SEVENTH FRAMEWORK PROGRAMME

THEME ICT-1-1.1

"Network of the future"

Project acronym: EFIPSANS

Project full title: Exposing the Features in IP version Six protocols that can be exploited/extended for the purposes of designing/building Autonomic Networks and Services

Proposal/Contract no.: INFSO-ICT-215549

GANA threat and trust models

Project Document Number: EFIPSANS/D3.5 PU[1] v1.0

Project Document Date: 31/12/2009

Workpackage Contributing to the Project Document: WP3

Deliverable Type and Security: R/PU

Editor: Sheila Becker (UL)

[1] Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Abstract: In this document, we analyze the GANA architecture from a security point of view, and based on this we first define a threat model. Then, we also compare different existing trust models in different scenarios and we explain how trust can be used within GANA framework. We introduce two types of Security Decision-Elements that are used for integrating the security mechanisms and functions in the GANA architecture. Data mining techniques are described, that are used for providing self-defending mechanisms that are accomplished by the proposed Decision-Elements. Furthermore, Vulnerability Detection for IPv6++ protocols is envisioned using Fuzzing to efficiently detect faults by providing unexpected input with the goal to harden the protocols.

Keywords: GANA, threat model, trust model, ONIX, data mining, fuzzing, EFIPSANS

Copyright

January 14th, 2010

Project Number	**INFSO-ICT-215549**
Project Name	**E**xposing the **F**eatures in **IP** version **S**ix protocols that can be exploited/extended for the purposes of designing/building **A**utonomic **N**etworks and **S**ervices
Document Number	**INFSO-ICT-215549**/WP3/D.3.5v1.0
Document Title	GANA threat and trust models
Workpackage	WP3
Editors	Sheila Becker (UL)
Authors	Vassilis Merekoulias (ICCS) Yacine Rehabi (FOKUS) Krzysztof Cabaj (WUT) Sheila Becker (UL)
Reviewers	Ranganai Chaparadza (FOKUS) Kristian Slavov (LMF) Peter Schaffer (UL) Timotheos Kastrinogiannis (ICCS)
Contractual delivery date	31st December 2009
Delivery Date	31st December 2009
Version	1.0

Executive Summary

This deliverable reports on the work and outcome produced mainly by the security task T3.4 "*Security issues in autonomic IPv6 Networks*", which aims at securing the network and application functionalities of the GANA architecture (described in detail in EFIPSANS deliverable D1.5), defined by EFIPSANS. The objective is to secure the optimized communication in multiple dimensions and provide secure services of high quality, and functionality as well as ubiquitous access. In accordance with the project's Description of Work in this Deliverable the following issues/objectives are covered:

(a) A detailed security model concerning GANA architecture and specific autonomic functionalities developed with project's framework will be proposed emphasizing on identifying risks and providing remedies.

(b) Different levels for the threat model are considered according to their impact in accordance to which various threats concerning Self-* functionalities (e.g. Self-Description, Auto-Dissemination (Self-advertisement of Capabilities Description Information)) are studied and classified.

(c) Concrete scenarios will be proposed investigating the corresponding security dimensions.

(d). Initial description of Security Management Decision Element

i. Trust Functions

ii. Authentication & Authorization

iii. DoS prevention

iv. Vulnerability Detection in IPv6 Protocols using Fuzzing

The structure of this deliverable reflects the above as follows. Chapter 3, refers to the the EFIPSANS specific analytical GANA threat model [**Deliverable's Objective: (a) and (b)**], while in Chapter 4, the EFIPSANS trust model in relation to GANA is described extensively [**Deliverable's Objective: (a) (c) and (d)**]. Furthermore, Chapter 5, details on the security functions important to EFIPSANS and GANA [**Deliverable's Objective: (c) and (d)**].

This deliverable covers the research work performed by Task T3.4 from M12 to M24. This is a final deliverable due in month M24 (Dec '09) and focuses on the development of Framework F11**.**

Table of Contents

1 INTRODUCTION

The objective of the security task T3.4 is to secure the network and application functionalities, provided by EFIPSANS, optimizing communication in multiple dimensions and providing services of high quality, and functionality, as well as ubiquitous access. Specifically, the objective of this task is to investigate and address security issues, concerning trust models (in sense of architectural components relations) and vulnerability in autonomic IPv6 enabled networks. Furthermore, security issues directly related to GANA architecture are studied and addressed via producing an analytical GANA threat model as well as proposing and specifying methods towards fortifying GANA.

One possibility to address security in an environment is to ensure three principal characteristics: Confidentiality, Integrity, and Availability. These three key words are known under the abbreviation CIA. All the analyses and propositions are made with the goal to have an environment where CIA is guaranteed.

The first step towards this procedure is to analyze the network and application functionalities. Therefore, we need to take a deep look into the proposed functionalities and architectures and identify the threats that are possible to occur. In EFIPSANS, the architecture GANA (Generic Autonomic Network Architecture) is proposed for modeling and specifying autonomic behavior. So, we focus on this architecture for creating the threat model.

In this document we also study trust-mechanisms and explore different algorithms. As confidentiality and integrity are important topics for security, it is needed to adopt a trust model in GANA. We compare different algorithms with the help of peer-to-peer based simulations. We also define the properties that are adaptable to the GANA architecture, to have an optimal trust mechanism.

The aspect of availability is handled in the proposed security functions. Towards enabling various security functions, we introduce two security management decision elements. One decision element works on node level, and the other on network level. We address also access control in relation to GANA and ONIX. We also propose self-defending mechanisms for autonomic networks using data mining techniques. Vulnerability detection for enhanced IPv6 protocols using fuzzing is also integrated as security function.

1.1 Scope of the Deliverable

The scope of the deliverable (i.e. objectives (a)-(d)) is to describe and clarify how security issues related to the various developments within EFIPSANS are addressed. This deliverable shows how we analyzed the GANA architecture from a security point of view, and how a threat model has been adopted for this architecture. Furthermore, it introduces the use of Trust in relation to GANA. It also illustrates the integration of security functions, as the definition of security decision elements, access control in relation to GANA, self-defending mechanisms, and vulnerability detection using fuzzing.

1.2 Structure of the Document

After giving an introduction to this deliverable, Chapter 2 contains all relevant terminology, by listing all abbreviations as well as provides the most important definitions, towards enhancing the understanding of this document. Thereafter, Chapter 3 explains and details our analytical GANA threat model. In Chapter 4, the proposed trust model in relation to GANA is described extensively. Furthermore, Chapter 5 details on the security functions important overall to EFIPSANS and GANA.

2 TERMINOLOGY

2.1 Abbreviations

AAA	Authentication, Authorization, and Accounting
ACL	Access Control List
CIA	Confidentiality, Integrity, Availability
DDoS	Distributed Denial of Service
DE	Decision (-making) Element
DHCPv6++	enhanced Dynamic Host Configuration Protocol version 6
DHT	Distributed Hash Table
DoS	Denial of Service
DSCP	Diffserv Codepoint
FSM	Finite State Machine
FTP	File Transfer Protocol
GANA	Generic Autonomic Network Architecture
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ICMPv6++	enhanced Internet Control Message Protocol version 6
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IP	Internet Protocol
IPv6	Internet Protocol version 6
ISP	Internet Service Provider

KDD	Knowledge Discovery in Databases
ND++	enhanced Neighbor Discovery protocol
NL_DE	Network Level Decision Element
ONIX	Overlay Network for Information eXchange
OS	Operating System
P2P	Peer-to-Peer
PKI	Public Key Infrastructure
QoS	Quality of Service
RED	Random Early Detection
ROCQ	Reputation, Opinion, Credibility, and Quality
RSVP	Resource reSerVation Protocol
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
ToS	Type of Service
UDP	User Datagram Protocol
VoIP	Voice over IP
WRED	Weighted Random Early Detection
XML	Extensible Markup Language

2.2 Definitions

Autonomic Behaviour (AB):

In GANA, we define an **Autonomic Behaviour** (AB) as a is a *behaviour* or *action* that may consist of a set of sub-behaviours or sub-actions triggered by a Decision-Making-Element (DME or a DE – in short) in an attempt to achieve the goal defined by how the Decision-Making-Element manages a Managed Entity (ies) – ME(s) under its control. The autonomic behaviour is considered as the behaviour of a DE, triggered as a result of reception of information from its information suppliers such as its associated Managed Entity (ies) in an attempt to regulate or reconfigure the behaviour of the Managed Entity (ies), ***OR*** starts as the behaviour spontaneously triggered by the DE. A behaviour triggered spontaneously by a DE is simply a spontaneous transition in the Finite-State-Machine describing the overall behaviours of the DE. An example of an autonomic behaviour is: ***self-description*** and ***self-advertisement***, ***self-healing***, ***self-configuration***, all triggered by a DE.

Black-box testing/fuzzing:

Testing or fuzzing a system, while having no knowledge of the mode of operation / source-code.

Bot:

Machine infected by malicious software allowing remote controlling and using it for all illegal activity, for example, performing scanning or DDoS and hosting phishing sites.

Botnet:

Group of bots that is controlled by one person or organization.

Data mining:

The most important step in Knowledge Discovery in Databases (KDD) that is responsible, for example, for discovery patterns, classification or clustering of data.

Frequent set:

One of the data mining patterns that represents a subset of items frequently appearing in analyzed data set.

Fuzzing:

Detecting vulnerabilities by generating unexpected input

Grey-box testing/fuzzing:

Testing or fuzzing a system with some knowledge of the exact mode of operation of the system

Incremental mining:

Data mining technique that allows obtaining new results when new data appear without recalculation of whole data.

Malware:

Abbreviation from “malicious software” used for all network treats like worms, bots and viruses.

Phishing:

Attack that persuades an innocent user giving personal data in forged site, for example, an attack that utilizes fake e-mail that directs user to malicious phishing site used for stealing personal data.

Phishing site:

Site used for phishing; it looks identically as attacked organization’s site used for gathering personal data, for example, login, password or credit card number, from innocent users.

Virus:

A malicious program that needs user interaction for propagating from one machine to another. It can take a form of specially crafted Web page, mail or executable program,

White-box testing/fuzzing:

Testing or fuzzing a system while having exact knowledge of the mode of operation of the system

Worm:

Network threat that can automatically infect other machines without user interaction using vulnerabilities in network software. In contrast, Virus needs person activity for infection.

3 ANALYTICAL GANA THREAT MODEL

3.1 Introduction

A threat model is an essential tool for analyzing security, risks, and mitigation approaches for an architecture. A threat model can be constructed in several ways, with solutions ranging from theoretical graph attacks/threat trees and up to methodology driven approaches like OCTAVE [ABPW99] or STRIDE [HLOS06]. We have followed a quantitative and pragmatic approach, consisting in the identification of threats, the assessment of their impact, and the proposal of mitigation actions.

The first part of this Chapter introduces the GANA architecture in particular the related self-* functionalities. The second part briefly discusses security in self-managing networks as well as the well known security threats that affect the current Internet and which need to be addressed in self-managing networks.

3.2 Background

3.2.1 The GANA architecture: a short overview

In this section, we would like to recall some of the aspects of the GANA architecture. GANA refers to Generic Autonomic Network Architecture and allows to model and to specify the autonomic behaviors while designing Future Internet. The GANA architecture considers different levels of abstraction of autonomic networking functions, namely:

- Self-routing
- QoS self-management
- Self-forwarding
- Self-dissemination of information
- Self-description of capabilities
- Self-configuration

The GANA architecture also considers hierarchical levels of control loops and their associated Decision-making Elements (DEs) for self-manageability. It also includes peering relationships between Decision-making Elements (DEs) that determine the autonomicity of a node(s).

For this document, we only consider the self-description, self-dissemination, and self-configuration for elaborating the GANA threat model as these functionalities are the key self-* functionalities in the GANA architecture. For more details, we refer to section 3.3.

3.2.1.1 The self-description functionality

Self-description is the ability of a functional entity to describe itself and more precisely, to describe its capabilities and its possible potential and current role(s) being played while in operation, as well as supported protocols and how to use its services. What must emerge as a result of self-description is a Capability Description Model. This can be achieved using a description language like XML. The Node-Main DE of a node/device should aggregate the capabilities of all the functional entities of the node/device and self-describe the capabilities by creating a Capability Description Model. More details about the aggregation of capabilities by the Node-Main DE and the self-description process please refer to [D2.2].

3.2.1.2 The self-dissemination functionality

Self-advertisement is the process by which a functional entity (especially a network Node/Device) spontaneously disseminates its description using a Capability Description Model to some other functional entities or in response to a solicitation from another entity. The dissemination can be done over a distributed dissemination system or architecture such as the ONIX system being developed by EFIPSANS, as well as via EFIPSANS proposed extensions to the Neighbor Discovery protocol (ND++) for allowing on-link neighbors to share information about their Capabilities directly.

3.2.1.3 The self-configuration functionality

In the context of GANA and the EFIPSANS project, Self-Configuration (Auto-Configuration) is defined as the ability of a network and its devices to configure themselves without any manual intervention. Thus it refers to the ability of the network and its devices to configure its functionalities based on some *Network Profiles*. A *Network Profile* in the context of EFIPSANS and GANA is defined as a detailed, structured and monolithic composition of all the information required to configure the network and realize its network goals and objectives. Thus a *Network Profile* is a direct translation of a set of textual business goals of a network into sophisticated and structured technical goals of the network. A *Network Profile* can thus be considered as composition of *Policies*, *Objectives*, *Topology Information* and *Configuration Data.* More details about the Network Profiles and the Auto-Configuration functionality can be found in [D2.2] and [D2.4].

3.2.1.4 ONIX

ONIX – Overlay Network for Information eXchange – is the EFIPSANS proposed solution for scalable and fault-tolerant resource discovery (where by resource we understand devices or services offered by devices, but also data in the form of configuration data, network policies, incidents description, monitoring data, etc.) in large-scale environments. ONIX is able to handle sophisticated resource description and queries. The descriptors of the stored resources are semi-structured XML and ONIX supports partial queries based on the XML descriptors (ONIX will resolve queries even in the case where queries contain just a subset of attributes originally advertised). Also each query can have a scope (global, local, n-hops away, etc). ONIX is an autonomic system and is able to adapt to the changes in the operating environment, including changes in resources state and network attachment point. ONIX is design to scale to very large numbers of resources spread throughout a wide network across different administrative domains. To achieve the above goals, ONIX relies on an efficient distributed hash table process (DHT) which it uses as a building block. ONIX system is independent of the underlying DHT protocol, but it inherits the benefits of a P2P systems.

Different nodes/devices that are interested in using the services offered by the ONIX framework will use one of the ONIX external protocols. These include the protocols used by the Node_Main_DE or by the different Network_Level_DEs for interacting with ONIX. DHCPv6++ can be used to access the ONIX services during the bootstrap time and probably an application level protocol will be design for the interaction during the normal operation time of the network. The first draft of DHCPv6++ is presented in [D2.3]. The resource descriptors will be replicated across the system so that ONIX will be able to resolve queries even in the case of failures of several nodes. The information that is meant to be stored in ONIX needs to have relative mid or long-term validity. For very fast exchange of information between some entities, other mechanisms need to be in place. More details about ONIX system can be found in [D2.2].

3.2.2 Security in self-managing networks

As EFIPSANS is about future self-managing networks, it is wiser to expose at the beginning some facts about security in self-managing networks in contradiction with security in traditional networks. This will help in better understanding the upcoming sections. The main facts about security in self-managing networks that have to be stated are the following,

- As self-management is considered as one of the main characteristics of Future Internet, security in this case has to be a requirement and not an added value service. *The experience has shown that it is difficult to add security to a protocol suite unless it is built in into the architecture from the beginning.*
- Security in self-managing systems will not be an entirely new kind of security.
- Traditional security issues will also appear in self-managing systems under the same or a different form.
- New threats proper to self-managing systems might emerge.
- Self-managing also means an increase in the needs for communication and cooperation among entities, within a node or/and among nodes.
- The new characteristics of self-managing systems will also help in making our systems more secure.
- Policies are basic stones in building autonomic systems.

If we check in the literature, security is a topic that has been widely addressed via and through multiple ways, even non-appropriate ones. In this part, we are not going to discuss the entire solutions, however, we will just go through the characteristics of some solutions and thus, we will reveal the corresponding limitations that they are suffering from with respect to autonomicity. These characteristics are summarized in the following table,

Current Internet security solutions	*Related issues*
Security policies are preconfigured to a static behavior	Unable to be seamlessly adapted dynamically to new constraints
Anomaly detection scenarios noticed by humans or specialized static s/w & h/w	• Unable to deal with a constantly changing environment • Unable to detect new attacks and recover from security problems
Passive reaction by relying on the decision of human	Long delay until action, which allows additional attacks or make the situation worse
Security is placed in each layer and application whenever it is needed	Efficiency and performance are questionable, Network does not provide Security

3.2.3 Threats that can be easily detected

Some network activities cause generation of additional amount of traffic. Consequently, these threats may be detected by monitoring of network traffic. Detection of these threats by the self-defending functionality of NODE_LEVEL_SEC_MNGT_DE is described in subchapter 5.4. The description of this kind of network threats and the characteristic of the traffic that is produced by them is the outlined in the following. Of course not of them are in fact attacks, for example, scanning or sudden rise of traffic destined to the particular machine. However, detection of this activity may be a first sign that something strange happen. If some of these activities are usual for certain machines, properly configured policy can be used for disabling false positives.

3.2.3.1 Scanning (Recon)

Scanning is a process of discovering either running machines or services provided by these machines. Depending on the purpose of scanning, there are many scanning techniques which involve different protocols and produce various patterns. In many cases the detected scanning pattern can describe both an attacker and its aim.

3.2.3.1.1 Finding running machines

The first step in most malicious activities is a detection of running machines in a given IP address range. For this purpose, the simplest scanning method that uses ICMP packets can be utilized. An ICMP echo request packet is sent for each IP address from the given address range. Each machine that receives such a packet should respond with an ICMP echo reply packet. This kind of scanning can be detected by the appearance of many ICMP packets to consequent IP address.

3.2.3.1.2 Enumerate running services

If IP addresses of running machines are known, the next step is often conducted. In this step an attacker tries to enumerate all running services on the given machine. Such activity produces a visible sign and can be generated by using many programs. One of the most known network scanners is called Nmap [NMAP]. However, not only users' intended actives can lead to scanning process. Today, in order to finding vulnerable machines, worms or bots perform scanning just after an infection. Skilled attackers can use stealth scanning techniques that do not leave any signs in application logs. These methods use protocols features that tear down connection during 3-way handshake (for example, nmap syn stealth scanning [F97]) or use packets with a specially crafted set of flags (fin stealth scanning). During scanning, a number of ICMP packets with icmp_type set to 3 (port unreachable) can be observed.

3.2.3.2 DoS/DDos attack

A successful Denial of Service attack causes that legitimate users cannot access resources that in normal circumstances are granted to them. Bandwidth, machine CPU, or access to service in remote machine can be such a resource. Currently, many types of Denial of Service attacks, which use various kinds of traffic, can be observed. The simplest attack is caused by many packets destined to victim. This attack can overload victims' bandwidth or CPU time. In this kind of attack, either ICMP or UDP protocol can be utilized. This version of DoS attack uses vast amounts of packets that can be easily detected in monitored traffic. However, clever attacks can use lower number of packets. These attacks use some features of an attacked protocol. The most known attack of this kind, called SYN flood, uses TCP handshake procedure. Many packets with set SYN flag are sent to an attacked port of the victim. Accordingly to TCP handshake procedure, a machine responds with a segment, which has set both SYN and ACK flags. An attacker never responds to this packet (in many cases source IP address is spoofed and there is no machine with such address), but victim repeatedly resend request using valuable resources. Additionally, in some implementations of network stack a number of such connections in a half-open state are limited, and when the limit is reached any, even allowed connections, cannot be established.

As botnets appear more often nowadays, a new kind of denial of service attack can be observed. In all above mentioned DoS attacks there is one attacker that sends hostile traffic. The attacker can be easily detected because it sends many packets to the victim machine. In recent modifications that utilize Botnets many attackers are present, which send traffic to the victim. Due to this behavior, this attack is called Distributed DoS. As this kind of attack is performed by a vast amount of machines, it is difficult to stop it. Traffic from many machines should be filtered or many machines must be localized and cleared from malicious software.

3.2.3.3 SPAM

Very often an infected machine, which is part of a Botnet, starts sending a vast amount of e-mail messages. This activity is associated with a well known problem - SPAM. These messages can be either a form of marketing or the first step in phishing activity, which can lead to stealing personal account information. This activity leads to many TCP connections to port 25. This is a well-known port of SMTP protocol used for exchanging mail messages between mail servers.

3.2.3.4 Hosting malware/phishing site etc ...

The infection mechanism of currently used worms and bots is similar. Some parts of this process can produce higher than average traffic, which helps detection of malicious machines. The description of each step is described below.

3.2.3.4.1 Exploitation of vulnerability

Vulnerability exploitation needs at least one connection or even one UDP packet [SQL Slammer]. If it concerns only one machine, it cannot be detected by described self-defending functionality, because it detects frequently appeared traffic pattern. However, when new vulnerability is detected, rapidly a new worm appears which uses it. In effect, this frequently appearing malicious activity can be detected by described self-defending functionality. This activity produces much traffic that can be detected by the proposed system which observes traffic destined and sourced from protected network. In this situation, detected traffic can be very specific, for example, a connection to a given protocol and port that exchanges strict amount of data.

3.2.3.4.1 Download of next stages

Currently used exploits contain only the most important instructions that lead to exploitation of vulnerabilities, so they are no larger than few hundredths of bytes. All other functionality is downloaded after a successful exploitation of the vulnerability. Due to this approach during spreading of malware, other machines that store a copy of next stages, are used. These machines run software that provides malicious files. The malware for this purpose can use many well know protocols, for example, TFTP, FTP, HTTP and even some custom made protocols. Malware can be hosted in one central machine or stored in each infected machine [PH08]. In the first case, detection is simpler because all infected machines try to download some files from one machine. In the second situation there are many machines that provide malicious files. Each infected machine provides the next stages for each machine that is attacked by it.

However, downloading next stages of malware causes that an attacked machine stores some data and makes it available to other Internet users. In addition, after an infection the attacker uses machines for its purpose and often these machines, especially with high bandwidth interfaces, are used for providing some services. The most common usage is associated with providing fake web sites of financial organizations, which are used for phishing, for example, for stealing users' passwords

Both of these situations cause an increase in traffic destined to a suspiciously behaving machine. In many cases detected traffic is destined to unusual ports, other than well known ports of such protocols like FTP, HTTP or TFTP. This simple technique used by attackers helps avoiding detection by IDS or IPS.

However, before going through this step, the detailed instruction must be obtained concerning what files should be downloaded from which IP address. For this purpose, many currently observed malware starts backdoor, a functionality that provides secret access to infected machine. The simplest solution, very often used in Windows platform, executes windows program called cmd.exe (OS command shell) and bind it to listening socket or connect back to the attacker.

3.2.3.4.2 <u>Finding other vulnerable machines</u>

Due to the first aim of both worm and Bot, which is rapid spread to the other victims, after infection and downloading of the main stage they start finding other vulnerable machines. This activity is identical to scanning for specific service running on machines. Details of this activity are presented under section 3.3.1.

3.2.3.4.3 <u>Infected machine activities</u>

When a machine becomes a part of a Botnet, any kind of hostile activity could be executed from it. Security experts that analyze Botnets activity in the Internet noticed that these machines are used for scanning, sending spam, performing DDoS attacks or even click on paid banners of their clients. Most of them are described in more details in above presented sections of subchapter 3.2.3.

3.3 The GANA Threat model roadmap

The first step being achieved within the EFIPSANS security roadmap was the development of the security GANA threat model. The latter is a proven methodology for figuring out the security vulnerabilities at the design phase of GANA. The GANA threat model allows a systematic approach to protect the GANA architecture from attacks and abuses while it is used for planning appropriate security solutions. Figure 1 simply depicts this model.

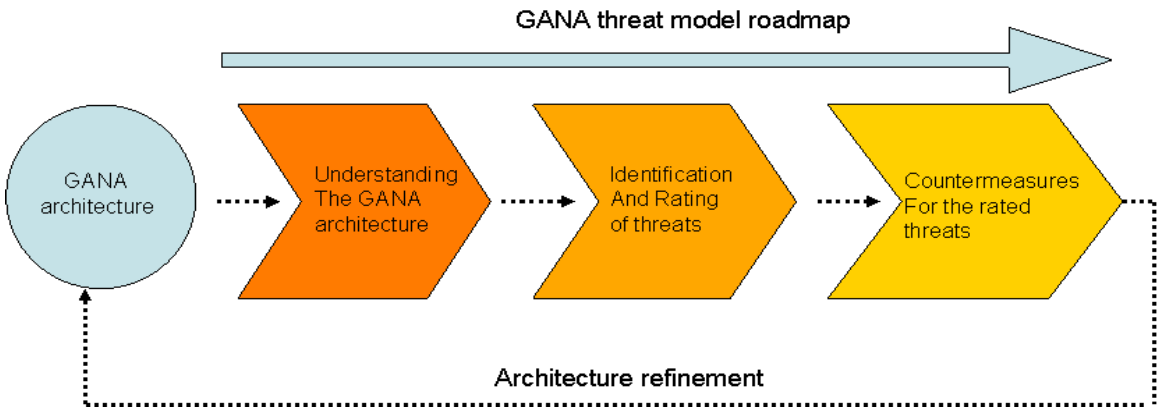

GANA threat model roadmap
GANA architecture
Understanding The GANA architecture
Identification And Rating of threats
Countermeasures For the rated threats
Architecture refinement

Figure 1: Threat model roadmap

In EFIPSANS, different kind of networks, functions, and protocols are being considered. Therefore, understanding and specifying what kind of security the EFPSANS framework needs is a hard task. Addressing all the security issues within EFPSANS cannot be achieved. Therefore, it is reasonable to direct the available resources to the assets and the innovative parts of the EFIPSANS framework, namely the security threats emanating from *"IPv6 and/or EFIPSANS specific architecture (protocols design, implementations used on entities, etc".*

It is very important to mention that the threat model built here is based on specifications discussed in the deliverable D2.2.

From our point of view, the GANA threat model has to address the following main questions,

- Which type of network/component/functionality needs to be protected?
- What kind of protection (detection and reaction) needed and against which threats?
- How to achieve the protection mechanisms in this new environment?

3.3.1 Threats Identifications

In short, the GANA architecture has specified some Decision Elements (DEs) components that need to advertise their capabilities, discover each other, and configure themselves through a central data repository called "ONIX". The mentioned mechanisms were explored, and the corresponding threats were identified. These threats were also classified based on how serious they are (see Table 1 in section 3.3.3).

A description of the methodology used to identify threats, we give examples based on the Self–Description, Self–Dissemination and Self–Configuration cases presented in section 3.2.

Within the EFIPSANS framework, specific entities capture a central role for the operation and the provisioning of services. Such entities should be analyzed and an architecture that prevents these important entities to be unavailable or compromised should be provided. For example, from the analysis of the current state of EFIPSANS proposed architecture specific bottleneck points were identified i.e. ONIX, Network Level DE, Node DE, Monitoring DE. Analysis of the above entities reveal that while ONIX is based on a distributed network while Network Level DE is not clear weather is implemented based on a single entity or not. Bottleneck points may be the target of future Denial (or Distributed Denial) of Service attacks or even the first points that malicious users will try to compromise. In any case bottleneck points should be avoided and relevant architecture that will enhance security (and resilience as a result) should be provided, within the EFIPSANS framework..

3.3.2 Threats Classification Criteria

In the analysis of the GANA architecture we classify the elaborated threats in different severity-levels. Different levels for the threat model are considered according to their impact:

- *Serious:* a threat is considered as serious if it affects bottleneck nodes or its impact can be difficult to recover. In this case, the whole network can collapse. For instance, if the ONIX collapses, the network will face problems.
- *Medium:* a threat is medium if it affects a set of nodes. This is the case when a node advertise to be a router and some other nodes will use it.
- *Low:* a threat is low if it affects a certain end node (end terminal). this is the case when a certain node attacks another node.

3.3.3 Risks and Remedies

In this section, we list the most crucial threats identified while analyzing the GANA architecture. We also rate these threats according to their impact. In addition to that, we discuss briefly some of the corresponding countermeasures.

	Threats	Level (serious, medium, low)	Security dimension
	Self-description		
1	False info about protocols and modules that are going to be used in self-description	Medium to high	Trust can be used here: there are different ways to achieve this, either by self-experience (testing regularly, previous knowledge) or using other nodes experiences
2	False info about role to be played, functionalities, interfaces and platforms to be supported	Medium to high	See 1
3	False info about cost to provide a certain service	Low	See 1
4	False info about monitoring data and format	Medium to high	See 1
	Self-dissemination		
5	Some malicious nodes might be in the neighborhood of a given node (secure discovery). Nodes may not want to disseminate info to malicious nodes.	Low	Disseminate a minimal set of capabilities – info and advertise more once trust is proven. Disseminate the capabilities to the trusted nodes (already authenticated and certified). A node might maintain a list of the trusted nodes around it. If some trust info is available somewhere (a central certification authority or/and near by nodes), it can also be used to ensure that these nodes are trusted
6	A malicious node might advertise wrong information to its neighbors	Medium to high	See 1

	Threats	Level (serious, medium, low)	Security dimension
7	A malicious node might flood its neighbors with capabilities advertisement messages	Medium to high	Flood detection mechanisms can be used Drop messages if the amount goes beyond a threshold. In any case there is no way to stop the malicious node to flooding the channel – line but the packets may be dropped before relaying and being processed.
8	A malicious node might ask for some capabilities description (that it needs) from its neighborhood	Medium to high	See 5
9	ONIX discovery: A fake ONIX	High	Authentication mechanisms of ONIX framework should be in place

	Threats	Level (serious, medium, low)	Security dimension
10	Malicious nodes might try to push faked capabilities to the ONIX	Medium to high	EFIPSANS, as a generic network framework, should be able to provide services and techniques in a variety of networks. This requires to the implemented techniques to be capable to provide services at networks that operate under a strict administrative domain as well as at ad-hoc network environments where no administrative authority exists. For example where a strict administrative domain exists only authenticated and trusted nodes should be able to push info into ONIX On the other hand if there is no administrative authority available, nodes may be authenticated by evidence of neighbor nodes. In any case ONIX must be able to provide information coupled with trust values according to the level of security and trust of data collected, based on the source of the data, network environment etc.
11	Malicious nodes might try to query some other nodes capabilities from the ONIX	Low to medium	See 5
12	Impersonate a certain node once its capabilities and its address are retrieved from the ONIX.	Low to medium	Identification of nodes and entities either by a central authority or by evidence of neighbor hooding nodes.
13	The malicious node might try to extend the lifetime of its capabilities in one manner or another just to push other nodes to use the corresponding wrong information	Low	Nodes should be identified and the refresh algorithms should be based on identified data. In case of identification by evidence data should be marked as such.

	Threats	**Level (serious, medium, low)**	**Security dimension**
14	A malicious node might flood the ONIX with updates regarding lifetime of its capabilities	Low	See 7
15	A malicious node might flood the ONIX with updates regarding its capabilities	Low	See 7
16	ICMPv6 (for Neighbour discovery) might have drawbacks that need to be investigated (related to the extension of the ND protocol). It seems that the SeND protocol is going to be used, so the related security issues should have been already mitigated	Medium	Enhance protocols security using fuzzing techniques to test design and implementation.
17	DHCPv6++ (between nodes and ONIX) is not secure enough, so related security issues have to be investigated	High	Enhance protocols security using fuzzing techniques to test design and implementation.
18	A malicious Network Level (NL)_DE might push malicious policies to the ONIX	Medium	Network Level DE should identify itself before any communication with ONIX or any Node DE.
19	A malicious node might listen to the communications between the ONIX systems in different domains	Low	Encryption should be used, at least in critical messaging channels.
20	Eavesdropping on the communications between the DEs at each level	Low	See 19

	Threats	**Level (serious, medium, low)**	**Security dimension**
21	If a certain DE is compromised in the hierarchy, this information might be passed to the other DEs in the hierarchy.	High	The trust model should also cover DEs within a node.
	Self-configuration		
22	A malicious node might get information about the NL-DE, so it is easy afterwards to attack it	High	Authentication and access control are mandatory here
23	If ONIX is absent, a malicious node might also be able to get information about the NL-DE, so it is easy afterwards to attack it	High	See 22
24	A malicious node might also contact the NL_DE in order to fetch the configuration file and use it for attacks purposes	Medium to high	See 22

Table 1 Threat Description and Classification

3.3.4 Conclusion

As security is considered as one of the main characteristics of Future Internet and not an added value service, security in EFIPSANS has been accompanying the development of the Generic Autonomic Network Architecture (GANA) from the start.

In short, the GANA architecture has specified some Decision Elements (DEs) components that need to advertise their capabilities, discover each other, and configure themselves through a central data repository called “ONIX”. The mentioned mechanisms were explored, and the corresponding threats were identified. These threats were also classified based on how serious they are.

The GANA threat model described earlier has been addressing the following main questions,

- Which type of network/component/functionality needs to be protected?
- What kind of protection (detection and reaction) needed and against which threats?
- How to achieve the protection mechanisms in this new environment?

After assessing the GANA architecture from the security point of view, it was decided to focus on the security topics below and explore them further in EFIPSANS as they cover most of the issues discussed earlier. These topics are,

- Identity management and secure communication between the DEs
- Trust management
- Security monitoring
- Fuzzing to identify vulnerabilities in the IPv6 protocols that are going to be extended to fulfill the EFIPSANS requirements.

4 Trust Models & Trust Management

4.1 Introduction

Autonomicity is built upon the ability of entities within systems to make decisions and take actions based on information gathered by their experience and other entities, feeding various self-* conscious mechanisms. Thus, information gathering occupies a central part of an autonomic environment. The quality of the recovered information is the main issue we cover in the discussion about trust that it follows. Entities are taking decisions based on the information available to them. These decisions include their behavior towards other entities, services that may ask by nearby entities as well as self conscious regarding their capabilities and state.

Humans make decisions exactly the same way, based on information we draft from various sources. Part of this information we believe – think it is true and we accept it, while for some other part we seem to be cautious or even negative. We combine this information with our previous knowledge and experience that most of us handle as trustworthy and we conclude, making decisions and taking actions. But how do we classify which piece of information is true and which not? Trust and trust management is playing a key role in this procedure. We accept a close friend or a relative as trustworthy, while we are sceptical with a stranger or a friend's friend. But except from information gathering from different sources, trust plays a key role when we are actually taking actions or interact with something. We buy fresh food from a trustworthy store, we see a doctor we really trust and we buy a car of specific brand based on trust. For trust building we use our previous experience, our friends experience and organizations that guarantee the credibility or reliability of a person, thing or organization.

In our work, we deal with and analyze trust models and trust management within EFIPSANS as a key concept of everyday life within the autonomic planet we try to enrich in EFIPSANS. Autonomicity requires both exchange of information, decision making upon it, as well as actions and interactions with a colorful population of autonomic entities. EFIPSANS environment will actually be looking like this, as a globe of multifunctional, multi-domain, multi-node and multi-user end entities that interact with each other providing network services and information in various layers. That's exactly why we think that trust modeling and trust management will turn to be core functions of the future autonomic networks.

We structure this chapter as follows, we first provide a view to different trust models we analyzed for the EFIPSANS environment, we then provide our ideas on how trust management could be coupled and introduced into GANA, proceeding we evaluate three different trust models that could be used within EFIPSANS and in the last section we present our conclusions and future plans.

4.2 Trust Models

Networking is based on actions taken by different nodes within a network, e.g. a packet is expected to be routed towards a destination node by a router, certain quality characteristics is being expected to be met by a large number of nodes, spanning different domains and/or even technologies. Autonomicity on top, is giving the nodes the ability to decide based on policies and information gathered by nearby nodes. At the same time, autonomicity introduces entities within nodes that make decisions, interact with each other and even with entities in "nearby" nodes. Thus the introduction of trust models among entities (Decision Elements) and nodes (Nodes Decision Elements – NDEs and Network Elements) is a necessity towards an autonomic network environment.

In EFIPSANS we consider trust among entities to be dynamic as new nodes and new entities are being introduced constantly within network neighborhoods and (not so often) even in nodes.

On the other hand networks are being developed based in different operational environments. There are networks operating under a strict administrative domain, networks operating under loose policies but still under a solid administrative domain and finally ad-hoc created networks lacking a central administrative domain. That's why in EFIPSANS, autonomic entities must be capable to achieve trust in two different scenarios:

- based on a central authority – framework and
- dynamically issued, based on reputation and experience collected by the entities.

4.2.1 Trust in networks under strict administrative authority.

Most of the research fields in this area are already in production. We believe that through EFIPSANS framework a network operator can adopt some of the following remedies – techniques towards providing a secure trustworthy network service to users and peers. These techniques are based on the existence of a central authority. Central here refers to the administrative authority, as the systems for security and scalability reasons could follow a distributed architecture providing seamless services under malicious attacks, DDoS events or rapid expansion of the network.

In these networks a central authority – framework provides both authentication and certification. This way, entities can identify themselves and provide the necessary credentials and references to whom may ask about, regarding their capabilities and credibility. This is the case of a network under a strict administrative domain. Autonomic nodes follow specific strict policies and they trust each other based on the credentials, references and roles provided by a central authority. For such networks trust based on certificates and a Certification Authority together with a Public Key Infrastructure framework may provide the required functionalities.

The first step towards establishing trust in an environment like the one under discussion is the creation of a secure communication channel between nodes. The main attributes of such a secure channel are:

- Confidentiality, making the information transmitted through the channel accessible only to the members of the conversation (sender and recipient, or in the case of a multipart channel the authenticated members). Data transmitted should be unreadable for an eavesdropper or any non-authenticated user. Thus encryption of data channel is a common technique used to achieve confidentiality.
- Integrity, ensuring that the information transmitted from the source will not be changed in the way towards the recipient. A mix of message digestion techniques and cryptography is applied to achieve data integrity.
- Non-repudiation, reassuring that the source and the recipient are the one mentioned to the headers of a network packet, making impossible for them to deny neither that they sent the packet nor that they received it. Non-repudiation can be achieved using digital signatures and cryptographic techniques.

A secure channel for communication between any two nodes within an autonomic network is a base service to provide security throughout the network. Through these secure channels entities communicate in order to access services that reassure the identity of each node, the rights to access or provide certain resources and so on. These services include:

- Authentication, providing a “name” or “id” to each network entity within the network. This way any entity could ask for any communicating entity to authenticate itself. This is provided by a central authentication authority.
- Authorization, giving the rights to entities to access or not certain functionalities or areas of the network. Authorization determines what a previously authenticated user is able to access and which type of access (e.g. only read, execute) he/she is allowed to do. An Authorization framework, typically as a part of an AAA (Authentication, Authorization and Accounting) service, can provide the relevant functionalities within EFIPSANS.

Even though a strict administrative authority is applicable in the case of a traditional network, autonomic networking may turn this into an impossible task. Nodes may be authenticated and authorized, coupled with specific keys but this can not happen in full extend for the entities inside nodes. New versions of software or/and firmware of routers, patches on machines and workstations as well as software installed by end user is almost impossible to be controlled, authenticated and authorized centrally. In this case a more loose management paradigm must be followed and EFIPSANS should be capable to provide such flexibility to the administrators and the network itself. We will present our ideas and view on this topic in a following section.

4.2.2 Trust in networks lacking an administrative authority.

Today’s networking and future networks and internet include significant parts of ad-hoc networking environments. In such environment, a central administrative authority is not available. These networks may play a central role in the future networking as they can provide coverage and quality of service in areas where providers’ infrastructure may prove to be inadequate. Networking provided by linking together user nodes can cover such areas making possible to end user to receive service where he would not otherwise.

In such networks trust models are based on previous experience collected by a node and input provided by its neighbours. We follow this view in EFIPSANS and examine some reputation based trust models that may be used towards establishing trust. As ad-hoc networking resembles to P2P overlay networks, in our work we examined reputation based trust models specifically designed for P2P applications. In this context we studied three reputation based models and examined their usage and behaviour. These models were, Eigentrust [KSM03], ROCQ [GAR04] and Bayesian [WV03] based model.

Eigentrust was introduced by S.D. Kamvar and others. It is based on the calculation of global trust values t_i based on the local opinion of node j has for node i, c_{ji}, based on previous experience of the j. The local trust values computed in a first step are normalized and aggregated to produce the global trust values. A distributed version and a secure version of the algorithm were proposed by the authors. The advantages of Eigentrust include easy adaptation and expansion of the algorithm, its clean mathematical structure and that it produces global trust values for all the nodes of the network. Its disadvantages include the distortion inserted by the normalization face, non-existence of negative trust values and the fact that the calculation of a global trust value may not be the best technique in hostile environments.

Reputation, Opinion Credibility and Quality (ROCQ) scheme was introduced by Anurag Garg and Roberto Battiti. It combines Reputation as a global value, Opinion reflecting a node's self-knowledge based on first-hand experience – transactions, Credibility of a reporting node and Quality of the provided values by a peer. The calculation of Reputation is taking place in a distributed way. In each step the Opinion matrix, the standard deviation of the Opinion matrix, the Quality matrix, the Credibility matrix and finally the Reputation matrix are being calculated. The advantages of ROCQ include the ability to produce global trust values for all the nodes of the network, while at the same time it takes under consideration the opinion of a local node and the credibility of the nodes and the ability to produce valid trust values even in hostile environments. Its disadvantages include the more complicated scheme than the one proposed by Eigentrust, the non existence of negative trust values.

A Bayesian trust model is based on the local trust values of a node. Each node keeps track of his previous transactions and the trust emanating by them as the ratio of the number of satisfactory transactions to the total number of transactions with each node. In any transaction node i uses the local trust value for node j unless it didn't have any transaction so far. In this case node i asks for the trust values of some of the nodes already in his local trust table and uses the Bayes law to produce the final trust value. The advantage of this model is the simplicity of the implementation and in specific cases (i.e. hostile environments) the usages of local trust values. As a disadvantage we can mention the difficulty of the model to operate and produce useful trust values in large networks of high mobility where the possibility for two nodes to meet – interact more than two times is minimal.

4.2.3 Trust in networks with a loose administrative authority.

As it was mentioned before, autonomic environments are expected to operate somewhere in between the previously described environments. A strict administrative domain may exist for some of the nodes of the network or part of Decision Elements within the nodes of the network. At the same time some or the majority of the nodes may operate with no administrative authority. For this reason we suggest for EFIPSANS framework the adoptation of such a scheme that will utilize the security infrastructure (i.e. authenticated and authorization authorities, PKI etc.) where available but will also operate a reputation based scheme for nodes or DEs that could not be authenticated or authorized.

Today's networks operate under distinct autonomous administrative authorities, the paradigm of Autonomous Systems. A same view we expect to be in operation in the Future Internet, as networks under different administrative domains will be called to cooperate and provide quality services. We also expect autonomicity to play a role in the way services will be provided among different domains and networks. In this case each domain may retain a global trust model but these values are expected to be local values for transactions among nodes of different domains.

In EFIPSANS we suggest a differentiated approach for trust modelling within an autonomic network. A global trust model, like Eigentrust or ROCQ, can be followed within a network domain. These models can re-enforce any centralized security mechanisms, like authentication and authorization infrastructure, installed and operating by the network management team. This way the today's security management paradigm is enriched and extended to cover autonomic networking. At the same time and for trust development between network domains a local trust model like Bayesian seems to be more appropriate. Our approach is this way covering all the needs for trust development and modelling covering inter and intra domain service provisioning within an autonomic network environment.

4.3 GANA & Trust Management

After presenting the trust models that can be used within EFIPSANS, analyzing different administrative environments we look into trust management approaches and we present trust functionalities that from our point of view GANA should be able to provide.

Different trust management techniques appear in literature. They can be categorized in the following three general containers:

1. Individual trust management, reassembling a real human society
2. Global trust management, where all the nodes cooperate to reproduce global trust values.

3. Federated trust management, where trust management is taking place in an isolated manner between groups of nodes, domains, where different strategies are being used to build trust among domains.

In individual trust management, each node builds its own trust values for the nodes participating in the network. The trust values t_i for a node i may be different in each node of the network, just like it happens in a human society. Algorithms based on this principle include Secure, Bayesian [WV03] models, Strudel [QLHCB06], H-Trust [ZL08], ROCQ [GAR04] and Mate [QH08].

In global trust management, the nodes cooperate to produce global trust values for all the nodes. This way within the network all the nodes share the same trust value t_i for a node i. Algorithms based on this principle include Eigentrust [KSM03] and models developed by Baras et al. [BJ05] and Donato et al [DPSCCL07].

In federated trust management, trust in multiple and heterogeneous security domains and autonomous systems is examined. Each domain retains its own trust management scheme while strategies for developing trust among domains are being built separately. Algorithms based on this principle include one developed by Chun and Bavier, regarding Decentralized Trust Management and Accountability in Federated Systems, and one proposed by Bhargav-Spantzel, Squicciarini and Bertino on intergrading federated digital identity management and trust negotiation [SSB07].

Future networks are going to face complex environments where global trust management could be used to provide trust values within a network with loose administrative authority, while federated trust management could be used to provide trust management among different administrative domains. This way GANA must be able to provide to the administrator and the end users the necessary tools to implement some of the above mentioned trust models, or even future ones. For this reason we propose the usage of IPv6 headers to disseminate trust and reputation values within a network and among different network domains. GANA in this sense should be able to provide trust metrics for:

- DEs within a node[2].
- DEs among nodes within a network domain.
- Nodes within a network domain.
- Nodes among different network domains
- Network domains.

Following the previous analysis we expect federated trust management models to be the most appropriate for use by GANA and EFIPSANS. We propose for this reason a usage of a mix of global and individual trust management models. Within each domain trust may be calculated globally, while for transactions crossing the boarders of two or more domains

[2] Information exchange and service requests may occur among different DEs within a node. Even if DEs at a certain point have been certified regarding operation and trustworthiness, something like thise may sooner or later be invalid. Nodes are operating in a hostile environment, they are being updated and DEs are changing their behaviour according to input – information. An environment like this makes trustworthiness and security an ad hoc attribute that has to be built upon trust management techniques even within a node.

local values and models could be used. In this approach we handle each domain as a single individual utilizing the network level DE for trust.

4.4 Simulations & Results

For evaluating the trust models described above, we based our analysis on simulation results. For the simulation we modified and used the P2P simulator developed by Andrew G. West and al. at university of Pennsylvania [WKLS09]. The simulator is developed as an open source project for evaluating Eigentrust algorithm. P2P simulator is considered as extremely efficient in evaluating reputation based systems as the models under discussion within chapter. For our purposes we enhanced and further extended P2P simulator by developing the code necessary for evaluating and comparing Eigentrust, ROCQ and Bayesian trust algorithms. Simulations are based on trace files, files with sequences of nodes and service requests made by them. In any moment service is being provided by a node capable for this service based on the trust values and the trust model we use.

For the purpose of our simulations we consider a set of nodes as a part of a network. Some of these nodes share common links making communication among peers available. In every moment each node is capable of requesting or providing a service, making it either a receiver or a provider, the same way as in a P2P network, where a peer requests or provides a file or some infrastructure in general. We consider two different modes for each transactions, either a node will provide trusty services fulfilling receiver expectations or the node will lie regarding the provided services, failing the relevant expectations by him. By the end of each transaction the receiver provides some feedback regarding the quality of the service experienced by the provider. Each node can provide different type of services, as in a p2p network each node can provide different files. The node is capable to remove or stop providing one or more of the services if it recognizes that a service is compromised and is not fulfilling expectations. We call this procedure clean-up in correspondence with the same mechanism that is being used in p2p networks.

For the needs of our simulations we classify a node as:

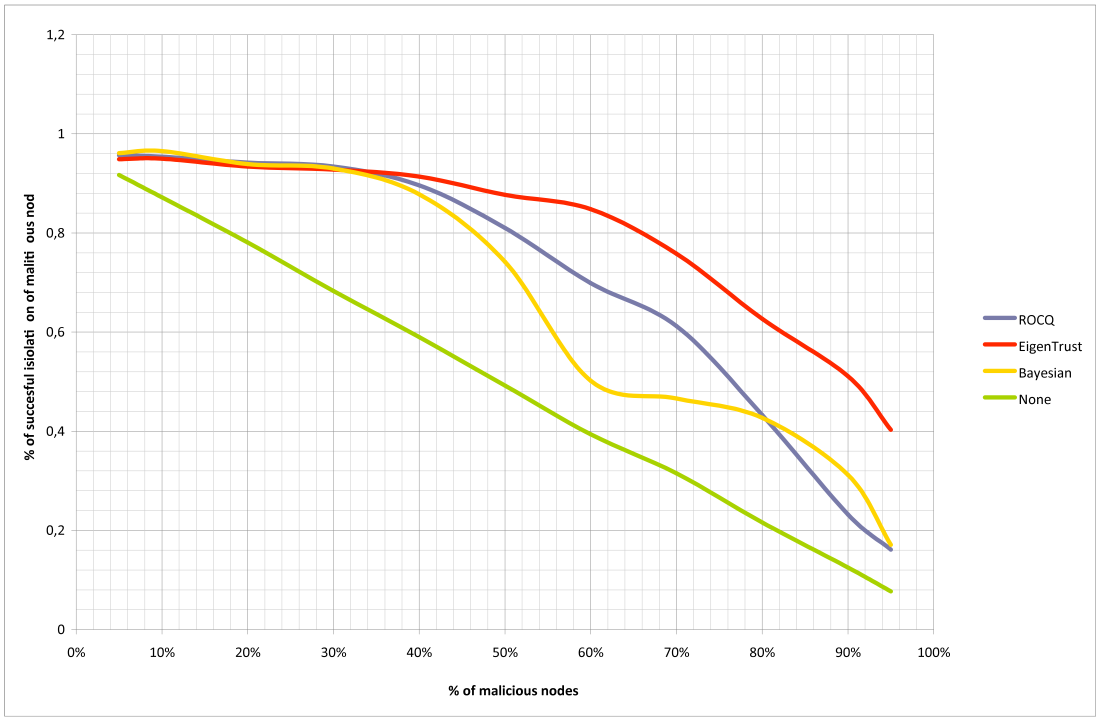
- Good, when it cleans-up 90 to 100 % of the compromised services and provides 100% trustworthy feedback.
- Purely Malicious, when it cleans-up 0% to 10 % of the compromised services and provides 0% trustworthy feedback.
- Malicious Provider, when it cleans-up 0% to 10 % of the compromised services and provides 100% trustworthy feedback.
- Feedback Malicious, when it cleans-up 90% to 100 % of the compromised services and provides 0% trustworthy feedback.
- Disguised Malicious, when it cleans-up 50% to 100 % of the compromised services and provides 50% to 100% trustworthy feedback.
- Sybil Attacker, when it cleans-up 0% to 10 % of the compromised services. A Sybil Attacker leaves the network after providing a bad or malicious service and tries to white-wash his identity entering back as a newcomer.

We note, even if we do haven't examined such scenarios in our work so far, that Malicious nodes can cooperate creating a botnet that we refer to as Malicious Collective. For our simulations we used Purely Malicious nodes in a fully connected network. As we calculate trust values for service provisioning we simulate the network as a fully connected one, as we assume that a node may requests a multi-hop connection towards a service provider if there is no direct connection with him.

The calculated metric for our simulations is the % of successive isolation of malicious nodes that is calculated as the ratio between the number of successive services good nodes consume to the total number of services consumed within the network. To understand this someone has to keep in mind that the more successive services good nodes consume, the more isolated and less service providing are the malicious node.

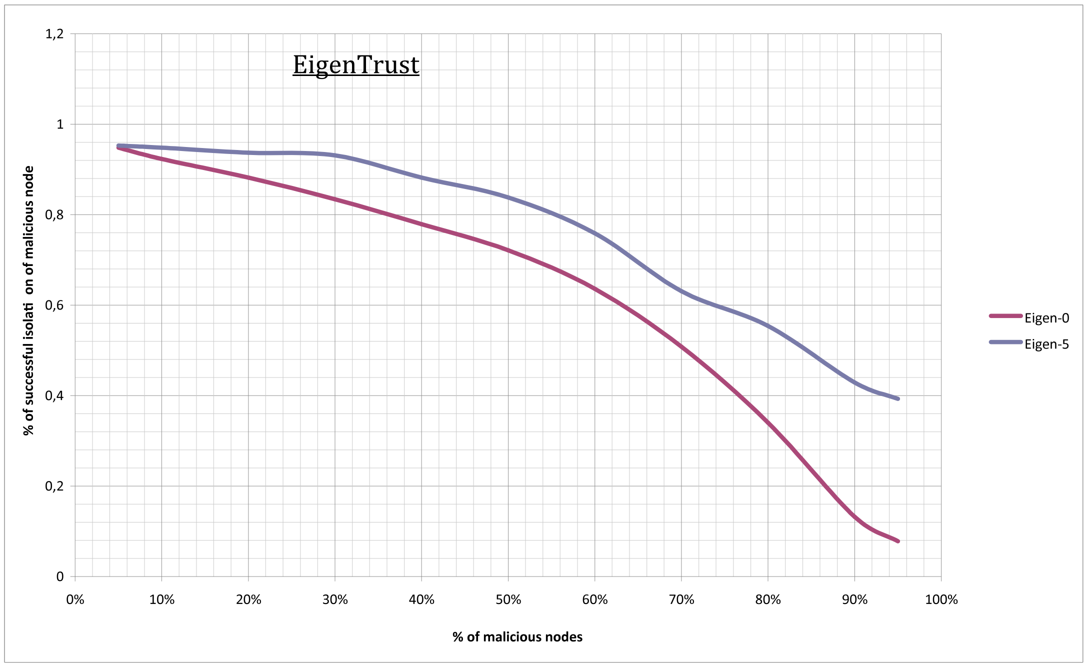
We tried six different scenarios to compare and evaluate the selected trust models.

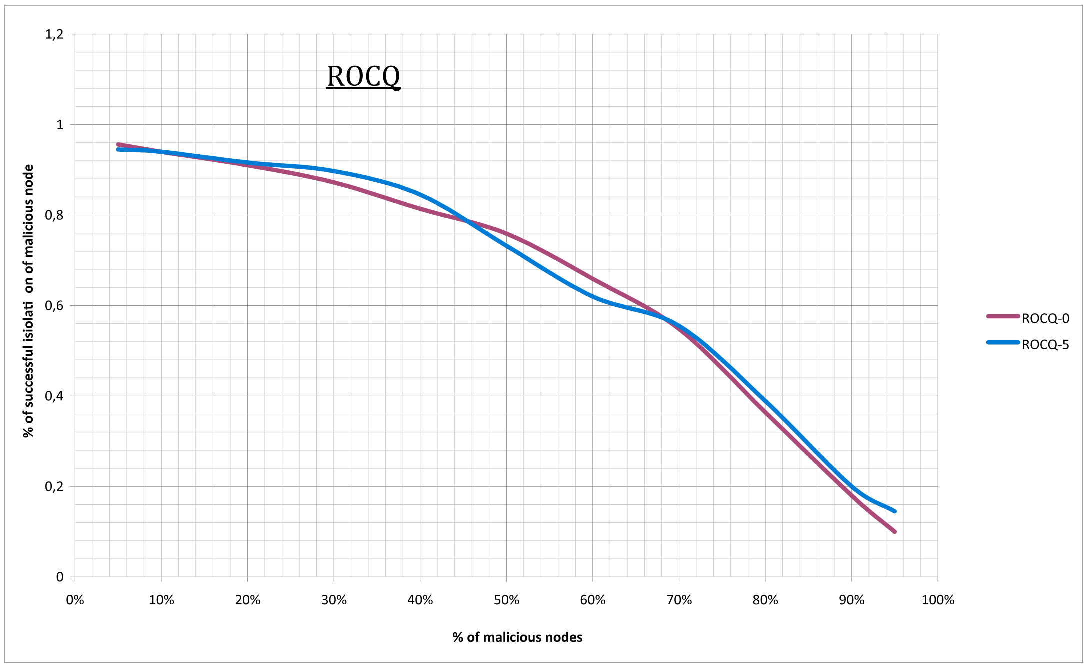
Scenario A: In this scenario we used a network of 200 nodes participating in 10.000 transactions – service requests. We increased the numbers of malicious nodes from 5% to 95% by an increment of 10%. For comparability we used the same trace among trust models each time. The results, compared to a network not using a trust model, appear in the following figure.



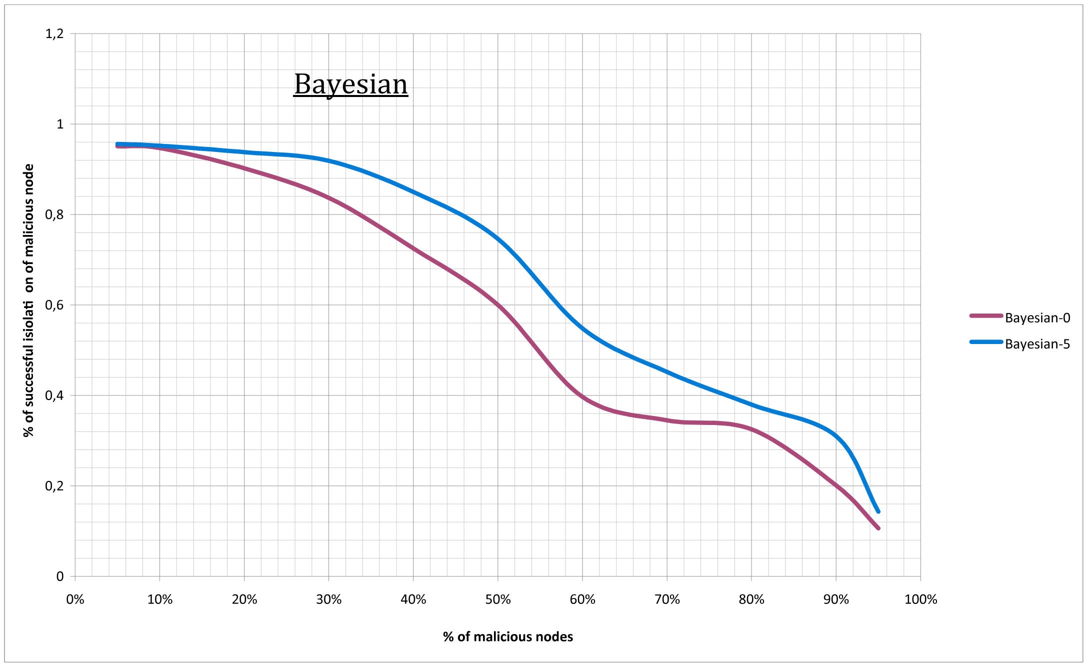
The usage of a trust model helps the network to provide better services to the end users. When we do not use a trust model the quality of the provided services are declining linearly. Among the three models, Eigentrust provides better results and quality of service toward good nodes.

Scenario B: In this scenario we introduced the idea of a pre-trusted node. A pre-trusted node is a node within the network that is well known and other nodes can use as reference. This can be the case of a network domain with loose administration where there are some nodes pre-trusted through a Third Trusted Party, but the majority of the nodes are not de facto trusted. We used a network of 200 nodes exactly as before, with 0 or 5 pre-trusted notes, where 5.000[3] transactions among nodes are taking place increasing the number of malicious nodes from 5% to 95% by an increment of 10% and repeating our simulations 10 times. The results of our simulations are depicted in the following figures.



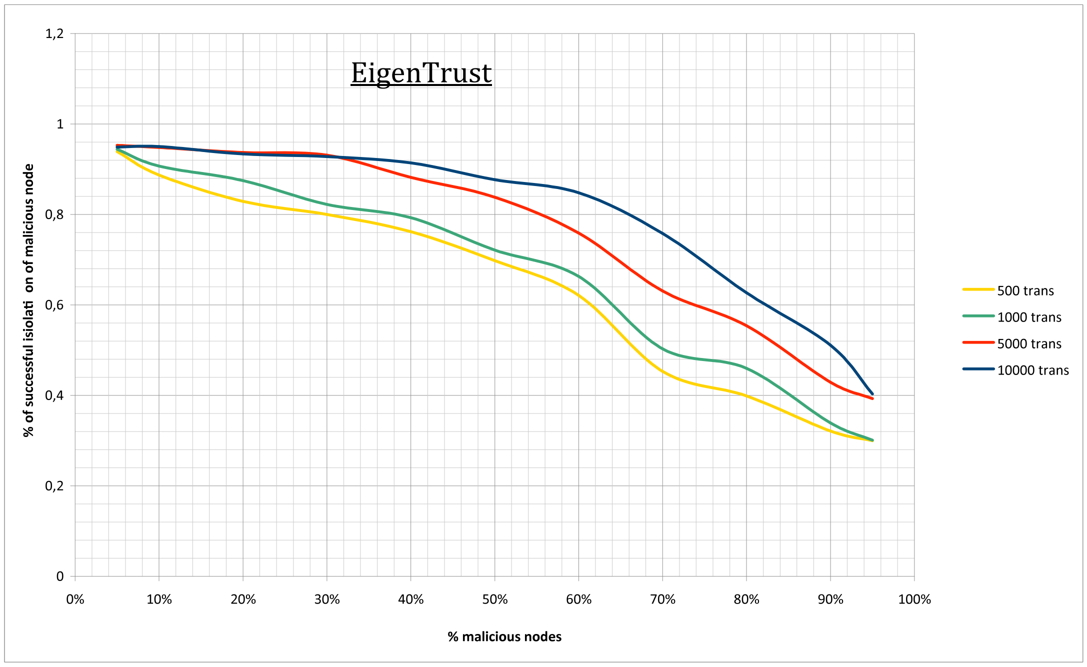


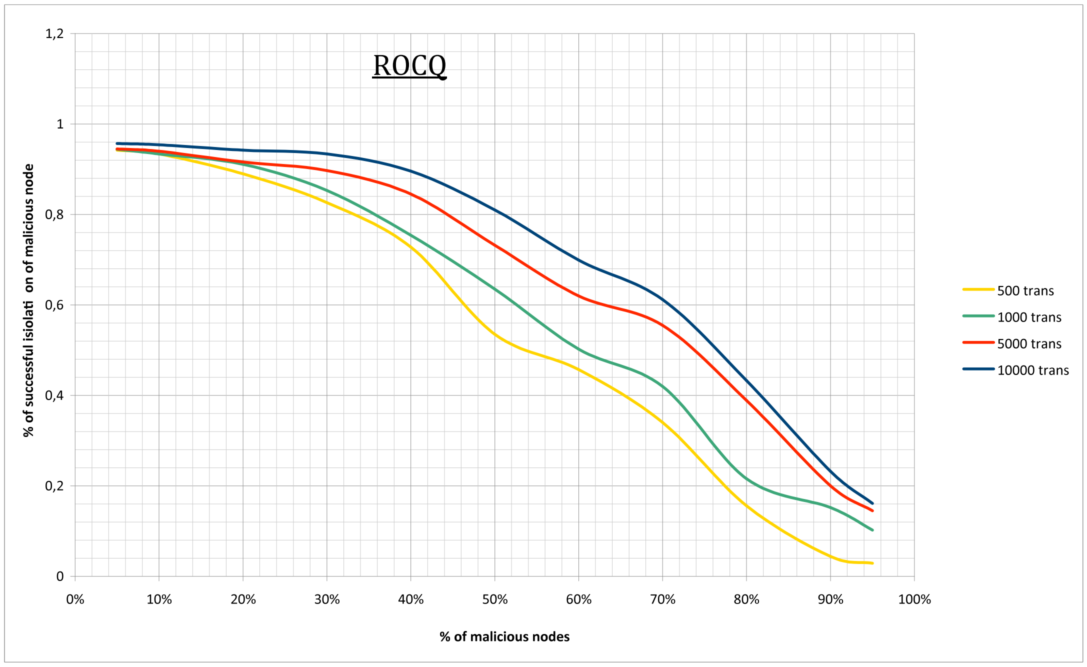
[3] We use 5.000 transactions from this point on, for faster simulations, as there is no difference in convergence compared with 10.000 transactions.

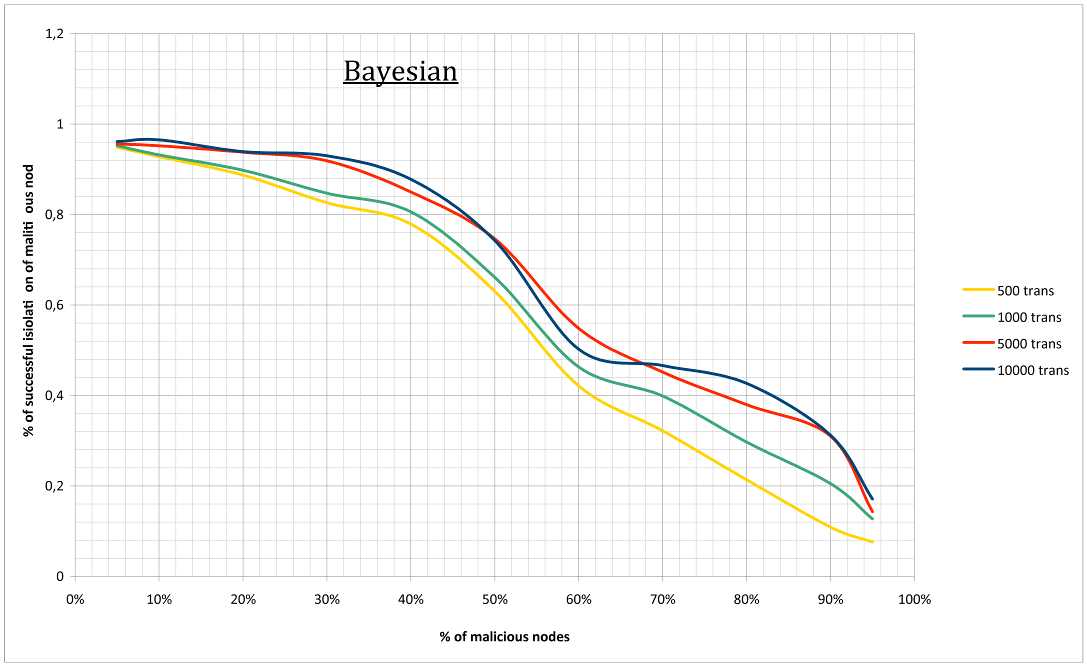


We observe that ROCQ's results are irrelevant to the number of pre-trusted nodes, while both Eigentrust and Bayesian are operating better when a small set (2.5%) of pre-trusted nodes are in place.

Scenario C: In this scenario we examine how trust models operate with respect to the number of transactions taking place within the network. We use a network of 200 nodes with 5 pre-trusted where 500, 1.000, 5.000 and 10.000 transactions are taking place. Exactly as we have done in the previous simulations, we increased the number of malicious nodes from 5% to 95% by an increment of 10% repeating our simulations 10 times. The results of our simulations are shown into the next figures.

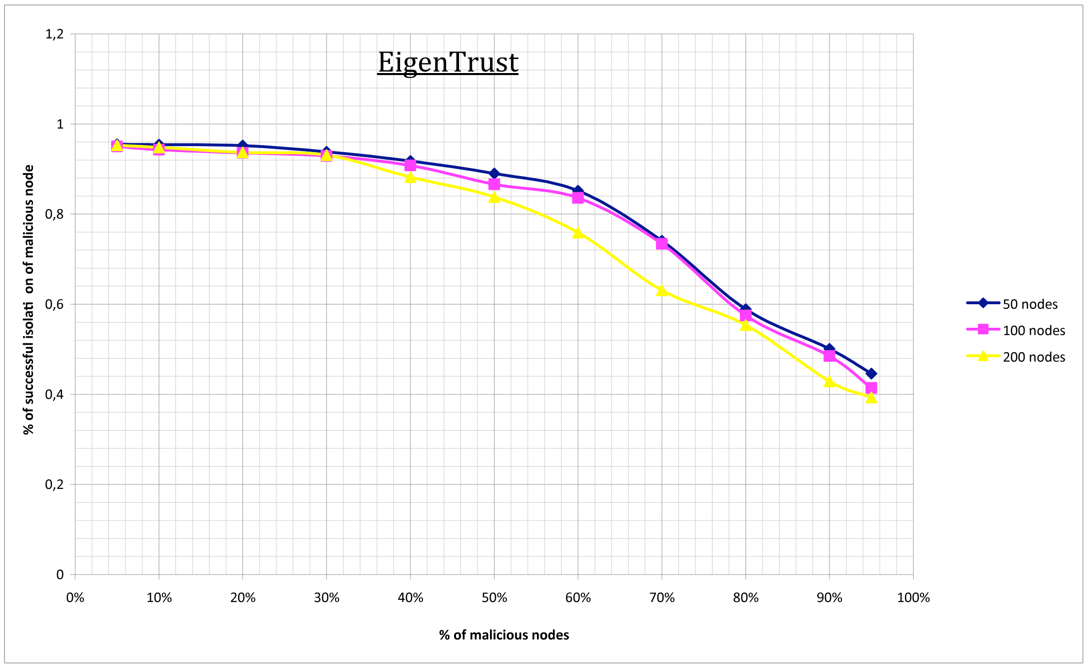


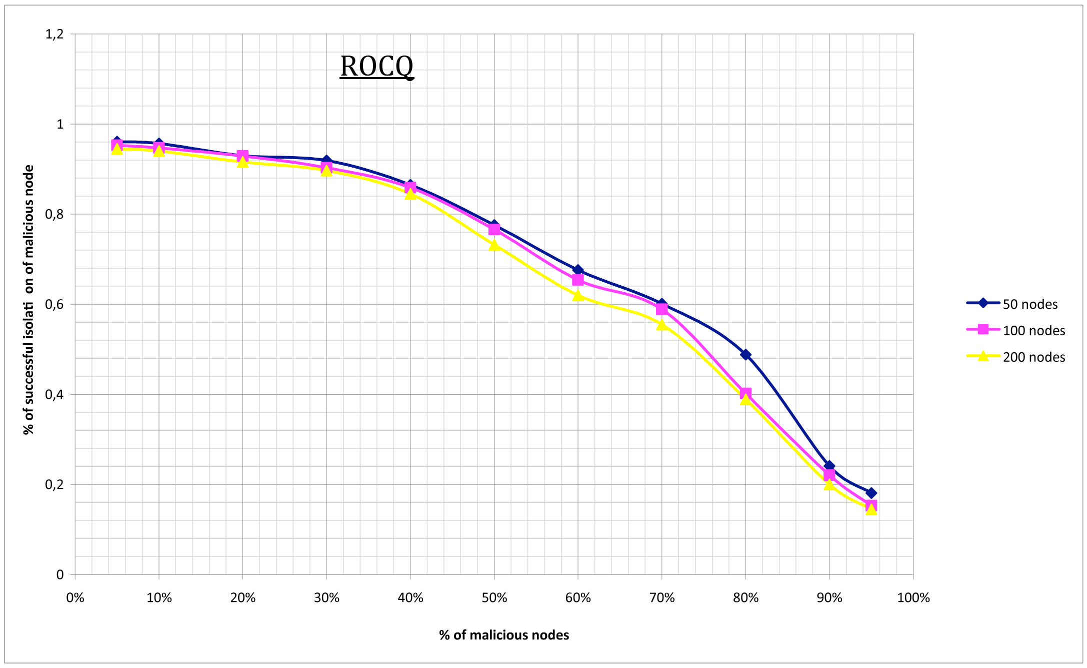


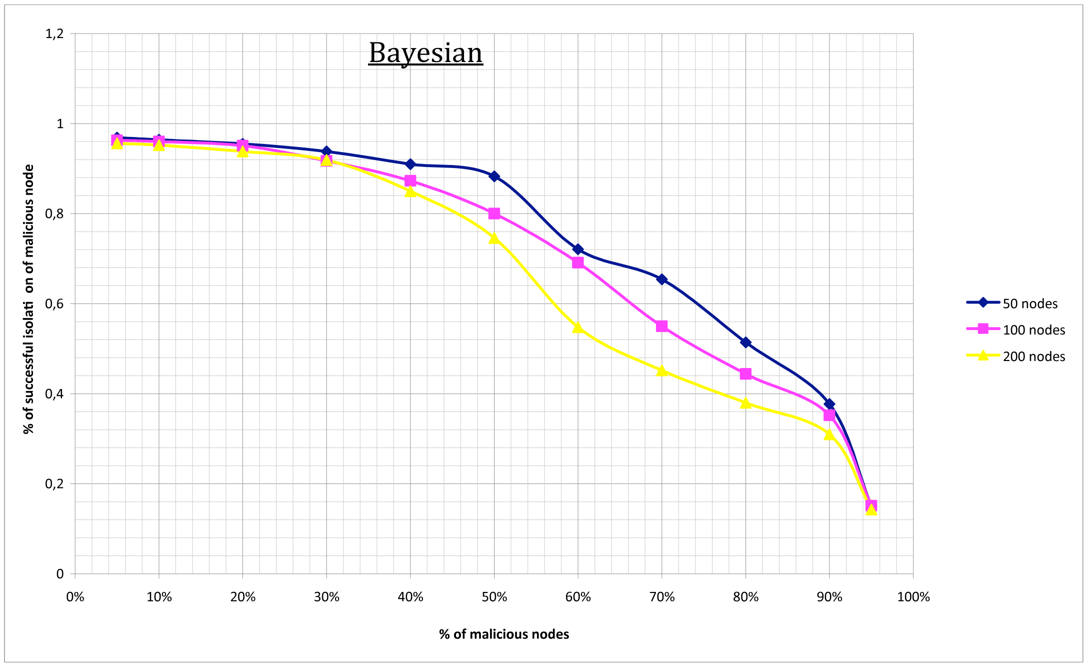


We notice that the number of transactions can help improving the effectiveness of the trust models, in the case of Eigentrust and ROCQ. While the Bayesian model seems to be less sensitive to this parameter. As a result we expect trust models to provide more accurate predictions and success rate in the case of networks with intensive service consumption and in the case of slow changing with time networks. This could be taken as a design parameter in future networks based on EFIPSANS network while enabling trust mechanisms.

Scenario D: In this scenario we examined trust models success in accordance with the number of nodes participating within the network. We simulated networks with 50, 100 and 200 nodes where 5.000 transactions are taking place. Exactly as we have done in the previous simulations, we increased the number of malicious nodes from 5% to 95% by an increment of 10% repeating our simulations 10 times. The results are presented into the next figures.

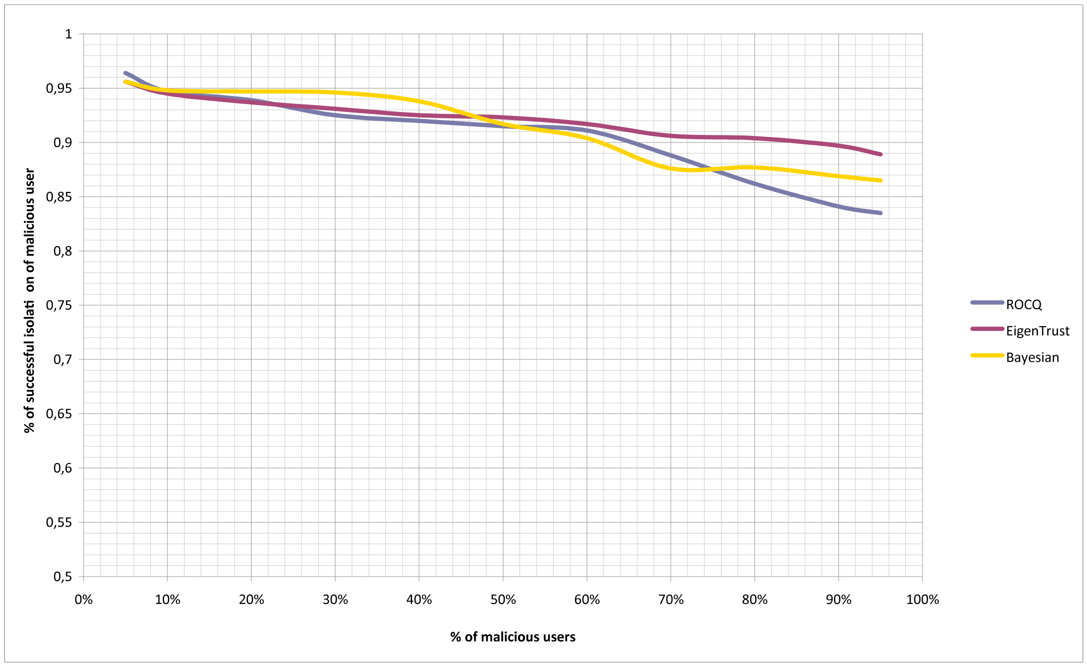






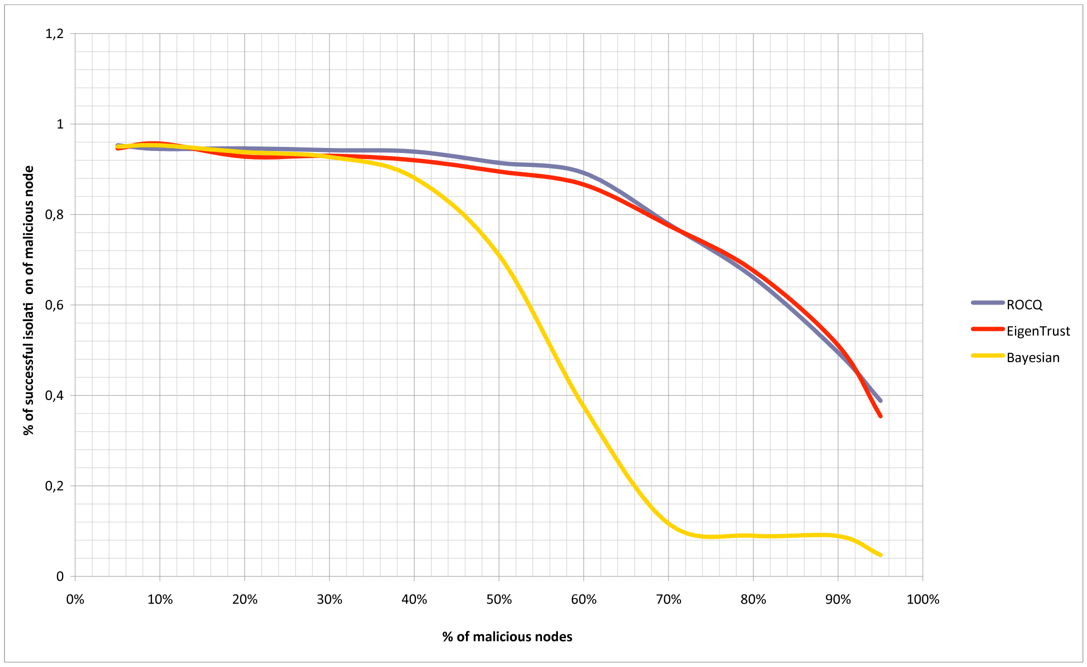
Decreasing nodes and keeping constant the number of transactions results into more information per node that is equivalent to keeping constant the number of nodes and increasing the number of transactions. So this scenario is equivalent to scenario C. So as expected we notice a reverse behavior compared with scenario C as the effectiveness of our models are declining as the number of nodes are increasing, provided that the number of transactions remain the same. The results for ROCQ seem though, to be less sensitive to the increasing number of nodes, compared with Eigentrust and Bayesian especially in the range of 70% malicious nodes.

Scenario E, in this scenario we tested the effectiveness of the models in the case of Disguised Malicious nodes. We used a network of 100 nodes where 5.000 transactions are taking place. The number of Disguised Malicious nodes was increased from 5% to 95% in increments of 10% as before with each simulation tested 10 times. The results we got are extremely positive and are depicted in the following figures.



All three models appear to be are extremely effective in this case, making comparison between them hard. This is explained partly because the Disguised Malicious nodes are far less aggressive compared with Malicious nodes used in previous scenarios. This way 70% Disguised Malicious nodes is more or less equivalent to 35% Malicious nodes used in previous scenarios.

Scenario F: In this last scenario, we tested the effectiveness of the three models in the case of Sybil Malicious nodes. We used a network of 100 nodes where 5.000 transactions are taking place. Sybil Malicious nodes were increased from 5% to 95% in increments of 10% as before, with each simulation tested 10 times. The results of our simulations are presented in the following figure



We notice that both ROCQ and Eigentrust are equivalent effective while Bayesian model fails at a threshold of Sybil Malicious nodes of around 55%. This is explained by the fact that as Sybil Malicious nodes increase there are not enough transactions taking place between good nodes. Thus local trust values of Bayesian model are of small importance resulting into model failure. For this reason, EFIPSANS could provide models based on Eigentrust and ROCQ for ensuring protection against these types of attacks.

5 Security Functions

5.1 Introduction

In previous chapters, we analyzed the GANA architecture by defining possible threats, and we compared different trust models in relation to GANA. We introduce in this chapter the Security Decision Elements with the objective to establish a security level into GANA. So these Security DEs handle inter alia access control, and self-defending mechanisms. Both are very important issues for an autonomic network. This chapter takes in charge the detailed description of both problems, namely; access control for the GANA architecture as well as self-defending mechanisms. We will also consider an adjacent problem of the robustness of network protocol implementations. An erroneous network protocol can cause many troubles and lead to communication inconveniences. Vulnerabilities in network protocols can be detected by fuzzing, a special kind of testing.

The structure of this chapter looks as follows: first the security DEs are introduced, second we address access control and secure communication between DEs. Afterwards, we explain our approach for self-defending, and finally vulnerability detection using fuzzing is clarified.

5.2 Security Management DEs

The self-management aspect in the GANA architecture is reflected by the control loops and in particular the Decision Elements (DEs) related to the different crucial functionalities identified and described through the different EFIPSANS deliverables. Security is another crucial functionality that has to be addressed by the GANA architecture in a harmonized way. Indeed, the introduction of a Decision Element (SEC_MNGT_DE) within the GANA architecture dealing with security fulfils the following requirements,

- The GANA architecture must be first secured at the node level.
- The GANA architecture must be secured at the network level.
- Like the GANA architecture itself, the corresponding security framework has to be adaptive.
- The GANA security framework must address the most known security issues (authentication, authorization, integrity,…).
- The GANA security framework should be flexible enough to be able to consider different security solutions (for instance using different authentication schemes,…).

Protection at node/network level

To protect the GANA architecture, we have introduced two types of Decision Elements (DEs) as depicted in Figure 2.

The one at the node level (NODE_LEVEL_SEC_MNGT_DE) will be in charge of the local security or the security on the node, however, the one residing in the network (NET_LEVEL_SEC_MNGT_DE) will be responsible for the global protection or the security of the entire network in addition to the coordination of the different security activities at the nodes level.

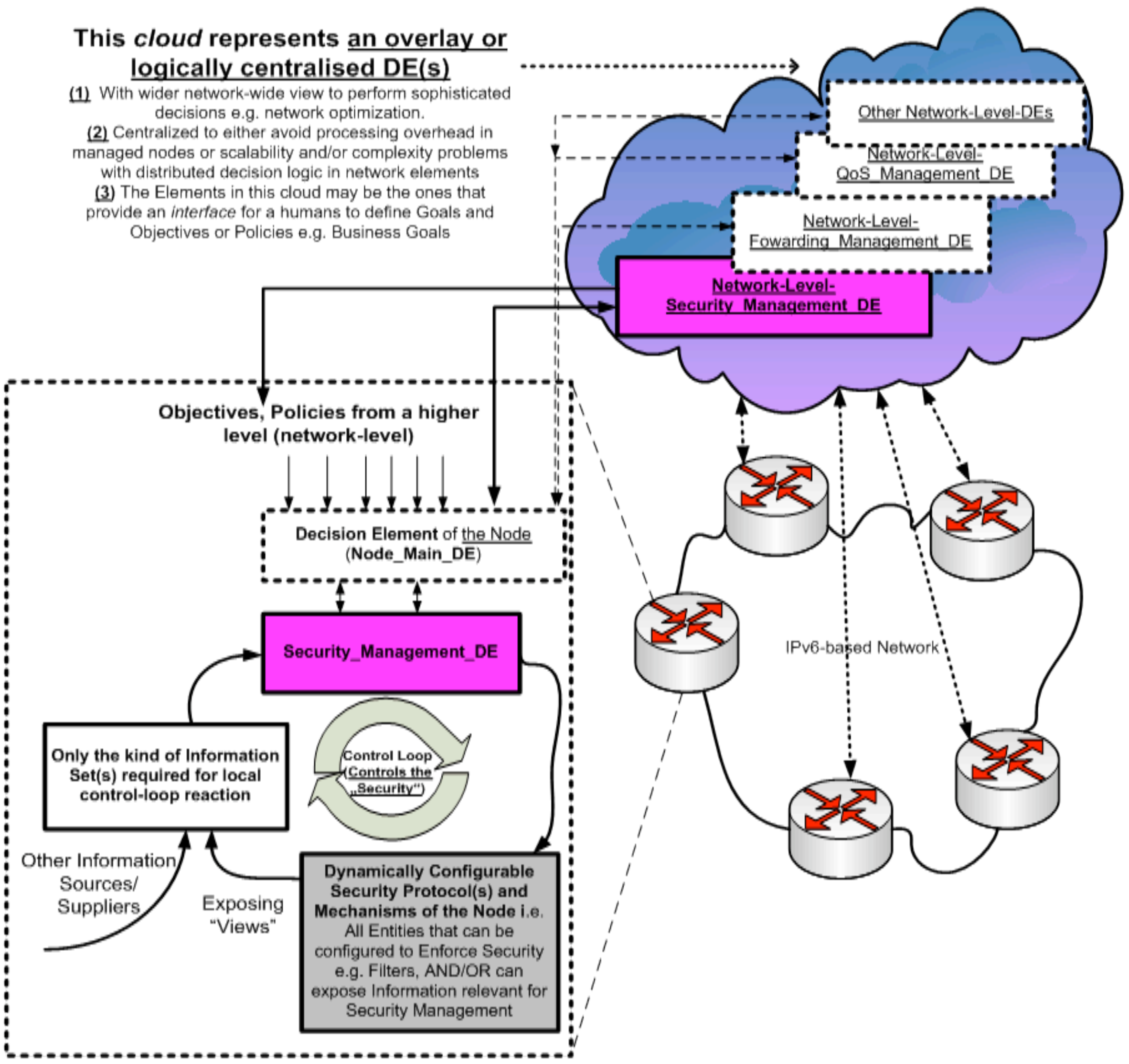

This cloud represents an overlay or logically centralised DE(s)
(1) With wider network-wide view to perform sophisticated decisions e.g. network optimization.
(2) Centralized to either avoid processing overhead in managed nodes or scalability and/or complexity problems with distributed decision logic in network elements
(3) The Elements in this cloud may be the ones that provide an interface for a humans to define Goals and Objectives or Policies e.g. Business Goals
Other Network-Level-DEs
Network-Level-QoS_Management_DE
Network-Level-Fowarding_Management_DE
Network-Level-Security_Management_DE
Objectives, Policies from a higher level (network-level)
Decision Element of the Node (Node_Main_DE)
Security_Management_DE
Only the kind of Information Set(s) required for local control-loop reaction
Control Loop (Controls the „Security")
Other Information Sources/ Suppliers
Exposing "Views"
Dynamically Configurable Security Protocol(s) and Mechanisms of the Node i.e. All Entities that can be configured to Enforce Security e.g. Filters, AND/OR can expose Information relevant for Security Management
IPv6-based Network

Figure 2 Security management DEs

Adaptive security

Introducing a DE dedicated to handle security for the GANA architecture reflects the autonomicity aspect as this is a part of a control loop that collects security information, analyses it and takes decision accordingly.

Addressing known security issues and various security solutions

The preliminary design of the SEC_MNGT_DE (see Figure 3) is intended to be generic enough to steer different security dimensions as well as various solutions or techniques for a given security dimension. For instance, we are considering security dimensions such authentication, access control, confidentiality, integrity and self-defending. These security dimensions will be specified and implemented based on the assigned role and the investigated scenarios. For instance, the authentication and access control dimensions will be specified and implemented if the SEC_MNGT_DE is the one at the network level. The security self-defending functionality will be specified and implemented on both Security DEs (on the node and in the network). However, the way this dimension will be specified and implemented at the node level will be different from the implementation at the network level, as the role of the security DE at the network level is more the coordination of the per node security activities.

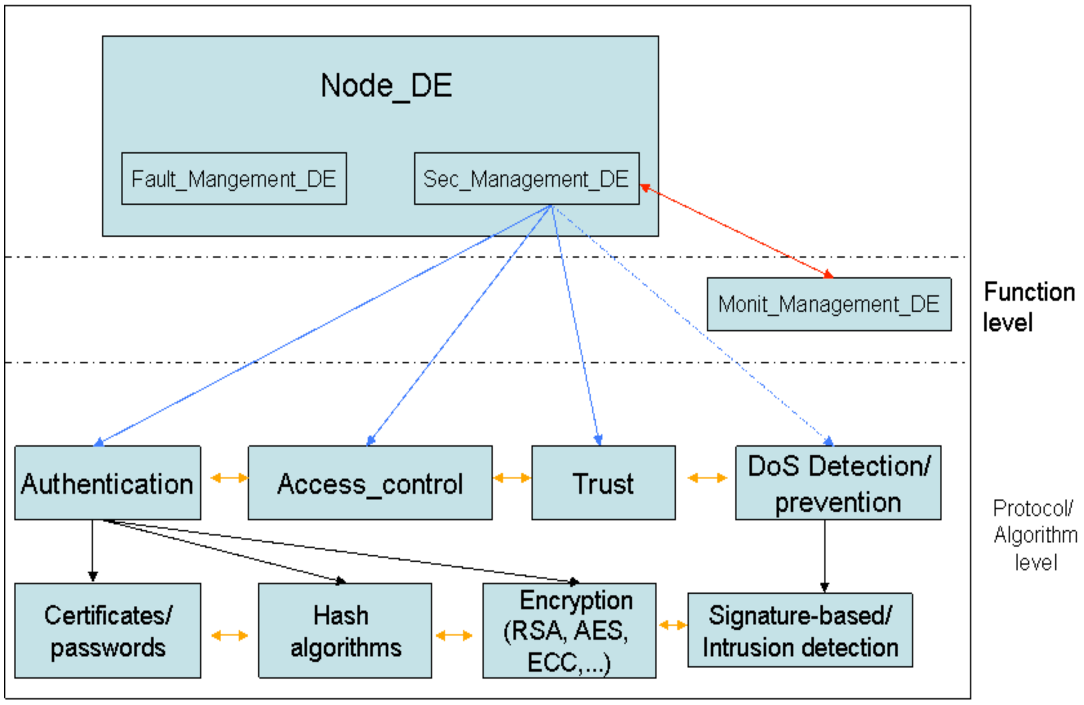


Figure 3: Security Management Decision Element : tentative design

5.3 Access control and secure communication between the DEs

The autonomic character in the GANA architecture is tightly linked to the storage framework (called ONIX and described in section 3.2). ONIX can be seen as central directory allowing in particular,

- An administrator to upload a configuration file describing for instance the topology of the network, the roles of the nodes as well as the related policies
- A node in the network to download the configuration file part related to it and configure itself accordingly
- the nodes in the network to publish their capabilities in order to make themselves visible to the rest of the network

- pushing (from the ONIX) certain type of information to some specific nodes that have subscribed with ONIX to be notified regarding this information

Based on the above description of ONIX, we can notice that one of the main threats to the GANA architecture is the management of the access to the resources (files stored on ONIX). Access control is a challenging problem in this case because of the following:

- The topology of the network depends heavily on the configuration file uploaded by the administrator. This means that we have to ensure that no entity (other than the administrator) can upload a configuration file on the ONIX system
- A node (in the network) with a certain role needs to fetch the appropriate part from the configuration file in order to configure itself accordingly. If a malicious node (or DE) got this opportunity, this would have a serious impact on the network behavior or topology.
- The roles assigned may change over time, so it is mandatory that a node whose role has changed, is not allowed to access data that its previous role allowed it to access it
- If some of the uploaded data on ONIX is encrypted, distributing the appropriate keys to other nodes in order to allow them to decrypt the data and use it, needs to be investigated

Based on the above basic requirements, the access control solution for the GANA architecture will focus mainly on some of the following topics,

- Definition and specification of the security profile that the administrator uploads on ONIX before the network boots up
- Design and specification of the role of the Sec_Management_DE in authenticating and authorizing the other nodes to access ONIX in order to upload or retrieve data to/from it. Here, key management and distribution will play a crucial role. We will investigate in particular,
    - whether the different nodes would need to carry some security keys and use them for authenticating themselves, accessing resources on ONIX and encrypting data flowing between the different DEs
    - or the DEs will get some sort of tokens generated by the Sec_Management_DE and which be used for accessing ONIX
- Design and specification of the interconnection between the Sec_Management_DE and the security DEs on the nodes
- Design and specification of an appropriate key revocation mechanism in order to prevent the nodes or the DEs to access previous information once their roles have changed
- Design and specification of a key management platform for key distribution if some of the data is going to be stored encrypted on ONIX

5.4 Self-Defending Mechanisms

In currently used networks there is a lack of self-protection and autonomous defending mechanisms. This situation leads to the spread of self-propagating malware which cause even more dangerous, significant threats i.e. Botnets [Cisco1]. Botnet is a group of infected computers (often called Zombies) that are controlled either by one person or an organization. Controlled machines can done any malicious work ordered by controller, called botmaster. SPAM sending, hosting phishing sites and performing DDoS attack are examples of malicious activities.

In the future self defending networks should detect such activities and protect other Internet users. As it was described in section 5.2, the proposed node security-management DE (NODE_LEVEL_SEC_MNGT_DE) introduce functionality, so important nowadays. This chapter describes NODE_LEVEL_SEC_MNGT_DE abilities to detect such threat and defend mechanism which with cooperation with other DEs, can be achieved in future networks.

5.4.1 Self-defending functionality description

The proposed self-defending functionality can be divided into two phases. In the first phase malicious or suspected activity is detected. In the second phase, accordingly to the policy, the appropriate reaction is performed. In figure 4 other DEs that take part in investigated functionality are presented.

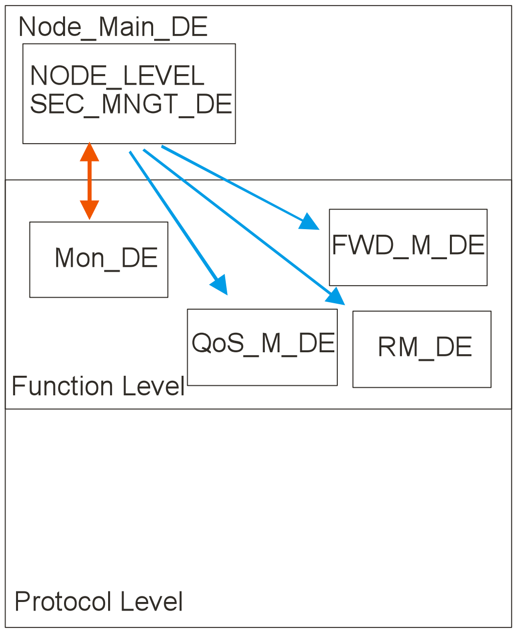


Figure 4: Security Management Decision Element and other DEs taking part in self-defending functionality

In the first phase suspicious activity must be detected. In this case monitoring services provided by other GANA DEs are used, especially those provided by Monitoring DE (MON_DE).For all interesting data NODE_LEVEL_SEC_MNGT_DE communicates with MON_DE. This communication is bidirectional: NODE_LEVEL_SEC_MNGT_DE sends requests for needed data and MON_DE provides acquired data. To ensure high performance some aggregations are performed by MON_DE. Acquired data is analyzed by NODE_LEVEL_SEC_MNGT_DE using data mining techniques. The class of attacks that can be detected is described in details in section 3.2.3. Section 5.4.2 describes data mining techniques that can be used for detecting suspicious activity.

The second phase is responsible for reaction to the previously discovered threats. The appropriate reaction steps are executed using the detected pattern, which describes a threat. Logs that inform about an observed event are generated and sent to the operator. It is possible to perform additional steps that can fulfill operators' needs. All parameters concerning this activity are placed in a policy. The policy connects protected machines' address with actions that should be performed in case determined threat is detected. It can be placed, for example, in the ONIX. Details of the policy will be defined in the course of the research. For example, accordingly to the policy, a detected malicious machine can be removed from network, malicious traffic can be completely removed or only maximally slowed down. In the policy, all parameters can be tuned, for example, a conducted reaction can depend on malicious machine IP address or type of detected threat. This flexibility of the policy tunings gives ability to implement protection reaction accordingly to all operators' needs. For example, an operator can immediately shut down an infected machine of home client. On the other hand, traffic of the business client's critical server can be only slow down. In this phase, services provided by Quality of Service Management DE (QoS_M_DE), Routing Management DE (RM_DE) and Forwarding Management DE(FWD_M_DE) are used. The detailed description of possible reaction is described in section 5.4.3.

5.4.2 Detection techniques

In the introduced self-defending functionality, any detection of malicious or suspicious activity is performed using data mining techniques. There are many techniques that could be used for this purpose, for example, neural networks, statistics or probabilistic methods. However, data mining approach was chosen due to simplicity and intelligibility of extracted knowledge. Using data mining, not only information that some activity is detected can be obtained but also a description of detected activity. This description is in a form of patterns which can be easily understood by human and easily converted by automatic systems for appropriate reaction.

Described in 3.2.3 section threats generate significant amounts of packets that have similar characteristic. For example, scanning for vulnerable machine activity produces vast amounts of packets that have the same source IP address, protocol and port. In data mining techniques this kind of patterns are known as frequent patterns. In literature there are few such patterns, for example, frequent sets, sequential patterns[AS95] or episodes[MTV95]. During the research, a decision on which patterns can be used in the proposed solution was made. The main impact on the decision is associated with the ability to easily use extracted knowledge in other collaborating systems. Currently, there is a lack of IDS/IPS or firewall systems that can utilize complex relations between some networks related events. Due to this fact, sequences and episodes are omitted and pattern called frequent set is chosen for further analysis. Moreover, this decision is justified by the fact that there are many algorithms that can be used for detecting this pattern. It should be emphasized that some of these algorithms can detect this pattern in incremental fashion.

As a result of this finding, in the solution further described in this paragraph, we are interested only in the pattern called a *frequent set*.

The *frequent set* idea was presented by Agrawal in 1993 [AI93]. Data that will be analyzed are treated as collection of item sets. A subset of items that frequently appears in analyzed data is called the *frequent set*. The parameter, called *minimal support,* is used for deciding what is frequent. A support parameter is calculated as a number of item sets that contain a given subset of items. Selection of a minimal support parameter decides which item sets are detected.

Frequent set pattern can be detected in data in many ways. The most known algorithm called A-Priori is described by Agrawal et al. [AS94]. The other known solutions that are using tree structures are FP-Trees [HPY00] and CATS trees [CZ03]. The last two mentioned algorithms are interesting, because of the ability of incremental mining. This advantage causes that patterns can start to be discovered when data are acquired, and result is immediately known when last item set is delivered. In this place one remark must be made. All presented algorithms discover frequent sets in whole data sets. Due to in our solution the detection of suspicious events should be in real time, acquired data is partitioned and analyzed in each partition.

Choosing from a variety of available data mining techniques is not the most important part of setting up the functionality; choosing parameters that will be used during data mining is even more crucial. During the research concerning currently observed threats, some of them are investigated, and a short description of this work is presented in section 3.2.3. of the document. Based on this research, interesting features are proposed. For further analysis only the following packet fields are used:

- protocol, this attribute can assume one value from this set: ICMP, UDP, TCP and OTHER,
- source and destination IP address,
- for TCP and UDP packets, source and destination port,
- for TCP protocol, interesting flags: SYN, RST, FIN,
- for ICMP packets, ICMP code and type,
- for UDP and OTHER protocol, packet/datagram size.

A simple case study that describes the detection of suspicious activity using frequent sets is presented below. For first experiments simple, not incremental algorithm extracting frequent sets is used.

First, data that have to be analyzed must be converted to a form appropriate for the above described algorithm. In our case, all interesting packets are converted to item sets. Each interesting packet's features presented in section 3 are converted to items. For example, TCP packet sent from IP address IP_A and source port 1234 to a machine with IP address IP_B and a destination port 80, with set SYN flag. Such packet is described as the following:

<TCP, SRC_IP_A, DST_IP_A, SRC_PORT_1234, DST_PORT_80, SYN>.

As it has been already mentioned, because used algorithm analyzes the whole delivered data, packets are acquired for a period of time and after that frequent sets are discovered.

In case in an analyzed period one machine starts scanning for vulnerable web servers, many packets are send to the same port but various destination IP addresses. In this situation a frequent set presented below will be discovered:

<TCP, SRC_IP_A, DST_PORT_80, SYN>

A detected pattern is easily understood and almost self describing; in analyzed period there are many packets using TCP protocol with SYN flag set sent from machine using IP address IP_A to port 80. Because in extracted pattern there is a lack of information about source port and destination address, it can be assumed that those features of packets are changing.

It may seem that this approach is similar to a simple counter that counts packets, for example, to given ports. However, the advantage of the method is that any combination of analyzed features that appears in monitored packets can be detected. For example, using this method a DDoS attack that uses UDP packets sourced from port 3333 that has 533 bytes long can be detected. Using counters each combination of protocol, packet port and size should have its own counter.

For assurance of the high performance, in presented scenario only a minimal amount of data is analyzed. In normal network activity only high level information about protocols is analyzed. Where something suspicious is detected, additional information about IP address is requested from MON_DE. After detection of suspicious IP address, additional detailed information is requested only for them. Simple scenario of cooperation between NODE_LEVEL_SEC_MNGT_DE and MON_DE is described in section 5.4.5.

5.4.3 Activities after detection

The detection of the threat is only the first step which leads to self-defending networks. The other very important step is associated with the reaction that can be applied to suspicious machines. There are few mechanisms provided by other DEs which can be utilized.

Below presented reaction could be configured in the policy on the level of both threat and machine. This gives an operator flexibility and ability to tune reactions accordingly to its needs. For example, if we are confident about malicious activity of home user, offending machine can be completely disconnected from protected network. Defining appropriate policy for all machines is crucial for minimising possible problems with false positives. In contrast to previously mention example, a detection of business critical server which sends suspicious traffic causes only generation of alert to operators' staff and slows down suspicious traffic. Additionally during creation of the policy we should be concern about it impact on performance and network behaviour. For example, enabling action which performs removing machines from routing table, in mass attack can overwhelms routers with blacklisting entries. In contrast, disabling RSVP tunnels negotiation can be sign for attacker that this is self-protected network an in effect deters him.

Below the proposed possible reactions are described with all details.

5.4.3.1 Stopping suspicious activity

The simplest method of protecting other machines when one in network is infected or misbehave is associated with removal of hostile traffic. Accordingly to the policy, either all traffic sent from and to this machine or only suspicious traffic can be removed.

This can be done in different ways, using various DEs. Below investigated mechanisms that can be used for this purpose are presented.

5.4.3.1.1 Dropping unwanted traffic

One of the simplest method for denying traffic is using of ACL (Access Control Lists). ACL is a list of rules, which allows or denies forwarding of defined traffic by network devices. By applying suitable rules, a network policy can be achieved. When this type of reaction is used, new rule that denies previously detected suspected traffic is generated and applied to FWD_M_DE. The decisive effect of this activity will be observed, when appropriate entries will be placed in ACL in access devices. In those cases malicious, unwanted traffic will be removed from network as soon as possible.

5.4.3.1.2 Removing IP's from routing table

When there is a lack of appropriate route for a given IP address in routing table, traffic destined to this IP is removed from network (this process is often called "dropping"). Deliberate removing of some routes can be treated as a method for removing unwanted traffic in case all packets to this destination should be removed. Some network devices introduce special interface (often called null interface) which is used for removing unwanted traffic. In some network devices this is a preferred and the fastest way to filter network traffic [Cisco2]. When this method of reaction is chosen, a special request to RM_DE with IP address that must be block is sent.

5.4.3.2 Slow down of suspicious activity

The above presented method of protection, which denies all traffic associated with malicious machine, may be in some situation inappropriate. Such situation can be observed when the detected machine is very important (business critical) or when detected activity is not for sure hostile (for example scanning). In these cases other reaction can be utilized, like slowing down suspected traffic to the time when future investigation gives new information. Till this time, machine has connection but it is slower.

Below some QoS mechanisms that can be used are described. What is important, depending on the policy, all traffic to or from this machine or only suspected traffic can be slowed down. This behaviour can be utilized because used data mining detection techniques give patterns that describe suspected traffic in detail.

5.4.3.2.1 Traffic marking and buffering strategy

When QoS is utilized, each packet can be appropriately marked and then forwarded using implemented policy. For example, VoIP packets that must be delivered with minimal delay are sent before FTP packets that are bigger and can wait for a while in devices' buffers. A decision what should be done depends on marking, which is placed in IP packet (TOS, DSCP fields).

When a self-defending functionality is implemented, suspicious packets can be appropriately marked and removed as the first packets from network, where congestion is discovered. Using well known QoS class names, detected suspicious traffic should be marked as "best effort" or "scavenger". In some situation special traffic class can be developed, which is used only for suspicious traffic. In this situation not only appropriate buffer strategy is associated with this marking but also QoS mechanisms, for example, RED (described in next section) are used.

In this case, after threat detection NODE_LEVEL_SEC_MNGT_DE sends request to QoS_M_DE for appropriate marking of suspicious traffic. Additional parameters of this request describe suspicious traffic and its marking.

5.4.3.2.2 RED/WRED

Random Early Detection and Weighted Random Early Detection are mechanisms used for congestion avoidance. Both mechanisms start removing randomly some small amount of packets when detecting congestion. This activity slows down traffic because dropped packets must be retransmitted after timeout expired.

This mechanism can be used for slowing down suspicious traffic. In the proposed solution, the suspected traffic can be deliberately removed with higher rates than other types of traffic. Special request with a description of suspicious traffic should be send to QoS_M_DE; consequently, RED mechanism is reconfigured.

5.4.3.2.3 Denial of RSVP tunnel negotiation

In addition to above presented reactions, when machine is suspected the ability to negotiating appropriate QoS level via RSVP should be disallowed. As in the case of two above mention mechanisms, a special request with IP address of suspected machines is sent to the QoS_M_DE.

5.4.4 Cooperation with Monitoring Team (WP4)

As described in previous paragraphs detection method could be easily integrated with MON_DE. MON_DE provides special services that are used by NODE_LEVEL_SEC_MNGT_DE. Cooperation with MON_DE assures high efficiency of presented mechanism as some simple aggregations are performed by it.

A simple scenario of communication exchange that leads to discovery of sample threats is presented below. The scenario consists of two scenes: in the first, an infected machine performing DoS attack is detected, in the second a malicious user who executes network scanning is detected.

These are common steps for scene 1 and 2.

1. NODE_LEVEL_SEC_MNGT_DE communicates with MON_DE and requests gathering of aggregation data for TCP, UDP, and ICMP protocols and sets aggregation period time. Aggregation data contains number of packet per each

protocol that was observed during defined aggregation period. This state of MON_DE is referred as aggregation mode.

2. After end of each period monitoring data are sent to NODE_LEVEL_SEC_MNGT_DE which analyzes them using data mining techniques. This activity is performed as long as no anomaly is detected.
3. When NODE_LEVEL_SEC_MNGT_DE detects traffic increase in one protocol, the request for more detailed data is send to MON_DE. From that moment MON_DE aggregates data in per IP address's basis. In both described scenarios an increase in number of TCP packets is detected.
4. Acquired detailed data are analyzed providing interesting IP. NODE_LEVEL_SEC_MNGT_DE sends a request to MON_DE for more details only for that IP. Detailed data contains information about used ports, packets size, interesting flags etc.

   Traffic associated with other IP addresses turns back to aggregation mode. This solution reduces number of gathered data which gives high performance.

Step 5 is applied for scene 1: infected machine performing DoS attack

5. Acquired detailed data related to interesting IP are analyzed. As a result, data mining pattern is revealed. The pattern describes activity generated by this IP as traffic consisting mostly packets with this same size and flags, directed to this same port. This pattern with high probability suggests DoS attack. Such information is used in the reaction phase.

Step 6 is applied for scene 2: user starts network scanning tool

6. Acquired detailed data related to interesting IP are analyzed. As a result, data mining pattern is revealed. Pattern describes activity generated by this IP as traffic consisting mostly packets with SYN flags, directed to many ports. This pattern with high probability suggests scanning activity. Such information is used in the reaction phase.

5.5 Vulnerability Detection using Fuzzing

During implementation or creation of specifications, usually it is not foreseen how a software or a protocol should react to wrong or unexpected input. It is only foreseen how systems should handle estimated input. Even testing techniques usually test if a system does what it is supposed to do for expected input. Fuzzing handles this incompleteness of software testing. Fuzzing is known as a special case of software testing. It is a method to discover faults by providing unexpected input while exceptions are monitored. In testing as well as in fuzzing three different types are possible: white-box, grey-box, and black-box testing.

A fuzzing method is called white-box when the method has complete knowledge of the source code of a software or of the fields and messages of a network protocol. Although, complexity for a complete analysis of the source code or protocol may be very high, it provides good coverage of detected faults. While in black-box testing, we consider to have no knowledge of the mode of operation of the system. Complexity is low and such a method is reproducible for other systems, but coverage is not guaranteed. In addition, there is the grey-box method, which lies in between of previously described methods. For this method we consider to have some knowledge of the mode of operation of the protocol or software.

We focus our work on white-box fuzzing. So we first need to analyze the protocol specifications. After analyzing the protocols, we need to define how we want to fuzz it, which fuzzing strategies we will adopt and how we will do fuzzing on a practical way. We have to decide whether to take an existing fuzzing tool or to start from scratch. So first we will give an overview of existing fuzzing environments, and other helpful tools. Afterwards, we explain how we analyze protocols and which fuzzing strategies we use to detect vulnerabilities in IPv6++ protocols.

5.5.1 Existing Tools

5.5.1.1 Fuzzing Environments

Nowadays, many different fuzzing tools exist. Although a lot of fuzzing tools are available, not all of them can be useful for our needs. This section gives an introduction to most popular fuzzing environments and explains which can be useful and which cannot.

One group of fuzzers is formed by EXE [CTGE06], Flayer [DO07], and CESE [MX07]. All of them use symbolic execution to check for vulnerabilities in system code. Instead of testing the code with manually generated input, the symbolic execution allows to define some variables as symbolic. The system code is then checked using a constraint solver with all possible value for these variables. This approach is not useful in our case, as this is a white-box approach for system code, and we do not know how the messages transferred are interpreted by the system. Another inconvenience of symbolic execution is non-proof for scalability.

Whereas, Autodafé [V05] is a taint-based fuzzer which tests for buffer overflows. This technique allows seeing where and due to which method an unsafe C function is used. This approach decreases complexity of fuzzing, as only relevant data is checked. Nevertheless, this approach is not useful for us, as it concentrates only on buffer overflows.

Furthermore, another tool called Sulley [Sulley] is interesting for our goal. Sulley is a python-based fuzzing framework. The main difference of this framework and other frameworks is, that it does not only provide data generation, but it also provides monitoring. It also provides packet-capture, which makes this tool interesting for us.

5.5.1.2 Useful Tools

Other useful tools exist that we could use for our fuzzing framework. One of these tools is Scapy [scapy], it allows us to create IPv6 packets, and to intercept and fuzz packets. It is a powerful and interactive packet manipulation program. Another interesting tool is ettercap [Ettercap], which provides sniffing, and content filtering of live connections.

5.5.2 Fuzzing IPv6++ Protocols

This subchapter explains how we can use fuzzing to detect unknown vulnerabilities in IPv6++. First, we need to analyze a protocol, as we will use white-box fuzzing to get best coverage. In Deliverable 2.3 [D2.3] some specifications for IPv6++ protocols are described: for DHCPv6++, ICMPv6++, and ND++. We choose to start our analysis with the specification for enhanced Neighbor Discovery Protocol (ND++). Second, we need to define the fuzzing strategies to apply to our analyzed protocol.

5.5.2.1 Protocol Analysis

Finite State Machines (FSM) can be used to model the behavior of protocols. An FSM consists of states and transitions between states. An FSM helps to see what messages are sent and received in which state.

We started our protocol synthesis by modeling the address-autoconfiguration that is part of Neighbor Discovery. So in following figure we can see the FSM for address-autoconfiguration.

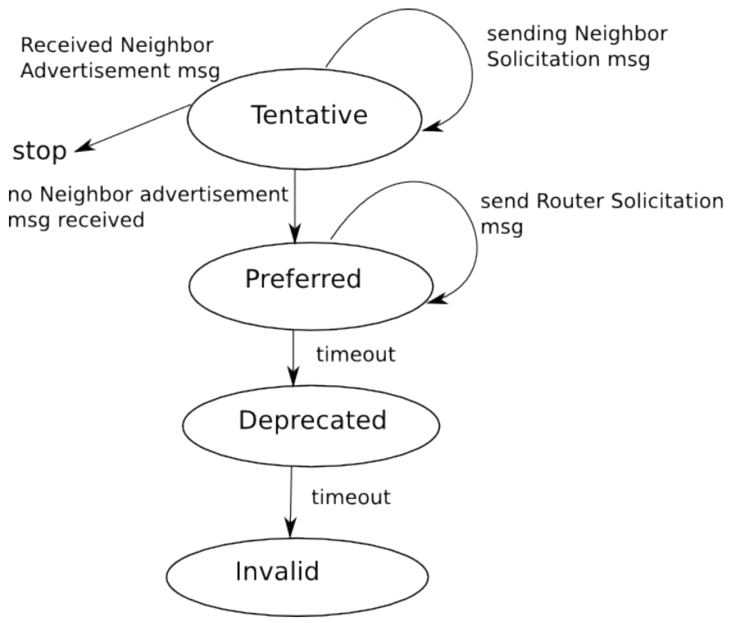

Received Neighbor Advertisement msg
sending Neighbor Solicitation msg
Tentative
stop
no Neighbor advertisement msg received
send Router Solicitation msg
Preferred
timeout
Deprecated
timeout
Invalid

Figure 5 modelling Address-Autoconfiguration

In order to have a complete view of a protocol, to model an FSM is not sufficient. We also need to analyze message formats.

5.5.2.1.1 Analyzing message formats :

In order to analyze message formats, we need to first list all the messages used in the protocol, and then synthesize what fields compose the message. Once we have all the fields of one message, we define what type each field has, and what the default value is of that field. In ND++ we have following messages:

- Neighbor Solicitation
    - 1-hop capabilities
    - multi-hop capabilities
- Neighbor Advertisement
    - 1-hop capabilities
    - multi-hop capabilities
- Router Solicitation
- Router Advertisement

ND++ compared to ND has some modifications on the Neighbor Solicitation and Neighbor Advertisement messages. Due to this reason we focus on these messages for this document. In [D2.3] the format of neighbor solicitation message with multi-hop capabilities (see Table 2), neighbor advertisement message with 1-hop capabilities (see Table 3), and one neighbor advertisement with multi-hop capabilities (see Table 4) is described. The neighbor solicitation message with 1-hop capabilities is not described as it has the structure of the original message format.

Neighbor Solicitation with multi-hop capabilities :	
–<IPsource> <IPdest> <HopLimit> <ICMP-Fields>	
<IPsource>	(bin, 128bits, prefix+interface_address)
<IPdest>	(bin, 128bits, multicast address)
<HopLimit>	(HopLimit, 8bits, 255)
<ICMP-field1>	(type, 8bits, 135)
<ICMP-field2>	(code, 8bits, 1)
<ICMP-field3>	(checksum, 16bits, ICMP checksum)
<ICMP-field4>	(hopcount, 8bits, hopcount)
<ICMP-field5>	(reserved, 24bits, unused)
<ICMP-field6>	(target address, bin, 128bits)

Table 2 Neighbor Solicitation with multi-hop capabilities message

Neighbor Advertisement with 1-hop capabilities :
–<IPsource> <IPdest> <HopLimit> <ICMP-Fields><MPR Parameters><MPR Announcement>

<IPsource>	(bin, 128bits, prefix+interface_address)
<IPdest>	(bin, 128bits, multicast address)
<HopLimit>	(HopLimit, 8bits, 255)
<ICMP-field1>	(type, 8bits, 136)
<ICMP-field2>	(code, 8bits, 0)
<ICMP-field3>	(checksum, 16bits, ICMP checksum)
<ICMP-field4>	(R, router flag, 1bit)
<ICMP-field5>	(S, Solicitation flag, 1bit)
<ICMP-field6>	(O, override flag, 1bit)
<ICMP-field7>	(reserved, 29bits, unused)
<ICMP-field8>	(target address, bin, 128bits)
<MPR Parameter-field1>	(Type, 8bits, 6)
<MPR Parameter-field2>	(Length, 8bits, variable)
<MPR Parameter-field3>	(Willingness, 16bits,[0,1,3,6,7])
<MPR Parameter-field4>	(Link Code, 8bits,)
<MPR Parameter-field5>	(AFN, 8bits, address fields number)
<MPR Parameter-field6>	(reserved, 16bits, unused)
<MPR Parameter-field7>	(Neighbour Address (1-n),.., neighbour addresses)
<MPR Announcement-field1>	(Type, 8bits, 7)
<MPR Announcement-field2>	(Length, 8bits, 2m+1)
<MPR Announcement-field3>	(reserved, 48bits, unused)
<MPR Announcement-field4>	(MPR address, (1-m)*32bits, announcement of MPRs)

Table 3 Neighbor Advertisement with 1-hop capabilities

Neighbor Advertisement with multi-hop capabilities :	
–<IPsource> <IPdest> <HopLimit> <ICMP-Fields><Topology and Link Control Info Option>	
<IPsource>	(bin, 128bits, prefix+interface_address)
<IPdest>	(bin, 128bits, multicast address)
<HopLimit>	(HopLimit, 8bits, 255)
<ICMP-field1>	(type, 8bits, 135)
<ICMP-field2>	(code, 8bits, 1)
<ICMP-field3>	(checksum, 16bits, ICMP checksum)
<ICMP-field4>	(R, router flag, 1bit)
<ICMP-field5>	(S, Solicitation flag, 1bit)
<ICMP-field6>	(O, override flag, 1bit)
<ICMP-field7>	(hopcount, 8bits, hopcount)
<ICMP-field8>	(reserved, 21bits, unused)
<ICMP-field9>	(target address, bin, 128bits)
<TLC info-field1>	(Type, 8bits, 6)
< TLC info -field2>	(Length, 8bits, variable)
< TLC info -field3>	(ANSN, 16bits, advertised neighbour sequence number)
< TLC info -field4>	(Link Code, 8bits, (neighbour type, link type))
< TLC info –field5>	(reserved, 16bits, unused)
< TLC info –field6>	(Neighbour Address, (1-m), neighbour addresses)

Table 4 Neighbor Advertisement with multi-hop capabilities message

5.5.2.2 Fuzzing Strategies

After analyzing the protocol and the messages, we need to define what strategies we want to apply in order to detect vulnerabilities efficiently. Many different methods and strategies exist for fuzzing [HSL08]. One possibility is to modify one or more data fields. This will have as consequence that an endpoint in the network has to handle wrong input. The data fields can be modified by deleting the fields, by inserting fields, or by modifying the value of the data field. Another possibility is to change the message type, by saying that this message is of another type. These described methods fuzz the content of the message. In fuzzing we also can use as strategy to send a different order of messages by inserting, repeating, or dropping messages inside one session.

Practically, this means that we need to create/mutate packets with changed data fields, based on the message analysis we have done previously. We could start by changing the type field in ND++ messages. Normally, the Neighbor Solicitation message with multi-hop capabilities has as value for the type field 135, in order to define that this message is a neighbor solicitation message. The scope is to change it to some other value, e.g. 137 and to notice if any unwanted behavior appears. If we are able to see that the destination host starts to behave bizarre or even that it crashes, we have detected a vulnerability. The same can be done by changing the date fields with the previously defined strategies. In Table 5 the different possibilities for fuzzing fields of messages are shown.

Fuzzing Strategies				
Fuzzing Strategy	Message to be fuzzed	Fuzzed Message field	Default value / structure	Fuzzed value / structure
Fuzzing type field	Neighbor solicitation (multi-hop)	ICMP-field : Type	135	137
Mutating field value	Neighbor Advertisement (1-hop)	Hoplimit	255	256
Deleting data field	Neighbor Advertisement (1-hop)	MPR Parameter – field1 : type	<IPsource> <IPdest> <HopLimit> <ICMP-fields> <MPR-parameter-field-1> <MPR-parameter-field-2> <MPR-parameter-field-3> <MPR-parameter-field-4> <MPR-parameter-field-5> <MPR-parameter-field-6> <MPR-parameter-field-7>	<IPsource> <IPdest> <HopLimit> <ICMP-fields> <MPR-parameter-field-2> <MPR-parameter-field-3> <MPR-parameter-field-4> <MPR-parameter-field-5> <MPR-parameter-field-6> <MPR-parameter-field-7> <MPR Announcement fields>

			<MPR Announcement fields>	
Inserting data field	Neighbor Advertisement (multi-hop)	TLC info field 2 : Length	<IPsource> <IPdest> <HopLimit> <ICMP-fields> <TLC info field 1> <TLC info field 2> <TLC info field 3> <TLC info field 4> <TLC info field 5> <TLC info field 6>	<IPsource> <IPdest> <HopLimit> <ICMP-fields> <TLC info field 1> <TLC info field 2> <TLC info field 2> <TLC info field 3> <TLC info field 4> <TLC info field 5> <TLC info field 6>

Table 5 Examples of the applying different fuzzing strategies

Furthermore, we can also use the previously defined Finite State Machine (FSM) of the address-autoconfiguration process in the Neighbor Discovery Protocol, to know which message type leads to which transition. Based on this, we can twist the order of the messages and follow-up if any abnormal behavior can be detected. Once, a node wants to start address-auto-configuration, it will send a neighbor solicitation to the network. So we can imagine the following scenario: another node will send just a few seconds after receiving a neighbor solicitation, another neighbor solicitation with the same configurations, asking for more information about that neighbor. So let's assume, the node did not yet have the time to configure itself correctly, as it is still waiting for a neighbor advertisement, which indicates that another node already obtains the same IPv6 address. Respectively, it is waiting for a timeout to configure itself correctly with the knowledge that the address is not in use. What will happen if that new node receives a neighbor solicitation message, will it anyhow respond with an advertisement message, although it is not yet sure that the address is unique?

One important thing in vulnerability detection is the reproducibility of vulnerabilities. Therefore, we need to be able to deduce what exactly caused the host to behave abnormally. Every event that has taken place before we detected the vulnerability can be part of the cause for the abnormal behavior. In order to have the reproducibility of vulnerabilities, we must be able to replay every event that caused the detection of the vulnerability.

5.6 Conclusion

In this chapter we proposed new Decision Elements for integrating a security level into the GANA architecture. Security management DEs are proposed on two different levels, one on node-level, and one on network level. We present an access control model in relation to GANA and ONIX. The same, we showed self-defending mechanisms for autonomic networks based on data mining techniques. Finally, we showed how to assess vulnerability detection for IPv6++ protocols using fuzzing mechanisms.

6 Conclusions & Future Work

This chapter summarizes the contributions of this deliverable. Moreover, it outlines the key future steps that will be followed towards fulfilling the goals of the security task.

Access control, DEs, and Secure communication

After the specification of the GANA threat model, the next steps will be,

- Definition and specification of the security profile that the administrator uploads on ONIX before the network boots up,
- Design and specification of the role of the Sec_Management_DE in authenticating and authorizing the other nodes to access ONIX in order to upload or retrieve data to/from it,
- Investigate how key management solutions can be used here,
- Design and specification of the interconnection between the Sec_Management_DE and the security DEs on the nodes.

Trust Models & Trust Management

We presented our work so far in Trust models and Trust management in the EFIPSANS framework. We consider trust management to be fundamental in the case of autonomic networks. After our analysis it appears that a mix of global and local trust management will be appropriate to use in an autonomic future internet.

Two models examined for global trust management produce similar results. On the other hand ROCQ retains the local values making it more suitable to be used in an autonomic environment. The Bayesian model produces on the other hand reliable results for the case of "friendly" environments while it collapses as the hostility increases to more than 60% malicious users in total.

Even though we produced some results regarding the appropriateness of the models, we still need to identify the way these models could be embodied in IPv6 and what this means in terms of scalability and efficiency. These are the two major issues that will be investigated during the third year of EFIPSANS.

Self-defending functionality

While performing a research on self-defending functionality, key points presented below have been identified. Using the proposed solution, classes of network threats that can be detected are defined and characterized. State of the art data mining techniques are reviewed and those which are able to detect previously mentioned treats are selected. Additionally, the research investigates possible reactions, which appear after a detection of malicious machine, that use some service provided by other DEs. The first draft of communication exchange, which leads to detection of malicious machine, has been agreed with WP4. It has been also decided that further research will focus only on data mining algorithms that find patterns called frequent sets.

The next step of the research is associated with defining details that allow making the proof of concept prototype. At first, all details concerning communication with monitoring DE will be investigated, in cooperation with WP4. During this research exact messages and their content will be developed. Additionally, all possible message exchanges will be examined.

Due to operators' need, the whole system should be highly tuneable. Therefore, all parameters concerning this functionality will be stored in a policy. The content of the policy will be proposed in the course of the research. This research is associated with defining all possible reactions that GANA framework could utilize. Details of communication with other DEs that take part in this process will be agreed.

The implementing phase will start, when all details are agreed, initiating a development of prototype self-defending functionality of SM_DE.

Fuzzing

We have shown our framework for vulnerability detection in IPv6++ protocols. The framework defines how to analyze protocols, and demonstrates it on the enhanced Neighbor Discovery protocol. Our white-box approach first elaborates a finite state machine and second investigates on the structure of the messages. After these examinations, strategies for vulnerability detection using fuzzing are presented. Our future work consists in elaborating a feedback-control system, which allows us to see how effectively the strategies are fuzzing a protocol. Furthermore, we want to use the output of the feedback-control system to produce a game theoretical model, with the objective to make a conclusion about which strategies act as optimal choices.

7 REFERENCES

[ABE01] K. Aberer and Z. Despotovic, Managing Trust in a Peer-2-Peer Information System. In Proceedings of the 10th International Conference on Information and Knowledge Management (ACM CIKM), New York, USA, 2001.

[ABPW99] Alberts, Christopher J.; Behrens, Sandra G.; Pethia, Richard D.; & Wilson, William R. Operationally Critical Threat, Asset, and Vulnerability EvaluationSM (OCTAVESM) Framework, Version 1.0 (CMU/SEI- 99-TR-017). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, June 1999.

[AL08] Mohamad Aljnidi, Jean Leneutre, "Security Solutions in Mobile Autonomic Networks", 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008, April 2008.

[AI93] R. Agrawal, T. Imielinski, A Swami, Mining Association Rules Between Sets of Items in Large Databases, Proceedings of ACM SIGMOD Int. Conf. Management of Data, (1993)

[AS94] R. Agrawal, R. Srikant, Fast algorithm for mining association rules, In J.B. Bocca, M. Jarke, and C.Zaniolo, editors.Proceedings 20th International Conference on Very Large Databases, pages 487-499, 1994

[AS95] R. Agrawal, R. Srikant: Mining Sequential Patterns, In Proceedings of 1995 Int. Conf. Data Engineering (ICDE'95), 3–14, Taipei, Taiwan, (1995)

[BAT04] R.Battiti, Self-management in Autonomic Communication: Trust and Reputation, 2004

[BJ05] J Baras, T Jiang: «Managing trust in self-organized mobile ad hoc networks»,proc.12th Annual Network and Distributed System Security Symposium, California, 2005

[CAS08] Roberto G. Cascella, Costs and Benefits of Reputation Management Systems

[Cisco1] [ONLINE] http://www.cisco.com/en/US/solutions/collateral/ns340/ns394/ns171/ns441/networking_solutions_whitepaper0900aecd8072a537.html

[Cisco2] [ONLINE] http://supportwiki.cisco.com/ViewWiki/index.php/Null_interface

[COR02] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, Choosing reputable servants in a p2p network. In Eleventh International World Wide Web Conference, Honolulu, Hawaii, May 2002.

[CPW03] Chess, D.M., Palmer, C.C., White, S.R., "Security in Autonomic Computing Environment", IBM Systems Journal 42, 2003.

[CTGE06] Cristian Cadar, Paul Twohey, Vijay Ganesh, and Dawson Engler. EXE: Automatically Generating Inputs of Death Using Symbolic Execution. In Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS), Virginia, USA, November 2006

[CZ03] W. Cheung, O. Zaïane, Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint, 7th International Database Engineering and Applications Symposium (IDEAS 2003), Hong Kong, China. IEEE Computer Society, (2003)

[D1.5] EFIPSANS Deliverable 1.5

[D2.2] EFIPSANS Deliverable 2.2

[D2.3] EFIPSANS Deliverable 2.3

[D2.4] EFIPSANS Deliverable 2.4

[DO07] Will Drewry and Tavis Ormandy. Flayer: exposing application internals. In WOOT '07: Proceedings of the first USENIX workshop on Offensive Technologies, pages 1-9, Berkeley, USA, 2007. USENIX Association.

[DPSCCL07] D. Donato, M. Paniccia, M. Selis, C. Castillo, G. Cortese and S. Leonardi, "New Metrics for Reputation Management in P2P Networks", in Proc. 3rd International Workshop on Adversarial Information Retrieval on the Web, Canada, pp. 65-72, May 2007

[Ettercap] http://ettercap.sourceforge.net/

[F97] Fyodor, [ONLINE] The Art of Port Scanning, http://nmap.org/nmap_doc.html

[GAR04] A. Garg and R. Battiti: The reputation, opinion, credibility and quality (ROCQ) scheme. Technical Report DIT-04-104, University of Trento, July 2004.

[GAR05] A. Garg, R. Battiti, and R. Cascella, Reputation management: Experiments on the robustness of ROCQ. In Proceedings of the 7th International Symposium on Autonomous Decentralized Systems (First International Workshop on Autonomic Communication for Evolvable Next Generation Networks), pages 725.730, Chengdu, China, Apr. 2005.

[GOL09] J.Goldbeck, Computing with social trust, Springer 2009.

[HLOS06] Shawn Hernan and Scott Lambert and Tomasz Ostwald and Adam Shostack, "Threat Modeling: Uncover Security Design Flaws Using The STRIDE Approach", MSDN Magazine 2006.

[HPY00] J. Han, J. Pei, Y. Yin, Mining Frequent Patterns without Candidate Generation, Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Dallas, Texas, United States, (2000)

[HSL08] Y. Htsu, G. Shu, D. Lee, "A Model-based Approach to Security Flaw Detection of Network Protocol Implementations", Network Protocols 2008.

[HYU01] Lee Hyun, rok: Multiple Selective Mutual Authentication Protocol For Peer-to-Peer System, 2001.

[KEP03] Jeffrey O. Kephart and David M.Chess, The Vision of Autonomic Computing IBM Thomas J. Watson Research Center, 2003.

[KNMC06] Andreas Klenk, Heiko Niedermayer, Marcus Masekowsky, Georg Carle, “An Architecture for Autonomic Security Adaption”, 2006.

[KSM03] S.D. Kamvar, M.T. Schlosser, H.G. Molina: “The EigenTrust Algorithm for Reputation Management in P2P Networks”, proc. 12th international conference on World Wide Web, pp: 640-651, 2003

[LPY08] H. Lutfiyya, M. Perry and C. Yew, “*Trust and Autonomic Systems*”, 2008 International MCETECH Conference on e-Technologies

[MTV95] H.Mannila, H. Toivonen, A.I. Verkamo, Discovering Frequent Episodes in Sequence, Proceedings of the First International Conference on Knowledge Discovery and Data Mining, Montreal, Quebec, 144–155, (1995)

[MX07] Rupak Majumdar and Ru-Gang Xu. Directed test generation using symbolic grammars. In ESEC-FSE companion '07: The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, pages 553-556, New York, NY, USA, 2007. ACM.

[NMAP] [ONLINE] http://nmap.org/book/man.html#man-description

[NT94] B.C. Neuman, T.o, Tsapos: «Kerberos: An Authentication Service for Computer Networks»,IEEE Communications Magazine, vol.32(9), pp 33-38, September 1994

[PAT06] Rinkesh Patel, Real-Time Trust Management for Agent Based Online Auction System, Fall 2006.

[PH08] N. Provos, T. Holz, Virtual Honeypots: From Botnet Tracking to Intrusion Detection, Addison-Wesley, ISBN 978-0-321-33632-3,2008

[PHM03] W. T. Polk , N. Hastings , A. Malpani: “Public Key Infrastructures that Satisfy Security Goals”, IEEE Internet Computing, pp. 60-67, July 2003

[PRE05] Baptiste Pretre, Attacks on Peer-to-Peer Networks. Dept. of Computer Science Swiss Federal Institute of Technology (ETH) Zurich Autumn 2005.

[QH08] D. Quercia and S. Hailes. MATE: Mobility and Adaptation with Trust and Expected-utility. International Journal of Internet Technology and Secured Transactions, 2008.

[QLHCB06] D. Quercia, M. Lad, S. Hailes, L. Capra and S Bhatti: «STRUDEL: supporting trust in the dynamic establishment of peering coalitions», proc. 2006 ACM symposium on Applied computing, pp: 1870 – 1874, Dijon, France, 2006

[RAH00] A. Abdul-Rahman and S. Hailes, Supporting Trust in Virtual Communities Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.

[Scapy] http://www.secdev.org/projects/scapy/

[SET05] G. Suryanarayana, J.R. Erenkrantz , R.N. Taylor: "An Architectural Approach for Decentralized Trust Management", IEEE Internet Computing, pp. 16-23, Nov. 2005

[SSB07] A. Bhargav-Spantzel, A.C. Squicciarini, E. Bertino: Trust Negotiation in Identity Management. IEEE Security & Privacy 5(2): 55-63, 2007

[Sulley] http://code.google.com/p/sulley/

[V05] Martin Vuagnoux. Autodafé an Act of Software Torture. In Proceedings of the 22th Chaos Communication Congress, pages 47-58, Berlin, 2005. Chaos Computer Club.

[WAN03] Yao Wang, Julita Vassileva, Bayesian Network-Based Trust Model. In Proceedings of Second International Workshop Peers and Peer-to-Peer Computing, July 14, 2003. Melbourne, Australia.

[WES08] Andrew G. West, Reputation Management Algorithms & Testing, 2008.

[WKLS09] Andrew G. West, Sampath Kannan, Insup Lee, Oleg Sokolsky, An Evaluation Framework for Reputation Management Systems, Department papers 2009.

[WV03] Y. Wang, J. Vassileva: "Bayesian Network-Based Trust Model", proc. 2nd International Workshop Peers and Peer-to-Peer Computing, Melbourne, Australia, July 14, 2003.

[XIO03] L. Xiong and L. Liu, A reputation-based trust model for peer-to-peer ecommerce communities. In IEEE Conference on E-Commerce (CEC'03), 2003.

[ZL08] H. Zhao and X. Li : « H-Trust: A Robust and Lightweight Group Reputation System for Peer-to-Peer Desktop Grid», proc. 28th International Conference on Distributed Computing Systems Workshops (ICDCS '08), pp. 235-240, 2008