Attack Detection using Cooperating Autonomous Detection Systems (CATS)

Falko Dressler, Gerhard Münz, Georg Carle

University of Tübingen, Wilhelm-Schickard-Insitute of Computer Science,
Computer Networks and Internet, Auf der Morgenstelle 10C
72076 Tübingen, Germany
Phone +49 7071 29-70522, Fax +49 7071 29-5220
{dressler, muenz, carle}@informatik.uni-tuebingen.de
http://net.informatik.uni-tuebingen.de/

Abstract. Today's communication networks are threatened by an increasing number intrusion attempts, worms, and denial of service (DoS) attacks. Apart from general measures for attack prevention, the possibility to detect ongoing attacks in order to take appropriate countermeasures constitutes an important asset for network security. We present a novel approach for attack detection based on cooperating autonomous detection systems (CATS). While a single detection system is able to identify ongoing attacks autonomously, cooperation with remote detection systems located in other parts of the network can improve the detection performance.

1 Introduction

The detection of intrusions, violations, and attacks is a significant task in nowadays communication networks. A brief history and overview to intrusion detection is, for example, given by Kemmerer and Vigna [13]. The number of denial of service attacks (DoS) and distributed denial of service attacks (DDoS) is increasing every day. Typically, important servers from government or industrial systems are attacked, yet we already see similar attacks against primary systems at universities. In addition, the effectiveness and malignance of such attacks is increasing [12].

In this paper we present a novel approach addressing these problems: CATS, a Cooperating Autonomous deTection System. The goal is to identify ongoing attacks using autonomously working detection system that are able to improve their detection performance through cooperation in a group of multiple detection systems. We use a common taxonomy of attack detection systems in order to classify and compare our approach with existing systems.

The proposed detection system will be implemented and used in the context of Diadem Firewall, an IST funded project [1].

The rest of the paper is organized as follows. A taxonomy of attack detection systems as well as a review of related work is presented in section 2. General requirements for an efficient detection system are discussed in section 3. Our novel approach for intrusion detection, the cooperating autonomous detection system, is presented in section 4. Section 5 evaluates the proposed system and compares it to existing detection systems. Finally, the conclusions are provided in section 6.

2 Taxonomy and related work

In order to classify our novel approach for cooperative autonomous detection of intrusions and DoS attacks, we first provide a taxonomy of detection systems and an overview of related work.

2.1 Taxonomy

Several attempts have been made in order to provide a taxonomy and classification of intrusion detection systems (IDS) as well as denial of service (DoS) attacks and detection techniques [2, 14, 15]. We provide a brief taxonomy of detection techniques and define the terms used in this paper.

First, we use the term “attack detection” instead of “intrusion detection” because we do not want only refer to the detection of intrusion attempts into a protected system, but also include the detection of attacks that aim to disturb the well functioning of the system, e.g. by causing system break-downs, resource exhaustion or any other kind of DoS to legitimate users. Furthermore, we define the term “attack” as any kind of intrusion or DoS attack. We define the term “detection system” as a system that performs detection of intrusions, DoS attacks, or both. This generalization is reasonable since most of today’s systems combine intrusion and DoS attack detection.

A common classification criterion distinguishes two types of attack detection: host-based and network-based detection. Host-based detection is restricted to a single host. Typically the detection system is directly executed on the protected host, which allows access to a wide range of security-related information, like log file entries, the state of running processes and logged-on users. It also allows analyzing network communication on any protocol layer, including the content of encrypted connections. In contrast, network-based detection addresses a network including the traffic from and to the connected hosts. Such a detection system is usually executed on a dedicated machine that passively captures and analyses the traffic on the network. As an advantage, network-based detection does not require any changes of the protected hosts and networks, and one system is sufficient for a larger number of hosts. On the other hand, the captured network traffic is the only source of information that can be used.

Another criterion refers to the detection method. We distinguish between knowledge-based detection, also known as rule-based detection, and anomaly detection. Knowledge-based detection comprises any method that disposes information about known attacks with the purpose to search for equal or similar occurrences. In case of network-based detection, systems that search the captured packets for attack signatures and bit patterns belong to this category, but also any systems that look for potential misbehavior and suspicious usage of protocols (e.g. during a port scan). Anomaly detection uses an opposite approach: based on information about normal network or system behavior, a significant derivation from this reference model is considered as indicator of a potential attack. Anomaly detection is able to recognize previously unknown attacks. However, derivation from normal behavior can have other reasons than attacks, resulting in false positives. Anomaly detection can be built on statistical tests and analysis.

If more than one host or larger networks are to be surveyed, a distributed detection system consisting of several subsystems can be deployed. These subsystems are located on different host or in different parts of the network. According to the relationship between the subsystems, we adopt the classification of [15] and differentiate between *autonomous*, *cooperative*, and *interdependent* subsystems. Autonomous subsystems act as independent and fully functional detection systems, i.e. they detect attacks independently without communication or interaction between them. Even if such subsystems deliver their detection results to a centralized entity e.g. a common database, they would still operate in an autonomous way. Cooperative subsystems are capable of autonomous detection. In addition, they can improve their detection performance through cooperation. Interdependent subsystems cannot operate autonomously, e.g. because one subsystem controls other subsystems in order to coordinate the detection process. Another example is a distributed detection system that organizes its subsystems in a hierarchical manner. In this case, two aspects of interdependency are possible: subsystems of higher hierarchy levels may control subsystems of lower levels, or subsystems of higher levels may depend on data delivered by lower-level subsystems.

Figures 1 to 3 show three examples of distributed detection systems with autonomous, cooperative and interdependent subsystems. Optional parts are drawn in dashed lines. In figure 1, the subsystems work autonomously without any cooperation. An optional centralized database can be used to gather all alerts generated by the subsystems. Figure 2 shows a detection system of autonomous subsystems that cooperate with each other. Again, alerts might be collected in a centralized database. Finally, a system of interdependent subsystems is depicted in figure 3. The detection subsystems are controlled by a centralized controller subsystem that might optionally adapt its control according to the reported alerts.

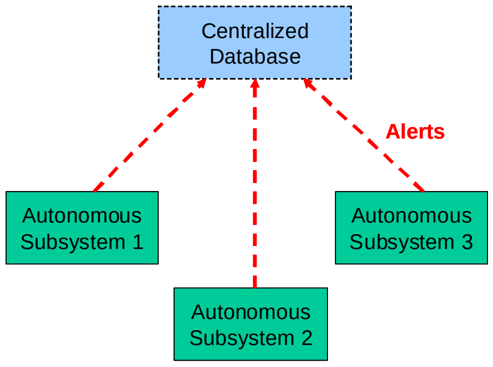

Fig 1. Distributed detection system with autonomous subsystems

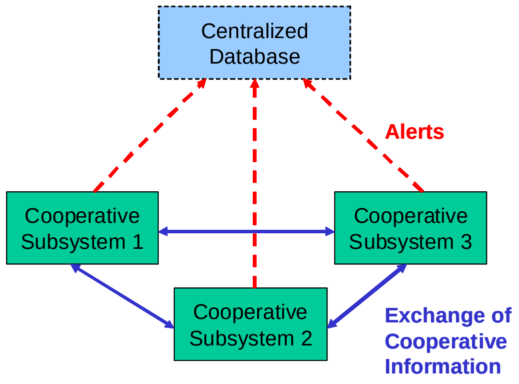

Fig 2. Distributed detection system with cooperative autonomous subsystems

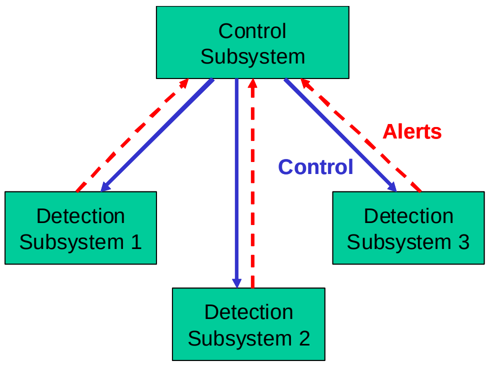

Fig 3. Distributed detection system with interdependent subsystems

2.2 Related work

In this section, we present four examples of relevant related work in the area of distributed detection systems. Some of them also provide automatic response functionality. However, in the context of this paper we concentrate on the detection mechanisms.

EMERALD [19] is a distributed detection and response system that has been developed at the SRI research institute in 1997. EMERALD was conceived to apply mainly host-based detection techniques, but its architecture is also suitable for network-based detection. It applies both knowledge-based and anomaly detection methods. The subsystems, called monitors, are organized in a three-level hierarchy. Each monitor of the two higher levels receives and correlates the detection results of various monitors of the corresponding lower level. Therefore, a subscription-based communication scheme is used that allows the subscriber to specify in which kind of detection results it is interested. According to our taxonomy, the EMERALD monitors can be classified as interdependent subsystems since monitors of higher levels depend on the detection results of lower-level monitors.

Prelude IDS [4, 21, 22] is an open-source project that consists of three functional components: sensors, managers, and countermeasure agents. The sensors deploy knowledge-based techniques for both network and host-based detection. Anomaly detection is not applied. Several sensors are associated to a manager that receives the detection alerts: it processes and correlates the alerts and decides on appropriate countermeasures. If more than one manager is used, the managers are organized in a hierarchy with a single one at the top. Since both sensors and managers are subsystems that are involved in the detection process and since managers depend on the results delivered by the sensors, the Prelude subsystems can be classified as interdependent.

D-WARD [16] is a network-based DDoS detection and defense system developed by Jelena Mirkovic at the University of California Los Angeles (UCLA). The D-WARD architecture consists of fully autonomous subsystems that are deployed at the entry points of so-called source-end networks. The subsystems analyze and control the outgoing traffic with the goal to prevent hosts of the observed networks from participating in DDoS attacks. Therefore each subsystem autonomously imposes rate limits on suspicious flows. The detection method is based on predefined models of normal traffic and can thus be classified as anomaly detection.

COSSACK [17] is a distributed, network-based DDoS attack detection and response system developed at the Information Science Institute (ISI) of the University of Southern California, funded by DARPA. The COSSACK subsystems, called watchdogs, are located at edge networks. If a watchdog detects an attack against the associated edge network, it multicasts an attack notification to the other watchdogs. Upon reception of such a notification, a watchdog checks if the attack flow or parts of it originate from its own edge network. If so, the watchdog tries to block or rate-limit the corresponding flow by setting filter rules in routers or firewalls. With respect to attack detection, the subsystems can be classified as cooperative since the notification messages help other watchdogs to detect an ongoing attack. COSSACK principally applies anomaly detection techniques.

Table 1 summarizes the classification of the presented systems. It already includes our novel approach CATS whose detailed description follows in section 4.

Table 1 Overview of existing distributed detection systems

System	Type of detection	Detection methods	Relationship between subsystems
EMERALD	host-based	knowledge-based and anomaly detection	interdependent
Prelude IDS	host-based and network-based	knowledge-based detection	interdependent
D-WARD	network-based	anomaly detection	autonomous
COSSACK	network-based	anomaly detection	cooperative
CATS	network-based	knowledge-based and anomaly detection	cooperative

3 General requirements for autonomous attack detection

In this section we concentrate on general requirements for distributed attack detection. Based on this set of requirements, existing detection systems as well as our novel approach, CATS, can be assessed. The requirements are introduced using two typical denial of service scenarios, which are described in the following section.

3.1 Denial of service scenarios

Denial of service attacks focus on the prevention of an offered service. This can be done in two ways: first, by exhausting network resources on the path towards the target server and secondly, by exhausting resources of that server. An example for the first scenario is a distributed ICMP flood attack. A TCP SYN flood attack is an example for the second scenario. Both scenarios are depicted in figure 4. Further information on distributed denial-of-service attacks can be found in [6].

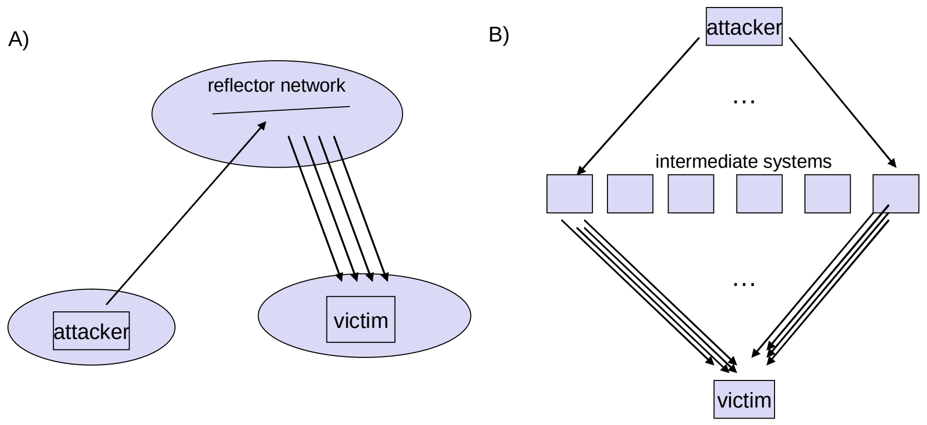

Fig 4. Typical denial of service scenarios. A: ICMP flood using a reflector network; B: TCP SYN flood using intermediate systems, e.g. compromised / malicious computers

3.1.1 ICMP flood attack

An ICMP flood attack can take place in two ways. First, so called broadcast pings can be employed utilizing an unsecured reflector network for forwarding the ICMP echo request messages towards a victim network. Secondly, IP address spoofing can be used by sending ICMP echo requests to multiple stations in the network with the IP address of the victim inserted in the source IP address of each packet. All the receivers of these ICMP request will answer by sending an appropriate response to the victim, from which they think the request was coming from. This scenario is shown in figure 4A. The result of this attack is an overload of the network paths near the victim. Therefore, normal service requests suffer from the artificial network congestion and cannot be served in an adequate time.

3.1.2 TCP SYN flood attack

The goal of a TCP SYN flood attack (see figure 4B) is to exhaust local resources at the victim. TCP is a connection oriented transport protocol. Thus, in order to transmit data, a connection has to be established first. This is done by sending a TCP SYN packet which is answered by an ACK+SYN. After the reception of the SYN packet, a half-open connection remains until it is timed out or the ACK+SYN is being answered.

Benefiting form this working principle of TCP, TYP SYN flood attacks employ compromised computers as a relay for a particular attack. All the relay hosts are commanded to send as many TCP SYN packets as possible to the victim. Resources required for state information of half-open connections are exhausted quickly, preventing the victim from receiving legitimate service requests.

3.2 Requirements for detection systems

Based on the described scenarios, general requirements for detection systems are derived in the following.

3.2.1 Detection of attacks

The capability of a detection system to detect anomalies and concrete attacks in a local context is evident. The capability of the detection system to detect anomalies and attacks in a global context is also important. The key properties of detection mechanisms are listed in the following.

Local context

Attack detection in a local context, i.e. based on information from packet data received at the detection system only is a straightforward process integrated in almost all detection systems. This capability requires no intercommunication or interaction with other detection systems. Both types of attacks (see figure 4) can be detected by a system near the victim. Nevertheless, only a system near the attacker is able to detect the source of the attack if IP address spoofing techniques are used.

Additionally, traceback mechanisms can be deployed to identify the source of a spoofed IP packet. Unfortunately, such mechanisms have significant resource requirements.

Global context

Using a global context, i.e. information gathered at multiple points in the network, allows improved detection of ongoing attacks in the network. Both scenarios described before can be detected with a global context. Therefore, this capability is an essential requirement. The communication overhead introduced by the different interacting detection systems is an important performance measure of the complete system.

Knowledge-based detection

Knowledge-based detection was the first kind of attack detection deployed in the Internet. While statistical conclusions are not possible, well-known attacks can be detected efficiently using this methodology.

Anomaly detection

The capability to employ anomaly detection mechanisms is a further requirement for highly accurate attack detection. With the increasing capacities of network links, pure knowledge-based detection systems suffer from their inability to process every single data packet. By employing statistical methods for anomaly detection, high-speed detection engines can be realized. Anomaly detection also allows to detect new kinds of attacks, or slightly modified variants of known ones, that cannot be detected by knowledge-based systems.

3.2.2 Autonomous behavior

To prevent a distributed detection system from becoming a target itself and to increase the availability of the overall system, each subsystem should perform attack detection autonomously. The intercommunication between the subsystems may increase the detection accuracy but should not become a pre-requisite for the functionality of the global detection system. Autonomous behavior requires the following capabilities:

Self-Configuration

A first requirement for autonomous behavior is the capability of self-configuration. Starting from a master configuration, or even starting from scratch, the system must be capable to set all required configuration parameters, such as the current location of the probe or the type and number of neighboring entities to which communication relationships are to be applied.

Self-Maintenance

Self-maintenance is the process of adapting the configuration parameters to the current situation. Autonomously working entities must be capable to adapt to a changing environment. This adaptation, typically realized by reconfiguration of runtime parameters, comprises of changes in the resource management and in the configuration of tasks and processes.

Self-Healing
Self-healing is an important function of autonomously working entities. In the case of problems, mechanisms must be available which determine the kind of problem and initiate a healing process. For example, if the system faces memory shortages, the attack detection must be modified by selecting algorithms and parameters which require less memory (while typically resulting in a lower detection rate).

Self-Optimization
Finally, self-optimization is an important requirement for autonomous systems. In this context we understand self-optimization as the ability to optimize the detection quality. This can be achieved by exchanging information about already identified attacks or suspicious network connections and also by statistically forwarding parts of collected data packets and network statistics to neighboring probes.

3.2.3 Distributed intelligence

A detection system can benefit from distributed intelligence. In the context of attack detection, we distinguish between two aspects of distributed intelligence: distributed detection, and separation of monitoring and detection functionalities.

Distributed detection
Distributed detection means that multiple detection systems are involved in the detection process, each analyzing a variable part of the monitored data. As a consequence, the detection load can be partitioned dynamically according to the available resources at the different detection engines.

Separation of monitoring and detection
Monitoring and detection functionalities should be separated in order to allow for an analysis of the monitored data at different locations and in different contexts. As a consequence, distributed intelligence may improve the detection performance and may increase the robustness of the whole system.

4 Cooperative autonomous attack detection

The objective of this section is to describe our novel approach for attack detection using cooperative autonomous detection systems. First, the architecture is presented followed by a classification of our approach based on the presented taxonomy.

4.1 Architecture of our novel autonomous detection system

The architecture of an individual detection system is depicted in figure 5. It consists of an outer part for network monitoring and an inner part for detection.

The network monitoring part is responsible for capturing packets and flow statistics from the network, either directly using a connected network interface, or by employing monitoring probes and the standardized IP flow export (IPFIX) [7, 20] and packet sampling (PSAMP) [8, 11] protocols. This part also performs necessary preprocessing of the gathered data, such as packet filtering or generation of statistical flow measurements needed by the detection part. It is further divided into a layer for packet monitoring and sampling and a layer for statistical measurements.

The detection part is divided into two detection engines, one providing statistical anomaly detection and the other applying knowledge-based detection mechanisms. The required packet data and statistical measures are provided by the network monitoring part.

The main reason for separating the network monitoring part and the detection part is to allow for a multi-hierarchy monitoring environment for capturing packets and flow statistics. The accounting NSLP protocol [10] can be employed for the configuration of the monitoring environment. This allows for deploying one detection system that analyzes data monitored at different points of the network. Furthermore, a detection system can become itself a source of information to other detection systems by exporting monitoring data.

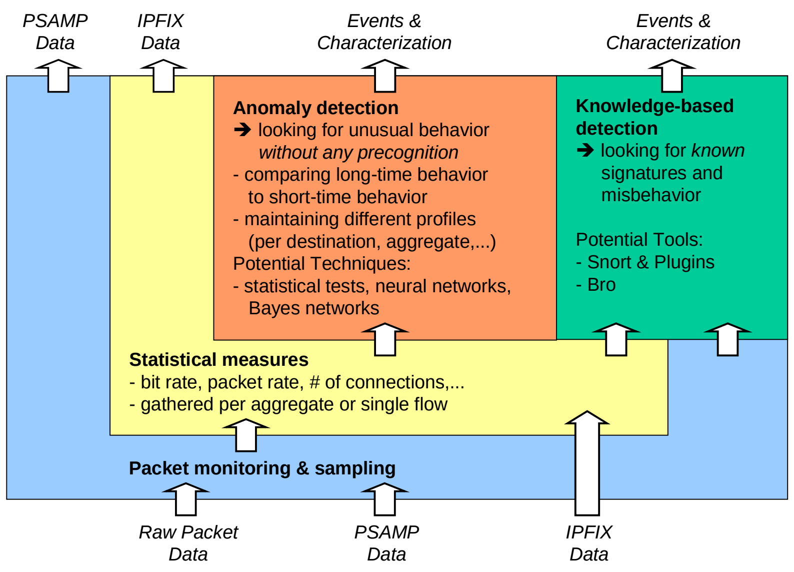
PSAMP Data
IPFIX Data
Events & Characterization
Events & Characterization
Anomaly detection
➔ looking for unusual behavior without any precognition
- comparing long-time behavior to short-time behavior
- maintaining different profiles (per destination, aggregate,...)
Potential Techniques:
- statistical tests, neural networks, Bayes networks
Knowledge-based detection
➔ looking for known signatures and misbehavior
Potential Tools:
- Snort & Plugins
- Bro
Statistical measures
- bit rate, packet rate, # of connections,...
- gathered per aggregate or single flow
Packet monitoring & sampling
Raw Packet Data
PSAMP Data
IPFIX Data

Fig 5. Architecture of our novel autonomous detection system

In the following subsections, the network monitoring part and the detection part of the detection system are described in more detail.

4.1.1 Packet monitoring and sampling layer

The architecture of our detection system allows two ways to capture packet data from the network: by using a directly connected NIC, and by employing PSAMP exporters, which send the collected information in a standardized way [9]. The packet monitoring and sampling layer is responsible for capturing of packet data received via NICs or PSAMP. Moreover this layer may preprocess the packet data. Filters or sampling algorithms may be applied to reduce the amount of packets being further processed.

Within the detection system, the collected packet data is used for two purposes. First it can be directly passed on to the detection part in order to look for known attack signatures. Secondly it can be forwarded to the statistical measurement layer that generates flow statistics from the packet data. Additionally, the detection system can export packet data to other detection systems using PSAMP. This functionality is described in section 4.1.5.

4.1.2 Statistical measurement layer

The statistical measurement layer generates statistical flow measures based on the packet data received by the packet monitoring and sampling layer, and the flow statistics received via IPFIX. Examples for statistical measures are the number of bytes and packets per flow or per aggregate, the number of connections per time, and the number of similar connections. The resulting statistical measures build the basis for further anomaly detections. For instance, an unusually high connection rate may indicate a distributed denial of service attack where typically each connection consists of only a single packet.

The statistical measurement layer does not only provide the data for the local detection mechanisms. It may also export the generated flow statistics via IPFIX. Using the terms of IPFIX [7], this corresponds to the functionality of an exporter or concentrator.

4.1.3 Anomaly detection

In our novel detection system, we integrate two separate, independently working detection engines - an anomaly detection engine and a knowledge-based detection engine - in order to achieve high detection rates. The detection of an attack results in the generation of an event that is combined with additional information for characterizing the attack. This information can be exchanged with other detection systems in order to improve the detection performance. On the other hand, it can be used to trigger appropriate countermeasures.

The anomaly detection works on statistical data received from the lower statistical measurement layer. This detection process is looking for unusual behavior without any precognition. It compares long-time behavior to short-time behavior and maintains different profiles, e.g. per destination, aggregate, and others. Potential techniques are statistical tests, neural networks, and Bayes networks. The architecture of our autonomous detection system allows to integrate a variety of other detection algorithms.

4.1.4 Knowledge-based detection

The knowledge-based approach represents the second main pillar of our detection engine. This engine searches the packet stream for known signatures and misbehaviors. Open-source tools such as snort [3, 5] and Bro [18], which are widely used in the Internet community, build the basis for this part of the detection.

4.1.5 Export of packet data and flow statistics

One detection system is capable of exporting packet data and flow statistics to other detection systems using PSAMP and IPFIX. This export capability is useful since one system might not be able to process all interesting or suspicious packet data and flow statistics. Through the export, it is possible to delegate parts of the detection work to other systems. A much more important advantage is the possibility to organize several detection systems in a multi-level hierarchy. In such a constellation, some systems may focus on processing data from their local environment. While performing attack detection using locally captured packet data, these detection systems also forward packet data and flow statistics to systems of a higher hierarchy level. Higher-level systems receive and process data from various parts of the network. Hence, a system of a higher-level may be able to examine certain aspects of an attack better than the lower-layer systems. This is particularly true for distributed attacks.

The detection systems use PSAMP and IPFIX both for data from monitoring probes, and for data from other detection systems.

4.2 Cooperation of multiple autonomous detection systems

So far, an individual, autonomously operating detection system can achieve a good detection rate by incorporating features from different approaches: knowledge-based detection and anomaly analysis. Additionally, the possibility to use a nearly unlimited monitoring network allows to gather packet data from multiple points in the network. From this respect, our approach is directly comparable to distributed detection systems such as EMERALD and Prelude IDS. However, our approach can achieve significant advantages due to its double detection mechanisms.

In this section we show how the detection quality can be enhanced further by loosely coupling multiple autonomous detection systems to cooperating ones, which additionally improves the overall detection quality.

Figure 6 shows a diagram of multiple interacting detection systems. Apart from the export of sampled packets and flow statistics using PSAMP and IPFIX (see section 4.1.5), the interaction between the systems is based on the exchange of information about suspicious network traffic.

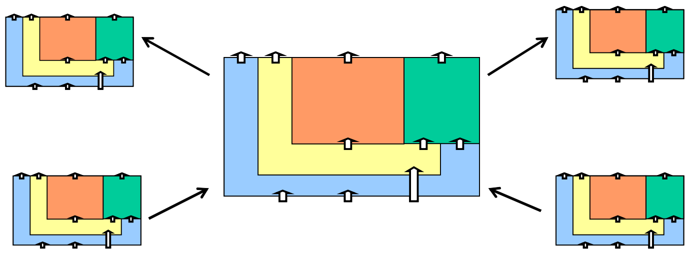

Fig 6. Interactions between multiple autonomously working detection systems

We assume that several types of attacks cannot be detected directly, especially distributed attacks. Instead, only assumptions can be generated about which packets or flows may belong to attacks. There are two main reasons for this:

1. Typically, a detection system obtains only a fraction of the packets belonging to an ongoing attack because it is not located directly at the attacker or at the victim.
2. In multi-gigabit networks, the capacity of monitoring probes and detection systems is limited. Frequently, sampling algorithms are employed for coping with the high data rates, which also drop packets belonging to an attack.

We address both problems by creating state information for all suspicious data flows. Starting from a first assumption of an ongoing attack, the detection system has to refine the analysis in order to confirm or reject the assumption. In a first step, the aggregation level will be decreased until the potential attack flows can be isolated and a corresponding filter rule can be formulated. Subsequently, the filters at the monitoring probe and the detection systems can then be programmed in order to capture and analyze all packets belonging to the suspicious flows. Ideally, sampling algorithms are applied only on packets that do not belong to suspicious flows.

In case that deeper analysis confirms an initial assumption, the state information is sent to other detection systems. Other systems that have not yet detected the same attack flow proceed as if the state information was a local assumption, trying to confirm or reject it by refining the analysis. As a result, the detection system can either affirm that it observes the same attack flow, or it dismisses the state information.

Therefore, the cooperation of detection systems allows to improve the detection rate significantly and helps to identify the path of the attack flows through the network.

5 Evaluation of cooperating autonomous detection systems

In order to evaluate our concept of cooperating autonomous detection systems, we first provide a classification according to our taxonomy. Then we show how the general requirements given in section 3 are fulfilled. Finally we compare CATS with existing approaches.

5.1 Classification

Our detection system provides network-based attack detection. It applies both knowledge-based and anomaly detection methods. Since the two methods are complementary, a combination allows to benefit from the advantages of both methods while mitigating the individual drawbacks. Finally, our concept can be classified as cooperative, since the individual detection systems work autonomously while improving their detection performance through cooperation.

5.2 Assessment based on the requirements

In section 3, we listed general requirements for attack detection systems. In this section, we show how CATS satisfies them.

First, we stated that attack detection should support both a local and a global context. Most existing detection systems are limited to attack detection in a local context, using monitoring data from the local environment. There exist some mechanisms for a global context, in particular the traceback mechanisms. However, in most cases attack detection in a global context is reduced to the deployment of independent, local detection systems at various locations in the network and a central entity that receives the detection results. Our system addresses the global context requirement in two ways. First, the clear separation of network monitoring and attack detection allows that each individual detection system processes monitoring data received from monitoring probes located at remote parts of the network. Secondly, the cooperation between different detection systems allows distributed detection in a global context.

In compliance to the requirements, our approach provides both knowledge-based and anomaly detection using two distinct detection engines.

With respect to the desired autonomous behavior, we outline some design aspects concerning the four categories self-configuration, self-maintenance, self-healing, and self-optimization. With respect to self-configuration, a discovery mechanism is needed that enables the detection system to discover monitoring probes as well as other detection systems for cooperation. For this purpose, service discovery mechanisms that have been developed for other problem domains can be employed. Self-maintenance and self-healing address the ability to adapt the monitoring and detection load according to the available resources of monitoring probes and detection systems by setting PSAMP and IPFIX parameters accordingly. An aspect of self-optimization is the capability to refine the applied analysis methods in order to confirm or reject initial attack assumptions. The usage of dynamic profiles instead of fix thresholds for anomaly detection allows self-optimization according to temporally changing network conditions.

Last but not least, our approach profits from distributed intelligence through the separation of monitoring and detection functionalities and the cooperation between multiple detection systems.

5.3 Comparison with other attack detection approaches

In section 2.2, we briefly presented four research projects in the area of distributed detection systems. The only cooperative system was COSSACK. COSSACK also is the most recent project, and appears superior to the other three systems. Therefore, we concentrate on a comparison of our detection system with the COSSACK system.

The main differences between COSSACK and our system are the employed detection methods, the location of deployment in the network, and the treatment of spoofed source addresses.

5.3.1 Employed detection methods

COSSACK applies so-called “blind” detection techniques, which corresponds to anomaly detection in our taxonomy. Our proposed detection engine integrates both knowledge-based and anomaly detection.

5.3.2 Location of deployment

COSSACK watchdogs are deployed at edge networks in order to detect attacks against hosts inside the surveyed network. Instead, our detection system can be deployed anywhere in the network. Moreover we think that a deployment of several detection systems at appropriate locations inside the network is promising. Such locations may be access routers, gateways, but also core routers.

Papadopoulos et al. discuss the question of location in [17]. Apparently they abandoned the idea to place their watchdogs inside the core of the network. First, they argue that monitoring and detection cannot be performed at line speed because of the very large link bandwidth. Our solution faces this problem using packet sampling and aggregated flow statistics. Hence, the monitoring and detection load can be controlled and limited by adapting the PSAMP and IPFIX parameters accordingly. As soon as a profound assumption of an attack exists, the sampling rate as well as the level of aggregation for the corresponding flow is lowered for deeper analysis of what is going on.

Papadopoulos [17] presents another reason for not deploying the watchdogs in the core network: Watchdogs classify flows according to predefined aggregate rate thresholds. These thresholds are set depending on the link bandwidth between the edge network and the core. However, in the core only coarse approximations of the edge link bandwidth are available. We do not use fixed thresholds but dynamic profiles for anomaly detection. These profiles do not rely on prior knowledge since they are created in a self-learning process.

5.3.3 Source address spoofing

COSSACK watchdogs identify the origin of an attack examining the source addresses of the corresponding packets. Obviously this method does not work if attackers use spoofed source addresses. Papadopoulos et al mention this problem without addressing it any further, arguing that there are sufficient technical remedies like egress and ingress filters that inhibit source address spoofing. We do not think that such filters can be applied in all possible cases (see also [6]). Instead we prefer not to rely on the correctness of source addresses. Using our cooperating detection systems, it should be possible to track the course of the attack flow by identifying the monitoring probes where the flow has been seen. Even though the origin networks cannot be identified exactly with this method, we still will be able to determine where the attack flow could be preferably blocked or rate-limited.

5.4 Assessment Summary

In Table 2, we compare CATS with related detection systems regarding their fulfillment of the requirements depicted in section 3.2. Obviously, CATS is the only system that fulfills all requirements.

Table 2 Requirements analysis

		EMER-ALD	Prelude IDS	D-WARD	COSS-ACK	CATS
Attack detection	Local context	yes	yes	yes	yes	yes
	Global context	no (host-based)	no	no	yes	yes
	Knowledge-based detection	yes	yes	no	no	yes
	Anomaly detection	yes	no	yes	yes	yes
Autonomous behavior		no	no	yes	yes	yes
Distributed intelligence	Sep. of monitor-ring & detection	no	no	no	no	yes
	Distributed detection	yes	partly	no	no	yes

6 Conclusions

In this paper we presented a novel approach for efficient and high-quality attack detection called CATS. The proposed detection system makes use of a distributed monitoring environment and achieves improved detection results through cooperation with remote detection systems. The detection quality is further increased by combining anomaly and knowledge-based detection mechanisms.

In order to assess our approach, we provided a conceptual comparison with existing systems. We conclude that our approach allows to achieve significant performance enhancements compared to existing approaches. Currently, we are implementing a prototype of the detection system in the context of the Diadem Firewall [1] project. This prototype will allow us to verify the capabilities of the detection system using real-world and simulated traffic.

7 References

1. "Diadem Firewall," EU FP6 Project IST-2002-002154, http://www.diadem-firewall.org (2004)
2. R. Bace and P. Mell, "Intrusion Detection Systems," NIST (2001)
3. J. Beale and B. Caswell, *Snort 2.1 Intrusion Detection*, 2nd edition ed, Syngress, 2004
4. M. Blanc, L. Oudot, and V. Glaume, "Global Intrusion Detection: Prelude Hybrid IDS," Technical Report (2003)
5. B. Caswell and J. Hewlett, "Snort Users Manual," The Snort Project, Manual (2004)
6. R. K. C. Chang, "Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," *IEEE Communications Magazine*, vol. 10 (2002) 42-51
7. B. Claise, M. Fullmer, P. Calato, and R. Penno, "IPFIX Protocol Specifications," in *draft-ietf-ipfix-protocol-03.txt* (2004)

8. B. Claise, "Packet Sampling (PSAMP) Protocol Specifications," in *draft-ietf-psamp-protocol-01.txt* (2004)
9. T. Dietz, F. Dressler, G. Carle, and B. Claise, "Information Model for Packet Sampling Exports," Internet-Draft, draft-ietf-psamp-info-01.txt (2004)
10. F. Dressler, G. Carle, C. Fan, C. Kappler, and H. Tschofenig, "NSLP for Accounting Configuration Signaling," Internet-Draft, draft-dressler-nsis-accounting-nslp-00.txt (2004)
11. N. Duffield, "A Framework for Packet Selection and Reporting," in *draft-ietf-psamp-framework-05.txt* (2003)
12. M. Handley, "Internet Denial of Service Considerations," draft-iab-dos-00.txt (2004)
13. R. Kemmerer and G. Vigna, "Intrusion Detection: A Brief History and Overview," *IEEE Computer* (2002) 27-30
14. R. B. Lee, "Taxonomies of Distributed Denial of Service Networks, Attacks, Tools, and Countermeasures," Princeton University, Technical Report (2004)
15. J. Mirkovic, J. Martin, and P. Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms," University of California, Los Angeles, Technical Report #020018 (2002)
16. J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the Source," Proceedings of 10th IEEE International Conference on Network Protocols (ICNP 2002), Paris, France (2002) 312-321
17. C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan, "COSSACK: Coordinated Suppression of Simultaneous Attacks," Proceedings of DARPA Information Survivability Conference and Exposition, Washington DC, USA (2003)
18. V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," *Comuter Networks*, vol. 31 (1999) 2435-2463
19. P. A. Porras and P. G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," Proceedings of National Information Systems Security Conference (1997)
20. J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export," in *draft-ietf-ipfix-reqs-16.txt* (2004)
21. K. Zaraska, "IDS Active Response Mechanisms: Countermeasure Subsytem for Prelude IDS," Technical Report (2002)
22. K. Zaraska, "Prelude IDS: current state and development perspectives," Technical Report (2003)