An extensible and flexible System for Network Anomaly Detection

Thomas Gamer, Marcus Schöller, and Roland Bless

Institut für Telematik
Universität Karlsruhe (TH), Germany

Keywords: Programmable Networks, Anomaly Detection, DDoS Attacks

Abstract. Network hazards like attacks or misbehaving nodes are still a great obstacle for network operators. Distributed denial of service attacks and worm propagations do not only affect the attacked nodes but also the network itself by wasting network resources. In wireless ad hoc networks even more hazards exist due to its self-organizing characteristic. A detection of such network hazards as early as possible enables a fast deployment of appropriate countermeasures and thereby significantly improves network operation. Our proposed detection system uses programmable network technology to deploy such a system within the network itself. Doing this without influencing the routing performance seriously demands a resource saving architecture. We therefore propose to use a hierarchical architecture which runs a very small basic stage all the time and loads specialized detection modules on demand to verify the network hazard. In this paper we introduce our system which can detect DDoS attacks, worm propagations, and wormhole attacks.

1 Introduction

In today's networks hazards are frequent and comprise various kinds of attacks as well as serious changes of the network itself. To automatically detect such hazards is still a challenge for network operators on the one hand. On the other hand more and more self-managed networks like ad hoc networks evolve. Such networks require an autonomic mechanism to detect hazards and employ fitting countermeasures autonomously. The range of network hazards we have in mind include network attacks like distributed denial-of-service (DDoS) attacks [6,9] or worm propagations [15,11] but also wormhole attacks [8] or misbehaving nodes in wireless ad hoc networks.

The earlier such hazards can be detected the better the network can be protected against them [17]. This requires a detection system within the network. Programmable networks enhance routers to flexibly and dynamically set up new services on that router. Furthermore it eases the update process of service modules since their functionality gets not tightly coupled to the packet forwarding but is loaded on demand. For these reasons we decided to build our system for anomaly detection based on programmable network nodes and to implement service modules which can generate indications of the occurrence of network hazards. If a new kind of hazards has to be detected we just want

to add specialized service modules which can detect this new kind of hazards without any changes to the rest of the system.

With DDoS attacks [6,9] which are a major threatening type the attacker does not exploit a weakness of the victim's operating system or application but aims to overload resources like link capacity or memory by flooding the system with more traffic than it can process. The attack traffic is generated by many slave systems which the attacker has compromised before. The attacker only has to coordinate all these slave systems to start the attack nearly at the same time against a single victim. As soon as the victim is not reachable anymore no reverse traffic is sent back to slave systems or error messages are generated by routers close to the victim. Such changes of the traffic can be detected by combining various anomalies.

Another threat to the Internet today are worms [15,11]. This piece of software automatically exploits security holes in operating systems or applications to infiltrate a system and starts to propagate itself to as many other systems as possible. Today's countermeasures to worms are signature-based detection systems scanning for well-known worms. These systems are typically located at the victim's edge of the internet preventing the worm propagation to a specific network. An earlier detection of such a worm propagation is possible if the detection system is located in the network itself. There a signature-based detection system is not applicable since it needs deep packet inspection which is infeasible without additional special-purpose hardware. Furthermore, a signature-based detection system is not able to detect previously unknown worms at all. To achieve a detection of unknown worms an anomaly-based detection system can be used. Such a system also has to be deployed within the network to ensure an optimal protection of the network. An anomaly-based detection system can collect hints on a worm propagation for example by analyzing the ratio of error messages due to closed ports generated by scanned systems to the total number of connection requests.

In wireless networks other hazards are possible in addition to DDoS attacks and worm propagations due to the different medium type. Since a wireless medium allows mobility and can not guarantee any knowledge about a participating node it is much easier to threaten the routing protocol or certain connections than it is in wired environments. A possible attack in wireless networks is the wormhole attack [8]. By establishing a wormhole an attacker aims at attracting as much traffic as possible to a node controlled by himself. If a proactive routing protocol is used this is achieved by influencing routing metrics in such a way that other nodes assume the attacker in their neighborhood due to the established tunnel. In fact however the attacker is far away. If a reactive routing protocol is used new routes are established through the tunnel since the tunnel enables the attacker to send a route request faster to the destination than this is possible over normal multi-hop communication. If the attacker succeeds in establishing a wormhole he can attack certain connections or the connectivity of great parts of an ad hoc network by dropping packets arbitrarily. Therefore, a wormhole attack causes an anomalous increase of traffic near the tunnel endpoints as well as an increasing drop rate of packets which are routed over the tunnel.

Our approach to build a hazard detection system uses anomaly-based detection functionality, e.g. stochastic anomalies, distribution anomalies or protocol anomalies. All these anomalies give hints to a current hazard. We analyzed various hazards in

different scenarios and concluded that a system which can easily be adopted to these scenarios would prove valuable. Such a system for an anomaly-based hazard detection is presented in this paper.

The paper is organized as followed: In section 2 we detail on the architecture of the system for anomaly detection and we explain the special characteristics of our system. Furthermore we describe two scenarios which we think the detection system can be deployed in – a small provider network and an ad hoc network. Section 3 presents implementation details for one of the described scenarios, the small provider network, and an evaluation of the system for anomaly detection. Additionally we will go into memory usage details. Section 4 gives a short summary.

1.1 Related Work

There are some existing approaches that try to detect DDoS attacks or worm propagations based on programmable networks. Some of these even use anomaly-based detection mechanisms. Approaches which mainly focus on mitigation and remediation are not discussed here. One approach of Sterne et al. [16] detects stochastic anomalies by using a simple threshold based DDoS detection mechanism. The system consists of three components: the DDoS flood detector, a management station which dispatches an active program in case of a DDoS detection and routers which are active networking nodes and are able to execute the active program. A drawback with this approach is that after detecting a DDoS flood by a sudden increase in packet distribution no further verification of the attack is done but immediately a rate limiter is installed on the active nodes. Another approach, IBAN [4], detects worm propagations based on active networks. Therefore, a management station distributes so called scanners on active networking edge routers. These scanners search for a specific vulnerability on all hosts which are connected to the edge router. This means that only known worms can be detected. For every newly announced vulnerability a new scanner has to be implemented first and distributed in the active network afterwards. If a vulnerable host is detected by such a scanner the management station is informed and the distribution of a blocker to the proper edge router has to be started manually. Thus, this approach uses some kind of signature-based detection and does not react automatically on worm propagations.

The pushback mechanism [10] is activated as soon as congestion occurs on a router. In this case a flooding attack is assumed and the packets which are dropped on the router due to congestion are inspected in more detail. The mechanism supposes that the distribution of dropped packets resembles the distribution of the whole packet stream and rate limits the highest bandwidth aggregate of packets. This is done for further aggregates until congestion has disappeared on the outgoing links. Afterwards rate limiters for the aggregates are installed in upstream active network routers to reduce congestion of the incoming links. This approach has several disadvantages: one of these is the fact that an attack can be detected not until congestion occurs on a router and hence a detection is only possible at the edge of the network. Another problem is the fact that no further verification is done if the rate limited aggregates really belong to an attack. AEGIS [3] proposes the deployment of so called shields which scan for several different anomalies and therefore, have to do a deep packet inspection which can cause a delayed forwarding of packets if no special-purpose hardware is used. Additionally,

a commander scans for further anomalies in the aggregated traffic behind the shields and tries to learn fingerprints of normal traffic to detect DDoS attacks. This procedure is likely to use much resources, primarily memory and processor time, and therefore, causes additional costs if new hardware is needed.

Our approach is based on the usage of a software system to achieve network anomaly detection. It is, however, possible to carry out some or all of the proposed detection mechanisms on network processors [13] or some other special-purpose hardware. This causes additional costs and changes to the currently deployed infrastructure but in some cases provides better performance of the system for network anomaly detection. In both approaches – software-based or hardware-based – resource management [7] is needed in order to use the available resources efficiently.

2 Architecture and Usage Scenarios

In subsection 2.1 we focus on the architecture of our system for network anomaly detection. Afterwards deployment of the system in two usage scenarios – a small provider network and an ad-hoc network – is introduced in more detail and anomalies usable for anomaly detection are described (see subsections 2.2 and 2.3).

2.1 Architecture

We developed a system for anomaly detection that is hierarchical, anomaly-based, extensible and flexible. A hierarchical system saves resources by splitting anomaly detection, respectively, into several stages. Thus, the common functionality of a node which runs an anomaly detection system is less affected by a hierarchical system than by a system that keeps its whole functionality in one stage. A hierarchical system for example ensures less memory usage on a router since only some easy calculations are needed in the basic stage. A system composed of only one stage has to maintain much more state for anomaly detection and therefore, causes a much higher memory usage. Thus, low resource consumption due to a hierarchical design ensures applicability in different kinds of networks. Because of this characteristic our system can even be deployed on nodes with limited resources like routers in small provider networks or PDAs in wireless ad-hoc networks.

Additionally, it is very easy to introduce new anomalies into the basic stage of the detection system or new aggregate specific anomalies. This flexibility also ensures an applicability in different network scenarios. Examples for the definition of aggregates in the basic stage and for detectable anomalies in suspicious aggregates in the second stage are given in this section.

The attack detection system analyzes in its basic stage the packet distribution within specific aggregates and scans for indications of an attack by detecting stochastic anomalies. Therefore, the packet stream is divided on the fly into intervals with a fixed length. Furthermore, aggregates of interest are defined for observation. The notion aggregate in this paper refers to a set of packets with the same characteristics and therefore, predefined aggregates are for example all TCP or all UDP packets in a small provider

network or all routing packets in an ad-hoc network. Then for each predefined aggregate the number of packets that belong to this aggregate is counted in every interval. An indication of an attack then is found if the observed number of packets exceeds a predefined packet threshold of the aggregate. To make the system self-adaptable to network load changes a dynamic *packet threshold* representing the average packet count in this aggregate for the last couple of intervals is calculated. At the end of every interval a check for each aggregate is performed if the observed number of packets exceeds the packet threshold. To prevent the system from generating false positive indications and starting the next stages for deeper inspections unnecessarily an *interval threshold* is defined. This interval threshold is necessary due to the self-similarity of internet traffic [12] which can cause normal traffic to exceed the packet threshold even though no attack is currently going on. Therefore, an indication only is generated and further detection stages are loaded if the packet threshold is exceeded in more consecutive intervals than the interval threshold. These detection stages will subsequently check the suspicious aggregate in more detail.

After detecting an indication for an attack or a network problem, respectively, by an exceeding of the threshold a second stage will be loaded. This second stage examines the suspicious aggregate in more detail by scanning for the currently defined aggregate specific anomalies. Additionally the detection system offers the possibility to load several consecutive stages for a more detailed analysis after detecting a stochastic anomaly in the basic stage. Because the suspicious aggregate is only a subset of the whole packet stream more detailed analysis can be done with low resource consumption and thereby low detraction of a node's common functionality.

The advantage of our system for anomaly detection is the fact that it can be used in very different networks and scenarios due to its flexibility. Dependent on the underlying network aggregates can be defined for observation in the basic stage which are likely to show some anomalies if an attack, a misconfiguration or a network problem occurs. Thus, in wired networks DDoS attacks and worm distributions are attack types that are common. Due to this a priori knowledge aggregates are defined based on transport protocols like UDP and TCP or on the network protocol ICMP. In a wireless ad-hoc network further attack types like the wormhole attack or network problems due to a misbehaving node are possible. Thus, in ad-hoc networks additional aggregates can be defined based on different packet types like routing and data packets.

2.2 Attack detection in a small provider network

The first example for the deployment of our system for anomaly detection is the detection of anomalies which indicate DDoS attacks or worm propagations in small provider networks since these attacks are the most prevalent hazards in this usage scenario. Therefore, the system scans for stochastic anomalies based on a packet and an interval threshold to detect attacks in the basic stage as described above. After detecting a stochastic anomaly in the basic stage the system loads two consecutive stages. The second stage uses a distribution anomaly to make a differentiation between DDoS attacks and worm propagations. This can be achieved by analyzing the distribution of packets into subnet prefixes based on destination addresses. Therefore, the whole address space

is divided into subnet prefixes based on the routing table of the node deploying the detection system. If large parts of the suspicious traffic – the number of packets by which the packet threshold was exceeded – are sent into exactly one subnet a DDoS attack is indicated since only one victim is currently attacked. If the suspicious traffic is equally distributed to all existing subnets a worm propagation is assumed since worms spread all over the internet. Dependent on the results of the second stage attack type specific protocol anomalies are scanned for in the third stage to identify either DDoS attacks or worm propagations in more detail. That means that only DDoS specific protocol anomalies are examined in the third stage if the second stage detected a distribution anomaly indicating a DDoS attack. Currently the anomalies used in our system for network anomaly detection offer no possibility to differentiate between DDoS attacks and legitimate traffic with the same characteristics, e.g. flash-crowd events [1]. Some example protocol anomalies that can be used to detect DDoS attacks or worm propagations in the third stage are described first. More protocol anomalies that can be used to detect worm propagations and other DDoS attacks can be found in [14].

Figure 1 shows the architecture of our detection system as deployed in a small provider network. It consists of three stages: The basic stage scans for stochastic anomalies within the predefined aggregates like *TCP packets*, *UDP packets*, and *ICMP packets*. The further stages are loaded on demand after a stochastic anomaly indicates a suspicious aggregate. The second stage differentiates between DDoS attacks and worm propagations by analyzing the destination address distribution in this aggregate. Depending on this analysis protocol anomaly detection modules for either a DDoS attack or a worm propagation are loaded.

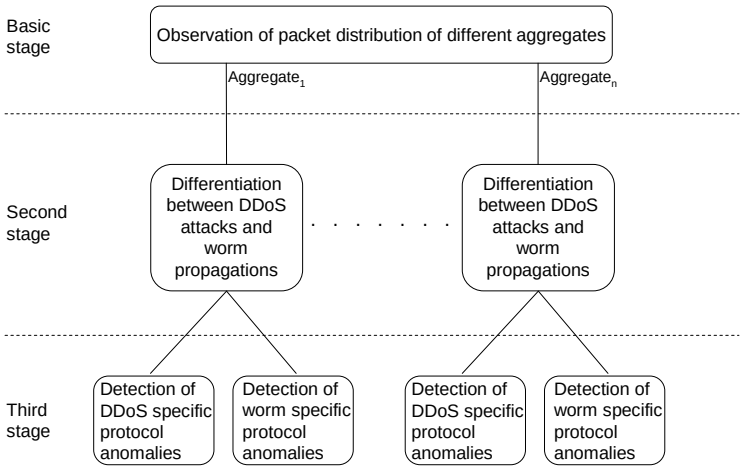


Fig. 1. Architecture of system for attack detection in small provider networks

As already mentioned the aggregates *TCP packets*, *UDP packets*, and *ICMP packets* are defined for attack detection in wired networks. The third stage of the detection

system is based on the fact that most of the existing DDoS attacks lead to a breach of symmetry between incoming and outgoing packet classes which belong together by protocol definition. A packet class here refers to a set of packets with the same characteristics, for example all TCP packets with SYN flag set. Thus, a DDoS attack can often be detected by a developing asymmetry of packet classes that belong together.

A SYN flooding attack for example tries to exhaust a victim's open connection storage space by flooding the victim with TCP packets with SYN flag set. Due to the mass of connection requests the victim can only respond to a part of all requests by sending TCP packets with SYN and ACK flag set. All remaining requests are dropped and the victim sends no response if storage space is already exhausted and the TCP instance is already down. This leads to an asymmetry between incoming TCP packets with SYN flag set and outgoing TCP packets with SYN and ACK flag set which can be used to detect this kind of DDoS attack.

Another possible DDoS attack is the smurf attack [2], named after its exploit program, which tries to exhaust the victim's bandwidth by flooding the victim with ICMP echo reply messages. Therefore, ICMP echo request messages with forged source addresses are sent to so called reflector systems. These reflector systems in turn response with ICMP echo reply messages to the forged sender address – the address of the victim. If an ICMP request packet is sent to the directed broadcast address of a whole subnet all systems in the subnet will answer the request and thus, amplify the strength of the DDoS attack. This attack leads to an asymmetry between ICMP echo request packets sent by the victim and ICMP echo reply packets received by the victim.

An example for a protocol anomaly that can be used for detection of a worm propagation utilizes the fact that a worm tries to infect other hosts randomly. Additionally, most vulnerabilities a worm tries to exploit are tied to a single port number. A worm propagation based on UDP sends packets with destination port set to the vulnerable port number to randomly selected hosts. If some of these hosts have patched their system already or the vulnerable port is closed anyway these systems send an ICMP packet containing the error message "port unreachable" back to the sender of the UDP packet. If the system or network does not exist at all an ICMP message "host/network unreachable" is generated. Because most worms propagate randomly the ratio of ICMP packets with these error message will increase during a worm propagation. This protocol anomaly can be used to verify a worm propagation detected already by a stochastic anomaly and a distribution anomaly.

2.3 Attack detection in an ad-hoc network

The second scenario we looked into are wireless ad-hoc networks. In ad-hoc networks the participating nodes themselves form the networking infrastructure in an ad-hoc fashion to achieve an autonomous, mobile, wireless domain. Our system can be deployed in such a network for anomaly-based detection of attacks, too. In addition to the previous described type of attacks ad hoc specific kinds of attacks can be detected, e.g. wormhole attacks. For this kind of attack an attacker uses a tunnel achieved for example by a directed antenna or by a wired link to send packets over only a few hops to an endpoint far away from the source node. Without the tunnel the packets would have been sent over multiple hops and would have taken significantly more time to reach the endpoint.

Due to the tunnel – and thus, the ”short” distance between source and destination – the attacker can achieve that much more traffic is sent over the tunnel endpoints controlled by himself than normal routing would do. Thereby, the attacker is able to attack certain connections or the connectivity of a great part of the network by dropping packets that are routed over the tunnel controlled by himself. Figure 2 shows the architecture for the our anomaly-based attack detection system in wireless ad-hoc networks.

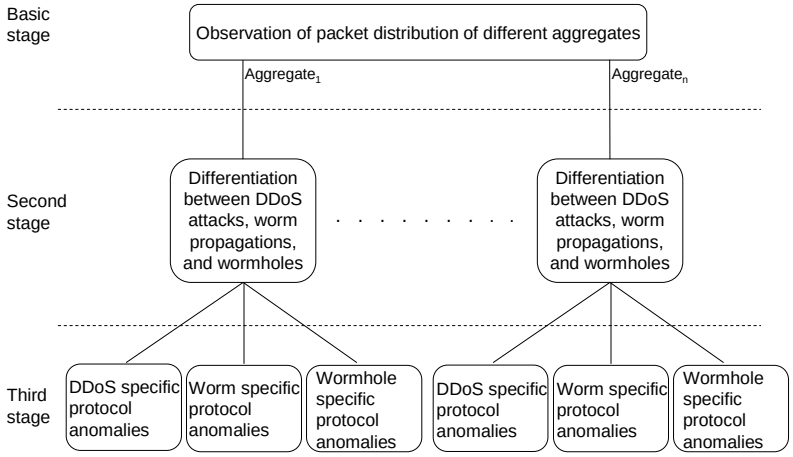


Fig. 2. Architecture of system for attack detection in ad-hoc networks

In case of an ongoing wormhole attack an increase in the number of data packets can be detected in the direct environment of the tunnel endpoints. This is a stochastic anomaly that can be detected by the basic stage. The subsequently loaded next stage analyzes the packets in more detail to distinguish the wormhole attack from DDoS attacks or worm propagations. The differentiation between DDoS attacks and worm propagations can be done by detection of a distribution anomaly described already in subsection 2.2 but with a slight difference: the address space is divided into host addresses instead of subnets due to the lower number of nodes in wireless ad-hoc networks. The differentiation between these two attacks and wormhole attacks can be done by another distribution anomaly. During an ongoing wormhole attack the distribution of next hop addresses in the forwarding table tends towards one address that is the next hop for almost all known destination addresses. To verify a suspected wormhole attack the third stage analyzes the suspicious aggregates and checks if the packet drop rate exceeds a certain threshold. This can be realized by a protocol anomaly module which scans for asymmetries in number of data packets to number of acknowledgement packets ratio for instance. Therefore, the third stage gets enhanced with this mechanism if deployed in a wireless ad-hoc network. The verification of worm propagations or DDoS attacks is done as described in the small provider scenario.

3 Implementation and Evaluation

The described attack detection system has been implemented on the programmable network platform FlexiNet [5] based on the Linux operating system. Thus, a service module can install iptables filter rules according to the PSAMP packet selection definition [18] which allows packet selection based on packet header field matching as well as packet sampling schemes. A selected packet is fed to the FlexiNet execution environment via the netlink interface. To preserve the packet order of all flows a copy of the selected packets is fed to the execution environment instead of the real packet which is forwarded regularly on the IP layer.

The basic stage of the anomaly detection system is implemented as a service module which is the only module loaded at system startup. This module processes the packets selected by the installed iptables filter rules. If the basic stage detects a stochastic anomaly in any aggregate by an exceeding of the packet threshold in more consecutive intervals than the interval threshold, specialized modules for stage two are loaded.

The basic stage monitors the number of packets within the predefined aggregates. For the simulation of the ICMP echo reply flooding attack we used a background traffic with an average data rate of about 3 Mbit/s. Due to the rather low bandwidth of the background traffic the following simulation and evaluation is only a first step towards a small provider scenario but nevertheless, we can show that the mechanisms of our detection system work. The average ICMP traffic within this background traffic was 1,8 kbit/s. The simulated attack traffic generated about 400 ICMP packets per interval resulting in an attack data rate of about 3,6 kbit/s of additional ICMP traffic. That is 0,12% of the overall traffic.

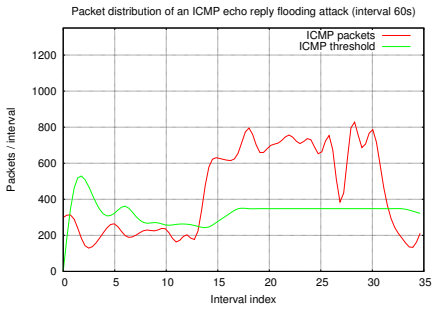
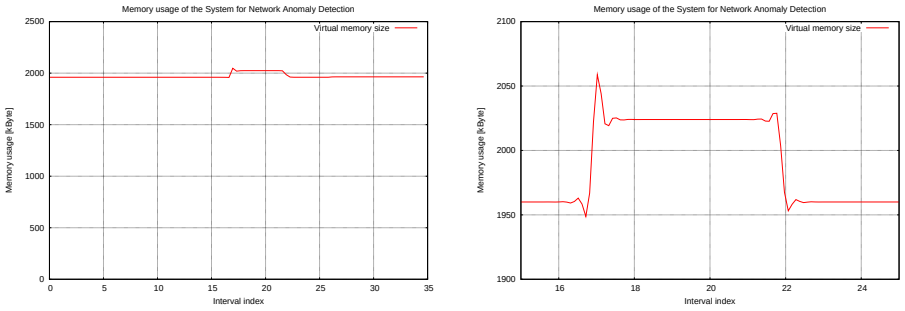


Fig. 3. Packet distribution of an ICMP echo reply flooding attack

The combined traffic – background and attack traffic – was analyzed by our attack detection system. This packet stream is shown in figure 3. Additionally the packet threshold which is used to detect attacks in the ICMP aggregate by a stochastic anomaly is shown. The red line shows the observed number of ICMP packets per interval whereas the green line shows the packet threshold. The threshold is calculated as described in

section 2.1 based on the aggregate's average packet count for the last couple of intervals. If an indication is generated due to an exceeding of the threshold for more consecutive intervals than the interval threshold, the threshold remains constant while the attack is running. We can clearly see that the simulated attack begins in interval 13 since the basic stage detected an indication for an attack in this interval. The exceeding of the packet threshold in more consecutive intervals than the interval threshold in the aggregate *ICMP packets* results in loading further stages of the detection system in interval 17. The second stage checks only the suspicious aggregate *ICMP packets* for a distribution anomaly which provides a differentiation between a DDoS attack and a worm propagation. In the simulation one specific subnet could be detected which most of the traffic is sent to by analyzing the distribution of suspicious packets to subnets. Thus, the third stage is loaded that checks only those packets of the suspicious aggregate that are sent into the suspicious subnet for DDoS specific protocol anomalies. In our simulation the third stage was able to detect an asymmetry between incoming ICMP echo reply packets and outgoing ICMP echo request packets as described in subsection 2.2.

Besides validating the functionality of our approach with this setup we also measured the resource consumption of our system. Figure 4a first shows the memory usage of the detection system during the ongoing attack detection described above. The graph shows the system's virtual memory size which is composed of code size, heap size and stack size. The lion's share of the memory usage of about 1750 kBytes is consumed by the execution environment itself. After loading the basic stage about 1960 kBytes virtual memory are used. In interval 17 memory usage increases due to the loading of the further stages. Figure 4b again shows the memory usage of the system during the attack detection phase but only intervals 15 through 25 with an adjusted y-axis scale. During this phase about 100 kBytes additional memory are allocated for the second stage. The third stage even needs just about 70 kBytes additional memory. After unloading the third stage in interval 22 the memory usage again is 1960 kBytes as it was before detecting a stochastic anomaly. These measurements show that our modular and hierarchical system needs just a small amount of virtual memory.



(a) Entire anomaly detection (b) A cutout

Fig. 4. Memory usage of the framework during an ongoing attack detection

4 Summary

In this paper we presented a system for network anomaly detection which is hierarchical, anomaly-based, extensible and flexible. These characteristics provide for the ability to be deployed in different network environments as well as the ability to detect various network hazards. DDoS attacks, worm propagations, and wormhole attacks are examples of such network hazards. We showed how stochastic anomalies, distribution anomalies, and protocol anomalies can be used to detect these and introduced the architecture of our detection system in a small provider network scenario and a wireless ad-hoc network scenario.

A simulation of an ICMP echo reply flooding attack shows that our anomaly-based system is able to detect DDoS attacks. Furthermore we verified that our hierarchical and modular system needs only a small amount of memory to realize the detection functionality.

In this paper simulation and evaluation were done only with low-bandwidth background traffic. Thus, future research has to address simulations using background traffic with a higher bandwidth to simulate a more realistic small provider network. In this context, the detection performance – e.g. the number of false positives – has to be examined. Furthermore, some work has to be done to achieve a differentiation between network hazards like DDoS attacks and legitimate traffic with similar characteristics, e.g. flash-crowd events.

References

1. I. Ari, B. Hong, E. Miller, S. Brandt, and D. Long. Managing flash crowds on the internet, 2003.
2. CERT. CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks, 2000. http://www.cert.org/advisories/CA-1998-01.html.
3. E. Y. Chen. Aegis: An active-network-powered defense mechanism against ddos attacks. In *IWAN '01: Proceedings of the IFIP-TC6 Third International Working Conference on Active Networks*, pages 1–15, London, UK, 2001. Springer-Verlag.
4. W. L. Cholter, P. Narasimhan, D. Sterne, R. Balupari, K. Djahandari, A. Mani, and S. Murphy. Iban: Intrusion blocker based on active networks. *dance*, 00:182, 2002.
5. T. Fuhrmann, T. Harbaum, M. Schöller, and M. Zitterbart. AMnet 3.0 source code distribution. Available from http://www.flexinet.de.
6. L. Garber. Denial-of-service attacks rip the internet. *Computer*, 33(4):12–17, 2000.
7. A. Hess, M. Schöller, G. Schäfer, A. Wolisz, and M. Zitterbart. A dynamic and flexible Access Control and Resource Monitoring Mechanism for Active Nodes. June 2002.
8. Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. Technical report, Department of Computer Science, Rice University, Dec. 2001.
9. A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks-extended. Technical Report ISI-TR-2003-569b, USC/Information Sciences Institute, June 2003. (Original TR, February 2003, updated June 2003).
10. J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium, Catamaran Resort Hotel San Diego, California 6-8 February 2002*, 1775 Wiehle Ave., Suite 102, Reston, VA 20190, February 2002. The Internet Society.

11. D. Moore, C. Shannon, and K. C. Claffy. Code-red: a case study on the spread and victims of an internet worm. In *Internet Measurement Workshop*, pages 273–284, 2002.
12. K. Park and W. Willinger. Self-similar network traffic: An overview. In *Self-Similar Network Traffic and Performance Evaluation*. Wiley Interscience, 1999.
13. L. Ruf, A. Wagner, K. Farkas, and B. Plattner. A detection and filter system for use against large-scale ddos attacks in the internet backbone. In *Proceedings of the Sixth Annual International Working Conference on Active Networking (IWAN 2004)*, October 2004.
14. S. Schober. Mechanismen zur Erkennung von Distributed-Denial-of-Service-Angriffen in IP-Netzen. Studienarbeit am Institut für Telematik, Universität Karlsruhe (TH), December 2003.
15. C. Shannon and D. Moore. The spread of the witty worm. *IEEE Security and Privacy*, 2(4):46–50, 2004.
16. D. Sterne, K. Djahandari, R. Balupari, W. L. Cholter, B. Babson, B. Wilson, P. Narasimhan, and A. Purtell. Active network based ddos defense. *dance*, 00:193, 2002.
17. L. Xie, P. Smith, M. Banfield, H. Leopold, J. Sterbenz, and D. Hutchison. Towards resilient networks using programmable networking technologies. In *Proceedings of IFIP IWAN 2005*, Nov 2005.
18. T. Zseby, M. Molina, F. Raspall, and N. G. Duffield. Sampling and filtering techniques for ip packet selection. Internet Draft, draft-ietf-psamp-sample-tech-06.txt, Work in Progress, Internet Engineering Task Force, February 2005.