AALTO UNIVERSITY

School of Electrical Engineering
Department of Communications and Networking

Maryam Pahlevan

Signaling and Policy Enforcement for Co-operative Firewalls

Master's Thesis submitted in partial fulfillment of the degree of Master of Science in Technology

Espoo, 26th February 2013

Supervisor: Prof. Raimo Kantola, Aalto University

AALTO UNIVERSITY **ABSTRACT OF THE MASTER'S THESIS**

Author:	Maryam Pahlevan
Name of the Thesis:	Signaling and Policy Enforcement for Co-operative Firewalls
Date:	26th February 2013 Number of pages:X+119
Department:	Department of Communications and Networking
Professorship:	S-38
Supervisor:	Prof. Raimo Kantola, Aalto University
Instructor:	D. Sc. (Tech.) Nicklas Beijar, Aalto University

The Internet environment has been changing dramatically during the recent years. Plenty of new requirements and problems such as the IPv4 address shortage, traversing middle boxes and mobility that cannot be solved with the current Internet architecture have emerged. Among those problems, the IPv4 address exhaustion is recognized as one of the key challenges in the original Internet design. The reason is the unprecedented growth in the number of user devices and regular network nodes which are addressed with IPv4 locators.

Several solutions including Network Address Translator (NAT) and IPv6 (a new version of IP) have been proposed to alleviate the scalability problem. NAT postpones the depletion of the IP address space by reusing the IPv4 address space and isolating customer networks from the public network; it makes hosts in the private address space unreachable from the public Internet. To be exact, a NAT, like a strict firewall, filters out all inbound data flows while outbound traffic from private hosts can go through it. The proposed NAT traversal techniques by IETF have many disadvantages.

The Customer Edge Switching aims to replace NAT with a new device called co-operative firewall and eliminate the problems of existing NAT traversal solutions. A CES device provides global connectivity over the Internet using different types of identifiers, global unique domain names and private addresses.

In this thesis, Customer Edge Traversal Protocol (CETP) is introduced as an edge-to-edge protocol and prototyped so as to tunnel data and control information while transporting the source and destination IDs from one customer network to another. This work also includes the policy management of co-operative firewalls. The testing results assure the suitability of CETP for inter-CES communication and edge-to-edge signaling.

Keywords: CES, NAT, NAT traversal, CETP, tunneling, Address exhaustion, Firewall

Language: English

Acknowledgments

I would like to thank my supervisor, Professor Raimo Kantola, for his valuable advice and helpful comments throughout my thesis work. His broad knowledge on my research work gives me opportunity to explore different aspects of computer networks.

I also appreciate my instructor, Nicklas Beijar. I found him a wonderful person, and I can't express how much I've taken away from working with him.

I want to extend an additional thanks to my friends for their support and encouragement. They helped me to have a balance in my life and kept me happy while doing my thesis.

I am extremely grateful to my siblings. Their successful education backgrounds always inspire me to pursue my education diligently.

Finally, I would like to thank my parents for their love, endless support and encouragement during my Master degree. They always keep me motivated for achieving my goals especially in terms of education.

Maryam Pahlevan

26th February 2013

Table of Contents

LIST OF ACYRYNOMS

ACL	Access Control List
ALG	Application Level Gateway
BEC	Best Effort Communications
CA	Certificate Authority
CES	Customer Edge Switch
CETP	Customer Edge Traversal Protocol
CSM	Connection State Machine
CuN	Customer Network
DDoS	Distributed Denial-of-Service
DNS	Domain Name System
DoS	Denial-of-service
DS	Directory Service
FSM	Finite State Machine
FTP	File Transfer Protocol
FQDN	Fully Qualified Domain Name
GTO	Global Trust Operator
HRL	Host Register List
ICMP	Internet Control Management Protocol
ID	Identity
IM	Instant Messaging
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service provider
ITF	Internet Trust Framework
LAN	Local Area Network
MD	Message Digest
MITM	Man In The Middle
MOC	Mobile Operator Certificate
MTU	Maximum Transmission Unit

NAT	Network Address Translator
NAPT	Network Address / Port Translator
NAPTR	Naming Authority Pointer
PAC	Packet Access Control
PN	private node
PRGW	Private Realm Gateway
RLOC	Routing Locator
RPF	Reverse Path Forwarding
RR	Resource Record
RRP	Return Routability Procedure
RTT	Round-trip time
SCTP	Stream Control Transmission Protocol
SIP	Session Initiation Protocol
SPN	Service Provider Network
STUN	Session Traversal Utilities for NAT
TTL	Time to Live
TURN	Traversal Using Relay NAT
UNSAF	Unilateral Self Address Fixing architecture
URI	Uniform Resource Identifier

LIST OF FIGURES

LIST OF TABLES

1 Introduction

As a result of the large scale deployment and success of the Internet over the last decades, the original Internet architecture has been stretched to the limit. In the future Internet architecture and protocols, several new problems and requirements that cannot be solved by the current Internet must be taken into account. The Internet Protocol version 4 (IPv4) address space exhaustion, traversing middle boxes such as Network Address Translators (NATs) and firewalls, unwanted traffic, mobility and multi-homing are some of such problems.

Since IPv4 addresses are required to address user devices and intermediate network elements in the global Internet, IPv4 address extinction is seen as one of the most important problems in the current Internet. There have been many proposals and solutions to alleviate scalability problems that exist in the original Internet design. Among all these proposals, NAT and Internet Protocol version 6 (IPv6) are gaining more attention and are deployed widely in practice. A NAT extends the IPv4 addressing lifetime by hiding local routing information of the customer networks from the core network and reusing IPv4 addresses, but makes it difficult to initiate communication with the hosts in the private network from the public network.

To solve the NAT traversal problem, IETF has proposed Unilateral Self Address Fixing architecture (UNSAF) [5] that operates by the help of costly adds-on servers, modifying user devices, cluttering application code with application specific NAT traversal code and adding keep-alive signaling. The keep-alive mechanisms and unsolicited traffic consume network resources rather fast and particularly for battery powered mobile devices cause additional drain to the batteries.

The Customer Edge Switching [23] has been proposed to replace NAT with a device called Customer Edge Switch (CES). A CES device resides at the edge of the customer network and provides extensions to the functionalities of a NAT. The main difference is that CES unlike NAT makes hosts and servers in the private networks globally reachable without using keep-alive signaling or even non-scalable and costly servers. In addition, CES in a similar way to firewalls can make a decision whether the incoming/outgoing packet is legitimate or not. Conceptually, a CES is a cooperative firewall, it allows the sender's network and the receiver's network to cooperate to counter the actions of the hosts that have been infected by viruses and Trojans and that may be joined in a botnet. A co-operative Firewall extends the options for decisions that the firewall can make from

admit/deny to admit/deny/query. Due to this, a CES helps to reduce unwanted traffic that reaches the destination host.

1.1 Research Problem

This master thesis is aimed at verifying and refining the specification of the Customer Edge Traversal Protocol through prototype implementation and testing. The starting point of the work was an outline protocol definition created by professor Kantola on a set of slides.

The CETP is an edge-to-edge tunneling protocol. The purpose of CETP is to tunnel data packets and control information while carrying chosen types of identities from one customer network to another. The protocol operates between two communicating CES devices which require desired types of identities, globally unique domain names and private addresses to provide global connectivity over the public Internet. The CETP gives the private network various tools like return routability check on naming or forwarding level to make an informed admission decision on incoming/outgoing packets and consequently enhances trust between two customer networks and two endpoints and reduces unwanted traffic that reaches a destination host.

CETP makes the inbound CES responsible for detecting and eliminating source address spoofing thus making it reasonable for the inbound node also to collect evidence of misbehavior of the sender and attribute the evidence to the sending host and at least to the customer network that serves the sender. Furthermore, we believe that CETP with the help of its on-demand routing capability can be used as a means to traverse multi-homed network boundaries. CETP can be modeled to run on top of UDP, over IP as a new protocol or over Ethernet by defining a new Ether-type for CETP.

All aspects of the CETP protocol are policy controlled. In this thesis, a significant effort is presented for policy control implementation and testing in connection with the CETP protocol.

1.2 Objectives

The purpose of this research work is to achieve a stable specification of CETP and further prototype the protocol as an essential component of our new co-operative firewall architecture. In addition, we implemented a policy enforcement module to enforce rules in co-operative firewalls. More specifically, the policy control of CETP is developed to evaluate the possibility of the idea of co-operative firewalls. This research is carried out in

a way that it can easily interoperate with the previous Master's theses that have prototyped the Customer Edge Switching concept, Private Realm Gateway (PRGW)[29] and Session Initiation Protocol (SIP) and File Transfer Protocol (FTP) Application Level Gateways (ALGs) [28].

To assure the correctness of the developed CETP packet structure and the logic of the policy processing algorithms, we tested the prototype with a significant number of testing scenarios. The key idea behind this extensive testing was to check the suitability of the CETP solution for the inter-CES communication while tunneling data packets and control information over the core network and transferring a variety of identities. The functionality of the program was also examined with well-known protocols such as SSH, SFTP and HTTP.

It is important to note that the developed prototype was not evaluated in terms of performance. Instead, we verified the logic of the implemented algorithms thoroughly. In addition, this work is completely transparent to end hosts.

1.3 Scope

This thesis presents a structure of the CETP message that emerged due to the work on this thesis. Additionally, it contains the policy control algorithms which are used for edge-to-edge signaling. However, the policy processing could be carried out in several ways; we only focused on a single model and developed that throughout this work.

The CETP implementation is integrated into the former CES prototype that has been developed as a proof of the CES concept. Therefore, the prototype only contains simplified Customer Edge Switching functionalities. We did not modify the core functions unless it was required in the CETP development process.

Since the purpose of this work was a rough prototyping of CETP and its relevant methods with Python, we did not conduct any performance testing. Moreover, the prototype was not tested with the mobile end hosts and multi-homed networks.

1.4 Structure

This thesis work is described in eight chapters. The basics of the network security are discussed in Chapter 2. Besides, some of the network attacks and corresponding counter measurements are explained in the subsections. Chapter 3 briefly explains some basic concepts of the Internet such as DNS, NAT and fragmentation. This chapter examines the

NAT concept in more depth and lists the issues that have emerged as a result of using NATs at the edge of the private networks. A Customer Edge Switching solution which has been proposed to obviate the NAT reachability problem is described in the fourth chapter. Understanding this chapter is essential to follow the next chapters. In the fifth chapter, the key ideas of developing CETP as an edge-to-edge tunneling protocol are presented in details. The packet structure of the CETP and the story behind each field in the CETP message are also discussed in this chapter. The sixth chapter is devoted to describing how the experimental environment is set up and how the networking elements involved in the prototype network are configured. Additionally, the algorithms enabling the policy processing for the CETP protocol are explained extensively in Chapter 6. Chapter 7 goes through some special cases of the testing scenarios and then evaluates the results that were obtained from the entire test cases. The final chapter provides the conclusion of the thesis.

2 The Basics of Network Security

This chapter discusses the different aspects of the network security and why it has become more and more important in recent years. The most contemporary network vulnerabilities and attacks will be listed and explained briefly. The end of this chapter is focused on cryptographic mechanisms and tools that are extensively used in Internet protocols as strong counter measurements to fraud and unauthorized activities.

2.1 Network Security

Over the recent decades, the Internet has evolved significantly and the number of the users who use the Internet for fulfilling their needs has increased exponentially. At the same time more attention is turned to malicious actions and unlawful access to the network's and the user's resources. The network attacks are usually targeted to capture or manipulate personal data or acquire control of the victim's system. By granting authority to tools with the malicious codes hidden inside them, insecure administration systems, vulnerable security systems, buggy operating systems and installed software provides numerous opportunities for attackers to initiate illegal activities. The malware like worms, viruses and Trojan horses, spam circulating, social engineering techniques such as phishing, identity theft and stealing personal and critical information are some of the dangerous network security threats.

There are several ways to infect a computer system by malicious codes. For instance, transferring malware from the Internet to the victim's system can happen by downloading a file, clicking on a link that redirects to a fake website or by opening spyware carried in a received email. Additionally, some browsers download malicious codes while surfing an infected website. In this scenario, the user does not do anything to trigger downloading malware from the visited website. [33]

2.1.1 Network Security Architecture

The security framework can be described using layers and planes. The purpose of the security layers is to provide security for the network infrastructure and facilities. Due to the layering design in the security layers, the end-to-end security between different applications can be built regardless of the type of the communicating applications. To achieve end-to-end security, security is split into infrastructure layer, services layer and application layer. Since there are various vulnerabilities on each network layer, each of the security layers strives to fulfill security requirements of the specific layer. For instance, the

Infrastructure layer tries to provide security across network elements such as servers, routers and physical links. The service layer takes care of the security of the services that are given to the Internet users by the Internet Service providers (ISP). In contrast, the application layer is associated with applications that function over the network.

The security planes protect a range of actions that are done over the Internet. The Management plane, the Control plane, and the End-User plane are three security planes which are aimed to meet security requirements of the activities like management activities. The management plane addresses the security of Administration, Maintenance and Provisioning activities. On the other hand, the control plane is responsible for providing the security of the signaling activities including setting up a connection between two communicating endpoints independent of the transmission facilities and network links. The end-user plane concerns the security of data and actions that reside or take place at end systems. The big picture of the network security architecture is shown in Figure 2.1. [50]

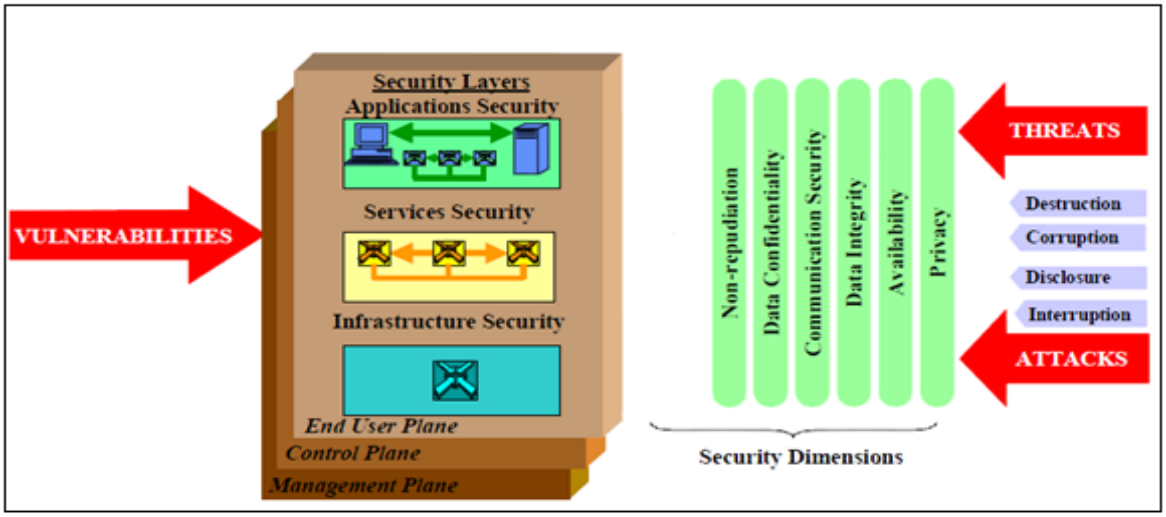

Figure 2.1: Network security architecture and dimensions [revised] (Zhao et al. 2003)

2.1.2 The Basic Components of Network Security

A closer look on Internet security shows that the basic idea behind the security concept is to preserve privacy. Privacy is about controlling the level of the visibility and accessibility of the personal information by its owner. In other words, privacy means that user's identity and his/her actions must be protected. [50]

Data security has four different dimensions: Data Confidentiality, Integrity, Availability and Non-repudiation. To provide data confidentiality, unauthorized access to data has to be hindered through encryption mechanisms, access control lists, and file permissions.

Also, an unauthorized party should not be able to generate, modify, remove, or replicate data. This feature is named data integrity. Data integrity cannot be guaranteed without knowing and verifying the sender of data. [10]

The availability dimension defines the ability to provide uninterrupted access to network resources, services and information storage for the right person. [44]

Non-repudiation is another aspect of the data security. It does not permit users to deny their activities that they did on the data including sending or receiving packets, making or receiving audio or video calls. There are two forms of non-repudiation ability that can be provided by the network. Non-repudiation with the proof of origin of data hampers any attempts by the sender to deny having sent messages. On the other hand, the recipient is unable to deny receiving the data through proof of delivery of data. [50]

2.1.3 Identification, Authentication and Authorization

In order to preserve confidentiality, integrity and availability of computing resources, firstly the user has to identify itself to the destination system, then the system that serves the customers' needs to verify the identity of the sender and eventually the system administrator should grant a specific access right of resources to the user. These three functions are named identification, authentication and authorization respectively.

The identification procedures often do not provide any certificate that can be used to assure the identity of users. The prevalent identification mechanism usually uses a username that consists of a mixture of the letters, numbers and other character's to identify the end system or user.

Authentication comes after the identification function as the next step. There are various authentication mechanisms in which the user is required to validate its identity. To prove user's identity, the user must provide certain entities to the destination system. These elements can be passwords, fingerprints or access cards.

Once the identification and authentication process in a destination system is in place, the system administrator must decide what level of access rights to assign to the user. This procedure of granting rights to the end system is called authorization. [10]

2.1.4 Trust in Networks

Trust is a concept involving different dimensions and disciplines. There is a mutual relationship between the essence of network security and absence of trust. Specifically, if trust is not provided across the networks, the Internet could not be used extensively for e-commerce and other serious services that need secure and trustworthy communication. However, the trust concept can be defined in different ways, in this thesis our focus is on the definition of trust that refers to it as a property of a relationship. This relationship consists of two parties. The first party is the trustor who relies on the legitimacy and rightfulness of the second party called trustee. When the trustor chooses to trust the trustee, it accepts a risk.

The purpose of trust modeling is to simplify connection establishment where the environment is hostile and also communicating end points have contrary aims. This can be achieved by minimizing risk in those situations.

In an act of communication between a sender and a receiver, the receiver is the trustor, while the sender is the trustee. The receiver's risk is realized when the received content turns out to be unwanted.

In the computer science, trust is treated as a logical concept. Consequently it is gained by examining many conditions and applying the respective propositional logic. It can be seen in the form of stateless packet filtering. This filtering function can be enhanced significantly by considering the previous activities of the source entity. It is worthy to mention that security risks cannot be always removed completely, therefore setting up connections must be allowed even in the existence of risks.

Authentication is being used as a strong tool to improve trust in the global network. As authentication is costly, a public network administrator can justify the expenses required for the essential authentication infrastructure only with the value-added services and usage based billing. [49]

2.1.5 Security Vulnerabilities and Threats

When a system is designed, implemented and used, a number of bugs and weak points appear in the system. An attacker can take advantage of the system's security vulnerabilities and break into the system. The security vulnerability concept differs from a security threat, risk and even network-based attack.

Any activity that tries to violate the confidentiality, integrity, or availability of resources is known as a security threat. A security threat can be active or passive. In an active threat like masquerading and denial of service, the attackers change the state of the system while making unauthorized access to the system's resources. In contrast, a passive threat including eavesdropping takes place without any changes in the system's state. [50]

The threats can be split into four categories: disclosure, deception, disruption and usurpation. A disclosure threat happens when an attacker accesses the resources without adequate privileges. Deception means that the system receives wrong information instead of valid data. In the disruption threat, malicious users prevent the system to function regularly. The usurpation is about supervising a destination system partially or completely. [9]

Vulnerabilities and threats in a separate context cannot make any security risk, whereas when a threat combines with security vulnerability, the security of the system encounters a risk. The following equation usually is exploited to explain a security risk. [10]

Risk = Threat × Vulnerability

For example, let us assume a web server attack as a risk. To evaluate this security risk, an overflow flaw in the desired application as a vulnerability is combined with the devastating activities such as an unauthorized access, using sophisticated tools and suitable knowledge about the system. The packet loss, data manipulation and loss of reputation are some of the security risk's results. [50]

2.2 Network-based Attacks

The action that causes a potential violation of the security to happen is called an attack. Lack of appropriate authentication and encryption mechanisms in the TCP/IP protocol stacks provides the basis for a wide range of attacks. Some of the potential attacks are described in the following subsections.

2.2.1 SYN Flooding

The three-way handshake that is used to set up a TCP session, gives a significant number of opportunities to attackers. For a TCP connection establishment, first an initiator sends a SYNchronize packet to the destination host and then the responder replies with the SYN-ACK message. The TCP session is established once the sender returns the ACK packet. The SYN flood that exploits this TCP's weakness to perform an attack can be considered

as an example. In this attack, a malicious user sends many SYN packets to the victim and does not respond to any of the acknowledgement packets. As a consequence, the receiver is overloaded with irrelevant packets and cannot reply to messages that are received from legitimate clients. The SYN cookie protects recipients against the SYN flood attack. In this solution, the receiver, instead of creating state upon the reception of the SYN packet, encrypts the SYN packet and sends it back inside the reply message. This procedure is quite easy to implement and eliminates the SYN flood effectively. This kind of mechanism is supported by the Stream Control Transmission Protocol (SCTP). However, this procedure is not supported by the TCP protocol. [4]

2.2.2 Smurfing

The Internet Control Management Protocol (ICMP) allows end users to send an echo request message in order to discover whether a destination host is alive or not. The echo packets can be sent and replied to using broadcast addresses in some implementations of the protocol.

To perform this attack, echo packets are sent with the victim's address forged as a source address and relayed to the broadcast address targeting all hosts in a subnet. These hosts that participate in the Smurfing attack are known as smurf amplifiers. A smurf amplifier sends back the echo replies to the victim entity and thereby it makes the victim system unresponsive to incoming packets.

Guarding network elements and users against smurfing can be done by excluding the broadcast ping capability from the Internet protocol implementation. [4]

2.2.3 Eavesdropping

In a compromised network, since a large number of connections established over the global network are unsecured and exchanged data flows are not encrypted; sniffing data traffic is a possible attack. Cryptography is proposed as a solid counter measure to snooping. [44]

2.2.4 Data Modification

An attacker may manipulate data which is intercepted on communication facilities. This attack targets the data integrity and in the same way as eavesdropping it can be prevented by encryption. [9]

2.2.5 Distributed Denial-of-Service Attack

The Denial-of-service (DoS) attack is one of the most common network layer attacks. In DoS attacks, an attacker brings down a single destination and does not allow the victim to operate normally. The DoS attacks can be eliminated by locating the hackers and filtering the data flow coming from those IP addresses. [10]

Since protecting systems against a DoS attack from a single IP address is quite straightforward, therefore the more complex form of DoS attack called Distributed Denial-of-Service (DDoS) is developed by attackers to defeat this weakness.

In this attack, the attacker firstly exploits a flaw in one computing system and makes use of it as a DDoS master. Then the master system gains control of a collection of hosts and installs the DDoS attack agents on the compromised machines. The DDoS attack is being launched when the intruder sends a trigger signal to the infected computers and as a result those machines overwhelm the specified destination with a large number of unsolicited packets. [4] The DDoS attack, unlike the DoS attack, is very difficult to defeat. In other words, the real hacker could not be discovered, even though the victim finds the sources of the attack because they are intermediate systems just being compromised to launch the DDoS attack and not the genuine attacker. In most cases, these controlled hosts take part in the DDoS attack unknowingly. [10]

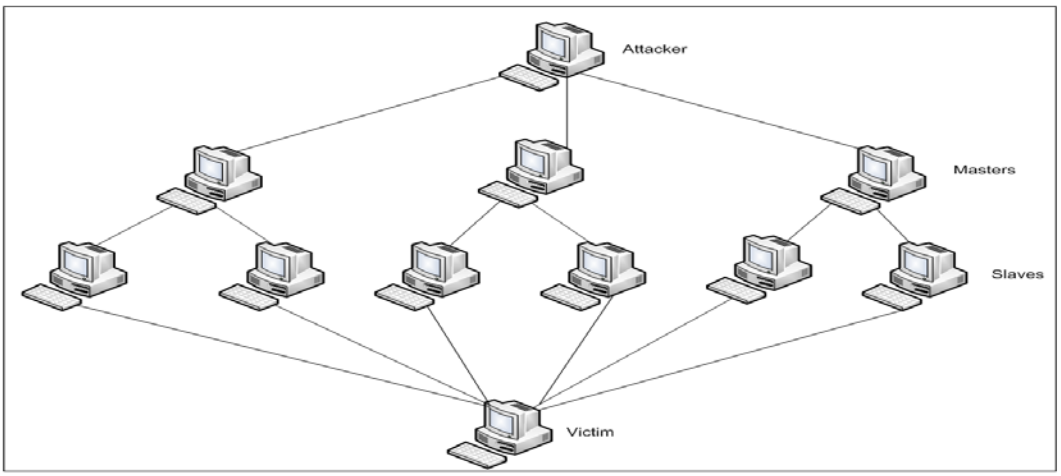

Figure 2.2: Distribute denial of service attack

The ICMP trace back message is being proposed and added to the standard in order to counter the DDoS attack. [7]

2.2.6 Spamming

The intention of a spammer launching a spam attack is fraud or selling something (like Viagra) or attracting the user to a site that will download a virus/Trojan to the victim. To do a spam attack, the spammer forges a new email account and starts sending huge volumes of spam to a list of the legitimate email addresses that can be derived e.g. from unscrupulous websites. [2]

2.2.7 IP Spoofing (Identity Spoofing)

For a long time malicious hosts have exploited various ways to conceal their real identities. IP spoofing is one of the well-known techniques used by hackers to mask their identities. [46] In IP spoofing, an attacker masquerades as a legitimate system by generating packets with fake IP addresses. To be exact, the source address field of the IP header is replaced with an IP address that does not belong to the actual sender. Address spoofing is usually feasible only in UDP, ICMP and SYN. After SYN in TCP it is not possible to spoofed address. [15]

Let us assume there is an active connection (e.g. UDP connection) between two communicating end users, the partner and the victim, as shown in Figure 2.3. Within the ongoing session, a pirate creates an IP packet with the partner's address as a source address and forwards it to the victim system. In the TCP/IP suite, the identities of the sender and the recipient are not verified by the network elements and hosts. Therefore, when the message with a spoofed address is received, the victim assumes that the message was originated from the partner; thereby it accepts the packets and sends back replies to the partner iteratively. In the described scenario, the partner can be the victim and the victim can be the reflector. The spoofing attack can have more destructive effects on the destination system if the reflectors are amplifiers. [9]

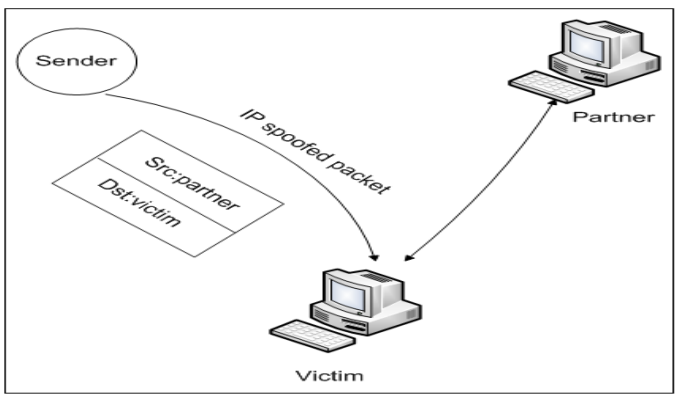

Figure 2.3: IP spoofing

Attackers aid DoS attacks with IP spoofing. It means a single attacker can send many packets with various spoofed IP addresses and thus take the targeted system down. Using random spoofed addresses in forged packets makes eliminating unwanted traffic very hard. Since it seems that the flood of traffic comes from different directions, consequently it is difficult to trace back the packets. [45]

Additionally, IP spoofing could be used in the applications and networks in which hosts are identified and authenticated based on their IP addresses, however, TCP packets cannot be used easily with spoofed source addresses. In these cases, the hacker pretends being a trusted system by using spoofed addresses, thereby gaining an unauthorized access to victim's data. This attack results in monitoring, manipulation and corruption of information. [8]

The IP spoofing is widely used to launch a wide range of attacks. These types of attacks can be divided into blind spoofing, non-blind spoofing and man-in-middle attack.

In the non-blind spoofing, an attacker and a destination system are residing in the same subnet. Consequently capturing the sequence and acknowledgement numbers is not a costly process. The attacker first breaks the ongoing connection down and then re-establishes the session with a correct sequence and acknowledgement number. This attack is known as session hijacking.

On the other hand, sniffing the sequence and acknowledgement numbers is not feasible in blind spoofing. Instead, the attacker sends many packets towards the victim system in order to guess the sequence and acknowledgement numbers precisely. According to the aforementioned explanation, it is obvious that blind spoofing is a more complicated and time-consuming process than the non-blind spoofing attack.

A limited number of countermeasures like ingress filtering and encryption have been proposed to eliminate IP spoofing from the global Internet. This fact implies that it is rather hard for the network elements to detect malicious hosts who exploit spoofed identities or addresses.

Adding the ingress and egress filtering functionality to border routers is a primary action usually taken in the spoofing protection technique. In the filtering mechanism, the inbound interface through the Access Control List (ACL) implementation drops the incoming packets with the private IP addresses. Furthermore, inbound messages with a local IP address as a source address should be filtered out by this interface. The outbound interface

can also help spoofing defense by specifying a range of the legitimate private IP addresses accurately. [45]

2.2.8 Man in the Middle Attack

The Man in the Middle (MITM) attack can be seen as active eavesdropping. It encompasses different types of attacks including two forms of IP spoofing which were described earlier. To launch a man in the middle attack, a hacker sits between two communicating end points and tries to gain control of the data traffic. The attacker often manipulates or corrupts the captured packets and then forwards them to the targeted victim. In the extreme cases, the malicious user establishes an independent connection with the victim systems and impersonates either the recipient or the initiator of the conversation. This makes the attacker able to access the victims' secret information while both sides of the communication believe that they are conversing with a trusted host. [45]

Mutual authentication is known as a main solution for preventing MITM attacks. For instance, the communicating parties could be authenticated and verified through a trusted certificate authority. [11]

2.3 Firewalls

A firewall is the most common and effective way to protect the internal network from the public Internet and hence provides a level of security for all outbound and inbound connections. By separating the trusted local network from the outside world, a firewall limits the impact of the security problems of the Internet on the internal network. Most often, a firewall stands between a corporate network and the global network. The existing firewalls in the market fall into two categories: The host based and network based firewalls.

Since all incoming and outgoing data traffic passes through a single access point where the firewall is residing, the firewall is capable of tracking what exactly is taking place in the network. Commonly, a firewall makes a decision whether a packet that is either sent to or received from the Internet is legitimate or not. It is more convenient to apply the security policies in one entity (i.e. firewall) rather than distributing the security decisions over a large number of network elements.

Each packet can be analyzed and filtered based on the different layers in the TCP/IP suite. According to this criterion, there are three types of firewalls: Packet filtering, circuit gateways, and application gateways. [53]

The packet filters usually incorporate with routers which interconnect various networks and thus deliver packets to particular destinations. They filter out packets according to the IP addresses and/or port numbers. Mostly, the packet filters are stateless. It means that they do not keep any record of the processed packets. Filtering on either the incoming or the outgoing interfaces being performed using a list of eligible hosts and services and a blacklist of the malicious users.

The spoofing attack can be prevented by packet filters. To accomplish this, they inspect inbound packets to block those malformed packets that use the private or loopback addresses as source addresses. This functionality is named *egress filtering*. In addition, outgoing packets must have the valid source addresses; otherwise they will be dropped by *ingress filtering*.

The complexity of filtering remarkably depends on the size of the internal network's address space. Moreover, the packet filtering is not accurate enough to apply a large portion of the security rules. [12]

Circuit gateways are another form of firewalls in which the packets are filtered at the transport layer of the Internet protocol stack. They are more prohibitive than packet filters and add many useful services to existing firewalls like encrypting data flows between two firewalls.

The circuit gateways cannot guard systems against the application level attacks. Hence the application gateway functionality in firewalls is proposed to fulfill this need. They serve one or more applications and apply the security policies that need finer control granularity than what packet filters and circuit gateways are able to do. Although, application gateways incredibly reduce the security risks over the network, they add a significant delay to the response time of the services residing behind firewalls. [4]

2.4 Cookie

In recent years, the DoS attacks are known as the most common security risks in that attackers bombard a targeted victim with connection initiation queries and thus deplete the victim's resources including memory and CPU cycles. To limit this attack and its destructive effects, Cookies can be used. The recipient does not create any state upon reception of the session initiation query until it ensures that the source address of the message is genuine. [30]

In this procedure, the recipient replies to the initiation request with a block of text called a cookie, which is used in order to figure out whether an initiator of the session uses a forged IP address. If the initiator resides at the claimed address, it can resend the session initiation request with the received cookie. After the two first messages, the responder just accepts those session initiation requests that carry the same cookie; otherwise it discards the packets originated from a malicious user.

Since in the described scenario, only the responder is involved in the cookie generation and examination process, the initiator does not need to be aware of what exactly is happening in the cookie generation algorithm and the algorithm can be defined locally. Due to this feature, the different implementations of protocols and standards use various procedures for generating a cookie. The only common principle among all these algorithms is that the cookie must be recoverable by the responder based on the received packet not using any saved state. [26]

2.5 Cryptography

Encryption is usually exploited to provide data confidentiality and integrity, whereas it does not help with the availability of data. There are various cryptographic tools that use different techniques and protocols for encrypting data, exchanging key information securely and authenticating the origin of data. The strength of a cryptographic system is measured based on its complexity. It is clear that a reasonable cryptographic block must be complex enough, so that launching an attack becomes more expensive than what the attacker is willing to pay. [50]

In the encryption process, a plain text is transformed into humanly unreadable text which is known as a ciphertext. The original text can be derived from the ciphertext with a reverse conversion named decryption. However, as the encryption algorithms are public, keys that are the fundamental inputs for these algorithms have to be secret. Figure 2.4 shows a general data encryption process.

$$Ciphertext \quad C = E_k(P) \quad (1) \;\; Encryption\ Process$$
$$Plaintext \quad P = D_k\big(E_k(P)\big) \; (2) \;\; Decryption\ Process$$

Figure 2.4: Data encryption terminology, E_k: encryption key, D_k: decryption key

A wide variety of encryption techniques are being used in order to encrypt cleartext into ciphertext. Among these techniques, substitution is the easiest one in which each letter of the cleartext is being replaced with the corresponding character in the ciphertext alphabet.

Transposition is another way to encipher plaintexts. In this approach unlike substitution, the order of characters in the cleartext is changed according to the value of the encryption key. For instance, if 4 is chosen as an encryption key, firstly the original text must be split into the fixed-sized segments (i.e. 4 characters). Then the first character of each segment is fit into the beginning of the ciphertext. The second letters in each part come after them. This pattern applied over and over until the ciphertext is built completely. [20]

Generally, there are two forms of encryption algorithms. The most popular one is *symmetric cipher*. The symmetric cipher uses the same key for encrypting cleartext and decrypting ciphertext. In contrast, some encryption algorithms are *asymmetric* and hence they have separate keys for encipher and decipher messages.

2.5.1 Symmetric Cipher

In a symmetric cipher, a key called shared secret is used for both the encryption and decryption function. The symmetric ciphers are categorized into two groups: block ciphers and stream ciphers. A block cipher tool firstly divides data into fixed and equal sized blocks and then enciphers each block of data. The encryption process of each block is performed atomically. If the length of data is not dividable by the block size of a block cipher algorithm, the input needs to be padded. In different block cipher algorithms like AES [18], CAST [1] and Blowfish [42] the block size varies.

A stream cipher unlike a block cipher operates on the data bit by bit. To encipher data in the stream cipher, initially a stream of bits is produced by using a key and then the generated stream is XORed with the input. The encryptor and decryptor of the stream cipher tool must be synchronized and they must use the same bit stream in the ciphertext to recover the corresponding bits in the cleartext. As a consequence, if these two functions get out of synchronization, the original input cannot be derived from the ciphertext. This synchronization problem makes stream ciphers such as RC4 [39] less popular than a block cipher. [16]

2.5.2 Asymmetric Cipher

Asymmetric cipher which is also called *public-key cryptography* exploits a pair of keys to encrypt and decrypt messages exchanged over insecure communication links. In this encryption system, a plaintext is enciphered by one of the two keys that is named public key and can be deciphered only by the other one which is called the private key. The private keys are produced internally and never distributed. Moreover, they are not

derivable from the corresponding public keys. Therefore, the public keys can be transmitted in the plaintext to any end systems who request encrypted communication. [20]

The application of a public-key cryptosystem can be divided into three different groups: encryption/decryption, digital signature and key exchange. The encryption procedure is implemented with the help of the public key described above. To generate a digital signature, the sender uses its private key in order to sign the whole message or a shorter form of it. Both communicating parties are able to swap key materials through their private keys.

However, public-key cryptosystems use a wide range of cryptographic algorithms, but all of them conform to similar principles. From a computational point of view, generating a pair of keys via public-key algorithms must be facile. Besides in these methods data encryption and decryption have to be computationally easy. For message authentication, it must be impossible to derive a cleartext from a ciphertext only by knowing the public key. [44]

2.5.3 The RSA Encryption Algorithm

Since in the symmetric cipher, the session key exchange has been problematic and secret keys need to be distributed securely, therefore a public-key system is proposed as an alternative solution and widely implemented.

The RSA [16] algorithm has been recognized as the most widely deployed public-key encryption since 1977. This algorithm was designed and developed by Rivest, Shamir and Adelman. To generate a pair of keys in the RSA scheme, firstly two large positive prime numbers, p and q, are chosen and their multiplication (i.e. n) is computed. After that Euler totient of n which equals to the product of $(p - 1)$ and $(q - 1)$ is calculated. As the next step, an integer e that is prime to the Euler totient of n is selected. Eventually, the integer d is computed based on the following formula.

$$d * e \; mod \; q(n) = 1 \tag{1}$$

In this public-key algorithm, the public key is

$$\mathrm{KU} = \{\mathrm{e}, \mathrm{n}\} \tag{2}$$

and the private key is

$$KR = \{d, n\} \tag{3}$$

Consequently, only the recipient who decrypts the ciphertext must be aware of d's value. On the other hand, *e* and *n* are available for both sides of the communication. To encrypt a message M, a sender calculates

$$\mathrm{C} = M^e \bmod \mathrm{n} \qquad (4)$$

and furthers sends C to the desired target. On the opposite side, the receiver deciphers C by computing

$$\mathrm{M} = C^d \bmod \mathrm{n} \qquad (5). [44]$$

2.5.4 Certificate Authority

The fundamental advantage of public-key cryptography is that public keys can be broadcasted to any end systems in a large scale. Although, this feature seems very beneficial, it has a serious defect. As there is no explicit authentication mechanism in the public key exchange, it is feasible for a malicious user to pretend to be a trusted party and send forged public keys to the targeted victim. After that, the attacker is able to decipher all encrypted messages sent by the victim and thus gain an unauthorized access to the victim's confidential information.

To defeat this weakness, the public-key certificate is considered as an appropriate proposal. Typically, a public key certificate ties a public key to the identity of the key owner and digital signature. In fact, the signature is the whole certificate block that is signed digitally using the private key of a trusted third party. Commonly, the trusted third party is referred as a *Certificated Authority (CA).* A participant announces its public key to a CA over secure channels and then acquires a certificate. Since then, the participant can advertise the certificate instead of the public key. The integrity of the certificate can be verified through examining the appended signature. [44]

2.5.5 One-way Hash Function

In hash functions, an arbitrary-size message is hashed into a fixed number of bits which is called a *Message Digest* (MD). A hash function is known as a one-way function, if the original block of data cannot be recovered only by knowing the corresponding hash value. Furthermore, in terms of computation, it should be infeasible to find two different integers ($x \neq y$) so that they are hashed to the same MD by the hash function. This characteristic is referred to as collision resistance. SHA-1 [38] and MD5 [21] are two examples of one way hash functions that are widely used in cryptosystems. [44]

2.5.6 Signature

Public-key systems can in addition to encryption be used to provide non-repudiation. To achieve this, a sender encrypts a block of data with its private key and then on the recipient side, the encrypted message is deciphered with the sender's public key. The ciphertext obtained in this scheme is known as a *digital signature*.

As only the sender accesses its own private key, no one else could encipher a message in such a way that it could only be decrypted with the sender's public key. In other words, attackers are unable to modify messages without knowing the initiator's private key. Therefore, the integrity of data and the origin of data can be preserved through digital signature. [20]

It is obvious that in terms of processing, encrypting the whole message is relatively expensive. Consequently, for signing a message, a hash function is exploited to compute a shorter version of the message (i.e. the MD) and further the MD instead of the entire message is encrypted with the private key. In this approach, the whole message except the signature is sent as clear text, thereby data confidentiality is not provided and eavesdropping cannot be eliminated. [44]

2.6 Return Routability Check

The main objective of the *Return Routability Procedure* (RRP) is to verify that the sender could receive packets at a claimed address. More specifically, the responder sends a message towards the address that is claimed to be the source of the communication. If the desired reply is sent back by the initiator, the validity of an address would be proven.

This approach does not require any form of authentication such as the public-key infrastructure and in terms of timing, commonly can be performed within one round-trip time. Given this, the RRP only provides a basic level of integrity and authenticity and does not establish any secure associations between communicating parties. Moreover, the return routability check reduces the chances of spoofing and DoS attacks significantly. RRP is an efficient method with the assumption that the attacker has not compromised the routing system in any of networks on the way from the sender to the receiver and rather has access only to the hosts. [21]

3 Some Basics of the Internet

This chapter is devoted to describing some basic concepts of the Internet including NAT, Domain Name System (DNS) and fragmentation. The NAT subsections briefly explain why the NAT schemes have been developed and how NAT devices work. The different NAT traversal methods are also listed and described in this section. The concept of DNS and motivations behind its existence are discussed in the DNS sections. At the end of the chapter, the essence of fragmentation, the procedure invoked when the size of packets exceeds Maximum Transmission Unit (MTU) of the communication links, is explicated.

3.1 NAT

The essence of IP address translation emerges when routing information, specifically IP addresses, within a private address realm cannot be advertised in the external networks because the same addresses are used in several private realms simultaneously and thereby are not valid globally. The Network Address Translation scheme is aimed to interconnect disparate address realms transparently. For this purpose, the NAT devices modify the packet headers when they are sent from a private address space to the public realm or vice versa.

Primarily, NAT was deployed to reuse the IPv4 address space and thus postpone IPv4 address exhaustion. In this approach, to extend the IPv4 addressing lifetime, one or a few addresses that are reserved for the Local Area Network (LAN) are exploited to communicate with the outside world. The hosts residing in the private network, behind the NAT device are addressed with private IP addresses which are not unique across the Internet.

NAT works well with the client-server model where end hosts behind a NAT make requests to arbitrary severs in the public network. To cite an instance, let us assume a scenario where a host in a private realm intends to initiate a connection with a specific server. The packet, on the path to the destination, firstly goes through the NAT device. The NAT device modifies the source address of the packet field from the host’s private IP address to the NAT’s public IP address. This mapping is written into the NAT’s mapping table and kept alive for a limited period of time. Next, the NAT device forwards the modified packets to the destination server. In the reverse direction, the server sends replies to NAT’s public address. If a relevant mapping still exists in the mapping table, the NAT device modifies the address fields of the packets correspondingly and delivers them to the end host.

However, there are a wide variety of NAT devices; the traditional NAT is the most commonly deployed type of NAT device. The traditional NAT is divided into two different categories: 1) the basic NAT, and 2) the Network Address / Port Translator (NAPT). In the basic NAT, only the IP address is mapped while the NAPT translates both IP addresses and transport identifiers. A NAT device regardless of its type performs the address binding statically with a fixed address assignment or dynamically at a session initiation.

3.2 Basic NAT

The basic NAT device assigns a certain range of IP addresses to the local hosts in the private address realm when they relay packets to another realm. To be exact, as a host in the private domain sends packets to a server in the external domain, the basic NAT which sits on the border of these two address realms allocates one of the public addresses for the host and changes the source IP address to this value. In the address binding phase, all related fields in the packet headers such as checksum that are dependent on the source address field must be updated correspondingly.

3.3 NAPT

NAPT permits numerous private hosts to be multiplexed into a single external IP address at the same time while each session is established with different transport identifiers. To do this, NAPT in addition to IP address translation, translates transport identifiers (TCP/UDP port numbers, ICMP query ID) as well. In other words, as a local endpoint originates packets to an external host, the NAPT maps the internal IP address and the internal transport identifier to the public IP address and the public transport identifier. The inverse translation is being done for the inbound packets. [43]

3.4 NAT Address Assignment Behavior

For knowing NAT's mapping behavior it is necessary to describe how one or more public IP addresses are reused for messages that are originated from a set of internal users and are destined to the external hosts. More formally, various mapping practices follow several techniques to execute the translation between the pair of private IP address and the private transport identifier and the pair of external IP address and the external transport identifier. The NAT's translation principles can be explained as follows.

Endpoint-Independent Mapping: The NAT only keeps one entry in the mapping table for all packets originating from the internal host with the same private IP address and transport identifier. These packets further can be routed to any external endpoints. It means

that as long as traffic is coming from the same local IP address and port, the corresponding mapping does not alter.

Address Dependent Mapping: There is a single mapping in the NAT table for packets being relayed from the same private IP address and transport identifier to the same external IP address. The transport identifier of the external destination does not play any role in this mapping.

Address and Port-Dependent Mapping: The NAT maintains different mappings for packets being sent from the same private IP address and port to the same external IP address but distinct ports. More specifically, as soon as the host with a private IP address and port starts conversing with a second public application that either has a different IP address or port, the NAT creates a separate mapping for this newly established connection. [6]

3.5 NAT Traversal Issues

The NAT must manipulate the packet headers at layer 3 and 4 in order to transmit packets from one realm to another. Furthermore, an inbound packet can be delivered to the private end system only if its addressing information is matched with one of the NAT table's states. Otherwise, it will be dropped by the NAT because it does not have a mapping in the NAT table and thereby it is infeasible for the NAT to find a relevant internal destination. The NAT traversal issues that are closely related to trust concept are fitted into four classes.

The first issue appears when some protocols (e.g. SIP) include the IP addresses within their payload. Since the conventional NAT does not function above layer 4, it fails in a situation where a sender resides in the corporate network and uses a locally significant IP address in outbound packets' payload. To overcome this, an ALG, which provides protocol specific processing, is proposed as a solution. However, the ALG fails when encryption is used. Moreover, for every application protocol, the separate functionality must be implemented in the ALG.

The second area where NAT devices cause difficulties is peer-to-peer applications. In the traditional Internet, the predominant communication paradigm was the client-server scheme. Consequently, the network address translation scheme was also designed with the assumption of asymmetric connection establishment in mind. For this reason, the NAT only operates where an internal host in the private realm initiates a connection with a

server residing in the public Internet. In contrast to client-server applications, peer-to-peer applications require bi-directional connectivity. In other words, in a peer-to peer application, the connection can be established by any of the parties and hence there is no difference between the communicating peers. This results is a problem in the NAT functionality particularly when either one of the hosts or both are located behind NATs. To give an example, let us assume that a host in the private network tries to provide a specific service for a set of clients. In this scenario, the host is unable to fulfill its clients' needs because there is no mapping for the incoming packets and thus they get dropped by the NAT.

The two first categories described above combine and form the third group. The Bundled Session Applications (e.g. FTP, SIP/SDP) uses realm-specific addresses within their payload fields for the additional connection establishment. In these applications, the first connection is known as a control plane, while the connection established subsequently is called data plane. Carrying realm-specific IP addresses in the payload is not the only problem here; rather the issue is the data session when it is initiated from the public realm to the private realm.

The last category is related to the unsupported protocols such as the Stream Control Transmission Protocol (SCTP) that have been developed recently. The NAT could not support these protocols even if a private host establishes the session with an external host. This is because the NAT does not have translation functionalities for these protocols. This group also encompasses protocols such as the encryption protocols in which the NAT cannot access layer 3 or layer 4 headers. [37]

3.6 NAT Traversal Techniques

In this section, some of the most deployed techniques for traversing NAT devices are explained briefly.

3.6.1 Session Traversal Utilities for NAT

Session Traversal Utilities (STUN) for NAT is a protocol that is used extensively by other protocols in the context of NAT traversal. A host through this protocol can learn the allocated IP address and port number by the NAT. Moreover, it can be used either for examining the connectivity between two hosts or keeping mappings in the NAT table active for a certain period of time. The STUN operates with a wide variety of NAT devices without making any changes in their operation. [41]

The STUN is a client-server protocol. As such, it needs a STUN server in the public Internet and a STUN specific application installed on all client end systems. Figure 3.1 demonstrates the message flow in the STUN system.

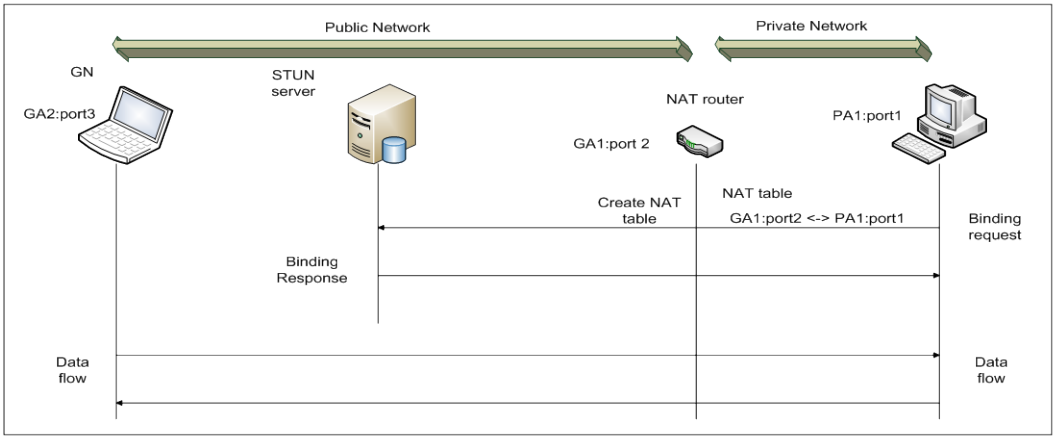

Figure 3.1: Message flow in STUN [revised] (Virtanen et al. 2009)

To determine the NAT binding, a private node (PN) that is located behind the NAT sends a binding request to a STUN server. The binding request goes through one or more NAT devices before it is forwarded to the STUN server. Therefore, the NAT will change the source address field of the packet and further add the respective mapping to its table. As a consequence, what exactly the server sees as a source transport address in the binding request is the public IP address and the port allocated for PN (i.e. GA1:port2) by the nearest NAT. This addressing information is named a reflexive transport address. The server sends back a binding response that carries that reflexive transport address within its payload towards the correspondent PN. After that, PN can advertise its public transport address across the public Internet and hence communicate with any external hosts. Note however, that depending on the type of NAT, it may be necessary to repeat this procedure for each pair of an external communications partner and application. [47]

3.6.2 Traversal Using Relay NAT

Typically, the “hole punching” technique is used to find a direct communication path between two hosts where a host behind a NAT tries to initiate a session with a preferred peer that might be located behind another NAT. This technique will fail if the hosts involved are behind NATs that have a special mapping behavior such as "address-dependent mapping" or "address- and port-dependent mapping".

The Traversal Using Relay NAT (TURN) protocol has been proposed to complement the limitation of the STUN system for which the direct communication path between two hosts cannot be discovered without intervention of any relay servers. The TURN system uses an additional network element called relay server to relay messages between two end systems. A relay server commonly resides in the public Internet and needs high-bandwidth connection to the Internet.

The TURN similarly to the STUN is a client-server protocol. It allows a client in the private realm behind one or more NAT devices to request a public transport address from a TURN server and thereby receive TCP or UDP sessions at the allocated address.

To begin the TURN operation, a TURN client must have the address of a TURN server. The address can be discovered by using for example the SRV records or from a preconfiguration file. As the next step, the client sends a TURN allocate request to the TURN server. The TURN system exploits a digest challenge mechanism to perform the authentication and integrity checks for both the allocate requests and the replies. As soon as the allocate request has been authenticated, the TURN server gives back an allocated address inside the allocate response to the client. The TURN client cannot receive data from a peer until it sends a send request message which contains data to the TURN server. The server forwards the data derived from the packet to the peer specified by the destination field. In the opposite direction, the data received at the allocated address from the peer is encapsulated into a Data Indication message and relayed back to the client. The simplified TURN operation is represented in Figure 3.2.

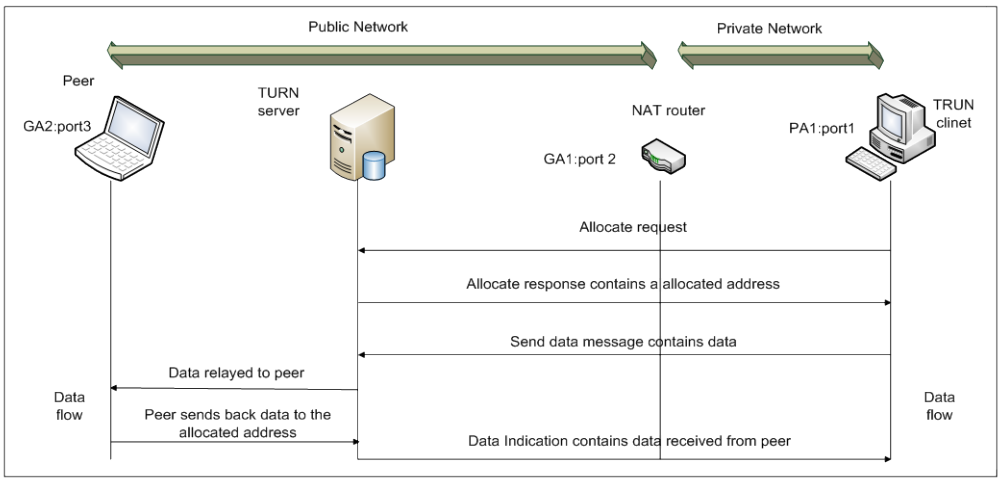

Figure 3.2: Basic message flow in TRUN [revised] (Virtanen et al. 2009)

If the TURN client wishes to establish connections to more than one peer, it must send a send request message to every peer separately. Besides, the allocated transport address can be published to any hosts that wish to communicate with the client. [47]

3.6.3 Interactive Connection Establishment

Interactive Connection Establishment (ICE) allows Bundled Session Applications to traverse any type of NAT device. More specifically, ICE enables peers to collect the adequate information about their topologies and thereby discover their potential communication paths by using the STUN and TURN techniques. Each of these paths is named a candidate. In addition, ICE is one of the most comprehensive and optimum solution among the proposals which have been presented for the NAT traversal problems, because it works even under very complicated topologies and makes use of relay only if it is needed.

The main objective of ICE is to discover which pairs of candidates can be used for communication by the agents. To do that, at first the agent must collect its candidates. The candidates can be separated into three categories. The first category is called host candidate and obtained from the local interfaces (e.g. Ethernet). The second one is server-reflexive candidate and the agent gathers them by sending request messages to the STUN server. The last category is relayed candidates and they are gathered from the TURN servers. In some cases, the TURN server provides both relayed and server-reflexive candidates of the agent simultaneously.

As the agent gathers and prioritizes its candidates based on a certain criteria, it sends an INVITE request message to the peer agent. This message carries the agent's candidates gathered before. Upon the reception of this message, the destination agent gathers and prioritizes its candidates in the same way. In the next step, it replies to the INVITE message with the message that contains its own candidates.

At this point, each agent knows the peer and its candidates. Therefore, each agent makes different combinations from each of its candidates and each of the candidates from the peer. Then, the STUN messages are relayed from one agent to another one in order to check the connectivity of each pair of candidates separately. A data session can be established between the agents once a point of communication is discovered. [40]

The disadvantage of ICE is that due to the erratic nature of NATs and packet loss, it uses up to 100 messages to select the best candidate pair for one connection. When an

application has 2 connections, the maximum is 200 messages. As a consequence, the maximum session setup delay grows to tens of seconds. [31]

3.6.4 ALG

ALGs are the protocol specific translation components that collaborate with NAT routers in order to connect hosts in disparate address realms. An ALG may interact with the NAT for several reasons. For instance, it modifies the application specific payload by using the NAT state information and thereby allows them to traverse NAT transparently.

In some cases, ALGs do not exploit NAT state information. Instead, they may evaluate and examine the application payload and further ask the NAT to add an extra mapping to its table. However, ALGs similarly to proxies provide the application specific processing especially for session-oriented applications, but unlike proxies they do not require any changes in application clients and also do not interact with clients by using a special protocol. [43]

3.7 DNS Overview

This section gives an overall picture of the DNS system which is extensively used across the Internet.

3.7.1 The Domain Name Space

DNS is a distributed and hierarchical database which was originally designed to provide a name resolution service across the Internet. Mainly, it is used by end users and various Internet services to translate host names (i.e. human readable domain names) into numerical IP addresses used to identify hosts. [3] The main component in the DNS structure is the name space. DNS name space is a hierarchical tree structure that is constructed from several nodes. Each node is associated with a set of resources and has its own parent. Additionally, each node has a label whose length can be expanded to up to 63 characters. The root node unlike other nodes does not have any parent and its label is null (zero-length). [48] The absolute domain name for every node is the sequence of labels which are read from that node towards the root. An absolute domain name which is also known as a Fully Qualified Domain Name (FQDN) must be unique in the respective tree. To achieve this, the sibling nodes may not have same labels. In domain names, each node label is separated from the following label on the path by a dot. [3]

The DNS name space is separated into zones. Each zone has a set of authoritative servers that usually appear in two flavors: primary name server and secondary name server. The primary name server maintains all the resource records belonging to that zone in the master file. This file must be updated by the administrator to make a newly added host to that zone globally reachable. In the secondary name server, the zone data is derived from the primary server. The data is transferred between authoritative servers by using the zone transfer query. [22]

3.7.2 Resource Records

The information associated with domain names is held in the Resource Records (RRs). Each RR contains the different fields including name, class, type, Time to Live (TTL) and data. The TTL field specifies how long the record in a caching server would be valid. Upon the corresponding timeout the query must be sent to one of the authoritative DNS servers. Also, RRs may present various data based on the given value for the name and type parameters. [35] The message structure of resource record is presented in Figure 3.3.

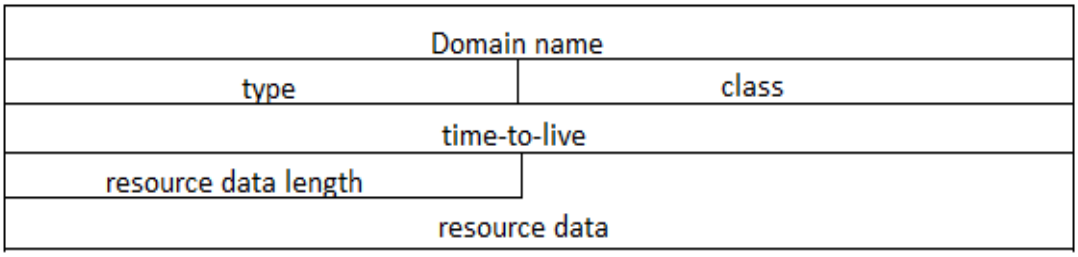

Figure 3.3: The message structure of resource record

For instance, if Type and Name fields are set to A value and a hostname respectively, the Value parameter will contain the relevant IP address. Using this RR, the hostname can be mapped to the corresponding IP address. The Value field presents the hostname of the authoritative server of the domain that is defined in the Name field if the Type parameter is NS. The authoritative DNS server is aware of the procedures that should be followed to obtain the IP addresses corresponding to the hosts inside the domain. If CNAME is assigned to the Type, the Value field has a canonical hostname of the Alias. [27]

Yet another type of resource record is called Naming Authority Pointer (NAPTR). NAPTR usually carries one or more Uniform Resource Identifiers (URIs). A URI is a flexible format that is used to identify end users engaging in any type of communication such as e-mail, VOIP, Instant Messaging (IM) and etc. This enables DNS to perform name lookups for a wide range of services. Each NAPTR record holds the list of contacts which is sorted

based on the Order and Preference fields. In a case that the list has more than one contact address, the client can switch to an alternative address once the destination host is not responding through one of the given contact addresses.

The most common usage of NAPTR is to code rule-sets in the DNS. NAPTR contains a regular expression that is used by an application in order to transfer a string into a new domain name. Since the regular expressions describe information in an extremely compact manner, thus large amounts of data by means of these expressions can be encoded in a relatively small DNS message. In a NAPTR record, flags and other parameters supervise rewriting and re-delegation operation. [32]

3.7.3 Resolvers

The DNS clients named resolvers are used by applications to retrieve information from the authoritative name servers. Commonly, a resolver generates a query upon a request from a program and sends it to the primary name server. When the resolver receives the DNS reply, it derives the required information and eventually sends this information back to the initiator application. [3]

3.7.4 DNS Message Structure

There are different types of DNS messages including query and response, in the DNS protocol. Additionally, there is one more DNS message called DNS update but it will not be examined in this thesis. All DNS queries and responses are encapsulated in the same format. This format is separated into 5 parts as shown in Figure 3.4. [35]

Identification	Flags
Number of Questions	Number of Answer RRs
Number of Authority RRs	Number of additional RRs
Questions	
Answer Resource Records	
Authority Resource Records	
Additional Resource Records	

Figure 3.4: DNS message format

The first part is the header and exists in all DNS messages. This section is 12 bytes long and contains several fields such as identification, flags, number of RRs, etc. Identification always is generated upon creating a DNS query and identifies the query. It also has to be copied to the corresponding response message.

There are various flags in the flags field. The query/reply flag specifies the type of message. The message is a query if this flag is set to 0; otherwise it is a reply. The name server sets the authoritative flag to 1 if it is the authoritative server for the queried domain name. In a query message, the querying host may set the recursion-desired flag to 1 and hence ask the queried name server to perform recursion resolution if it does not have the respective record. Also, the recursion-available flag is set to 1 in the case that the DNS server supports recursion. As the following parts in the message are variable-length, the number of fields determines their length separately. [47]

The four other sections that come after the header section are encoded in RR format. The question section, which uses fields such as query type and query name, conveys a query to a DNS server. The answer part may carry one or more RRs that are sent in response to the received questions. The authority section includes RRs that refer to an authoritative name server. [35]

3.7.5 Name-to-Address Resolution

However, DNS provides a large number of services for TCP/IP-based networks, the basic idea of DNS is to perform the name-to-address resolution. The purpose of the name-to address resolution process is to resolve the IP address of the queried hostname using the distributed DNS databases; however, a host does not necessarily have a name. Typically, the mapping process takes place when an application running on the host wants to start communication with a remote host. [20]

To do that, firstly a resolver running on the initiator machine sends a DNS query containing the remote host name to a local name server that acts as the primary name server. If the name server finds the IP address corresponding to the remote host name in its DNS database, it simply replies to the querying host with this IP address. In contrast, if the referred name server is not an authoritative name server for the zone of the queried domain name, its database does not contain the host name. In this case, the local name server usually forwards the query to the root DNS server. Currently there are 13 root domain name servers across the global Internet. They are authoritative servers for a huge number of organizational domains. [35]

The resolution process can be carried out in two ways. In the recursive resolution, the root name server returns a referral to the second-level name server that is responsible for the queried domain name. Upon reception of the reply, the local name server forwards the same query message to that server and gets a referral to the next level DNS name server. This method is applied over and over until the local name server queries the authoritative name server and hence obtains the IP address of the remote host from the response message. A recursive name resolution is shown in Figure 3.5.

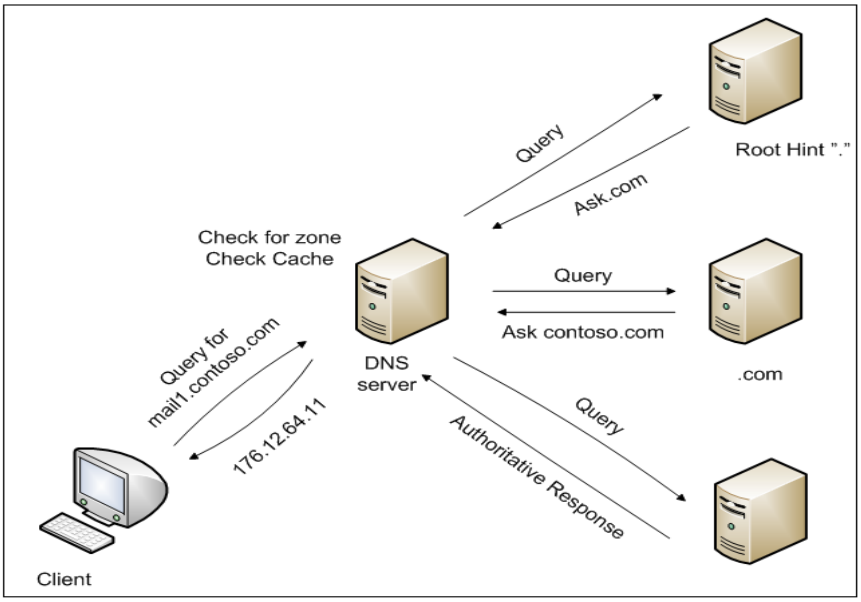

Figure 3.5: A recursive name resolution

On the contrary, when the resolution method is iterative, the local name server will not query the root name server if it does not have the queried domain name. Instead, it sends the query to the closest name server in the name space hierarchy. The request is passed on to next-level domain name servers repeatedly until the authoritative name server that has the IP address of the requested host name is found and thereby returns the destination's address to the local name server. [20]

3.7.6 Address-to-Name Resolution

In addition to the name-to-address resolution there is another resolution process called the address-to-name mapping. The queries sent for this purpose are called pointer queries. They are usually generated by system programs in order to handle management and debugging activities. Furthermore, the email servers and file servers use these queries to verify end users.

To perform a reverse DNS lookup, the resolver sends a pointer query containing an IP address to a name server. Upon the request, the name server returns the host name for the given IP address. In the hierarchical structure of the DNS namespace, the domain name is used as a search key. Therefore, inverse queries would need to check all domains attached to the nodes in the tree to find the requested IP address. As this searching process is impractical, the in-addr.arpa domain is defined for inverse queries. In this domain, the nodes are labeled with IP addresses of hosts but in the reverse order. The reverse name lookup can be performed in a more efficient way by executing a traversal of node hierarchy under in-addr.arpa. [20]

3.8 Fragmentation and Reassembly

Fragmentation happens when the size of a packet exceeds the MTU of any link along the route of the packet. As a result, the router cannot forward the packet due to its size and splits the payload inside the packet to multiple smaller segments. Next, each segment that is also referred to as a fragment is encapsulated in a separate packet whose length is not greater than the smallest MTU of involving networks in the path and further forwarded to the destination. [14] On the recipient side, the IP layer reassembles the data fragments derived from the incoming packets and thereby reconstructs the original packet.

A set of fields in the IP header including identification, flags and fragmentation offset are defined specifically to facilitate the fragmentation and reassembly operation. The identification field has the same value in all fragments, thus the remote host uses this parameter to identify each received fragment and relates it to the respective original packet. The total length always determines the size of the original packet before fragmentation. Consequently, each smaller packet contains the same value in this field. As an additional benefit, the IP of the destination host can use this value to recognize whether all fragments related to the original packet are received completely or not.

The offset field specifies the position of first byte of the payload within each fragment relatively to the original payload. This field is expressed in multiples of 8 bytes. The flags field contains two flag bits: don’t fragment (D-bit) and more fragment (M-bit). When the sender sets the D-bit to 1, the routers on the path to the destination host should not fragment the packet. The M-bit is set to 1 in all fragments except in the last one. The basic idea of the time-to-live field is to detect packets stuck in loops. Also it determines the maximum time period a host has to wait for each fragment that might be delayed, discarded or corrupted.

Despite that the fragmentation operation seems rather easy; it has bad effects on a number of protocols. For instance, the TCP protocol generates an acknowledgment message when the destination host receives all fragments belonging to the original packet. Therefore, if one of the fragments contained within the smaller packets gets lost due to any reason, the source host after a certain time period (i.e. specified in the time-to-live field) will retransmit the initial packet completely. It results in rapid consumption of the network resources and reduction in network performance. To overcome this, in most of the current TCP implementations, fragmentation is not allowed. This is done by defining the maximum data block size (limit). [20]

The path MTU discovery can be used as an alternative solution. In this procedure, the Don't fragment bit in the IP header is used to discover the smallest MTU of the links on a path. To do that, a sender considers the MTU of its first hop as the MTU of entire path and then sends a number of datagrams with the DF bit on towards a destination host. If the size of some of the datagrams exceeds the maximum MTU of any networks on the path, the router residing in that network will drop them and send back an ICMP Destination Unreachable message with a code that means "fragmentation needed and DF set". When the initiator host receives these messages, it chooses a smaller value for the MTU of the path. Once any of the datagrams is received by the destination host without fragmentation, the MTU discovery process completes and the assumed MTU is considered as the smallest MTU of the path. [36]

3.8.1 Incoming Fragmented Packets at NAT Device

Fragmented packets may be delivered to a NAT in an arbitrary order, depending on the packet ordering, network status and implementation of the fragmentation mechanism. Upon the reception of fragmented packets, different NAT devices exhibit different behaviors. Some NATs only forward the fragments arriving in order. This behavior is referred to as "Received Fragments Ordered". In contrast to this, "Receive Fragments Out of Order" relates to the NATs that are able to receive the fragments in any order. There are NATs in which any fragmented packet is discarded regardless of the receiving order. This property is known as "Receive Fragments None".

Fragmentation has been a target for a number of attacks. More specifically, attackers take advantage of either passing fragments through NATs or storing out-of-order fragments to launch several attacks. For instance, DoS attacks are likely to happen where a NAT has to store out-of-order fragments prior to processing them as a whole message containing all

headers. The problem of fragmentation for NATs is that the fragments after the first one do not have port information while the NAT depends on it. [5]

Let us assume a scenario where a malicious user divides a packet into rather small fragments so that some fields of the transport header are forced to be carried within the second fragment. Consequently, an illegitimate packet may go through a NAT, because the filtering rules that are related to those fields cannot be applied only by knowing the initial fragment. This attack is referred to as a tiny fragment attack. To eliminate this type of attack, a router can simply impose a specific fragment size limit on incoming packets to ensure that the first fragment includes all compulsory header fields.

The current reassembly implementation of the IP protocol allows each fragment to overlap with other fragments which are received before and thereby overwrite over their segments. There is an attack called overlapping fragment attack that has been formulated based on this weakness. To launch this attack, a hacker generates a number of fragmented packets so that the first fragment while carrying a legitimate data could go across a NAT. The subsequent fragments that have non-zero offset, might overlap with the previous-received fragments and hence change them. This opens up an opportunity for passing unauthorized packets through a NAT. [52]

4 Customer Edge Switching

This chapter introduces the main architecture and detailed description of the Customer Edge Switching. The chapter initially presents the basic idea of CES and further discusses for which reasons this scheme has been proposed. The subsequent section examines the CES operation in deeper detail.

4.1 Objectives

The CES concept is closely related to the reachability problem that the current Internet has due to intervening middle boxes especially the NATs. More specifically, the reachability problem appears where a host in a private address realm waits passively for incoming sessions from other address realms and wants to be globally reachable. Several proposals of which some were explained in Chapter 3.6 have been developed in the context of the reachability topic, but still a need for more powerful and efficient solution exists. To meet these requirements, the CES scheme has been proposed and implemented in the Department of Communications and Networking in the Aalto University. This proposal aims to shift the Internet towards a more secure and trustworthy direction. It also enables hosts residing in the public Internet to initiate a connection with a remote host in a private network behind a NAT. [23]

4.2 Requirements

Customer Edge Switching provides global connectivity between hosts that reside in different private address realms. The connectivity is based on globally unique names, locally or globally significant Identities of hosts/services or users and locally or globally unique addresses. An enhancement to state of the art is that a private host can be a server, not only a client. A CES is an extension to Network Address Translator. A CES hides the addressing of the customer network it serves. The inbound CES is responsible for detecting and eliminating source address spoofing. Given that this is effective this makes it reasonable for the inbound node to collect evidence of misbehavior of the sender and attribute the evidence to the sender's customer network or directly to the sending host. Because the trust requirements of various applications are very different, Customer Edge Switching offers a choice of different types of Identities and a set of checks the inbound node can make before admitting a new flow.

CES operates by means of DNS and NAT and acts as a gateway of a customer network to the Internet. To be exact, the gateway, containing DNS and NAT functions is the only

element where a network is adapted to the CES concept. However, most of the network elements including end hosts will not require any changes to interoperate with the proposed architecture. Although the Customer Edge Switching architecture reuses existing DNS protocols and record types to resolve the addressing of remote hosts, it introduces a special configuration of the DNS information. It was also assumed that the core network between private networks could be based either on Ethernet or IP. [47]

Additionally the proposal needs no keep-alive signaling from hosts. In this solution, users can be reachable from the public network because of regular network capabilities. It is also desirable to fit all necessary network capabilities into regular network elements rather using costly add-on servers. Moreover, each address realm can use its own forwarding technology such as IPv4, IPv6, carrier grade Ethernet or legacy technologies. [24]

4.3 Customer Edge Switching Overview

The CES solution has been proposed to obviate reachability issues that exist in the current Internet. It can also be seen as one way of implementing the Trust-to-Trust communication model introduced by David Clark [13]. A purpose of CES is to provide global connectivity for hosts that reside in private address realms.

A CES is a replacement and an extension to a NAT. A CES node in addition to the fundamental NAT features contains other attributes to resolve the reachability problem of private hosts. It allows inbound connection initiations with no need to any keep-alive signaling from hosts in the private address space. As described before, only some special configuration needs to be made in the existing DNS infrastructure without introducing changes to neither the DNS protocol nor the DNS record types. A CES controls the reachability and packet admission by applying diverse policies.

CES helps protecting the hosts from unwanted traffic. In this approach, it is possible to point evidence of misbehavior to the sender's network or at least detect RLOC spoofing and block the unwanted traffic. This motivates ubiquitous collection of evidence that can be further processed automatically for fast detection and blocking of bot machines infected by Trojans.

When user traffic is being transferred through a CES device, the IP address fields in the packet headers are modified from private addresses to public ones or vice versa. This conceals private addressing from external networks and reuses the private address ranges several times. As a result, the address shortage problem of IPv4 mitigates and IPv6

deployment becomes less urgent. Furthermore, a CES exploits globally unique domain names, private IP addresses and arbitrary type of identifiers to provide global communication across the Internet. It maps identifiers to locally significant addresses and thereby delivers packets to hosts residing in the customer network. This behavior improves trust and security on the address space boundaries. [23]

In terms of functionality, a CES operates like the tunnel router in LISP [17] especially when it is connected to an IP core network. The difference is that instead of End Point Identifiers that are actually addresses, CES introduces communications identifiers that are never used for routing. It also resembles a realm boundary node in TRIAD [19] architecture. The difference can be explained by differences in their routing approaches. The CES uses the ID to address translation instead of the source routing used in the TRIAD to make the private hosts globally reachable. This mapping is done based on the trust paradigm. Moreover, a CES supports powerful procedures including return routability checks to filter out unauthorized inbound traffic. [38]

4.4 CES Architecture

The CES concept classifies networks into Customer Networks (CuN) and Service Provider Networks (SPN). From the IP routing point of view the former are stub networks i.e. they do not carry transit traffic on IP layer. The end users may be attached to different CuNs. There is at least one CES in every CuN. CES itself has a firewall and a pool of local IP addresses for referring to user entities. On the other hand, each SPN encompasses several provider edge nodes. To be exact, two separate edge switches associated with different owners sit on the boundary of disparate networks. As a consequence, the ownership of the network is divided into appropriate blocks. [23] A Customer Network is a trust domain served by one or more CES devices. One or more Internet Service providers provide the connection from one customer network to another. The ISP networks are seen as one federated trust domain. The borders of trust domains conform to the boundaries of network ownership. A trust domain provides trust services for its users. User traffic can be tunneled over Ethernet core or IP or IP/MPLS core networks. Furthermore, the SPN network includes a Directory Service (DS) that may be implemented for example by using the DNS.

In this approach, an end host or a service is identified by an identity (ID) that is created from a unique host name by applying hash or alternative internal procedures. The purpose is to avoid publishing the private IP addresses across the Internet and thus isolate the access, corporate and customer networks from the core network. The IDs can be classified

into three different groups: globally unique IDs, locally deterministic 32-bit IDs like in LISP and reasonable unique random values. Assurances of the validity of an ID vary from privacy protection by the sender's network to network operator or 3rd party certificates. As the source and the destination IDs may need to be changed when special situations such as attacks occurs in the network, therefore using globally unique ID is not necessarily easy and cost efficient. On the other hand, according to the birthday paradox, it has been argued in [23] that the length of random IDs should be from 60 to 80 bits where a customer network aims to serve around 1M hosts. As a consequence, an IPv4 address field is not long enough to carry a sufficiently unique random ID. Hence, the random IDs may be carried in an ID protocol. A CES device or an ID server at the edge of a private network uses the locally defined algorithms to generate and manage random IDs. Due to this property, the process involving changing random IDs is less expensive than changing global IDs. As an additional advantage, this type of IDs can remain invariable while packets are crossing a multi-homed interface or end users are roaming into a foreign network. The CES concept is demonstrated in Figure 4.1. [38]

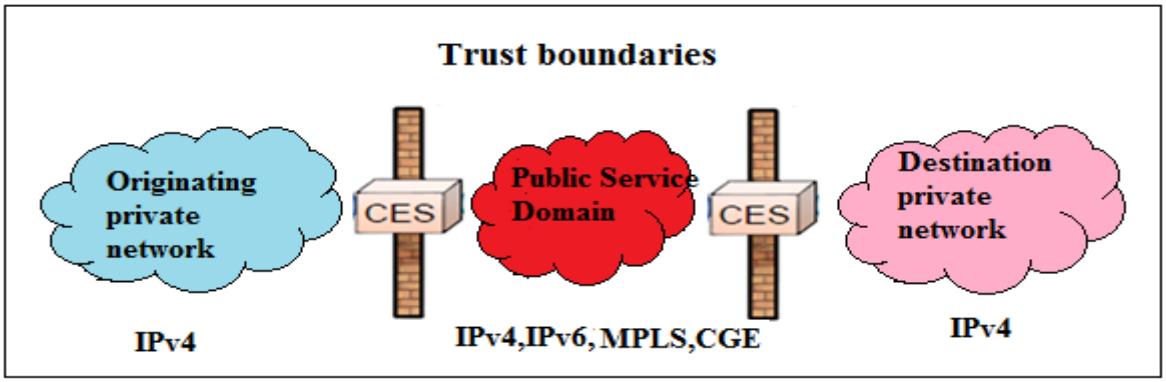

Figure 4.1: CES concept

4.5 Message Flow Across Trust Domains

The Customer Edge Switching solution reuses the existing DNS infrastructure including common DNS protocols and resource record types to implement the DNS server resolving names of remote hosts. Thus, if a server in a private realm wants to serve external client hosts, firstly it has to register its own domain name in the DNS server. Besides, its NS record must point to the authoritative name server of the corresponding customer network. [38]

As Figure 4.2 depicts, a message flow begins with a DNS lookup query from Host A. In other words, at first a client program running on Host A sends a DNS query for the A record of Host B's domain name. This message is routed to the DNS server via CES-A.

The CES-A acts as an enhanced DNS proxy. Thereby, it first checks whether the queried destination host is in the same private network. Then, CES-A picks up the DNS query, changes it into a DNS request for the relevant NAPTR record and ultimately forwards the modified DNS query to the DNS server. Commonly, the destination's NS record refers to an ID Server hosted by the inbound CES. Therefore, the DNS server passes the request to CES-B. [23]

The addresses of the CES public interfaces are called Routing Locators (RLOCs). An RLOC is a public address with a preference and has the properties: order and address type. The RLOC types vary depending on the forwarding technologies such as IPv4, IPv6 and Ethernet. The preference and order determine the selection of an RLOC from a set of active RLOCs. [38]

CES-B sends back the DNS reply containing its RLOCs and the destination ID to the outbound CES. The destination ID is carried in two parts within the reply message: ID type and ID value.

Upon the reception of the reply (the second message in Figure 4.2), CES-A allocates a local address for communication with Host B from its address pool. Then, it creates a connection state containing the allocated local address, the destination ID and the RLOCs referring to the inbound CES. The outbound CES does not forward the original DNS reply. Instead, it changes the destination ID to the local address (oc: b) representing the remote host to Host A and further returns the modified DNS reply to the local host.

After the name resolution phase, Host A establishes a connection with the destination host using the stored information in the connection state. To be exact, Host A sends a message with the local address as the destination address field to the CES-A (the third message in Figure 4.2). The source ID can be generated in two ways. It may be derived from a source host name and ports. The other option is to allocate an ID by CES. The destination ID is retrieved from the corresponding connection state held in the CES. As the next step, the packet is encapsulated in the IPv4 packet because the core network runs over IPv4. The actual address that is used for routing the packet to the inbound CES is carried in the outer header.

The generated state in CES-A in addition to the destination ID and respective proxy address contains the source addressing information and the sender ID. Moreover, CES in a similar way to NAT assigns certain timeout to each connection state. [23]

Once the message from CES A enters customer network B, the inbound CES allocates a proxy IP address representing the sender to Host B. [47] It also creates a state containing a mapping from the source ID to the allocated address. The destination ID within the message is exploited for the packet access control. More specifically, the inbound CES based on the destination ID can make a decision whether the incoming connection is legal or not. Furthermore, it maps the destination ID to the destination local address that specifies the route to the receiver across the target realm. The simplified process of message flow in CES is given in Figure 4.2.

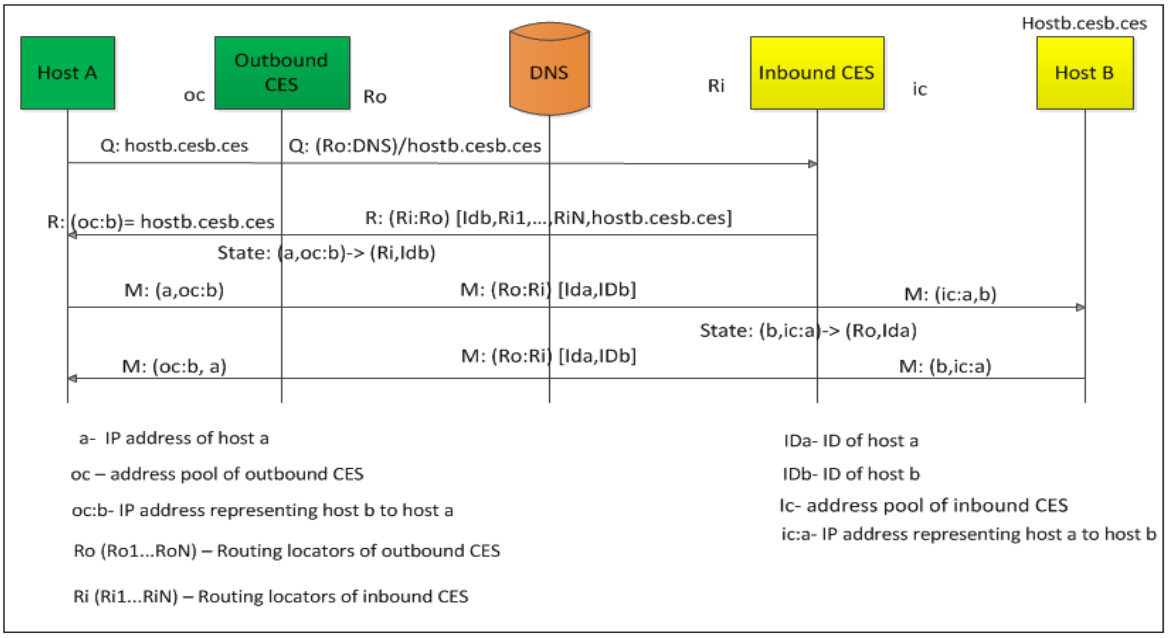

Figure 4.2: The message flow in Customer Edge Switching architecture [revised] (Kantola et al. 2012)

The inbound CES obtains the RLOC of CES-A and the ID of Host A upon the first packet of the service flow. On the other hand, the domain name of Host A cannot be learned from the incoming packets. Therefore, the inbound CES is unable to perform a DNS query and discover the additional RLOCs of the outbound CES. A remote-end-on-demand routing protocol such as CETP can be used to resolve this limitation. In special cases, the destination domain does not return the destination ID and RLOCs within a DNS reply. Instead, it refers to a separate ID protocol and ID server. However, using a separate ID protocol adds several round trips to the name resolution phase and has bad effects on performance. At the same time, it may reduce the risk of eavesdropping substantially. [38]

From the Host A point of view, CES-A works like a NAT. Generally the difference is in admitting communication invitations from external networks where all destinations reside in the private network. In general, a CES node operates in a similar way to a traditional

NAT except that it provides bi-directional connectivity for private hosts and resolves the hairpaining problem in a more efficient manner. [47]

In the described scenario, the outbound CES may receive several NAPTR records in response to the DNS inquiry. This happens when the inbound CES belongs to a multi-homed stub network and thus has multiple RLOCs. Since the CES only contains a DNS-like interface, the NAPTR records can be generated dynamically. The outbound CES may use a subset of RLOCs identifying the route to the inbound CES in parallel depending on the forwarding protocols that are used across the core network and other parameters specified in the NAPTR records.

This approach is compatible with well-behaved protocols like HTTP in which a DNS lookup is used for learning the destination identifier and address, whereas it does not work properly with the protocols that obtain the destination routing information outside the DNS. For each protocol of this type, CES requires a protocol specific state machine. The session-oriented protocols including FTP and SIP/SDP are examples of such protocol. The protocol specific treatment in CES is relatively similar to what exactly happens in state-full firewalls. Moreover, this proposal assumes that emerging application protocols use unique domain names instead of IP addresses as identifiers and all destinations seem to reside in the private network. Several ALGs for the Customer Edge Switching solution have been implemented within other thesis works by Petri Leppäaho and Jesus Llorente. [28], [29]

When a host residing in a private address realm initiates communication with a public server, CES node falls back to the regular NAT behavior. To make this possible, CES cannot re-use the global address space for the local communication. For interworking with legacy clients, a CES can contain the Private Realm Gateway (PRGW) functionality providing connectivity to servers in a private address space. The idea of PRGW has been prototyped by Jesus Llorente. [29]

5 Customer Edge Traversal Protocol

This chapter examines CETP that is proposed as a new tunneling protocol in the Communication and Networking Department of Aalto University in deeper details. [25] The chapter is divided into three sections. The first section discusses the main ideas behind the CETP proposal. Next, the chapter describes CETP packet formatting including the header fields and the payload segment precisely. At the end, the chapter explains how CETP assists the current Internet architecture to fulfill various upcoming demands while making minimal changes in network elements.

5.1 CETP Objectives

The growth of the Internet to its current size causes a large variety of problems and requirements that cannot be resolved by the original Internet architecture. The address shortage of IPv4, existence of middle boxes such as firewalls, unwanted traffic, mobility and multi-homing are the most common examples of such problems.

CETP is proposed as a part of an Internet Trust Framework (ITF) which consists of a number of Hosts, subscribers, agents of communicating parties called Customer Edge devices, ISPs, a Global Trust Operator (GTO) and applications involved in a communication. CETP with the other components of Customer Edge Switching aims not only to resolves existing problems in the Internet which are listed above, but also takes into account future needs.

The protocol is designed for packet delivery from one CES to another while transporting the source and destination IDs. In other words, the purpose of CETP is to tunnel a data flow that was originally received from a private host between the network-edges and also transport in-band signaling from one customer network to another. More specifically, CETP differentiates between the payload and the control signaling. The payload includes a tunneled data packet, whereas the control information section contains the signaling information that is being exchanged between two CES devices. This type of signaling is a step towards the implementation of cooperative firewalls.

Also, the CES with the help of CETP can be seen as a way of implementing the identifier/locator split idea. To make this possible, CETP transports different types of identities in the source and destination ID header fields. This method is more flexible and scalable than present procedures like HIP that have been proposed to resolve the problem of the dual role of address as an identifier and as a locator.

CETP collaborates with a CES device to isolate a customer network from the core network in terms of technology choices and routing protocols. To facilitate multi-homing and use of parallel technologies in the global Internet, the protocol transports a number of addresses on the public network called RLOCs. The RLOC types differ depending on the forwarding technologies used over the core network. These addresses can be also sorted based on preference and order values.

If a host can run CETP, RLOC spoofing becomes possible. This can be corrected by agreeing on a network engineering principle that CETP is always carried over a different VLAN in the core network than the plain IP traffic. If this is done, Inbound CES (iCES) can be sure that no host can spoof the source RLOC.

If a CES is compromised and spoofs its RLOC, this can be detected by iCES and the destination host will not be disturbed. The condition is that iCES applies a suitable policy. The same methods that now apply to tracing address spoofing apply to this case. In this case evidence cannot easily be attributed to the compromised CES.

Due to eliminating the possibility of RLOC spoofing by the sender, a CES can always attribute evidence of misbehavior of a host to the source customer network.

To reduce the chance of ID and address spoofing, CETP provides a range of tools for the private networks. CETP places the responsibility of using those tools to the receiver's network. This remarkably enhances trust between two corporate or customer networks involved in a communication. Furthermore, the protocol enables a CES node to define and apply various policies for each ongoing session. A policy describes the recipient's expectations (e.g. required ID type, etc.) from the sender. In other words, CETP allows the inbound edge to specify which ID types must be used to identify a remote host, whether protection against source address spoofing is required, whether the CETP control signaling needs to be signed to ensure non-repudiation and also whether the integrity of edge-to-edge communication has to be preserved by encryption or not.

The protocol facilitates host IP trace back, assurance of RLOCs and IDs of involved hosts and return routability checks on either naming or forwarding level. This provides non-repudiation of communication and trust enhancement between two communicating customer networks. In addition, a trust domain administration using those tools can more easily locate malicious senders and attackers.

An idea of CETP is to create a shift in the basic communication paradigm used in the Internet. The Best Effort paradigm in which the network puts its best effort to deliver sender's packets to the destination is by far the most popular communication model. The Publish/Subscribe is another paradigm for communication. In this model, none of the published messages would be available for the receiver unless it subscribes to the content beforehand. CETP moves towards the Best Effort Communications (BEC) that is a synthesis between the classical Best Effort and Publish / Subscribe approach. In the BEC model, the network should make its best effort to serve both the sender and the receiver and thus balance their interests in the context of communication. To achieve this goal, the subscription in the Publish / Subscribe solution is replaced by the policy in the BEC. This helps to limit the packet admission based on the receiver's consent. The interest of the receiver means that the recipient only admits packet flows that it wants and are dropped the rest. [25]

The protocol only operates between two CES nodes; therefore the entities residing in the access and customer networks are not required to be aware of CETP. Furthermore, CETP provides the Trust incident reporting for other ITF components that support the concepts of reputation and trust for each Internet entity. [51]

In CETP, all control information and also IDs are encoded in a TLV (type, length, value) format. Using the TLV data formatting addresses flexibility and future expandability. Hence, the protocol facilitates implementing extensions.

As an additional feature, CETP together with the server side PRGW improves the procedures that delay the extinction of IP address space for the mobile devices and objects in the Internet of Things. As a number of wireless user devices continues to rise, the protocol would relieve the IPv4 address space shortage.

5.2 Requirements

Within this thesis work, three models for carrying CETP are defined. In the first model, CETP is defined as a new Ethertype and then transported in an Ethernet frame over the core network. It is also feasible to define a new well known port for the CETP. To be exact, in this case, CETP models as a new protocol on top of UDP. Moreover, CETP can be directly carried over an IPv4 core network as a new transport protocol like SCTP or TCP.

To make CETP operational, a DNS server must be able to encode a wide variety of ID types in existing resource records (e.g. NAPTR record). Furthermore, a CES needs to process CETP packet flows in addition to other types of packets.

5.3 CETP Packet Structure

The protocol definition given in this thesis is the second iteration that emerged as the result of our prototyping and verification work. As illustrated in Figure 5.1, the CETP packet structure consists of two separate parts: the CETP header and payload. The CETP header is also divided into the compulsory control header and the optional control TLVs. The compulsory control header contains a fixed size segment and two variable length fields for the sender and destination IDs. Additionally, there is a flag in the compulsory control header that specifies whether the optional part of the header carries any TLV fields or not. The control TLVs are mainly used for the edge-to-edge signaling that is usually exploited by the inbound edge to decide about the packet admission.

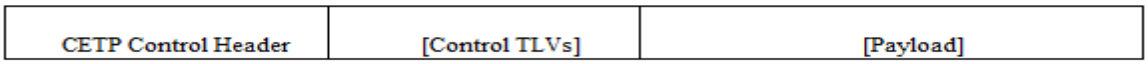

Figure 5.1: CETP packet structure

On the other hand, the data packet received from a private host is encapsulated into the payload field. This field always starts at a 32-bit boundary. A CETP packet containing a control TLV may have a non-empty payload field. In contrast, if there is no control TLVs within the CETP header, a payload must be present in the packet.

5.4 Protocol Compulsory Control Header

The CETP compulsory control header structure is shown in Figure 5.2. The compulsory control header is made of several fields including Version, Flags (C, R), Header Length, Payload Length, Source ID, and Target ID. The Version field indicates the protocol version. The Flags parameter consists of two bits: the C and R bits. If the CETP message includes at least one control TLV, the C bit must be set to one. On the contrary, the C bit would be zero when the CETP packet only contains a payload. The second bit, R bit, is reserved for future use.

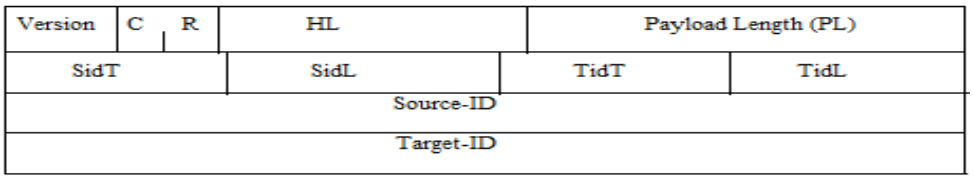

Figure 5.2: Protocol header

Due to the flexible size of the ID fields, the CETP compulsory control header length is variable. Therefore, the Header Length (HL) parameter is defined to specify the header length in octets. HL includes the length of the fixed part of header, the source ID, the target ID and the optional control TLVs. The total length of the CETP control header cannot exceed 2048 octets as HL is limited 11 bits. To be exact, HL is calculated from the following formula:

$$HL = \Sigma\,(3 + Length - in - TLV\ +\ TLV\ padding) + 8\ +\ SidL\ +\ TidL \\ +\ (nrof\ TLVs > 127)\quad all\ control\ TLVs\ included\ in\ the\ sum \qquad (1)$$

In the CETP packet, the payload size is given in the Payload Length field in octets.

5.4.1 Identity Encoding

The source ID and the target ID are defined via three parts: type, length, and the actual identity value. Due to this structure, the ID can have variable length, flexible format and various types. The type parameter, which is called SidT for the source ID type and TidT for the target ID type, indicates the ID generation algorithm used by the ID server. For example, IDs can be random values that a CES generates using internal algorithms. Furthermore, CETP by means of mobile operator infrastructure can use globally unique mobile operator assured IDs for end-to-end communication. Mobile Operator assured IDs can be, for example, MSISDN numbers, IMSI numbers and user certificates that can be verified by HSS/HLR. Unlike end-to-end protocols that identify the end hosts with the same assured IDs, CETP protects a mobile target device against the DDoS attacks and unwanted initial packets that are not carrying the assured source ID type.

The size of ID value is determined in the Length field which is called SidL for the source ID length and TidL for the target ID length in the protocol header. The ID itself fits into the value part. There is no limitation on the ID formatting in the current CETP specification.

In the first prototype of CETP, the source and the destination IDs were encoded in a similar way to the control TLVs, although in the current specification they are formatted as described above to facilitate packet processing.

5.4.2 Control TLV Format

It is feasible to separate the control information section from the payload within the CETP message structure. The control signaling contains one or more control TLVs and signals

control information between two communicating CES nodes residing at the edges of the trust domains. The control TLV format is depicted in Figure 5.3.

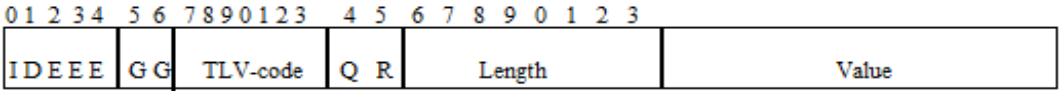

Figure 5.3: Control TLV format

The control TLV includes three segments: type, length and value. The type field occupies two bytes and is represented as follows: 5-bits for flags, 2-bits for group, 7-bits for TLV code and a 2-bit operation. The group specifies the high-level type of control TLV, whereas the code is more detailed and determines the type of TLV element in the group. Table 5.1 presents four control TLV groups.

Bits (GG)	Control TLV group
00	ID types
01	Payload types
10	RLOC types
11	Control information types

Table 5.1: Different control TLV groups

Appendix A presents TLV mnemonics for all existing control information. There are four possible operations including Query, Response, Reliable Response and Acknowledgement which are defined by means of the operation bits (Q and R bits). In a TLV query, a sender asks for a responder's value for a certain TLV type and then waits for a reply. The response TLV contains the answer (i.e. sender's value) to the previous query. In this case, the responder does not expect an acknowledgement TLV. In contrast to this, the reliable response that carries the same information as the response TLV requires an acknowledgement leading to a 3-way handshake edge-to-edge. So, query, response and reliable response always carry the sender's value. The acknowledgement may or may not carry the receiver's value. Table 5.2 lists all possible operations. [34]

Flags(QR)	Operation	Description
00	Query	TLV query that can also carry sender's value and sender expects a response.
01	Response	TLV response provides sender's value; in this case ACK TLV is not expected.
10	Reliable Response	Sender expects an ACK TLV from the destination. Besides, TLV includes a sender's value.
11	Acknowledgement	ACK TLV is sent as an acknowledgement to pervious reliable response TLV. It may carry the receiver's value

Table 5.2: Possible operations for control TLVs

Moreover, the type field has compatibility bits (named I and D bits) that specify how an unknown or an unsupported TLV type should be processed. To give an example, if both the I and D bits are set to zero; the sender asks the receiver to ignore the TLV if its type is not understood. Table 5.3 presents a different combination of compatibility bits. The flags also contain the extension bits reserved for future use.

Flags(ID)	Description
00	If type not understood, sender tells the receiver to ignore the TLV.
01	If type not understood, sender tells the receiver to ignore the TLV and send a Backoff code in response.
10	If type not understood, sender tells the receiver to delete all control TLVs in the message.
11	If type not understood, sender tells the receiver to delete all control TLVs in the message and send a Backoff code in response.

Table 5.3: List of combinations of compatibility bits

The length field size varies between one and two octets based on the value of the first bit. To be exact, if the first bit is set to zero, the length field occupies one byte while its size extends to two bytes in a case that the value of the first bit is one. This is done to optimize

the CETP message size as one octet length field is usually sufficient for specifying the value length.

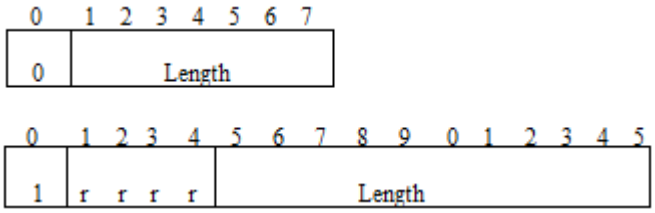

Figure 5.4: Length encoding for TLVs and IDs
L = 0 -> 7-bit length (0-127), L = 1 -> 11-bit length (0-2047)

The value is defined depending on the TLV type. The TLV type numbering space comprises ID types, payload encapsulations, reachability information, and signaling information. The type number space is common between the control signaling and payload. As a consequence, it is feasible to address specific information between the planes. It is noteworthy to mention, a TLV always is padded up to a 32-bit word boundary, however, the padding is not counted in the length field.

5.5 CETP Control Signaling

In this section, the format of control TLVs is defined according to the latest CETP specification.

5.5.1 RLOC TLV

The location of the source or the destination CES device is determined by the RLOCs. More specifically, an RLOC is a globally unique address of a given type. Each RLOC identifies the route to a CES node. The type specifies one of three currently defined RLOC types: IPv4 addresses, IPv6 addresses, and MAC addresses. In the RLOC TLV, the 2-octet length field encoding is chosen to keep the alignment of the rest with 32-bit boundary. The length is equal to the number of RLOCs multiplied by the size of each RLOC of a given type.

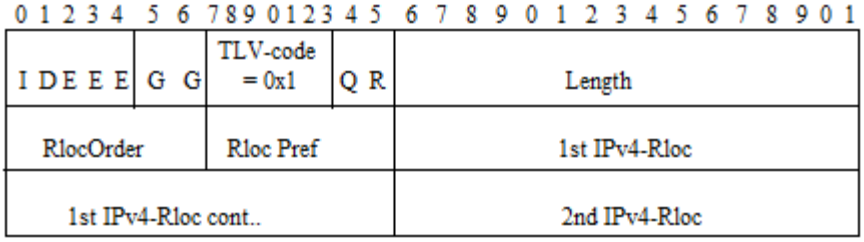

Figure 5.5: RLOC TLV format

As shown in Figure 5.5, each RLOC has a preference and order. The RLOCs are ordered based on the priority (i.e. combination of order and preference). As a result, the load can be distributed between several CES interfaces in a similar way as when order and preference appear in DNS address records.

The outbound CES obtains a set of inbound CES RLOCs and their default states from the DNS response, while the inbound CES needs to use CETP in order to learn alternative RLOCs for the outbound CES. For a legitimate packet flow, the reachability information transported by CETP should match the reachability information obtained from the DNS. Furthermore, the communicating CESs can use CETP to monitor and notify changes in the reachability information during a session. This can be used to split the load effectively, support multi-homing and to report unavailable RLOCs. The pace of monitoring is specified by the timeout of connection state.

If an inbound edge wants to use different RLOCs from the default ones stored in the DNS, the current state of RLOCs must be announced to the outbound edge by using the CETP RLOC TLV. Additionally, if an ongoing session does not meet the admission requirements of the inbound edge, the CES drops all queries originated from that source to eliminate network scanning. Depending on the policy, a CES may admit data flows from all alternative outbound edge RLOCs.

When a CES receives a CETP packet containing the sender's RLOC TLV with 0xFE preference value, it should switch to a new remote CES RLOC and update the current state of the destination RLOC instantly. Upon an RLOC switchover request, CES has to start admitting packets from a new RLOC. Admitting packets for an ongoing session on an alternative destination RLOC may be limited to a certain time period in order to make the DDOS attacks less likely. The state mirroring from an active CES to a standby CES is necessary when a hot swap of a session takes place. If the RLOC to RLOC delays vary substantially, a hot swap operation becomes complex. To avoid the performance penalty resulting from the state mirroring, the session hot swap may be only applied to the crucial and long term flows.

5.5.2 Timeout of the Customer Edge State

In CETP, the timeout TLV specifies how long the state of communication can be valid. This value will be restarted upon the reception of a message with the same source ID. The purpose of the TOUT TLV is to carry the desired timeout of sender's state information

during the negotiation phase. The timeout value is given in seconds. Figure 5.6 draws the Timeout TLV format.

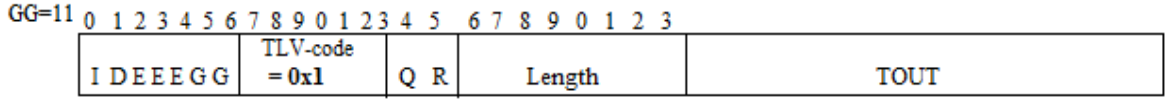

Figure 5.6: Timeout TLV structure

In addition, an edge node can notify the peer about the removal of outdated state information or ID revocation by mean of TOUT TLV with a zero value.

5.5.3 Cookie TLV

Cookie TLV encoding, which is illustrated in Figure 5.7, has the same format as other control TLVs. CETP defines the Cookie control TLV for implementing forwarding protocol level return routability check.

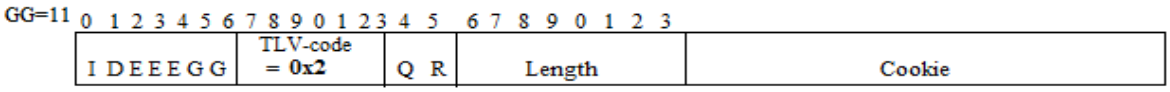

Figure 5.7: Cookie TLV formatting

To be exact, upon the first packet of a new flow, the inbound CES does not need to create a connection state. This is for reducing the chance of address spoofing. Instead, it can postpone the state creation to the next message, sending back a response message with a cookie TLV. If the source RLOC has not been spoofed, the outbound edge will respond with the same cookie. Once the inbound edge receives the next message with the same cookie and thus ensures that the source RLOC is genuine, it admits the traffic and creates an appropriate mapping state. In the Cookie TLV, the actual cookie is accommodated in the value field while the length specifies the cookie size in octets.

5.5.4 New ID Type Query and CA Address TLV

As a result of encoding the ID in the TLV format, both the source and the destination IDs can have different types, variable length and flexible format. CETP allows the edge node to tell what kind of ID is required to be used as the remote host identifier. Therefore, if an inbound edge receives a message with the source ID type which is different from the expected type, it creates a TLV query for the desired ID type and sends it back to the

outbound edge. Upon the reception of a New ID type TLV query, if the sender supports the required ID type, the outbound edge replies with a message carrying the queried source ID.

The ID TLV has a similar format to other control information. It belongs to the ID types group and its TLV-code is initialized by the required ID type for the sender identity while the value field in this TLV is always empty.

In some cases, the new ID type query and the CA address query are sent together because the inbound CES requires the validity assurance of the new source ID. However, if the new ID is assured by a trusted third party, CA address TLV query is not needed. The value of the CA address TLV gives the address (e.g. HSS address) that can be used for executing assurance queries.

5.5.5 Domain Information

The domain information TLV falls into the control TLV group and its value carries the sender's domain name. In the current version of CETP definition, there are two main usages for the domain information TLV: (1) replying to a reverse DNS query, and (2) performing a return routability check on naming level.

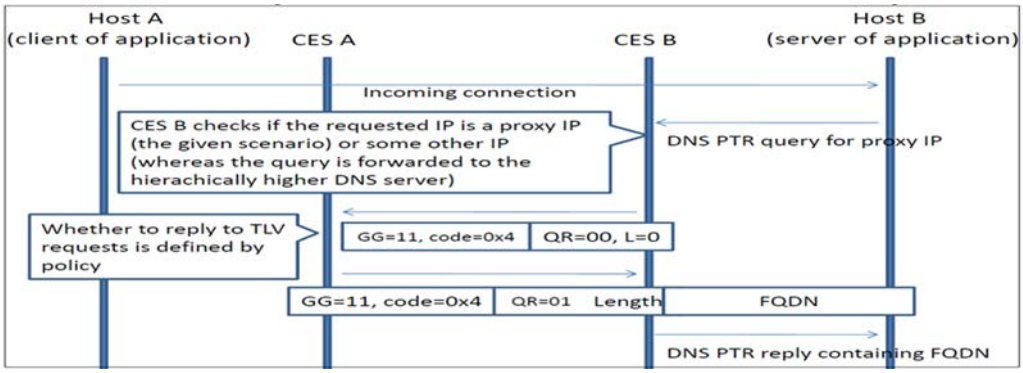

Figure 5.8: Example of reverse DNS query

In the first usage, as shown in Figure 5.8, once an edge node receives a PTR query from a local host, it forges a domain TLV query and forwards it to the peer. If the remote edge finds the FQDN corresponding to the queried ID, it will return a domain TLV response containing the host domain name to the sender edge. If the ID is matched to multiple FQDNs and also the responder edge wants to provide all, each FQDN must be sent in a separate TLV. On the other hand, if no FQDN related to the queried ID is found, the responder edge may reply with the empty domain TLV. Upon reception of the response, the edge node generates a relevant PTR reply using the received FQDN and forwards it to the querier.

For performing a naming level return routability check, an inbound edge follows a similar query/response procedure as in the reverse DNS lookup. The only difference is that the inbound CES performs a DNS lookup for the received FQDN. The return routability check procedure will be discussed in more detail later.

5.5.6 Signing CETP Header

The signature TLV is particularly defined to carry a signed CETP header including RLOC TLVs. The purpose of the signature is to prove the integrity of the protocol header and the origin of data while transporting across different trust domains. This implies protection against identity theft and man-in-the-middle modification of information. To support the signature mechanism, the responder CES needs to have its own identity. It also requires a registration in a certificate authority (e.g. HSS). Furthermore, the signature must appear after all other TLVs as it includes a signed full CETP header (both mandatory header and optional control TLVs).

5.5.7 Reporting Unexpected Messages

If an inbound edge receives an unexpected CETP message from a specific sender, it starts counting these messages. Once the inbound CES receives a given number, *N* of unexpected response messages, the CES deduces that the destination is a victim of a reflector attack and it reports the incident to the remote CES by sending a CETP message containing the reporting unexpected message TLV. The fact that *N* is chosen as a limit for the number of received unexpected messages prevents the amplification effect. As indicated in Figure 5.9, in the control TLV designed for the aforementioned reason, the length is set to *M* because the value carries the first *M* bytes of an unexpected message.

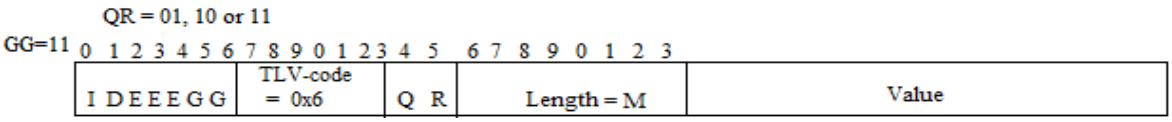

Figure 5.9: Unexpected message report TLV

This TLV is used to report that iCES suspects that it is a victim of a reflector attack and that the peer is being used as a reflector.

On the outbound edge, upon receiving the TLV, the reflector should tighten the policy for the incoming flows. For example, CES can eliminate the source address spoofing by executing naming and forwarding level return routability checks.

5.5.8 Backoff TLV

There are some error situations in which termination of the connection is required. CETP facilitates reporting such conditions by two separate Backoff code TLVs. The first type of Backoff codes relates to the error conditions associated with communicating CES devices and local parties. A congested CES, unavailable destination host and unknown connection are examples of such problems. Each of these problems is identified by a specific error code in the value field. The unrecognized TLV types including unknown and unsupported TLV types are reported by another type of Backoff codes. In this case, the value contains the unrecognized TLV in addition to the relevant error code. The unknown TLV error code is defined for the backwards compatibility with the future definitions of CETP, while the unsupported TLV error code indicates a strict policy or absence of some resources.

5.6 Reporting Unwanted Traffic and Malware

CETP does not support reporting malware and unauthorized traffic. Instead, other elements of ITF provide such reporting. There are two main reasons behind this design decision. First, detecting unwanted traffic and malware is a time consuming process and needs deeper packet inspection; therefore it is possible that the connection has been terminated before completing detection operation. Moreover, such reporting can be seen as a source of numerous security weaknesses such as badmouthing innocent hosts. So, we assume that instead of reporting evidence of misbehavior directly to the sender that may be compromised by a Trojan, the evidence should be aggregated and validated first by a trust management system. It will then be up-to the trust management systems to take action against the misbehaving host.

5.7 Payload in CETP

In the current version, the payload CETP uses two types of encapsulations: a minimal encapsulation of IPv4 payload and an Ethernet frame for a tunneled raw IPv4, IPv6 or other packets. In the payload part, the type contains the group bits and the payload TLV-code but no other flags. Thus, a negotiation for the payload encapsulation type must be performed by control TLVs. The control TLVs used for this purpose have a similar format to other control information and the control TLV-code is always set to sender’s preferred payload type. It is important to notice that TLV’s value in these signaling information does not offer any additional information to the peer. In other words, it is always empty.

5.7.1 Header Compression for IPv4 Payload

This encapsulation type resembles RFC-2004 (i.e. Minimal Encapsulation within IP) and is used in the case of IPv4 core network. The basic idea of Minimal Encapsulation is to reduce the header overhead resulting from tunneling. Since the destination IP address is derived from the mapping information with the help of the destination ID, this field is not present in the compressed IPv4 header. Figure 5.10 illustrates the IPv4 header compression format. Upon reception of a payload with IPv4 encapsulation at iCES, the IP header parameters are generated as follow:

Version = 4, IHL =20, Type of service – based on the local policy, Total length = Payload length + 16, Fragmentation cannot be supported, TTL = core IP TTL – 1, Protocol = copied from the original IP packet, Header Checksum – calculated locally, Source IP address: allocated by CES locally, Destination IP address – mapped by CES based on the connection state

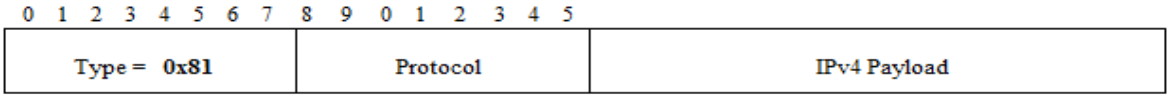

Figure 5.10: IPv4 header compression format

5.7.2 Ethernet Encapsulation for any Payload Protocol

CETP can carry any payload protocols that are transported on top of Ethernet. To achieve this goal, a new Ethertype for CETP is required. This encapsulation may be used to generate nested CETP messages. In other words, a CETP packet may contain another CETP message within its Ethernet payload. The structure of Ethernet payload is presented in Figure 5.11.

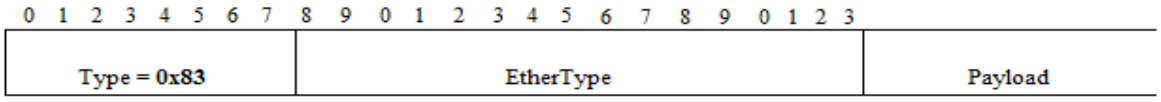

Figure 5.11: Ethernet encapsulation structure

The Ethernet encapsulation provides means for CETP to support fragmentation. Currently, the CETP header does not have any fragmentation fields. Instead, it handles packet fragmentation by switching between the payload types without adding the overhead of two byte fragmentation field in the header. More precisely, when the length of the data packet originated from a private host exceeds the MTU threshold and the DF bit is off, the CES fragments the packet into smaller portions and puts each segment into a CETP Ethernet payload. The experience of CETP usage will tell whether this kind of support for

fragmentation will be enough or a new payload format that would support fragmentation will be needed after all.

5.8 Security Mechanisms Implemeneted with CETP

In the Customer Edge Switching architecture, a CES makes the decision whether the incoming packet is legitimate and should be delivered to the destination host or not. The main result of this mechanism is establishing trusted communication between two endpoints. To achieve this level of trust, CETP offers a range of tools and procedures to CES devices. Some of these security mechanisms are listed in Table 5.4 and described in the following subsections.

Security Threat	Counter measurement implemented with CETP
ID theft	Signature, executing assurance query via CA address TLV
Eavesdropping, Man in the middle attack, Data modification,	Signature
IP(ID) spoofing, spamming, SYN flooding	Return routability check on naming and forwarding level
DDoS attack	Reporting attack via reporting unexpected message TLV

Table 5.4: Trust mechanisms of CETP

5.8.1 Return Routability Checks

Most of the protocols used across the Internet do not require authentication and confirmation of identity from senders. A malicious sender takes advantage of this breach; uses a spoofed address as its routing address and bombards the destination with unauthorized packets. Consequently, the receiver's resources are exhausted quickly and legitimate packets cannot be processed by the receiver.

CETP offers Return routability checks on naming and forwarding level as an effective procedure to detect packets with the spoofed source address or ID. In this approach, the receiver edge before accepting communication, checks whether the sender is accessible at the claimed address or not. CETP exploits control TLVs including cookie and FQDN to implement the return routability check mechanism.

When an inbound CES receives a first packet of a new flow, it may perform return routability check either on naming or on forwarding level. For performing naming level return routability check, the inbound edge requests the domain name of the initiator with the domain information TLV query and gets the sender's FQDN in the TLV response. After that, the inbound CES performs a DNS lookup for the received FQDN and retrieves all outbound edge's RLOCs. Then, it examines whether the sender routing information in incoming packets matches to one of the RLOCs derived from the DNS response. The described scenario is drawn in Figure 5.12.

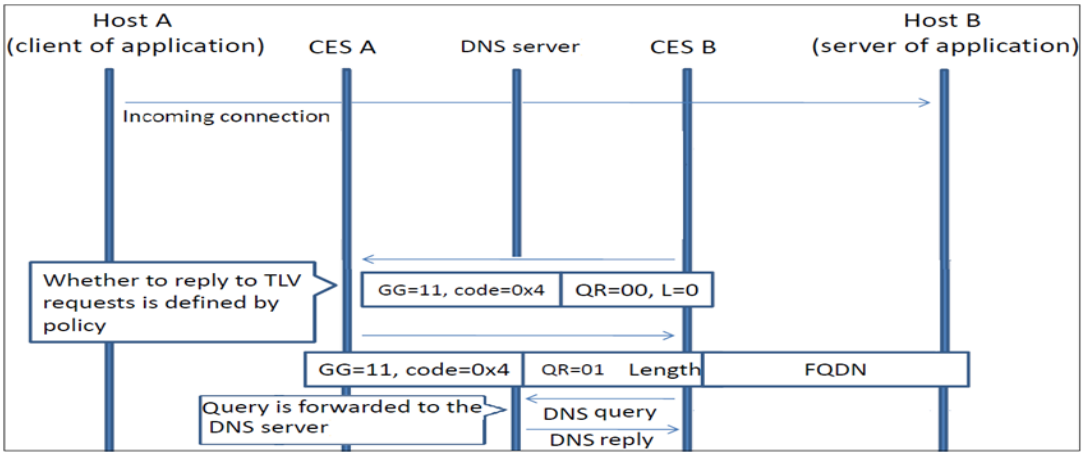

Figure 5.12: Example of return routability check on naming level

The cookie mechanism in CETP can be seen as one way of performing a return routability check on forwarding level. We borrow the cookie mechanism from SCTP that supports the cookie mechanism. In TCP it is possible to use light weight cookie mechanism; the receiver may drop the first SYN and generate a special 32 ISN (initial sequence number). When the receiver gets an ACK with a valid ISN it can create session state. Host implementations of TCP cannot interoperate with this technique, so the outbound and inbound CES will have to hide its use from hosts.

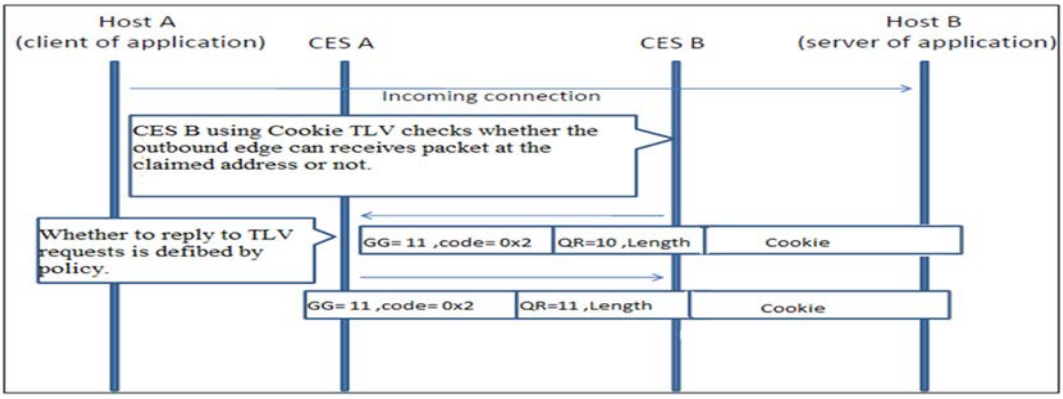

Figure 5.13: Example of forwarding level return routability check

In this approach, as shown in Figure 5.13, the querying edge sends a cookie TLV with the reliable response operation (QR=10) towards the outbound CES and then in the next message from the remote edge it expects to receive an acknowledgement TLV (QR=11) containing the same cookie.

5.8.2 State Management

Upon reception of a new legitimate flow, the inbound edge creates a connection state. Each communication state depending on its timeout can be active for a specific period of time. The state's timeout will be restarted once CES receives any packet with the ID stored in the state. On the other hand, the state entry will be removed if no relevant packets are received within the timeout.

Removal of outdated state information is signaled and synchronized between CES devices. In other words, when the timeout of state information expires, the CES deletes the state, informing the peer about the state expiry and requests the removal of the corresponding connection state by sending a timeout TLV with a zero value. The remote edge removes the corresponding mapping state upon the notification message and if it is still interested in continuing communication, it has to perform a DNS lookup and establish connection again. The edge node should also consider that the destination ID may have been changed after the connection abortion. Typically, the local host is informed about the removal of connection state.

5.8.3 ID Management

Due to the flexible format of IDs in CETP packets, two endpoints can negotiate about the types of ID that must be used by the other end. To perform an ID negotiation, an edge node creates a TLV query for the required ID type and directs it to the peer. On the other end, if the queried ID type is supported by the sender, the next message is built with the requested source ID. Upon the first message with a new ID, if all other requirements are also met by the remote edge node, the inbound edge creates a new connection state and delivers the data packet to the destination host. In the described scenario, using a cookie TLV for return routability check and CA address TLV for assurance queries are optional. The example of a new ID type query at the beginning of a new flow is illustrated in Figure 5.14.

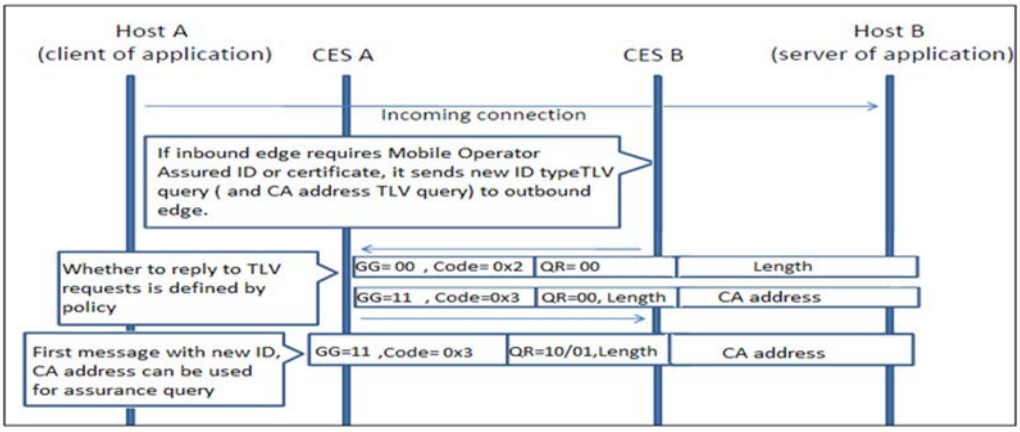

Figure 5.14: Changing ID type example

In addition, an ID can be revoked for example if an identity theft has been detected by a trust domain. For the ID revocation, a CES in a similar way to the removal of outdated state information will send timeout control information with a zero value to the remote end.

5.8.4 Signature

A plain text return routability check that is used for detection of source RLOC spoofing cannot disclose a stolen RLOC from a compromised ISP network. In other words, when an inbound edge uses the plain text routing addresses of a sender e.g. RLOCs for return routability checks, hijacking of RLOCs is still possible in compromised networks. This security weakness can be a target of several attacks. To avoid such incidents and attacks, the RLOC TLVs can be signed cryptographically.

5.8.5 Reporting Attacks

Upon attack detection, the trust domain can report the problem to the concerned party if the party is unlikely to be compromised. Let us consider a reflector attack as an example. To launch a reflector attack, a type of DDoS attack, an attacker takes control of a host in order to send legitimate queries towards the reflector's address. The hosts that are not compromised and send messages to the victim are known as reflectors. The compromised host accomplishes this by spoofing the victim's address as the source address into legitimate packets that it sends to the reflectors. The reflector only replies to messages with the spoofed IP-address of the victim according to protocol principles. An example is sending DNS queries with spoofed source addresses to the reflector. To counter this, an inbound edge may report the unexpected messages to the reflector. However, it does not make sense to report anything to an attacker that is under the control of a Trojan. Upon

reception of the unexpected message report, the reflector should tighten the policy for incoming flows. For example, the outbound CES can counter source address spoofing by executing naming and forwarding levels return routability checks.

5.8.6 Policy Control of CETP

On an inbound edge, a customer network can decide whether the incoming packet is eligible or not. To be exact, the trust domain requires certain information from the remote edge before admitting a new flow. These requirements are defined in a policy. A policy specifically defines the information that can be offered to the peer, the required source ID type, the required checks that need to be applied before establishing a connection and the state information that must be maintained for the duration of the session.

In most cases, a network administrator controls such admission policies in a similar way to the current firewall policies. However, it is desired that the policy associated with flows from / to a specific end user can also be managed by the end system. Controlling policy by end devices has its own security challenges and problems. Furthermore, the packet admission rules can be changed dynamically as a function of hostile activity using the information about suspected or detected attacks or according to the trustworthiness of the source and the destination ID.

Since the focus of CETP is on receiver's interest, it leaves to the inbound edge node to control packet admission. An admission policy can be controlled at least in two modes: lax and strict. In the strict mode, the inbound CES would usually reply minimally to the outbound edge queries and also does not deliver data packets destined to the private host before the peer responds to all its requirements. Compared to strict packet admission, the inbound edge with a lax policy offers more information to the initiator edge and forwards the packet to the target end device even before successful negotiation. An example of successful flow with the lax and the strict admission policy are given in Figure 5.15.

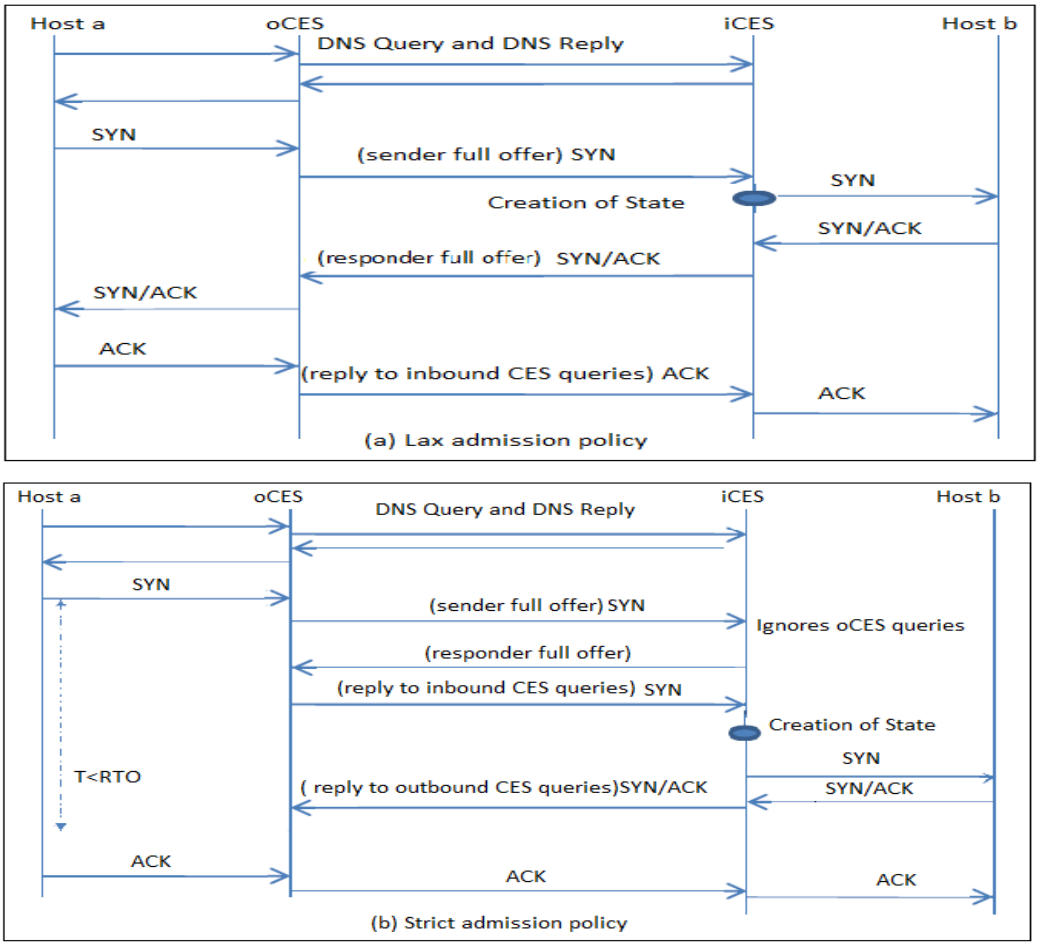

Figure 5.15: Example of successful flow with the lax and the strict admission policy

6 Principles of CETP Policy Processing

This chapter is dedicated to describing the CETP policy management framework. The chapter first explains the different components that are developed for policy control of CETP. Later in this chapter, some of the algorithms used for enforcing principles of CETP policy processing are examined in more details.

6.1 Policy Control of CETP

A CES can apply a wide range of policies to a data flow depending on the role of the CES (i.e. inbound or outbound), the source and destination ID types, destination port number and desired strength. The strength parameter enables the CES to apply more strict policy when the respective network is under attack or the peer endpoint is recognized suspicious based on the information gathered about suspected or detected attack attempts. However, in this prototype, the policy initialization is simplified and the policy for each new connection is initialized only according to the role of CES. More precisely, each host in the private network has separate admission policies for incoming and outgoing connections. Thus, upon reception of a flow associated to the host, a CES enforces the policy corresponding to that host considering the direction of the connection. Each control TLV belongs to its own interaction. CETP allows many types of interactions such as query – response, response, query - reliable-response – acknowledgement and reliable-response - acknowledgement. Table 6.1 lists different types of interactions.

Interaction	Description
Query - Response	With this interaction a sender asks for a control TLV and a responder replies to the query.
Response	With this interaction a sender offers its control TLV.
Reliable-response - Acknowledgement	With this interaction a sender offers its control TLV and in response a receiver has to send an acknowledgement upon reception of the TLV.
Query – Reliable-response – Acknowledgement	With this interaction a 3-way handshake for edge-to-edge signaling is implemented. A sender asks for a control TLV and a responder replies to that query. At the end, a sender has to acknowledge reception of the TLV.

Table 6.1: Different types of interactions

In addition, the communicating peer nodes must follow a set of principles (named policy constraints) so that their policies can be matched and they can communicate with each other. In other words, policy constraints define rules that limit the acceptable and feasible policies of involved parties. Some of policy constraints that the prototype takes into account during policy processing are:

- Since the policy control of CETP focuses on fulfilling the receiver's needs rather the sender's requirement, an inbound CES does not reply to the sender's queries before its own requirements are met by the outbound CES. Thus, if the sender cannot continue conversing with the peer before accessing certain control information of the receiver, it must repeat its demands in a following message.
- Each edge node must access to at least one RLOC type of the common RLOC types determined by the peer.
- The peers must support a common payload encapsulation.

The policy control of CETP is implemented through three classes: PolicyEngine, Policy and FSM (Finite State Machine).

6.2 Policy Class

Each policy is determined using a number of policy vectors. Each policy vector defines an aspect of policy for one or more TLV types. In other words, it specifies how each TLV type should be handled and processed. In this class, a policy is defined by the following policy vectors:

Role- It is assumed that the policies that the CES applies to incoming flows are different from the ones associated with outgoing connections. Hence, this vector specifies the role of the CES enforcing that policy.

ID-Reqc- This field defines what kinds of ID types identifying the peer host are acceptable from the perspective of the private host.

Reqc- This vector identifies certain control information (e.g. FQDN) that a home CES requires from the peer edge before admitting a new flow.

RR-Reqc- This vector includes TLV types that a CES offers and requires a reliable response operation and thereby expects to receive an acknowledgement from the peer.

Offerc- This vector defines TLV types that CES can offer or solicit to the peer particularly at the beginning of a session.

Available- This field indicates TLV types that are allowed to be offered to the peer upon requests.

As described before, policies are processed differently on inbound and outbound edges. To calculate acceptable policies, we focus on the negotiation phase prior to connection establishment. On an outbound edge, a sender host can be identified with the four ID types (Random, FQDN, MOC and MAID) and also can decide whether to provide a required ID type for the peer (6 choices). An outbound CES can reply to control TLV queries in five different ways: ignoring a query, sending a Backoff code with the response or reliable response interaction, providing its information with the response or reliable response interaction. For offering each control information, both outbound and inbound CESs have five options: not offering, offering its control information with response or reliable response, querying a TLV type with or without soliciting its own information. A receiver same as a sender can be identified with the four ID types and also can ask from a sender to change its ID (54 choices). With the current CETP specification (including eleven control TLV types), the number of possible policies is calculated as follows:

$Number of Outbound Policies\ 2{,}29E+15 = (ID\ policies\ 24) * (offer\ policies\ 5^{10}) * (response\ policies\ 5^{10})(1)$ *An outbound CES cannot send a message with its cookie.*

$Number of Inbound Policies\ 1{,}13E+09 = (ID\ policies\ 58) * (offer\ policies\ 5^{10} * 2) * (\text{response policies } 1)\ (2)$ *Operation of cookie cannot be query or response.*

6.3 FSM Class

Upon reception of the first packet of a new flow, a CES usually creates one instance of FSM for that connection. This object holds the current state of the ongoing session, the policy associated with the connection, previously sent TLVs towards the peer CES, list of control TLVs received from the sender edge and pending TLVs that should be returned to the peer as soon as the data packet is received from the local host. Furthermore, each FSM has several timeout parameters though they are not implemented in this thesis and were left for future implementation. Table 6.2 lists all timeouts that are defined in the FSM class.

Timeout	Description
timeout_p	oCES timeout for pending state.
timeout_on	oCES timeout for ongoing state.
timeout_CA	oCES/iCES timeout for CA response.
timeout_ka	keep_alive interval of CES whose timeout is longer.

Table 6.2: List of FSM timeouts and their description

6.4 Policy Engine Class

This class plays the main role in controlling the policy of CETP flows. When the CES prototype receives a packet, it examines whether the packet is to/from a legacy host or to/from a host behind another CES node. Moreover, each ongoing session may go from one phase to another occasionally. If the packet is related to CES-to-CES communication, the respective method in the policy engine class is invoked.

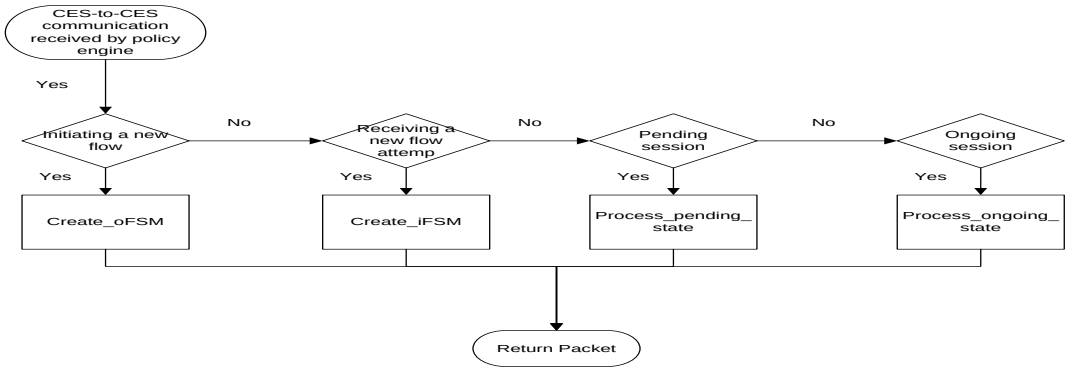

Figure 6.1: Policy engine Algorithms

Figure 6.1 shows how the policy engine behaves in the differing phases. On the outbound edge, the CES must initiate a flow upon a request from a local host while an inbound edge has to process flow arrivals. The state of the session is set to pending when the home CES cannot guarantee that the peer would respond to the sent message or not. Once the outbound edge responds to all queries sent by the inbound CES and thereby fulfills all destination hosts' needs, the inbound session goes to the ongoing state. Upon reception of TOUT TLV with zero value, the state of the incoming or outgoing session changes to idle. In addition, if the CES receives a query for changing the ID type of the local host, it sets the state of the corresponding session to pending. It is noteworthy to mention that an outgoing session will go to the ongoing state even if inbound CES does not fulfill its requirements. The reason is that we assume that the connection is established based on receiver's interests rather than sender's consent. Figure 6.2 illustrates the finite state machine of the policy engine.

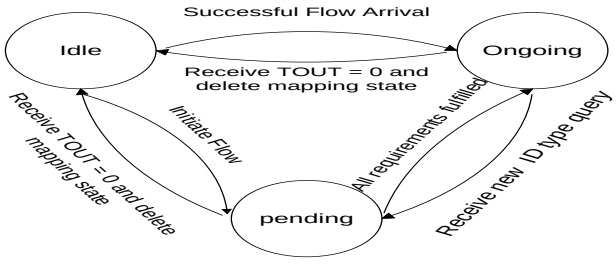

Figure 6.2: Finite state machine of policy engine

In this thesis, the DNS lookup for CES-to-CES communication is performed in a similar way to the first version of the prototype. The only difference is that when the prototype receives the DNS reply from the authoritative server, unlike the original prototype, it does not forward the message to the initiator local host. Instead, it buffers the DNS message and further starts negotiating with the peer edge. If the inbound edge verifies the legitimacy of the sender host and the connection is set up between the two edges, the outbound CES returns the DNS reply to the corresponding private host. After that, the host begins dispatching data packets towards the destination. This model is named postponing DNS message. The clear advantage of this model is that it avoids the need for buffering packets during the negotiation phase. This procedure is effective especially when the size of data packets sent by a local host is large (e.g. Video frames) and buffering packets would require a large amount of memory. Since the inbound edge does not deliver data packets to the destination host before the negotiation is completed successfully, an outbound edge would have to store data flows from private hosts during the negotiation phase, otherwise they would be dropped by the peer edge and negotiation as promised in the protocol could not be transparent to end points anymore.

Before implementing the postponing DNS message model, in an early version of the prototype, a DNS reply was forwarded to a querier preceding the negotiation phase. For avoiding problems that emerged due to buffering data packets, we decided to implement the postponing DNS message model rather than the original procedure.

In the initialization phase, the CES prototype derives the information related to each private host including its identifier and policy number pointing to the specific admission policy from the configuration file. This file also presents the list of policies with arbitrary policy vectors. The prototype stores these policies using policy class objects. Each packet admission policy has a unique number and a private host based on its policy number is linked to a certain policy. Furthermore, as described before, this prototype does not use the public key infrastructure. Instead, the peer CES’s certificate and home CES’s private key are given in the setting file.

6.5 Auxiliary Methods in Policy Engine Class

The policy engine uses a number of generic algorithms in addition to the methods performing the actual policy processing. They help the policy engine with creating the full requirements, processing and replying to the control information from the peer, creating cookie and checking cookie. Each of these methods will be explained in detail later.

6.5.1 Policy Engine Algorithm for Creating Full Requirements

At the beginning of the negotiation, each edge node depending on its policy declares the required information from the peer within the CETP control TLVs field. The edge node requests this control information in order to verify the validity of the peer. In this phase, it may also offer some information about itself such as its signature to the peer. The CETP packet whose control signaling is initialized in this way is called the full requirement. To make such a CETP control signaling, the policy engine uses the Creating Full Requirement algorithm. Figure 6.3 draws the process of creating a full requirement.

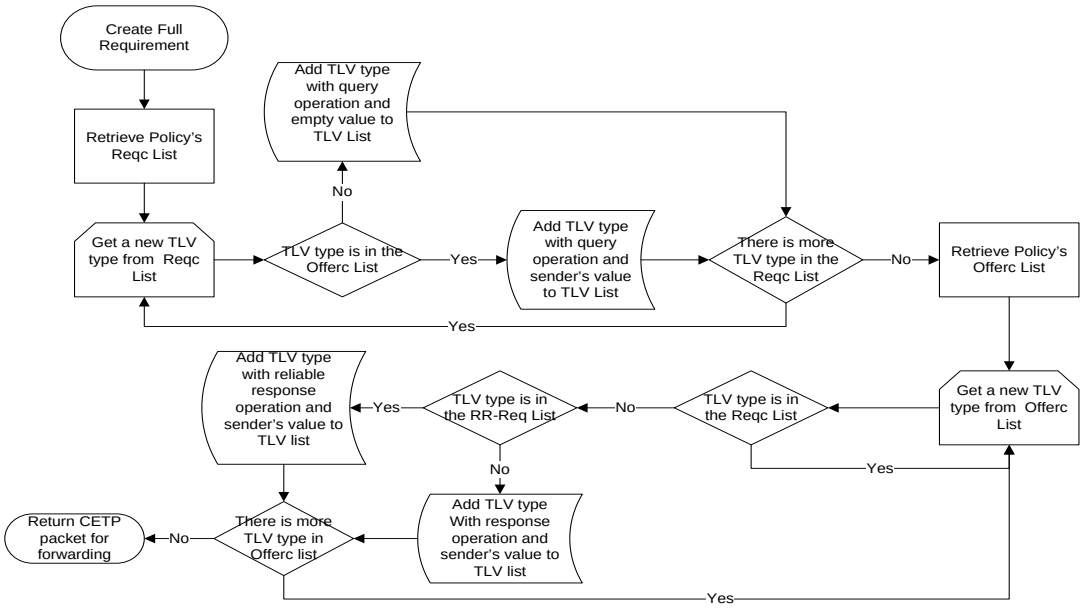

Figure 6.3: Creating full requirement algorithm

This algorithm first iterates over the Reqc list of given policy and then for each TLV type in the list determines whether it is also present in policy's Offerc list or not. By this procedure, the appropriate parameters for each required TLV type can be defined and further added to the CETP TLVs list. If the TLV type in the Reqc list is also found in the Offerc list, it means that the edge in addition to inquiry for that control information is also willing to offer its own information in the TLV's value field; otherwise the value part of the TLV is empty.

After adding all required TLV types to the CETP control signaling, it comes to add the information that is going to be solicited by the sender in the full requirement. To do this, the TLV types in the Offerc list are retrieved one by one and checked whether they are also in the Reqc list. If not, the algorithm examines whether the TLV type exists in the RR-Reqc list of the policy or not. If it is found in that list, the TLV's operation is assigned to

reliable response in order to tell the peer that it expects an acknowledgment for that TLV; otherwise the operation of the TLV is set to response. In both cases, TLV's value field contains the home CES information for that specific type. It is important to notice that if the TLV type is present in both the Reqc and Offerc list, there is no need to add a separate TLV for each query and offer operation. Instead, the policy would be applied by a single TLV field with the query operation and the value segment carrying the sender's value.

6.5.2 Policy Engine Algorithm for Processing Control Information

When the policy engine receives a CETP packet from the peer edge, it has to process the control signaling and payload separately.

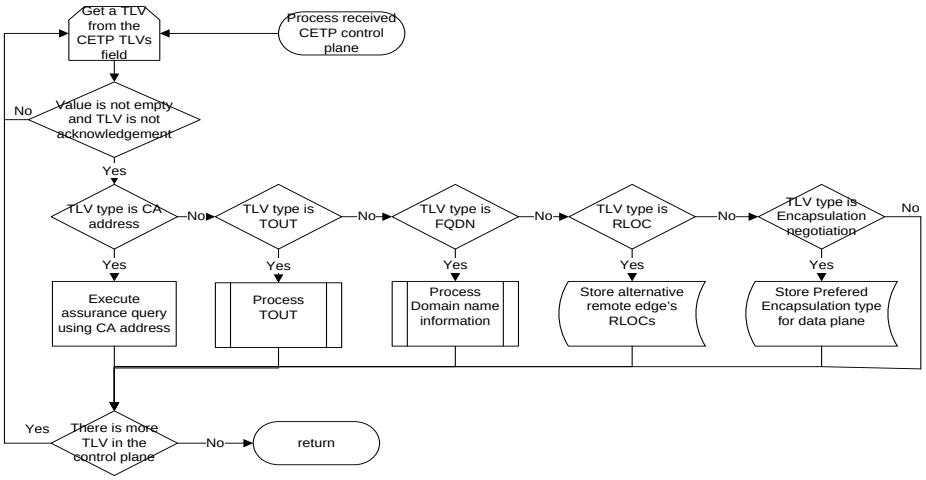

Figure 6.4: Policy engine algorithm for processing of CETP control information

For processing control TLVs, the policy engine uses the procedure which is illustrated in Figure 6.4. This algorithm first retrieves each control TLV of the CETP TLVs field and checks the TLV's value. If it is not empty and its operation is not acknowledgement, a certain operation is invoked based on its type and operation.

If the TLV type is CA address, an assurance query may be performed using the address in the value segment to validate the remote edge's certificate, however, this functionality is not implemented in this thesis work and left for future work.

Assuming the TLV type is not CA address, the algorithm checks whether the type is TOUT or not. If yes, the Process TOUT procedure is called. This method first checks the role of CES. If the role of CES is outbound, the TOUT value is examined. This is done because if iCES accepts oCES TOUT, this opens vulnerability. Since the peer edge sends the TOUT TLV with zero value to notify connection termination, thus if the value is zero, the algorithm removes the corresponding mapping information and stops processing the

remaining control TLVs. On the contrary, if the TLV's value is not zero and is also greater than the local timeout, the connection state's timeout is set to this value. The described algorithm for the TOUT processing depicts in Figure 6.5.

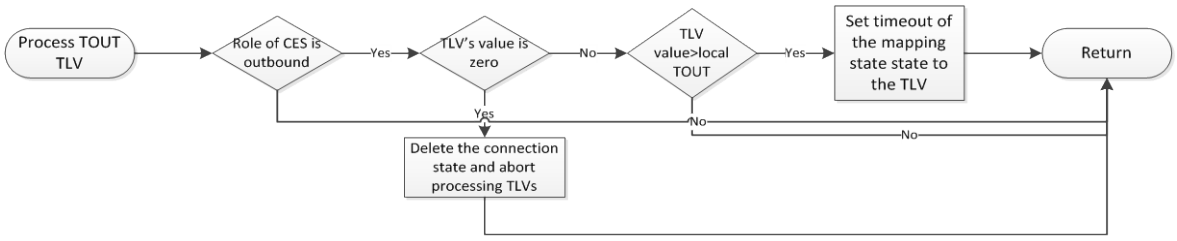

Figure 6.5: Policy engine algorithm for TOUT processing

If the TLV type is neither CA address nor TOUT, the algorithm checks whether it is domain information or not. Assuming the TLV type is FQDN, the control information is handled by the Process Domain Information algorithm which is visualized in Figure 6.6. The method first tests whether this domain information is received in response to the previously sent FQDN query which was forged upon the PTR query from the local host. This is done by checking the DNS query list that maintains all unanswered DNS queries. If the relevant PTR query is found, the PTR reply with the received domain information is generated and sent back to the private host. In contrast, if no PTR query matches with the FQDN TLV, the offered domain information is used for naming level return routability check. To do this, the algorithm executes a DNS query using the domain name and compares the remote CES's RLOCs obtained from the DNS reply with what the peer edge claims as its address in the source address field. If the sender address does not match with any RLOCs in the DNS reply, it means that the source address is spoofed and the connection must be terminated. In the developed algorithm, it is assumed that a DNS server always replies to the sent queries, however if any DNS responses are not received upon queries, the algorithm goes to a deadlock state.

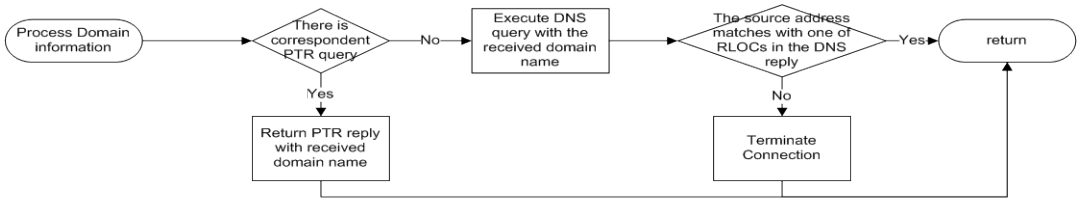

Figure 6.6: Policy engine algorithm for domain information processing

If the TLV type does not match with any of the previous control TLV types, the algorithm examines whether it belongs to the RLOC TLVs category. If the answer is positive and there is a corresponding mapping state in the connection table, each RLOC in the TLV's value segment is stored in the connection state as an alternative address identifying the

route to the peer. The peer's RLOCs in the connection state can be sorted based on their preferences and orders; however, this prototype does not sort RLOCs using these priorities as multi homing and hot switchover are not implemented and thereby there is no need to change the destination's RLOC during a session.

If the TLV type does not pass the RLOC type test as well as other described control information, the algorithm checks whether it falls into the payload type negotiation TLVs or not. If it does and the home CES supports the peer's preferred encapsulation type, this payload type will be used by the CES for encapsulating the future data flows.

6.5.3 Policy Engine Algorithm for Signature Verification

The policy engine invokes the Signature Verification algorithm in order to check the legitimacy of the peer's signature. This algorithm that is presented in Figure 6.7 first generates the MD of the CETP header in a similar way to the signature generation operation. Then, it verifies the validity of the signature using the peer's public key. If the signature is not validated, it means that the CETP header is modified and signed by a third party. As a result, to protect the customer network against MITM and DDoS attacks, the connection would be torn up. The local edge may notify the remote edge about the connection termination by sending a TOUT TLV with a zero value. However, this notification message opens a way of scanning for malicious users; in the developed prototype it is always sent upon connection terminations for debugging purposes.

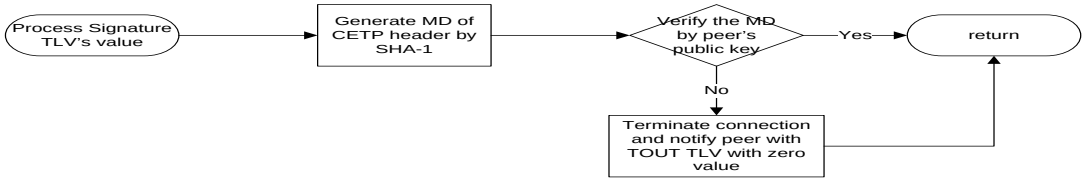

Figure 6.7: Policy engine algorithm for signature processing

6.5.4 Policy Engine Algorithm for Replying to Control Information

After processing the control information of the received CETP packet, the policy engine starts making responses to each control TLV carried in the CETP control signaling.

The algorithm used in the policy engine for this purpose is shown in Figure 6.8. This method, same as the previously described algorithm, iterates over all received TLVs in the CETP TLV field and for each one of them builds an appropriate control TLV reply based on the session's policy, TLV type and operation. As the policy engine exploits separate

procedures for handling the ID type negotiation process, thus the algorithm ignores replying to the control TLVs sent for the ID negotiation.

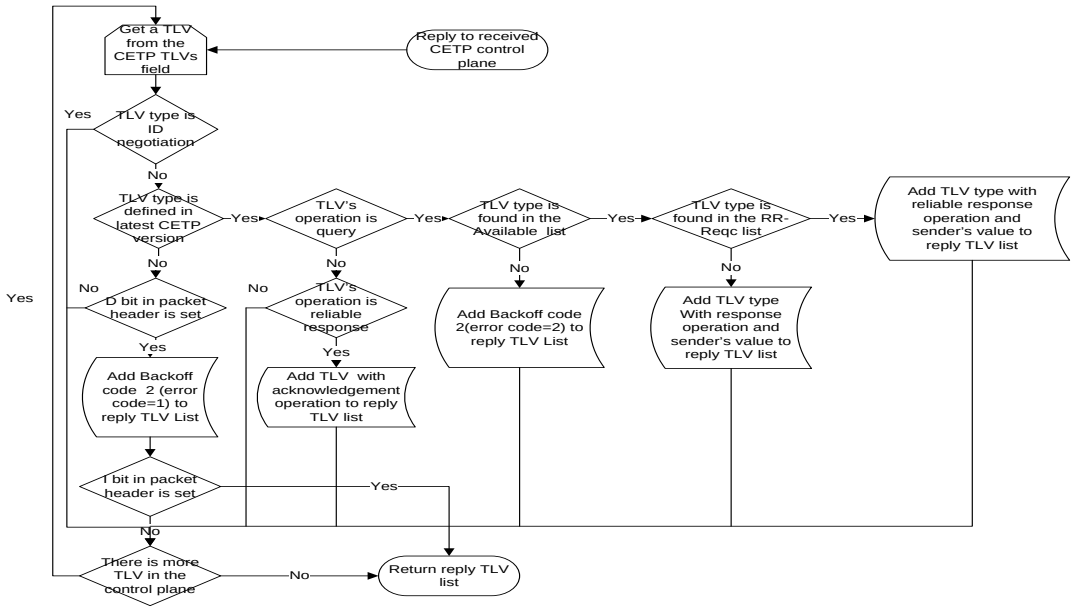

Figure 6.8: Policy engine algorithm for replying to control information

It is likely that the old versions of CETP do not know about the future modifications. Therefore once a new feature is added into a new edition of CETP (without changing the version number), its treatment by old editions should be specified by the ID bits that are understood by all CETP versions. To add backward compatibility to the protocol processing, this algorithm checks whether the TLV type according to the latest CETP specification is known or not. If the type is unknown and the peer edge asks to be notified about unknown TLV types via setting the D bit in the CETP header, the algorithm adds the Backoff code 2 (and error code = 1) which is defined for this sort of error conditions to the control TLV reply list. However, this prototype regardless of the D bit's value returns the Backoff error code when unknown TLV type is detected. This is done to ease tracking negotiation messages and debugging. Assuming the TLV type is unknown and the Backoff code is already added, the algorithm examines the value of the I bit in the packet header. If the bit is on, the algorithm aborts the control information processing and does not provide any replies to the received CETP control signaling. In contrast to this case (i.e. I bit is zero), the next TLV would be derived and processing starts from the next TLV.

On the contrary, if the TLV type is recognizable based on the current CETP version, the procedure exhibits various behaviors considering the TLV's operation. If the operation is query, the algorithm checks whether the home CES according to the policy's Available list

is allowed to offer the required information to the peer or not. If the answer is yes, it examines whether the TLV type is found on the policy's RR-Reqc list or not. If it is found in that list, the TLV's operation is set to reliable response; otherwise the TLV's operation is assigned to response. In both cases, TLV's value field is filled with the home CES information for that specific type. On the other hand, if the CES could not offer the requested information, the Backoff code 2 (and error code = 2) that aimed for unsupported TLV types is added to the reply list.

Assuming the TLV type is known and the operation is not query, the algorithm checks whether the operation is reliable response or not. If yes, the TLV with the acknowledgment operation and the same as received TLV's value is added to the reply list. It is worth noting that in this prototype, the inbound CES does not return any reply to cookie with the reliable response operation.

6.5.5 Policy Engine Algorithm for Cookie Generation

As a cookie TLV is generated and also checked by the same edge node, each communicating party can have its own internal algorithm for cookie generation. In this prototype, a cookie is produced as follows. First, the expiry time is calculated by incrementing the current time of the system by 5 minutes. Next, the identity of the destination host is concatenated with the string constant named SECRET and further the resulting string is given as an input to the MD5 hash function. Then, the constant string namely VERSIONofSECRET, the generated hash value and the expiry time period are merged together to create the corresponding cookie. At the end, the algorithm encrypts the cookie with its secret that is generated by the AES algorithm. The cookie generation formula is presented in Figure 6.9.

$$cookie = VERSIONofSECRET + hash(SECRET + Destination\ ID) + expiry\ time \quad (1)$$

$$encrypted\ cookie = encrypt(cookie, shared\ secret) \quad (2)$$

Figure 6.9: Cookie generation algorithm

The purpose of including expiry time in the cookie string is to eliminate reply attacks. In other words, it is infeasible for malicious users to keep sending an overheard cookie after its expiry time as the CES verifies the validity of the cookie while processing it. Exploiting two constant strings (VERSIONofSECRET, SECRET) in the cookie generation algorithm allows another way of replay attack prevention. These strings must be changed at regular intervals in order to avoid unauthorized users from using an intercepted cookie for an unlimited time period. It is worthy to note that the current prototype uses the time-invariant

constants in cookie generation as the cookie is exchanged only during the negotiation phase which is very short. Additionally, this procedure binds the cookie to the specific session by means of the destination ID that is used in the cookie structure. Due to the cookie encryption, the peer edge can neither access the actual cookie content nor generate a new cookie and use it instead of the original one.

6.5.6 Policy Engine Algorithm for Cookie Verification

To check whether the received cookie is identical to the one previously sent towards the peer, the Cookie Verification algorithm first decrypts the cookie with its secret. Then, it follows a similar procedure to the Cookie Generation method and produces the prefix_cookie as follows:

$$Prefix\ Cookie = VERSION of SECRET + hash(SECRET\ +\ Destination\ ID) \quad (1)$$

It is noteworthy to mention that the destination ID used in this string is equal to the source ID field of the packet. In the developed algorithm, it is assumed that the sender ID remains unchanged during cookie interactions.

After generating the prefix of the cookie, the algorithm compares the string with the same portion of the received cookie string. If they were identical, next the validity of the cookie needs to be examined. To do this, the cookie's expiry time is extracted from the cookie and it is compared with the current system time. If the expiry time is greater than the system time, it means that the cookie is valid; otherwise the cookie has expired. This prototype terminates the session in the second case. As described before, for ease of debugging, once the connection is terminated, the prototype returns a notification message (CETP packet with TOUT TLV with a zero value) to the peer.

6.5.7 Policy Engine Algorithm for Checking ID Requirement

The policy engine exploits the Check ID Requirement algorithm which is depicted in Figure 6.10 so as to figure out whether a sender's ID type meets the ID requirement of the session's policy or not. To do that, the algorithm compares each ID type in the ID-Reqc list with the source ID type of the CETP packet. If the sender's ID type is not found among the required ID types, the procedure creates a CETP packet with an ID TLV query for the desired ID type (IDQ). As described in Chapter 5, in order to enhance security and trust between communicating edges, this message may also carry a cookie, CA Address and signature TLV queries depending on the connection's policy. Additionally, this algorithm does not give any priority to any of the required ID types and always makes a query for the

first ID type in the ID-Reqc list. However, a future implementation may have a certain mechanism to differentiate between the preferred ID types.

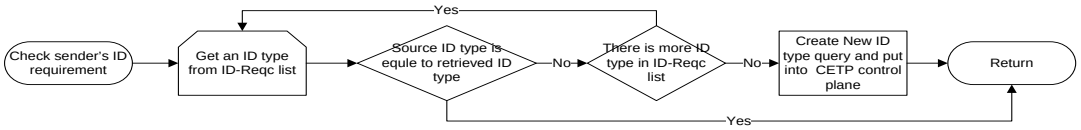

Figure 6.10: Check ID requirement algorithm

6.6 Policy Engine Algorithm for Creating oFSM

As described before, this prototype implements the policy control of CETP using the postponing DNS message model. Therefore, upon reception of the DNS reply from an authoritative server which is sent in response to local host's DNS lookup, the policy engine on the outbound edge begins negotiation with the peer through the Creating oFSM algorithm. Figure 6.11 shows how the policy engine applies the policy when a flow is initiated.

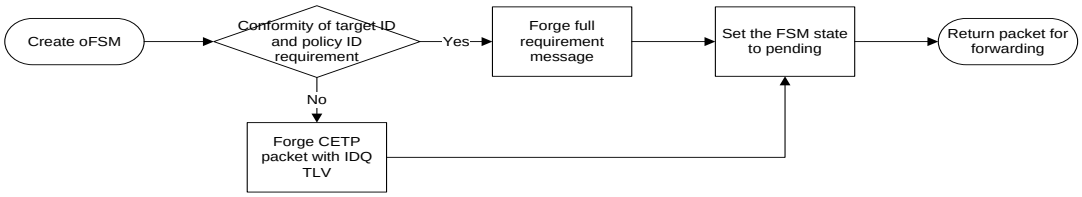

Figure 6.11: Policy engine algorithm for creating oFSM

Before invoking this algorithm, the connection state and the FSM instance are created for the session and both are initialized by using the DNS reply information. At this step, the FSM's state and transaction parameters are set to Idle and Initiate Flow respectively. According to the value of the session's state and transaction, the policy engine calls the Creating oFSM algorithm. The algorithm first derives the policy associated with the local host and then checks the conformity of the destination host ID type obtained from the DNS reply with the ID requirement of the private host's policy via the Check ID Requirement method. If the ID requirement is not fulfilled, the algorithm returns the CETP packet with a New ID type TLV query and other related control information; otherwise it returns a full requirement message which is built using the Creating Full Requirement procedure. It is important to note that in both cases the payload of the generated CETP packet is empty and also the FSM's state is set to pending before exiting the algorithm.

6.7 Policy Engine Algorithm for Creating iFSM

Upon reception of a packet of a new flow from a peer, the policy engine invokes the Creating iFSM algorithm that is depicted in Figure 6.12.

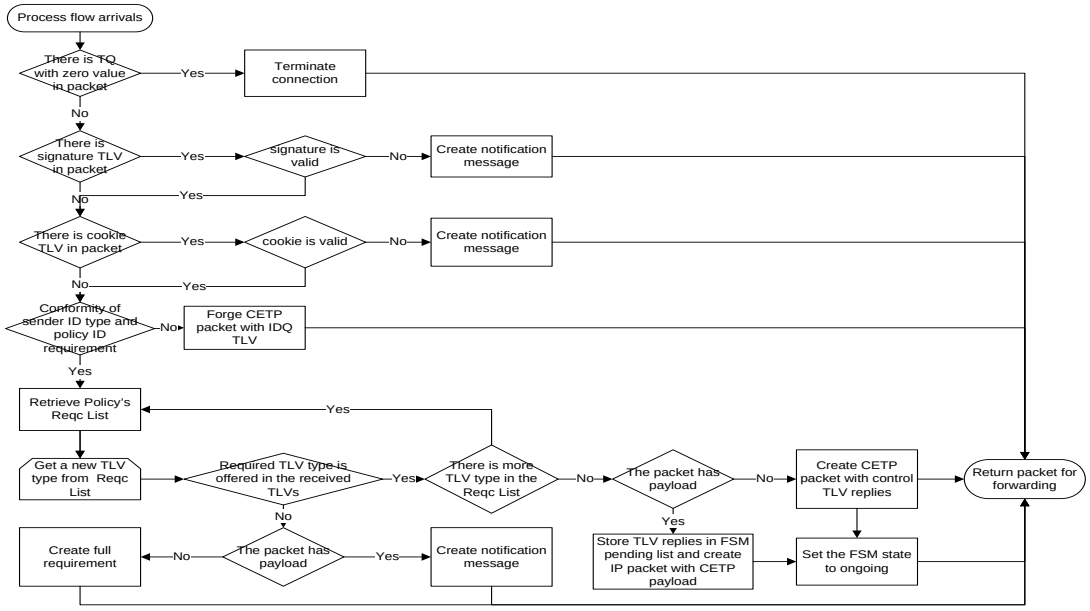

Figure 6.12: Policy engine algorithm for processing flow arrivals

Since on the inbound edge the prototype does not keep any state for a new flow before a successful connection setup, once the policy engine detects a packet that is not related to any entry in the connection table, it calls the Creating iFSM algorithm. In other words, this algorithm is applied to all packets of a new flow preceding connection establishment. The algorithm first checks for a cookie TLV. If the cookie is found in the CETP control signaling, it verifies the validity of the cookie using the Cookie Verification method. If invalidity of the cookie is proven, the algorithm terminates processing and for debugging purposes returns the notification message (i.e. CETP packet with TOUT TLV with zero value). A similar procedure applies to the received signature TLV using the signature verification algorithm. The algorithm also aborts policy processing if the CETP packet contains the TOUT TLV with a zero value. In the current prototype, this message is only meant to notify terminating connection on the peer edge.

Assuming the CETP packet does not contain an invalid cookie, an invalid signature or a notification message, the Creating iFSM procedure derives the inbound policy corresponding to the destination host. Then, with the help of the Check ID requirement procedure it examines whether the sender's ID type meets the ID requirement of the destination policy. If the ID requirement is not met, the algorithm returns the CETP packet

with the New ID type TLV query for the preferred ID type and other related control information.

On the contrary, if a sender is identified by one of the preferred ID types, the algorithm goes to the next step and examines whether the peer edge fulfills other destination host's requirements or not. To achieve that, it iterates over the policy Reqc list and looks for the relevant values within the CETP control signaling for each required control information. Then, it uses the processing control information method to check the received TLVs. If any of the required information cannot be answered with the received control TLVs, the algorithm returns a full requirement message. It is important to notice that in this case if the CETP packet carries the payload along with the control information, the Creating iFSM instead of a full requirement returns a notification message so as to tell the peer that the connection cannot be established. This design decision is made because in the postponing DNS model it is assumed that the first message does not contain any payload. Therefore, from algorithm's perspective, the CETP packet containing a non-empty payload means that the inbound edge has already sent a full requirement and the outbound CES failed to reply to queries appropriately. Consequently, there is no need to repeat a full requirement.

In contrast, assuming all requirements are met, first a connection state and FSM for the ongoing session are generated and the state of the session is set to ongoing. Next, the replies to the received TLVs are made via the Replying to control information algorithm. After that, the algorithm checks the CETP payload. If the packet has payload, the data packet is retrieved from the payload field and the generated control TLV replies are stored in the FSM pending TLV list. Depending on the payload type, the algorithm exhibits differing behaviors. In case of the IPv4 compressed encapsulation type, the complete IPv4 header with the destination host IP address as the destination address and the allocated proxy address as a source address are added to the data packet and returned for forwarding to the destination. For the Ethernet encapsulation, the entire IP packet is available in the CETP payload. Thus, only the source and destination address fields are modified to the destination IP address and proxy address respectively. But if the payload is empty, the algorithm returns a new CETP packet with the control TLV replies and an empty payload for forwarding to the peer CES.

However, there are different types for RLOCs and encapsulations and it is feasible that an inbound policy includes more than one preferred RLOC and encapsulation type; it is enough for the peer to support only one of the required RLOC and encapsulation types. As an additional assumption, this algorithm ignores a changing ID type request from a sender

in order to minimize the exposure of the destination host information before assuring the sender's legitimacy.

6.8 Policy Engine Algorithm for Processing Pending State

As described before, on an outbound edge the session goes to the pending state after sending the first CETP packet. Therefore, upon reception of the response message from the inbound CES, the policy engine exploits the Processing Pending State algorithm to process the incoming CETP message and forge relevant packets.

Figure 6.13 illustrates how the Processing Pending State algorithm works. The algorithm first checks whether the CETP packet carries TOUT TLV with a zero value. If the answer is positive, it terminates policy processing and deletes all state information and FSM related to that session. The algorithm exhibits the same behavior if either invalid cookie or invalid signature TLV is found in the packet's control signaling In this case, a DNS error message is returned to the local host as a notification of unsuccessful connection establishment.

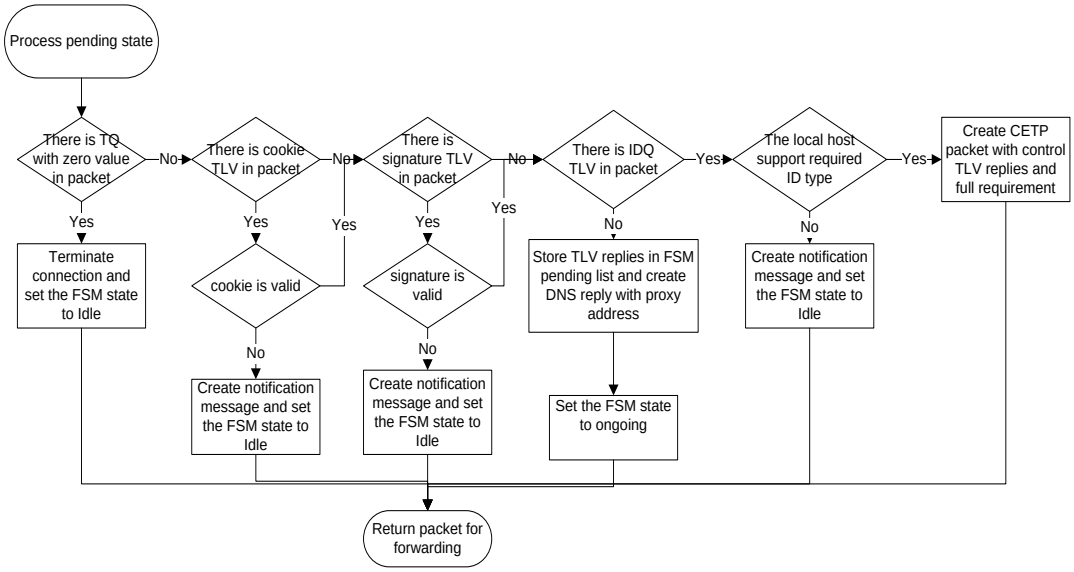

Figure 6.13: Policy engine algorithm for processing pending state

Assuming the CETP packet is not a notification message; the algorithm looks for the IDQ TLV within the control signaling. If ID Query is found, the algorithm checks whether the local host supports the required ID type or not. If the host cannot be identified by that ID type, the procedure returns a notification message to tell the peer that the connection cannot be established. In contrast, if the local host registers with the requested ID type as well as the default random ID in the CES, the local host ID in the corresponding

connection state and FSM changes to the value of the required ID type and the CETP packet with the new source ID containing the control TLV replies and full requirement is returned for forwarding to the peer. It is important to note that in this case the session remains in the pending state.

When there is no IDQ TLV in the packet, the algorithm sets the state of the session to ongoing. Then, the replies to the received TLVs are made via the Replying to Control Information algorithm and are stored in the FSM pending TLV list. Finally, the DNS reply containing the proxy address representing the destination host is sent back to the local host by the algorithm.

6.9 Policy Engine Algorithm for Processing Ongoing State

The policy engine processes a packet that belongs to a session in ongoing state using the Processing Ongoing State algorithm. Figure 6.14 shows the algorithm which is designed for policy processing of ongoing state.

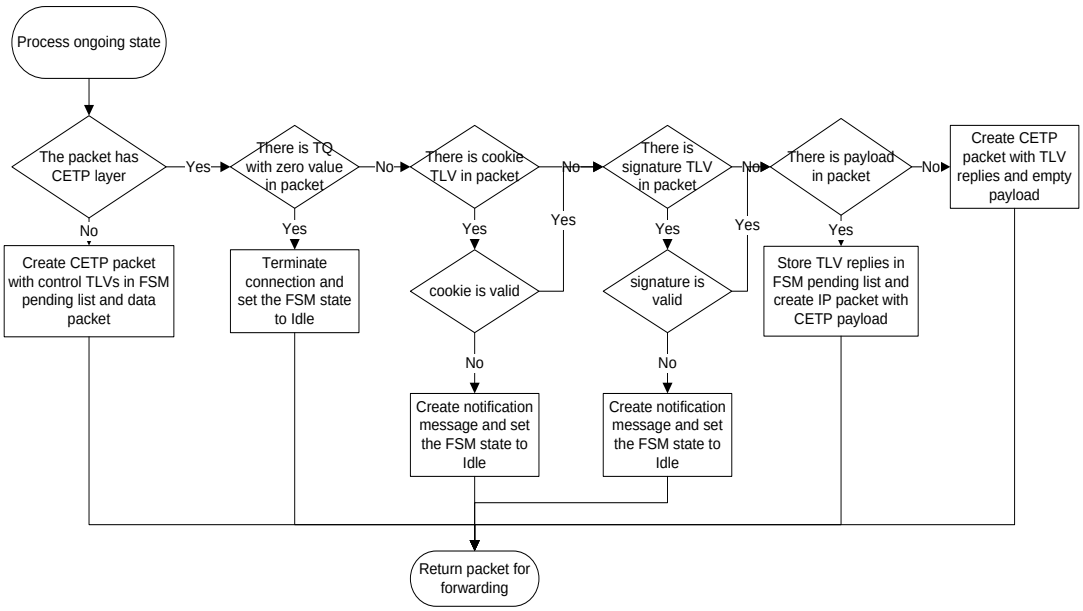

Figure 6.14: Policy engine algorithm for processing ongoing state

The algorithm first checks whether a packet has CETP layer or not. If the packet is not CETP message, it means that the message is originated from a local host and need to be forwarded to the peer edge. Therefore, a CETP packet whose control signaling contains control TLVs that were previously stored in the FSM pending TLV list and its payload that were the actual data packet received from the private host is generated and returned.

Assuming the packet has a CETP layer, the algorithm looks for TOUT TLV with a zero value. If this TLV is found, it terminates policy processing and deletes all state information and FSM related to that session. The algorithm exhibits a similar behavior if either invalid cookie or invalid signature TLV is found in the packet's control signaling. In all aforementioned error cases, it returns an ICMP message (with an error message of a Host is Unreachable) to the local host as an indication of the connection termination.

If no notification message, invalid cookie or invalid signature was detected, the algorithm goes to the next step and checks the CETP payload. If the payload is not empty, the data packet is extracted from the payload field. The algorithm modifies the IP header of the data packet based on the payload type. If the encapsulation type is IPv4 compression, a new IPv4 header with the destination host IP address as a destination address and the proxy address as a source address is added to the data packet and is returned for forwarding to the destination. In contrast, in the Ethernet encapsulation, the whole IP packet exists in the CETP payload. Hence, only the source and destination address fields are modified to the destination IP address and proxy address respectively. On the contrary, if the payload carries nothing; the algorithm returns a new CETP packet with the control TLV replies and empty payload for forwarding to the peer.

However, any of the communicating parties may request from the peer to change its ID type while the session is in ongoing status, but this case has not been taken into account in this prototype and left for future implementation.

7 Implementation and Evaluation

This chapter describes a prototype that we implemented to verify the protocol logic and its policy control explained in the previous chapter in details. At the beginning of the chapter, the scope of the test network, the required experimental setup and necessary network elements that were used in the test environment are discussed briefly. Next, the chapter presents the test results which are collected from more than one hundred different test cases. The general configuration of the testing environment is explained. Then some of test cases are examined in more details to show how the prototype actually works. The difficulties and issues we faced while designing and prototyping CETP and relevant functionalities are discussed later. Finally, the effectiveness of the protocol for edge-to-edge tunneling and policy control of cooperative firewalls is evaluated.

7.1 Test Network and Components

The prototype network is simulated on a PC running the Linux/Debian operating system. Since the prototype requires a set of computers (at least five) to examine various test scenarios, KVM (Kernel-based Virtual Machine) solution is used for virtualization (available at http://www.linux-kvm.org). Using KVM, multiple virtual machines can be run on a single PC so that each virtual machine has the virtualized hardware such as a network card and is running the Linux/Debian operating system. To access virtual machines from the main PC and run the source code files on the respective virtual PC, SSH connections are established.

7.2 The Scope of Implemented Prototype

This thesis work is integrated into the primary CES prototype that has been implemented as a proof of concept. Therefore, this prototype, like the original prototype, contains a simplified implementation of the Customer Edge Switching architecture described in the fourth chapter. Moreover, there are some limitations that were set to this prototype. To facilitate the implementation work, the host registration operation is excluded. Instead, the assumption was made that hosts are already registered into the respective CES and each CES node maintains the valid information of all active hosts residing in its customer network. It is also deemed that all virtual machines are behind a NAT and there is no connection from the external network to the virtual PCs. In addition, the prototype does not implement the hash algorithm that is used to calculate sufficiently unique random type of IDs. The hash values are generated beforehand and stored along other ID types of that host in the Host Register List (HRL) and the DNS database.

This prototype does not include the dynamic initialization of admission policy as a function of network condition, the source and destination ID types and other parameters. It is assumed that each host in the private network is tied to a certain admission policy that remains invariant under any situations. As CETP packets can be transported over the Ethernet or on top of the IP layer, the CES device exploits the NAPTR resource record type to forward a DNS query originated from a local host. The peer RLOC that is a routable address to that CES and destination's ID are encoded in a regular expression of the focusing type within the response section of a NAPTR record:

''*A B U* ''ID+idprotocol'' !^(.*)$!dest: *C*,*D*?*E*=*F*!''

The *A, B, E* and *F* parameters describe the order, preference, address type and the actual address of a given type respectively. The *C* and *D* parts indicate the destination ID type and the actual value of ID.

In the original prototype, for CES-to-CES communication, the source and destination IDs identifying communicating endpoints are carried in address fields of the IP packet and the Ethernet frame, whereas in this implementation, CETP is used to tunnel data and signal control related information between CES nodes while transporting different types of ID.

For implementing the signature TLV of CETP, the prototype does not exploit a public key infrastructure. Instead, it is assumed that each CES has certificates of remote CES nodes beforehand. Also, it does not use the diameter protocol to assure certificates of other CESs. In the test network, the IPv4 routing is used within the private networks while the core network can be built over Ethernet, IPv4 or IPv6.

The prototype by default identifies user devices in a private network with a unique random type of IDs; however, upon a request to change ID type, any other type of ID can be used as a user identifier. Furthermore, this work does not include on-demand multi-homing functions and it was left for a future implementation.

7.3 Experimental Setup

The purpose of the test network is to simulate two customer networks and a core network between them. The test environment consists of a number of private hosts (e.g. Host A, Host B), two CES devices at the edge of each private network (CES A, CES B) and an authoritative DNS server. Figure 7.1 illustrates the prototype network that is built using virtualization on a single PC (cesproto).

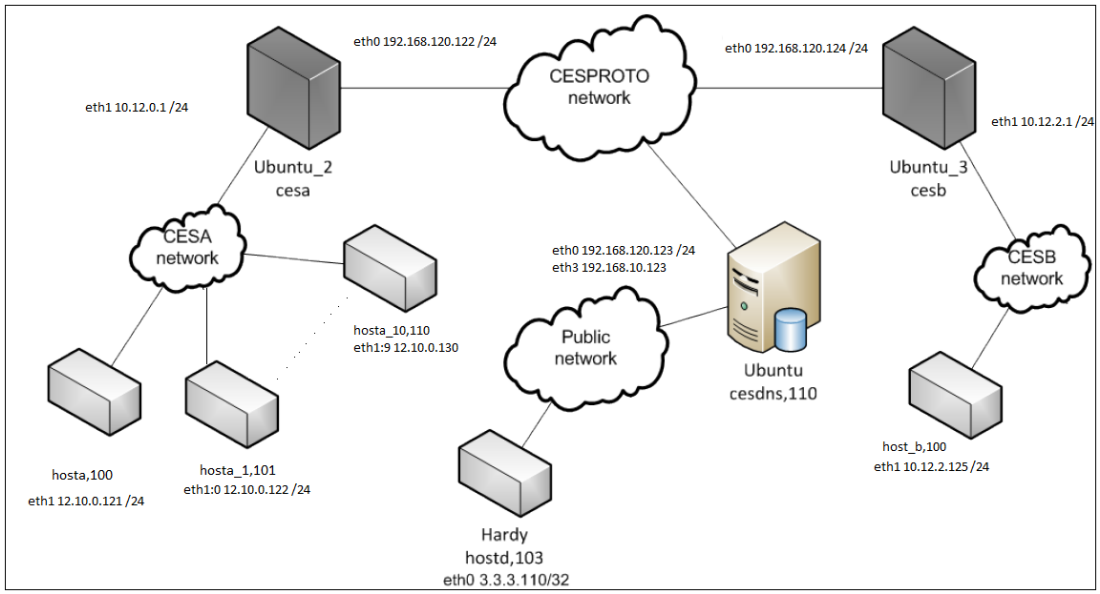

Figure 7.1: The prototype network

Due to the prototype network topology and physical connection between all virtual machines, care has been taken in terms of address allocation to make sure that Host A and CES A are in the same private network and Host B and CES B reside in another private network.

To configure the prototype network, each virtual machine must be configured individually. The network configuration can be made via */etc/network/interfaces* and */etc/resolv.conf* files as each virtual machine is running the Linux operating system. Additionally, a private host uses the CES device that it is registered in as the name server.

In Linux, it is straightforward to simulate several networks by adding routes to respective routing tables. Using this technique, the prototype network is divided into three different networks while they are actually residing in the same physical network. To give an example, the Host A virtual machine is configured so that all packets originated from Host A are routed through CES A. This is done by the following Linux command:

For IPv4: "route add default gw <CES A IPv4 Address> dev eth0"

For IPv6: "route -A inet6 add default gw <CES A IPv6 Address> dev eth0"

The same procedure is applied to other virtual machines in order to add corresponding routes. Furthermore, the necessary software and servers like Lynx are installed on the host machines.

The DNS server (Bind 9) is installed on the virtual machine that serves all DNS queries coming from other nodes in the prototype network. For the DNS machine, configuration is

created via */etc/bind/named.conf.local* and */etc/bind/zones/db.ces*. Moreover, all virtual machines are administered using virsh. Virsh provides the main and stable interface for controlling the virtualized operating system.

7.4 Network Elements

This subsection is dedicated to describing the network elements exploited to simulate different scenarios in the test environment.

Host: Host nodes are designed to simulate regular IP stack end systems. As described before, at first they are registered to the corresponding home CES. After that, the CES has their correct information including domain names, local addresses and IDs in its HRL. Each host can establish a connection to a target machine by executing a DNS lookup for the destination domain name, decoding received DNS responses and eventually dispatching data flows to IP addresses obtained from DNS replies.

CES: CES devices reside at the edge of private networks and exchange packets between the customer and core network in a similar way to NATs. More precisely, a CES passes DNS inquiries from private hosts to the DNS server. Upon the reception of a reply, it allocates a proxy IP address representing the destination host to the sender from its IP address pool, stores the corresponding mapping information in the connection table and sends a modified DNS reply back to the initiator. A CES can also negotiate with the remote CES about some parameters (e.g. source ID type) before admitting a new flow. The negotiation phase is fully policy controlled. Additionally, Customer Edge Switches unlike NAT devices provide bi-directional connectivity for private hosts that they serve. In other words, they accept the requests of connection invitation from other networks and deliver them to the targeted host.

DNS Server: From the prototype perspective, this network element resembles an authoritative name server. Therefore, it loads a master file at the beginning and generates a set of resource records using the file content. After initialization, it is ready to receive DNS queries and return appropriate replies.

7.5 External Libraries

Three external libraries are used in this implementation work. There are Scapy, Crypto and DNSPython.

7.5.1 Scapy

Scapy (available at http://www.secdev.org/projects/scapy/doc/installation.html) is an interactive packet manipulation program which is written in python. It allows a user to forge or dissect packets of different protocols, send and receive them, match queries and responses and many other operations. Consequently, the prototype uses this program to perform all packet manipulation and data transmission.

7.5.2 DNS Python

DNSPython is a DNS library for Python (available at http://www.dnspython.org/). Generating DNS queries and replies, sending them on wire and other DNS related operations in the prototype are done with DNSPython.

7.5.3 Python Cryptography

The Python cryptography toolkit (available at https://www.dlitz.net/software/pycrypto/) includes a wide variety of cryptographic functions. It is aimed to provide a simple, reliable and stable interface for existing cryptographic algorithms. In this thesis work, generating a public and private key pair, encryption and decryption of messages and creating and verifying signatures are done with the Python cryptography program.

7.6 Prototype Software Structure

The class diagram of the prototype is depicted in Figure 7.2.

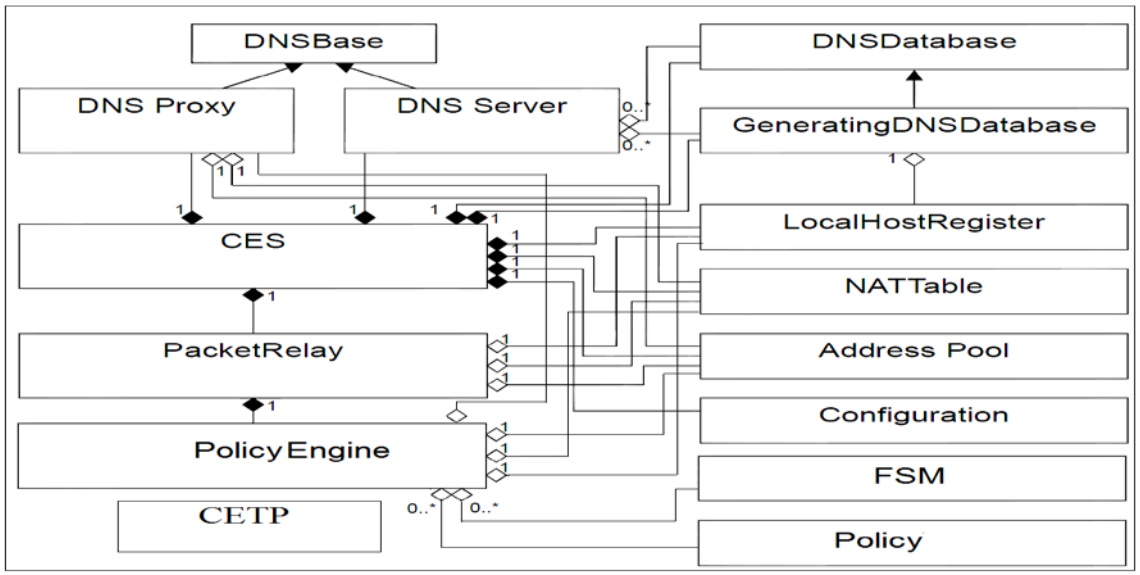

Figure 7.2: The class diagram of CES prototype

To implement the protocol and relevant methods, many changes have been made in the following classes of the original prototype: PacketRelay, LocalHostRegister, NATTable, GeneratingDNSDatabase, CES and DNSProxy. The CETP class is also added to the current prototype in order to forge and dissect CETP packets. Additionally, the policy control of CETP is implemented via the Policy Engine, Policy and FSM classes.

7.7 CETP Class Implementation

As all data transmissions in the prototype are done with Scapy, we decide to implement the CETP packet class by extending the Scapy's packet class and defining the packet parameters in the field_list attribute. The different parameters of a CETP packet are defined using built-in fields of Scapy. For example, as described in Chapter 5, the actual ID value can have variable length. Thus, the source and target ID fields are defined with Scapy's StrLenField whose length is specified by the ID length parameters.

Moreover, the optional control TLVs fit into a string whose length is calculated as follows:

$$TLV\ length = HeaderLength - (SourceIDLen + TargetIDLen + 8) \quad (1)$$

Given this, two separate functions are used for adding a control TLV to the field and parsing the TLV string captured from wire respectively.

Since signature TLV carries a signed packet header, in the adding control TLV functionality, it is added after all other TLVs. To generate a signature, first the signature TLV is filled with zeros (size of the value segment is fixed and is 128 octets) is added to the CETP header. Then, the packet header is hashed to the corresponding MD with SHA-1 and finally the MD is signed by the private key of the home CES. The payload part is also defined with a string field and encoded depending on the encapsulation type. On the other hand, the data can be decapsulated from the CETP payload string and further may be forwarded to the targeted private host. Additionally, there are three different methods in this class that model CETP over Ethernet, on top of IPv4 or IPv6. The CETP packet class is illustrated in Appendix B.

7.8 Testing Scenario

For testing purposes practically for checking the policy engine, ten hosts residing in the private network (i.e. corporate network A) register in the outbound CES (i.e. CES A). Each host is simulated using a virtual interface on Host A virtual machine and tied to the specific packet admission policy regardless of ID type. The outbound policies associated with the

local hosts are listed in Appendix C. Testing has been done in this manner because it is rather difficult to keep track of policy initialization while ID types are being changed during the session.

The testing process is divided into ten different parts. In each part, a certain policy is assigned to the server residing in the private network B behind the inbound CES (CES B) and ten hosts in the network A try to ping that server at the same time.

The described operation is done via a simple client program running on Host A. At the beginning of this program, the DNS lookup for the remote server (i.e. Host B) is performed on behalf of each private host and after the DNS resolution echo messages are directed to the destination element. This program runs ten times and each time Host B is tied to different packet admission policy. These policies are listed in Appendix D. As described in Chapter 6, information of each host is defined in a setting file. Therefore, before each testing part, the policy number of Host B in CES B's configuration file is modified manually. However, it seems easier to perform all testing scenarios at once and automatically, but we found this method more convenient to capture exchanged packets and analyze the results. The entire testing results are listed in ten separate tables and presented in Appendix E. According to the obtained results, the policy processing module of the prototype works as expected and we did not find any case that the prototype behaves strangely. In the following sections, some special test cases are described in order to provide an overall picture of testing.

7.9 Example Run of Inbound Policy Without any Requirements

It is possible that most of the applications running on a specific server require no extra information from a client preceding the connection establishment. This test example is designed to simulate such conditions. In this scenario, Host A behind CES A tries to connect to Host B behind CES B while the policy of Host B has empty Reqc policy vector. As previously mentioned, this prototype does its best to meet receiver's preferences while it is assumed that the sender tends to continue communication as he/she initiates the connection. The following polices are defined for Host A and Host B in the correspondent CES devices.

Policy of Host A:

```
Role=outbound
ID-Reqc=RANDOM
Reqc=
RR-Reqc=TOUT, IPv4_RLOC, IPv6_RLOC
Offerc=IPv4_RLOC
Available=FQDN,TOUT,IPv4_RLOC,Cookie,CA_Address,Signature
```

Policy of Host B:

```
Role=inbound
ID-Reqc=RANDOM
Reqc=
RR-Reqc=FQDN, Signature
Offerc=
Available=FQDN,TOUT,IPv4_RLOC,IPv6_RLOC,Cookie,CA_Address,Signature
```

Due to the DNS postponing model, upon reception of the DNS reply from the DNS server, CES A buffers this message and starts negotiations with the remote edge whose routing locator is obtained from the DNS inquiry. The policy engine uses the Creating oFSM algorithm to enforce Host A's policy. According to the ID-Reqc policy vector, the sender's ID type needs to be a unique random number. Since in this prototype hosts are identified with locally generated random numbers by default, a new ID type TLV is not required in the CETP control signaling. In addition, as the Reqc list is empty and Offerc policy vector contains IPv4 RLOC TLV, IPv4 RLOC TLV containing alternative IPv4 addresses of CES A is the only control information which is added to the CETP packet. Since this TLV also exists in the RR-Reqc policy vector, the TLV's operation is set to reliable response. Finally, CES A routes the forged CETP packet towards CES B. Figure 7.3 draws the message flows in this test case.

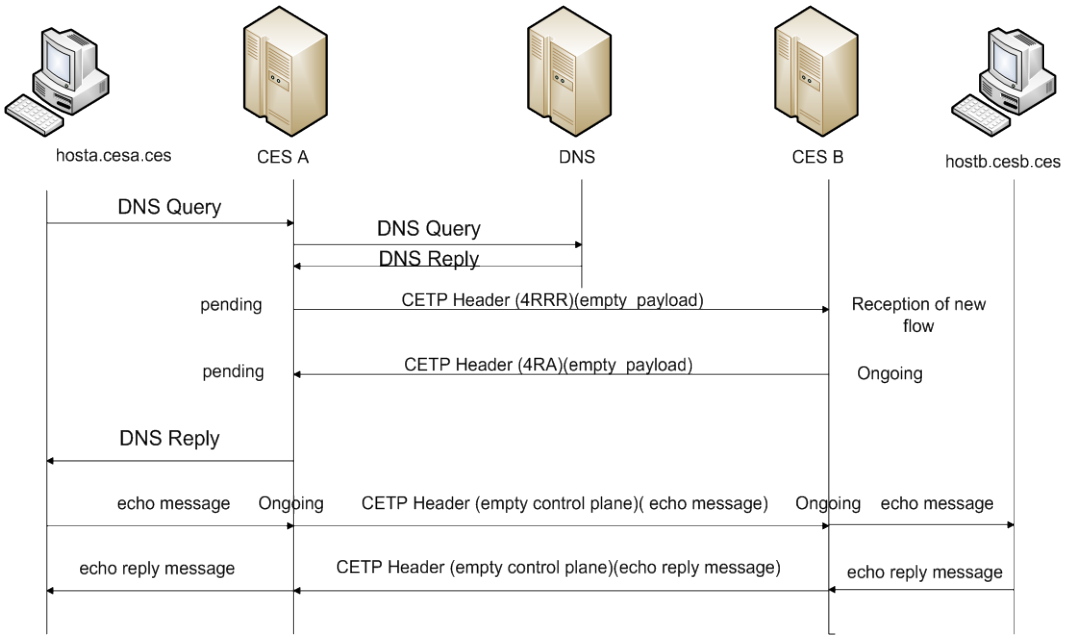

Figure 7.3: Example of successful connection setup for a destination without any requirements

On the inbound edge, the policy engine exploits the Creating iFSM procedure to process the received CETP message. From the Host B perspective, the preferred ID type is random. Thus, there is no need to add a changing ID request to the CETP reply message. As the

next step, the algorithm checks whether the Host B's requirements can be answered by the incoming packet. Since the Host B's Reqc policy vector does not include any control TLV, upon the first message the connection between two end nodes is established and relevant state information for the session is stored in CES B's connection table. To forge CETP reply packet, IPv4 RLOC TLV with the acknowledgement operation is added to the control signaling. On the other hand, as the Offerc list, same as the Reqc policy vector, is empty, no additional control TLV is incorporated into the message.

When CES A receives the CETP packet from CES B, it forwards the DNS reply to Host A and changes the session state to ongoing. Once Host A obtains the destination routing information, it sends an echo message towards Host B. On the way to Host B, CES A picks up the data packet, encapsulates it into a CETP message and eventually directs it to CES B.

On the opposite side, CES B extracts the original data packet from the CETP payload and further delivers it to Host B. The echo reply message is delivered to Host A in a similar way.

7.10 Example Run of Inbound Policy With A Specific ID Requirement

Some applications such as mission critical tasks only establish a connection with an end user who is identified by a more secure ID type than locally generated random numbers. Hence, if an inbound edge node receives a CETP packet with such sender's ID type, it requests the originator to change its ID to the preferred ID type. To test this scenario, following policies are assigned to Host A and Host B respectively.

Policy of Host A:

```
Role=outbound
ID-Reqc=RANDOM
Reqc=
RR-Reqc=TOUT, IPv4_RLOC, IPv6_RLOC
Offerc=IPv4_RLOC
Available=IPv4_RLOC,Cookie,CA_Address,Signature,FQDN
```

Policy of Host B:

```
Role=inbound
ID-Reqc=MOC
Reqc=Signature,CA_Address,FQDN,IPv4_RLOC,IPv6_RLOC
RR-Reqc =
Offerc=Signature,IPv4_RLOC,IPv6_RLOC
Available=FQDN,TOUT,IPv4_RLOC,IPv6_RLOC,Cookie,CA_Address,Signature
```

Since Host A policy is identical to the previous example, after the DNS resolution, the Creating oFSM forges a similar CETP message (containing the IPv4 reliable response TLV and empty payload) and forwards it to CES B.

Upon reception of the CETP message, CES B invokes the Creating iFSM to apply Host B's policy. According to the ID-Reqc policy vector, the sender is required to use the Mobile Operator Certificate (MOC) identifier. As local hosts are identified by Random IDs within this prototype, thereby the algorithm adds the changing ID type query for the MOC ID type to the CETP reply message. As previously mentioned in Chapter 5, it is assumed that only Cookie, Signature and CA address TLVs could company New ID type queries. Thus, the algorithm checks whether the control information is present in the Reqc and Offerc policy vectors. Since Host B's Reqc policy vector contains Signature and CA address TLVs; in addition to the New ID type TLV, Signature and CA address TLV queries are incorporated into the reply message. It is important to note as Signature also exists in the Offerc list, CES B provides its own signature within the value segment of the Signature TLV. The message flow exchanged between two endpoints is illustrated in Figure 7.4.

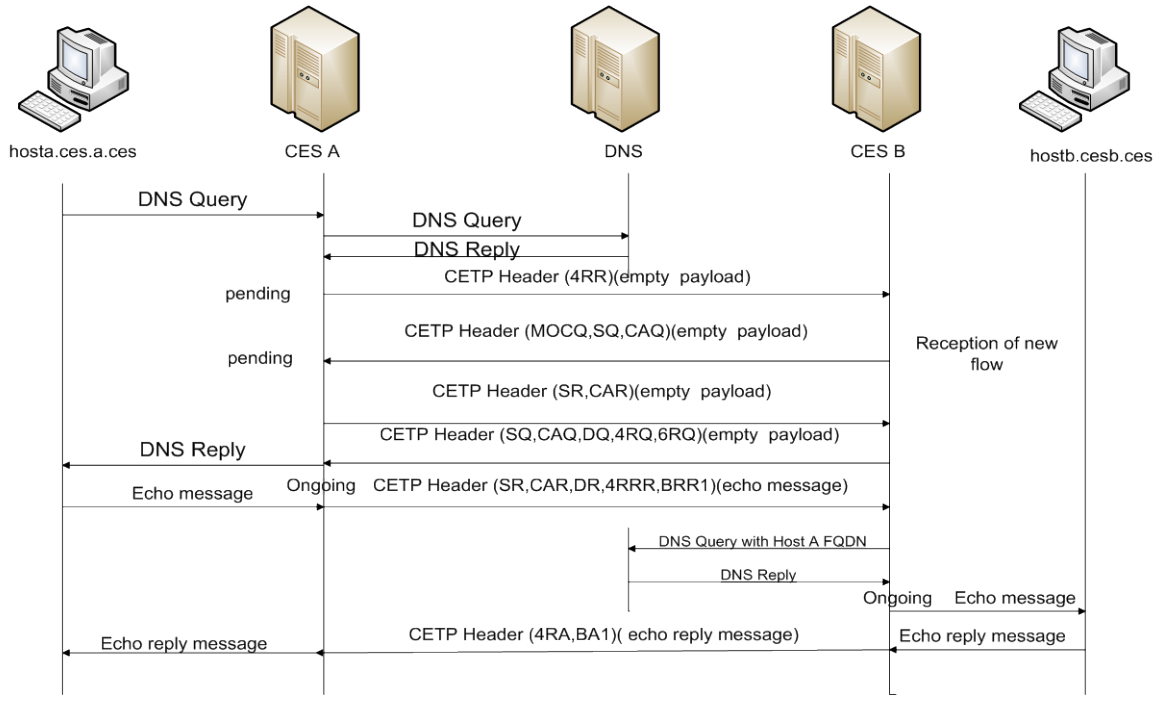

Figure 7.4: Example of successful connection setup for a destination with MOC ID requirement

Since the CETP message that arrives at CES A contains the peer's signature and New ID type Query, the Pending State Processing algorithm first examines the validity of the

signature and then checks whether Host A supports the MOC ID type or not. As the signature is valid and Host A also registers with the MOC identifier in CES A, the algorithm changes the local host's ID from the random value to the MOC in the corresponding connection state and builds the CETP reply message with the new source ID. The reply message carries the CES A's Signature and CA address that could be used for an assurance query.

On the Inbound edge, when the Creating iFSM procedure receives the message with expected sender's ID (MOC), it looks for replies to other Host A's requirements within the message control signaling. The packet only carries replies to CA address and Signature queries. Therefore, the algorithm creates a full requirement and returns it to CES A. Signature, CA Address, FQDN, IPv4_RLOC and IPv6_RLOC TLVs are specified as Host B's requirements on the Reqc list. Consequently, the full requirement message contains TLV queries for all these control TLVs. In addition, since Signature, IPv4_RLOC and IPv6_RLOC also exist on the Offerc list; CES B offers its signature, alternative IPv4 and IPv6 addresses in the value fields of signature, IPv4_RLOC and IPv6_RLOC TLVs respectively.

Upon reception of the second CETP packet from CES B, CES A by means of the Pending State Processing algorithm generates replies to the received TLV queries and stores them in the session's FSM pending TLV list. The reply TLV list contains Signature, CA Address and FQDN response TLV. It also includes the IPv4_RLOC reliable response TLV as this TLV is available in the RR-Reqc list as well as the Reqc policy vector. As IPv6_RLOC does not exist on Host A's available list and thereby could not be offered to the peer, Backoff code 2 (error code = 2) is added to the reply TLV list. In the next step, CES A forwards the pending DNS reply to Host A. After that Host A dispatches the echo message destined to Host B. The echo message on the way to Host B first goes through CES A. CES A creates a CETP message and incorporates the control information within the FSM pending TLV list to the packet's control signaling. The echo request is also fit into the payload. Finally, CES A sets the state of the session to ongoing and routes the CETP message to Host B.

On the opposite side, CES B using the Creating iFSM first checks whether Host B's queries could be replied with the control information in the message. The algorithm initially examines CES A's signature. As the signature is valid, it initiates a DNS lookup using Host A's domain name derived from the FQDN TLV. This is done to execute the return routability check on naming level. After the successful return routability check, it

comes to examine replies for remaining queries. Since the appropriate control information for CA address and IPv4_RLOC TLVs (one of the requested RLOC type) is received from CES A, the procedure sets the session's state to ongoing and derives the original data packet from the CETP payload. Then, it directs the echo message to Host B.

The CETP message containing the echo reply message along with the IPv4_RLOC and Backoff code 2 TLV acknowledgements is sent to CES A. Finally CES A delivers the echo reply to Host A.

7.11 Some Example of Unsuccessful Connection Establishments

In this section, some test scenarios in which connections did not end up to exchange of data flows between two end points are examined in more detail.

7.11.1 Example Run of Unsupported ID Requirement

Let us assume that unlike the previous example, Host A does not support the MOC ID type. Thus, when CES A receives the MOC ID query from the inbound edge and then it did not find the MOC ID of Host A in its HRL, a notification message is sent back to the remote edge and the session state information is deleted completely. At the end, CES A sends the DNS error reply to the local host as a notification of the unsuccessful connection setup. Figure 7.5 represents this testing scenario.

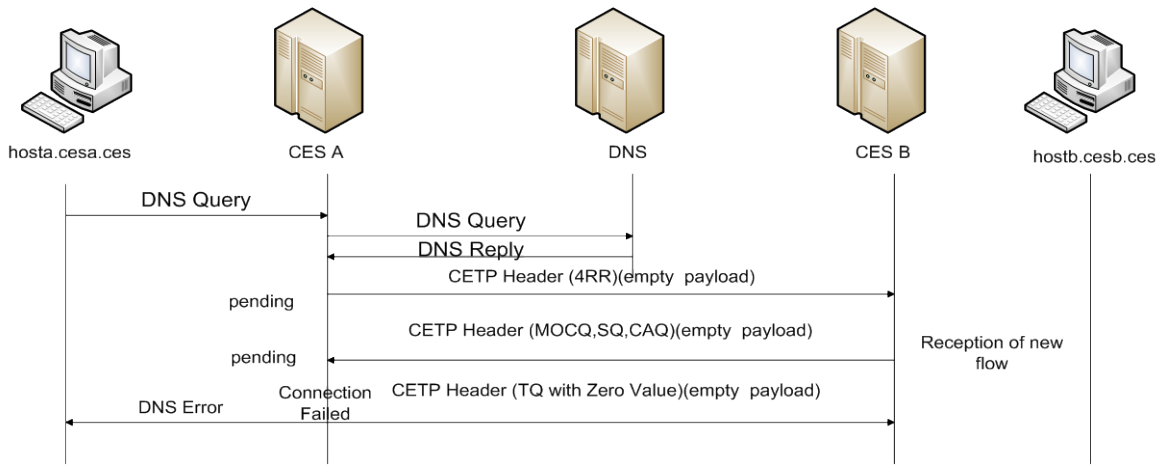

Figure 7.5: Example of unsuccessful connection establishment due to unsupported ID requirement

7.11.2 Example Run of Unsuccessful Return Routability check

For this test case, we again refer to the example of inbound policy with the MOC ID requirement. To simulate a failure in the naming level return routability check, Host A

registers in CES A's HRL with a fake domain name. Hence, when CES B executes a DNS query using Host A's domain name that is derived from the FQDN response TLV, the DNS server returns an error DNS reply. The reason is that no NS entry in the DNS database is matched to the requested domain name. Upon reception of the error message, CES B concludes that the sender is trying to masquerade as someone else and consequently blocks the received echo message destined to Host B and terminates the connection with the other end by sending a notification message. This action (i.e. terminating the connection) is taken in order to eliminate unauthorized access. On the opposite side, CES A removes the corresponding mapping state upon the notification message and informs Host A about the connection setup failure using an ICMP error message. The described scenario is depicted in Figure 7.6.

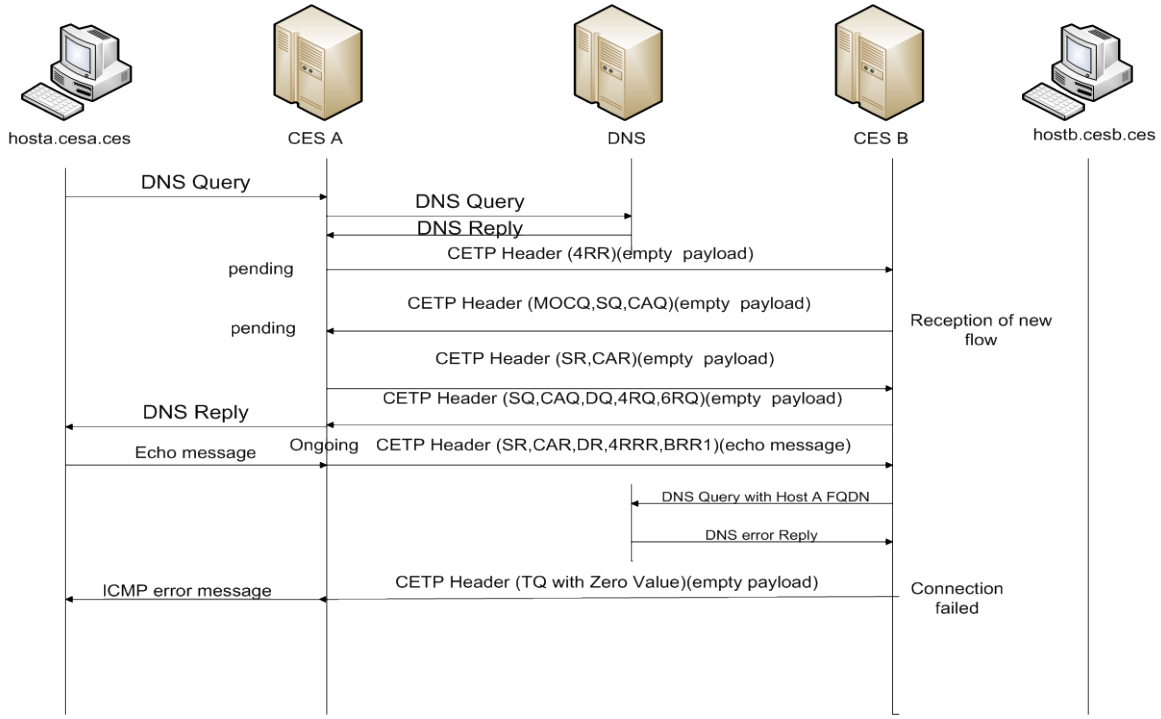

Figure 7.6: Example of unsuccessful connection setup due to unsuccessful return routability check

7.11.3 Example Run of Unsupported Receiver Requirement

As described in Chapter 6, if a destination host inquires for specific control information that a sender does not support according to its Available policy vector, the connection setup fails and the inbound edge terminates the session. This scenario is tested with the following Host A and Host B policies.

Policy of Host A:

```
Role=outbound
```

```
ID-Reqc=RANDOM
Reqc=TOUT
RR-Reqc=
Offerc=IPv4_RLOC,MAC_RLOC,IPv6_RLOC,FQDN
Available=FQDN,IPv4_RLOC,IPv6_RLOC,Cookie,CA_Address,Signature
```

Policy of Host B:

```
Role=inbound
ID-Reqc=RANDOM
Reqc=TOUT,Signature,CA_Address,IPv4_RLOC,Report_unwanted_msg
RR-Reqc=Signature
Offerc=Signature,TOUT
Available=FQDN,TOUT,Ether_Payload_encapsulation,IPv4_Payload_encap
sulation,Signature
```

Figure 7.7 shows the message flow of the unsuccessful connection setup in which destination host's requirements are not fulfilled completely. After the DNS resolution, CES A generates a CETP message containing TOUT TLV query, IPv4_RLOC, MAC_RLOC, IPv6_RLOC and FQDN response in order to enforce Host A policy and further directs it to CES B.

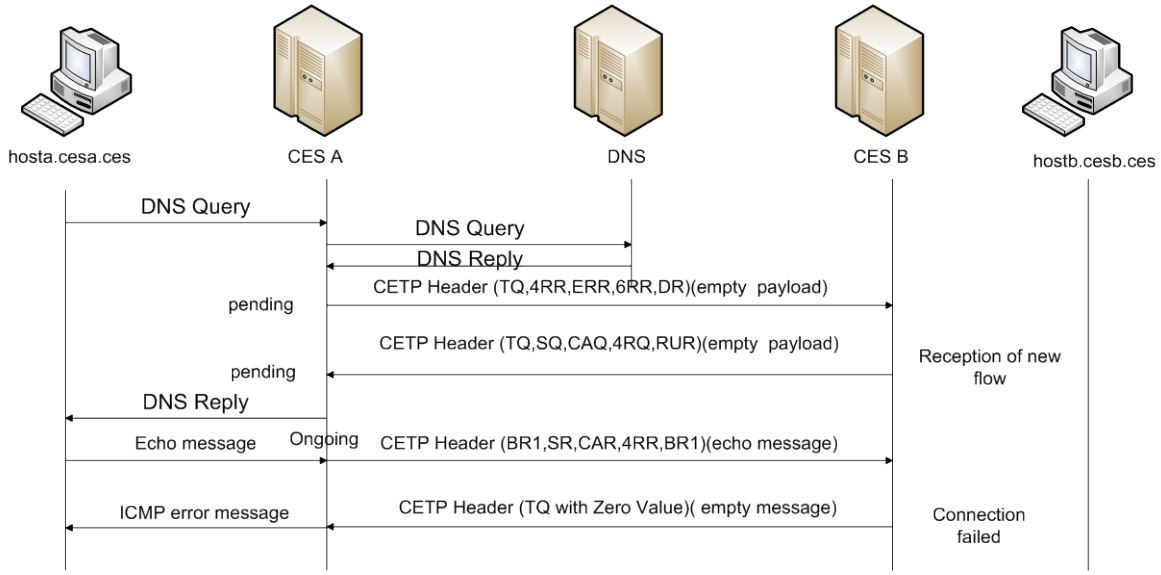

Figure 7.7: Example of unsuccessful connection establishment due to unsupported receiver requirements

On the inbound edge, CES B processes the received message and forges the CETP reply packet. As Host B requirements cannot be answered by the control signaling of the CETP message, CES B returns the full requirement message which carries TOUT, Signature, CA address, IPv4_RLOC and Report_unwanted_msg query. It is important to note that CES B provides its own signature and TOUT information to the peer within the message control signaling. To limit receiver's visibility before connection setup, replies to the sender's queries are not included in the full requirement.

Upon reception of the full requirement, CES A forwards the DNS reply to Host A and waits for data packets from the sender. Once the data flow is received from Host A, CES A encapsulates it into a CETP message and along with the control TLV replies sends back to the inbound edge. The control signaling of the reply CETP contains the Backoff code 2 (error code = 2), Signature, CA address, IPv4_RLOC and Backoff code 2 (error code = 1) response. The Backoff code 2 (error code = 2) response is added to the control signaling as a response to the TOUT query. The reason is that the TOUT TLV does not exist on Host A's Available policy vector and thereby Host A does not offer its TOUT information to the peer. Moreover, as Report_unwanted_msg TLV type is not defined in the current prototype, the CETP message also carries Backoff code 2 (error code = 1) response.

When CES B receives the CETP message containing the control information that is unable to fulfill all Host B's requirements and non-empty payload, the inbound edge node terminates the connection and returns a notification message to CES A.

On the outbound edge, CES A deletes the connection state upon the notification message and sends the ICMP error message towards Host A as a notification of connection setup failure.

7.11.4 Example Run in Disruptive Network

In real networks, it is highly probable that one of the parties involved in communication interrupts for a while and recovers once the disruptive factors are eliminated. To examine the described condition, let us assume the example of inbound policy without requirements. After successful connection establishment, we tried to simulate disruptive network by restarting either CES A or CES B. Let us consider a situation where CES A is restarted while other communicating nodes including Host A, CES B and Host B are running. In this case, CES A drops the data packets originated from Host A since it could not match the messages to any entry in the connection table. As the next step, Host A is notified about the connection failure via the ICMP error message from CES A. On the contrary, in the above test example if CES B goes down and restart, CES B discards data flows received from the other end (Host A) and returns a notification message (containing TOUT TLV with a zero value) towards CES A. It is worthy to mention that if in the described scenarios TCP connection is running between two hosts, the session will fail and cannot recover once each of CES devices is restarted. To recover broken connections, Host A should initiate communication and perform DNS lookup again.

7.12 Problems Throughout the Research Work

At the beginning of the thesis work, we decided to implement a rough prototype of the proposed protocol (CETP) so that we could get an overall picture of disparate protocol functionalities; because it was clear that correct specification of such a complicated functionality just on paper is not feasible. We implemented the packet structure several times to obviate defects spotted during prototyping. To give an example, in the early versions of the protocol, Q and R flag bits were a part of the fixed header and specified the operation of the entire CETP message rather than certain control information. The implementation work led us to shift these operation bits to the control TLV format. The reason was that these signaling flags have no meaning for the payload which is aimed to tunnel data flows; however, they are closely related to the CETP control signaling.

Although, at the first glance, this modification seems rather straightforward to implement, in practice this design decision resulted in considerable additional complexity of the protocol processing. During prototyping, we realized that systematic procedures need to be designed to handle signaling that is happening via CETP control signaling. Consequently, various algorithms were proposed and implemented before we came to an agreement for the final implementation of policy control of CETP that we have done within this research work. Furthermore, since the policy engine may operate in many ways, we prototyped and evaluated only a limited number of them (e.g. postponing DNS model) and the rest is left for further studies.

In addition, we encountered problems while we use Scapy as the primary tool for data transmission and packet manipulation in the prototype. The main issue emerged when we strived to modify Scapy classes and built-in functions for implementing the CETP packet and policy engine class. Due to lack of documentation, we started probing different aspects of the Scapy program by means of trial and error method. This approach can be applied easily, but it is very time consuming and might not reveal all exceptional cases.

Even though, implementing CETP packet via extending packet class was a challenging task due to alignment and padding considerations, undoubtedly the design, prototyping and testing of the policy processing algorithms were the most difficult part of this thesis. In the first place, we did not have a clear picture of the policy engine concept; however the essence of this module was proven to us. Moreover, testing the policy engine so that it can be claimed that the method works appropriately in all cases was a complicated task. It was assumed that the policy processing algorithms are generic and thereby must be capable of enforcing a wide variety of packet admission policies seamlessly. To obtain reliable

results, one hundred test cases were designed in such a way that the policy engine algorithms were examined thoroughly.

7.13 Evaluation of Results

In the developed prototype, CETP is used for inter CES communication. In addition to policy processing test cases, the prototype was tested with the well-known protocols: SSH, SFTP and HTTP. The testing results including packet captures indicate that the prototype operates exactly as we expected. In all testing scenarios, both end points reside in the private networks and thus data flows have to go through the CES devices before delivery to the destination host. Moreover, the different messages with the variable content size are sent from a private host to the other end in order to test fragmentation functionality. Whenever the message size after adding the CETP header exceeds the CES machine's MTU and the DF bit is also off, the prototype fragments the data packet into smaller pieces and encapsulates each segment using the Ethernet payload type. However, on the outbound edge, the fragmentation task is carried out properly, but the inbound CES faces the problem in reassembling the fragments and delivering the original packet to the local host. As it is very rare that a sender sets the DF bit in the IP header to zero, we did not resolve the problem within this thesis work and it is left for future implementation.

We also found out that in some situations especially during the negotiations it is possible that an edge node needs to send an empty CETP packet (with no control TLV and no payload) towards the peer. However, according to the flags field (C and R bits) of the CETP header, if the CETP packet does not carry any control information, its payload must be non-empty. As a consequence, with the current specification we cannot create an empty CETP message. To counter this, the definition of the CETP flags needs to be modified.

7.14 Discussion

Although, in the first CETP specification, the Q and R flag bits were indicators of the CETP message operation, the developed prototype made us to shift these operation bits to the control TLV field. We made this decision because these signaling flags are meaningless to tunneled data packets; rather, they are tied to the CETP control signaling concept.

Additionally, in the early version of CETP, only two different operations were defined (i.e. Query and Response) using the Q and R bits. Further studies led us to take advantage of all four possible combinations of the operation bits and defined Query, Response, Reliable

Response and Acknowledgment operations. These operations are aimed to assist the CES devices in enforcing various packet admission policies. In other words, the edge nodes using these operation bits can inquire for specific control information, reply to the received queries from the remote edges and acknowledge the received replies. This design decision had a significant impact on the way of defining policy vectors within this work. For instance, the Reqc list is defined to declare the control information that must be queried from the other end with the Query operation while the RR-Reqc list includes the control TLVs that an edge node expects to receive the acknowledgement messages upon sending them with the Reliable Response operation.

Since the purpose of policy control of CETP is to fulfill the receiver's needs rather than the sender's requirement, the policy processing cannot be done in the same way for both incoming and outgoing sessions. To differentiate between the policies for the inbound and outbound connections, the Role parameter was added to the policy definition.

Another issue that captured our attention during the implementation work was two different definitions of the length field in the control TLV format. After prototyping, we realized that however choosing an appropriate length field (one-byte or two-byte length parameter) according to the size of the value segment optimizes the packet size, it adds a considerable overhead and complexity to the protocol processing. To simplify CETP packet structure and avoid performance penalty, we decide to use the fixed-size length parameter for all control TLVs regardless of their value segment size in a future prototype.

As described before, the CETP packet structure is divided into three parts: the fixed header, optional control information and payload. The basic idea of introducing the payload and the control signaling in CETP messages is to send signaling information along a tunneled data flow in one packet rather in separate packets. As a result of this design decision, the number of packets transporting over the core network is reduced substantially. We also believe that with this packet format we can optimize the firewall processing overhead and become close to the optimum Round-trip time (RTT). However, for proving these claims we need to conduct performance testing on our prototype in a future work.

On the other hand, to make ID encoding flexible and facilitate implementing future extensions in CETP, ID fields and all control information are encoded in a TLV format. Due to this encoding, CETP can be used for transporting arbitrary types of ID with a variable size.

The robustness testing of the CETP development was out of scope of this work. In a separate research, for example we should introduce a separate TOUT for RLOC polling for robustness reasons.

We have noticed that the term CA in CETP is not quite correctly used. In a future version we may rename it to something more appropriate.

For optimization reasons, once oCES notices from the CETP response that it is unable to meet the receiver's requirements, instead of storing the reply and sending DNS response to host it could send DNS NACK to host and drop the session state. However, this is not done in the current prototype and left for a future work.

In the current prototype, upon the second message from iCES, outbound CES forwards the DNS reply to the private host. After this, the local host keeps sending data packets, although iCES might need to communicate with a third party for example for return routability check before establishing the connection. Thus data packets are received while iCES is conversing with a third party might be lost. To counter this problem, we can buffer data flows either in iCES or in oCES for limited period of time. This buffering would allow iCES to complete the negotiation phase and further deliver data flows to the destination host.

8 Conclusion

The purpose of this thesis was to develop CETP as a new tunneling protocol over an IP layer or directly on top of an Ethernet frame in order to transport data flows and control information from one customer network to another. This research work also includes the policy control of CETP. The developed prototype was evaluated with various testing scenarios. The test results prove that CETP can be used as means for edge-to-edge communication and signaling.

Within this thesis, Customer Edge Switching with the help of CETP provides global connectivity for hosts and servers residing in the private address spaces. This approach relieves the IPv4 address shortage by separating the dual role of an IP address as an identifier and locator at the edge node. In the test environment, we have assumed that both communicating parties reside in two different private networks and are identified by arbitrary type of IDs and located using private addresses. The reason is that in this work; we only studied inter-CES communication in that all traffic from/to the private host must go through the CES.

The Customer Edge Switching is targeted to eliminate address spoofing and improve trust and security among disparate customer networks. CETP assists CES devices to achieve this goal through performing a return routability check on naming and forwarding level and by signing the control information. Additionally, the CES by means of CETP enforces various packet admission policies to each ongoing session. More specifically, the CES takes advantage of the CETP control TLVs to declare the information which the destination host requires from the sender before successful connection establishment.

As this thesis has been done as a part of a larger project whose different components were implemented with Python, this prototype was also programmed with Python and thereby was not optimized in terms of performance. The performance of the current prototype can be enhanced considerably, if it is recoded with C / C++. The developed algorithms within this prototype are well-documented so that further functionalities can be added smoothly.

However, this prototype applies static admission policies to ongoing connections, in a future work; the policy initialization can be dynamic and changed according to the network condition and end-point's needs. We believe that this topic must be studied in separate research work. Furthermore, a future prototype instead of hardcoded certificates can exploit the public key infrastructure to derive required certificates and perform assurance queries.

The developed prototype uses some basic policy constraints to enforce different policies. However, we expect that in a future implementation each CES device can define more complex policy constrains that specify the acceptable and feasible policies systematically. For example, the policy definition is not really useful if its Offerc policy vector includes certain TLV type while that TLV does not exist on the Available policy vector. This means that the CES cannot reply to the queried control information, however, according to the Offerc policy vector the CES is allowed to provide the control information for that TLV type.

It is worthy to mention that this thesis work was integrated to the latest version of the CES program which is adapted to the open flow design and tested with a wide variety of test cases. On the contrary, the mobility and multi-homing functionalities for CETP are not implemented within this work and are left for the future implementation.

References

[1] C. Adamas. 1997. The CAST-128 Encryption Algorithm. RFC 2144.

[2] B. Agrawal, N. Kumar, and M. Molle. 2005. Controlling Spam Emails at the Routers. IEEE International Conference on Communication. Seuol, Korea.

[3] P. Albitz, and C. Liu. 1998. DNS and BIND. Third Edition. O'Reilly, Inc. ISBN 1-56592-512-2.

[4] R. Andreson. 2008. Security Engineering: A Guide to Building Dependable Distributed systems. Second Edition. John Wiley & Sons, Inc. ISBN 0-47-006852-6.

[5] F. Audet, and C. Jennings. 2007. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787.

[6] F. Audet, and C. Jennings. 2007. Network Address Translator (NAT) Terminology Behavioral Requirements for Unicast UDP. RFC 4787.

[7] S. Bellovin. 2003. ICMP Traceback Messages. RFC 2026.

[8] S. Bellovin. 1998. Security Problems in the TCP/IP Protocol. Computer Communication Review. Vol. 19:2. pp 32-48. ISSN 0146-4833.

[9] M. Bishop. 2004. Introduction to Computer Security. First Edition. Addison-Wesley Professional, Inc. ISBN 0-32-124744-5.

[10] M. Chapple. 2003. The GSEC Prep Guide: Mastering SANS GIAC Security Essentials. First Edition. John Wiley & Sons, Inc. ISBN 0-7645-3932-9.

[11] Z. Chen, S. Gue, R. Duan, and S. Wang. 2009. Security Analysis on Mutual Authentication against Man-in-the-Middle Attack. First International Conference on Information Science and Engineering. Nanjing, China.

[12] W. Cheswick, S. Bellovin, and A. Rubin. 2000. Firewalls and Internet Security: Repelling the Wily Hacker. Second Edition. Addison-Wesley. Inc, ISBN 0-20-163466-X.

[13] D. Clark. 30-31, 2007. MIT Communications Futures Program. Bi-annual meeting. Philadelphia. PA.

[14] E. Comer. 2006. Internetworking With TCP/IP – Principles, Protocols, And Architecture. Fifth Edition. Pearson Education, Inc. ISBN 0-13-187671-6.

[15] Computer Incident advisory Committee (CIAC). 1995. Advisory Notice F-08 Internet Spoofing and Hijacked Session Attacks [Retrieved on 17.08.2012]

Available: http://www.sans.org/reading_room/whitepapers/threats/introduction-ip-spoofing_959

[16] N. Doraswamy, and D. Harkins. 2003. IPSec. Second Edition. Prentice Hall, Inc. ISBN 0-13-046189-X.

[17] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. 2009. Locator/ID Separation Protocol (LISP). draft-farinacci-lisp-12.txt. Work-in-progress.

[18] Federal Information Processing Standards Publication 197. 2001. Announcing the Advance Encryption Standard (AES). United States National Institute of Standards and Technology (NIST). [Retrieved on 17.08.2012]

Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[19] M. Gritter, D. Cheriton. 2001. An Architecture for content routing support in the Internet. USITS'01 Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems. San Francisco, California, USA.

[20] F. Halsall. 2005. Computer Networking and the Internet. Fifth Edition. Addison-Wesley Professional, Inc. ISBN 0-321-26358-8

[21] D. Johnson, C. Perkins, and J. Arkko. 2004. Mobility Support in IPv6. RFC 3775.

[22] J. Kangasharju, and K. Ross. 2000. A Replicated Architecture for the Domain Name System. IEEE INFOCOM. Tel Aviv, Israel.

[23] R. Kantola. 2010. Implementing Trust-to-Trust with Customer Edge Switching. in press for AMCA in connection with AINA.

[24] R. Kantola. 2009. Customer Edge Switching – A Trust-to-Trust Architecture for the Internet. Helsinki University of Technology, Department of Communication and Networking.

[25] R. Kantola, N. Beijar, Y. Zheng, and M. Pahlevan. 2012. Customer Edge Traversal Protocol (CETP). [Retrieved 29.3.2012]

Available: http://www.re2eee.org

[26] C. Kaufman. 2005. Internet Key Exchange Protocol (IKE2). RFC 4306.

[27] F. Kurose, and W. Ross. 2005. Computer networking – A Top-Down Approach Featuring the Internet. Third Edition. Pearson Education, Inc. ISBN 0-321-26976-2.

[28] P. Leppäaho. 2012. Design of Application Layer Gateway for Collaborative Firewalls. M.Sc thesis. Comnet/Aalto. Espoo.

[29] J. Llorente. 2012. Private Realm Gateway. M.Sc thesis. Comnet/Aalto. Espoo.

[30] D. Maughan, M. Schertler, M. Schneider, and J. Turner. 1998. Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408.

[31] J. Mäepää, V. Andersson, G. Camarillo, and A. Keränen. 2010. Impact of Network Address Translator Traversal on Delays in Peer-to-Peer Session Initiation Protocol. IEEE Global Communication Conference (GLOBECOM). Miami, Florida, USA.

[32] M. Mealling, and R. Daniel. 2000. The Naming Authority Pointer (NAPTR) DNS Resource Record. RFC 2915.

[33] T. Mendyk-Krajewska, and Z. Mazur. 2010. Problem of Network Security Threats. Third Conference on Human System Interactions. Rzeszow, Poland.

[34] MEVICO Deliverable D 3,1. Packet Transport Architectures and Technologies for the Mobile Networks.Work in progress.

[35] P. Mockapetris. 1987. Domain Names – Implementation and Specification. RFC 1035.

[36] J. Mogul, and S. Deering. 1990. Path MTU Discovery. RFC 1191.

[37] A. Muller, G. Carle, and A. Klenk. 2008. Behavior and Classification of NAT Devices and Imlications for NAT Traversal. IEEE Network. Vol. 21:5. pp 14-19. ISSN 0890-8044.

[38] R. Rivest, and A. Shamir. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communication of the ACM. Vol. 21:2. pp 120-126. ISSN 0001-0782.

[39] M. Robshaw. 1995. Stream Ciphers. RSA Laboratories Technical Report TR-701. [Retrieved on 17.08.2012]

[40] J. Rosenberg. 2010. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245.

[41] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. 2008. Session Traversal Utilities for NAT (STUN). RFC 5389.

[42] B. Schneier. 1994. Description of a New Variable-Length Key, 64-Biet Block Cipher (Blowfish). Fast Software Encryption: Second International Workshop. Leuven, Belgium.

[43] P. Sirsuresh, M. Holdrege, and Lucent Technologies. 1999. IP Network Address Translator (NAT) Terminology and Consideration. RFC 2663.

[44] W. Stallings. 2003. Network Security Essentials Applications and Standards. Second Edition. Pearson Education, Inc. ISBN 0-13-035128-8.

[45] M. Tanase. 2003. IP Spoofing: An Introduction [Web Document]. SecurityFocus (Symantec Connect) [Retrieved on 17.08.2012]

[46] V. Velasco. 2000. Introduction to IP Spoofing [Web Document]. SANS Institute Infosec Reading Room. [Retrieved on 17.08.2012]

Available: http://www.symantec.com/connect/articles/ip-spoofing-introduction

[47] L. Virtanen. 2009. Communicating Globally Using Private IP Addresses. M.Sc thesis. Comnet/TKK, Espoo.

Available: ftp://ftp.rsasecurity.com/pub/pdfs/tr701.pdf

[48] P. Vixie, J. Dunlap, and J. Karels. Name Server Operations Guide for BIND Release 4.9.5. CSRG. Department of Electrical Engineering and Computer Science. University of California. Berkeley. California. [Retrieved on 17.08.2012]

Available: http://docs.freebsd.org/44doc/smm/10.named/paper.pdf

[49] Y. Zhang. 2007. Trust Management for Mobile Computing Platforms. Doctoral Dissertation. Comnet/TKK. Espoo. [Retrieved 29.3.2012]

Available: http://lib.tkk.fi/Diss/2007/isbn9789512291205/

[50] H. Zhao. 2003. Security in Telecommunications and Information Technology. First Edition. International Telecommunication Union.

[51] Y. Zheng, R. Kantola, and Y. Shen. 2012. Unwanted Traffic Control via Hybrid Trust Management. IEEE TrustCom. Liverpool, UK.

[52] G. Ziemba, D. Reed, and P. Traina. 1995. Security Considerations for IP Fragment Filtering. RFC 1858.

[53] E. Zwicky, S. Cooper, and B. Chapman. 2000. Building Internet Firewalls. Second Edition. O'Reilly Media, Inc. ISBN 1-56-592871-8.

Appendices

Appendix A

TLV	TLV-Code range	Query mnemonics	Response mnemonics	Reliable Response mnemonics	ACK mnemonic
New ID type	0x4×(0x1.0x7F)	IDQ	--	--	--
IPv4 Payload Encapsulation	0x204…0x207	4PEQ	4PER	4PERR	4PEA
IPv6 Payload Encapsulation	0x208…0x20B	6PEQ	6PER	6PERR	6PEA
Ethernet Payload Encapsulation	0x20C…0x20F	EPEQ	EPER	EPERR	EPEA
IPv4 RLOC	0x404…0x407	4RQ	4RR	4RRR	4RA
IPv6 RLOC	0x408…0x40B	6RQ	6RR	6RRR	6RA
Ethernet RLOC	0x40C…0x40F	ERQ	ERR	ERRR	ERA
TOUT	0x604…0x607	TQ	TR	TRR	TA
Cookie	0x60A…0x60B	--	--	CORR	COA
CA Address	0x60C…0x60F	CAQ	CAR	CARR	CAA
Domain information	0x610…0x613	DQ	DR	DRR	DA
Signature	0x614…0x617	SQ	SR	SRR	SA
Unexpected Message report	0x619…0x61B	--	UR	URR	UA
Backoff Code 0	0x681…0x683	--	BR0	BRR0	BA0
Backoff Code 1	0x685…0x687	--	BR1	BRR1	BA1

Appendix B

```
class CETP(Packet):
    name = "CETP"
    fields_desc = [
       # Version field
        BitField("version", 1, 3),
        #Flags Part
        FlagsField("flags", 0, 2, "CR"),
        #Header Length
        BitField("HL",0,11),
        #Length of whole packet
        ShortField("PayloadLength",0),
        #source identification
        ByteField("SourceType", 0),
        ByteField("SourceLen", 0),
        #destination identification
        ByteField("TargetType", 0),
        ByteField("TargetLen", 0),

        StrLenField("sourceID", "", length_from=lambda pkt:pkt.SourceLen),
        #Destination ID
        StrLenField("targetID", "", length_from=lambda pkt:pkt.TargetLen),
        # List of TLVs
       ConditionalField(StrLenField("TLVs", "", length_from=lambda pkt:pkt.HL-
      pkt.SourceLen-pkt.TargetLen-8),lambda pkt:pkt.flags == 2),
        #data plane in CETP
        StrField ("Payload", ""),
      ]
```

Appendix C

Outbound Policy Number One:
Role = outbound
ID-Reqc = RANDOM
Reqc=
RR-Reqc = TOUT, IPv4_RLOC, IPv6_RLOC
Offerc = IPv4_RLOC
Available= FQDN, TOUT, IPv4_RLOC, Cookie, CA_Address, Signature, Ether_Payload_encapsulation

Outbound Policy Number Two:
Role=outbound
ID-Reqc=RANDOM
Reqc= TOUT
RR-Reqc=TOUT, IPv4_RLOC, IPv6_RLOC
Offerc= Error_code1
Available=FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature, MAC_RLOC

Outbound Policy Number Three:
Role = outbound
ID-Reqc = RANDOM
Reqc = FQDN, IPv4_RLOC, Cookie, Error_code1
RR-Reqc= Error_code1
Offerc =
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature

Outbound Policy Number Four:
Role = outbound
ID-Reqc = RANDOM
Reqc = FQDN, MAC_RLOC, IPv6_RLOC, Cookie
RR-Reqc = IPv4_RLOC
Offerc = IPv6_RLOC, FQDN
Available= IPv4_Payload_encapsulation, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie

Outbound Policy Number Five:
Role = outbound
ID-Reqc = RANDOM
Reqc = TOUT
RR-Reqc =
Offerc = IPv4_RLOC, MAC_RLOC, IPv6_RLOC, FQDN
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature

Outbound Policy Number Six:
Role = outbound

ID-Reqc = RANDOM
Reqc = FQDN, Cookie
RR-Reqc =TOUT
Offerc = IPv4_RLOC, MAC_RLOC, IPv6_RLOC, FQDN
Available = Cookie, CA_Address, Signature

Outbound Policy Number Seven:
Role = outbound
ID-Reqc = RANDOM
Reqc = FQDN, IPv4_RLOC, IPv6_RLOC, Cookie
RR-Reqc = IPv4_RLOC, IPv6_RLOC
Offerc =IPv6_RLOC
Available = IPv4_RLOC

Outbound Policy Number Eight:
Role = outbound
ID-Reqc = RANDOM
Reqc = MAC_RLOC
RR-Reqc = MAC_RLOC, Signature
Offerc = IPv4_RLOC
Available = Ether_Payload_encapsulation, FQDN, TOUT, IPv4_RLOC, CA_Address, Signature

Outbound Policy Number Nine:
Role = outbound
ID-Reqc = RANDOM
Reqc = Signature, CA_Address, IPv6_RLOC
RR-Reqc = TOUT, Signature
Offerc = IPv4_RLOC, MAC_RLOC
Available = FQDN, TOUT, IPv4_RLOC, Cookie, Signature

Outbound Policy Number Ten:
Role = outbound
ID-Reqc = RANDOM
Reqc=
RR-Reqc = IPv6_RLOC
Offerc =
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature

Appendix D

Inbound Policy Number One:
Role = inbound
ID-Reqc = MAID
Reqc = Signature, CA_Address, FQDN, IPv4_RLOC, IPv6_RLOC,
Ether_Payload_encapsuation, IPv4_Payload_encapsulation
RR-Reqc = IPv4_RLOC, MAC_RLOC, TOUT
Offerc = FQDN, IPv4_RLOC, IPv6_RLOC
Available = Cookie, CA_Address, Signature, IPv4_RLOC, MAC_RLOC

Inbound Policy Number Two:
Role = inbound
ID-Reqc = RANDOM
Reqc = MAC_RLOC, IPv4_RLOC, IPv6_RLOC, Signature, Cookie,
IPv4_Payload_encapsulation
RR-Reqc = CA_Address
Offerc = TOUT
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature,
Ether_Payload_encapsulation

Inbound Policy Number Three:
Role = inbound
ID-Reqc = MOC
Reqc = CA_Address, FQDN, IPv6_RLOC, MAC_RLOC
RR-Reqc=
Offerc = FQDN
Available = FQDN, TOUT, CA_Address, Signature

Inbound Policy Number Four:
Role = inbound
ID-Reqc = RANDOM
Reqc = TOUT, Signature, CA_Address, IPv4_RLOC
RR-Reqc = Signature
Offerc = Signature, TOUT
Available = FQDN, TOUT, Ether_Payload_encapsulation, IPv4_Payload_encapsulation,
Signature

Inbound Policy Number Five:
Role = inbound
ID-Reqc= MAID
Reqc = IPv4_Payload_encapsulation, Cookie
RR-Reqc = Signature, TOUT
Offerc = IPv4_RLOC, IPv6_RLOC
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature

Inbound Policy Number Six:
Role = Inbound

ID-Reqc = RANDOM
Reqc = Signature, CA_Address, FQDN, IPv4_RLOC, IPv6_RLOC, Cookie
RR-Reqc = IPv4_RLOC, IPv6_RLOC
Offerc = MAC_RLOC
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie

Inbound Policy Number Seven:
Role = inbound
ID-Reqc = MOC
Reqc = Signature, CA_Address, FQDN, IPv4_RLOC, IPv6_RLOC
RR-Reqc=
Offerc = Signature, Error_code1, IPv4_RLOC, IPv6_RLOC
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature, MAC_RLOC

Inbound Policy Number Eight:
Role = inbound
ID-Reqc = RANDOM
Reqc =
RR-Reqc = FQDN, Signature
Offerc =
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature

Inbound Policy Number Nine:
Role = Inbound
ID-Reqc = MAID
Reqc = TOUT, Error_code1, FQDN, IPv4_RLOC, IPv6_RLOC
RR-Reqc = CA_Address, Signature
Offerc = Signature, TOUT
Available = FQDN, TOUT, IPv4_RLOC, IPv6_RLOC, Cookie, CA_Address, Signature

Inbound Policy Number Ten:
Role = inbound
ID-Reqc = RANDOM
Reqc = Signature, CA_Address, FQDN, IPv4_RLOC, IPv6_RLOC
RR-Reqc = TOUT
Offerc = CA_Address, Signature, IPv6_RLOC
Available = Cookie, CA_Address, Signature, TOUT

Appendix E

Testing Scenario Number 1:

In this test case, all local hosts in the private network A behind CES A tried to ping host B residing in the private network B behind CES B at the same time. For this scenario, host B applies inbound policy number one to all incoming connections.

Outbound Policy Number	Host Address	Setup Time (ms)	DNS Process Time(ms)	Ping Waiting Time (ms)	Expected Result	Failure Reason
1	10.12.0.121	59	77	216	Success	----------
2	10.12.0.122	X	Failure	X	Fail	Host does not support MAID ID type.
3	10.12.0.123	X	76	Failure	Fail	Not support any encapsulation type.
4	10.12.0.124	X	61	Failure	Fail	Not support FQDN TLV.
5	10.12.0.125	X	Failure	X	Fail	Host does not support MAID ID type.
6	10.12.0.126	X	64	Failure	Fail	Not supported any RLOC.
7	10.12.0.127	X	42	Failure	Fail	Not support signature.
8	10.12.0.128	X	Failure	X	Fail	Host does not support MAID ID type.
9	10.12.0.129	X	75	Failure	Fail	Not support CA address.
10	10.12.0.130	X	67	Failure	Fail	Not support any encapsulation type.

NB1: inbound CES based on its policy required MAID ID type.
NB2: each host starts its negotiation with random ID type.
NB3: hosts with 10.12.0.121, 10.12.0.125 and 10.12.0.128 IP addresses do not support MAID ID type.
NB4: In case of success, in this testing scenario set up CETP Connection could be completed in two round trips. In contrast, in case of failure three round trips are needed.
NB5: Setup CETP Connection Time = since the DNS reply is received from the DNS server until the modified DNS reply is forwarded to the host by outbound CES.
DNS process Time = since a host sends the DNS Query till the modified DNS reply is forwarded to the host by outbound CES.
ping waiting time = since a sends a ping request till it receives the ping reply from the server residing behind the inbound CES.

NB6: Number of sent TLVs by inbound CES = 5 (full requirement)

Testing Scenario Number 2:

In this test case, host B applies inbound policy number two to all incoming connections.

Outbound Policy Number	Host Address	Setup Time(ms)	DNS Process Time(ms)	Ping Waiting Time(ms)	Expected Result	Failure Reason
1	10.12.0.121	X	46	Failure	Fail	Not support IPv4 encapsulation type.
2	10.12.0.122	X	34	Failure	Fail	Not support IPv4 encapsulation type
3	10.12.0.123	X	43	Failure	Fail	Not support IPv4 encapsulation type.
4	10.12.0.124	X	43	Failure	Fail	Not support signature.
5	10.12.0.125	X	88	Failure	Fail	Not support IPv4 encapsulation type.
6	10.12.0.126	X	43	Failure	Fail	Not supported any RLOC.
7	10.12.0.127	X	43	Failure	Fail	Not support signature.
8	10.12.0.128	X	39	Failure	Fail	Not support any encapsulation type.
9	10.12.0.129	X	65	Failure	Fail	Not support any encapsulation type.
10	10.12.0.130	X	50	Failure	Fail	Not support any encapsulation type.

NB1: In case of success, in this testing scenario CETP connection setup could be completed in one round trip. In contrast, in case of failure two round trips are needed to complete negotiations.

Testing Scenario Number 3:

In this test case, host B applies inbound policy number three to all incoming connections.

Outbound Policy Number	Host Address	Setup Time (ms)	DNS Process Time(ms)	Ping Waiting Time(ms)	Expected Result	Failure Reason
1	10.12.0.121	X	48	Failure	Fail	Not support any RLOC.
2	10.12.0.122	28	39	131	success	----------

3	10.12.0.123	X	Failure	X	Fail	Host does not support MOC ID type.
4	10.12.0.124	X	45	Failure	Fail	Not support CA address.
5	10.12.0.125	74	100	198	success	----------
6	10.12.0.126	X	Failure	X	Fail	Host does not support MOC ID type.
7	10.12.0.127	X	48	Failure	Fail	Not support required TLVs.
8	10.12.0.128	X	46	Failure	Fail	Not support any RLOC type.
9	10.12.0.129	X	Failure	X	Fail	Host does not support MOC ID type
10	10.12.0.130	26	43	131	success	----------

NB1: inbound CES based on its policy required MOC ID type.
NB2: each host starts its negotiation with random ID type.
NB3: hosts with 10.12.0.123, 10.12.0.126 and 10.12.0.129 IP addresses do not support MOC ID type.
NB4: In case of success, in this testing scenario CETP connection setup could be completed in two round trips. In contrast, in case of failure three round trips are needed.
NB5: Number of sent TLVs by inbound CES = 7 (full requirement)

Testing Scenario Number 4:

In this test case, host B applies inbound policy number four to all incoming connections.

Outbound Policy Number	Host Address	Setup Time (ms)	DNS Process Time (ms)	Ping Waiting Time (ms)	Expected Result	Failure Reason
1	10.12.0.121	40	57	181	success	----------
2	10.12.0.122	45	56	177	success	----------
3	10.12.0.123	49	73	202	success	----------
4	10.12.0.124	X	59	Failure	Fail	Not support FQDN TLV.
5	10.12.0.125	82	106	255	success	----------
6	10.12.0.126	X	63	Failure	Fail	Host does not support ID type.
7	10.12.0.127	X	63	Failure	Fail	Not support time out TLV.
8	10.12.0.128	45	58	158	success	----------

9	10.12.0.129	44	62	Failure	Fail	Not support CA address.
10	10.12.0.130	45	63	189	success	----------

NB1: In case of success, in this testing scenario CETP connection setup could be completed in one round trip. In contrast, in case of failure, two round trips are needed to complete negotiations.
NB2: We just consider successful cases that have been completed successfully.
NB3: Number of sent TLVs by inbound CES = 4 (full requirement)

Testing Scenario Number 5:

In this test case, host B applies inbound policy number five to all incoming connections.

Outbound Policy Number	Host Address	Setup Time (ms)	DNS Process Time (ms)	Ping Waiting Time (ms)	Expected Result	Failure Reason
1	10.12.0.121	X	51	Failure	Fail	Not support IPv4 encapsulation type.
2	10.12.0.122	X	Failure	X	Fail	Host does not support ID type.
3	10.12.0.123	X	46	Failure	Fail	Not support IPv4 encapsulation type.
4	10.12.0.124	152	172	468	success	----------
5	10.12.0.125	X	Failure	X	Fail	Host does not support ID type.
6	10.12.0.126	X	44	Failure	Fail	Not support any required TLV.
7	10.12.0.127	X	44	Failure	Fail	Not support any required TLV.
8	10.12.0.128	X	Failure	X	Fail	Host does not support ID type.
9	10.12.0.129	X	58	Failure	Fail	Not support IPv4 encapsulation type.
10	10.12.0.130	X	49	Failure	Fail	Not support IPv4 encapsulation type.

Testing Scenario Number 6:

In this test case, host B applies inbound policy number six to all incoming connections.

Outbound Policy Number	Host Address	Setup Time(ms)	DNS Process Time(ms)	Ping Waiting Time(ms)	Expected Result	Failure Reason
1	10.12.0.121	19	37	146	success	----------
2	10.12.0.122	15	27	134	success	----------
3	10.12.0.123	24	42	154	success	----------
4	10.12.0.124	X	46	Failure	Fail	No signature TLV.
5	10.12.0.125	60	84	21	success	----------
6	10.12.0.126	X	32	Failure	Fail	Not support any RLOC.
7	10.12.0.127	X	37	Failure	Fail	Not support IPv4 encapsulation.
8	10.12.0.128	18	34	139	success	----------
9	10.12.0.129	X	33	Failure	Fail	Not support IPv4 encapsulation
10	10.12.0.130	20	37	156	success	----------

NB1: In case of success, in this testing scenario CETP connection setup could be completed in one round trip. In contrast, in case of failure, two round trips are needed to completed negotiation.
NB2: I just consider successful cases that have been completed successfully.
NB3: Number of sent TLVs by inbound CES = 7 (full requirement)

Testing Scenario Number 7:

In this test case, host B applies inbound policy number seven to all incoming connections.

Outbound Policy Number	Host Address	Setup Time(ms)	DNS Process Time(ms)	Ping Waiting Time(ms)	Expected Result	Failure Reason
1	10.12.0.121	104	119	251	success	----------
2	10.12.0.122	106	120	247	success	----------
3	10.12.0.123	X	Failure	X	Fail	Host does not support MOC ID type.
4	10.12.0.124	X	93	Failure	Fail	Not support signature.
5	10.12.0.125	149	173	276	success	----------
6	10.12.0.126	X	Failure	X	Fail	Host does not support MOC ID type.

7	10.12.0.127	X	100	Failure	Fail	Not support any of required control TLVs except RLOC.
8	10.12.0.128	83	94	199	success	----------
9	10.12.0.129	X	Failure	X	Fail	Host does not support MOC ID type.
10	10.12.0.130	103	123	253	success	----------

NB1: inbound CES based on its policy required MOC ID type.
NB2: each host starts its negotiation with random ID type.
NB3: hosts with 10.12.0.123, 10.12.0.126 and 10.12.0.129 IP addresses do not support MOC ID type.
NB4: In case of success, in this testing scenario CETP connection setup could be completed in two round trips. In contrast, in case of failure three round trips are needed.
NB5: Number of sent TLVs by inbound CES = 6 (full requirement)

Testing Scenario Number 8:

In this test case, host B applies inbound policy number eight to all incoming connections.

Outbound Policy number	Host Address	Setup Time(ms)	DNS process Time(ms)	Ping waiting Time(ms)	Expected result	Failure reason
1	10.12.0.121	18	31	106	success	----------
2	10.12.0.122	15	31	106	success	----------
3	10.12.0.123	16	32	121	success	----------
4	10.12.0.124	20	35	120	success	----------
5	10.12.0.125	59	86	185	success	----------
6	10.12.0.126	16	36	114	success	----------
7	10.12.0.127	20	32	108	success	----------
8	10.12.0.128	11	24	100	success	----------
9	10.12.0.129	43	55	133	success	----------
10	10.12.0.130	18	30	107	success	----------

NB1: In case of success, in this testing scenario CETP connection setup could be completed in one round trip. In contrast, in case of failure, two round trips are needed to completed negotiation.
NB2: Number of sent TLVs by inbound CES = 0 (full requirement)

Testing Scenario Number 9:

In this test case, host B applies inbound policy number nine to all incoming connections.

Outbound Policy number	Host Address	Setup Time (ms)	DNS process Time(ms)	Ping waiting Time(ms)	Expected result	Failure reason
1	10.12.0.121	53	68	Failure	Success	----------
2	10.12.0.122	X	Failure	X	Fail	Host does not support MAID ID type.
3	10.12.0.123	X	65	Failure	Fail	Unknown required TLV error_code1.
4	10.12.0.124	X	87	Failure	Fail	Unknown required TLV error_code1.
5	10.12.0.125	X	Failure	X	Fail	Host does not support MAID ID type.
6	10.12.0.126	X	72	Failure	Fail	Unknown required TLV error_code1.
7	10.12.0.127	X	69	Failure	Fail	Unknown required TLV error_code1.
8	10.12.0.128	X	Failure	X	Fail	Host does not support MAID ID type.
9	10.12.0.129	X	64	Failure	Fail	Unknown required TLV error_code1.
10	10.12.0.130	X	73	Failure	Fail	Unknown required TLV error_code1.

NB1: inbound CES based on its policy required MAID ID type.
NB2: each host starts its negotiation with random ID type.
NB3: hosts with 10.12.0.121, 10.12.0.125 and 10.12.0.128 IP addresses do not support MAID ID type.
NB4: In case of success, in this testing scenario CETP connection setup could be completed in two round trips. In contrast, in case of failure three round trips are needed.
NB5: Number of sent TLVs by inbound CES = 7 (full requirement)
There is not any successful connection in this scenario.

Testing Scenario Number 10:

In this test case, host B applies inbound policy number ten to all incoming connections.

Outbound Policy number	Host Address	Setup Time(ms)	DNS process Time(ms)	Ping waiting Time(ms)	Expected result	Failure reason
1	10.12.0.121	44	55	176	success	----------

2	10.12.0.122	45	59	18	success	----------
3	10.12.0.123	42	60	197	success	----------
4	10.12.0.124	X	55	Failure	Fail	Not support signature.
5	10.12.0.125	80	105	254	success	----------
6	10.12.0.126	X	52	Failure	Fail	Not support FQDN.
7	10.12.0.127	X	58	Failure	Fail	Not support CA address.
8	10.12.0.128	45	65	167	success	---------
9	10.12.0.129	X	58	Failure	Fail	Not support CA address.

NB1: In case of success, in this testing scenario CETP connection setup could be completed in one round trip. In contrast, in case of failure two round trips are needed to completed negotiation.
NB2: Number of sent TLVs by inbound CES = 5 (full requirement)