UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Facoltà di Ingegneria

Corso di Dottorato di Ricerca in Ingegneria Informatica ed Automatica
XX Ciclo
Dipartimento di Informatica e Sistemistica

On the Effective Exploitation of Distributed Information for Cooperative Network Security and Routing Optimization

Francesco Oliviero

Ph.D. Thesis

Tutor
Prof. Simon Pietro Romano

Coordinator
Prof. Luigi Pietro Cordella

Cotutors
Prof. Carlo Sansone
Prof. Edward W. Knightly

November 2007

Abstract

The wide deployment of more and more sophisticated services and application over Internet imposes to careful project solutions for face the threats that this spreading involved. Unfortunately, the computer networks were designed neglecting the problem of security, since they ware thinking as "open systems" where anyone could use them as it liked, and malicious users was not considered.

The lack of security requires to study alternative solutions that integrate the existing network technologies. The interest in computer network security is growing in the last year and new ideas have been proposed.

In this thesis we present a cooperative approach to network security. By sharing information about evidence of anomalous user's activities it is possible to improve the effectiveness of the overall system by a careful prevention and detection of attacks. In this way the cooperation is able to overcome the intrinsic design limits of existing computer networks.

By analyzing several collaborative systems, we define a general framework for the design and implementation of a reliable cooperative solution.

Based on this framework, we implement two systems that share common project's principles: a cooperative DoS detection system and a new secure routing protocol for wireless networks.

The results prove the goodness of design and the capability of framework proposed to be extended to several problems of network security.

Acknowledgments

I would like to dedicate this manuscript to my family, my father Antonio, my mother Maria Grazia and my brother Salvatore. Thank you so much for your support and enthusiasm. I could not have done this without you.

I am deeply grateful to my advisor Prof. Simon Pietro Romano, or just Simon Pietro, for being a great mentor, but firstly a great friend. I sincerely thank you for your patience and support that you always gave me.

My sincere thanks goes to Prof. Carlo Sansone. His suggests and critiques have been so precious for my works. I would like to thank to my colleague and friend Claudio Mazzariello, who shared with me all these years of hard studies.

I would like to thank to the entire COMICS group, I have found a great atmosphere. It is a great family, and I am proud of being part.

I would like to express my gratitude to Prof. Giorgio Ventre, for believing in me and for giving me the possibility to begin this hard but gratifying road of PhD.

I would like to thank to Prof. Edward Knightly and the Rice Network Group at Rice University, for giving me the opportunity to join them and to live a great professional experience.

Finally, let me thank to all my friends, and all people that during these years are and have been by my side.

Contents

List of Figures

List of Tables

Chapter 1

Cooperation and Distributed Information

1.1 Introduction

One of the main strength of Internet since its origin has been the capability of connecting people and systems around the world, increasing the opportunities of communication, collaboration and cooperation among them. The well-known *Information Society* is a result of the growing's process of Internet. More and more people are beginning to use the network to benefit of new services, and new social and geographic scenario is going to open to services providers. The information exchanging is the most important merits of Internet, bringing in contact different cultures and helping for their development.

In spite of all others communication systems as press, telephony or TV, Internet better represents the main expression of the *Power of Communication*. Indeed, its features of *anywhere*, *anytime* and, usually, for *anyone* are the reasons why Internet is more and more perceived by the people as an extraordinarily powerful mean of communication and an inexhaustible source of information. The motivation is that Internet is able to go beyond the temporal and spatial locality.

Several definitions for power of communication have been proposed. We think that the must appropriate one is that power of communication is the

capability to have any kind of advantages from exchanging information.

This is definitely true for people. The communication, indeed, has had a great influence for society development, and it is expression of freedom. It is well-known that the absence of communication produce poverty, unfairness and lack of democracy. Totalitarian regimes find every time ways to control and counter the political impact of communication media, in particular of Internet [1].

The information, or better, the knowledge of the "experiences" around the world can help a community, a nation, to correctly grow. In few words we can say that everything happens around the world can contribute to improve locally the prosperity of people if they know about these events. So we can claim that global information can improve any "local" human process.

This is not true just for the human beings. Far from to be a treaty on the social implications of communication and Internet, indeed, we want just to observed that the information exchange could be a key factor for any computation system too. In this context the communication is just a way to collect information around in order to improve the performance of a system.

Any time we share information among different entities we are talking of "cooperation". In computer system context, as for the human beings, the cooperation can significantly contribute to improve the "performance".

The cooperation sometimes is required as solution for overcome some infrastructure constrains. Several problems, indeed, drive for encouraging the collaboration among the different elements of a distributed system in order to improve the performance.

First of all, the networks are becoming more and more heterogeneous infrastructures, where different technical and methodological solutions cohabit all together. The interoperability between Internet and the cellular phones network is just an example of this heterogeneity. Sometimes providing a service, as a VoIP call, needs to involve several elements of the network which usually have different technologies, softwares, and resources availability. A priori knowledge of these situations can help to properly deploy the service

with better performance in terms of users' satisfaction.

The dynamism of infrastructures is also a reason for adopting collaborative solutions. The networks are more and more dominated by mobile technologies. Such technology has certainly changed the way to communicate. It has introduced remarkable advantages for people, but unfortunately it has required a great effort for providing a suitable coordination needed for such technology. The sensor networks, for example, require to exploit cooperative mechanism to solve the lack of physical network infrastructure. Routing protocol is an example of mechanism that can take advantage from the a cooperative solution.

Sometimes the peculiarities of the infrastructure don't allow to have the reasonable standard of security. Again the wireless networks are the better example of vulnerable infrastructures that can be easily compromised. Frequently, indeed, a malicious elements can easily evade the traditional security solutions attempting the safety and the correctness of the whole system. A collaborative solution, based on information exchange, should mitigate such risks. In such a way also resiliency capabilities should be assure to make the network robust to the faults.

Allowing a system to be self-configurable self-manageable can be realize by a cooperative solution. For example, by adopting the cooperation among the elements of a distributed system should increase the capability of the system itself to isolate the inefficiencies of some elements.

The cooperative scheme seems to be the more fashionable solutions to meet all the mentioned requirements. Generally we can claim that every distributed system which requires a coordination can take advantages by adopting a distributed collaborative solution.

Basement of any cooperative mechanism is the *distributed information*. Every element usually perceives a kind of "limited vision" of world around it. For this reason we have a sort of "local information" distributed in every single node. By collecting all this information we can produce a "global" one. We claim that by sharing global information we can improve local decisions

and computations. Indeed, by merging local and global information it is possible to observe a meaningful improvement in performance also for local process. Isolated entity can improve it capability to take local decision by collecting also data from several points into the network. Wherever a local decision can influence global results of a system, it is important to collect also data of the overall components of the system. This is usual for a cooperative system.

By summarizing, we can claim that several benefit can be gained by adopting a system design solution based on distributed information and cooperation approach for improving the overall performance of system. A global information as an aggregation of local experiences can result an advantage for the effectiveness of a distributed system.

So we can conclude saying that the cooperation is an extraordinary mean for developing people and machines.

1.2 Objectives

Cooperative approach is more and more exploited for improving the effectiveness of distributed systems. As stated in the previous section, cooperative solution seems to be conform to the requirements of all the systems characterized by high dynamism, heterogeneity and vulnerability. Solutions for network security, both in wired and wireless scenario, also should benefit more from this approach. This is true whether we want to improve the effectiveness of existing network security systems by creating a collaborative environments, or we want to study new solution and algorithms.

Every existing solutions, in fact, can benefit from using an approach that allows them to share their results in order to improve the capability of identify correctly a threat. A distributed attack, indeed, can exploit the fact that using multiple sources it can easily prevail over a single security system. For example a Denial of Service (DoS) attack, a common threat for a network, exploits the limited resources of a system to reduce the availability of it. A distributed version of this attack can quickly exhaust the resources before a

single system is able to report any anomalous situation. A solution based on cooperation among single elements, instead, can contribute to detect attacks opportunely and effectively. An isolate security system, indeed, could easily identify an ongoing attack if it has evidences of the attack activities coming from elements close the source. This can be consider an example of "collaborative detection capability" where every local evidence of attack can be supported by the remote ones.

An approach based on information sharing and cooperative computation can be also a useful mechanism to extend with security functionalities traditional distributed protocol as routing algorithm. These protocols are intrinsically cooperative so any secure approach can benefit from this peculiarity.

Generally it is possible to identify three main reasons that suggest to adopt a mechanism based on distributed information and cooperation in the network security context.

Firstly some infrastructural weaknesses can affect meaningfully the security standard of any network environment. Sensor networks and wireless mesh network, for example, are exposed to threats that can exploit the lack of physical infrastructure to launch an attack. A malicious node can easily take legitimate nodes' place creating serious damages to the overall networks. By allowing the nodes to exchange information about the anomalous situation we can obtain a good level of collaboration that can contribute to isolate the malicious elements.

Solutions for network security are growing rapidly in the last years. Intrusion Detection Systems (IDSs) and firewall are just some example of this evolution. Such situation creates an heterogeneity of solutions and approaches more and more sophisticate. These systems, even though efficient suffer sometime to be "isolated". A good solution could be to create a network of security systems to enforce the capability to identify attacks by sharing reports on the malicious activity of some users. We can connect IDSs and firewalls in order to cooperate and improve their performance.

The main reason for the growing interest of both academic and industrial communities for adopting cooperative systems in the network security is the wide diffusion of more and more sophisticated distributed attacks. The Distributed Denial of Service (DDoS) are commonly exploited to launch threats against a system. The cooperation is a good solution for mitigating the problem of distributed attack by means of a massive use of information exchange among the elements of the network trying to properly limit the damages.

The analysis of several example of cooperative network security schemes suggests that an integrated approach is required to better design such solutions. A framework starting from the data collection of the traffic to the methodologies to share information among the systems elements seems to be more appropriate. This framework should also hold in due consideration the peculiarity of the network infrastructure.

Provide a solution that takes into account a global vision of problems of collaboration is one of the main challenges. In this work we want to investigate some aspects of a cooperative system for security network stating from a general approach which can be characterized to different security contexts. We want to analyze the benefits of adopting a framework for distributed collaborative system.

By identifying the essential requirements of a general cooperative approach based on distributed information we will prove the goodness of our assumptions and their capabilities of generalization by applying them in two different security contexts: the volume-based Denial-of-Service detection and the secure optimization of routing protocol in wireless network.

In particular, in chapter 2 we will present the state of art of network security, focusing, in particular, on Denial-of-Service (DoS) threats and the possible cooperative approach to properly defeat them. The main requirements for a collaborative system in the context of network security will be dealt with in chapter 3, providing at the some time several design principles that can be apply for every cooperative solution. Based on these principle in chapter 4 and chapter 5 we will propose our distributed solution for detection

of DoS. The same principle well be applied to design of a new secure routing protocol for wireless mesh networks in chapter 6. Final considerations and future work will conclude this work in chapter 7.

Chapter 2

Distributed Information for Cooperative Network Security: the State of the Art

In this chapter we propose a brief survey of existing methodologies and architectures for cooperative network security. In particular, we want to investigate the current solutions related to the following well-known security problems:

- volume-based Denial-of-Service (DoS) attacks,
- attacks to routing protocols for wireless networks.

These threats both aim to deny or restrict the legitimate usage of network resources. They are widely discussed in the scientific community and present a good margin of investigation.

We will analyze separately the state of the art in both fields. This analysis reinforces our idea to exploit a cooperative approach based on distributed information for their improvement. This survey will be the starting point for further discussions and will serve as an introduction to our original contributions.

2.1 Security Issues in Networked Systems

The problem of security is one of the main challenges for a computer network. In any networked system the need to share data coming from different points increases irremediably the probability that information can be corrupted or acquired by unauthorized users. Moreover, any distributed system requires that each of its components works correctly and that it is always available in terms of both data and service offered. All these problems require solutions that allow to reduce the risk of tampering. Generally it is possible to define some characteristics for network information and resources that must be preserved by a security solution. Network security, indeed, is based essentially on three components:

Confidentiality: It refers to the characteristic of data to be accessed exclusively by authorized users. Bank accounts data are an example of information which needs to preserve confidentiality.

Integrity: It refers to the capability to prevent the corruption and alteration of data. It is important, for example, to assure data integrity during a banking transaction.

Availability: Data must be protected against any attempt to reduce their availability to legitimate users.

By summarizing, every time a remote user exploits network to require a service, for example booking a flight or reserving a hotel room, he first needs that the service is available, and then that his own private information, such as credit card number, is protected against illegal eavesdropping attempts. Likewise, the server needs to know the real identity of the user for bill accounting and the user needs to know the identity of the server which he is providing his own credit card number to.

Any security solution usually attempts to address one of these issues. It is unthinkable, indeed, to propose a global solution to all these challenges. The critical nature of the network security problem, together with the complexity

of the single issues above stated above have suggested, during the last years, to focus on specific solutions to specific problems.

The correctness of user authentication in conjunction with the data privacy and secrecy, usually referred to integrity and confidentiality requirements, are well assured by AAA (Authentication, Authorization, and Accounting) infrastructures [2] and cryptography algorithms [3]. The former category is a framework capable to authentical users, handle authorization requests, and collect accounting data. The AAA system can be considered as the interface between a remote user and the resource it want to access. A distributed system could not operate without a robust AAA server for management of critical information.

Cryptography algorithms define a set of mechanisms and policies to assure privacy and secrecy of data. By encoding the messages that pass through the network, cryptography attempts to hide the real content of data.

Widely used in computer networks, these solutions are well-established, and their general schemes are commonly adopted as de facto standards, as well as new algorithms have been proposed along the years.

On the other hand, the availability of network resources and services of remotely stored data, are not yet completely explored issues, and a standard approach has not been proposed by the scientific community. In particular, the problem of Denial-of-Service attacks has recently attracted the interest of several research groups.

The DoS identifies an attack that prevents legitimate users from accessing a specific network resource by exploiting some weaknesses of the available infrastructure.

The lack of standard solutions is due essentially to the nature of the network, and consequently the difficulty to efficiently detect and fight these threats.

The current network technologies, either wired or wireless, are based on the *end-to-end paradigm* [4]: the network provides just the forwarding data process from the source to destination in a best-effort way, leaving to end

users the problem of ensuring effectiveness and security of the communication. In its fundamental design, the forwarding process just provides fast data transmission, considerably limiting elaborations on them. The largest computation load is located at end users, legating to them also all the issues related to security. If problems occur during an end-to-end communication, no intermediate node can assume control to limit the damages. Furthermore, attacks are usually launched by compromising some nodes in the network. For this reasons any security strategy is unequivocally related to securing each single part of the network, and this is impossible to assure.

For this reason only stable solutions for data privacy and profile authentication are provided, thus neglecting the problems related to security inside the network.Hence, DoS defence solutions are still open issues.

2.2 Denial-of-Service Attacks

Mirkovic et al. provide an interesting taxonomy for DoS, in particular for distributed DoS (DDoS) [4]. Generally, several parameters can be considered for DoS attacks classification. It is possible to discriminate them based on *degree of automation*, *attack rate dynamics* or *source address validity*. The most interesting classification criterion is based on *exploited weaknesses*: it is possible to distinguish DoS in *semantic* and *brute-force* attacks. The former exploit protocols features or application bugs run at victim is side to consume resources and to deny to others their usage. TCP-SYN attack is an example of semantic DoS attacks. The brute-force attack, instead, represents a DoS attack performed by launching a huge amount of requests of service to victim greater then it is able to handle in order to disrupt server's functionality. Some DoSs can fit both classes: for example a TCP-SYN attack could be considered both a semantic and a brute-force attack.

Another interesting classification is based on *victim type*. The *infrastructure attack*, in particular, targets some distributed services that are critical for network operations. An example of these attacks are attacks to DNS, or against network core routers and routing protocols.

A classification about the *impact on the victim* is also provided. We can distinguish between *disruptive* effects and *degrading* effects. A disruptive attack can completely deny access to victim is services to legitimate users. These attacks might require a manual recovery of the victim by human intervention, a self-recovery or semiautomated recovery, or, in the worst case, they might lead to non-recoverable effects. Sometimes the attacks can have marginal effects on victims. The degrading attacks attempt to consume a portion of victim resources in order to limit the number of users that can use them. Detection of these attacks is hard since they don't deny completely the victim services. The economical effects of these attacks also can be more disastrous then disruptive attacks, since low customer satisfaction for a service provided by a company can move their preferences to different provider.

As suggested in [4], unfortunately a rigid classification of DoS cannot be provided and sometimes attacks can fall in different categories or newer ones can require the definition of new classes. A DoS, indeed, identifies all attempts to deny a service to users by compromising a resource provided by the network. Any denial of service that involves some intermediate entities or networks can be considered a DoS.

In this chapter we will focus exclusively on two types of DoS attacks: *Infrastructure Volume-based DoS* attack and *Routing protocol DoS* attacks. The latter, in particular, will be analyzed for wireless scenarios, where the problem represents a critical challenge. The objective of this thesis is not to provide an exhaustive solution to DoS defence. As stated previously, we want to prove the applicability of cooperative solutions to improve the efficiency of detection. We have chosen these problems since they represent the most debated ones in the scientific community. Based on the same approach we claim that mechanisms and results provided for these two threats can be effectively extended to other specific DoS threats. We will discuss about this at the end of this thesis.

2.3 DoS Defensive Solutions

Any defence mechanism, a part from the specific DoS attack, is influenced from several factors [4]. In particular, the nature of attacks, especially DDoS, makes it essential to design distributed systems to face them. Leaving the responsibility to handle attacks just to the victims can create serious damages. Cooperative solutions can better response to DoS and DDoS. In this way it is possible to block the attack attempts further from the victim, thus reducing the risks. Moreover a distributed attack detection can make it easier to apply a proper response. Unfortunately, since the networks are administered by different entities, a fully distributed solution is not favourably accepted by the entire scientific community.

The mechanisms for DoS defence are usually classified in two classes based on the approach adopted to deal with the attack:

- preventive solutions,
- reactive solutions.

The first ones try to eliminate the potential system weakness exploited by attackers during their actions. Resources redundancy, mechanisms for resiliency, secure code and protocol design, users accounting tools, are common approaches adopted to prevent a DoS attack. Unfortunately this approaches cannot assure the complete security of the network. A security strategy that exclusively applies prevention policies can effectively operate only if all its mechanisms work correctly; otherwise an attacker can exploit a fault in one of its parts to bypass the active security strategy. For this reason both detection and responsive solutions are essential.

Reactive solutions have the responsibility to mitigate the effects of one attack in progress. They require firstly to detect the attack and then to respond to it. From a complexity point of view, the main challenge of a reactive mechanism is the attack detection. The well-known solutions to this problem are based on the adoption of Intrusion Detection Systems (IDS).

Generally an IDS provides two fundamental tasks: (i) monitoring the traffic data in order to discriminate the activities on the network; (ii) classifying the monitored activity as normal or malicious.

The IDS is the critical component of a reactive strategy against DoS attacks, so it has attracted a great interest from several researchers in the last years. IDS systems have been largely studied and numerous solutions have been proposed.

In the next section we will provide details on current solution for IDS. Cooperative systems also will be dealt with.

2.4 Intrusion Detection Systems

The network monitoring and traffic inspection are critical tasks for a intrusion detection system. Generally an IDS provides network traffic monitoring in order to identify inappropriate, incorrect or anomalous activity within a system, be it a single host or a whole network. As like a "bouncer", IDS observes the traffic passing through the network, discriminates among different users by analyzing properly the different packets flows, and finally classifies them in normal or malicious based on well-known classification criteria. All this must be done in a transparent fashion, without interfering with legitimate user activities.

Several IDS classification criteria have been proposed. Usually an IDS can be grouped into three main categories:

- Network-based Intrusion Detection Systems (N-IDS),
- Host-based Intrusion Detection Systems (H-IDS),
- Stack-based Intrusion Detection Systems (S-IDS).

This classification depends on the information sources analyzed to detect an intrusive activity. N-IDS [5, 6] analyze packets captured directly from the network. By setting network cards in promiscuous mode, an IDS can monitor

traffic in order to protect all of the hosts connected to a specified network segment. On the other hand, H-IDS [7, 8] focus on a single host's activity: the system protects such a host by directly analyzing the audit trails or system logs produced by the host's operating system. Finally, S-IDS [9] are hybrid systems, which operate similarly to a N-IDS, but only analyze packets concerning a single host of the network. They monitor both inbound and outbound traffic, following each packet all the way up the TCP/IP protocol stack, thus allowing the IDS to pull the packet out of the stack even before any application or the operating systems processes it. The load each IDS must afford is lower than the total traffic on the network, thus keeping the analysis overhead into reasonable bounds; hypothetically, each host on the network might run a S-IDS.

Intrusion Detection Systems can be roughly classified as belonging to two main groups as well, depending on the detection technique employed:

- anomaly detection,
- misuse detection,
- signature-based detection.

The first two techniques rely on the existence of a reliable characterization of what is *normal* and what is not, in a particular networking scenario.

More precisely, anomaly detection techniques base their evaluations on a model of what is normal, and classify as anomalous all the events that fall outside such a model. Indeed, if an anomalous behavior is recognized, this does not necessarily imply that an attack activity has occurred: only few anomalies can be actually classified as attempts to compromise the security of the system. Thus, a relatively serious problem exists with anomaly detection techniques which generate a great amount of false alarms. On the other side, the primary advantage of anomaly detection is its intrinsic capability to discover novel attack types. Numerous approaches exist which determine the variation of an observed behavior from a normal one. A first approach is

based on statistical techniques. The detector observes the activity of a subject (e.g. number of open files or TCP state transitions), and creates a profile representing its behavior. Every such profile is a set of "anomaly measures". Statistical techniques can then be used to extract a scalar measure representing the overall anomaly level of the current behavior. The profile measure is thus compared with a threshold value to determine whether the examined behavior is anomalous or not. A second approach, named *predictive pattern generation*, is based on the assumption that an attack is characterized by a specific sequence, i.e. a *pattern*, of events. Hence, if a set of time-based rules describing the temporal evolution of the user's *normal* activity exists, an anomalous behavior is detected in case the observed sequence of events significantly differs from a normal pattern.

Misuse detection, also known as *signature detection*, is performed by classifying as attacks all the events conforming to a model of anomalous behavior. This technique is based on the assumption that an intrusive activity is characterized by a signature, i.e. a well-known pattern. Similarly to anomaly detection, misuse detection can use either statistical techniques or even a neural network approach to predict intrusions.

Signature-based detection is the most used to detect an attack. SNORT[1][10] and Bro[2][5] are well-known systems based on this approach. Intrusions are coded by means of a set of rules: as soon as the examined event matches one of the rules, an attack is detected. A drawback of this approach is that only well-known intrusive activities can be detected, so that the system is vulnerable to novel aggressions; sometimes, few variations in an attack pattern may generate an intrusion that the IDS is not able to detect.

The main problem related to both anomaly and misuse detection techniques resides in the encoded models, which define normal or malicious behaviors. Although some recent open source IDS, such as SNORT or Bro, provide mechanisms to extend the detection ability of the system in order to include either anomaly or misuse approaches[11, 12], behavior models are

[1]http://www.snort.org
[2]http://www.bro-ids.org

usually hand-coded by a security administrator, representing a weakness in the definition of new normal or malicious behaviors. Recently, many research groups have focused their attention on the definition of systems able to automatically build a set of models. Data mining techniques are frequently applied to audit data in order to compute specific behavior models (MADAM ID [13], ADAM [14]).

Data mining algorithm is referred to as the process of extracting specific models from large stored data [15]. Machine learning or pattern recognition processes are usually exploited in order to realize this extraction (SLIPPER[3] [16]). These processes may be considered off-line processes. In fact, all the techniques used to build intrusion detection models need a proper set of audit data. The information must be labelled as either "normal" or "attack" in order to define the suitable behavioral models that represent these two different categories. Such audit data are quite complicated to obtain. The data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, the 1999 KDD data[4] [13][17], is probably the most well-known example of this kind of information, representing a processed version of the DARPA Intrusion Detection Evaluation Program database, collected and managed by the MIT Lincoln Laboratory. The DARPA database contains *tcpdump* data related to seven weeks of network traffic generated over a military emulated LAN. KDD is filled with five million connection records labelled as "normal" or "attack".

2.5 Cooperative Solutions for Attacks Detection

The current "isolated" IDS solutions suffer from two major problems: the high false alarm rate, and the "locality" of detection that limits the capability of the system to properly face distributed and coordinated attacks as DDoS.

New distributed solutions for intrusion detection have been recently pro-

[3]http://www-2.cs.cmu.edu/~wcohen/slipper/
[4]http://kdd.ics.uci.edu/

posed, supported also by the consideration that by correlating alerts it is possible to reduce the false alarm rate [18].

In this section we deal with solutions for network security that have successfully exploited cooperation and distributed information to improve the ability of a system to properly detect attacks to the network. The common feature of these systems is to define architectures that are independent of the specific intrusion detection mechanism, since they just attempt to enlarge the functionality of system and to increase the capability of detection by adopting a coordinated and collaborative approach based on the exchanging of security information. All of them usually adopt existing solutions for local attack detention and try to increase the effectiveness of the overall process based on the assumption that sharing evidences of attacks might provide a clearer vision of the ongoing situation. This can improve the capability of correct detection, through a decrease of both false positives and false negatives, and thus bring to a prompt response.

2.5.1 DefCOM

The Defensive Cooperative Overlay Mesh (DefCOM) system, proposed by the Laboratory for Advanced System Research (LASR) at University of California in Los Angeles (UCLA) [19], is a well-known example of distributed system based on a cooperative approach to face the problem of distributed DoS. Its architecture is shown in Figure 2.1. DefCOM is composed of heterogeneous defensive elements, organized in an overlay peer-to-peer network. We can identify four different defensive elements in the architecture:

- Legacy Router,
- Core Defence Node,
- Classifier Node,
- Alert Generator.

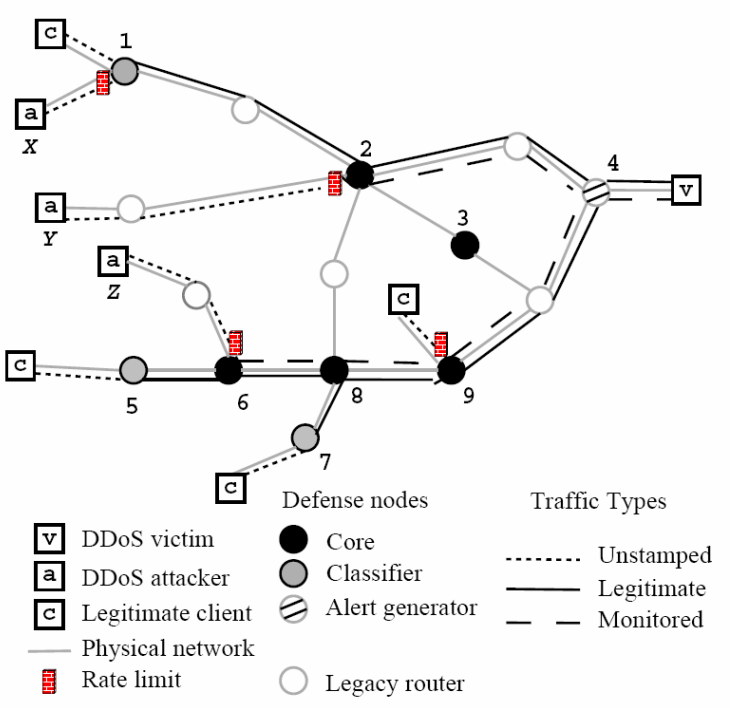


Figure 2.1: DefCOM Architecture

The core of the architecture is the Traffic Tree Discovery by which the framework attempts to identify the *victim-rooted traffic tree*. When a DDoS attack occurs, the alert generator detects it and sends an `alert` message to all the nodes in the network, except those nodes that don't forward traffic to the attack target. All the nodes that observed the traffic directed to the target are called active nodes. The nodes cooperate in order to identify the traffic tree from attackers to victim by stamping the traffic. In particular, each active node selects a stamp to place in the header of packets it forwards to the victim, and communicates it to its neighbors in a secure way. Moreover, an active node observes the stamps selected by its neighbor in order to create a parent-child relationship; a node that observes the traffic stamped by its neighbor becomes a "parent" and so it communicates to the neighbor its new "child status" by explicit message. The security of the stamp exchanging mechanism is assured by changing frequently the stamp and by a mix of encryption and authentication mechanisms.

Once the traffic tree from attackers to victim is identified, the defense elements start to cooperate both to limit the traffic of attackers and to assure correct treatment of the traffic of the legitimate users. The best solution requires to apples rate limiting techniques as close as possible to the attackers in order to reduce the risk to limit legitimate traffic. The rate limit is propagated by the root of the tree to the defense elements closest to attackers. Each node assigns an equal amount of its rate limit to its children and communicates this through a `rate-limit request`. The rate limit assigned by a parent can be modified by a child in order to assure traffic to its legitimate users: this is communicated by child through a `resource request` message to parent.

The traffic statistics of each node are reported to *root of tree* in order to allow it to determine the attack evolution and communicate this to the first defense node that raised the alarm.

In order to provide differentiated services to legitimate and malicious traffic each defense node maintains two stamps, `approved stamp` and `monitored stamp`. A Classifier Node monitors the traffic and marks the legitimate traffic with an `approved stamp` and all the other traffic with `monitored stamp`. If no information about a traffic profile is known, a node could mark this traffic as unknown. The rate-limit algorithm assigns the bandwidth resource firstly to "approved" traffic, then to "monitored" one, and finally to "unknown" traffic.

2.5.2 COSSACK

COSSACK is another distributed solution for automatic detection and response to DDoS, proposed in the context of the YOID project at Information Sciences Institute (ISI) [20].

As shown in Figure 2.2, the basic components of the architecture are the watchdogs. Placed at the egress of each network, they are responsible for scanning network topology in order to identify potential victims in the network. They are also equipped with traditional IDS that are responsible for

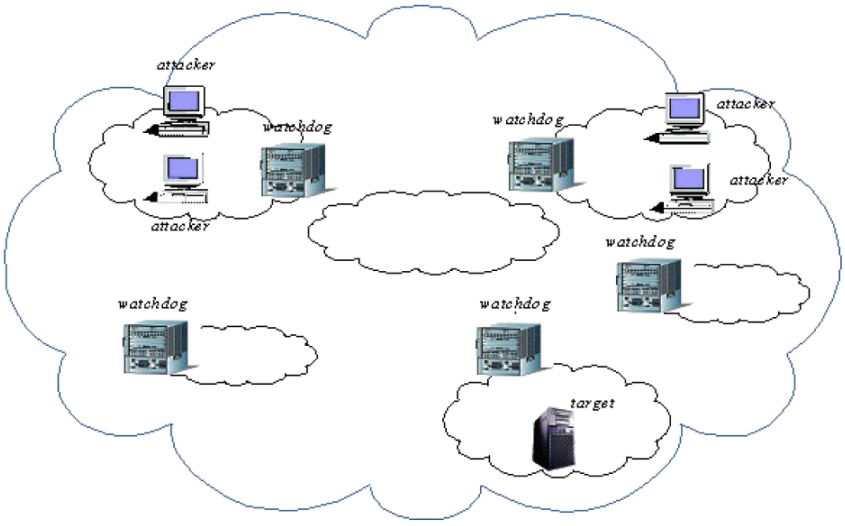


Figure 2.2: COSSACK Architecture

monitoring the traffic and discovering threats to its network. In normal conditions the watchdog is dormant, while the IDS monitors the traffic network. Once an alert is raised by the IDS related to a target host in its network, the watchdog changes its status to "awake" and creates a YOID multi-cast communication group with the others watchdogs of the infrastructure. In this way the watchdog alerts watchdogs close the attack source to adopt the suitable countermeasures to limit the effects of the attack. Once the source of the attack is identified, the watchdogs instruct the related routers to enable filtering on malicious traffic.

In order to reduce risk of source address spoofing, the watchdogs at source networks monitor the outgoing traffic to identify spoofed addresses. Once detected such an address, the watchdog checks for existence of YOID multicast group related to spoofed source's destination. At this point the watchdog joins the group solving the problem of address spoofing.

2.5.3 Cooperative Defence against DDoS Attacks

Nodes cooperation is also the basis for Cooperative Defence against DDoS Attacks (CoDeDDos) architecture [21], a distributed approach for defence

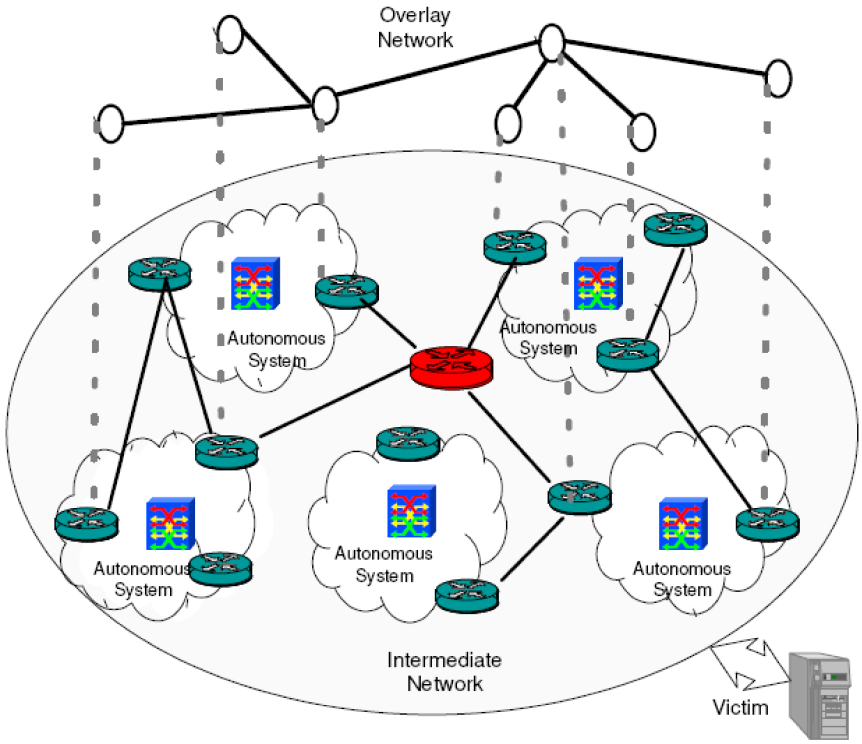


Figure 2.3: CoDeDDoS Architecture

against distributed denial of service proposed by Zhang et al.

This architecture, shown in Figure 2.3, consists essentially of two stages. At the first stage each system node detects the attack autonomously by exploiting a variety of existing IDS solutions, for example Snort [10]. The victim is protected by adopting a local rate limit on traffic directed to it from a suspicious node, according to a local policy. Then, in the second stage each node adjusts dynamically its rate limit according to the information shared with others nodes in the infrastructure. The information exchange exploits gossip communication mechanisms.

In the architecture proposed, each defence element is placed at the edge router of an Autonomous System (AS) and monitors traffic locally to identify DDoS attacks ongoing and in case limits their damages. Single elements are connected by peer-to-peer overlay network; this allows them to exchange information on attacks' evidences.

By adopting a set of metrics M_i each node locally profiles the traffic monitored. Based on these metrics the single element assigns to suspicious traffic a confidence degree $conf$, weighting each metrics with its reliability in terms of false positives or negatives. Thanks to such $conf$ value, the node limits suspicious traffic. Moreover, by using gossip communication mechanisms, it provides his neighbor peers in the overlay network with information on the suspicious traffic profile, the metrics values related to it, and its confidence $conf$.Gossip communication, in particular, provides a "light" and reliable mechanism for sharing information; it exploits solution based on well-known epidemic theory algorithms. The data exchange on DDoS attack evidence allows a more accurate process of traffic limiting by adopting algorithms that aggregate information.

2.5.4 SABER

Proposed by Keromytis et al. [22], SABER (Survivability Architecture: Block, Evade, React) integrates several defensive solutions in order to provide unique coordinated architecture to detect and effectively face different kinds of attacks. Basic idea is that isolated and uncoordinated solutions can fail to react properly to all possible threats. An overview of SABER architecture is shown in Figure 2.4.

SABER consists of different components, that it coordinates and selects properly based on the specific threat that it has to face. The components are:

DoS Resistent Architecture placed at the perimeter of network, rapidly detects whether a service request is legitimate or not, hence limiting the illegal resource usage;

Intrusion Detection tools , placed both inside and outside the network, provide effective solutions to detect ongoing attacks;

Process Migration and Software Patching tools available inside the network, to respectively: (i) a process to automatically suspend itself and

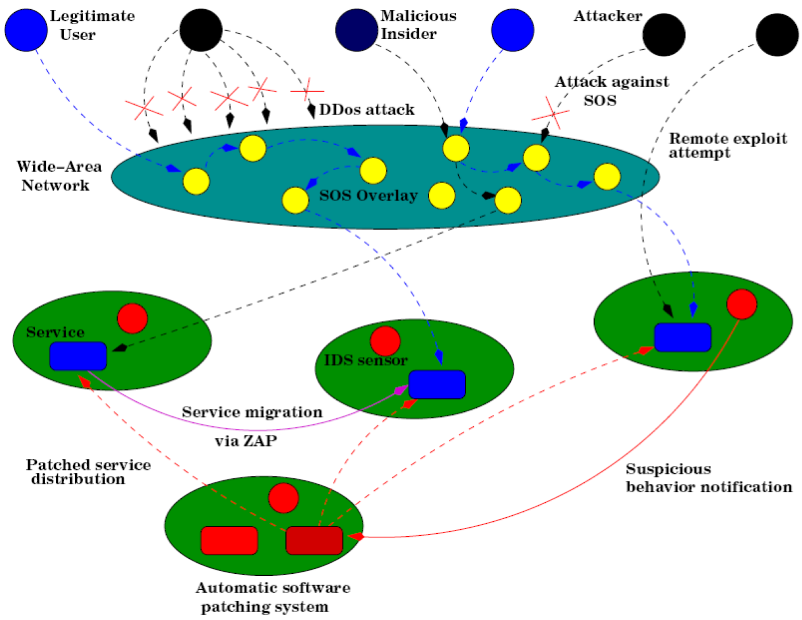


Figure 2.4: SABER Architecture

to move to a more secure node; (ii) to reduce risk of vulnerability code by an automatic application of patches.

Coordination and Control Infrastructure which allows the different components to communicate and correlate themselves in a suitable way,by adopting a distributed publish-subscribe event-based approach.

Each component of the SABER framework is provided with two different mechanisms for communication: an alerts reports and a control acceptance. Information is exchanged through MEET (Multiply Extensible Event Transport) that provides an efficient distributed architecture for the publish-subscribe paradigm. The events coming from different components are collected, managed and aggregated by XUES (XML Universal Event Service).

By using the coordination and control infrastructure, SABER is able to activate the proper components to better respond to an attack attempt.

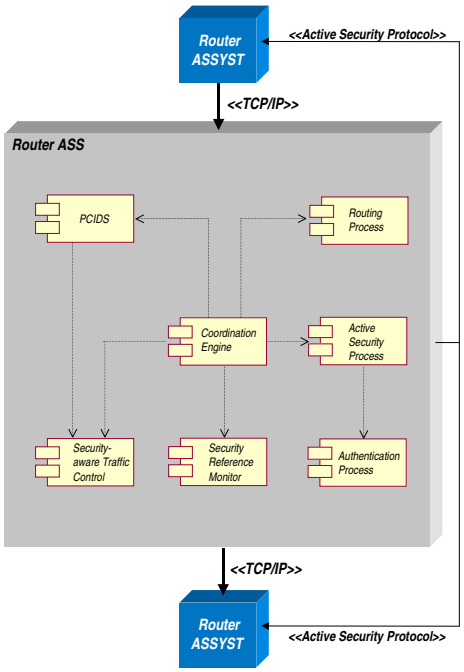


Figure 2.5: ASSYST Router

2.5.5 ASSYST

Proposed by COMICS (COMputer for Interection an CommunicationS) Research Unit at Federico II University of Napoli [23], Active Security SYSTem (ASSYST) is a distributed solution for active network protection against DDoS attacks, involving routers and exploiting a new protocol for information exchange called Active Security Protocol (ASP).

In spite of traditional solutions for attack source traceback, ASSYST avoids for marking packets, but exploits only information exchange among network routers with the objective to go back up to attacker. Indeed, the main ASP purpose is to provide a "light" solution for cooperation of network elements involved in malicious activity.

An ASSYST router consists of the following components, as shown in Figure 2.5:

Packet Classifier Intrusion Detection System which allows the detec-

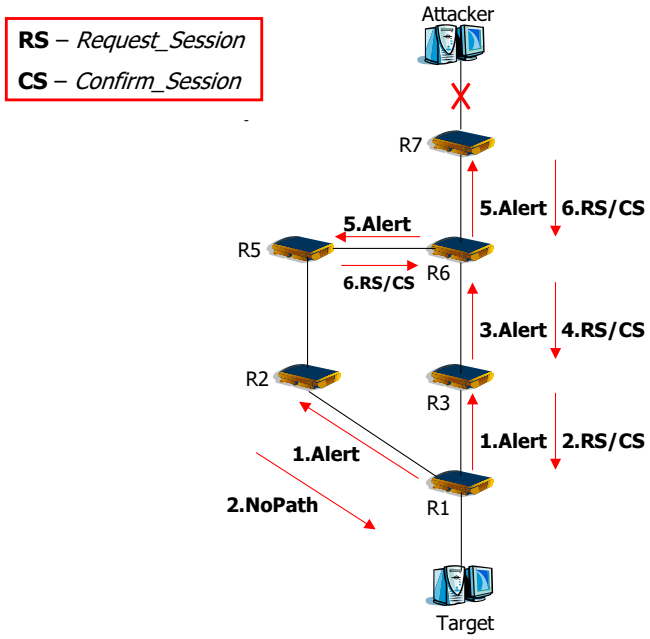


Figure 2.6: ASP Process

tion of an attack in progress;

Security Reference Monitor which manages the information needed by others components cooperating;

Policy Response Process which manages the allocation of queues for attack monitoring based on specific monitoring policy;

Coordination Engine which allows to orchestrate all the network elements involved in the detection process;

Authentication Process needed for securely authenticate the cooperating routers;

Active Security Process for the interaction with ASP;

Routing Process which is responsible for local routing operations.

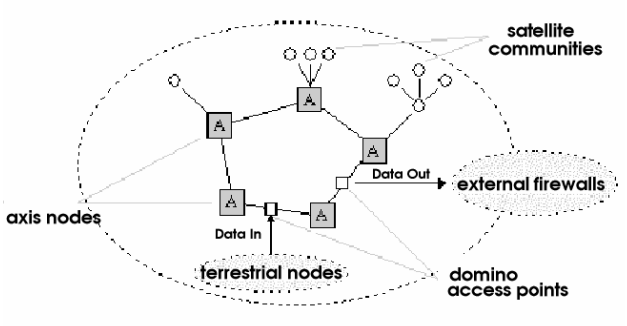


Figure 2.7: DOMINO Architecture

When a router detects suspicious activity it propagates an `Alert` message to all the neighbor nodes with an identifier of attack session (Figure 2.6). Such information is summarized in unique `IDSession` and in a `TrafficDescriptor` which describes the characteristics of the attack.

Upon reception of an `Alert` message, all routers start to monitor the traffic based on the information contained in the `TrafficDescriptor`. If no traffic with such features is detected the router replies to the source with a `NoPath` message, otherwise it sends a `RequestSession`. Upon reception of the `RequestSession` message, the router which started the process first confirms the session and then delegates to upstream routers all monitoring processes, since those routers are closer to the attack source.

Once an attack is over, the ASP protocol provides mechanisms to release all resources allocated for the detection process.

2.5.6 DOMINO

Proposed by the University of Wisconsin [24], DOMINO (Distributed Overlay for Monitoring InterNet Outbreaks) provides a dynamic framework for information sharing aimed at improving the intrusion detection capability of a set of nodes located within the network.

It is possible to identify three different classes of components (Figure 2.7):

Axis Overlay It is the core of the DOMINO architecture. It provides an overlay network for communication and information sharing among the

elements of the architecture. It consists of several parts: the *DOMINO Access Point* for external connectivity, the *Active-Sinks*, which simulate virtual machines and which exploit unused IP address space to collect data on attacks attempts (honey-pots), *NIDS/Firewall* for monitoring activities of a local segment of network, the *DOMINO Query Engine*, which creates an interface for importing and exporting the data about intrusions, as well as for setting parameters, and the *DOMINO Summary Exchange Protocol*, which allows periodically exchange of reports with information about intrusions.

Satellite Communities They are small networks or single nodes implemented as local DOMINO systems. These communities are hierarchally organized and communicate with the outside world through either an Axis Node or a DOMINO satellite. Generally data obtained at satellite level are less trustworthy then data collected at an Axis Node.

Terrestrial Contributers They identify nodes, usually NIDS or Firewall, that don't implement the DOMINO architecture but contribute to the DOMINO system by providing daily reports on intrusions and port scans.

Information sharing allows to aggregate local and global views of the intrusion activities. A *Cooperative Intrusion Detection Module* (CRIM) can be exploied to merge information in order to have a global view of the situation. This process can also be opportunely weighted.

2.5.7 Cooperative Anti-DDoS Entity

Cooperative Anti-DDoS Entity [25] is a modular software that offers to participating domains both a communication and response coordination service. The main objective is to create a community of cooperativing partners that by exchanging security information are able to identify and respond to attacks locally, without any traceback process.

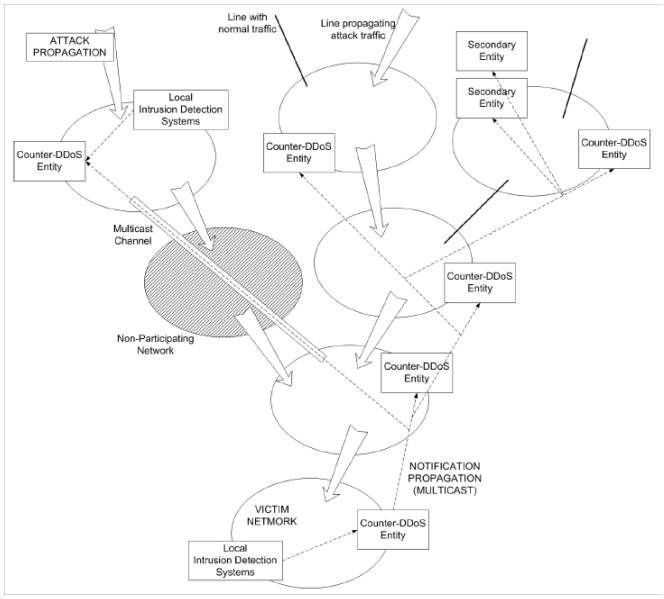


Figure 2.8: Deployment of Cooperative Anti-DDoS Entity

The basic component is the *Entity*, which operates as a high-level coordinator of a local hierarchy of IDSs (Figure 2.8). By exploiting both information of local IDSs and information coming by other peers in the community, the Entity is able to infer about the presence of an ongoing attack. The Entity is also able to interact directly with a local IDS in order to temporarily configure filters for DDoS suppression.

In normal conditions the Entities periodically transmit and receive messages that confirm their "aware status" to the community. When an Entity receives an `Alert` message either from a local IDS or from another peer, it moves to *suspicious* status. As soon as the number of `Alerts` exceeds a threshold the Entity changes it status to *alerted*, and asks a management console for the policy to be adopted related to these alerts. If a security policy is needed, the Entity moves to the final status of *reacting*.

The communications within the community are allowed by means of a multi-cast model, and messages exchanged are formatted according to XML Intrusion Detection Message Exchange Format (IDMEF) [26].

2.6 Security Issues in Wireless Networking Scenario

Cooperation seems to be also the key strategy to prevent and reduce the risk of attacks in a intrinsically weak infrastructure like a wireless network. The lack of physical connectivity, indeed, represents a critical factor that a malicious user can exploit to compromise the functionality of a wireless network. Thanks to the "broadcast" nature of the wireless channel, malicious packets can not be prevented from reaching some of the nodes as opposed to wired networks where filtering policies at network access can easily be applied to reduce the risks of malicious actions. Although authentication processes can be exploited, in wireless network an intruder can more easily obtain physical access. In this scenario some services, like the routing process, become more exposed to attacks, and for this reason well-designed security solutions are required.

As we will see in section 2.7 the cooperation among elements of a wireless infrastructure can prevent or limit some damages provided by malicious activities, thus overcoming the intrinsic weakness of its fundamental functionality. In the following we will discuss the possible risks coming from a DoS in a wireless network, paying more attention to routing attacks.

In wireless scenarios DoS attacks represent much more serious threats than in wired infrastructures. Although authentication, confidentiality and integrity of data still play a fundamental role and can be achieved by adopting traditional security mechanisms, DoS attacks deserve special attention, since they can easily compromise functionality of the infrastructure.

The availability of low cost hardware, open-source software and public access to ISM band (also adopted by IEEE 802.11), have made it easy to launch a DoS attack to a wireless network compared. IEEE 802.11 technology is an appealing target since it provides good connectivity range; Furthermore, an intruder can reach a node undisturbed by using ISM band with a low cost technology.

Obviously the different wireless network technologies determine new threats that exploit them. Although several mechanisms typical of wired scenarios, (like volume-based DoS detection) can still be applied in a wireless context, new solutions are required to properly handle new attacks.

The DoS attack to physical layer is just an example of these new threats. The jamming problem, for example, exploits the peculiarities of wireless link to compromise communication reliability. By jamming the channel with a noise signal it is possible to reduce seriously the bandwidth availability of the channel, thus limiting total throughput of the network. For this reason by interfering with the radio channel these attacks can reduce the performance of the network. The problem dramatically raises if the attack is launched in strategic points of the network. In a wireless mesh network, for example, performing signal interference in proximity of a gateway can irreparably compromise the communications of all the network. Unfortunately no effective solutions have been provided to contrast physical layer attacks.

Differently from wired networks, wireless infrastructures are also particularly vulnerable to MAC layer DoS attacks. As for the physical layer, these attacks can compromise the effectiveness of communication since the sharied nature of the medium can allow an attacker to contend the MAC channel with legitimate users.

Attacks to routing protocol are also a serious risk against correct functionality of wireless networks. As stated by Salem et al. in [27], the multi-hopping nature of the most common wireless infrastructures (Ad-hoc Networks and Wireless Mesh Network) makes routing a very important mechanism. For this reason it is becoming an appealing target of more and more attack attempts. Since the routing protocols are critical operations they require robust and secure solutions. According to Pirzada et al. [28], "there is no advantage in protecting the data if it never reaches its required destination".

There are two sources of threats to routing protocols. The first is due to "external attackers". These threats usually attempt to limit the functionality of routing by injecting incorrect routing information, modifying or destroying

such information, or replaying old routing messages.

The second source of threats are the "internal attackers". They usually consist of compromised nodes that can exchange incorrect information with the other nodes, block the process of forwarding when packets pass through them or modify the forwarded data.

Detailed information about internal attackers is provided by Yih-Chun et al. in [29]. They define a routing *blackhole* as a typical compromised node that, by using forged routing messages, can attract traffic to it in order to drop maliciously data packets. Typically this is performed by announced short distance to destination. Sometimes an attacker can substitute a blackhole with a *grayhole* that selectively drops packets. In particular it can forward just routing packets, while dropping data packets. In this way it joins the path discovery process but precludes nodes from reaching the destination, once the grayhole is on the path selected. An attacker can also intentionally announce a longer distance to a destination node in order to avoid the forwarding data process through it. This attack is known as *gratuitous detour*. In a *wormhole* attack, instead, a compromised node records the data at one location, sends them by tunnelling to another compromised node located in another place, and injects again packets from there in the network. This attack can create serious problems to routing protocols by denying its capability to find paths longer than one or two hops. By quick dissemination of illegal `ROUTE REQUEST` messages which suppress any later legitimate `ROUTE REQUEST`, a *rushing* attack might seriously compromise the protocol routing process.

Fortunately, several solutions have been proposed to protect the routing mechanisms in wireless context. According to the classification proposed in section 2.3, it is possible to identify two typical approaches to the problem:

- prevention,
- detection and reaction.

Both of them require both information exchanging and the cooperation

among nodes of the infrastructure.

Prevention mechanisms usually exploit private-key dissemination and certification authorities to ensure both the correctness and the reliability of the routing information exchanged. Unfortunately, as stated previously, prevention cannot assure the effectiveness of security in the network. Detection and reaction approaches are still needed.

Several cooperative solutions for detection and reaction to routing protocol attacks will be presented in the next section.

2.7 Cooperative Solutions for Secure Routing in Wireless Networks

In this section we describe some solutions to detect and mitigate misbehavior in routing protocols for wireless networks. Their main objective is to provide a robust mechanism to select "secure paths" based on node trustworthiness principles. As stated previously, the common feature of all these solutions is cooperation.

2.7.1 Watchdog and Pathrater

Watchdog and Pathrater are two mechanism to detect and mitigate misbehavior in routing processes [30]. They have been designed to extend the functionality of the well-known DSR (Dynamic Source Routing) [31] routing protocol for mobile networks.

Watchdog is responsible for detecting nodes that don't forward packets. It is implemented at each node by maintaining a buffer with all the packets forwarded recently, and by comparing packets in the buffer with the packets heard by interface in promiscuous mode. If packet captured doesn't match the data in buffer or a packet remains in buffer a time longer than a timeout, a fault event is added to the failure list for that node.

Implemented at each node Pathrater performs a path computation by combining node misbehavior rating with data of the links. The path is se-

lected by adopting a metric that is the average of nodes ratings along the path: the highest metric path is chosen. The ratings are periodically computed based on variation of nodes behavior.

2.7.2 CONFIDANT

CONFIDANT (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks) [32] is an interesting solution for cooperation in an ad-hoc network to reduce risks of attacks to routing protocols. CONFIDANT works as an extension of DSR routing protocol. The solution proposed is based on detection of misbehavior and subsequent reaction to it: upon detection of malicious behavior of a node, the system responds by blocking the forwarding process of packets coming from that node. A sort of *re-integration* possibility, after an "expiation" period, is offered to misbehaving nodes which return to work correctly.

The cooperation is the main contribution of the CONFIDANT protocol. It provides two mechanisms to detect malicious nodes:

- learning correctness of neighbor nodes behaviors from their direct observation;
- sharing information about malicious nodes with other components of the network.

These two mechanisms allow the network to isolate nodes that don't have an "exemplary conduct".

CONFIDANT consists of four components (Figure 2.9):

- the *Monitor*,
- the *Reputation System*,
- the *Path Manager*,
- the *Trust Manager*.

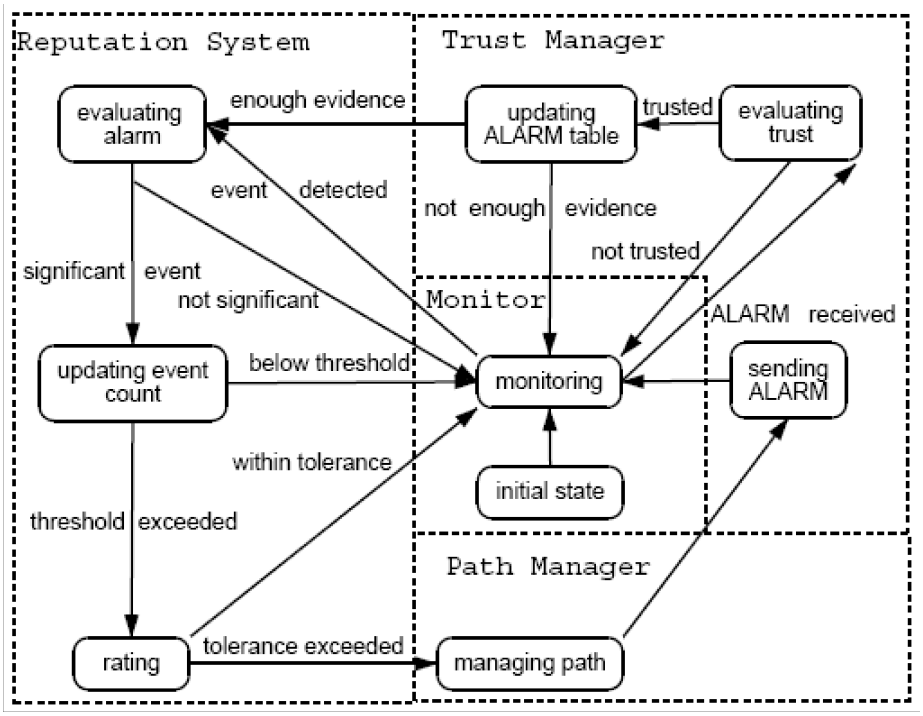


Figure 2.9: CONFIDANT components

The Monitor is responsible to control behavior of neighbor nodes. In a wireless network scenario the monitoring of neighbors is extremely easy to perform. The intrinsic broadcast nature of a wireless link represents an advantage to control the operation of a close node: the traffic sent by a node can be sensed from all the others in signal range. In this way every node controls its neighbor node and vice versa. This feature is fundamental to implement any detection solution in a wireless networking scenario. The CONFIDANT Monitor observes the behavior of neighbor nodes related to a specific activity, for example the forwarding process or the routing messages injection, and registers any deviation from a "normal conduct".

The Trust Manager allows to create a distributed trust management. When a node wants to advice others of a malicious neighbor, it sends an `ALARM`. The receivers have to control the trustworthiness of the `ALARM` received. For this reason the CONFIDANT Trust Manager is provided of several components: an *alarm table*, for storing information about the alarms

received, a *trust table*, containing the trust levels of nodes which are used to access the trustworthiness of the alarm received, and a *friend list*, that provides information about the nodes that potentially can send an alarm, and in which a node trusts.

CONFIDANT Reputation System is a distributed component responsible for management of nodes reputation. It is provided with a table where all the reputation ratings are stored. Anytime node misbehavior evidences are enough, the Reputation System modifies the rating for that node. The update is computed by a function that merges the different evidences of a misbehavior, by assigning different weights to respectively direct experience and to other nodes observations.

The information produced by the Reputation System is used by the Path Manager. The Path Manager is responsible to delete paths that contain malicious nodes. Finally it manages path requests coming from malicious nodes.

2.7.3 CORE

CORE (COllaborative REputation) [33], similar to CONFIDANT (Section 2.7.2), is another solution for improving routing security in wireless scenarios through a distributed reputation model. Adopted as reinforcement of DSR protocol, CORE defines two different kinds of reputation:

- Subjective Reputation
- Indirect Reputation

The former is the reputation observed locally by a node with regard to other nodes. By monitoring temporal evolution of node's behavior, the subjective reputation is computed by giving more relevance to past observations, than to recent ones. This is because isolated misbehaving activities can occur due to link breaks.

The Indirect Reputation is reputation provided to a node from others. It doesn't come from direct observations but rather from observations of other entities.

Subjective Reputation and Indirect Reputation are merged by means of a weighted combining formula to compute a final value of reputation to be adopted in DRS protocol.

The reputation model is provided with a Reputation Table (RT), which contains information about the reputation of each node, and a Watchdog Mechanism (WD).

The reputations in RT are updated only if a misbehavior is detected, as well as only positive reputation value are disseminated during the distribution phase.

Chapter 3

Requirements for a Cooperative System based on Distributed Information

In chapter 2 we presented several solutions for network security that adopt a cooperative approach based on exchanging information. The results confirm that cooperation can considerably improve the performance of the overall system. A security solution can benefit from exchanging information since it can contribute to elaborate a better perception of critical events in progress, and so a suitable reaction policy is possible.

The importance of security application requires cooperative systems that own some fundamental characteristics, by means of which the system can correctly and effectively operate.

In this chapter we will briefly analyze the main requirements for a cooperative security solution. We want to provide useful considerations that can be exploited to correctly design these systems. This analysis allows us to propose a global framework for designing and testing approaches to network security based on distributed information.

Firstly we want to justify from a theoretical point of view the need of a distributed solution to security. Several technical aspects of system components will also be considered.

Based on these assumptions, in the next chapters we will present our solu-

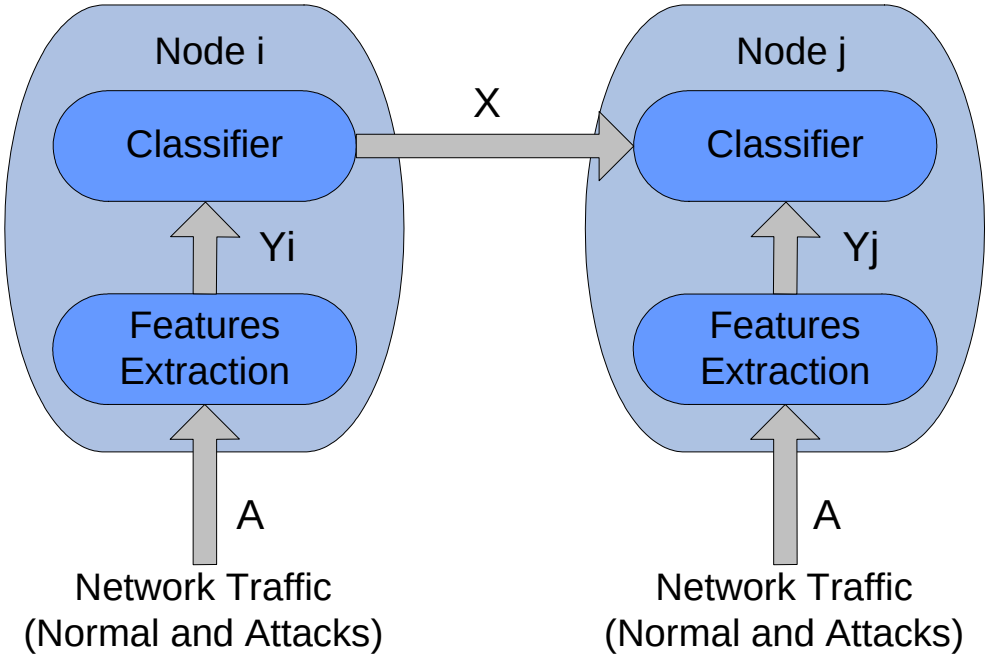


Figure 3.1: Model for Cooperative System

tions for cooperative detection of volume-based DoS attacks and for securing routing protocol.

3.1 Advantages of Cooperation: A Simple Analytical Model

The effectiveness of a cooperative system is well supported by several solutions. Every solution proves the goodness of its design by empirical analysis, but it doesn't always provide a theoretical analysis to justify the choices adopted.

Based on a simple idea of cooperative systems, in this section we want to justify from a theoretical point of view that the performance of such a system improves by adopting a distributed approach to attack detection. We will prove it by using the *Information Theory*. The model can be easily extended to several cooperative solutions for network security.

The proposed system model consists of two nodes, i and j, with an iden-

tical scheme to detect anomalies in network traffic. Each node provides a first elaboration to summarize some features of the monitored traffic, and a classification process to identify anomalies in the data. No assumptions have been made on the features extraction process and classification in order to provide a model as general as possible.

Let us consider an attack event as a random variable A, which takes value 1 if an attack is in progress, 0 otherwise. The discrete probability distribution of the attack is

$$p(a) = \begin{cases} 1-p & \text{if } a = 0, \\ p & \text{if } a = 1. \end{cases} \tag{3.1}$$

Based on Information Theory, the a priori uncertainty on the presence of the attack is given by the entropy of the variable A

$$H(A) = -p * log(p) - (1-p) * log(1-p) \tag{3.2}$$

Given a probability distribution, the entropy defines how much is the uncertainty on variable classification: if $p = 1$ or $p = 0$ the uncertainty is 0.

By considering a new random variable Y_i, an elaboration of the data traffic provided by the features extraction process, as input of the classifier at node i, it is possible to observe a decrease in the uncertainty on the variable A. Generally, indeed, we can state that:

$$H(A|Y_i) \leq H(A) \tag{3.3}$$

This proves that the classification process might be improved by an elaboration on the data independently from the classifier adopted. The equality only holds if the variables A and Y_i are independent. This just means that if Y_i does not carry information about the variable A then it cannot reduce the uncertainty on A, otherwise it gives us more information to reduce the uncertainty.

Let us now focus on the cooperative elaboration of data. Let Y_j be the traffic data extracted at node j, and X a random variable representing an estimate of the variable A provided by node i. Generally, we can state that

$$H(A|Y_j, X) \leq H(A|Y_j) \tag{3.4}$$

with equality only if A and X are independent given Y_j, i.e. if

$$Pr(A = a|X = x, Y_j = y) \leq Pr(A = a|Y_j = y) \tag{3.5}$$

Based on these results, we claim that if node j also receives as input to the classification process an estimate X of the attack event A provided by node i, we might further reduce the uncertainty on event A, unless the information carried by X about A is already contained in Y_j. In such a case, we don't have a decrease in the uncertainty on A, but anyway the probability of bad classification does not increase.

Generally, the conditional entropy $H(A|Z)$ is related to the error of estimate

$$\hat{A} = g(Z) \tag{3.6}$$

By adopting Fano's inequality [34] for the binary random variable A, indeed, we can state that:

$$Pr(error) = Pr(\hat{A} \neq A) \leq H(A|Z) \tag{3.7}$$

We have thus proved that, by reducing the error probability in the classification of malicious activities, a cooperative system might improve its performance with respect to a centralized approach. At least, the performance doesn't degrade when the information exchange is related to different events.

By exploiting an estimate of A provided by node i, which represents the result of a local elaboration, at node j we conclude that the uncertainty on the attack variable A at j decreases, improving in this way its capability of events detection.

3.2 Exchanged Data

The results about the opportunity to exploit a cooperative approach for network security introduce the problem to determine what kind of information nodes need to exchange in order to improve considerably the network security operations. As stated in Section 3.1, indeed, a notable result for distributed detection can only be observed if the data exchanged are statistical dependent. A statistical independence of data, as shown by equation 3.5, does not add useful information in order to reduce the uncertainty on the classification process of the variable A. This might happen if X provides an estimate of a variable A^* completely independent from variable A. In distributed detection, for example, a node i might provide node j with information related to observations of other traffic patterns that node j itself observes.

This raises two problems related to the effectiveness of a distributed system: what kind of information we need to share, and which partners a node must send information to. In the following we will discuss about the first issue, providing an analysis of node partners selection in section 3.3.

Since an attack uses the network to perform its action, it is possible to observe its so called "footprints" in several points through the network. Furthermore, a distributed attack leaves more pronounced"footprints", even if its "form" could change from a location to another, since an attack could aggregate its multiple evidences in some network points. Fortunately, this can only contribute to enforce the certainty of the attack.

Apart from the type of attack, it can be perceived as a "network phenomenon". The evidences of it, its location and its variation are not random features, but they meet a well-known logic summarized in the following two fundamental characteristics:

- spatial proximity,
- temporal proximity.

If an observer senses a phenomenon, it is very likely its spatially close

observer senses the same event too. We define spatial proximity this characteristic. Furthermore, if an observer detects a phenomenon at time t, it is very likely it is going to observe the same event after a fraction of time, for example at time $t + \delta$. This peculiarity of network attacks is called temporal proximity of attack events.

The proximity features reflect in temporal and spatial correlation of the observations of a phenomenon provided by multiple sources. In particular, we can summarize these as follows:

Spatial Correlation Spatially proximal node observations are highly correlated. The degree of correlation increases with internode spatial separation, for example hops distance.

Temporal Correlation The degree of correlation between consecutive node observations may change according to temporal variation characteristics of phenomenon.

Proximity peculiarities could be considered for selecting what kind of information to share. Spatial and Temporal correlations assure that the data exchanged are not independent. By adopting appropriate merging algorithms it is possible to exploit these data peculiarities to reduce the uncertainty of the classification process and to improve global performance of system.

3.3 Data Dissemination Process

Data dissemination concerns the problem to exchange information among nodes. Clearly this is a fundamental issue since the cooperation is based exclusively on the node capability to share data.

The design of a suitable protocol for dissemination must satisfy several requirements, directly connected to the specific security application that we want to realize, and the performance that this application needs to achieve. The mechanism proposed might be the result of a trade-off among these conflicting requirements.

Generally it is possible to identify four fundamental characteristics as follows:

- peer selection;
- time of reporting;
- interference with normal network operations;
- speed of dissemination.

A dissemination protocol requires to properly select a set of peers which a node sends information to. This process is closely related to the security application served. Not all partners of network could be interested in receiving data from a remote location. Sometimes only a limited number of them are really involved in the cooperation process.

The principle of proximity seems a good solution to be effectively exploited for defining the best solution to adopt. By using the spatial proximity, for example, the right set of nodes to involve in dissemination process can be selected.

Several solutions have been proposed. Peer-to-peer communication paradigm is usually exploited. This might be the best solution if the security application knows exactly which peer to send data to. Otherwise a broadcast solution could be implemented. This approach maximizes the number of peers involved in the dissemination process, and it could be adopted when a specific security application requires an event that is known by all partners of community. Intermediate solutions can also be realized. Multi-cast communication paradigm is widely exploited for existing cooperative approaches. This implies that a node must have the capability to identify partners that could be involved in a cooperation process. Furthermore, this process must be dynamic and based on specific events that require cooperation. An alternative solution could be to select partners by using a probabilistic approach. Each node selects a set of nodes to which to send data based on the probability that those nodes are actually involved in the attack. This solution

seems to be the approach when a node does not know exactly the peers that could be involved.

Another basic parameter that characterizes dissemination protocols is the time of reporting. Also this parameter is closely related to the security application that we want to implement, and it could considerably influence the protocol design. Reporting time concerns the time at which the dissemination protocol is exploited to share the information. Clearly the reporting time problem is closely connected to the problem of data synchronization among peers.

Some applications could require to send information periodically at regular interval T. Usually these applications do not have real-time constraints on cooperation, so a periodic reporting is enough for assuring the functionality of cooperation.

Sometimes an event-driven reporting solution might be needed. Some cooperative approaches, indeed, require to respond promptly to critical situations. The cooperation request could be closely connected to the evolution of threats, and for this reason pseudo real-time solutions must be adopted. Every time an attack event evolves a cooperation request could be raised. Similarly, event-driven reporting sometimes could be triggered by application events themselves, instead of threat events.

Finally, some security applications have a strictly real-time reporting constraint. The evolution of monitoring parameters related to a malicious activity, for example, could be continuously communicated to partners in the effort to cooperate for a distributed attack detection.

Clearly this is just a general classification of possible approaches for reporting time in cooperative security systems. Several applications exploit hybrid solutions based on their specific requirements.

Connected with the reporting time is the problem of interference of dissemination protocol with normal network operations. Traffic overhead is introduced by the protocol and it increases with the frequency of data reporting. The information we want to disseminate and its size in terms of

bytes could affect the interference degree of the protocol. For this reason the protocol design and implementation face a trade-off between the frequency and information exchanged, which directly influences the effectiveness of cooperative actions, and the interference with normal network operations. Since security applications are critical for network dependability, sometimes it might be better to introduce a higher overhead in order to have excellent performance of the cooperative system.

The problem of overhead could be partially solved by using existing protocols for spreading information to partners. This requires the dissemination process to be driven by an existing protocol itself. Fortunately, sometimes this is possible, in particular when a cooperative approach extends security functionalities of existing network mechanisms.

The last parameter that characterizes the data exchanging process is the speed of dissemination. Connected with the time of reporting, this parameter defines how long the information takes to reach all partners of community. Sometimes the spatial proximity principle is not enough to create a proper protocol. The application could require that information, after an elaboration by each node traversed, reaches the widest number of partners. This happens usually when an existing protocol is adopted or a virtual chain of peer is involved for security application.

3.4 Data Aggregation Process

Another important issue for the correct design of a cooperative system is the mechanism for data aggregation. Clearly, this process depends on the specific data that the system exchanges, and the specific application that the distributed framework implements.

The elaboration degree of data coming from partners can change based on applications. Some of them, indeed, use data coming from partners only as an input for triggering operations or applying local policies. No complex data elaborations are required, and so no aggregation of information is effectively performed.

On the other hand, sometimes to reinforce locally evidences applications perform an elaboration of the data received. Algorithms that temporally correlate local data with external data are examples of this process. Usually this aggregation process exploits statistical elaboration which is referred to as *Information Fusion*. Information fusion concerns the techniques for merging data from multiple sources despite differing conceptual or contextual representations.

The objective of aggregation is to have a more accurate vision of the reality occurring in the network, in order to improve the detection and reaction to specific anomalies. Statistical correlation or data mining process and multi-classifiers solutions seem to meet this requirement.

Any data aggregation approach requires suitable performance evaluation criteria. Since aggregation in our context has the main objective to improve the security of network, some typical performance parameters of it or classification processes could be adopted to evaluate the data fusion process. False positive rate or false negative rate are analyzed to verify the goodness of aggregation process, and in general improvements of adopting a cooperative solution.

Other performance parameters could be exploited related to the network functionalities themselves, in spite of cooperative applications. Since some attacks attempt to reduce effectiveness of infrastructure, network performance parameters could be evaluated. The service availability time or number of users that at the same time require a service are well-known parameters to exploit to verify the solution performance.

3.5 Cooperation, Reliability and Trustworthiness

At this point it is clear that cooperation can definitely assure improvements for network security, but, at the same time, it could introduce some risks, since when several partners participate for a common purpose the probability

of misinformation among them is very high. The main risks come from the following observations:

- heterogeneity of components cooperating,
- local information more reliable than remote one,
- heterogeneity of events perceived by partners,
- risk of misbehavior of partners.

Usually cooperative systems exploit different components performing heterogeneous techniques in order to increase the effectiveness of action. Although this can certainly yield benefits, it could introduce some risks. By adopting different detection techniques for IDS, for example, cooperative detection process could be compromised. An anomaly detection technique performs zero-day attack identification better then a misuse detection, which could ignore them. A solution able to limit the opposing effect of these two techniques is needed to increase the effectiveness of the system.

Whenever two partners cooperate, the information exchanged might be weighted in different ways. Usually the local information is more reliable that the remote one, since a node is sure for its information generated locally, but it could have some "doubts" about that from partner. It comes directly from the human behavior, where the one's own opinions are more important than the opinion of the others. This is true also in cooperating systems. In statistical aggregation processes, for example, the local information might be weighted more then the remote one, in order to avoid generation of misbehavior analysis.

In a cooperative system events observed by partners could be contrasting, since they are placed widely through the network. Each event is strictly correlated to the local context where it happens so judgments could be influenced by this peculiarity. For limiting the effects of information coming from location "marginal" with respect to the event, a solution that takes into account the locality of information by weighting suitably them is needed.

Finally for the perfect cooperative system we need the assumption of "loyalty" of all the partners. This is true for a "perfect world", not for the real networks, where the threats and risks increase more and more. The trustworthiness is a fundamental component of a cooperation, in particular for a critical application as network security. Some partners could be subverting themselves, and this could seriously compromise the performance of the overall system. A cooperative security solution is weak to malicious node misbehaver so a model limiting the effects of this nodes is required. The solutions commonly adopted are based on partners trustworthy and reputation.

Chapter 4

A Framework for Intrusion Detection

Any cooperative approach to network security requires several mechanisms in order to operate correctly. By analyzing all solutions presented in chapter 2, it is possible to identify four essential components (Figure 4.1). First, *local agents* are needed, which locally analyze network events by monitoring and classifying data. Existing solutions and mechanisms as IDS are usually exploited and integrated in the cooperative framework. Then, a *dissemination protocol* that allows agents to exchange information about their local situation is also a fundamental "ingredient". In order to exploit data coming from different sources, a cooperative system requires mechanisms for *data aggregation*. Finally common reactive policies are designed for limiting damages due to attacks.

Keeping in mind the necessity of these "essential ingredients", as announced in chapter 1, we want to provide our original contributions to network security by proposing a cooperative approach to secure detection and reaction to DoS, in particular volume-based and routing protocol attacks. The peculiarities of these two attacks and the different context in which they operate suggested us to deal with them separately, even if the some conceptual principles discussed in the previous chapter 3 are common to both solutions. In the following two chapters we will deal with the problem of volume-based DoS, by proposing our original solution; a cooperative system

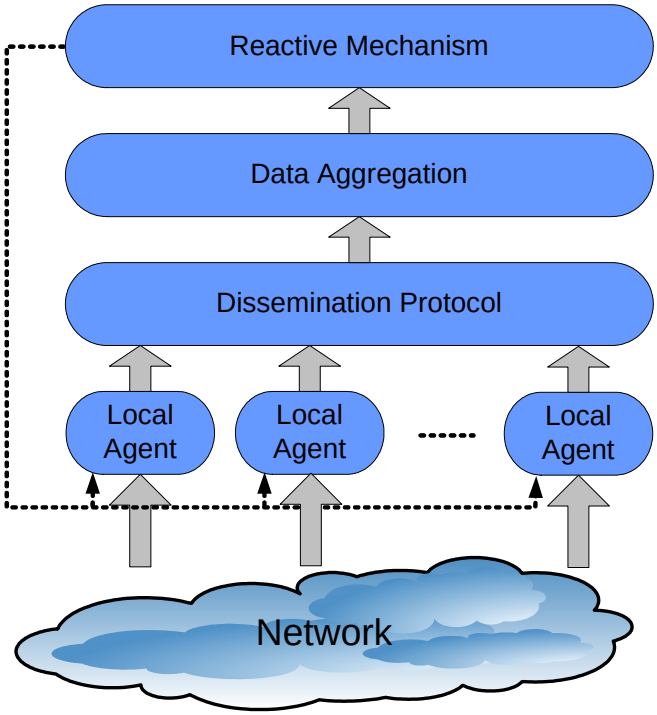


Figure 4.1: A General Framework for Cooperative Security System

for routing protocol attacks detection in wireless scenarios will be in turn dealt with in chapter 6. Specifically in this chapter we will focus on our proposal for a framework for IDS based on traffic monitoring and classification of events by means of data mining techniques.

4.1 An Approach to Intrusion Detection

The work of an IDS consists in analyzing and classifying raw network traffic. Based on its analysis and classification processes, the system can be ascribed to one of the classes of IDS described in section 2.4.

Despite the inherent differences among IDS classes, some common building blocks can be identified, with respect to the high level functionality needed for fulfilling the task of detecting intrusions. Such components, depicted in Figure 4.2, are:

Monitor: a component is needed which can read data and convert them to

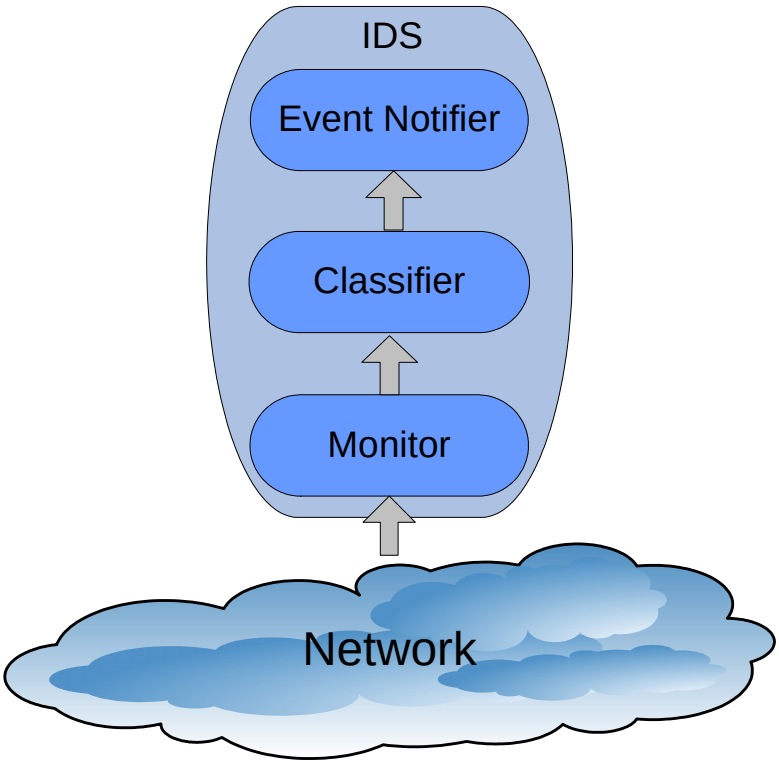


Figure 4.2: Canonical IDS structure

a format which is compatible with the one required from the classifier. The conversion into such a format sometimes involves the extraction of some parameters of interest aimed at synthesizing the properties of the data which are of greater interest for the problem at hand. In the case of the proposed intrusion detection system, network packets are usually decoded, all the header fields are evaluated, and a set of traffic features are computed, related to some statistical properties of the traffic.

Classifier: once data are modeled into a common format, they need to be analyzed and classified. In principle, the classifier component of such an IDS might be independent of the type of data. It needs to be aware of a set of criteria aimed at detecting some particular properties in the analyzed data and, when at least one out of such criteria is matched, notify an entity about the occurrence of such an event. If each criterium

is associated with the most likely cause which might have generated the event it's related to, the analyzer not only is able to notify in case of the occurrence of some particular events, but it is also able to ascribe such events to a generating cause, thus enabling the classification of each reported event.

Event Notifier: any time the classifier reports the occurrence of some events, it is necessary to enable the whole system to communicate with the external world, in order to allow the notification of such occurrences. The event notifier is in charge of interpreting the results of the analysis and correctly formatting the messages required for communicating with the system users.

The detection process involves a set of classification criteria and relies on the definition of a suitable behavior model. Thus, the IDS requires that a proper representation of the user's behavior is inferred from network data by the monitoring module. This process can be realized through a careful analysis of the network traffic and it is more complicated in a real-time scenario, where great volumes of traffic have to be analyzed as quickly as possible in order to reduce packet losses. Moreover, user's behavior is also needed to define the set of classification criteria used by the classifier to identify anomalous activities.

According to this requirement, it is possible to identify two main challenges in IDS development:

- real-time extraction of user's behavior from network traffic;
- definition of a suitable behavioral model to be used in the detection process.

Commonly used IDSs typically analyze packets captured from the network, finding in the current packet the signature of an attack in-progress. However, malicious activity cannot be detected by examining just a single packet: some types of attacks generate in a certain time interval a great

amount of packets belonging to different sessions. Hence, an efficient detection needs statistical parameters taking into account the temporal relation between sessions. These parameters collectively describe the user's behavior. Therefore, in a real-time process we have to associate a set of parameters, i.e. a "pattern", with each packet. Patterns include information related to the single packet, as well as to the connection which the packet belongs to, and to all the connections sharing some properties with the current one. This is needed in order to retrieve information about "network context", which is a fundamental ingredient needed to correctly infer the behavior of a single user.

As it is clear, the user behavior extraction process involves a heavy computation load; moreover, suitable data structures have to be adopted in order to compute the needed parameters. With regard to the behavioral model used to classify users with respect to potential malicious activities, two different approaches can be embraced. The former can be defined as a "punctual classification" approach. According to this criterium, every single model is able to classify a specific user behavior: for example, every network attack is codified by means of a "signature". Usually, every attack signature is represented by a rule (rule-based systems). Unfortunately, the definition of a rule for every attack is not an effective solution. On one hand, this approach is not able to detect novel attack patterns; on the other hand, the definition of new attacks has a negative impact both on the computation load and on the average time required to analyze every single packet (hence, the related packet loss problem). Recently, in order to overcome the above mentioned drawbacks, a new "non-punctual" approach to intrusion detection has been adopted, based on the concept of "behavior generalization": each classification model generalizes the fundamental properties of a set of potential user behaviors. For this reason, every model used in the classification process is able to identify a set of attacks sharing common properties. This generalization approach involves methodologies belonging to the *data mining* research area. When applied to Intrusion Detection, data mining algorithms can be

exploited to encode models which define normal or malicious behaviors.

In order to meet the above requirements, we have developed a framework for intrusion detection exploiting pattern recognition techniques so that novel attacks can be identified [11]. The reference framework is depicted in Figure 4.3.

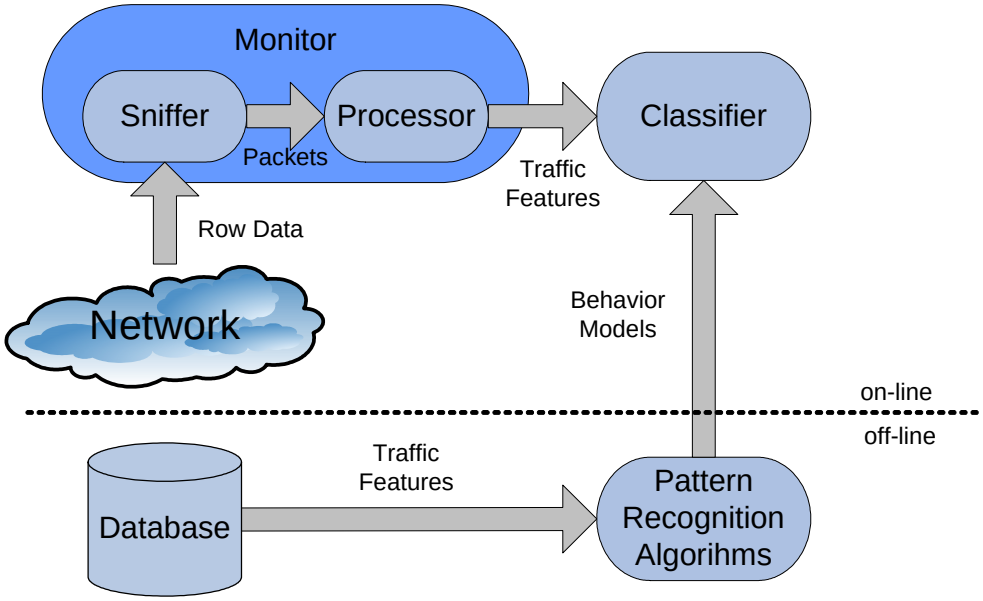


Figure 4.3: A Framework for Intrusion Detection

The overall model is composed of two parts: the former is a real-time intrusion detection system which monitors and classifies network traffic based on well-known user behavioral models; the latter is a pattern recognition process, which extracts such behavioral models from pre-elaborated network traffic, and consists of a database of labeled network traffic features, together with a set of pattern recognition algorithms.

In particular we execute off-line elaborations on a suitably chosen data set in order to extract a set of behavioral rules; such a set is then used in the real-time classification process realized by the IDS.

4.2 Traffic Monitoring: a Behavioral Network Engineering Problem

The on-line system presents some operational blocks which perform the functionality described for a canonical IDS architecture. The lowest block is the monitor module. Connected directly to the network infrastructure, the monitor firstly performs the sniffing task, capturing and decoding in a human-readable data all the packet on the wire. Then, it elaborates the captured packets in order to extract higher level information; such information, commonly called *connection features*, is needed to improve the behavioral classification process. The connection features represent a summarization of the network user behavior. The greater the capability of the set of features to discriminate among different categories, the better the classifier.

Usually there are three levels at which feature sets may be defined:

- The features may be referred to the single packet captured from the network: although this set is easy to compute, it is not able to identify all the potential attack types.
- A set of features related to the entire session which the packet belongs to may also be defined: this is due to the fact that some intrusions may be realized by means of a sequence of packets belonging to either the same connection or different connections.
- The computed set of features may perform a statistical analysis of the relation between the current session and the other ones: this is needed in order to capture intrusions which affect the interrelation among different sessions.

The data aggregation process is probably the main task for the monitoring module. Extracting information of users' habits can clearly influence the classification stage. Thus, modelling correctly the network activities becomes a critical issue for an IDS. In order to design a proper solution for

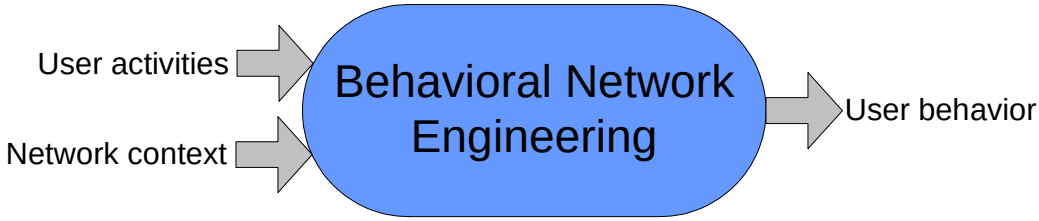


Figure 4.4: Behavioral Network Engineering Approach

monitoring we identify some principles for users' behavior extraction that we called Behavioral Network Engineering [35]. Although we adopted it in an intrusion detection context, such principles can be easily extended to all applications that requires to discriminate users' behaviors for their operations. In the following we will provide some details about such approach.

Firstly we observe the strict correlation between a user's behavior and network resource usage anytime the user asks for a service. Thus it becomes of fundamental importance to get information on users habits in order to optimize all network operations. Generally, Behavioral Network Engineering aims at exploiting data about user's behavior to effectively manage, secure and dimension the network. The knowledge of user's habits represents a valuable input for the Behavioral Network Engineering black-box, which also uses information about the current overall network status to define the proper actions to be performed onto the network. Therefore, both situation awareness and user profiling contribute to produce behavioral information needed to make the network secure (Figure 4.4).

The wide deployment of more and more complex services and applications over the Internet imposes a new approach for characterizing all potential threats coming from either accidental events or malicious users [36]. According to the new paradigm named *Autonomic Communication* [37], network users can be grouped into *communities*, each community representing a set of entities sharing some features and interests and cooperating in order to reach a well-defined objective. Clearly an IDS needs to know what all users are doing, in order to effectively face the threats carried out by the malicious ones. Moreover, this information must also take into account the

relationships among the sets of users sharing common features, resources and purposes. Unfortunately, the current real-time intrusion detection systems do not adopt any user characterization including such "inter-user" information. A Distributed Denial of Service (DDoS), for example, is generally realized through coordinating activities involving actors which are widely distributed throughout the network. According to this property, we claim that the detection of such attacks also requires the knowledge of information regarding the "cooperation" among different network entities. Such entities share the same "purpose" (i.e. the compromission of either a service or a single host) and make use of the same resources (e.g. those belonging to the network infrastructure) in order to fulfill their task. For this reason, the set of hosts involved in a DDoS attack can be considered as a "community", whose members all aim to achieve the same result. Thus, the information about this community as a whole can also contribute to gain a deeper knowledge, i.e. the behavioral information, needed to improve the detection process.

Keeping in mind the assumptions on Behavioral Network Engineering principles, we can define the main requirements imposed by intrusion detection to a monitoring framework.

From the implementation standpoint, the tasks required to a monitoring module in the context of intrusion detection are:

- packets classification and data aggregation,
- behavior extraction.

A common way of classifying packets is grouping them in *flows*. Generally, a traffic flow is referred to as "a set of packets passing at a network point during a time interval and having common properties". This is a general and flexible definition allowing even a single packet, or a few packets to be considered as a flow.

User behavior is described by some parameters, the so-called *metrics*, which are measured by analyzing the properties of the observed packets. In order to increase the required flexibility of the entire monitoring system

and, therefore, its capability to support different kinds of security application approaches, metric definition should be a process driven by users through, for example, the use of suitable programming interfaces.

Thus, any monitoring framework should perform the following operations:

- capturing packets from the network,
- associating them with a flow by enabling a customizable flow definition,
- updating data records containing flow-related metrics

Measured data are, then, collected in order to make them available to the classification process.

Traffic flows which are of interest to an IDS can be classified in two main categories: fine-grained flows and coarse-grained flows. Fine-grained flows refer to traffic generated by a single user or a small set of users and are monitored in order to detect specific kinds of attack. On the other hand, coarse-grained flows transport information describing network context and, then, they are analyzed with the aim of identifying widely distributed attacks such as a Distributed Denial of Service (DDoS). Such flow classification inspires the metric definition and computation process in the sense that depending on the attacks to be identified the IDS requires monitoring system to measure specific metrics on a given class of flows.

We introduce two vectors: the vector of fine-grained metrics, named $\overline{M}_f$, and the vector of coarse-grained metrics, named $\overline{M}_c$(see Figure 4.5). For each flow, either the vector $\overline{M}_f$ or the vector $\overline{M}_c$ will be produced depending on the flow granularity. Starting from these vectors, the intrusion detection system extracts the "context", i.e. a synthetic view of the overall network status.

Therefore, according with a Behavioral Network Engineering approach, the implementation of an intrusion detection system implies developing a *context extraction algorithm* whose inputs are the vectors $\overline{M}_f$ and $\overline{M}_c$ and whose output is a new vector, called $\overline{M}_w$. This algorithm uses both living

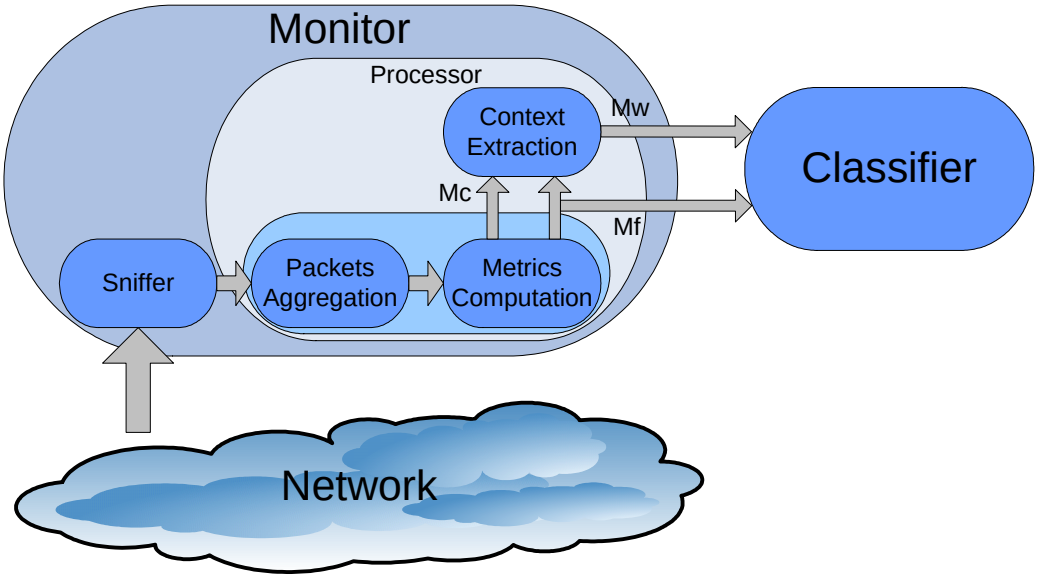


Figure 4.5: Behavioral Extraction for Intrusion Detection

and timed-out behavior user metrics, since both the current state and the recent history of the network concur in determining the context.

Finally, $\overline{M}_w$ and $\overline{M}_f$ are inputs to an intrusion detection algorithm checking whether or not the analyzed flow is an attack. This algorithm is interested in analyzing only living metrics as its task is to trigger a prompt operation on the network. A background classification process involving timed-out flows can be useful to provide forensics as well as to help perform "context" measurements.

In order to better illustrate the proposed approach we provide an example of what $\overline{M}_f$, $\overline{M}_c$ and $\overline{M}_w$ can represent in a particular attack scenario.

Let us suppose we are interested in detecting denial of service attacks against a certain server. We define two kinds of flows: a fine-grained, bidirectional flow, identified by the 4-tuple *(source IP address, source port, destination IP address, destination port)*, and a coarse-grained flow composed of packets having as destination address the server's IP address. Vector $\overline{M}_f$ is computed on the fine-grained flows and reports the number of bytes together with TCP connection status per flow. On the other side, vector $\overline{M}_c$ is

computed on the coarse-grained flows and contains the number of bytes per flow.

Let us notice that the number of bytes received by the server can be also inferred from the sum of all bytes related to fine-grained flows having the server as destination. However, it is more straightforward to evaluate such metric by directly monitoring coarse-grained flows, even though this implies moving computation task in the monitoring system. With reference to our example, a possible context extraction algorithm performs a two-fold task: on one hand, it determines whether server bandwidth utilization is anomalous and, on the other, computes the percentage of flows and the percentages of received bytes per server port number. Such information is reported in vector $\overline{M}_w$. In order to determine anomalous server bandwidth utilization daily statistics are needed. As the usage of a server can vary over the 24 hours of a day, the algorithm compares data contained in vector $\overline{M}_c$ with historical information to detect anomalies in the server bandwidth utilization. Then, vector $\overline{M}_w$, which reports a measure of the level of anomaly in the server bandwidth utilization and the distribution of the port utilization in terms of flows and bytes, describes the overall context and drives the intrusion detection algorithm in identifying malicious flows. More precisely, the intrusion detection algorithm analyzes the context vector and decides whether every single fine-grained flow has to be checked in order to detect specific attacks. For example, a deeper analysis performed by the intrusion detection algorithm might consist in controlling the status of a TCP connection. If the client has sent a *SYN* packet without replying to the server's *SYN/ACK*, then such a behavior might be hiding an attack; therefore, the IDS analysis should trigger an immediate counteraction on the network.

4.3 A Traffic Monitor for IDS

In order to support our framework for traffic monitoring, we implement a system that responds the requirement stated in the previous section [38]. The overall structure is shown in Figure 4.6.

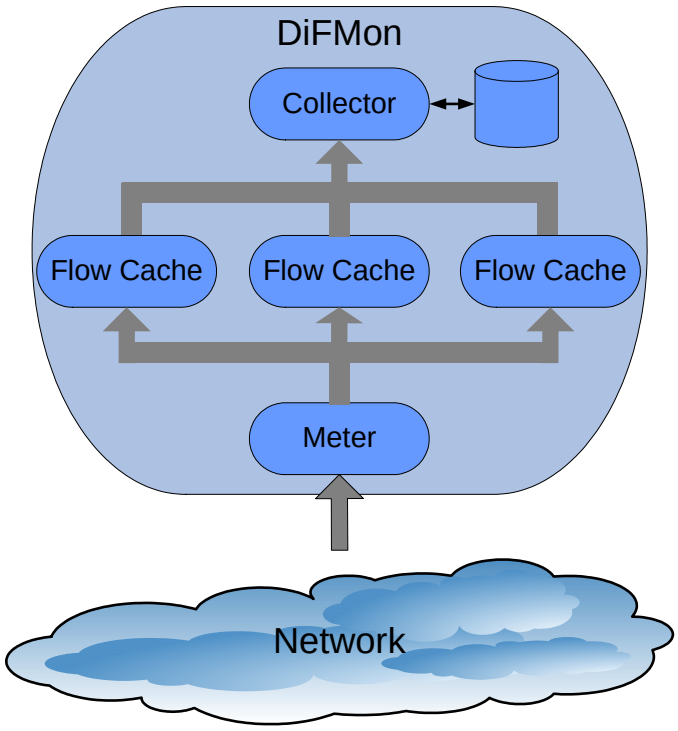


Figure 4.6: DiFMon Architecture

DiFMon (Distributed Flow Monitoring) is in charge of capturing packets from the network, associating them with a flow, by the means of a customizable flow definition, and updating a record containing flow-related metrics. Measured data are, therefore, collected in order to make it available to an IDS classifier. The system architecture comprises a module that stores all the flow records related to *living* flows, where a living flow is a flow which is still receiving packets before a timeout occurs. The main issue concerning this task is represented by the fact that the number of living flows is very high (up to millions) and the packet inter-arrival time is very short on high speed links. This implies that the time interval spent to search for the record associated with a captured packet can be longer than the packet inter arrival time, in case of a huge number of flow records. For this reason we decided to adopt a distributed approach making it possible to divide the task of keeping the records related to the living flows among multiple processes. The proposed architecture is made of the following modules:

1. *Meter*, which captures the packets from a network interface, or equivalently from a trace file, associates them with a flow identification number, the so-called *flow id*, and passes them to the next component.

2. *Flow Cache*, which stores the metrics related to the living flows observed and updates the records. This is the most challenging component as it has to search and update the metrics within the inter arrival time. For this reason we decided to introduce more flow caches and implement each of them as a separate process. Flow records are associated to them on the basis of their flow id. The flow cache is also responsible for *exporting* all the information related to timed-out flows to a further component, i.e. the collector.

3. *Collector*, which collects the metrics related to the flows observed by all the flow caches. The collected data can be used by classifier.

Furthermore, valuable capabilities, such as on-line packet sniffing and filtering, packet analysis based on a trace file, customizable definition of a flow via a scripting language, and customizable definition of a metric via an API are provided.

The software is available on the SourceForge web-site[1].

4.3.1 DiFMon Meter

The Meter performs the following tasks:

1. capturing packets from the network interface or from a trace file;

2. associating a flow id with every packet;

3. providing a time stamp in order to keep track of when the packet was captured;

4. identifying, by means of the flow id, the flow cache which is related to the flow;

[1] `http://difmon.sourceforge.net`

5. sending the packet (together with its flow id and time stamp) to the selected flow cache.

The flow id computation relies on a set of rules defined by the user by means of a standard language. Such language supports the flow definition according to the contents of the packet headers from the IP level to the application level.

The selection of the flow cache is done by means of a hash function applied to the flow id. The chosen function is the "mmh" which is fast enough and has good stochastic properties that enable the uniform distribution of the flow records among the various flow caches.

4.3.2 DiFMon Flow Cache

The flow cache keeps track of living flows in order for metrics to be updated in real time. It receives the captured packet and the related flow id from the meter and, then, establishes whether a corresponding flow record already exists. If so, the flow cache updates the related metrics, otherwise a new flow record is created.

Based on the assumption that the user should be able to define specific metrics, an API is provided. This makes the system very flexible and capable to support different kinds of classification processes.

Moreover, the flow cache periodically sends the data related to the no-more-living flows to the collector. The definition of a living flow can be based either on the introduction of a timeout or on the analysis of TCP sessions.

Since the classification process may require the exporting of some still-living flows. In this case the flow cache selects living flows to be exported through a heuristic function.

The main challenge is the development of a fast and effective flow cache. In particular it is necessary to implement a suitable data structure together with an ordering mechanism to maintain information about living flows. We intend to apply an LRU (Least Recently Used) ordering as it is the main solution used in caching algorithms. This ordering algorithm allows addressing

two issues: it provides a fast way to detect timed out flows as well as a good heuristic to select the so-called *heavy hitters* flows, where a heavy hitter flow is a high rate flow.

In fact, by scanning the LRU queue from the tail and by checking for each record whether the difference between the record's last update time and the current time exceeds the timeout, it is possible to find every timed-out flow. Therefore, if the search for heavy hitters flows is always done by scanning the LRU list from its head, one will step into heavy hitters flows with high probability. The exporting process can take advantage of this ordering mechanism simply exporting the first N records of the flow cache queue.

4.3.3 DiFMon Collector

The collector is the module responsible for collecting the metrics of all the observed flows and sending them to the running classifier. In case it receives flow records related to the no-more-living flows from the flow caches, then such records are both stored into a file and sent to the applications. In case the collector receives living flows, it passes them directly to the classifier performing real time operations. Like the flow cache, the collector provides an API.

4.3.4 DiFMon Management Protocol

In this section we present the protocol for managing interactions among the monitoring system components. The aim is to make the system robust, flexible and tolerant to every kind of faults and errors. To this purpose, when designing such a protocol we made the following assumptions:

1. the system modules run on hosts belonging to a dedicated network separated from the network to be monitored. This is for two reasons: first, the traffic generated by the monitoring system must not affect the behavior of the monitored traffic. Second, the network connecting

system components should be faster than the monitored network as it has to re-transmit every captured packet plus some further information.

2. The modules may run on different machines as well as two or more modules may run on a same machine.

3. The meter is fast enough to perform packet capturing and classification within the mean packet inter arrival time. Both the meter and the collector have to be properly designed and implemented since their behavior largely affects system performance.

4. Both the meter and the collector use well defined port numbers to send or receive signaling messages, while data transfers between system modules happen by using port numbers dynamically chosen.

4.4 Pattern Recognition Techniques for Classifying of Threats

In this section we will deal with the problem of classifying users' behavior provided by the lower monitor module, and the technique to extract models for improve the detection capability of possible threats.

In the framework proposed the classifier represent the core of the IDS. This component analyzes the current data on users' behavior and classifies them. As stated in chapter 2 several are the techniques to detect malicious activities. In our IDS framework we adopted a misuse detection approach since it represents a good trade-off between signature-based solution and anomaly detection in terms of false positives and true negatives.

Based on this approach, the classification process uses a set of rules extracted by means of pattern recognition algorithms in the off-line process of our framework: such rules encode the misuse activities in the network. The metrics' pattern about the users' behavior extracted by monitor are compared against all the rules in the set; when the examined information matches at least one rule, an intrusive action is detected.

Clearly inferring classification criteria by means of pattern recognition algorithms is one of the main issues for identifying dangerous activities in traffic network. In particular the application of these methodologies to IDS context concerns the definition of a proper data set, containing user profiles on which the data mining processes work in order to extract the models. In principle, an efficient set of rules for the detection has to contain all of the possible user behaviors. Moreover, according to all pattern recognition processes, the data set has to properly label the behavior profile items with either "normal" or "attack". Although this might look like an easy task, labelling the data imposes a pre-classification process: you have to know exactly which profile is "normal" and which is not.

In order to solve the issue related to data set building, two main approaches are possible: the former relies on simulating a real-world network scenario, the latter builds the set using actual traffic.

The first approach is usually adopted when applying pattern recognition techniques to intrusion detection. The most well-known dataset is the so-called KDD Cup 1999 Data, which was created for the Third International Knowledge Discovery and Data Mining Tools Competition, held within KDD-99, The Fifth International Conference on Knowledge Discovery and Data Mining[2] that was created by the Lincoln Laboratory at MIT in order to conduct a comparative evaluation of intrusion detection systems, developed under DARPA (Defence Advanced Research Projects Agency) and AFRL (Air Force Research Laboratory) sponsorship[3].

This set was created in order to evaluate the ability of data mining algorithms to build predictive models able to distinguish between a normal behavior and a malicious one. The KDD Cup 1999 Data contains a set of "connection" records, our users' behaviors, descending from those defined by Stolfo, and coming out from the elaboration of raw TCPdump data. Each connection is labelled as either "normal" or "attack".

Although widely employed, some criticisms have been raised against the

[2]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[3]http://www.ll.mit.edu/IST/ideval

1999 KDD Cup Data [39]. Indeed, numerous research works analyze the difficulties arising when trying to reproduce actual network traffic patterns by means of simulation [40]. Actually, the major issue resides in effectively reproducing the behavior of network traffic sources.

Another issue is related to the types of traffic contained in the 1998 DARPA IDEVAL data set. Today, the ever-changing paradigms of networking applications have deeply modified network traffic profiles with respect to 1998. Some researches demonstrate that new traffic types, for example P2P traffic, are replacing the traditional traffic patterns with respect to the data volume generated. Intrusion detection systems depend critically on the accuracy of the traffic used to construct the network behavioral models by means of data mining processes. For example, normal traffic can be misinterpreted as anomalous in case some traffic samples are not present in the training data set.

Based on the considerations above, we can conclude that the KDD Cup 1999 Data can just be used to evaluate the effectiveness of the pattern recognition algorithms under study, rather than in the real application of intrusion detection.

Collecting real traffic can be considered as a viable alternative approach for the construction of the traffic data set [41]. Although it can prove effective in real-time intrusion detection, it still presents some concerns. In particular, collecting the data set by means of real traffic needs a data pre-classification process. In fact, as stated before the pattern recognition process needs a data set in which packets are labelled as either "normal" or "attack". Indeed, no information is available in the real traffic to distinguish normal activities from malicious ones in order to label the data set. So we have a paradox: *we need pre-classified traffic in order to extract the models able to classify the traffic.*

In our framework we adopted a real traffic collection approach for extracting the network behavior models. The data set has been built by collecting real traffic on the local network at Genova National Research Council

(CNR). The *raw traffic* data set contains about one million packets, equivalent to $1GByte$ of data. The network traffic has been captured by means of the TCPdump tool and logged to a file. Such data have been elaborate to properly label them.

After defining the data set, the model extraction requires to manage such data in order to realize the pattern recognition process. Every element in the data set is usually a record of features, our metrics, about a specific user behavior. Indeed, just few features can be used to tell apart normal from anomalous traffic in the analyzed network scenario. In fact, some attacks can be classified only with a small set of behavior features. This can be considered as an advantage: we can reduce the dimensional space of the data set, letting the pattern recognition process become simpler. Common to all the data mining processes, the issue of feature subset selection is known as *feature selection problem*. In our context, we have adopted ToolDiag[4], a pattern recognition toolbox, in order to carry out feature selection [12].

The last step in our work has concerned the extraction of network behavior patterns from the data set. To this purpose, we have adopted the SLIPPER[5] [42] tool. SLIPPER is a rule-learning system exploiting the Boosting technique [43]. On the basis of Cohen's RIPPER program, a widely adopted learning system [13], the SLIPPER algorithm creates a rule set by iteratively boosting a greedy rule-builder. At each iteration, the rule-builder generates an individual rule, by splitting the training data into a *growing set* and a *pruning set*; a single rule gradually "grows" by using the first set. The rule itself is a sequence of conditions on the values of the features. Such conditions are needed in order to correctly classify the items of the data set belonging to the "positive class" (i.e. the class associated with attacks). A final default condition classifies the complementary "negative class" (i.e. normal traffic). The new rule generated overfits the training data; then, the final sequence of conditions is pruned off from the rule, which is finally tested by using the pruning set. If the rule shows a good accuracy, it is added to the rule set.

[4]http://www.inf.ufes.br/~thomas/home/tooldiag.html
[5]http://www-2.cs.cmu.edu/~wcohen/slipper/

Train Error Rate	Test Error Rate	Hypothesis Size	Learning Time
0.20%	0.36%	10 Rules, 37 Conditions	217.33s

Table 4.1: Detection accuracy after feature selection – Average values

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	33.59%	0.06%
2nd Half	**1st Half**	50.41%	0.03%

Table 4.2: Detection accuracy after filtering and feature selection

The SLIPPER algorithm uses boosting to reduce the weight of the data set items covered by the new rule.

4.5 Experimental Results

In this section we will provide some results about the solution proposed. We observe that the main contribution to the effectiveness of the framework comes from the behaviors' models for classification, since they influence the capability of system to discriminate among normal activities and malicious ones. Based on this consideration we will focus on the performance analysis of data mining techniques, in particular on the missed detection rate and, more important, on the false alarm rate, which is a critical requirement for an effective intrusion detection system [44].

Though in other pattern recognition applications a false positive rate below 5% may be a very satisfactory value, in intrusion detection such a rate may not be acceptable. For example, if we imagine to work on a network with a packet rate of 1000000 packets per hour, a false alarm rate of 0.1% would lead to 1000 annoying alert messages sent to the administrator every hour: though characterized by a very low false alarm rate, the number of unjustified alerts would be too high and would lead the administrator to ignore or eventually switch the intrusion detection system off.

We ran different tests on some previously collected data (see section 4.4). First of all, we decided to subsample the data by a factor of 1/10 in order to reduce the computation time of the results; as stated before, we use ToolDiag

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	13.57%	0.16%
2nd Half	**1st Half**	55.32%	0.07%

Table 4.3: Detection accuracy without feature selection

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	13.79%	0.16%
2nd Half	**1st Half**	62.19%	0.05%

Table 4.4: Detection accuracy after filtering without feature selection

for the feature selection step and SLIPPER for the classification. In the first experiment we subsample the data-set by choosing one behavior record out of ten, then we split the subsets in two parts. On each of the half-subset obtained we perform feature selection and, by examining the discriminating power and the number of occurrences over the whole data set of the selected features, we choose an "optimum" set of features. By "optimum" feature, we mean a feature whose ability to discriminate between attacks and normal traffic, within the training data, is the highest with respect to the discriminating power of all the examined features. We consider then, in turn and for each subset, the first half as the training set, and the second half as the test set; then we swap training and test sets, using the second half of each subset as the training set and the first half as the test set. All these experiments are useful to understand which is the best data set we have, as we suppose to have no prior knowledge about the discriminating power of the connection records included in each one of them. In table 4.1 we point out the average values emerging from the analysis of the presented results.

It is worth pointing out that the data we are working on contain some records tagged as *uncertain*. These represent data for which it was impossible to assign for certain neither "normal" nor "attack" class. In this first experiment we decided to label the corresponding packet as normal. We performed a second experiment discarding these uncertain packets. Thus we built and processed a "filtered out" data set, made up by all the records corresponding to packets whose classification was clear enough, obtained by deleting from

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	4%	0%
2nd Half	**1st Half**	0%	0%

Table 4.5: Detection accuracy without feature selection – Trin00 attack

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	0%	0%
2nd Half	**1st Half**	0%	0%

Table 4.6: Detection accuracy after filtering and without feature selection – Trin00 attack

the set the *uncertain* connection records.

Again we proceeded with feature selection and obtained, in the same way as before, the best set of features. On the filtered data we decided to deploy a test by using the whole dataset, with no subsampling. We divided the dataset in two halves and, in Test 1 we considered the first half as the training set, and the second half as the test set; in Test 2, instead, we consider the second half of the data set as the training set and the first half as the test set.

Furthermore, to test the effect of feature selection on the detection capability of the system, we decided not to apply subsampling, and to test the classifier on the datasets before and after the filtering process described above (tables 4.3, 4.4).

We notice a very low false alarm rate, which is good, and a missed detection rate sometimes around 60%. This might seem a not so good result, but it is not; missing an attack packet does not mean to miss the whole attack itself; in fact, an attack pattern may consist of a burst of packets thus, not detecting a few of such packets doesn't mean to lose the attack. Stressing again the false alarm rate problem, we notice that the rate obtained within our experiments is very low, and encouraging for the development of this kind of detection techniques.

In order to strengthen these observations, we also sketch, in Table 4.5 the detection capabilities tested over a particular DoS attack, Trin00, which is always correctly detected by our IDS without rising any false alarm. This confirms the applicability of the realized IDS within the proposed framework

for intrusion detection and reaction.

Finally, as we have a little lower missed detection rate when not using feature selection, we noticed an increase of one order of magnitude in rule calculation time and number of rules. This is due to the fact that we have to strike the balance between detection accuracy, number of adopted criteria and computation time.

4.6 A Distributed Solution

The system described is a centralized solution: all of the components are tightly coupled and are supposed to run on a single node. The higher and higher bandwidth capacity of the current network infrastructure suggests to study robust and flexible solutions that are able to support the great amount of traffic to analyze.

By this assumption, in this last section we want to propose a possible solution for improving flexibility and robustness of the system by appropriately distributing the skills of a centralized solution to several independent components across the network [45].

The solution proposed consists of several entities, whose orchestrated operation concurs in ensuring that IDS works correctly (Figure 4.7).

- *Preprocessor*

  Each such component is in charge of:

  1. capturing raw data from the network;
  2. extracting behavioral information from captured packets;
  3. sending the computed information to a *broker* entity.

- *Broker*

  The broker is a mediation component whose main tasks are:

  1. collecting behavioral information coming from preprocessor;

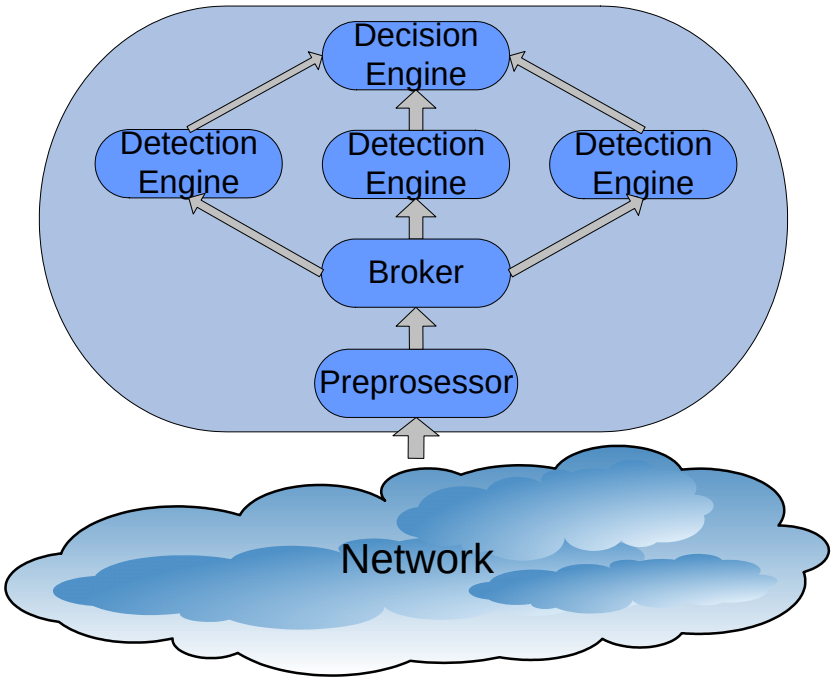


Figure 4.7: A model for a Distributed IDS

2. distributing the collected information to one or more *detection engines*. The distribution strategy is defined by means of a policy-based approach.

- *Detection Engine*

  Detection engines receive behavioral information from the broker and decide on whether or not such information represents a potential attack pattern, based on a specific detection technique.

- *Decision Engine*

  The main task of the decision engine resides in collecting information coming from detection engines and coming up with a final decision concerning the current network situation. This might be done through a number of approaches, ranging from simple majority voting to much more complex solutions in which the various inputs coming from the detection engines are appropriately weighted on the basis of their de-

gree of reliability (with respect to their own capability of detecting a particular set of malicious activities).

Dynamic deployment of the distributed system components is actually needed in order to ensure both flexibility of the architecture and robustness in the face of changes in network and traffic conditions. As an example, the IDS might need several detection engines, each exploiting the best fitting detection technique according to the system status.

Chapter 5

A Distributed Cooperative System for Network Security

As the computer attacks become more and more sophisticated, the need of new approaches and solutions to provide effective detection and reaction increases. In particular, one of the main challenges consists in appropriately facing distributed attacks, that are becoming the most effective threats to network security. Unfortunately, "isolated" security systems, as that presented in chapter 4, are bound to fail in identifying potential distributed threats, since usually the *locality principle*, either *spatial* or *temporal*, can't be fruitfully exploited to detect this type of attacks.

To cope with such lack of locality, it is possible to implement an "alliance" between isolated systems in order to face coordinated and distributed attacks. In a cooperative system for network security, each component exchanges, with its peer entities, information about the evidence of anomalous behaviors. Hence intuitively, the reliability of the network security system can be increased by adopting strategies for correlating the information provided by multiple sources. According to these assumptions, in the following of this chapter we will present a new cooperative solution for DDoS detection, which is able to improve the capability of the overall system to enforce the effectiveness of detection process.

5.1 DCube: Distributed DDoS Detection

Let us consider a potential target of an attack, be it a single host or an overall network. It is intuitive that it is easier to detect a suspicious activity close to the target (the borderline case being the target itself who is in charge of detecting the attack) rather than far away from it. This is particularly true in the presence of a threat foreseeing the coordinated interaction of a number of distributed nodes. In the depicted scenario, in fact, the closer to the victim, the more information can be retrieved in order to effectively detect a potential attack. On the other hand, detecting a threat far-off from the victim enables a quicker identification of the attack source(s), thus allowing for a prompt mitigation of the attack effects, as well as a reduced impact on the quality of service perceived by legitimate users of the network. Furthermore, it has been demonstrated in chapter 3 that solutions relying on the combination of information coming from multiple sources clearly could outperform legacy approaches based just on the elaboration of local data, which do not contain any context-related information. Local analysis, even though capable to infer information about the relations existing among a set of different traffic flows as shown in chapter 4, is strictly linked to data traversing a specific area of the network. Such an analysis is thus totally unaware of the global context, which is by no doubt in some relation with locally monitored events. This is the reason why we firmly believe that out-of-context information can prove as successful as context-aware data only in close proximity to the victim.

Let us consider the scenario shown in figure 5.1, where a distributed attack is depicted. Using a coordinated action the attackers quickly deny the availability of the target service to legitimate users. We can define a direction of the malicious activity from the attackers to the target. The "disruptive" power grows while the attack's flows direct towards the victim, thanks to their aggregation effects. The main idea of our approach is to prevent the catastrophic damage by means of a cooperative action of the nodes involved somehow in the attack process: each node supplies to other nodes its "talent" to detect an attack. The exchanged information flows in the same direction

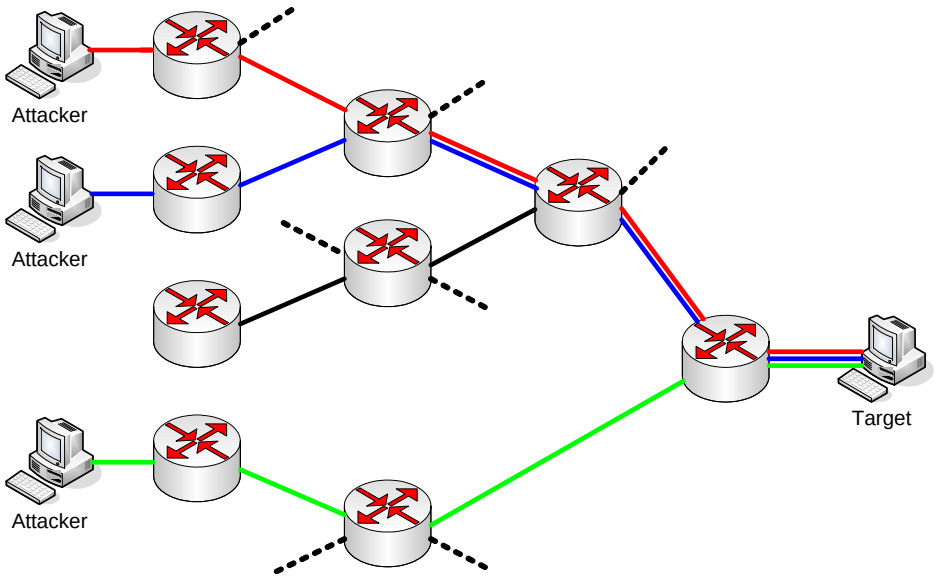


Figure 5.1: A typical distributed attack

as the attack, i.e. from the attackers to the target.

Each node will in some way use information sent by other nodes in order to improve its own detection capability. Once done with a decision about potentially harmful flows, it will in turn make such information available to the others (e.g. the downstream nodes along the path of the suspicious flow). This process perfectly fits the previous assumption about the improvement of the detection capability near the victim, thanks to broader information availability. Furthermore, thanks to this technique it is possible to adopt reactive policies (e.g. dropping of packets belonging to harmful flows, application of traceback techniques, etc.) faraway from the victim. The information about the ongoing attack keeps on growing as long as we move from the attackers towards the target.

Keeping in mind the above considerations, we proposed a distributed context-aware solution to issue of identifying distributed security threats. It has been called *DCube* (or D^3), standing for *Distributed DDoS Detection*. A possible deployment of system is depicted in Figure 5.2

The system consists of several components (DCube nodes) spread across the network, which are entitled to express their opinion about a specific

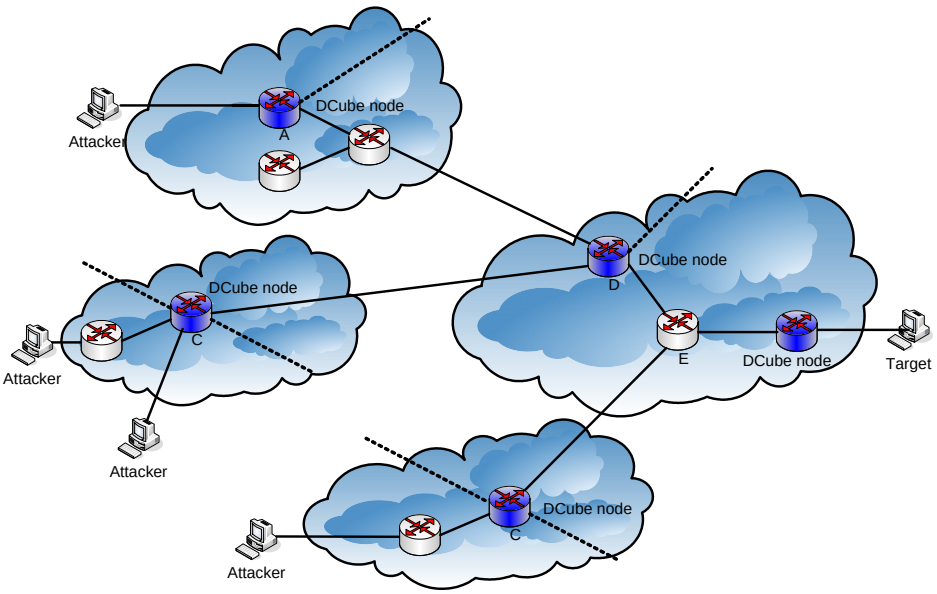


Figure 5.2: A DCube Deployment

situation. As an example, these nodes might be intrusion detection systems used to detect potential attacks to a network infrastructure; every node might implement a specific algorithm or techniques in order to identify security threats independently from the other nodes. At the occurrence of specific events (e.g. a new packet is captured, a timeout for exporting flow information expires, etc.) each DCube node will emit its own verdict depending on the inner behavioral rules it is based upon. Then, the verdict is sent to neighbor by a communication protocol for implementing the collaborative detection.

As already mentioned in chapter 3, several are the requirements for a cooperative solution.

In the following subsections we will describe the solutions proposed for meeting the issues mentioned above.

5.1.1 DCube Information Exchange Protocol

This section deals with the critical challenge of defining an effective protocol to exchange information among the network nodes. This protocol has the main task to support the system by providing it with the "relationships"

needed to make the system itself cooperative. Furthermore, in a collaborative and coordinated scenario, each node makes a decision taking into account the "context" of the network; this context, also known as "network status", is provided by the information exchange protocol. Like in a human community, the "news" spreads node by node so that a growing number of nodes gradually learns the network context. All node behaviors are thus influenced by the "opinion" of the entire node community.

The main task of this protocol is to let information spread through the node community. Whenever the network context changes new information must be sent to update the community members. This scattering process can be obtained in different ways. Borrowing from sensor network approaches [46, 47, 48], we adopt a hop by hop solution, whereby a node sends information to all its neighbors except the node which the "news" has come from.

Two different actions can be undertaken whenever a news arrives, based on whether or not the node is on the attack path. If the node is on the path, it reinforces or weakens the news based on its local "experience", and sends the new information to other nodes except the previous one on path. Otherwise, the node just forwards the information without its local contribution if it is not on the path. This solution is complied with need to assure the spatial proximity of the data exchanged. The neighbor nodes, especially these on the path, are interested in local experiences much more that the faraway ones.

Anyway, the protocol achieves a wider information spreading by sending data also to nodes not directly involved in the attack; in such way the "local" attack evidence can reach nodes involved in the same distributed and coordinated attack action, thus reinforcing their detection capability.

In order to reduce the overhead due to dissemination protocol the information exchanged must marginally weigh on current network load. For this reason we assure the data exchanging only during an attack ongoing, and design a data format which restrict the information to the essential one.

Keeping in mind these assumptions, for each critical flow we define:

CFID - Critical Flow ID: the ID of the critical flow, for example the 5-

Figure 5.3: DCube Packet

tuple (SRC_IP, DST_IP, SRC_PORT, DST_PORT, PROTOCOL) or any of its subsets.

MTR - Metric: a metric or a feature detecting an anomalous behavior of the flow.

CONF - Confidence: a single value or a vector of values representing the estimated reliability of the detection result for the critical flow related to the metric (e.g. I'm telling you that I detected a potential attack with 99% reliability).

LOC - Location: a flag that indicates whether or not the incoming information arrives from the same path of the critical flow or from a different path.

AGE - Ageing: a parameter indicating the distance in hops from the first node which has detected an anomaly in a monitored flow.

So, we define *DCube* Packet – $D3P$ – the vector:

$$D3P = (CFID; MTR; CONF; LOC; AGE) \tag{5.1}$$

Each node exchanges D3Ps with its neighbors (by means of the protocol described in the previous subsection) every time a new anomaly related to a flow is observed, or a D3P is received from a neighboring node and a new $CONF$ value is computed. This process is sketched in Figure 5.4.

In Figure 5.5 we draw the data flow diagrams for detection process carried out at a generic node. Every node analyzes a set of flows and for each flow in the set measures information according to the associated metrics. In case the monitoring activity raises an alarm for a certain metric concerning

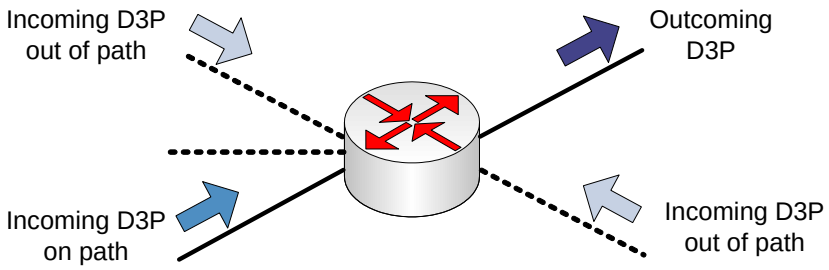


Figure 5.4: Exchanging of D3P

a specific flow, the node in question creates a new D3P packet and immediately thereafter starts the information spreading process by sending it to its neighbors. Moreover, if a new D3P packet arrives at the node carrying information about a flow which is not deemed to be critical on the basis of the local analysis, the node nonetheless merges its local information about such flow with the global information contained in the incoming D3P and then creates an updated D3P which is sent to its neighbors.

5.1.2 DCube Information Fusion

The main issues related to algorithm depicted in Figure 5.5 are the *MTR Analysis* and the *Information Fusion* process. This chapter deals mainly with the latter point. The former one, indeed, is related to the problem of local detection of malicious activities discussed in chapter 4: the IDS solution proposed previously can be exploited for assuring an effective detection.

The information fusion is a critical task for our system. It concerns the techniques for merging at node "local" knowledge about a critical flow with the "global" information about the same flow incoming from the other nodes.

As stated in chapter 3 a common problem in distributed collaborative system is how to combine the decisions coming from different modules with the final aim of improving both the accuracy of the classification process and the reliability of the decision-making process [49].

Several techniques have been proposed for combining data from multiple information sources. The simplest fusion mechanism is *majority voting* [50].

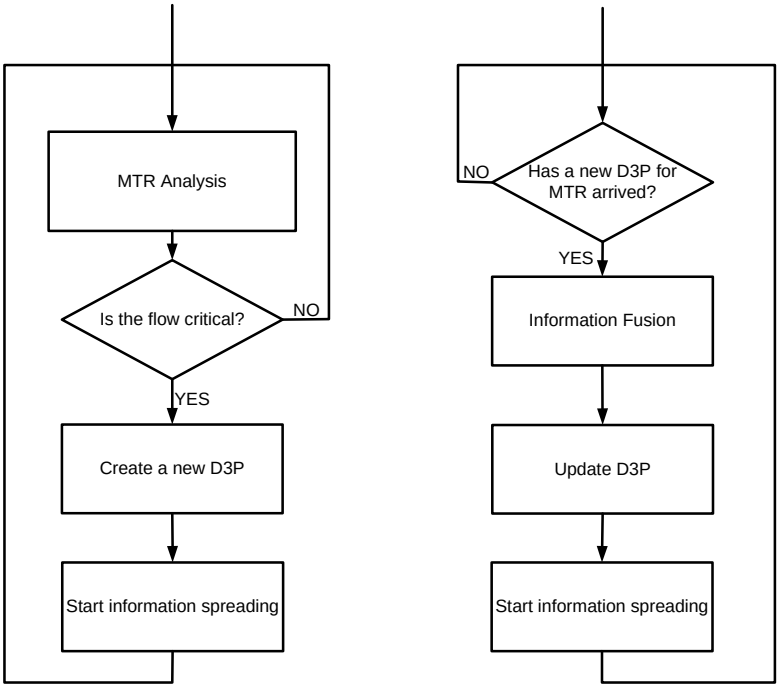


Figure 5.5: Data Flow Diagram

Such a method is very simple, and prone to errors or information injection attacks. A weighted majority might solve the problem of assigning a fair share of trustworthiness to each component of the whole system, though it does not express correctly the mutual relationships among detection sensors due to their relative position in the network, as well as their past interactions. This leaves the system exposed to failures due to misbehaving detection nodes.

The *bayesian fusion* method is, instead, the best-known one [51]; according with the Bayes formula, we can compute the conditional probability of an event E_i given the *a priori* probability of the event itself, and the conditional joint probability of the evidences on E_i from different sources:

$$P(E_i \mid S_1, ..., S_n) = \frac{P(S_1, ..., S_n \mid E_i)P(E_i)}{\sum_j P(S_1, ..., S_n \mid E_j)P(E_j)} \tag{5.2}$$

By correctly assigning the values of both a priori and conditional probability, it is possible to express the relative trustworthiness of each detection module. Still, no mechanism is used to cope with reciprocal trustworthi-

ness. Obviously, in order to use the bayesian fusion method, we need to be able to exactly characterize the possible events probabilitywise. This is, unfortunately, unpractical in most cases.

The *Dempster-Shafer* (D-S) theory, on the other hand, provides an interesting alternative to traditional bayesian models for the mathematical representation of uncertainty [52, 53, 54]. The basic principle which this theory is based upon consists of the fact that no a priori probability is needed for the events at hand. Indeed, the D-S theory aims at characterizing the degree of belief in the occurrence of an event based on the observation of a number of *witnesses*. In our case, the event set contains both the occurrence of a malicious behavior and a normal behavior; the sensors, instead, represent the witnesses reporting the occurrence of the monitored events. From this perspective, such a theory can be interpreted as a generalization of the classical probability theory, where probabilities are assigned to sets or intervals rather than to mutually exclusive singletons.

Moreover, the most innovative and interesting aspect of the D-S theory is the criterion which allows to combine evidence coming from multiple sources and to model conflicts among them. In fact, such criterion was calculated once and for all when such theory first came out, and is applicable regardless of the probabilistic characterization of both the monitored events and the detectors. It is worth discussing some of the formal details involved in D-S theory in order to make things clear.

Given a set of events X, we will consider its power set $P(X)$, and a generic set A such as $A \in P(X)$. Three fundamental functions are used in the framework of D-S theory, in order to express in a rigorous way the concepts of belief and reliability discussed so far: the *basic probability assignment* function (*bpa*), the *Belief* function, and the *Plausibility* function.

The basic probability assignment is a primitive of the evidence theory, and defines a mapping of the power set to the interval between 0 and 1.

$$m : P(X) \rightarrow [0, 1] \tag{5.3}$$

More exactly, the value $m(A)$ of the *bpa* for a given set A expresses the proportion of all relevant and available evidence that supports the claim that a particular element of the considered global set belongs to the set A but to no particular subset of A.

Formally, this description of *bpa* can be represented with the following equations:

$$m(\emptyset) = 0 \tag{5.4}$$

$$\sum_{A \in P(X)} m(A) = 1 \tag{5.5}$$

By means of equations 5.4 and 5.5, the upper and lower bounds of an interval can be defined. Such interval contains the probability of a subset of X, or an element of $P(X)$, and is bounded by two nonadditive continuous values called, respectively, *Belief* and *Plausibility*.

The lower bound, *Belief*, for a set A is defined as the sum of all the basic probability assignments of the proper subsets B of the set of interest A, such as $B \subseteq A$.

The upper bound, *Plausibility*, is the sum of all the basic probability assignments of the sets B that intersect the set of interest A, such as $B \cap A \neq \emptyset$.

Formally, for all sets A that are elements of the power set, $A \in P(X)$,

$$Belief(A) = \sum_{B|B \subseteq A} m(B) \tag{5.6}$$

$$Plausibility(A) = \sum_{B|B \cap A \neq \emptyset} m(B) \tag{5.7}$$

The D-S theory provides a rule to combine evidences from independent observers. To make things simple, let's assume two of them: O_1 and O_2. The combined *bpa* is expressed by:

$$m_{12}(A) = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{\sum_{B \cap C \neq \emptyset} m_1(B) m_2(C)} \tag{5.8}$$

The depicted D-S combination rule implies that we equally weight all the observers. This assumption normally does not hold in real environments. In our case, with an overlay infrastructure for security, sensors span multiple domains [53]. Hence, we need to take into account different levels of reliability. First, information coming from different remote sources is considered less reliable than the one produced by local sources. Second, multiple sources are located in different places; hence, they may capture different traffic profiles. An extended D-S combination rule has been proposed, which considers a "conditioned" view of evidence and proposes a modified combining rule able to take into account the above mentioned weights [53]. This new solution properly addresses the fact that we cannot trust all the sensors equally, and that given observers might have different effectiveness in detecting individual misuse types. The proposed combining rule is:

$$m_{12}(A) = \frac{\sum_{B \cap C = A} [m_1(B)]^{w_1} [m_2(C)]^{w_2}}{\sum_{B \cap C \neq \emptyset} [m_1(B)]^{w_1} [m_2(C)]^{w_2}} \tag{5.9}$$

It can prove useful to adopt a weight for each evidence based on parameters like the neighbors' historical performance figures. So we consider the D-S combining rule expressed in 5.9, where the weight w_i for the i-th evidence is a function of location parameter LOC, and the ageing parameter AGE:

$$w_i = f(LOC, AGE) \tag{5.10}$$

In the following we will focus on the these two parameters, which both deal with issues related to a potential attack scenario and that must be considered to make the system more robust by improving its capability to react to threats as soon and as effectively as possible. We base our discussion on the following fundamental observations related to the detection mechanism:

- the farther the distance from the target, the less the information available to the detection process. Hence, as detection capabilities improve gradually from the attacker to the target, a *CONF* value too high in a node far-off from the target might compromise the detection by biasing it around a bad decision. A mechanism for appropriately weighing the *CONF* value based on the distance from the target is needed.

- It is important to make a distinction between critical flow information coming directly from a node belonging to the attack path and information received from nodes outside the path.

As an example of the application of the above considerations, let us illustrate a potential information fusion algorithm. Let us consider the following hypothesis. The possible values for w_i range from 0 to 1; in particular, w_i is equal to 0 when AGE is equal to 1, and LOC is equal to 0. Contrariwise, w_i is equal to 1 when AGE is greater then a certain threshold T, and LOC is equal to 1.

So if we consider:

$$w_{AGE} = \begin{cases} 0 & \text{if AGE} = 1, \\ f(AGE) & \text{if } 1 < \text{AGE} < \text{T}, \\ 1 & \text{if AGE} > \text{T}. \end{cases} \tag{5.11}$$

then we can adopt the following formula:

$$w_i = \frac{LOC + w_{AGE}}{2} \tag{5.12}$$

Anyway, it is also possible to assign a different meaning to the two parameters above. For example, we might consider that LOC is more important than the AGE: in this case equation 5.12 becomes:

$$w_i = \frac{c_l LOC + c_a w_{AGE}}{2} \tag{5.13}$$

with

$$c_l = 0.6, c_a = 0.4 \tag{5.14}$$

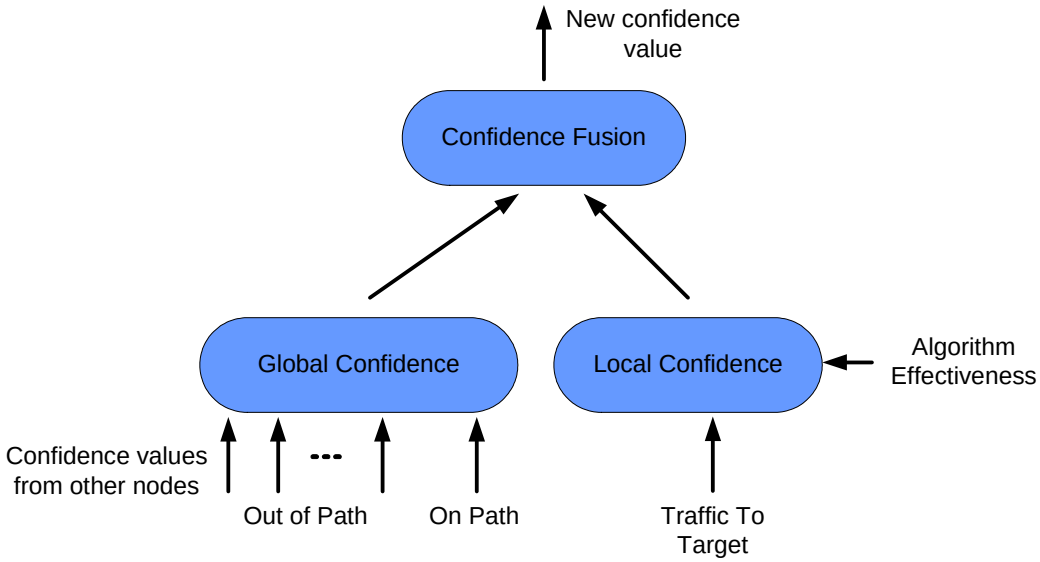


Figure 5.6: Information fusion scheme

According to the assumptions above, we propose an algorithm for aggregation of evidence information (Figure 5.6).

A top-down view of the picture, shows how a new confidence value can always be computed by appropriately combining both local and global confidence information. As to the global contribution, the figure highlights that foreign confidence data can come from nodes which happen to be either on- or off-path with respect to the specific attack pattern. As already mentioned, on-path nodes might have a stronger impact on the computation of global confidence than off-path nodes which can often be supposed to have been looking at attack data from a narrower perspective. Finally, local confidence might be influenced by both information extrapolated from local traffic analysis and beforehand information about the actual effectiveness of the locally employed detection algorithm(s).

From an implementation point of view we can consider MTR analysis is performed by an IDS, for example the system presented in chapter 4. We assume that each IDS classifies network traffic by associating each packet with a specific "class". For the sake of simplicity, we herein make the hypothesis that

the classes involved are the following: *attack* and *normal*. Hence, the power set involved in our discussion is $P(X) = \{\{attack\}, \{normal\}, \{attack/normal\}\}$. This means that upon reception of a packet, each DCube node classifies it depending on its own *bpa*, which is seen as a vector of three possible values, associated, respectively, with the probability that the packet in question is either an attack, or a normal piece of information, or something which the IDS is not able to classify as belonging to either of the above classes.

The output of the different nodes is then normalized in order to have a uniform view of the various responses. By adopting a model proposed by Gargiulo et al. [55], the normalization process is performed by computing an appropriate *indicator*, X, which in our case is expressed by:

$$X = \frac{m(attack) - m(normal)}{1 - (m(attack/normal))} \tag{5.15}$$

As explained above, the value $m(A)$ of the *bpa* for a given set A and a given detection module expresses the fraction of all relevant and available evidence supporting the claim that a particular observed event belongs to the set A and to no particular subset of A.

The above indicator has been conceived in such a way as to guarantee that the following properties are always respected:

- the sign of the function must depend just on the numerator;
- the variability range of the indicator must be $[-1, +1]$. The closer to 1, the more likely the classified packet belongs to an attack session; similarly, the closer to -1, the more likely the packet contains normal data.
- The value of the denominator has no influence on the sign of the indicator. Though, the higher the value assigned to the composite hypothesis (i.e. *attack/normal*), the farther the indicator will be from both -1 and $+1$ limit values.

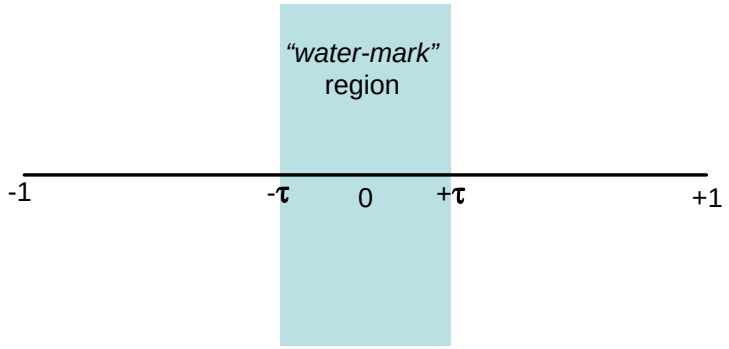


Figure 5.7: The water-mark region

We remark that such indicator proves useful only in case we consider two alternative classification clusters (like the classes *attack* and *normal* in our example, plus the third class *attack/normal* which is just used to deal with uncertainty). In case more classes were present, a different indicator should be considered.

Given the above formulation, we can try to balance between the number of rejected packets (i.e. those packets for which we are unable to issue a verdict) and the misclassification rate, by introducing an ad hoc threshold value τ (Figure 5.7). The classification process will be performed as follows:

- Attacks: all packets for which $X > \tau$;
- Normal: all packets for which $X < -\tau$;
- Rejected: all packets for which $-\tau \leq X \leq \tau$.

Once a peer receives attack evidence from an DCube neighbor it applies the Dempster-Shafer's weighted combining rule reported in equation 5.9, using the weights w_i in order to obtain a verdict that takes into account both local observations and the observation of the neighbor. The response comes in the form of an indicator representing the new CONF value, which is sent to the neighbors in $D3P$ packet.

5.2 REFACING: A Reputation Model

Although this approach based on weighted D-S combining rule 5.12 is more general than the basic one and can be applied to several kinds of attack scenarios, it still leaves an important open issue. This is mainly due to the fact that information fusion approaches are mostly focused on the development of effective analytic methods to evaluate combined evidence. However, we have to take into account the actual deployment of sensors exploiting these approaches in real-world scenarios, in which the inherent dynamism forces both evidences and weights to change in order to adapt combining rule's results to the *situational-context*.

Our work mainly deals with the practical application of the mentioned theoretical results to the effective management of components for network security. One major issue concerns the procedure for adjusting the weights that have to be adopted while evaluating the enhanced D-S combining rule, and it can be addressed by equipping a system with self-management functionality. We claim that in the above mentioned scenarios (and especially in those related to network security) a broader vision should be considered in order to make the deployment of the information fusion process more concrete. To this aim, in this section we propose an improvement of the D-S formulation by introducing a reputation model for a *self-management* approach to network protection. We will propose an interpretation of Information Fusion as a *situational-aware* and *automatically adaptive* decision-making process.

Several solutions have been proposed for estimating the reputation in a cooperative system [56, 57, 58].

There are essentially three approaches to reputation estimation. The first relies on a completely centralized solution, where an authority assigns a trustworthiness value to each element; the components interact by using such reputation labels. In the second approach, the trustworthiness value of a component is estimated by collecting all the opinions about it coming from the other elements of the community; usually, a centralized system is responsible for collecting information and publishing the reputation values.

A completely distributed solution for reputation characterizes the third approach: each element "locally" builds the opinions about other components, just based on its direct and indirect interactions with them. Although the third solution is more similar to the concept of "social network", we claim that, at present, a centralized entity holding a global reputation value for each component better fits the security requirement of networks. "Local" solutions, based on the direct interaction among the nodes, could be vulnerable to violations coming from deliberately misbehaving nodes. Some compromised components could, in fact, spread false information, in order to reduce the effectiveness of the detection. Anyway, a completely distributed solution, with some innovative approach to cope with false information spreading is currently the subject of our work. According to the second approach, we propose a multi-layered model for estimating the reputation of network components taking part to the distributed detection process, called *REFACING* (*RElationship-FAmiliarity-Confidence-INteGrity*). The lowermost layer provides information about the existence of some form of *interaction* among "community" components. The absence of connection indicates the actual impossibility of carrying out any form of *social relationship* with the other nodes of the network. Otherwise, the second layer in the stack can prove useful to *quantitatively* measure the level of interaction existing between each pair of network nodes. The more we interact, the more familiar we are with each other. Yet, this does not necessarily imply that we trust each other: I can know you quite well, but (or even better, just because of this) I can hardly trust you, if our past interactions showed me that you are not that reliable. This is the reason why we introduce the third layer of the trustworthiness stack, which deals with *confidence*. If I have relations with others, and if I am familiar with the others as well, I can much more objectively determine their level of trustworthiness with respect to our social interactions. This said, to further foster the capability of assessing someone else's *loyalty* level related to his/her interactions within the network, one more dimension should be taken into account to somehow reflect the *variability* in the behav-

ioral interaction patterns of each node. To make things clearer, the fact that some node has showed a blameless behavior in one single interaction does not necessarily mean that such node shall be irreproachable also in its subsequent interactions. Some form of estimation of the *line of conduct* over time is definitely needed for all nodes: the more coherent my behavior has been in the past, the less probable it will be that I will behave too differently in the near future. This issue is dealt with at the uppermost layer, which provides information about the level of *integrity* of network nodes. We do believe that the adoption of such a multi-layered model helps add objectivity to the assessment of network nodes' reputation, since it takes into account a number of complementary, though highly correlated, facets. The reputation system has a global view of the physical topology of the network, and is thus capable to determine whether or not there exists some form of relationship (layer 1 in the trustworthiness stack) between the networks node, by interacting with them. Furthermore, by exploiting monitoring, it can also determine the frequency of the interactions among them (layer 2 of the stack). Information pertaining to the third layer can be retrieved through a comparison between each evaluation provided by any single node and the global *opinion* of the system (e.g. my *confidence* level gets higher if my personal evaluation was found in accordance with the final decision taken by the distributed detection system, after analyzing all single decisions coming form the security nodes). Finally, data at the fourth layer can be computed by statistically analyzing the information related to all past interactions among all underlying nodes (e.g. my *integrity* level gets higher if my *confidence* level has kept on growing over the past interactions). After each evaluation turn, the management layer can compute a set of labels (one for each network node involved in the detection process), which are assigned to the nodes through, for example, a policy-based approach. The label computation process can be as general as possible, and will normally be influenced by information belonging to all of the layers in the trustworthiness stack (in a simplistic scenario, it might for example be a simple weighted sum of the values computed at

each of the four layers). The labels are then used by all nodes whenever they start a new interaction. Each label acts like a *business card* for the node involved in the interaction and can be used by the other nodes, in order to assign a weight to the information that each of them receives from its partners. In order to better highlight the potential application of this novel self-management approach to improve the information fusion process, let us consider the weighted D-S combining rule expressed in (5.9). In this formula, w_i is the weight for the generic observer O_i. As claimed above, depending on the situational-context, the management process can evaluate the level of trustworthiness of each node involved in a generic transaction, by quantitatively measuring the relationship, familiarity, confidence and integrity metrics.

During each transaction, each node has to evaluate the i_{th} weight to be employed in the D-S combining rule. To this aim it will apply a *utility function*, which we call *management function* (MF), to the previously measured values as expressed below:

$$TRUST_i = MF(Rel, Fam, Conf, Int) \tag{5.16}$$

For administrative purposes the *management function* can be appropriately suited to meet the specific domain's high-level goals and/or requirements.

The weight w_i in weighted D-S combining 5.9 becomes:

$$w_i = \frac{LOC + w_{AGE} + TRUST_i}{3} \tag{5.17}$$

or:

$$w_i = \frac{c_l LOC + c_a w_{AGE} + c_r TRUST_i}{3} \tag{5.18}$$

5.2.1 REFACING and the Byzantine Generals Problem

First to focus on the implementation issues related to REFACING model, in this subsection we comment on the possibility that some of the peering entities in the DCube community act as liars, like in the well-known Byzantine Generals Problem [59]. It looks clear that in any networked environment whose correct operation is based on the exchanging of information among the various network entities, the hypothesis that none of these entities is a liar must necessarily hold. This is true also in the case of DCube. In fact, as already explained, our architecture envisages that each peer sends to its neighboring entities a D3P each and every time something worth noting happens. The REFACING model provides the current trustworthiness value that the node has been assigned. The *TRUST* parameter actually represents the value of the trustworthiness label that has been computed by the system in the latest update cycle. Nothing prevents a node from lying, by treacherously inserting in the exchanged D3Ps modified values. The risks by this issue are worked around by leaving the management layer to explicitly communicate to each peer the values of the labels of all of its neighbors. In this way, at least for what concerns the *TRUST* parameter, the system becomes capable to self-protect against the mentioned issue. It is understood that this approach only solves the problem of ensuring correct information exchanging with respect to sensitive information about critical flows and associated metrics, which might in any case keep on lying on other aspects linked to the communication between peer and the overlay management system.

5.3 REFACING Reputation System

In this section we present our proposal for "trustworthiness" estimation by means of the REFACING model. The informal flow diagram in Figure 5.8 depicts the behavior of the system we realized.

In our view, the solution is implemented as an overlay layer which man-

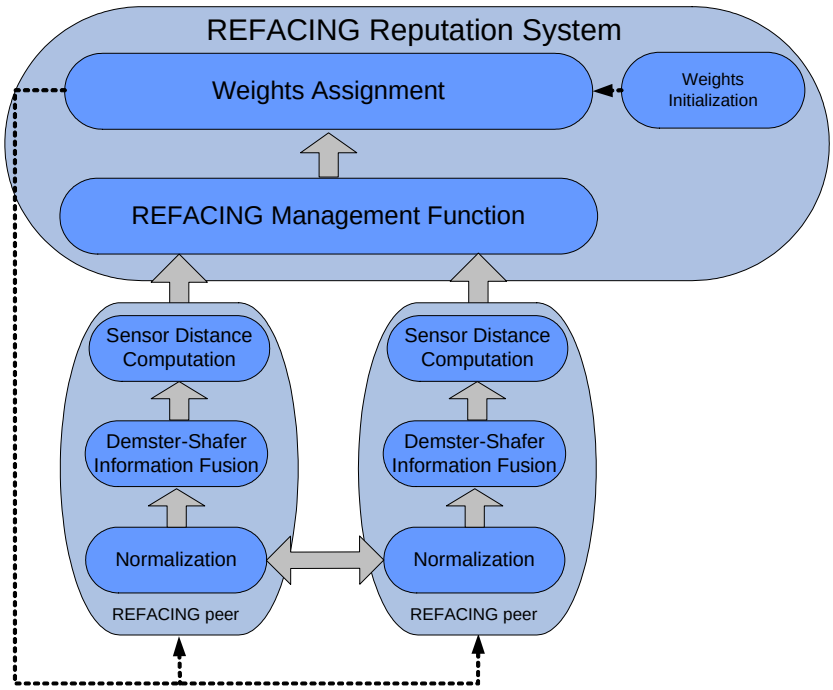


Figure 5.8: REFACING System Data Flow

ages the overall DCube infrastructure (Figure 5.9).

In order to allow the REFACING system to perform the estimation of the weights, each DCube node computes the distance Δ between the value X_{local} of its own indicator before the fusion process and the value X_{global} received by the neighbor, as mentioned in the section 5.1.2:

$$\Delta_{Fusion} = |X_{local} - X_{global}| \tag{5.19}$$

The value of Δ_{Fusion} sent to the REFACING system can be considered as a measure of the level of disagreement between the node's verdict and the verdict provided by the neighbors in the previous steps.

It is possible to determine whether or not a peer's response was in accordance with the global response by first analyzing the product $(X_{local} \cdot X_{global})$ between the node's indicator and the global one. If such product is negative, there is disagreement in the responses; on the other hand, a positive result means that the specific node and its neighbor have converged on a common

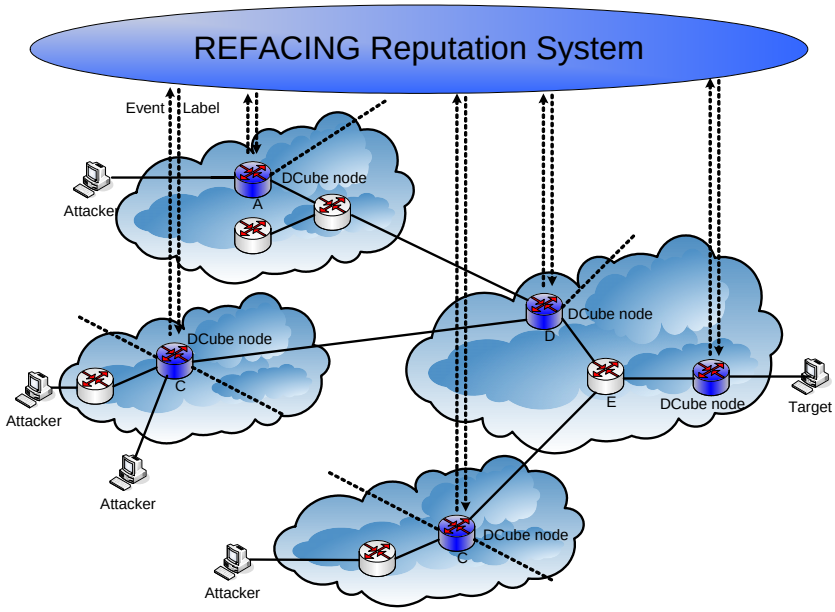


Figure 5.9: REFACING Reputation System

decision.

The value of Δ_{Fusion} helps quantify the level of agreement/disagreement between the two entities and it is used in the REFACING management function to update the node's reputation weight, as it will be thoroughly explained in the following of this section. For the example above, we have derived the following management function, used to perform the weight updating process. Let us define:

- $TRUST_i$: the value of the trust of a generic node at the i-th iteration;
- RD_i: the degree of correlation of the node at the i-th iteration;
- A_i: the mean value of the node's trust at the i-th iteration;
- V_i: the variance of the node's trust at the i-th iteration;
- SR_i: the number of transactions during which the node has actively contributed (i.e. it has issued his own verdict) to the distributed detection process, at the i-th iteration;

- NR: the total number of transactions.

First, we compute the instantaneous value of the trust at the i-th transaction, based on the values it assumed in the past:

$$TRUST_i = A_{i-1} + \Delta \tag{5.20}$$

where Δ is computed as follows:

$$\Delta = \begin{cases} \Gamma * \frac{1-A_{i-1}}{2} * (1 - \frac{\Delta_{Fusion}}{2}), & \text{Agreement,} \\ -\Gamma * \frac{A_{i-1}}{2} * (\frac{\Delta_{Fusion}}{2}), & \text{Disagreement.} \end{cases} \tag{5.21}$$

In equation 5.21, the term Γ is computed as follows:

$$\Gamma = \alpha * A_{i-1} + \beta * RD_{i-1} + \chi * V_{i-1} \tag{5.22}$$

with:

$$\alpha + \beta + \chi = 1 \tag{5.23}$$

In our case, the parameters above have been set by trial and error and have the following values:

$$\alpha = 0.6,\ \beta = 0.3,\ \chi = 0.1.$$

At this point, we can update the old values, which will be used to compute the instantaneous weight at the occurrence of the next transaction:

$$RD_i = 1 - \frac{NR - SR_i}{NR} \tag{5.24}$$

$$A_i = \frac{SR_{i-1} * A_{i-1} + TRUST_i}{SR_i} \tag{5.25}$$

$$V_i = \frac{SR_{i-1} * V_{i-1} + (TRUST_i - A_{i-1})^2}{SR_i} \tag{5.26}$$

5.4 Experimental Results

In this section we present a performance evaluation of our solution. We show the improvement achieved by our system with regards to previous solutions.

The analysis is conducted through an extensive simulation-driven campaign. We developed a simulator, called *RefacingSimulator*[1], allowing us to test the performance of the solution adopted in a number of different scenarios. In the following subsection we first briefly provide a description of the simulator, then we describe some interesting experimental results.

The RefacingSimulator is a software tool implemented in Java following the classical Object Oriented design patterns. We implemented a number of sensor categories, which have been used for our experimental measurements.

These categories are briefly described in the following:

- *Reliable Sensor* the ideal class of sensors; it provides a response which is always in accordance with the event generated by the *Attacker*;
- *Lying Sensor* the worst conceivable sensor; it always disagrees with the event generated by the attacker, thus providing an error detection rate of 100%;
- *Shy Sensor* not always participates in the detection process. Though, in case of participation, it provides a response which is always in accordance with the event generated by the *Attacker*;
- *Variable Sensor* decides randomly, based on a configurable probability value, whether or not to provide a response which is in accordance with the event generated by the *Attacker*;
- *Shy Variable Sensor* implements a sensor showing a hybrid behavior, influenced by both the *shy* and the *variable* paradigms. More precisely, it does not always participate in the detection process, but in case of involvement it randomly decides whether o not to provide a response which is in accordance with the event generated by the *Attacker*;

[1] http://refacing.sourceforge.net

Scenario 1 (1000 Transactions, Attack Probability = 0.5)	
Sensors	**Detection Errors**
10 Reliable Sensors	0
10 Variable Sensors (Pe=0.3)	250
Majority Voting	0
D-S / No REFACING	6
D-S + REFACING	0

Table 5.1: Scenario 1

Scenario 2 (1000 Transactions, Attack Probability = 0.5)	
Sensors	**Detection Errors**
10 Reliable Sensors	0
10 Variable Sensors (Pe=0.7)	702
Majority Voting	0 (1 rejection)
D-S / No REFACING	29
D-S + REFACING	0

Table 5.2: Scenario 2

- *Bursty Lying Sensor* represents a liar, as in the case of the simple *lying sensor*, with the difference that the lies it tells are concentrated in bursts whose duration can be a priori configured.

In order to evaluate the effectiveness of the proposed solution, we performed some simulations in different scenarios. For all the scenarios we are going to describe, we performed 1000 transactions, by choosing an event generator characterized by an attack generation probability of 0.5 (i.e. one out of two generated events represents, on average, an attack). For the sake of simplicity, the following *bpa* function has been adopted for all peers: $bpa(Normal) = (0.896, 0.0, 0.103)$ and $bpa(Attack) = (0.0, 0.896, 0.103)$. Such a value for the *bpa* indicates a sensor which is quite good at detecting both the *attack* and the *normal* class, characterized by a low, though non zero, level of uncertainty. For the sake of completeness, it's worth pointin out that the issue of sensor characterization in the practice is beyond the scope of this paper. The number of detection errors is calculated by summing up the number of false negatives and the number of false positives. The number of rejections, indicated in parentheses for majority voting, is not included as part of the errors, hence it is reported separately.

Scenario 3 (1000 Transactions, Attack Probability = 0.5)	
Sensors	**Detection Errors**
2 Reliable Sensors	103
12 Variable Sensors (Pe=0.3)	336
5 Lying Sensors	897
10 Bursty Variable Sensors	134
Majority Voting	153 (57 rejections)
D-S / No REFACING	520
D-S + REFACING	103

Table 5.3: Scenario 3

Scenario 4 (1000 Transactions, Attack Probability = 0.5)	
Sensors	**Detection Errors**
10 Variable Sensors (Pe=0.5)	378
Majority Voting	312 (134 rejections)
D-S / No REFACING	480
D-S + REFACING	508

Table 5.4: Scenario 4

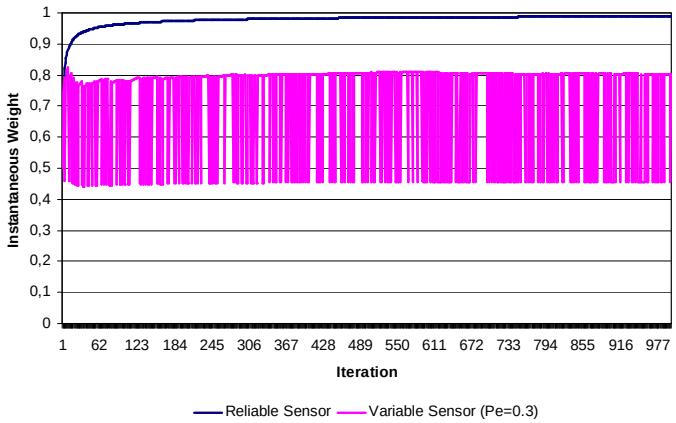


Figure 5.10: Scenario 1: Instantaneous Weight

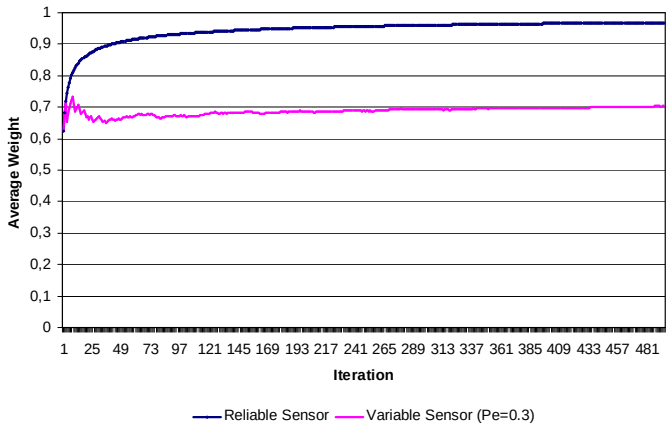


Figure 5.11: Scenario 1: Average Weight

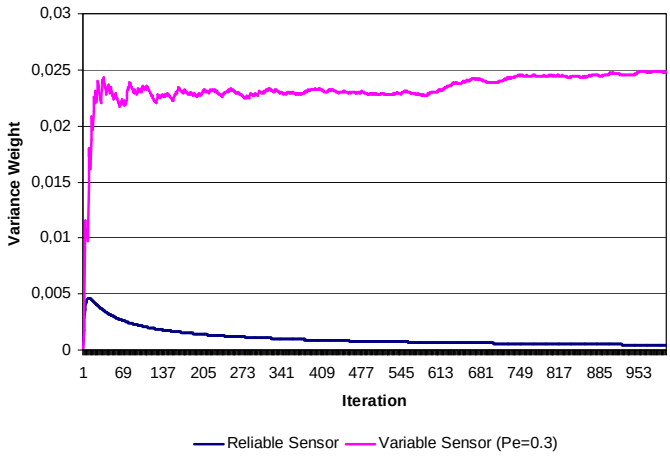


Figure 5.12: Scenario 1: Variance Weight

5.4.1 Scenario 1

Scenario 1 presents a configuration, composed of a total number of 20 REFACING peers, with 10 reliable sensors and 10 variable sensors, with a detection error probability of 0.3 (Table 5.1).

From the simulation results we firstly observed that the *variable* peers present an average number of detection errors equal to 25% of the total number of transactions. This is in accordance with the detection error probability they had been assigned. By applying the basic D-S formula (i.e. in the absence of REFACING), we observed, on average, a reduction in the number of detection errors equal to 97%. On the other hand, by applying the weighted D-S formula with the REFACING approach we observed a further decrease in the number of detection errors, which are in this case completely eliminated.

Compared with the majority voting, our approach does not provide any improvement. This can be justified by observing that the number of variable nodes is exactly the same as the number of reliable ones. For this reason the majority of nodes expressing the same decision always includes the set of reliable nodes, together with a few variable ones and, therefore, the detection errors are zero.

From a more detailed analysis of the graphs we can also draw that the instantaneous value (and thus the average value) of the weight assigned to *reliable* sensors keeps on growing during the simulation. This means that the decisions taken by reliable peers become more and more important (i.e. they have more and more influence on the final verdict) as long as the system evolves. This result is justified by observing the average behavior of such peers, which present the maximum degree of correlation (equal to 1 all along the simulation), a very small variance, and a very high average weight.

On the other hand, *variable* peers show an ever-decreasing trend with respect to the values of their weights (which are also rather low). This is due to the fact that their detections suffer from both errors and discontinuities. Weight decreases in case of detection errors are practically the same

(in module) as weight increases in case of correct detection. This is justified by the "constant" behavior of all peers in terms of both average values and variance of their weights. Finally, weight reductions obtained for variable sensors during the first transactions are due to the fact that the detection errors produced by such peers are highly biased around the first few simulation transactions.

5.4.2 Scenario 2

Scenario 2 presents the same configuration of scenario 1, except for detection error probability of 0.7 for variable sensors (Table 5.2). Compared to the results from the scenario 1 we can draw almost the same considerations. Though, differently than before, we can now observe a greater decrease in the value of the weight assigned to variable sensors. This is clearly justified by the higher degree of uncertainty associated with detection errors. More precisely, instantaneous weight values (and hence average weight values), are much lower than in the previous case. This entails a reduced impact of unreliable peers, and definitely improves the system's detection capability (thanks to the decrease in the number of detection errors). In this case, the REFACING approach presents almost the same results when compared to majority voting. In this scenario, since the error probability of the variable sensors is higher, variable sensors might be simultaneously in accordance between each other and, at the same time, in disagreement with the reliable nodes.

5.4.3 Scenario 3

In scenario 3 we tried to draw a more realistic environment composed of 20 REFACING peers, with 2 *reliable*, 12 *variable* with a detection error probability of 0.3, 5 *lying* peers, and finally 10 *bursty variable* peers with a detection error probability of 0.3 and a burst of 50 detection errors all in the first transactions (Table 5.3).

We observed that *reliable* peers present a false negative rate equal to

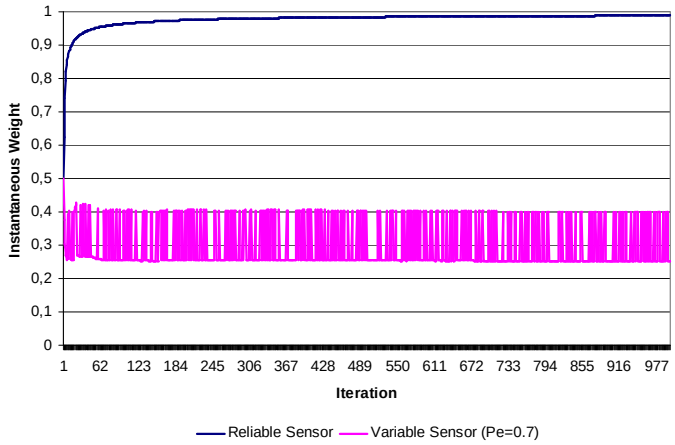


Figure 5.13: Scenario 2: Instantaneous Weight

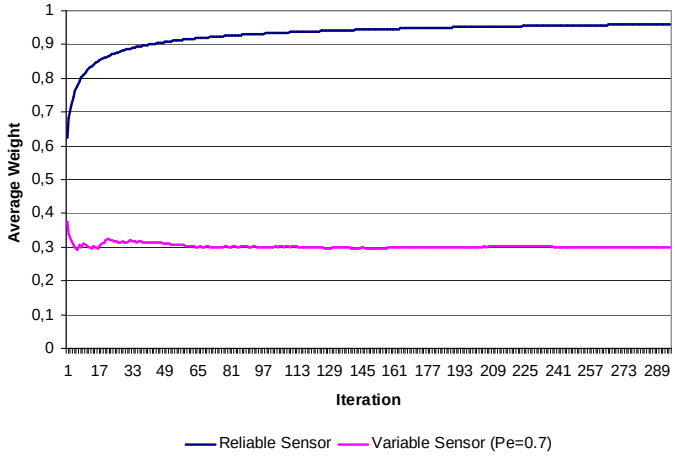


Figure 5.14: Scenario 2: Average Weight

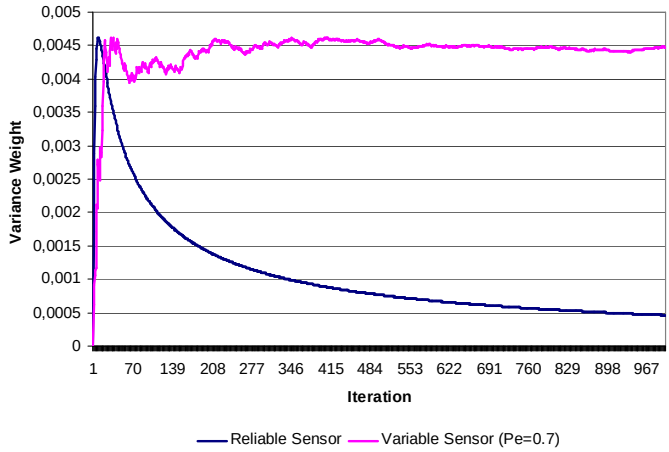


Figure 5.15: Scenario 2: Variance Weight

10, 3% and 33, 6% of the total transactions respectively for *reliable* peers and *variable* ones. The latter figures are in accordance with the detection error probability assigned to the peers. *Lying* and *bursty* peers, instead, present an average number of detection errors equal to 89, 7% 13, 4% and of the total number of transactions. Furthermore, by applying the weighted D-S formula (with the REFACING approach) we observed a very sensitive decrease in the number of misdetections: the number of detection errors is, in fact, 75% less than in the case the basic D-S formula is adopted. This means that the application of the REFACING model to the weighted combination procedure allows for an optimization of the overall system's behavior: the number of detection errors (after the information fusion process) for this scenario is equal to the error detection rate of the involved *reliable* peers.

In order to further confirm the validity of our method, a comparison with the majority voting approach has also been carried out. More precisely, an improvement of 10, 7% in terms of the average detection error rate has been achieved. By analyzing the graphs from simulations, we can also draw that the instantaneous value (and thus the average value) of the weight assigned to *reliable* sensors keeps on growing during the simulations. This means that the decisions taken by reliable peers become more and more important (i.e. they have more and more influence on the final verdict) as long as the

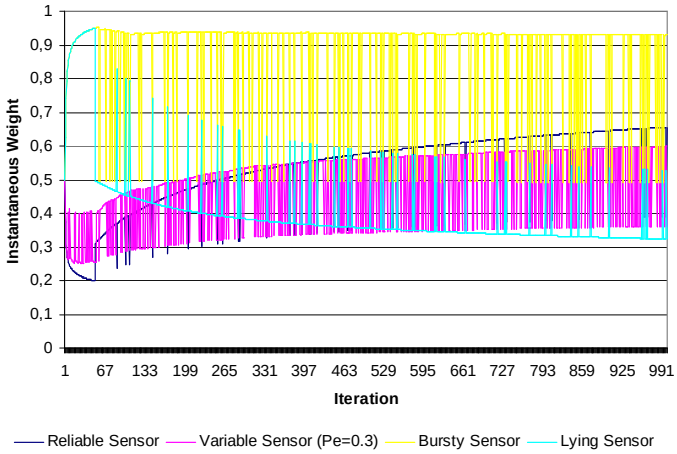


Figure 5.16: Scenario 3: Instantaneous Weight

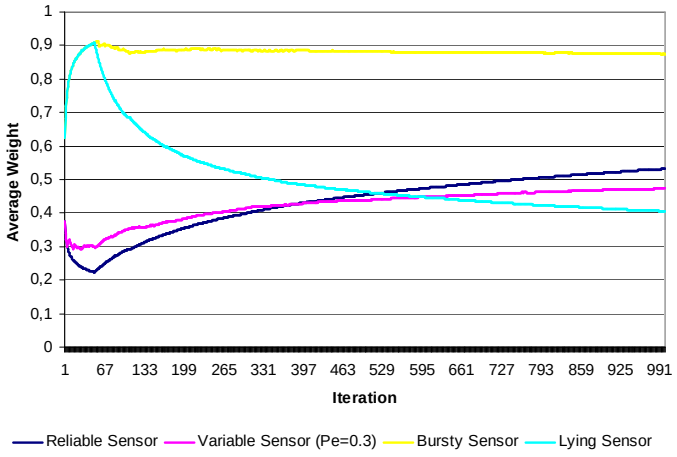


Figure 5.17: Scenario 3: Average Weight

system evolves.

This result is justified by observing the average behavior of such peers, which present the maximum degree of correlation (equal to 1 all along the simulation), a very small variance, and a very high average weight. On the other hand, *variable* peers show an ever-decreasing trend with respect to the values of their weights (which are also rather low). This is due to the fact that their detections suffer from both errors and discontinuities. Weight decreases in case of detection errors are practically the same (in module) as weight increases in case of correct detection. This is justified by the "constant"

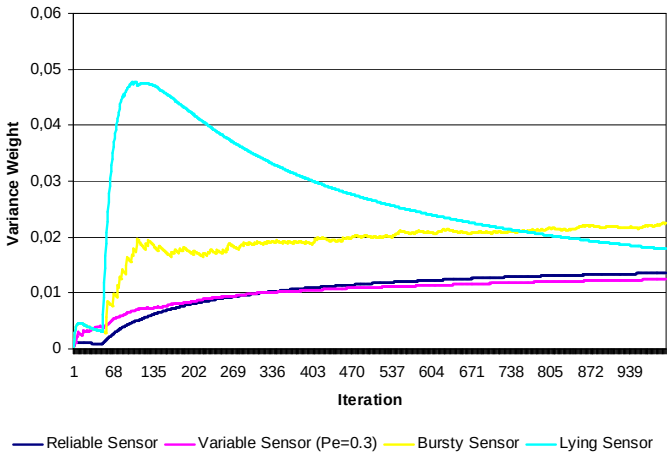


Figure 5.18: Scenario 3: Variance Weight

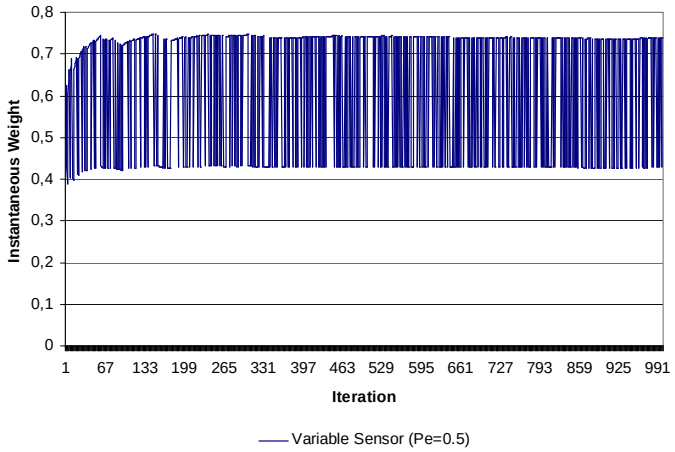


Figure 5.19: Scenario 4: Instantaneous Weight

behavior of all peers in terms of both average values and variance of their weights. Finally weight reductions obtained for variable sensors during the first transactions are due to the fact that the detection errors produced by such peers are highly biased around the first few simulation transactions.

5.4.4 Scenario 4: not all that glitters is gold

Scenario 4 presents a simple configuration of 10 variable REFACING peers, with a detection error probability of 0.5. From simulation results we have an average number of detection errors equal to $37,8\%$. Unfortunately the

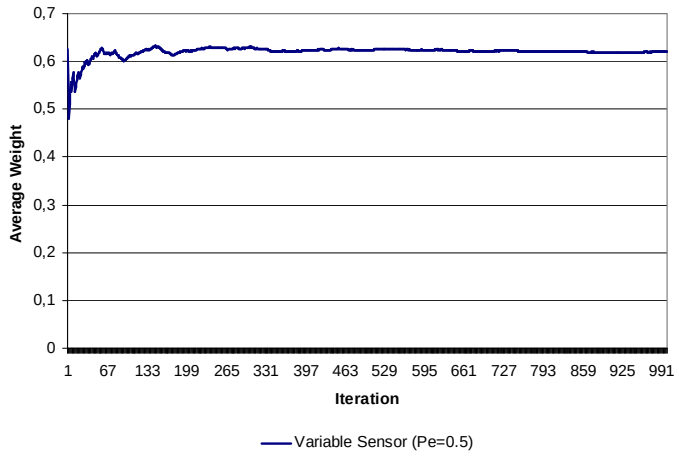


Figure 5.20: Scenario 4: Average Weight

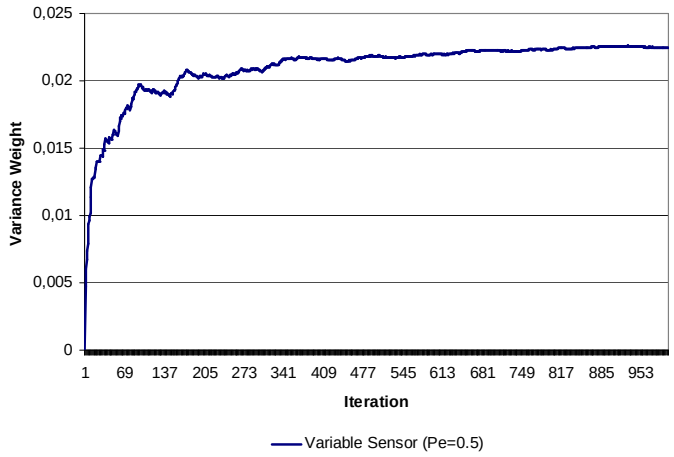


Figure 5.21: Scenario 4: Variance Weight

weighted D-S formula (with the REFACING approach) increases the number of misdetections with respect to the case with just basic D-S formula. The same consideration can be extended for the majority voting approach, which has provided an experimental improvement of 6.2% with respect to the REFACING method. We figured out that the high false negative rate characterizing the variable peers (with no mitigation effect from the reliable peers as it happened in the previous scenarios) brings to assigning high instantaneous weight values to these sensors. This has the effect of inverting in most cases the output of the fusion process, thus increasing the percentage of detection errors with respect to the case when the basic D-S formula is applied. This scenario clearly represents an example of a potential case in which the very low reliability level of the detection peers causes an appreciable reduction of the otherwise positive effects of the weighted fusion process exploiting the REFACING approach.

5.4.5 Summary considerations

To summarize the considerations we made, we herein provide in tabular form the main performance figures characterizing each of the aforementioned scenarios. Furthermore, in Figures 5.22 and 5.23 we analyze the performance of the proposed Management Function. Figure 5.22 shows the percentage of detection errors respectively in the presence and in the absence of the REFACING management function, and in the case of majority voting. It can be easily noticed that in all but the last scenario we achieve a considerable performance improvement. The best figures are obtained for both scenarios 1 and 2. Indeed, Figure 5.22 highlights an error reduction to 0%. Good results are also attained with scenario 3, with an overall decrease in the error detection rate that is close to 40% with respect to the absence of the REFACING function, and to 10% with respect to the majority vote. In all cases we observe that the adoption of the REFACING approach guarantees an error rate which is less than 10%. Again, with reference to scenario 4, the trend inverts and shows a performance decrease close to 6%. As already

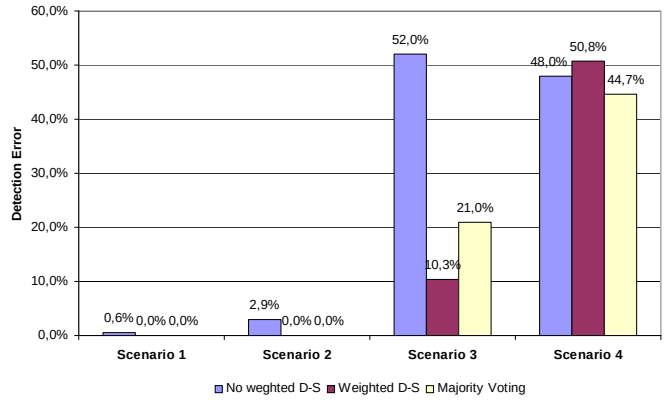


Figure 5.22: Detection Error

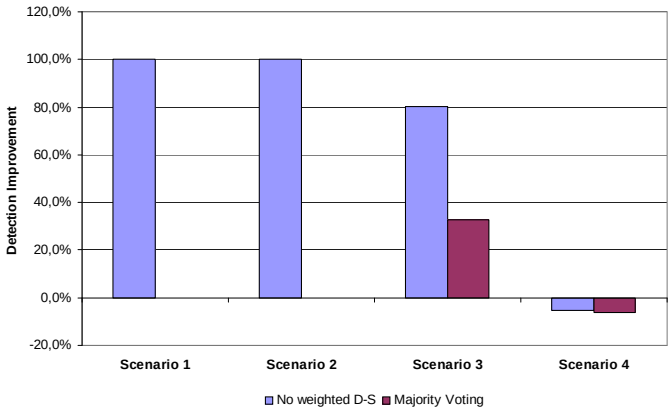


Figure 5.23: Detection Improvement

discussed, the reason behind such performance debacle can be ascribed to the presence of a high number of sensors characterized by very poor detection capabilities.

Chapter 6

A Secure Routing Protocol for Wireless Mesh Networks

Nowadays, human activities rely more and more on wireless infrastructures. Such technology is changing people's behaviors since the possibility of mobile communications has increased the number and the types of services provided by the network.

Unfortunately, the growing diffusion of wireless technologies also opens several issues, related in particular to network security. As stated in chapter 2 the lack of wired connections increases the risks of damages due to malicious users.

The attempts to deny wireless services are becoming more and more frequent and nowadays the DoS attacks are the main threats for these networks.

In this section we will deal with security problems in wireless networks. We will then propose a cooperative solution to protect the network against a specific class of DoS, namely the attacks to routing protocols.

6.1 Introduction

In chapter 2 we widely debated the problems of security in wireless networks. In particular, we observed the criticality of attacks to routing protocols that can compromise a fundamental functionality of these infrastructures. We analyzed several attacks pointing out, in particular, the dangerousness of

blackhole and grayhole attacks (see section 2.6). Subverted nodes can maliciously interrupt forwarding mechanisms by dropping data packets that pass through them. These attacks are very dangerous since they can considerably reduce the availability of network services.

The problem of blackhole and grayhole attacks has been widely debated and several solutions have been proposed [30, 32, 33]. A common features of these solutions is the exploitation of cooperation among network elements. Wireless networks, in fact, represent the best example of self-configured and self-managed system due to cooperation. Nodes continuously collaborate to offer network services since the system is extremely dynamic and there is no centralized coordination. As the nodes cooperate to provide routing services, similarly they might cooperate to identify and isolate the malicious elements by sharing information.

By embracing this principle, we proposed an integrated solution that includes the peculiarities of cooperative systems presented in the previous work. The idea is to extend the mechanisms and the results proposed for cooperative IDS in the context of wireless network security in order to strengthen the goodness of the cooperative solution presented.

The idea is to include mechanisms for cooperation into the routing protocol, in order to provide a new secure protocol. We will exploit the principle of REFACING to design a reputation model that associates a "trustworthiness" label with each node. These labels are used to avoid routes that include malicious nodes. The definition of trustworthiness requires to exchange evidence of anomalous activities of nodes. Activities detection, data exchange, information fusion and the reactive mechanisms constitute the basis for our cooperative solution.

The variety of technologies and solutions designed for wireless networks suggested us to focus on a specific type of wireless networks and then extend the principles to all the other environments. We decided to test the effectiveness of our approach on a specific wireless technology, namely the Wireless Mesh Networks (WMNs). A WMN represents an emerging solution in the

wireless scenario. Due to its commercial success, WMNs are obviously becoming the favorite target of attacks, and a network inefficiency might have serious economical and social impacts.

In the following section we will first provide some details about the WMNs and then we will introduce our solution for a secure routing protocol based on cooperative approach.

6.2 Wireless Mesh Networks

WMN is a key technology in the emerging context of wireless networks scenario [60]. More and more deployments are currently available: campus and company networking, community and neighborhood Internet access, building networks represent only some examples of the spreading of this technology. Due to its flexibility, Internet providers, government agencies, no-profit corporations are developing WMNs to allow network access in areas where wired solutions are too expensive to deploy, as for example low-income and under resourced communities[1] [61].

The main reason of its success is the low cost of technologies. Conventional nodes such as desktops, laptops, PDAs, phones, Access Points equipped with wireless interfaces can be included in a WMN deployment. Furthermore, developing a WMN is not so difficult since it can exploit the protocols designed for ad-hoc networks, from which a wireless mesh network come.

As in ad-hoc networks, indeed, WMN nodes can exchange data, although they are not in the transmission range of each other, by means of a multi-hop communication made possible by intermediate nodes.

By deploying gateways and bridges, a WMN is capable to interconnect different networks such as mobile ad-hoc networks and cellular networks, and to connect itself to the Internet.

Moreover, due to its self-organizing and self-configured nature, a WMN

[1] http://www.techforall.org/tfa_wireless.html

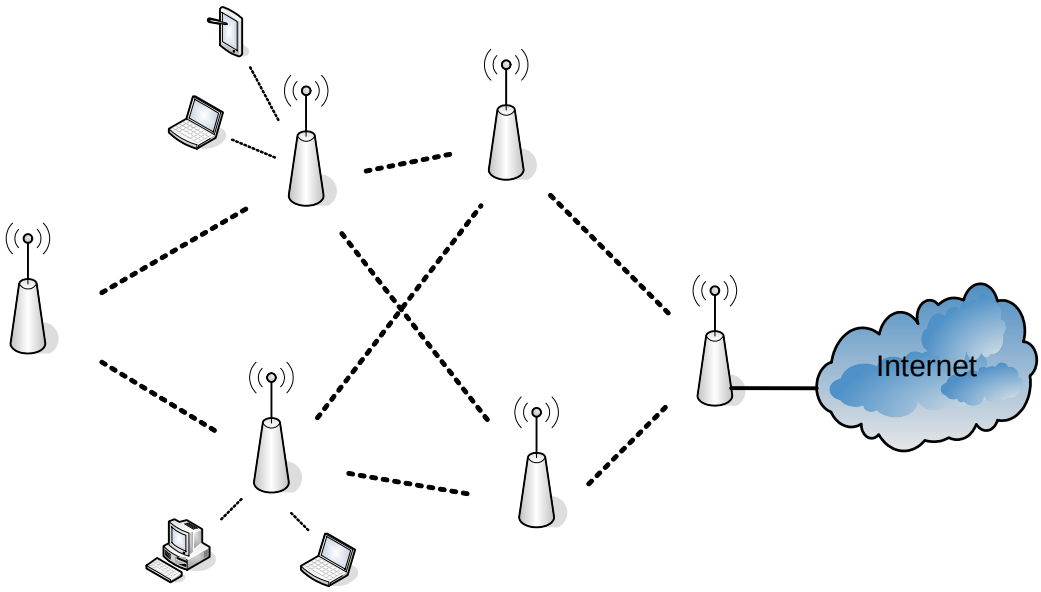


Figure 6.1: A typical WMN deployment

can easily grow in size since adding a new node do not require to reconfigure the entire network. The more are the WMN nodes the more is the effectiveness of the network.

A typical WMN consists of two types of nodes (see Figure 6.1): mesh routers and mesh clients. Mesh routers are fixed nodes, and they provide a backbone infrastructure by means of routing protocols. Every mesh router is equipped with one or more wireless interfaces, which may also implement different medium access control (MAC) protocols. Mesh clients may be either fixed or mobile. They are terminal hosts of the network and do not provide routing functionality. The technologies adopted for both clients and routers may be the same.

Based on this classification, a WMN architecture usually consists of two connection tiers. An access tier that allows the client to connect to a mesh router in the *Infrastructure Architecture*, and a backhaul tier for the interconnection of mesh routers.

In order to reduce the interference, the two tiers use different channels or implement different MAC protocols. Anyway it is common to observe WMNs

using the same radio technology for both access and backhaul tiers in order to reduce the cost of deployment. Unfortunately this approach increases the interference and reduces the overall throughput since clients and routers contend for the same channel.

As stated above, since WMNs share common features with ad-hoc networks, usually routing protocols developed for these networks can also be exploited for WMNs. Several routing protocols are adopted for WMNs such as the active protocols DSDV (Destination-Sequenced Distance Vector) [62], AODV (Ad-hoc On-demand Distance Vector) [63], DSR (Dynamic Source Routing) [64], or the proactive ones such as OLSR (Optimized Link State Routing) [65].

Despite the availability of several solutions for ad-hoc networks, routing for WMNs is still an open research field. The mentioned protocols were designed for satisfying the requirements of ad-hoc networks that are different from those of WMNs. In particular some issues related to the security or the optimization are neglected.

6.3 A Secure Routing Protocol: AODV-REX

As anticipated above in the chapter, we want to present a cooperative solution for secure routing in the context of wireless mesh networks.

In chapter 3, we described the design requirements needed to implement a cooperative system. In this section we briefly describe the "ingredients" required to develop our system.

Firstly, a mechanism that is in charge to observe other nodes and to discriminate between legitimate and illegitimate behavior is needed. Similar to the IDS presented in chapter 4, this mechanism must detect illegal activities of a set of nodes.

Another fundamental mechanism is the wide dissemination of behaviors' data into the network. This opens several issues. Since the messages exchanged introduce an overhead, it is important to define a protocol for dissemination that limits the interference with the normal network operations.

The reporting time also represents a challenge for a spreading protocol.

As observed for the cooperative IDS network, once information reaches nodes we have the problem to define the best solution that, by fusing data, can take the maximum advantage for the dissemination process.

Finally, a reaction mechanism is needed to exploit the result of cooperation.

From an implementation point of view, we have two choices for designing a cooperative mechanism: implementing the system from scratch, or exploiting existing mechanisms and protocols for providing the cooperation. Since information exchange is the core of any collaborative system, the latter seems to be the best solution in order to reduce the overhead provided by a new protocol.

In the context of routing protocols, we observe a close correlation between the activities of route selection and the operation for cooperative security. This suggests to merge the two mechanisms together, in order to improve the overall effectiveness of the routing protocol.

Although it seems the best solution for facing the problem of secure routing, unfortunately this integrated approach requires a great effort for designing an entirely new protocol. An alternative solution is to exploit existing routing protocols ad extend it with a cooperative mechanism that enforces the security of the protocol.

Based on this assumption we present an extension of a routing protocol for wireless networks, AODV, that introduces the principle of cooperative mechanisms for secure routing protocol. We called this new protocol AODV-REX (AODV Reputation EXtention).

We chose AODV since it is the most common protocol in WMNs. The choice of the protocol does not matter since our objective is to prove the effectiveness of the proposed cooperative approach.

According to our idea of collaborative system, this new approach is based on local information and global information. In particular, the approach exploits a local and global reputation of the nodes suggested by the REFAC-

ING reputation model. Similar to the solution proposed for the cooperative IDS, these two sources of information, properly exchanged and integrated, assure an improvement of the routing protocol security.

In the following we will provide several details about our new protocol. However, prior to introducing AODV-REX, it is necessary to briefly present the AODV protocol.

6.3.1 AODV

AODV [63] is an on demand protocol for Ad-hoc mobile network proposed by Perkins et al. [66]. It is a reactive protocol since a route is established only if explicitly requests for the source node. The route is maintained as long as it is needed, afterwards it is deleted. The reactive feature of this protocol is due to the need of an ad-hoc network to establish connection only when it needs, since the mobile nature of infrastructure does not require to maintain permanently the routes.

AODV supports both the unicast and multicast communications.

Although it have designed for Ad-hoc network, AODV is widely adopted also in wireless mesh.

The protocol defines three different UDP messages:

`RREQ` - **Route REQuest** This message is generated for discovering the path between a source, that requires it, and a destination.

`RREP` - **Route REPly** It is sent by the destination or intermediate nodes upon reception of a `RREQ`.

`RERR` - **Route ERRor** This message is sent to announce problems on an unreachable destination.

When a source needs to send data and a route to destination does not already available, the node generates a `RREQ` message with information on the destination that it want to reach (Figure 6.2).

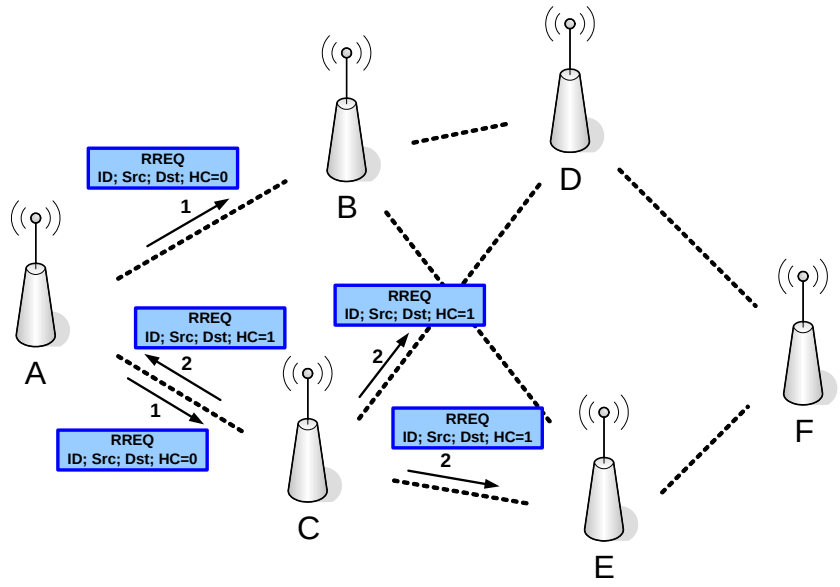


Figure 6.2: Route Request

Usually this happens if the route to destination is unknown, or a problem on the route occurs, or the destination is unreachable. The `RREQ` is sent in broadcast to all the neighbors. A related ID allows to identify univocally different requests. Furthermore a correct management of Sequence Number is needed in order to avoid loops in the route discovery process.

After broadcasting the request, the node waits for a `RREP` with information about the destination to reach until a timeout expires, afterwards a new `RREQ` is sent. In order to reduce the overhead due to unsuccessful requests, a binary exponential backoff mechanism is implemented for `RREQ` generation.

When a node receives a `RREQ` it checks if a request from the same IP source and with the same ID have been already received. In this case the request is dropped. Otherwise, if the node is not the destination demanded it creates or updates a reverse route, needed in case it will receive the `RREP` to send back to source node. Finally the node updates the fields of message, increments the Hop Count and broadcasts the message to the neighbors.

A `RREP` message is generated if, upon reception of the `RREQ`, node is itself the destination or it has an active route to the destination demanded. The

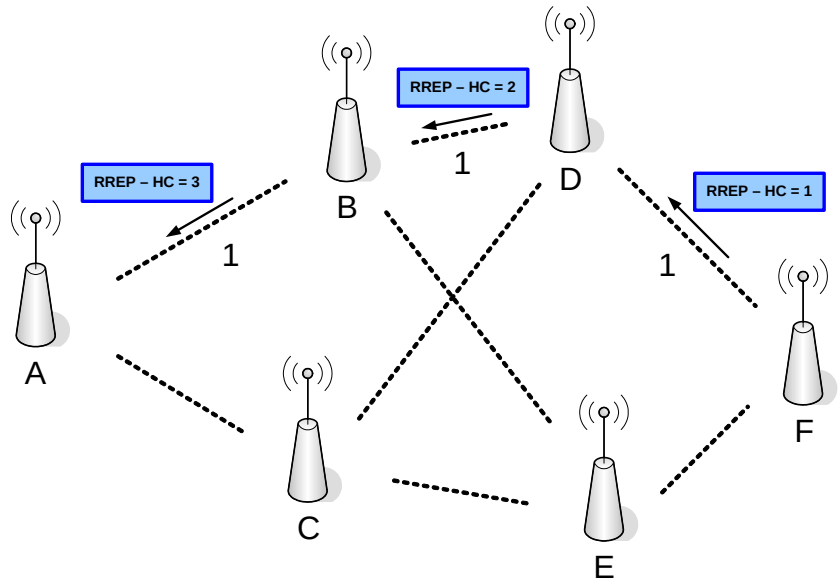


Figure 6.3: Route Reply

`RREP` is sent back in unicast through the next hop on the reverse path toward the `RREQ` generator. The hop count is set whether one, if the node is the destination requested, or the known distance in hops from the real destination (Figure 6.3).

When an intermediate node receives a `RREP` it creates a forward route to destination, in order to use it for the future data forwarding, increments the Hop Count and sends back it in unicast toward the source by the reverse route.

The `RRER` is generated for informing about a problem in routing process. In particular a node can send a `RRER` both in unicast and multicast to its neighbors if it detects a break for the next hop of an active forwarding route, or if it does not have an active route for data packets received, or if it receives a `RRER` from its neighbor.

As stated above, the routing table entries are temporary since the protocol is on-demand: any time a new route is activated a `Lifetime` is associated to it. A new process of request is needed when a route is not more activated.

A node may inform the neighbors about its connectivity by broadcasting

them an `HELLO` message. Such message should be sent exclusively by the nodes involved in an active route. If a node has not sent a broadcast message (e. g. `RREQ`) in a time interval, it broadcasts a `RREP` message with destination IP address its own address, the Hop Count 0, and a TTL 1. This assures neighbors drop the message upon reception.

A Local Repair process is provided by the protocol when a link break on an active route occurs. Locally the node upstream the break may decides to repair the route if the distance to destination is less a well-known value. It sends a new `RREQ` for the unreachable destination increasing the Sequence Number of request.

AODV does not present any solution for improve the security of route selection. A node that exhibits malicious behavior can not be excluded from route if it correctly exchanges routing information. The shortest path is selected even if a malicious node is included in it.

Our idea is to extend by AODV-REX the protocol providing it with a mechanism that based on cooperation are able to reduce the risks that a malicious node is selected in the route. In the following we describe this new protocol.

6.3.2 The Watchdog

In this section we introduce the first of components that integrate the AODV protocol: the *watchdog*.

The watchdog is the module that is in charge of control the behavior of neighbors. According to the classification of the system proposed in chapter 2, the watchdog could be consider a intrusion detection module, since it discovers neighbors that exhibit anomalous behavior.

Based on the model proposed by Marti et al. in [30], watchdog observes if the neighbors complete the forwarding process required when data packets are sent to them. This is possible since the wireless channel is a broadcast medium; if the reception and transmission ranges are the same, every forwarding activities of a node is sensed by all its neighbors. In order to allow

the watchdog implementation it needs to set the interface in promiscuous configuration.

The watchdog works directly as an extension of forwarding process provided by the routing algorithm. Every time data packet is sent to next node on the route, and if this node is not the final destination, the watchdog store information about the packet in a buffer and actives a timer. The information stored are usually enough to discriminate among different packet sensing on the interface. Usually the buffered data are related to destination of packet, the node at which the packet is sent, and the packet ID.

Based on the neighbors' behavior the watchdog update the reputation values of them. Two are the event that generate the reputation update. If the node senses a packet forwarded by its neighbor and this is present in the buffer a positive result to its activity is assigned and the data in buffer are deleted. Otherwise, if a timeout for a packet stored occurs, the watchdog provides to assign the neighbor a negative observation results.

The observations are the base for determining the value of direct reputation provided by node for its neighbors. A queue of N observed events is implemented in order to estimate node reputation that takes into account past interaction between node and its neighbor. The main objective of this solution is to introduce an "history" of events that could contribute to mitigate sporadic misbehavior which does not due to malicious attack.

Forwarded routing protocol data and broadcast packets are not buffered since no useful information to determinate the behavior of neighbors can be extracted from them. In particular, malicious node could forward routing protocol packets and drop data packets in order to be involved anyway in route establish process: it could attract data packets for dropping them later.

As stated above, the events observed by the watchdog are local to node, so only a local reputation can be evaluate from them.

A more sophisticated process have to be implemented to provide a global reputation of every node in the network. By a global reputation the nodes can have a better view of the situation and they can provide a more effective

routing process.

In the next section we will propose our solution for exchange information on reputation in order to improve the capability to prevent ineffectiveness in routing algorithm.

6.3.3 The Dissemination Protocol

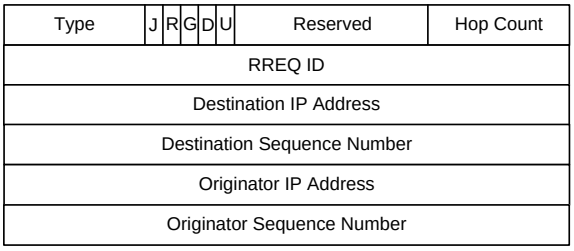
In this section we introduce the mechanism provided by AODV-REX for spreading information about the local observation into the network. This is fundamental for defining the global reputation of node as we will see in the section 6.3.4.

The main problem for a cooperative system is how to disseminate the local information all around the partners in order to allow them to use information to reinforce the opinion about a specific situation. Furthermore, we observe that in wireless environment a selective dissemination process can not be exploited due to the broadcast nature of the channel. For this reason the solution for spreading data on local observation requires that information must be carefully selected in order to avoid that malicious node can take advantage from knowing the "opinion" of its neighbors.

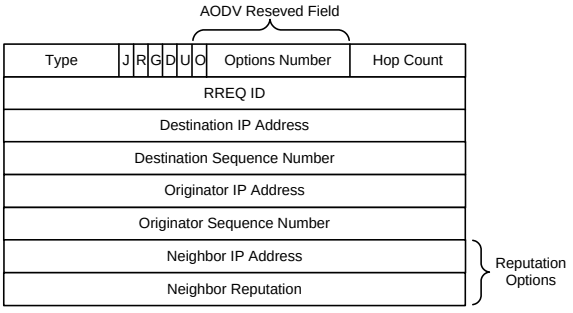
Related to this problem we propose to propagate exclusively a reputation that is a merging of local reputation and the reputations of other nodes in the network. In this way the malicious node can not identify exactly its neighbor that has begun to propagate bad reputation on it, and so it can not take adequate countermeasures. In the section 6.3.4 we will provide details about the merging of local reputation and the reputation coming from other nodes.

The information about the reputation are encapsulated in `RREQ` message of the AODV protocol. In particular an "option" extension, called `Reputation Option`, is applied to normal `RREQ` message. For each neighbors the node inserts a `Neighbor IP` address and a `Neighbor Reputation` (Figure 6.4(b)).

In order to allow nodes that implement AODV-REX to work in conjunction with nodes that do not support the reputation extension we can exploit



(a) AODV



(b) AODV-REX

Figure 6.4: RREQ massage format

the `Reserved` field in the `RREQ` message. This field is usually sent as 0 and it is ignored on reception. By exploiting this field we can differentiate the processing of `RREQ` from the protocol. Since AODV does not use `Reserved` bits, we can inform about the presence of `Reputation Option` with the first bit in the field, and with the other 10 bits we can number the `Reputation Option` (Figure 6.4).

According to the AODV protocol, every time a node has data to send it generates a `RREQ` message. Together with the usual AODV information, our protocol fits the message with reputation on its neighbors. The `RREQ` is broadcasted all the neighbor. The neighbors that receive the `RREQ`, it checks for the presence of `Reputation Option` by `Reserved` field and retrieves information in the `RREQ`. If the a node present in `Reputation Option` is not a neighbor for receivers, this descartes the information leaving unmodified it in `RREQ`. Otherwise it exploits the reputation value, that can be consider as a summarization of trustworthiness on a node from all the other nodes in

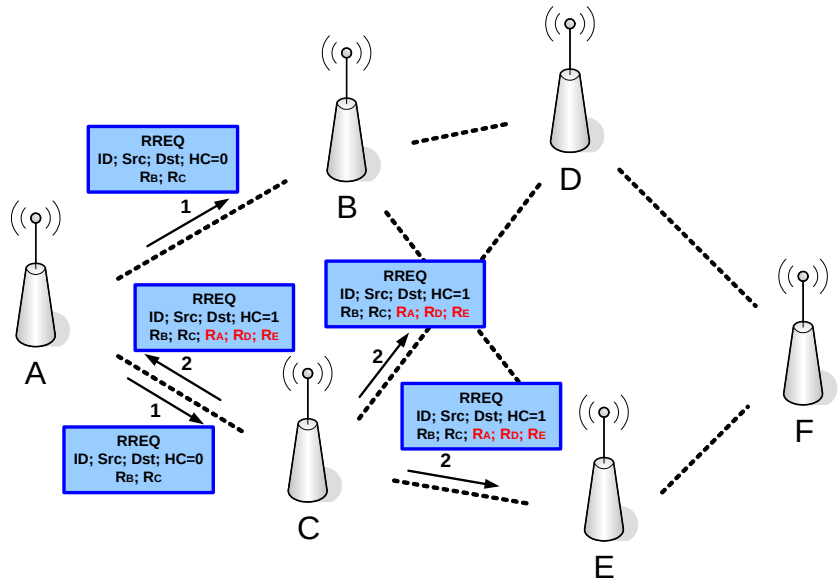


Figure 6.5: RREQ process in AODV-REX

the network, to update with its local observation its new reputation value. This new value is then fitted in the `RREQ` with together reputation values of neighbors do not already present in the message. Finally this new message is broadcasted all its neighbors (Figure 6.5).

This process assures that all the nodes in the network concord on the reputation of every single node. Anyway a process that guarantees bed information does not propagate in through the nodes is needed. In section 6.3.4 we will provide a mechanism that assures such requirement.

6.3.4 The Reputation Extraction Process

The reputation extraction is the core of our proposal. As widely demonstrated, data dissemination can contribute to improve the effectiveness of a system, but it is important that data exchanged are correctly handled in order to maximize the benefit coming form a cooperative approach.

In this section we will deal with the problem of merge local and global information.

As announced above in the chapter, ADOV-REX is based on two reputa-

tion levels which represent respectively the global and the local information: a *local reputation*, coming from the observations provided by the watchdogs, and a *global reputation* supplied by other nodes through dissemination protocol. Both local and global reputation are merged to define the reputation that can be exploit to determinate the real behavior of node.

The problem of information merging can be described as follows.

Let $R_{B_A}(i)$ be the reputation of node B at node A at $i-th$ iteration, while $R^G_{B_A}(i)$ represents the global reputation of node B at node A at $i-th$ iteration. Anytime node A receives a new reputation value $R_{C_B}(i)$ on node C is sent by node B through the `RREQ` message, the node A first weights this information with the reputation $R_{B_A}(i-1)$ that it has with regard to B, in order to reduce the risk of bad news propagated from malicious nodes. This new information is then compared with the last reputation value on node C at node A

$$R^w_{C_B}(i) = R_{B_A}(i-1) * R_{C_B}(i) \tag{6.1}$$

$$\Delta_{C_{AB}} = |R^G_{C_A}(i-1) - R^w_{C_B}(i)| \tag{6.2}$$

The new global reputation of node C at node A is

$$R^G_{C_A}(i) = \frac{1}{2} * (1 + \Delta_{C_{AB}}) * R^G_{C_A}(i-1) + \frac{1}{2} * (1 - \Delta_{C_{AB}}) * R^w_{C_B}(i) \tag{6.3}$$

Local and global reputation extraction are supported by a mechanism that stores the recent events, both direct observations by watchdogs and global reputation by other nodes, and on these computes the current reputation values (see 6.3.2). Local and global reputations are extracted as a mean of recent samples stored.

The mean formula exploits an approach that allows to node a "reintegration" if it has shown a faultless behavior recently. The longer is the time from the last misbehavior the less is the weight in the average. For this reason we

implement a *Weighted Moving Average* WMA. Let w_{lr} be the weight of contribution that we want assign the most recent value. We have the following formula for the mean:

$$\bar{R}_{B_A} = w_{lr} * \sum_{i=1}^{N-1} (1 - w_{lr})^{i-1} * R_{B_A}(i) + (1 - w_{lr})^{N-1} * R_{B_A}(N) \qquad (6.4)$$

Now, according to REFACING model, the reputation extraction process is implemented as follows. Whenever a new event happens, for example a `RREP` message is received or a watchdog timer expires, we need to update the reputation of the neighbor. We adopt an increment or decrement Δ_{B_A} of the mean of previous reputation values $\bar{R}_{B_A}$

$$R_{B_A}(i) = \bar{R}_{B_A}(i-1) + \Delta_{B_A}(i-1) \qquad (6.5)$$

In spite of the local and global reputation average defined by (6.4), for the reputation we adopt a simple mean of all the previous reputation samples.

In order to set the value of Δ_{B_A} we need to evaluate the agreement between the global reputation and the local one of node B at node A. Let us define $\bar{R}^G_{B_A}$ e $\bar{R}^L_{B_A}$ the averages respectively of global and local reputation. We compare this two value to evaluate the "distance" between them.

$$\Delta^{LG}_{B_A} = |\bar{R}^L_{B_A} - \bar{R}^G_{B_A}| \qquad (6.6)$$

Clearly global and local information agree if $\Delta^{LG}_{B_A}$ is lower then a well-known threshold T, for example 0.5, otherwise they disagree. So in the first case we adopt an positive value for Δ_{B_A}

$$\Delta_{B_A} = \Gamma * (1 - \bar{R}_{B_A}) * \frac{(1 - \Delta^{LG}_{B_A})}{2} \qquad (6.7)$$

If local and global disagree we observe that it can be a good solution reduce the reputation since the overall system reputation of node B presents some anomalous behavior. In this case it is better adopt a "conservative" approach by setting negative value for Δ_{B_A}

$$\Delta_{B_A} = -\Gamma * \bar{R}_{B_A} * \frac{\Delta_{B_A}^{LG}}{2} \tag{6.8}$$

The value of Γ represents a weight that takes into account the reputation, the interaction degree ID between the node and its neighbor, and the variance of reputation:

$$\Gamma = \alpha * \bar{R}_{B_A} + \beta * ID + \omega * V_{B_A}^R \tag{6.9}$$

with:

$$\alpha + \beta + \omega = 1 \tag{6.10}$$

In this case we set the following values

$$\alpha = 0.6,\ \beta = 0.2,\ \omega = 0.2. \tag{6.11}$$

Once we compute the new value for reputation we have to update all the value for reputation mean $\bar{R}_{B_A}$ and variance $V_{B_A}^R$ adopting the following formulas:

$$\bar{R}_{B_A}(i) = \bar{R}_{B_A}(i-1) + \frac{R_{B_A}(i) - \bar{R}_{B_A}(i-1)}{N+1} \tag{6.12}$$

$$V_{B_A}^R(i) = \frac{N * V_{B_A}^R(i-1) + (R_{B_A}(i) - \bar{R}_{B_A}(i)) * (R_{B_A}(i) - \bar{R}_{B_A}(i-1))}{N+1} \tag{6.13}$$

The reputation value computed by 6.5 provides a unique vision of nodes behaviors since it is a merge of what is observed locally and what the other nodes observe. This information is the basement of our secure routing solution as will see in the next section.

This information is propagate to the nodes by the protocol described in section 6.3.3.

6.3.5 The Reputation and Route Selection

Once reputations have been propagated widely into the network, we need to project an effective mechanism by which the cooperation can be exploited for realize a secure routing protocol. The idea is to use nodes' reputation to influence the route selection process.

We observe that AODV choices the route based on hop count value: the path selected is the shortest path in term of hop count. Our idea is to modify the hop count values in order to carry also information about the nodes' reputations along the path. We want to substitute the idea of *shortest path first*, common to all routing protocol, with *"most trustworthy" path first* of our solution. We suggest to "lengthen" the path traversing nodes with bad reputation in order to avoid its selection.

As stated in the subsection 6.3.1 the length is defined during the reply process, by counting the number of hops traversed: every time a node receives the RREP message it increments the Hop Count value by one to account for the new hop through the intermediate node. Our idea is to define a new distance between two nodes that is not just a physical distance but a virtual distance that take into account the reputation level of the node connected to the link: the distance of two neighbor nodes increase by decreasing the reputation of one of them. For this reason we introduce a new link metric called *Reputation Metric* from node A to node B as follows:

$$RM_{\vec{AB}} = \lfloor (1 - R_{B_A}) * ND \rfloor \tag{6.14}$$

with R_{B_A} the reputation of the node connected at the link, and ND the `Network Diameter`, the maximum network diameter define by the AODV protocol. The Reputation Metric is added to physical distance.

By this solution, if reputation is 0 the metric is equal to the maximum diameter of the network, otherwise the metric RM is 0 every time reputation of my neighbor is maximum. The use of floor function allows us to be "indulgent" if rarely neighbor has a irregular behavior.

In the Figure 6.6 is depicted an example of AODV-REX reply process.

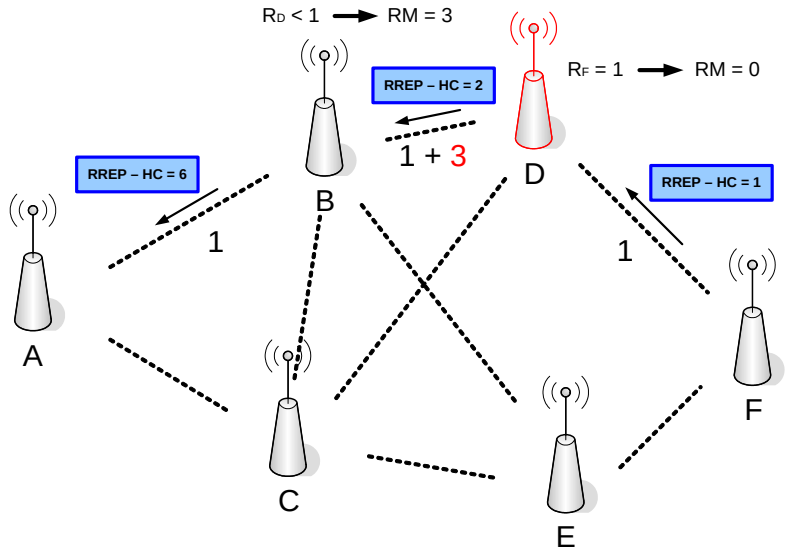


Figure 6.6: RREP process in AODV-REX

When a node generates a `RREP` it set the Hop Count one as in AODV. Upon the reception of `RREP` the AODV-REX protocol checks in its `Reputation Neighbor Table` the value of reputation for the node that sent the `RREP`. Then it computes the RM value by 6.14, and adds this to Hop Count. It is further incremented in order to take in to account the physical hop. In this way if RM is 0 Hop Count is incremented similar to AODV protocol.

When the RREP reaches the source of request this evaluate the RM with regard to message sender, adds it to Hop Count in RREP and in case it starts to use the route for sending data packets.

Again we observe that the AODV-REX is designed in order to work in conjunction with AODV node: no modifications at `RREP` message format are needed, so an AODV-REX `RREP` can be handled also by AODV protocol.

The solution to increment the distance by RM in `RREP` process assures an intrinsic security of the protocol: by adopting this mechanism a subverted node can not modify the distance since the the RM is added by downstream nodes. It could modify the distance of `RREP` message received but anyway its reputation is reflected in the RM computed by downstream node.

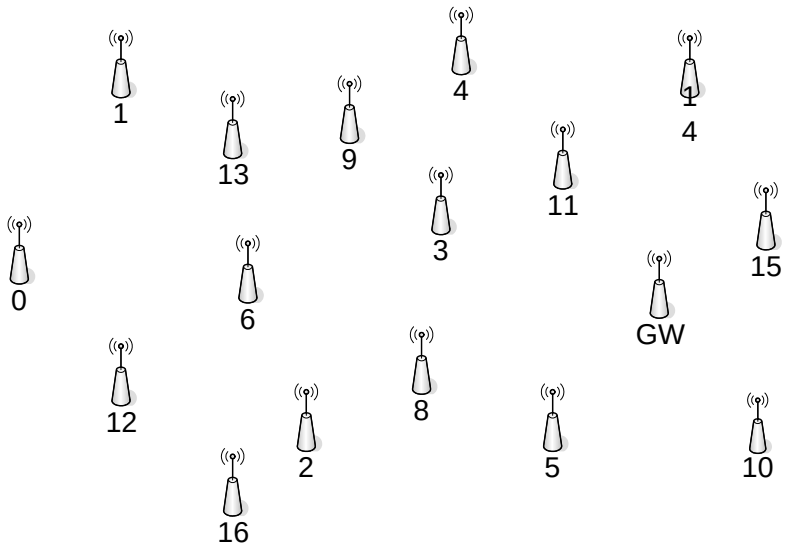


Figure 6.7: Experimental Topology

6.4 Experimental Results

In this final section we want to provide an evaluation of performance of protocol proposed. We implemented the protocol as module of well-known Network Simulator `ns2`[2] [67].

The scenario simulated consists of a random network topology of 17 WMN nodes (Figure 6.7). The *GW* node is the gateway to Internet. Each node is equipped with a single 802.11 interface with an omnidirectional antenna.

The malicious nodes are implemented by modifying in the code the forwarding process. We designed a *malicious bursting node* where burst of packets are dropped with probability of 0.3. The burst length is 20 packets. The packets dropping is apply exclusively to data packets since we want that also the malicious node are involved in the routing protocol operations. It allows subverted nodes to be potentially included in routes.

We compare the AODV-REX and AODV. The choice of parameters to evaluate is very important. We analyze exclusively the throughput. We claim that the throughput represent a parameter that effectively synthesizes

[2]http://www.isi.edu/nsnam/ns/

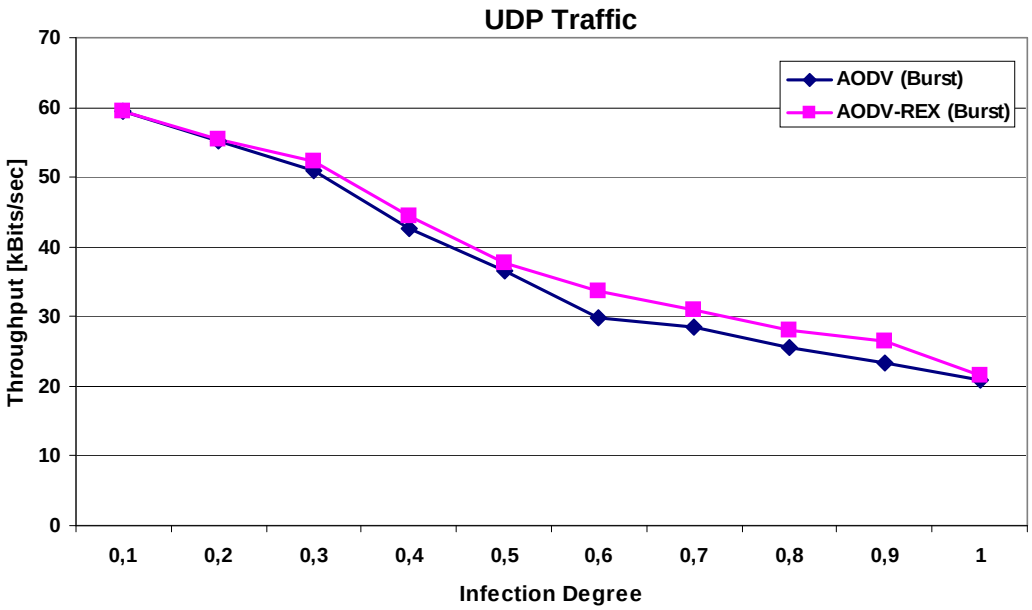


Figure 6.8: UDP Throughput

the improvement of our solution. The objective of malicious node is to deny the possibility for users to fully exploit the network services: by dropping packet a node can decrease the overall throughput of network. Our idea is to prove that by adopting AODV-REX instead of AODV we can optimize the routing functionality, since ineffectiveness due to subverted node can be properly faced.

So we compared throughput of network both with AODV and AODV-REX, first with UDP traffic, then with TCP traffic. We consider 6 periodic flows from several sources to the single gateway *GW* of our topology. The choice to adopt periodic flow is required in order to allow the information on nodes' reputation to spread widely into the network.

Furthermore, we increased the numbers of nodes infected in order to compare the behavior of AODV and AODV-REX in critical conditions.

In the graph 6.8 we depicted the trend of UDP throughput by increasing the infection degree of the nodes. The UDP flows have a bit rate of 64 kbit/sec.

Generally we observed an improvement in the performance of AODV-

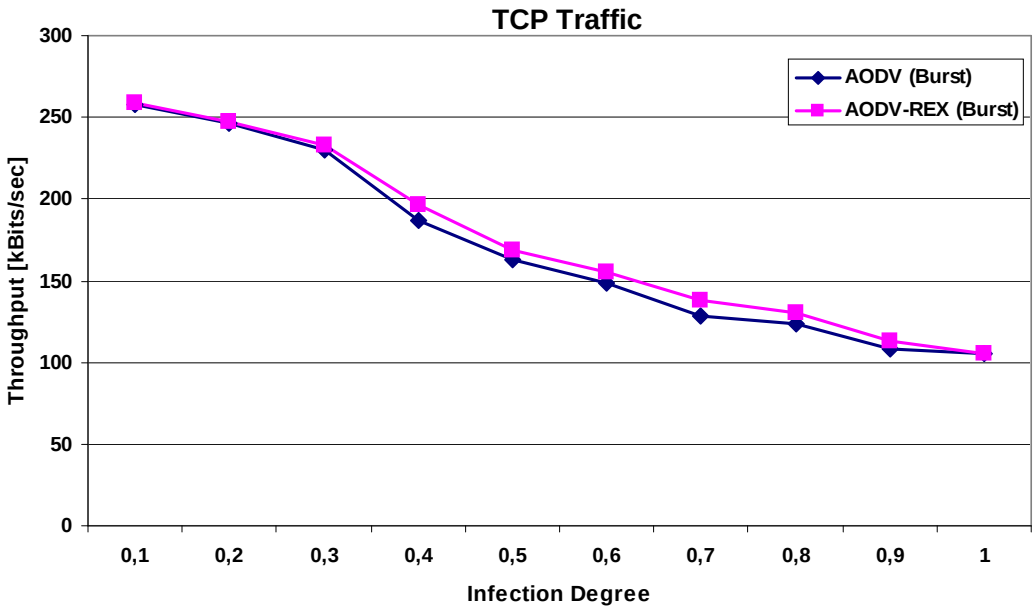


Figure 6.9: TCP Throughput

REX with respect to AODV, except for low value of infection and for a full network infection. As we have supposed, we observe a greater gain for value of infection degree between 0.5 and 0.9. The reason is due to the grater probability of AODV-REX to select "secure path" related to AODV: when the infection degree is low the probability the AODV selects a bad path is low, comparable with the AODV-REX one. Increasing the number of nodes infected the probability to select paths passing through compromised nodes increases. No gain can be observed when the network is completely infected.

In the graph 6.9 the results of the TCP throughput for both the AODV and AODV-REX are shown.

Again we observe a gain in the performance of AODV-REX compare to AODV. In this case the advantage to adopt our solution is lower then for UDP traffic. The reason is that TCP protocol is connection oriented protocol so mechanisms for reliable communications can limit the effects of malicious packet dropping.

As observed for the UDP throughput, the greater performance gain is present for a value of infection degree between 0.4 and 0.5.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis we have dealt with security issues in computer networks. Security in networked system is a critical challenge since the people more and more relies on services provided by the network.

This problem is widely debated by research communities and several solutions have been proposed. Unfortunately the heterogeneity of infrastructure, technologies, services and users make extremely complex to propose a global approach to security problems.

An alternative is represented by the possibility to adopt different solutions and integrate them through a coordinated and collaborative approach.

In this context we offered our original contributions by investigating the opportunities offered by a cooperative approach for network security based on distributed information.

We analyzed the main requirements needed to design a system exploiting such approach. Mechanisms for data dissemination with together the demand of effective algorithms for combining them represent the main challenges that we had to deal with.

Closely connected with the problem of information fusion is the issue to avoid that bad information might compromise the effectiveness of data sharing. Some information, in fact, could be deliberately or casually altered

producing a decrease in system performance.

Based on these assumptions, we proposed our solutions for prevent and detect attacks to computer networks. In particular we focused on two type of threats, both related to Denial of Service attacks: the volume-based and the wireless protocol routing attacks.

By adopting a general framework for cooperative approach based on distributed information, we designed a distributed intrusion detection system and a new secure routing protocol for wireless network. Both solutions share the some design principles. This confirms the goodness of framework proposed since it is possible to adopt it in different security context.

Furthermore we defined a new model for reputation that we applied successfully to both the context analyzed. We observed that the reputation is fundamental for limiting both malicious activities and inefficiencies of single elements of distributed system.

The results showed that the cooperation exploiting distributed information can effectively improve the performance of security system in both wired and wireless network scenarios. In particular volume-based attacks can be adequately faced by integrating different elements in a distributed system that exploits the collaboration of its single parts to detect the attacks. At the same time the risks due to some specific class of DoS at routing protocol in a wireless scenario can also be limited by sharing evidence of anomalous behaviors and exploiting this information for building reputation of partners.

Moreover we proved that the new reputation model improves the capability of a distributed system to limit impact of misbehaving elements, as well as it can be exploited to successfully implement a new routing protocol and metric for a wireless mesh network.

7.2 Future Work

The results of our approach allow to be confident related to the possibility of extend it in several directions.

Firstly it might be possible analyze the effectiveness of solutions proposed

to face new DoS attacks. The independence of cooperation mechanisms from the local detection do not limit the typologies of DoS attacks that can be identified.

Moreover exploiting the framework's principles it might be possible to design new mechanisms for new type of attacks. Since the approach proposed is extremely generic, in fact, we could extend the cooperation also for emerging threats as botnets.

Finally, some improvements should be required for the centralized system that is in charge to evaluate the reputation of the partners in distributed IDS. A full distributed solution could be explored to increase its effectiveness and to reduce the risks coming from a single point of failure.

Bibliography

[1] Shanthi Kalathil and Taylor Boas. *Open Networks, Closed Regimes: The Impact of the Internet on Authoritarian Rule*. Washington, DC: Carnegie Endowment for International Peace, 2003.

[2] C. De Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. Rfc 2903 - generic aaa architecture. Internet draft, IETF, Jenuary 2000.

[3] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[4] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defence mechanisms. *ACM SIDCOMM Computer Communication Review*, 34(2):39–53, 2004.

[5] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23–24):2435–2463, December 1999.

[6] G. Vigna and R. Kemmerer. Netstat: a network based intrusion detection system. *Journal of Computer Security*, 7(1), 1999.

[7] D. Andersson. Detecting usual program behavior using the statistical component of the next-generation intrusion detection expert system (nides). Technical report, Computer Science Laboratory, 1995.

[8] M. Tyson. Derbi: Diagnosys explanation and recovery from computer break-ins. Technical report, 2000.

[9] B. Laing and J. Alderson. How to guide - implementing a network based intrusion detection system. Technical report, Internet Security Systems, Sovereign House, 57/59 Vaster Road, Reading, 2000.

[10] Andrew R. Baker, Brian Caswell, and Mike Poor. *Snort 2.1 Intrusion Detection - Second Edition*. Syngress, 2004.

[11] Marcello Esposito, Claudio Mazzariello, Francesco Oliviero, Simon Pietro Romano, and Carlo Sansone. Real time detection of novel attacks by means of data mining techniques. In *Proceedings of 7th International Conference of Enterprise Information Systems, ICEIS*, May 2005.

[12] Marcello Esposito, Claudio Mazzariello, Francesco Oliviero, Simon Pietro Romano, and Carlo Sansone. Evaluating pattern recognition techniques in intrusion detection systems. In *Proceedings of the 5th International Workshop on Pattern Recognition in Information Systems, PRIS 2005*, May 2005.

[13] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):227–261, November 2000.

[14] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu. Adam: Detecting intrusion by data mining. In *Workshop on Information Assurance and Security*, pages 11–16. IEEE, 2001.

[15] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, pages 37–52, 1996.

[16] W. W. Cohen and Y. Singer. A simple, fast, and effective rule learner. 1999.

[17] C. Elkan. Results of the kdd99 classifier learning. In *SIGKDD Explorations*, volume 1, pages 63–64. ACM, 2000.

[18] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transaction on Dependable and Secure Computing*, 1(3):146–169, July–September 2004.

[19] Jelena Mirkovic, Max Robinson, Peter Reiher, and Geoff Kuenning. Alliance formation for ddos defense. In *Proceedings of New Security Paradigms Workshop, ACM SIGSAC*, August 2003.

[20] Christos Papadopulos, Rober Lindell, John Mehringer, Alwfiya Hussain, and Ramesh Govindan. Cossack: Coordinated suppression of simultaneous attacks. In *Proceedings of the DARPA Information Survivability Conference and Exposition, IEEE DISCEX*, 2003.

[21] Guangsen Zhang and Manish Parashar. Cooperative defence against ddos attacks. *Journal of Research and Practice in Information Technology*, 38(1):69–84, February 2006.

[22] Angelos D. Keromytis, Janak Parekh, Philip N. Gross, Gail Kaiser, Vishal Misra, Jason Nieh, Dan Rubenstein, and Sal Stolfo. A holistic approach to service survivability. In *Proceedings of First ACM Workshop on Survivable and Self-Regenerative Systems, SSRS*, October 2003.

[23] Domenico Cotroneo, Lorenzo Peluso, Simon Pietro Romano, and Giorgio Ventre. An active security protocol against dos attacks. In *Proceedings of Seventh International Symposium on Computers and Communications, ISCC*, 2002.

[24] Vinod Yegneswaran and Paul Barford Sommesh Jha. Global intrusion detection in the domino overlay system. In *Proceedings of 11th Annual Symposium on Network and Distributed System Security, NDSS*, February 2004.

[25] Georgios Koutepas, Fotis Stamatelopoulos, and Basil Maglaris. Distributed management architecture for cooperative detection and reaction to ddos attacks. *Journal of Network and Systems Management*, 12(1):73–94, March 2004.

[26] H. Debar, D. Curry, and B. Feinstein. Rfc 4765 - the intrusion detection message exchange format (idmef). Internet draft, IETF, March 2007.

[27] Naouel Ben Salem and Jean-Pierre Hubaux. Securing wireless mesh network. *IEEE Wireless Communications*, 13(2):50–55, April 2006.

[28] Asad Amir Pirzada, Chris McDonald, and Amitava Datta. Performance comparison of trust-based reactive routing protocols. *IEEE Transaction on Mobile Computing*, 5(6):695–710, June 2006.

[29] Yih-Chun Hu and Adrian Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, 2(3):28–39, June 2004.

[30] Sergio Marti, T.J. Guili, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of 6th Annual International Conference on Mobile Computing and Networking, MobiCom*, August 2000.

[31] David B. Johnson, David A. Maltz, and Yih-Chun Hu. Rfc 2026 - the dynamic source routing protocol for mobile ad hoc networks (dsr). Internet draft, IETF, April 2003.

[32] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the confidant protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks). In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing, MobiHOC*, June 2002.

[33] Pietro Michiardi and Refik Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad-hoc networks. In *Proceedings of Communications and Multimedia Security Conference, CMS*, September 2002.

[34] Robert M. Fano. *Transmission of Information: A Statistical Theory of Communications*. MIT Press and John Wiley and Sons, New York, 1961.

[35] Salvatore D'Antonio, Marcello Esposito, Francesco Oliviero, Simon Pietro Romano, and Dario Salvi. Behavioral network engineering: making intrusion detection become autonomic. *Annales des Telecommunications/Annals of Telecommunications*, 61(9–10):1139–1151, 2006.

[36] David Geer. Behavior-based network security goes mainstream. *IEEE Computer*, 39(3):14–17, March 2006.

[37] Michael Smirnov. Autonomic communication - research agenda for a new communication paradigm. White paper, Fraunhofer FOKUS, Institute for Open Communication Systems, November 2004.

[38] Claudio Mazzariello, Francesco Oliviero, Salvatore D'Antonio, and Dario Salvi. A distributed multi purpose ip flow monitor. In *Proceedings of 3th International Workshop on Internet Performance, Simulation, Monitoring and Measurement, IPS MoMe*, March 2005.

[39] J. McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294, November 2000.

[40] V. Paxson and S. Floyd. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, 2001.

[41] M. Mahoney. *A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic*. PhD thesis, Florida Istitute of Technology, 2003.

[42] W. W. Cohen and Y. Singer. Simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conferance an Artificial Intelligence*, pages 335–342. AAAI Press / The MIT Press, 1999.

[43] R. Meir and G. Ratsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures an Machine Learning*, pages 119–184. Springer Verlag, 2003.

[44] S. Axelsson. *The base-rate fallacy and the difficulty of intrusion detection*, volume 3 of *ACM transaction on information and system security*, pages 186–205. ACM, August 2000.

[45] Claudio Mazzariello and Francesco Oliviero. An autonomic intrusion detection system based on behavioral network engineering. In *Proceedings of 2nd IEEE INFOCOM Student Workshop*, April 2006.

[46] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks. In *USENIX/ACM Symposium on Network Systems Design and Implementation*, 2004.

[47] Jonathan W. Hui and David Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *2nd ACM Conference on Embedded Networked Sensor Systems*. ACM Press, 2004.

[48] Philip Levis and David Culler. The firecracker protocol. In *11th ACM SIGOPS European Workshop*. ACM Press, 2004.

[49] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[50] Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of ACM MobiCom*. ACM, 2000.

[51] Daniel J. Burroughs, Linda F. Wilson, and George V. Cybenko. Analysis of distributed intrusion detection systems using bayesian methods. In *International Performance Computing and Communication Conference*, April 2002.

[52] Ian Ruthven and Mounia Lalmas. Using dempster-shafer's theory of evidence to combine aspects of information use. *J. Intell. Inf. Syst.*, 19(3):267–301, 2002.

[53] D. Yu and D. Frincke. Alert confidence fusion in intrusion detection systems with extended dempster-shafer theory. In *Proceedings of the 43rd annual southeast regional conference, ACM Press*, pages 142–147, March 2005.

[54] H. Wu, M. Siegel, R. Stiefelhagen, and J. Yang. Sensor fusion using dempster-shafer theory. In *Proceedings of IEEE Instrumentation and Measurement Technology Conference*, 2002.

[55] Francesco Gargiulo, Claudio Mazzariello, and Carlo Sansone. A self-training approach for automatically labeling ip traffic traces. In *Book Computer Recognition Systems 2*, volume 45/2008 of *Advances in Soft Computing*. Springer Berlin / Heidelberg, 2008.

[56] E. Friedman K. Kuwabara P. Resnick, R. Zeckhauser. Reputation systems. *Communications of the ACM*, 43(12):45–48, December 2000.

[57] Sepandar D. Kamvar, Mario T. Schlosser, and Hector GarciaMolina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of WWW2003*, Budapest, Hungary, May 2003. ACM.

[58] Sonja Buchegger and Jean-Yves Le Boudec. Nodes bearing grudges:towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, EUROMICRO-PDP*, 2002.

[59] Leislie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[60] Ian F. Akyldiz, Xudong Wang, and Weilin Wang. Wireless mesh network: A survey. *Computer Networks*, 47(4):445–487, March 2005.

[61] Joseph Camp, Joshua Robinson, Christopher Steger, and Edward Knighly. Measurement driven deployment of a twotier urban mesh access network. In *Proceedings of ACM Fourth International Conference on Mobile Systems, Applications, and Services, MobiSys*, June 2006.

[62] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of ACM Conference on Communications Architectures, Protocols and Applications, SIGCOMM*, pages 234–244, 1994.

[63] C. Perkins, E. Belding-Royer, and S. Das. Rfc 3561 - ad hoc on-demand distance vector (aodv) routing. Internet draft, IETF, July 2003.

[64] D. Johnson, Y. Hu, and D. Maltz. Rfc 4728 - the dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4. Internet draft, IETF, Febraury 2007.

[65] T. Clausen and P. Jacquet. Rfc 3629 - optimized link state routing protocol (olsr). Internet draft, IETF, October 2003.

[66] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications, WMCSA*, pages 90–100. IEEE Computer Society, 1999.

[67] Kevin Fall and Kannan Varadhan. *The ns Manual*. A Collaboration between UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 2007.