Combining IDS and Honeynet Methods for Improved Detection and Automatic Isolation of Compromised Systems

Stephan Riebach, Birger Toedtmann, Erwin Rathgeb

Computer Networking Technology Group
Institute for Experimental Mathematics
University Duisburg-Essen, Germany
{riebach|btoedtmann|erwin.rathgeb}@iem.uni-due.de

Abstract. The growing penetration of mobile terminals increasingly limits the usefulness and efficiency of classical firewall architectures and, therefore, increases the need for the deployment of Intrusion Detection Systems (IDS). In this contribution a concept is proposed that combines an anomaly-based IDS with mechanisms typically applied in honeynets as well as with additional functions. A first prototype implementation of this concept indicates that it allows to combine the advantages of both approaches in order to improve the accuracy of anomaly detection while at the same time avoiding unnecessary restrictions for the users of suspicious systems as well as minimising the risk for the rest of the production network.

1 Introduction

Despite intensive efforts to achieve security objectives and to actively detect successful attacks in IT infrastructures, a vast number of undetected intrusions with severe impact on system security can be observed. One of the reasons is the drastic reduction in product life cycles of hardware and software due to a highly competitive market. This results in rapid changes of IT infrastructures and, as a consequence, in a continuous introduction of new and previously unknown vulnerabilities. In addition, attackers continuously refine and adapt their methods to new vulnerabilities as well as to the countermeasures developed in response. While significant progress has been made in the area of intrusion detection (see e.g. [1], [2], [3]), the inherent problem of all pattern recognition approaches remains unsolved: there is always a – small but finite -– probability for non-existing, falsely detected and existing, but not detected incidents. Both, the so-called ”misuse” detection applying static filters to decide on the existence of security violations as well as the “anomaly” detection trying to identify deviations from “normal” system or network behaviour are subject to those *false positives* and *false negatives* to a certain degree [3]. Unfortunately, it is exactly this property of IDSs that leads to a negative feedback process in real life application: overwhelmed and annoyed by too many false alarms, network administrators tend to underestimate or overlook serious alarms, and sometimes they even reduce the sensitivity of the IDS. On the other hand, one single undetected intrusion can seriously undermine the confidence in the IDS.

In order to allow fast and reliable identification and analysis of new attack patterns and signatures, the application of so-called honeynets [6] has been proposed. Honeynets are artificial networks (i.e. networks with no real users or traffic) exposing computer systems (honeypots) openly (i.e. without full firewall protection) to attacks in a tightly controlled and monitored environment. Due to their comprehensive traffic and activity logging capabilities, honeynets can be used to gather statistical data on the number and type of attack attempts [7]. In addition they allow in-depth forensic analysis (online and offline) of successful attacks to gain insight into the methods, strategies and motivations of attackers [6], [8]. However, our own experience has shown that operating a honeynet requires a significant effort. Furthermore, the results obtained from a honeynet are not directly usable for intrusion detection purposes. Therefore, the benefits of using honeypots and honeynets to support IDSs in production networks are disputed in the IDS community.

The concept presented in this paper combines mechanisms from the areas of intrusion detection and intrusion response with honeynet mechanisms. The basic idea is to isolate systems generating suspicious (but not yet positively identified as malicious) traffic automatically in a tightly controlled honeynet environment for further observation before making a final decision. During this “quarantine”, harmless traffic from these systems is still forwarded to the production network to allow users to continue working while all potentially harmful traffic is contained within the honeynet. Thus, it is possible to reduce the number of false alarms without generating an unacceptable risk for the production network.

In section 2, the relevant characteristics of IDSs and honeynets will be discussed before presenting the proposed concept and its components in some detail in section 3. Section 4 presents a first prototype implementation which demonstrates the feasibility of the concept. Section 5 provides a summary and an outlook on further work.

2 Features and limitations of IDSs and honeynets

Mobile terminals used both within private (enterprise) networks and outside while at home or travelling can only be partially controlled by the network administration. Therefore, they provide multiple entry points for malware and limit the efficiency of classical firewall concepts. To the same degree, the need to detect the violation of security objectives and to contain their impact increases. Intrusion detection and intrusion response systems are deployed to provide a way for dealing with these conditions by reporting observed incidents. The (additional) use of honeynets and honeypots has also been proposed to investigate new attack types.

These concepts originally developed for completely different purposes have to some extent complementary strengths which can be combined to improve the overall security achieved by these techniques.

2.1 Intrusion detection und intrusion response

Intrusion Detection Systems (IDS) are used to detect malicious attack activities in computer systems or within a computer network. A classification is made distinguishing rule based detection approaches, also called “misuse detection”, and approaches trying to detect deviations from normal operation, also known as ”anomaly detection” [4]. Furthermore, a distinction is made between IDSs monitoring all data traffic in a local network (Network Intrusion Detection System, NIDS) and those checking for manipulations in the software of computer systems (Host Intrusion Detection System, HIDS). A rule based NIDS will thus search for known patterns (signatures), e.g. typical keywords or bit combinations, in data packets sent over the network. Inside a computer system, a HIDS will check for modified, deleted or newly added (system) files and unusual activities – in particular accesses to program libraries -– which could indicate a violation of security objectives [4].

IDSs are per definition passive systems. Their task is alarm generation rather than system protection by data filtering or by repairing the file system. Combining an IDS with such mechanisms yields a so-called “Intrusion Prevention System” (IPS). A problem when applying an IPS is the (unjustified) isolation of computer systems in case of false alarms, which typically leads to long periods of severe usage restrictions due to the human intervention required for correction.

Attack detection by using IDSs is subject to some significant limitations reducing their effectiveness in many real world scenarios. Typically IDSs are not designed to cope with realtime requirements. HIDSs are mostly offline systems performing periodic checks of the file system rather than continuously monitoring system activity. The latter would require severe modifications in the operating system, e.g. replacement of internal system calls like open(). In an offline system, the duration of the check interval obviously limits the realtime capabilities. Although NIDSs continuously monitor network traffic, alarms are typically gathered in log files which have to be evaluated manually by the administrator, thus limiting the capabilities for realtime response. Our measurements have shown that the commonly used NIDS *Snort* [10] needs up to 40 seconds to report an attack signature under heavy network load conditions. In addition, many IDSs such as e.g. the popular Snort or the HIDS *Aide* [11] do not correlate isolated incidents they have detected. As a consequence, every single signature or modification is logged separately resulting in huge log files. It is up to the administrator to evaluate this huge amount of data and to identify and correctly interpret incidents as port scans, execution of shell code or a replaced system file. Detection of multi-phase attacks is not supported by these tools. Furthermore, they provide no means to isolate suspicious systems even if they can be identified. Hence, again human intervention is required to reinstate network integrity.

IDSs applying misuse detection are only able to detect known attacks described by predefined detection rules (signatures) — new and unknown attacks remain undiscovered. As these signatures are the result of intensive offline analysis of attack activities, such a system always lags behind the current threat situation. An alternative are anomaly-based detection systems scanning for unusual network or system activities. Such systems, however, require a significantly higher effort for administration. Especially during the deployment phase when "normal" system and network behaviour is learned by the IDS, a lot of false alarms are typically generated. Subsequent changes in the network, e.g. reconfigurations or the deployment of new applications and users can also cause an increase of false alarms. Furthermore, anomaly detection requires significantly more computing power than rule-based detection and performance is also a concern in heavily loaded networks or systems. The main advantage of anomaly-based systems is their ability to also cope with previously unknown attacks and, therefore, a significant research effort is spent in this area.

2.2 Honeypots und Honeynets

Honeypots are computer systems explicitly deployed to be found, probed and compromised by attackers [6]. This however does not mean that a honeypot is per definition a system which is reachable from the internet and is completely unprotected. In fact a honeypot is generically a security resource used to detect and analyse attack attempts from arbitrary sources. Honeypots are equipped with comprehensive HIDS sensors and logging facilities, e.g. recording keystrokes and registry modifications.

A honeynet is an artificial network including honeypots but typically no production systems. Data traffic in the honeynet is tightly monitored by NIDS sensors. In addition, all network traffic is logged on packet level. Since no production systems have to be protected and no (disturbing) production traffic is present in a honeynet, successful attacks can be allowed in a controlled fashion and a medium to long term observation of attacker activities is possible. Even detailed offline forensic analysis is possible due to the comprehensive traffic logging. Thus, a honeynet allows far more detailed and flexible attack observations than IDSs in production networks, where immediate countermeasures are required.

Honeynets can be operated in parallel with production networks. However, they require a significant amount of resources, not only in terms of hardware but also in terms of administration effort [8]. As a general rule, it makes sense to separate the honeynet completely from the production network to make the forensic analysis of successful attacks more efficient. As a consequence, honeynets usually give no direct indication about intrusions in the production network even if they are co-located.

3 Improved anomaly detection by using automatically configured quarantine networks with honeypots

The proposed scheme attempts to combine the advantages of host and network based IDSs with those of honeynets in order to reduce the probability of false alarms which would cause severe usage restrictions for suspected users and systems. We assume that the majority of computer systems in a typical production network is not protected by a sophisticated HIDS – due to the effort that has to be spent and the expertise that is required -– and, therefore, intrusions on these systems will remain undetected.

The basic idea of our scheme now is to isolate the corresponding computer system fast and automatically if the NIDS reports a suspected intrusion there. However, the system is not fully detached from the production network until a successful attack has been confirmed. Instead, limited usage of production network resources is still allowed as long as there is no severe risk for the other systems. Since the restrictions for the user of the potentially compromised system may be tolerable, the network administration can afford the time for an in depth evaluation of the incident.

The second part of the proposed scheme is to expose (one or more) honeypots that have been configured like a typical production system – and are thus vulnerable to the same exploits – directly to the suspected machine. If one of the honeypots is being compromised successfully, its HIDS will report this, resulting in a positive proof that the suspicion was correct. If, on the other hand, no further attack attempts are being reported, a false alarm can be assumed with sufficient probability and the isolated system can be released back into the production network.

In this way, the main features of honeynets, namely *isolation* of suspicious activities and exposure to *artificial vulnerabilities*, are used to allow for a more comprehensive evaluation of suspicious activities, enhancing intrusion detection accuracy. At the same time, massive disruption of normal operation is avoided.[1]

3.1 Concept overview

Starting from a traditional anomaly-based NIDS, additional functions have to be provided to automatically initiate an in-depth inspection if a suspicious but not clearly malicious activity is detected. The suspicious system has to be partially confined to a quarantine network. There it can interact with the preconfigured honeynets representing the vulnerabilities of the typical production systems in order validate the suspicion. Figure 1 shows the main components of the proposed system, their interaction and basic functionality. A detailed description of the various functions is given in the following subsections.

[1] The principle of isolation and verification of an initial suspicion is also well known from controlling epidemics and identifying criminals, where it is quite successfully applied.

The proposed scheme is incident driven. The triggering incident is an alarm from the NIDS monitoring traffic in the production network which cannot be positively identified as malicious. To minimise reaction time, a "topology monitoring" function proactively collects information on the physical attachment points (switch ports) of all computer systems in the networks. When the mechanism is triggered for a specific computer system, the IP address of the suspicious machine is being identified, the physical port where it is attached to the network is retrieved and the system is moved to the quarantine network by an "isolation/rehabilitation" mechanism located at OSI layer 2 (see section 3.4 for details) – which is transparent to the application. A corresponding packet filter function provides limited access from the quarantine network to the production network such that some preconfigured basic services (e.g. WWW) will remain available in order to limit the impact for the user during the subsequent investigation phase. All other activities which could be related to a successful intrusion are contained within the quarantine network (as in a classical honeynet) such that only the honeypots are exposed to them. If the NIDS in the quarantine network or the HIDSs within the honeypots detect signs of an attack, e.g. illegal file accesses, it is proven that the suspicious system has been compromised. A corresponding feedback to the isolation/rehabilitation function will result in a complete and permanent detachment of the compromised system from the production network. Of course, this action is documented and signalled both to the user of the compromised system and the network administrator. If -– in case of a false alarm -– no attacks on the honeypots can be observed for a preconfigured time period a "quarantine timer" function will initiate the release of the falsely suspected system.

3.2 Anomaly detection with two thresholds

The approach chosen for our scheme is based on the anomaly-based search for intrusions and exploitation of vulnerabilities. Therefore, a NIDS sensor is assumed to monitor unusual activities in the production network and to document them in a log file.

A commonly used scheme for anomaly detection is to perform a training phase to gather information about "normal" system operation. The relative frequency with which a specific activity x occurs is measured and used as an estimate for the probability $P(x)$ of its occurrence. A specific inverse function such as

$$A(x) = -log_2(P(x))$$

yields the anomaly indicators – which are obviously high for unexpected events and low for expected ones. As an example, anomaly indicators can be calculated for combinations of IP addresses and TCP destination ports in data packets. The training phase for an email server with IP address 10.0.0.1 might result in the probabilities $P(10.0.0.1; 25) = 0.9$ and $P(10.0.0.1; 80) = 0.1$ (a mail server is usually not a WWW server). Therefore, the anomaly indicator $A(x)$ of an email packet (using destination port 25) to this server would be roughly 0.15, whereas the anomaly indicator for a http packet (port 80) to the same server would be 3.32 [13].

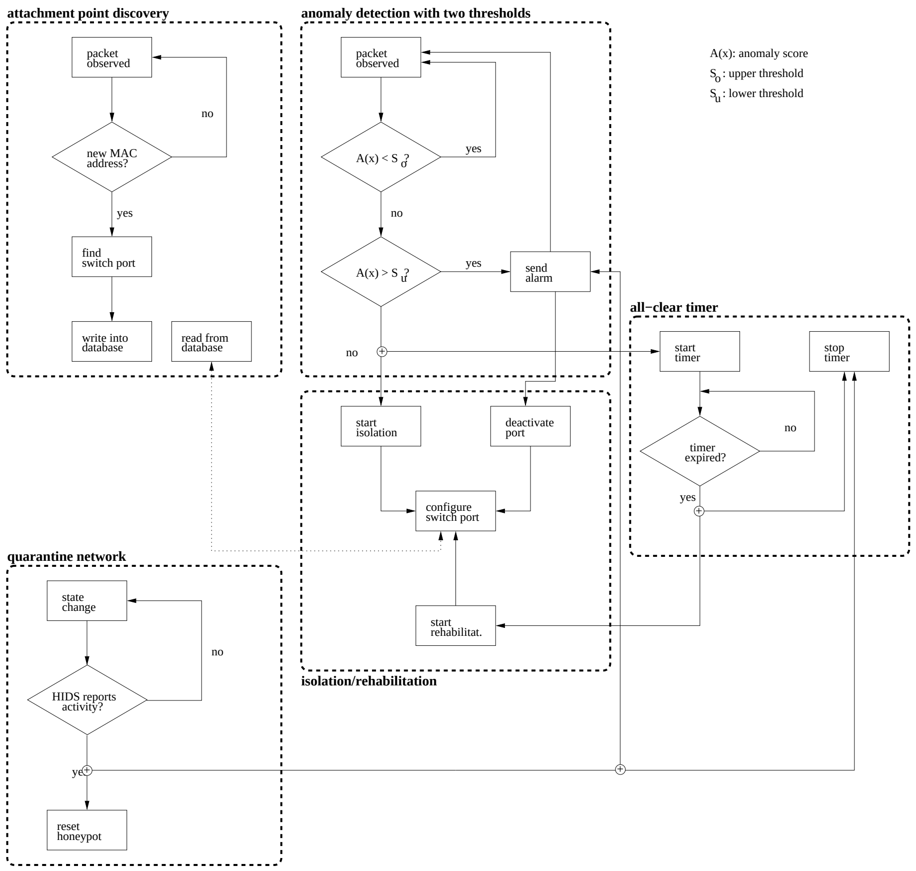

Fig. 1. Main functional blocks, sequence of operation diagram

To be able to identify and report a significant deviation from normal operation, the definition of a threshold by the network administrator is necessary. This threshold value is consequently also responsible for the rate of false positives and false negatives: if it is set too low, a massive number of false alarms is generated. If it is set too high, the system becomes insensitive and real attack signatures will remain undetected.

To adapt this mechanism to our scheme, the anomaly assessment of the detectors is modified to take into account *two* thresholds that distinguish three areas:

- Normal: Anomaly indicator below lower threshold, event is definitely rated as normal activity
- Suspicious: Anomaly indicator between lower and upper threshold, further observation in the quarantine network is initiated.
- Alarm: Anomaly indicator above upper threshold, event is definitely rated as dangerous, countermeasures (e.g. shutdown of the switch port) are immediately initiated

This function is permanently active and performs a continuous monitoring of the network traffic. Of course, more sophisticated algorithms for anomaly detection can be used as well. Since the definition of a second threshold only changes the interpretation of the computational result, the algorithm as such does not need to be modified.

3.3 Topology monitoring

Most applications in today's networks are IP based, i.e. they expect stable IP parameters like source and destination IP addresses, transport protocol type and source and destination port number. The protocol parameters of the underlying (OSI layer 2) protocols, however, are normally not visible for the application. This means that, e.g. MAC addresses can be modified without major impact on active applications.[2]

This property is used in our approach to perform the isolation of a sucpicious system in realtime and (as far as possible) transparent for running applications. For this VLANs (Virtual LANs) are used which can also be maintained in complex, hierarchical switched-LAN topologies if IEEE standard 802.1q is applied. As a prerequisite for rapid isolation, the correct physical attachment port of the suspicious system has to be identified. Although the IP addresses related to suspicious traffic are in principle known to the IDS, two problems remain:

- The MAC addresses are not necessarily known, because the suspicious system might be located in a different broadcast domain.
- The reported IP address is not necessarily the correct one because it might be manipulated (spoofed) by the compromised system.

Another problem is that a reactive scheme starting to locate the port only after suspicious traffic has been detected doesn't react fast enough. By using the Simple Network Management protocol (SNMP [15]), the information for reliably locating the physical access port can be collected automatically from all network nodes. As the query and response process as well as the correlation of the resulting data takes time, we propose to apply a proactive approach where all systems have been already tracked down when they generated traffic for the first time after being attached to the network. The initial packets are traced back to their origin by first establishing the IP-to-MAC address mapping (by monitoring

[2] If ARP caching is used, some temporary performance degradation may occur.

the Address Resolution Protocol, ARP) and subsequently polling the Bridge-MIBs [18] of all switches via SNMP to derive the MAC-to-port mapping. Thus a complete map identifying the physical attachment port of all active systems in the network can be generated and stored. The topology monitoring function is continuously active and updates the system location map dynamically.

In case several systems are connected to the same port via a hub, the whole segment attached to the port is isolated if necessary. If authentication mechanisms on OSI Layer 2, e.g. IEEE 802.1x [20] and EAP [19], are used in the network, locating the systems is even simpler because the relevant information is explicitly exchanged in these protocols.

3.4 Isolation and Rehabilitation

The purpose of this functional block is to isolate suspicious systems from the production network (triggered by the request from the IDS) by moving them to the quarantine network by means of a preconfigured VLAN. A prerequisite is that all switches are IEEE 802.1q enabled, i.e. all switches are able to recognise the VLAN tags and will process them accordingly. A special quarantine VLAN configuration exists that ensures that all data packets from an isolated system are tagged accordingly forwarded based on these VLAN tags to the central monitoring system. From there they are relayed either to the quarantine network or the production network, depending on preconfigured forwarding and filtering rules.

Once the isolation process is triggered, the switch where the suspicious system is attached to and its physical port number within the switch are retrieved from the system location map. In the next step, the switch is instructed via SNMP to move the suspicious system from the default VLAN — where the production network is located — to the quarantine VLAN.

Depending on the result of the investigations carried out during the confinement, the isolation/rehabilitation function is instructed to either disconnect the suspicious system completely from the production network by shutting down the physical switch port, or to reconnect it to the production network by moving it to back the default VLAN.

As a result, we establish three distinct network partitions which are connected to the central monitoring system:

1. the production network,
2. the suspicious system separated from the production network by the VLAN and
3. the quarantine network.

A filter logic there defines how the packets are forwarded:

– All three network partitions are members of a common bridge such that the partitioning is transparent for the applications.
– Traffic originating from the suspicious system that has been classified as harmless based on predefined rules -– e.g. access to web pages —is passed to the production network.

– All other (potentially dangerous) traffic generated by the suspicious system is diverted to the quarantine network where it can reach only the honeypots.
– Traffic from both the production and the quarantine network destined to the suspicious system is delivered there.
– *All* traffic between the production and the quarantine network is blocked.

In such a scenario, a basic service can be provided to the isolated system during the quarantine period. The potential risk for the production network emanating from the applications classified as harmless can be further minimised by using a local Intrusion Prevention System, e.g. Snort_inline [12]. With this extension, Snort can modify the monitored traffic in such a way that specific attack patterns are normalized.[3]

This function block is being triggered by the various sensor blocks if systems have to be isolated, blocked or reconnected.

3.5 Quarantine network

In the quarantine network -– which is essentially a honeynet -– there are only honeypots which are fully exposed to the suspicious systems under observation. The honeypots represent typical systems similar to those found in the production network rather than arbitrarily weak systems. This reflects the actual threat level for the production network.

Due to their sophisticated HIDS sensors, the honeypots are able to detect intrusions quickly and reliably and will report them to the central monitoring system. It is crucial for the effectiveness of the overall system that the HIDS should operate with a low latency, i.e. rather than only performing periodic checks of MD5 checksums, tighter monitoring of local process activity, disk and (system) file access is necessary. However, since there are no real users on the honeypots, this is not really problematic as any activity going beyond the usual internal system maintenance should be flagged as intentional malicious. These techniques provide the quarantine network and its honeypots with examination and observation capabilities way beyond those feasible in the production network.

The quarantine network is preconfigured and continuously active. However, it has to be made sure that compromised honeypots are restarted with a clean configuration once the isolated system has been removed.

3.6 Quarantine timer

The partial isolation of suspicious systems is controlled by a timer which is preconfigured by the network administrator to balance out the detection accuracy and the restrictions imposed on the isolated systems. If no attack activities are detected during this time period, it is assumed that the suspicion was unsubstantiated. The system is moved back to the production network in this case and, thus, fully rehabilitated.

[3] This makes sense especially if email is still allowed, to avoid spreading of viruses.

4 Prototype implementation

We decided to set up a first prototype for the concept described above with a Linux system that serves as the IDS as well as the host for the quarantine network. The production network was built from two Cisco switches and several WindowsXP systems. The switches were configured to support 802.1q VLAN tagging on their uplinks, which are then by definition trunk ports. All other ports are usually members of the production VLAN. We could not use the default VLAN as the Linux system will not bridge between a tagging and non-tagging interface if they use the same physical device. We therefore configured the VLAN id "2" for this network. For the quarantine VLAN we used id "3", and under normal circumstances no switch port is member of that VLAN. Within the Linux system, we configured two pseudo-interfaces, `eth0.2` and `eth0.3`, to connect to both VLANs, while the physical device `eth0` was plugged into a trunk port on the topmost switch. As IDS software we used *Snort*. In order to put Snort into an anomaly detection mode, we installed the plugin *Spade* (Statistical Packet Anomaly Detection Engine) [10], [13]. Normally Spade would have been extended to support two thresholds instead of one,[4] however, for the prototype we simply used Spade's single threshold as the lower bound. Thus we had no upper bound which could indicate a "definite incident" but had rather a huge margin of suspicion. This means that when the score Spade calculates for a specific observed packet is above the threshold configured, our isolation procedure was triggered.

The honeynet that is part of the quarantine network was reduced to a single honeypot, for which we simply used a representative WindowsXP installed as guest operating system inside a VMware [21]. The WindowsXP was configured just as the client systems in our production network and it is automatically started inside the Linux system when it boots up. The honeypot's network interface inside the hosting Linux system, `vmnet1`, was bridged via the *brctl* utility to both `eth0.2` and `eth0.3`. Because the honeypot should not be visible to the production network, we used the link layer filtering capabilities of Linux and its accompanying utility *ebtables* to drop all packets destined to or coming from the Ethernet MAC address of the VMware guest when they try to pass `eth0.2`.

For the detection of newly connected systems we used the program *arpwatch* [14]. This utility allows to trigger external programs via the switch "-s" (originally intended to invoke a sendmail process) which is employed in our prototype to start a small script that searches for the physical switch port of the new system. Here we simply issue three SNMPv3 "get" requests: the first extracts from the port table of the Bridge-MIB, `dot1dTpFdbPort`, the bridge port of the system with the observed MAC address. This has then through a second request to be mapped to the `ifIndex` of the Interface-MIB by retrieving `dot1dBasePortIfIndex` from the Bridge-MIB of the switch. The third request extracts the VLAN of the interface we found by reading `vmVlan` from Cisco's VLAN-MEMBETSHIP-MIB. If no membership exists, we know it is the trunk

[4] Work has been started to implement this feature in Spade.

port of the switch and can safely discard the results as the physically attached port will be on another switch in the hierarchy. At worst we have to issue three requests and wait for three responses for all switches in the network to obtain a valid result.[5] We then save the information `switch-ip:if-index` for later use by the isolation/rehabilitation function in a file with the MAC address as file name.

When Snort flags a suspicious system, it calls the isolation/rehabilitation function, which is a script in our prototype. This looks up the switch and the port of the offender and moves the system into the quarantine VLAN by sending a single SNMPv3 "set" request to the switch which contains `vmVlan.[ifIndex]` as OID and the new VLAN id "3" as value. The performance of this operation depends on the speed of the network and is usually completed within milliseconds as it takes almost no time at the switch.

We now ensured that upon Snort issuing the first alert (under slight network load this is done between 1-2 seconds after observing the suspicious activity), our mechanisms isolated the corresponding system very fast and efficiently. To enforce the confinement, we configured *ebtables* in such a way that only WWW traffic is allowed to pass from `eth0.3` to `eth0.2`, thus leaving the quarantine network and entering the production network. All other packets were manipulated by the destination network address translation mechanisms (DNAT) Linux offers: we set the destination MAC and IP addresses to be those of the VMware-based honeypot. This allowed us to safely bridge all traffic from `eth0.3` to `vmnet1`, to which the honeypot is connected, without reconfiguring the network of it all the time. On the other hand, when the honeypot now sends response packets back to the suspicious system, we can switch the source IP addresses back by using Linux' connection tracking mechanism, however, the source MAC addresses cannot be reinstated. This will not affect the applications running on the systems but may serve malicious code to detect our detour to the honeypot.

The last building block of our system was the development of a fast HIDS for the honeypot. As this was VMware-based, we chose to utilize VMware's methods to make disk images non-persistent, which means that changes to the disk image made by the guest operation system will not be written to the original file but to a so-called REDO file. This not only provided us with an efficient way to clean a compromised honeypot very easyly, it also made finding unauthorized file access possible. In order to accomplish this, we had to install the WindowsXP that serves as the guest system into a FAT32 filesystem because the NTFS filesystem is not very suitable for finding differences. In fact VMware writes single sectors that have been changed by the guest to the REDO file, thus it contains not a filesystem but only small parts of it. With FAT32 it is nevertheless possible to observe the names of newly created files within the guest system. We tested this by periodically making a copy of the REDO file and letting the utility *xdelta*

[5] This process can also be parallelized. As a consequence, finding the port through which a system attaches to the network can be done in under a second. The performance depends mostly on the network's speed as the requests take almost no calculation time within the switches.

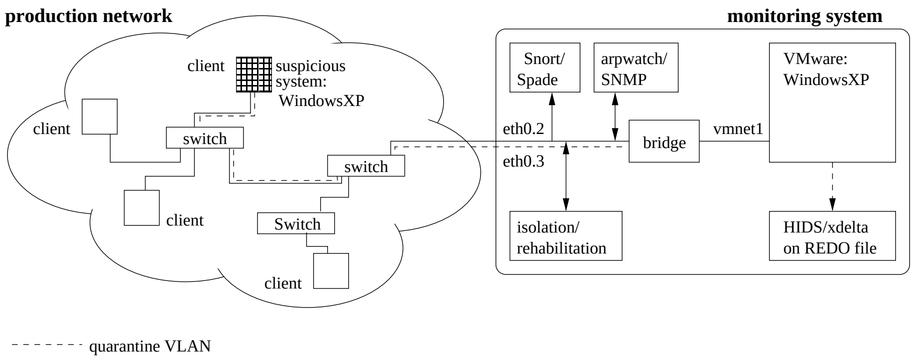

Fig. 2. Prototype implementation and testing scenario

compare both REDO files after 10 seconds. In first tests we were able to detect the creation of the file “explorer.exe” by the MyDoom.B virus within the system directory. Thus we had built a small but effective HIDS that reports suspicious file creation just in time.

The prototype has just been built and is being tested, therefore no detailed experimentation results can be given right now. Also the feedback mechanism for the rehabilitation or disconnection of the offending system has not been implemented yet. However, we could demonstrate that some technical obstacles such as bypassing default bridge logic by link layer redirection with MAC address DNAT can be overcome and that the system itself is achievable.

5 Conclusion and outlook

In this paper, we presented a concept which combines the strengths of IDS, IPS and honeynet approaches. As a result, it is possible to substantiate (or abandon) an unspecific initial suspicion issued by an anomaly based IDS and, thus, to improve its detection accuracy. By using VLANs and packet filtering it is possible to maintain basic service to the suspicious system during the observation period while minimising the risk for the production network on the other hand.

A prototype implementation of this concept is already partially operational and first tests have already indicated that a realisation is feasible and that promising results can be obtained. Thus, this highly automated system requiring only moderate implementation and administration effort has the potential to improve security in networks where classical firewall concepts alone are not sufficient, e.g. due to a large number of mobile end systems which are dynamically added and removed.

Further studies are planned to refine the relatively simple implementation of some components of the prototype implementation. In particular, it will be investigated if some of the statically preconfigured functions can be replaced by more dynamically adaptive solutions. The refined prototype will allow to evaluate robustness, scalability and performance of the proposed scheme in more detail. Furthermore, additional tests will provide more insight into the optimisation of system parameters, e.g. timers and thresholds. An open issue is how several suspicious systems can best be observed at the same time, i.e. if it is better to provide several (virtual) quarantine networks or to try allocate incidents in the quarantine network to specific suspects by correlating HIDS and NIDS data. At a later stage, investigations on suitable scenarios are planned because our concept currently focuses on highly autonomous malware (such as viruses) and it is yet not clear whether remotely controlled intrusions could be safely contained in the quarantine network we developed.

References

[1] Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J. and Stoner, E.: ,,State of the Practice of Intrusion Detection Technologies", CMU/SEI-99-TR-028 (2000)

[2] E., Cloete, E., Venter, L.M.: ,,A comparison of Intrusion Detection systems", Computers and Security, 20 (2001), S. 676-683

[3] Lazarevic A., Ozgur A., Ertoz L., Srivastava J., Kumar V.: ,,A comparative study of anomaly detection schemes in network intrusion detection", in: SIAM International Conference on Data Mining (2003)

[4] Debar H., Dacier M., Wespi A.: ,,Towards a Taxonomy of Intrusion-Detection Systems", Computer Networks, 31(8): S. 805–822, April 1999.

[5] McLaughlin, Laurianne: ,,Bot Software Spreads, Causes New Worries", IEEE Distributed Systems online 1541-4922, Vol. 5, No. 6; June 2004, http://csdl.computer.org/comp/mags/ds/2004/06/o6001.pdf

[6] The Honeynet-Project: ,,Know Your Enemy: Learning about Security Threats", Indianapolis: Addison-Wesley, 2004, http://www.honeynet.org/

[7] S. Riebach, B. Toedtmann, Erwin P. Rathgeb: ,,Efficient deployment of honeynets for statistical and forensic analysis of attacks from the Internet", to appear in the proceedings for Networking 2005 conference, Waterloo Ontario, Canada, 02.-06. May 2005

[8] S. Riebach, B. Toedtmann, Erwin P. Rathgeb: ,,Risk assessment of production networks using honeynets - some practical experience", to appear in the proceedings of ISPEC05 conference, Singapur, 12.-14. April 2005 r

[9] The Honeynet Project: ,,Know your enemy: Passive Fingerprinting", Whitepaper, March 2003, http://www.honeynet.org/papers/finger/

[10] Roesch, Marty; Caswell, Brian: ,,Snort's official homepage", http://www.snort.org, last seen: 2. Feb. 2005

[11] Lehti R., Pablo Virolainen, P., van den Berg, R.: ,,AIDE (Advanced Intrusion Detection Environment)", http://sourceforge.net/projects/aide, last seen: 2. Februar 2005

[12] Metcalf, W.: ,,Snort_inline Projekt Homepage", http://snort-inline.sourceforge.net/, last seen: 2. Feb. 2005

[13] Biles, S.: ,,The SPADE Project", last seen: 11. Oct. 2004, http://www.bleedingsnort.com/article.php?story=20041011095505501

[14] LBNL's Network Research Group: ,,arpwatch", http://www- nrg.ee.lbl.gov/, last seen: 2. February 2005

[15] Zeltserman, D.: ,,A practical guide to SNMPv3 and network Management", New Jersey 1999

[16] Linux netfilter/iptables project: ,,netfilter/iptables", http://www.netfilter.org/, last seen: 2. Febraury 2005

[17] Schuymer, Fedchik, Borowiak: ,,ebtables", http://ebtables.sourceforge.net/, last seen: 2. February 2005

[18] Decker E., Langille P., Rijsinghani A., McCloghrie K.: ,,RFC 1493 - Definitions of Managed Objects for Bridges", Internet Standard, 1993

[19] Blunk L., Vollbrecht J.: ,,RFC 2284 - PPP Extensible Authentication Protocol (EAP)", Internet Standard, 1998

[20] IEEE: ,,Port-Based Network Access Control", New York 2001

[21] VMware, Inc.: ,,VMware Workstation 4", http://www.vmware.com/, last seen: 2. February 2005