Security monitoring for network protocols and applications

Vinh Hoa La

NNT : 2016SACLL006

THESE DE DOCTORAT
DE
L'UNIVERSITE PARIS-SACLAY
PREPAREE A
TELECOM SUDPARIS

ÉCOLE DOCTORALE N°580
Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique

Par

M. Vinh Hoa La

Security Monitoring for Network Protocols and Applications

Thèse présentée et soutenue à Evry, le 21 octobre 2016 :

Composition du Jury :

M, Farid NAIT-ABDESSELAM, Professeur, Université Paris Descartes (Paris V), Rapporteur
M. Marcelo DIAS DE AMORIM, Directeur de recherche, CNRS-LIP6-UPMC, Rapporteur
M. Patrick SENAC, Professeur, ENAC – Toulouse, Examinateur
Mme Fatiha ZAIDI, Maître de Conférences, HDR, Université Paris Sud, Examinatrice
M. Adrien BECU, Head of Research & Technology, Airbus DS Cybersecurity, Examinateur
M. Wissam MALLOULI, Dr., Ingénieur de recherche, Montimage, Examinateur
Mme Ana Rosa CAVALLI, Professeur, Telecom SudParis, Directeur de thèse

Titre : Monitorage des Aspects Sécuritaires pour les Protocoles de Réseaux et Applications.

Mots clés : sécurité, détection d'intrusion, surveillance de sécurité, supervision de réseaux

Résumé : La sécurité informatique, aussi connue comme la cyber-sécurité, est toujours un sujet d'actualité dans la recherche en sciences informatiques. Comme les cyber-attaques grandissent de plus en plus en volume et en sophistication, la protection des systèmes ou réseaux d'information devient une tâche difficile. Les chercheurs dans la communauté de recherche prêtent une attention constante à la sécurité, en particulier dans les deux directions suivantes: (i) - la conception des infrastructures sécurisée avec des protocoles de communication sécurisés et (ii) - surveillance / supervision des systèmes ou des réseaux afin de trouver et de remédier aux vulnérabilités. La dernière vérifie que tout ce qui a été conçu dans la première fonctionne correctement et en toute sécurité. Ceci étant le sujet principal de cette thèse.
Cette dissertation présente un cadre de surveillance de la sécurité en tenant en compte des différents types de jeu de données d'audit y compris le trafic de réseaux et les logs dans les applications. Nous proposons également des approches innovantes fondées sur l'apprentissage supervisé pour prétraiter et analyser l'entrée de données. Notre cadre est validé dans une large gamme de cas d'études, y compris la surveillance des réseaux traditionnels TCP / IP (v4) (LAN, WAN, la surveillance de l'Internet), la supervision des réseaux de objets connectés utilisant la technologie 6LoWPAN (IPv6), et également l'analyse des logs d'autres applications.
Dans chaque cas d'étude, nous décrivons comment nous collectons les jeux de données d'audit, extrayons les attributs pertinents, traitons les données reçues et décodons leur signification de sécurité. Pour ces objectifs, l'outil Montimage Monitoring Tool (MMT) est utilisé comme le cœur de notre approche. Nous évaluons également la performance de la solution et sa possibilité de marcher dans les systèmes "à plus grande échelle" avec les jeux de données plus volumineux.

Title : Security Monitoring for Network Protocols and Applications.

Keywords : security, intrusion detection, security monitoring, network monitoring

Abstract: Computer security, also known as cyber-security or IT security is always an emerging topic in computer science research. Because cyber-attacks are growing in both volume and sophistication, protecting information systems or networks becomes a difficult task. People in research community give an ongoing attention in security including two main directions: (i) - designing secured infrastructures with secured communication protocols and (ii) - monitoring/supervising the systems or networks in order to find and remediate vulnerabilities. The former can assist the later by forming some additional monitoring-supporting modules. Whilst, the later verifies whether everything designed in the former is correctly and securely functioning. This is the main topic of this thesis.
This dissertation presents a security monitoring framework that takes into consideration different types of audit data set including network traffic and applications' exchanged messages. We propose also some novel approaches based on supervised machine learning to pre-process and analyze the data input. Our framework is validated in a wide range of case studies including traditional TCP/IP(v4) network monitoring (LAN, WAN, Internet monitoring), Internet of Things (IoT) using 6LoWPAN technology (IPv6), and other applications' logs. In each case study, we describe how we collect the audit data set, extract the relevant attributes, handle received data and decode their security meaning. For these goals, Montimage Monitoring Tool (MMT) is used as the core of our approach. We assess also the solution's performance and its possibility to work in "larger scale" systems with more voluminous datasets.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Abstract

Computer security, also known as cyber-security or IT security, is always an emerging topic in computer science research. Because cyber attacks are growing in both volume and sophistication, protecting information systems or networks becomes a difficult task. Therefore, researchers in research community give an ongoing attention in security including two main directions: (i) - designing secured infrastructures with secured communication protocols and (ii) - monitoring/supervising the systems or networks in order to find and re-mediate vulnerabilities. The former assists the later by forming some additional monitoring-supporting modules. Whilst, the later verifies whether everything designed in the former is correctly and securely functioning as well as detecting security violations. This is the main topic of this thesis.

This dissertation presents a security monitoring framework that takes into consideration different types of audit dataset including network traffic and application logs. We propose also some novel approaches based on supervised machine learning to pre-process and analyze the data input. Our framework is validated in a wide range of case studies including traditional TCP/IPv4 network monitoring (LAN, WAN, Internet monitoring), IoT/WSN using 6LoWPAN technology (IPv6), and other applications' logs. Last but not least, we provide a study regarding intrusion tolerance by design and propose an emulation-based approach to simultaneously detect and tolerate intrusion.

In each case study, we describe how we collect the audit dataset, extract the relevant attributes, handle received data and decode their security meaning. For these goals, the tool Montimage Monitoring Tool (MMT) is used as the core of our approach. We assess also the solution's performance and its possibility to work in "larger scale" systems with more voluminous dataset.

Résumé

La sécurité informatique, aussi connue comme la cyber-sécurité, est toujours un sujet d'actualité dans la recherche en sciences informatiques. Comme les cyber-attaques grandissent de plus en plus en volume et en sophistication, la protection des systèmes ou réseaux d'information devient une tâche difficile. Les chercheurs dans la communauté de recherche prêtent une attention constante à la sécurité, en particulier ils s'orientent vers deux directions principales: (i) - la conception des infrastructures sécurisées avec des protocoles de communication sécurisés et (ii) - surveillance / supervision des systèmes ou des réseaux afin de trouver et de remédier des vulnérabilités. La dernière vérifie que tout ce qui a été conçu dans la première fonctionne correctement et en toute sécurité, ainsi détectant les violations de sécurité. Ceci étant le sujet principal de cette thèse.

Cette dissertation présente un cadre de surveillance de la sécurité en tenant en compte des différents types de jeu de données d'audit y compris le trafic de réseaux et les messages échangés dans les applications. Nous proposons également des approches innovantes fondées sur l'apprentissage statistique, la théorie de l'information et de l'apprentissage automatique pour prétraiter et analyser l'entrée de données. Notre cadre est validé dans une large gamme des études de cas, y compris la surveillance des réseaux traditionnels TCP / IP (v4) (LAN, WAN, la surveillance de l'Internet), la supervision des réseaux de objets connectés utilisant la technologie 6LoWPAN (IPv6), et également, l'analyse des logs d'autres applications. Enfin, nous fournissons une étude sur la tolérance d'intrusion par conception et proposons une approche basée sur l'émulation pour détecter et tolérer l'intrusion simultanément.

Dans chaque étude de cas, nous décrivons comment nous collectons les jeux de données d'audit, extrayons les attributs pertinents, traitons les données reçues et décodons leur signification de sécurité. Pour attendre ces objectifs, l'outil Montimage Monitoring Tool (MMT) est utilisé comme le cœur de notre approche. Nous évaluons également la performance de la solution et sa possibilité de marcher dans les systèmes "à plus grande échelle" avec des jeux de données plus volumineux.

Acknowledgments

In the first place, I would like to thank my supervisor, Prof. Ana Rosa Cavalli, who gave me the opportunity to do this research and has been providing me a myriad of help, guidance, and encouragement throughout my doctoral study. She always does her best regarding both scientific advice and financial resources to send me to conferences, workshops and seminars. These were genuinely favorable opportunities in which I can access novel research trends and technologies, present my research achievements and receive feed-backs from experts. I really appreciate and enjoy the time working with her.

I must thank my colleagues and friends in "Ana's team" for the joint cross-topic work we have done together that in its own way made a difference to this research. Many ideas in this research have their origins in countless discussions with Raul Fuentes whom I have some collaborative work with. I am more grateful to everyone in the lab: Jorge, Diego, Olga, José, Natalia, Anis, Stephane, Nina, Huu Nghia, Khalifa, Javier, Fatma, Ichrak, Joao, Pramila, Anderson, Xiaoping, Hien, Fabrice, Mohamed, Jeevan. The friendly atmosphere that they have created helped me to go through the endless days on campus. Special thanks to my Vietnamese and French-Vietnamese friends in TSP who helped me a lot during my arrival and for passed three years. I will miss our interminable disputations on political issues.

Thanks especially go to Brigitte Laurent, Valerie Mateus, Veronique Guy and Sandra Gchweinder for their beautiful nature of helping me (and everyone) with administrative paper-works; they have made it easy for everything.

During the preparation for my dotoral thesis, I had chance to work with excellent people in Montimage: Edgardo, Bachar, Wissam, Luong, and Huu Nghia, who helped me a lot concerning practical designs and implementations in this research. I own them a debt of gratitude.

I would like to thank also people from IDOLE project (Investigation et Détection Opérées à Large Echelle) from which my PhD research was funded. I really appreciate the chance to participate in the project which provided me a practical view and contributed a lot to the *state of the art* part in this thesis.

Last but not least, thank you my beloved wife Lana, my dearest parents, my stubborn sister, and my dear departed grandfather. Had it not been for their enormous love and support, I could not accomplish this mission.

Thanks to all of you!

To Lana, Mama, Papa, and Titi

Contents

List of Tables

List of Figures

Acronyms

6LoWPAN IPv6 over Low power Wireless Personal Area Networks. i

BR Border Router. 63, 67–69

CPU Central Processing Unit. 19

DAO Destination Advertisement Object. 73

DIO DODAG Information Object. 73

DIS DODAG Information Solicitation. 73

DM Diffusion Map. 35

DPI/DFI Deep Packet/Flow Inspection. 35

DTLS Datagram Transport Layer Security. 60, 62

HIDS Host-based Intrusion Detection Systems. 17

HSTS HTTP Strict Transport Security. 42

HTTPS Hypertext Transfer Protocol Secure. 41–43

IDS Intrusion Detection System. 16

IMAP Internet Message Access Protocol. 43

INSENS Intrusion-tolerant routing protocol for wireless SEnsor NetworkS. 81

IoT Internet of Things. i

IPv4 Internet Protocol version 4. i

IPv6 Internet Protocol version 6. i

IT Information Technology. 7

ITSRP Intrusion Tolerant Secure Routing Protocol. 83

LAN Local Area Network. i, 39

LRT Local Route Table. 1, 83

MAC Media Access Control. 39

MMT Montimage Monitoring Tool. i, ii, 31

NIDS Network-based Intrusion Detection. 17, 18

PCA Principal Component Analysis. 35

POP Post Office Protocol. 35, 39

RP Random Projection. 35

RPL Routing Protocol for Low-Power and Lossy Networks. 59, 73

RSUs RoadSide Units. 110

SMTP Simple Mail Transfer Protocol. 35, 47

SNMP Simple Network Management Protocol. 47

SQL Structured Query Language. 44, 46–48, 51, 52, 54, 92

SQLI SQL Injection. 92

SSL Secure Sockets Layer. 41, 43, 47

SVM Support Vector Machines. 22

TCP Transmission Control Protocol. i

TLS Transport Layer Security. 41, 42

UDP User Datagram Protocol. 27

VANET Vehicular Ad-hoc Networks. 108

VM Virtual Machine. 91–93, 97

WAN Wide Area Network. i

WSN Wireless Sensor Network(s). i

XML Extensible Markup Language. 31

XSS Cross-site Scripting. 45–47, 54

Chapter 1

Introduction

Contents

1.1 Motivation

Computer security, also known as *cyber-security* or IT security, has been always an emerging topic for decades. It is predicted to continuously attract a lot of attention due to the increasing reliance on computer systems everywhere. Computer systems here are not limited in the zone of ordinary desktops or laptops but include also *smart devices* (e.g., smartphones, connected objects, sensor devices). Additionally, the incredible growth of Internet and wireless networks such as Bluetooth and Wi-Fi and the concept *Internet of Things* promise to make the world become *Internet of Every Things*.

However, at the same time, cyber attacks are growing in both volume and sophistication. According to a study made by *Symantec* [1] in 2015, nearly one million of new malware threats are released every day. Two-thirds of Internet users have been victims of cyber-crime, with more than 1.5 million new victims every day. All over the world, people are somehow influenced by cyber-crime. Indeed, the governments are paying several millions of U.S dollars each year in consequence of cyber-attacks. Fig. 1.1 depicts the cost caused by cyber-crime according to a report of Ponemon Institute [2] in eight countries which are the most affected in recent three years. In fact, the cost stays high and increases in some big countries (e.g., United States, United Kingdom) despite the effort and investment for countermeasures.

Due to the attacks, protecting information systems or networks become a difficult but indispensable task. People in research community give an ongoing attention in security. Indeed, as demonstrated in the Fig. 1.2, there are around 5000 papers or more published each year on IEEE Publisher dealing with the problem of "information security". This number is rapidly rising in the recent two years.

[1] https://www.symantec.com/security-center/threat-report
[2] Ponemon Institute homepage: https://www.ponemon.org/

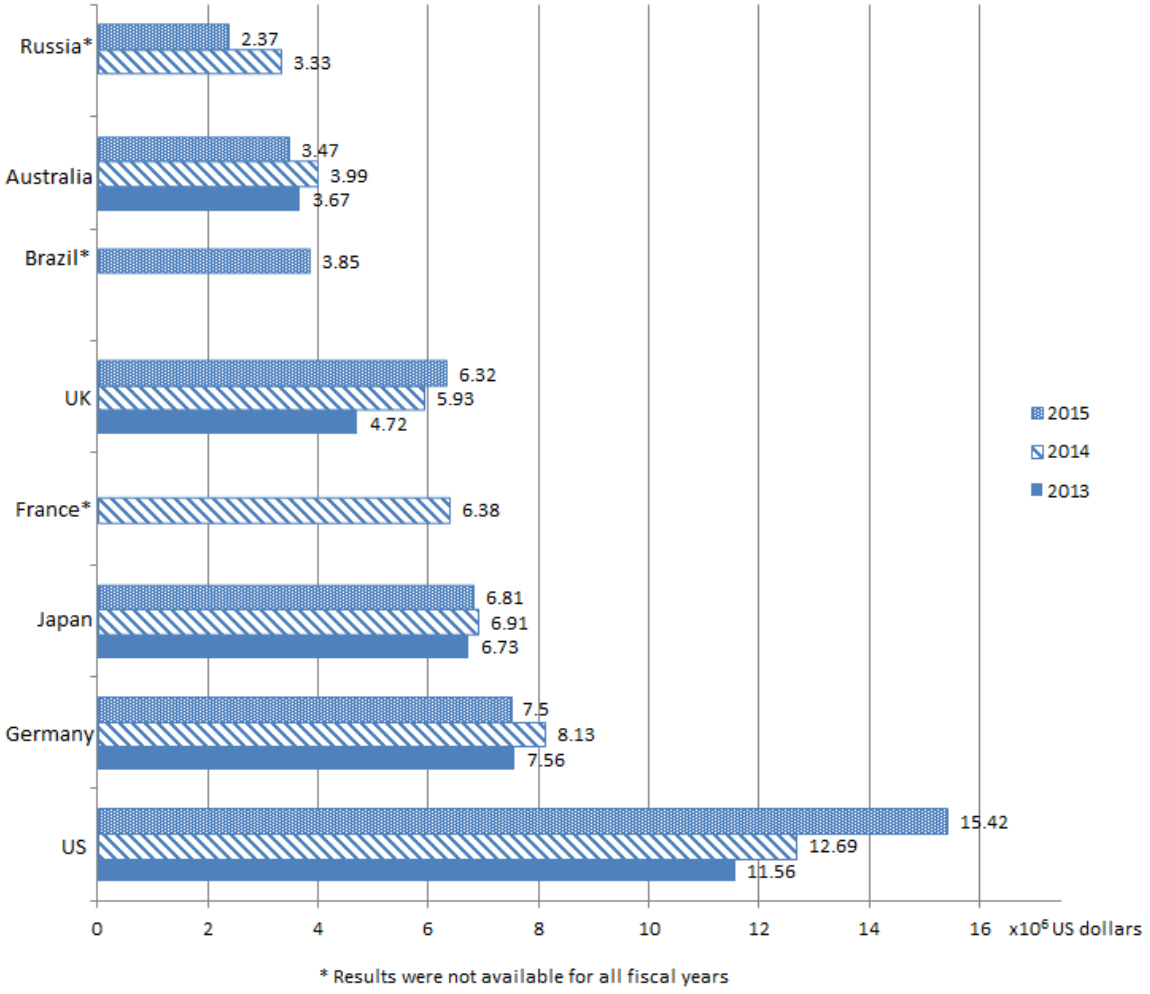


Figure 1.1: Total cost of cyber crime in eight countries

We can divide the research on *information security* into two main directions: The former is to design secured infrastructures with secured communication protocols. The later is to monitor/supervise the systems or networks in order to find and re-mediate vulnerabilities. They are actually mutual tasks. The former can assist the later by forming some additional monitoring-supporting modules. Whilst, the later can verify if everything designed in the former is correctly and securely functioning as well as detecting security violations.

More specifically, looking inside research papers on "information security", we observe around 25 percents and 15 percents focusing on "security monitoring" and "network security monitoring" respectively (Fig. 1.2). Despite the low proportion, *security monitoring* still plays an important role in industrial enterprises. According to the survey made by PricewaterhouseCoopers LLP (PWC) [3] in 2016, "real-time monitoring and analytic" hold the first position in the list of popular solutions to face to cyber-risks in companies (Fig. 1.3). In the other words, *security monitoring* deserves more attention from researchers, even more than the rising interest that it currently receives. This is the main topic of this thesis.

[3]PWC homepage: http://www.pwc.com/

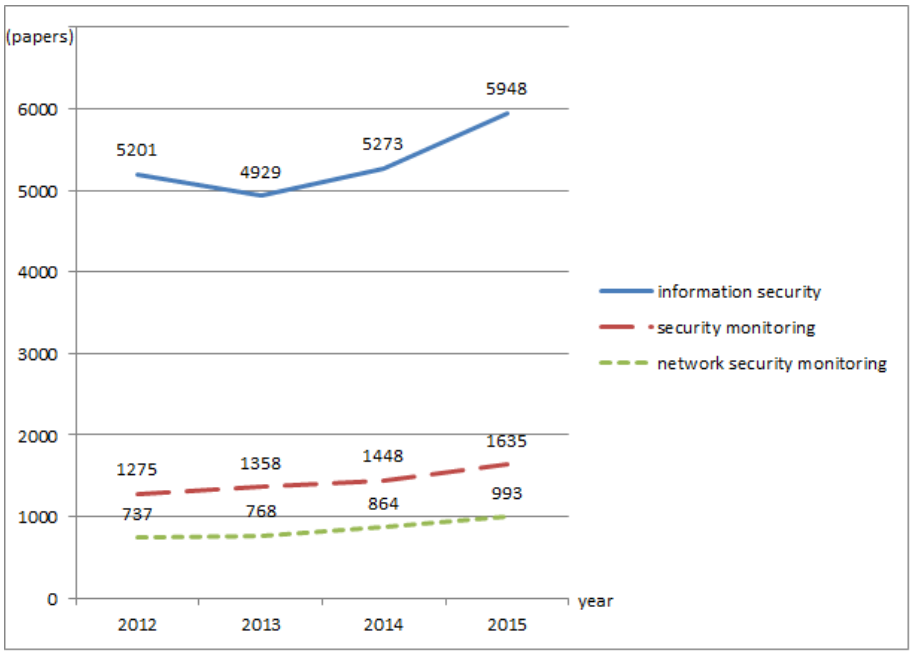


Figure 1.2: Amount of research papers containing corresponding keywords in recent four years (investigation on IEEE Xplore Digital Library)

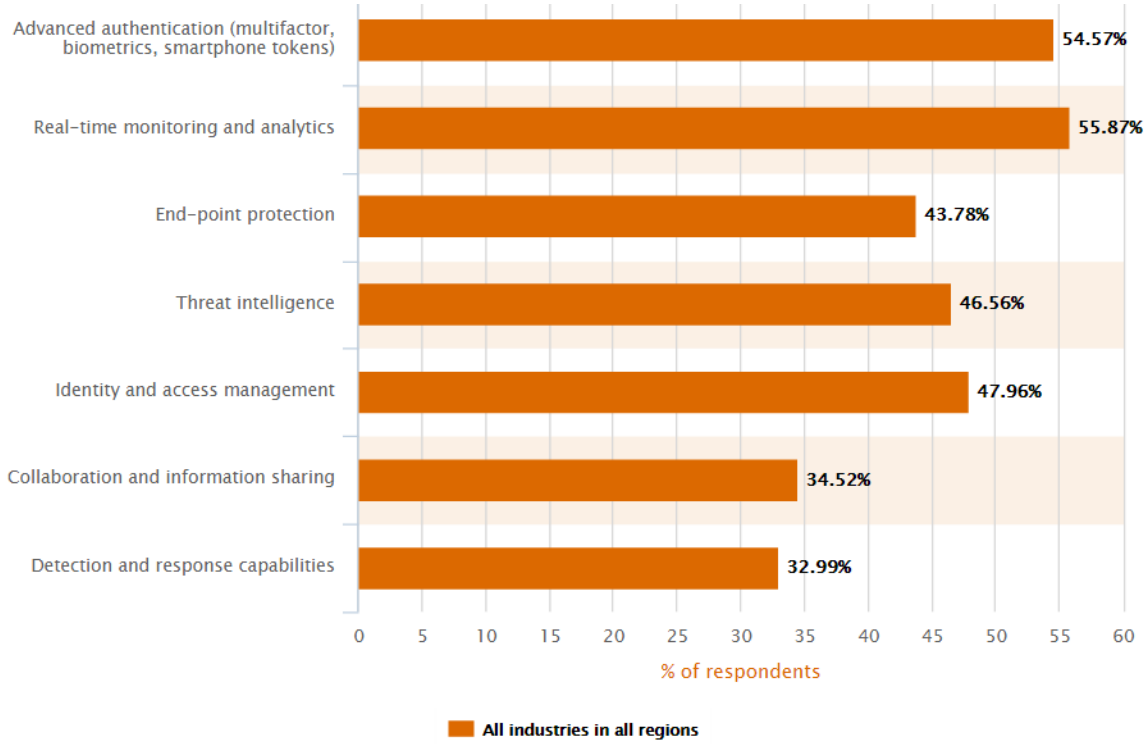


Figure 1.3: Preferred cloud-based solutions chosen by companies to face to cyber-risks

Many establishments assume that their IT infrastructure is sufficiently protected by an up-to-date anti-virus tool and a reliable firewall. However, cyber-criminals are more and

more intelligent to develop more sophisticated methods to access company computers and servers. On the other side, many systems are able to defense very well against attacks coming from outside but stay vulnerable against those from inside. Sadly, employees are commonly not well trained enough to keep up with the security menaces and sometimes, they can be abused by the attackers. As a result, an automated network security monitoring solution becomes a critical challenging task for the network operator, the service provider in order to keep the network operation stable, smooth and safe. If a security violation occurs beyond the company network without being detected, it will be just a matter of time before the entire network has been compromised. The service provider's ability to deliver secure and high-quality services would be compromised too.

As a matter of fact, the monitoring agents must detect every security threats and should warn the administrators in *real-time* or *"near-real-time"* to perform suitable countermeasures. However, in addition to the growing sophistication of threats, there are several other challenges to modern monitoring solutions, namely:

- The rising scale and complexity of running systems or networks under-monitored: From a few network devices or less than a hundred computers in the past to thousands in the presents.
- The higher bandwidth of modern networks: From 10 or 100 Mbps (Megabit per second) before up to 10 Gbps (Gigabit per sec) now.
- The appearance of new devices, new technologies, new network standards and protocols.

The first two challenges prompt the huger traffic which need to be analyzed. The monitors should thus have a better performance by applying novel techniques to deal with voluminous data. Whilst, the last challenge requires the modern monitoring solutions flexible and extensible in order to adapt in new platforms with new traffic. Indeed, those are our main goals in this thesis.

To date, the research on attack and intrusion detection systems is mainly focused on how to detect as many attacks as possible, as soon as possible, with the minimum of the false alarm rate. However, a growing recognition is that a variety of critical applications need to continue to operate or provide a minimal level of services even when they are under attack or have been partially compromised. Additionally, attacks are more and more sophisticated and thus, they are difficult to be captured. Hence, beside intrusion detection, the future IT systems in our opinion should be able to tolerate the intrusion or attacks. It will be surely precious in the cases where IDSs (Intrusion Detection Systems) are too expensive or unpractical. We consider that the *intrusion tolerance* can be also the "next-step" of *intrusion detection*. For example, if a system detects an intruder, it can react somehow to keep a proper operation despite the existence of the intruder, in the other words, tolerate the intrusion. This topic will be discussed in the chapter 6.

1.2 Contributions

The main contribution of this thesis is the proposition and implementation of a security monitoring framework that takes into consideration different types of audit datasets including network traffic and applications' logs. We propose some novel approaches based on

supervised machine learning to pre-process and analyze the data input. From our point of view, these techniques can leverage the data processing speed to assure quick detection even in large scale systems with high traffic.

Our framework is validated in a wide range of case studies including traditional TCP/IP(v4) network monitoring (LAN, WAN, Internet monitoring), IoT using 6LoWPAN technology (IPv6), and other applications' logs. In each case study, we describe how we collect the audit dataset, extract the relevant attributes, handle received data and decode their security meaning. To achieve these goals, the tool MMT (Montimage Monitoring Tool) is used as the core of our approach. We assess also the solution's performance and its possibility to work in "larger scale" systems with more voluminous dataset.

More specifically, we implement the framework for the following case studies:

1. Traditional TCP/IP networks:
   We started the research over common TCP/IPv4 networks which have been the most used all over the world.

   - ARP spoofing:
     Although *ARP spoofing* is a very classic attack, we could still perform it to some LANs. Based on this basic attack, we could realize further attacks to exploit and receive sensitive information from the network's members, e.g., *user-name* and *password*. We proposed then some countermeasures to avoid those vulnerabilities.
   - HTTP User-Agent field case study:
     HTTP is an Internet protocol that is predominant and widely used in every computer networks nowadays. User-Agent is a field in HTTP request message which is modifiable and exploitable to perform attacks. We specified this issue, proposed the methodology and implemented the framework to monitor that field and detect attacks. The framework was also extended to HTTP traffic in general in considering techniques aiming improve the performance (e.g., dimension reduction, machine learning). We proved that *Android malware detection, Web pop-up and Spam avoidance* can be achieved similarly by the same approaches.

2. 6LoWPAN-based WSNs:
   To our knowledge, 6LoWPAN traffic was not understandable to any monitoring solution. That is the reason why we adapted MMT to work in 6LoWPAN-based WSNs by adding several new plug-ins. Then we proposed a number of algorithms and techniques to detect anomalies in such networks based on supervised learning including statistical learning, information theory. Experiments proved our proposition's applicability, extensibility and its reasonable performance.

3. Intrusion tolerance: Possible next-step of intrusion detection

   - Intrusion tolerant routing in WSNs:
     We studied two most famous intrusion tolerant routing protocols by theoretical analysis and by simulations. We pointed out their missing issues and proposed some improvements possibly making them practical and more efficient.
   - Emulation-based intrusion detection and tolerance:
     We proposed an extension of the framework to detect and tolerate attacks at the same time. This work is validated by the SQL injection case study.

1.3 Dissertation Outline

The remainder of this dissertation is organized as follows. Chapter 2 reviews the literature of related topics including security monitoring, intrusion detection approaches and tools, as well as recently proposed detection techniques. Chapter 3 provides a high-level view of our security monitoring framework. The application of our framework to two practical platforms (TCP/IPv4 networks and 6LoWPAN-based WSNs) is presented in the following Chapter 4 and Chapter 5 respectively. Chapter 6 discusses the possibility to enable the concept *intrusion tolerance* by design. Finally, Chapter 7 concludes the dissertation and discusses the future work.

Chapter 2

Background

Contents

2.1 Security Monitoring

Monitoring is the process of dynamically collecting, interpreting and presenting metrics and variables related to a system behavior, in order to perform management and control tasks [3]. *Security monitoring* is thus a sub-domain of monitoring which focuses mostly in security issues.

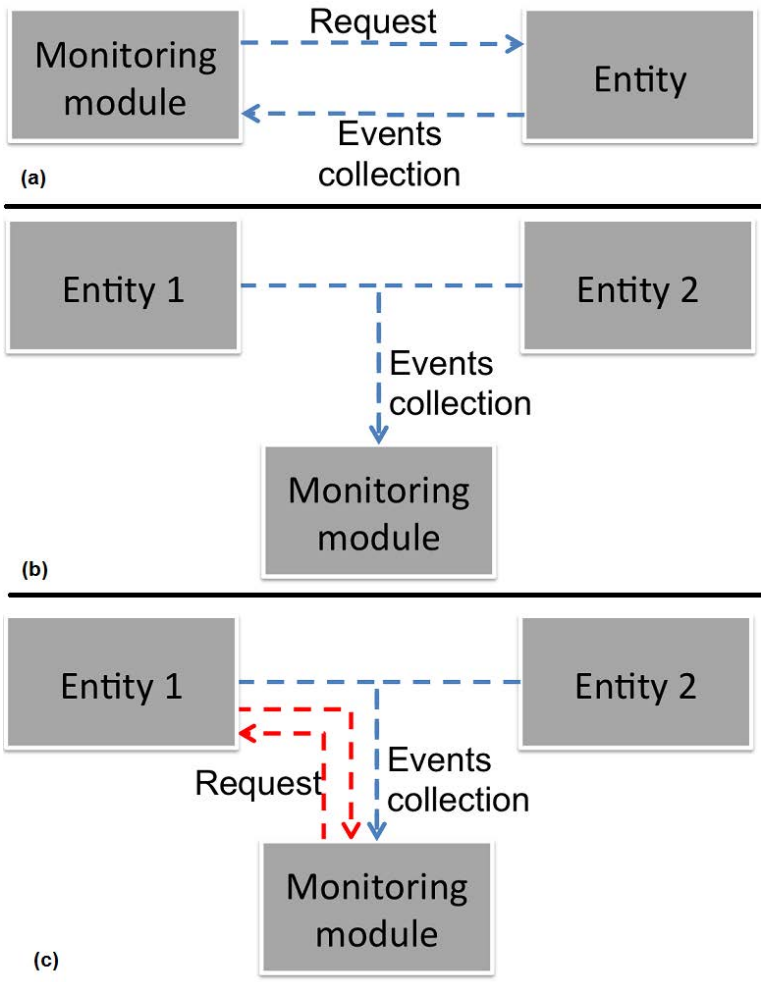


Figure 2.1: Basic active (a)/passive (b)/hybrid (c) monitoring deployment

A typical monitoring platform is basically composed of three main elements:

- Monitoring probes: they directly collect data from the sources to be further analyzed by the analysis module. Probes can be distributed in diverse locations of the system (e.g., network, application, user side, etc.) in order to obtain a global view of the system.
- Database: diverse databases can be utilized. Commonly, one database is used to store the raw data gathered by the probes, and other databases containing diverse information (e.g., rules, security, performance) are also consulted to correlate the collected data in order to extract the required information.

- Analysis module: it contains the means to examine the input collected by the probes, correlate it with the information stored in the database and produce an output that can be used for different purposes. For example, in user access monitoring, the monitoring component can detect the number of unsuccessful connection attempts from a specific source; if this number reaches a predefined threshold, the access for this source can be blocked for a certain period of time.

Additional modules can be added with the aim of performing supplementary tasks. For example, visualization modules may be used to show the requested statistics to the involved actors.

Depending on the real scenario, the audit data input of the monitoring tool can be network traffic, system trace, application logs, users' activity or heterogeneous data source. Regarding the way the data source is collected, we can classify monitoring techniques into three main categories: active, passive and hybrid approaches (Fig. 2.1).

2.1.1 The range of Network Monitoring

Network monitoring is a critical challenging task for a network operator, a service provider or a corporate infrastructure in order to keep the network operation stable, smooth and safe. Depending on the network range, the suitable monitoring technique can be chosen.

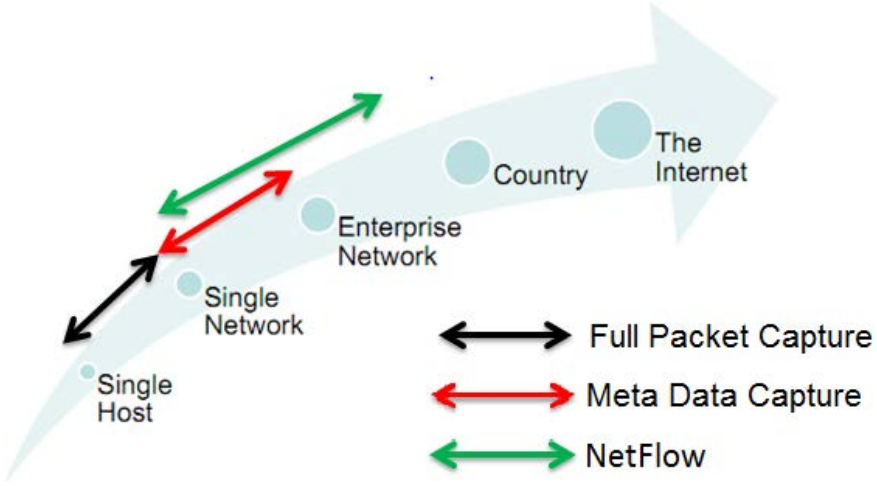


Figure 2.2: The range of Network Monitoring

The Fig. 2.2 presents the range of network monitoring that we can divide into three collections of approaches:

- Full Packet Capture: Monitoring module captures "everything" that goes across the network (e.g., PCAP). This technique is typically used on a single network.
- Meta Data Capture: Monitoring module captures data associated with a particular network activity, typically in the form of logs. For example, it can capture: *from, to, subject, date, attachments* regarding email traffic and *source IP, destination IP, URL, User Agent String* regarding web traffic capture.

- NetFlow: NetFlow aggregates related packets into unidirectional flows. The flow records are collected and stored for later analysis, e.g., SiLK, Argus, etc.

2.1.2 Monitoring requirements

The main objectives of monitoring are functional verification of the system under test, performance analysis, verification of security properties, and detection of security vulnerabilities and attacks. In order for a monitoring platform to be effective in the early detection of any performance, security, or privacy violation, the following requirements must be fulfilled:

- Data capturing performance: traffic speed and data volume should not affect the collection of system information.
- Extensibility: it should be easy to incorporate new services to be monitored without requiring a great effort from the administrators.
- Scalability: the monitoring system should be adaptable to the increase of data to be collected, services and applications to be monitored. This can be done by reducing the data collected through the use of efficient data capturing mechanisms and preprocessing techniques.
- Near real-time operation: in order to achieve early detection of performance or security is- sues, data capture must be aligned with real-time analysis capabilities.
- Granularity: the monitoring system should be capable of differentiating among the different protocols, services and applications that are monitored.
- Diversity: diverse network devices, protocol stacks, services, applications, etc. should be supported by the monitoring platform.
- Low cost: the amount of computing, storage and communication resources consumed by the monitoring system should be as lower as possible.
- Security: the monitoring system should not add new vulnerabilities to the system.
- Transparency : the monitoring functions should not interact or disturb the normal operation of the system.

2.2 Intrusion Detection

First of all, an intrusion refers to any set of actions perpetrated to compromise the integrity, confidentiality or availability of a resource (e.g., a network, an information system, a set of applications or software). An intrusion detection system is a process that monitors events occurring within a system and analyzes them to detect possible signs relating to a security problem.

Initially, intrusion detection was performed manually by human analysts whose task was to examine the log files looking for suspicious activity. This approach could not work in large scale, the idea of automating this process quickly gave birth to intrusion detection systems. Thereafter, an Intrusion Detection System (IDS) became a software/tool designed to analyze the activities of a computer system and to identify the legitimate and illegitimate

ones. Once identified, incidents (i.e. illegitimate actions) are listed and an alert is generated to inform the person in charge of incident management.

In this section, we revise the classical intrusion detection approaches. These approaches are first classified according to the perimeter targeted by detection (network-based or host-based) and according to the detection approach used (anomaly-based or signature-based).

2.2.1 Host-based and Network-based Intrusion Detection

Intrusion detection systems based on the use of probes whose main task is gathering and analyzing information for the detection of computer attacks. Depending on the location of the probe and its scope of action, we distinguish two types of intrusion detection systems detection systems on host (Host-based Intrusion Detection Systems - HIDS) and detection systems network (Network-based Intrusion Detection Systems - NIDS).

2.2.1.1 Host-based Intrusion Detection

Intrusion detection on the host involves installing a software agent on the host system which need to be monitored. Common capacities of Host-based Intrusion Detection Systems (HIDS) include the analysis of activity logs (system, network and applications), event correlation, verification of system integrity and files, the implementation of the policy, rootkit detection (e.g., stealth malware), alerting, etc. They frequently have also the ability to detect the system configuration changes and provide an general overview of the current state of the local host. In some specific implementations, the HIDS agents also enable connectivity and/or compatibility to other security systems. For example, *Cisco Security Agent* (CSA) has the ability to send host data "upstream" to *Cisco Intrusion Prevention System* (IPS), *Checkpoint integrity* can be integrated with *Checkpoint Secure Client* (VPN Client), and *IBM Proventia Desktop* can play the role as a Cisco Network Admission Control (NAC) solution.

Nowadays, most HIDSs have the opportunity to actively prevent malicious or abnormal activity on the host system. This is called an active HIDS. If a malicious change or unauthorized activity is detected, the HIDS can: (a) alert the user via a pop-up, (b) alert the central management server, (c) block the activity or a combination of the three. The decision is based on the existing policy in the system. However, because of the potential impact that the HIDS actions can have on the end user, these systems are often deployed in passive mode. The system then merely alerts the administrator who will choose the appropriate response.

To be effective in an environment with multiple hosts, a HIDS is usually managed from a central location. On the management system, a policy is configured for deployment of local staff. There may be a single policy for all computers, but often several policies are used depending on the operating system, the type of machine and the type of user.

There are many types of HIDS software available. Tab. 2.1 presents a list of the most common including free and commercial ones.

2.2.1.2 Network-based Intrusion Detection

Detecting network intrusion based on the capture and analysis of the network traffic. A HIDS is capable to detect in real-time an attack taking place at one of the machines in the network. It can detect also malicious activities and virus dissemination. Network-based

Free	Commercial
OSSEC (Open Source Host-based Intrusion Detection System)	IBM Proventia Desktop
Tripwire	Cisco CSA
AIDE (Advanced Intrusion Detection Environment)	Checkpoint Integrity
Prelude Hybrid IDS	Tripwire Enterprise
	Symantec Endpoint Protection
	McAfee Host Intrusion Prevention

Table 2.1: Principal HIDSs

Intrusion Detection (NIDS) should then report an alert to the appropriate entity (e.g. user, administrator, analyst, etc.)

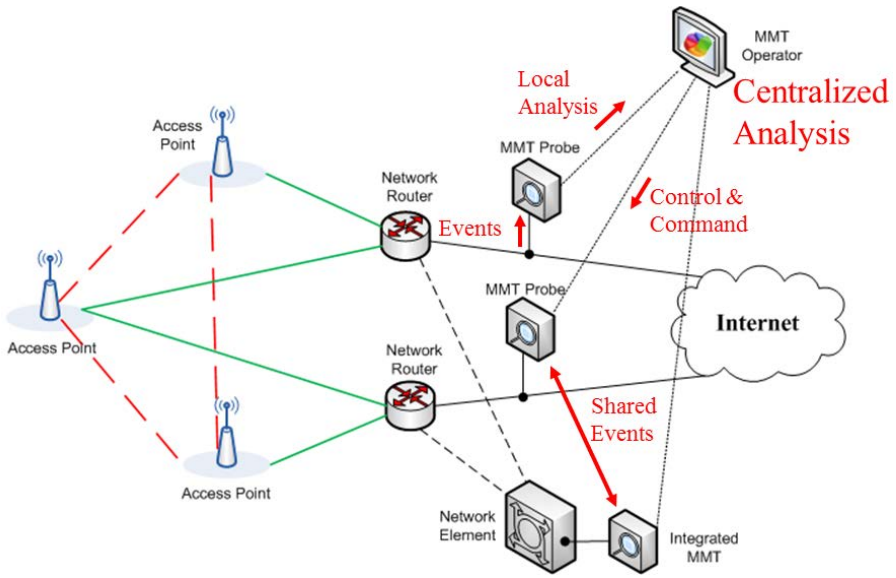


Figure 2.3: A NIDS architecture with distributed (MMT) probes

A NIDS manages and analyzes data packets. A big NIDS server can be installed on the network backbone in order to handle all the traffic. Alternatively or in addition, smaller systems can be deployed in specific links to manage traffic to a specific equipment such as a server, a switch, a gateway, or router. Another type of NIDS could be installed on a central server in order to scan all file systems to ensure the integrity and detect unauthorized activities.

There are several common NIDS architectures. Firstly, in the early warning mode, the NIDS is deployed outside the perimeter firewall. Thus, all traffic entering to hosts or the local enterprise network is scanned by the NIDS. Another architecture is internal deployment: In this mode, the NIDS probes are deployed in order to control the traffic through any link within the monitored network. An example is illustrated in Fig. 2.3 where Montimage Monitoring Tool (MMT) is used as the NIDS. This architecture provides a high level of security. NIDS is deployed near the access routers, bridge nodes, and network boundaries. Finally, a third architecture is to attach a NIDS to each host. NIDS are different from HIDS in the fact that they stay decoupled from the host's operating system and can be managed by a network administrator from a centralized location in the topology.

2.2.2 Signature-based and Anomaly-based approaches

Once deployed (in the host machine or network), the probe collects all or selective data passing through it for analysis. At this point, there are two approaches: signature-based and anomaly-based. In the first approach we will compare the data collected with our prior knowledge of the attacks while in the second, it is based on our knowledge of the normal system behavior.

2.2.2.1 Signature-based approaches

Signature-based approaches rely on the construction of a database of signatures representing already known attacks or security violations. The intrusion detector then compares the observed activities with each element of the database and raises a warning if there is a match. The advantage of this approach is the implementation simplicity and the high relevance of its detection. Two sub-techniques can be used inside this approach: the pattern matching and detection by inference.

- Pattern matching
  The simplest form of signature-based detection considers the sequences of operations recorded in the audit as a language. The signature database is a subset defined by a set of patterns for that language. Intrusion detection becomes classic problems of language recognition that can be resolved by models such as the *Turing machine, State Automata, Petri net*, etc.

  To avoid a combinatorial explosion due to the astronomical number of possible states if we considered all operations in auditing a computer system, the transitions are provided with guards. These are Boolean functions which allow either simplifying the specification of patterns or reducing the exploration space.

  The pattern matching is a deterministic and accurate method. The main difficulty lies in the construction of patterns themselves. They must be both specific enough to discriminate a large number of cases and to avoid generating false positives, and generic enough to detect different variants of the same attack. In addition, an attacker knowing signature database could build an undetectable scenario.

- Detection by inference
  The pattern matching is a heavy technique and seemingly unusable on a large scale. Therefore, many intrusion detectors complement the scenarios with a probabilistic inference algorithm based on the principle of *Bayesian inference*. In this model, known attacks constitute the hypotheses that could explain the observed facts. We consider that a given attack can be detected not only by symptoms appearing in the form of events in audit, but also statistical data as in the case of anomaly detection (memory usage, CPU load, etc.). Given a set of symptoms, *Bayesian inference* can calculate the probability of each known attack scenario. When a scenario has a high probability, an alert should be raised.

  Suppose that the database of attacks contains a set of hypotheses A_i, some of which refer to known attacks and the probabilities $p(A_i)$ and $p(A_i|S_j)$ with S_j is an observed symptom. The construction of this database requires significant expert work for the formulation of hypotheses, the statistical study of possible cases, and the probability

calculations. A *Bayesian inference-based intrusion detector* recursively constructs a decision tree, according to the inference rule:

$$p(A_{n+1}) = P(A_n|S_n) = p(A_n) * p(S_n|A_n) * a \tag{2.1}$$

The initial information, S_0, is an early symptom observed (e.g., by pattern matching) which permits assigning a probability to each of the possible hypotheses. The arrival of each new element S_i modifies these probabilities and adds a new tree node. In the case of a known intrusion, the algorithm eventually generate a node providing a high probability to a certain hypothesis A_q, which is defined as this attack.

Each element observed in the audit can be confronted with different hypotheses and an attack scenario is defined as a combined presence of a set of symptoms, not a particular sequence of events. This minimizes the risk that an attacker can exploit his knowledge of basic attacks to go through without being detected. The development of an undetectable scenario requires the construction of a series of operations realizing the desired attack, but in which nothing actually confirmed any of the hypotheses. This is very difficult in the case of non-trivial attacks. The principle of inference enables detecting number of variants of a known attack and can be applied in the case where the attacker tries to drown the attack inside the noise, in generating a large number of trivial operations. However detecting new attacks is still not possible.

2.2.2.2 Anomaly-based approaches

The *anomaly-based approach* (also known as behavioral approach or misuse approach) consists of two steps: Firstly, a definition of "normal" behavior of the system must be elaborated. This step represents a learning phase and is generally carried out automatically and progressively. The normal behavior definition can be after that set to evolve over time. It is thus a qualified empirical approach in which the behavior definition should be "learned" during the time, by observation. The second step is to analyze the trace to detect any deviation from the normal behavior defined above.

There are several ways to perform this task including the probabilistic approach and the statistical approach.

- The probabilistic approach
  In this approach, we consider that the behavior of a system is characterized by several events which we will associate a probability of occurrence to each one. Some qualified normal events will have a high probability of occurring while doubtful events have a low or zero probability.

- The statistical approach
  Although superficially similar, the statistical approach is more accurate and comprehensive than the last. It is a quantitative measure of system resources used in the context of a normal behavior. Subsequently, the same activity generating different statistics may or may not (depending on the margin of error) associated with an attack. The efficiency of this approach lies in developing the statistical profile of normal behavior. If it is skewed by an attempted attack, it will not be able to detect similar attacks later.

2.2.3 Limitations of classical approaches

Signature-based detection approaches are relatively easy to implement, require no learning curve. This eliminates the risk of over-training or voluntary deformation of the profile that can be observed in behavior-based approaches.

However, these approaches require an active maintenance and very frequent updates of the signature database to integrate any new attack discovered. Indeed, the update cannot be performed automatically as in the case of behavior-based detection. This fact implies a higher rate of false negatives. The problem arises especially with very recent attacks for which signatures have not been included in the database yet. Also, the absence of a standard pattern description language limits the usefulness of signatures described in a given language since interoperability between different detector is probably not possible. If some signatures descriptions are too simplified for performance reasons, which makes them likely to correspond to the legitimate actions and therefore to trigger false positives.

Anomaly-based detection approaches have several interesting features. First, as the hypotheses are made only on the normal behavior of the system and not on possible attacks, detection is exhaustive. Indeed, the system allows a prior to detect all that "differs" from established normal behavior. Thus, it becomes possible to envisage detection of unknown attacks and no specific knowledge about the attacker is required. All necessary information is collected within the system. On the other hand, once the learning phase terminates, the IDS does not require particular update. The definition of normal behavior evolve only slightly if any.

Nevertheless, a high rate of false positives is the main weakness of these approaches because it is sometimes difficult to define the "normal behavior". Sudden changes in the environment can have an impact on behavior. This sudden change in behavior will be considered as an anomaly and an alert will be generated. Also, since the first phase is dedicated solely to the development of the definition of "normal behavior", this one is particularly vulnerable to attack. Indeed, the presence of signals related to an attack in the learning trace will result in skewing the definition of behavior. Thereafter, any similar attack will be treated as a normal behavior. The information used during this first phase in the optimal condition must be totally free from damage. In practice, it is frequently impossible to have such perfect environment.

2.3 Novel Advanced Detection Techniques based on Machine Learning

As mentioned in the section 2.2.3, classic approaches have themselves several weaknesses that can be exploited by attackers and/or limit their utilization in large scale. This section presents some prominent novel techniques which attempt to deal with those limitations.

2.3.1 Supervised learning

The automatic *supervised learning* presupposes the existence of a database used for learning in which there are data entries (packet attributes, signatures, etc.) and a label (e.g., class) associated to each entrance. Supervised learning algorithms use the correlation between the input and the class to modify the internal parameters and thus find the transfer function of the system. Once the training/learning phase is done, we achieve the internal parameters

of the model and apply them on new data (which were not used for learning) to have the classification. The most-used supervised learning algorithms are neural networks, decision trees, Support Vector Machines (SVM) and Bayesian networks. We present in the following three of those.

2.3.1.1 Neural networks

A *neural network* is defined as a set of nodes (or neurons, units) connected by their inputs and outputs in a predetermined scheme forming a topology. The learning of a neural network consists of modifying the weights of the connections between neurons while satisfying a convergence criterion. Effectively, these models are difficult to interpret.

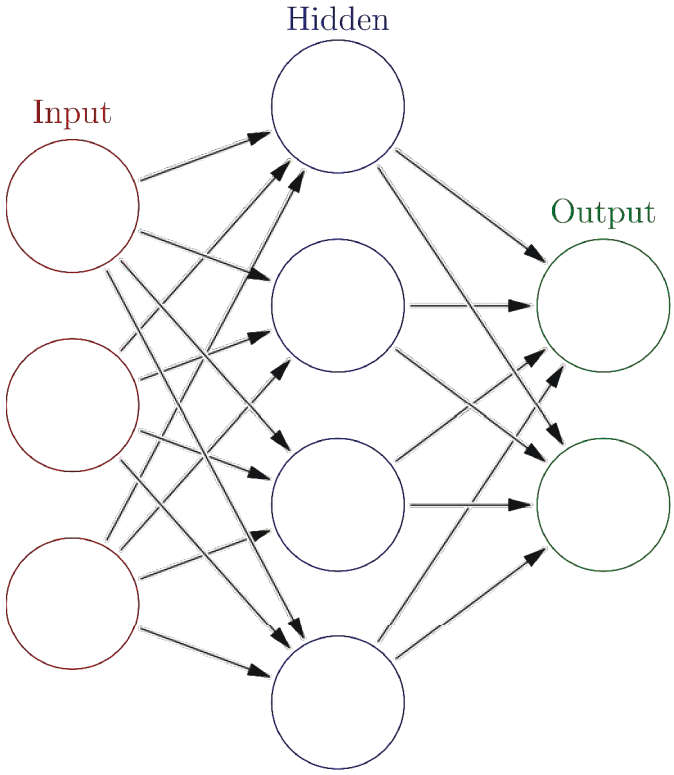


Figure 2.4: A generic example of Neural Networks

The neural topology is organized in three layers (Fig. 2.4): (i)- Input layer: The number of nodes equals to the number of attributes that characterize the data input in the learning phase; (ii)- Output layer: The number of nodes equals to the number of classes desired by the model. Each node gives the probability that the given sample input is classified in the appropriate class; (iii)- Hidden layers: The number of nodes depends on the implementation. A big number of hidden layer nodes enables learning a complex function, but increases also the complexity and the time required for learning.

Like any other supervised learning algorithms, neural networks operate in two phases. In the first phase, the network learns how to adjust the weight of its connections so that the output classes correspond to values of the input attributes on labeled samples of learning data. The weights are usually initialized randomly. Before the learning stage, we must

decide the desired number of nodes in each layer. In the detection or classification phase, the neural network will be used for classifying the corresponding attribute values in new data.

2.3.1.2 Decision trees

A *decision tree* is a tree structure permitting to represent a set of rules. The rules are represented by a series of conditions which define a path from the root to a leaf containing the decision to apply. If the target variable takes a finite set of values, tree models are called *classification trees*. In this case, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. If the target variable takes continuous values (typically real numbers), decision trees are called *regression trees*.

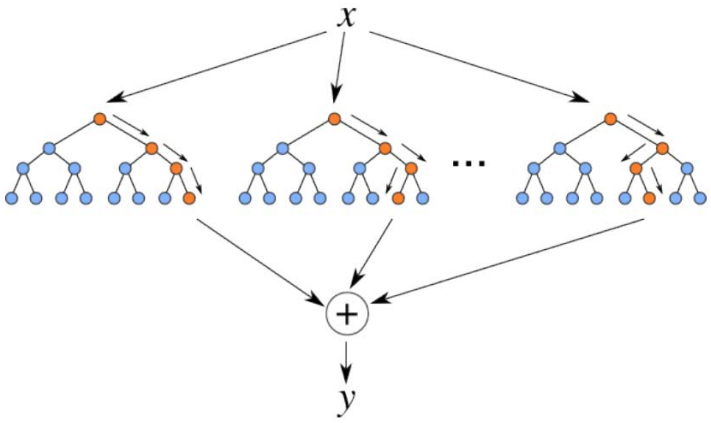


Figure 2.5: A generic example of Decision trees

2.3.1.3 Support Vector Machines

Support Vector Machines (SVM) consist of a class of alternative algorithms to *neural networks*. Given a set of training examples, each one is marked for belonging to one of two categories. An SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall into.

The use of SVM is widespread in the academic world as well as in the industrial world. It is conceptually easy to understand and it is an optimization problem of a convex function. SVM is therefore a global optimum and do not fall on local optima as neural networks. In addition, SVM has a set of vectors easier to interpret than a neural network which is a black box. In addition to linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

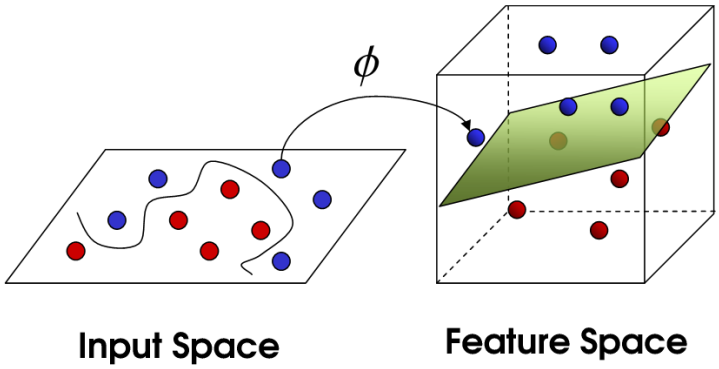


Figure 2.6: A generic example of Support Vector Machines

2.3.2 Unsupervised learning

When we do not have a labeled learning data, supervised learning is not possible and an unsupervised learning approach is required instead. In this case, the algorithm must find classes that make sense, in the other words, group similar examples and separate distinct examples. The most-used unsupervised learning algorithms are *association rules* and *k-means*.

2.3.2.1 Association rules

Association rules contain rules in the form $X \rightarrow Y$, where X can consist of a set of conditions and Y is usually restricted to only one conclusion (Fig. 2.7). Learning a set of association rules is to find all instances where X and Y are co-present and to retain only the rules satisfying statistical criteria, including *support* and *confidence*. Association rules are individually legible but are generally very numerous.

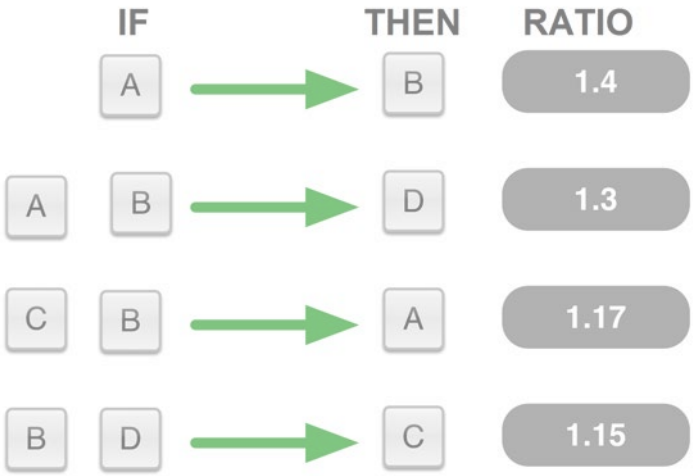


Figure 2.7: A generic example of Association rules

2.3.2.2 The k-means

The *k-means* algorithm is an algorithm partitioning data to separate the examples into k classes where each instance is assigned to the class whose center is the closest in the sense of distance. Learning such a classification is to determine the k cluster centers. This algorithm requires to choose the distance and the number k (Fig. 2.8). It is very effective for a lot of problem and popular due to its simplicity.

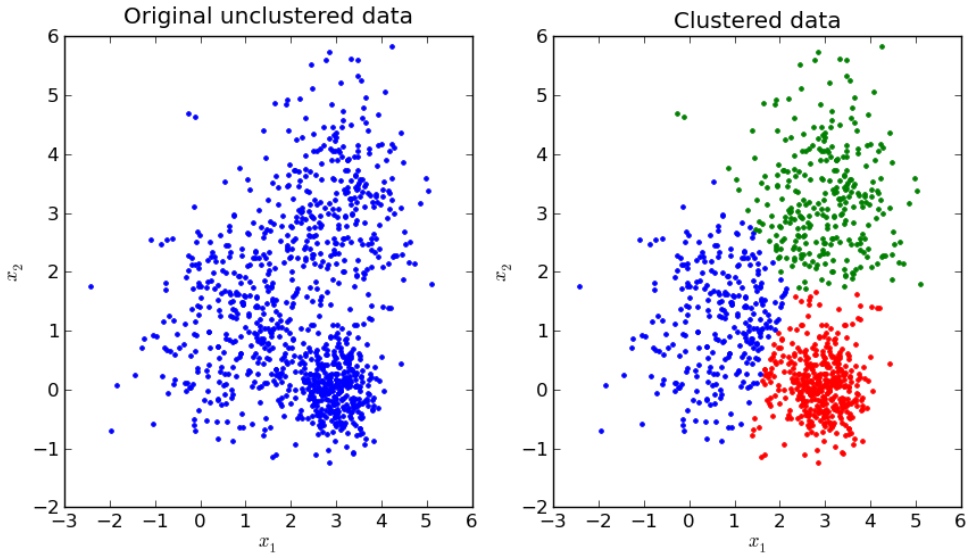


Figure 2.8: A generic example of k-means clustering with $k = 3$

2.4 Classic Intrusion Detection Systems

In reality, there exists many IDSs proposed both in academic institutions and industrial companies. They use different techniques to identify malicious behavior and/or to leverage the data processing speed. From the author's point of view, there are mainly three classic IDSs which are the most well-known and widely used, namely Snort, Bro and Suricata.

2.4.1 Snort

According to [4], Snort is an open source intrusion prevention system capable of real-time traffic analysis and packet logging. Snort is a typical example of signature-based IDSs. It is able to perform protocol analysis, search/match the content based on a signature database. Thus, it can be used to detect a variety of attacks such as buffer overflows, port tests, and worms. Signatures are represented by *rules* which can be actively contributed by an open Snort Community. The latest versions of Snort can combine a set of signatures to detect anomalies.

Snort can operate as a packet sniffer capturing packets traversing the network link, and display them on the console. It can be also a packet logger saving complete packets into a log file for later analysis. Most commonly, it plays the role of a NIDS which monitors the network traffic and verifies the packets in using a set of rules predefined. A Snort rule set

is a file containing the signature of a malicious activity or attack with respective actions in response. Snort uses *pattern matching technique* to verify whether a rule is matched. Nevertheless, the current version of Snort does not provide communication mechanisms which allows Snort instances being deployed in different nodes to correlate data input or share attack detection information. As a result, Snort does not support distributed intrusion detection and collaboration between the participating nodes to identify malicious nodes in a consensual manner.

2.4.2 Bro

Bro [5,6] is a representative network security monitor that focuses not only in security but also more general network traffic analysis. As any security monitoring tool, Bro passively supervises the network traffic for detecting signals of security attacks and intruders. In fact, Bro has two major components: (i)- An *event engine* transforming the filtered network traffic stream into a series of higher-level events. (ii)- A *policy script interpreter* executing a set of *event handlers* written in a custom Bro scripting language. *Event handlers* can be the expressions of security policies, i.e., responsive actions to take if a pre-defined activity is detected. The script can be used also to update the global state information, synthesizing new events, scheduling events, recording information to disk files, and generating alerts. In a nutshell, users are free to create their own script to specify their policies. This fact makes Bro flexible and extensible.

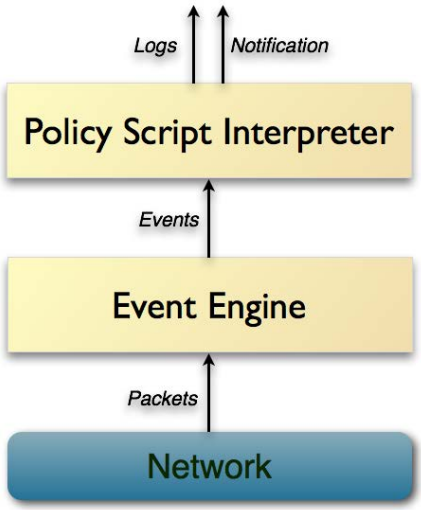


Figure 2.9: Bro's internal architecture

People in Bro project are actively transforming Bro from an originally academic tool to an industrial product by providing open source version as well as commercial support. However, in our opinion, the current version of Bro is still more suitable for academic researcher with expert knowledge in scripting language rather than general users.

2.4.3 Suricata

Suricata [7] is an open-source NIDS developed by *Open Information Security Foundation (OISF)*. The first beta version was released in December 2009. Despite being released for not very long time, Suricata has been rapidly known and used by the community. The novelty of Suricata is its multi-threaded architecture that enables taking advantage of multiple core and multiple processor host architecture. Suricata can, thus, increase the processing speed and the efficiency of traffic analysis in using hardware acceleration. For example, in can achieve 10 gigabit speeds on real life traffic without sacrificing rule-set coverage. This fact can potentially resolve the problem concerning more and more high scale and voluminous network traffic under-test.

As Snort, Suricata is a (signature) rule-based IDS, i.e., it matches a set of pre-defined rules to the captured traffic to recognize an attack if any. Therefore, Suricata retains the same architectural limitations: the inability to provide a cooperative scheme among nodes to detect sophisticated attack that demands mutual works.

2.4.4 Evaluation studies of classic IDSs

Regarding the performance, there are a number of assessment comparing those classic IDSs. We review in the following two of them:

In [8], the authors present the results of three realistic experiments to compare Snort and Suricata by observing:

- Their real-time performance while monitoring live traffic.
- Their performance while analyzing big (off-line) recorded traffic.
- system in response to malicious packets generated by the Pytbull tool [9].

Overall, Suricata performs at least as well as Snort, and even better at most cases. Indeed, Suricata can handle larger volumes of traffic than Snort with similar accuracy. His performance increases more or less linearly with the number of processors. However, there is no significant advantage in speed or accuracy of Suricata in comparison with Snort observed in a certain amount of cases. The results concerning false positive and false negative can be explained by the weaknesses of the set of rules used for testing. It is inconclusive whether Suricata or Snort has a better detection algorithm, but a 64-bit machine is recommended for both to allow loading comprehensive rules. The ability to use multi-threading techniques in a multi-CPU environment will leverage Suricata in the future while network traffic is continuously increasing. But Snort can remain in service for the near future before Suricata becomes more stable.

The work presented in [10] attempt to evaluates Suricata, Snort and Bro in different scenarios. The metrics measured in the experiments are composed of the CPU usage, the number of packets lost and the number of alerts. Regarding the CPU usage, Bro has a pretty stable performance even if the traffic rate increases while Snort and Suricata are more CPU consuming. Those three IDSs react differently in terms of packets lost. Bro is the best in case of TCP traffic but it is also the worst in case of UDP traffic. No clear trend is found on the subject of the number of alerts. Indeed, this metric depends largely on how users define security rules rather than the nature of IDSs themselves.

Regardless of limitations in two aforementioned studies ([8, 10]) concerning the presentations and not thorough evaluation metrics, we still find the inspiration for our own

evaluations in order to evaluate our propositions. This will be further discussed in the following chapters.

2.5 Intrusion tolerance

Intrusion or attack tolerance of a system is generally understood as the capability to continue to function properly with minimal degradation of performance, despite intrusions or malicious attack [11]. In terms of networks, this concept means the ability to maintain the overall connectivity and diameter of the network as nodes are removed.

There are four categories of common architectures [12]:

- Detection Triggered [11] (e.g., Sitar, Dpasa, Willow, etc.): These architectures build multiple levels of defense to increase system survivability. Most of them rely on intrusion detection that triggers recovery mechanisms.
- Algorithm Driven (e.g., Pasis, Maftia, ITUA, etc.): These systems employ algorithms such as the voting algorithm, threshold cryptography, and fragmentation redundancy scattering (FRS) to harden their resilience.
- Recovery Based (e.g., SCIT): These systems assume that as soon as a system goes online, it is compromised. Periodic restoration to a known good state is necessary.
- Hybrid (e.g., COCA): These systems are the combinations of different techniques.

In terms of techniques, *intrusion tolerance* can be realized thanks to the following approaches:

- Redundancy and Diversity: Redundancy refers to the extra reserved resources allocated to a system that are beyond its need in normal working conditions. Whilst, diversity means that a function should be implemented in multiple ways, differently at different times. For example, research has made it practical to automatically generate diverse executable from the same source code or automatically change the configuration of a system from time to time to confuse the attacker.
- Voting: Voting is used to resolve any differences in redundant responses and to arrive at a consensus result based on the responses of perceived non-faulty components in the system. It has two complementary goals: masking of intrusions, thus tolerating them, and providing integrity of the data. The process involves comparing the redundant responses and reaching agreement on the results to find the "correct" response.
- Acceptance Test: This issue usually consists of a sequence of statements that will raise an exception if the state of the system is not acceptable. If any exception is raised by the acceptance test, the module is said to have failed or been compromised.
- Threshold Scheme and Distributed Trust: The general idea is to devise a method to divide data D into n pieces in such a way that it needs at least k shares to reconstruct original data D. Anything less, reveals no information at all. This elegant idea has found many applications in key management schemes as well as cryptography.

- Dynamic Reconfiguration: Reconfiguration after the detection of an intrusion in traditional systems is mostly reactive and generally performed manually by the administrator, thus, involves some downtime. Survivable systems need a dynamical and adaptive reconfiguration to be proactive instead.
- Indirection: The common goal of all indirection techniques is to separate clients and servers by an additional layer that play the role as protection barriers. There are four main types used in intrusion tolerant systems, namely proxies, wrappers, virtualization, and sandboxes.

Chapter 3

Proposed Security Monitoring Framework

Contents

3.1 Framework Overview

The Fig. 3.1 summaries the high-level design of our framework which provides security monitoring capacities in different platform including networks, systems and applications. We will go to the details of each modules of the framework in the following sections. In general, we would like to integrate to it the following possibilities:

- Hybrid approach
  The classic intrusion detection approaches have themselves some limitations (Section 2.2.3), e.g., incapacity in detecting new attacks of signature-based approaches and false positive concern of anomaly-based approaches. Our framework provides hybrid approaches based on both known signatures and a set of expected legitimate behaviors.

- Novel advanced techniques
  We integrate novel investigation and detection techniques in dealing with input data by using statistical/machine learning. The goal is to efficiently learn and define normal/abnormal behaviors.

- High performance to monitor large scale systems. We take into consideration the concern about voluminous data input (e.g., Big Data) created by large scale networks or complicated systems. The performance of our solution in some case studies will be evaluated by experiments and discussed in the next chapters.

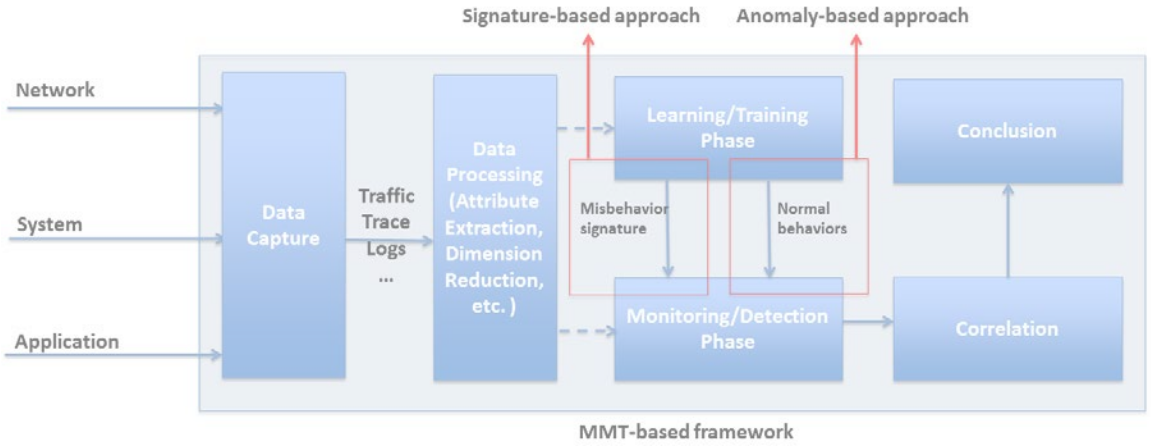


Figure 3.1: Overview of proposed framework

- Flexibility
  Our framework can be extended to understand new type of data input (e.g., network traffic, log files and any structured data). Users have also the flexibility in adding their rules to detect security incidents in varied situations.

- Scalability
  Our framework can realize not only individual tasks at a single monitoring point inside a network, a system, or a host, but also cooperative missions by correlating the results received from other monitoring points as well as collaborating with other third party solutions.

3.2 Montimage Monitoring Tool

Montimage Monitoring Tool (MMT) plays the important role in the framework to achieve such goals. In this section, we provide a brief introduction on Montimage Monitoring Tool (MMT) which is the core of our framework.

MMT is a monitoring tool that allows capturing and analyzing network traffic in both on-line and offline manners. MMT supports network traffic inspection by extracting necessary attributes and referring to a set of security rules. Fig. 3.2 illustrates the architecture of MMT. MMT uses Deep Packet/Flow Inspection (DPI/DFI) techniques and consists of three principal modules:

- **MMT-Extract** is enables the extraction of network protocol fields of not only offline structured traffic (e.g., PCAP files) but also real-time on-line network traffic passing by an interface. It is possible to build a new plug-in for the addition of new protocols and the parsing of proprietary structured data. In practice, this module permits monitoring different observable application, system, or network. In the case of application, the input can be the exchanged messages or events log.

- **MMT-Security** contains security rules written in XML that refer to both expected and unexpected behaviors. MMT-Security model is inspired from Linear Temporal

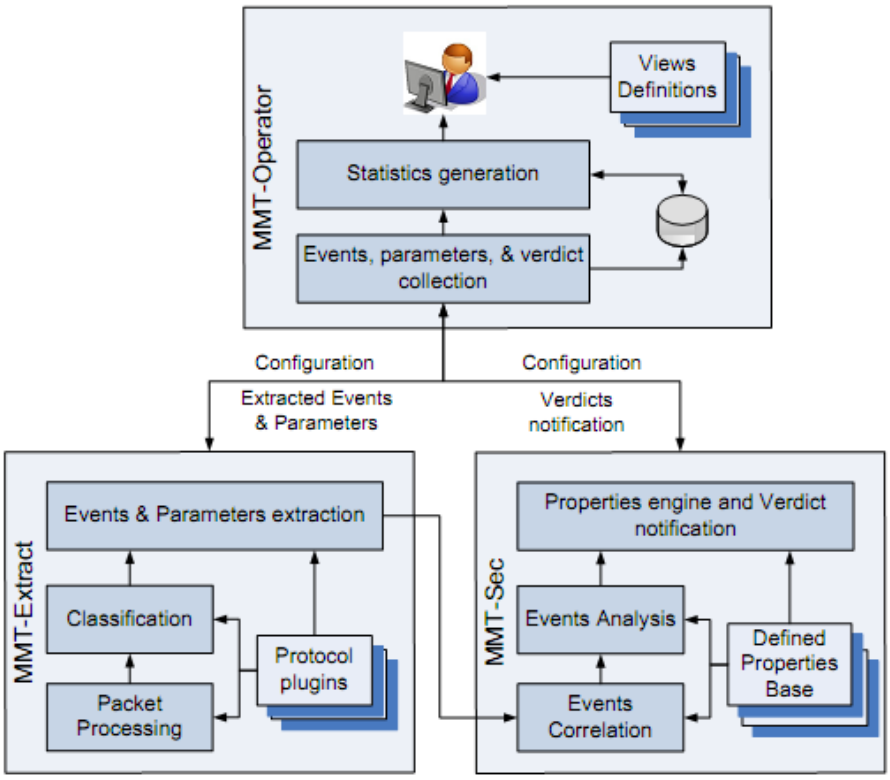


Figure 3.2: MMT global architecture [1]

Logic. Different rules can be correlated in order to detect security incidents. Rules in XML provide the advantage of simple and straightforward structure verification. A property is an IF $<$ *context* $>$ THEN $<$ *trigger* $>$ relation(Fig. 3.3). The trigger is checked if and only if the context is valid. If the trigger is found valid, then the property is satisfied. Otherwise, the property is violated.

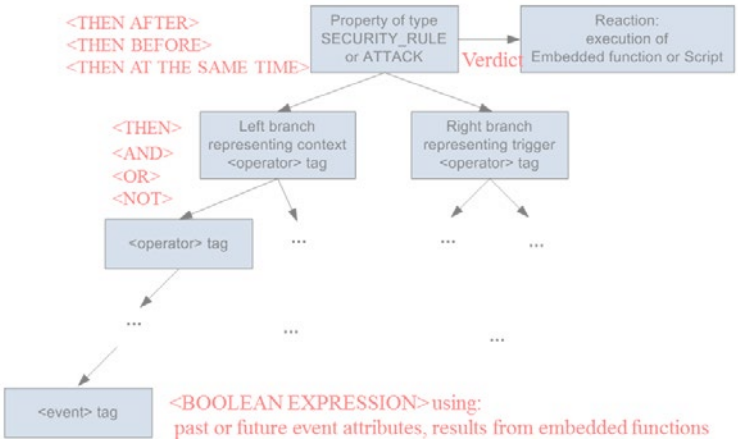


Figure 3.3: An MMT Security Property sample

- **MMT-Operator** allows a graphical user interface which is customizable to display the result.

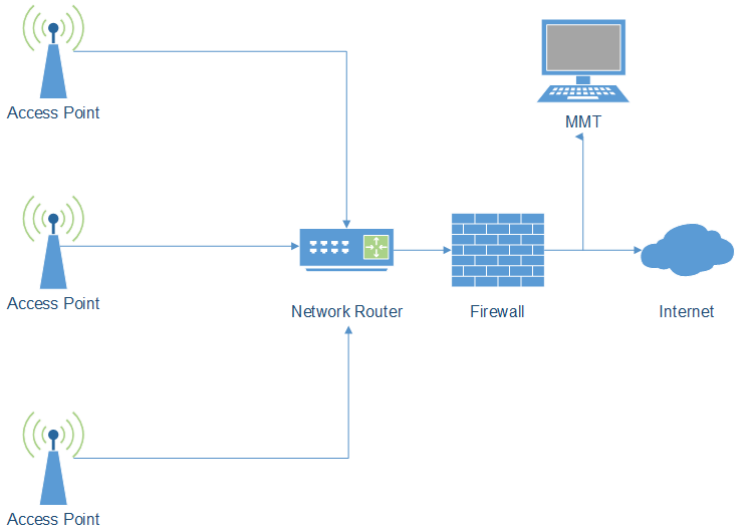


Figure 3.4: MMT's position to listen to live traffic

Fig. 3.4 illustrates a typical position of MMT in monitoring an organization's network, i.e., Network-based IDS (NIDS). For example in [13], MMT is implemented in a host of a wired LAN (Local Area Network) to detect ARP spoofing attack. There could be one or several distributed MMT-Agents (Probes) in different positions of the network collecting a better global view and correlating different events and security properties.

Nonetheless, MMT can be installed in an individual local host as a Host-based IDS (HIDS) to listen to traffic passed by one or several interfaces. Web administrators can also integrate MMT in their web-servers to inspect incoming requests before processing them. MMT has been recently put on cloud to become an on-line monitoring tool. Users can inspect their network by creating a tunnel to MMT VPN server thanks to PPTP (Point-to-Point tunneling protocol) [1]

MMT is chosen to be the core in our framework principally because of its flexibility and extensibility that make it adaptable to different scenarios. Compared to existing intrusion detection techniques, the originality of the MMT is that MMT is not based on only pattern matching (i.e., signature-based) as SNORT nor requires writing executable scripts as in BRO. MMT is a flexible solution that can integrate pattern matching, statistics and machine learning [14] techniques depending on the actual problem. MMT property rules are descriptive and straight forward. They can be written and added to describe normal/abnormal behaviors. Furthermore, MMT is open for developer to add plug-in in order to deal with new structured input as well as to pre-process the attributes before analyzing them in the module MMT-Security.

[1] https://mmt-cloud.montimage.com/

3.3 Data capture

As demonstrated in the Fig. 2.3 and Fig. 3.4, thanks to MMT, our framework is able to capture the live traffic and trace passing by a monitoring point. The data captured can be saved for later investigation (*offline*) or can be inspected immediately (*real-time* or *"near" real-time monitoring*).

One important issue in distributed monitoring solutions is monitoring points where we should locate the probes (i.e., sniffers, agents). Intuitively, they should be somewhere so that we can collect the most important traffic/trace without negatively affecting the system under-monitored. If the solution is installed at a host machine, this host must satisfy minimum requirements in terms of memory (to store the data captured) and processing capacity. For example, in some critical systems, a real-time monitoring is demanded to raise an alert and trigger suitable countermeasures in maximum five seconds after the occurrence of the incident. The machine hosting the security monitoring solution in this case must be quick enough to realize necessary computations and actions provoked inside and outside the security monitor.

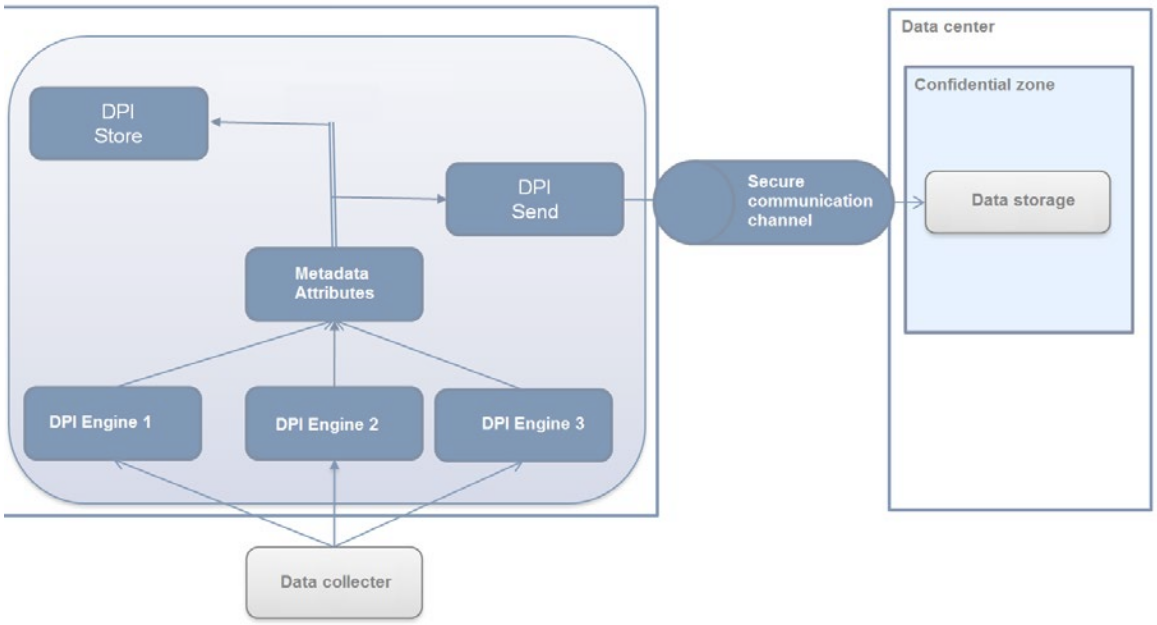


Figure 3.5: A proposed architecture to sanitize audit data before analyzing

In some cases, the traffic and logs are not directly captured by the monitoring tool. Many organizations have themselves means to collect and store (for a duration of time from some days to some months) the traffic/logs in their network/systems (e.g., Fig. 3.5). Due to the privacy concern, they do not provide a direct access to a third party security monitor. Instead, they pre-sanitize (e.g., obfuscation) the data before the analysis. To deal with this problem, our proposed framework supports both *online mode* (i.e., listening to the interface) and *offline mode* (e.g., reading data provided by the third party). We take advantage also a *plug-in approach* that enables understanding "new data input" by adding new plug-in defining the new structure.

3.4 Input pre-processing

3.4.1 Attribute extraction

Looking inside the data input, it is typically not so important to look at the entire contents. Depending on the case study, some attributes can be interesting. But overall, the security monitoring solution must be able to extract as many attributes (e.g., fields) as possible. Then it is up to the users extracting their desired attributes to verify corresponding security properties.

Our framework inherits the capacity to decode and extract fields in more than 150 widely used Internet Protocols in TCP/IP network (HTTP, POP, SMTP, etc.). We have been adding also new plug-ins concerning protocols related to IoT/6LoWPAN technologies.

We used the Deep Packet/Flow Inspection (DPI/DFI) technique to inspect the traffic. In the other words, we look inside each packet, determine the protocol's identifier, identify its type and extract valuable attributes in referring to data structure that was predefined for the corresponding protocol. Packets from the same flow are grouped and correlation analysis can be necessary. There are three types of extracted attributes:

- Real attributes: Protocol field value which can be directly extracted from the inspected packet.
- Meta attributes: Attributes linked to each packet to describe capture information (e.g., timestamp)
- Calculated attributes: Attributed indirectly calculated from real attributes and meta attributes (e.g., delays, packet loss rate, jitter, link weight)

3.4.2 Dimension reduction

If the volume of the data input is huge, an additional step to reduce its complexity is considered. The benefit is that we can save storage space thanks to data compression and reduce the size of the processed data if the computing capacity is limited. The reduction techniques are also used to eliminate noise before the data treatment.

We used so far the following *dimension reduction algorithms*:

- Principal Component Analysis (PCA)
- Random Projection (RP)
- Diffusion Map (DM)

Depending on the corresponding case study, they will be detailed in the section 4.4

3.5 Training/learning phase

The Fig. 3.6 illustrates the high-level diagram of the training/learning phase. The main purpose of this phase is to define a database of signature related to malicious activities and/or a set of expected normal behaviors. A signature can be simple or complicated depending on the abnormal behavior that it refers to. Whilst, normal behaviors can be represented by an expected interval of a variable.

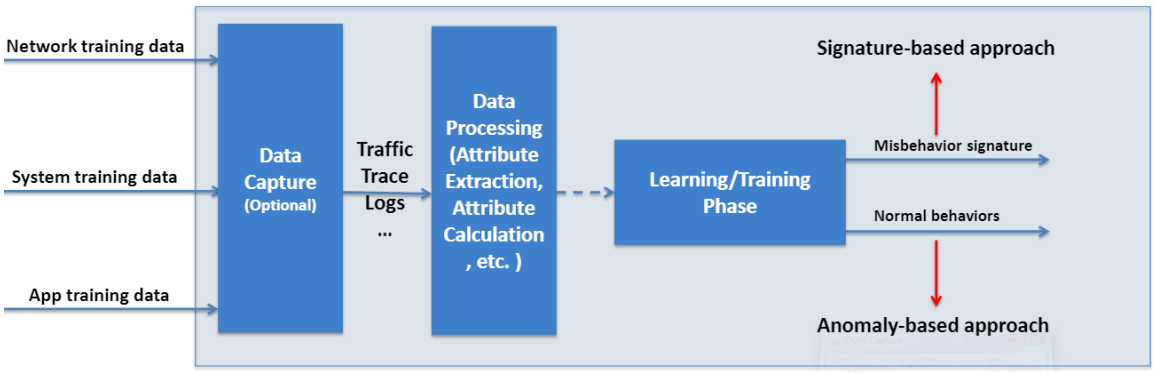


Figure 3.6: Training/Learning phase diagram

The input of this phase is the training data captured by the monitoring tool itself or provide by a third party. *Dimension reduction* can be optionally performed. Valuable attributes serving the learning algorithm are then extracted. In the context of this thesis, we use only labeled data (i.e., supervised machine learning) as the input. In the other words, we know that the input is assumed as normal or related to attacks before learning it. The detailed description concerning the learning techniques will be discussed later together with the corresponding case study.

3.6 Detection/Monitoring phase

After the termination of the learning phase, the set of *misbehavior signatures* as well as *normal behaviors* will be saved to be referred in the *monitoring/detection phase*. Signatures coming from the third party are also encouraged because the more redundant the signature database is, the more likely we can detect the malicious intruder or attacks.

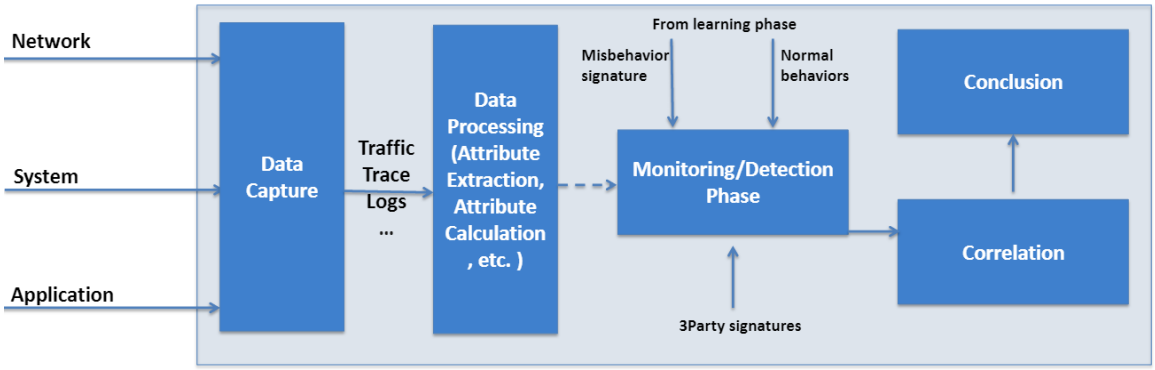


Figure 3.7: Detection/Monitoring phase diagram

The input of this phase is the live data captured from the environment or the offline data stored for analysis. The operations (e.g., dimension reduction, attribute extraction, variable calculations) are performed similarly to those in the learning phase. Then we refer to the signature database and the expected zone of normal behaviors to determine whether

an observed activity is legitimate or malicious. Sometimes *event correlation* is needed to detect complicated attacks.

Three architectures can be taken into account:

- Local analysis: the collected data-under-test is analyzed for security purposes in one probe (i.e., agent) that collects trace from one or several interfaces.
- Centralized analysis: the traffic capture is distributed but the security analysis is centralized. All data sources (i.e., probes) send their collected trace (filtered or not) to the same master server that correlates the traces. It is worth noting that a synchronization among probes is a need in this case.
- Distributed analysis: Each probe perform a local analysis and share the conclusion to each other. This analysis can be very interesting in some specific case studies like ad hoc networks.

Chapter 4

TCP/IP Network Security Monitoring

Contents

4.1 Introduction

Network monitoring is a critical challenging task for a network operator, a service provider or a corporate infrastructure in order to keep the network operation stable, smooth and safe. If the network becomes vulnerable, under attack or breaks down, even for a small period of time, the service provider's ability to deliver secure and high-quality services would be compromised. Network administrators must be proactive rather than reactive. This means monitoring the network traffic and performance at all times, and verifying that security threats do not occur within the network perimeter.

When analyzing the network traffic, it is typically not so important to look at the contents of the packets; rather the information about them, where they are going and how they got there. This "network metadata", often referred to as NetFlow data, can reveal interesting information about your network and often uncover mis-configurations, policy abuses and security incidents. The network metadata, which is obviously more succinct than raw packets in terms of volume, can be a rapid and useful source for detecting abnormal behavior. This technique is able to reduce computations for the monitoring, thus, decreasing the time consumption for each detection. Analyzing network metadata can be also the first step to early recognize malicious attacks before further investigation in detail. Its results probably simplify the investigation over raw data by limiting the analysis to malicious zone-under-test. For example, if network monitoring over metadata has detected a set of abnormal activities of an IP address, then a warning should be raised and other investigation over this IP address should be performed to avoid serious compromises.

In this chapter, we nominate MMT as a polyvalent monitoring tool which is the core of our framework. The advantages of MMT are the flexibility and the extensibility to different domains and technologies, and the scalability with low cost and high performance. Indeed, counting upon the defined security properties, MMT is able to determine the attributes to extract them from network traffic, application logs, or whatever structured-data flows. These attributes can be: (i)-real attributes directly extracted from the inspected input (e.g., protocol field value); (ii)-calculated attributes derived from the calculations of extracted attribute and made available for the security analysis engine (e.g. delays, jitter, packet loss rate); (iii)-meta attributes linked to each packet to describe capture information (e.g., time-stamp).

4.2 LAN monitoring

Local Area Network (LAN) including Wired LAN (Ethernet) and Wireless LAN (Wi-Fi) is very common in daily life. A LAN interconnects computers within a limited area such as a residence, school, laboratory, university campus or office building. In this section, we applied our framework for security monitoring this kind of network in taking into consideration ARP spoofing attack. We witnessed a high proportion of assessed LANs still vulnerable to this classic attack.

4.2.1 ARP spoofing: An attack still alive

4.2.1.1 ARP Definition

ARP is an Ethernet Address Resolution Protocol defined in RFC826 used for converting layer 3- network layer addresses (IP addresses) to link layer 2- link layer addresses (MAC addresses) (Fig. 4.1).

4.2.1.2 ARP spoofing/ poisoning

Each host in a LAN stores an ARP cache containing (IP, MAC) addresses of other hosts. This cache is built upon ARP request and reply. ARP spoofing is the kind of attack in which the attacker sends modified ARP Reply Message so that the victims mis-recognize MAC address of a certain hosts. For example, in Fig. 4.2, the attacker pretends to be POP server. The traffic that the POP client want to send to the POP server would be rerouted

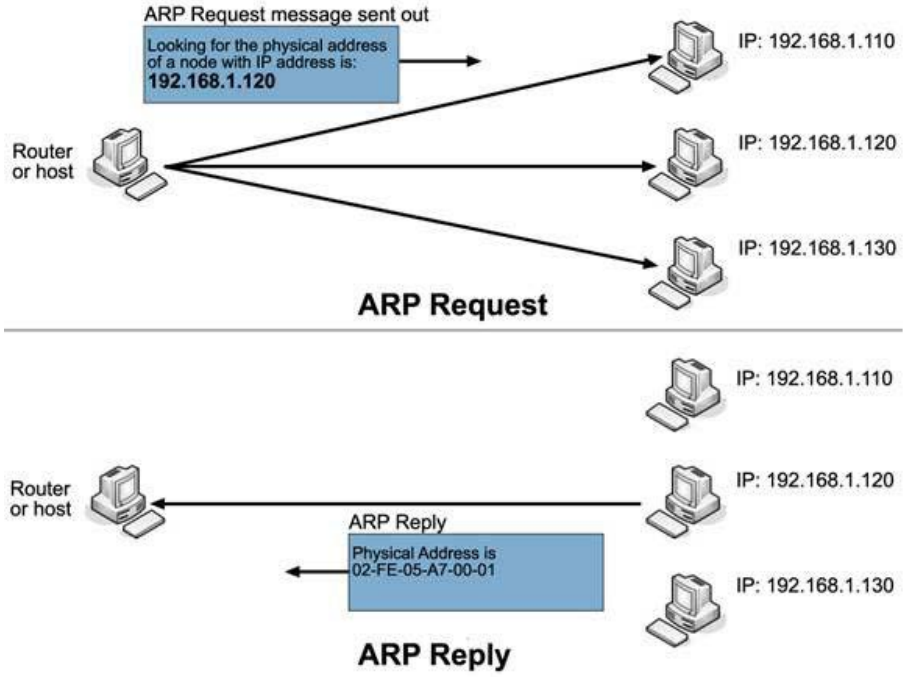


Figure 4.1: ARP: An example

via the attacker. The attacker gains the possibility to deny/ modify this traffic (i.e., Man in the middle attack)

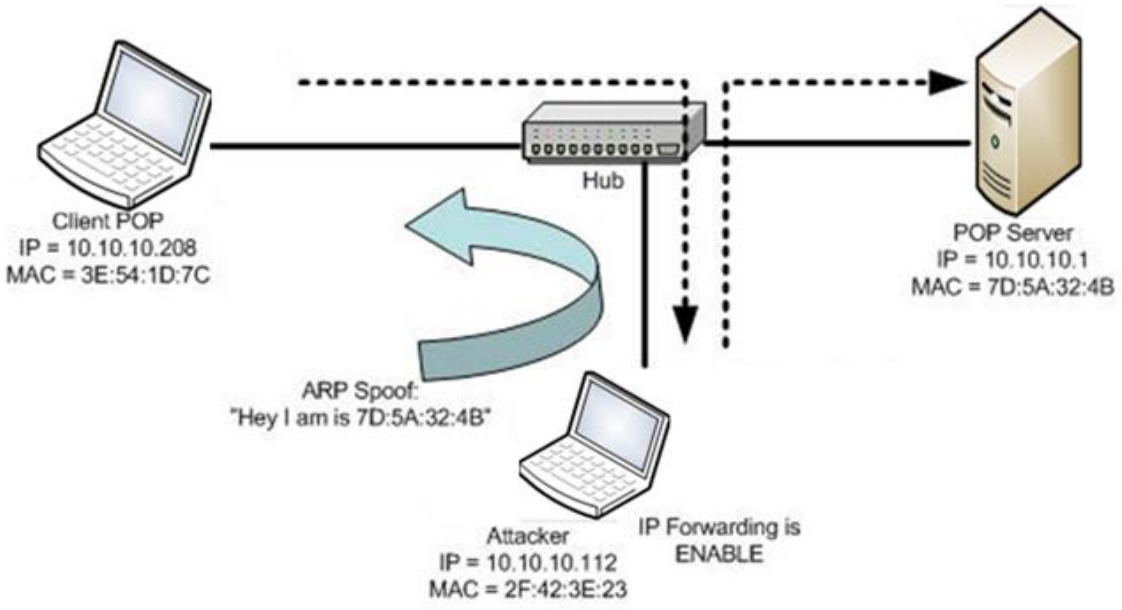


Figure 4.2: ARP spoofing/ poisoning: An example

4.2.1.3 Practical experiment

We used a Ubuntu laptop installed ettercap 0.8.2 [1] to perform ARP spoofing attack. We tried to perform the attack in different real LANs including Wired LAN and Wireless LAN in student residence, campus, commercial center, home, hotel, restaurant, etc. Here below we present the experiment in which our computer (attacker) connected to LAN in our office by a wired Ethernet cable (Fig. 4.3).

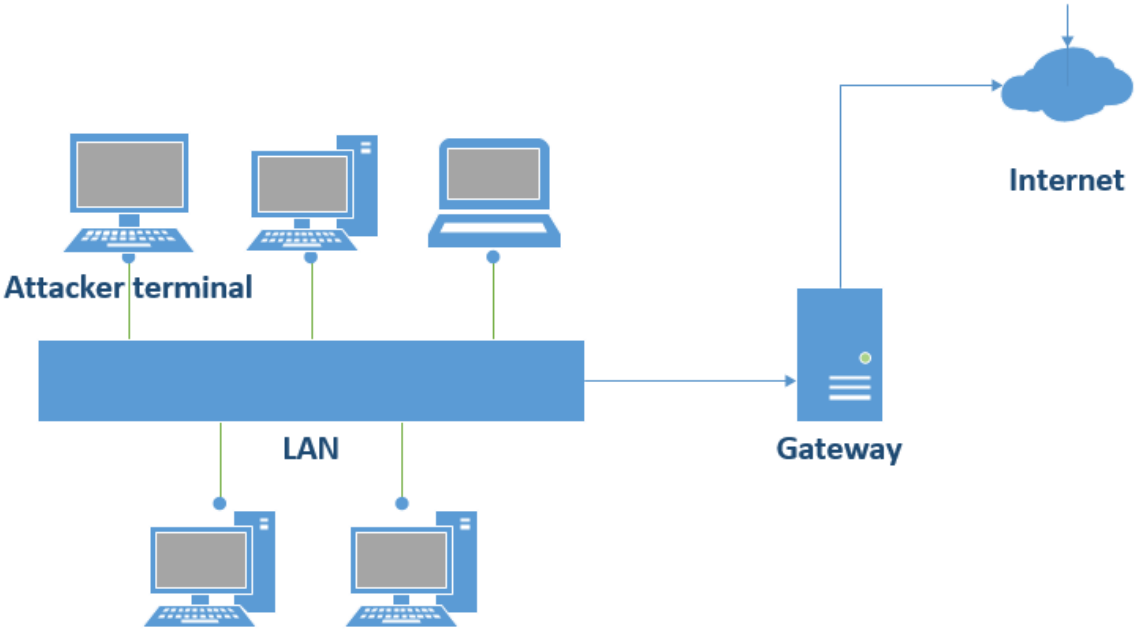


Figure 4.3: ARP case study: Experiment architecture

Using Wireshark to capture the traffic passing by our Ethernet interface (eth0), we admited the success of ARP spoofing attack. We sniffed and analyzed the traffic with the recognition that we could achieve different sensitive information as follows:

Firstly, all the HTTP traffic of poisoned hosts could be recorded and seen in plain text. The primary concern is the privacy. Evidently, almost all users do not want their history of visited websites easily disclosed by someone else. Nowadays, HTTPS allows securing communications between users with must-be-secure web-servers (e.g., the cases of email, online banking, online shopping, etc.). HTTPS traffic is encrypted with a session key which is normally created based on Diffie Hellman key exchange. It is, thus, safe and sound from our attack.

However, it is worth emphasizing that there are still many vulnerable websites using HTTP even for authentication processes. In this case, username and password are easily caught by attackers who stay in the middle and listen to the traffic. Those vulnerable websites are generally not very popular (e.g., forum chat, reselling second-hand object webs, software/driver download webs, etc.). Other trusted-websites such as Google, Facebook, Twitter or banks' websites are certificated with SSL/TLS certificates and obligated to be retrieved under HTTPS connection. However, the problem is that users normally do not

[1] Ettercap: https://ettercap.github.io/ettercap/

create each password for each source but reuse passwords. Some even use the same password for every accounts which increases in number very fast in the modern life. Therefore, if a hacker gains a password extracted from seemingly useless HTTP traffic, he/she can try to use it for other cases and probably gain some sensitive information.

Indeed, in our experiment, we found the ID and password of a colleague when he logged in an insecure website (e.g, http://fr.tchat.tchatche.com/). Having the full power sniffing his traffic (encrypted and unencrypted), we could track his surfing history and steal other web-identities. We could try those passwords to log in his Gmail, school email, Facebook, etc., but we decided not to go further.

Secondly, we deploy the filter to perform a downgrade attack. The idea is to downgrade HTTPS connections to HTTP in which the traffic is not encrypted. For this goal, we use filter to modify captured traffic to force web-servers and clients to use HTTP instead of HTTPS. We did not want to damage the whole network so from the laptop, we performed ARP spoofing only between our own machine and the network's gateway. Then we opened the browser to connect to some popular HTTPS websites namely gmail.com, facebook.com and youtube.com. We figured out that connections were denied and our attack turned out a DoS attack because probably two following reasons:

- The web-servers are configured to use HSTS [2] - a novel protocol enabling web sites to declare themselves accessible only via secure connections and/or for users to be able to direct their user agent(s) to interact with given sites only over secure connections.
- Internet browsers in the victim's computer are well configured to avoid HTTPS/TLS downgrade attack and ignore manipulated HTTP handshakes.

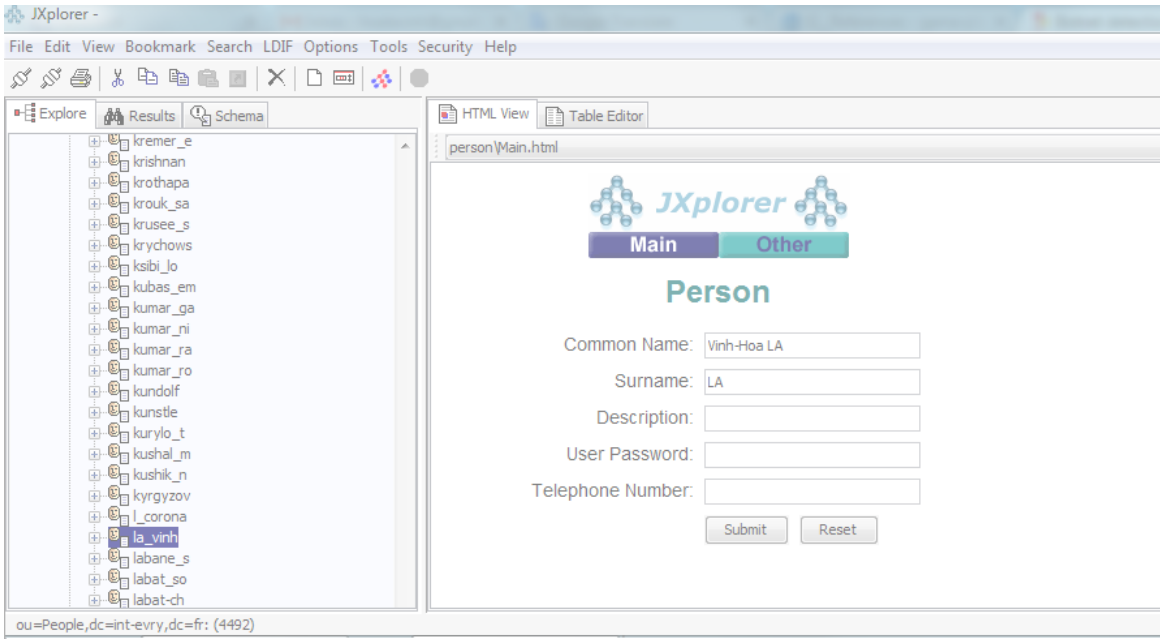


Figure 4.4: JXplorer interface

However, this experiment is still under extension. The filter can be improved. Downgrading HTTPS to HTTP seems feasibly impossible but downgrading from HTTPS/TLS

[2] HTTP Strict Transport Protocol, https://www.ietf.org/rfc/rfc6797.txt

1.2 to a weaker encryption HTTPS/SSLv3 is realizable. In fact, a successful ARP spoofing rewards the attacker a powerful role and he/she can easily drop instead of forward the traffic passed by to trigger DoS attack. Nevertheless, ARP attackers normally silently listen to the network traffic and inject/modify it as their wish instead of brutally kill the service and get attention from network administrators. As the man in the middle, the attacker can also inject a fake certificate. To web browsers, attacker claims to be the secure web server. To the web server, attacker mimics the web browser. We actually used sslstrip[3] and Fiddler2[4] to realize such intention but without success, possibly because two aforementioned reasons. Nevertheless, outdated web browsers and many HTTPS websites are still vulnerable. It is worth re-mentioning that users usually reuse passwords, thus, a single divulged password can be more harmful than expected.

Thirdly, using ettercap for ARP poisoning and sniffing the traffic between every hosts in the LAN with the gateway, we eventually captured several sensitive attainments. Our experiment did not necessitate any complicated filter but some simple rules to extract unencrypted userIDs and passwords. The results obtained were really surprising. We did indeed catch an ID and password concerning LDAP (Lightweight Directory Access Protocol) traffic from the IP address 157.159.10.39, port 389. For further investigations, we installed an open source LDAP client called Jxplorer to connect to that IP address which we assumed a LDAP server. And we did succeed. From the windows of Jxplorer, we could explore a database containing information including *Full name*, *ID* to connect to the IT infrastructures (Fig. 4.4) of all employees and students. It seemed like we could also modified data or even delete but we did not go that far.

Lastly but the most severely, doing the same last-mentioned attack during a whole day, we accidentally received some IMAP passwords corresponding to IDs used to log in the email service which is supposed to use HTTPS. These results were totally out of our expectations because normally such traffic is always encrypted. It is worth noting that after signing in the email service, we can also win the connection to other online services. We believed that there were mis-configurations for IMAP server. Indeed, we figured out the big vulnerability of the email service that it accepted both HTTP and HTTPS.

4.2.2 Countermeasures

There are in reality a wide range of solutions for ARP spoofing either by design (e.g., static IP addresses) or by additional tools/framework (e.g., IDS). Our MMT-based framework can detect such kind of attack in real-time and send an alert (e.g., email) to the administrator if necessary. A demo can be found in [13]. Fig. 4.5 illustrates a simple rule to detect the attack in inspecting the attributes concerning ARP. This is a ready-to-use feature in the original version of MMT.

In addition to measures avoiding the attack, users should be aware of such security threats and train themselves to battle against. The most important thing is to be careful with the personal sensitive information and never enter such data to any website not using HTTPS with a valid certificate.

[3] https://github.com/moxie0/sslstrip
[4] http://www.telerik.com/fiddler

```
<property value="THEN" delay_max="10" property_id="1"
        type_property="ATTACK"
       description="IPv4 address conflict detection (RFC5227). Possible ARP spoofing.">
   <operator value="THEN" delay_max="1">
     <event value="COMPUTE" event_id="1"
           description="ARP who has request"
           boolean_expression="(ARP.OPCODE == 1)"/>
     <event value="COMPUTE" event_id="2"
           description="ARP reply MAC address"
           boolean_expression="((ARP.OPCODE == 2)&amp;amp;&amp;amp;
           (ARP.SRC_PROTO == ARP.DST_PROTO.1))"/>
   </operator>
   <event value="COMPUTE" event_id="3"
         description="ARP reply but with different MAC address"
         boolean_expression="(((ARP.OPCODE == 2)&amp;amp;&amp;amp;
         (ARP.SRC_PROTO == ARP.DST_PROTO.1))&amp;amp;&amp;amp;
         (ARP.SRC_HARD == ARP.SRC_HARD.2))"/>
</property>
```

Figure 4.5: MMT security property example to detect ARP spoofing attack

4.3 WAN/Internet monitoring

The work presented in this section is under the context of the IDOLE project in which we are developing advanced monitoring techniques for detection and investigation using metadata from different sources. This section regards our first work on HTTP which is so far a predominate communication protocol. More precisely, the User-Agent field in HTTP request headers is exposed to various types of security attacks, namely SQL injection, Stored and Reflected cross-site scripting and DoS-type attack. Fast or real-time detection of abnormal traffic concealed among a huge volume of legitimate HTTP traffic is more and more difficult, especially due to the increasingly high rate of network traffic.

The User-Agent is a field of a HTTP request whose value is a string. This string describes the user agent generating this HTTP request (e.g., the browser used by the user) and thus, can be considered as metadata of the user agent. However, attackers can modify the User-Agent field to perform attacks to the entities dealing with HTTP requests (e.g., web-server). Our work attempts to automatically detect the occurrences of the suspect traffic generated by attackers exploiting the User-Agent vulnerabilities. Besides, our detection approach is also useful for malicious traffic corresponding to malware, botnets or virus generated intentionally or unintentionally by infected users or proxies. The experiments proved the practicality of our implementation on both network's live traffic and offline captured packets (e.g., PCAP files). We also displayed the improvements provided by our approach in comparison with SNORT, the classical famous intrusion detection system. Comparing with the common packet analyzer TCPdump, regarding extraction performance, our implementation performed a higher rapidity and a lower resource consumption.

4.3.1 HTTP User-Agent field case study

The User-Agent field is defined by RFC 2616 to identify the user agent generating HTTP requests (e.g., web browsers), so that appropriate data can be returned. It was initially designed for three following main purposes.

- ***Statistical purposes:*** Websites can store a history of visited user agents, then guide developers to the best suitable views with respect to the browsers sending requests. For example, if a website witnesses a dominance of Android user agents, it should be tailored to be better convenient for Android equipment.
- ***The tracing of protocol violations:*** If a website observes a constant signal of error with a particular User-Agent string, the user agent associated should be examined or even blacklisted.
- ***Automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations.*** For instance, thanks to the User-Agent string, the mobile version of the website is returned by the web-server in response to the requests coming from mobile devices. This version should support slower connection and/or a smaller screen with probably a larger font.

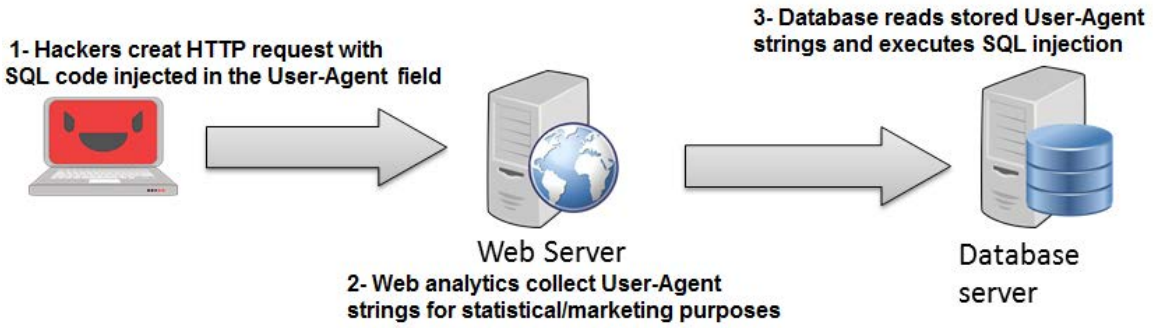


Figure 4.6: SQL injection: a generic example

However, the emerging development of tools which allow modifying User-Agent strings and thus, lying on the kind of browser sending HTTP GET requests, makes the field prone to malicious attacks. Indeed, in [15], they performed a SQL injection attack based on the User-Agent field. It is worth noting that when a web-server receives a HTTP GET request, it extracts the User-Agent string and look it up in a database to provide a suitable return. The User-Agent string can be also stored in another database for statistical/marketing purposes. Cleverly modifying the User-Agent string and analyzing the web-page given back, the authors of [15] could inject their own code and retrieve more and more information about the database, e.g., version, number of tables and columns etc. Step by step, they extended their knowledge and discover the vulnerabilities. Similarly, Fig. 4.6 illustrates another example in which a hacker realized an attack by injecting SQL code to the User-Agent field. If stored User-Agent strings are not validated before being read, injected SQL code can be executed.

Furthermore, modifiable User-Agent field is possibly abused by attackers to trigger Stored and Reflected XSS (Cross-site Scripting). In [16], the author demonstrates a simple

example in which the hacker injects a script to User-Agent strings. Without validation, a web analytical tool can store modified User-Agent strings together with the malicious script that can be accidentally executed by the administrator. It is even not necessary for attackers to send modified-User-Agent strings to the web-server by themselves, there are various manners for them to achieve that goal. They can send malware to the victim user and poison the user's Internet browsers in order to change User-Agent strings or maybe perform a MiM (Man in the Middle) attack and alter HTTP headers via filters. Several tools supporting this attempt are listed in [17] namely XUL, XAML, Active X and Mocha.

It must be noted that SQL injection and XSS are ranked 1^{st} and 3^{th} in the list of top 10 web vulnerabilities according to OWASP (Open Web Application Security Project) Foundation [18]. In addition to them, there are plenty of vulnerabilities based on the User-Agent field. For instance, a DoS attack can be generated if HTTP requests containing intensely long User-Agent strings are continuously sent to a web-server. Furthermore, a hacker presented in [19] his experience to divulge the back-door of a Wireless Broadband D-Link router and surprisingly, the back-door to pass all the authentications of the router is a constant string value of the User-Agent field. The vendor of this router secretly leaves a back-door for probably automatically firmware updating but this back-door was unfortunately discovered.

In short, the User-Agent field is prone to various type of attacks. Besides aforementioned attacks, we witnessed, in extracting User-Agent strings from malware traffic samples, that many malware use specific User-Agent strings in their HTTP request packets. We even extracted the string "ArchitextSpider" when we listened to the real-time traffic in our network. By some simple investigation, we recognized an infected host and the traffic containing that User-Agent string corresponded to a botnet communication. There is another similar experience presented in [20]. In this web-post, the author witnessed that the infected hosts (i.e., bots) made callbacks and communicated with the command and control (C&C) server by injecting what they want to exchange in the User-Agent field. Similarly, there are several works [16,21] [22,23] invested on this topic. They affirmed that User-Agent strings potentially depict an underrated source of abnormal activity detection and analyzing User-Agent strings can be the first step for fast detection before further thorough investigation should be triggered. Although existing proposed method or countermeasures are still manual and intuitive, they are still a very good source of inspiration. A thorough consideration is, however, a need.

4.3.2 Methodology and implementation

Our goal in this work is to validate the usability of User-Agent strings to detect abnormal behaviors hidden amongst a huge volume of legitimate traffic. It is worth noting that there are two kinds of abnormal activities including (i)-attacks directly abusing the User-Agent field to inject/attach evil intention and (ii)-attacks which are not directly related to the field but can be detected based on User-Agent strings. For this goal, we attempt to create a plug-in and several security rules integrated with our original version of MMT to automatically detect those activities.

In reality, MMT is applicable for both network and application level. However, in this case study, we nominate MMT rather as a network monitoring tool. The Fig. 4.7 and Fig. 4.8 represent our methodology to deal with problems provoked by the User-Agent field vulnerabilities. In general, the network traffic is the input of our analysis. MMT permits

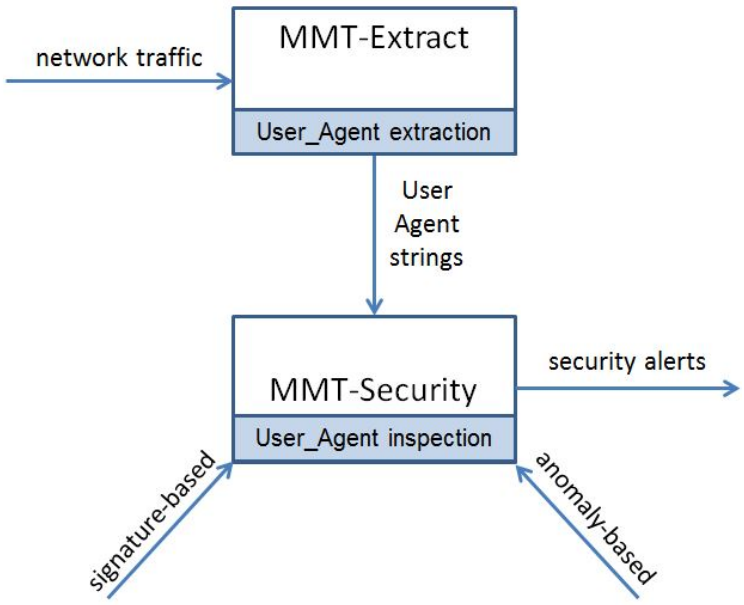


Figure 4.7: Proposed methodology to detect abnormal behavior using the User-Agent field.

capturing a live traffic (Fig. 3.4) or taking PCAP (packet capture) files into consideration. MMT is able to support the extraction of plenty of protocols's attributes (e.g., HTTP, SNMP, SMTP, SMTPs, TCP, UDP, SSL, etc.). In the work presented in this section, we called the function that allows extracting the User-Agent field of passed by/captured HTTP traffic.

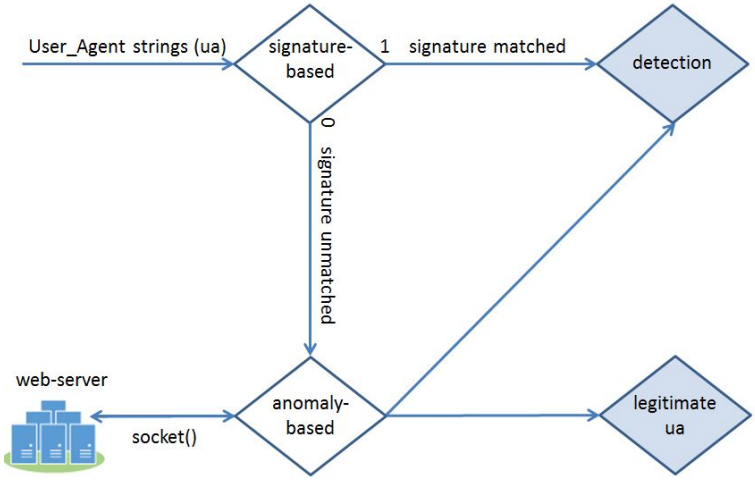


Figure 4.8: User-Agent strings analysis diagram

After that, we developed a plug-in combining signature-based and anomaly-based approaches that allows analyzing and detecting malicious User-Agent strings. It is worth noting that malicious strings consists of: (i)-the ones corresponding to attacks based on the User-Agent field (e.g., SQL injection, Stored and Reflected XSS and DoS) as well as

(ii)-well-known strings corresponding to popular attacks and/or random strings created by chance. The former should be identified by our embedded function whilst, the latter should be referred to our database of already-known evil or benign strings.

Regarding the embedded function, currently we attempt to recognize SQL injection and DoS attack (i.e., based on intense long User-Agent strings). Our current version use "keyword matching" (e.g., DROP, UNION, single quote, double quote, double hyphen, double pipe) to detect the injection. The evasion, if any, will be caught when the string is compared with those in the database. Within the framework of this section, we built our User-Agent strings database based on those from [24] (http://www.user-agents.org/). In reality, the administrators of an organization's network can construct their own database based on machine learning techniques. Especially with organizations in which new application installation policies are very strict, their list of user agents is rather static. However, building such database based on machine learning techniques is out of this work's scope.

Fig. 4.8 illustrates how an extracted User-Agent string (UA) is processed. Two aforementioned type of malicious strings will be detected effectively. In the next sections, we will prove the applicability of the solution in determining such User-Agent strings. However, there are still two issues that we need to consider:

Firstly, if a User-Agent string is created by an exotic browser or an application that performs directly the HTTP queries with a crafted header, this string may not be present in either the sub-database of normal strings or the one of malicious strings. The application/browser may be considered as an attacker. This leads to a false positive. To mitigate this problem, the database should be regularly updated. In a strict network, e.g., the network in a bank office, where users are very restricted in new application installation, the database in this case does not change a lot.

Secondly, because the User-Agent field is modifiable, attacker can use a fake normal User-Agent string for their evil requests. This attack should be detected by other solutions (e.g., other rules of MMT) than the work concerning the User-Agent field presented here. Our goal in this case study is to detect attackers who want to abuse the User-Agent field to perform their evil intention (i.e., SQL injection, DoS attack) and to support the early detection of abnormal entities (e.g., malware, botnets or virus) who use very frequently (until now) their particular User-Agent strings. Detecting a malicious User-Agent string cannot reveal all harmful user agent in the network but can leverage that procedure.

4.3.3 Experimental results

In order to evaluate MMT in general and in the case study of User-Agent strings, we realized a number of experiments with both offline (i.e., PCAP files) and on-line (i.e., live traffic) network traffic. We focus our evaluation on the processing speed as well as the number of false positive and negative in considering the capacity of our implementation to deal with a big volume of traffic. We performed the similar experiments with SNORT [4] in which we disabled all rules except those related to User-Agent strings in order to reduce the execution time. In the scope of this work, we did not compare MMT with Suricata or Bro, other two famous network monitors. Suricata [25,26] is multi-threaded, thus, it allows taking advantage of all the CPU/cores available. Nevertheless, it is not advantageous in the actual case study in which the network traffic is not very huge. Therefore, the utilization of Suricata can lead to a waste of resource. Whilst, BRO [5] is rather complicated to set up and demands a challenging learning curve. It is more suitable for researchers. SNORT

seems to be the best candidate to be compared with MMT. Those two are flexible enough for both research community and real-world network administrators.

4.3.3.1 Experiments with PCAP files

Firstly, we applied our solution and SNORT for the PCAP files downloaded from the website of NETRESE, an independent software vendor with focus on Network Forensics and Network Security Monitoring. The PCAP files contain different malware traffic within normal one. As demonstrated in Tab. 4.1 in which the input contains 214036 HTTP GET packets, we noticed not only the deficiencies of SNORT in terms of detection but also a slight dominance of MMT regarding extraction issue. The reason for the low number of detections of SNORT is that it utilizes only rules identifying blacklisted User-Agent strings, in other words, only a signature-based technique. Therefore, SNORT is incapable against new abnormal behavior. Constructing new rules for SNORT to fix this issue is possible but out of scope of our research. It must be noted that the packet loss rate is calculated as follow:

$$packet\_loss\_rate = \frac{number\_of\_packets\_lost}{number\_of\_packets\_sent} \tag{4.1}$$

Secondly, in order to test MMT and SNORT handling huge network traffic, we realized

	MMT	SNORT
Number of packets	214036	214036
Number of extractions	213978	213794
Packet loss rate	0.03%	0.11%
Number of detections	83209	585

Table 4.1: MMT and SNORT in case of offline traffic

experiments over the dataset captured in Clarkson University [27] by a Linux Mint 17.2 64-bits machine with Intel Core i5-5200U CPU @ 2.20Ghz (2 cores) and 8 GB of RAM. The dataset consists of 80 files PCAP containing 83,850,638 packets with total volume of 39.2 GB. We performed 10 tests and in each test, we measured the time consumption of MMT, SNORT and TCPdump, which is a popular open-source command-line packet analyzer used for network traffic capture as well as the analysis of PCAP files. We applied them to read and extract User-Agent strings in the whole dataset. Tab. 4.2 depicts the results. The processing rate is calculated as follow:

$$processing\_rate(Mbps) = \frac{traffic\_volume(GB) * 1024 * 8}{average\_execution\_time} \tag{4.2}$$

In the first five tests, we ran MMT, SNORT and TCPdump all alone. That means we limited in maximum parallel programs that could consume CPU/RAM resource or network bandwidth. We experienced a small dominance of MMT over TCPdump and a bigger dominance over SNORT. The performance of MMT is really positive with a pretty high processing rate which is approximately 419 Mbps (52 MBps). The Fig. 4.9 illustrates how execution time increases in function of traffic volume. In later five tests, we ran several applications at the same time and we found that TCPdump was highly affected while SNORT just slightly changed. In contrast, we even did not notice any considerable difference over MMT. This fact was explained when we calculated the resource consumed by MMT,

Test N^0	MMT [s]	SNORT [s]	TCPdump [s]
1	807	1010	858
2	835	1004	862
3	743	1219	862
4	783	1006	860
5	720	1003	863
6	739	1005	2181
7	758	1143	2227
8	730	1283	2013
9	740	1307	2574
10	807	1212	2304
Average	766.2	1119.2	1638.4
Processing rate (Mbps)	419	287	196

Table 4.2: Execution time and processing rate of MMT, SNORT and TCPdump in reading PCAP files

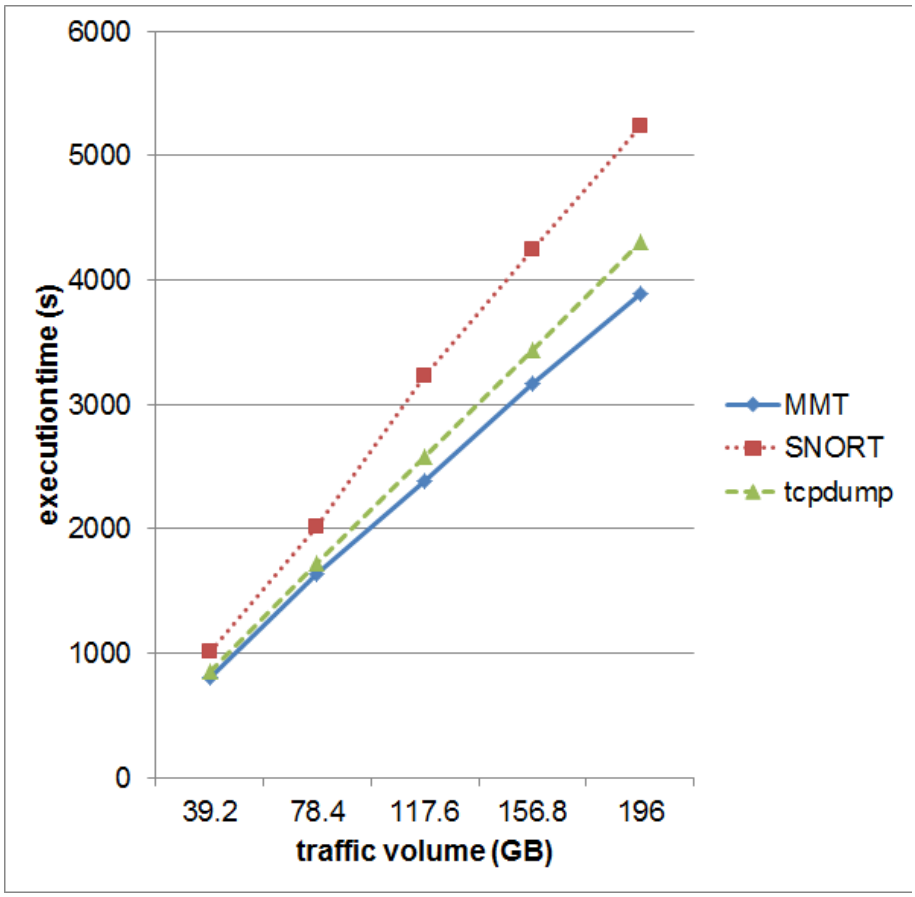


Figure 4.9: Execution time of MMT, SNORT and TCPdump in function of traffic volume

SNORT and TCPdump separately to read the same PCAP file of 300MB (Tab. 4.3), we observed that MMT expended less resource than the others (the memory consumption differences were relatively small but CPU usage of TCPdump was considerably higher than SNORT and MMT).

	MMT	SNORT	TCPdump
CPU usage	3.4%	4.5%	6.0%
Memory consumption	12.8%	13.3%	13.0%

Table 4.3: Average resource consumption of MMT, SNORT and TCPdump

4.3.3.2 Experiments with live traffic

Test N^0	SQL injection	DoS	Random UA	Known malicious UA	
	MMT [ms]	MMT [ms]	MMT [ms]	MMT [ms]	SNORT [ms]
Test 1	0.901	0.735	0.868	0.776	0.920
Test 2	0.790	0.655	0.773	0.938	0.939
Test 3	0.700	0.555	0.704	0.881	0.942
Test 4	0.590	0.443	0.645	1.118	0.967
Test 5	0.482	0.192	0.988	1.116	_
Test 6	0.334	0.109	0.934	1.117	0.927
Test 7	0.167	0.978	0.870	1.052	0.959
Test 8	1.002	0.874	1.109	0.851	0.989
Test 9	0.895	0.783	1.136	0.944	0.993
Test 10	0.810	0.695	1.142	0.906	_
Average	0.667	0.602	0.917	0.970	0.955

Table 4.4: Detection latency of MMT and SNORT

We used a personal computer running Ubuntu 14.04.3 LTS with 4 * Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz processor and 8GB of RAM. We used also the Mozilla Firefox's Add-on named TAMPER DATA to edit manually the User-Agent field and thus, to generate malicious HTTP requests. In addition, we created a simple C application that was able to read normal/abnormal User-Agent strings prepared in a text file and passed the HTTP requests containing them to a web-server. They were captured by MMT when they went through Ethernet interface (i.e., eth0). The goal is to verify whether our solution is influenced if HTTP requests are generated at low (i.e., in case of TAMPER DATA) and high speed (i.e., in case of our own application).

The Tab. 4.4 and Tab. 4.5 present the results when we produced a number of HTTP requests containing both legitimate and malicious User-Agent strings. We used four categories of malicious strings including ones identifying SQL injection and DoS-type attack as well as random and known malicious ones (e.g., User-Agent strings of HTTP requests corresponding to famous attacks or malware).

In regards to the detection latency, we noticed that our solution is able to detect in real-time abnormal User-Agent strings that very probably belong to malicious HTTP requests. The signature-based detection approach applied on SQL injection and DoS allows a little faster (30%) responses in comparison with the approach based on our database User-Agent strings which is used to raise alerts over random and known malicious User-Agent strings. Whilst, the current handle of SNORT for detecting malicious User-Agent strings seems limited. Indeed, we experienced an incompetence of SNORT regarding every other types of User-Agent strings except known malicious ones. Even for this category, SNORT missed some evil objects. We were not surprised because of the results seen above in the

experiments over offline traffic and the reason aforementioned. The rules about blacklisted malicious User-Agent strings cannot cover new and random strings. In similar detections, SNORT and MMT anyhow showed generally the same latency. With respect to the number

	MMT	SNORT
Number of extractions	212	212
Number of detections	40	8
False positive	0	0
False negative	0	32

Table 4.5: False positive and false negative of our solution and SNORT

of false negative (Tab. 4.5), the weakness of SNORT is obvious. The exigence of a regular update limits SNORT on the subject of new attack detection. In this point, our proposition with MMT proves a dominance based on a heterogeneous approach that allows detecting both known and unknown threats. MMT will not be greatly affected by outdated database or rules in terms of false negative thank to anomaly-based sub-module. In this practical experiment, the false positive in case of MMT and SNORT are equal. MMT is, in fact, more likely prone to trigger false positive alerts than SNORT, especially with exotic browser/application as explained in section 4.3.2. However, in any case, a false positive is still better than a false negative.

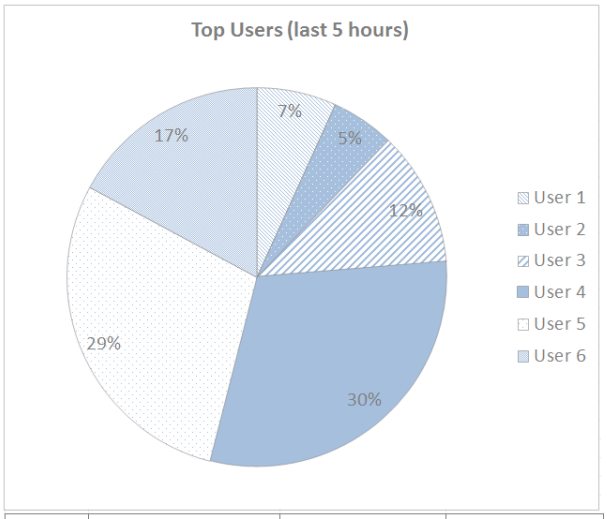
4.3.3.3 Discussion

Analyzing the User-Agent field in HTTP requests has attracted the attention since recent few years. As far as we know, there was no official conference or journal paper validating the usability User-Agent strings as a good source for network monitoring. The idea of using them to gain more knowledge about the network as well as their vulnerability are only presented in some technical reports and blogs.

As an example, according to [21], up to 75% of Web application scanners are incapable to catch HTTP headers parameters related flaws. This article confirms that several fields of HTTP headers including the User-Agent field are exposed to SQL injections. Examples of attacks and tools were briefly presented. The author introduced also a list of 14 scanners that analyze web-based applications source-code and detect their vulnerabilities. This work, thus, focuses on source code analyzing and potential vulnerabilities detection rather than network monitoring and intrusion detection.

In [23], [16], the authors affirmed the vulnerabilities of the User-Agent field and presented their analysis on offline network traffic (i.e., PCAP files). They proved the relevant relationship between malware/abnormal activities with strange User-Agent strings collected from their own network. Especially in [23], VRT (Sourcefire Vulnerability Research Team) studied their network traffic and observed that unique or less-appeared User-Agent strings are very probably linked with suspicious activities. In [16], the authors also proposed a list of tools (e.g., Wireshark, Snort and TCPdump) which could be used to manually hunt down malicious user agents in analyzing their own organization's network traffic. A number of tutorials showing how to edit and inject User-Agent strings as well as how to catch and inspect them were presented. Both of two aforementioned works concluded that User-Agent strings could be a potential source for detecting malicious behavior in networks. Their approaches were, nevertheless, still intuitive and rather manual.

We proposed in this section an automated solution based on MMT that allows real-time detection of malicious User-Agent strings. In the presented case study, we concentrate only on security issues. In practice, looking for abnormal behavior is not the unique use case for MMT. MMT can also monitor user activities and troubleshoot the network (e.g., discover an employee using Skype, or spending too much time watching YouTube instead of working). The Fig. 4.10 depicts an example concerning top network traffic consumers. Another example related to the User-Agent case study is that we can look at the respective traffic of each user agent to figure out common user agents in the organization's network. The website and other service developers should tailor their products to better adapt those user agents'characteristics. Not only HTTP but also other network protocols can be taken into account to provide an improved view in terms of both depth and breadth.



User	Service	Traffic volume (GB)	Percentage (%)
User 1	Finance	3.4	6
User 2	Human Resources	2.7	5
User 3	Marketing	5.8	11
User 4	IT	15.2	30
User 5	Engineering	14.5	28
User 6	Student Labs	8.6	17

Figure 4.10: An example on using metadata to monitor users'activities

The aforesaid experiments proved that our proposition provides, in the context of the User-Agent case study and in general network monitoring, some outstanding features as follow:

- **Heterogeneous intrusion detection approach:**
  SNORT takes the User-Agent field into account but SNORT simply defines rules over blacklisted User-Agent strings that must be updated regularly with new ones. This is truly a huge weakness of SNORT because the fact that malware is being generated faster than people can keep up with. Writing adequate signatures for detection is thus very difficult. Our methodology combines both two classic intrusion detection approaches including signature-based and anomaly-based. In theory, the limitation of signature-based approaches is the incapacity over new attacks. That of anomaly-based approaches is the high rate of false positive. Our solution allows detecting both wanted (e.g., security rules) and unwanted (e.g., attacks) behavior. That means we

can detect even new suspicious User-Agent strings and also reduce the number of false positives.

- **High-speed extraction and real-time detection:**
  Experimental results in case of on-line and offline traffic display positive enhancements of our solution against SNORT and TCPdump regarding both extraction speed and real-time response to suspicion. This fact suggests the potential capability of MMT about the passage to large scale monitoring.

- **Attribute extraction and legal problems:**
  Our technique does not require the payload of packages. The private data contained in payload is therefore not inspected, then our approach is applicable for high privacy communications. Thanks to the fact that we extract only attributes (i.e., User-Agent field) instead, the amount of information required is reduced. This leads to the ability of utilization on large networks even when real-time processing is a must.

Moreover, our solution is always under improvement and extension. As mentioned, MMT is able to work over a wide range of network protocol fields. It allows the extraction of various performance indicators and attributes of not only protocol packet formats but also structured application generated messages (e.g. traces, logs). MMT is applicable for even applications using non-standard port numbers (e.g., P2P, Skype). As a modular solution, the utilization of MMT is flexible and we can combine MMT with other systems. MMT-Security uses rules to define security properties which are feasible to be correlated with other MMT rules or even other rules of different systems to passively monitor or actively generate tests. The goals of MMT are not uniquely to detect anomaly or complicated attacks but also to verify access control policies or to report performance parameters.

Detecting a malicious User-Agent string is evidently not enough to determine a harmful user agent. But it is still a very good starting point of network traffic inspection. The related IP address and/or domain, payload data sent and received by this host and other correlated hosts should be investigated. It is worth reminding that a proxy server or an infected web-browser/operating system can rewrite HTTP headers that are in transit. Our detection approach covers two kinds of threats: (i)-attacks in which attackers modify intentionally the User-Agent field in order to perform their evil intention (e.g., SQL injection, Stored and Reflected XSS, and DoS) and (ii)-malicious traffic corresponding to suspicious threats (e.g., malware, botnets or virus) generated intentionally or unintentionally by infected users or proxies. For example, statistics show that a malware usually uses random User-Agent strings and in reality, many signatures of User-Agent strings used by well-known attacks are also registered. All of these will be detected by our solution. After identifying a malicious host, a maximum information about it should be collected and a thorough investigation will return it's real intention.

4.4 Framework extension

4.4.1 An extension from HTTP User-Agent field case study

This section aims to provide an extension of the framework validated for HTTP User-Agent field case study. More specifically, we optimize the framework for a LAN (e.g., office, home, company, institution network) in which the set of applications/browsers generating HTTP

requests is supposedly fixed after a long enough duration of utilization. This assumption costs us virtually nothing because indeed, in many companies, the computer provided to employees are strictly installed only with a pre-defined list of verified programs. If the user wants to install a new one, he/she must contact the IT service. Sometimes, they even restore the computers back to their original configuration each time the computers restart. The work done by the employees is stored online.

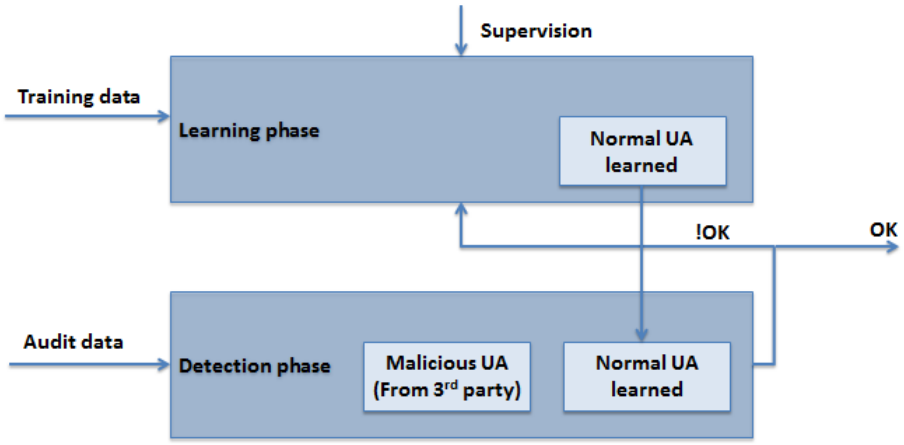


Figure 4.11: User-Agent case study extension

The aim of the learning phase is therefore to learn and create a database of normal User-Agent strings (Fig. 4.11). This phase can be a supervised learning thanks to the support of experts. The detection phase inherits this database as well as integrates another set of malicious signatures (e.g., from 3rd party) to leverage the detection. If an audit User-Agent string is determined abnormal, it can be learned against with the inspection of IT service.

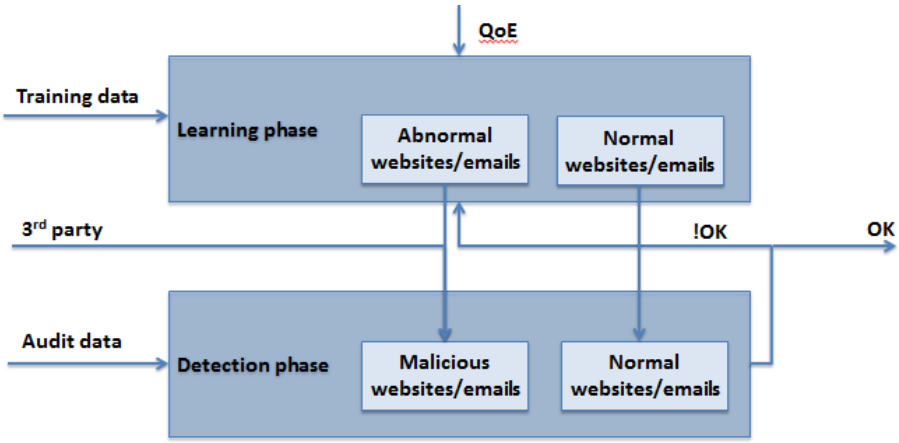


Figure 4.12: QoE-based web pop-up and spam avoidance

4.4.2 QoE-based web pop-up and spam avoidance

Another idea is taking into consideration the QoE (Quality of Experience) to determine the normal behaviors. The Fig. 4.12 illustrates the framework to avoid web pop-up and spam.

The QoE parameter can be based on the reaction of the users in opening a website or an email. For example, the quicker a website is closed or an email is deleted, the more probably it is a spam or an unwanted object. Normal and abnormal entities will be learned in addition to the signature database from third parties. This framework can be implemented in the form of a browser's adds-on (Firefox) or Extensions (Chrome) that takes into account the Internet traffic and browser's logs.

4.4.3 Smartphone-based security monitoring

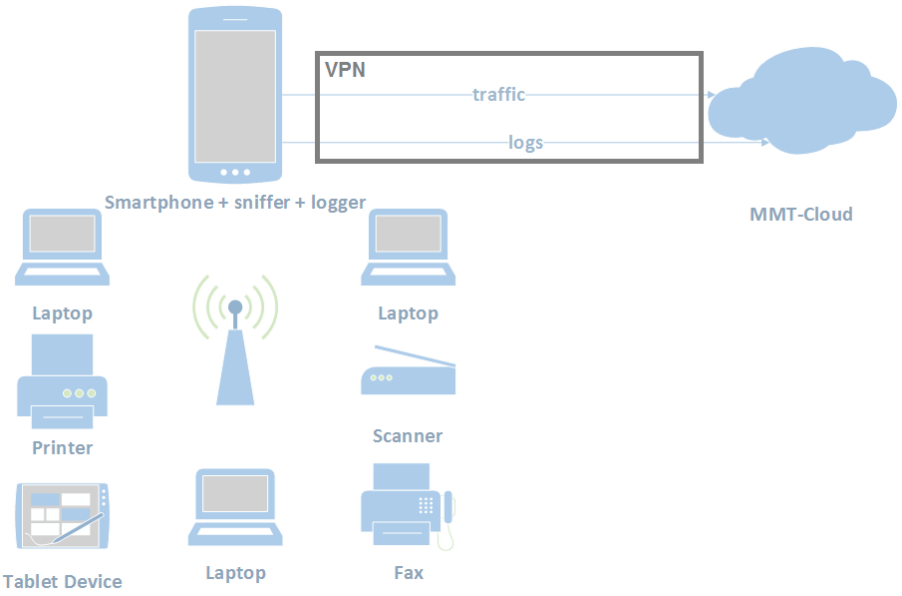


Figure 4.13: Smartphone-based security monitoring example

The Fig. 4.13 illustrates our proposition for security monitoring a smartphone and the Wi-fi network that it connects to. The main idea is to deploy a sniffer and/or a logger making a systematic recording of network traffic, events, observations, or measurements. The aforesaid traffic can be the packets passing by the wireless interface of the smartphone. However, due to the wireless communication characteristics, the smartphone can sniff also other messages (not encrypted) broadcasted in the Wi-fi network inside its radio range. Analyzing this traffic is useful not only for the smartphone itself but also the whole network. Whilst, the logs can be related to sensitive application containing personal information (e.g., username, password, bank account).

The smartphone can connect and send the audit data to MMT-Cloud via a VPN connection [5]. MMT verifies whether there is any security issue, e.g., the utilisation in case necessary of HTTPS, the content payload of the traffic concerning sensitive information must be encrypted, etc.

[5] https://mmt-cloud.montimage.com/docs/setup-vpn-client

Chapter 5

6LoWPAN-based IoT Security Monitoring

Contents

5.1 Introduction

There are nowadays about 15 billion devices on the Internet of Things (IoT) and there would be 50 billion connected devices by 2020 according to a report by Cisco and DHL [28]. As a representation, WSNs are more and more widely used in various domains, e.g., to monitor physical and environmental conditions in the regions where human access is probably limited. Smart cities are also another application domain based on the collaboration of a number of WSNs. However, researchers observed three principal difficult challenges in designing and implementing a secure WSN, namely:

- The vulnerable characteristics of wireless communication nature:
  For example, eavesdropping, unauthorized access, spoofing, replay and Denial of Service (DoS) attacks, etc.

- The severely resource-constraints of sensor devices: Typical WSNs are composed of a large number of low-power tiny sensors and actuators. Those nodes own typically limited energy lifetime, slow embedded processors, severely constrained memory and low-bandwidth radios. For example, Waspmote [29], the modern open source sensor device distributed by Libelium, contains simply a 14 MHz micro-processor, 3.3 V - 4.2 V battery voltage, 8 KB SRAM, 128 KB flash memory and 4 KB EEPROM to save sensed data and to run an operating system and application programs. These resource constraints limit the degree of encryption, decryption, and authentication that can be deployed, thus, the concept *security* and *WSNs* sound likely contradictory.

- Additional physical security risks: WSNs are commonly deployed in inaccessible terrains or unattended and even hostile environments to sense data or to observe the occurrence of certain events. They can self-organize into an ad-hoc style wireless network that collects and forwards sensor data to an information sink (e.g., a base station acting as a gateway to the wired network).

To date, the research on IoT/WSNs is mainly focused on how to make the concept IoT realistic and practical. In the other words, most of the research works are enabling this technology by standardizing the communication protocols, ameliorating the performance of the IoT systems, optimizing the resource consumption, etc. Security is always considered as an important issue but difficult to thoroughly achieve because it seems contradictory with the system's performance.

Moreover, the research on IoT security mostly concentrates on **designing** secure communication protocols, light encryption, authentication, ect. For example, Fig. 5.1 displays the complete security scheme proposed by Libelium that deals with common security issues including *access control (privacy), authentication, data confidentiality, data integrity, data freshness (avoiding packet injection)* and *non-repudiation*. In general, there are so far several security propositions for 6LoWPAN-based IoT:

- Hop by hop security: TinySec, Minisec, ContikiSec, IEEE802.15.4 security mechanism, WSNSec.

- End-to-end approach: WSN-ETESec

Recently, it exists more and more research works on monitoring in general and intrusion detection in particular for IoT/WSNs. However, some existing approaches are still in design level and not implemented yet. Some others focus only in routing problem (e.g., Foren6) or seemingly affect the performance of the systems (e.g., SVELTE). Therefore, security monitoring with the minimum influence to the running system is the topic that we study in this chapter.

5.1.1 6LoWPAN overview

Traditional battery-powered networks or low-bit-rate networks (e.g., 802.15.4) were considered incapable of running IP due to their typical characteristics:

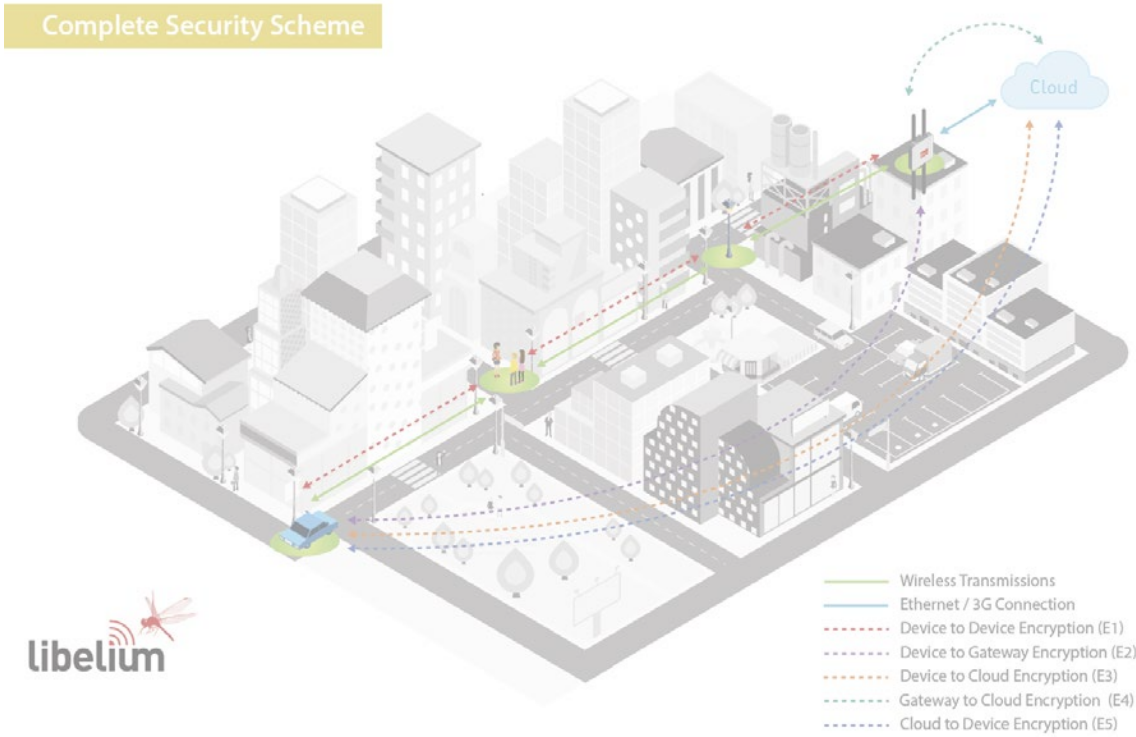


Figure 5.1: Complete security scheme proposed by Libelium

- Limited processing capability: From 8-bit clock speed processors.
- Small memory capacity: From a few kilobytes of RAM with a few dozen kilobytes of ROM/ flash memory.
- Low power: From a few dozen of milli-amperes.
- Short range: Normally from 10 meters to 100 meters.

Whilst, a huge majority of LANs and WANs are running IP. As a result, 6LoWPAN has been designed to work on top of 802.15.4 networks as an adaptation layer which makes the layer 2 compatible with layer 3 routing and inter-network technology. 6LoWPAN supports uniquely IPv6 (no IPv4 support available) and is promising to allow low-power and lossy devices connecting to other IP-based networks, without intermediate entities like translation gateways or proxies. This success will enable reusing existing IP-based technology including tools for monitoring, diagnostic and management.

6LoWPAN standards [30] are basically completed. Fig. 5.2 demonstrates the protocol stack of a 6LoWPAN including following standardized protocols [31]:

- 6LoWPAN: IPv6 over Low-Power WPAN (RFC 4919, Aug 2007) [32]
- RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks (RFC 6550, Mar 2012) [33]
- CoAP: Constrained Application Protocol (RFC 7252, June 2014) [34]

- DTLS: Datagram Transport Layer Security (RFC 6347, January 2012) [35]

6LoWPAN standards are still being complemented to satisfy routing needs and to extend to other link layer technology. A 6LoWPAN typically includes devices realizing a combined

TCP/IP protocol stack				6LoWPAN protocol stack	
HTTP	SSL/TLS		Application	CoAP	DTLS
TCP	UDP	ICMP	Transport	UDP	ICMP
IP			Network (Adaptation)	RPL	
				6LoWPAN	
Ethernet MAC			Data Link	IEEE 802.15.4 MAC	
Ethernet PHY			Physical	IEEE 802.15.4 PHY	

Figure 5.2: 6LoWPAN protocol stack in comparison with TCP/IP

work: collecting the physical or environmental parameters and send them to real-world applications. The most seemingly popular devices are wireless sensors, although a 6LoWPAN is not necessarily comprised of sensor nodes only, but also actuators. Fig. 5.3 illustrates an example of typical WSN/IoT solutions proposed by Libelium. Data collected from sensors can be stored in a local or external database which will be queried by cloud-based applications.

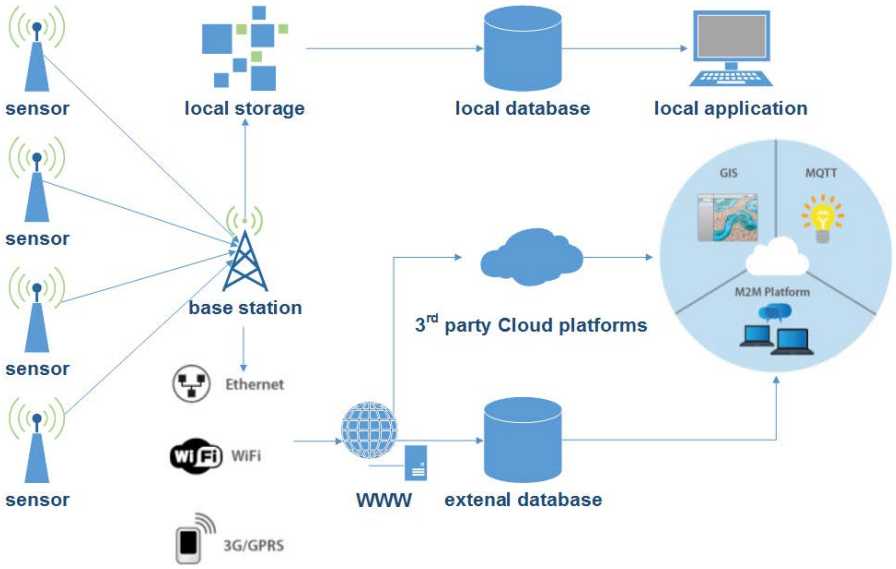


Figure 5.3: An example of actual application of WSN/IoT

However, in reality, sensors are usually affected by noises, mis-configurations, and other malicious nodes. In addition, 6LoWPAN devices are themselves unreliable due to various reasons, namely uncertain radio connectivity, battery drain, device lockups, physical tampering. Network monitoring and detecting abnormal activities become therefore a need.

5.1.2 IoT/WSNs Security requirements

- Availability: the network should be available even if it is under an attack with a minimum performance degradation. In some cases, it can be understood as the *resilience* or *survivability* against attacks.

- Confidentiality: ensure that the data is only readable by the proposed destination. This requirement is closely related to the concept *authentication*.
- Integrity: ensure that the information contained in the original message is kept intact. In the other words, data or messages delivered among nodes are not altered by attackers.
- Privacy: ensure that only the desired sensor devices and gateways are part of the network. Other relevant security requirements are *anonymity*, *liability* and *trust*.
- Authentication: ensure that the supposed sender is the real sender.
- Non-repudiation: a sender should have mandatory responsibility in transmitting the messages for the investigation that will determine the correct sequence and content of messages exchanged before the security incident.

5.2 MMT Adaptation for 6LoWPAN-based WSNs

Having been standardized by IETF, 6LoWPAN-based WSNs consist of Low Power objects equipped with sensors. They use IEEE 802.15.4 as the physical layer standard. However, they are exposed to various types of security threats due to the intrinsic characteristics and the lack of security considerations in the design of protocols for them. The failure of nodes may result in network partition, decreasing the cover ratio, reducing the availability of the sensor network and even producing the entire network failure. An adapted monitoring tool that takes into account the particular characteristics of 6LoWPANs (e.g., resource constraints) is therefore a need.

Nevertheless, to our knowledge, there has not been any official monitoring solution for such kind of networks yet. The initial propositions concentrate only in routing issues and they are likely impossible to allow a deep inspection on the network traffic. We aim to fulfill this missing. Indeed, we have adapted our original version of MMT which has been well working over TCP/IP networks [1, 13, 36]. Our goal is to consider not only theoretical topology of the network but also ready-to-use elements in network traffic to monitor itself (i.e., passive monitoring). Avoiding creating additional traffic, which is costly in 6LoWPAN, is an important priority throughout our work. We validate MMT integrated with new 6LoWPAN plugins over a real test-bed in analyzing real-world 6LoWPAN traffic. Experimental results prove the applicability of our tool which can be useful for both research community and industrial companies.

5.2.1 MMT plugin for 6LoWPAN

Attempting to adapt MMT for 6LoWPANs, we have built several 6LoWPAN-plugins in addition to the original version working properly over TCP/IP networks. These plugins take into consideration the encapsulation and header compression mechanisms that allow IPv6 packets able to be sent to and received over IEEE 802.15.4 based networks. Attributes and protocols can be thus recognized and extracted for being analyzed. To the best of our knowledge, existing monitoring tools and IDS (Intrusion Detection System) (e.g., Suricata, SNORT) have not provided any official support to IEEE 802.15.4 or 6LoWPAN yet.

Fig. 5.4 presents an example of a packet captured while nodes were exchanging topology information for routing. It should be noted that there are three different header structures

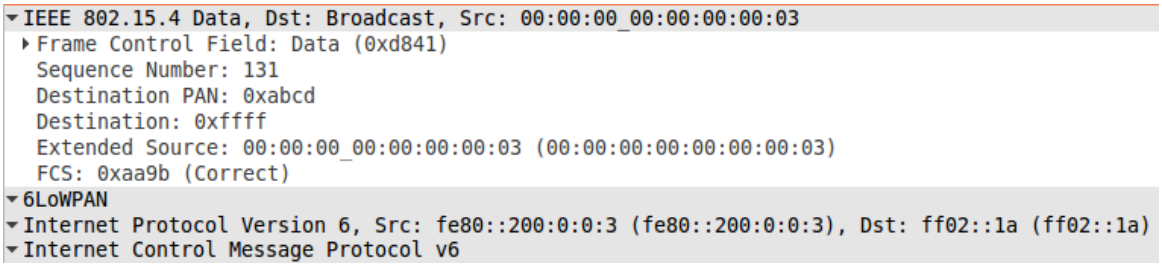


Figure 5.4: A sample captured packet with IEEE 802.15.4 fields

```
Protocol {
        Properties {
                label = "IEEE802154"
                id = "802"
                context = "false"
                session = "false"
                encapsulation = "true"
                encoding = "network"
        }
        Attributes {
                struct header {
                        uint16_t FrameControl;
                        uint8_t SeqNum;
                        uint16_t DESPAN;
                        uint16_t DEST;
                        uint64_t SRC;
                        uint16_t FCS;
                    }
        }
}
```

Figure 5.5: Attribute definition for IEEE 802.15.4

corresponding to IEEE 802.15.4 ACK packets, IEEE 802.15.4 DATA Unicast packets and IEEE 802.15.4 DATA Multicast packets. The field "Frame Control" plays the role of their identifier. Fig. 5.5 displays a simple description about the header's structure of the IEEE 802.15.4 DATA Multicast packets. This is only a simple straw description used to generate the skeleton of the plugin which is in C language.

Our plugins aim to cover all possible structures of packets. Fig. 5.6 briefly resumes our plugins and their supporting protocols (i.e., packet structures) at the time of writing this thesis. They include already-done ones (black boxes), almost-done and under-tested ones (blue boxes), and to-be-done one (yellow box). For the moment, we are mostly focusing on routing control packets that can identify efficiently the network's state. For the long term goal, we would like to verify other protocols in higher layers, especially security-related-protocols (e.g., DTLS).

Actually, building a new plugin for any structured data/ traffic/ event logs is a feasible task. Researchers and industrial network administrators can build the plugins themselves taking into considerations their own interesting data to extract. Montimage provides supporting tools to create skeletons for new plugins based on pre-defined attributes which are in need of being extracted.

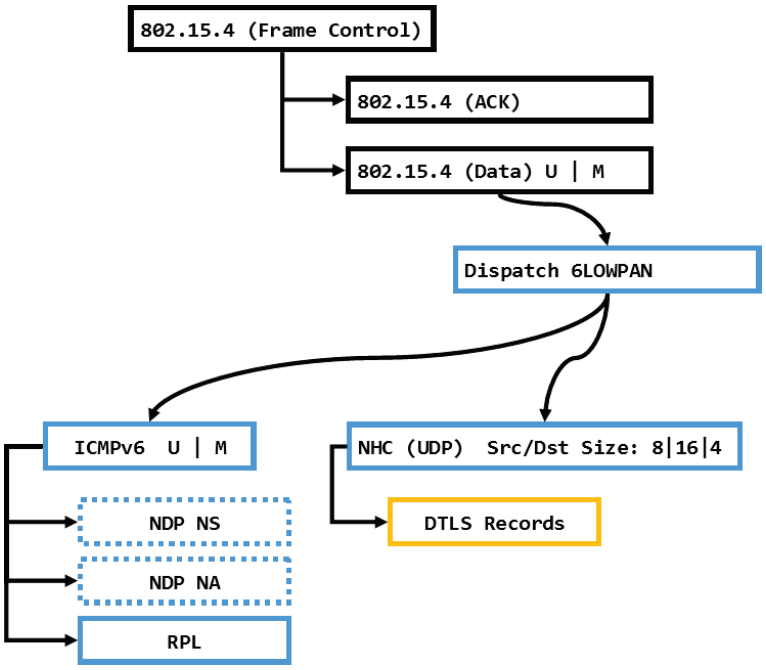


Figure 5.6: List of MMT plugins corresponding to supported protocols

5.2.2 Related Work on 6LoWPAN monitoring/intrusion detection

As far as we know, there have been not many monitoring tools for IoT in general and for 6LoWPAN-based WSNs in particular. Foren6 is seemingly the most famous one which is compared in the section above. However, the current version of Foren6 mainly focuses in visualizing the network topology and analyzing routing issues. Whilst, MMT is an extendable monitoring tool that allows adding plugins to define new input as well as writing rules describing both wanted and unwanted behaviors from the input. This flexibility makes MMT open to different types of input as well as able to adapt to different scenarios.

Regarding research works dealing with 6LoWPAN objects's security, SVELTE [37] has been presented as the most well-known among very few intrusion detection tools working over such small devices. SVELTE consists of three main centralized modules including lightweight modules and mini-firewalls deployed in SNs and central modules called 6Mapper located in BRs. 6Mapper collects the routing information thanks to their "little" collaborators located in SNs. Experiments have been carried out by the authors and their team to evaluate SVELTE. In comparison with our approach, SVELTE is more active and creates additional traffic to realize their goal. Whilst, we attempt to passively monitor the network based on the network's traffic to avoid additional costs which might hamper 6LoWPAN.

Concerning IDS solutions proposed for WSNs which are not specifically based on 6LoWPAN technology, some of the most important ones are studied in [38]. Nevertheless, similarly to SVELTE, they have a common issue: IDS modules installed in nodes use the same wireless medium to communicate among themselves. This fact triggers additional cost that could be expensive and even unaffordable in case of DoS attacks (e.g., jamming or flooding). Consequently, detecting attacks, the primary task of an IDS, becomes inefficient or even unachievable. To overcome this problem, the authors of [39, 40] proposed DEMO as an IDS framework combining several existing open source technologies. They focused especially on

DoS attacks detection by real-time monitoring of various physical parameters. They integrated to the 6LoWPAN some distributed IDS probes acting as sniffers. These IDS probes have the mission to send relevant information to the IDS (e.g., Suricata) through wired connections. Wired connectivity, which is certainly more reliable, allows the framework to be resistant to attacks. However, WSNs are frequently deployed in inaccessible terrains or even hostile environments. A wired connection in that case is seemingly unrealistic. Meanwhile, MMT is able to play the same role of Suricata in DEMO framework, without the need of decoders (Suricata does not officially support IEEE 802.15.4 and 6LoWPAN yet).

5.3 Detection methodology and algorithm

In this section, we summary our methodologies and algorithms to detect anomalies. The learning phase is realized by utilizing supervised learning approach, i.e., we knew the label (normal or abnormal) of the audit traffic before learning it. More specifically, we propose two detection algorithms, one based on statistical learning and another based on information theory (entropy).

5.3.1 Misbehaving node detection algorithm based on statistical learning

We suppose that s is a sink node (i.e., base station node, gateway) and n_i is the i^{th} sensor node. For a node n_i at the moment t, $W_i(t)$ denotes the weight of the link between n_i and s. Depending on real-world case study and requirements, W_i can be defined and calculated differently.

Our detection algorithm consists of two phases: ***learning phase and monitoring phase***.

5.3.1.1 Learning phase

We assume that $W_i(t) \sim N(\mu_i, {\sigma_i}^2)$, i.e., W_i is distributed normally with mean μ_i and variance σ_i. $N(\mu, \sigma^2)$ is the normal (or Gaussian) distribution in probability theory [41].

According to *3-sigma rule*, approximately 95% and 99.7% of values drawn from a normal distribution lie correspondingly within two and three standard deviations σ away from the mean μ. This percentage increases according to the gap away from the mean. In case of 7σ, the percentage approaches up to 99.99999999974%. In other words, the probability that X is within $[(\mu - 7\sigma), (\mu + 7\sigma)]$ is high up to 0.9999999999974.

In the learning phase, we assume that every node functions normally. This phase should be performed right after the sensor network starts operating. In fact, multiple attempts in learning phase could be useful to identify "the most common normal status of the network" thus, to determine the best values for μ_i and σ_i for the node n_i. We define then $[(\mu_i - \varepsilon_i), (\mu_i + \varepsilon_i)]$ as the promising interval that W_i should lie within. ε_i is a customizable parameter which defines the frontier between normal and abnormal behaviors. Its value is generally from 3σ to 7σ.

5.3.1.2 Monitoring phase

In this phase, we listen to the network and calculate $W_i(t)$ for every node. We evaluate whether a node n_i is normal or abnormal by comparing $W_i(t)$ with μ_i defined in the learning phase.

- *Step 1: Malicious path identification.*
  Let S_i be the state of n_i, $S_i(t) = 0$ if n_i operates normally at the moment t. Otherwise, $S_i(t) = 1$ and there must be (a) misbehavior node(s) somewhere. Hence, $S_i(t)$ is deduced as follows:

$$S_i(t) = \begin{cases} 0 \; if \; W_i(t) \in [(\mu_i - \varepsilon_i), (\mu_i + \varepsilon_i)], \\ 1 \; otherwise \end{cases} \tag{5.1}$$

  It is worth noting that the fact that $S_i(t) = 1$ does not lead to the conclusion that n_i is malicious. The problem can also come from another sensor node within the path from n_i to the sink node s. Our mission then is to identify a misbehavior node that we know definitely within the path from n_k to s. This is the goal of the step 2.

- *Step 2: Misbehavior node identification.*
  Suppose that we are (passively) monitoring in real-time a 6LoWPAN-WSN and suddenly we witness the occurrence of the event "$S_k = 1$". Thus, there must be a malicious node within the path $s \longrightarrow n_1 \longrightarrow n_2 \longrightarrow ... \longrightarrow n_k$.
  Thanks to the learning phase, we have already known $S_1, S_2, ..., S_{k-1}, S_k$. Then we have the following logic deduction:
  $\exists j \in \mathbf{N},\ j \geq 1 \mid (S_i = 0 \; for \; every \; i \in \{0, 1, ..., j-1\}) \bigwedge (S_j = 1)$
  The sink node s is considered as S_0 ($S_0 \equiv s$).
  Evidently, n_i is legitimate for $i \in \{0, 1, ..., j-1\}$ and n_j is logically the first misbehavior node detected. We continue to test the other nodes including n_{j+1} until n_k:
  $s \longrightarrow ... \longrightarrow n_j \longrightarrow n_{j+1} \longrightarrow ... \longrightarrow n_k$.
  $S_0 = 0 \longrightarrow ... \longrightarrow S_j = 1 \longrightarrow S_{j+1} = 1 \longrightarrow ... \longrightarrow S_k = 1$
  We define α as the difference (i.e., delay) between the link weight in reality and the expected link weight: $\alpha_i = W_i - \mu_i$, and β as the additional link cost to the neighbor caused by the node n (Fig. 5.7) ($\beta = 0$ if and only if n is normal). α is directly calculated thanks to known values of W and μ, whilst, β would be deducted indirectly Obviously, 5.1 equals to two following expression:

$$S_i(t) = \begin{cases} 0 \; if \;\; \mid \alpha_i(t) \mid > \varepsilon_i, \\ 1 \; otherwise \end{cases} \tag{5.2}$$

$$S_i(t) = \begin{cases} 0 \; if \; \beta_i(t) = 0 \\ 1 \; otherwise \end{cases} \tag{5.3}$$

Figure 5.7: Additional link cost to the neighbor

Thus,

$$(\alpha_{j+1} - \alpha_j) = (W_{j+1} - W_j) - (\mu_{j+1} - \mu_j) \tag{5.4}$$

Obviously, $\mid \alpha_j \mid > \varepsilon_j \; and \mid \alpha_{j+1} \mid > \varepsilon_{j+1}$. μ_j and μ_{j+1} were derived from the learning phase. W_j and W_{j+1} are calculated in real-time monitoring.

Because all nodes from the sink node to n_{j-1} function normally, α_j exists principally as a result of the communication delay between n_{j-1} and n_j. In other words,

$$\alpha_j \approx \beta_j \tag{5.5}$$

In (5.4), $(\mu_{j+1} - \mu_j)$ is the weight costed by the link $n_j \longrightarrow n_{j+1}$ in normal condition (in theory); $(W_{j+1} - W_j)$ is the one in under-monitored condition (in practice). The right side is, thus, the additional cost caused over the link $n_j \longrightarrow n_{j+1}$, i.e., $\beta_j + \beta_{j+1}$ Therefore,

$$(\alpha_{j+1} - \alpha_j) = \beta_j + \beta_{j+1} \tag{5.6}$$

Because of 5.5, 5.6, β_{j+1} can be inferred as:

$$(\beta_{j+1} \approx \alpha_{j+1} - 2 * \alpha_j) \tag{5.7}$$

We have achieved identifying the status of n_{j+1}.
Continuously, now we are testing n_{j+2}:
Similarly to (5.4), we have:

$$(\alpha_{j+2} - \alpha_{j+1}) = (W_{j+2} - W_{j+1}) - (\mu_{j+2} - \mu_{j+1}) \tag{5.8}$$

The right side of (5.8) is the additional cost caused over the link $n_{j+1} \longrightarrow n_{j+2}$, i.e., $\beta_{j+1} + \beta_{j+2}$
To sum up,

$$\beta_{j+2} = \alpha_{j+2} - \alpha_{j+1} - \beta_{j+1} \tag{5.9}$$

All elements in the right side are disclosed, thus, (5.9) gives us the condition to determine whether n_{j+2} is normal or not.

Similarly, we repeat the aforementioned steps to verify the status of the rest:

$$\{n_{j+3}, n_{j+4}, ..., n_k\}$$

5.3.2 Anomalies detection based on Information Theory

This subsection aims to take Information Theory into consideration in order to provide theoretical base for the learning phase of our framework. These measures can be defined and calculated from extracted attributes. They can be useful to describe the characteristics of an audit dataset, define a suitable detection model, as well as evaluate the performance of the model.

1. Entropy
   Entropy [42] is an important concept measuring the uncertainty (or impurity) of a collection of data items. Let X is the collection including N classes of data items x_i (i=1, 2, ..., N). The entropy of X is defined as:

$$H(X) = H(x_1, x_2, ..., x_N) = -\sum_{i=1}^{N} P(x_i) * logP(x_i) \tag{5.10}$$

   where $P(x_i)$ is the probability of x_i in X for i=1, 2, ..., N. The "purer" dataset has a smaller entropy, i.e., the class distribution is skewer. The smallest possible value

of the entropy is 0 in case the dataset has only one class of items, i.e., there is no uncertainty because we know for sure every items belong to this unique class. When the data is more "impure" the uncertainty increases, the entropy value is bigger.

In the context of this thesis (anomaly detection), we use entropy to measure the regularity of the data input. For example, a trace file can be translated to a set of events $E = \{e_1, e_2, ..., e_N\}$. The high regularity refers to the fact that many events are repeated and they will likely appear again in the future. Additionally, if a system works in the mode of *duty cycle* and frequently (e.g., WSNs in which sensors periodically send sensed data), its regularity is seemingly stable. This assumption is assessed in the experimental section.

2. Conditional Entropy
   The conditional entropy of the dataset X given the dataset Y is defined as:

$$H(X|Y) = -\sum_{j=\overline{1,M}}^{i=\overline{1,N}} P(x_i, y_j) * logP(x_i|y_j) \tag{5.11}$$

   where x_i, y_j are classes of data items of X and Y respectively ($i = \overline{1,N}, j = \overline{1,M}$), $P(x_i, y_j)$ is the joint probability of x_i and y_j and $P(x_i|y_j)$ is the conditional probability of x_i given y_j.

   This concept can be used to measure temporal or sequential characteristics of complex audit datasets corresponding to temporal user, program and network activities. For intrusion detection point of view, this is usable for detecting complicated attacks demanding the correlation of different events.

3. Other measures
   In addition to two aforementioned measures, there exists several concepts that can be taken into consideration, namely *relative conditional entropy, information gain and classification, and information cost*. They can be useful for building an anomaly detection model as well as evaluating it.

5.4 Experimental results

5.4.1 Proof-of-concept architecture

In order to validate our framework, we deploy a real 6LoWPAN-based WSN using the open platform provided by FIT-IoT lab [43]. Fig. 5.8 depicts the hierarchical architecture of our network acting as the proof-of-concept. The BRs (Border Routers) play the role as sink nodes equipped with a sniffer which allows capturing live traffic. In the context of this work, we used A8 and M3 nodes to deploy BR nodes. Whilst, M3 and WSN430 nodes [1] were utilized to implement sensor nodes. Our nodes were running Contiki as the operating system. Sniffers were integrated with BRs to capture and pass the network traffic to MMT. Extracted attributes were stored in a local database and further computations would be performed to detect the problems.

The network deployment was step by step realized as follows:

[1] Hardware information about nodes: https://www.iot-lab.info/hardware/

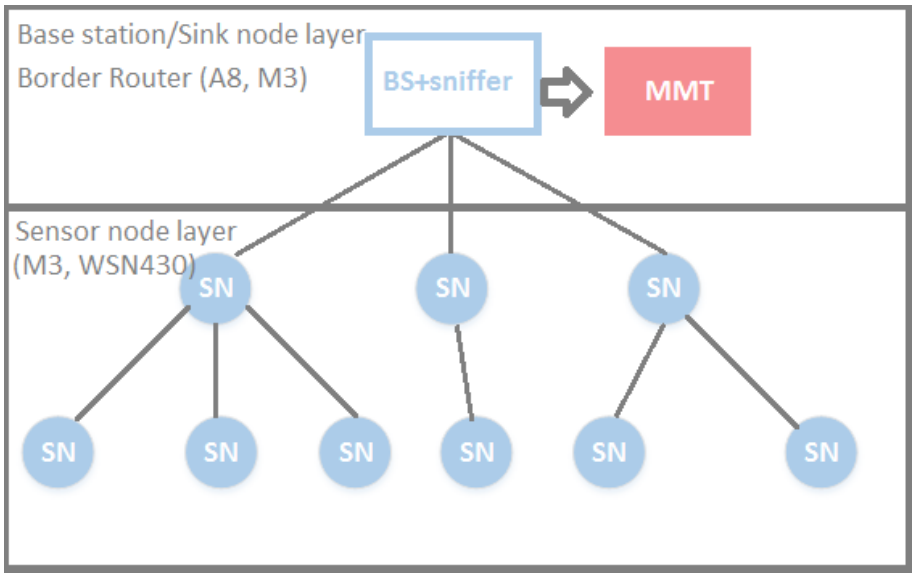


Figure 5.8: Hierarchical architecture of the 6LoWPAN-based WSN in our experiment

- Selecting available sensor nodes in FIT-IoT test-bed for the experiments: M3/A8 node as BR node and M3/WSN430 nodes as sensor nodes.
- Starting *tunslip6* [2] to bridge the BRs to the front-end network.
- Loading suitable firmware for each node: BR firmware with sniffer integrated and sensor node firmware with HTTP server code included.
- Booting nodes and starting the experiments. Sensor nodes would periodically (every 10 seconds) send sensed data to their corresponding BR. MMT would take the traffic captured by sniffers as the input.

5.4.2 Experimental results

5.4.2.1 Case 1: Statistical Learning

1. Performance evaluation with offline traffic (PCAP files)
   Firstly, we assessed the processing speed of MMT in the cases of different sizes of the network (i.e., the number of nodes). In each case, the sniffer recorded the traffic passing by the BR for 5 minutes and saved as a PCAP (packet capture) file. MMT would analyze the PCAP files and extract all attributes that we had defined by the plugins. Fig. 5.9 summarizes the results. Evidently the more nodes we inserted to the network, the more traffic they generated and the more time MMT required to process. However, MMT has indeed shown a promising processing rate which is always around 420 Mbps. The processing rate is calculated as follows:

$$processing\_rate(Mbps) = \frac{traffic\_volume(kB) * 8}{1024 * average\_execution\_time} \tag{5.12}$$

[2] https://www.iot-lab.info/tutorials/build-tunslip6/

This processing rate introduces MMT as a potential candidate for monitoring even big network consisting of hundreds or thousands connected objects.

Nb of nodes	Traffic (kB)	Processing time (mS)	Processing rate (Mbps)
5	47	0.87	422
10	118	2.21	417
15	235	4.38	419
20	393	7.33	418
25	648	12.14	417
30	1038	19.35	419
35	1334	24.87	419
40	2096	39.08	419

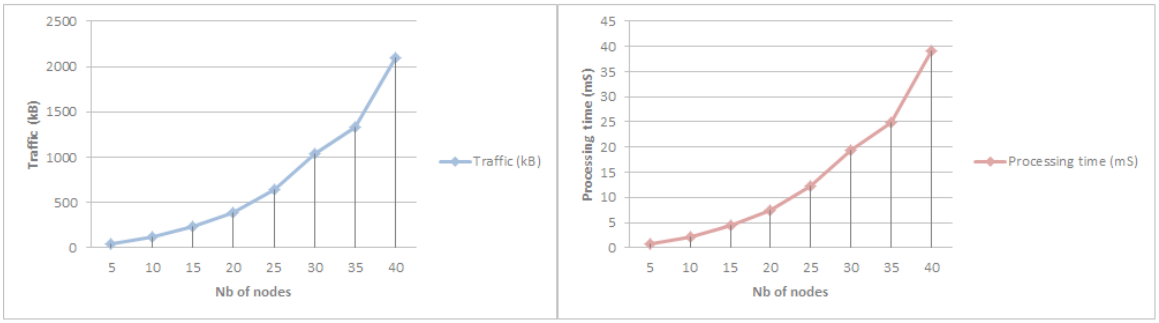


Figure 5.9: Volume of traffic and processing time depending on the size of network

2. Real-time monitoring and response delay

   Secondly, we validated and compared our solution's performance with Foren6, which is one of the first and the most well-known open-source debugging tools for IoT, while detecting abnormal activities triggered by some misbehavior nodes in the network. In fact, similarly to MMT, Foren6 permits passively capturing 6LoWPAN traffic and renders the network state in a graphical user interface. Although it is able to detect abnormal behaviors in routing, it is mainly used to reconstruct a visual and textual representation of network (i.e., network troubleshooting). There is, for the moment, no specific Foren6-based application for detecting security violations. In our experiments, we created abnormal activities by modifying the firmware loaded to a number of nodes and forcing them to delay message forwarding process or even sometime avoid forwarding messages (selective forwarding attack). These misbehaving nodes would affect all downstream nodes that use them as the forwarder to reach the BR.

   In order to detect these behaviors, we applied our own detection algorithm, which is explained in detail at the section 5.3.1 as well as in [44]. The general idea is to calculate *travel time* of each packet coming from the BR to each node or vice versa. This task was realized by extracting suitable attributes in packets. The *travel time* would act as the link weight mentioned in the algorithm. We saved the results when the network was properly functioning and when aforementioned abnormal activities were performed.

   In case of proper conditions, we observed every node and extracted two attributes *time stamp* and *MAC address* of every packet coming in and out. After that, based on those values stored, we calculated the *travel time* related to each node (i.e., the necessary time for a packet delivered from this node to the BR or vice versa). We considered this as a random variable and statistically analyzed them in using RStudio

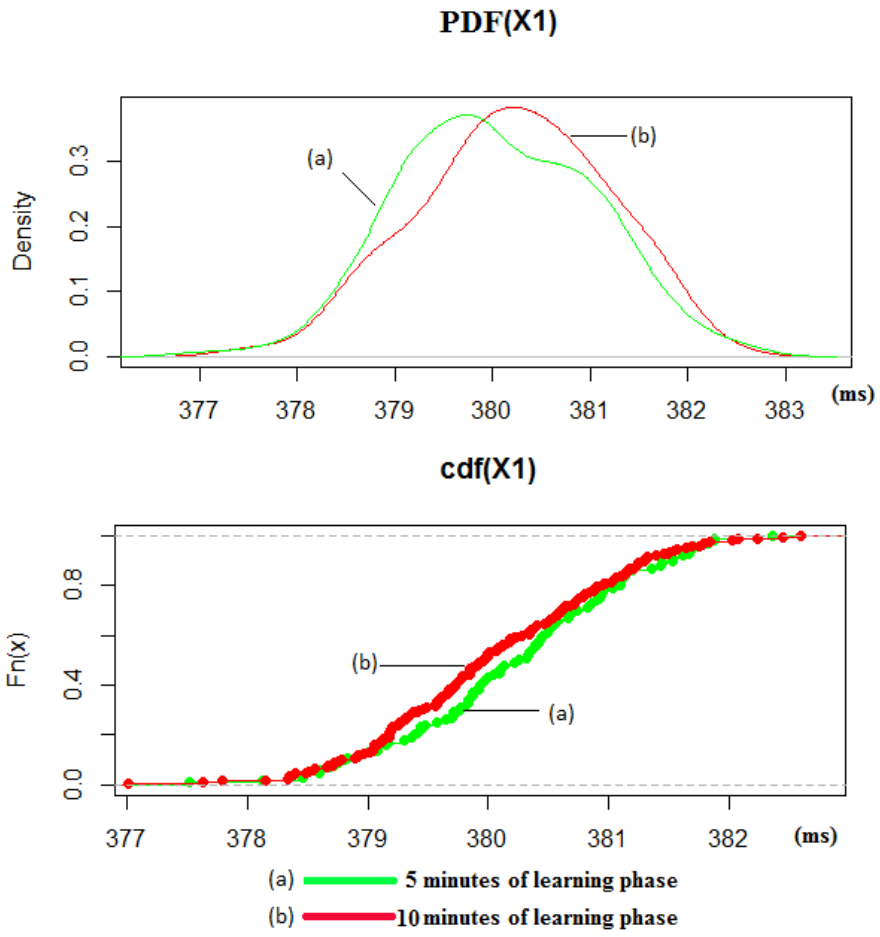


Figure 5.10: Probability Density Functions and Cumulative Distribution Functions of the *travel time*

[3]. We received the same result regardless of using MMT or Foren6. Fig. 5.10 presents the histogram of two PDFs (Probability Density Functions) and two CDFs (Cumulative Distribution Functions) of the *travel time* of packets concerning a sample node. Line (a) corresponds the case where we took into account five minutes of monitoring, whilst, line (b) corresponds the case of ten minutes. We witnessed that it is likely normally distributed (Gaussian distribution [41]). For both two cases, the *mean* (i.e., expectation) of the distribution is approximated at 380 milliseconds. We call this observation as ***the learning phase***.

In case of abnormal activities added to the network, we repeated that procedure (extraction and calculation) and compared received values with the ones derived from the learning phase. As seen in the case above, if we suppose $X_i(t)$ is the random variable representing the *travel time* related to the node n_i and the sink node s at the moment t, $X_i(t)$ can be dealt as a Gaussian distribution:
$X_i(t) \sim N(\mu_i, {\sigma_i}^2)$ where μ_i is the mean and σ_i is the variance.

[3]https://www.rstudio.com/

According to *3-sigma rule* [41], approximately 95% and 99.7% of values drawn from a normal distribution lie within two and three standard deviations σ correspondingly away from the mean μ. This percentage increases according to the gap away from the mean. Therefore, we could define then $[(\mu_i - \varepsilon_i), (\mu_i + \varepsilon_i)]$ as the promising interval that X_i should lie within at whatever moment. ε_i should be customizable and generally between 3σ and 7σ. Each occurrence of the event when we witness a value fall outside this interval should trigger the alert about an abnormal activity. In such case, our detection algorithm would be applied to determine the misbehaving node.

Nb of nodes	Nb of malicious nodes	MMT (ms)	Foren6 (ms)
5	1	13.87	13.43
10	2	31.9	32.14
15	3	48.54	49.11
20	4	66.58	64.87
25	5	84.61	85.22
30	6	108.22	110.56
35	7	128.92	131.94
40	8	152.57	155.15

Table 5.1: Comparison of detection delay between MMT and Foren6 (in millisecond)

While performing experiments with MMT and Foren6, we successfully detected the evil nodes which were loaded malicious firmware. Tab. 5.1 depicts the detection delay of MMT and Foren6 depending on the network size. In those experiments, we fixed the number of malicious nodes equivalent to 20% of total nodes in the network. We witnessed that the demanding processing time to identify misbehavior nodes are strikingly increased with the number of nodes (both normal and abnormal ones). This delay consists of the time for extracting the attributes, calculating variables and performing the detection algorithm to determine malicious entities. It was growing with the number of nodes and paths under test because the number of computations used for the algorithm increases correspondingly. In any case, we observed basically the same performance for both MMT and Foren6.

3. Real-time monitoring and the detection algorithm performance:
Thirdly, in order to evaluate the influence of the threshold ε_i to the accuracy of our algorithm, we repeated the experiments with different threshold in counting the number of false positive and false negative. In the framework of this research, we did not observe any false negative. Fig. 5.11 illustrates the false positive and accuracy rate related to an observed node N_i in function with the threshold ε_i.
It is computed as follows:

$$false\_positive\_rate(\%) = \frac{nbr\_false\_positive * 100}{nbr\_detection} \tag{5.13}$$

Since there was no false negative:

$$accuracy\_rate = 100 - false\_positive\_rate(\%) \tag{5.14}$$

Indeed, we observed a very good accuracy when the threshold ε_i is bigger than $3 * \sigma_i$. These results validated once again the ***3-sigma rule***.

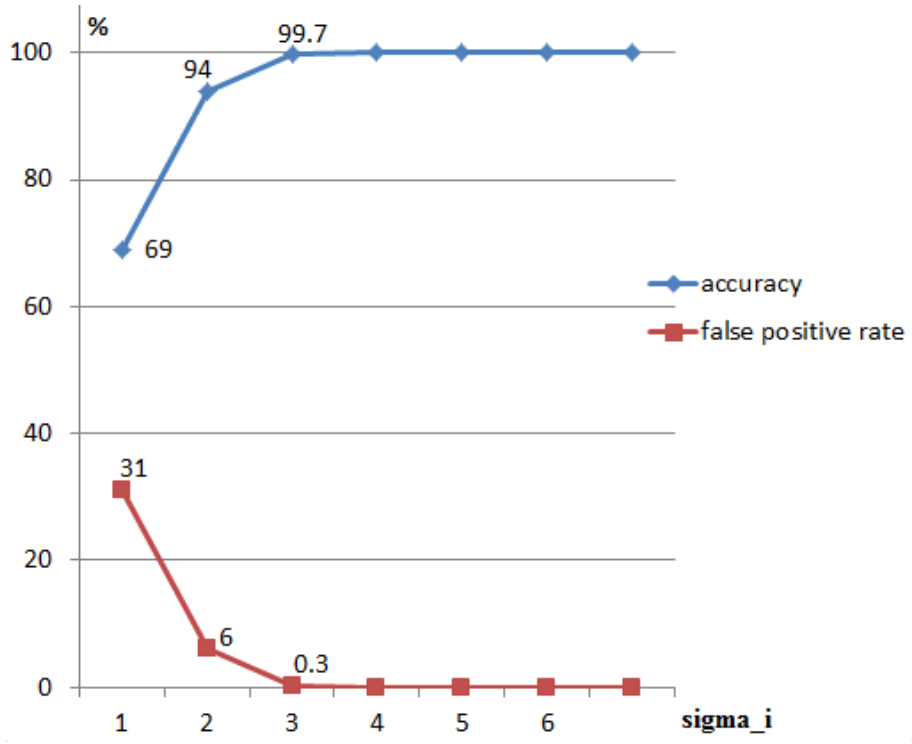


Figure 5.11: Proposition's false positive and accuracy rate in function with the threshold ε_i

4. Algorithm extension
   Although the accuracy witnessed in aforementioned experiments was high, we recognized that certain abnormal nodes detected were not really valuable. Sensor nodes are weak and sometimes fall to failure but only momentarily and then come back to normal state. The temporary fault state should be tolerated. To deal with this issue, we attempted to replace the momentary state in monitoring phase by a more long-lasting state. X_i would not be calculated at a single moment t but as an average value in the period from $t - \tau$ to t, where τ is the observation duration and pre-defined based on the characteristics of the network. We performed our experiments with $\tau = 1$ minute.
   Evidently, the detection algorithm with and without the presence of τ must have

	Without τ	With τ
AVG(processing time) (ms)	383	528
Throughput (Kbps)	104	76

Table 5.2: Solution's average processing time and throughput

different performance in terms of response delay (i.e., processing time) due to the additional time for querying other older values of X_i and for extra calculations (Tab. 5.2, *AVG* means *average (of)*). Experiments demonstrated that our solution allowed processing network traffic with bit-rate up to 104 Kbps (without τ) and 76 Kbps (with τ) which is sufficient for 6LoWPAN-based WSNs.

In conclusion, looking to some sample events occuring in some specific moment is

not enough for a thorough security monitoring. In stead, monitoring should be a continuous process taking into account the history and the sequence of events.

5. Related work
   There are actually several malicious and abnormal node detection schemes proposed in the literature for WSNs in general and recently for 6LoWPAN specifically. As a result of energy issues, most of them are based on a distributed model, using either neighbor coordination or clustering. For example, Curiac et al. [45] proposed an auto-regression technique to detect malicious node. They saved past and present values provided by each sensor as the input of an auto-regressive predictor to estimate an expected value. If the received value is too different from the expected one, the related sensor node must be questionable. The similar point of this work to our one is that we both predict an expected range based on the received one, i.e., anticipate the future from the past. Falling outside from this range signifies an abnormal behavior. However, [45] specifically copes with suspicious nodes sending malicious data, in other words, it cares more about the content of message rather than other aspects of the network, e.g., delay, bitrate and packet loss rate which is the main concern taken in our work.

   Whereas, Atakli et al. [46] proposed another scheme do detect compromised node using *weighted-trust evaluation*. The authors utilized a clustered topology for their hierarchical WSN network and built their detection scheme based on *weighted-trust evaluation*. They divided their network into three layers including AP (Access Point), FN (Forwarding Node) and SN (Sensor Node) layer. FNs assigned a *weighted-trust* to each SN and an algorithm was proposed to update this value based on what FNs receive from their SN. Nonetheless, [46] presents simply some preliminary results derived from some simulations and the performance and the scalability of the solution are still a problematic that the author left as their future work. As an improvement, Seo Hyun Oh et al. [47] proposed another scheme using *dual-weighted trust evaluation* to reduce mis-detection rate while maintaining comparable performance. Although *weighted-trust* is very close to our idea in using *link weight* represented by *packet's travel time*, FNs in our case are also sensor devices which are not powerful enough to perform computations.

5.4.2.2 Case 2: Information Theory

The main idea of these experiments is to monitor the entropy value of the system (6LoWPAN-based WSNs) and see if it can be useful to design an anomaly detection model. Similarly to the experiments in the section 5.4.2.1, we deployed 6LoWPAN-based WSNs but this time, we cared only about routing packets. For each packet, we extracted the set of attributes consisting of *source's MAC address, destination's MAC address, timestamp, type of routing packet*. So far we defined five different routing packet types: *RPL DIS, RPL DIO, RPL DAO, Neighbor solicitation* and *Router Advertisement*. An event e_i is defined as a triplet <source's MAC address, destination's MAC address, type of routing packet>. We analyzed the traffic and recorded events received then calculated the entropy of the set of all received events as a temporal variable.

Firstly, we performed five experiments on the networks of 10 nodes. We monitored the entropy of the set of received events in approximately 40 minutes (from the booting of

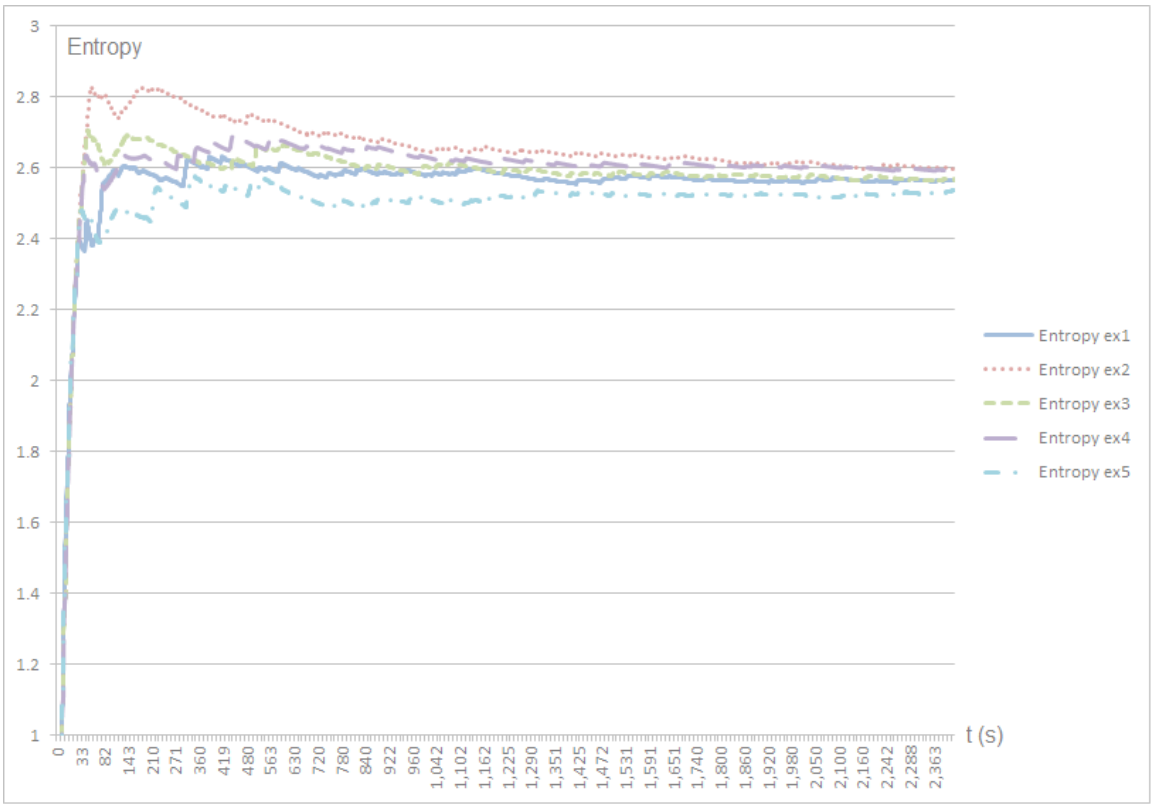


Figure 5.12: Entropy monitoring of 10 nodes under normal condition

sensor nodes). The topology of the networks in five experiments were fixed but we loaded BR firmware to five different nodes. The sniffer was always located in the BR node.

As displayed in the Fig. 5.12, after first two or three minutes increasing very fast, the entropy became quite stable and slightly oscillate around a convergence value. We witnessed this in all five experiments regardless the convergence values are a little bit different among five cases. We observed also that the more the BR node was located in the center of the network (i.e., the more symmetric the topology is), the smaller the entropy became (i.e., the purer the set of events is).

Secondly, in order to observe the possible change on larger networks, we repeated the experiments another three times but on the networks of 30 nodes. On the one hand, we received the similar results (Fig. 5.13) (the entropy quickly increased in the booting duration until a pretty stable value). On the other hand, we noticed that the entropy regarding the 30-node-networks is higher than the one regarding the 10-node-networks. This is obviously understandable, because the larger systems with a bigger number of entities likely become impurer than the the smaller systems.

Thirdly, we performed another two experiments on the networks of 30 nodes. However, we rebooted the BR several times to see how the entropy variable reacted. Indeed, it reacted like we restarted the experiment (ex4 and ex5 in Fig. 5.14) with the stable point a little bit higher than before the reboot. This is because of the fact that the reboot event and necessary routing events performed after the reboot make the set of caught events impurer.

Last but not least, we injected some routing attacks to the networks. The Fig. 5.15 (ex6)

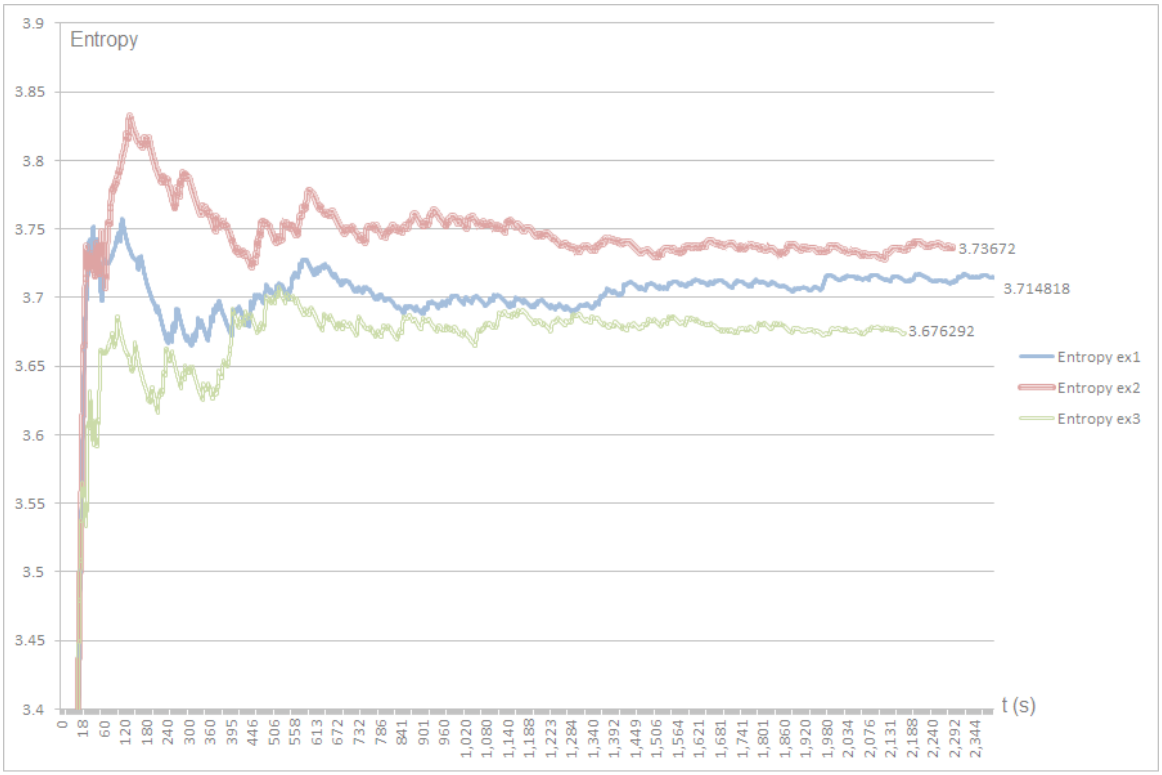


Figure 5.13: Entropy monitoring of 30 nodes under normal condition

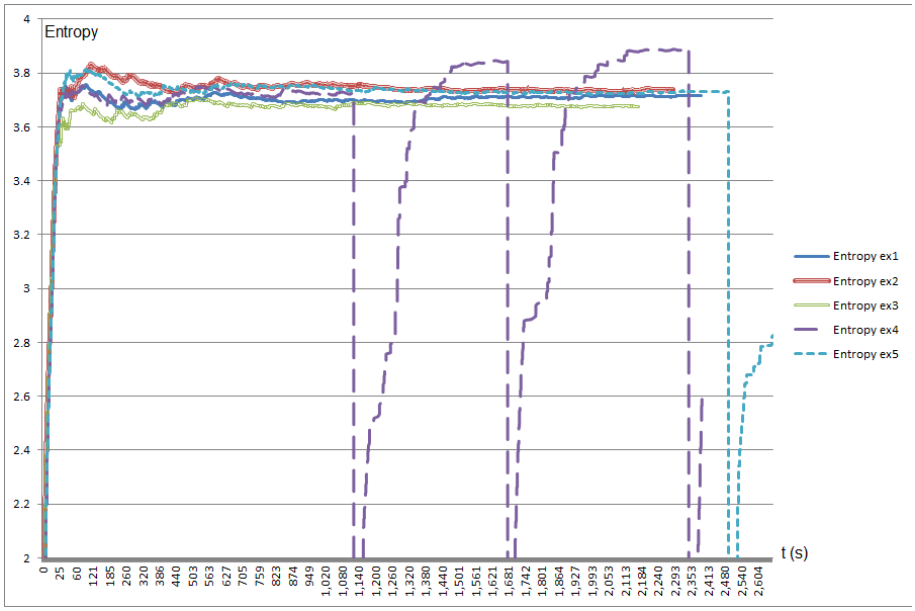


Figure 5.14: Entropy monitoring of 30 nodes under rebooting

depicts the results when we forced some nodes to perform the *flooding attack* (attack 1) and the *selective forwarding attack* (attack 2) [48]. An almost immediate augmentation of the entropy is noticed at the moment of the attack. When the attack is terminated, the entropy

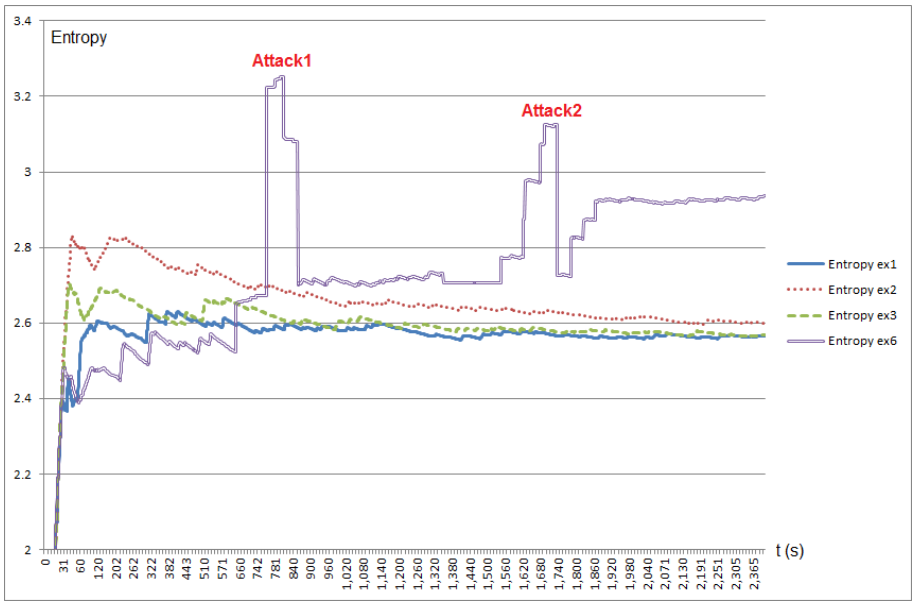


Figure 5.15: Entropy monitoring of 10 nodes under attacks

return the stable state at a point higher than before the attack. This can be explained by the fact that our model had taken into account also the attack and thus, taken events are impurer. Indeed, any routing attack affecting considerably the proper normal routing process of the network can trigger a sudden change in the entropy value.

In short, we acknowledged the usefulness of entropy as a metric to monitor the 6LoWPAN-based WSNs. A simple entropy-based MMT-Security rule was created to automatically detect the abnormal change in the entropy value, and then to detect attacks. The main idea is as follows: (i)- Monitor the network and learn the normal expected zone for the entropy value $(e-\varepsilon, e+\varepsilon)$; (ii)- Keep monitoring the network and raise an alert if the temporal entropy value fall outside the pre-learned normal zone. The learning phase (including the determination of the expectation e and the threshold ε) is repeated up to the network administrators. The bigger threshold can lead to the higher rate of false negative. The smaller one can increase the rate of false positive. The Tab. 5.3 displays the performance of our solution in different networks consisting of different number of sensor nodes, in which we defined that: $\varepsilon = 5\% * e$. Evidently the more nodes we inserted to the network, the more traffic they generated and the more processing time (PT) MMT required to process. However, MMT has indeed shown a promising processing rate which is always around 420 Mbps. The processing rate is calculated as follows:

$$PR(Mbps) = \frac{traffic\_volume(kB) * 8}{1024 * PT} \tag{5.15}$$

This processing rate (PR) introduces MMT as a potential candidate for monitoring even big network consisting of hundreds or thousands connected objects.

To conclude, in addition to other usable metrics, entropy can be a good candidate for systems where we need to define the normal states. Particularly speaking about RPL-based WSNs, from our point of view, the link-weight can be another metric providing the local view of the network [44], while the entropy can provide a global view of the whole system.

Nb of nodes	Traffic (kB)	PT (mS)	PR (Mbps)
5	47	0.86	424
10	118	2.19	420
15	235	4.38	419
20	393	7.33	418
25	648	12.16	417
30	1038	19.34	419
35	1334	24.83	419
40	2096	39.06	419

Table 5.3: Traffic volume, processing time and processing rate depending on the size of network

Chapter 6

Enabling Intrusion Tolerance by Design

Contents

6.1 Introduction

For a long time people invest much research on security mechanisms to prevent or detect intrusions and attacks. However, attacks are more and more sophisticated and thus, they are difficult to be captured. A system may fail to complete its mission whenever a successful attack occurs and it may be impossible to recover quickly. In the recent few years, research community starts to spread the issue attack tolerance that will allow a system strong enough to tolerate attacks.

Intrusion or attack tolerance of a system is generally understood as the capability to continue to function properly with minimal degradation of performance, despite intrusions

or malicious attacks [11]. In terms of networks, this concept means the ability to maintain the overall connectivity and diameter of the network as nodes are removed.

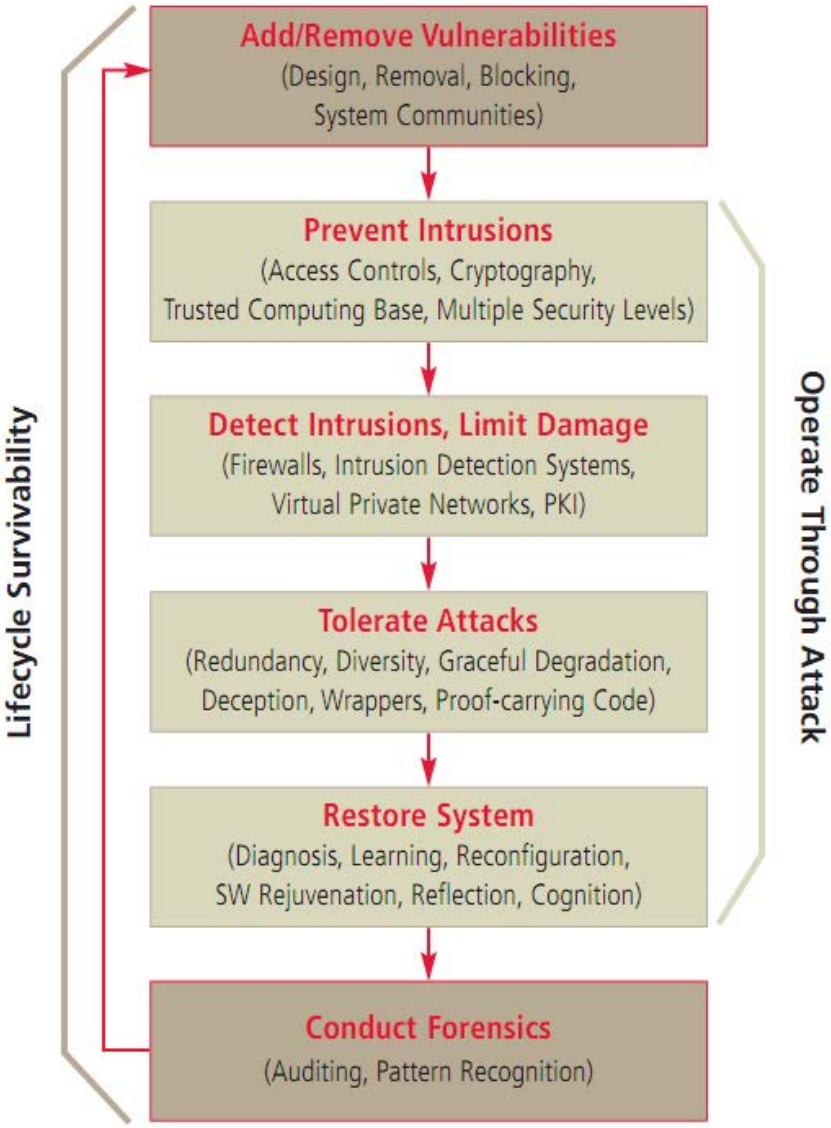


Figure 6.1: Security features for modern systems

The Fig. 6.1 [49] indicates features which should be integrated to a modern system or network. To gain intrusion/attack tolerance, systems employ redundancy, diversity, and reconfiguration to remove unwanted intrusions and recover the normal state.

This chapter provides a study on the concept *intrusion tolerance by design*. We first assess the possibility to integrate *intrusion tolerance* to the routing issue in WSNs. Then we propose an emulation-based extension of aforementioned security monitoring framework to simultaneously detect and tolerate intrusions.

6.2 Intrusion Tolerant Routing in WSNs

In the current era, WSNs are rapidly emerging as an important area in both the research community and the public. However, there are three principal difficult research challenges to design and implement a secure WSNs, namely:

- the vulnerable characteristics of wireless communication nature (e.g., eavesdropping, unauthorized access, spoofing, replay and DOS attacks).
- the severely resource-constraints of sensor devices. Typical WSNs are composed of a large number of low-power tiny sensors and actuators that own a limited energy lifetime, slow embedded processors, severely constrained memory and low-bandwidth radios, with sensing, computation, and wireless communication capabilities. For example, Waspmote [29], the modern open source sensor device distributed by Libelium, contains simply a 14 MHz micro-processor, 3.3 V - 4.2 V battery voltage, 8 KB SRAM, 128 KB flash memory and 4 KB EEPROM to save sensed data, run and operating system and application programs. These resource constraints limit the degree of encryption, decryption, and authentication that can be deployed, thus, the concept *security* and *WSNs* were likely contradictory.
- additional physical security risk because of being deployed in inaccessible terrains or unattended and even hostile environments to sense data or to observe the occurrence of certain events by self-organizing into an ad-hoc style wireless network that collects and forwards sensor data to an information sink (e.g., a base station acting as a gateway to the wired network).

In addition to the intrinsic characteristics, due to the lack of security considerations in the design of protocols, WSNs are exposed to various types of security threats. The failure of nodes may result in network partition, decreasing the cover ratio, reducing the availability of the sensor network and even producing entire network failure. The integration of an intrusion detection system (IDS) is presumably too expensive in terms of resource. Therefore, it is essential to take the concept of **intrusion tolerance** into consideration in order to sustain the sensor network functionality without interruption despite malicious attacks and sensor node failures.

The idea on intrusion- tolerant routing for WSNs first appeared around the middle of last decade, however, there is very little research on this topic. Beside the well-known INSENS, ITSRP is uniquely another proposition which is much less popular. Nevertheless, ITSRP still deserves considerations because it took into account the problem "energy consumption" that is a very critical issue of WSNs.

This section aims to present a study on INSENS and ITSRP. We attempt to briefly describe how these two protocols work and which properties make them intrusion-tolerant. We also remark their missing issues, carry out some simulations to assess their practical performance and suggest some possibilities to upgrade these two protocols. Although the doubt upon the feasibility of our suggestions, we consider this study as good first step to approach a further better intrusion- tolerant protocol as well as an intrusion-tolerant framework that could be integrated to WSNs.

Intrusion tolerance [50–52] is generally understood as the capability to continue to function properly with minimal degradation of performance, despite intrusions or malicious attacks. A great deal of work has been done to address the sensor network security problems

recently so that the WSNs can tolerate and/or prevent intrusions [53,54]. As other kinds of networks, routing is one of the most critical features in WSNs. In the following subsections, we attempt to briefly identify two current approaches used for achieving intrusion-tolerant routing and focus our analyzes on their intrusion-tolerant properties. The more detailed descriptions illustrating how INSENS and ITSRP function can be found in [2,55] and [56].

6.2.1 INSENS - Intrusion-tolerant routing protocol for wireless SEnsor NetworkS

Intrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS), proposed by Jing Deng et al [2,55,57], aims to assure communication authentication and integrity check while forwarding tables are being constructed, and to prevents dangerous attacks and malicious activities.

INSENS is possibly subdivided into two phases: ***Route Discovery*** phase and ***Data Forwarding*** phase (Tab. 6.1). The goal of the first phase is to collect topology knowledge and to construct appropriate forwarding tables for every node. Whilst, the second phase simply enables forwarding of data from each sensor node to the base station and vice versa. It is worth mentioning that every communication between nodes is one-way forwarded (i.e., unicast) via base station.

Phase 1: Route Discovery			Phase 2: Data Forwarding
Route Request	Route Feedback	Computing and Propagating Multi-path Routing Tables	

Table 6.1: Two phases and three rounds in the first phase of INSENS

Intrusion-tolerant properties of INSENS are mostly gained thanks to the ***Route Discovery*** phase, that is composed of three rounds: *Route Request, Route Feedback*, and *Computing and Propagating Multi-path Routing Tables*. In the beginning (or when the topology may have changed substantially because of nodes' mobility), the *base station* floods **limitedly** a *request message* to all the reachable sensor nodes in the network. After receiving a *request message* for the first time, a sensor node x broadcasts in return another *request message* that includes a *path* from the *base station* to x and also the *identity of* x. Whenever receiving duplicate *request messages*, it records the *identity* of the sender as a neighbor, but stop re-broadcasting the duplicate request. The *base station* authenticates the *feedback messages* received from sensor nodes (authentication manner will be further discussed in the follow parts of this section). After that, it constructs a topological picture of the network from the authenticated neighborhood information, computes the forwarding tables for each sensor node, and sends the tables respectively to nodes using a *routing update message*. To address the influences of compromised nodes, INSENS builds redundant multi-path routing tables containing disjoint paths. Therefore, even if a single node or path is taken down by an intruder, secondary path will substitute.

Tab. 6.2 details the format of a *request message*: The *type* field indicates whether the message is a *request, feedback, routing*, or *data message*. The *path* field contains the path

from the base station to the current node (the node that sends this request message). The *size* field contains the length of this path. In the scope of this work, we concentrate our interest in the OWS (One-Way Sequence) field and the *MAC (MACR and MACF)* (Message Authentication Code Request/Forward) field that support the intrusion-tolerant properties of INSENS.

request message

type	OWS	size	path	MACR

feedback message

type	OWS	parent_inf	path_inf	nbr_inf	MACF

Table 6.2: Request message and feedback message format

One-way sequences (OWS) concept is actually proposed by $\mu TESLA$ protocol [58]. It consists of a function F generating a sequence of numbers K_0, K_1..., K_n, such that K_i = F(K_{i+1}), where $0 < i < n$ and F satisfies the condition that it is computationally unfeasible to compute K_{i+1} in a limited time by only knowing F and K_i. Initially, every node is pre-configured to know K_0 and F. In the i^{th} *Route Discovery phase*, the base station includes K_i in the *request message* that it broadcasts. After receiving a *request message*, a node verifies if the sequence number did indeed originate from the base station by checking whether K_0 = $F^i(K_i)$. Due to the characteristics of F, a malicious node would be computationally impossible to guess the next OWS to spoof the base station by generating ***new*** OWS. On the other hand, a sensor node stores the most up-to-date or freshest OWS_{fresh} that it has just seen from the base station. This fact resists an intruder to disrupt the network by using ***old*** *OWS* to flood ***old*** *request messages*.

In addition to OWS, keyed MAC (Message Authentication Code) algorithm is another factor that provides intrusion-tolerant properties for INSENS. Each sensor node is initially configured with a separate *secret key* that is shared only with the base station. When a node x receives a *request message* for the first time, before forwarding it, x appends its identity to the path list, and then generates a MAC of the complete new path with its key: $MACR_x = MAC(size|path|OWS|type, key_x)$ where " |" denotes concatenation. The value of MACR is also appended to the *request message* that is then forwarded downstream. This MACR field will eventually be exploited by the base station to verify the *integrity* of the path contained in the packet. The aforementioned concerns about fake or modified *request message* can not harm base station thanks to this integrity check. Additionally, even if a node is compromised, only its secret key will be revealed, so an intruder cannot compromise the entire network.

In the second round (Route Discovery-Route Feedback), as illustrated in Tab. 6.2 concerning the format of *feedback messages*, keyed MAC is applied one more time to protect the integrity of *feedback messages* (i.e., MACF). List of neighbors *nbr_info* and the path *path_info* to a node x are protected by the following *keyed* $MACF_x$:

$MACF_x = MAC(path\_info \mid nbr\_info \mid OWS \mid type, key_x)$

In feedback messages, *parent_info* field determines which of a child's upstream neighbors is the parent who will take part in forwarding the *feedback message* to the base station. Instead of simply using the identity of the parent node, INSENS requires a child node to put its parent's $MACR_p$, that is derived from the parent's original *request message*, into the *parent_info* field. This $MACR_p$ is tightly linked with the current state of the OWS request-

feedback cycle, as well as to the path to the child node. In other words, the $MACR_p$ play a role as not only an addressing function but also a security function. A casual attacker, that only knows *node_id*, would be unable to forward a spurious feedback message.

6.2.2 ITSRP - Intrusion Tolerant Secure Routing Protocol

Intrusion Tolerant Secure Routing Protocol (ITSRP) [56] is a secure routing protocol which focuses on the design of some fields to emphasize the security accounting to the key exchange, but, as INSENS, not result in the complexity of the protocol with a reasonable price regarding energy factor.

ITSRP assumes that each node stores a *Local Route Table* (LRT) containing entries whose format is demonstrated in Table 6.3. The first field contains the unique *Tag* for a route. The second and the third field record ancestor node and successor node respectively. The fourth field indicates the total estimated energy for sending out the message from the source node, via intermediate nodes. This field attempts to construct secure and reliable routing. The fifth field records a timer to limit the validation of a route. The entry will be removed if this timer hits 0.

Tag	Ancestor	Successor	Energy	Lifetime

Table 6.3: Format of an entry in LRT (Local Route Table)

ITSRP consists of three phases: *the path discovery phase, the path reverse phase and the data transfer phase* [56]. These three phases will be principally described in following parts to illustrate the procedures whenever a source node N_0 wants to send a confidential message M to the sink node N_n but it lacks neither an available *route path* nor a *shared session key* with N_n. A more detailed description could be found in the original paper [56]. It is worth mentioning that each node is initially issued a Distributed Key (DK) that is shared only between itself and the sink node (e.g., the base station)

1. Path discovery phase

In this phase, N_0 must establish a *route path* and a *session key SK* uniquely shared with N_n through intermediate nodes. The source node N_0 first generates the unique *Tag* for this route and realizes these following steps:

(1) Select randomly a secret session key SK_0, compute the necessary energy E_0 for sending out M.

(2) Set $M_0 = [Tag|N_0|N_n|SK_0]$ (" |" depicts the concatenation) and encrypt M_0 in using the DK_0: $C_0 = E_{dk}(M_0)$.

(3) Encapsulate the packet $[Tag|N_n|C_0|E_0]$ and broadcast it to all nodes within its wireless transmission range.

(4) Store the entry $(Tag, 0, ?, E_0, T_0)$ to its LRT_0 where T_0 is the timer for the route and starts when the entry is added, " ?" depicts a field that will be fulfilled later

Each node close to N_0 receives packet $[Tag|N_n|C_0|E_0]$, it checks and drops if this packet has been already received before. Otherwise, it broadcasts this packet within its range and store $(Tag|N_0|?|E_0|T_0)$ in its LRT_1

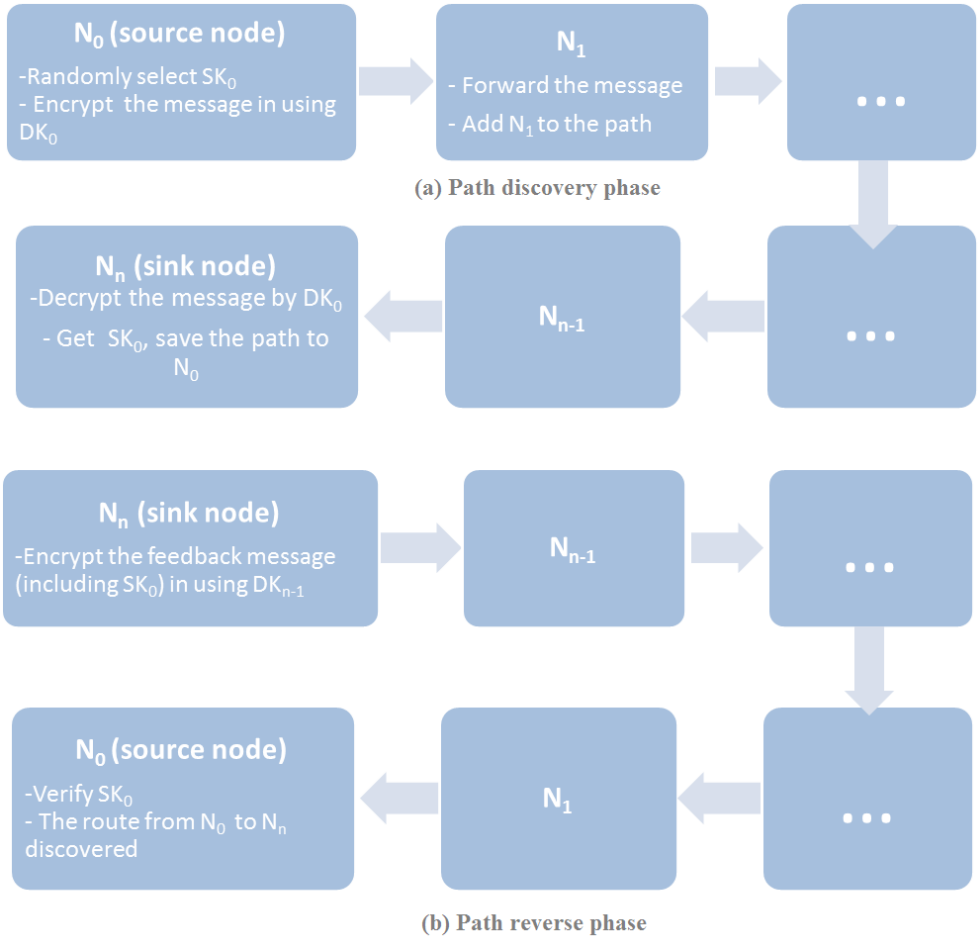


Figure 6.2: Summary of the first two phases of ITSRP

Without loss of generality, assuming that packet passes the intermediate nodes $N_1, N_2, ..., N_{n-1}$ and reaches the sink N_n. Node N_i stores $(Tag|N_{i-1}|?|E_0|T_i)$ in its LRT_i.

2. Path reverse phase

In this phase, the sink node N_n must send back to the source node N_0 the reverse path. N_n works as follows.

(1) Knowing DK_0, the sink node N_n is able to decrypt C_0 to get M_0, and then get SK_0

(2) Make $M_n = [Tag|N_0|N_n|SK_0]$ and use DK_0 to encrypt M_n as C_n

(3) Look up its ancestor node N_{n-1} according to Tag in its LRT_n, then use DK_{n-1} to encrypt Tag as C_{n-1} and sends $[C_n|C_{n-1}]$ to N_{n-1}.

When the node N_{n-1} receives $[C_n|C_{n-1}]$ from N_n, it works as follows.

(1) Use DK_{n-1} to recover Tag.

(2) Look up in its LRT_{n-1} according this Tag and update the entry saved from the first phase: $(Tag|N_{n-2}|N_n|E_0|T_{n-1})$.

Similarly to the end, N_0 gains the entry $(Tag, 0, N_1, E_0, T_0)$ in its LRT_0. Not only has the route from N_0 to N_n been discovered but also a shared session key SK between of them has been established.

3. Data transfer phase

In this phase, the source node N_0 need to confidentially send message M to the sink node N_n after knowing the route and having successfully established the shared session key SK. The data can be simply transferred by using the session key SK to encrypt M as C. After that, C (and Tag) will be forwarded along the route path by looking up the LRTs of intermediate nodes.

6.2.3 Missing issues of INSENS and ITSRP

As any other security protocol, INSENS and ITSRP have themselves missing issues that have not been thoroughly solved yet. For INSENS, a malicious node m is still possible to flood a modified *request message* in using the ***current*** OWS in a valid *request message* which it has just received from the base station. Such attack is discussed as *rushing attack* in [59]. However, nodes in the tree (Fig. 6.3), that are closer to the base station than the malicious node m, will receive the valid *request message* first. These nodes will drop m's spurious request messages coming later because, as mentioned above, nodes do not rebroadcast duplicate request messages (contain the same OWS). Even if neighboring nodes of m accept to forward the fake *request message* created by m, they forward only once. Nevertheless, the problem is that attacker could pack a *fake path* into its spurious *request message* or drop the *request message* instead of forwarding it. Some nodes can be harmed (not getting a *request message* or not being able to forward their feedback message to the base station in the second round) but, as shown in Fig. 6.3, the damage is locally confined to the nodes nearest to and downstream from the intruder. This conclusion seems to be logical but remains intuitive and deserves further evaluation to know whether such damage can still seriously disrupt the network.

With regard to ITSRP, in [56], the authors affirm that their proposition is immune to *Wormhole attacks* but this affirmation seems not surely correct. In fact, Wormhole is a kind of attacks in which two malicious nodes profit a secret high speed channel or tunnel to pretend that they are very close to each other to join in the shortest paths of some routes. After that, they can disrupt this tunnel or do whatever they want because they have gained a very powerful role in routing activities. In our opinion, a wormhole attack can still harm ITSRP, even though the probability of this event can be very low. Indeed, in the *path discovery phase*, each node, after receiving a path discovery packet, will broadcast it within its wireless communication range. The shortest path will be found based on the fastest arrived-discovery packet and duplication will be ignored. If somehow two legitimate intruders can exchange this packet faster than the normal rate of other nodes, they are still able to set up a tunnel and thus, generate an incorrect route path. The possibility that they can do it depends on which mean of wireless communication used in the network, e.g., Wi-Fi, GPRS, Bluetooth, ZigBee, etc. This problematic deserves more tests and evaluations.

6.2.4 A Comparative Evaluation

To evaluate the performance of INSENS and ITSRP, we performed the simulations by OMNeT++ with Castalia plug-in integrated. We simulated a rectangular region of $150 * 150$ m^2 in which the wireless sensor nodes were deployed in random topology, with transmission range of 10 meters. The number of sensor nodes is from 20 to 150 in increment of 20 nodes. We assumed that every node was static and equally owned the initial energy of 100000 mW and one communication corresponding to the action of sending or receiving a packet

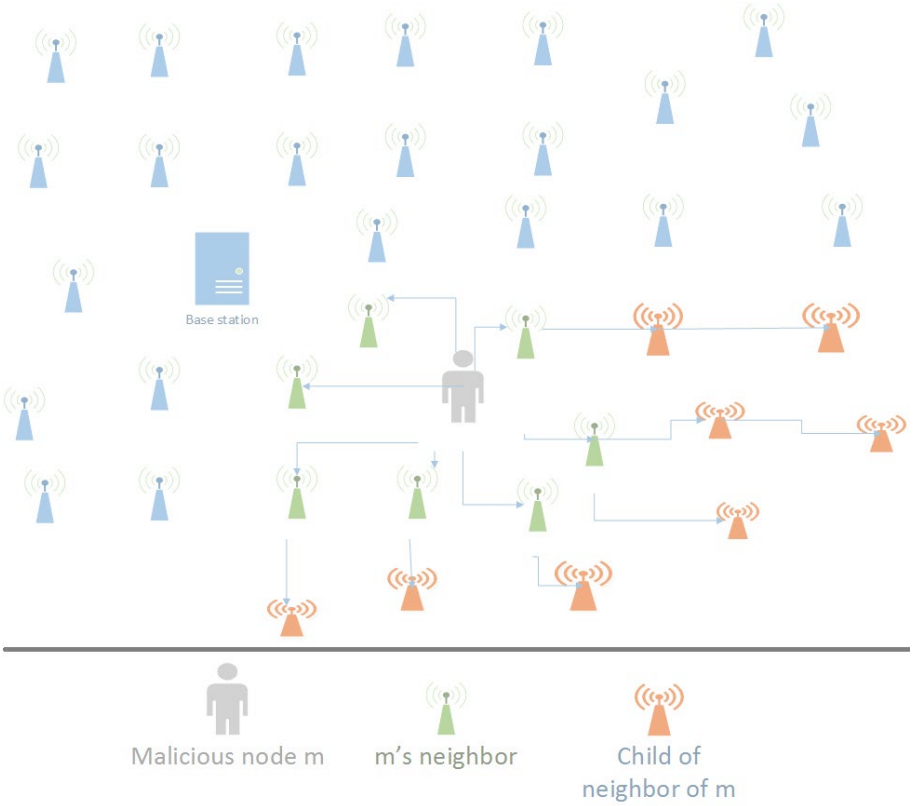


Figure 6.3: Confined portion of the impact caused by a malicious node m [2]

consumed 100 mW. The base station was assumed to have a wired energy supplier. We performed the simulations under two following kinds of attacks:

- *Passive attack*: i% of nodes misbehaved as dead nodes that refused to forward packets.
- *Active attack*: i% of nodes kept repeatedly sending *fake feedback messages* to the base station to block the wireless medium (DOS attack).

In each simulation, we measured the throughput, the overhead, and the lifetime of the network when DD (Direct Diffusion - a naive protocol without intrusion-tolerant features [60]), INSENS and ITSRP utilized. Fig. 6.4, 6.5, 6.6 present our results with $\mathbf{i = 25}$. The suffix "p" and "a" refer to *passive attack* and *active attack (DOS)* correspondingly.

6.2.4.1 Network throughput

In these cases, the throughput of network could be understood as the packet delivery ratio, i.e., the percentage of successfully received packets by the base station over all sent packets. As demonstrated in Fig. 6.4, throughput regarding respectively three metrics decreased more or less. However, the network experienced a considerable decrease in case of DD (Directed Diffusion) protocol, but only a slight change if INSENS or ITSRP was applied. This result proves one more time the intrusion-tolerance properties of ITSRP and INSENS. In terms of network throughput, INSENS is better than ITSRP if the number of sensor nodes enormously increases while ITSRP is the better choice for small-size WSNs. Another

conclusion is that *DOS attack* likely affected the throughput of the network more than the *passive attack* did.

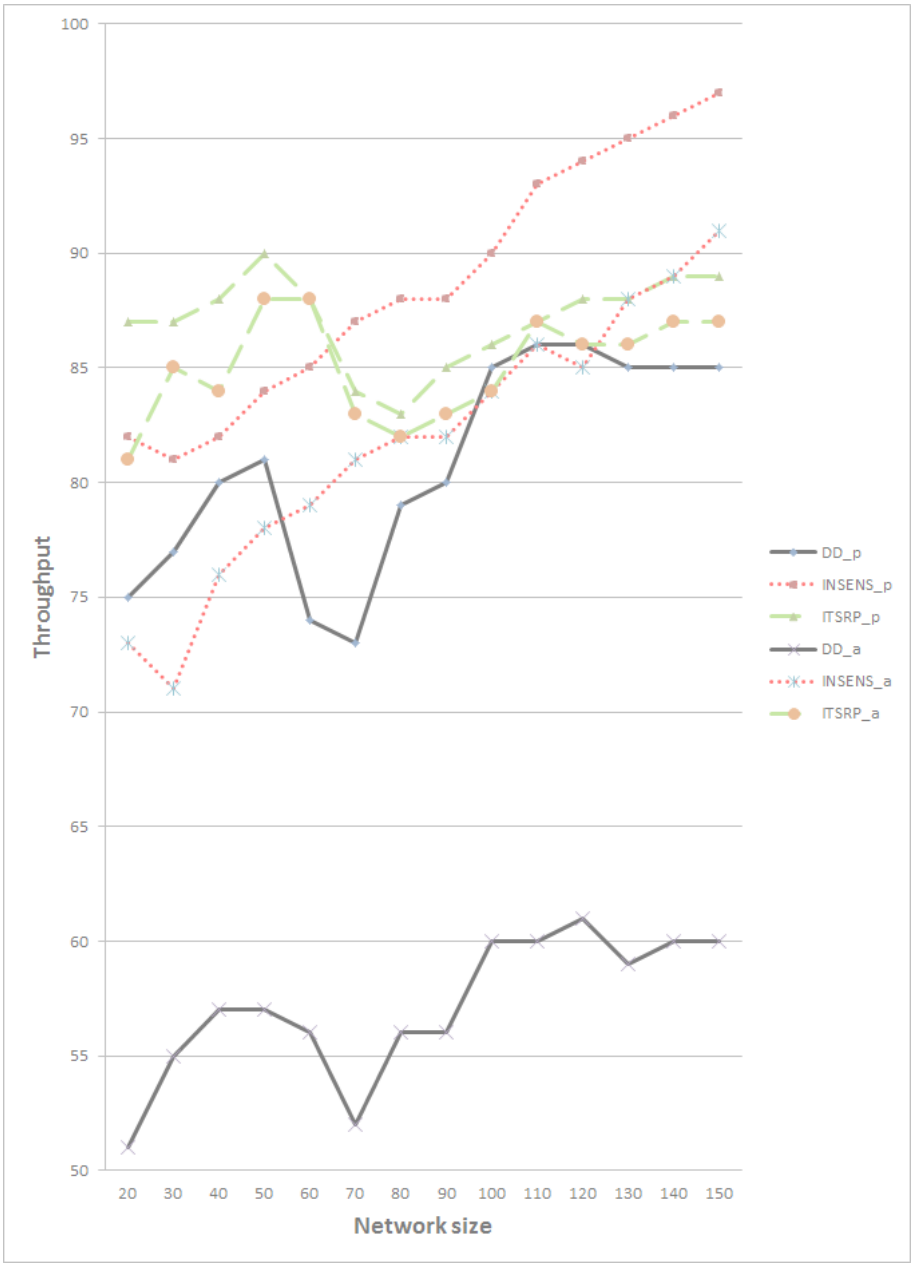


Figure 6.4: Comparison of average network throughput

6.2.4.2 Network overhead

The overhead of the network in our simulations is computed as the number of packets exchanged for the route discovery phase, i.e., the necessary packets to collect enough topology knowledge and then to build routing tables. This parameter corresponds to the price or the waiting- duration that the network need to pay to get ready.

We experienced in Fig. 6.5 that no matter what protocol were used, the overhead in case of *DOS attack* was higher than the one in *passive attack*. In other words, *DOS attack* impacted more severely the network. However, thanks to intrusion-tolerant properties brought by INSENS and ITSRP, the network overhead was kept much more stable than the one in case of DD that was extremely huge. In big size networks, INSENS is better than ITSRP with a slightly lower overhead.

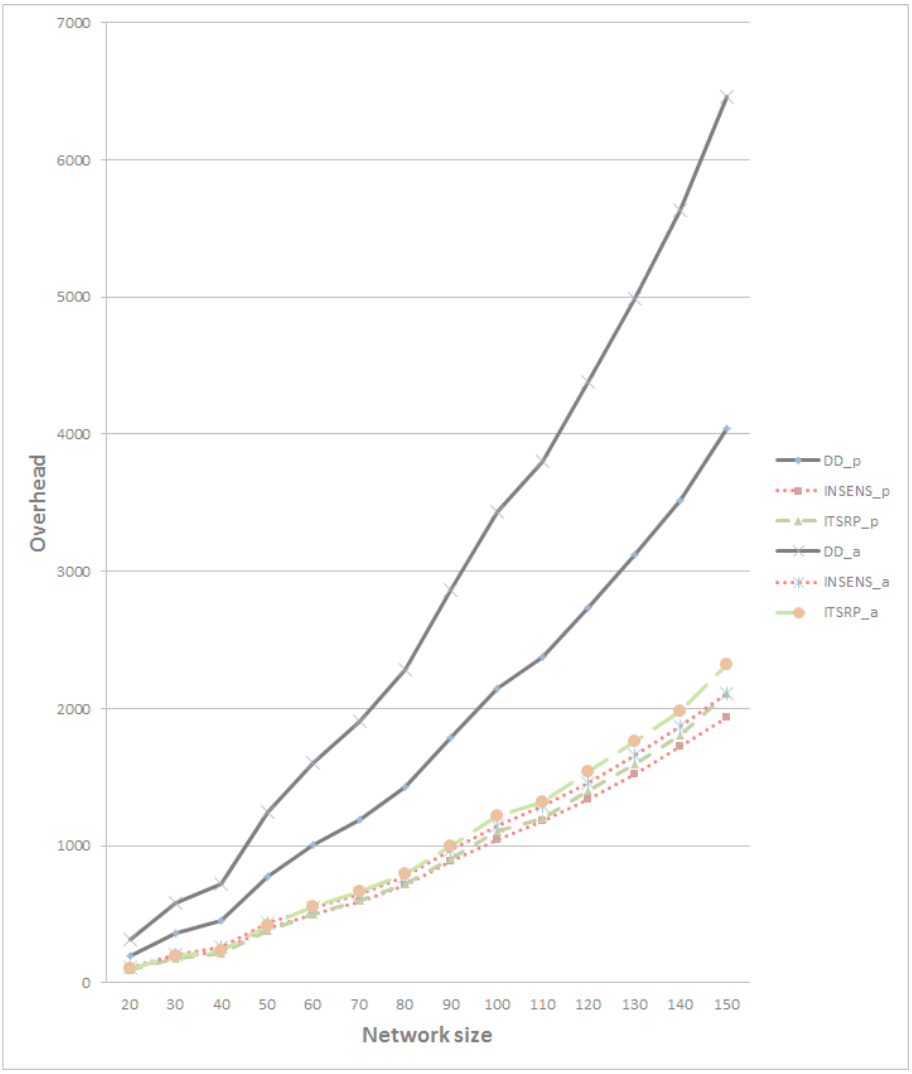


Figure 6.5: Comparison of average network overhead

6.2.4.3 Network lifetime

The network lifetime was defined as the duration since the simulation had been triggered until the first failure of a legitimate node. This parameter depends on two factors: the survivability of nodes (i.e., their ability to tolerate attacks) and the energy consumption volume. In Fig. 6.6, we can easily witness that the lifetime of the network using DD was strikingly smaller than the one when the network used INSENS or ITSRP, *DOS attack* wasted more energy than *passive attack* (thus, more reduced the lifetime of the network)

and INSENS presented itself as the best solution for dense WSNs with the highest lifetime (i.e., lowest energy consumption).

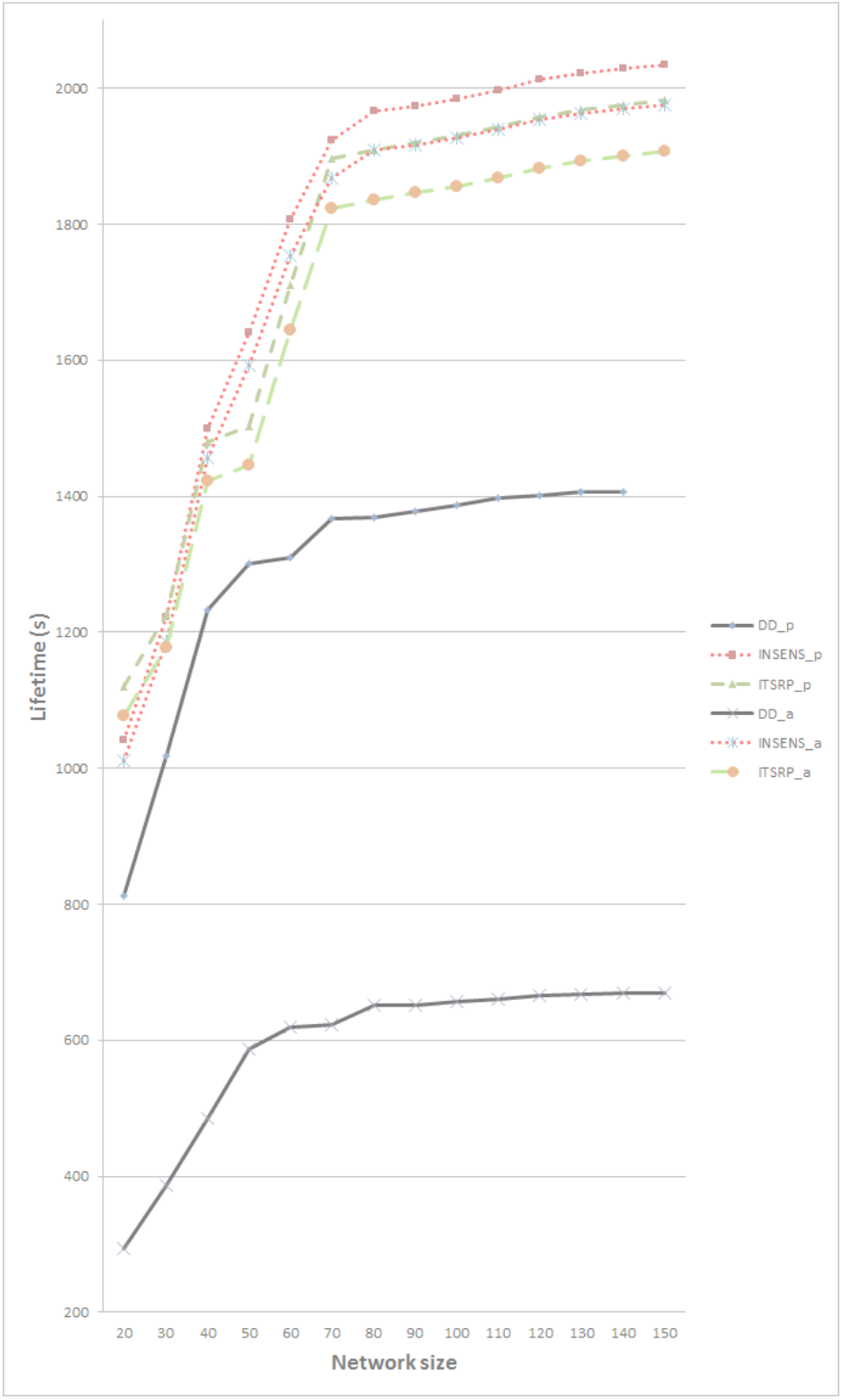


Figure 6.6: Comparison of average network lifetime

6.2.5 Improvement propositions

In this section, we would like to suggest some directions to improve two aforementioned protocols by fulfilling the missing issues as well as integrating more helpful functionality. These suggestions will be also our future works. Our wish is to achieve in near future a novel intrusion tolerant routing protocol that must be more efficient and stable.

Firstly, considering the results from aforementioned simulations, we found that INSENS seemingly dominates ITSRP. However, from our point of view, ITSRP still proves itself as a potential solution if it is invested more amelioration. INSENS is not as flexible as ITSRP and is expensive regarding possibly consumed energy to guarantee OWS and MAC computations. In contrast, as demonstrated, ITSRP uses LRTs for nodes where we can add fields to support our goal, i.e., intrusion tolerance. For example, in above-mentioned simulations, we assumed that the field *Energy* to send a message from N_i to N_k equals to the number of nodes in the routing path linking them. This assumption is for facilitating our simulations but in reality, *Energy* field deserves more investigations to help network still intrusion- tolerant but at a reasonable price. Benefiting this field or adding some other useful fields could be good ideas to upgrade ITSRP. Some good features of INSENS could be also integrated with ITSRP to gain a wise hybrid version.

Secondly, we only performed simulations in which the proportion of intruders i equals to 25%. In fact, there is a threshold i^* where the network will be rapidly corrupt if i exceeds i^*. We are considering this threshold in case of INSENS and ITSRP as a function of network size. Our goal, a better intrusion- tolerant routing protocol, should have a higher i^*.

Thirdly, other attacks like MitM (Man in the middle attack) should be also considered to test the protocols. Kinds of attacks depend on communication environments in which wireless sensor nodes are deployed. In fact, we are looking for a concise case study from an industrial project to better plan our simulations as well as to modify the protocols in order to fit practical criteria. Based on a detailed case study, we can also plan the strategies to prevent attacks and thoroughly overcome the missing issues of INSENS and ITSRP.

6.3 Emulation-based intrusion detection and tolerance

The idea of the approach presented in this section was inspired by [61, 62] in which the authors studied an emulation-based network intrusion detection systems to detect the presence of shellcode in network traffic. Their main idea is to try to execute (portions of) the network packet payloads in an instrumented environment and then check the execution traces for signs of shellcode activity. This inspired us thinking about an emulation-based detection module staying in the middle of the service requester and the service provider. An emulated infrastructure can be created to test the requests before they are sent to the real service provider. If these requests are determined free of security concern, they can approach the real infrastructure. Otherwise, a denial response will be returned.

Evidently, this approach will affect more or less the performance of the system (e.g., increase the response delay). However, looking at the bright side, the fact that the requester (insider or outsider) is tested before having access to real platform, enables both *intrusion detection* and *intrusion tolerance*.

6.3.1 General methodology

Fig. 6.7 illustrates the general methodology allowing simultaneously detect and tolerate intrusions. Suppose that the main entity/service provider (e.g., a database, a server, a gateway) receives a request from an authenticated entity (e.g., a user, a host). Instead of immediately processing the request, it refers to an additional *intrusion detection and tolerance framework*. This framework is an *emulated environment* which consists of a *VM (Virtual Machine)* acting the role of the requester and a *copied entity* acting the role of the service provider. The VM copies the initial request including original parameters and sends to the copied entity. A detection module is installed to analyze the request and the response, as well as detect abnormal intentions if any. Of course we can use the security monitoring framework aforementioned in this thesis here. If any security violation is detected, the request will be denied and a denial notification will be sent by the main entity to the requester. Otherwise, a normal response is provided.

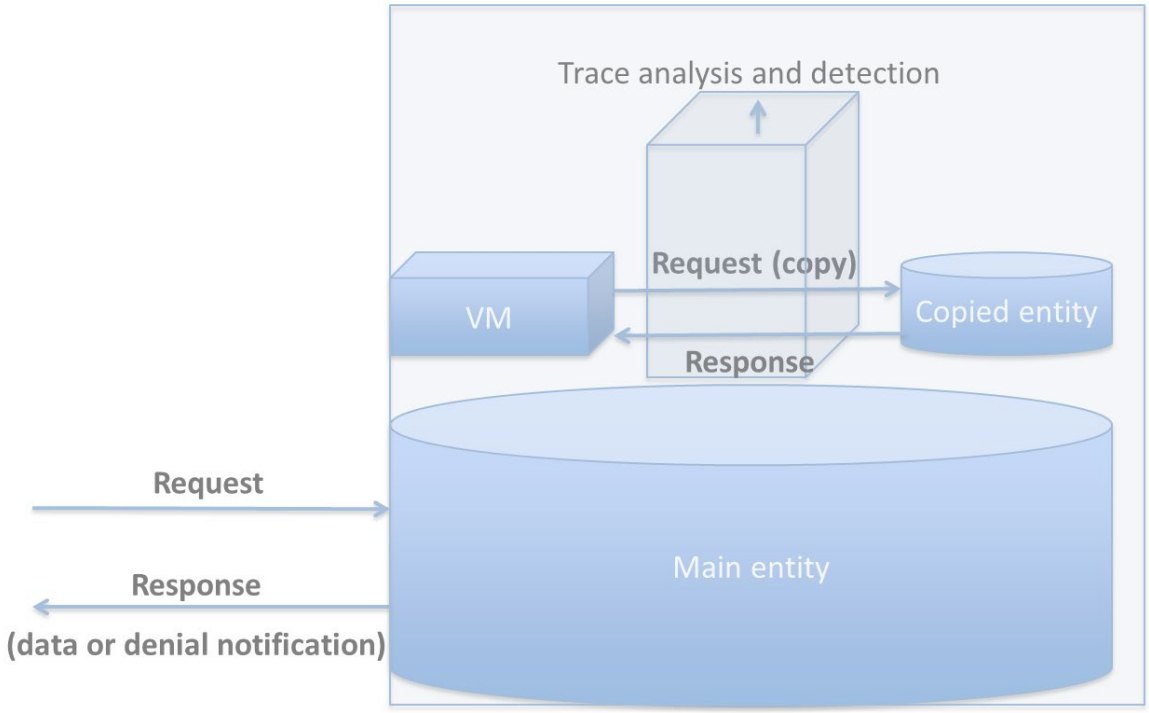


Figure 6.7: Emulation-based intrusion detection and tolerance methodology

In the common cases, security monitoring module is installed between the requester and service provider to analyze the trace of communications (i.e., request and response). If an anomaly is detected, the request will be denied. However, this approach requires the detection module a real-time reaction. That means it has to react nearly immediately after the incident. A late detection can lead to information leakage, network failure or data loss. In many cases, the real-time detection cannot be always assured.

Our emulation-based approach can overcome this challenge, with the assumption that in the additional detection and tolerance module we can reduce the analysis complexity to have a real-time detection. This assumption is realistic if we consider the following measures:

- The *copied entity* is a reduced version of the *main entity*. For example, if the main

entity is a database, the copied entity can have the same structure but just some sample data. Depending on the real case study, we can define the copied entity differently in order to speed up the analysis and detection.

- A set of normal behavior can be achieved by learning the emulated system in response to a set of normal requests. This demands a thorough learning phase in which we have to input as many normal requests as possible. This can be done by using testing techniques and generating a comprehensive test-suite.
- In the emulated system, in addition to the trace analysis, security monitoring can be applied as well on the VM and the copied entity to facilitate the detection.

6.3.2 A novel approach for SQL injection detection and tolerance

SQL Injection (SQLI) is a wide spread devastating security problem holding the first place in the list of top 10 web vulnerabilities according to OWASP (Open Web Application Security Project) Foundation [18]. The work presented in [63] revealed that approximately 63% of websites are still believed to be vulnerable to SQLI attacks. Exploitation of SQL injection vulnerabilities in web-based application can result in serious consequences such as authentication bypassing, leakage of sensitive personal information and data loss. It is worth noting that people usually reuse usernames and passwords so the leakage of these data in unsafe websites can damage also the safe websites. In this section, we provide a novel approach for SQLI detection and tolerance, principally in the context of database-driven Web applications.

6.3.2.1 Problem statement

Many web-based programs store and retrieve sensitive information (e.g., usernames, passwords, names, addresses, phone numbers, and credit card details) from databases by executing SQL queries. These queries often include user supplied inputs. If these are not sanitized properly before being included and passed to the back-end database, the attacker can leverage the syntax and capabilities of SQL by adding their own evil SQL code. Consequently, the intended structures of dynamic queries get altered and result in SQL Injection (SQLI) attacks.

As illustrated in the Fig. 6.8 [64], a database-driven Web application commonly has three tiers: a *presentation tier* (e.g., a Web browser or rendering engine such as Internet Explorer, Safari, Firefox, etc.), a *logic tier* (a programming language (e.g., C#, ASP, .NET, PHP, JSP, etc.), and a *storage tier* (i.e., a database such as Microsoft SQL Server, MySQL, Oracle, etc.). The *presentation tier* sends requests to the *logic tier*, which services the requests by making queries and updates against the *storage tier*. Sometimes there can be additionally an *application tier* between the Web server and the database interacting with the data store and imposing application and business logic.

SQLI can be performed if the attacker is able to insert SQL code into application/user input parameters and this code is passed and executed in the back-end SQL database. The simple countermeasure eliminating SQLI is to sanitize the input parameters first before passing. However, attackers have their techniques to evade. That is the reason why we propose a SQLI detection and tolerance framework to immunize the running system in case there is somehow an evasion.

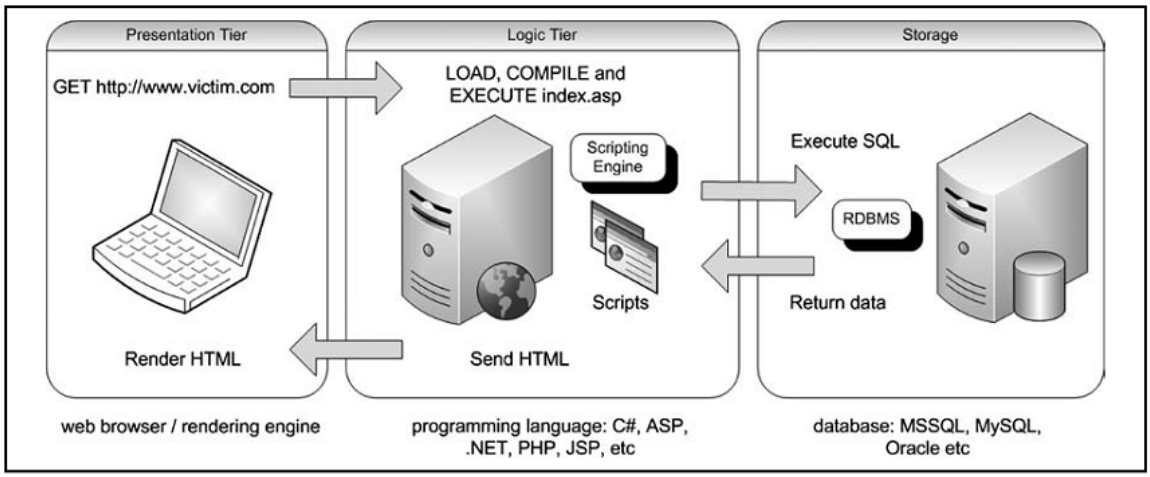


Figure 6.8: Simple Three-Tier Architecture of database-driven Web Applications

6.3.2.2 Methodology

The Fig. 6.9 depicts the concrete methodology for the SQL injection case study, in applying the method presented in Section 6.3.1. The VM queries the copied database instead. A detection module is installed to inspect the query.

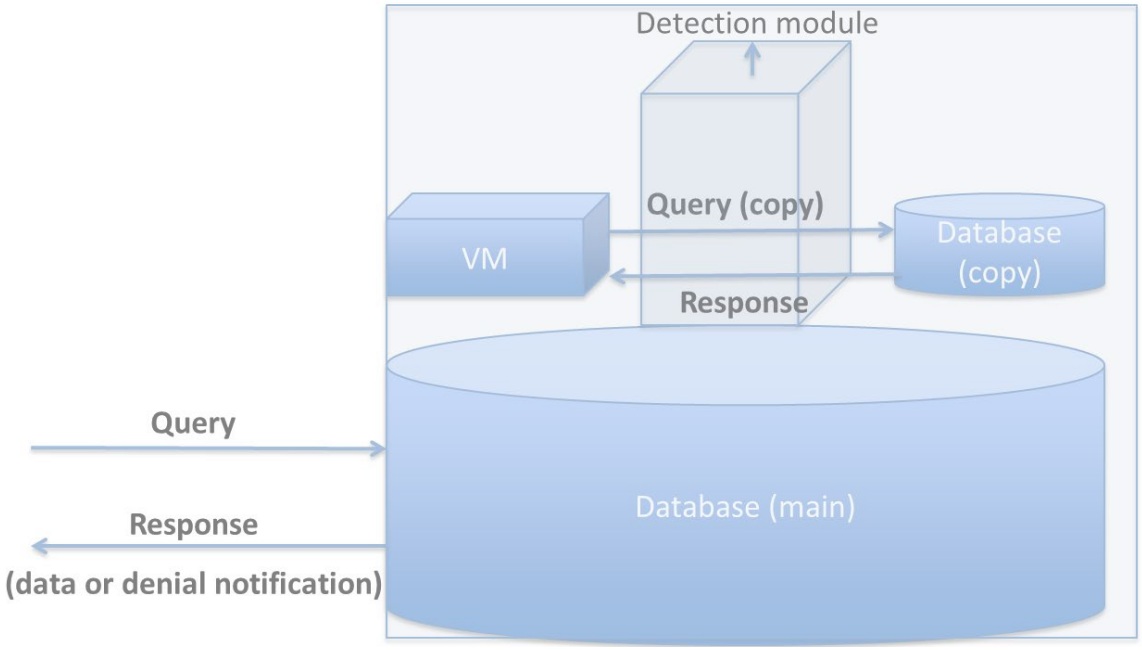


Figure 6.9: SQL injection detection and tolerance methodology

In the implementation validating the approach, we used the *information theoretic detection technique* inspired by [65]. More specifically, we calculate the entropy of the query sent by the VM and the query executed in the copied database. Any difference between them signalizes an anomaly, i.e., most probably a SQL injection. Assuming that $\{x_1, x_2, ..., x_N\}$

is the set of all tokens presented in the query q, the entropy related to q is calculated as follows:

$$H(q) = H(x_1, x_2, ..., x_N) = -\sum_{i=1}^{N} P(x_i) * logP(x_i) \quad (6.1)$$

where $P(x_i)$ is the probability of the token x_i appearing in q.

For instance, let q = "select id, password from tlogin where login ="".$sLogin. "' and password ="".$sPassword. ""'. The number of tokens in total is 12 (Tab. 6.4)

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
select	id	password	from	tlogin	where	login	=	sLogin	and	password	sPassword
1/13	1/13	1/13	1/13	1/13	1/13	1/13	2/13	1/13	1/13	1/13	1/13

Table 6.4: Token probability distribution for a benign query

The corresponding entropy is measured then:

$$H(q) = -11/13 * log(1/13) - 2/13 * log(2/13) = 3.54 \quad (6.2)$$

The detection approach composes two phases: training phase and detection phase (Fig. 6.10). In the training phase, we enter a complete legitimate test-suite to the input (i.e., white box, supervised learning). The *server side script code* is analyzed and the *entropy* corresponding to each legitimate query is computed. The *entropy* is recorded at the *instrumented code*. In the detection phase, when a query comes, the generated dynamic query is analyzed to compute the entropy. The we compare this value with the value learned. If a deviation is found, a malicious query is detected.

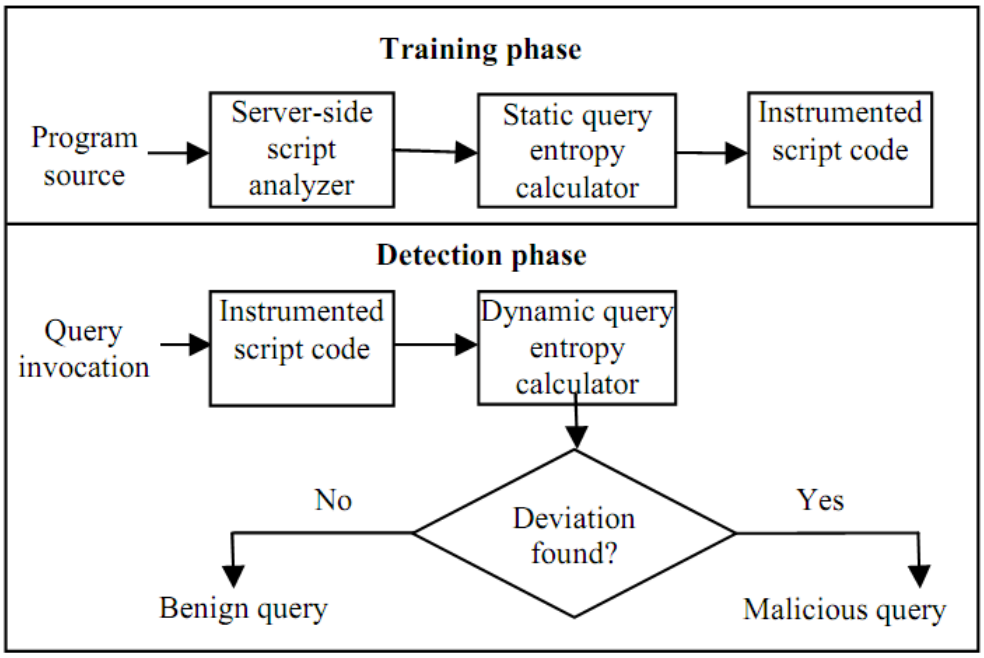


Figure 6.10: Information theory-based SQLI detection approach

Let's go back to the aforementioned instance, suppose that we receive a malicious query "select id, password from tlogin where login =' ' or 1=1 –' and password =' "'. The token distribution corresponding to the query executed in the storage tier is presented in the Tab. 6.5.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
select	id	password	from	tlogin	where	login	=	or	1
1/12	1/12	1/12	1/12	1/12	1/12	1/12	1/6	1/12	1/6

Table 6.5: Token probability distribution for a malicious query

The new entropy is calculated:

$$H(q) = -2/3 * log(1/12) - 1/3 * log(1/6) = 3.25 \tag{6.3}$$

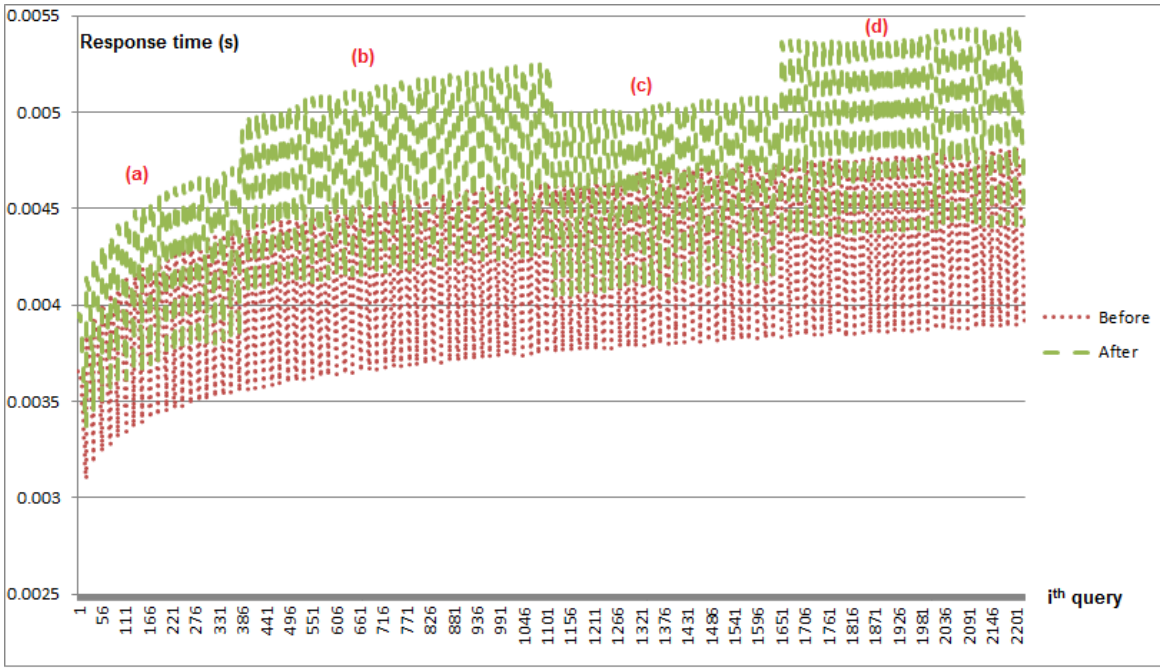


Figure 6.11: The response time with and without adding the detection and tolerance module: (a), (c) - Malicious queries; (b),(d)- Benign queries

6.3.2.3 Implementation

We implemented the framework in an Apache web-server. In order to evaluate the proposed model, we used three open-source PHP programs available from *sourceforge.net* including PHP-Address [1] (address and contact information manager), Serendipity (blog management program) [2], and PHP-fusion (content management system) [3]. These three open-source

[1] http://sourceforge.net/projects/php- addressbook/?source=directory
[2] Serendipity PHP Weblog System: http://www.s9y.org/
[3] http://sourceforge.net/projects/php-fusion

programs are ranked among the top five most downloaded applications for the website development.

For malicious queries entering the input, we inherited test inputs from OWASP security testing guide [18] in taking into account tautology, union, piggyback, inference, and hex encoded queries. We divided the queries into four sets (two sets of malicious ones and two sets of benign ones), and passed them in the following order: malicious - benign - malicious - benign. It is worth reminding that in each set, we sorted the queries based on their length (in tokens).

The main concern that we would like to assess is the performance of the overall system before and after integrating the *detection and tolerance module*. Obviously, additional detection and tolerance module requires additional processing time, thus, the system performance is more or less negatively affected. The metric that we used to evaluate the performance is the "response time", i.e., the total amount of time the system takes to process a query and return a response (data or denial notification). The results are displayed in the Fig. 6.11 and Fig. 6.12 in which we noticed several phenomena as follows:

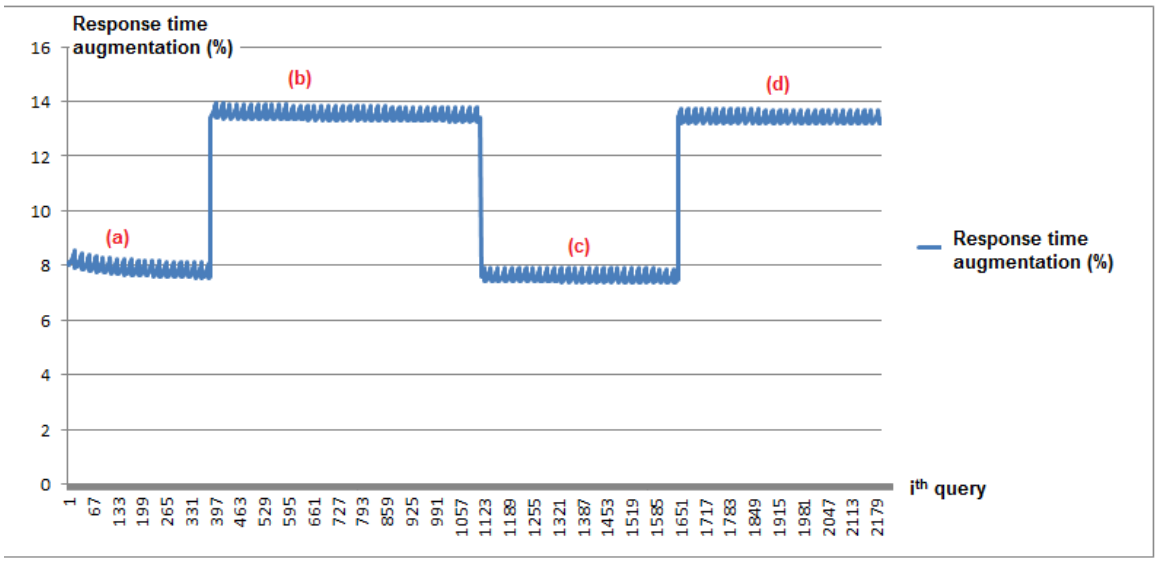


Figure 6.12: The augmentation in response time caused by the additional module: (a), (c) - Malicious queries; (b),(d)- Benign queries

- From the local point of view, the longer the query was, not necessarily the more time the system required to process (i.e., higher response delay). That means the fact that "the query q_1 has more tokens than the query q_2 does" does not result in the event "the response time corresponding q_1 is bigger than the one corresponding q_2". However, from the global point of view, if s_1 and s_2 are two sets of queries and any query in s_2 is longer than every query in s_1, the average response time related to queries in s_2 will be likely bigger than the one in s_1. We can see this trend in the Fig. 6.11.

- The system without our proposition reacted basically the same regardless the malicious or legitimate queries. When the intrusion detection and tolerance module was added, it took more time for the system to deal with the sets of benign queries and with those

of malicious queries. This phenomenon can be explained by the fact that in case of SQLI queries, the systems simply returned a denial notification without querying the database. Indeed, when a query comes in, either it is legitimate or abnormal, the same processes are performed to inspect it. If it is determined abnormal, the system simply returns a refusal. Otherwise, the query is executed in the data storage side and data is returned. The benign queries take thus more time than the malicious ones.

- In terms of performance degradation represented by the augmentation in response time (Fig. 6.12), we witnessed a pretty stable response time augmentation around 9% regarding the malicious queries ((a) and (c)) and a higher loss up to 14% regarding benign queries ((b) and (d)). This can be explained in the same way like the phenomenon mentioned above: the system with our proposition integrated need more time to process normal queries than abnormal queries, whilst, the raw system reacts almost the same against those two types.

- We did not recognize any noticeable difference in response time regarding the injection type (tautology, union, piggyback, inference, and hex encoded queries).

6.3.3 Discussion

Although in the SQLI case study, the response time increases only approximately 9% to 14% due to the extra module, the third phenomenon presented in the Section 6.3.2.3 is really the weakness of the approach. Indeed, the system with the intrusion detection and tolerance module included need more time to process normal requests than abnormal requests. This is a bad point because the normal requests are more common than the abnormal ones.

Nevertheless, the approach can be optimized thanks to the following possible improvements:

- Detection technique
  A quick detection surely speeds up the overall system. However, it must be efficient enough. In our case study, the SQLI detection technique based on information theory was chosen instead of a lot of others because it can avoid evasion tricks from attackers. A study on the integration of other SQLI techniques to our framework is scheduled as a future work.

- The configuration of the VM and the definition of the copied entity
  Evidently, the more powerful the VM is, the faster the system reacts. Whilst, the definition of the copied entity must be detailed enough to help the detection process, but at the same time, it should be abstract enough to reduce the time and resource consumption.

- Big Data
  As demonstrated above, the response time augmentation is mainly caused by the execution of queries in the storage tier. The more voluminous the database is, the slower the system processes. Therefore, the techniques dealing with Big Data can be considered to resolve this problem.

Chapter 7

Conclusion and Future Work

Contents

7.1 Conclusions

In summary, this dissertation presents a security monitoring framework that takes into consideration different types of audit dataset corresponding to different platforms. Its originality includes the proposition of some novel approaches based on Supervised Machine Learning to pre-process and analyze the data input. We took advantage of the novel detection techniques to leverage the data processing speed and assure quick detection even in large scale system with high traffic. In addition, we implemented an adaptation to 6LoWPAN traffic of IoT devices. To the best of our knowledge, this is the first time a security monitoring tool has been adapted to work in such resource-constraint environment.

Our framework is validated in a wide range of case studies as follows.

1. Traditional TCP/IP networks:
   We started the research over common TCP/IPv4 networks which have been the most used all over the world.

   - ARP spoofing:
     Although *ARP spoofing* is a very classic attack, we witnessed a high probability that it is still possible to perform it in some Wired and Wireless LANs. The problem is that attackers can realize further attacks or investigations to exploit and receive sensitive information from the network's members based on this basic attack. In such cases, *availability, confidentiality and integrity* of the data passing throughout the network cannot be assured. We presented some sample results that we successfully hacked including personal log-in information (e.g., user-name, password, hobbies), mis-configurations in some servers, etc. We proposed then some countermeasures to avoid those vulnerabilities and advice for Internet users to protect themselves against evil intentions.

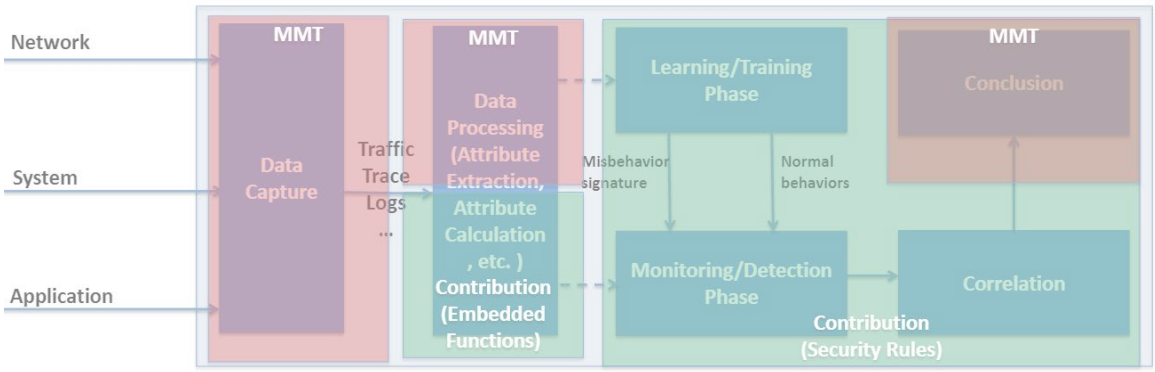


Figure 7.1: Contribution summary in HTTP User-Agent field case study

- HTTP User-Agent field case study:
  HTTP is a predominant communication protocol which is widely used in every computer networks nowadays. User-Agent is a field in HTTP request message which is modifiable and exploitable to perform attacks. We studied existing works dealing with this problem, clarified some possible attack scenarios (e.g., SQL injection, XSS), as well as cited some actually successful attacks presented in literature. After that, we proposed the methodology and implemented the framework to monitor that field and detect attacks (Fig. 7.1). We affirmed that even if User-Agent field is not intentionally abused by the attackers, identifying suspicious or strange User-Agent strings is not trivial. In contrast, it can be good source to detect security violations in their early steps. Our framework based on MMT proved its good performance in comparison with Snort and Tcpdump.

  The framework was also extended to HTTP traffic in general in considering techniques aiming improve the performance (e.g., dimension reduction, machine learning). We suggested that *Smartphone-based security monitoring, Web pop-up and Spam avoidance* can be achieved by the similar approaches.

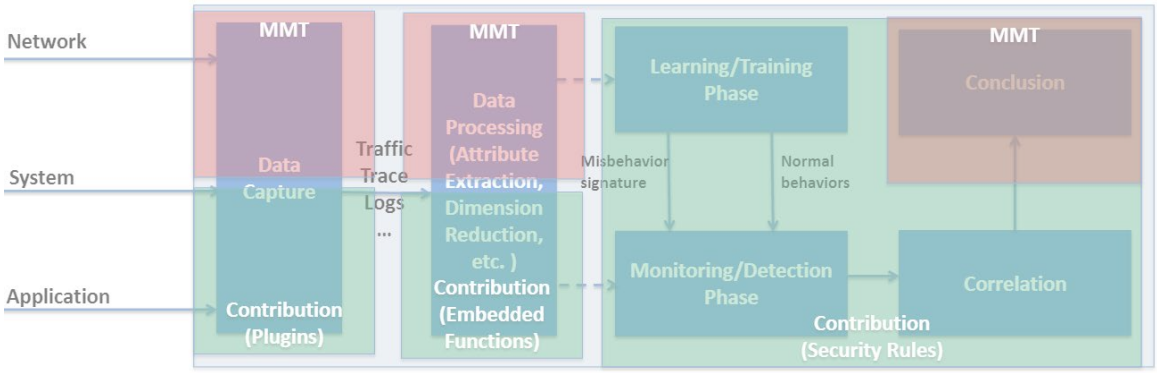


Figure 7.2: Contribution summary in 6LoWPAN-based WSNs case study

2. 6LoWPAN-based WSNs:
   To our knowledge, 6LoWPAN traffic is so far not understandable to any monitoring

solution, including Snort, Suricata and Bro. That is the reason why we adapted MMT to work in 6LoWPAN-based WSNs by adding several new plug-ins ((Fig. 7.2)). After having the ability to analyze 6LowPAN traffic, we proposed then a number of algorithms and techniques to detect anomalies in such networks based on supervised learning including statistical learning and information theory. We explained in which scenarios they can be applied and what metrics can be used.

Experiments proved our proposition's applicability and its reasonable performance. We validated the usability of the link-weight (e.g., the travel time) and the entropy but many other metrics can be probably exploited. The approaches are also applicable to other case studies, namely communications in Ad-hoc environment.

3. Intrusion tolerance by design:
   We emphasized that *intrusion/attack tolerance* is a feature that should be integrated into modern computer systems. This goal can be achieved by design or by reactive responses after detecting a security threat to tolerate it. We focused on the former type of approaches.

   - Intrusion tolerant routing in WSNs:
     We studied two most famous intrusion tolerant routing protocols namely INSENS and ITSRP. We provided a theoretical analysis demonstrating how they can tolerate common attacks. A comparative simulation-based performance evaluation was presented to assess their applicability. The performance metrics used in the simulations consists of network throughput, network overhead and network lifetime. INSENS seemingly dominated ITSRP but the flexible structure in ITSRP packets open the windows for new amelioration. For both two protocols, we pointed out their missing issues and proposed some improvements possibly making them practical and more efficient.
   - Emulation-based intrusion detection and tolerance:
     We proposed an extension of the framework to detect and tolerate attacks at the same time. A general architecture was presented, beside a validation regarding SQL injection case study.

Throughout the case studies, our techniques do not require any touch to the payload of packages. The private data contained in payload is therefore not inspected, then our approach is applicable for high privacy communications. Thanks to plug-in architecture, the framework is extensible to analyze new type of audit data. Therefore, organizations who want to keep their data private can perform an obfuscation phase first before passing the data to the input gate of our framework.

7.2 Future Work

Throughout the dissertation, we have discussed the limitations of our security monitoring framework in each case study which were unfortunately unable to be fulfilled in the context of the thesis. This section synthesizes several ones which are, in our opinion, important and deserve more investments. In short, there are actually a number of open problems and future research directions, including but not limited to the following issues:

- Detection and investigation in large scale systems and the Big Data issue
  Computer networks are continuously more complex and compose a larger number of entities. The network bandwidth is also higher and higher. It is thus not surprising if the audit data under-monitored becomes Big Data. This is also the main concern in the context of IDOLE project. The problem relies on the research on *data mining* as well as powerful *machine learning algorithms*. Good algorithms will allow not only the rapidity of data processing but also efficiently learning and detecting events related to security incidents. *Dimension reduction* is also another direction to reduce the number of variables (i.e., attributes) under consideration. This is surely useful when the audit datasets are high-dimensional (e.g., more than 10). There are a number of possible algorithms which can be taken into account, for instance K-means, PCA, LDA, RP, etc. We initial assess some of them in this thesis but just as demonstrations in some simple cases.

- Distributed approach and optimization problem
  The more the systems become sophisticated, the harder the monitoring tool can detect violations. Sometimes it is impossible to realize this task in listening to the traffic passing by a single point inside the network. Distributed approaches with collaborative works are therefore a need. This is one of our goals in designing the framework and should be a feasible task thanks to the open architecture of MMT. But how the agents collaborate themselves and where we should put the agents are still problematic. One idea is to take into consideration the concept *Multi-Agent* processing. However, this issue merits further thorough considerations.

  Another idea coming up from 6LoWPAN-WSNs case study is to use entropy to determine the best listening points to put the probes. As concluded in the section 5.4.2.2, the probes should be installed in points so that we can get the audit data logs owning the smallest entropy.

- Mobility concern and the possibility for VANETS
  In the work done with 6LoWPAN-based WSNs, we normally assumed that our networks were static rather than mobile. Nevertheless, sensor nodes in reality sometimes moves. In these cases, the efficiency of the detection could be damaged. Mobility models are there for a need to avoid significant false positive rate. Together with distributed issues aforementioned, this problem refers to agent-based modeling and geographical information data. To our knowledge, it can be resolved by using GAMA platform [1]

  If the mobility concern is overcame, the framework can be extended for VANETs because VANETs and WMNs have multiple similarities. The main difference between them is the mobility since VANETs consider moving cars as nodes.

- Adaptive intrusion tolerance
  We consider that *intrusion tolerance* can be the next-step after detecting an intrusion. In the other words, the running system will react somehow to be able to continue to function properly with minimal degradation of performance. For example, the system was designed in different models. In case of being attacked, new variant model is

[1] GAMA homepage: http://gama-platform.org/

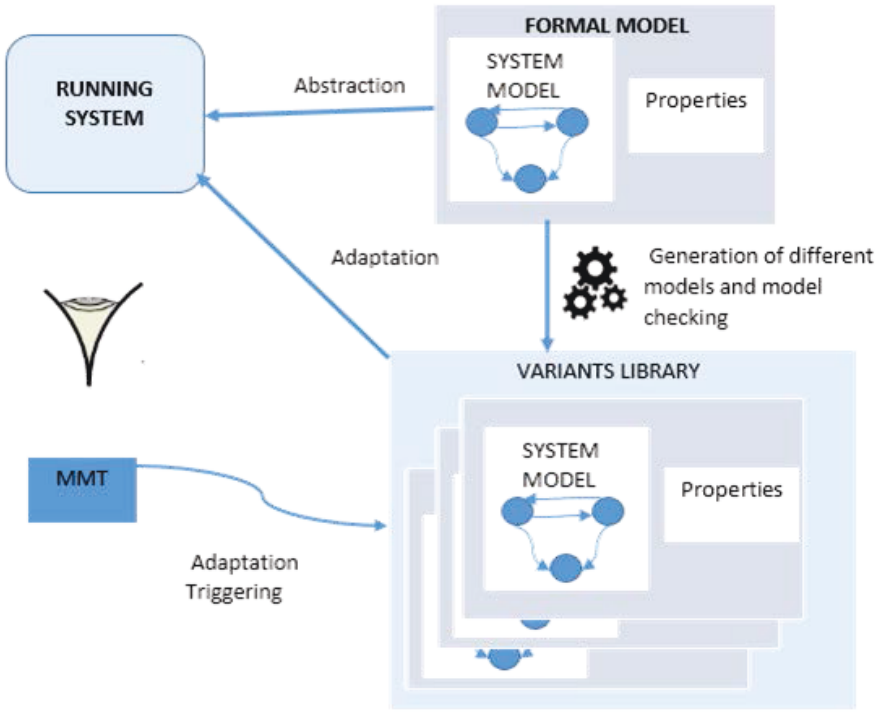


Figure 7.3: An adaptive intrusion tolerance example

called then the attack is detoxified (Fig. 7.3). This topic deserves more considerations and in fact, it is actually an active subject in research community.

Appendix A

FIT IoT-Lab Hardware Information

A.1 WSN430 Open Node

The WSN430 open node is a WSN430 node based on a low power MSP430-based platform, with a fully functional ISM radio interface and a set of standard sensors. Concerning the radio, two versions are developed: version 1.3b presents an open 868 MHz radio interface while version 1.4 has an IEEE 802.15.4 radio interface at 2.4 GHz.

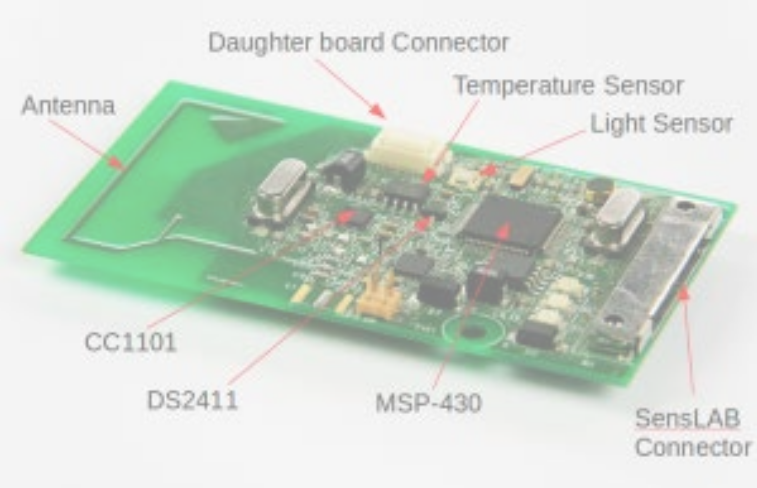


Figure A.1: WSN430 Open Node

In details, the main hardware components contained in the node are :

- Micro-controller MSP430 offering 48kbyte ROM, and 10kbyte RAM
- Sensors (Temperature, Sound, Ambient light)
- Radio: 868MHz radio interface for version WSN430 1.3b and 2.4GHz radio interface for version WSN430 1.4
- Serial Number: An EEPROM serial number is available thanks to a Maxim DS2411 chip, giving each node a unique identifier, readable by the MPS430 firmware over a 1-Wire interface.

- Flash memory: 1MByte
- Battery charger is controlled by a Microchip MCP73861 chip. It allows battery charge, and is supervised by MSP430 through 2 digital outputs
- Three LEDs (green, red, blue)

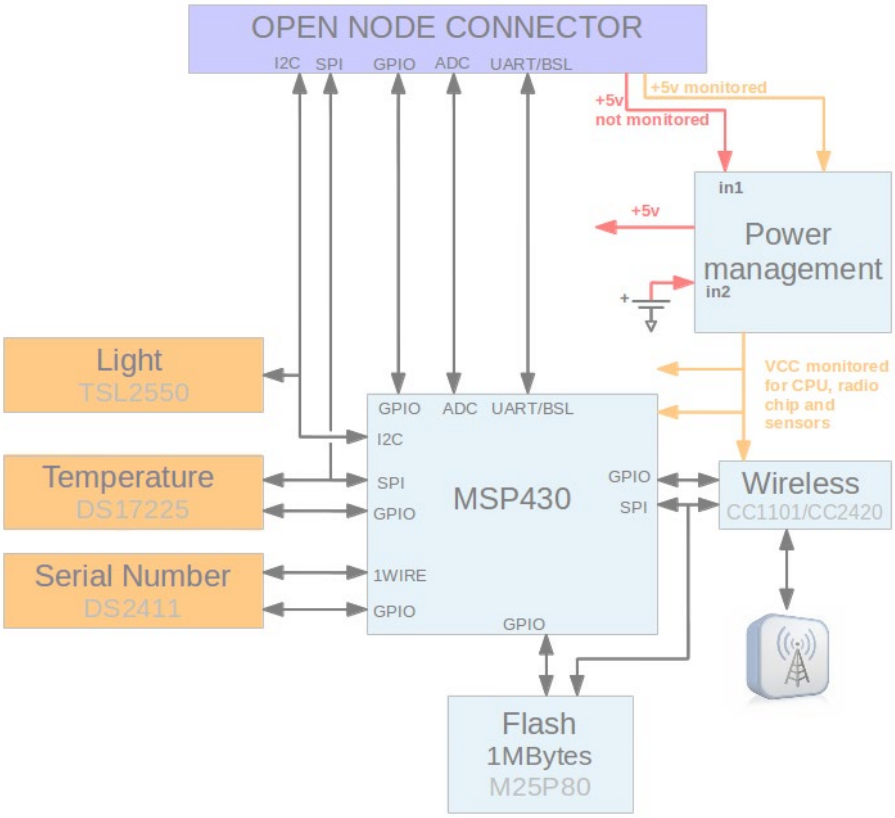


Figure A.2: WSN430 Open Node's hardware in detail

A.2 M3 Open Node

The M3 open node is based on a STM32 (ARM Cortex M3) micro-controller. Like the WSN node this next generation contains a set of sensors and a radio interface. Main evolutions are a more powerfull 32-bits processing, a new ATMEL radio interface in 2.4 Hz and more sensors.

In details, the main hardware components contained in the node are :

- ST2M32F103REY (72 MHz, 32bits, 64kB RAM)
- Radio interface 2.4 GHz AT86RF231
- Sensors
  - Light sensor ISL29020

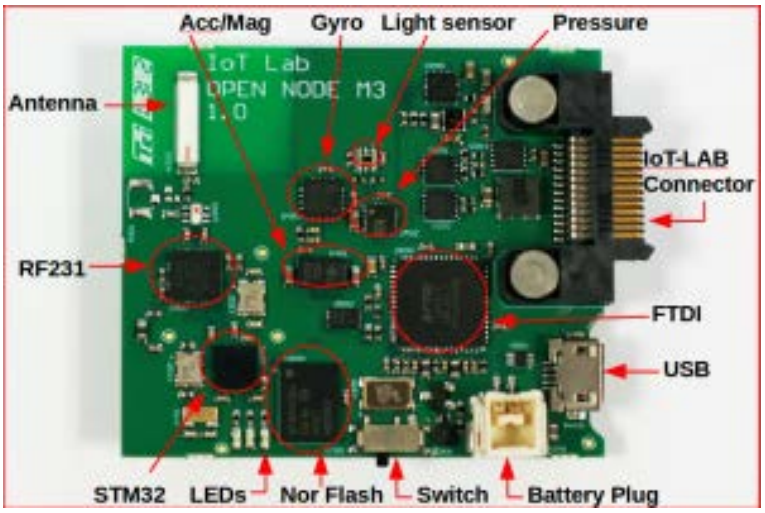


Figure A.3: M3 Open Node

  - Pressure sensor LPS331AP
  - Tri-axis accelerometer/magnetometer LSM303DLHC
  - Tri-axis gyrometer L3G4200D
- External Nor flash (128 Mbits) N25Q128A13E1240F
- Three LEDs (green, red, orange)
- 3,7 V LIPO battery (650 mAh)

A.3 A8 Open Node

The A8 open node is the most powerful IoT-LAB node and allows to run high-level OS like Linux. The main processor is a TI SITARA AM3505 (Arm Cortex A8) combined with a STM32 micro-controller and a radio interface. It enables to run applications used in advanced devices such as set-top box or smart-phone/tablet in order to concentrate sensors information coming from a wireless sensor network.

Main hardware components contained in this node are :

- A Variscite VAR-SOM-AM35 CPU which a high performance System On Module. It is a board of the shell based on a TI SITARA AM3505 (600 Mhz, 256 MB)
- A “co-microcontroller” based on ST2M32F103REY (72 MHz, 32bits, 64kB RAM) which controls :
  - Radio interface 2.4 GHz AT86RF231
  - Tri-axis accelerometer/magnetometer L3G4200D
  - Tri-axis gyrometer LSM303DLHC
- A USB device FTDI2232H to control UART and JTAG

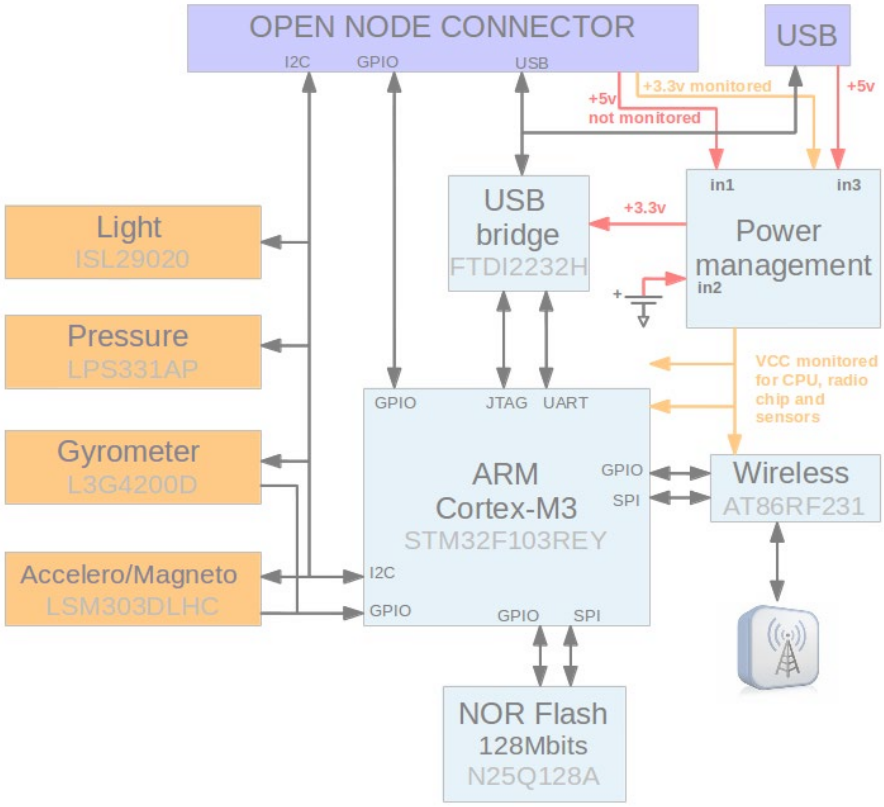


Figure A.4: M3 Open Node's hardware in detail

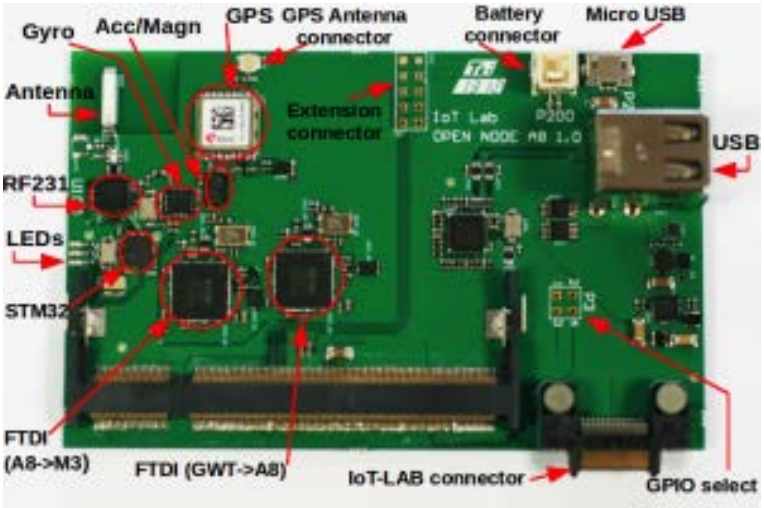


Figure A.5: A8 Open Node

- A GPS device MAXQ (optional)
- Three LEDs (green, red, orange)
- 3,7 V LIPO battery (600 mAh)

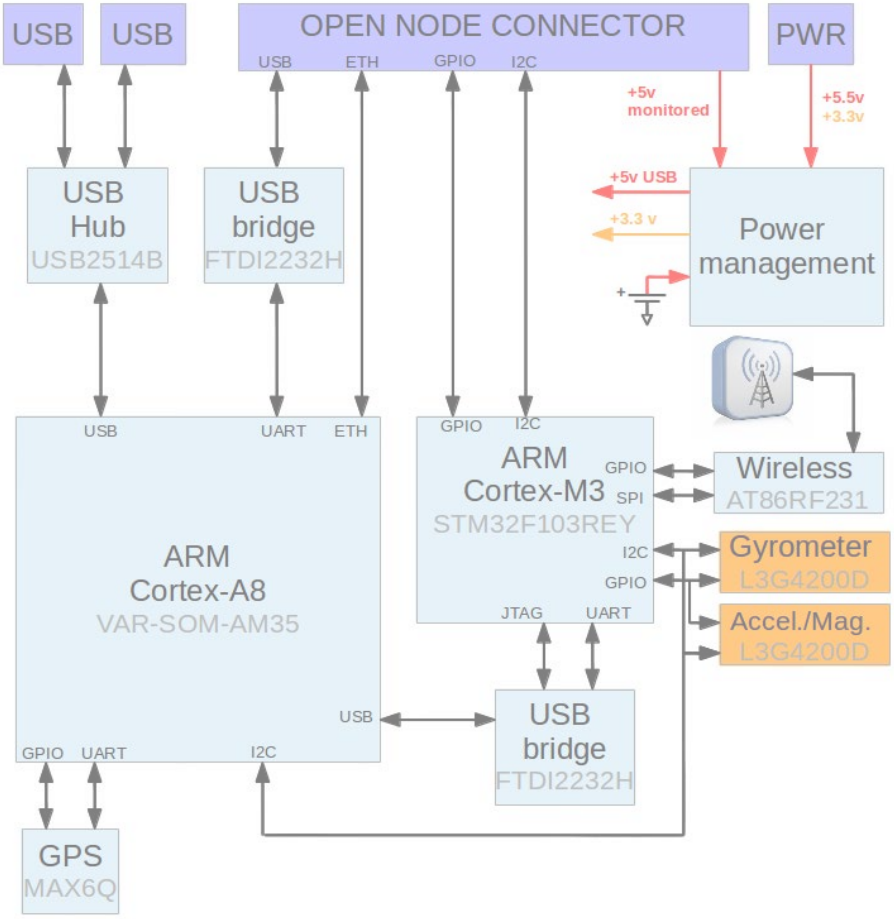


Figure A.6: A8 Open Node's hardware in detail

Appendix B

A Taxonomy of Attacks in Vehicular Ad-hoc Environment

The Tab. B.1 summaries a taxonomy of common attacks in Vehicular Ad-hoc Networks (VANET) in which we characterize the attacks by three attributes: (1) Type of attacker, (2) Violated Security Properties, (3) Class of attacks. A detailed description of the attacks can be found in [66]

Attack	Type of attacker	Violated Security Properties	Class of attacks
Sybil	I.*.A.*	Authentication	NA
Bogus Info Bus telegraph	I.R.A.*	Integrity Authentication	AA
Impersonation Masquerade	*.*.A.L	Authentication	NA, MA
Timing	I.M.A.*	Integrity Authentication	TA
GPS Spoofing	I.R.A.L	Authentication	NA, AA
Hidden vehicle	I.M.A.L	Authentication	AA
Tunnel	I.R.A.L	Authentication	NA, AA
Illusion	I.R.A.L	Authentication	NA
ID Disclosure	I.R.A.L	Authentication Privacy	NA MA
DoS & DDoS	*.M.A.L	Availability	NA
Black Hole	I.M.A.L	Availability	NA
Wormhole	I.M.A.E	Availability	NA
Malware & Spam	I.M.A.*	Availability	NA
MiMA	*.*.A.L	Confidentiality Authentication	NA

Table B.1: A Taxonomy of Attacks in Vehicular Ad-hoc Environment

B.1 Type of attacker

Inspired by [67,68], we characterize an attacker by **Membership. Motivation. Method. Scope** where:

- Membership stands for Insider (I) or Outsider (O)
- Motivation for Malicious (M) or Rational (R)
- Method for Active (A) or Passive (P)
- Scope for Local (L) or Extended (E)
- A star (*) indicates that the corresponding field can take any value.

For example, an attacker I.R.A.L is an insider who behaves rationally, and performs active attacks in restricted areas.

B.1.1 Insider vs. Outsider

If the attacker is a member node who can communicate with other members of the network, it will be known as an Insider and able to attack in various ways. Whereas, an outsider, who is not authenticated to directly communicate with other members of the network, have a limited capacity to perform an attack (i.e., have less variety of attacks).

B.1.2 Malicious vs. Rational

A malicious attacker uses various methods to damage the member nodes and the network without looking for its personal benefit. On the contrary, a rational attacker expects its own benefit from the attacks. Thus, these attacks are more predictable and follow some patterns.

B.1.3 Active vs. Passive

An active attacker can generate new packets to damage the network whereas a passive attacker only eavesdrop the wireless channel but cannot generate new packets (i.e., less harmful).

B.1.4 Local vs. Extended

An attacker is considered as local if it is limited in scope, even if it possesses several entities (e.g., vehicles or base stations). Otherwise, an extended attacker broadens its scope by controlling several entities that are scattered across the network.

B.2 Violated Security Properties

B.2.1 Confidentiality

In VANETs, the definition of confidentiality refers to “confidential communication” [69]. In a group, none except group members are able to decrypt the messages that are broadcasted to every member of group; and none (even other members) except a dedicated receiver member is capable to decrypt the message devoted to it.

B.2.2 Integrity

It ensures that data or messages delivered among nodes are not altered by attackers. This concept in VANETs often combines with the concept "authentication" to guarantee that: A node should be able to verify that a message is indeed sent and signed by another node without being modified by anyone. In order to gain this property, Data Verification is also required: Once the sender vehicle is authenticated, the receiving vehicle performs data verification to check whether the message contains the correct or corrupted data.

B.2.3 Availability

The network should be available even if it is under an attack without affecting its performance. This concept of VANETs is not different from itself in other kinds of networks but not easy to ensure because of the mobility in high speed of vehicles.

Besides three main security requirements above, the following security aspects should be also satisfied in VANETs:

B.2.4 Privacy

The profile or a driver's personal information must be maintained against unauthorized access. We consider the following two cases:

- Communications between vehicles and RoadSide Units (RSUs): Privacy means that an eavesdropper is impossible to decide whether two different messages come from the same vehicle.
- Communications between vehicles: Privacy means that determining whether two different valid messages coming from the same vehicle is intensely burdensome for everyone except a legitimate component (e.g., tracing manager [70]).

Monitoring Attacks
Social Attacks
Timing Attacks
Application Attacks
Network Attacks

Table B.2: Attack classification

B.3 Class of attacks

Tab. B.2 depicts a classification of attacks. In first class, Network Attacks, attackers can directly affect other vehicles and infrastructure. These attacks are on the high level of danger because these affect the whole network.

Whilst, in Application Attacks class, the objectives of attackers are applications that provide added service in VANETs. The attacker is mainly interested in changing contents used in applications and abusing it for their own benefits.

The third class, Timing Attacks, is a type of attacks in which attackers'main objective is to add some time slot in original message, for example, to create delays in order to block

this message come to the receiver before the expiration of its lifetime. All unmoral messages, which trigger bad emotions of other drivers, are classified into the class Social Attacks.

Finally, attacks in which monitoring and tracking activities are performed are laying in the class Monitoring Attacks.

Publications

[1] Vinh Hoa La and Ana R. Cavalli. Security Attacks and Solutions in Vehicular Ad Hoc Networks: A Survey. In *International Journal on AdHoc Networking Systems (IJANS)*, pages 1–20, 2014 (Published).

[2] Vinh Hoa La and Ana R. Cavalli. A study of Intrusion-tolerant routing in Wireless Sensor Networks. In *Proceedings of the Institute for System Programming of RAS, volume 26, 2014. Issue 6. ISSN 2220-6426 (Online), ISSN 2079-8156 (Print)*, pages 99–110, 2014 (Published).

[3] Vinh Hoa La and Ana R. Cavalli. A Comparative Evaluation of Two Intrusion-Tolerant Routing Protocols for Wireless Sensor Networks. In *10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA 2015)*, pages 6–12, Nov 2015 (Published).

[4] Vinh Hoa La, Raul Fuentes, and Ana R. Cavalli. Network monitoring using mmt: An application based on the user-agent field in http headers. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 147–154, March 2016 (Published).

[5] Vinh Hoa La and Ana R. Cavalli. A misbehavior node detection algorithm for 6LoWPAN Wireless Sensor Networks. In *Proceedings of 36th IEEE International Conference on Distributed Computing Systems (ICDCS 2016), Second IEEE International Workshop on Security Testing and Monitoring (STAM 2016)*, 2016 (Published).

[6] Vinh Hoa La, Raul Fuentes, and Ana R. Cavalli. A Novel Monitoring Solution for 6LoWPAN-based Wireless Sensor Networks. In *2016 IEEE 22nd Asia-Pacific Conference on Communications (APCC)*, August 2016 (Published).

[7] Vinh Hoa La and Ana R. Cavalli. Information-Theoretic Approach for Anomaly Detection: A Case Study in RPL-based Internet of Things. In *EAI Endorsed Transactions on Security and Safety*, 2016 (Under review).

[8] Vinh Hoa La and Ana R. Cavalli. Novel Learning Techniques for Anomaly Investigation and Detection in Large Scale. In *Journal of Computers*, 2016 (Under review).

[9] Raul Fuentes, Vinh Hoa La, and Ana R. Cavalli. Secure M2M Communication Monitoring in 6LoWPAN-based Wireless Sensor Networks. In *Wireless Networks: the journal of mobile communication, computation and information*, 2016 (Work in process).

Book Chapter

[1] Vinh Hoa La and Ana R. Cavalli. Intrusion detection and intrusion tolerance for 6LoWPAN-based Wireless Sensor Networks. In Georgios Kambourakis, Asaf Shabtai, Constantinos Kolias, and Dimitrios Damopoulos, editors, *Intrusion Detection and Prevention for Mobile Ecosystems*. CRC Series in Security, Privacy and Trust - Taylor & Francis, 2016 (Accepted).

Presentations

[1] Vinh Hoa La and Ana R. Cavalli. Intrusion-tolerant Routing in Wireless Sensor Networks. In *Journée sur la sécurité des SCADA et des infrastructures critiques, Paris, France*, October 2014.

[2] Vinh Hoa La and Ana R. Cavalli. Intrusion-tolerant Routing in Wireless Sensor Networks. In *French- Russian seminar on Software Verification, Testing and Quality Estimation, Paris, France*, November 2014.

[3] Vinh Hoa La and Ana R. Cavalli. Network Security Testing using MMT: A case study in IDOLE project. In *TAROT Summer School 2015, Cadiz, Spain*, July 2015.

[4] Vinh Hoa La, Raul Fuentes, and Ana R. Cavalli. Network Monitoring using MMT: An application based on the User-Agent field in HTTP headers. In *Journées non thématiques RESCOM, Lille, France*, January 2016.

[5] Vinh Hoa La and Ana R. Cavalli. Security Monitoring for Network Protocols and Applications. In *TAROT Summer School 2016, Paris, France*, July 2016.

Bibliography

[1] B. Wehbi, E. Montes de Oca, and M. Bourdelles, "Events-Based Security Monitoring Using MMT Tool," in *IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), 2012*, April 2012, pp. 860–863.

[2] J. Deng, R. Han, and S. Mishra, "INSENS: Intrusion-tolerant Routing for Wireless Sensor Networks," *Computer Communications*, vol. 29, no. 2, pp. 216–230, 2006.

[3] S. Vladimir and S. Lorenzo, "A survey on online monitoring approaches of computer-based systems," in *Centre for Software Reliability, City University London*, 2009.

[4] Cisco and its affiliates, "SNORT official homepage," https://www.snort.org/, last checked: 15.07.2016.

[5] V. Paxson, "Bro: A system for detecting network intruders in real-time," in *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, ser. SSYM'98. Berkeley, CA, USA: USENIX Association, 1998, pp. 3–3. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267549.1267552

[6] T. B. Project, "Bro official homepage," https://www.bro.org/, last checked: 15.07.2016.

[7] S. homepage, "Open Source Network-based Intrusion Detection System," http://www.openinfosecfoundation.org, last checked: 15.07.2016.

[8] E. Albin and N. C. Rowe, "A realistic experimental comparison of the Suricata and Snort intrusion-detection systems," *Proceedings of 26th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2012*, pp. 122–127, 2012.

[9] P. homepage, "Open Source Network-based Intrusion Detection System," http://pytbull.sourceforge.net/, last checked: 15.07.2016.

[10] K. Thongkanchorn, S. Ngamsuriyaroj, and V. Visoottiviseth, "Evaluation studies of three intrusion detection systems under various attacks and rule sets," *Proceedings of 10th IEEE Region Annual International Conference TENCON*, vol. 6, pp. 6–9, 2013.

[11] F. Wang, R. Uppalli, and C. Killian, "Analysis of techniques for building intrusion tolerant server systems," in *Military Communications Conference, 2003. MILCOM '03. 2003 IEEE*, vol. 2, Oct 2003, pp. 729–734 Vol.2.

[12] Q. Nguyen and A. Sood, "A Comparison of Intrusion-Tolerant System Architectures," *IEEE Security Privacy*, vol. 9, no. 4, pp. 24–31, July 2011.

[13] Montimage, "MMT-Security: User Guide. https://github.com/Montimage/MMT_Security/blob/master/MMT_Security_User_Guide.pdf, 2013," https://github.com/Montimage/MMT_Security/blob/master/MMT_Security_User_Guide.pdf, 2013, last checked: 05.09.2015.

[14] J. Pokhrel, B. Wehbi, A. Morais, A. Cavalli, and E. Allilaire, "Estimation of QoE of video traffic using a fuzzy expert system," in *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, Jan 2013, pp. 224–229.

[15] Bricks, "Advanced SQL Injection on user agent," http://sechow.com/bricks/docs/content-page-4.html, 2013, last checked: 05.09.2015.

[16] D. Manners, *The User Agent Field: Analyzing and Detecting the Abnormal or Malicious in your Organization. 2012*, SANS Institute, 2012.

[17] MustLive, "XSS attacks via user-agent header," http://websecurity.com.ua/5195/, 2011, last checked: 05.09.2015.

[18] O. Foundation, "Top 10 web vulnerabilities in 2013," https://www.owasp.org/index.php/Top_10_2013-Top_10, 2013, last checked: 05.09.2015.

[19] Craig, "Reverse Engineering a D-Link Backdoor," http://www.devttys0.com/2013/10/reverse-engineering-a-d-link-backdoor/, 2013, last checked: 05.09.2015.

[20] J. Nieto, "Pytbull-IDS/IPS Testing Framework," http://www.behindthefirewalls.com/2013/11/the-importance-of-user-agent-in-botnets.html, last checked: 15.07.2016.

[21] Y. Aboukir, "SQL Injection through HTTP Headers," *Application Security*, 2012, last checked: 05.09.2015.

[22] C. Sanders, "Using application layer metadata for network security monitoring," http://chrissanders.org/2011/09/using-application-layer-metadata-for-network-security-monitoring/, 2011, last checked: 05.09.2015.

[23] V. Labs, "Analyzing User-Agent Strings For Use In Real-World Detection Scenarios," https://labs.snort.org/papers/ua-analysis.html, 2013, last checked: 05.09.2015.

[24] Psychedelix, "List of User-Agents (Spiders, Robots, Crawler, Browser)," http://www.user-agents.org/, last checked: 05.09.2015.

[25] E. Albin and N. Rowe, "A Realistic Experimental Comparison of the Suricata and Snort Intrusion-Detection Systems," in *26th International Conference on Advanced Information Networking and Applications Workshops, 2012*, 2012, pp. 122–127.

[26] K. Thongkanchorn, S. Ngamsuriyaroj, and V. Visoottiviseth, "Evaluation studies of three intrusion detection systems under various attacks and rule sets," in *TENCON 2013 - 2013 IEEE Region 10 Conference (31194)*, Oct 2013, pp. 1–4.

[27] T. F. J. L. Jeanna Matthews, Joshua White, "Quantitative Analysis of Intrusion Detection Systems: Snort and SuricataClarkson University. 2011," http://web2.clarkson.edu/class/cs644/ids/#content, last checked: 05.09.2015.

[28] J. Macaulay, L. Buckalew, and G. Chung, "Internet of Things in Logistics," *DHL Trend Research*, vol. 1, no. 1, pp. 1–27, 2015.

[29] L. C. D. S.L., "Waspmote Datasheet," Product description, 2014.

[30] O. Hersent, D. Boswarthick, and O. Elloumi, *The internet of things : key applications and protocols*. Chichester, West Sussex: Wiley, 2012. [Online]. Available: http://opac.inria.fr/record=b1133974

[31] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1389–1406, 2013.

[32] X. Ma and W. Luo, "The analysis of 6lowpan technology," in *Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on*, vol. 1, Dec 2008, pp. 963–966.

[33] J. P. Vasseur, N. Agarwal, J. Hui, Z. Shelby, P. Bertrand, and C. Chauvenet, "RPL: The IP routing protocol designed for low power and lossy networks," *In Internet Protocol for Smart Objects (IPSO) Alliance*, no. April, p. 20, 2011.

[34] D. Karaman, N. Gozuacik, M. O. Alagoz, H. Ilhan, U. Cagal, and O. Yavuz, "Managing 6lowpan sensors with coap on internet," in *Signal Processing and Communications Applications Conference (SIU), 2015 23th*, May 2015, pp. 1389–1392.

[35] S. Raza, D. Trabalza, and T. Voigt, "6lowpan compressed dtls for coap," in *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, May 2012, pp. 287–289.

[36] V. H. La, R. Fuentes, and A. R. Cavalli, "Network Monitoring Using MMT: An Application Based on the User-Agent Field in HTTP Headers," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, March 2016, pp. 147–154.

[37] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661–2674, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2013.04.014

[38] A. H. Farooqi and F. A. Khan, "Intrusion Detection Systems for Wireless Sensor Networks: A Survey," *Communications in Computer and Information Science*, no. 56, pp. 234–241, 2009.

[39] P. Kasinathan, C. Pastrone, M. a. Spirito, and M. Vinkovits, "Denial-of-Service detection in 6LoWPAN based Internet of Things," *International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 600–607, 2013.

[40] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. a. Spirito, "DEMO: An IDS framework for internet of things empowered by 6LoWPAN," *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, no. October 2015, pp. 1337–1340, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2508859.2512494

[41] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[42] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, Jan. 2001. [Online]. Available: http://doi.acm.org/10.1145/584091.584093

[43] F. I. Lab, "IoT experimentation at a large scale," https://www.iot-lab.info/, 2015, last checked: 28.01.2016.

[44] V. H. La and A. R. Cavalli, "A misbehavior node detection algorithm for 6LoWPAN Wireless Sensor Networks," in *Proceedings of 36th IEEE International Conference on Distributed Computing Systems (ICDCS 2016), Second IEEE International Workshop on Security Testing and Monitoring (STAM 2016)*, 2016.

[45] D.-I. Curiac, O. Banias, F. Dragan, C. Volosencu, and O. Dranga, "Malicious Node Detection in Wireless Sensor Networks Using an Autoregression Technique," *International Conference on Networking and Services (ICNS '07)*, pp. 83–83, 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4438332

[46] I. M. Atakli, H. Hu, Y. Chen, W.-S. Ku, and Z. Su, "Malicious Node Detection in Wireless Sensor Networks using Weighted Trust Evaluation," *Proceedings of the 2008 Spring simulation multiconference*, pp. 836–843, 2008.

[47] S. H. Oh, C. O. Hong, and Y.-h. Choi, "A Malicious and Malfunctioning Node Detection Scheme for Wireless Sensor Networks," vol. 2012, no. March, pp. 84–90, 2012.

[48] I. Grand, E. Nancy, and T. Nancy, "A Taxonomy of Attacks in RPL-based Internet of Things," vol. 18, no. 3, pp. 459–473, 2016.

[49] Raytheon, "Intrusion-Tolerant Systems," *Technology Today Journal*, 2007.

[50] D. P. Yves Deswarte, "Intrusion-tolerance on the Internet (tolerance aux intrusions sur internet)," Presentation at LAAS-CNRS, Toulouse, France, 2005.

[51] B. B. Madan and K. S. Trivedi, "Security Modeling and Quantification of Intrusion Tolerant Systems Using Attack-response Graph," *High Speed Networks*, vol. 13, no. 4, pp. 297–308, Oct. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1085412.1085416

[52] F. Wang, R. Uppalli, and C. Killian, "Analysis of techniques for building intrusion tolerant server systems," in *Military Communications Conference, 2003. MILCOM '03. 2003 IEEE*, vol. 2, Oct 2003, pp. 729–734 Vol.2.

[53] W. Liang-Min, J. feng Ma, C. Wang, and A. Kot, "Fault and intrusion tolerance of wireless sensor networks," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, April 2006, pp. 7 pp.–.

[54] R. Ma, L. Xing, and H. E. Michel, "Fault- Intrusion Tolerant Techniques in Wireless Sensor Networks," in *Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, ser. DASC '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 85–94. [Online]. Available: http://dx.doi.org/10.1109/DASC.2006.30

[55] R. H. J. Deng and S. Mishra, "INSENS: Secure and Intrusion Tolerant Routing for Wireless Sensor Networks," Department of Computer Science. University of Colorado, Boulder, CO, 2004, Tech. Rep., 2004.

[56] J. Zhou, C. Li, Q. Cao, and Y. Shen, "An intrusion-tolerant secure routing protocol with key exchange for wireless sensor network," in *Information and Automation, 2008. ICIA 2008. International Conference on*, June 2008, pp. 1547–1552.

[57] J. Deng, R. Han, and S. Mishra, "A performance evaluation of intrusion-tolerant routing in wireless sensor networks," in *Technical report, University of Colorado*, 2003, pp. 349–364.

[58] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, Sep. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1016598314198

[59] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[60] C. Intanagonwiwat, R. Govindan, D. Estfin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 2–16, 2003.

[61] A. Abbasi, J. Wetzels, W. Bokslag, E. Zambon, and S. Etalle, *On Emulation-Based Network Intrusion Detection Systems*. Cham: Springer International Publishing, 2014, pp. 384–404. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11379-1_19

[62] M. Polychronakis, K. G. Anagnostakis, and E. P. Markatos, *Network–Level Polymorphic Shellcode Detection Using Emulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 54–73. [Online]. Available: http://dx.doi.org/10.1007/11790754_4

[63] N. Antunes and M. Vieira, "Defending against web application vulnerabilities," *Computer*, vol. 45, no. 2, pp. 66–72, Feb 2012.

[64] J. Clarke and R. M. Alvarez, *SQL Injection Attacks and Defense*, 2009. [Online]. Available: http://www.amazon.com/Injection-Attacks-Defense-Justin-Clarke/dp/1597494240/ref=sr{_}1{_}1?s=books{&}ie=UTF8{&}qid=1309070123{&}sr=1-1

[65] H. Shahriar and M. Zulkernine, "Information-Theoretic Detection of SQL Injection Attacks," *2012 IEEE 14th International Symposium on High-Assurance Systems Engineering*, pp. 40–47, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6375635

[66] V. H. La and A. R. Cavalli, "Security Attacks and Solutions in Vehicular Ad Hoc Networks: A Survey," in *International Journal on AdHoc Networking Systems (IJANS)*, 2014, pp. 1–20.

[67] A. K. Al-kahtani, Salman bin Abdulaziz, "Survey on security attacks in Vehicular Ad hoc Networks (VANETs)," in *6th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2012, pp. 1–9.

[68] J. P. H. M. Raya, "Securing vehicular ad hoc Networks," in *Journal of Computer Security, vol.15, January 2007*, 2007, pp. 39–68.

[69] C. T. Wing, "Secure and Privacy-preserving Protocols for VANETs," in *PhD thesis at The University of Hong Kong*, August 2011.

[70] L. Zhang, "Research on Security and Privacy in Vehicular Ad Hoc Networks," in *PhD thesis at Universitat Rovira i Virgili*, June 2010.