Automatic IPv4 to IPv6 Transition D3.1 - Secured Transition Engine

Frédéric Beck, Isabelle Chrisment, Olivier Festor

Automatic IPv4 to IPv6 Transition
D3.1 - Secured Transition Engine

Frederic Beck, Isabelle Chrisment, Olivier Festor

June 1, 2010

Contents

1 Introduction and Context **2**

2 Site Global Security Policy **5**
2.1 General recommendations . . . 5
2.2 Global variables . . . 6
2.3 Site-ISP interconnection: bogon prefixes filtering . . . 7
2.4 Subnet templates . . . 8
2.5 Common rules . . . 8
2.5.1 Anti-spoofing and abnormal traffic filtering . . . 8
2.5.2 ICMPv6 . . . 12
2.5.3 Multicast . . . 14
2.5.4 Firewall INPUT/OUTPUT traffic . . . 16
2.5.5 Blacklist . . . 18

3 Pinholes **19**
3.1 Specification . . . 19
3.1.1 Services aliases . . . 19
3.1.2 Custom rules . . . 21
3.2 Consistency . . . 21

4 Implementation **22**
4.1 Planned architecture . . . 22
4.2 XML schemas for firewall rules . . . 23
4.2.1 Ip6tables and cisco . . . 23
4.2.2 Generic schema . . . 24
4.3 Site policy . . . 24
4.3.1 ACLs generation . . . 25
4.4 Pinholes . . . 25
4.4.1 Service aliases . . . 25
4.4.2 Consistency . . . 27
4.5 Serialization . . . 28
4.6 GUI integration . . . 28
4.7 Future work . . . 31

5 Versus simple/advanced security drafts **32**

6 Next steps and future work **33**

7 Conclusion **34**

Abstract

Over the last decade, IPv6 has established itself as the most mature network protocol for the future Internet. While its acceptance and deployment remained so far often limited to academic networks, its recent deployment in both core networks of operators (often for management purposes) and its availability to end customers of large ISPs demonstrates its deployment from the inside of the network leading to the edges.

For many enterprises, the transition is seen as a tedious and error prone task for network administrators.

In the context of the Cisco CCRI project, we aim at providing the necessary algorithms and tools to automate the transition. In this report, we present the security policy that must be deployed on the site in conjunction with the addressing plan.

Chapter 1

Introduction and Context

IP networks are widely spread and used in many different applications and domains. Their growth continues at an amazing rate sustained by its high penetration in both the Home networks and the mobile markets. Although often postponed thanks to tricks like NAT, the exhaustion of available addresses, and other scale issues like routing tables explosion will occur in a near future.

IPv6 [5] was defined with a bigger address space (128 bits) and comes along with new built-in services (address autoconfiguration [12], native IPSec, routes aggregation, simplified header...). Despite its slow start, IPv6 is today more than ever the most mature network protocol for the future Internet. To faster its acceptance and deployment, it has however to offer autonomic capabilities that emerge in several recent protocols in terms of self-x functions reducing and often eliminating the man in the loop. We are convinced that such features are also required for the evolutionary aspects of an IP network, the transition from IPv4 to IPv6 being an essential one.

In this project, we are interested in the scientific part of the technological problems that highly impact human acceptance. Many network administrators are indeed reluctant to deploys IPv6 because, first, they do not know well the protocol itself, and they do not have sufficiently rich algorithmic support to seamlessly manage the transition from their IPv4 networks to IPv6. To address this issue, we investigate, design and aim at implementing a transition framework with the objective of making it self-managed.

As the IPv4 to IPv6 transition is a very complex operation, and can literally lead to the death of the network, there is a real need for a transition engine to ease and secure the network administrator's task; the ideal being a "one click" transition.

We made the assumption that the IPv6 network is constructed from scratch, without direct mapping of the IPv4 network into IPv6, as for many different reasons (different physical devices, different topologies...) both network are not necessarily identical. In order to offer the possibility to the administrators to adapt the IPv6 network to their needs and requirements, we defined a set of constraints that our algorithms use to define the addressing plan or security policy. These constraints permit an administrator, for example, to force a prefix on a subnet or link, to force the upstream router in case of multihoming, reserve some prefixes on a router for planned network extensions, or specify the behavior of a particular subnet (e.g. DMZ or IPv4 NAT). We focus on network devices and do not modify or configure anything on the end hosts.

The network topology is represented by an oriented graph, providing a logical view of the network to address. The root of the graph is the border router connected to the IPv6 Internet. The interconnection between the border router and the IPv6 provider is out of scope here, as there are already many documents and guidelines that can assist the administrator. On this part however, this interconnection is considered from the filtering point of view, as we set firewall rules at the border to protect the network. From the interior, the IPv6 connection is seen as native, and both the configuration and addressing are based on that assumption.

We defined a metric standing for the needs of a router or interface in terms of /64 prefixes it requires when transitioning to IPv6. This metric is propagated from the leaves to the root in the graph as shown

in figure 1.1.

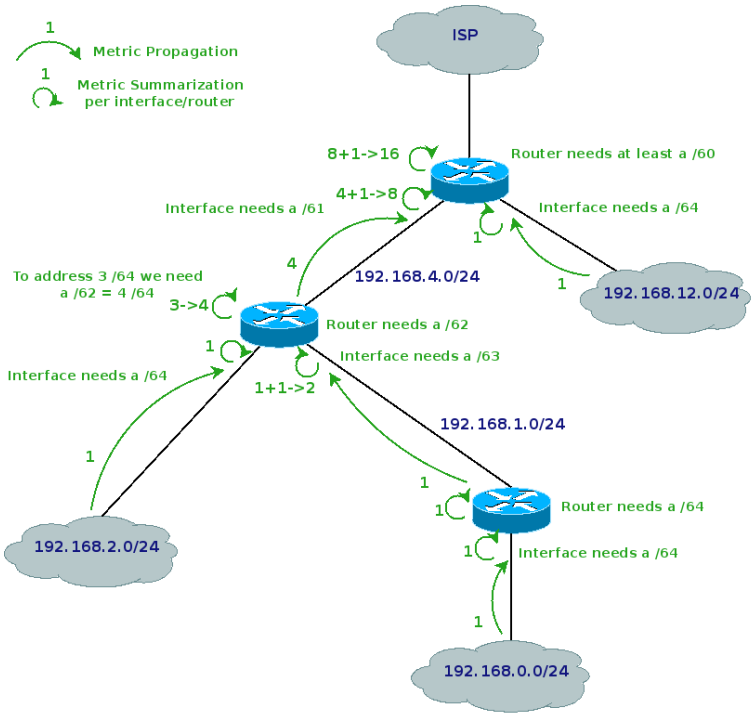

Figure 1.1: Metric Propagation

Then, the addressing plan is defined from the root to the leaves as depicted in Figure 1.2.

These algorithms have been implemented in a Transition Engine prototype and where successfully tested and validated on several topologies.

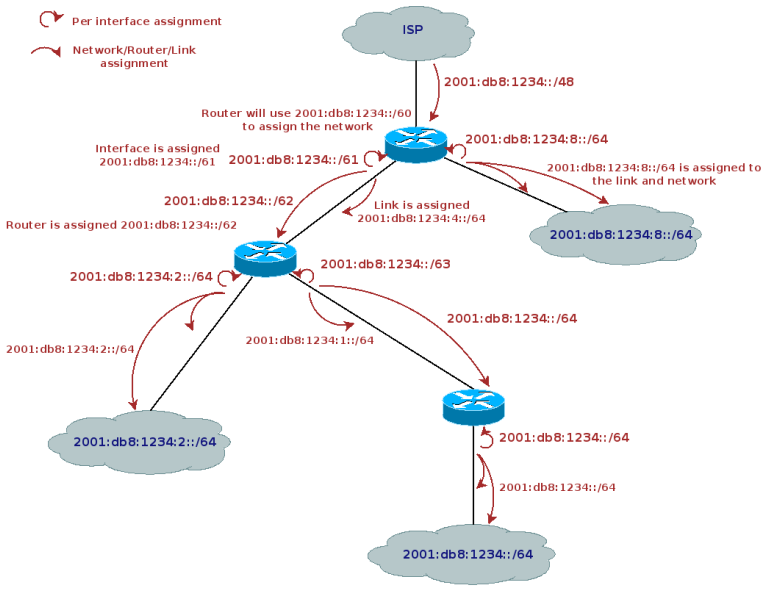

Figure 1.2: Addressing Heuristic

The next step in this study consists in describing the security policy and filtering rules that have to be deployed in conjunction with this addressing plan in order to follow the transition procedure we defined to ensure a smooth and safe transition:

1. Identify the network infrastructure
2. Identify the needs in term of addressing and request site prefix
3. Set ingress filtering
4. Connect the border router to the IPv6 world
5. Identify the constraints
6. Determine an addressing plan
7. Define and set firewall rules
8. Update DNS entries
9. Configure routing infrastructure and address routers
10. Address nodes
11. Verify addressing of core services
12. Advertise the prefix and DNS

In this report, we present a study on the security of the newly generated IPv6 network. We will first present the site global security policy and the related filtering rules. Then, we will specify the security engine and present its implementation. Finally, we will discuss our study regarding the existing work.

Chapter 2

Site Global Security Policy

In order to represent the security rules to apply on the network devices, we need to define a site security policy. To do so, we define a default policy. But this policy must be configurable and adaptable to the needs and constraints of each network. Therefore, we define a data model representing this security policy. This data model is expressed by a RelaxNG XML schema. In this section, we present the different aspect of the security policy we considered and how they can be configured in the data model.

In this data model, we have 2 levels of security policies. The first one is the global policy of the site that will be applied on the border router. The second one is a subnet level policy, where some subnets may require specific rules, such as ones for a DMZ. In this case, the rules are applied on all the upstream routers connected to that subnet.

In the following section, we detail the security policy of the site. We present the default behavior and actions to perform, these aspects being configurable in the data model. We present some general recommendations to follow when defining such a policy. Then, we describe some global variables that can be used in both site and subnet levels. Then, we present some specific aspects to the site (at the interconnection with the ISP) or to a subnet, before detailing the policy to be applied.

2.1 General recommendations

In this section, we highlight the general recommendations that should be considered when defining the site global policy:

- All firewalls deployed are stateful firewalls.
- All firewalls deployed have strict Reverse Path Filtering (RPF) enabled.
- Their default policy is to drop all inbound packets that do not match explicit pinholes.
- The outbound default policy can be DROP if the traffic is restricted or ACCEPT if it is unrestricted. If the outbound traffic is restricted, some pinholes need to be defined to allow the desired services to successfully bypass the filtering.
- The pinholes are as restrictive and detailed as possible (n-tuples with source, destination addresses and ports, protocol...).
- The minimum default set of permitted services if the outbound traffic is restricted are HTTP(S), SSH and FTP.
- If a packet is filtered, the firewall should send an ICMPv6 Destination Unreachable (Administratively Prohibited).

- If a DMZ is present with a DNS server, name resolution is permitted toward the server in that subnet, otherwise it is allowed toward the Internet.
- All Neighbor Discovery [10] packets with a hop limit different from 255 should be dropped.
- In general, all packets with link local source or destination with a hop limit different from 255 should be dropped.
- When addressing the routers, it is recommended not to use an obvious sequential addressing (e.g. ::1, ::2, ::3...). This feature can be configured within the transition engine, so that it is taken into account in the addressing plan. It is possible to use a configured sequential addressing (with a given start an increment), EUI-64 or identifiers following RFC 4941 [9].

Packet inspection and connection tracking is activated as well. As stated in RFC5382 [6], the time-out for TCP established idle connections is 7440 seconds (2 hours 04 minutes), and the transitory time-out is 0. For UDP, the idle time-out is 20 seconds. This gives us the follow inspection rules on a Cisco router:

```
ipv6 inspect audit-trail
ipv6 inspect max-incomplete low 150
ipv6 inspect max-incomplete high 250
ipv6 inspect one-minute low 100
ipv6 inspect one-minute high 200
ipv6 inspect udp idle-time 20
ipv6 inspect tcp idle-time 7440
ipv6 inspect tcp finwait-time 3
ipv6 inspect tcp synwait-time 15
ipv6 inspect tcp max-incomplete host 40 block-time 0
ipv6 inspect name v6_fw tcp timeout 300
ipv6 inspect name v6_fw ftp
ipv6 inspect name v6_fw udp
ipv6 inspect name v6_fw icmp
```

These rules will be set for inbound and outbound traffic inspection on all interfaces. For Netfilter firewalls, these operations are performed with the *state* module.

As we consider the internal to be native IPv6, 6to4 and Teredo tunneling are prohibited. Outbound rules deny the usage of such addresses as source, and inbound filtering drops packets to such a destination, while permitting to communicate with nodes using these technologies in the Internet. Prohibiting these two technologies requires IPv4 filtering, by blocking UDP port 3544 and the resolution of `teredo.ipv6.microsoft.com` for Teredo or protocol 41 (IPv6 in IPv4 encapsulation) for 6to4. This can be bypassed by using 6to4 relays within the site. RFC 3964 [11] contains useful information on how to counter this kind of threat.

2.2 Global variables

To simplify the definition of the security policy, we proposed a set of aliases that may be used to identify the source or destination of the filtering rules:

- **internal** All global or ULA prefixes deployed within the site
- **local** Only ULA prefixes deployed within the site
- **external** Global prefixes from partners and providers to allow access to specific services or hosts for business related communications
- **management** Allowed to perform management operations
- **any** The default when nothing is specified

These aliases are lists of prefixes and/or hosts that match the description. They can be used for the site global policy or subnets policy.

2.3 Site-ISP interconnection: bogon prefixes filtering

Protecting the network with traffic filtering is not sufficient. The announces received by the routing neighbors must be filtered as well to protect against advertisement of deprecated or bogus prefixes. This operation is called Bogon Prefixes Filtering, and should be enabled for all inbound and outbound routing protocol updates.

We identified the following prefixes that should be filtered in both the inbound and outbound prefix-list:

- **::/8 le 128** IPv4 Compatible and Mapped addresses
- **fe00::/9 le 128** Link and Site Local Addresses
- **fc00::/7 le 128** Unique Local Addresses
- **ff00::/8 le 128** Multicast Addresses
- **2001:db8::/32 le 128** Document Addresses
- **3ffe::/16 le 128** 6Bone Addresses
- **2001:10::/28 le 128** ORCHID

Teredo and 6to4 are special cases. The prefix used for these tunneling mechanisms should be accepted, to enable traffic sent to these hosts, but smaller prefixes should be dropped:

- **Teredo** permit 2001::/32
  deny 2001::/32 le 128
- **6to4** permit 2002::/16
  deny 2002::/16 le 128

Finally, the prefixes assigned to the site should be filtered as well. At the inter-connection with the ISP, the prefix must be filtered for inbound announces: *deny* P_{Site} *le 128*. Inside the network, inbound announces of the site prefix must be accepted, to allow routing between the site subnets: *permit* P_{Site} *le 64*. For outbound announces, at the site-ISP inter-connection, we permit the site prefix (*permit* P_{Site}), and within the site, each router is allowed to advertise the prefix it routes: *permit* P_{Router}.

To express the prefixes that are valid, we defined two mode of bogon filtering:

- **relaxed** permit 2000::/3 le 48; does not require regular update, at least not until all 2000::/3 have been assigned to the Regional Internet Registries (RIRs)
- **strict** explicitly permit only prefixes assigned to the RIRs by the IANA [1]; requires an update when a new prefix is assigned (last prefix assigned the 13th of May 2008)

By default, the relaxed mode is set.

In the configuration, it is possible to specify whether the router should accept announces containing the default route. This may be useful in some cases within the site. by default, even if this feature is activated, it is not set on the inter-connection between the site and the ISP.

In all cases, the default rule is to deny everything that has not been explicitly permitted: deny ::/0 le 128

[1] `http://www.iana.org/assignments/ipv6-unicast-address-assignments/`

2.4 Subnet templates

Subnets can have different roles in the network. Based on these roles, different filtering rules or behaviors are expected, which are expressed by the following templates:

- **DMZ** Stateful firewall, deny as default policy for inbound and outbound traffic, pinholes to allow traffic
- **IPv4 NAT** Stateful firewall, deny as inbound default policy with pinholes to allow traffic, outbound default policy of accept
- **Local** no Internet access, all non-ULA prefixes are explicitly filtered, limit source and destination addresses of the packets to the internal alias

IPv4 NAT means that the IPv4 subnet migrated to IPv6 was using this technology. In this case, we want to keep the basic security implied by the address translation. Here, we also recommend the usage of Privacy Extensions [9] for that subnet, but as it is host-oriented and does not require any configuration at the network level, we can not do it automatically. Other benefits of NAT can be achieved as stated in RFC4864 [4], but they are hard to implement on an automated basis, and we are not convinced that these benefits are really required in such a case.

For other types of subnets, we do not recommend the usage of Privacy Extensions, and in particular the temporary addresses, as it makes difficult the monitoring and local tracking of hosts, even if the benefits at a larger scale have been proven.

Some tools such as NDPMon [2] can help resolving that problem of host tracking locally if the usage of Privacy Extensions is desired.

Mechanisms such as NAT66 [14] or other IPv6 NAT proposals can be of some interest if they reached the standard status and could be added as another template here, and handled as another constraint in the addressing algorithm.

Of course, it is also possible to define a different policy for a subnet by setting the different parameters explicitly.

A local subnet is restricted to traffic from and to the site. It is possible to specify whether the inbound traffic is restricted or not. If it is, the only services that are accessible are the ones that generate pinholes. Such a pinhole is propagated within the site, but not on the inter-connection with the ISP. As the first rules that are set concern reverse path filtering and a local pinhole has an internal prefix as source, it would be dropped anyway. If the subnet is not restricted, all traffic initiated from the site itself passes through the firewall. Moreover, as no packets from the Internet can reach this subnet, to avoid routing such packets through the network when it is not necessary, all traffic to/from this subnet is denied at the inter-connection between the site and the ISP.

2.5 Common rules

In this section, we define the default filtering rules that should be applied on the border and on each internal firewall. These rules are set in all inbound or outbound ACL as defined in the following subsections.

2.5.1 Anti-spoofing and abnormal traffic filtering

As it has been done for the routing advertisements, we must filter the bogon and deprecated prefixes in the packets themselves.

Moreover, as stated in RFC 5095 [1], an IPv6 packet with Routing Header 0 should be dropped, and other types should be accepted.

[2] `http://ndpmon.sf.net`

Figure 2.1: From the Internet to the Site

Action	Source	Destination	Protocol	Port	Comment
DROP	any ::/128	::/128 any	any any	any any	Unspecified Address
DROP	any ::1/128	::1/128 any	any any	any any	Loopback Address
DROP	any ::/96	::/96 any	any any	any any	IPv4 Compatible Addresses
DROP	any ::ffff:0:0/96	::ffff:0:0/96 any	any any	any any	IPv4 Mapped Addresses
DROP	any fe80::/10 or longer	fe80::/10 or longer any	any any	any any	Link Local Addresses
DROP	any fec0::/10 or longer	fec0::/10 or longer any	any any	any any	Site Local Addresses
DROP	any fc00::/7	fc00::/7 any	any any	any any	Unique Local Addresses
DROP	any 2001:db8::/32	2001:db8::/32 any	any any	any any	Document Addresses
DROP	any 3ffe::/16	3ffe::/16 any	any any	any any	6Bone Addresses (deprecated)
DROP	any 2001:10::/48	2001:10::/48 any	any any	any any	ORCHID (Overlay Addressing)
DROP PERMIT	any 2001::/32	2001::/32 SITE	any any	any any	Teredo as destination Teredo as source
DROP PERMIT	any 2002::/16	2002::/16 SITE	any any	any any	6to4 as destination 6to4 as source
DROP	any	any	any	any	Default rule

Figure 2.2: From the Site to the Internet

Action	Source	Destination	Protocol	Port	Comment
DROP	any ::/128	::/128 any	any any	any any	Unspecified Address
DROP	any ::1/128	::1/128 any	any any	any any	Loopback Address
DROP	any ::/96	::/96 any	any any	any any	IPv4 Compatible Addresses
DROP	any ::ffff:0:0/96	::ffff:0:0/96 any	any any	any any	IPv4 Mapped Addresses
DROP	any fe80::/10 or longer	fe80::/10 or longer any	any any	any any	Link Local Addresses
DROP	any fec0::/10 or longer	fec0::/10 or longer any	any any	any any	Site Local Addresses
DROP	any fc00::/7	fc00::/7 any	any any	any any	Unique Local Addresses
DROP	any 2001:db8::/32	2001:db8::/32 any	any any	any any	Document Addresses
DROP	any 3ffe::/16	3ffe::/16 any	any any	any any	6Bone Addresses (deprecated)
DROP	any 2001:10::/48	2001:10::/48 any	any any	any any	ORCHID (Overlay Addressing)
DROP PERMIT	2001::/32 SITE	any 2001::/32	any any	any any	Teredo as source Teredo as destination
DROP PERMIT	2002::/16 SITE	any 2002::/16	any any	any any	6to4 as source 6to4 as destination
DROP	any	any	any	any	Default rule

Traffic from the site to the Internet needs to be specifically permitted, and restricted to the site's prefix as source only. In the same way, traffic from the Internet to the site must be limited to the site's prefix as destination, and dropped if a packet coming from the Internet has the site prefix as source. Reverse Path Filtering must be activated on all interfaces and explicit filtering rules should be set as well.

Action	**Source**	**Destination**	**Protocol**	**Port**	**Comment**
PERMIT	SITE	any	any	any	Legitimate Traffic
DROP	any	any	any	any	Spoofed source

Figure 2.3: From the site to the Internet

Action	**Source**	**Destination**	**Protocol**	**Port**	**Comment**
DROP	SITE	any	any	any	Spoofed source
PERMIT	any	SITE	any	any	Legitimate traffic

Figure 2.4: From the Internet to the site

2.5.2 ICMPv6

As specified in RFC 4890 [3], the following rules should be applied on ICMPv6 traffic.

In the following table, we present the rules that should be applied in the FORWARD table. The terminology used is the same than RFC 4890. We take the assumption that we want the same rules from and to the Internet. This could be modified by the administrator to match its own preferences, for example by dropping Echo Request packets from the Internet to the site.

Action	Source	Destination	Protocol	Type	Code	Comment
PERMIT	any	any	icmpv6	1	any	Destination Unreachable (MUST NOT)
PERMIT	any	any	icmpv6	2	any	Packet too big (MUST NOT)
PERMIT	any	any	icmpv6	3	0	Time Exceeded (MUST NOT)
PERMIT	any	any	icmpv6	4	1-2	Parameter Problem (MUST NOT)
PERMIT	any	any	icmpv6	128	any	Echo Request (MUST NOT)
PERMIT	any	any	icmpv6	129	any	Echo Reply (MUST NOT)
PERMIT	any	any	icmpv6	3	1	Time Exceeded (SHOULD NOT)
PERMIT	any	any	icmpv6	4	0	Parameter Problem (SHOULD NOT)
PERMIT	any	any	icmpv6	144	any	HA Addr Disco Req (SHOULD NOT)
PERMIT	any	any	icmpv6	145	any	HA Addr Disco Resp (SHOULD NOT)
PERMIT	any	any	icmpv6	146	any	Mobile Prefix Sol (SHOULD NOT)
PERMIT	any	any	icmpv6	147	any	Mobile Prefix Adv (SHOULD NOT)
DROP	any	any	icmpv6	139	any	Node Info Query(SHOULD)
DROP	any	any	icmpv6	140	any	Node Info Response (SHOULD)
DROP	any	any	icmpv6	138	any	Router Renumbering (SHOULD)
DROP	any	any	icmpv6	100-101	any	(SHOULD)
DROP	any	any	icmpv6	127	any	(SHOULD)
DROP	any	any	icmpv6	200-201	any	(SHOULD)
DROP	any	any	icmpv6	255	any	(SHOULD)
DROP	any	any	icmpv6	any	any	Default rule

Figure 2.5: ICMPv6 Traffic to Filter (FORWARD)

The rules for types 144 to 147 should only be applied if MIPv6 [8] is used on the site, which is something that needs further investigation.

We must also consider that two modes can be defined for ICMPv6, as it has been defined for the Bogon Prefix List:

relaxed ICMP filtering rules defined in RFC 4890 and presented here

strict More restrictive filtering to protect against the discovery of the internal topology (e.g. echo requests and replies would be denied from the Internet).

Action	Source	Destination	Protocol	Type	Code	Comment
PERMIT	any	IP_{FW}	icmpv6	1	any	Dest Unreachable (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	2	any	Packet too big (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	3	0	Time Exceeded (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	4	1-2	Parameter Problem (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	128	any	Echo Request (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	129	any	Echo Reply (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	133-134	any	RS/RA (MUST NOT)
PERMIT	any		icmpv6	135-136	any	NS/NA (MUST NOT)
PERMIT	any	FF02::/16	icmpv6	133-134	any	RS/RA (MUST NOT)
PERMIT	any	FF02::/16	icmpv6	135-136	any	NS/NA (MUST NOT)
PERMIT	::/128	IP_{FW}	icmpv6	135	any	NS for DAD
PERMIT	any	IP_{FW}	icmpv6	141	any	Inv NDP Sol (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	142	any	Inv NDP Adv (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	130	any	MLD Query (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	131	any	MLD Report (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	132	any	MLD Done (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	143	any	MLDv2 Report (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	151	any	Mult Router Adv (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	152	any	Mult Router Sol (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	153	any	Mult Router Term (MUST NOT)
PERMIT	any	IP_{FW}	icmpv6	3	1	Time Exceeded (SHOULD NOT)
PERMIT	any	IP_{FW}	icmpv6	4	0	Parameter Problem (SHOULD NOT)
DROP	any	IP_{FW}	icmpv6	148	any	Cert Path Sol (MUST NOT)
DROP	any	IP_{FW}	icmpv6	149	any	Cert Path Adv (MUST NOT)
DROP	any	IP_{FW}	icmpv6	137	any	Redirect
DROP	any	IP_{FW}	icmpv6	139	any	Node Info Query
DROP	any	IP_{FW}	icmpv6	140	any	Node Info Response
DROP	any	IP_{FW}	icmpv6	100-101	any	(SHOULD)
DROP	any	IP_{FW}	icmpv6	127	any	(SHOULD)
DROP	any	IP_{FW}	icmpv6	154-255	any	(SHOULD)
DROP	any	IP_{FW}	icmpv6	any	any	Default rule

Figure 2.6: ICMPv6 Traffic to Filter (INPUT)

where IP_{FW} stands for any address set on the firewall interface on which the packet is received, and *FF02::/16*.

NDP messages RA/RS and NA/NS are also allowed with FF02::/16 as destination. This permits to receive messages multicasted to the all-nodes, all-routers... addresses.

NDP NS with undefined source is accepted as well in INPUT in the scope of the DAD procedure. This rule must be set before the rules defined in section 2.5.1.

Rules for types 148 and 149 must be set to PERMIT if SEND [2] is used on the site. Otherwise, these rules must apply the DROP action. As we do not consider SEND in our study, and we do not believe that it is something that may be of any interest for the networks, regarding its complexity, we set these rules to DROP.

Rules for types 128 and 129 (echo request and reply) are shown here in mode *relaxed*. A *strict* mode has been defined as well that limit this messages to the management alias. All other sources or destination are prohibited to ping the hosts on the network. However, the hosts are allowed to ping any computer in the Internet.

Action	Source	Destination	Protocol	Type	Code	Comment
PERMIT	IP_{FW}	any	icmpv6	any	any	Default rule

Figure 2.7: ICMPv6 Traffic to Filter (OUTPUT)

where IP_{FW} stands for any address set on the firewall interface from which the packet is emitted.

2.5.3 Multicast

Packets with a multicast address as source should be dropped in any case. Moreover, special attention must be brought to the scope of the packets: no packet must be forwarded beyond its scope.

To do so, we must consider 4 scenarios.

Interconnection with the ISP

At the site boundaries, only multicast packets which have a multicast destination with global scope should be forwarded. Packets with node, link, site or organization scope should be dropped.

Action	Source	Destination	Protocol	Port	Comment
PERMIT	any	FF0E::/16			ASM Global Multicast
PERMIT	any	FF1E::/16			ASM Global Multicast
PERMIT	any	FF3E::/16			SSM Global Multicast
DROP	any	FF00::/8			Default rule
DROP	FF00::/8	any			Source Multicast

Figure 2.8: Multicast from the Internet to the Site

Action	Source	Destination	Protocol	Port	Comment
PERMIT	P_{Site}	FF0E::/16			ASM Global Multicast
PERMIT	P_{Site}	FF1E::/16			ASM Global Multicast
PERMIT	P_{Site}	FF3E::/16			SSM Global Multicast
DROP	any	FF00::/8	any	any	Default rule
DROP	FF00::/8	any			Source Multicast

Figure 2.9: Multicast from the Site to the Internet

Forwarding within the site

Action	Source	Destination	Protocol	Port	Comment
PERMIT	any	FF0E::/16			ASM Global Multicast
PERMIT	any	FF1E::/16			ASM Global Multicast
PERMIT	any	FF3E::/16			SSM Global Multicast
PERMIT	P_{Site}	FF05::/16			Site scope
PERMIT	P_{Site}	FF08::/16			Organization Scope
DROP	any	FF00::/8	any	any	Default rule
DROP	FF00::/8	any			Source Multicast

Figure 2.10: Forwarding within the site

INPUT on the firewall

Action	Source	Destination	Protocol	Port	Comment
PERMIT	any	FF0E::/16			ASM Global Multicast
PERMIT	any	FF1E::/16			ASM Global Multicast
PERMIT	any	FF3E::/16			SSM Global Multicast
PERMIT	P_{Site}	FF05::/16			Site scope
PERMIT	P_{Site}	FF08::/16			Organization Scope
PERMIT	P_{Link}	FF02::/16			Link Local Scope
PERMIT	FE80::/10	FF02::/16			Link Local Scope
DROP	any	FF00::/8	any	any	Default rule
DROP	FF00::/8	any			Source Multicast

Figure 2.11: INPUT on the Firewall

OUTPUT from the firewall

Action	Source	Destination	Protocol	Port	Comment
ACCEPT	IP_{FW}	FF00::/8	any	any	Default rule
DROP	FF00::/8	any			Source Multicast

Figure 2.12: OUTPUT on the Firewall

2.5.4 Firewall INPUT/OUTPUT traffic

Incoming traffic to the firewall should be strictly restricted. Besides ICMPv6, bogus and multicast filtering presented in the earlier sections (2.5.1, 2.5.2 and 2.5.3), all traffic should be dropped. The only traffic that should be permitted is management traffic. This includes SNMP, NTP, Telnet or SSH for remote configuration, FTP or TFTP for configuration up/download... This traffic should be restricted to explicit IPv6 addresses, i.e. the management alias defined earlier.

In the following table, IP_M stands for the IPv6 addresses of the management alias, and IP_{FW} for the IPv6 address of the firewall. If ULAs are deployed on the network, it is preferable to use these addresses for management access to the firewall.

Action	Source	Destination	Protocol	Port	Comment
PERMIT	IP_M	IP_{FW}	tcp	ssh	Access to the SSH server
PERMIT	IP_M	IP_{FW}	tcp	telnet	Access to the Telnet Server
PERMIT	IP_M	IP_{FW}	tcp	snmp	Access to the SNMP Server
DROP	any	any	any	any	Default rule

Figure 2.13: Traffic to the Firewall (INPUT)

Depending on the configuration method selected in the Transition Engine, we will open Telnet and/or SSH for remote configuration. Special attention must be brought to the routing protocol used, and rules may be set accordingly, to avoid dropping packet used for internal or external routing. Depending on the routing protocol used, the following rules are applied on each concerned interface of the router.

RIPng uses TCP or UDP port 521 and multicast group FF02::9 or unicast to the IPv6 Link Local address on the router's interface. Even if UDP towards the multicast group is mainly used, all rules should be permitted to avoid missing any information.

Action	Source	Destination	Protocol	Port	Comment
PERMIT	any	FF02::9	udp	521	
PERMIT	any	FF02::9	tcp	521	
PERMIT	any	IP_{FW}	udp	521	
PERMIT	any	IP_{FW}	tcp	521	

Figure 2.14: RIPng in INPUT

Action	Source	Destination	Protocol	Port	Comment
PERMIT	IP_{FW}	FF02::9	udp	521	
PERMIT	IP_{FW}	FF02::9	tcp	521	
PERMIT	IP_{FW}	any	udp	521	
PERMIT	IP_{FW}	any	tcp	521	

Figure 2.15: RIPng in OUTPUT

OSPF runs directly over IP with the protocol number 89. It uses the multicast groups FF02::5 and FF02::6, or unicast to the IPv6 Link Local address on the router's interface.

Finally, BGP uses TCP on port 179 with known BGP neighbors (specified in the transition engine).

FTP or TFTP traffic is generally issued by the firewall. There is no need to allow the return packet as the stateful filtering will allow this traffic. The traffic issued by the firewall does not require any filtering, besides the common rules for anti-spoofing, multicast and icmpv6.

The rules defined above for routing protocols in output are thus not required.

Action	Source	Destination	Protocol	Port	Comment
PERMIT	any	FF02::5	89		
PERMIT	any	FF02::6	89		
PERMIT	any	IP_{FW}	89		

Figure 2.16: OSPFv3 in INPUT

Action	Source	Destination	Protocol	Port	Comment
PERMIT	IP_{FW}	FF02::5	89		
PERMIT	IP_{FW}	FF02::6	89		
PERMIT	IP_{FW}	any	89		

Figure 2.17: OSPFv3 in OUTPUT

Action	Source	Destination	Protocol	Port	Comment
PERMIT	$IP_{neighbor}$	IP_{FW}	tcp	179	

Figure 2.18: BGP in INPUT

Action	Source	Destination	Protocol	Port	Comment
PERMIT	IP_{FW}	$IP_{neighbor}$	tcp	179	

Figure 2.19: BGP in OUTPUT

Action	Source	Destination	Protocol	Port	Comment
PERMIT	IP_{FW}	any	any	any	Default rule

Figure 2.20: Traffic from the Firewall (OUTPUT)

2.5.5 Blacklist

It is possible to define a blacklist of hosts or prefixes that are not allowed to communicate with the site or subnet. These hosts or prefixes will be explicitly prohibited in the ACLs.

Chapter 3

Pinholes

As already said in section 2.1, we use stateful firewalls. This implies that we need to explicitly open pinholes to allow access to the services within the site or a subnet of it. These pinholes create rules as detailed as possible that permit some traffic to pass the firewall.

The pinholes can be set in input (e.g. to access a WEB server in the DMZ), in output if it is restricted (e.g. permit host within a NAT subnet to surf on the Internet), or both (e.g. a DNS server in the DMZ must be accessible from the internal network, but also needs to join other DNS servers in the Internet for recursion). Other pinholes are by default restricted to the management alias (e.g. SSH access to the servers in the DMZ).

3.1 Specification

The pinholes can be expressed via service aliases or via explicit rules

3.1.1 Services aliases

An alias consists in a couple composed of the name of the service and the host or prefix to use as destination. If the service runs on another port than the default one, it can be specified. Moreover, the access can be restricted to a given set of sources (hosts, prefixes or aliases), the default being any.

We defined rules for the following aliases: SSH, HTTP(S), NTP, DNS, SMTP(S), POP3(S), IMAP(S), Windows Shares, NFS, FTP, SNMP, SCTP, DCCP, IPSec (AH+ESP).

The definition of DNS differs from the other aliases, because we need to allow recursion over the Internet. We must thus specify a primary and eventually a secondary server for which the recursion will be permitted.

Moreover, we do not consider here the security that should be performed on the server itself, e.g. disabling recursion for the DNS server if the request is issued by an external node.

All aliases must follow the following RelaxNG Schema:

```
<element name="service">
  <interleave>
        <!-- the name of the service
             acts as the name of the alias
             Aliases for far: http, https, smtp, smtps, ssh, snmp, dns
        -->
        <element name="name">
            <data type="string"/>
        </element>

        <!-- The host on which the service is running
```

```
     if not set, the whole site/subnet is used as destination
-->
<optional>
    <element name="destination">
        <oneOrMore>
            <choice>
                <element name="prefix">
                    <ref name="ipv6-network-repr"/>
                </element>
                <element name="host">
                    <ref name="ipv6-address-repr"/>
                </element>
                <!-- aliases to the variables defined at the start -->
                <element name="internal">
                    <empty/>
                </element>
                <element name="local">
                    <empty/>
                </element>
                <element name="external">
                    <empty/>
                </element>
                <element name="management">
                    <empty/>
                </element>
                <element name="any">
                    <empty/>
                </element>
            </choice>
        </oneOrMore>
    </element>
</optional>

<!-- The port, if different than the default one from the alias -->
<optional>
    <element name="port">
        <ref name="port-or-range-or-service-content"/>
    </element>
</optional>

<!-- The source can be
     default: any
     internal
     local
     external
     management
     a prefix
     a host
     For each source a rule is set
-->
<optional>
    <element name="source">
        <oneOrMore>
            <choice>
                <element name="prefix">
                    <ref name="ipv6-network-repr"/>
```

```
                        </element>
                        <element name="host">
                            <ref name="ipv6-address-repr"/>
                        </element>
                        <!-- aliases to the variables defined at the start -->
                        <element name="internal">
                            <empty/>
                        </element>
                        <element name="local">
                            <empty/>
                        </element>
                        <element name="external">
                            <empty/>
                        </element>
                        <element name="management">
                            <empty/>
                        </element>
                        <element name="any">
                            <empty/>
                        </element>
                    </choice>
                </oneOrMore>
            </element>
        </optional>
    </interleave>
</element>
```

All the aliases must follow the same interface, and it must be possible and easy to add dynamically new aliases. It is important to document each alias correctly to avoid misconfigurations.

3.1.2 Custom rules

It is impossible to have default aliases for all possible services. Therefore we allow the definition of custom rules by subclassing the generic rules presented in section 4.2.2.

3.2 Consistency

As we have several firewalls in our network, the consistency of the security policy between them is important and will require special attention. If the usage of the common bogon and ICMPv6 filtering will ensure part of the consistency, the pinholes are more challenging. We have to propagate the pinholes on all firewall on the path to the border.

As the security and transition engine are meant to work together to achieve out the transition procedure, we have access to the information about the network representation as a graph, and more specifically on the ancestors and successors of a node. Given this information, we need to define an algorithm, that ensures the correct propagation of the pinholes.

Studies addressing the consistency, verification and validation of security policies exist. The algorithms and mechanisms described in these studies can be applied, or modified to be applied, to this consistency to verify that the propagation of the rules is in adequacy to the site policy.

Chapter 4

Implementation

In this section, we present the first thoughts and conclusions on the implementation of the security engine.

4.1 Planned architecture

The security engine will work in cooperation with the transition engine, as we need information about the network graph. It takes the security policy defined in section 2 as a parameter and outputs the rules for the firewalls in XML files respecting the schema defined in 4.2.2. These files can then be translated in the schema corresponding to the firewalls implementation and pushed on the device via Netconf.

Figure 4.1 presents the planned architecture.

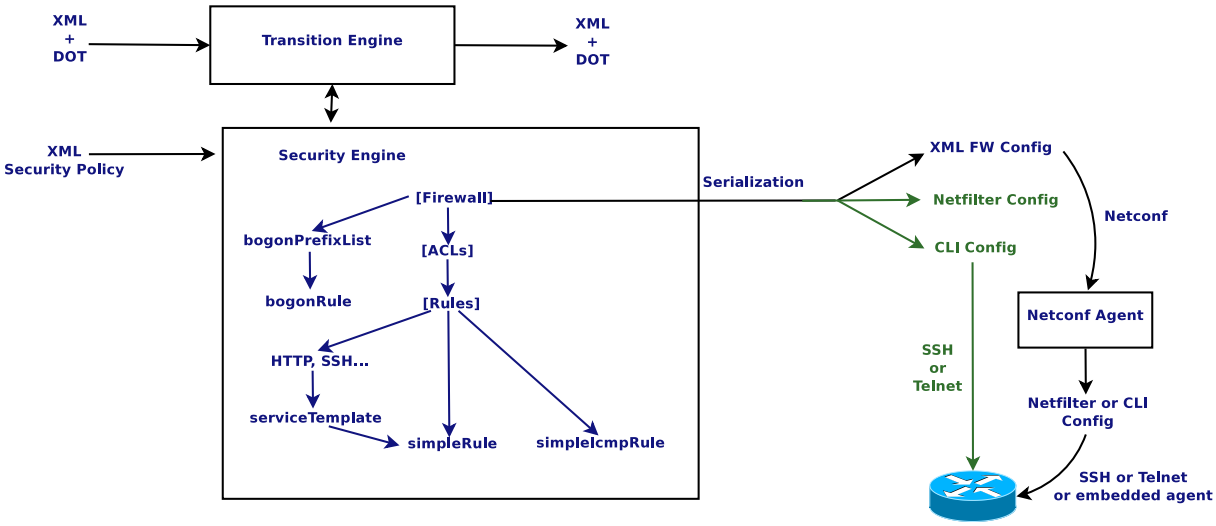

Figure 4.1: Security Engine Architecture

In the current version of the framework, the XML file will be serialized in CLI or Netfilter script and pushed on the devices via SSH or Telnet. The work on Netconf will be kept as a background task.

4.2 XML schemas for firewall rules

As we decided since the beginning of the project to work with XML data models, in order to reuse them later on with Netconf, we define some schemas to handle the firewall rules.

We begin by the schemas for ip6tables and the Cisco IOS firewall, which are the two firewalls implementations we selected in this study. Then, by analyzing both the similarities and our needs, we designed a generic XML schema.

4.2.1 Ip6tables and cisco

We defined two RelaxNG XML schemas for ip6tables version 1.4.4 and Cisco IOS Firewall version 12.4(11)T2. The main difference resides in the architectural organization of the two firewalls.

In ip6tables, the rules are split in several independent tables, each one taking care of a specific task: FILTER is the default table used for filtering, MANGLE is used for packet alteration, and RAW is used for configuring exemptions from connection tracking. In each table, the rules are organized in chains (a chain is an ACL according to the Cisco terminology), and we have some predefined chains (e.g. INPUT, FORWARD and OUTPUT in the FILTER table) or user-defined ones. A chain or a rule can be applied to an interface inbound or outbound traffic. It can also be applied to the transit between two interfaces. Several chains can be set on the same interface, in the same table or not, if the conditions are not overlapping.

In Cisco, the rules are organized in ACLs. An ACL can be set on one or more interfaces in input or output, or both, but we can only set one ACL per interface in each direction.

The difference in organization (ip6tables tables) can be easily bypassed. In our scenario, we can limit ourselves on the FILTER table, and thus make the assumption that all chains are set in that table. If we do so, we have the following representations for a rule allowing a host to connect on the SSH server on another:

```
ip6tables -A INPUT -s 2001:db8:4501:32::2/128  -d 2001:db8:4501:8::1/128
          -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
```

```
<ip6tables>
  <INPUT>
    <rule rulenum="1">
      <source type="host">2001:db8:4501:32::2/128</source>
      <destination type="host">2001:db8:4501:8::1/128</destination>
      <interface-in>eth0</interface-in>
      <protocol>
        <tcp >
          <destination-port type="value">22</destination-port>
        </tcp>
      </protocol>
      <target>ACCEPT</target>
    </rule>
  </INPUT>
</ip6tables>
```

```
permit tcp host 2001:db8:4501:32::2 host 2002:9851:7278:8::1 eq 22
```

```
<access-lists>
  <access-list>
    <name>internal-in</name>
    <interface>
      <name>eth0</name>
      <in/>
    </interface>
    <rule>
      <action>permit</action>
      <source type="host">2001:db8:4501:32::2/128</source>
      <destination type="host">2001:db8:4501:8::1/128</destination>
      <protocol>
        <tcp >
          <destination-port type="value">22</destination-port>
        </tcp>
      </protocol>
    </rule>
```

```
  </access-list>
</access-lists>
```

The main difference that subsists concerns the handling of the interface on which the rule is applied, and the presence of the predefined chains in ip6tables. But, if we correctly define the usage of these differences and adapt the way we use ip6tables to one similar to Cisco, we still manage to express the simple/basic security policy we want to apply.

Thus, we defined a generic representation for firewall rules.

4.2.2 Generic schema

As presented in the previous section, it is possible to express an ip6tables or Cisco firewall in a common representation. In the example, we see that the source/destination and protocol representations are identical. It is the case for all usual protocols (TCP, UDP, ICMP) or options (DSCP, mobility and routing headers). We thus defined a first version of a generic representation for firewalling rules, which takes only the parts identical in both implementations (always making the assumption that all chains are set in the FILTER table of ip6tables).

This gives the following representation for our example:

```
<firewall>
  <stateful/>
  <rpf>
    <strict/>
  </rpf>
  <group>
    <name>internal-in</name>
    <interface-in>eth0</interface-in>
    <default-policy>DENY</default-policy>
    <rule>
      <action>permit</action>
      <source type="host">2001:db8:4501:32::2/128</source>
      <destination type="host">2001:db8:4501:8::1/128</destination>
      <protocol>
        <tcp >
          <destination-port type="value">22</destination-port>
        </tcp>
      </protocol>
    </rule>
  </group>
</firewall>
```

Depending on the configuration method chosen, this representation will be translated to the specific XML schema if we use Netconf, or to a configuration script if we use a traditional CLI configuration mechanism.

We introduced also the notion of default policy, which is an ip6tables terminology. In Cisco, this is translated in a *deny any any* rule with the highest sequence number possible.

When defining the basic security that should be deployed when performing the transition, some new needs in terms of protocols or options have raised (SCTP, DCCP...) and other will too (ESP, AH when we will consider the VPNs). The differences between both implementations in this case are minor. One rarely used option will be present in one and not the other, or they will not define exactly the same aliases for some types for example. But in all cases, it is possible to define a generic representation, not always optimal for one of the firewalls, but that permits to express the rule in a generic way without loosing information and syntactically valid, while keeping the translation to the specific rule easy.

4.3 Site policy

The site global security policy follows a RelaxNG XML schema. It will not be presented here, but it shipped with the source code. It permits to define and tune all the rules we presented in chapters 2 and 3. the interesting part is the parsing of this policy.

1. Parse and create the source/destination aliases

2. Parse the Site-ISP interconnection policy
    - Create Border Firewall
    - Create ACLs IN and OUT on upstream interface
3. Parse subnets policies: for each subnet:
    - Identify subnet, predecessor and predecessor interface
    - Subnet template ?
    - Create Firewall corresponding to the predecessor, IN and OUT ACLs on its interface connected to the subnet, parse and set pinholes
4. On all generated firewalls, create ACLs on the interfaces that do not have any
5. Ensure consistency of pinholes

Adding ACLs on the firewalls interfaces where none was set permits to ensure the firewall is protected in INPUT and OUTPUT on these interfaces as well, and performs the minimum filtering (anti-spoofing, ICMPv6...). As by configuration no security was required on these interfaces, we could set the default policy to accept on these interfaces. This solution works well for Cisco firewalls, as the packet can go through several ACLs automatically. However, in Netfilter, a packet goes through the FORWARD table, and is then redirected to the ACL corresponding to the incoming or outgoing interface, but, unless we set a manual jump to another ACL, the packet will not be filtered by another ACL. Thus, if we set the default policy to ACCEPT in Netfilter, a packet may bypass security rules set on the site. For example, if we have missing ACLs with ACCEPT as default policy set on an interface, and ACLs protecting a DMZ set on the other one, a packet entering the firewall from the missing ACL and directed to the DMZ will match the default rule on the missing ACL and will bypass all the rules set in the DMZ. To avoid this issue, instead of being ACCEPT, the default policy is to jump to the ACL corresponding to the interface to which the packet is forwarded. Therefore, the packet will go through several interfaces and will not bypass any security rule.

4.3.1 ACLs generation

We decided to set ACLs for inbound and outbound traffic on all interfaces on the firewalls. In order to generalize the naming of these ACLs to simplify the implementation, we decided that an interface IFACE will have 2 ACLs called *IFACE-IN* and *IFACE-OUT*.

For example, in figure 4.2, the interface *Gi0/0* has two ACLs defined: *Gi0/0-in* and *Gi0/0-out*. The direction of the inbound and outbound traffic is inverted between the upstream and the other interfaces. In this example, on the upstream interface, outbound traffic is going out on *Gi0/0*, whereas on *Gi0/1* it is coming in.

In Netfilter firewalls, we have 2 extras ACLs, INPUT and OUTPUT, which handle the traffic directly accessed or emitted by the firewall. These ACLs differ from the other ones because they are not linked to an interface.

4.4 Pinholes

4.4.1 Service aliases

All pinholes follow the same interface. In fact, this interface is a python class called *service_template* from which all services inherit. This template defines a default constructor and all the serialization functions.

All the aliases are stored in the directory *./6tea/security/services/*. As long as the defined service name, file name and class name are rigorously identical, the service alias will be dynamically loaded by the security engine without any modification of the engine itself.

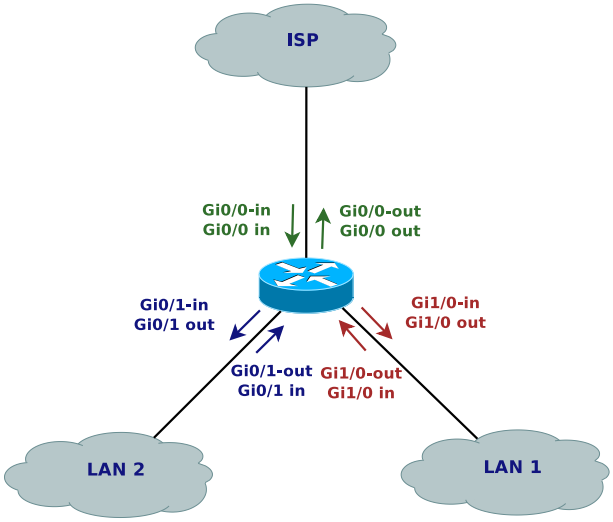

Figure 4.2: ACLs Naming

For example, to add a service alias for HTTP, we create the following *http.py* file and place it in *./6tea/security/services/*.

```
from template import service_template

class http(service_template):

  def __init__(self,id=0,dst=[],source=[],protocol="tcp",port=80,inbound=True,outbound=False):
     service_template.__init__(self, id=id, dst=dst, source=source, protocol=protocol, \
                               port=port, inbound=inbound, outbound=outbound)

  def __repr__(self):
    txt = "HTTP: \n" + service_template.__repr__(self)
    return txt
```

Then, in the security policy XML file, we add the service description:

```
<service>
    <name>http</name>
    <destination>
        <host>2001:660:4501:3208:212:3fff:fe77:74a</host>
    </destination>
</service>
```

As the service name in the XML policy, the file name and the class definition match, the service alias will be handle perfectly.

Some services may require several filtering rules on different ports or firewalls. In that case, the constructor must be rewritten as it was done for example for RIPng, to use directly the class defining a rule, *simpleRule* and create the list of rules to apply.

```
from simple_rule import simpleRule
from template import service_template

class ripng(service_template):
```

```
def __init__(self,id=0,dst=["ff02::9"],source=[],port=521,inbound=True,outbound=False):
  self.rules = []
  self.ruleId = id

  if inbound:
    if len(dst) == []:
      dst=["ff02::9"]

    for d in dst:
      r = simpleRule(id=self.ruleId, dst=d, protocol="udp", port=port, inbound=inbound, outbound=outbound)
      self.rules.append(r)
      self.ruleId = self.ruleId+1
      r = simpleRule(id=self.ruleId, dst=d, protocol="tcp", port=port, inbound=inbound, outbound=outbound)
      self.rules.append(r)
      self.ruleId = self.ruleId+1
  else:
    for s in source:
      r = simpleRule(id=self.ruleId, dst="ff02::9", protocol="udp", port=port, inbound=inbound, outbound=outb
      r.source = IP(s)
      self.rules.append(r)
      self.ruleId = self.ruleId+1
      r = simpleRule(id=self.ruleId, dst="ff02::9", protocol="tcp", port=port, inbound=inbound, outbound=outb
      r.source = IP(s)
      self.rules.append(r)
      self.ruleId = self.ruleId+1
      r = simpleRule(id=self.ruleId, dst=None, protocol="udp", port=port, inbound=inbound, outbound=outbound)
      r.source = IP(s)
      self.rules.append(r)
      self.ruleId = self.ruleId+1
      r = simpleRule(id=self.ruleId, dst=None, protocol="tcp", port=port, inbound=inbound, outbound=outbound)
      r.source = IP(s)
      self.rules.append(r)
      self.ruleId = self.ruleId+1

def toCLI(self, reflect=None, inbound=False, outbound=False):
  return service_template.toCLI(self, reflect=None, inbound=inbound, outbound=outbound)


def __repr__(self):
  txt = "RIPNG: \n" + service_template.__repr__(self)
  return txt
```

In this example, we created rules for UDP and TCP port 521, for the given list of source and destination, and the reserved multicast address FF02::9. On top of that, we modified the *toCLI* function to force the reflect parameter to None. This parameter permits the use of reflexive ACLs on Cisco firewall, which must be set explicitly, unlike Netfilter where the state module handles it automatically. By default, all pinholes generate an entry in the global reflexive ACL on the firewall.

4.4.2 Consistency

All pinholes opened on a firewall ACL (even on the firewall INPUT and OUTPUT ACLs for management operations) must be propagated through the network graph up to the root. Pinholes set in a subnet following the *local* template are also propagated to the root but not on the interconnection with the ISP.

When setting a pinhole for a subnet, we first propagate it locally on the upstream inbound interface. Then we go to the predecessor firewall and do the same, until the border firewall is reached. Outbound

traffic in response will be automatically permitted thanks to the connection tracking system.

4.5 Serialization

All objects in the implementation (firewall, ACL, service alias, bogon prefix list...) follow the same serialization interface:

toCLI Serialize into Cisco CLI

toNetfilter Serialize into Netfilter script using the ip6tables command

toXML Serialize into the generic firewall representation

To ensure that all filtering rules are processed in the right order, each one of them is assigned a rule ID (sequence number in CLI terminology), and the rules are sorted regarding this rule ID before serialization.

4.6 GUI integration

The security engine has been integrated with the Transition Engine via its graphical interface. To do so, we created two new tabs.

The Security tab permits to load the security policy XML configuration file, and displays information about the site global policy. Once the policy is loaded, the new network map is updated to show which routers have a firewall activated. With the following security policy example, we obtain the new network shown in figure 4.3.

```
<security-policy>
  <management>
    <host>2001:660:4501:1:213:72ff:fe35:745</host>
  </management>
  <internal>
    <prefix>2001:660:4501:3200::/56</prefix>
    <prefix>fdcd:2141:44be::/48</prefix>
  </internal>

  <site>
    <bogon-filtering>relaxed</bogon-filtering>
    <stateful/>
    <icmpv6>strict</icmpv6>
    <rpf>strict</rpf>
    <restricted>
      <service>
        <name>https</name>
        <destination>
          <host>2001:200:0:8002:203:47ff:fea5:3085</host>
        </destination>
      </service>
      <service>
        <name>ssh</name>
        <source>
          <management/>
        </source>
      </service>
    </restricted>
    <blacklist>
```

```
    <host>2001:660:4501:123::1</host>
    <prefix>2001:660:4501:456::/64</prefix>
  </blacklist>
  <services>
    <service>
      <name>telnet</name>
    </service>
  </services>
</site>

<subnet>
  <id>lan1</id>
  <dmz/>
  <restricted>
    <service>
      <name>http</name>
    </service>
  </restricted>
  <services>
    <service>
      <name>http</name>
      <destination>
        <host>2001:660:4501:3208:212:3fff:fe77:74a</host>
      </destination>
    </service>
    <service>
      <name>ssh</name>
      <host>2001:660:4501:3208::/64</host>
      <source>
        <management/>
      </source>
    </service>
  </services>
</subnet>

<subnet>
  <id>lan4</id>
  <local/>
  <services>
    <service>
      <name>ftp</name>
      <source>
        <internal/>
      </source>
    </service>
  </services>
  <rules>
    <rule>
      <action>ALLOW</action>
      <source type="network">2001:660:4501:3200::/56</source>
      <protocol>
        <tcp>
          <destination-port type="value">12345</destination-port>
        </tcp>
      </protocol>
    </rule>
```

```
    </rules>
  </subnet>

  <subnet>
    <id>lan5</id>
    <nat/>
    <services>
      <service>
        <name>ssh</name>
      </service>
    </services>
  </subnet>
</security-policy>
```

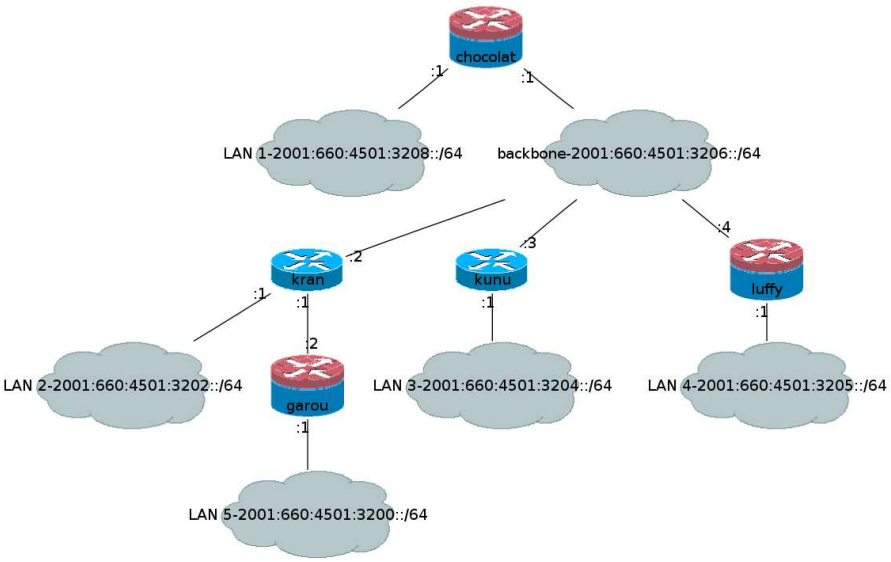

Figure 4.3: Secured Network

The Firewall Details tab shows the detailed information about each newly generated firewall. It gives access to the global configuration, or from a per ACL point of view. It is possible to configure independently each firewall, or all of them at the same time via the security tab. It is possible to save each firewall or ACL configuration in script (CLI or Netfilter) or XML representation via the firewall details tab. Figure 4.4 illustrates such a tab.

At the same time, we implemented the transition procedure:

1. Launch 6Te@ and load the XML and DOT configuration files for the Transition Engine
2. Go to the addressing tab and run the engine
3. Check the proposed addressing plan via the new network and router/network details tabs
4. Validate the addressing plan in the addressing tab
5. Secure the network in the addressing tab
6. Load the site policy XML file in the security tab
7. Check in the new network and firewall details tab the security rules

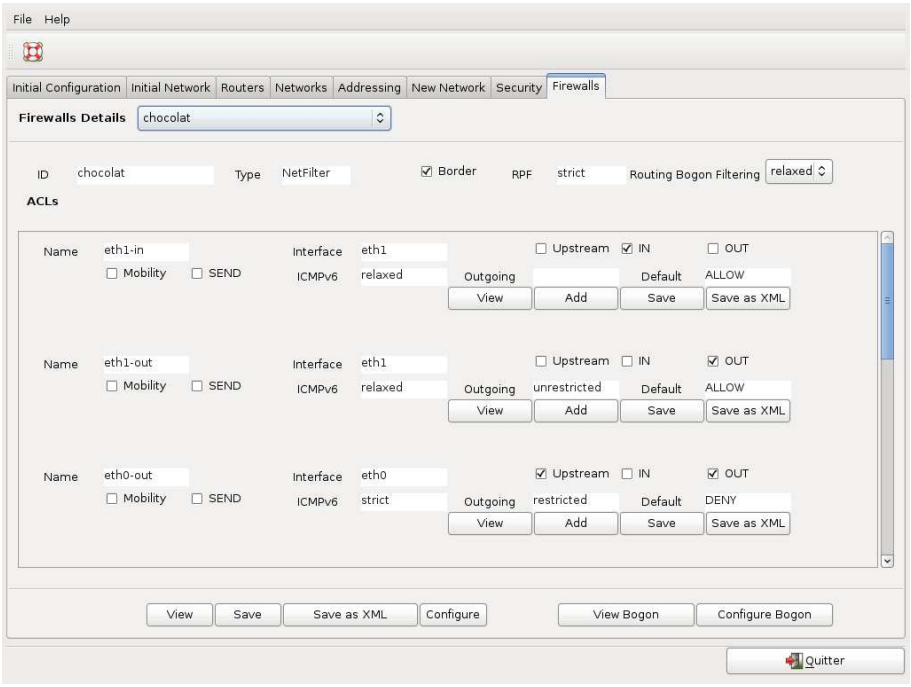

Figure 4.4: Firewall Details

8. Configure the routing bogon filtering and the firewalls in the security tab
9. Configure the routers in the addressing tab
10. Eventually, save the configuration in the security tab
11. The network transition is performed, quit

The tool does not permit a one click transition, as the administrators validation is required before configuring the network devices, but once he configured everything according to his needs and requirements, the tools performs all the operations automatically.

4.7 Future work

Some minor improvements must still be performed. At the moment, we set the rules everywhere without checking that the source or destination prefix match the aggregated prefix assigned on an interface. This does not create security holes, but simply adds unnecessary rules.

For example, if the management alias contains a host which address is outside the site prefix, it would be only necessary to create the ACL on the firewall's upstream interface.

Chapter 5

Versus simple/advanced security drafts

Two IETF drafts address simple [7] and advanced [13] security in Customer Premise Equipments (CPE) residential gateways. These proposition are focused on home residential network, where a single subnet is present, and where the IPv4 network is using NAT technologies. These drafts have some interesting recommendations, and we share a subset of these in our study.

However, we are dealing with a more complex scenario, as we are focused on small and medium business networks. We address the same problem, at a larger scale. Thus, we have to consider several subnets, several routers, and routing protocols, which are not considered in these drafts. Moreover, the subnets in our network can have different roles (DMZ, NAT, local) that require specific and different security rules. This implies that we can have several firewalls and that we have to deal with the consistency of the filtering on all of them, to ensure a correct propagation of the pinholes.

As we are oriented on business networks, our approach is more restrictive. The default policy and filtering is tighter in our study, but via configuration the same pinholes can be opened. For example, when in residential networks the default policy for outgoing traffic would be to allow everything, we prefer to deny everything that is not explicitly permitted.

We still need to investigate some points (see section 6), and we are aiming at publishing these recommendations as an Internet Draft, which is why any feedback is welcomed.

Chapter 6

Next steps and future work

We still need to address several points in this study.

First of all, we need to improve some aspects of the security policy. For example, we want to define some templates for ICMPv6 as we did for the bogon filtering, to permit or deny ping6 and other messages that may permit to scan the network and infer its internal topology easily. Another template would concern Mobile IPv6.

We did not consider the interconnection between the border router and the IPv6 provider in the transition engine, but we need to consider it in the security engine. Thus, we need to define the rules to apply if a tunneling mechanism such as 6to4 is used to interconnect with the IPv6 Internet [11], or if relays are deployed within the site.

Other points that we must investigate are IPv6 VPNs and other IPv6-in-IPv6 tunneling mechanisms. We must define the filtering rules that will permit the usage of IPv6 VPNs within the site, without opening security holes where the filtering rules may be bypassed by the encapsulation.

Finally, we want to translate the generic firewall representation in the Yang language and propose it in the Netconf community. This model will be implemented in the Ensuite Netconf framework [1].

[1] `ensuite.sf.net`

Chapter 7

Conclusion

During this study, we proposed several algorithms performing the transition of a network from IPv4 to IPv6. In this report, we presented the operations required to secure this transition.

We proposed the minimal set of filtering rules that should be set on a newly transitioned SME network. We also proposed a global site policy definition, and presented how it can be deployed on a network.

Finally, we presented the implementation of these filtering rules within 6Te@, our transition engine, offering a tool capable of performing the transition of a network to IPv6, from the addressing to the security of this new network.

Bibliography

[1] J. Abley, P. Savola, and G. Neville-Neil. Deprecation of Type 0 Routing Headers in IPv6. RFC 5095 (Proposed Standard), December 2007.

[2] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971 (Proposed Standard), March 2005.

[3] E. Davies and J. Mohacsi. Recommendations for Filtering ICMPv6 Messages in Firewalls. RFC 4890 (Informational), May 2007.

[4] G. Van de Velde, T. Hain, R. Droms, B. Carpenter, and E. Klein. Local Network Protection for IPv6. RFC 4864 (Informational), May 2007.

[5] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFC 5095.

[6] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh. NAT Behavioral Requirements for TCP. RFC 5382 (Best Current Practice), October 2008.

[7] Ed. J. Woodyatt. Simple Security Capabilities in Customer Premises Equipment for Providing Residential IPv6 Internet Service. draft-ietf-v6ops-cpe-simple-security-08, October 2009.

[8] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[9] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), September 2007.

[10] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), September 2007.

[11] P. Savola and C. Patel. Security Considerations for 6to4. RFC 3964 (Informational), December 2004.

[12] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007.

[13] E. Vyncke and M. Townsley. Advanced Security for IPv6 CPE. draft-vyncke-advanced-ipv6-security-00, October 2009.

[14] M. Wasserman and F. Baker. IPv6-to-IPv6 Network Address Translation (NAT66). draft-mrw-behave-nat66-02 (Internet-Draft), November 2008.