Efficient Signature Management for Intrusion Detection in Mobile Ad-hoc Networks

Stefan Karsch, Florian Zaefferer, Jens Haag
Cologne University of Applied Sciences
Institute of Informatics
Steinmüllerallee 1, 51643 Gummersbach
Germany

{stefan.karsch, jens.haag}@fh-koeln.de,
florian.zaefferer@gmx.de

Marko Jahnke
Research Institute for Communication,
Information Processing and Ergonomics
Neuenahrer Str. 20, 53343 Wachtberg
Germany

jahnke@fgan.de

ABSTRACT

Signature-based Intrusion Detection Systems (IDS) are an accepted concept for the detection of attacks on computer networks. The reliability of these systems depends on continuous updates of the signature database. Usage of outdated databases makes the most current attacks – and according to experience the most prevalent attacks – undetectable. Deploying signature-based IDS in wireless mobile ad hoc networks (MANETs) adds the demand for efficiency in performing these updates due to the limited transmission bandwidth and local resources of small scaled mobile devices. Additionally, the special characteristics of ad hoc networks have to be taken into account. Especially the connection volatility and the constantly changing network topology during a mission have to be considered in the design of an update strategy. In this contribution, a method to efficiently manage the signature updates for network-based IDS on mobile devices in a MANET with central connectivity to a wide area network is discussed. The developed methodology was implemented in a MANET test bed using the publicly available IDS Snort as an example.

1 INTRODUCTION

Compared to wired computer networks, wireless networks are exposed to greater security risks. To gain access to wired networks, direct access to the cable is necessary. In wireless networks, typically no physical access protection is available. Radio contact is sufficient for accessing the information on the network.

Wireless networks are either operated in infrastructure or ad hoc mode. In infrastructure mode, every network station connects to a central instance (access point) which is responsible for the organization of the network. In ad hoc mode, the participants connect spontaneously with each other and take on the administration of the network themselves. If more than two participants are supposed to communicate via an ad hoc network, special routing protocols allow the creation of the MANET [1]. Nodes which have no direct link to other nodes (due to being out of radio range) can communicate across intermediate nodes with routing functionality.

2 MITE

Our work is a part of the MANET Intrusion Detection for Tactical Environments (MITE) research project [2]. MITE focuses on developing a prototypical IDS for tactical MANETs, especially for infantry units. Its reference scenario considers a network consisting of 15-20 so-called lightweight nodes (small-scaled devices like UMPCs or PDAs), carried by infantry troops, and one or more fully-equipped nodes (a laptop with sufficient power supply, e. g. in a carrier vehicle). The examined exemplary military mission is a HR

(Hostage Rescue) scenario, and it considers the liberation of a human hostage from an enemy location.

The MANET is used to support the troops with communication and information capabilities, i. e. Command and Control Information Systems (C2IS), voice communication (VoIP), text-based communication, and additional information services.

Due to the sensitivity of the information that is transmitted, stored and processed, high protection demands arise. Beside adequate physical access restriction, authentication and encryption mechanisms, intrusion detection and response capabilities are necessary. Attacks against the tactical MANET were identified and evaluated in [4], such as interference or jamming the wireless communication, physical takeover of an authenticated MANET node, attacks on the MANET routing and forwarding, and conventional attacks on different logical levels of the wireless network.

The MANET-specific attacks are addressed in [2], [3], [4], [5], and [6]. Most of the conventional attacks are already known from wired networks.

3 NETWORK-BASED IDS IN MANETS

Network-based Intrusion Detection Systems (NIDS) allow the detection of attacks on a network or its components. Ideally, the reliable detection of an attack triggers an adequate reaction, such as notifying administrative personnel, or even an automatic response to the attack, such as reconfiguring the network in order to exclude the attacker from further communication. Additionally, NIDS can also detect improper behavior of applications and attacks which use security gaps in applications to gain access to the target host or network [7].

Signature-based NIDS rely on a widely used and successful underlying concept. To detect attacks, network traffic is recorded and analyzed by the IDS. The analysis is performed by matching network traffic with attack signatures which are created by signature developers when an attack occurs for the first time. Signatures can refer to single or different, similar attacks and contain characteristics which clearly identify the attack, a so-called signature pattern.

In our research, we have chosen the publicly available open source IDS Snort ([9], [10]) as an example for a network and signature-based IDS [15]. The Snort signature database consists of several files with each file containing the attack signatures of a certain attack type or attack family. The official Snort signature package is created, tested, and supported by the Sourcefire Vulnerability Research Team (VRT), a commercial company [10]. Moreover, there are other communities which develop signature packages of their own [11].

To guarantee the reliable detection of attacks, the Snort signature database must be kept up-to-date, similar to the signature database of antivirus software. The official VRT signature package is usually updated every few days.

4 CONCEPT OF THE UPDATE MANAGEMENT SYSTEM

A prerequisite for a signature update is a connection to an intranet or the internet, across which the signatures can be transferred towards the network stations within the MANET. Typically, only one connection to such a primary network exists. Moreover, it is assumed that the node which establishes this external connection has more resources than the other devices in the MANET (fully-equipped node). Due to its position in the infrastructure of the network, the role of the *signature update server* can be assigned to this device. The other devices within the network can correspondingly take on the role of the *signature update client*. As shown in Figure 1, the signatures are transferred from the intranet to the server and from there they are distributed to the clients.

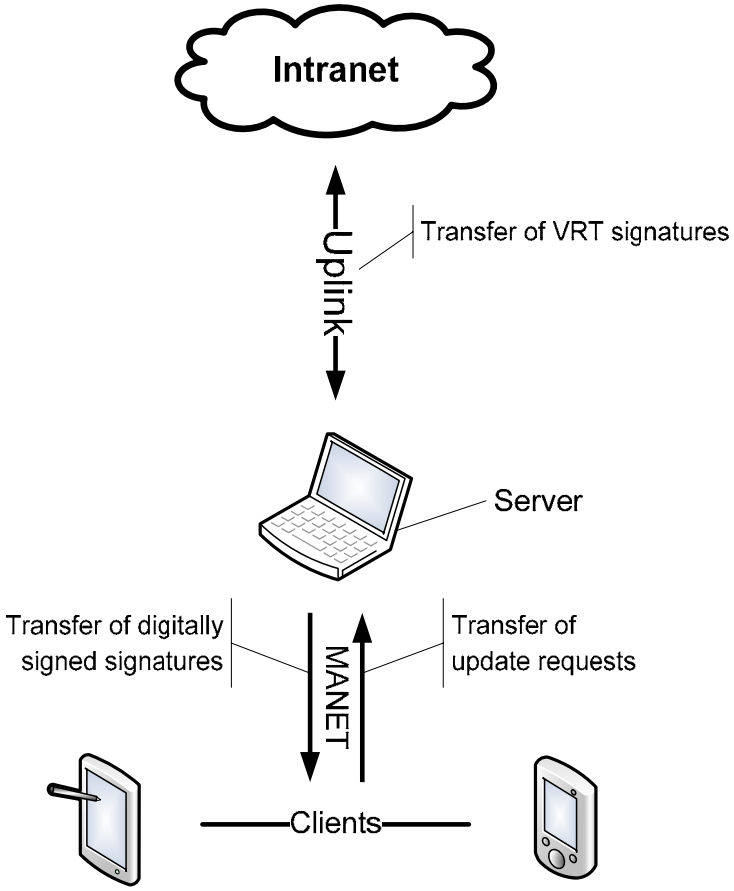

Figure 1: Architecture for the signature distribution in the MANET

The de facto standard for the update of Snort signatures is a software package called Oinkmaster [12]. Oinkmaster is-based on Perl [13] which support download, local storage, and customization of Snort signature packages for various scenarios. Oinkmaster manages signature updates by downloading the complete signature package (e. g. the VRT package with a size of approx. 5 MB) for every update. Whereas these downloads are usually processed in most wired networks without any impact on other components, this reveals that Oinkmaster was not designed for networks with limited bandwidth.

In the reference scenario, the naive usage of Oinkmaster would result in a separate transmission of the signature package to each of the lightweight nodes – which would lead to a significant load on the network, and due to the volatility of the links, it is at least questionable whether the transmissions are

reliable, or need to be repeated several times.

This high load would probably disturb the services (e. g. VoIP) which are potentially essential for the troops supported by the MANET. For this reason, a much more efficient update procedure needed to be designed to keep the load of MANET and clients as low as possible and distribute signature changes in a time-efficient and resource-optimized way. The first idea to reduce the consumption of network bandwidth by signature updates was to perform incremental signature updates.

The analysis of the differences of VRT signature packages during a given time period shows, that an update with an incremental method promises a considerable efficiency profit. An incremental update transfers only changed (added, modified or deleted) signatures to the client.

Table 1: Number of signatures and updates in selected signature packages within a representative time period

package release date	total signatures	updated signatures	difference in days
March 8, 2007	7207	-	
April 16, 2007	7964	757	39
September 4, 2007	8346	3196	141
October 02, 2007	8536	907	28
October 10, 2007	8536	10	8
October 17, 2007	8551	24	7
November 13, 2007	8591	297	27
Total		5191	250

Table 1 shows the evaluation of a count of signature packages selected by chance, which were released over a period of 250 days. The modifications (including addition and deletion of signatures) in the new signature packages compared to the old ones were counted respectively. In arithmetic terms (5191 modifications in 250 days), this results in 21 modifications in 24 hours. Since this value is far below the number of absolute signatures (nearly 8600 on November 13, 2007 with an upward trend), the implementation of an incremental update procedure is worthwhile.

These considerations show the potential of incremental updates. Since the connectivity of each MANET node cannot be predicted, the method for the incremental updates has to take into account that clients might have missed an unknown number of incremental updates in the past. The update method should be able to adapt to this without performing each missing increment sequentially for each client. Moreover each signature affected should be updated in one step to save network bandwidth. This requires keeping track of the update status of each signature of the complete signature set. Each client has to receive an individually assembled update package, depending on its update status in comparison to the current signature base on the server. The following sections describe the method for an update procedure which meets these demands.

5 INCREMENTAL UPDATE PROCEDURE

Using an incremental signature update, a comprehensive version handling is needed. It must ensure that server and client can comprehend all possible modifications of a signature state to the next signature state. This includes the modification, the addition, and the deletion of a signature. To identify signatures clearly and to detect the respective version of the signature, a distinctive identification and a version state, the attributes Snort ID (SID) and Revision (REV) are contained in every signature.

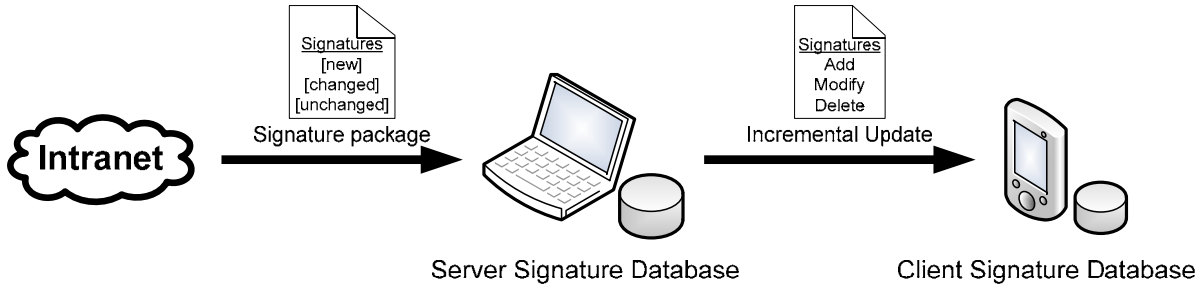

Figure 2: Total context for version handling, with intranet, server and client

Different signature states can occur, as figure 2 shows, in two situations:

- At the server

The server distributes the signatures to the clients. Thus, it has to manage a database of signatures. If this database is updated with new signatures, the differences mentioned above arise. Note that if a signature is no longer contained in the new signature package, it must have been deleted.

- At the client

The Snort signatures of the client are updated continuously. Therefore, the differences between the new signature state of the server and the old signature state of the client arise again.

The signature database of the server is amended by recording the chronological sequence of all modifications of the signatures. This is implemented by consecutive, distinctive serial numbers assigned to each signature ID as a time stamp substitute, as the real alteration time is not relevant for our update procedure. Table 2 shows an example for the allocation of serial numbers (SN) when the SIDs were inserted from left to right in the database.

Table 2: Example for the allocation of SIDs and SNs

max. SN	5				
SID	226	227	236	1854	1855
REV	7	7	7	8	8
SN	1	2	3	4	5
Signature	alert [...]	alert [...]	alert [...]	alert [...]	alert [...]

On reception of a new signature package, the server checks the modifications to the existing signature package via SID and REV. Signatures which were changed, are replaced by the new version. The old highest serial number is increased by 1 and assigned to the corresponding signature. The old highest serial number remains assigned to the signature changed before. When adding a new signature, a new serial number is assigned too. The same happens when deleting a signature, in which the signature itself can be removed. However, the information about the deletion of the signature must be saved, so it can be removed on the client.

6 UPDATE REQUESTS

To keep the signature database of the client up-to-date, it must contact the server continuously. The client sends its maximum serial number to the server. If the server has the same maximum serial number, no update is necessary. If the server has a higher maximum serial number, it can identify and send exactly the signatures, which have changed since the last state of this particular client. The increased computation

effort, which is necessary to compare the old signature package with the new one, remains at the server. The server also carries the computation effort for finding the signatures required by the client to reach the current signature state. The network is charged only in such a way as it is necessary, since only the signatures that the client needs will be transferred. There is little overhead left.

7 TRANSMISSION

Starting point for the distribution of the signature packages to the MANET nodes is the source of the signature package (e. g. intranet) in the wide area network. Thus, the package has to be downloaded to the update management server first. The procedure for incremental updates described in the previous section minimizes the network traffic as far as possible.

Since connections can be frequently interrupted in a MANET due to the characteristics of the network, the Transmission Control Protocol (TCP) on layer 4 of the IP protocol stack is not a good choice for the transfer of the signatures. TCP creates avoidable overhead by retransmissions as well as renewed connection set-up at a connection abort. This implies that many protocols, services or programs, which use TCP to transmit data, should not be used.

The remaining and obvious alternative is UDP. As it does not establish any firm connection with the communication partner and does not check whether a packet has arrived. The UDP packets are simply sent without advance notice after each other. Therefore the header of UDP with its 8 bytes is considerably smaller than the 20 byte TCP header (without options). Protocols like the Real-Time Transport Protocol (RTP [14]) – frequently used for e. g. IP telephony – use UDP as underlying transport protocol as well. For this use case, a missing packet would merely cause a short interfering noise in the audio stream. In contrast to this, the signature data has to be transferred completely and precisely. Missing parts would make the signature or the complete signature file become by Snort. Thus, connection control and reliability of the signature transmission via UDP must be ensured on the application layer of the SUMS. In our scenario we implement this by using a counter: The server answers an update request of a client with the expected number of signatures to be transferred. During the transmission, an index is added to every signature. By this, the client can detect, which signatures are missing after the completed transmission. The missing signatures can be re-requested separately at the server, if necessary. Otherwise the complete update process can be repeated.

8 SECURING THE TRANSMISSION

During the transmission of the attack signatures the authenticity of the signature server as well as the integrity of the transferred signature packages have to be ensured. An attacker, who pretends to be the server, might distribute wrong or faulty signatures and could suspend Snort on all devices in the network. In addition, a so-called insider could represent a danger by distributing manipulated signatures. Confidentiality of the packages is considered unnecessary, because signatures of Snort are not typically sensitive data. The computing power which would be necessary for the encrypted transmission of the signatures can therefore be saved.

By means of a digital signature, both authenticity of the server and integrity of the transferred signature packages can be guaranteed. Only the correct server owns the private key to sign the data, which is only verifiable with the appropriate public key. The server authenticates by digitally signing every data packet sent to the client separately. By this, the integrity of the data packet can be guaranteed too. Upon receipt, the client first checks the signature and then continues with the processing of the packet. In this way, a wrong sender (server) as well as manipulations of the packet can be detected. Only a packet which is not modified and signed with the private key of the server can be verified with the public key of the client. The keys used for this secure transmission have to be stored under strong protection. This also applies to

the public key of the client.

An authentication of the clients was omitted. Since confidentiality was not identified as a security requirement, unauthenticated clients are allowed to request signatures. If client authenticity were required and implemented on SUMS application level, a key management would be necessary, since the public keys of all clients must be stored on the server. Adding one client to the MANET would cause an update of the key database on the server. This would be in conflict with the idea of a MANET and its volatile structure.

9 RESULTS

Figure 3 illustrates the whole context of the developed update system including the architecture of the network and assignment of tasks for server and clients.

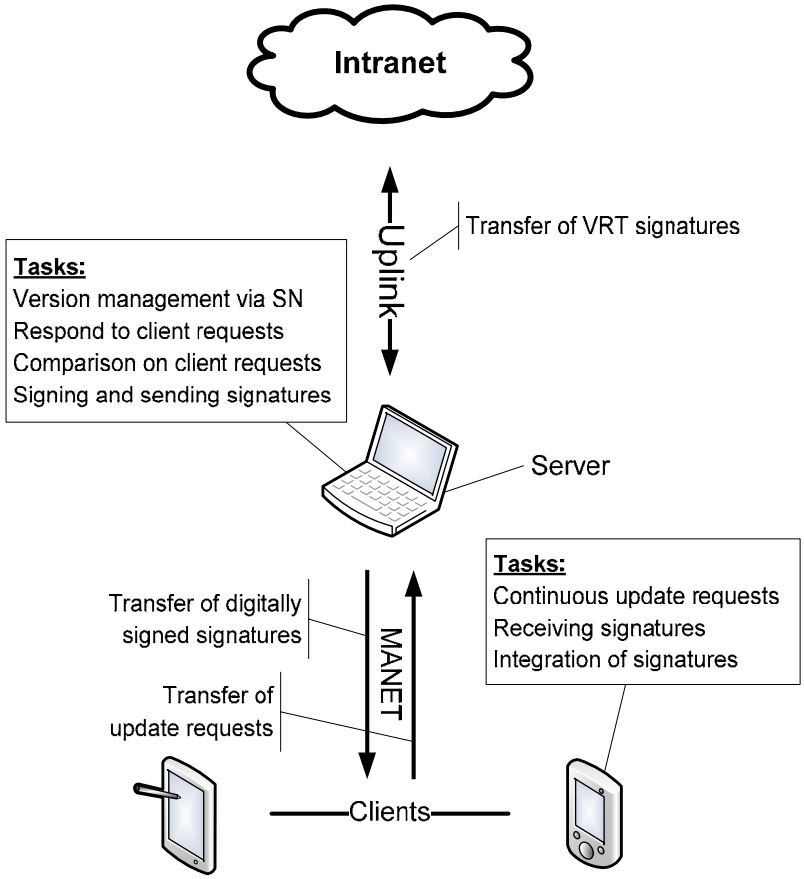

Figure 3. Architecture and assignment of tasks for the signature distribution in the MANET.

By means of examples, a comparison can give an approximate impression of the increased efficiency which can be reached by the SUMS with the following assumptions:

- Average size of a signature: 900 bytes
- Number of already existing signatures: 500
- No connection abort takes place.

- Transmission volume:
  - Contains TCP and UDP header as well as connection establishment and connection termination.
  - Oinkmaster also transfers only signatures (no comments, documentation or other files which are usually contained in the signature package)
  - Overhead by the application protocol (e. g. https ~ 1-2 kb) is not taken into account at Oinkmaster
  - Also contains the digital signature and control information (e. g. request of the client) for the program flow at the SUMS
  - For SID, Index etc. the maximum size was calculated at the SUMS (integer > string = 10 bytes)

Table 1. Changing 50 signatures

Update-System	Oinkmaster	SUMS
Transmission volume	479 kB	60 kB

Table 4. Deleting 1 signature

Update-System	Oinkmaster	SUMS
Transmission volume	478 kB	0,5644 kB

Table 5. Adding 50 signatures

Update-System	Oinkmaster	SUMS
Transmission volume	527 kB	0,0585 kB

Table 6. Changing 500 signatures

Update-System	Oinkmaster	SUMS
Transmission volume	479 kB	579 kB

The overhead for the SUMS in Table 6 results from the digital signature (256 byte) which is transferred with every signature. This example equates to an initialization of the signature database of the client.

10 CONCLUSION

In this contribution, the Signature Update Management System (SUMS) has been presented. It implements an efficient method for signature updates of the open-source IDS Snort, especially in low-resource environments, such as tactical MANETs. Particularly the consumption of resources of the client component, the minimization of the network load, as well as reliability and security of the complete system were taken into account.

The computation effort for the version management as well as for comparing the clients' and server's signature databases remains at the server. In addition, the network is charged only with the signatures which need to be transferred to the client. With the usage of UDP as a transport protocol, the transmission of signatures is adapted to the restrictions given in a MANET environment. An additional overhead of the transport protocol is avoided. The reliability and security of the transmission is guaranteed by application logic and usage of digital signatures. SUMS test results in the MITE reference scenario reveal a fast, secure and resource-optimized way for delivering Snort signature updates to MANET clients.

11 REFERENCES

[1] IETF MANET Working Group. *Mobile Ad-hoc Networks (MANET).* http://www.ianchak.com/manet, April 18, 2008.

[2] M. Jahnke et al. *Research Project Documentation for MITE - MANET Intrusion Detection for Tactical Environments, Project Reference E/IB1S/5A779/2F005.* Federal Office for Information Management and Information Technology of the German Forces (ITAmtBw), Germany 2005-2007.

[3] J. Tölle. *Intrusion Detection durch strukturbasierte Erkennung von Anomalien im Netzverkehr*. PhD thesis, University Bonn, Germany. GCA-Verlag, 2002.

[4] A. Wenzel. *Sensorik für Intrusion-Detection-Systeme in mobilen Ad-Hoc-Netzen*. Diploma thesis, Cologne University of Applied Sciences, Cologne, Germany, 2006.

[5] S. Marti et al. *Mitigating routing misbehavior in mobile ad hoc networks*. Proceedings of the 6th annual international conference on Mobile computing and networking, International Conference on Mobile Computing and Networking, Boston, Massachusetts, United States, 2000.

[6] M. Jahnke et al. *Methodologies and Frameworks for Testing IDS in Adhoc Networks*. Proceedings of the 3rd ACM workshop on QoS and security for wireless and mobile networks, Q2SWinet 2007, Chania, Crete Island, Greece, 2007.

[7] A.Wenzel et al. *Verteiltes Packet-Sniffing als Sicherheitswerkzeug in MANETs*. Proceedings of the D*A*CH Security 2007, Klagenfurt, Austria, 2007.

[8] Sourcefire Vulnerability Research Team (VRT) http://www.snort.org/vrt/

[9] M. Roesch. Snort – *Lightweight Intrusion Detection for Networks*. Proceedings of the 13th Conference on Systems Administration (LISA-99), Seattle, Washington, USA, 1999.

[10] Sourcefire, Inc. *Snort - Open Source Network Intrusion Prevention System*, http://www.snort.org, April 18, 2008.

[11] Bleeding Edge Threats. *Bleeding Edge Ruleset.* http://doc.bleedingthreats.net/bin/view/Main/AllRulesets, April 18, 2008.

[12] Andreas Oestling. *Oinkmaster - Script for updating and managing Snort-signatures* http://oinkmaster.sourceforge.net, April 18, 2008.

[13] The Perl Foundation. *Perl, free platform independent script and programming language* http://www.perl.org, April 18, 2008.

[14] H. Schulzrinne et al. *RTP: A Transport Protocol for Real-Time Applications*. http://www.ietf.org/rfc/rfc3550.txt, Internet Engineering Task Force, RFC3550, 2003.

[15] J.Haag, S.Karsch. *Optimized Sensors for Intrusion Detection in Mobile Ad-Hoc Networks*. Proceedings of the Military Communications and Information Systems Conference (MCC), Bonn, Germany, 2007.