Online Network Traffic Security Inspection Using MMT Tool

Wissam Mallouli*, Bachar Wehbi*, Edgardo Montes de Oca* and Michel Bourdellès†
*Montimage, 39 rue Bobillot, 75013 Paris - France
{wissam.mallouli, bachar.wehbi, edgardo.montesdeoca}@montimage.com
†THALES Communications, SRP Department
160, Bd de Valmy - BP 82, 92704 Colombes Cedex - France
{michel.bourdelles}@fr.thalesgroup.com

***Abstract*—MMT (Montimage[1] Monitoring Tool) is a monitoring solution that allows near real-time QoS and security analysis based on deep packet inspection techniques. For security monitoring, it relies on a formal description of conditions on sequences of events called security properties to define security rules (i.e., rules that should be respected) or attacks and misbehaviours. These security properties can also integrate: advanced analysis techniques based on statistics and machine learning, notifications, alarms (e.g., using syslog), and countermeasures (e.g., using the iptables library).**

In this paper, we give an overview of MMT's architecture showing its extensibility and adaptability to multi-domains. We also demonstrate the realibity of the tool by its application to an industrial case study provided by Thales Group dealing with a QoS-aware ad-hoc radio communication protocol.

Keywords: Events extraction, Monitoring, Security Analysis.

I. Introduction

A. Motivation and challenges

New and more critical vulnerabilities are constantly being introduced by the evolution of the Internet and mobile communications where critical infrastructures are more and more open and corporate IT is more and more dematerialised (e.g., using cloud services). This is pushing towards the need for more proactive and automated mechanisms for detecting and preventing anomalies (due to attacks or misbehaviours). In this context, Deep Packet Inspection (DPI) is considered as a key element in the shift towards advanced monitoring. DPI is the process of capturing network traffic, analysing and inspecting it in detail to determine accurately what is really happening in the network. This "core component" feeds the different monitoring applications with high added value information. Starting from this perception, the requirements of a network monitoring system can be summarized as follows:

- High capturing performance. It must be able to capture traffic at high speeds and under high traffic volume. This depends on what is to be monitored and where.
- Extensibility. If new services are integrated in the network, it must be possible to deploy effortlessly new monitoring mechanisms for these specific services. In addition, if new analysis techniques are needed, it should be possible to integrate them as effortlessly as possible.
- Scalability. It must be able to handle the increase of traffic data as network link speeds and the number of probes increase in the network without performance degradation. Scalability can be achieved by reducing the traffic information collected using efficient packet capturing mechanisms and traffic pre-processing.
- Real time functioning. It must implement real time mechanisms in order to quickly detect network security/performance problems and allow timely execution of automated or manual countermeasures.
- Granularity. It must be able to track the security and performance of each service by capturing and analysing the traffic belonging to the application of interest.
- Diversity. It has to support the network's diversity as it contains different types of network devices from multiple vendors, protocols stacks, and applications to provide services to the user.
- Low cost. It should not use excessive amount of computing, storage, and communication resources so the cost of deploying and operating the monitoring infrastructure is low for service providers.
- Secure. It should not add vulnerabilities to the network, or disturb normal network operation.

B. Paper content

In this paper, we present MMT tool an online monitoring solution that allows providing a real-time visibility of network traffic. It provides network, application, flow and user level visibility. MMT facilitates network security and performance monitoring and operation troubleshooting. MMT's rules engine can correlate network and application events in order to detect operational, security and performance incidents.

In section II, we will describe the security properties formalism used to specify the security requirements of the system/network under test. Then, we will present MMT tool architecture and security features in section III and show how they can answer the security monitoring challenges described above. The application of MMT to an industrial case study provided by Thales Group is given is section IV. Finally, we conclude the paper and provide some future directions in

[1]Montimage is an innovative company created in 2004 and located in Paris. It is specialized in software development and testing services.

section V.

II. MMT-Security properties formalism

A. Formalism description

The main objective of MMT security properties is to formally specify security goals and attack behaviors related to the application or protocol under test. The "MMT-Security property" model is inspired from LTL logic [3] and can refer to two types of properties: "Security rules" and "Attacks" described as follows:

- A Security rule describes the expected behavior of the application or protocol under-test whether it is functional or security oriented. The non-respect of the MMT-Security property indicates an abnormal behavior, e.g. the access to a specific service must always be preceded by an authentication phase.
- An Attack describes a malicious behavior whether it is an attack model, a vulnerability or a misbehavior. Here, the respect of the MMT-Security property indicates the detection of an abnormal behavior that might indicate the occurrence of an attack, e.g. a big number of requests from the same user in a limited period of time can be considered as a behavioral attack.

It must be noted that the events that we take into account within MMT-Security properties are related to observable system/network communications. In the case of a telecommunication network, they refer to traffic packets and flows. In other contexts, they can relate to any action that can be stored in a server/database/software log file. In the following, we formally present the concepts of MMT-Security properties in the context of telecommunication networks. The main definition of an MMT-Security property is provided by definition number 10. The other definitions allow understanding the basics of the used model. In the rest of this document the terms: "packet", "message" and "event" are used interchangeably.

Definition 1: (Live Trace) A collected live trace during a period of time is a set of ordered captured packets.

- *A live trace* $T = \bigcup_{i=1}^{n} p_i$ *where n is the number of the captured packets,* p_1 *is the first packet captured in the trace and* p_n *the last one.*
- *Each packet* p_i *has a rank* r_i *that corresponds to its position in the trace* T.
- $\forall p_i \in T$, $p_i = \bigcup_{j=1}^{m_i} f_{i,j}$ *where* $f_{i,j}$ *is a field of the packet* p_i *and* m_i *is the number of fields of the packet* p_i. *Each field* $f_{i,j}$ *of the packet* p_i *has a value* $v_{i,j}$.
- $\forall p_i \in T$, $\exists f_{i,j} \in p_i$ / $f_{i,j} = t_i$ *where* t_i *is the timestamp when* p_i *was captured.*
- $\forall r_i$, r_j *where* r_i *is rank of* p_i *and* r_j *is rank of* p_j, *if* $r_i > r_j$ *then* $t_i > t_j$

Definition 2: (Value function ϕ*) Let* T *be a collected trace of n packets,* F *the set of fields of all the packets* p_i *of the trace* T, V *the domain of values and P the set of packets.* $V = \mathbb{R} \cup S \cup NULL$ *where S is a finite set of strings. We define the function:* ϕ*:* $P \times F \rightarrow V$ *as the function allowing to provide the value of a field in a specific packet of the trace* T*:*

- $\phi(p_i, f_{m,n}) = v_{i,n}$ *if* $f_{m,n} \in p_i$ *and*
- $\phi(p_i, f_{m,n}) = NULL$ *if* $f_{m,n} \notin p_i$

An MMT-Security property is an IF-THEN property. It allows expressing specific constraints on network events. Each event is a set of conditions on some of the field values of the exchanged packets.

Definition 3: (Conditions) Conditions are predicates on packets' fields values. Let p_i *and* $p_{i'}$ *be two captured packets,* V *be the domain of values,* $f_{i,j}$ *be a field of the packet* p_i, $f_{i',j'}$ *be a field of* $p_{i'}$ *and* $x \in V$. *Let op be an operator element of* $O_R \cup O_S$ *where* O_R*={*$\leq, \geq, =, \neq, \in$ *etc.}* *and* O_S*={equal, not equal, contain, not contain etc.}*[2]. *Two types of conditions can be defined:* c_s *(simple condition) and* c_c *(complex condition):*

- $c_s ::= \phi(p_i, f_{i,j})$ *op x. We say that the packet* p_i *satisfies* c_s *iff* $v_{i,j}$ *op x is true.*
- $c_c ::= \phi(p_i, f_{i,j})$ *op* $\phi(p_{i'}, f_{i',j'})$. *We say that packet* p_i *satisfies* c_c *iff* $v_{i,j}$ *op* $v_{i',j'}$ *is true.*

Definition 4: (Basic event) An event e_j *is a set of conditions on relevant fields of captured packets.* $e_j = \bigcup_{k=1}^{m_j} c_{j,k}$, m_j *being the number of conditions (simple and/or complex).Let* p_i *be a packet and* e_j *an event with* m_j *conditions and* $c_{j,k}$ *the* k^{th} *condition of* e_j. *A packet* p_i *satisfies an event* e_j *if and only if* $\forall\ k \in [1, m_j]$, $c_{j,k}$ *is true.*

Definition 5: (Abstention of having an event) If e is an event, then $\neg e$ *is also an event.* $\neg e$ *is satisfied if no packet that satisfies the event e occurs in the collected trace.*

Definition 6: (Repetition an event) If e is an event and $n \in N^*$, *then* $n \times e$ *is a complex event.* $n \times e$ *is satisfied if n packets satisfying the event e occur in the collected trace.*

Definition 7: (Complex events: Successive events) Let $n \in \mathbb{N}^*$, $t \in \mathbb{R}^{+*}$ *and* e_1 *and* e_2 *be two basic events.* $(e_1; e_2)_{n,t}$ *is a complex event. It is composed of two basic events.* $[p_1, p_2]$ *satisfies* $(e_1; e_2)_{n,t} \quad \Leftrightarrow$

- p_1 *satisfies* e_1 *and*
- p_2 *satisfies* e_2 *and*
- *time*$(p_1) <$ *time*$(p_2) <$ *time*$(p_1)+t$ *and*
- *rank*$(p_1) <$ *rank*$(p_2) <$ *rank*$(p_1)+n$.

In other words, $[p_1, p_2]$ satisfies $(e_1; e_2)_{n,t}$ iff p_2 follows p_1 and they are separated by at most n packets and t units of time.

Definition 8: (complex events: AND) Let $n \in \mathbb{N}^*$, $t \in \mathbb{R}^{+*}$ *and* e_1 *and* e_2 *two basic events.* $(e_1 \wedge e_2)_{n,t}$ *is a complex event.*

[2] O_R is the classical set of operators that can be applied on real numbers in the domain $\mathbb{R}$. O_S is the classical set of operators that can be applied on strings of the domain S.

It is composed of two basic events. $[p_1, p_2]$ *satisfies* $(e_1 \wedge e_2)_{n,t}$ $\Leftrightarrow$ $[p_1, p_2]$ *satisfies* $(e_1; e_2)_{n,t}$ *or* $[p_1, p_2]$ *satisfies* $(e_2; e_1)_{n,t}$.

Intuitively, p_1 and p_2 satisfy $(e_1 \wedge e_2)_{n,t}$ iff p_2 and p_1 are separated by at most n packets and t units of time.

Definition 9: (complex events: OR) Let e_1 *and* e_2 *two basic events.* $(e_1 \vee e_2)$ *is a complex event.*p_1 *satisfies* $(e_1 \vee e_2) \Leftrightarrow p_1$ *satisfies* e_1 *or* p_1 *satisfies* e_2.

Definition 10: (MMT-Security property) Let $W \in \{BEFORE, AFTER\}$, $n \in \mathbb{N}$, $t \in \mathbb{R}^{+*}$ *and* e_1 *and* e_2 *two events (basic or not). An MMT-Security property is an IF-THEN expression that describes constraints on network events captured in a trace* $T = \{p_1, ..., p_{m'}\}$. *It has the following syntax:*

$$e_1 \xrightarrow{W,n,t} e_2$$

This property expresses that if the event e_1 *is satisfied (by one or several packets* p_i, $i \in \{1, ..., m\}$*), then event* e_2 *must be satisfied (by another set of packets* p_j, $j \in \{1, ..., m\}$*) before or after (depending on the W value) at most n packets and t units of time.* e_1 *is called triggering context and* e_2 *is called clause verdict.*

B. Formalism implementation

The MMT-Security property model allows expressing complex security properties derived from security best practices and from domain-specific security requirements. These MMT-Security properties are described using an XML format to make interpretation easier for both humans and software.

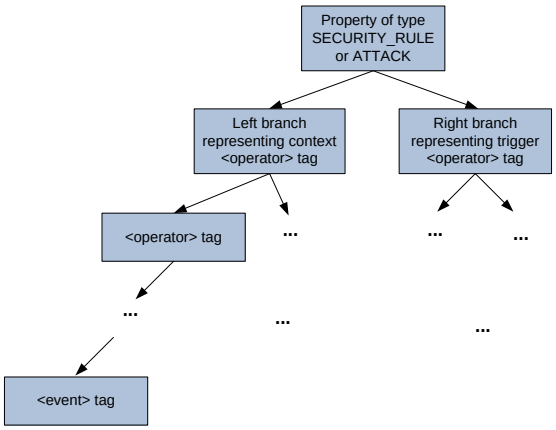

Figure 1. MMT property structure.

Each property begins with a <*property*> tag and ends with <*/property*>. A property is a "general ordered tree" as shown in figure 1. The nodes of the property tree are: the property node (required), operator nodes (optional) and event nodes (required). The property node is forcibly the root node and the event nodes are forcibly leaf nodes. The left branch represents the context and the right branch represents the trigger. This means that the property is found valid when the trigger is found valid; and the trigger is checked only if the context is valid. In other words:

- If the context is verified and the trigger is not, then a property non-respect occurrence is detected.
  - In the case of a "security rule", this means that the context of the rule has been found and, since the trigger was not, we conclude that the "security rule" has been violated.
  - In the case of an "attack", this means that the context of an attack has occurred but the trace was attack free.
- If the context and the trigger are verified, then a property respect occurrence is detected.
  - In the case of a "security rule", this means that the context of the rule has been found, as well as the trigger. We conclude that the "security rule" has been respected.
  - In the case of an "attack", this means that the context of an attack has occurred, as well as the trigger. We conclude that the attack has been detected.

Table I illustrates a security property derived from the case study described in section IV.

C. Multi data sources management for security analysis

In the context of MMT, DPI (Deep Packet Inspection) and DFI (Deep Flow Inspection) are used to help detect and tackle harmful traffic and security threats; and, to throttle or block undesired behaviours. We define a set of security properties for network traffic, at both control and data levels, to detect interesting events. Indeed, based on the defined security properties, we register the attributes to be extracted from the inspected packets and flows. These attributes are of three types:

- **Real attributes:** They can be directly extracted from the inspected packet. They correspond to a protocol field value.
- **Calculated attributes:** They are calculated within a flow. Packets from the same flow are grouped and security/performance indicators are calculated (e.g. delays, jitter, packet loss rate) and made available for the security analysis engine.
- **Meta attributes:** These attributes are linked to each packet to describe capture information. The time of capture of each packet (timestamp attribute) is the main meta attribute in the current version of MMT.

The extracted attributes needed for security analysis can emanate from different data sources (probes and/or interfaces). This is managed in the MMT monitoring solution during the specification phase of the security properties. Indeed, the data sources identifiers are part of the meta-attributes that can be used in the specification of the relevant events for security analysis. Three architectures are taken into account in MMT:

- **Local analysis:** the collected traffic is analysed for security purposes in one probe that captures network traffic from one or several interfaces.
- **Centralized analysis:** the traffic capture is distributed but the security analysis is centralized. All data sources send

Table I
EXAMPLE OF MMT-SECURITY PROPERTY

XML code	Explanation
<property value="THEN" property_id="1" type_property = "SECURITY_RULE" description="If one node receives two successive MSG_SPHY_DATA_IND messages from the same source, then these two messages must to be separeted by 50 slots">	Security rule (that should be respected) with id=1 of the form: *if the context holds then the trigger should have occurred.*
<operator value="THEN" delay_min="0+" delay_max="99">	Two successive events that occured within a delay in]0,99].
<event value="COMPUTE" event_id="1" description = "MSG_SPHY_DATA_IND message" boolean_expression = "((THALES_META.MSG_CODE == 8193) && (MSG_SPHY_DATA_IND.SLOT_TYPE == 0))" />	An event with id=1 that satifies the boolean expression. It is an SPHY_DATA_IND message identified by the message code = 8193 and a slot type = 0 (SCH broadcast channel type).
<event value="COMPUTE" event_id="2" description = "MSG_SPHY_DATA_IND message" boolean_expression = "((THALES_META.MSG_CODE == 8193) && ((MSG_SPHY_DATA_IND.SLOT_TYPE == 0) && ((MSG_SPHY_DATA_IND.ADDRESS_SOURCE == MSG_SPHY_DATA_IND.ADDRESS_SOURCE.1) && (THALES_META.NODE_ID == THALES_META.NODE_ID.1))))"/>	An event with id=2 that satifies the boolean expression. It is an SPHY_DATA_IND message identified by the message code = 8193 and a slot type = 0 (SCH broadcast channel type). It has the same node source address as event 1 and is received by the same node.
</operator>	End of the operation.
<event value="COMPUTE" event_id="3" description = "MSG_SPHY_DATA_IND messages must to be separeted by 50 slots" boolean_expression = "((THALES_META.TIME_SLOT.1 + 50) == THALES_META.TIME_SLOT.2)"/>	An event with id=3 that checks that the time slot between the two first events are seperated by 50.
</property>	End of property

their collected traffic (filtered or not) to the same master server that correlates the traces (i.e., need to synchronize probes to be able to perform this task).

- **Distributed analysis:** the traffic capture is distributed and the analysis is performed by all the probes that communicate together to share information. This analysis can be very interesting in some specific case studies like ad hoc networks. The communication between probes is an on-going work for MMT tool.

The originality of the MMT security properties with respect to existing intrusion detection techniques lies in that they are not based on just pattern matching (i.e., signatures) as in SNORT [5] nor requiring writing executable scripts as in BRO [4]. They allow a more abstract description of sequence of events that can represent normal/abnormal behaviour. They can also integrate pattern matching, statistics and machine learning techniques; but describing this here is out of scope for this paper.

III. MONTIMAGE MONITORING TOOL

A. MMT-Security architecture

MMT-Security is composed of three complementary, but independent, modules:

- **MMT-Extract** is the core packet processing module. It is a C library that analyses network traffic using Deep Packet/Flow Inspection (DPI/DFI) techniques in order to identify network and application based events by analyzing: protocols' fields values; network and application Quality of Service (QoS) parameters; and, Key Performance Indicators (KPI). In a similar way, it also allows analyzing any structured information generated by applications (e.g., traces, logged messages). MMT-Extract incorporates a plugin architecture for the addition of new protocols or messages, and a public API for integration into third party probes.
- **MMT-Security** is a security analysis engine based on MMT-Security properties. MMT-Security analyzes and correlates network and application events to detect operational and security incidents. For each occurrence of a security property, MMT-Security allows detecting whether it was respected or violated.
- **MMT-Operator** is a visualization application for MMT-Security currently under development. It allows collecting and aggregating security incidents to present them via a graphical user interface. MMT-Operator is conceived to be customizable, i.e., the user will be able to define new views or customize one from a large list of predefined views. With its generic connector, MMT-Operator can be integrated with third party traffic probes. At the time of writing this paper, a web based representation of the analysis results is provided.

B. MMT-Security features and innovation

Granular traffic analysis capabilities: MMT allows parsing a wide range of protocol packet types (e.g., TCP, UDP, ARP, HTTP, etc.) and extracting various performance indicators. The extraction is powered by a plugin architecture that allows adding the analysis of new protocol packet formats or even structured application generated messages (e.g., traces, logs).
Application classification: Prior to extracting protocol packet attributes, MMT uses DPI techniques for application identification and classification. This is essential when analyzing applications that use non-standard port numbers (e.g., P2P, Skype). To be able to classify encrypted packets such as Skype, both signature detection and flow state are used [2].
Properties engine: Allows the detection of complex sequence of events that conventional monitoring does not detect (see

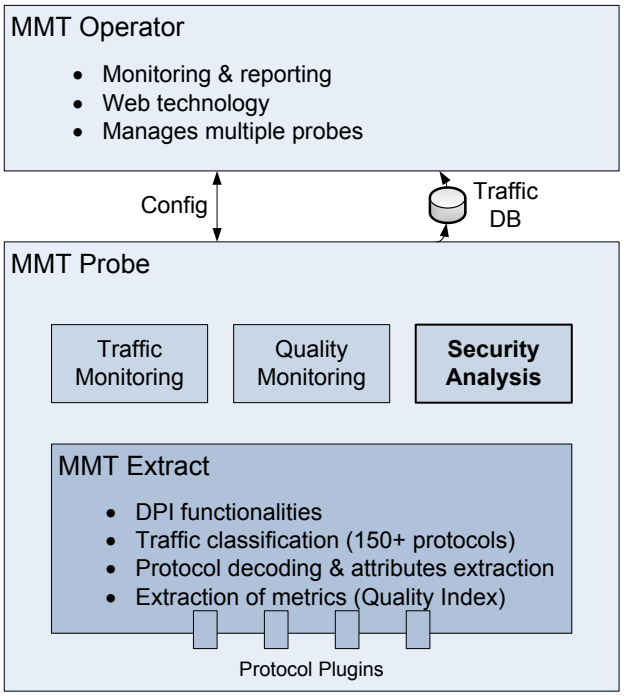

Figure 2. MMT global architecture.

description in section II). This is used to monitor: i) access control policies (e.g., verify that authorized users are authenticated prior to using a critical business application); ii) anomaly or attacks (e.g., detect excessive login attempts on the application server); iii) performance (e.g., identify VoIP calls with QoS parameters exceeding acceptable quality thresholds); etc.

Configurable reports: MMT traffic reports and charts can be configured by the user. The user can edit pre-configured reports and create new ones. Different chart types and graphs can be used including: pie, histograms, time charts, stacked area charts, sequence charts, tables, hierarchical tables, etc.

Multi-platform solution: MMT is available on Windows and Linux based distributions. It can be installed as software on commodity hardware or optimized for integration with dedicated probes.

Modular solution: MMT is a modular solution composed of three components: MMT-Extract for the traffic processing and data decoding; MMT-Security for properties analysis; and, MMT-Operator for data aggregation, correlation and reporting. It is possible to integrate MMT-Extract and MMT-Security in third party traffic probes and to connect MMT-Operator with existing systems.

The novelty in the approach used by MMT is that it allows:

- detecting both wanted (e.g., security rules) and unwanted (e.g., attacks) behaviour;
- using performance indicators, e.g., to detect bottlenecks caused by attacks;
- defining countermeasures, e.g., change the iptable;
- combining active and passive approaches (e.g., can be used to verify that generated tests passed or failed);
- analysing any structured information (e.g. network packets, messages, application logs); and,
- combining centralized and distributed analysis to detect 0-day attacks using machine learning techniques (work in progress).

Furthermore, MMT uses an algorithm that only stores the information needed to verify the properties and does not do any backtracking to verify, for instance, that an event happened before.

MMT allows defining properties with a high degree of expressiveness that include conditions on time, order of events, packet parameters, payload information, KPI, statistical and machine learning analysis. The expressiveness of the properties allows detecting complex events. This makes MMT a very flexible tool that can be applied in several domains. It also makes it possible to deploy probes that will work together to obtain a more complete view of the network. This work in progress can also be used as a Complex Event Processing engine for Business Activity Monitoring.

MMT can be installed as (1) a standalone tool that allows the analysis of live or pre-recorded traces or as (2) a set of two libraries for integration into third party probes.

IV. Case Study

A. Experiment description

MMT tool has been used to analyze an industrial case study provided by Thales Group. This case study is based on an ad-hoc radio network. The security issues that were explored focused on intrusion capabilities from over-the-air threats and also on radio network disruptions due to incorrect radio unit behaviour. Since these disruptions could affect routing capabilities of a network of several dozens of radio units, the verification of these security issues has been done using a simulation platform based on OMNET++ [6]. OMNET++ acted as a client and MMT as a server that received all the exchanged data between ad-hoc network nodes to analyse them.

A set of 25 security properties derived from identified radio protocol requirements were formally specified. They were then checked using the MMT-Security tool on a set of collected traces delivered by Thales and generated using the CertifyIt Smartesting tool [1]. This tool allows performing active testing based on test purposes.

The security properties specified and validated are based on the neighbourhood management of a radio nodes. Each node manages the resource allocation of the 1 and 2 hops radio nodes it can communicate with. The neighbour node detection and release is processed by exchange of specific PDU information exchange between these nodes. As all the network topology is built from these peer nodes dynamic detection, this protocol part is very vulnerable to attacks and misfunction of these PDU message exchange and processing.

B. Preliminary analysis results

The specified security properties allowed to detect the different attack scenarios. At least one specified security property

Table II
SUMMARY OF RESULTS ACCORDING THE SPECIFIED ATTACK SCENARIO 1

id	Property description	Respected	Violated
1	Security rule: If one node receives two successive MSG_SPHY_DATA_IND messages from the same source, then these two messages must to be separeted by 50 slots	12	2
2	Security rule: If one node receives two MSG_SPHY_DATA_IND messages from different sources, then these two messages must have two differents slot ids	15	0
3	SECURITY RULE: If node A receives from B an MSG_SPHY_DATA_IND message claiming A as a neighbor, then this means that A received from B at least 4 MSG_SPHY_DATA_IND messages in the last 5 periods (One period = 50 TS)	14	0
4	SECURITY RULE: DataUMAC within MSG_SPHY_DATA_IND (SCH) message must have a management status equal to 10 and a channel presence equal to 10 (hexa values)	30	0
5	SECURITY RULE: Number of neighbors must be between 0 and 127	30	0
6	SECURITY RULE: DataUMAC within MSG_SPHY_DATA_IND message should have K, J, C bytes (broadcast channel) as follows: K between 1 and 255, J between 3 and 11 and C between 0 and 7	30	0
7	SECURITY RULE: DataUMAC within MSG_SPHY_DATA_IND message is well formatted (can replace rules 4, 5, 6 and more !). All neighbors should respect the broadcast channels limitations defined in rule 6	19	11
8	SECURITY RULE: The declared neighbors of a node are distinct	30	0
9	SECURITY RULE: The neighbors declared by a node A do not contain the source node A	30	0
10	SECURITY RULE: The bit Z1 in KJC must be equal to 0 (Tolerance X% = 0)	30	0
12	SECURITY RULE: The directivity byte in BLOC3 (if any) must be equal to 0 or 1)	30	0
13	SECURITY RULE: KJCs in BLOC2 (if any) must be different from KJC in BLOC 1)	30	0
14	SECURITY RULE: All channels in BLOC2 are distinct	30	0
15	SECURITY RULE:If node A receives from B an MSG_SPHY_DATA_IND message claiming A as a neighbor with a bidirectivity bit = 1, then this means that A received from B at least 4 MSG_SPHY_DATA_IND messages in the last 5 periods (One period = 50 TS)	16	0
..	...	..	..
..	...	..	..
25	Security rule: If a node receives two SPHY_DATA_IND messages from two different nodes, the two messages need to have different broadcast Channels	15	0

was violeted during an attack. This demonstrates the efficiency of MMT and its relevance in detecting intrusions in ad hoc networks. In the following paragraph, we present an attack example simulated by OMNET+ and the results provided by MMT.

Security requirement: Every node must periodically send a notification message that includes the list of its neighbors on its allocated service slot. In order to verify this requirement, we should check that two consecutive notifications from the same source are seperated by a specified number of slots (50 in our example) and that two notifications from different sources have different slots ids.

Attack scenario 1: A malicious node sends a message on a non allocated service slot.

Specified security properties: Two security rules to detect this attack are specified (c.f. properties 1 and 2 in table II). The first property is described in details in the example of table I.

Table II describes an example of results provided by MMT tool during an online analysis. The next step (ongoing work) is to allow collaboration between network node to identify intruder(s) and perform a suitable countermesearument according a some pre-defined strategies. This is only possible if communication between nodes (and probes) is possible.

V. CONCLUSION

The MMT monitoring tool allows bringing near real-time visibility and operational intelligence into system communications so that the quality and reliability can be studied and verified. By developing this tool, Montimage targets different technologies (wireless, mobile, web) and industrial domains (telecom, banking) and aims to raise different monitoring challenges presented in this paper.

Unlike active testing, passive monitoring does not inject traffic in the network, nor modify the traffic that is being transmitted in the network. Nevertheless, active testing, combined with passive testing, can be very useful to stimulate critical systems and detect vulnerabilities and security flaws early in the development process. The results obtained show that passive monitoring for detecting security flaws can help improve the reliability of the resulting products.

ACKNOWLEDGEMENT

The research leading to these results has received partial funding from the French ANR PIMI Project and the European ITEA2 Diamonds project.

REFERENCES

[1] http://www.smartesting.com/index.php/cms/en/product/certify-it.

[2] Kimberly C. Claffy Alberto Dainotti, Antonio Pescapè. Issues and future directions in traffic classification. *IEEE Network (NETWORK)*, 26(1):35–40, 2012.

[3] Alessandro Armando, Roberto Carbone, and Luca Compagna. LTL model checking for security protocols. *Journal of Applied Non-Classical Logics*, 19(4):403–429, 2009.

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

[5] Martin Roesch. Snort: Lightweight intrusion detection for network. *Proceedings of LISA '99: 13th Systems Administration Conference*, pages 229–238, 1999.

[6] András Varga. Omnet++. In Klaus Wehrle, Mesut Günes, and James Gross, editors, *Modeling and Tools for Network Simulation*, pages 35–59. Springer, 2010.