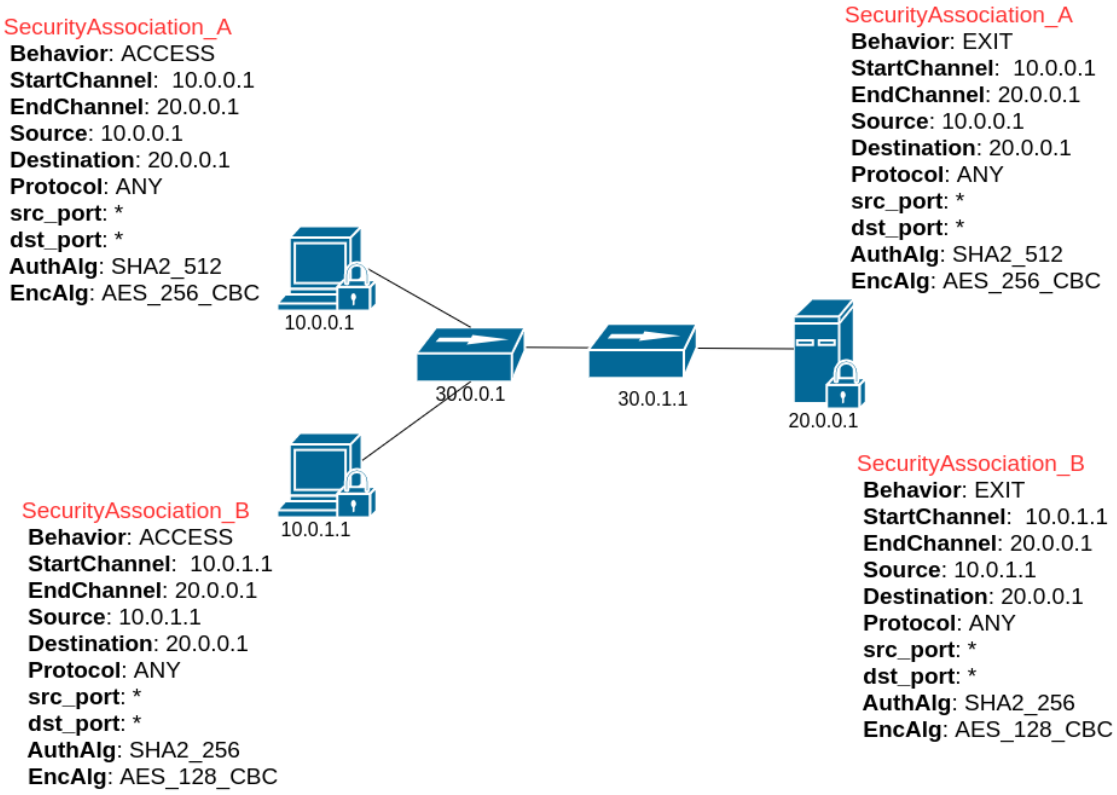


Figure 7.17. Output Service Graph of solution 2

In this solution two end-to-end channels are created to satisfy the communication protection requirements: one from the end-host 10.0.0.1 to the end-host 20.0.0.1 and another one from the end-host 10.0.1.1 to the end-host 20.0.0.1

7.3 Testing

A set of tests have been performed in order to evaluate how the implemented solution scales and which are the factors that have the most impact on the scalability of the system. All the tests have been performed with a Device having an i7-10750H CPU and 32 GB of RAM, for each input the tests have been repeated 50 times. The results are shown in the following by means of whisker plots where each box represents the first and the third quartile, the segment inside it represents the median and the cross indicates the average of the collected data; the external segments, i.e. the whiskers, are delimited by the highest and lowest values while the point external to the them denote the exceptional values.

The metrics which have been identified as the most relevant for the scalability tests are:

- the number of Allocation Places;
- the number of communication protection requirements.

First of all, focusing on the aforementioned metrics, the approach which has been followed to understand which one has the higher impact on the scalability of the implemented solution has been to increase one metric to a higher value, while keeping the other fixed. The results are presented in the following two charts, where the chart in Figure 7.18 shows how the solution scales to increasing of the communication protection requirements keeping fixed the number of the allocation places to 2, one for the access VPN Gateway and another one for the exit VPN Gateway; the figure 7.19, instead, shows how the solution scales to increasing of the number of allocation places, considering a single communication protection requirement :

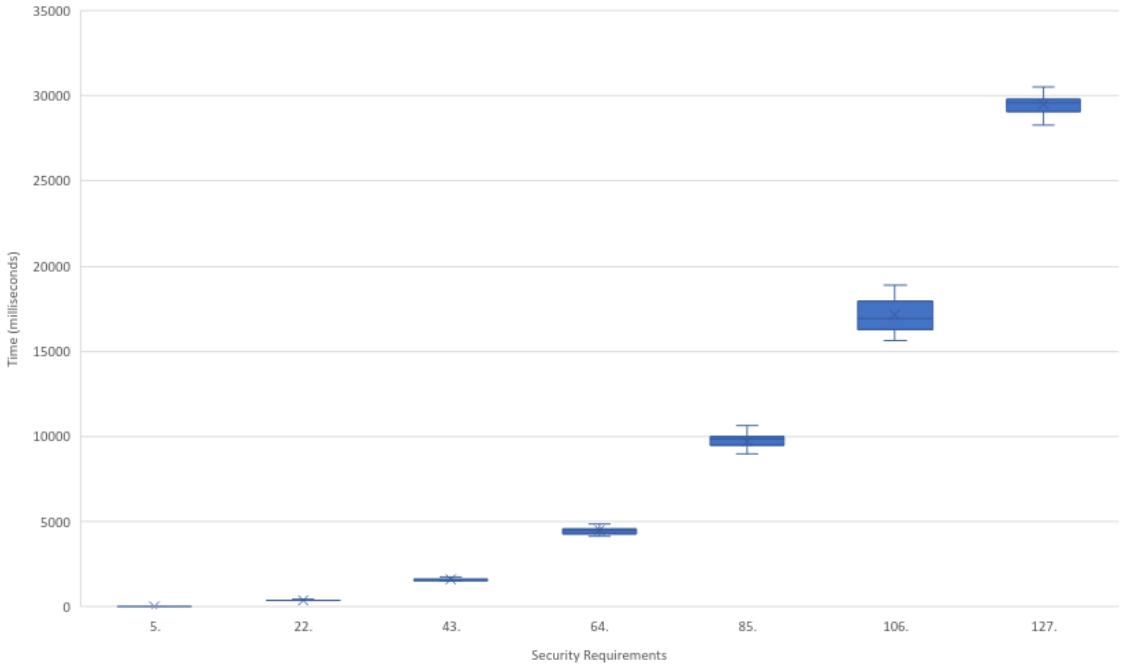


Figure 7.18. SR impact

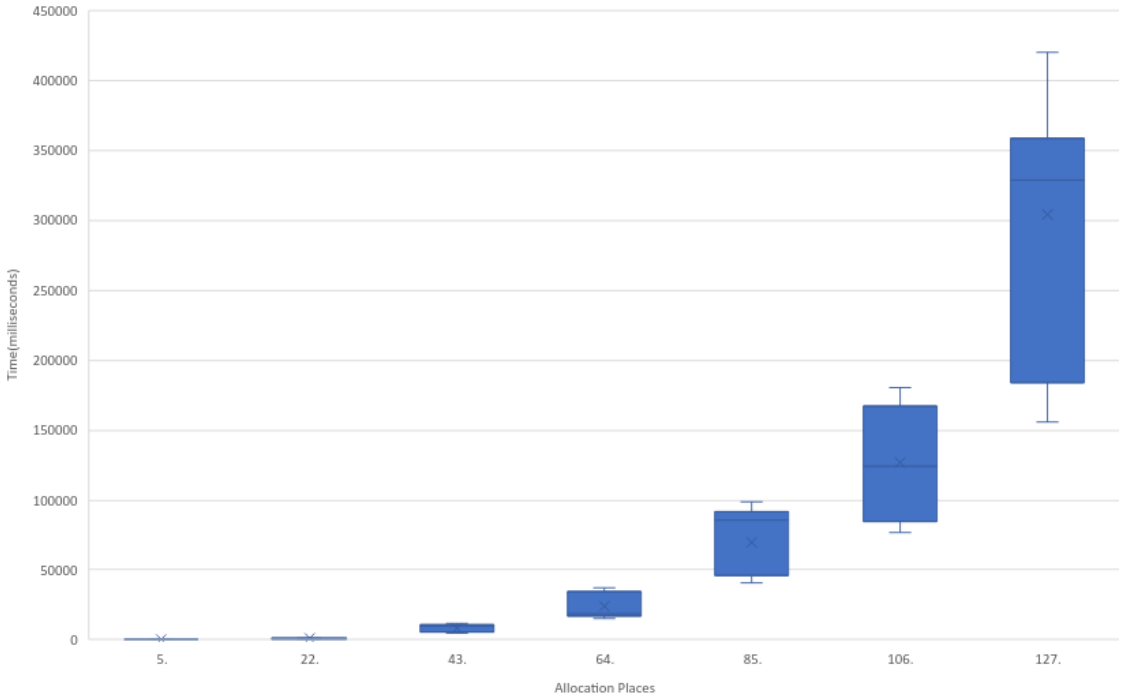


Figure 7.19. AP impact

By analyzing the two charts, it is evident that the increase of the execution time is not exponential, independently from which metric is considered (Allocation Places or Network Security Requirements); this is a fundamental result, given the intrinsic computational cost of a MaxSMT problem. Then, by comparing the two charts it is evident that the metric with the higher impact on the system scalability is the number of allocation places; moreover it can be notice that for each considered value of the x axis in the second chart the distribution of the values along the Y axis is more heterogeneous respect the first chart, this is due to how Z3 manages the integer composing the ip addresses.

A real scenario, instead, expect situations in which the security administrator specifies a set of communication protection requirements and that the allocation graph, inserted in input or generated by the ADP module starting from the input service graph, has a certain number of allocation places available in order to allocate the necessary CPSs to satisfy the requirements. For this reason in the following test both the number of Allocation Places and the number of communication protection requirements are progressively increased in order to evaluate how the solution scales with the growth of these values looking at the computational time to find the optimal solution. Since the number of allocation places has a higher impact on the scalability respect to the number of communication protection requirement, this value is incremented in a way that each security requirement pass through 4 allocation places. The results are shown in the chart depicted below:

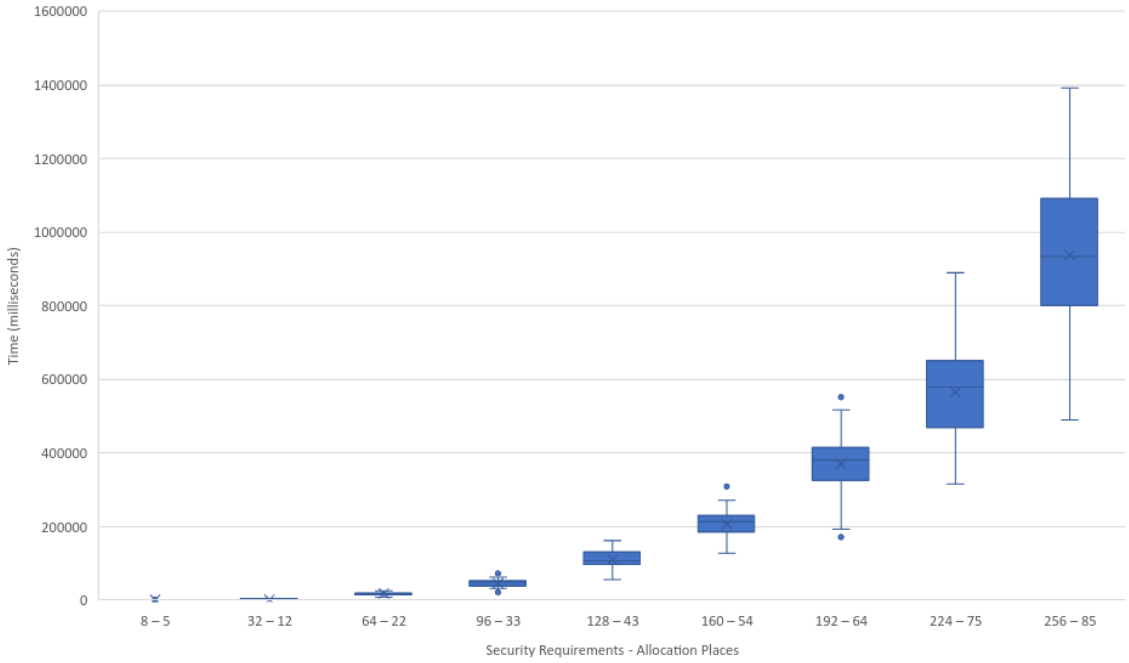


Figure 7.20. scalability test graphic

As can be inferred analyzing the graph the implemented solution scales well and allows to manage Allocation Graph with a high number of Allocation Places and a large set of security requirements in a reasonable time; for instance in a Allocation Graph having 54 Allocation places and 160 communication protection requirements the execution time does not overcome the 300000 ms, i.e approximately 5 minutes.

Chapter 8

Conclusions

During this thesis work a new version of the ADP module of VEREFOO has been modeled and implemented, capable of automatic allocate and automatic configure Channel Protection System in order to satisfy network security requirements related to the enforcement of secure communication.

First of all, a new type of network security requirements has been designed in order to allow to the security administrator to specify, exploiting either a HSPL which would be subsequently refined to obtain the corresponding requirement in the MSPL or directly exploiting a MSPL, the intention of creating secure communications.

Then, functions able to create secure communications have been made available to the ADP module, i.e. those functions that could be automatically allocated and/or automatically configured to satisfy communication protection requirements; in particular a new network security function, the VPN Gateway, has been designed and implemented from scratch in order to protect the traffic generated by other nodes, and the already existing network functions belonging to the category of the End-Host have been extended enabling them to protect its own traffic;

Furthermore, the concepts of technology and capabilities have been introduced with the purpose of allowing to the security administrator to specify a certain technology, such as IPSEC, or determinants capabilities, such as a certain authentication algorithm or a certain encryption algorithm, in the communication protection requirements.

A central work has been the formulation of the maxSMT problem by which the ADP module is capable of performing an automatic allocation and/or an automatic configuration of the necessary CPSs following some optimality criteria, receiving in input the set of communication protection requirements, the CPSs and the Service/Allocation graph; the MaxSMT problem is formulated so as to provide also a formal verification that the achieved solution is formally correct.

Finally, the reachability assurance of the packets belonging to a secure communication , flowing from the source to the destination when a packet filter is present on the traversed path has been granted, in order to avoid that the packet filter would drop those packets.

After all these works have been concluded, a series of performance tests have been carried out to evaluate the impact the number of communication security requirements and the number of allocation places have on the system scalability; the result which has been achieved is that the framework is capable of handle with a large set of requirements and with Allocation Graphs having a high number of allocation places in a reasonable time.

A possible future work about the ADP module of VEREFOO could be the implementation of a solution able to automatically allocate and automatically configure different network security functions at the same time, considering different kinds of requirements in a single iteration. Currently the ADP module is able to automatically allocate and automatically configure two different kinds of network security functions: Packet Filters and the Channel Protection Systems, but only in a sequential manner; in fact in order to allocate and configure both the types of network security function it must perform two iteration. In the first one the ADP module considers the Isolation and Reachability requirements and the Service Graph which produces in output with the eventual packet filters allocated and configured becomes the input Service Graph of the second iteration which deals with the allocation and configuration of CPSs. Furthermore additional future works could be to introduce a larger set of network functions which the service designer can exploit to define the Service Graph, to model additional Network Security Functions such as Intrusion Detection System and Web Application Firewalls to enrich the capabilities of the framework and to define new types of Network Security Requirements which could consider additional features.

Bibliography

[1] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "Automated optimal firewall orchestration and configuration in virtualized networks," in *NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, April 20-24, 2020*. IEEE, 2020, pp. 1–7. [Online]. Available: https://doi.org/10.1109/NOMS47738.2020.9110402

[2] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7665.txt

[3] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "Towards a fully automated and optimized network security functions orchestration," *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, pp. 1–7, 2019.

[4] F. Valenza and A. Lioy, "User-oriented network security policy specification," *J. Internet Serv. Inf. Secur.*, vol. 8, pp. 33–47, 2018.

[5] F. Valenza, C. Basile, D. Canavese, and A. Lioy, "Classification and analysis of communication protection policy anomalies," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2601–2614, 2017.

[6] Z. Fu and S. F. Wu, "Automatic generation of ipsec/vpn security policies in an intra-domain environment," in *DSOM*, 2001.

[7] Y. Yang, C. Martel, and S. F. Wu, "On building the minimum number of tunnels: an ordered-split approach to manage ipsec/vpn policies," *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507)*, vol. 1, pp. 277–290 Vol.1, 2004.

[8] C.-L. Chang, Y.-P. Chiu, and C. Lei, "Automatic generation of conflict-free ipsec policies," in *FORTE*, 2005.

[9] Yanyan Yang, Z. Fu, and S. F. Wu, "Bands: an inter-domain internet security policy management system for ipsec/vpn," in *IFIP/IEEE Eighth International Symposium on Integrated Network Management, 2003.*, 2003, pp. 231–244.

[10] M. Roßberg, G. Schäfer, and T. Strufe, "Distributed automatic configuration of complex ipsec-infrastructures," *Journal of Network and Systems Management*, vol. 18, pp. 300–326, 2010.