An effective attack detection approach in wireless mesh networks

Felipe Barbosa Abreu, Anderson Morais and Ana Cavalli

Télécom SudParis,
9 Rue Charles Fourier,
91000 Evry, France
Email: felipe_barbosah@hotmail.com
Email: anderson.morais@telecom-sudparis.eu
Email: ana.cavalli@telecom-sudparis.eu

Bachar Wehbi, Edgardo Montes de Oca and Wissam Mallouli*

Montimage,
39 rue Bobillot,
75013 Paris, France
Email: bachar.wehbi@montimage.com
Email: edgardo.montesdeoca@montimage.com
Email: wissam.mallouli@montimage.com
*Corresponding author

Abstract: Wireless mesh network (WMN) is a recent technology that is gaining significant importance among traditional wireless networks. It is considered a suitable solution for providing internet access in an inexpensive, convenient, and rapid manner. Nonetheless, WMNs are exposed to various types of security threats due to their intrinsic characteristics such as open broadcast medium and decentralised architecture. For instance, a compromised node can generate malicious traffic in order to disrupt the network routing service, putting the entire mesh network at risk. In this paper, we provide an efficient method for detecting active attacks against the routing functionality of a mesh network. The approach relies on the analysis of the protocol routing behaviour by processing the traces produced by each node using the Montimage Monitoring Tool (MMT), which outputs routing events that are correlated between nodes to detect potential intrusions. We demonstrate the approach feasibility by using a virtualised mesh network platform that consists of virtual nodes executing 'better approach to mobile ad hoc network' (BATMAN) routing protocol. The experimental results show that the proposed method accurately identifies malicious routing traffic diffused by an attacker through the network.

Keywords: wireless mesh network; WMN; routing attack; attack detection; network trace.

Reference to this paper should be made as follows: Abreu, F.B., Morais, A., Cavalli, A., Wehbi, B., Montes de Oca, E. and Mallouli, W. (xxxx) 'An effective attack detection approach in wireless mesh networks', *Int. J. Space-Based and Situated Computing*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Felipe Barbosa Abreu received his Master's degree from the University of Ceara, Brazil in 2013. He spent a training period of one year at Telecom SudParis in 2012 in the framework of the Brafitec programme. His research interest are intrusion detection, mesh network, protocol testing and fault injection.

Anderson Morais received his Master's in Computer Science from the Institute of Computing of UNICAMP (University of Campinas) in Sao Paulo, Brazil and his BSc in Computing Engineering at UNICAMP. He worked as a Software Engineer specialising in development of mobile embedded software, mobile protocols and hardware simulation for mobile devices. His main research interests are intrusion detection, mesh networks, security testing, protocol testing, fault injection and mobile services. He received his PhD in Computer Science from Telecom SudParis in 2012.

Ana Cavalli obtained her Doctorat d'Etat es Mathematics Science and Informatics from the University of Paris VII, in 1984. She is currently a Full Professor at Telecom Sudparis (ex Institut National des Telecommunications) since 1990. She is the Director of the Software for

Networks Department and also a member of the CNRS research laboratory SAMOVAR. Her research interests are on specification and verification, testing methodologies for conformance and interoperability testing, active testing and monitoring techniques, the validation of security properties and their application to services and protocols.

Bachar Wehbi is a Senior R&D Engineer at Montimage. He received his Master's degree from Paris VI University in 2005 and his PhD in Computer Science from Telecom and Management SudParis (France) in 2008. His topics of interest cover telecommunication protocols and wireless networks. He has a strong background in monitoring and testing network security based on formal methods.

Edgardo Montes de Oca graduated as an Engineer in 1985 from Paris XI University, Orsay. He has worked as a Research Engineer in the Alcatel Corporate Research centre in Marcoussis, France and in Ericsson's Research centre in Massy, France. In 2004, he founded Montimage, and is currently its CEO. He is the originator and main Architect of Montimage Monitoring Tool (MMT). His main interests are: network and application monitoring and security; detection and mitigation of cyber-attacks; and building critical systems that require the use of state-of-the-art fault-tolerance, testing and security techniques.

Wissam Mallouli is a Senior R&D Engineer at Montimage and has graduated from the National Institute of Telecommunication (INT) Engineering School in 2005. He received his Master's degree from Evry Val d'Essonne University also in 2005 and his PhD in Computer Science from Telecom and Management SudParis (France) in 2008. His topics of interest cover formal security testing of embedded systems and distributed networks. He is an expert in testing methodologies. He has a strong background in monitoring and testing network security (protocols and equipment).

This paper is a revised and expanded version of a paper entitled 'An effective attack detection approach in wireless mesh networks' presented at the AINA Conference, Barcelona, Spain, 25–28 March 2013.

1 Introduction

Wireless mesh network (WMN) is an alternative architecture to IEEE 802.11 standard for data traffic, voice, and video transmission to end-users beyond cable and traditional wireless infrastructure. WMNs advantages include low cost, easy deployment, having fault tolerance capabilities, and can provide access to remote users, which would be more difficult to reach using conventional wired networks.

WMNs resemble much mobile ad hoc networks (MANETs) since both are self-organised networks and use multi-hop routing strategy for transmitting traffic from a source node to a destination node. The main difference between these two technologies lies in the fact that, in WMNs, the architecture is composed of static mesh routers, which form a fixed wireless mesh backbone where the mesh clients, e.g., mobile devices, connect to access the internet.

The concept of mesh networking brings a series of advantages (Akyildiz et al., 2005), which make it increasingly interesting to implement, such as:

1 low cost network: since sharing bandwidth resources significantly reduces the total cost of the network, then enabling the creation of community networks

2 easy deployment: WMNs have the characteristic of being self-configurable, as a result they are easy to deploy since they do not require complex configuration, and there is no need to reconfigure the nodes if new nodes join the network

3 fault tolerant: the ability of dynamic routing combined with the existence of multiple access routes to a node enables the network to easily recover from failures, such as disconnection of communication links

4 reliable: one of the most prominent features of WMNs is that the reliability and connectivity increase as more nodes are added to the network, therefore, the increase of network size, unlike traditional networks, is not an issue for this type of network.

However, WMNs have some limitations (Zhang et al., 2008). Most of the problems faced by mesh networks can be attributed to the recent attention given to this kind of network, such as:

1 the absence of standardisation: the lack of a common standard supported by industry and academic community makes it impossible to adopt this technology on a large-scale

2 the absence of efficient security schemes: security is still an open research field in WMNs, in addition to regular security threats from traditional wireless networks, there is still the problem of ensuring integrity and authenticity of routing and data packets travelling between nodes in the network.

With respect to security, WMNs are similar to WLANs, in that an attacker can passively intercept radio transmissions and exploit them. Thus, messages transmitted between

nodes can be captured and altered, e.g., the network can be inappropriately accessed by malicious nodes and be a target of, for instance, denial of service (DoS) attacks. Because of the decentralised nature and multi-hop architecture of WMN, it may take a longer time to detect and mitigate such attacks, giving unwanted intruders a considerable advantage over the network administration. Consequently, attacks in WMNs should be precisely identified for each individual node.

In WMNs, it is important to ensure confidentiality, integrity, authentication, access control, and availability for the routing operations, data services, and for the nodes of the mesh infrastructure. The network traffic can be protected at different layers, including physical, MAC, network, transport, and application layers, by using different schemes for encapsulating frames, authentication protocols, and encryption algorithms. However, an adversarial node owning a legitimate key, or even a non-authenticated mobile node that has access to the network, can broadcast malicious traffic, intercept and modify packets, or refuse to forward traffic by discarding packets received from other nodes.

Possible attacks in the WMN can be characterised into two types:

1 external attacks, in which attackers outside of the mesh network can impersonate nodes (e.g., carry out spoofing attacks or inject invalid information)

2 insider attacks, which enable attackers to execute more severe threats from compromised nodes participating in the network.

These are more difficult to prevent since the node has a valid key.

In this work, we focus on detecting message alteration attacks and, more specifically, routing manipulation attacks (Morais and Cavalli, 2011b). These attacks are internal intrusions where a malicious node sends fakes routing messages on behalf of another node in order to violate the integrity of the network routes. We show how to diagnose this routing attack by making use of the traces generated by nodes executing better approach to mobile ad hoc network (BATMAN) routing protocol (Neumann et al., 2008). We then apply the Montimage Monitoring Tool (MMT) tool (Wehbi et al., 2012) to analyse the traces and generate routing events in a sorted manner. These events represent the protocol routing behaviour at each node. Then, we employ attack detection algorithms using the tool output to identify malicious traffic for the nodes.

The innovative aspect of this work is mainly that the approach for detecting active attacks against the routing functionality of a mesh network is based on a formal tool-supported specification approach. This approach enables the detection of normal or abnormal routing behaviour and allows automating the generation and analysis of routing events. In this way, it is possible to efficiently identify compromised nodes that generate malicious traffic with the intention of disrupting the network routing service.

The remainder of this paper is structured as follows. Section 2 discusses the related work. In Section 3, we describe the attack detection approach. Section 4 introduces the MMT tool we implemented to detect the attacks. Section 5 presents the evaluation procedure and results of the approach. Finally, Section 6 concludes this paper.

2 Related work

Many studies have been conducted to conceive security mechanisms for the existing routing protocols in WMNs. For that purpose, various secure routing protocols have been designed such as ARAN (Sanzgiri et al., 2002), Ariadne (Hu et al., 2005), and SAODV (Zapata, 2002). Most of the approaches rely on key management or cryptographic technologies to authenticate and protect the messages exchanged among nodes, and to guarantee that unauthorised nodes will not be able to join the network. However, these schemes alone cannot prevent a compromised (which has a valid key) or a non-authenticated mobile node from launching attacks against the mesh nodes to disrupt the routing communication and data delivery services, as we try to address in the present article.

Various security methods have been proposed for WMNs, but not all are fully applicable in realistic WMN scenarios. In Wu and Li (2006), the authors propose a private routing algorithm to address the problem of privacy in WMNs. The approach is based on layered encryption to hide the routing information at the mesh routers. However, the authors do not show how the mesh nodes implement authentication, key distribution, and key agreement. In addition, the approach performance is not demonstrated as cryptographic schemes usually incur high computational overhead to the devices.

Santhanam et al. (2007) introduced an active cache-based defence scheme at the mesh routers for detecting flooding attacks in the network. The approach applies a ‘most frequently used’ cache mechanism to detect malicious flows. But the authors do not specify which type of malicious flow the approach aims at identifying, whether routing traffic or data traffic. Islam et al. (2009) proposed a secure layer-2 path selection (SLPS) mechanism for WMNs based on IEEE 802.11s. The scheme uses cryptographic extensions to provide authenticity and integrity for the routing messages in order to prevent unauthorised manipulation of mutable fields in routing messages. Nevertheless, the proposed SLPS protocol is still susceptible to message fabrication attacks.

Routing manipulation attacks (or link spoofing attacks), which is a type of routing disruption attack, has not been sufficiently studied in WMNs. Kannhavong et al. (2006) presented a collusion attack against optimised link state routing (OLSR) protocol in which a pair of colluding attackers prevent routes to a specific node from being established. The attacker advertises fake two-hop neighbour links to the target node in order to take control over the target node’s routes. The authors implemented the attack in a network simulator, and presented a method for detecting

this routing disruption attack by adding information of two-hop neighbours in the routing messages broadcasted by the nodes. The drawback of this detection scheme is that the target node cannot detect the routing attack if the attacker modifies the content of the routing messages transmitted by all the neighbouring nodes to this target node.

Raffo et al. (2005) proposed a location-based detection scheme to identify link-spoofing attacks against OLSR. The method utilises signatures together with timestamps and a GPS device. The technique adds the geographical position and the timestamp of the sending node in the control messages so that every node knows the correct position of every other node in the network. This method detects link spoofing by comparing the geographical data to the routing data received by the node. If contradictory link information is found, the node discards the false routing message. The principal inconvenience with this approach is that it will not function in a scenario where mobile nodes are allowed to join the network without having a GPS device installed.

3 The attack detection approach

In this section, we present the attack detection approach. First, we introduce BATMAN routing algorithm. Then, we describe the steps for detecting routing attacks in a mesh topology by making use of the nodes' traces.

3.1 BATMAN protocol

We opted for BATMAN as routing protocol since we applied it in previous work (Morais and Cavalli, 2011a) for security analysis related to routing attacks in WMN. However, our attack detection approach can be applied to other routing protocols as well.

BATMAN routing algorithm is detailed as follows:

1 Each node, also referred as originator (*Orig*), periodically broadcasts hello messages, known as originator messages (OGMs), to inform its neigbors about its existence. An OGM contains at least: the *Orig* address, the source node address (*Src*), the previous sender address (*PSrc*), a unique sequence number (*Seq*), a time to live (TTL), and a transmission quality (TQ) value.

2 As a neighbouring node (*Nb*) receives an OGM, it modifies the *Src* address to its own address and rebroadcasts the OGM according to BATMAN forwarding rules to inform its neighbouring nodes about the existence of the node that originated the OGM, and so on and so forth.

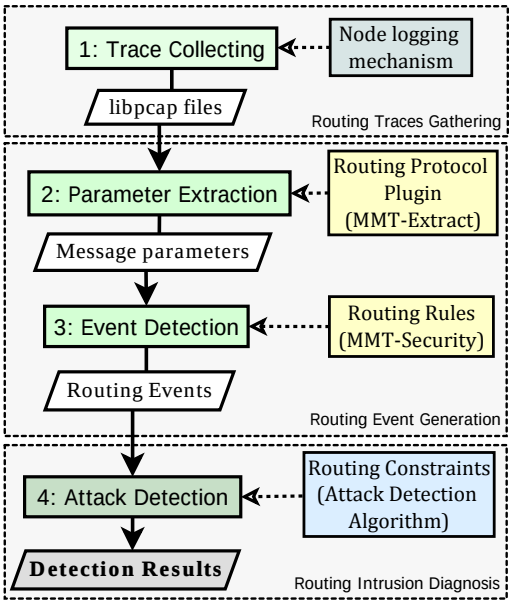
3 Hence, the mesh network is flooded with OGMs until every node has received it at least once, or until they are discarded because of packet loss in the wireless link, or until their TTL value has expired.

4 The *Seq* of OGM is used to verify the message freshness, i.e., to distinguish between new OGMs and duplicated ones in order to guarantee that each OGM is only counted once. The amount of OGMs, i.e., the total number of *Seq*, received from an *Orig* via each *Nb* is used as a metric to calculate the route quality. Thus, BATMAN will choose the *Nb* from which it has received the highest amount of *Seq* (OGMs) recorded in a sliding window (i.e., the packet count metric) as the best next-hop to the *Orig*.

3.2 Method of attack detection

Figure 1 illustrates the methodology we apply for detecting routing attacks, which comprises four steps.

Figure 1 Steps of the attack detection process (see online version for colours)



The first step of the approach is trace collecting, where output files generated by each node are collected separately for further analysis. For obtaining such traces, we employ the logging mechanism of the node, for instance, a monitoring tool sniffing the traffic travelling over the node network interface. A trace file contains communication information for a given node, i.e., all the routing messages exchanged between the node and its neighbouring nodes. For convenience, we assume that the file format of traces is libpcap (http://www.tcpdump.org/) since it is the main capture file format used in most of networking tools. The libpcap file format is a basic format to save captured network data. The typical file extension of libpcap-based files is: .pcap. Therefore, in this work, we refer to the traces as 'PCAP files'.

In the second step, we perform parameter extraction. First, the collected traces are examined in detail individually and relevant message parameters are extracted according to the routing protocol described in Section 3.1, and also taking into account the protocol specification. A trace file is composed of a PCAP file. Each PCAP file is composed of a set of captured BATMAN packets, i.e., OGMs received or

broadcasted by the node that are ordered with respect to the reception and transmission time respectively.

Then, we use the routing protocol plug-in, i.e., the BATMAN plug-in, which was integrated to MMT-extract library of MMT tool, for extracting the protocol parameters. MMT-extract provides a plug-in architecture, which allows adding new protocols and packet structures for these protocols for checking attributes for the protocol packets. BATMAN plug-in performs the following steps:

1 analyses the PCAP file

2 identify relevant BATMAN packets

3 extracts specific message parameters from each BATMAN packet.

After the extraction is terminated, we will have access to the following parameters from the OGMs that were received or broadcasted by the node: $P = \{Seq, Orig, Src, PSrc\}$. Moreover, we could extract additional parameters from the OGM packet, e.g., BATMAN version and TTL, in order to verify if such parameters are modified by an attacker during the message transmission in the context of man-in-the-middle attacks, or corrupted due to communication failures.

The next step of the attack detection approach is event detection. In the first place, we have to define the routing events we aim identifying in the routing messages based on the parameters *P* extracted from each routing packet. Thus, we establish the following routing events, which represent BATMAN protocol behaviour at the node.

- *OGM_rcv*: means that an OGM is received by the node from a neighbouring node *Nb*.
- *OGM_brd*: means that the node broadcasts an OGM to its neighbour *Nb*.
- *OGM_rebr*: means that the node rebroadcasts an OGM received from a neighbouring node *Nb*.

To detect these routing events, we implement routing rules for MMT-security module of MMT tool. A routing rule is a matching condition expressed in the XML format defined by MMT-security to specify security properties as temporal logic rules. MMT-security essentially formulates security properties and rules in the form of IF-THEN conditions, which makes use of parameters obtained from the protocol packets. Therefore, we implement routing rules for each routing event defined for BATMAN protocol.

- *R*1 for *OGM_rcv*:

  if $Src_i \neq N_{src}$ **then** $Rcv \leftarrow P_i$, for $1 \leq i \leq n$;

  where *n* is the number of OGMs analysed by BATMAN plug-in; N_{src} is the node address, i.e., the MAC address; and *Rcv* is the set of OGMs received by the node, i.e., the set of routing events *OGM_rcv*, that contains a set of parameters *P* for each OGM received by the node.

- *R*2 for *OGM_brd*:

  if $(Src_i = N_{src}) \wedge (Src_i = Orig_i)$ **then**
  $Brd \leftarrow P_i$, for $1 \leq i \leq n$;

  where *Brd* is the set of OGMs broadcasted by the node, i.e., the set of routing events *OGM_brd*, that contains sets of parameters *P* for all the OGMs broadcasted by the node.

- *R*3 for *OGM_rebr*:

  if $(Src_i = N_{src}) \wedge (Src_i \neq Orig_i)$ **then**
  $Rebr \leftarrow P_i$, for $1 \leq i \leq n$;

  where *Rebr* is the set of OGMs rebroadcasted by the node, i.e., the set of routing events *OGM_rebr*, that contains sets of parameters *P* for all the OGMs rebroadcasted by the node.

The sets *Rcv*, *Brd*, and *Rebr*, which stand for the routing events detected for BATMAN routing protocol, are the outcome of this event detection step. The results are provided to the attack detection algorithm for identification of routing intrusions in the network traffic.

With the use of MMT tool, we have some advantages, for example, easy extraction of message attributes since once the routing protocol plug-in is integrated to MMT-extract, we can readily identify the suitable routing parameters from the PCAP trace files collected for each node. In addition, the employment of routing rules allows sorting the routing traffic in a practical manner, so we can analyse particular aspects of the routing behaviour of the protocol, not to mention that the entire process of generating routing events is automated.

3.3 Attack detection algorithm

The last step of the approach is attack detection. As mentioned earlier in this work, we aim at detecting message fabrication attacks. For this, we define routing constraints relying on the routing protocol specification in order to identify message fabrication misbehaviour that disrupts the correct behaviour of the routing protocol. The routing constraints make use of the routing event sets produced in the preceding step of event detection. In the case that a routing constraint is violated, we count the number of inconsistencies *F* diagnosed for each violated routing constraint separately. Rcv_{Nb} and Reb_{Nb} are the routing events detected for neighbouring node *Nb*.

- *C*1: $\forall Seq_i \in Rebr$, **if** $Seq_i \notin Rcv$ **then** F_1++

  Checks if Seq_i rebroadcasted by the node for $Orig_i$ was received before by the node from neighbour Src_i, where F_1 is the number of fake OGMs, i.e., *Seq*, fabricated and rebroadcasted by the node for *Orig*.

- *C*2: $\forall Seq_i \in Rcv$, **if** $Seq_i \notin Rcv_{Nb}$ **then** F_2++

  Checks if Seq_i received by the node for $Orig_i$ was firstly received by its neighbour Src_i which should be received before from $PSrc_i$, where F_2 is the number of fake

OGMs, i.e., *Seq*, received by the node that are fabricated and rebroadcasted by *Nb*.

- *C*3: $\forall Seq_i \in Rcv$, **if** $Seq_i \notin Rebr_{Nb}$ **then** F_3++

  Checks if Seq_i received by the node for $Orig_i$ was previously rebroadcasted by its neighbouring node Src_i, where F_3 is the number of fake OGMs, i.e., *Seq*, received by the node that are fabricated and rebroadcasted by a malicious node.

We then apply routing constraints *C*1, *C*2, and *C*3 for detecting message fabrication attacks in a neighbourhood according to the attack detection algorithm. The algorithm analyses the routing behaviour of the neighbourhood by simultaneously examining the routing behaviour of every node and its respective neighbours. In that case, we have a global view of the network activity for the vicinity that enables us to accurately track down the source of intrusion, i.e., the malicious node, in a collaborative manner.

For a better understanding of the algorithm, we consider that *Rcv* and *Rebr* are the routing event sets generated for node O_1, and Rcv_{Nb_i} and $Rebr_{Nb_i}$ are the corresponding routing event sets generated for the neighbours $N = \{Nb_1, \ldots, Nb_k\}$, $1 \le i \le k$, where k is the number of neighbouring nodes of node O_1. This algorithm should be executed for each node O_j in the mesh network, $1 \le j \le l$, where l is the number of nodes. We assume each node O_j has at least one neighbour Nb_i.

Attack detection algorithm

Input: *Rcv*, *Rebr*, Rcv_{Nb_i}, $Rebr_{Nb_i}$

Output: F_{rebr}, F_{RbN}, F_{Rcv}

```
1.  for all Seq ∈ Rcv then
2.      if Seq ∉ Rcv then
3.          F1++;
4.      end if
5.  end for
6.  for all Seq ∈ Rcv do
7.      for all Nbi ∈ N do
8.          if Seq ∉ Rcv_Nbi do
9.              F2++
10.         else if Seq ∉ Rebr_Nbi then
11.             F3++
12.         end if
13.     end for
14. end for
```

As a result, this step provides the detection results $F = \{F_1, F_2, F_3\}$, which reflect potential malicious routing behaviour observed on the network traffic captured at node O_1, or in the traffic collected from one or more of its neighbouring nodes Nb_i.

4 The MMT tool architecture and features

In this section, we present the MMT tool and how we used it to analyse the protocol traces. First, we describe the security properties formalism used to specify the security requirements of the system/network under observation. Then, we present the MMT tool architecture and security features, and show how they can answer the security monitoring challenges described above. Finally, we present the application of MMT to the BATMAN protocol case study. A more detailed description of MMT can be found in Mallouli et al. (2012).

The MMT tool is an online monitoring solution that provides real-time visibility of network traffic, application communication, flow and usage levels. It facilitates network security, performance monitoring and operation troubleshooting. MMT's rules engine can correlate network and application events in order to detect operational, security and performance incidents.

4.1 MMT tool and its security properties formalism

The main objective of the MMT security properties is to formally specify security goals and attack behaviour related to the application or protocol being observed. The 'MMT-Security property' model is inspired from LTL logic (Wehbi et al., 2012) and can refer to two types of properties: 'security rules' and 'attacks' described as follows:

- A security rule describes the expected functional or security behaviour of the application or protocol. If it is violated then it indicates an abnormal behaviour.
- An attack describes a malicious behaviour whether it is an attack model, a vulnerability or a misbehaviour. If detected then it indicates abnormal behaviour that could be due to an attack.

It must be noted that the events that we take into account within the 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. The main definition of an MMT-security property is provided by the definition that follows. Other definitions that allow understanding the basics of the model used can be found in Mallouli et al. (2012).

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 meta values or field values of the exchanged packets.

4.1.1 Definition of MMT-security property

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

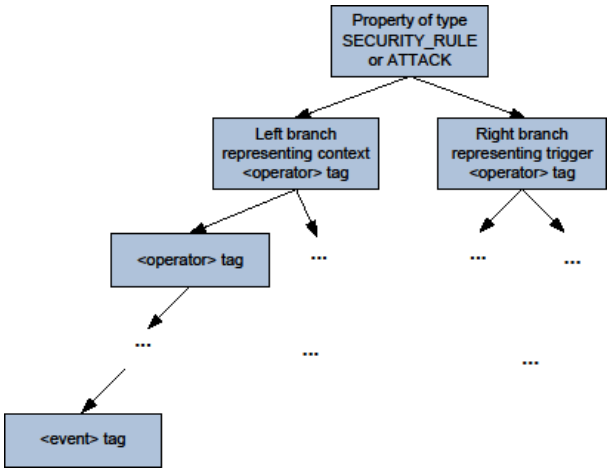
$$e_1 \xrightarrow{W,n,t} e_2$$

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

4.1.2 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 their interpretation easier for both humans and software. The main structure of a property is given in Figure 2.

Figure 2 MMT property structure (see online version for colours)



Each property begins with a <property> tag and ends with </property>. A property is a 'general ordered tree' as shown in Figure 2. 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*.

4.1.3 Multi data sources management for security analysis

In the context of MMT, deep packet inspection (DPI) and deep flow inspection (DFI) are used to help detect and tackle harmful traffic and security threats; and, to throttle or block undesired behaviour. 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.

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:

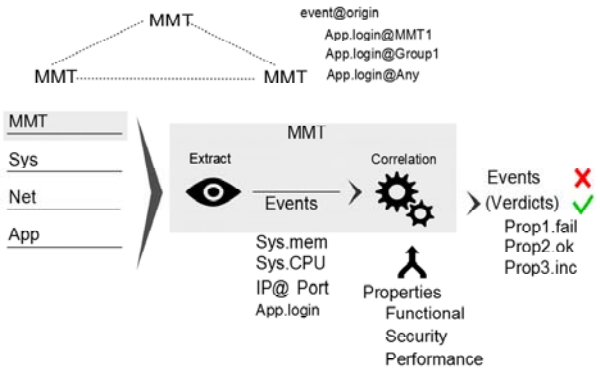
- *Local analysis*: the collected traffic is analysed for security purposes in one probe that captures network traffic from one or several interfaces.
- *Centralised analysis*: the traffic capture is distributed but the security analysis is centralised. All data sources send their collected traffic (filtered or not) to the same master server that correlates the traces (i.e., need to synchronise 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 particularly for ad hoc network case studies. The communication between probes is an ongoing work for the MMT tool.

Figure 3 shows how the MMT monitoring tool works. Input can be provided from different raw data sources (e.g., network, system, application). It can also be provided by remote MMT probes in the context of a distributed architecture. The extraction engine allows to retrieve relevant attributes for the further analysis and the correlation engine allows the correlation of extracted events in time in order to check the validity of MMT properties. These properties specify the functional, security or performance behaviour that needs to be validated. The correlation engine will then produce the verdicts for each property (fail, ok or inconclusive).

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 (Hu et al., 2005) nor requiring writing executable scripts as in BRO (Sanzgiri et al., 2002). They allow a more abstract description of a 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.

Figure 3 MMT distributed correlation (see online version for colours)

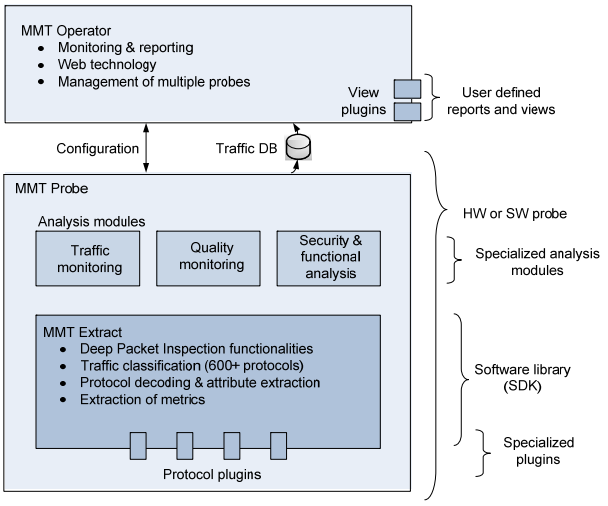


4.1.4 *MMT-security architecture*

MMT-security is composed of three complementary, but independent, modules as shown in Figure 4:

- *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 analysing: protocols' field values; network and application quality of service (QoS) parameters; and, key performance indicators (KPI). MMT-extract incorporates a plug-in architecture for the addition of new protocols or message types, and a public API for integration into third party probes.
- *MMT-security* is a security analysis engine based on MMT-security properties. MMT-security analyses 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. Other analysis modules exist (e.g., for video quality analysis and traffic analysis) or new ones can be added.
- *MMT-operator* is a visualisation 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 customisable, i.e., the user will be able to define new views or customise one from a large list of predefined views. At the time of writing this paper, a web-based representation of the analysis results is provided.

Figure 4 MMT global architecture (see online version for colours)



4.1.5 *Application of MMT to the BATMAN protocol case study*

To be able to apply the tool to the BATMAN protocol case study we first created the BATMAN plug-in for the MMT-extract library and then specified the properties that we wished to detect for the MMT-security module.

An MMT-extract plug-in will serve to initialise a protocol structure that contains the required information regarding the protocol attributes, as well as the functions allowing extracting the data corresponding to these attributes. For creating a MMT-extract plug-in (written in the C language), a set of utility structures, utility functions and generic extraction functions are available as a C language API to simplify the task.

To create the BATMAN plug-in we defined a C header file that specifies the different protocol attributes that make up a BATMAN communication packet, and a C source file that implements the functions needed to extract the attribute values from this packet. This task is relatively easy when using an existing plug-in as a template. Implementing and testing it can be done in one day.

To be able to detect the security-related events needed for identifying malicious nodes, we specified the security properties that are used by the MMT-security module. These properties were done to identify badly formed packets and identify routing events: *Rcv*, *Brd*, and *Rebr* for each node. The properties were written in XML and are described in Section 3.2.

The time needed for analysing the PCAP traces is very small (a few ms) due to the fact that the trace files are small (less that 100 Mb). The efficiency of the MMT-security module for the type of properties used was in the order of the number of properties *pr* times the number of packets to analyse *pa* ($O(n^2)$).

5 Experimental evaluation

In this section, we use a virtualised network environment to demonstrate the viability of our trace-based attack detection solution and assess the efficiency of the approach.

5.1 Experimental platform

The mesh topology is emulated by virtual machines (VMs), which consists of QEMU (Claffy et al., 2012) instances, with Linux OS installed, running BATMAN routing protocol. *Batman-adv* v.2011.1.0 (QEMU, http://wiki.qemu.org) is compiled from source code and loaded into the Linux OS of each QEMU instance as kernel module.

The hardware configuration of a QEMU instance consists of standard IBM PC 32Bit computer architecture with one CPU, that is the default QEMU 32Bit CPU type, and 256 MB of main memory, and compatible with Linux OS. We assume that each node in the network has an equivalent hardware configuration to that one of the QEMU VM that corresponds to common wireless mesh devices such as Wi-Fi routers or mobile devices with sufficient hardware resources.

The virtual nodes, i.e., the VMs, are interconnected by a virtual switch (Open-Mesh.net, http://www.open-mesh.net/), which emulates bi-directional link communication between the nodes and corresponding degradations in the quality of the links due to the packet loss rate. For the experiments, we assume a fixed mesh topology.

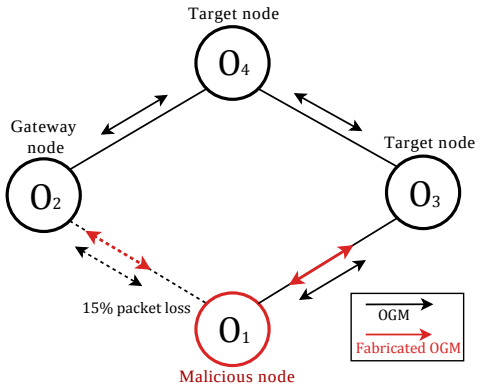
During the experiment execution, we use the logging mechanism of QEMU to dump the network traffic of each node into separate PCAP logging files. The PCAP files are synchronised with a global clock, i.e., the time of the main machine that executes the VMs. This procedure of saving PCAP files for each mesh node apart concerns the trace collecting step of the approach described in Section 3.2.

5.2 Attack emulation scenario

Our next step is to define a mesh network scenario where routing communication is carried out among the nodes, so we can analyse the types of routing events that may occur. The emulated network scenario is composed of four nodes: O_1, O_2, O_3, and O_4, where each node only communicates with its neighbours, as seen in Figure 5.

Since the link between nodes O_1 and O_2 has a packet loss rate (15%) higher than the one between nodes O_2 and O_4 (0%), nodes O_3 and O_4 normally prefer a route toward gateway node O_2 via node O_4. For instance, in the routing table of node O_3, the route entry of node O_2 has as best next-hop node O_4. In this scenario, node O_1 is compromised, and creates and sends spoofed OGMs on behalf of node O_2, i.e., node O_1 fabricates *Seq* for *Orig* = O_2, which in fact it did not received from node O_2. In this manner, node O_1 can divert the routes of target nodes O_3 and O_4 toward gateway node O_2 to the own malicious node O_1, thus characterising a routing manipulation attack.

Figure 5 Mesh topology emulated for the experiment (see online version for colours)



As routing decisions rely on statistical analysis of the amount of OGMs, i.e., *Seq*, received rather than information contained in the routing messages, malicious node O_1 has to fabricate a number of fake OGMs that are numerous enough to surpass the legitimate OGMs broadcasted by node O_2 in order to redirect the routes of nodes O_3 and O_4 to node O_1. In other words, the malicious node O_1 has to continuously win the neighbour ranking of target nodes O_3 and O_4 towards gateway node O_2. For instance, node O_3 has to constantly choose *Nb* O_1 as best next-hop to O_2. As a result, the routing table of target nodes O_3 and O_4 will update their route entry of node O_2 for best next-hop to node O_1.

The routing manipulation attack is implemented using the Ethernet packet generator packETH (VDE Switch, http://wiki.virtualsquare.org/wiki/index.php/VDE) which permits node O_1 to create and broadcast bogus OGMs with continuous valid *Seq* on the network interface of own node O_1.

5.3 Attack detection results

After collecting the PCAP files generated by the nodes, we will examine the trace files for nodes O_1, O_2, and O_3 since these nodes are directly involved in the malicious routing behaviour performed by node O_1. Thus, we employ the MMT-Extract tool to extract the OGM parameters defined in the second step of Parameter Extraction. We obtain three parameter sets: P^{O_1}, P^{O_2}, and P^{O_3}, which comprise the parameters P extracted from all the OGMs in each collected PCAP file. For instance, for node O_1 we have $P^{O_1} = \{P_1, ..., P_n\}$, where n is the number of OGMs found by BATMAN plug-in when analysing the PCAP file of node O_1.

By using the parameter sets P^{O_1}, P^{O_2}, and P^{O_3} obtained from the traces along with the routing rules implemented in third step of event detection, we can precisely identify the routing events for each node. Then, we apply the MMT-Security tool to these parameter sets to generate the routing event sets: *Rcv*, *Brd*, and *Rebr* for nodes O_1, O_2, and O_3.

Once we have the routing event sets, we execute the attack detection algorithm in the context of the routing

behaviour of nodes O_1, O_2, and O_3, which corresponds the last step of the approach. For instance, for node O_1 we consider the set of neighbours $N = \{O_2, O_3\}$ and the corresponding routing event sets: $\{Rcv_{O_i}, Rebr_{O_i}\}$, $2 \leq i \leq 3$, in addition to the routing event sets $\{Rcv, Rebr\}$ generated for node O_1.

The output of the algorithm for each node, i.e., the detection results $F = \{F_1, F_2, F_3\}$, are exhibited in Table 1. Based on this, we remark that node O_1 is the source of intrusion in the neighbourhood since node O_1 detects a significant number of fake OGMs F_1 broadcasted by the own node. And its neighbours O_2 and O_3 also detect a considerable number of fake OGMs F_2 received from their respective neighbour O_1 that are fabricated by node O_1.

Table 1 Detection results

Number of inconsistencies	*Node* O_1	*Node* O_2	*Node* O_3
F_1	1,630	-	-
F_2	-	1,948	1,510
F_3	-	-	-

The duration of the experiment was 30 minutes. We used the default BATMAN OGM broadcasting rate of 1,000 ms for the routing attack, and node O_1 starts broadcasting fake OGMs at 48 seconds. Therefore, the total number of fake OGMs broadcasted by node O_1 during the experiment execution is $T = 1{,}752$. We observe nodes O_1 and O_3, except node O_2 that is affected by the link packet loss, detect less fake OGMs than expected, i.e., $F_1 < T > F_2$. The reason is that the detection algorithm does not consider the arrival time of OGMs when checking *Seq* in the routing event sets of the nodes, so if a fake OGM broadcasted in the past has the same *Seq* value as a legitimate OGM transmitted at the present time, the algorithm considers this fabricated OGM as valid. Since OGM broadcasting rate is 1 sec, the algorithm should verify *Seq* between the routing event sets of nodes considering a time difference of 2 seconds.

The ratio of attack detection with regard to the malicious routing behaviour identified in the traces is defined as: $R = \frac{|T-F|}{T}$. Then, we calculate R using the number of detected intrusions F in Table 1, and we obtain: $R_{O_1} = 6\%$, $R_{O_2} = 12\%$, and $R_{O_3} = 14\%$. We notice that the smallest difference concerns the attack detection in the perspective of node O_1 since the number of detected fake OGMs F_1 relies exclusively on the routing events *Rebr* and *Rcv* identified in the local trace of node O_1, which is not affected by external transmission factors, such as packet drop.

However, from the point of view of nodes O_2 and O_3 we note a higher deviation in the attack detection compared to node O_1. In fact, the trace file collected at node O_2 is particularly impacted by the high packet loss rate between nodes O_1 and O_2, which results in loss of OGMs exchanged between these nodes. Thus, the number of intrusions F_2 that is calculated taking into consideration the routing event sets *Rcv* of both nodes O_1 and O_2 is surely influenced by the packet loss rate present in the link.

Moreover, the divergence R in terms of F_2 for nodes O_2 and O_3 is caused by the lack of synchronisation between the logging mechanism of nodes, at the beginning and at the end of the experiment, when the logging mechanism starts and stops saving routing messages to the respective trace files. For example, at the time node O_3 stops recording traffic data to the trace file, its PCAP log file contains more routing information from nodes O_1 and O_2 than the respective PCAP files of node O_1 and node O_2 which stopped capturing the routing packets earlier. Therefore, metric F_2 that is basically calculated correlating routing message parameters between neighbouring nodes will produce incorrect results since it is missing routing information in the routing event sets.

The additional fake OGMs detected by node O_2 with respect to T, since $F_2 > T$ for O_2, do not properly correspond to attack attempts and can be considered as false positives, i.e., false alarms. They are mainly caused by degradations in the link quality, i.e., the packet loss rate of the link. Therefore, a threshold based on transmission impairments should be defined in order to distinguish between false positives and malicious routing behaviour when analysing the attack detection results F of each node.

6 Conclusions

In this work, we have implemented the BATMAN plug-in for MMT-extract. Then, we applied MMT-extract to the BATMAN traces supplied by each node in order to obtain relevant BATMAN properties regarding the routing protocol semantics. We define routing events that correspond to the routing behaviour of the protocol in execution at the node. With the help of MMT-security, we are able to properly generate such routing events in a suitable output format, which are then used as input for the attack detection algorithm. We managed to successfully detect intrusion attempts performed by a malicious node participating in the mesh network and broadcasting phoney routing messages in its neighbourhood.

As a future work, we envisage evaluating the approach in malicious scenarios where two or more nodes collude to carry out routing misbehaviours, i.e., coordinated attacks.

Acknowledgements

This research work is partially financed by the CelticPlus project SAN (http://projects.celtic-initiative.org/SAN/).

References

Akyildiz, I.F., Wang, X. and Wang, W. (2005) 'Wireless mesh networks: a survey', *Computer Networks*, March, Vol. 47, No. 4, pp.445–487 [online] http://dl.acm.org/citation.cfm?id=1071646.

Claffy, K.C., Dainotti, A. and Pescapè, A. (2012) 'Issues and future directions in traffic classification', *IEEE Network (NETWORK)*, Vol. 26, No. 1, pp.35–40.

Hu, Y-C., Perrig, A. and Johnson, D.B. (2005) 'Ariadne: a secure on-demand routing protocol for ad hoc networks', *Wireless Networks*, January, Vol. 11, Nos. 1–2, pp.21–38.

Islam, Md.S., Hamid, Md.A., Choi, B.G. and Hong, C.S. (2009) 'Securing layer-2 path selection in wireless mesh networks', *Information Security Applications, Lecture Notes in Computer Science*, Vol. 5379, pp.69–83, DOI: 10.1007/978-3-642-00306-6_6.

Kannhavong, B., Nakayama, H., Kato, N., Nemoto, Y. and Jamalipour, A. (2006) 'A collusion attack against OLSR-based mobile ad hoc networks', *Proc. Global Telecommunications Conference (GLOBECOM '06)*, November, pp.1–5.

libpcap, *Portable C/C++ Library for Network Traffic Capture*, [online] http://www.tcpdump.org/ (accessed 1 June 2014).

Mallouli, W., Wehbi, B., Montes de Oca, E. and Bourdelles, M. (2012) 'Online network traffic security inspection using MMT tool', in the *9th Workshop on System Testing and Validation (STV)*, Paris, France, October.

Morais, A. and Cavalli, A. (2011a) 'Detection of attacks in wireless mesh networks', *Proc. 5th Latin-American Symposium on Dependable Computing (LADC 2011)*, April, pp.45–54.

Morais, A. and Cavalli, A. (2011b) 'Route manipulation attack in wireless mesh networks', in *2011 IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 22–25 March, pp.501–508, DOI: 10.1109/AINA.2011.11.

Neumann, A., Aichele, C., Lindner, M. and Wunderlich, S. (2008) *Better Approach to Mobile Ad-Hoc Networking (B.A.T.M.A.N.)*, April, IETF Internet-Draft (expired October 2008) [online] http://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00 (accessed 23 May 2014).

Open-Mesh.net, *B.A.T.M.A.N. (Better Approach to Mobile Ad-Hoc Networking)* [online] http://www.open-mesh.net/ (accessed 16 May 2014).

QEMU, *Machine Emulator and Virtualizer* [online] http://wiki.qemu.org (accessed 3 April 2014).

Raffo, D., Adjih, C., Clausen, T. and Muhlethaler, P. (2005) 'Securing OLSR using node locations', *Proc. 11th European Wireless Conference 2005*, April, pp.1–7.

Santhanam, L., Nandiraju, D., Nandiraju, N. and Agrawal, D. (2007) 'Active cache based defense against dos attacks in wireless mesh network', *Proc. 2nd International Symposium on Wireless Pervasive Computing (ISWPC '07)*, February, pp.5–7.

Sanzgiri, K., Dahill, B., Levine, B.N., Shields, C. and Belding-Royer, E.M. (2002) 'A secure routing protocol for ad hoc networks', *Proc. IEEE International Conference on Network Protocols (ICNP 2002)*, November.

VDE Switch, *Virtual Distributed Ethernet Switch* [online] http://wiki.virtualsquare.org/wiki/index.php/VDE (accessed 29 March 2014).

Wehbi, B., Montes de Oca, E. and Bourdelles, M. (2012) 'Events-based security monitoring using MMT tool', *Proc. IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST '12)*, April, pp.860–863, DOI: 10.1109/ICST.2012.188.

Wu, X. and Li, N. (2006) 'Achieving privacy in mesh networks', *Proc. 4th ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '06)*, ACM, New York, pp.13–22.

Zapata, M.G. (2002) 'Secure ad hoc on-demand distance vector routing', *ACM Mobile Computing and Communications Review (MC2R)*, July, Vol. 6, No. 3, pp.106–107.

Zhang, W., Wang, Z., Das, S.K. and Hassan, M. (2008) 'Security issues in wireless mesh networks', in *Book Wireless Mesh Networks: Architectures and Protocols*, Springer, New York.