NTNU

Norwegian University of
Science and Technology

Security Analysis of Future Internet Architectures

Carlos Ballester Lafuente

Master in Security and Mobile Computing

Submission date: June 2010

Supervisor: Danilo Gligoroski, ITEM

Co-supervisor: Tuomas Aura, Aalto University - School of Science and Technology

Norwegian University of Science and Technology
Department of Telematics

Problem Description

The objective of the Thesis is to study and to possibly reveal the potential security flaws/problems that Future Internet architectures (and protocols designed in order to implement those) might have. Future Internet architecture projects that will be studied are, but not limited to, PSIRP, Accountable Internet Protocol, Networking Named Content, DONA, Postcards-from-the-edge and Scribe.
Most of these architectures focus on a identifier-location split and make use of paradigms such as publish-subscribe.
Claims are made that these new architectures can provide strong protection against DoS and DDoS attacks. Thesis will investigate these claims in order to try to verify whether they are right or not. Analysis based on RFCs and documentation, possibly confirmation of findings by checking code or testing it on a live environment when possible.

Assignment given: 10. February 2010
Supervisor: Danilo Gligoroski, ITEM

Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences

Degree Programme of Computer Science and Engineering

ABSTRACT OF THE MASTER'S THESIS

Author: Carlos Ballester Lafuente

Title: Security Analysis of Future Internet Architectures

Date: June 22, 2010 **Language:** English **Pages:** 5 + 55

Professorship: Data Communications Software **Code:** T-110

Supervisors: Professor Tuomas Aura
Professor Danilo Gligoroski

Abstract:

During the last decades, Internet has evolved from *host-centric* to *information-centric* in the sense that it is information and data what matters, regardless of where it is located. Meanwhile, Internet's architecture still remains the same as it was in its origins and still focuses on host-to-host communication, putting too much emphasis on the *"where"* rather than putting it on the *"what"*.

Original Internet's architecture also introduces several security flaws such as DoS and DDoS, spoofing and spam, and other non-security related problems such as availability or location dependence related issues. In order to address these issues, several new architectures and protocols have been proposed. Some of them aim at redesigning totally the architecture of Internet from scratch, while others aim at improving it without redesigning it totally.

The aim of this Master Thesis is to analyze these new protocols and architectures from a security point of view in order to determine whether the security claims made are true or not. The security analysis is made based on RFCs, technical papers and project deliverables. The results obtained have uncovered some security issues in several of the new protocols and architectures and have provided some insight into further improving them.

Keywords: Future Internet, architectures, protocols, security analysis

Acknowledgements

This thesis was conducted at the Department of Computer Science and Engineering of the School of Science and Technology of Aalto University, Espoo, Finland.

I would like to express my gratitude and heartiest thanks to my supervisors Professor Tuomas Aura from the School of Science and Technology of Aalto University and Professor Danilo Gligoroski from the Norwegian University of Science and Technology (NTNU) for their support, guidance, suggestions and collaboration through all my Master Thesis work.

Also, I would like to thank Mikko Särelä, M.Sc. (Tech.) from Ericsson Nomadic Labs for the suggestions and advices given and for the time dedicated to read through my material and to give feedback on it.

Finally, I would like to thank and to dedicate this Thesis work to my parents and my brother for all the support given through these two years of Master studies, to my family, and to my friends, which have made my time in Norway and Finland a great and very enjoyable time.

Espoo, June 22, 2010

Carlos Ballester Lafuente

Contents

1 Introduction **1**

2 Future Internet Protocols and Architectures **3**

2.1 Publish-Subscribe Internet Routing Paradigm 4

2.1.1 Publish-Subscribe Paradigm 4

2.1.2 Architecture Components 5

2.1.3 Design Considerations: mobility and security 9

2.2 Scribe 10

2.3 Accountable Internet Protocol 11

2.4 Networking Named Content 14

2.5 Layered Naming Architecture 16

2.6 Internet Indirection Infrastructure 18

2.7 Data-Oriented Network Architecture 20

2.8 Postcards from the Edge 22

3 Methodology **25**

4 Security Analysis **27**

4.1 PSIRP Security Analysis 27

4.1.1 Forwarding 28

4.1.2 Possible Solutions 33

4.2 Scribe Security Analysis 34

4.3 AIP Security Analysis 36

4.3.1 Spoofing issues 36
4.3.2 Forwarding 37
4.3.3 Forcing Shut-Off by Replay techniques 38
4.4 NNC Security Analysis 38
4.4.1 Content Spoofing and Trust Management 39
4.4.2 Forwarding 40
4.5 LNA Security Analysis 41
4.5.1 Name Resolution Layers 41
4.5.2 Forwarding 42
4.6 I3 Security Analysis 42
4.7 DONA Security Analysis 44
4.8 Postcards from the Edge Security Analysis 44
4.9 Architectural Similarities 46

5 Discussion **47**

6 Conclusion and Future Work **50**
6.1 Conclusion 50
6.2 Future Work 51

Chapter 1

Introduction

> *"**The Internet only just works.** The core Internet protocols have not changed significantly in more than a decade, in spite of exponential growth in the number of Internet users and the speed of the fastest links. The requirements placed on the net are also changing, as digital convergence finally occurs."*, M. Handley [13]

Internet was designed with the goal of resource sharing on mind, thus the original model of Internet's architecture aimed at establishing an end-to-end communication between two hosts. Nowadays, Internet has reached a point where what matters is the content accessed, regardless of where it is physically located, but still, Internet's model remains unchanged and focuses on the ***"where"*** rather than on the ***"what"***. This communication model introduces several issues related with availability, security and location dependence.

The current IP layer has several flaws that allow for DoS and DDoS attacks, spam sending, and address hijacking and spoofing being easily undetected. Many solutions have been proposed in order to solve these issues, many of them being fixes of the current IP layer. However, these kind of solutions require either implementing new mechanisms that are too intricate in their own nature, involving external sources to trust into, or require an effort from the operators in order to work properly.

While these solutions being a best effort and technologically its best and totally plausible, the problem lies in building them over the existing IP layer which is somehow trying to fix something broken rather than building from scratch something that implements everything properly from the beginning.

During the past few years, many proposals for new architectures and protocols in order to solve these previously mentioned issues have been developed. The aim of many of these new architectures and protocols is to address these issues from a completely new point of view, instead of trying to fix the current Internet [3, 14, 8, 2, 5, 11, 12, 29, 7], while some others are proposed as partial changes or overlay solutions based on the current Internet architecture [19, 4, 16, 21, 25]. Many of these new Future Internet projects are still ongoing under the European Union Seventh Framework Programme (FP7) initiative [20, 26, 18, 10].

The objective of this Thesis is to study and to possibly reveal the potential security flaws/problems that Future Internet architectures (and protocols designed in order to implement those) might have, focusing more concretely in those connected with DoS/DDoS. The analysis of these new architectures and protocols will be done based on RFCs, project deliverables and/or technical papers.

The rest of the document is structured as follows. In Chapter 2 the most relevant Future Internet architectures and protocols are presented and explained. In Chapter 3 the methods used in order to select the architectures and to conduct the security analysis of these are defined. Chapter 4 conducts the security analysis and outlines some possible solutions to some the problems found. Chapter 5 discusses the outcome of the security analysis and finally, Chapter 6 summarizes the findings of this Thesis and describes the possible future work related with it.

Chapter 2

Future Internet Protocols and Architectures

> *"Current Internet standards bodies and core Internet protocols are ossifying to such an extent that security and performance requirements for next-generation applications will require a totally new base platform. If current Internet base protocols survive, it will be as a substrata paved over by new-generation smarter ways of connecting."*, Ian Peter, Ian Peter and Associates and the Internet Mark 2 Project

This first chapter introduces the *Future Internet* protocols and architectures that will be analyzed in Chapter 4.

For each of the architectures whose security properties will be analyzed, a technical description on how they work, their desired properties, their objectives and their approach to the problem is presented in order to provide a better understanding for the reader when performing the security analysis. The technical description is based either in RFCs, technical papers and/or project deliverables.

While the description aims at providing a general picture of the architecture, a deeper level of detail and understanding can be achieved by consulting the references at the end of this document, and the reader is encouraged to do so.

2.1 Publish-Subscribe Internet Routing Paradigm

Publish-Subscribe Internet Routing Paradigm, in short PSIRP [26], aims to redesign the current Internet host-centric approach and to move to a new information-centric approach. In order to achieve this goal, PSIRP moves away from the current sender-driven Internet model towards a more receiver-driven model, on which publish-subscribe paradigm plays a central role.

The basis of PSIRP is that everything is considered information, and all the information is uniquely identified with a label to be used by the rendezvous system to match between publishers and subscribers. This label or identifier is called the **rendezvous identifier** (RId).

Individual pieces of information can be grouped together in a logical way using **scope identifiers** (SId), being those a subclass of RIds. This allows to build information networks in order to apply access control rules (i.e. a collection of pictures that should only be accessed by family members can be grouped under the scope "family"). Furthermore, by using the so called **algorithmic RIds**, which are RIds generated by a well know function or algorithm, information within information networks can be grouped into smaller groups called information collections, which are several information units that all together belong or represent another single information item (i.e. all the pictures belonging to a same picture album). Also in a similar way, information networks can also be grouped using algorithmic SIds.

In the next subsections, basic notions of publish-subscribe model, PSIRP architecture components, and design considerations will be explained.

2.1.1 Publish-Subscribe Paradigm

Publish-Subscribe paradigm is based in the asynchronous sending of messages where the sender is not intended to send data to an specific receiver. In publish-subscribe systems, receivers express their interest in certain data and they only receive the data they expressed an interest into, without the burden of having to know who is the specific sender. This decoupling between senders and receivers allows for higher scalability and topology dynamism.

The vast majority of publish-subscribe architectures rely on some intermediate elements (i.e. rendezvous systems or brokers), which are in charge of matching interests and publications in a decentralized and distributed manner.

Publish-subscribe systems can be further classified into two classes: topic-based and content based. In topic-based systems the messages are published to topics, which act as a sort of logical channels with a common topic and subscribers join the channels to receive the messages they are interested in. Subscribers in this case receive all the messages published to the channel and all the subscribers receive the same messages.

In the other hand, content-based systems only deliver messages to the subscriber if the content or attributes of the message match some interest definitions imposed by the subscriber. This second class of publish-subscribe system allows for the subscribers to define their interest through some parameters, making sure they only receive what they want, while in topic-based approach they might be messages of little or no relevant interest for some of the subscribers.

2.1.2 Architecture Components

PSIRP architecture is based on a non-layered approach called **"the component wheel"**. In this approach, the outermost part of the wheel is occupied by the APIs, while the center is the so called blackboard. The middle area between the APIs and the blackboard hosts the core components of the architecture, such as the rendezvous, the forwarding, the topology management and formation and the like, also leaving open possibilities for the addition of new components. This architectural approach is depicted in Figure 2.1.

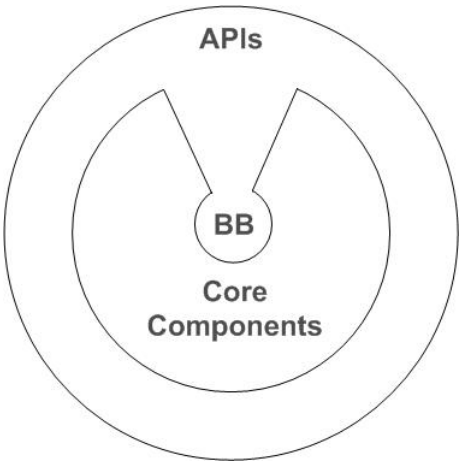

Figure 2.1: PSIRP's component wheel architecture.

Following, each of the core components of PSIRP's architecture is described in more detail.

As stated previously, several types of identifiers are used within PSIRP's architecture. While some of them have already been described in Sec. 2.1, there are still other identifier types that should be introduced, namely **application identifiers** (AId), **forwarding identifiers** (FId) and **algorithmic identifiers** (AlgId). AIds are human readable identifiers used by applications and can appear in several different flavors. They are mapped to RIds through different mechanisms such as search engines, directory services and the like. FIds are used to establish a delivery path between publishers and subscribers and will be explained in more detail when introducing the forwarding component of PSIRP. AlgIds are identifiers that have been generated by a well known algorithm and they implement information collections. They are used to create relations between identifiers and allow for many different functions such as subscription management and aggregation, caching, coding, flow control and many others. Describing this functions in detail is out of the scope of this section and more information on those can be found in [26].

The second component of PSIRP's architecture are the **helper functions**,which are classified into three different categories:

- network management functions, which are used to collect network information about resources, performance and others,
- remote service functions, which are used for segmentation, forward error correction (FEC), re-coding of content and caching, and
- host service functions, which are tools intended for the hosts to ease their task of dealing with the network use.

The third architectural component is the **rendezvous** component, whose main task is to match subscriber interests with publisher data matching those interests. Rendezvous nodes (RN) are grouped into rendezvous networks, and rendezvous nodes can contain several rendezvous points.

Rendezvous networks interconnection can be done both by means of central control entities or in a distributed fashion using virtual overlays. In the last case, there should be one root rendezvous node in each network which is in charge of communication with its peer root RNs in the other networks. For the rendezvous system to operate correctly a bootstrapping mechanism is needed. This mechanism works in such a way that every end-node in the

system knows at least one rendezvous node, all the RNs inside the same network know the rest of RNs within that network so they are able to establish a network topology, and root RNs from different networks find their peers and are able to form the virtual overlay as shown in Fig. 2.2.

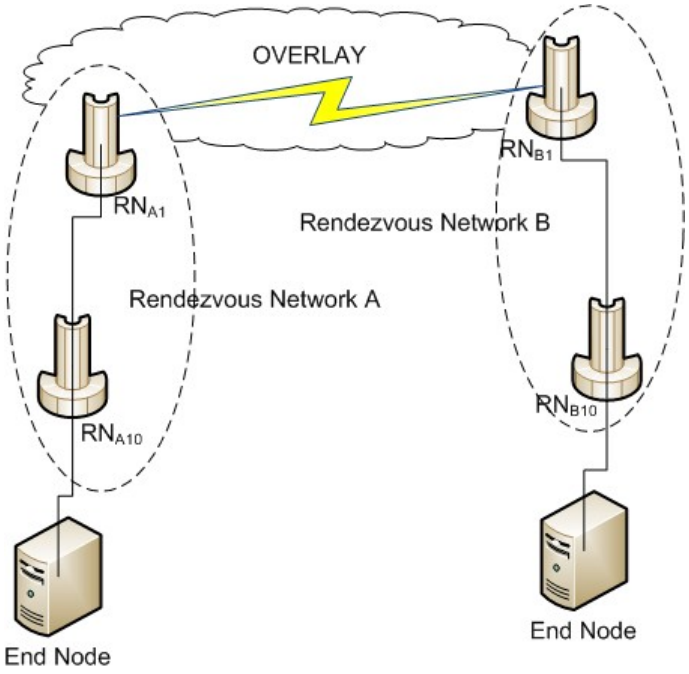

Figure 2.2: Rendezvous networks overlay.

When a publisher wants to publish to a certain SId and RId, first it sends the publication to the rendezvous node he is aware of. When the the publication reaches this first RN, depending if the node is in charge of the given scope or not, can either create a new rendezvous point for the given scope, or in the second case forward it further into the rendezvous network. When the publication reaches the RN that can register the given scope, it becomes the authoritative rendezvous point for the scope. Then, the new scope is advertised through the rendezvous network so all the nodes are aware of it and enables them to forward subscriptions to that scope to the appropriate RP.

Subscription works in quite a similar fashion, being the first step for the subscriber to contact the RN that is know to it. The subscription includes a SId and a RId the host wants to subscribe to. Similarly to the publish situation, the RN checks whether it is a RP for the given scope or not.

If not, the subscription is forwarded through the rendezvous network until eventually the matching RP is found (if the subscription reaches the root RN and there is still no match, it is forwarded via the overlay to the other rendezvous networks). When the RP is found, the rendezvous takes place and a forwarding path is established from the publisher to the subscriber.

The next architectural component in PSIRP is the **topology management and formation**, which can be further divided into intra and inter-domain cases. In order to achieve intra-domain topology management and formation, each network needs to have at least one **topology manager** (TM), which implements the topology management function. Then, each forwarding node maps its local connectivity creating a neighbor list and publishes it to the TM, which can then construct the full topology of the network based on this information.

For inter-domain topology management and formation, each TM from each autonomous system (AS) publishes the relevant peering information and routes to the **inter-domain topology formation function** (ITF) via an special SId provided for this function. Then, paths between publishers and subscribers can be created combining both intra and inter-domain information.

Regarding the next PSIRP component, namely **forwarding**, we can also subdivide it into intra and inter-domain forwarding. Intra-domain forwarding is based in the previously mentioned forwarding identifiers (FIds) and in link identifiers. Each link is associated with two link identifiers, one for each direction of the link. Link identifiers are m bits long, where k of those bits are set to 1, k is much smaller than m and m is large enough to avoid collisions with a high probability. Data paths are then encoded in the header of the packets using **in-packet Bloom Filters** (called in PSIRP architecture *zFilters*), created by ORing all the link identifiers the packet should traverse. Forwarding decision is made at each forwarding node by ANDing all its outgoing link identifiers, one by one, with the zFilter contained in the packet's header. The packet is forwarded through the link(s) whose link identifiers match the result of the previous operation. Multicasting is easily implemented by adding to the zFilter more than one link identifier from the same forwarding node. When the multicast tree is too dense, virtual links can be created. Virtual links are a collection of single links grouped all under the same link identifier which create some state in the nodes in exchange of reducing the false-positive percentage. Intra-domain forwarding also implements fast recovery and loop prevention.

Inter-domain forwarding is not fully defined yet in the document. Instead, a series of assumptions, design goals and technical considerations are provided.

The last component of PSIRP's architecture to be described is the **network attachment**. In order to attach to a certain network, a node willing to do so can subscribe to well known SIds used for this purpose or also can advertise its intention by means of publications. After the attachment procedure is started, a two-way control channel between the node and the attachment point (AP) can be established. The channel can be further used to negotiate subscription or service terms and accounting, authentication and/or authorization.

2.1.3 Design Considerations: mobility and security

Mobility in PSIRP for publishers and subscribers is achieved in a rather straightforward manner. It is enough for subscribers to re-subscribe or for publishers to re-publish at their new locations and let the rendezvous system handle the rest of the details. Topology must also be updated in order to maintain the multicast trees and data paths. Publications can not be lost as they are buffered during the hand-offs. Router and network mobility can be achieved in a similar fashion.

Security is a main concern for PSIRP and has been taken into account since the first design phases of the architecture, being naturally integrated into it. In order to provide a better protection at the forwarding level, it uses a special kind of zFilters that are calculated dinamically using a function called zFormation. This concept is further explained in Sec. 4.1.1 during the security analysis phase.

Other security mechanisms included into PSIRP's architecture are authorization mechanisms at the network attachment and the rendezvous system level, preventing possible spam through notarization and a technique known as **Packet Level Authentication** [17], which authenticates every packet sent by cryptographic means such as in-packet certificates and signatures. A much more detailed and elaborated overview of PSIRP's security mechanisms can be found on the deliverable 2.4 [27].

2.2 Scribe

Scribe [7] is a wide scale event notification infrastructure based on the publish-subscribe paradigm described in Sec 2.1.1. It is built over Pastry routing protocol, which will be explained later in this section.

Scribe uses Pastry to create topic groups and to build multicast trees to deliver the events to all the group subscribers. In Scribe, all nodes can create new topics and/or subscribe to already existing topics, and they can be the root of a multicast tree and/or a node pertaining to a multicast tree at the same time. Once inside the group, given that the node has enough permissions, it can publish events that Scribe forwards to the other members of the group in a best effort manner and without any delivery order guarantee. The groups have no limit on the amount of publishers and subscribers pertaining to it and nodes have no limitations regarding the amount of groups they can pertain to.

Each group in Scribe is identified uniquely by a **topicId** and the node with the most similar identifier to the one of the topicId implements the functions of the group's root node, being the root for the multicast tree for the given topic. The multicast tree grows as the amount of subscribed nodes to the topic grows and each of the intermediate nodes of the multicast tree maintains a table of the children nodes pertaining to the topic the tree is in charge of. The table is periodically refreshed by the children nodes and after a certain amount of inactivity children nodes are deleted from the table assuming they are no longer part of the group.

In the case of an intermediate node losing connectivity, and in order not to leave out of the multicast tree any leaf nodes, intermediate nodes should send control messages to its children nodes. If a children node realizes that there are no control messages arriving from its parent node, assumes that it has lost connectivity and rejoins the multicast tree again in order to regain connectivity. In the same fashion, if a node wishing to leave the group is in charge of any children node, it has to remain as a part of the multicast tree to keep forwarding the events, even though not being member of the group anymore.

As said before, Scribe is built over Pastry [23]. Pastry is a routing protocol which interconnects an overlay of nodes that are uniquely identified by a nodeID. The identifier space is circular and identifiers are 128 bits long. Identifiers represent a position inside the circular space and are assigned in a random way, allowing for physically distant nodes to have close nodeIDs.

Each node keeps track of the neighboring nodes, leaf nodes and a routing table. The leaf nodes are the set comprised by the closest nodes in both directions of the circular space and they maintain a coherent network structure and shorten the search time. The neighboring nodes are the n closest nodes according to some given network metrics and are used to maintain the route table.

The routing table contains a row for each assigned address block, and blocks are generated splitting the local nodeID into groups of b bits and grouping the nodes according to a prefix matching basis between the local nodeID and other nodes. Messages can be directed to any nodeID, whether it exists or not, and they are forwarded through the circular network until reaching the node with the matching nodeID or the one with the longest match. When sending a message, the node first checks if it has a direct route to the receiver, and if not it sends the message to the node with the longest prefix matching in his routing table. In this way it is assured that the message gets each hop closer to the destination.

2.3 Accountable Internet Protocol

As stated by Andersen et al. [3], almost all of the security problems related to the current Internet model, are caused by a lack of accountability in the current IP layer, that is, the impossibility of tracking *who does what*. The authors propose the Accountable Internet Protocol (AIP) not as a fix to the current IP layer but as a completely new approach to replace it.

AIP addresses are presented in the form *AD:EIP*, being AD the identifier of the administrative domain the host is attached to and EIP the end-point identifier, that is, the host itself. As names (addresses), are self-certifying, both ADs and EIPs are the public keys of the domain and the host respectively. In order to keep addresses to a fixed length, what is used is a hash taken from those public keys, making addresses 160 bits long as shown in Figure 2.3

By the use of public keys as identifiers, accountability can be achieved in a cryptographic safe way. Due to a host being able to be attached to an AD that is organized hierarchically or having multiple addresses at the same AD, generally speaking addresses are in the form $AD_1 : AD_2 : ... : AD_n : EID_{Ifn}$, where AD_k is the identifier for a single level in the administrative domain hierarchy and EID_{Ifn} is the identifier for a single interface of the host, using

Figure 2.3: AIP address structure.

the last 8 bits of the address to identify it.

Forwarding is done in a simple manner, and usually involves routers only having to inspect the destination AD field of the packet. Mobility is handled in an efficient way as the EID part of the address remains unchanged when a host roams from one administrative domain to another, and only the AD part of the address needs to be updated.

AIP aims at solving different security related issues that remain unsolved in nowadays IP layer, being one of those detecting and preventing source address spoofing. In IP, mechanisms such as ingress filtering are used already in order to prevent address spoofing, but as they rely in the operators configuring and maintaining proper and correct filters they are not usually so effective and, in addition, they introduce other problems like triangular routing in the case that a host needs to send packets with a different address or interface than the one that it uses to receive them. In the other hand, given the self-certified cryptographic nature of AIP addresses, spoofing can be avoided in a rather simple and convenient way. AIP focuses into preventing spoofing by entities, that is, a router in the middle of a data path can still pretend to spoof packets from host A to host B, but those packets will not be properly signed thus allowing the receiver to determine whether the data packets come from the right source or not.

Source address validation is done in two points; EID validation at the first-hop routers and AD validation at the edge AD routers:

- EID validation uses a verification packet V, given that the source is not already included into the first-hop router cache. The source needs to prove his identity by returning the verification packet signed with his own private key. If the verification packet is deemed valid at the first-hop router, the source is included in its cache and all the subsequent packets coming from the source are directly forwarded.

- AD validation involves 3 cases; if the AD trusts that the neighboring

AD has performed the required checks already, it forwards the packets without further actions required. If not, it uses uRPF in order to determine if the packet is valid, and if uRPF fails, then it sends a verification packet directly to the source like specified in the EID validation section.

In order to keep the cache within an acceptable size, the routers will insert wildcards in the form *AD:** if a certain threshold in the amount of verification packets containing the same AD part has been reached.

While these previously mentioned measures against spoofing also help to mitigate certain DoS attacks on which spoofing is involved, they certainly do not prevent all forms of DoS attacks. That is why a **shut-off protocol** has been included in AIP, in order to further protect hosts against those attacks. Shut-off protocol relies on the well-intentioned users - and regular users are considered always well-intentioned - installing smart-NIC cards into their machines, which will keep track internally of the most recent sent packets and which will be able to accept SOP packets. SOP packets include a hash of the packet that originated it and a TTL, all signed by the host sending the SOP packet in order to provide authentication. If the hash contained in the SOP packet matches one of those stored by the smart-NIC and the SOP packet is verified as authentic, the smart-NIC card will install a filter to suppress the traffic to the host that sent the SOP with an expiry time equal to the TTL contained in the SOP packet.

AIP can be also used to secure BGP as the route announcements can be signed by the originator, avoiding the propagation of unwanted routes which can result into a DoS attack by isolating an AD.

Other issues that AIP deals with and solves apparently in a correct way are those related to key management and crypto-algorithm compromise by introducing simple mechanisms in order to revoke keys and to switch or upgrade crypto-algorithm when needed by using the version field shown in Figure 2.3. AIP also takes into account scalability issues. Regarding those, a deep study on scalability of AIP has been done, taking into account predicted hardware improvement/growth rates and also estimating the amount of hardware resources and times needed by an Internet sized network using AIP.

2.4 Networking Named Content

Networking Named Content (NNC) [14] focuses in naming data and information rather than hosts in order to solve the availability, security and location dependence issues related to the current Internet model. NNC introduces the concept of **content-centric networks**, from now on **CCN**, on which the addresses used to establish communication refer to content rather than to location. NNC aims to replace the current IP layer with a new CCN layer improving strategy and security while maintaining all the characteristics that made and make IP attractive.

In NNC, communication is receiver driven instead of sender driven; receivers only get the data on which they have expressed an interest into. There are two basic packet type, *Interest* (**I**) packets and *Data* (**D**) packets. Data packets consume interest packets in order to preserve the flow balance, in a similar way as IP preserves the flow balance with ACK packets. The matching of interests and data is done on a prefix-match basis and it can be context dependent as well.

CCN nodes are composed by three main elements as depicted in Figure 2.4:

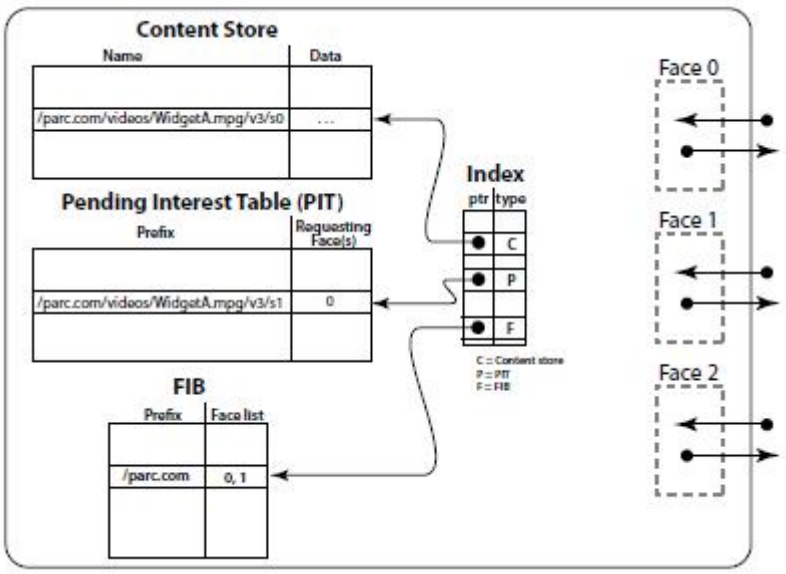

Figure 2.4: CCN node elements, [14]

- Forwarding Information Base (FIB) which is responsible for getting the

data that matches the arriving interests,

- Content Store (CS) which acts as a cache for data and
- Pending Interest Table (PIT) which stores the interests that haven't been satisfied yet.

Interest and data packet processing is handled by these three main elements mentioned before. Interest packets that arrive to a node are first compared to check whether there is a match in the ContentStore. If there is any data cached that matches the interest, the data packet is forwarded to the face where the interest packet came and the interest packet is discarded (to preserve the flow balance as explained before). If there was no match, the packet is compared against PIT entries and in the case a match is found, the face of the packet is added to the PIT entry and the interest is discarded. Finally, if there is a match in the FIB, the data is requested and the interest is added to PIT in order to forward the data as soon as it arrives to the node. If there is no match in any of the elements, the interest is discarded as the node doesn't know how to retrieve the data matching the interest.

Data packet processing is done in a similar match-case fashion as interest packet processing; if an arriving data packet matches the ContentStore, it is discarded as the data is already cached. If it matches an entry in the FIB, it means that the data is unsolicited as there is no match in the PIT, and if it matches an entry on the PIT, the data packet is forwarded the the list of faces stored in that entry.

Similarly to IP, CCN operates over unreliable packet delivering services, that is, interest and/or data packets can get lost or damaged. Retransmission is used to cope with this, but nevertheless it is the receiver who is responsible for asking again about interests that have not been satisfied. Duplicate packets are discarded as explained previously and nonces are used to prevent interests from looping. Flow control is performed at each hop and every data consumes an interest at each hop.

The naming structure used in CCN is hierarchical and it is structured in trees. Every piece of data has a version and a segmentation number so asking for correlative pieces of data or for new versions can be handled with ease. Mobility is possible as CCN handles data and not end nodes, so data exchange is always possible as long as it is physically feasible. The actions, triggers and attributes used to mark and forward data conform the strategy layer. All data available locally is obtained directly thus no routing is needed unless if the data is not found locally.

CCN provides several mechanisms in order to provide security. Built around the concept of **content-based security**, CCN authenticates every single data packet with a digital signature, and encrypts private content. In this way, private data can only be read by intended recipients and every recipient can authenticate the data it receives. CCN also provides mechanisms to manage trust regarding keys, network security and policy enforcement.

2.5 Layered Naming Architecture

Layered Naming Architecture (LNA) [4] proposes a new architectural style that involves three levels of name resolution in order to comply with what they call their *"four basic design principles"*. Such three levels of name resolution are, as stated by the authors, *(1) from user-level descriptors to service identifiers, (2) from service identifiers to endpoint identifiers and (3) from endpoint identifiers to IP addresses*. A flat namespace structure is also proposed for the service and endpoint identifiers. The benefits that LNA introduces are, among others, that services and data become the main objects rather than hosts, that mobility and multi-homing can be handled in an easy way and that middleboxes (such as firewalls and NATs) can be gracefully accommodated and no longer violate IP semantics.

The first LNA basic principle states that *"Names should bind protocols only to the relevant aspects of the underlying structure; binding protocols to irrelevant details unnecessarily limits flexibility and functionality."* Current Internet violates this principle as it binds services to end hosts and the location of those end hosts. In order to solve this issue, two new naming layers are required, one that will name services using service identifiers (*SIDs* from now on) and will bind user-level descriptors to those SIDs and another that will identify hosts in a unique way, regardless of their topology and location, using endpoint identifiers (*EIDs* from now on). Furthermore, this new two naming layers need two additional layers of name resolution in order to function properly: one that will resolve EIDs from SIDs and one that will resolve IPs from EIDs. In this way, it is only IP itself who deals with IP addresses thus making mobility and multi-homing possible in an efficient way. This new layering structure is depicted in Fig. 2.5.

The second basic principle states that *"Names, if they are to be persistent, should not impose arbitrary restrictions on the elements to which they refer."*. In order to achieve this, LNA proposes using a flat namespace for SIDs and EIDs.

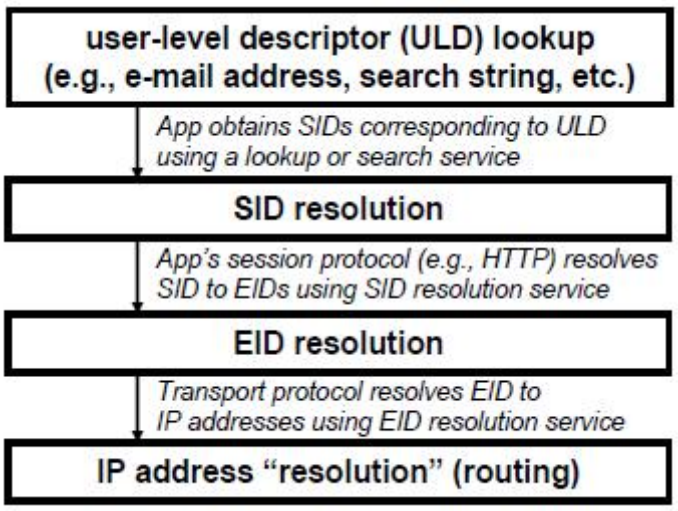

Figure 2.5: LNA naming layers, Balakrishnan et al. [4].

The third basic principle on which LNA is based says that *"A network entity should be able to direct resolutions of its name not only to its own location, but also to the location or names of chosen delegates."* When a machine request a connection to a service, the request destination entity can choose to redirect the requesting host to a delegate of its choice providing the same service/data. Trust relations are not affected by this, as the destination entity trusts the delegate, so the requesting host should also trust on it. The delegate concept allows to easily integrate middle-boxes in the architecture while also providing some extra DoS protection.

The last LNA principle states that *"Destinations, as specified by sources and also by the resolution of SIDs and EIDs, should be generalizable to sequences of destinations."* By being able to specify a list of SIDs or EIDs as destination instead of a single SID/EID, senders and receivers can choose which path or series of endpoints their data will traverse, giving much more control over it.

As SIDs and EIDs are presented as a flat namespace, LNA replaces DNS with distributed hash tables in order to be able to cope with this new naming model. Also it provides authentication as flat identifiers can hold some cryptographic meaning such as the hash of a public key and accompanying meta-data can provide also some means of verification by including some extra authentication information such as cryptographic statements from certifying entities.

2.6 Internet Indirection Infrastructure

Internet Indirection Infrastructure (i3) [25] proposes a general Internet indirection overlay that will provide mobility, multicast, anycast and service composition all-in-one in opposition to the current overlay or application based solutions which are completely disjointed and cover just one of the requirements previously mentioned.

As stated by Stoica et al. [25], *"the purpose of i3 is to provide indirection; that is, it decouples the act of sending from the act of receiving"*. In order to accomplish this objective, i3 nodes act as a rendezvous point where identifiers (id s from now on) of content and triggers are matched. In this way none the sender(s) or the receiver(s) need to be aware of the number nor location of each other(s). In its simplest way of operation, senders use packets in the form (id, data) and receivers insert triggers in the form (id, addr). Should an id matching occur between a packet and a trigger, data packets with id *ID* should be forwarded to the host(s) that inserted a trigger with id *ID* to the address *"addr"* specified in the trigger.

Matching is done in a longest-prefix match fashion where id s have **m** bits and there is a threshold of **k** bits that should be exact in order for the id s to match. Threshold k is selected such as $\mathbf{k} < \mathbf{m}$, but large enough to provide collision free id s. In the case of i3, this is $\mathbf{m} = \mathbf{256}$ and $\mathbf{k} = \mathbf{128}$.

In order to provide an efficient mechanism for longest-prefix matching, each node on i3 is responsible for a given set of unique id s. Furthermore, it should be a requirement that all the id s that have the same k significant bits should be managed by the same i3 overlay node. I3 routes packets with id *ID* to the node responsible for it, where the trigger matching and the forwarding are done. It is important to remark, tough, that data packets are never stored in the i3 overlay infrastructure, they are only routed to their destination and only a best-effort service such as the one present in IP is used.

As can be seen in Figure 2.6 (a), mobility is easily achieved as the receiver only needs to update the address field in his trigger with a new address in order to keep receiving the matching data, id remains unchanged. In the case of sender's mobility, there is no additional operation needed.

In the case of multicast, as depicted in Figure 2.6 (b), receivers that want to get the same data just need to insert their triggers with the same id so the data packets will be forwarded to all of them. For anycast, the first k bits of the id have to be identical in order to define an anycast group, and then the

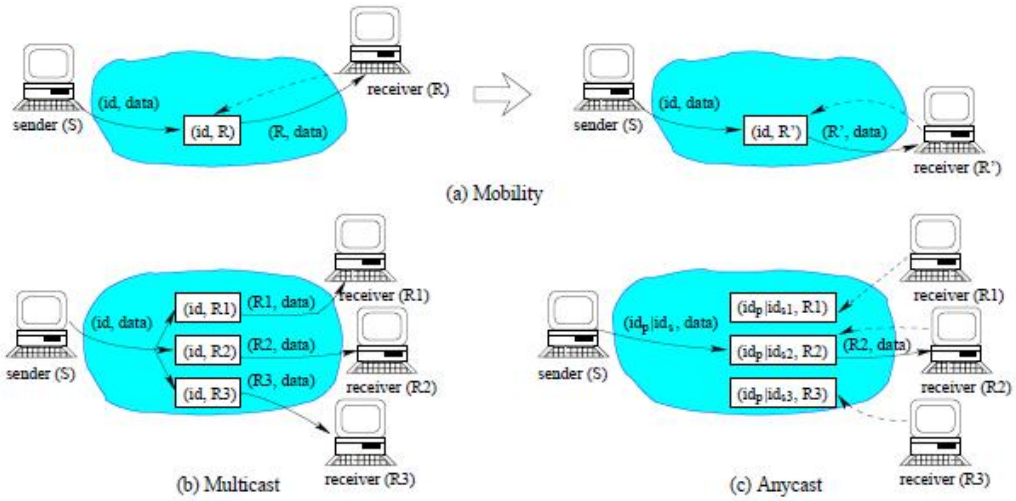

Figure 2.6: I3 mobility, multicast and anycast, as depicted by Stoica et al. [25]

unique receiver is chosen using a longest-prefix matching of the rest of the bits in id, as shown in Figure 2.6 (c).

Finally, in order to allow for source routing or service composition, id s can be stacked in data packets in the form of *(idstack, data)* or in triggers in the form of *(id, idstack)* where idstack can contain several ids and/or addresses. In this way, senders can specify a set of nodes the packet should traverse much like in source routing, and receivers can specify which nodes a data packets need to traverse in order to achieve service composition.

In order to improve the security characteristics of i3, two types of triggers are defined: **public and private**. Public triggers can be used to contact well known services while private ones can be used to establish private communication between two hosts and can be exchanged between those hosts using public triggers. Robustness is achieved in an easy manner as receivers refresh triggers periodically, thus damaged or lost triggers are restored as soon as the update takes place. In the case that the refreshing interval is too high, i.e. applications with highly demanding time constraints, either a backup trigger can be inserted by the end-host in another i3 node, to switch to it in the case that the original trigger suffers any problem, or the i3 infrastructure itself can be configured in such a way that will replicate triggers and store them in the nearest i3 node to the one containing the original one.

In order to avoid hot-spots to become bottlenecks on i3 because too much

routing being done on them, when the rate of packets matching a trigger is bigger than a given threshold, a copy of the trigger is inserted into as many nearby nodes as needed until the load is well-balanced.

Security is a big concern in i3, and security problems have been addressed thoroughly. Security problems that might arise are those related with:

- attacks related to the use of triggers pointing to end-hosts, such as eavesdropping, impersonation or reflection (which can be used in order to launch a DoS attack) and
- attacks related to forming arbitrary topologies using cyclic triggers causing loops or dead-ends, and thus, exhausting resources (in order to carry out a DoS attack).

In order to avoid all of these previously listed problems, i3 provides with 3 techniques to mitigate them:

- constrained triggers on which id construction is tied to one-way functions in order to solve eavesdropping, impersonation and loops,
- push-back mechanisms in order to avoid dead-ends, allowing to remove subsequently triggers that point to a dead-end and,
- trigger challenges, where a trigger can only be inserted into the i3 overlay after a challenge-response has been solved. This avoids reflection and also dead-ends.

Last but not least, i3 provides a good level of anonymity as sniffing the traffic generated by a sender/receiver will not reveal the identity of its counterpart.

2.7 Data-Oriented Network Architecture

The Data-Oriented Network Architecture (DONA in short) [16], focuses yet again into redesigning the Internet's naming and name resolution infrastructure.

As stated by many others such as in Jacobson et al. [14] or Nikander et al. [19], nowadays users care much more about data and information regardless of its

location, contrary to the initial Internet communication model, which focuses in host-to-host communication.

DONA aims into addressing some user-relevant issues such as:

- persistence, or the property of data and services to have names that remain valid until needed, even if the data or service changes its physical location, i.e. it is moved to another server,
- availability, by means of reliability and low-latency and,
- authenticity, so the users can be sure that the data source is the one intended and not a malicious user spoofing it.

In order to achieve these goals, DONA proposes replacing the current naming infrastructure with flat and self-certifying names and the current name resolution mechanisms with an anycast primitive that can be used for several resource discovery kinds. In DONA, persistence and authentication are achieved through the use of a flat, self-certifying namespace as previously mentioned, while availability is achieved through the use of an efficient name resolution mechanism, i.e. the anycast primitive previously mentioned.

As stated by Koponen et al. [16], *"to provide availability the name resolution process should (a) guide requests to nearby copies of the data , and (b) avoid failed or overloaded servers"*.

DONA uses a route-by-name approach as name resolution mechanism in order to accomplish these two previous requirements, as routing protocols are designed both to use shortest paths and to route around failed or overloaded points.

As in many other approaches for new namespaces, DONA uses public-key pairs as a mean of identifying data, services, hosts or any other named entity. Each named entity is associated with a principal and every principal has a public key associated with it. Names then come in the form **P:L**, where P is the hash of the public key of the principal and L is a label to ensure uniqueness. Data is sent in for of triples *(data, key, sig)* in order to provide authentication; a user receiving this previously mentioned triplet can check that it came from the appropriate principal by taking a hash over the public key and verifying that it matches with P and also checking that the public key attached in the triplet is the one that generated the signature, as only the principal has the appropriate private key to produce the signature.

As flat names can not be learned in an easy way by users DONA proposes the use of a human-readable namespace that would map into those unreadable flat names and some external mechanisms in order to solve that mapping, i.e. search engines.

Regarding the name resolution mechanism, DONA introduces the concept of **"resolution handlers"** (RHs from now on), which by means of **FIND** and **REGISTER** messages will manage the name resolution process. FIND messages are in the form *FIND(P : L)* and REGISTER messages are *REGISTER(P : L)*. If a host is serving all the data associated with a certain principal, a *REGISTER(P : *)* can be used instead. Longest-prefix matching is used in order to match the entries in the RHs against the FIND messages. An especial type of **UNREGISTER** message is also available in order to allow servers to specify that they are not longer serving certain data.

Security in DONA is mostly implemented by relying on external mechanisms or providers. For bandwidth exhaustion attacks (a type of DoS attack), DONA relies in IP-level mechanisms which would drop the streams that are overwhelming it, and for resource overload attacks (also classified as DoS attacks), DONA relies into providers restricting the amount of FIND/REGISTER messages that a host can send per minute. Other solutions like puzzle solving are also proposed but not really explained in-depth by the authors.

In order to avoid malicious RHs isolating clients, DONA offers the possibility to clients of being able to access copies matching their interests other than the closest one. In that way DONA ensures that clients will be able to avoid misbehaving RHs and will always be able to access the data the requested.

Regarding key security, DONA proposes to introduce key revocation mechanisms, but rather than implementing them into the architecture, it relies again on external sources to provide such mechanisms.

Last but not least, DONA offers solutions also for content-caching, mobility, multihoming and multicast.

2.8 Postcards from the Edge

As stated by Yates et al., Postcards from the Edge [21] is *"a cache-and-forward architecture that exploits the decreasing cost and increasing capacity of storage devices to provide unified and efficient transport services to end*

hosts that may be wired or wireless; static, mobile, and/or intermittently disconnected; and either resource rich or poor".

Postcards from the Edge focuses on optimizing large file transfer as a separate service, while acknowledging the need for the traditional best-effort delivery service for other kind of services such as VoIP, video and audio streaming and the like. In a cache-and-forward architecture storage is performed at every node, regardless of it being a core router in the backbone, an edge access point or even a mobile host.

In its simplest scenario, in order to deliver a file to a mobile node, the steps taken are the following. Every mobile node has tied to itself a set of **post-office** nodes (PO from now on) and there is a name resolution service, much like DNS, which resolves the POs for a given host. Once the POs for the mobile host are retrieved, the sender forwards the file to those PO(s) and they keep the file stored until the mobile host is available for delivery. When the mobile host is available, the file is delivered from the PO(s) to the host.

Postcards from the edge introduces several new protocols for the transport layer while leaving untouched the IP layer to be used for control purposes.

The architecture proposes a network composed both by traditional nodes, like today's routers and the like, and cache-and-forward nodes (CNF nodes). It also proposes a naming convention for files in the form *UFID.FQDN* where **UFID** is a unique identifier for the file (MD5 hashes are proposed for this purpose) and **FDQN** is the fully qualified domain name of the home location of the file. In order for this naming infrastructure to work properly, the concept of *File Name Resolution System* (FNRS) server is introduced.

Each host has an authoritative name resolution server which is in charge of maintaining and updating the set of POs tied to a particular host. A *Name Resolution Protocol* (NRP) is used to maintain and update this list, so each time that a mobile host informs of a new post office node, the list of POs related to that certain host will be updated in the NRS server.

The *Routing Protocol* (RP) used in Postcards from the Edge is yet not well defined, but an overview of the most basic features is depicted. The first step involved would be to retrieve the list of POs related with the target CNF node by means of the name resolution service. For a CNF node that is not mobile (i.e. is wired), its PO would be itself and routing would be done in a similar fashion than in the current Internet. In the other hand, if the CNF node is a mobile host, then a list of PO(s) where the file can be sent would be retrieved. The implementation of the routing mechanism for this

last scenario remains as a research topic.

The link protocol (LP) has two main components, namely the *Link Session Protocol* (LSP), which is used to establish the link, and the *Link Transport Protocol*. There is also an additional *Link Management Protocol* for diagnostic purposes, like monitoring errors or sending ACKs.

The architecture also implements a *Caching Service Protocol* (CSP), which is in charge of retrieving files and returning them to the appropriate node(s) (i.e. the node(s) which made the request for that particular file), and a *Transport Protocol* (TP), which is responsible for fragmentation and reassembly of files. The main composition of all the previously mentioned protocols can be seen in Fig. 2.7.

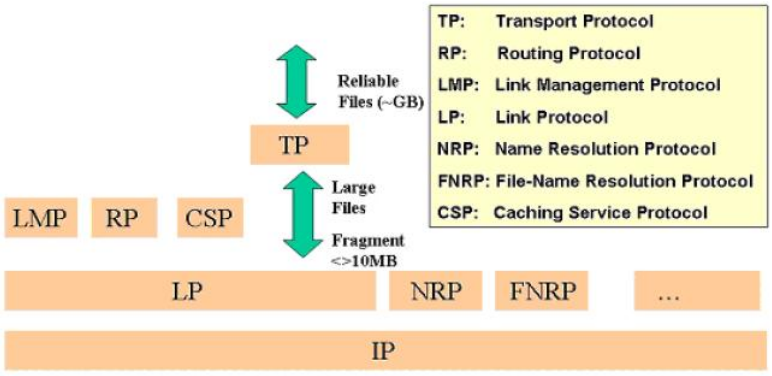

Figure 2.7: Postcards from the Edge data and control plane protocols, Yates et al. [21].

Postcards from the Edge functions on a hop-by-hop basis, which means that transmission to the next hop can not start until the transmission to the previous hop has been completed. This working methodology, which could seem a burden on the performance of the architecture, actually improves the overall performance in mobility scenarios by greatly reducing interferences.

Chapter 3

Methodology

> *"It is common sense to take a method and try it. If it fails, admit it frankly and try another. But above all, try something."*, Franklin D. Roosevelt.

This chapter describes the methodology employed in this Master Thesis work both when approaching the background theory part and the security analysis part. Following, the criteria on how the selection of architectures has been done and how the security analysis has been performed is described.

Architecture Selection

After selecting a broad range of papers, RFCs and project deliverables and documentation dealing with Future Internet Architectures, it was obvious that the amount of material was too extensive to be all suitable for the Thesis. Therefore, all the material was inspected and classified into valid and non-valid. Criteria for this classification tried to assure that the architectures selected represent a broad set of different approaches to the research problem of the Thesis, meaning that some are fully new architectures and others are just redesigning some parts of the current Internet (i.e. naming) while maintaining some of the former design. In this way a good balance is obtained and pros and cons for both types of approaches can be seen.

Security Analysis

The analysis is presented on a "per architecture" basis. Presenting it organized by architectural elements was unpractical, due to not all the papers describing each architectural element for every architecture analyzed.

Security analysis has been done on each architecture over critical architectural elements such as forwarding, topology management, central elements (i.e. rendezvous) and key elements relevant to particular architectures (i.e. shut-off protocol in AIP). For each of these elements, the security mechanisms developed by the authors were studied and based on that, possible vulnerabilities have been outlined, if any.

Vulnerabilities have been only described or demonstrated in a theoretical way, as trying them on a live environment has been in most of the cases not possible, due to the implementation not being public or not existing at all. In the cases that a prototype implementation has been available, the difficulty of setting up the environment or the lack of a testbed infrastructure has been too big and falls out of the scope of this Thesis.

Chapter 4

Security Analysis

> *"The mantra of any good security engineer is: 'Security is a not a product, but a process.' It's more than designing strong cryptography into a system; it's designing the entire system such that all security measures, including cryptography, work together."*, Bruce Schneier.

In this Chapter, the security analysis of the architectures presented on Chapter 2 is performed. The analysis is presented on a "per architecture" basis, as presenting it organized by architectural elements was unpractical due to not all the papers describing each architectural element for every architecture analyzed. Focus is placed over DoS/DDoS protection, as it is the kind of attack that prevails the most and the one that most architectures intend to eradicate. Also, special emphasis is given to the analysis of the forwarding implementation of each architecture, as it is the most likely venue for DoS/DDoS attacks to happen, if vulnerabilities are found on it. Following, the security analysis is presented.

4.1 PSIRP Security Analysis

PSIRP provides an extensive set of security mechanisms such as rendezvous level authentication, data integrity and confidentiality, secure inter-connection of rendezvous networks, packet level authentication, network attachment security and spam prevention through notarization. As the documentation for PSIRP is very extensive and many of the security properties and possible attacks and countermeasures for the previously mentioned mechanisms are

explained with a high level of detail in [27], the focus of the analysis for this architecture has been put on the forwarding plane, as it is the most sensitive one regarding DoS/DDoS attacks. Following, the security analysis of PSIRP's forwarding is presented.

4.1.1 Forwarding

PSIRP [26] uses in-packet **bloom-filters** to implement its underlaying forwarding fabric. In-packet bloom-filters have been proposed as one of the possible solutions to implement DoS and DDoS resistant forwarding and they can be used to establish a forwarding path between the publishers and subscribers in a source routing fashion, by adding to the filter the link IDs of the forwarding nodes the data packet should traverse. In principle, the bloom filter is calculated on demand and it is only known to the publisher, once a successful publication-interest match has occurred into the rendezvous system.

Due to their probabilistic nature, one of the only possible methods to forge a bloom filter that will establish a forwarding path between two hosts is using brute force until a valid filter is obtained. While trying to guess valid bloom filters by means of brute forcing is well possible, the computational effort to achieve it is huge and increases drastically as the number of hops the attacker is separated from the target increases.

In-packet bloom-filters are called in PSIRP **zFilters**. In order to make the approach even more resistant to DoS and DDoS, PSIRP introduces the concept of **zFormation**, which dynamically calculates the zFilters and link IDs by applying a function over some in-packet information such as the flow identifier, the in and out interfaces and a shared secret K, which changes over time. This makes the brute force approach even more complicated, as every time the shared secret changes, the zFilter needs to be recalculated and zFilters are tied to the flowID they were calculated for and to the specific pair of inbound and outbound interfaces. However, it is demonstrated that brute-forcing a 1-hop zFilter is a relatively easy achievable task that can be carried out in a reasonable amount of time, even when using zFormation [22].

ZFilters and Forwarding

ZFilters have several parameters, namely **m**, which is its length in bits (typically ranges from 128 to 256), **k**, which denotes the amount of bits set to one

in the filter (typically 5) and the **maximum fill factor** (p), which limits the amount of bits that can be set to one in a given zFilter.

The basic notion behind zFilter forwarding is constructing it in such a way that it will contain all the link IDs that the data packet from the publisher to the subscriber should traverse. In order to achieve this, all the link IDs of the forwarding nodes that are involved in the delivery path are ORed together in one single bloom filter defining a unique (sometimes subject to certain amount of false positives) forwarding identifier. When the data packet is sent, every forwarding node does an AND operation of the zFilter and each of its out link IDs, and if the result matches the link ID, then the packet is forwarded through it. This is depicted in Fig. 4.1

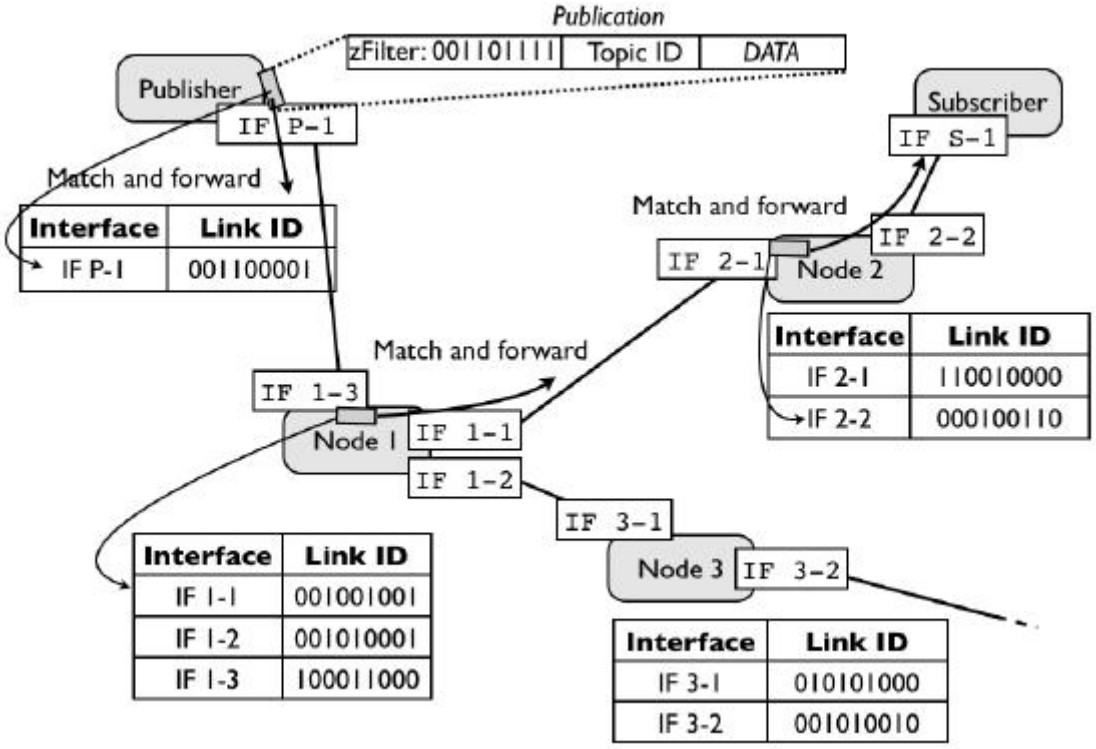

Figure 4.1: Basic forwarding with zFilters, Jokela et al. [15].

In order to improve DoS resistance, zFilters and link IDs can be calculated dynamically by using a function called zFormation. The basic forwarding idea depicted in last paragraph remains the same, but in this case the zFilter and every link ID is calculated on the fly by applying the zFormation function

over the flow ID, the in and out interfaces and a shared secret K between the topology manager and the forwarding nodes. The shared secret changes over the time so the link IDs change and brute-force guessing valid zFilters is made more difficult, as the zFilter has to be recalculated by brute-force every time the shared secret K changes, and the delivery path gets tied to the flow ID, deeming it unusable for any other flows.

Nevertheless, it is demonstrated that for a fill factor of $\mathbf{p} = 0.5$, the number of attempts needed to guess a valid 1-hop zFilter with probability 1/2 is somewhere near 10^2 and for a 2-hop zFilter 10^4, which makes guessing this path length zFilters, even if they are computed using zFormation, an achievable task in an acceptable amount of time [22]. Sec. 4.1.1 makes use of this fact in order to perform a DDoS attack.

DDoS using 1-hop brute-forcing and legitimate zFilters

This attack scenario involves the attacker owning a bot-net which has one of its controlled hosts 1 or 2 hops away (1 hop would be the ideal situation) from the victim machine, or in case of not having any, it relies on luring some machine (which is a less probable option but still achievable) which is 1 or 2 hops away into subscribing to some (legal) publication issued from the bot-net. From now on we will call that previously mentioned machine *intermediary*. A generic representation of this scenario can be seen in Fig. 4.2.

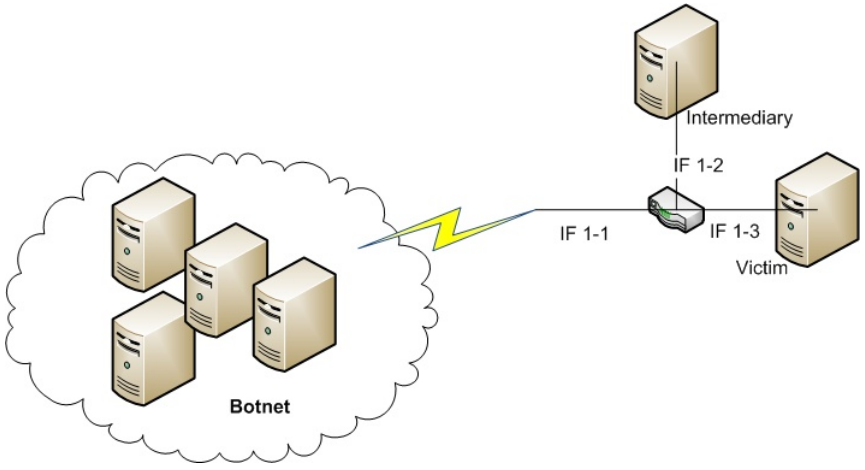

Figure 4.2: Attack scenario.

By combining a valid zFilter from the bot-net machines to the intermediary

with a 1 or 2 brute-forced zFilter from the intermediary to the victim, we reduce the attack computational cost to that of guessing (brute-forcing) a valid zFilter to a machine that is 1 or 2 hops away. Of course, for this attack to work, we need to make some assumptions such as that the semantics of the system provide a reply packet, in order for the intermediary machine to know whether it has found the appropriate filter or not. Following, a step by step explanation of the attack is provided and a forwarding and data flow diagram representing it can be seen in Fig. 4.3:

- The intermediary gets a valid zFilter (F_{I-V}) to the victim machine by brute-force. This has been demonstrated to be a not-so-much computational effort consuming task.
- The other machines in the bot-net issue regular publications to which the intermediary subscribes. In this way, all the machines of the bot-net have a valid zFilter (F_{B-I}) that establishes a forwarding path to the intermediary.
- The machines of the bot-net subscribe to a publication issued by the intermediary containing the brute-forced zFilter obtained in the first step.
- By ORing the valid zFilters with the brute-forced one, every machine in the bot-net obtains a zFilter (F_{B-V}) that establishes a forwarding path to the victim machine.
- All the machines in the bot-net send data packets through those paths to the victim machine in order to flood it and to cause a DDoS.

In the case that the attacker doesn't control an intermediary machine, the attack is still possible if the attacker can obtain valid zFilters to a machine that is located 1 or 2 hops away from the victim, by luring it into subscribing to some publication.

Regarding the use of zFormation, in which both the in and the out interfaces are used to construct the valid zFilter, the approach previously mentioned would not be effective, as the in-out interface pair from the intermediary to the victim would be probably different that those in-out pairs from the bot-net machines to the victim. Nevertheless, the attack still can be modified in order to adapt to these new constraints. The modification of the attack is following explained:

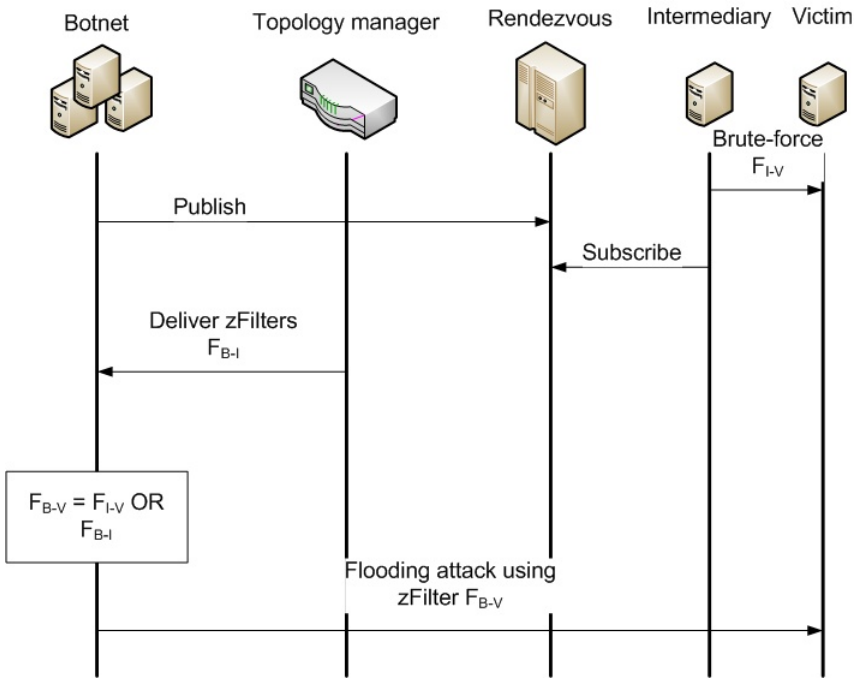

Figure 4.3: Attack flow.

- The machines in the bot-net issue regular publications to which the intermediary subscribes. In this way, all the machines of the bot-net have a valid zFilter (F_{B-I}) that establishes a forwarding path to the intermediary.
- Using the previously zFilter and brute-forcing from each of the bot-net machines the last segment left, namely a zFilter that will include IF1-1 and IF1-3 as depicted in Fig. 4.2, which is equivalent to a one-hop brute-forcing attack, every machine would get a valid zFilter to the victim.
- All the machines in the bot-net send data packets through those paths to the victim machine in order to flood it and to cause a DDoS.

While still possible, now every machine in the bot-net has to brute-force a 1-hop filter, which would make the attack slightly more difficult.

Path overloading by zFilter combination

This attack scenario involves the attacker also owning a bot-net and it assumes the attacker has one controlled machine on each end of the path that has to be overloaded. By combining zFilters in certain ways, it is possible to make all the traffic traverse a selected set of forwarding node(s), thus trying to overload those node(s) with an excess of traffic.

The attack is carried out as follows:

- The first step involves obtaining a zFilter that will establish a forwarding path between two of the bot-net machines, each of them situated at one end of the path that the attacker wants to overload. In order to get such a zFilter, it suffices issuing a publication from one of the machines and having the other to subscribe to it.
- After performing this first step, the attacker needs to get valid zFilters from every machine to the machine that issued the publication in the first step. By issuing publications from those machines and having the last one to subscribe to those publications, every machine in the bot-net gets a zFilter that establishes a forwarding path from them to the *"middle"* machine.
- By ORing each of those zFilters with the one obtained in the first step, every machine in the bot-net gets a zFilter whose forwarding path crosses a common set of node(s), a.k.a. the nodes included in the zFilter obtained in the first step.
- All the machines start sending data packets through their forwarding paths, thus overloading the desired segment of the network by filling its capacity.

Note that no illegal step has been taken in getting the zFilters, as they have been obtained by legal publish-subscribe operations.

4.1.2 Possible Solutions

The attacks previously introduced in the last two sections would most probable work in the case of zFilters generated without the zFormation technique. Whether they would work also in the case of using zFormation as described

in [22] is something left for further study. A probable guess would be that it would work for the first attack scenario, but not for the second.

Given that flow IDs can be selected at will by the intermediary machine, it would suffice to brute-force the 1-hop zFilter using the same flow ID that the one used when getting the legitimate zFilters from the other machine(s) of the bot-net. In the second attack scenario, as all zFilters obtained are valid zFilters through publish-subscribe operations, it would be difficult to merge them having different flow IDs, and it is unknown to the author whether flow IDs can be easily spoofed or not and if the filters would be still valid in that case.

Regarding the possible solutions to mitigate these attacks, the most reasonable proposal would be that the topology formation manager would always deliver zFilters with the maximum fill factor permitted. In this way, it would not be possible to add extra edges to the path, thus making impossible to combine zFilters. While it seems a good solution, it involves choosing a bigger k and perhaps reducing the maximum fill factor in order to enable short paths to be represented as zFilters using the whole capacity of the fill factor. In order to ensure that the zFilters are not to be combined, two different approaches can be taken, which are briefly outlined following:

- to have maximum fill factor parameter in the packet header, which is modified to be as close to the actual fill factor as possible.
- to have a parameter that varies the k accordingly to the needs of the filter.

Both of this approaches have their pros and their cons, probably causing some impact on the scalability of the architecture, and further study of these solutions is left open as a future work.

4.2 Scribe Security Analysis

As mentioned in Sec. 2.2, Scribe is built over Pastry. Pastry uses distributed hash tables in order to store routing entries, and DHTs suffer from several security weaknesses specific to them, that could be exploited by a malicious user to carry out different types of attacks. The three most remarkable weaknesses are the Sybil attack, the Eclipse attack and routing and storage attacks, which are briefly described following and depicted in Fig. 4.4:

- sybil attack refers to a malicious user creating several pseudonymous entities in order to gain a big influence of the system thus thwarting its redundancy,
- eclipse attack refers to a malicious user creating references in well-intentioned nodes pointing to malicious nodes in order to corrupt the routing tables and,
- routing and storage attacks refers to malicious nodes trying to corrupt data or not routing data in a proper way.

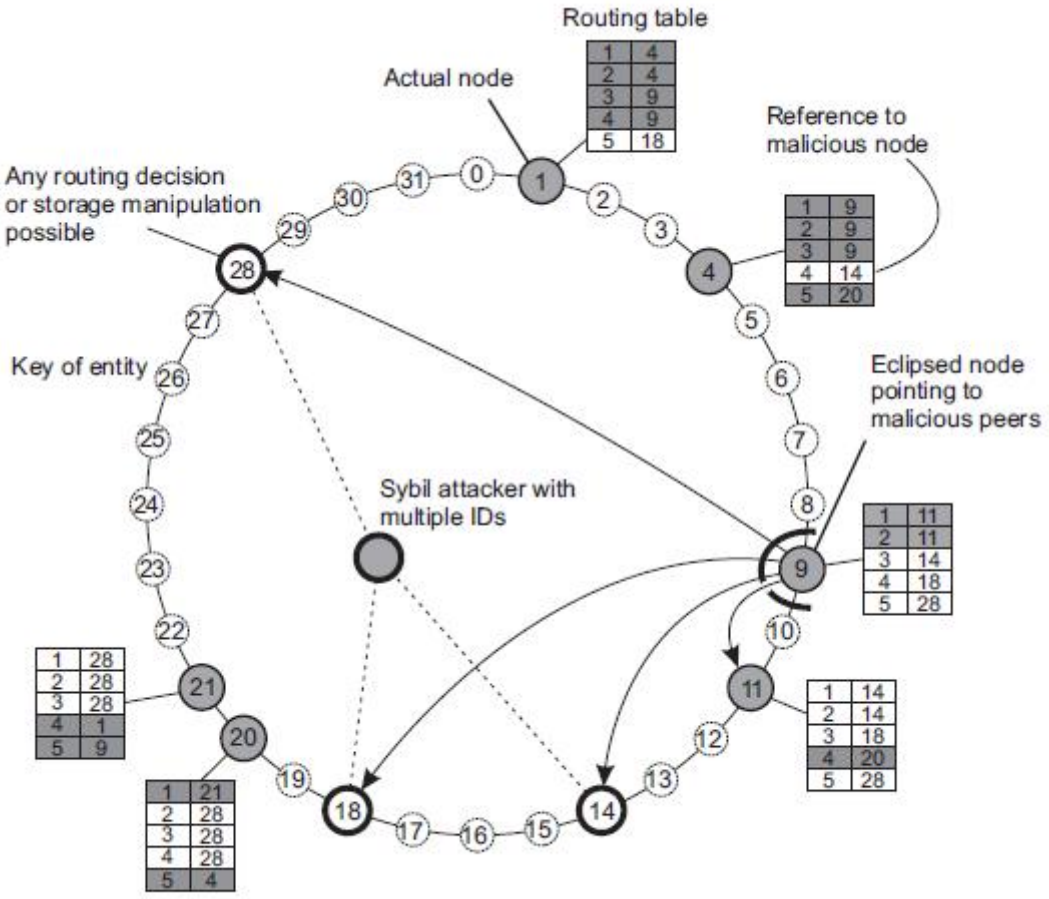

Figure 4.4: DHT vulnerabilities, Urdaneta et al. [28].

The main mechanisms that should be used in order to avoid this kind of attacks are secure assignment of node identifiers, maintaining the routing

table in a secure way, and implementing a secure message forwarding. A thorough analysis on this kind of measures and mechanisms is presented by Druschel et al. [6].

Possible solutions regarding the secure use of DHTs and the attacks they are vulnerable to can be found at [28, 24, 9].

4.3 AIP Security Analysis

AIP enforces security through accountability. The fact of keeping track on who does what can prevent many types of attacks present on the current Internet's model, such as DoS, spoofing and the like. AIP uses self-certifying addresses making spoofing a lot harder and authentication of content much easier. AIP also prevents DoS attacks using a special protocol called shut-off protocol.While these previously mentioned measures offer a better protection level against those threats, some possible vulnerabilities have been encountered during the analysis, which are presented following.

4.3.1 Spoofing issues

As explained in Sec. 2.3, in order to verify ADs and EIDs, routers along the data path use some verification measures such as verification packets, uRPF or trust relations. When a packet is deemed to be valid, an entry for the AD:EID combination is added to the router's accept cache. In order to maintain the size of the cache within some reasonable limits, when many entries containing the same AD part are found in the same accept cache, all the entries are deleted and an entry in the form AD:* is then added to represent those which were deleted.

Given that an attacker controls enough hosts in the same administrative domain in order to upgrade the entry of the accept cache to one of the form AD:*, or that the attacker happens to be attached to a router which has installed such an entry in its accept cache, then he can spoof EIDs at will, as the router will accept every packet coming from that AD.

4.3.2 Forwarding

AIP forwarding is based on a next-hop lookup approach, where the intermediate routers just inspect the next-hop destination AD (administrative domain) field in order to forward the packet. AIP packet headers allow to stack destination ADs by using the *"dest AD stack"* field in the packet header, which has a size of N*160 bits, being N the number of ADs in the stack and 160 bits the regular length for an AD or EID.

While this approach makes routing and forwarding decisions simple and straightforward, it can also allow an attacker to consume large amounts of bandwidth by making the packet to loop for some time between two ADs.

Assumptions:

it is not specified whether N has an upper limit in order not to allow "infinite" ADs in the dest AD stack. Nevertheless, we can assume that at least the field can contain a reasonable amount of ADs, which would suffice for our purpose.The attacker owns a bot-net, and needs to install some state in the system, namely valid entries of AD:EID in the accept cache of the routers he needs to traverse. For this, it would suffice to send a valid regular packet though the route in order for the routers to add a cache entry.

Attack:

Once the attacker has got the needed entries in the accept caches of the routers, he needs to craft special packets which will contain in the dest AD stack some sort of (finite) loop. In order to do this, the attacker can insert in the field a stack of ADs in the following form:

$AD_1 : AD_2 : AD_3 : AD_m : AD_n : AD_m : ... : AD_n : ... : AD_{dest}$

Given that the source address of the bot-net machines used in the attack is in the accept cache, which should be as a first valid regular packet has been sent from them to the route being used, if several machines use this approach and they loop through the same pair of ADs, they can cause a huge amount of traffic to loop constantly through the ADs edge routers, wasting resources and consuming bandwidth.

A possible solution for this attack could be that the border routers keep a small cache with the hashes of the last n packets that have traversed them. In the case of detecting the same hash for a small certain amount of times, they would just have to drop the packet in order to get rid of the traffic excess.

4.3.3 Forcing Shut-Off by Replay techniques

AIP uses what they call **"Shut-off protocol"** in order to prevent DoS attacks. The idea of this protocol is based on every computer being equipped with a *"smart-NIC"* card, that will control the network behavior of the host by being able to stop certain traffic flows or to limit them. This smart-NIC card keeps track of the most recently sent packets, and it accepts SOP packets. The key point here is that SOP packets cannot be forged, as they are signed by the sender and they include a hash of the data packet that originated the SOP packet, which assures that malicious hosts cannot forge SOP packets to force a DoS towards a victim by making a machine to stop to communicate with the victim.

Nevertheless, it is still possible, under certain conditions, to abuse the use of SOP packets. If an attacker is able to sniff such a packet between the victim and some other host, the attacker just has to replay the packet from time to time in order to further block the communication between the victim and the other host. The replayed SOP packet will still appear to be valid as it has the victim's signature and it contains the hash of a recently sent packet.

Another possibility to abuse the shut-off protocol by using replay techniques could work as follows. Given that the attacker is able to sniff regular traffic, there is nothing that prevents him from replaying the sniffed traffic, as proper entries in the accept caches have been installed by the original packet and he is not tampering the contents of the packet or spoofing any address, so the replayed packet would look exactly as the original one.

If the attacker starts to replay such sniffed traffic all the time, the hosts that the victim is communicating with will start to receive many duplicated packets, but all of them will seem to be originated by the victim machine. Whether this will make those hosts to send SOP packets to the victim or not, it is not clear on the paper, but a good guess would be that after some time receiving many duplicated traffic, it would be considered as some kind of flooding attack and SOP packets would be sent to the victim.

4.4 NNC Security Analysis

Networking Named Content presents a rather interesting approach into preventing DoS/DDoS attacks and also spoofing of content. NNC names data instead of hosts, and all data units are signed for authentication, and en-

crypted if privacy is needed also. This fact, assures hosts the authenticity of the data they retrieve, being the original or a mere copy of it cached in some intermediate CCN node.

NNC implements forwarding in a very intelligent way, mostly described in Sec. 2.4. Interest packets are the only packets that are actually routed, and while being routed, they leave a trail behind, namely in the form of entries on the PITs of each CCN node they traverse. When the data matching an interest is retrieved, it has only to follow the trail left by the interest packet which requested it, in order to reach the receiver. Thus, data moves in a hop-by-hop fashion, and it consumes the interest that originated the request in every CCN node it traverses back to the receiver. In this way, DoS/DDoS attacks by means of data flooding are not possible to carry out, unless only locally through the local link. Any other attempt of data flooding will just not work, as there will be no matching interest in the CCN node, and data will be automatically discarded as explained by NNC forwarding model.

Following, some considerations and possible venues for attacks in NNC are described.

4.4.1 Content Spoofing and Trust Management

While it is true that having a signature over every data packet is an efficient way of preventing spoofing of content, this can cause a false sense of confidence in users which can result perhaps in a bigger problem than benefit.

Regular users usually don't pay much attention nor put much effort into actually verifying whether content, and the signature or certificate that proofs that the data is the correct one, match one to another. This behavior can be already seen in the current Internet and the use of certificates to authenticate websites and the like. Phising websites without valid certificates, or websites with a certificate that has already expired are still accessed by the vast majority of regular users, which are not *"security-minded"*. In the scenario presented by NNC, all the data is supposed to be authenticated, fact that can give users a false feeling of safety and to make them think that in this kind of architecture every piece of data they will get is properly authenticated and therefore, safe.

While this is not a problem of the architecture itself, it is important to remark that an architecture cannot rely in users verifying every piece of data they get or reading through a certificate in order to determine if it is valid or not.

This verification should be done by the end-user applications automatically in order to work properly, and shouldn't rely on the user to determine when to block or not incoming data packets.

Another problem regarding content spoofing, specific to this architecture, is that in the case that an attacker can effectively spoof some content, that spoofed content will remain in the caches of every CCN node it traverses, and every time a client will request the data, the spoofed copy can be stored in more and more caches, making the spoofed data copies to grow exponentially, and eventually to maybe take over the original valid data.

4.4.2 Forwarding

The fact that forwarding is implemented in a way that every data packet should consume an interest in order to be able to progress towards the receiver, makes NNC DoS/DDoS resistant to data flooding. Nevertheless, it is also true that data flooding can still be performed through the local link, affecting only the closest CCN node. While this fact can seem a priori something unimportant, if an attacker owns an insider machine to some target company or organization, it could still cause several damage by flooding through the local link other hosts or the CCN node, preventing the rest of machines to communicate or to get their requested data.

The fact that local links can still be flooded with data is not something that should be ignored, as one single CCN node can serve several users, and flooding it is still a form of denial-of-service attack.

In the other hand, a distributed flooding can still be performed using interest packets. Given that the packets don't share the same name components, so they are not combined into one interest, a massive sending of interest packets could overwhelm CCN nodes by filling their PITs or consuming all the bandwidth, resulting also in a denial-of-service attack. Several measures are proposed by the authors of the paper [14] in order to mitigate interest flooding attacks. Whether that measures are fully effective or not, is subject to discussion.

4.5 LNA Security Analysis

The Layered Network Architecture focuses at redesigning the name resolution levels in the Internet. In order to do so, they introduce three levels of name resolution and their corresponding resolution layers. While this approach helps for sure solving some of the problems that the current Internet's architecture suffers from, like mobility or multi-homing, it also introduces new venues for new attacks.

Nowadays, Internet only has one name resolution layer, namely DNS, and yet, plenty of vulnerabilities have been discovered on it, like DNS cache poisoning, information disclosure by zone transfers, DNS rebinding attacks and many others. Introducing new name resolution layers, if not implemented perfectly and in a really 100 percent safe way, could introduce many other new venues for attacks, which could make the situation worse than it is right now.

LNA proposes **distributed hash tables** (DHTs from now on) as a way to implement a name resolution infrastructure that can resolve flat names such as SIDs and EIDs as depicted in LNA's architecture in Sec. 2.5. They also propose maintaining current IP as the forwarding infrastructure on which their architecture will be founded. Following some of the biggest security concerns of these decisions are exposed.

4.5.1 Name Resolution Layers

As previously mentioned, LNA uses three resolution layers, from user-level to SIDs, from SIDs to EIDs and from EIDs to IPs. The first layer is implemented as a search or lookup service while the other two are implemented using DHTs. As previously mentioned in Sec. 4.2, DHTs suffer from several security weaknesses specific to them.

Another concern about the resolution layers involves the concept of delegates introduced in the paper. If an attacker could subvert a DHT record in order to alter the delegates related to a service or host, adding one of his own controlled machines, it could allow him to sniff all the traffic being the man in the middle without the victim ever noticing, as it is the task of delegates to redirect the traffic to the original service provider.

While the implementation of the DHTs and their algorithms are not explained in the paper, it is important to remark the possible attacks and to try to implement the resolution layers in such a way that will be resistant to those

in order to improve the security of the system. It is not the aim of this Thesis to go deeper into DHT security issues, and for further information the reader is encouraged to consult [28, 24, 9].

4.5.2 Forwarding

LNA includes SIDs and EIDs into the packet headers in order to perform some of the routing, like when service composition is needed, by stacking several SIDs in the same header, but ultimately, the routing is done at IP level.

IP introduces several vulnerabilities previously mentioned, such as spoofing being easily undetected or DoS/DDoS attacks being quite possible. By using IP, LNA introduces all these vulnerabilities into their architecture too, as even if services and end-points are represented as SIDs and EIDs, it is still possible to reach whatever host by means of its IP address, making possible the delivery of unwanted traffic or huge amounts of non-solicited data.

The fact that SIDs or EIDs can be stacked at will by the senders in order to allow them to dictate the path of packets doesn't make it any better as not only malicious users can send data to selected victims, but they can also choose the path the data will follow.

4.6 I3 Security Analysis

The Internet Indirection Infrastructure is an overlay architecture over regular IP, which apart from solving problems of mobility, multicast and anycast, aims at solving some security issues affecting Internet, such as DoS attacks. In order to do so, the overlay provides several mechanisms that improve the current Internet's situation, such as hiding IP addresses, giving end-hosts control against attacks and mechanisms to avoid new vulnerabilities that might arise. Following, we will analyze each of the three mechanisms.

Hiding IP addresses

In order to avoid DoS attacks at the IP level, I3 proposes nodes and end-hosts not disclosing their IP addresses by communicating exclusively over IDs. In order for this measure to work, it is assumed that all the nodes using

the overlay will only communicate through it, so theoretically, end-hosts communicating exclusively over I3 would be safe. While this approach can be seen as effective, relying on *"security through obscurity"* has been proven several times not a good way to offer protection. It is really difficult that nodes will only communicate through the overlay, as there can be services that are not part of it and that the end-hosts need to use. Also, while I3 is coexisting with the regular Internet, mechanisms such as DNS should still be working, so it shouldn't be difficult to discover the IP of the desired victim.

With this protection model, DoS attacks to random end-hosts and spam sending are still possible, as the attacker doesn't need any specific IP. Also, attacking directly I3 nodes storing triggers can render all the end-hosts storing their triggers on that node unreachable.

Giving more control to end-hosts

In order for end-hosts to be able to stop attacks by themselves, I3 proposes that end-hosts under attack should be able to remove their private or public triggers. In the case of private triggers, removing it would stop the attack completely without any side effect, as private triggers are used to communicate 1-to-1, and the rest of legitimate users will still be able to communicate through their own private triggers.

In the case of removing a public trigger in order to stop a flooding attack, the hosts already communicating through private triggers would not be affected, but the server would become unreachable for new clients until a new public trigger is inserted.

Protecting against new vulnerabilities

In order to protect against new vulnerabilities, I3 introduces the concepts of constrained triggers, pushback mechanisms and trigger challenges, as explained in Sec. 2.6. A detailed explanation on these three mechanisms is available at Adkins et al. [1].

4.7 DONA Security Analysis

Data Oriented Network Architecture makes use of an anycast primitive and a route-by-name approach in order to redesign the naming and name resolution from a clean-slate state. As explained in Sec. 2.7, it uses FIND and REGISTER messages in order to point out were content can be found. This messages are handled by some core entities called RHs, which provide much the same functionality as rendezvous nodes in PSIRP. In fact, the find and register messages are forwarded from the lowest RHs to the root or Tier-1 RHs in a very similar fashion that PSIRP forms rendezvous networks and forwards subscriptions and publications.

While all these previously operations and messaging is done through the RHs and not over IP, once the nearest available copy of data has been found, the rest of packet exchanges is done over IP. As mentioned in previously analysis, as long as forwarding over IP is still enabled, DoS attacks are still well possible, as data can still be sent to unintended recipients using IP directly.

To cope with certain attacks like bandwidth DoS attacks, DONA relies in IP-level mechanisms that can throttle down unwanted packet streams. This is equivalent to the current level of protection against DoS attacks which exists in the existing Internet architecture, and still doesn't solve the issue of DDoS attacks, where the unwanted packet streams are not only one but many, and come from many different machines.

For resource exhaustion attacks against RHs, DONA relies on contractual limits on the amount of FIND or REGISTER messages that a customer can send per minute, imposed by the providers.

Finally, in order to avoid malicious RHs, DONA allows clients to request the data not from the closest copy but from the n'th closest copy. While this helps avoiding misbehaving RHs, it could be used by an attacker owning a bot-net in order to request a lot of copies from the same RH, resulting in a resource exhaustion attack against that RH.

4.8 Postcards from the Edge Security Analysis

Similar to some of the other architectures analyzed, Postcards from the Edge still maintains IP in order to carry some of its tasks, such as control mes-

saging. This brings the same problems as stated in the previous analyses, as it is still possible to send unwanted traffic via IP, meaning that DoS attacks are then still possible.

As the paper doesn't addresses the security issues that might arise from the architecture nor describes any explicit security mechanisms, the vulnerabilities that are to be explained following cannot be fully proven.

Regarding the cache-and-forward nodes, it is unclear which amount of storage capacity they have, but it is obvious that it should be limited. If we assume this, it could be well possible to send very huge files with the TOS byte set to popular, in order to steal caching space or in order to completely fill the storage capacity of some node. Also, the files that are waiting in a queue of a forward-and-cache node are sent with a different priority according to their TOS byte. This fact could be abused by misbehaving users in order to get their files sent with a higher priority than others.

Post Office addresses are stored as an extra record in regular DNS. As long as it is possible to perform a DNS cache poisoning attack in the current Internet, it is also possible modify the PO addresses of a mobile node in oder to isolate it or to retrieve data that was intended for it.

Also, by performing a denial of service attack using IP against the POs of a mobile node, which can be retrieved from DNS, we can perform a DoS attack against that mobile node meanwhile it is offline, as any file that was intended for it will never reach its destination. This opens new venues for DoS attacks, as now it is not even needed that the victim is online, but it suffices performing a DoS attack against its PO nodes in order to effectively DoS the victim as well.

4.9 Architectural Similarities

The following table spots the basic similarities between the different architectures introduces in Chapter 2. It is intended only to get a better overview on the shared characteristics between the different architectures. It is left for future research the question on whether it can be proven useful to extrapolate certain attacks between architectures and to improve it further more.

Architecture	Pub/Sub	IP use	Extra resolution layers	DHTs
PSIRP	yes	no	yes	yes
Scribe	yes	no	no	yes
AID	no	no	yes	no
NNC	yes	no	no	no
LNA	no	yes	yes	yes
i3	yes	yes	yes	yes
DONA	no	yes	yes	no
Postcards	no	yes	yes	no

Chapter 5

Discussion

> *"A scientist's aim in a discussion with his colleagues is not to persuade, but to clarify. "*, Leo Szilard.

This Chapter presents a short discussion about the outcomes and shortcomings of the security analysis of the Future Internet architectures performed in Chapter 4.

The aim of this Thesis work was to assess the security of several proposals for Future Internet architectures. More concretely, the security analysis should focus mainly in DoS/DDoS protection mechanisms and possible vulnerabilities on them, as it is the most prevailing kind of attack on Internet and most of the architecture proposals try to address that issues. To this extent, we could say that the objective of the Thesis has been accomplished, as vulnerabilities that allow attackers to carry out this kind of attacks previously mentioned have been found in most of the architectures.

Regarding the feasibility of those attacks to pose a real threat against the given architectures or not, and their verifiability, the subject can be discussed further. In one hand, most of the described vulnerabilities seem to be consequent with what it is written on either the scientific papers, the RFCs or the project deliverables that have been employed to conduct the analysis and to gain a good understanding on how the architectures work and which are the mechanisms implemented in order to protect them against this kind of attacks. In the other hand, the lack of practical verification in a live test environment and the different implementations that some architectures provide for the same architectural element(s), added to the, sometimes, lack of some details and/or information on how that mechanisms work exactly,

could render some of the discovered vulnerabilities partially unusable. Nevertheless, even in those cases, the aforementioned vulnerabilities represent a good starting point and definitely should be taken into account.

Another issue that can be subject of discussion is whether the analysis is complete enough or not. While it is true that the security analysis could have been much more deep, and could have included also a wider variety of attacks, time constraints and the need to focus more in a particular attack, as focusing in many different kind of attacks would have resulted into the security analysis being less detailed and probably too general to render significantly useful results, have shaped it this way. It remains as an open possibility for the future to investigate other kind of attacks and to revisit the ones already explained in order to add more details or to double check them in order to assure as much as possible their feasibility, for example by conducting tests on prototype implementations.

The difference on the extension of the security analysis from architecture to architecture is due to the amount of technical detail found for each of them, the quantity and quality of their security mechanisms and their relevance to the Thesis. It doesn't mean that some architectures have been analyzed in a worst way or with less dedication than others, they have been all analyzed at the same level and employing the same methods.

Regarding the possible solutions presented for some of the vulnerabilities found, they have not been studied in depth and they don't present enough detail level in most of the cases. It was not the aim of this Thesis to give detailed solutions but to point out possible vulnerabilities in the architectures, so those solutions should be considered only possible approaches that are subject to further study and that may set up a good starting point.

A general result that can be derived from the security analysis is that the forwarding plane is one of the most critical components of an architecture. Forwarding is in charge to deliver data to a recipient in last instance. If there are vulnerabilities at the forwarding level that can allow an attacker to send non-solicited data to a victim, whatever other security mechanisms implemented at other levels may be rendered useless. This is specially obvious in the case of architectures that make use of IP level forwarding as a part of their implementation. As long as an attacker is able to send unsolicited data to a victim using IP directly, DoS attacks are still possible, despite any other security mechanisms implemented at other levels, as can be seen for example in Sec. 4.5 or 4.6.

Last but not least, is important to understand that security is not something that may be added to an architecture after the complete development process is done. This is not only a dangerous idea, but also a great increase on the development effort, as patching an already implemented architecture requires much more work than integrating security into it from the very beginning. Some people may argue that integrating security all the way through the development process is costly as well, and it does require some extra time and effort, but it is proven to be much more effective at the end.

Chapter 6

Conclusion and Future Work

> *"I think and think for months and years. Ninety-nine times, the conclusion is false. The hundredth time I am right."*, Albert Einstein.

This Chapter presents the conclusion of this Master Thesis and proposes some directions for future work.

6.1 Conclusion

In this Thesis, we have presented several Future Internet architectures. Some of them present a complete clean-slate design for the Internet while others focus on redesigning only certain parts of it. We have started by giving an overview about their technical description, their desired properties, their objectives and their approach to the problem. Then we have analyzed from a security point of view their main architectural elements such as forwarding, topology and the like. Finally, we have discussed the results obtained in the security analysis part and presented some possible solutions for the security issues that have arisen.

The analysis has revealed some potential security flaws in many of the architectures, most of them located on the forwarding plane. It has been proven that when designing and implementing a new architecture, security has to be integrated since the very first moment and that it is always better to aim for a clean-slate architecture that for a redesign of the existent one. While many of these Future Internet architectures have shown to improve the cur-

rent security status of the Internet model, total security is very difficult to achieve, and new designs can always introduce new venues for attacks.

However, most of these architectures are still in a design phase and changes are being introduced every now and then, leaving still room for security improvement.It is our hope that the results obtained in the analysis will help them in that improvement.

6.2 Future Work

Although the security analysis of this Thesis has been performed in all the architectural elements described for each architecture, it has been focused mostly on DoS/DDoS vulnerabilities. It could be useful as a future work to analyze those elements focusing in other kind of vulnerabilities.

Also, all the results obtained have been purely theoretical, based on what the papers, RFCs and project deliverables state. It could be subject for future work to reproduce those vulnerabilities found in a live environment, in order to verify if they work as intended and to investigate further how they work and other possible venues for attacks.

Some of the solutions presented in Sec. 4 can be subject of further study and can be an interesting starting point to improve some of the security mechanisms of the architectures analyzed.

Last but not least, despite being a quite ambitious task, all the information gathered in this Thesis could be used to try to design a new architecture that will reunite all the worthy features present on them, while taking into account all the vulnerabilities found, in order to develop an even safer architecture.

Bibliography

[1] Adkins, D., Lakshminarayanan, K., Perrig, A., Stoica, I., Adkins, D., Lakshminarayanan, K., Perrig, A., and Stoica, I. Towards a more functional and secure network infrastructure. Tech. rep., 2003.

[2] Ahlgren, B., Arkko, J., Eggert, L., and Rajahalme, J. A node identity internetworking architecture. In *INFOCOM* (2006), IEEE.

[3] Andersen, D. G., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D., and Shenker, S. Accountable internet protocol (aip). In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication* (New York, NY, USA, 2008), ACM, pp. 339–350.

[4] Balakrishnan, H., Lakshminarayanan, K., Ratnasamy, S., Shenker, S., Stoica, I., and Walfish, M. A layered naming architecture for the internet. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2004), ACM, pp. 343–352.

[5] Caesar, M., Condie, T., Kannan, J., Lakshminarayanan, K., and Stoica, I. Rofl: routing on flat labels. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2006), ACM, pp. 363–374.

[6] Castro, M., Druschel, P., Ganesh, A., Rowstron, A., and Wallach, D. S. Secure routing for structured peer-to-peer overlay networks.

[7] Castro, M., Druschel, P., Kermarrec, A.-M., and Rowstron, A. Scribe: A large-scale and decentralized application-level multicast

infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC 20* (2002), 2002.

[8] Clark, D., Braden, R., Falk, A., and Pingali, V. Fara: reorganizing the addressing architecture. In *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture* (New York, NY, USA, 2003), ACM, pp. 313–321.

[9] Douceur, J. R. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems* (London, UK, 2002), Springer-Verlag, pp. 251–260.

[10] Fischer, A. Deliverable 3.1: Taxonomy of p2p, overlays and virtualization techniques with respect to service resilience, 2009. project deliverable resumeNet.

[11] Ford, B. Unmanaged internet protocol: taming the edge network management crisis. *SIGCOMM Comput. Commun. Rev. 34*, 1 (2004), 93–98.

[12] Gupta, A., Sahin, O. D., Agrawal, D., and Abbadi, A. E. Meghdoot: content-based publish/subscribe over P2P networks. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware* (New York, NY, USA, 2004), Springer-Verlag New York, Inc., pp. 254–273.

[13] Handley, M. Why the internet only just works. *BT Technology Journal 24*, 3 (2006), 119–129.

[14] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. Networking named content. In *CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies* (New York, NY, USA, 2009), ACM, pp. 1–12.

[15] Jokela, P., Zahemszky, A., Arianfar, S., Nikander, P., and Esteve, C. LIPSIN: Line speed Publish/Subscribe Inter-Networking. In *ACM SIGCOMM* (Barcelona, Spain, August 2009).

[16] Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. A data-oriented (and beyond) network architecture. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2007), ACM, pp. 181–192.

[17] Lagutin, D. *Redesigning Internet—The Packet Level Authentication Architecture*. Licentiate's thesis, Helsinki University of Technology, Department of Information and Computer Science, 2008.

[18] Mödeker, J., and Marikar, A. Deliverable 1.1: System deployment scenarios and use cases for cognitive management of future internet elements, 2009. project deliverable Self-Net.

[19] Moskowitz, R., and Nikander, P. Host Identity Protocol (HIP) Architecture. RFC 4423 (Informational), May 2006.

[20] Ohlman, B. Deliverable d6.1: First netinf architecture description, 2009. project deliverable 4WARD.

[21] Paul, S. Postcards from the edge: A cache-and-forward architecture for the future internet. NeXtworking07, 2nd COST-NSF Workshop on Future Internet, Berlin, Germany, April 2007.

[22] Rothenberg, C. E., Jokela, P., Nikander, P., Sarela, M., and Ylitalo, J. Self-routing denial-of-service resistant capabilities using in-packet bloom filters. In *the 5th European Conference on Computer Network Defense (EC2ND)* (2009).

[23] Rowstron, A., and Druschel, P. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)* (Nov. 2001), pp. 329–350.

[24] Sit, E., and Morris, R. Security considerations for peer-to-peer distributed hash tables. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems* (London, UK, 2002), Springer-Verlag, pp. 261–269.

[25] Stoica, I., Adkins, D., Zhuang, S., Shenker, S., and Surana, S. Internet indirection infrastructure. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2002), ACM, pp. 73–86.

[26] Tarkoma, S. Deliverable d2.3: Architecture definition, component descriptions, and requirements, 2009. project deliverable PSIRP.

[27] Trossen, D. Deliverable d2.4: Update on the architecture and report on security analysis, 2009. project deliverable PSIRP.

[28] Urdaneta, G., Pierre, G., and van Steen, M. A survey of DHT security techniques. *ACM Computing Surveys* (2009). `http://www.globule.org/publi/SDST_acmcs2009.html`, to appear.

[29] Yang, X., Clark, D., and Berger, A. W. Nira: a new inter-domain routing architecture. *IEEE/ACM Trans. Netw. 15*, 4 (2007), 775–788.