

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Telecommunications Software and Multimedia Laboratory

HOST IDENTITY PROTOCOL ENABLED FIREWALL

A Prototype Implementation and Analysis

Master's Thesis

Essi Vehmersalo

Telecommunications Software and Multimedia Laboratory
Espoo 2005

Author:	Essi Vehmersalo	
Title of thesis:	HOST IDENTITY PROTOCOL ENABLED FIREWALL A Prototype Implementation and Analysis	
Date:	September 6 2005	Pages: 10 + 70
Professorship:	Telecommunications Software	Code: T-110
Supervisor:	Professor Antti Ylä-Jääski	
Instructor:	Janne Lindqvist, M.Sc. (Tech.)	
<p>This thesis focuses on interactions between network firewalls and Host Identity Protocol (HIP). HIP is an emerging technology that introduces cryptographic Host Identities (HI) into the Internet architecture. As a result it has profound effects on security as well as mobility and multihoming.</p> <p>The objective of this thesis was to design and implement a firewall prototype that enables traffic filtering based on the HIP HIs and is able to correctly filter HIP traffic. Furthermore this thesis analyzed the effects that different aspects of HIP have on firewall technologies. These include for example added security features, such as traffic authentication, and more flexible handling of mobility and multihoming.</p> <p>The implemented firewall meets the requirements defined for it. Traffic filtering is done based on HIs of the end-hosts and cryptographic properties of the HI are used for authenticating the HIP control traffic. The firewall also implements stateful connection tracking mechanisms to associate the Encapsulating Security Payload (ESP) data traffic to the HIP association. Possible directions for further development and research are also identified. These include for instance extending the firewall to include the HIP registration capability and further research of firewall performance and efficiency.</p>		
Keywords:	HIP, HI, firewall, middlebox, packet filtering	
Language:	English	

Tekijä: Essi Vehmersalo	
Työn nimi: HOST IDENTITY PROTOCOL ENABLED FIREWALL A Prototype Implementation and Analysis	
Päiväys: 6. syyskuuta 2005	Sivumäärä: 10 + 70
Professori: Tietoliikenneohjelmistot	Koodi: T-110
Työn valvoja: Professori Antti Ylä-Jääski	
Työn ohjaaja: DI Janne Lindqvist	
<p>Tämä diplomityö keskittyy Internet palomuurien ja Host Identity Protocol (HIP) -teknologian väliseen yhteistoimintaan ja vuorovaikutukseen. HIP on kehitteillä oleva teknologia joka tuo kryptografiset, päätelaitteita kuvaavat identiteetit, Host Identity (HI), Internet arkkitehtuuriin. Siksi HIP:lla onkin syvällisiä vaikutuksia sekä turvallisuuteen että liikkuvuuden hallintaan.</p> <p>Tämän diplomityön tavoitteena oli suunnitella ja toteuttaa palomuuriprotyyppi, joka mahdollistaa liikenteen suodattamisen päätelaitteiden identiteettien perusteella sekä kykenee käsittelemään HIP-liikennettä. Lisäksi diplomityö analysoi HIP:n eri osa-alueiden ja toimintojen vaikutusta palomuuritekniologioihin. Näihin kuuluvat esimerkiksi uudet turvallisuusominaisuudet, kuten liikenteen luotettava tunnistaminen, sekä liikkuvuuden joustavampi käsittely.</p> <p>Työn tuloksena syntynyt palomuuuri toteuttaa sille asetetut vaatimukset. Liikenne suodatetaan päätelaitteiden identiteettien perusteella ja identiteettien kryptografisia ominaisuuksia käytetään hyväksi HIP-liikenteen autentikoinnissa. Palomuuuri toteuttaa myös tilaa ylläpitävän yhteyksien valvonnan, jonka avulla Encapsulating Security Payload (ESP) -data-liikenne kyetään liittämään tiettyyn HIP-yhteyteen. Myös mahdollisia jatkokehitys ja -tutkimus aiheita on nimetty. Näitä ovat esimerkiksi HIP rekisteröintitoiminnon lisääminen palomuuuriin sekä palomuurin tehokkuuden ja suorituskyvyn tutkiminen edelleen.</p>	
Avainsanat: HIP, HI, palomuuuri, pakettisuodatus	
Kieli: Englanti	

Acknowledgements

This thesis was done at HUT Telecommunications Software and Multimedia Laboratory (TML) as part of the joint InfraHIP project of TML and Helsinki Institute for Information Technology (HIIT). I would like to thank my supervisor, Professor Antti Ylä-Jääski, and instructor, M.Sc. (Tech.) Janne Lindqvist, for their valuable feedback and guidance. Also, the entire InfraHIP project group has provided important views and insights to HIP and related aspects, special thanks to Miika Komu for his help with the HIPL.

Last but not least I would like to thank my friends and family and especially T-P for the love and support and the refreshing and inspiring distractions during the writing of this thesis.

Helsinki, September 6th 2005

Essi Vehmersalo

Abbreviations and Acronyms

DoS	Denial of Service
DSA	Digital Signature Standard
ESP	Encapsulating Security Payload
HI	Host Identity
HIP	Host Identity Protocol
HIPL	Host Identity Protocol for Linux
HIT	Host Identity Tag
HMAC	Keyed-Hashing for Message Authentication
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IRTF	Internet Research Task Force
IP	Internet Protocol
IPsec	Internet Protocol security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
NAT	Network Address Translation
RSA	Rivest Shamir Adleman public key algorithm
RVS	Rendezvous Server
SA	Security Association
SPI	Security Parameter Index
SPKI	Simple Public Key Infrastructure
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network

Contents

Abbreviations and Acronyms	v
1 Introduction	1
1.1 Problem Statement	2
1.2 Scope	3
1.3 Organization of the Thesis	3
2 Background	4
2.1 Firewalls	4
2.1.1 Firewalls as Network Elements	4
2.1.2 Different Functionalities Provided by Firewalls	5
2.2 Host Identity Protocol (HIP)	8
2.2.1 Protocol Overview	8
2.2.2 Mobility and Multihoming	9
2.2.3 Registration protocol	10
2.2.4 Rendezvous	10
2.3 Summary	11
3 HIP Enabled Firewalling	12
3.1 HIP with firewalls	12
3.1.1 Basic Functionality	13
3.1.2 Registration Protocol	15
3.1.3 HIP Rendezvous Service	17

3.1.4	Mobility and Multihoming	17
3.1.5	Asymmetric Routing	20
3.1.6	HIP Certificate Parameter	22
3.1.7	Potential Security Vulnerabilities	22
3.2	Scenarios for HIP Enabled Firewall	24
3.2.1	General Firewalling Scenario	24
3.2.2	Road Warrior and Virtual Private Network (VPN) So- lution	25
3.3	Summary	25
4	Requirements	27
4.1	Functional Requirements	28
4.1.1	Firewall Policy Management	28
4.1.2	Overall Functionality	29
4.1.3	Stateless Packet Filtering	29
4.1.4	Stateful Packet Filtering	30
4.2	Non-Functional Requirements	31
4.2.1	Requirements Regarding the Design	31
4.2.2	Security	32
4.2.3	Secondary Requirements	32
4.3	Summary	33
5	Design	34
5.1	Design Alternatives	34
5.1.1	Linux Netfilter Extension	34
5.1.2	Independent HIP Firewall Solution	39
5.2	HIP Enabled Firewall Design	39
5.2.1	Firewall Main Module	40
5.2.2	Firewall Policy Management	41
5.2.3	Packet Filtering Functionalities	41
5.2.4	Connection Tracking	42

5.3	Summary	43
6	Implementation	44
6.1	External Components	44
6.2	Firewall Main Module	45
6.3	Packet Filtering Functions	45
6.4	Firewall Policy Management	45
6.4.1	Data Structures	46
6.4.2	Parsing Rules	47
6.4.3	Interface for Firewall Management	47
6.5	Connection Tracking	48
6.5.1	Functionality	49
6.5.2	Data Structures	50
6.6	Interaction Between Components	51
6.7	Summary	53
7	Analysis	54
7.1	Evaluation Against Requirements	54
7.1.1	Test and Development Setting	54
7.1.2	Overall Functionality and Interfacing to the Commu- nication System	56
7.1.3	Firewall Policy Management	56
7.1.4	Stateless Packet Filtering	57
7.1.5	Connection Tracking	57
7.1.6	Non-Functional Requirements	59
7.2	General Analysis of HIP Enabled Firewalling	59
7.2.1	Role of HIP Enabled Firewall	60
7.2.2	Registration Requiring Firewall	60
7.2.3	HIP Protocol Implications to Firewall Design and Im- plementation	61
7.3	Summary	62

8	Conclusions	63
8.1	Future Work	64
8.1.1	Supporting Updated HIP Specifications	64
8.1.2	Extending Firewall to Include Registration	65
8.1.3	Production Level Firewall Solution	65

List of Figures

2.1	HIP base exchange	9
2.2	HIP base exchange through Rendezvous Server (RVS)	11
3.1	State maintained by intermediate firewalls	16
3.2	HIP mobility signaling	19
3.3	Asymmetric routing scenario	20
3.4	Asymmetric routing scenario with SPISIG signaling	21
4.1	Format of the firewall rule	28
4.2	Format of stateless filtering options	29
4.3	Format of the state option	30
5.1	High level architecture	35
5.2	Netfilter hooks and packet traversal	36
5.3	A simplified connection table structure of conntrack module	37
5.4	Overall design of the HIP enabled firewall	40
6.1	Rule data structure	46
6.2	Functions for managing firewall rules	48
6.3	Connection tracking data model	50
6.4	Overall functional sequence of the system	52
7.1	Test setting	55

Chapter 1

Introduction

Host Identity Protocol (HIP) [22, 25] is a proposed protocol for providing security, mobility and multihoming to the current Internet architecture. HIP establishes a new name space of Host Identities (HI) for representing hosts independent of their locations in the network. By contrast, the current architecture uses the Internet Protocol (IP) addresses to identify hosts.

The host identity implies also a change in the Internet architecture. HIP creates a conceptual layer between the network and transport layers. With this architectural change, transport level associations can be bound to an identifier representing the host rather than the topological location of it. This has profound impacts on handling mobility and multihoming [24]. The current Internet architecture is based on the assumption that hosts have a single static network attachment point. This was a reasonable supposition at the time Internet architecture was developed. However, it no longer applies as mobility and multihoming are becoming all the more viable and desirable in networking.

Possibly the most significant contribution of HIP is the inherent security it adds to the architecture. The identity of the host is in fact the public key of a cryptographic public/private key pair possessed by the host. Accordingly, the very identity of the host can be used for authenticating its owner. This makes HIP essentially different from other available communication protocols. It adds deeply embedded security into Internet communication.

The other main theme of research in this thesis are Internet firewalls. Firewalls protect networks by filtering traffic passing through them, to and from the protected network. To enhance and ensure security, firewalls analyze properties of the intercepted traffic.

In order to perform filtering reliably, a firewall needs to be able to trust the validity of the information it analyzes in the packet headers. Furthermore, probably the most important filtering criteria are the source and destination end-point of the traffic. This is, however, problematic as the current end-point identifier, the IP address, is by nature insecure and can easily be forged to impersonate another host.

This thesis analyzes implications and effects of HIP in the context of Internet firewalls. As HIP is a relatively new proposed technology, it is essential to evaluate what consequences it has on different aspects of the existing architecture and already well established technologies. Furthermore, the security and the architectural restructuring provided by HIP are likely to have an impact also on firewall technologies. Therefore, the general motivation of this thesis is not simply developing a firewall that allows HIP traffic to traverse. Instead, the thesis studies the implications and possible benefits that HIP protocol has from the point of view of firewall technologies.

1.1 Problem Statement

The filtering mechanisms of current firewalls are largely based on the IP address as an end-point identifier. In effect, they are often vulnerable to IP address spoofing if the firewall is not able to properly authenticate the end-point.

Furthermore, firewalls are a widely established security mechanism. The current firewalls do not, however, support filtering of HIP traffic. As a consequence, HIP traffic is in general blocked by firewalls [30, 31]. Still, traversing middleboxes is a necessary property for any protocol in order for it to be successfully deployed. The enhanced security provided by HIP could therefore be an important motivation for adding HIP support into firewalls. This could in turn aid the deployment of HIP.

This thesis analyzes the possibilities and effects of traffic filtering based on HIP host identities. The objective is to design and implement a firewall solution for using host identities for access control decisions. Accordingly, the firewall will need to filter traffic based on HIP associations. This will require maintaining necessary state regarding the HIP association. The firewall will also take advantage of the traffic authentication mechanisms provided by HIP. Thesis will further analyze different aspects of HIP protocol with regard to firewall interoperability and interactions. This includes issues such as mobility, asymmetric routing and denial of service vulnerabilities.

1.2 Scope

The analysis of the thesis covers different aspects of HIP protocol and a HIP enabled firewall should be implemented with regard to these aspects. As HIP is an emerging new technology, all parts of the protocol are not yet covered by the existing HIP implementations. In general, the HIP for Linux (HIPL) [1, 7] is used as the reference implementation. Therefore, firewall features may be implemented to the extent that the protocol implementation covers them. The design of the system should, however, consider the analyzed issues in order to be extendable to include them in the future.

The thesis provides a prototype implementation of the HIP enabled firewall. Therefore the implementation will serve as a proof of concept rather than a production level firewall.

1.3 Organization of the Thesis

Rest of this thesis is organized as follows. Chapter 2 presents background information on the field of the thesis. In Chapter 3 different aspects of HIP enabled firewalling are analyzed and this also provides basis for design and implementation of a HIP enabled firewall. Chapter 4 describes the requirements for the implementation. The design of the solution is presented in Chapter 5 and the implementation related issues in Chapter 6. The solution is analyzed in Chapter 7 and Chapter 8 will then present conclusions of the thesis along with directions for possible future research.

Chapter 2

Background

This chapter describes closely related background information on the field of the thesis. The chapter gives an overall introduction to firewalls with their functionalities and requirements. HIP protocol functionality is also presented on general level. Additional aspects of HIP, that have effects on firewall functionalities are then described in more detail. These concern for example additional functionalities that HIP can provide to firewalling or different types of traffic that a HIP enabled firewall may encounter.

2.1 Firewalls

In general, firewalls are entities that in some way screen network traffic and accordingly filter out unwanted traffic [6]. Within this definition there is quite a diversity of firewall solutions operating on different levels of protocol stack and providing different sets of functionality. Furthermore, firewall may also be a separate, designated network entity or it may be integrated together with other functionalities of a network node. The following chapters first discuss the nature of firewalls as network elements. This provides basic constraints and requirements that will be further addressed in the rest of the thesis. Also some of the functionalities provided by firewalls are then identified.

2.1.1 Firewalls as Network Elements

Firewalls are generally used as a security perimeter protecting a certain part of network. The protected part of network may range from a single host to a large segment containing several subnets. However, firewalls enable

defining and upholding level of security for all these cases in a centralized manner. With a limited number of designated firewalls, an organization is less dependent on security measures of multiple individual end-hosts. This also evens out the general asymmetric setting between an attacker and a target: The target needs to cover all potential vulnerabilities, but for an attacker finding a single security hole suffices. With centralized security, also the target is able to limit the number of potential vulnerabilities. Accordingly, firewalls are a widely deployed security mechanism.

There is, however, controversy regarding the role of firewalls as network elements. One of the guiding design principles of Internet, the end-to-end argument [28], states that some essential functions can and should only be performed by the end-hosts. Consequently, the communication between end-points should not be dependent on intermediate network elements. This implies that there should not be additional complexity in the intermediate network elements that enables packets to traverse. However, the very purpose of firewalls is to uphold security by blocking traffic [8]. Furthermore, the firewall's ability to let through legitimate traffic often requires understanding the particularities of certain protocols or even maintaining state information regarding the traffic.

There is an apparent need for a compromise between the requirement for centralized security and the end-to-end argument. Even though the end-to-end argument certainly holds value as a design principle, centralized security mechanisms are necessary for organizations to uphold unified security policies. Therefore the need for security provided by firewalls is understandable and there needs to be a balance between these two requirements. This issue is addressed by the transparency rule [13], stating that a firewall must not interfere with legitimate, standards-compliant traffic. In effect, up keeping the transparency should be an important principle in firewall design. Furthermore, even though the burden of transparency rule is ultimately on the firewall implementer, also the design of a protocol should consider different middleboxes, such as firewalls. If not for purely architectural reasons, then from the self-serving reason of aiding the future deployment of the new technology.

2.1.2 Different Functionalities Provided by Firewalls

Firewalls may perform different sets of functionalities for screening and filtering packets. Also, there is no actual standardization for firewalls and therefore the naming conventions are somewhat variable. The following presents

general issues and the main categories of firewall functionality. This categorization concentrates on features that are most closely related to HIP enabled firewalls although other functionalities exist as well.

Policy vs. Functionality

When considering the functionality of a firewall, it is first necessary to separate the concepts of the filtering functionality and the firewall security policy. Here, the filtering functionality refers to the actual mechanisms used for filtering, whereas the policy defines which of the filtering mechanisms are used and which packets they are applied to. Referring back to the previous chapter, it is legitimate to block anything that would be considered malicious by the firewall policy. However, related to the transparency rule, when certain type of traffic is allowed by the policy, there must be nothing in the filtering mechanisms that blocks that traffic.

For the most part, this thesis concentrates on the filtering functionality. It analyzes how these functionalities should be implemented with regard to HIP protocol so that they can be used to enforce different policies. Policies are used to describe more concrete scenarios, where HIP enabled firewalls could be of use. In effect, the policies are dependent on the security requirements of a particular organization or network in question.

Level of Protocol Stack

Firewalls function on different levels of the protocol stack [9]. On the network access layer, the Media Access Control (MAC) addresses could be analyzed. In the network layer logical filtering attributes would be IP addresses and for example properties of the Internet Control Message Protocol (ICMP) packets. Transport layer protocols can be analyzed through protocol ports and other protocol information. On application level, firewalls may focus on details of certain application level protocols, such the Hypertext Transfer Protocol (HTTP) or e-mail protocols. Firewall functionalities may also stretch over several layers.

Stateless vs. Stateful Firewalls

Another categorization of firewall functionalities depends on whether state is maintained by the firewall [26, 10]. Stateless firewalls provide filtering based on static information available in the traffic. This may include source or

destination IP addresses, protocols and ports used, as well as other properties of the packet.

Stateless packet filtering is not as effective and sophisticated method as the stateful inspection of packets. It is however more efficient as it often requires less computation and lacks the effort of storing state related to packets. As the security provided by stateless packet filters is quite minimal, they are in many cases insufficient for protecting a network.

A stateful firewall bases filtering decisions on the connection that traffic relates to. Therefore, traffic can be filtered by whether it is starting a new connection or belongs to an existing one. A necessary concept for state keeping is a flow identifier. Flow identifier is the piece of information that firewall uses to recognize traffic that belongs to a certain connection. In general, this information includes at least the protocol used and identifiers for connection end-points. A flow identifier could include for example Transmission Control Protocol (TCP) as protocol and IP addresses and ports used by the communicating end-points.

Stateful filtering can also be extended to traditionally stateless protocols, such as User Datagram Protocol (UDP). A packet can be interpreted as belonging to same connection, if it shares the same protocol and end-points as the previous packets. In this case the firewall must remove the state for example after a timeout value as there is no explicit sign of closing the connection.

Stateful firewalls can be further divided based on the nature of state keeping [9]. With hard state, the connection between end-hosts must be restarted, if the intermediate firewall loses its state. In the soft state approach the state in the firewall is regularly refreshed and at the same time also recreated, if necessary.

A traditional part of firewalls performing stateful filtering is also analyzing the state transitions that the intercepted packets cause in the connection. Corresponding functionalities for TCP and general principles also applicable to other stateful protocols are described in [36]. Analyzing the state transitions allows filtering out packets that are invalid in the context of the protocol. Invalid packets may be sent by potentially malicious nodes trying to inject packets in to the connection. This could be a sign of trying to impersonate one end-point of the connection or attempting to cause state transition that would break the connection between the original hosts.

Transparent vs. Explicit

Firewalls may also be categorized based on whether they explicitly communicate with the end-hosts. With transparent firewalls the end-hosts may remain unaware of the firewall. Explicit firewalls, on the other hand, require either or both of the end-hosts to communicate directly with the firewall. A stateful firewall that is transparent to the end-hosts is often called a dynamic packet filter [10]

2.2 Host Identity Protocol (HIP)

HIP was already briefly introduced in Chapter 1. Here it is presented more extensively, with emphasis on issues that may influence firewall functionalities. HIP is currently being specified by the HIP working group of the Internet Engineering Task Force (IETF) and the research group of the Internet Research Task Force (IRTF).

The new identifier, HI, introduced by HIP, enables referring to hosts independent of their location and number of network attachment points [22, 25]. In contrast, the current architecture uses the IP address, describing hosts location in the network topology, also as an identifier. In effect, the new name space corresponds to a new host identity layer between the transport and network layers. This enables binding transport layer associations to the invariable HI instead of the potentially inconstant location.

As mentioned, the Host Identity namespace also provides unique security properties, as a cryptographic public key of a host is used as its HI. A more convenient format Host Identity Tag (HIT), a 128 bit hash of the HI, is used to refer to a host in communication. This way the public key is inherently bound to its owner's identity, unlike in several other technologies.

All HIP protocol packets, except for the connection initialization, are cryptographically signed by the sender. This enables the receiver to authenticate the traffic. For the convenience of middleboxes, the signature and lot of the other protocol information are not encrypted, but are visible also for intermediate network entities.

2.2.1 Protocol Overview

The basic functionality of HIP is defined in detail in the HIP Internet-Draft [22]. HIP association is established with a four-way base exchange procedure.

The base exchange as well as the following data traffic is illustrated in Figure 2.1. The initiator host first requests connection with I1 packet. The request may contain the HIT of recipient or when an initiator does not know or does not need to have control over the selection of the receiver HI, it can be left out. The latter case is referred as opportunistic mode of operation.

The responder host answers with R1 packet containing among other things a puzzle for the initiator to solve. Initiator responds with I2 containing the solution. Responder finally verifies the solution and responds with R2 packet. The last three packets of the base exchange also comprise a Diffie-Hellman key exchange and contain signatures for authenticating the sender. Another noteworthy part of the protocol is that with pre-computed R1 messages the responder can defer state creation until receiving the I2 packet. This, together with the cost imposed on the initiator by the puzzle solving, provides resistance to Denial of Service (DoS) attacks.

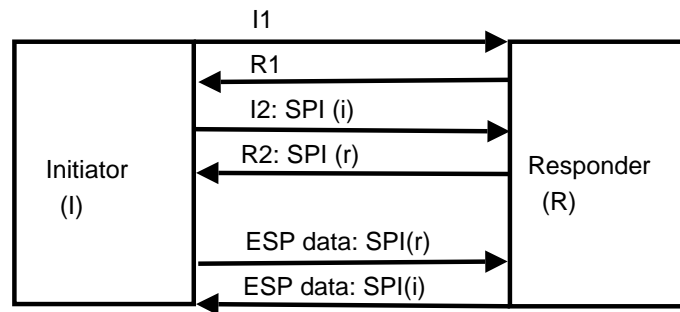


Figure 2.1: HIP base exchange and the following data traffic

The actual data traffic of HIP connection uses IPsec Encapsulating Security Payload (ESP) [17] security association. The payload data is shielded inside ESP as defined in [16]. The ESP Security Parameter Index (SPI) values of the parties are delivered in I2 and R2 packets as illustrated in Figure 2.1. The SPIs are used to identify the HIP association that data traffic belongs to. Externally the HIP data traffic appears as regular IPsec traffic whereas the HIP control packets include the HIP protocol header.

2.2.2 Mobility and Multihoming

As a host may have several network interfaces and may change its location in the network, the location is represented by a dynamic set of IP addresses. To enable mobility and multihoming, HIP allows end-points to signal each other

the changes in the set of IP addresses they can be reached through. The multi addressing functionalities are specified in the mobility and multihoming draft [24] of the HIP working group.

The signaling is done with UPDATE packets, which are also used for updating the SPIs of the HIP association. The end-point wishing to announce changes sends to the other party an update message, including a sequence number that the recipient must acknowledge. The recipient responds with the acknowledgement and an echo request parameter. The echo request is used to check the reachability of the other end-point in the newly announced address. The initial host finally responds with an echo response parameter. The address reachability verification is used to prevent flooding attacks where a host would redirect traffic intended for itself to IP address of another host.

2.2.3 Registration protocol

A general purpose registration protocol is also proposed as part of the HIP standardization [19]. The registration protocol can be used by end-hosts to obtain different HIP related services. Some of the services envisioned are rendezvous service, discussed in Chapter 2.2.4, and registration with firewalls and other middleboxes. The registration protocol is, however, defined to be generic in order to be applicable to a variety of different services. With middleboxes, an initiating host would request a service of for example firewall traversal or Network Address Translation (NAT) before sending any of the actual traffic. This enables authenticating the initiating host before creating a state for its traffic.

The registration protocol reuses the DoS resistant HIP base exchange. Security associations are not, however, created in the case of registration. The R1 packet is used to announce the available services to the end-host. The end-host then requests service as part of I2 packet and server acknowledges this in the R2 packet. Registration exchange may also be done with update packets if host already has a HIP association with the service provider.

2.2.4 Rendezvous

HIP proposes also a mechanism for initial rendezvous, for mobile hosts that can not be reached through a static address. The rendezvous server functionality is described in the HIP rendezvous draft [18]. The mobile host uses the HIP registration protocol to create an association with an entity called Rendezvous Server (RVS). As a result the address of the RVS can be used

to initiate HIP base exchange with the mobile host as the RVS forwards the traffic to the current location of the mobile host.

The functionality of the rendezvous server is similar to that of a Mobile IP home agent. The essential difference is that the rendezvous server operates by forwarding only the initial I1 packet to the responder. The remainder of HIP communication will then take place directly between the initiator and responder hosts. The rendezvous server operation is presented in Figure 2.2

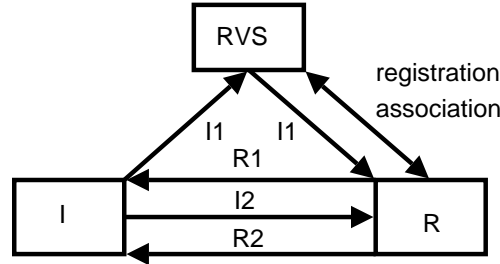


Figure 2.2: HIP base exchange through Rendezvous Server (RVS)

2.3 Summary

This chapter presented two different technologies that both have significant effects on security. Firewalls are already widely established network technology and provide uniform security for networks ranging from large organizations to single hosts. HIP, by contrast, is an emerging technology that guarantees security for end-to-end communication between two hosts.

In general, these two technologies are used by different stake-holders in a communication environment, and accordingly, for somewhat different purposes. HIP serves the end-user, providing confidential, authenticated data transfer. Meanwhile, the firewalls are mostly used by network administrators, concerned with details of the traffic and whether the traffic is authorized to enter the network. Oftentimes, some of the end-to-end security measures may also hinder firewalls as traffic is not transparent to intermediate network entities. Nevertheless, in case of wider scale deployment of HIP, these two technologies need to coexist. The following chapter analyzes this issue more closely.

Chapter 3

HIP Enabled Firewalling

The previous chapter introduced firewalls and HIP as well as the different security provided by them. In this chapter, it is further analyzed how also firewalls can benefit from the security provided by HIP. The chapter first discusses the general effects that HIP as a protocol has on firewalls. It also describes a basis for designing and implementing a HIP enabled firewall. In practice, there are limitations on what features can be implemented by the firewall solution of this thesis. Some of the features may not be implemented even by the existing HIP protocol implementations. Therefore, this analysis covers a wider scope of issues than the implementation.

Three different types of middleboxes, and accordingly firewalls, are identified in relation to HIP [32]. The HIP unaware firewalls are the current firewalls that do not provide support for HIP. Transparent firewalls are firewalls that are HIP aware, but operate implicitly from the end-host point of view. The third category is registration requiring firewalls that explicitly communicate with end-hosts. The actual implementation of the thesis is a transparent HIP aware firewall. However, both HIP aware firewall types are discussed here.

3.1 HIP with firewalls

One of the design objectives for HIP is that the protocol needs to cooperate with different middleboxes [21, 22]. HIP does provide strong security for data authentication, confidentiality and integrity. Yet, despite the traffic confidentiality, all essential information for HIP association flow identifier is visible to intermediate network entities. Furthermore, HIP is not only comprehensible to middleboxes, but enables them also to take advantage of

the security features it provides [21].

This chapter describes the basic functionality of a HIP enabled firewall. Although HIP has several benefits for firewalls, some difficulties also exist. Therefore, the chapter suggests some possible changes for better cooperating with middleboxes.

Lot of this chapter also applies to other middleboxes that need to keep up state of the HIP association and that benefit from authenticating the state changes. An example of such middlebox would be HIP enabled Network Address Translator (NAT). NAT and firewall functionalities are in fact often paralleled in the referenced HIP literature.

3.1.1 Basic Functionality

The basic functionality of a HIP enabled firewall is to perform traffic filtering based on HIs. HIs are in practice presented with HITs. In the context of firewall, HIT is a convenient identifier to be used for example in expressing access control information. HIT is also a natural selection as the flow identifier for HIP traffic. Filtering also concerns the ESP data traffic relating to the HIP control traffic. This implies that the firewall must be stateful in order to properly filter all HIP traffic.

The actual data traffic is carried over IPsec ESP. The IP addresses and the SPIs of the destination end-points can be used to identify the flow with ESP traffic [34]. The data traffic flow identifier can be first deduced from the base exchange packets. The initiator sends out its SPI in the I2 packet and the responder delivers its SPI in the R2 packet. During the lifetime of the HIP association, the update packets have to be monitored to keep track of the changes in end-point SPIs and the IP addresses they use.

Opportunistic Mode with Firewalls

One consideration with the HIP base exchange is that the opportunistic mode may be problematic with firewalls. For the most part the issue relates to security policies and can be tended to by choosing firewall rules with consideration for this. The firewall security policy may define which hosts are allowed destinations for traffic. However, the destination host identity of a HIP I1 packet using opportunistic mode can not be determined. In this case, the firewall should discard the packet if it can not verify the destination HIT of the packet against that in a firewall rule.

This kind of situations may be avoided, if destination HITs are not defined for hosts that are likely to be contacted with opportunistic mode. These hosts might include for example general purpose servers that are contacted by clients that are previously unaware of the server.

The opportunistic mode has effects also on stateful filtering as the destination HIT is used in the flow identifier. When the destination HIT of the I1 packet is not available a HIP enabled firewall could temporarily use the destination IP address in the flow identifier. After intercepting the corresponding R1 packet the firewall would fill in the missing HIT value. Again, this could be a firewall policy issue whether a firewall may establish state without the responder HI.

In the case of a registration requiring firewall, the NOTIFY parameter could be used for signaling about failed I1 packets. The NOTIFY parameter defines a message type for informing a peer host when opportunistic mode fails due to policy of the host. This could also be used by an intermediate firewall as part of the registration packets.

Authentication of HIP Traffic

As first pointed out in [22] and further discussed in [35, 34], the HIP signatures are visible to intermediate network entities. This allows HIP enabled firewall to authenticate the senders of control messages by validating the signatures. This property makes HIP traffic essentially different from other protocols from the point of view of a firewall.

Naturally, the traffic authentication also implies that the state information maintained by the firewall itself is in fact valid. This is important as HIP control packets are used to create state information that enables traversal of the HIP data packets.

For authenticating the packets the firewall needs to be aware of the end-point HIs corresponding to the HITs. As HITs are used in the access control list of the firewall, the HI relating to each HIT can also be defined. Furthermore, the responder HI is carried unencrypted in the HIP base exchange R1 packet. A firewall is therefore able to intercept the HI from traffic and to verify the responder signatures.

By contrast, the initiator HI is delivered encrypted in the I2 packet of the base exchange. This is done to protect the privacy of the initiator by not revealing the identity to outsiders. The validity of this reasoning is, however, questioned in [5]. It is further suggested that the initiator HI should be

transmitted unencrypted for the benefit of intermediate middleboxes.

Proper authentication of traffic has also implications to basic firewall functionalities. Traditionally, stateful firewalls have traced the details of different protocols to filter out packets that can not be a part of a valid, established connection. Despite of this, it is still possible that a packet that looks perfectly valid is spoofed. This could happen if an attacker is located on the connection path and is equally aware of the connection state. Furthermore, the firewall must then store much of the protocol logic and is more prone to errors. The communication between end-hosts is therefore increasingly dependent on the middlebox functionality.

In effect, HIP provides a more straightforward method that reliably authenticates the sender. This also simplifies the role of firewall as a middlebox, as most details of the protocol logic can be kept only at the end-hosts. However, a HIP enabled firewall needs certain level of protocol state keeping. This is necessary for being able to obtain necessary information for recognizing the ESP data packets of the connection.

3.1.2 Registration Protocol

The HIP registration protocol was already briefly introduced in Chapter 2.2.3. One group of the systems that could benefit of the registration protocol are middleboxes, such as firewalls. In fact, the registration protocol was first discussed in the context of middleboxes [33]. The middlebox traversal using registration protocol is further specified in the NAT and Firewall Traversal for HIP -draft [34]. Also, a prototype of the registration protocol has been implemented and is presented in [32]. The use of registration protocol essentially separates HIP aware middleboxes as implicit or explicit middleboxes as categorized in [9].

As described earlier, the registration protocol reuses the HIP base exchange procedure. This may be initiated by the end-host either explicitly sending an I1 message to the firewall. Alternatively a firewall may intercept an I1 message intended for the responder. In either case, the firewall then responds with R1 message as with regular base exchange. During the registration protocol exchange, the firewall inspects whether the traffic is allowed to traverse according to the security policy. If so, the firewall finally acknowledges the end-host's request.

A HIP enabled middlebox should not introduce new Denial of Service vulnerabilities, as pointed out in [33, 32]. Accordingly, the middlebox should be able to authenticate the end-point before creating state. The base exchange

procedure provides same benefits to the registration as to overall HIP. Both the firewall and the end-host can authenticate each other. Cost is imposed on the initiating end-host and the firewall does not need to create state before verifying the puzzle solution and authenticating the end-host. Extending these feature to firewall traversal, prevents firewalls from becoming the weak point of the HIP protocol.

Besides establishing the registration, also terminating it has significance. The registration uses a soft-state approach, where the registration times-out and must be periodically renewed. This increases the freshness of the state information in middleboxes. An example of the advantages here is presented with HIP mobility in Chapter 3.1.4. The registration may also be cancelled by either party. This could be useful for an end-host that wishes to stop receiving unwanted traffic in an expensive wireless environment [33].

In practice, traffic may need to traverse several firewalls, which would cause several associations to be considered in an individual firewall. An end-host would be required to register with several firewalls, for instance, the firewalls protecting the networks of the end-host and the peer. Therefore, the firewall functionality must recognize also the other registration associations, as well as, the actual HIP association. This scenario is presented in Figure 3.1.

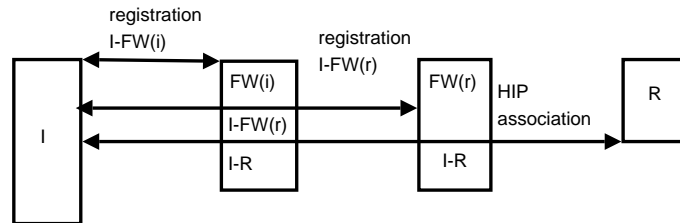


Figure 3.1: Associations between different network entities and the state maintained by the firewalls for each association.

Initializing the registration directly between the initiator end-host and a firewall may cause same problem as the opportunistic mode. Also in this case, the HIT of the actual responder host is not conveyed to the firewall. Due to this, the case where a firewall intercepts the I1 packet intended for the responder host should be favored. This allows firewall to make the access control decision based on both the source and destination HITs. Furthermore, in this case the initiator does not need to have explicit knowledge of the firewall before initiating the connection. Alternatively, the desired destination HIT would need to be conveyed as part of the registration. However, the registration protocol as such can not accommodate the destination HIT

value as services are negotiated with eight bit registration types.

3.1.3 HIP Rendezvous Service

Initializing HIP association may include use of a rendezvous server, RVS, as described in 2.2.4. A logical consequence of the rendezvous is that the IP addresses of the I1 and R1 packets may not be consistent. The destination addresses for the ESP flow identifiers are therefore best taken from the I2 and R2 packets carrying the SPIs.

Topologically there are different scenarios from the point of view of a firewall. These depend on whether the rendezvous server or either of the end-hosts, or a subset of these three is located in the network protected by the firewall.

In a case where either the rendezvous server or both the initiator and the responder are protected by a firewall, the firewall only detects the I1 being sent to rendezvous server and forwarded back to the responder. In this case the firewall needs to remove the state information created for the connection.

A registration requiring firewall has an additional consideration in one particular scenario where either the responder or both the rendezvous server and the initiator are located in the network protected by the firewall. When the rendezvous server forwards the packet it may need to also rewrite the source address of the packet. Otherwise, the packet may be rejected by ingress filtering as having a forged source address. In this case, the rendezvous server inserts into the packet a FROM parameter containing the original initiator address. A registration requiring firewall must then use the address of the FROM parameter, if one is present, in sending the R1 registration packet to the initiator.

3.1.4 Mobility and Multihoming

As further described in the HIP mobility and multihoming draft [24], HIP enables host mobility and multihoming by allowing hosts to signal each other about the changes in their network addresses. Moreover, HIP allows each host to have a set of addresses through which it communicates. These addresses are grouped under one or more Security Associations (SA) that HIP connection establishes. Consequently, a HIP enabled firewall needs to associate a dynamic set of SPIs, representing the SAs, to a single HIP connection. Each SPI may then include a variable set of IP addresses. The mobility and multihoming draft does, however, define a preferred address that each host

should announce. The preferred address should be the primary destination address that peer hosts send data to.

One clarifying aspect is that each address is suggested to be associated to its own SA. Alternatively a set of addresses that are expected to experience faith sharing could be grouped under an SA. An example of this would be the addresses of a same network interface that would be included under a single SA. In some cases, communicating hosts may have different number of interfaces they use for the connection. The main policy is, however, that the SAs should even then be formed pairwise between the hosts.

The actual signaling procedure uses HIP update packets as depicted in Figure 3.2. Here a host moves to a new network and receives a new address. The new address may be added to an existing interface or the new network may provide network access point to a new interface. The information about the SPI is included in a REA parameter, along with the address. In the latter case the packet also include NES parameter as rekeying is performed.

The response packet from the peer includes acknowledgement for the previous message, SPI or NES parameter and echo request parameter for address verification. Therefore, this message is sent to the newly announce address. This way also a firewall protecting the new location of the mobile host is able to intercept it and obtain the peer SPI value. The draft mentions that address verification could be skipped in some cases. It further warns that this may lead to incompatibilities with middleboxes. A HIP enabled firewall could also enforce the use of address verification. In that case, no data traffic is allowed to a new address until the firewall has encountered the related address verification packets.

For assuring the traversal of firewalls protecting the new location, the first update message should preferably be sent from the new address. This is not explicitly required in the draft. As a result, the firewall may not be able to intercept the first update packet and can not acquire the SPI of the mobile host. After that, also the following messages of the update exchange would be blocked by the firewall as the connection is previously unseen. A more complicated issue is announcing several new addresses in a same update message. In this case, if the preferred address is among the addresses, it could be preferred as a source address.

Another point of view is that of a firewall that protects the previous location of the mobile host. Alternatively, the host can be multihomed and has acquired new address for another network interface. If the newly acquired address is used as the preferred address, traffic may not traverse the old firewall any longer. In both cases the firewall is not able to intercept the

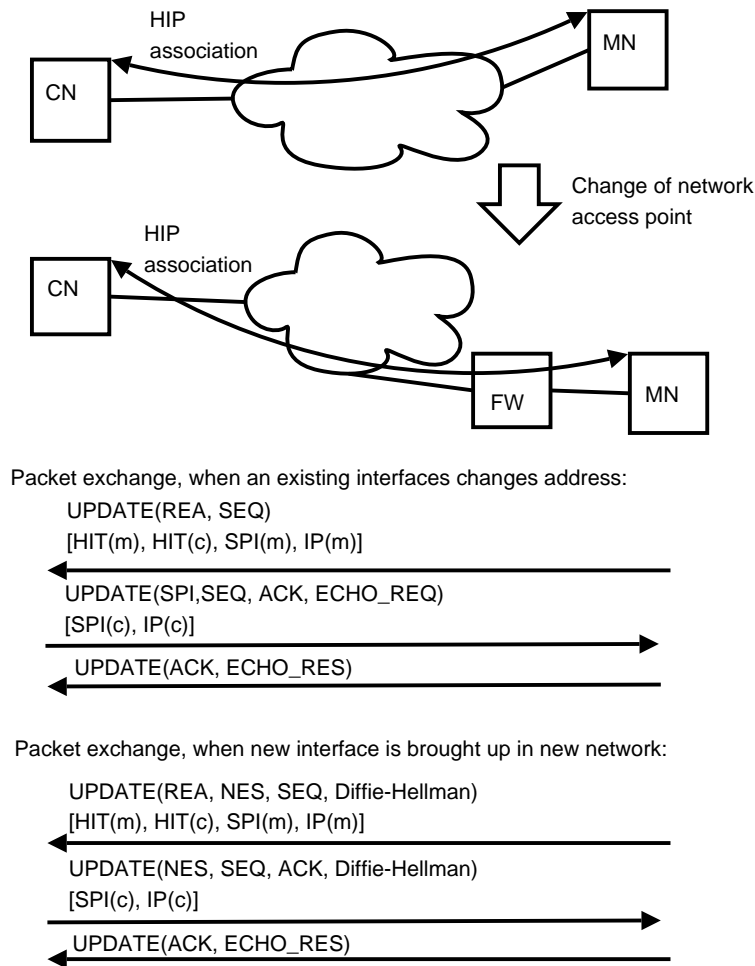


Figure 3.2: Packet exchange between mobile node (MN) and corresponding node (CN). The figure presents HIP packet exchanges in two cases where either existing network interface receives an address in the new network or new network interface is brought up. Information extracted by the firewall (FW) from each packet, is listed inside brackets.

remainder of the connection, including the close sequence. For practical reasons, a transparent firewall might need to remove the connection state for instance after a certain time of idle connection. This protects the firewall from using memory resources for potentially non-existent connections. With a registration requiring firewall, the state is automatically removed as the registration times out.

3.1.5 Asymmetric Routing

Asymmetric routing may cause problems with firewall state creation. In certain cases, traffic between two end-hosts may travel along different paths in different directions. This causes problems for networks with several firewalls if incoming traffic traverses different firewall than outgoing traffic. In this case neither of the firewalls is able to intercept the SPI values needed for the connection state. This scenario is presented in Figure 3.3. The problem of asymmetric routing with HIP has been introduced in [34] and is further discussed in [32].

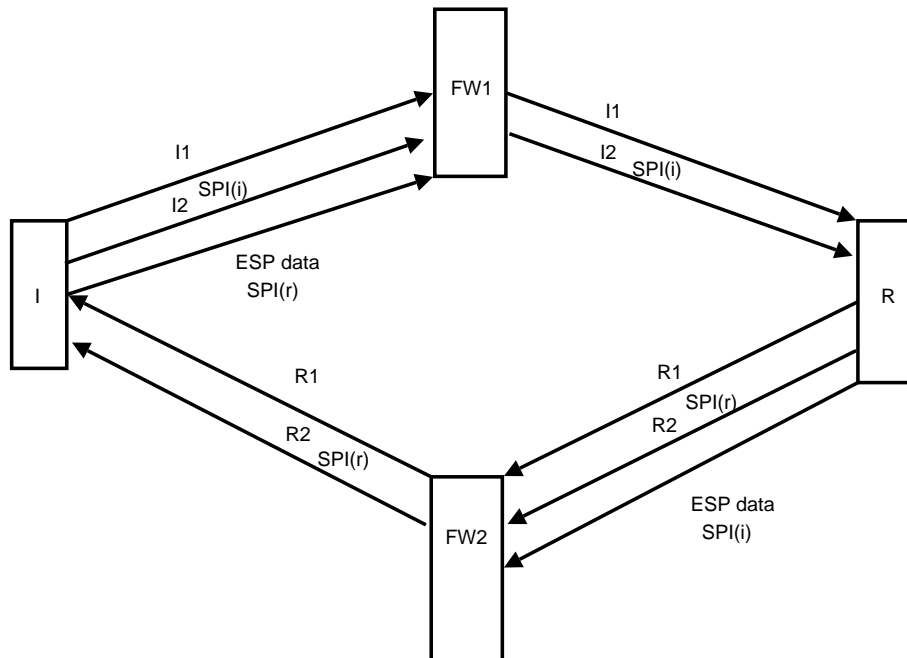


Figure 3.3: Asymmetric routing scenario.

One solution proposed is including the SPI value of the peer host to another message that is sent back to the peer [32]. This way, the initiator SPI would be included in R2 packet. For the responder SPI a special I3 packet would have to be sent after the actual base exchange. Extending the base exchange with one more packet is not, however, considered a desirable solution.

As another solution with registration requiring firewalls, a message for signaling SPI values is proposed [32]. An end-host would use this SPISIG message to send the SPI value it has chosen for the security association to the firewall. This would, however, require that the end-host has explicit knowledge

of the firewall. Even when a registration association is created, it is not done with the firewall that will miss the SPI of the host. Furthermore, the asymmetric routing may take place in both initiator's and responder's home network. In effect, the end-hosts would have signal the SPI value to any firewall experiencing asymmetric routing along the path from the peer to the host.

Possibly a more straight forward solution would be combining the approaches of the first and the second solution. As a result a host would use the SPISIG message to signal the value chosen by the peer host. This way a host would have knowledge of at least any registration requiring firewalls on the path. This is presented in Figure 3.4.

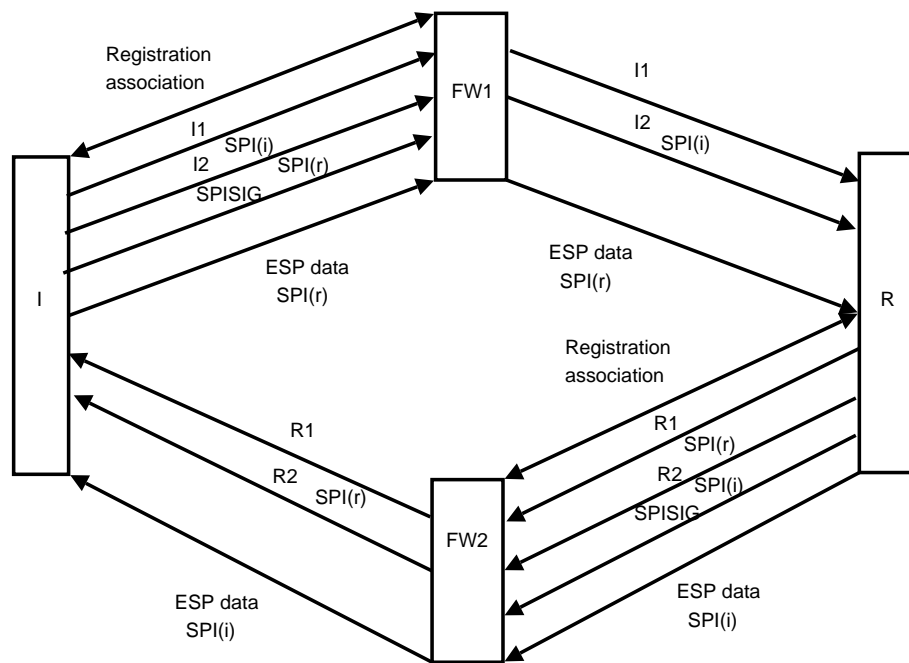


Figure 3.4: Asymmetric routing scenario with SPISIG signaling. Host signals the SPI value of the peer host.

To conclude, the asymmetric routing is somewhat problematic issue for stateful firewalls. However, this problem is in no way HIP specific, but exists with several other protocols as well.

3.1.6 HIP Certificate Parameter

HIP protocol defines a parameter type for delivering certificates [22]. This provides a generic method for certificate use, which can be further extended for different purposes and can benefit a variety of different situations. The actual usage of the certificates is left to be defined.

Using certificates with the firewall registration is first mentioned in [33] and is further discussed in [32]. For authorization, Simple Public Key Infrastructure (SPKI) [12] certificates are suggested. The use of certificate parameters is not explicitly mentioned in the registration protocol draft [19]. However, the certificate parameter could be used as part of the registration protocol for distributing information for authentication and authorization [32]. Certificates could be similarly intercepted also by transparent HIP aware firewalls when they are not encrypted by the sender.

For firewalls, certificates could be an attractive mechanism for authorization [12]. However, certificate revocation mechanisms could complicate the system. The authorization information stored in the access control list will need to include public key material for authentication. With use of certificates this authorization material can be limited to the public keys used to issue authorization certificates to hosts.

As traffic is initialized the firewall can obtain the HI of the host from the certificate along with the authorization information. The authorization information can then be used to allow certain type of firewall traversal. After the connection is closed the firewall may clear the public key of the host from its memory. This can considerably decrease the amount of data that HIP enabled firewalls must store for the security policy.

In this case also management of a distributed firewall system would be more efficient and straightforward. Authorization issued with a certificate takes an effect instantly anywhere the certificate is used. Updating firewall rule sets of several firewalls on the other hand require operations with several different entities. Furthermore, certificate expiration times allow issuing rights for a fixed period of time. In a traditional firewall this would require updating the firewall rule set twice for also removing the authorization.

3.1.7 Potential Security Vulnerabilities

Traditionally stateful firewalls need to establish state after the very first packet in order to associate the later packets to the same connection. This consumes firewall resources and may cause a risk of similar DoS vulnerabili-

ties that HIP helps avoid in the end-hosts.

In the case of HIP enabled firewall, resources can be exhausted mainly in two ways. Firstly, the signature verification consumes CPU cycles in the firewall. Secondly establishing state related to HIP associations requires memory.

With the initial registration a firewall may avoid the problem of early state creation and the puzzle mechanism imposes a cost on the initiating end-host. The difficulty of the puzzle may also be adjusted for different situations such as for potential DoS attack. A transparent firewall may attempt to alleviate problem with a suitable timeout value after which connection is removed if valid I2 packet is not received. This is still problematic and no directive timeout value can be deduced from HIP protocol, since state is not established after I1.

The signature verifications can not be avoided when a firewall needs to authenticate traffic. Performance analysis of cryptographic operations of a HIP implementation indicates that especially the Digital Signature Standard (DSA) signature creation and verification are relatively strenuous compared to other operations [14]. The exact results may be implementation dependent, but the general observations should be valid. HIP implementations are also required to support the Rivest Shamir Adleman (RSA) as the public key algorithm. The use of RSA benefits an intermediate firewall as the signature verification is significantly faster than with DSA [29].

With well behaved end-hosts signature verifications are in many cases relatively infrequent. In general, they are only needed at connection establishment and when end-points change their set of network attachment points and as the connection is closed. Nevertheless, a malicious host may cause additional signature verification attempts by sending spoofed HIP packets to the firewall. This, of course, requires that the attacker is aware of some pair of HITs that have an ongoing connection through the firewall. With end-hosts the additional keyed-hash message authentication code (HMAC) in the HIP packets can be verified with less effort. Unfortunately the HMAC verification is not available for intermediate entities as it is based on the shared secret between the end-hosts.

The traffic authentication only applies to control traffic, while data traffic is simply recognized through the flow identifier. In effect, if a third party has knowledge of the SPI values and IP addresses of the hosts it may create false ESP traffic that penetrates the firewall. This causes additional traffic in the network being protected, but the end-hosts will be able to discard spoofed ESP packets.

3.2 Scenarios for HIP Enabled Firewall

Following discusses general scenarios for a HIP enabled firewall. It provides examples of scenarios where HIP seems to conveniently aid traditional firewall functionalities.

A general consequence of HIP in firewall systems is that details of the upper layer protocols are hidden from firewalls. Oftentimes organizations monitor and filter traffic on different levels of the protocol stack and with different protocols. On one hand HIP simplifies things and provides much needed security. On the other, the firewalls will not have say on upper level issues or the traffic contents. This may essentially change the role of firewall systems in corporate and organization networks.

3.2.1 General Firewalling Scenario

One typical policy for stateful firewalls is to allow connections to be established only from the trusted network. From the untrusted network the only traffic allowed to traverse is then related to the established connections. In effect, no unsolicited traffic from untrusted side is accepted.

As a result, the initiators are in this case hosts of the trusted network. For a transparent firewall that does not require registration the initiators are mainly the hazardous party of the connection. They pose a potential risk of abusing the firewall state establishment. In the scenario where the initiators can be assumed more reliable, also the risk of the DoS attacks is alleviated.

Furthermore, depending on the strictness of the security policy it may not be necessary to authenticate the traffic from the trusted network. The risk of a host impersonating another to penetrate a firewall is not as great for hosts of the trusted network and little is gained by doing so. In this case, the firewall rule set does not need to include the HIs of these hosts and memory can be saved.

On the contrary to the initiator, the responder of the connection is located in the untrusted network along with several potentially malicious hosts. Accordingly, it is more of a concern to authenticate the traffic of the responder. The responder is also chosen from a vast group of potential hosts. Therefore, it is favorable that the responder identity is available in the base exchange packets and can be dynamically added to the state information. As a result the firewall does not need to store the HIs statically. Instead, the information most important for ensuring security is provided by HIP protocol itself.

3.2.2 Road Warrior and Virtual Private Network (VPN) Solution

Virtual Private Network (VPN) is a technology for securely interconnecting networks that may be dispersed across Internet [15]. It uses secure tunneling mechanisms, such as IPsec, to transport traffic between the access points of different networks. In effect, VPN creates a system which can be virtually considered a single private network from security perspective. In addition, individual end-hosts may connect to a network using the VPN solution. A term road warrior is used for a user connecting to a corporate network from varying external locations. In VPN the designated access points of the network, VPN gateways, perform the encryption and authentication required for achieving confidentiality and integrity of the traffic.

HIP can also be applied to a Road Warrior VPN scenario [23]. In brief, the VPN-like solution would consist of end-hosts communicating securely over HIP and designated HIP aware firewalls for enforcing access control and authentication. HIP hosts would therefore take care of the encryption of traffic. The centralized entity would only perform the cryptographic operations necessary for authenticating traffic and control access to the network.

In case of HIP some functionality of the VPN gateway is shifted to the end-hosts. This decreases the load on the centralized entity without compromising security. In case where end-host would explicitly require data encryption for its traffic it would have to perform the encryption itself anyway. In traditional VPN there would then be duplicate encryption. This is an example of the classic end-to-end principle: a function should be implemented by the end-host, whereas implementations in intermediate components may only provide improvements [28]. Another apparent benefit of HIP is that it provides solutions for mobility and multihoming in the process.

In this scenario the firewall would need to require registration from the remote end-hosts wishing to connect to a corporate network. As the hosts initiating the connection are in the untrusted network, there is a high risk that the IP packets can be spoofed. Creating state at a firewall before authenticating the initiator could easily cause possibility of a DoS attack.

3.3 Summary

This chapter analyzed different aspects of HIP that may have effect on firewall functionalities. As a conclusion, HIP provides several benefits even though

there are also problematic aspects. Reliable traffic authentication and the transparency to intermediate network entities, are likely to enable developing more reliable firewall functionalities. On the other hand, a HIP enabled firewall must also consider different HIP functionalities in order to handle the protocol traffic correctly.

The chapter also analyzed potential security vulnerabilities. The registration protocol, defined for HIP, would be useful in providing additional security for the firewall as the end-host can be authenticated before using firewall resources on creating state. Asymmetric routing also continues to be problematic, even though possible solutions have been proposed. However, the asymmetric routing is in no way HIP specific problem.

Chapter 4

Requirements

This chapter presents the requirements placed on a HIP enabled firewall solution. It includes both the functional requirements for the firewall implementation as well as the non-functional requirements. One source of constraints are the general requirements defined for firewall technologies. For middle-boxes, such as firewalls, HIP end-point identifiers and the visible signaling in HIP provide new opportunities in filtering the intercepted traffic. Therefore, another set of requirements as well as possibilities emerges from HIP.

The implementation is a generic firewall system with a simple management interface. Therefore the implementation does not include defining rules for individual security policies. Accordingly also the level of security provided by the firewall depends on the security policy used. However, the firewall design and implementation must not have additional flaws or weaknesses that would weaken the security.

As the HIP standardization and protocol implementation is on going, the scope of the implementation is in general limited to features covered by the HIPL HIP protocol implementation [1]. This enables testing and verifying the firewall features. One main restriction is that the HIPL currently only supports HIP IPv6 traffic. Therefore also the firewall implementation will be IPv6 specific. Linux is chosen as the platform, which presents practical constraints. Nevertheless, the high level design of the solution must still be generic enough to be independent of platform.

4.1 Functional Requirements

The following presents the functionalities that a HIP enabled firewall must provide. To depict the services provided by the firewall, the chapter describes also example usage and firewall rules.

4.1.1 Firewall Policy Management

A firewall must provide a mechanism for defining the security policy by which it operates. For this particular implementation, an elaborate user interface is not necessary. Instead, firewall policy configuration may be done with a simple configuration file that the firewall processes at start up. For further development of the firewall management, the firewall should provide interface for more interactive maintenance. This interface needs to provide functions for updating the security policy of the running firewall, and accordingly need to take concurrent operations into account.

The format of the security policy should be simple, axiomatic and well-defined. The firewall policy is defined with a set of rules that define what kind of analysis is performed and which packets are allowed to traverse. Ideally the rule format would be also easy to learn and adopt by a user. A desirable solution could be therefore provided by using a rule format of an existing firewall solution as basis. The firewall implementation is done in Linux environment. Furthermore, the Linux Netfilter/Iptables framework [3] is a well-established and widely used firewall system. Therefore the syntax and semantics of rules should preferably follow that of the Iptables rules [4].

The general format of a firewall rule is presented in Figure 4.1. Here the HOOK defines through which of the hooks in networking stack (INPUT, OUTPUT or FORWARD) the packets are received. The match defines the packets that the rule concerns and TARGET defines whether the packets are accepted or dropped. The match may be constructed from several different options which all must match the packet properties for target to be executed. Examples for rule options will be further presented in following chapters for illustrating the firewall requirements.

HOOK [match] TARGET

Figure 4.1: Format of the firewall rule.

The implementation needs to provide rule parsing for checking syntax of the rules. Although the firewall should allow flexible expressive use of filtering options, some limitations are needed in order to keep semantics of the rules rational. This should, for instance, eliminate contradicting options, such as defining incoming interface for a rule in OUTPUT hook. The firewall policy management must provide this semantic inspection for rules.

4.1.2 Overall Functionality

For effectiveness of the firewall system it is essential that the firewall is able to intercept all traffic that concerns it. This requires that the firewall software has an interface to the networking stack of the communication system. The HIP enabled firewall needs to be able to intercept all IP packets that carry either HIP or ESP as payload. Furthermore, this should be done in different branches of packet traversal. The firewall must have access to incoming and outgoing packets as well as packets being forwarded. In addition, the firewall must also be able to enforce that a given packet will be accepted or dropped.

4.1.3 Stateless Packet Filtering

Stateless packet filtering functionalities provide simple inspection of certain properties of HIP packets. These properties include source and destination HITs, the type of a HIP packet as well as incoming and outgoing network interfaces. The properties may also be negated. Options for these functionalities are presented in Figure 4.2.

```
-src_hit [!] <hit value> -src_hi <file name>

-dst_hit [!] <hit>

-type [!] <hip packet type>

-i [!] <incoming interface>

-o [!] <outgoing interface>
```

Figure 4.2: Format of stateless filtering options.

Access Control Based on Host Identities

Stateless packet filtering must also contain option for enforcing access control based on the cryptographic identity of the host. Along with source HIT, a rule may also define the corresponding HI, which will then be used to verify the sender signatures. To ensure validity of the rule, firewall should also inspect that the HI matches the host identity, before accepting the rule.

4.1.4 Stateful Packet Filtering

Stateful packet filtering provides filtering based on the connection state. Stateful packet filtering can be used with the state option and by defining the state (NEW or ESTABLISHED). This command may also be combined with other filtering options to match for example new connections that have certain source or destination HIT. Format of the state option is presented in Figure 4.3.

```
-state [!] <state> -verify_responder -accept_mobile
```

Figure 4.3: Format of the state option.

The vital issue with stateful filtering is determining which packets are actually part of the connection. The firewall should be able to ensure that the state acquired from the protocol packets is valid and not a result of spoofed packets. With cryptographic identities and signed protocol packets, HIP provides effective methods for ensuring this. The traditional measure for analyzing packets is maintaining protocol state information and comparing received packets with that. Also HIP enabled firewall needs to provide suitable level of protocol state checking.

HIP provides possibility to dynamically obtain the responder HI from base exchange packets as the connection is initialized. Verifying signatures of the responder assures sender invariance [34] in cases where the responder identity is not previously known in the firewall policy. The firewall should provide this as an optional feature of stateful connection tracking. Crucial part of obtaining the responder HI is examining that the public key in fact produces the responder HIT as hash value.

HIP protocol contains two different data streams; the protocol data and the payload data carried in ESP. Firewall needs to identify the protocol

traffic using the HITs as flow identifier and data traffic using the SPIs and destination addresses [34]. The SPI values are intercepted as base exchange or rekeying packets are analyzed. When connection is closed by the endpoints with close packets, the connection state must be removed.

When a mobile host moves to a network protected by a firewall, the signaling data will in some cases traverse the firewall. The firewall should establish state from mobility signaling when allowed by the security policy, and when it is made possible by the mobility signaling

The firewall must also allow rendezvous traffic to traverse when it is authorized according to the security policy. The rendezvous functionality is not, however, currently completely supported with the HIP reference implementation. However, the scenarios of rendezvous traffic should be considered in the design.

4.2 Non-Functional Requirements

Some non-functional requirements were already identified when presenting the functional requirements. For example, properties of the rule format were discussed. This section further defines which non-functional requirements are essential for the design and implementation. In addition, it is also analyzed why some general requirements may not be as important for this particular system.

4.2.1 Requirements Regarding the Design

General requirements for any architecture include that it is modular, clear and as simple as possible. The design also needs to model the problem domain and interactions within it. In many cases the viability of architecture is truly weighed only after the design needs to be changed or maintained.

As HIP is an emerging technology and the standardization is ongoing, changes to the protocol are expectable and even likely. Even during writing of this thesis different aspects of the protocol have been modified as new versions of the drafts have been published. This affects, besides the implementations, also other systems concerned with the details of the protocol, such as HIP aware firewalls. Accordingly this stresses the designs of these systems as changes are adopted within the existing architecture.

The design of a HIP enable firewall should take this into account. As a result

the design should be modular by nature. This way the structure of the design is clear as unnecessary interconnections between different elements will not complicate the design. A modular design is often also easy to extend to support new features and functionalities. Another constraint could be that the parts of the design are generic and could therefore be reused for new emerging aspects.

Potential changes concerning HIP include for example new types of packets and parameters in the core protocol. Alternatively, existing parameters or packets may also be removed or modified. Also, alternative methods for performing functions, such as new algorithms, could be adopted. The registration extension also enables introducing new types of services relating to HIP.

For an end-host it may be adequate to only implement a subset of functionalities and operate using those. A middlebox analyzing traffic of other hosts may, on the other hand, come to contact with different types of traffic. In general, a middlebox should not block legitimate traffic that is in accordance with the protocol definition.

4.2.2 Security

The very purpose of a firewall is to provide security for an external user. In general, the field of the thesis for a large part concentrates on issues relating to security. Chapter 3 discussed what kind of security measures are possible and on the other hand necessary in the context of HIP. The functional requirements further defined what kind of functionalities a firewall needs to provide in order to deliver these security measures. In addition to these the internal security of the firewall is also important. This includes for instance reliable management of the firewall rules.

4.2.3 Secondary Requirements

There exist a number of qualities that are in general desirable for a system of this kind. These include for example efficiency, usability and manageability. These are essential properties for a firewall system used in a production level environment. However, the scope of this thesis limits to a prototype implementation that demonstrates the feasibility of this technology. In this context, some of the requirements important to production level implementations are secondary concerns for this system. For future development of the technology it is however important to also identify these properties.

Efficiency of a firewall system is an essential goal. The risk is that a firewall system will create a bottleneck for network traffic to and from the secured network. To an extent this can be controlled with CPU power of the firewall. The design and implementation of the firewall system play a role here as these factors can both hinder and support the efficiency of a system. Furthermore, HIP firewall functionalities include some potentially strenuous operations, such as the verification of cryptographic signatures.

Usability was, to some extent, discussed with the format of the firewall rules. Another aspect of usability is the manageability of the firewall system. This includes the system used for altering the firewall security policy. One aspect is the availability of the management interface and includes also remote maintenance of the firewall. The necessary user interface for this system is rather minimal. However, further development of a larger scale management interface should be addressed by providing a necessary interface for updating the firewall rule set.

4.3 Summary

The functional requirements defined by this chapter included management of the firewall policy as well as the actual packet filtering functionalities. The firewall filtering options were also defined and used here to illustrate the different features provided by the firewall.

The firewall policy management includes inspecting the validity of the rules and provides necessary functionalities for managing the rule set. Different packet filtering functionalities include stateless filtering as well as stateful connection tracking. The stateless filtering options can be used for filtering packets based on source and destination HITs, HIP packet type and incoming and outgoing network interfaces. Also the source HI may be defined for verifying the HIP packet signatures using the given public key. The state option is used for filtering based on connection status. It also contains additional options for verifying signatures of responder packets and accepting connections of mobile hosts.

In addition, different non-functional requirements were identified. These include properties of the design and general security aspects.

Chapter 5

Design

Firewall technologies are in general quite well established but have not been actually standardized. This chapter uses some available reference architectures in analyzing different architectural considerations in the context of HIP enabled firewalling.

5.1 Design Alternatives

Two high level design alternatives are presented. First one extends the existing Linux Netfilter firewall solution [3] and second implements an independent HIP firewall prototype. Former of these was the initial choice for design and was therefore relatively extensively studied. The latter was, however, selected as it better suited the setting of the project.

Even in the independent solution the Netfilter framework has its influence. It is used for receiving packets from the networking stack and the general format of rules has been adopted for the independent implementation. In addition, the Netfilter architecture has affected some of the design choices of the independent solution. Hence, this chapter also describes the Netfilter architecture in more detail.

5.1.1 Linux Netfilter Extension

Extending the Linux Netfilter [3] for HIP firewalling has several benefits. Accordingly, Netfilter was initially considered as a good candidate for basis of the design. In general, it is freely available and designed to be extensible. Netfilter is also the designated and established place for firewall and NAT

functionalities in Linux. Therefore, it could be desirable to integrate also the HIP functionalities into the existing implementation instead of implementing an alternative firewall only for HIP based filtering. Using existing implementation also provides the generic components, which would otherwise have to be implemented separately. In the case of Netfilter, the Iptables user interface would be available for modifying rules of the running firewall.

Drawbacks of Netfilter include mainly practical reasons and suitability for the particular project. An important feature of HIP traffic filtering is the state keeping for connections. Furthermore, the HIPL [1] HIP implementation, which is used as a reference implementation for the firewall testing, currently only supports IPv6. The IPv6 stateful connection tracking in Netfilter is, however, only supported in Usagi kernel and changing over to it was not a desirable choice. Furthermore, the signature verification used for authenticating end-points, could be more conveniently done in user space.

Following presents the overall architecture of the Netfilter framework as well as architectural changes for extending Netfilter to support HIP traffic filtering. Also the Iptables -management interface and the changes concerning it are presented. The high level architecture of a possible solution is presented in Figure 5.1.

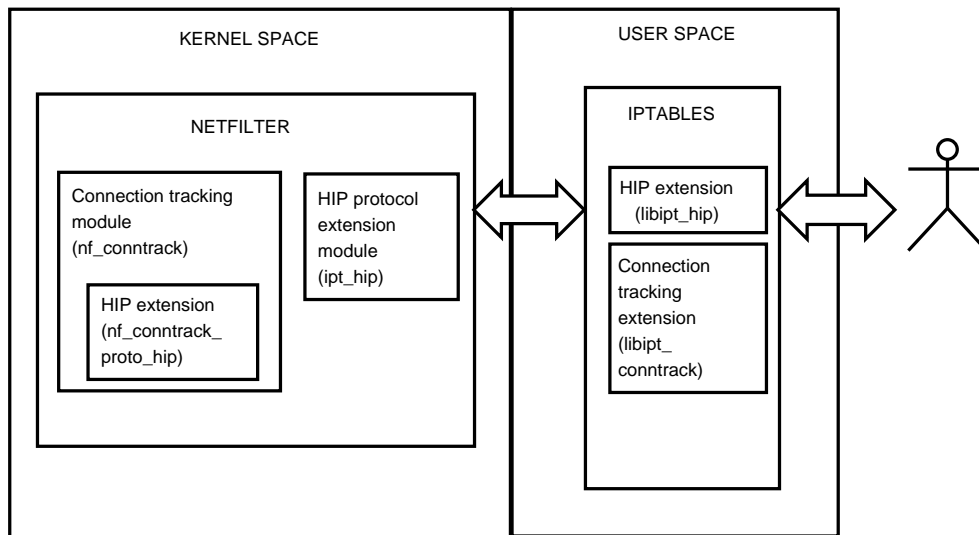


Figure 5.1: High level architecture.

Netfilter Design

Netfilter is a framework enabling firewall and NAT functionalities in Linux. It is implemented as a set of kernel modules, containing the core functionalities and additional modules for extending Netfilter to support different protocols and to provide additional features. The core functionalities manage the necessary hooks in the protocol stack and manage calling functions of different extensions. The Netfilter framework design is therefore quite modular and extensible.

Netfilter interfaces to operating system protocol stack with a series of hooks for receiving packets. This is represented in more detail in Figure 5.2. Firewall management is conducted by specifying a series of rules for different Netfilter chains. Packet filtering, which is the main focus of this thesis, can be done in the input, output and forward chains referring to local in, local out and forward hooks respectively. [27]

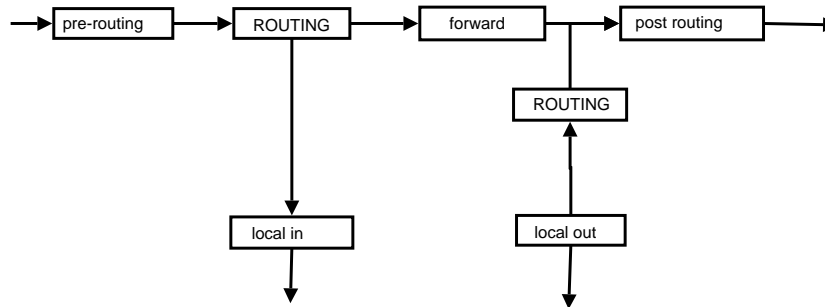


Figure 5.2: Netfilter hooks and packet traversal [27].

Netfilter provides an interface that extension modules implement. The functions include initializing and closing the module as well as a function for checking validity of the Iptables rules. For the actual packet filtering the module provides functions selecting packets and determining whether a packet matches a given rule. [27]

HIP Protocol Extension Module

The HIP protocol extension module, also presented in Figure 5.1, enables packet filtering of individual HIP packets. The filtering is done based on packet properties such as source and destination HIT and packet type. The extension module provides a standard Netfilter interface of functions, through which it is called.

Connection Tracking

Netfilter provides stateful filtering functionalities under a specified connection tracking module. Under the connection tracking there are extension modules for supporting tracking of different protocols.

The connection tracking module maintains necessary state information for identifying connections between two end-points. The information is contained in a data structure called `nf_conntrack_tuple` which includes IP addresses, protocol and the original direction of traffic, as well as additional protocol specific information. Tuples are contained in a hash table structure, where they can be efficiently searched and matched with packets. Each connection, `nf_conn` structure, holds reference to two tuples, one for each direction. Also a reference back to connection can be derived from a tuple hash structure. The relations of these data structures are presented in Figure 5.3.

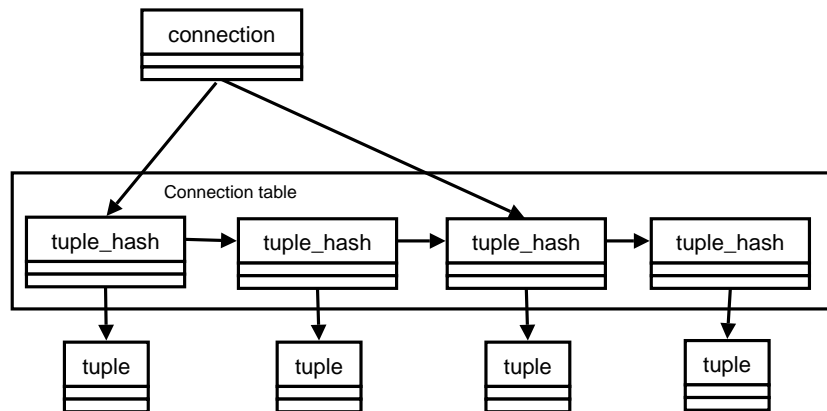


Figure 5.3: A simplified connection table structure of conntrack module.

Extending Connection Tracking Core

In the tuple, the main information for identifying the connection is based on the source and destination IP addresses. HIP however provides end host identities, that are independent of the location dependent IP addresses. To accommodate this within the Netfilter framework the HIT needs to be used as a flow identifier for the HIP protocol traffic. As the 128 bit HITs can be conveniently stored in IPv6 address structures, the HIT could replace the IPv6 address in the tuple. Also, a dynamic set of IP addresses needs to be

associated with the flow. In effect, even though the Netfilter framework is rather extensible, HIP implies changes to the more fundamental assumptions in the architecture.

As HIP traffic contains two separate connections, the HIP protocol traffic and the ESP data traffic, there needs to be a way to interconnect these two flows. For this, the connection tracking provides a concept of related connections. A protocol extension may implement and register a protocol helper, which is used to point out the expected related connections [27]. In the case of HIP, the protocol helper will create an expected ESP connection with certain IP addresses and SPIs whenever such connection is possible. This would occur when SPIs and IP addresses are exchanged during the base exchange, readdressing or rekeying.

HIP Connection Tracking Extension Module

The HIP extension module for connection tracking needs to provide standard interface of a connection tracking extension. That includes most importantly the functions for defining the HIP specific part of a tuple, creating a new connection and giving verdict whether a packet matches tuple.

The protocol helper functionalities are used to analyze the HIP signaling data. This includes, the responder HI, the SPIs and changed IP addresses. With this information the HIP connection properties and related ESP connections can be updated, created or deleted when necessary.

HIP protocol specific information in the tuple data structure needs to include the host identities, when available. The HITs would be already stored in the main part of the tuple holding the flow identifier.

ESP Connection Tracking Module

Extension for stateful tracking of ESP traffic is necessary as ESP is used for carrying HIP payload traffic. With ESP traffic the flow is identified with the SPI values and destination IP addresses. Consequently, the ESP protocol specific part of the tuple structure contains the SPIs.

Iptables Management Interface

Iptables is a user space management interface for the Netfilter functionalities. It provides means to view and alter the firewall and NAT rules for different

Netfilter hooks. Iptables contains extensions corresponding to Netfilter extension modules. Each extension contains functions for parsing and checking the validity of the inserted rules. [4]

Iptables user interface could be extended to support HIP traffic filtering by implementing an Iptables extension library. This extension module would contain options for HITs, HIs and HIP packet types.

For the stateful filtering the existing Iptables module handling connection tracking would have to be extended with HIP specific sub-options.

5.1.2 Independent HIP Firewall Solution

An alternative for extending an existing solution is to construct a separate HIP firewall system. The implications of this are twofold. Design choices of an independent firewall are not bound to an existing implementation. Consequently, the design and implementation may be chosen to better serve the particular protocol instead of a generic design. Also the system may be constructed in either user space or kernel space. Due to HIP signature verification, a user space implementation is more desirable for the system.

Then again, a separate implementation may not be able to take advantage of the generic parts of an existing solution. These include for example user interface, the general framework and the interface to the protocol stack for intercepting packet traversal. Fortunately, the Netfilter framework provides a mechanism for user space applications to participate in the packet filtering. The following section presents the resulting design in more detail.

5.2 HIP Enabled Firewall Design

Even though an independent HIP firewall implementation is chosen, the Netfilter framework does contain some well-founded design choices. Furthermore these design choices have been proved to work in practice. Therefore some aspects in the Netfilter architecture have been adopted also for the independent solution.

While Netfilter provides a practical reference for firewall architecture, a more conceptual model has also been studied [26]. The model presented is not adopted as such, but is rather used for identifying different functional entities and their interactions. These are further adapted to needs of this particular system.

The following chapters present the overall design of the solution. Essential functional components from [26] are also identified in the design. The design is also illustrated in Figure 5.4.

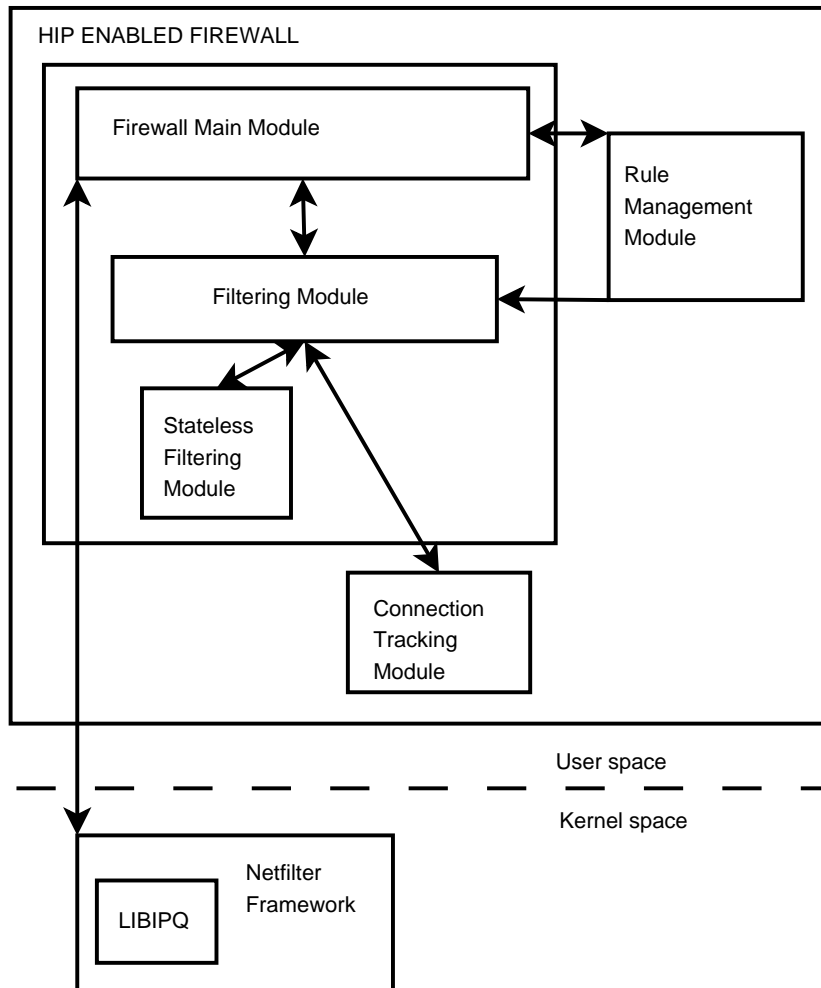


Figure 5.4: Overall design of the HIP enabled firewall. The interactions between components are shown with the arrows.

5.2.1 Firewall Main Module

The firewall main module contains functions for receiving packets for analysis and issuing verdicts on whether the packets are allowed to traverse or not. It uses rest of the firewall components for producing these verdicts based on properties of the packets received.

The main module contains mechanisms for interfacing the firewall to the actual communication system. This is referred as integration and enforcement module in [26]. Integration and enforcement functionalities must guarantee that the firewall is able to intercept the packet traversal in right parts of the system protocol stack.

The firewall design uses the Netfilter framework for integration to the actual system. Netfilter module in turn contains necessary hooks in the Linux networking stack through which packets are intercepted. The packets interesting to the HIP firewall are directed to Netfilter QUEUE target, which is used to transmit packets to user space applications. The firewall system uses the LIBIPQ library to register for receiving queued packets as well as issuing verdicts on them.

5.2.2 Firewall Policy Management

The firewall rules that define the security policy are contained by an entity called rule set [26]. Here the firewall policy management module contains also functionalities for managing the rules and verifying the rule syntax.

There are two basic entities interacting with the rule set. Firstly, the firewall system needs to regularly read the rule set to determine faith of each packet. Secondly, the firewall manager needs to set the rules for defining the firewall security policy. For this a configuration file is used and is read in the firewall system at start up.

Due to the minimal user interface, the policy management module also provides an interface for further development of the management system. The interface consists of functions for altering the rule set of a running firewall. These functions can be used to implement a more interactive user interface for firewall management. The rule management also needs to provide necessary synchronization for potentially concurrent operations.

5.2.3 Packet Filtering Functionalities

For the actual packet filtering mechanisms two different modules are identified [26]. The analysis module is used for conducting analysis on the packet data. The decision module uses the firewall rules and result of the analysis to determine whether packets are allowed to pass.

In this design these functionalities are contained as a single logical entity under the main module of the firewall. The decision module calls different

analysis functionalities based on the contents of each rule. The results are then gathered to issue a final verdict on each packet. The result is finally returned to the main module.

Different analysis functionalities include filtering based on identity (HIT and HI), HIP packet type, incoming and outgoing interfaces and the Netfilter hook, through which the packet was received. The stateful filtering module is described in more detail below. The identity based filtering compares the HIT defined in a rule with that found in the packet. Additionally rule may define also the sender HI. In this case the HIP packet signature is verified with the HI to authenticate the packet sender. In effect, the filtering module also contains functions of an authentication module [26].

5.2.4 Connection Tracking

The connection tracking module is conceptually part of the analysis functionalities. It is, however, separated to its own module, as it contains more elaborate and extensive analysis. The connection tracking module contains functionalities necessary for maintaining state information about HIP associations. The packet filtering module calls connection tracking when packets need to be filtered with the state option. Still, all HIP packets are analyzed by the connection tracking functionalities, independent of the filtering options used. This ensures that necessary information regarding the HIP association is obtained from the packets.

In addition, the connection tracking provides similar signature verification functionality as the authentication done in the packet filtering module. Here the responder HI is extracted from HIP traffic dynamically. It is then used to authenticate the end-point in further communication. The authentication here has different nature than in an actual authentication module. The identity itself is not essential, as it is when defined in a firewall rule. Instead, the connection tracking attempts to assure the property of sender invariance [34]. This guarantees that, independent of the actual identity, the traffic can be trusted to be sent by the same end-point throughout the connection. Accordingly the state information maintained by the firewall is authenticated and reliable.

5.3 Summary

This section first described the two overall design alternatives that were studied for the firewall solution. First of these, extending the Linux Netfilter framework, would have included implementing a HIP specific extension module as well as modifying the existing connection tracking mechanisms to support HIP.

The second alternative, a separate HIP firewall solution, was selected due to better suitability for the project. Still, some well-founded design choices were adopted from the Netfilter framework also for this design. The main components of the design include the firewall main module, the firewall policy management, the packet filtering module with stateless filtering functionalities and the connection tracking module.

Chapter 6

Implementation

The overall architecture and the main components of the firewall solution were presented in the Chapter 5. This chapter presents the implementation of the firewall solution in more detail. The external components, that the implementation depends on, are first summarized. Implementation of each firewall component is then presented. The chapter gives examples of the data structures used and presents selected core functionalities in more detail. Finally, a sequence diagram is used to illustrate larger entities and the interconnections from functional point of view.

6.1 External Components

The Implementation relies on certain external libraries and systems. These include LIB_IPQ library from Netfilter framework, GLib library and the HIPL [1] HIP protocol implementation. Of these, the LIB_IPQ is used for receiving packets sent to the host and for issuing verdicts on them. The GLib library provides useful data structures and methods for manipulating them as well as other necessary functionalities. GLib is used for example for list structures and their manipulation and for thread and time functionalities.

HIPL provides several data structures and functions relating to HIP protocol and for manipulating HIP packets. For instance, HIPL contains detailed data structures representing the HIP protocol packets and the different parameters. It also includes functions for searching parameters in HIP packets and for example verifying the packet signatures.

6.2 Firewall Main Module

As described in Chapter 5, the main module ties together the other components of the firewall and calls each component as necessary. The main module initializes the necessary components and starts up the firewall. This includes also issuing calls to parse the firewall rules from a file defining the firewall security policy. The file name is defined as an argument when firewall is started. The main module also uses the LIB_IPQ interface for registering the firewall software to receive the packets intercepted by Netfilter. For the actual packet filtering the main module receives the packets through the LIB_IPQ interface. It then issues calls for analyzing each packet and finally delivers verdict for each packet back to the Netfilter system.

6.3 Packet Filtering Functions

Packet filtering functions are in practice included in the main module, but are logically a separate set of functions. Packet filtering contains simple functions that analyze different properties of the packet in relation to a certain option of a firewall rule.

Source and destination HITs are matched by a function comparing the HIT values. Source HI may also be defined in a firewall rule as a sub option to the source HIT. Source HI matching verifies that the packet signature has been created with the HI defined in the rule. In effect, this authenticates the packet. HI matching uses the signature verification functions provided by HIPL.

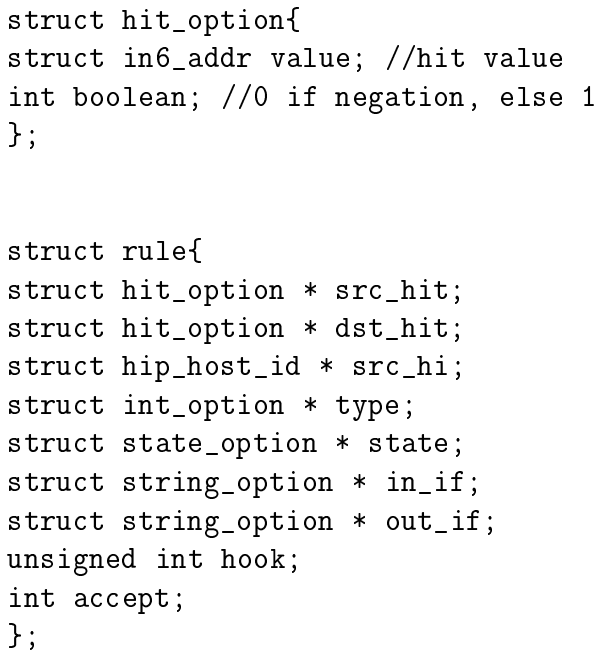
Other filtering functions are used for matching the type of packet and the incoming and outgoing network interface of the packet. Information about the network interfaces is passed to the firewall from the Netfilter system along with the packet.

6.4 Firewall Policy Management

Firewall policy management contains data structures representing firewall rule and its different options. It also provides functions for parsing a rule from character string representation to the rule data structure and for manipulating the set of rules that define the firewall security policy.

6.4.1 Data Structures

The rule data structure contains information defining which packets it is applied to and what should be done with these packets. It consists of different options, the hook through which the packet came in and an accept value to indicate whether to accept the packet or not. When an option is defined in the rule it contains a pointer to a structure defining the details of the option. For undefined options the pointer value is set to null. A firewall rule may also define a so called default target for a certain hook. This rule contains none of the options and in effect matches any packet coming in from a certain Netfilter hook.

An example of an option is shown in Figure 6.1 along with the actual rule structure. All the option structures follow a same basic format but are differentiated by the type of data and possible sub option values they store. In the option structure the value contains the actual value of the option and boolean indicates whether the option was negated with '!'.


```
struct hit_option{
struct in6_addr value; //hit value
int boolean; //0 if negation, else 1
};
```

```
struct rule{
struct hit_option * src_hit;
struct hit_option * dst_hit;
struct hip_host_id * src_hi;
struct int_option * type;
struct state_option * state;
struct string_option * in_if;
struct string_option * out_if;
unsigned int hook;
int accept;
};
```

Figure 6.1: Rule data structure and an example of an option structure.

The option data structures are intended to be generic, so that they could be used for variety of different options that need to store a value of certain data type. For example `string_option` data structure may be used for any

option that has a string value. The `hit_option` could be alternatively used for storing options with an IPv6 address.

6.4.2 Parsing Rules

As the user defines rules in string representation, it is necessary to parse them to the internal representation used by the firewall. This includes ensuring that the values defined for options are meaningful and that the overall syntax and semantics of the rule are correct.

For parsing source or destination HITs, the HIT values are converted from strings to `in6_addr` structures. Source HIs are defined with a path to a file containing the public key. When parsing HI option, it is first ensured that a file exists. The file name must contain either “`_dsa_`” or “`_rsa_`” for identifying either DSA or RSA as the algorithm of the public key. The public key is then read into a `hip_host_id` structure. This uses modified functions from HIPL for loading the key from a file. The implementation then checks that hash value of the HI equals the source HIT for which the source HI was defined.

For type option and for hook the string values are matched against the type and hook names and converted to integers. Input and output interfaces are string values that are limited by the maximum length.

6.4.3 Interface for Firewall Management

The firewall policy management module contains a simple interface which can be used for extending the firewall management. Currently, the firewall rules may only be inserted through a configuration file, which is read in during start up. In a more advanced system the firewall rules should be updated interactively at run time. The interface follows the format of the Iptables user interface in the Netfilter system. It includes methods for inserting a rule, deleting a rule as well as listing and removing all rules, as illustrated in Figure 6.2.

Managing the rules concurrently, as the firewall is operating, requires synchronization. There could be potentially multiple threads reading and writing into the rule lists, while the firewall analyzes the rules for filtering a packet. The rule lists must remain consistent during these operations. The situation can be modeled as a classic readers and writers problem and can be controlled with synchronization mechanisms presented in [11]. The solution guarantees

```
void insert_rule(const struct rule * rule, int hook);
int delete_rule(const struct rule * rule, int hook);
GList * list_rules(int hook);
int flush(int hook);
```

Figure 6.2: Functions for managing firewall rules.

that readers and writers may not operate simultaneously, but several read operations may occur concurrently. In the case of firewall, write operations would be updates for firewall rules. The read operations would be performed by both the rule managing and the firewall packet filtering.

Within this solution there are two choices of preference. The solution may favor writers, so that when ever a writer wishes to write, no new reader is allowed to read before the write operation has occurred. Alternatively read operations may be prioritized and no reader has to wait unnecessarily if write is not taking place at the moment. The first solution decreases concurrency and is potentially less efficient. The latter solution could, however, result in writers waiting indefinitely if read operations occur as a steady stream.

The first solution, enabling faster writes was selected. The firewall management is expected to be rather infrequent and the management may be controlled by an administrator. Thereby, write operations are not likely to burden a firewall excessively. Furthermore, the read operations occur whenever a packet needs to be filtered and are potentially a steady stream of operations. Also, this load depends on network traffic and can not be easily controlled, unlike the management operations. In effect, the solution prevents large network loads from blocking the necessary management operations.

6.5 Connection Tracking

Main function of connection tracking is to maintain necessary state information for recognizing packets that belong to a connection. HIP protocol packets are simple to manage as they all carry source and destination HITs and can be authenticated when HI is available. With data packets the flow identifier must be deduced from HIP protocol packets when the association is created or updated.

Connection tracking provides two public entry point functions that analyze packets. The `filter_state` function is used for analyzing packet in relation

to a given state option. The `contrack` function is called for packets that are not filtered by a state option, but must be analyzed in order for the connection tracking to be aware of any changes in the connection state. Both of these functions call same internal analysis function for the packet. The analysis performed with each packet is more closely described in the following section.

6.5.1 Functionality

HIP connection is initialized with a base exchange procedure. When analyzing the base exchange packets, the connection tracking code extracts data that will be needed in further filtering of packets of the particular connection. This data includes SPIs and destination addresses used in the data packets and the responder HI, when this is required in a state option of a firewall rule. During base exchange, necessary data structures are also created for the connection.

When the base exchange is completed the connection tracking code is able to recognize the ESP data packets relating to the connection. Also when verifying responder packets is required in the filtering rules, the connection tracking uses the HI information to authenticate the responder packets.

Connection tracking analyzes update packets sent in the HIP connection. When a new destination address under an SPI or an altogether new SPI is announced by a host this information is stored in the ESP data structures related to the connection. The connection tracking must take into account that the two end-points each maintain separate state information. Due to this, changes announced by one party can not be considered to be known by the other party immediately. This affects for example rekeying situations, where old information must remain valid until the other end-point has acknowledged the new information. In practice, data packets with old SPI, could still be on the way when new SPI is announced. This principle is also discussed in [36] in the context of TCP protocol.

The HIP connection closing was still under development in the HIPL while the firewall was implemented. Due to this, connection timeout checking was developed as an alternative method for being able to remove HIP connections from the firewall memory. The implementation allows setting a timeout value after which unused connections are removed. If zero or negative value is specified, the timeout checking is not performed.

The connection tracking module inserts a time stamp into connection data structure. The time stamp is then updated whenever packets of the connec-

tion are encountered. For detecting idle connections these time stamps are periodically checked against a certain timeout value. The timeout value is defined as an argument when firewall is started and passed from the main module to connection tracking when timeout checking is initialized. When idle time of a connection exceeds the timeout value the connection and all data relating to it are removed. In practice the timeout checking is implemented with a separate thread.

6.5.2 Data Structures

Connection tracking models the connections with structures similar to those of Netfilter connection tracking. The structures are illustrated in Figure 6.3. In the HIP firewall, the model does not need to be as generic as with Netfilter framework as only a single protocol is supported. Accordingly, for instance relating HIP and ESP data is done in more straightforward manner.

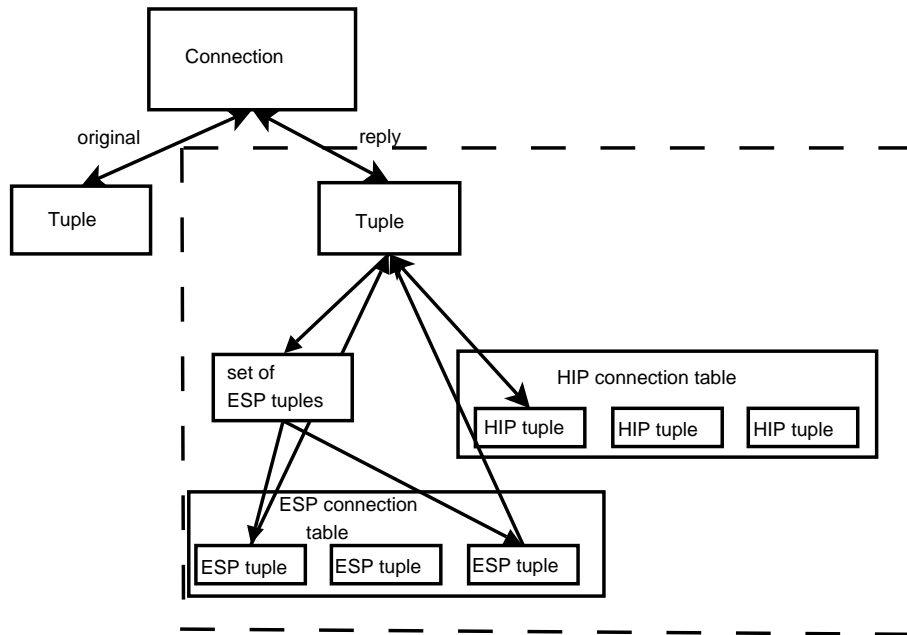


Figure 6.3: Connection tracking data model. Arrows represent pointer references between the data structures.

As with Netfilter, a tuple data structure contains information that directly translates into information carried by a packet. The implementation provides tuples for both HIP and ESP packets. The tuples are contained in HIP and

ESP connection tables. As a packet is received the connection tracking code searches for a HIP or ESP tuple in connection table that matches the packet. If a tuple is found, there exists a connection into which the packet belongs to.

A joint tuple structure contains the HIP tuple and all the ESP tuples of a certain direction of a connection. The HIP and ESP tuples also have pointers back to this joint structure. The actual connection contains two tuples representing the two directions of the connection. Both of these tuple structures also contain pointer to the connection structure as noted in Figure 6.3.

The HIP and ESP connection tables are implemented as linked lists. This is adequate choice for this type of application. However for a more performance critical implementation, a hash table would be more efficient data structure as elements could be searched in constant time. For HIP packets the HIT values could be used to calculate hash value for a tuple and for ESP tuple, the SPI value and the destination address could be used to produce unique hash value.

6.6 Interaction Between Components

This section summarizes the overall functionality of the system. It also describes how different parts of the system interact together. This is illustrated in Figure 6.4.

At start up the main module uses `LIB_IPQ` to first register the firewall application to receive packets intercepted by Netfilter. It also issues call to firewall policy management to process the file defining the firewall rules. Based on this processing the policy management module forms a list of rules for each of the Netfilter hooks, `INPUT`, `OUTPUT` and `FORWARD`. These rules can then be queried by the main module and the packet filtering functionalities as necessary. Finally, the main module calls the connection tracking module to initialize the connection timeout checking functionality.

After this the firewall is ready to start receiving and processing packets. From Netfilter, the firewall receives packets which are defined to be queued for processing of registered user space applications. The packet type is analyzed to determine whether packet is HIP or ESP packet or of some other type. Filtering function is called for HIP and ESP packets.

Filtering function, in turn, calls policy management for getting the firewall

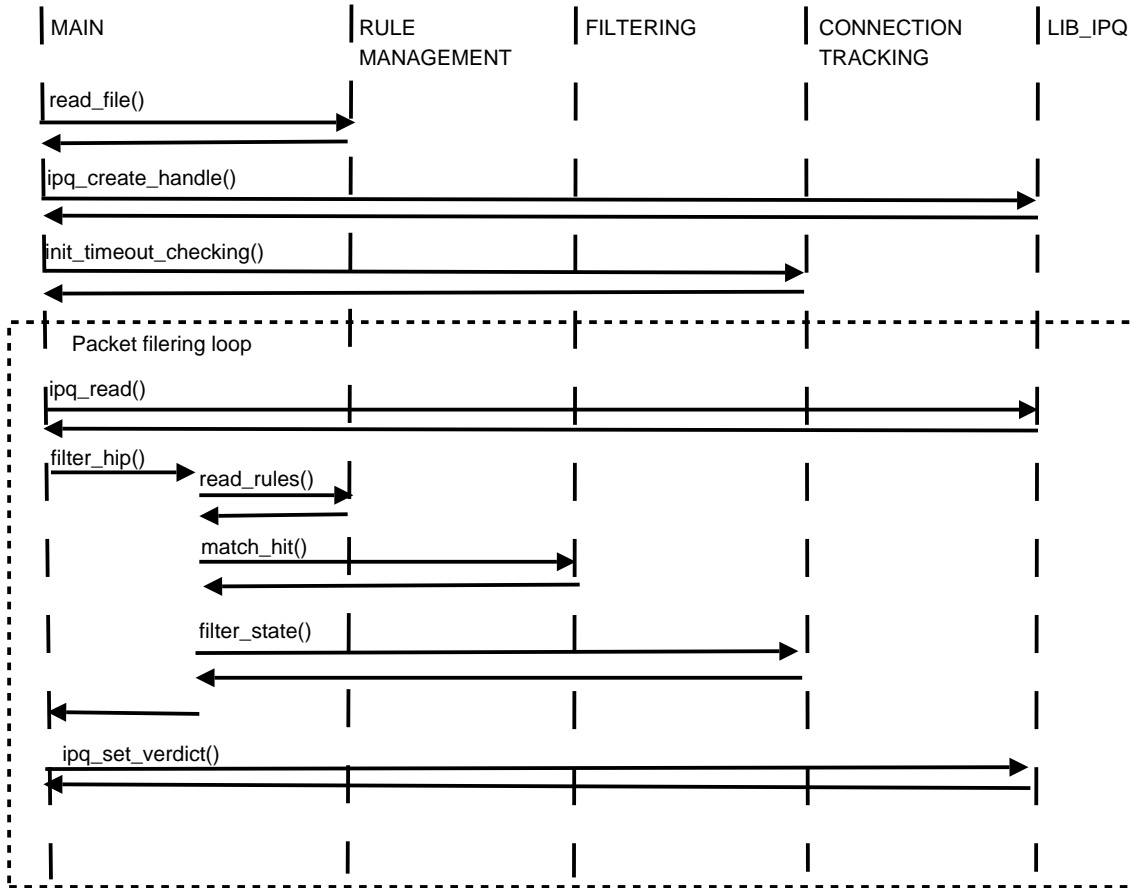


Figure 6.4: Overall functional sequence of the system. The figure presents a slightly simplified example of the firewall call sequence.

rules for the particular hook through which the packet came in. The filtering function traverses the list of rules and looks for a rule matching the properties of the packet. Matching is done by calling the filtering functions or connection tracking as necessary. When a matching rule is found, the function returns the target of the rule to caller. If none of the firewall rules match the packet and no default target is defined, accepting verdict is returned as a default response.

For each rule, the filtering function analyzes each option that has been defined. Filtering function calls the specific packet filtering functions for each of the options. These filtering functions and the `filter_state` function of connection tracking analyze the packet properties in relation to the option passed as an argument and return boolean value indicating whether packet

matches the option.

6.7 Summary

This chapter described implementation of each of the firewall components. Further details were also presented about selected issues. These included for example the readers and writers -synchronization mechanism selected for the firewall policy management interface and the connection tracking functionalities. The overall functionality was also illustrated to provide a general view of interactions between the firewall components.

Chapter 7

Analysis

This chapter first evaluates the firewall implementation against the requirements specified in Chapter 4. The focus here is on the functionalities that have further implications or that require additional consideration. Also the testing environment is presented.

Second point of view for analysis includes the more general implications of HIP to firewalls. Many such issues were already discussed in Chapter 3 and laid a groundwork for implementing the HIP enabled firewall. Therefore this chapter focuses on implementation related observations and issues that have affected the implementation.

7.1 Evaluation Against Requirements

This section addresses the requirements introduced in Chapter 4. Ongoing development of protocol poses some restrictions on features of the firewall and to what extent some functionalities could have been implemented. As mentioned in requirements, the implementation limits to features implemented by the HIPL [1]. At the time of implementation the HIPL supported HIP base draft version 01 and HIP mobility and multihoming draft version 00, with some limitations.

7.1.1 Test and Development Setting

The main test method used for verifying the implementation against requirements was system level testing of the firewall solution. Test functions have also been implemented for rule management functions. For the actual firewall

functionalities functional system level testing was, however, considered the best option. Especially state keeping involves analyzing sequence of packets which all have side effects to data maintained by the firewall. Therefore, the system was tested with actual HIP traffic traversing through the firewall. As the firewall is a prototype implementation, also reasonable amount of debug information is printed out. The debug information describes the analysis that leads to packet being accepted or dropped. This enables monitoring actions and the internal state data of the firewall.

The testing environment uses VMware Workstation virtual machines. This allows simulating a network of several hosts in one physical host machine. The test setting is depicted in Figure 7.1.

The main testing and development setting includes two virtual networks connected by a virtual host (*FW*) acting as the firewall. The firewall host is running a Linux kernel with the HIPL user space HIP implementation which also includes the HIP enabled firewall program. The virtual HIP hosts (*host 1*, *host 2* and *host 3*) contain Linux kernels with the HIPL kernel version. The firewall implementation was tested by creating HIP associations through the firewall host, between the HIP hosts. This setting was necessary, as the HIP enabled firewall is based on the HIPL user space implementation, which at the time of implementing was being developed. Therefore the kernel implementation of HIPL was used for establishing the HIP associations.

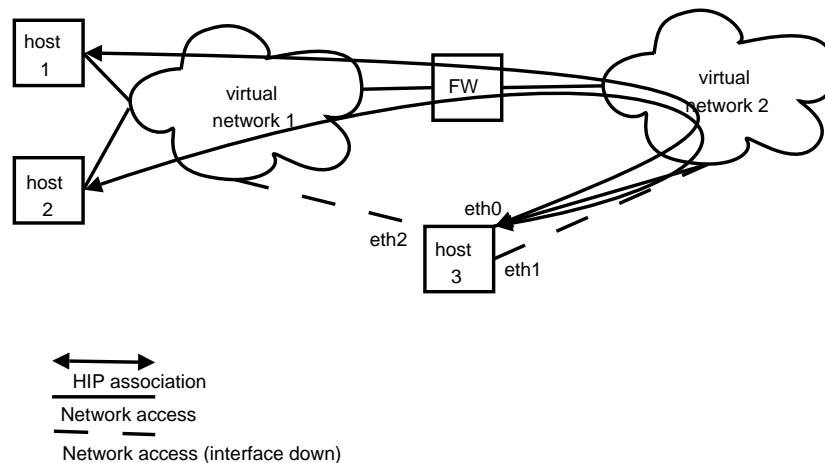


Figure 7.1: Test setting with virtual machines and -networks.

The HIP hosts (*host 1*, *host 2* and *host 3*) had multiple network interfaces and could be connected to both networks, as illustrated with *host 3* of the Figure

7.1. As a consequence, the setting could be used to simulate mobility situation, where a mobile host with ongoing HIP connection enters into a network protected by the firewall. In the Figure 7.1, this could be accomplished by first creating a HIP association from *host 3* through network interface *eth2* to either of the hosts in *network 1*. After that the interface *eth2* would be brought down and a new interface, *eth1*, would be brought up in *network 2*. This sequence was used to test state establishing for a mobile host and is further discussed in section 7.1.5.

7.1.2 Overall Functionality and Interfacing to the Communication System

The firewall implementation interfaces to the communication system through the Netfilter framework, as mentioned earlier. This also requires management through the Netfilter user interface, the Iptables. The Iptables rules must define that the HIP and ESP packets must be queued for user space handling.

Besides requiring user input through two different interfaces, there could, in principle, be complications in a system if several different firewall solutions need to be used. In a production level system this type of solution could be too cumbersome. Instead, all firewall functions are ideally integrated in a single system. For a proof of concept level implementation this solution is, however, adequate.

7.1.3 Firewall Policy Management

The external interface provided for managing the firewall policy is largely as defined in the requirements. The same format of rules is provided by the firewall as presented in Figures 4.1 - 4.3. The firewall rules and their options also map quite directly to the different features provided by the firewall. The syntactical and semantic analysis was tested with a set of different configuration files that were parsed by the firewall.

Besides checking the syntactic and semantic issues with the rules, the rule management provides functions for managing the rule lists in the firewall. This was necessary for providing interface for managing firewall also at runtime. Test functions were implemented for rule management functions.

7.1.4 Stateless Packet Filtering

The stateless packet filtering options require quite straight forward functionalities. The implemented functionalities follow those defined in the requirements. These filtering options are applied to the HIP packets and, as possible, also to the ESP packets.

For ESP packets, the input and output interface options are, of course, valid as well as the state option, while the HIP packet type option is not applicable. The source and destination HITs can not be directly derived from the packet but can be analyzed in the context of the state option filtering. Therefore the HITs are analyzed from the state information maintained for the connection. Applying these options also to ESP packets makes the firewall solution in some ways more flexible and adaptable.

7.1.5 Connection Tracking

The connection tracking functionality satisfies the general requirements placed on it. The HIP packets of an association are recognized and analyzed and necessary information is derived for filtering the related data traffic. Changes in data traffic flow identifier are also detected from traffic. When required by the firewall rules, the responder HI is extracted from the base exchange packets and used for authenticating rest of the control packets.

In practice, the connection tracking features were tested with the test setting described earlier. HIP associations were created through the firewall to analyze base exchange handling. The processing of mobility and multihoming signaling was verified by adding and removing addresses of a host during an ongoing HIP connection. For creating update exchanges where also rekeying was performed, new interface was brought up in an established HIP association. Different sets of firewall rules were also used to test for example removing the established connections due to blocked HIP and ESP packets.

Mobility

Due to mobility, an already established HIP connection may need to traverse through a new firewall. In practice, this can occur when a host with ongoing HIP connection moves into a network protected by a firewall. In this scenario the firewall is not aware of the connection, but the mobility signaling may provide enough information for establishing state for the connection.

Implementing state establishment through mobility signaling was somewhat

problematic due to ongoing development of the HIP implementation. The SPI parameter sending was under development and therefore state establishment requiring the SPI parameter could not be implemented. In practice, this refers to a situation, where a new address is added to an existing network interface of the host. However, in the case where a new interface in the new network is used for communication, the state could be established. Here the corresponding host SPI is received from the NES parameter. In practice this was tested with the setting described in section 7.1.1. Even though also the first scenario would be useful feature in the firewall, this functionality demonstrates how HIP protocol information enables establishing state even for ongoing connection.

An additional consideration here is that when state is not established from the actual base exchange, the responder HI is not available for the firewall. This limits the security that a firewall is able to provide for these mobile connections. Therefore an explicit option “`-accept_mobile`” exists for allowing this functionality with connection tracking. Due to this, a firewall rule with state option is not allowed to have both `-verify_responder` and `-accept_mobile` sub options defined.

Connection Closing

At the time of implementing, connection closing mechanism was still under development in the HIPL. Missing the close packet sequence, however, further emphasized the need for a connection timeout mechanism. The ongoing connections reserve memory of a firewall and the firewall with its resources is often vital to communication with the protected hosts.

Also, it is possible that the firewall is not able to intercept the proper closing sequence of a connection. This may happen for instance if one or both hosts become unavailable. Mobility or multihoming may also cause situations where the firewall is no longer able to intercept packets of the connection. In this case as packets no longer traverse the firewall, the connection is left open in the firewall and the data it reserves continues to be stored in the firewall memory.

Due to these issues, it may be practical for a firewall manager to be able to set some timeout value after which unused connections are removed. Connection timeouts in transparent middleboxes are however controversial issue. From protocol point of view this does not follow the transparency rule discussed in Chapter 2. This may cause situations where packets of a legitimate connection are blocked. In the case of mobility and multihoming this may happen

even if the timeout value is set to exceed the general Unused Association Lifetime value mentioned in the HIP draft [22].

In practice, the suitable timeout value needs to be balanced between the needs of the organization and the requirement for flexibility. The timeout checking may also be disabled, for cases where idle time of the connection is not wished to be limited. Also, this is one of the issues where the registration requiring firewall with its soft state approach, would be very beneficial.

7.1.6 Non-Functional Requirements

The identified non-functional requirements focused on properties of the design and implementation, such as modularity and maintainability. The general suitability of the design could be evaluated during the implementation. The division of functionalities into modules seemed practical and no major changes were needed. Also the planned interaction between the components was followed in implementation. Only exception to this was the stateless HIT filtering options applied to the ESP packets, as this needed to be performed in the connection tracking module. This functionality was discussed in section 7.1.4.

As mentioned in requirements, the main challenge for a HIP enabled firewall implementation is the ongoing development of the HIP specification and the implementations. This requires also updating the firewall implementation. The firewall implementation uses data structures and functionalities from the HIPL. Therefore the firewall will also benefit from further development of this HIP implementation. Besides this, the firewall is likely to require updating of its internal functionalities, as the protocol evolves. The firewall implementation may also be developed further. The main foreseeable addition would be the registration capability. How the current implementation could be extended to support registration is further discussed in section 8.1.

7.2 General Analysis of HIP Enabled Firewalling

General interactions between firewalls and HIP protocol were already analyzed in Chapter 3. This section discusses some issues and implications of HIP that affect especially a firewall implementation. These issues focus especially on how HIP as a protocol influences firewall design and implementation.

7.2.1 Role of HIP Enabled Firewall

As HIP introduces security features for end-to-end communication, this also calls for re-evaluation of the role of firewalls. Firewalls have traditionally attempted to provide authentication and for instance encryption has been often provided by VPN solutions. In the context of HIP these are, however, integral part of the end-to-end communication. Therefore the role of a HIP enabled firewall would include mostly managing the access control information and making the decisions based on this information. Here some responsibility of the security, such as end-to-end encryption of data traffic, is shifted to the end-hosts.

Furthermore, with its authentication mechanisms HIP enabled firewall serves better as a centralized security perimeter of an organization than as a personal firewall protecting a single host. Even though HIP is in many ways transparent to middleboxes, the end-hosts have still more efficient means in authenticating the traffic. By analyzing the HMAC parameter, a HIP host is able to first ensure the message validity, with less use of CPU power. Therefore, having a HIP enabled firewall program in an end-host first analyze the packet signature, would undermine the benefits of HMAC inspection. Nevertheless, a HIP enabled firewall would also here be a natural place to maintain access control lists and enforce them.

7.2.2 Registration Requiring Firewall

The firewall solution of this thesis implements a transparent HIP enabled firewall. In the case of a registration requiring firewall the initiating host communicates directly with the firewall. As a result, the transparent firewall acts as the first stage of HIP enabled firewalling for HIP end-hosts that do not yet include the registration capability.

The transparent HIP firewall can be also more easily deployed, as it does not necessarily require changes in the host protocol stack. The firewall host does not necessarily need to be a HIP host. Here the firewall implementation is built on top of the HIPL user space implementation to avoid copying the HIP code unnecessarily to the firewall implementation. However, a transparent HIP enabled firewall is not dependent of HIP implementation, even though some parts of a HIP implementation may be useful in the firewall implementation.

Even though transparent firewall has the advantage of supporting the hosts that lack the registration capability, the registration provides significant ben-

efits to the firewall. As pointed out before, especially the soft state functionality makes the firewall function in a more flexible manner. State is automatically deleted if it is not updated for a certain period of time and the state is re-established without re-establishing the connection. This is also beneficial in managing resources used by the firewall, as unused resources can be more easily freed.

7.2.3 HIP Protocol Implications to Firewall Design and Implementation

The HIP HIs provide much needed true identifier for end-hosts. As many other systems operating within the Internet architecture, also firewalls suffer from the semantic overloading of IP address. Therefore, mobility, multi-homing and unreliability of the IP address as an identifier are straining also viability of firewalls. HIP HI provides invariable identifier that is not affected by changes of location or the particular network interface used. Due to the introduction of HI, HIP communication can also be associated to a correct connection, even when there may be third parties involved in delivering it. By contrast, for example use of Mobile IP is problematic for stateful firewalls particularly because third parties are involved and the association in firewall is still identified with the IP addresses in the packets [20].

The cryptographic properties make HIs essentially different identifiers from IP addresses. IP addresses are rather unreliable as end-point identifiers. To address this issue, firewalls currently try to analyze intricate protocol data to obtain further assurance that the sending host is in fact the one that IP address indicates. As pointed out earlier, this analysis only ensures that the sending host is as aware of the connection state as the firewall itself. It may not, however, ensure that the sender is in fact the other end-point of the connection. Instead, HIP traffic is reliably authenticated with less complicated mechanisms.

In both above mentioned aspects the traditional firewalls would need to contain detailed information about the protocol in order to be able to filter traffic. Furthermore, different application level protocols operate in very different manners and create different sets of transport level connections in the process. This leads to overly complicated designs of stateful firewalls, which are then prone to errors and costly to develop and maintain. This issue has been generally recognized. One proposed solution is signaling between the middleboxes and the end-hosts, which is currently investigated by the Midcom working group of IETF [2].

Explicit signaling is also possible with HIP as discussed in the context of registration requiring firewalls. Yet, HIP as such already simplifies firewall functionality. The approach in design and implementation here has been to limit protocol logic to what is necessary for obtaining data packet flow identifiers and HI for authentication. That, in fact, is the essential information conveyed in HIP.

7.3 Summary

This chapter first analyzed the implemented HIP enabled firewall solution in relation to the requirements specified in Chapter 4. The implementation satisfies the requirements defined for it. However, some firewall functionalities could still be further developed as the HIP protocol implementation is still progressing. The chapter also presented the testing methods used for verifying the functionalities.

Implications of HIP to firewall design and implementation were also analyzed in a more general level. To summarize, the security properties and visible signaling information of HIP support and simplify stateful firewall design and implementation. Still, use of the HIP registration protocol could further improve the HIP enabled firewall functionalities and provide more flexibility and robustness.

Chapter 8

Conclusions

Both firewalls and HIP are strongly security oriented technologies. Their focuses are, however, somewhat different. Firewalls analyze intercepted traffic to provide centralized security perimeter for a set of hosts. HIP, on the other hand, secures communication between two communicating end-hosts. This thesis analyzed how these two technologies should coexist and what benefits or challenges this may raise.

HIP takes middleboxes, such as firewalls, well into consideration. HIP HIs and the HITs derived from them provide a useful identifier also for access control information. HIP is also transparent to middleboxes as necessary information in protocol packets is left unencrypted. Traffic belonging to a HIP association can be recognized by a stateful firewall even when host is mobile or uses multiple network interfaces.

Possibly the most significant benefit are the security features that HIP provides to firewalls. By verifying the signatures in protocol packets, the firewall is able to reliably authenticate the sender host. In effect, the security is deeply embedded into the communication instead of being an add-on to the technology.

Different aspects of HIP may have effects on firewall functions. One of the most direct would be the registration protocol defined in HIP. Especially the soft state approach benefits stateful firewalls and enables more flexible and robust firewall functionalities.

For the HIP enabled firewall, two main design alternatives were considered. The implementation could have extended an existing firewall system, the Linux Netfilter, or it could be implemented as a separate HIP firewall system. The latter alternative was selected for the firewall solution. However, well-

founded design choices were also adopted from the Netfilter framework.

The implemented firewall solution demonstrates the feasibility of HIP enabled firewall technology. It provides transparent HIP firewalling and satisfies the requirements set to it. In general, the firewall implementation follows the extent to which the reference HIP implementation was implemented. The firewall includes both stateful and stateless packet filtering functionalities. As required, the firewall is able to authenticate traffic using the HIs of connection end-points.

To conclude, HIP enabled firewalls can provide significant benefits compared to traditional firewall functionalities. These include added security and more flexible handling of traffic, even in the case of multihoming and mobility. Accordingly, HIP enabled firewalling could be one of the factors that further aid the deployment of this emerging technology.

8.1 Future Work

This section outlines possible directions for further development of the HIP enabled firewall prototype. These include both development of additional functionalities as well as further improvement and analysis of the currently implemented features.

8.1.1 Supporting Updated HIP Specifications

The obvious line of further development is to extend the firewall to cover the aspects of HIP protocol that are currently missing. This can continue along with developing the protocol implementation, which has progressed during the firewall implementation.

Two features already mentioned were handling the close packets, which is currently already included in the protocol implementation, and the SPI parameter sending. The latter could be used to enable creating firewall state in the mobility situation described in analysis.

Another change concerns the parameters delivered in update packets. The REA and NES parameters have been changed into LOCATOR and ESP_INFO parameters. The update information is, however, transmitted in similar manner as before.

8.1.2 Extending Firewall to Include Registration

Currently the firewall implementation is transparent to the end-hosts. It could be also extended to include the registration capability, which would provide additional security to the firewall itself. This way state would not be established before authenticating the initiating end-host. This section outlines the necessary interactions between the firewall program and the HIP protocol implementation for implementing registration in the firewall solution. The registration capability was discussed in Section 3.1.2.

In this case, the registration protocol functionality would first need to be adopted into the HIPL protocol implementation. Here it is assumed that the actual registration protocol communication would be best performed by the protocol implementation. This would be reasonable as the registration protocol reuses HIP functionalities.

The firewall registration functionality would then require interaction between the firewall system and the HIP protocol implementation. This is straightforward as the firewall is built alongside the user space version of HIPL, which makes passing data back and forth simpler. Accordingly, there needs to be an interface to the actual service, here providing firewall traversal, that the protocol implementation can use. It is possible that this interface definition could accommodate multiple different services.

According to current definitions, the registration may be initialized either directly with the firewall or then firewall may intercept the I1 packet. In the first case, the HIP protocol implementation comes to contact with the packet and must consult the firewall service. The firewall must make access control decision based on the properties of the packet and the HIP protocol implementation can then proceed to accept or deny request.

In the second case, the firewall intercepts HIP packets intended for other hosts. The HIP implementation must therefore provide information of established and expired registration associations for filtering these packets. The firewall may then trigger the registration R1 packet sending in the HIP implementation if a HIP packet with missing registration is encountered. The firewall can also use the registration status information to remove connections from memory as the registrations expire.

8.1.3 Production Level Firewall Solution

In situation where HIP is more extensively deployed and used more widely, the firewall solution also needs to be more advanced. This would require

further development and more extensive quality assurance.

A production quality firewall needs to be extensively verified for all possible error conditions that a malicious host may cause in the firewall. One required aspect is therefore more exhaustive testing to further ensure the quality of the solution. The analysis would need to include testing with different distorted HIP packets, which could cause problems in HIP packet handling. Testing should also include different cases of abnormal behavior from end-hosts, such as sending packet sequences different from specifications.

Another aspect would be a more extensive study of the efficiency of the firewall and requirements posed by that. As HIP enabled firewall includes potentially CPU intensive operations, such as the signature verification, there should be analysis concerning the resource consumption of the firewall. This should produce estimates of necessary amounts of system resources, including CPU power and memory, for different scenarios and load conditions. Also, this should not limit to normal operation with well-behaved hosts. The analysis should also include estimates on the effects that malicious hosts or attacks of different magnitude can cause on the firewall performance.

Bibliography

- [1] HIPL: HIP for Linux. Helsinki Institute for Information Technology, <http://infrachip.hiit.fi/hipl/about.html>.
- [2] Middlebox Communication (midcom) working group, IETF.
- [3] The Netfilter/Iptables project. <http://www.netfilter.org/>.
- [4] ANDREASSON, O. *Iptables Tutorial 1.1.19*, 2001.
- [5] AURA, T., NAGARAJAN, A., AND GURTOV, A. Analysis of the HIP Base Exchange Protocol. In *ACISP'05* (July 2005).
- [6] BELLOVIN, S., AND CHESWICK, W. Network firewalls. *Communications Magazine, IEEE* 32, 9 (September 1994), 50 – 57.
- [7] CANDOLIN, C., KOMU, M., KOUSA, M., AND LUNDBERG, J. An Implementation of HIP for Linux. In *Proc. of the Linux Symposium* (July 2003).
- [8] CARPENTER, B. Internet Transparency. Fequest for Comments 2775, IETF, February 2002.
- [9] CARPENTER, B., AND BRIM, S. Middleboxes: Taxonomy and Issues. Request for Comments 3234, IETF, February 2002.
- [10] CHESWICK, W. R., BELLOVIN, S. M., AND RUBIN, A. D. *Firewalls and Internet Security: Repelling the Wily Hacker*, second ed. Addison-Wesley Professional Computing Series. Addison-Wesley, 2003.
- [11] COURTOIS, P. J., HEYMANS, F., AND PARNAS, D. L. Concurrent control with "readers" and "writers". *CACM* 14, 10 (October 1971), 667 – 668.

- [12] ELLISON, C. M., FRANTZ, B., LAMPSON, B., RIVEST, R., THOMAS, B., AND YLÖNEN, T. SPKI Certificate Theory. Request for Comments 2693, IETF, September 1999.
- [13] FREED, N. Behavior of and Requirements for Internet Firewalls. Request for Comments 2979, IETF, October 2000.
- [14] HENDERSON, T. R., AHRENHOLZ, J. M., AND KIM, J. H. Experience with the Host Identity Protocol for Secure Host Mobility and Multihoming. In *WCNC 2003 - IEEE Wireless Communications and Networking Conference* (March 2003), vol. 4, pp. 2120 – 2125.
- [15] HERSCOVITZ, E. Secure virtual private networks: the future of data communications. *International Journal of Network Management* 9, 4 (1999), 213–220.
- [16] JOKELA, P., MOSKOWITZ, R., AND NIKANDER, P. Using ESP transport format with HIP. Internet-Draft, IETF, February 2005.
- [17] KENT, S., AND ATKINSON, R. IP Encapsulating Security Payload (ESP). Request for Comments 2406, IETF, November 1998.
- [18] LAGANIER, J., AND EGGERT, L. Host Identity Protocol (HIP) Rendezvous Extension. Internet-Draft, IETF, February 2005.
- [19] LAGANIER, J., KOPONEN, T., AND EGGERT, L. Host Identity Protocol (HIP) Registration Extension. Internet-Draft, IETF, February 2005.
- [20] LE, F., FACCIN, S., PATIL, B., AND TSCHOFENIG, H. Mobile IPv6 and Firewalls Problem statement. Internet-Draft, IETF, August 2004.
- [21] MOSKOWITZ, R., AND NIKANDER, P. Host Identity Protocol Architecture. Internet-Draft, IETF, December 2004.
- [22] MOSKOWITZ, R., NIKANDER, P., (EDITOR), P. J., AND HENDERSON, T. Host Identity Protocol. Internet-Draft, IETF, October 2005.
- [23] NIKANDER, P. HIPpy Road Warriors Jumping Hoods over Road Blocks. IRTF Host Identity Protocol (HIP) Research Group, Workshop on HIP and Related Architectures, Washington DC, November 2004.
- [24] NIKANDER, P., ARKKO, J., AND HENDERSON, T. End-Host Mobility and Multi-Homing with Host Identity Protocol. Internet-Draft, IETF, February 2005.

- [25] NIKANDER, P., YLITALO, J., AND WALL, J. Integrating Security, Mobility, and Multi-Homing in a HIP Way. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS'03)* (February 2003), Internet Society, pp. 87–99.
- [26] POHLMANN, N., AND CROTHERS, T. *Firewall Architecture for the Enterprise*. Wiley Publishing, Inc, 2002.
- [27] RUSSELL, R., AND WELTE, H. *Linux netfilter Hacking HOWTO*, 2002.
- [28] SALTZER, J. H., REED, D. P., AND CLARK, D. D. End-to-end arguments in system design. In *ACM Transactions on Computer Systems (TOCS)* (November 1984), ACM, pp. 277 – 288. ISSN:0734-2071.
- [29] SCHNEIER, B. *Applied Cryptography, Second Edition*. John Wiley & Sons, Inc., 1996.
- [30] STIEMERLING, M., QUITTEK, J., AND EGGERT, L. Middlebox Traversal of HIP Communicationg. IRTF Host Identity Protocol (HIP) Research Group, Workshop on HIP and Related Architectures, Washington DC, November 2004.
- [31] STIEMERLING, M., QUITTEK, J., AND EGGERT, L. Middlebox Traversal Issues of Host Identity Protocol (HIP). Internet-Draft, IETF, February 2005.
- [32] TSCHOFENIG, H., NAGARAJA, A., SHANMUGAM, M., YLITALO, J., AND GURTOV, A. Traversing Middleboxes with Host Identity Protocol. ACISP'05, July 2005.
- [33] TSCHOFENIG, H., NAGARAJA, A., AND TORVINEN, V. HIP Middlebox Traversal. IRTF Host Identity Protocol (HIP) Research Group, Workshop on HIP and Related Architectures, Washington DC, November 2004.
- [34] TSCHOFENIG, H., NAGARAJAN, A., TORVINEN, V., YLITALO, J., AND GRIMMINGER, J. NAT and Firewall Traversal for HIP. Internet-Draft, IETF, February 2005.
- [35] TSCHOFENIG, H., TORVINEN, V., AND ERONEN, P. Advanced HIP-based Firewall Traversal. IRTF Host Identity Protocol (HIP) Research Group, Workshop on HIP and Related Architectures, Washington DC, November 2004.

- [36] VAN ROOIJ, G. Real Stateful TCP Packet Filtering in Ipfilter. 2nd International SANE Conference, Maastricht, The Netherlands, May 2000.