USING LOCAL NETWORK AUDIT SENSORS AS DATA SOURCES FOR INTRUSION DETECTION

Daniel Hamburg [*,1] **York Tüchelmann** [*]

[*] *Integrated Information Systems Group, Ruhr University Bochum, Germany*

Abstract: The increase of attacks on public and private networks led to the development of various methods to detect and prevent those attacks. To detect attacks, network traffic has to be captured and analyzed.
We propose a new sensor framework to capture network packets directly on a monitored host. The framework consists of three sensors, each capturing relevant parts of a packet at different levels of the network stack.
The sensors do not impose any special purpose detection engine thus allowing the integration into existing Intrusion Detection systems.

Keywords: Multi-Layer Sensors, Intrusion Detection, Intrusion Prevention, Network Audit, Network Security

1. INTRODUCTION

Computer systems and networks -significant in the context of infrastructure and work flows in modern companies- require reliable services and strong security functions to protect sensitive data from attackers.

In the last years different technologies securing computer systems and networks were developed, targeting different attacks. Most of those security systems protect a system or network from external attackers who try to compromise the system or reveal and alter sensitive data.

The inability of these systems to protect computer systems from malicious insiders or outsiders who have gained access on an internal system led to the development of additional mechanisms, such as Intrusion Detection Systems (IDS).

An IDS captures data and analyzes it to detect attacks on the network or a single computer. When it detects a malign event it generates an alarm. According to the type of data captured by the IDS sensors, IDS can be grouped in two categories. *Host based intrusion detection systems* (HIDS) capture local system data e.g. access logs of a supervised host. *Network intrusion detection systems* (NIDS) capture network packets of a network segment. While HIDS are used to monitor a single host NIDS are employed to protect whole networks.

Another commonly used criterion to group IDS bases on analysis methods namely *pattern-based* and *anomaly-based* IDS. Pattern-based IDS use specific patterns for every separate attack. However on the one hand this method leads to poor performance in case of a large attack pattern list, on the other hand to a reduced amount of false positives. False positive denotes an alarm generated by the IDS when there was actually no attack. Anomaly-based systems try to detect deviations of the actual system state. While those

[1] Many thanks to Sebastian Gajek for his very helpful suggestions.

systems are highly performing, they tend to generate high amounts of false positives.

Initially, IDS where designed only to generate alarms after detecting an attack. Meanwhile new systems emerged, which try to block malicious network traffic. Those systems are called Intrusion Prevention Systems (IPS).

However the effectiveness of all detection systems depends on the captured data. In this paper we introduce the use of local multi layer sensors to audit local network data. Our framework consists of three sensors that capture data inside the network stack of the monitored host. The sensors capture only significant parts of a network packet and are construed to send them immediately to a detection engine enabling a fast analysis and in case of an IPS a fast response.

The sensors can be seen as an effort to close the gap between HIDS and NIDS. Like a HIDS they reside on the monitored host, but contrary to analyzing log flies, like NIDS they analyze incoming and outgoing network traffic. While the combination of HIDS and NIDS features has been proposed in related works, the way our sensors capture traffic is new and helps to improve the performance of the IDS.

The sensors do not need any additional hard- or software but can be configured to cooperate with existing detection engines. Though we enable a faster and also more precise detection of attacks.

This paper is outlined as follows: In Section 2, we introduce the Multi-Layer Sensors framework. Section 3 gives a review of related works. Finally, we conclude our work and point out future work in Section 4.

2. USING MULTI-LAYER SENSORS TO CAPTURE NETWORK TRAFFIC

Traversing the network stack, each layer of the network stack alters packet data. This altering means more than just removing the corresponding headers. For example fragmented IP packets are put together on the Internet layer before they are handled to the Transport layer. Moreover IPSec packets must be decrypted inside the Internet layer before cleartext data is sent to Transport layer. Different actions are performed on different layers and different OS and applications may act varying on the same type of packets. Regarding the different behavior of systems Ptacek et. al. (Ptacek and Newsham, 1998) made the following statment

> " A packet, by itself, is not as significant to the system as the manner in which the machine receiving that packet behaves after processing it. "

The basic idea of our sensor framework is to capture the header and payload data exactly before it is processed by the OS or the application.

2.1 Network stack and sensor placement

Figure 1 shows a network stack and the placement of the Multi-Layer Sensors (MLS) and of standard NIDS sensors.

Let us take a closer look at the data that each sensor captures. For sake of simplicity we will deal only with packets that arrive at the monitored host. [2]

A standard NIDS sensor captures packets before any header data is processed. When a packet arrives at the host, the NIC produces a copy of the packet and sends it to the NIDS sensor. Standard NIDS sensors are able to capture the data contained in the headers of all network layers and the application payload.

In comparison to standard NIDS sensors, our Multi-Layer Sensors (MLS) consist of three different sensors installed on the monitored host, named Transport Layer Sensor (TLS), Application Layer sensor (ALS) and Application sensor (AS).

Transport Layer Sensor (TLS) captures data inside the Internet Layer, that is after IP fragments have been reassembled and before the Internet Layer performs any other task. Further, our TLS does not capture all packet data, but only the Internet- and Transport Layer headers. Application Layer headers and application payload is ignored.

Application Layer sensor (ALS) captures packets at the interface between the operating system and the application. Note that some HIDS use network sensors on the same network stack level. In contrast to them, our sensor does not capture the application payload but only the Application Layer header.

Application sensor (AS) captures the application payload.

After capturing data, sensor immediately sends data to the detection engine. There is no need for a sensor to wait until the sensors at higher network layers captured the same packet. The detection engine analyzes data of the sensors independent from each other but should also offer the possibility to correlate the data coming from

[2] Our sensors are also able to capture packets that leave the monitored hosts.

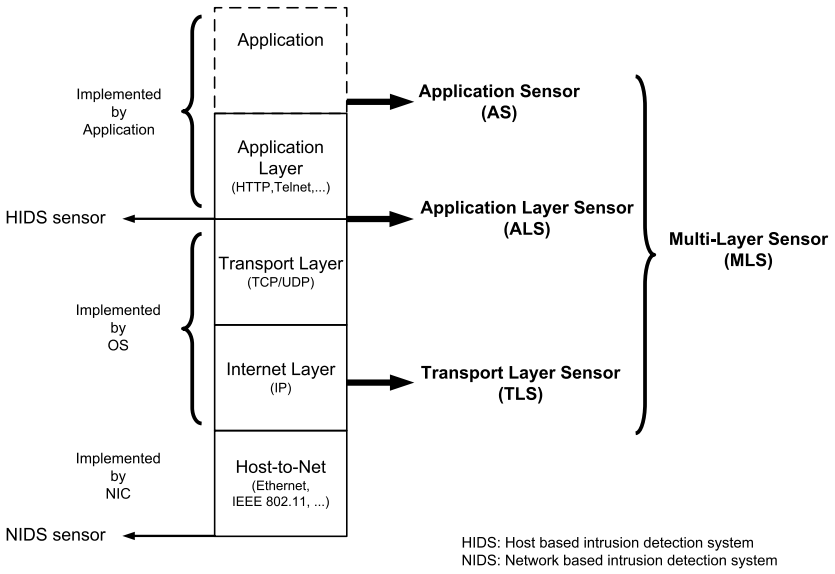

Fig. 1. Network stack and sensors

different sensors to detect more complex attacks offline.

2.2 MLS/TLS vs. NIDS sensor

Standard NIDS capture packets consecutively after they are picked up from the network by the network interface card (NIC). The packets contain data and headers of all layers. To detect an attack that the monitored system is vulnerable to the NIDS has to know how the network stack of this particular host handles the packet. With this knowledge, the NIDS can simulate the behavior of the host and detect -in case of an IPS also avert- attacks that harm the monitored system. For example the NIDS has to simulate how the monitored host combines IP-fragments to reconstruct the initial IP packet. If the NIDS does not exactly simulate the host's behavior, it may miss some attacks because of a different reassemble of IP fragments. (Ptacek and Newsham, 1998) gives some examples how the phenomenon of IP fragment handling can be used to bypass detection systems.

The requirement to simulate the behavior of the monitored host results from the need to know the exact behavior of the host's network stack. It is nearly impossible for a NIDS to handle the amount of simulation data if it monitors a large number of heterogeneous hosts. Because of the resource intensive task of simulating the behavior of different hosts, the NIDS fails to offer realtime attack detection. In addition it is not possible to exactly predict the behavior of systems which are closed source.

Another problem occurs if the NIDS operates in prevention mode. When a packet reaches the host the NIPS has to take a decision whether to forward the packet to the system or to drop it. This decision can be taken only after the host's behavior has been simulated and analyzed to detect possible attacks. Because of the time consuming nature of host simulation the decision can be taken only after a delay. This may negatively influence the usability of the monitored host.

Because of this drawbacks, most NIDS do not simulate the system's behavior but apply rules for groups of systems that because of similar software parameters like OS and installed applications are assumed to have similar behavior. This enables the optimization of NIDS resources but implies also higher false alarm rates and the partial miss of attacks.

The MLS captures data after it has been processed by the underlying layers of the stack. Therefore the data captured by our sensors is identical to the one processed by the host's network stack and it is not necessary to simulate the behavior of the system. The TLS e.g. captures data after the reassembly of IP-fragments but before any other actions on the Internet or Transport layer have been performed. Based on this data, the detection engine can detect attacks that use data inside the Internet and Transport layer headers. Operated in IPS mode, the detection engine has to analyze only a part of the packet data at a time, namely the header and payload captured by one of the sensors. When the packet passes the TLS, the detection engine has to analyze only the header of the Internet and Transport Layer header. If the data inside those headers reveals an attack the packet is dropped. If not, the packet is

passed back to the system. This results in a higher performance when operated in prevention mode.

This gain of performance and especially the lack of delay makes the generation of groups summarizing similar systems obsolete which results in a lower false alarm rate and higher amount of detected attacks compared to NIDS.

2.3 MLS/ALS & AS vs. HIDS sensor

A HIDS sensor captures data at the same level of the network stack as the ALS. Because Transport and Internet Layer headers are removed by underlying parts of the network stack, the sensor captures only Application Layer header and payload. Therefore, HIDS misses all attacks that use information of the Internet or Transport Layer [3] .

TLS captures the Internet and Transport Layer header data therefore enabling the detection engine to detect those attacks.

After a packet has been handed over from the operating system to the application, the application layer header is processed. This process may include the reassembly of application payload from different packets into one stream, or even the altering of payload, e.g. the conversion of strings into the character encoding used by the application.

To detect if the payload of a packet constitutes an attack against the monitored host, the HIDS has to recognize the reaction of the monitored host to the payload. To do so the HIDS must know the payload as processed by the application. Therefore it has to simulate the behavior of the host's application layer stack. As mentioned in the context of NIDS, this results in a decreased performance and eventually false alarms if the simulation fails. To circumvent this we decided to divide the capturing of Application Layer header and application payload on two sensors, namely ALS and AS.

ALS is used to detect attacks that use Application Layer header data, e.g. a HTTP-request string for a server-side script causing a buffer overflow. AS is used to detect attacks that use malicious application payload, e.g. client-side Javascript revealing the user's password.

3. RELATED WORKS

Kerschbaum *et. al.* (Kerschbaum *et al.*, 2000) propose *embedded sensors* inside the operating system and applications on the monitored hosts.

> "Those sensors are pieces of software that monitor a specific variable, activity or condition of a host."

While this approach enables a very exact monitoring of all actions on the host, which goes far beyond the events captured by our MLS, the method involves a very high complexity to design those sensors and adapt existing software. In addition, the use of closed source software makes the use of embedded sensors obsolete. On systems, where the software can be adapted, it is reasonable to implement embedded sensors to monitor events that can not be detected by analyzing network packets.

Zhang and Lee (Zhang and Lee, 2000) recommend the use of a distributed IDS [4] to detect attacks on elements of a wireless ad-hoc network. The authors propose to combine sensors that analyze network data similar to a NIDS sensor, ones that analyze system calls and log files and also sensors that capture parameters about the physical environment of the network elements. The network data analyzed is provided exclusively by the NIDS sensor which results in the problems discussed in Chapter 2.2.

Wu *et. al.* (Wu *et al.*, 2003) proposed a similar idea of applying multiple specialized sensors at different layers. The authors' sensors operate at the network layer capturing packets from the NIC, the kernel layer intercepting certain OS activities (e.g file access, illegal signals) and the application Layer catching calls to a set of library functions. Alerts are collated using a central detection engine, which scans for local and distributed attacks. As incoming network packets are captured between the Host-To-Net and the Internet layer (of the TCP/IP network model) the authors' sensors have same shortcomings as standard NIDS.

The use of sensors that analyze system calls and log flies can be found in both (Zhang and Lee, 2000) and (Wu *et al.*, 2003). This type of sensors are typical for HIDS that do not capture network traffic. Those sensors do not capture network packets that comprise the attack, but the impact on the system caused by this attack. Therefore those sensors in contrast to MLS completely lack the ability to prevent an attack.

4. CONCLUSION AND FUTURE WORK

In this paper we introduce a new sensor framework to capture network packets directly on the monitored host. The framework consists of three sensors which are integrated inside the operating system (Transport Layer Sensor, Application

[3] An enumeration of attacks that use such information inside low level layers can be found in (Daniels and Spafford, 1999).

[4] An IDS without a central detection engine

Layer Sensor) and the application (Application Sensor) thus capturing network packets before they are processed, enabling high performance for analysis and prevention mechanisms. Multi-Layer Sensors (MLS) are a new contribution to the field of network audit, aiming at network traffic of sensitive hosts, e.g. servers.

Using the sensors does not impose any special purpose detection engine because they are used only for data gathering and designed to use standard communication protocols, e.g. the Intrusion Detection Exchange Protocol (Feinstein *et al.*, 2002), to communicate with existing detection engines.

The three sensors capture data independent of one another which enables them to send data to different detection engines. As a result we can use various detection engines to enable load balancing and to omit a single point of failure.

Our future work aims at analyzing encrypted traffic as used with standard security protocols, such as SSH. In addition, we will analyze existing protocols and eventually design a new one to optimize the secure transmission of sensor data to a detection engine and to efficiently coordinate data originating from different sensors on the same host.

REFERENCES

Daniels, Thomas E. and Eugene H. Spafford (1999). Identification of host audit data to detect attacks on low-level IP vulnerabilities. *Journal of Computer Security* **7**(1), 3–3.

Feinstein, B., G. Matthews and J. White (2002). The intrusion detection exchange protocol (idxp). Technical report. Internet draft.

Kerschbaum, Florian, Eugene H. Spafford and Diego Zamboni (2000). Using embedded sensors for detecting network attacks. In: *Proceedings of the 1st ACM Workshop on Intrusion Detection Systems* (Deborah Frincke and Dimitris Gritzalis, Eds.). ACM SIGSAC. CERIAS TR 2000-25.

Ptacek, Thomas H. and Timothy N. Newsham (1998). Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report. Secure Networks, Inc.. Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6.

Wu, Y., B. Foo, Y. Mei and S. Bagchi (2003). Collaborative intrusion detection system (cids): A framework for accurate and efficient ids. In: *Proceeding of 19th Annual Computer Security Applications Conference.*

Zhang, Y. and W. Lee (2000). Intrusion detection in wireless ad hoc networks. *ACM MOBICOM.*