Combining Multiple Intrusion Detection and Response Technologies in an Active Networking Based Architecture

A. Hess, M. Jung, G. Schäfer

Telecommunication Networks Group, Technische Universität Berlin, Germany
Email: {hess,jung,schaefer}@ft.ee.tu-berlin.de

Abstract: With the ever growing number of hosts connected to the Internet, representing potential sources of malicious attacks, and increasing sophistication of attacking techniques and automated attacking tools, network intrusion detection and response has evolved into a very active field of research in recent years and a wide variety of approaches has been developed [LFG$^+$00, NN01]. However, isolated operation of specific intrusion detection and defense technologies generally exhibits only the specific strengths and drawbacks of one particular approach. In order to allow for a co-ordinated combination of existing and emerging security technologies (e.g. signature based detection, anomaly detection, DDoS response mechanisms, honeypots, etc.) we propose a flexible intrusion detection and response framework called FIDRAN [HJS03] that is based on active networking technology. Principal findings so far are that active networking proves to be a well suited technology for intrusion detection and response, that the load of intrusion detection can be distributed among multiple systems with this approach, and that the overhead stays in acceptable ranges.

1 Introduction

Recent developments show that securing communication networks with singular and isolated techniques proves to be insufficient to cope with the vulnerabilities of today's networks in a timely manner. The reasons behind this trend originate from multiple developments. First, the steadily increasing number of hosts connected to the Internet implying an accordingly increasing number of vulnerable hosts offers an ever growing number of potential targets for malicious activities. Second, many private and professional users are not sensible to security vulnerabilities affecting their own machines or they are just overstrained patching these. Furthermore, many users believe that they will never become the target of an attack, due to irregular on-line times, changing IP-addresses or having the perception that their system or data, respectively, is not of value for potential hackers. Unfortunately, this is not true: As, for example, Lance Spitzner writes in his book [Spi02]: "On February 28, 1999, at 20:15 I put the honeypot online ... Within 15 minutes of my connecting the honeypot to the Internet, an attacker identified, probed, and exploited it". Beyond this, he states that a home network was scanned on average by 31 different systems a day in the beginning of 2002.

Another reason for the rising danger arising of malicious activities is the alarming evolution of the execution speed of computer attacks. Consequently, the time window to invoke countermeasures in order to limit the harm of an attack is shrinking [BAMF01]. Weaver claims in [Wea] that it is possible to construct hyper-virulent active worms which are capable of infecting all vulnerable hosts of the Internet in approximately 15 minutes to an hour. Furthermore, the authors of [SJ] argue that under certain conditions a small worm "can infect almost all vulnerable servers on the Internet in less than thirty seconds".

As can already be seen from this short abstract of security problems in current communication networks, the currently existing security technologies on their own are not capable to react or be adapted to new attacks and changing requirements in a sufficiently timely manner. However, as each security technology has its specific advantages and drawbacks, a security infrastructure is needed that is able to combine as many technologies as possible to minimize their drawbacks and to combine their strengths.

In order to realize such a security infrastructure we developed *FIDRAN - a Flexible Intrusion Detection and Response Framework for Active Networks*. This framework allows the cooperation of traditional (firewall, intrusion detection system, etc.), innovative (DDoS defense, honeypots, etc.) and emerging security technologies in order to adequately secure communication networks. FIDRAN is build on top of an underlying active networking environment which allows to dynamically deploy new security modules on FIDRAN-hosts. Its modular design provides the infrastructure for a constructive cooperation between modules of different security technologies. In addition, the active networking infrastructure facilitates maintenance work and enables the distribution of security tasks among different FIDRAN-hosts.

2 Security Technologies

In this section we give a short introduction to different security technologies, including their strengths and drawbacks.

A *firewall* is the juncture between a protected subnetwork and a less trusted network. Thereby the network firewall is responsible for the following tasks:

- it restricts traffic to entering at one carefully controlled point.
- it restricts the traffic that may enter corresponding to rules (source-address, destination-address, protocol, etc).
- it restricts traffic to leaving at one carefully controlled point.

In general firewalls cannot protect against new threats, viruses and malicious insiders.

Intrusion Detection is the process of identifying and responding to malicious activity targeted at computing and networking resources [Amo99]. In general IDSs are categorized into the two types network intrusion detection systems (NIDS) and host-based IDS. A NIDS monitors packets on the network wire and attempts to discover if a hacker is attempting to break into a system. A typical example is a system that watches for large number of TCP connection requests (SYN) to many different ports on a target machine thus, discovering if someone is attempting a TCP port scan.

The evaluation method whether or not an attack is taking place can be classified into anomaly and signature detection. The anomaly technique assumes that all intrusive activities are necessarily anomalous. Thus, if it would be possible to create a "normal" behavior pattern of a system / communication network, every occurrence that does not accord to this pattern would be identified as an intrusion. Thereby, the main problem is the creation of the behavior patterns. Different approaches have been discussed including e.g. neuronal networks but still many problems exist. Another drawback of this method is that it is not possible to identify the attack by name. The system detects an anomaly but it is not able to link the anomaly with a concrete attack and hence, it is difficult to invoke adequate countermeasures.

In contrast thereto, the signature detection technology scans the network traffic for known attack signatures. "When a signature for an attack matches observed traffic, an alert is generated. A simple signature example is the Land attack; the source and destination IP addresses in a packet are the same” [SN02]. The strength of this approach is that each detected attack is named correctly which helps invoking the correct response. But one drawback hereby is that a signature-based IDS is only able to detect known attacks and that the signature database must be continuously maintained. Furthermore, through the crafting of numerous packets which match attack signatures, alarms on an IDS can be conditioned or disabled and then exploited [PYD]. An attacker who uses existing evasion techniques could trick signature-based IDSs. For example one instruction belonging to the Nimda-attack [nim] trying to exploit the so-called “Directory Traversal” vulnerability could look like:

- GET /SCRIPTS/../../winnnt/system32/cmd.exe?/c+dir,
- GET /SCRIPTS/../../winnnt/system32/cmd.%65xe?/c+dir,
- GET /SCRIPTS/..%c0%af../winnnt/system32/cmd.exe?/c+dir.

The given instructions look different but they are identical. Either an attack signature is generated for each thinkable permutation or appropriate functions are integrated into the IDS. In the first case, an enormous amount of attack signatures must be managed by the IDS. Thus, a modern IDS should be able to upgrade the set of attack signatures and to integrate new functionalities in order to react on newly detected evasion techniques. In section 2.1 we discuss in detail signature-based NIDS, as they are predominantly used today.

Honeypots are systems designed to be probed, attacked and compromised by an attacker [Pro01]. A honeypot can be used as alerting mechanism, to trick attackers, to slow down attacks or to gain information about attackers.

It must be emphasized that a honeypot is only of value when it is spotted by an attacker. Generally there are two types of honeypots. The first type could be directly contacted by an attacker and any communication between such a honeypot and another host is suspicious due to the simple fact that a normal user does not have any reason to contact the honeypot. Thereby, the detection of unknown automated attacks is possible, as such an attack would sooner or later contact all vulnerable hosts. The second type are respond mechanisms to certain attacks. In the case that an attack has been detected by a NIDS, the attack is redirected to an adequate honeypot in order to collect more information about the attacker or to slow it down.

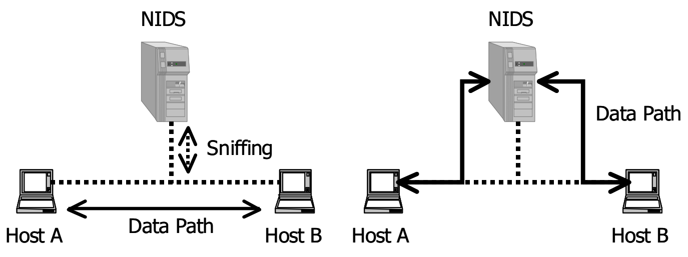
Table 1: A Comparison of existing Security Technologies

Technology	Function	Strength	Drawback
Firewall	blocks or permits traffic according to specified rules	real-time filtering	no detection functionality
Signature-based IDS	screens network traffic for known attack traffic patterns	clear identification of attacks	unknown attacks, evasion and insertion techniques
Anomaly-based IDS	observes network traffic for anomalies	detection of unknown attacks	high false alarm rate
Honeypot	security device designed to be attacked	detection of unknown attack and low false alarm rate	only useful when attacked

2.1 Signature-Based Network Intrusion Detection Systems

According to the Common Intrusion Detection Framework (CIDF) [SC] an IDS consists of an event generator, an analysis engine, a storage mechanism and under circumstances it also includes a set of countermeasures. Regarding a NIDS, the event generator is the component that gathers the data from the underlying network. Most current NIDSes do a so-called 'passive' protocol analysis. This mean a promiscuous network device or a sniffer obtains copies directly from the network regardless of their destinations (see figure 1) and hands them to the analysis engine. The strengths of the passive protocol analysis are once that it does not degrade the network performance and beyond this, the presence of a promiscuous network device is hardly to detect by other machines. But there are also obvious drawbacks. Passive protocol analysis IDSes are so-called 'fail-open', which means that the IDS does not provide any further protection in case that it is disabled. In contrast, a 'fail-closed' IDS would block all traffic in case that it crashes. Another problem of the passive protocol analysis approach is the reassembling of overlapping packet fragments. As different OS reassemble overlapping fragments in different manners, the NIDS should behave in the same manner as the end-hosts which it protects. This means that in case of overlapping fragments addressed to a Windows NT 4.0 host, the NIDS must favor old data while reassembling. If the fragments are addressed to a Linux host, the NIDS must favor new data while reassembling. Consequently, a NIDS must have a certain knowledge about the network it protects in order to interpret the packets accurately. A related problem is the question which packets a NIDS should accept in order to process the identical packets as the end-hosts. For example some end-hosts will accept source routed packets, but many will not.

The other possibility in contrast to the passive protocol analysis would be to capture the packets from the network. This would diminish the above described problem of accuracy. The NIDS-host would process all incoming packets in the same manner such that for example all packets are reassembled in the same way by the NIDS-host. Furthermore, this



(a) Passive (b) Active

Figure 1: Protocol Analysis

approach is fail-closed, as no further packets would be processed in case of a NIDS crash. One drawback is the impact on the network performance.

Next, the analysis engine evaluates whether or not an attack is taking place. It compares the information provided by the event generator to a large databases of known attack signatures. Finally, the task of the storage mechanism is to log a specified amount of packets and additional information in order to make this data available to the network administrator. Finally, some NIDS are equipped with a set of countermeasures, as the reconfiguration of the local firewall, in order to prevent further attacks.

2.2 Why Active Networks

At first, the network, that we assumed, consists of routers, active nodes, a service repository (SR), a network administrator and end-systems. An active node is able to execute services which can dynamically be downloaded from the SR.

While each of the main intrusion detection and response technologies, e.g. signature based detection, anomaly based detection, honeypots, IP traceback and pushback, general packet filtering, etc., has its principle advantages, isolated deployment of one single technology is often not sufficient to counter with dedicated and coordinated attacks. Therefore, we advocate a coordinated operation of multiple intrusion detection and response technologies on top of an active networking environment in order to combine the strengths and overcome the weaknesses of specific approaches.

One example for this is the weakness of signature based intrusion detection systems (IDS) to be quickly overloaded when being intentionally triggered by a malicious attacker [PYD]. However, analyzing the output of a signature based IDS with an anomaly detection based IDS may allow to identify attacks that are directed against the signature based IDS. Another example is the combination of a signature based IDS to identify well known attacks with honeypots that allow to detect unknown attacking patterns.

A further example is the distribution of security operations. According to the argumentation given in section 2.1, we decided to realize an intrusion detection framework that does an active protocol analysis. Consequently, the design of the framework must be aware of the potential performance impact (see also section 4) and thus, our framework provides the possibility to distribute security operations among different hosts (see figure 2).

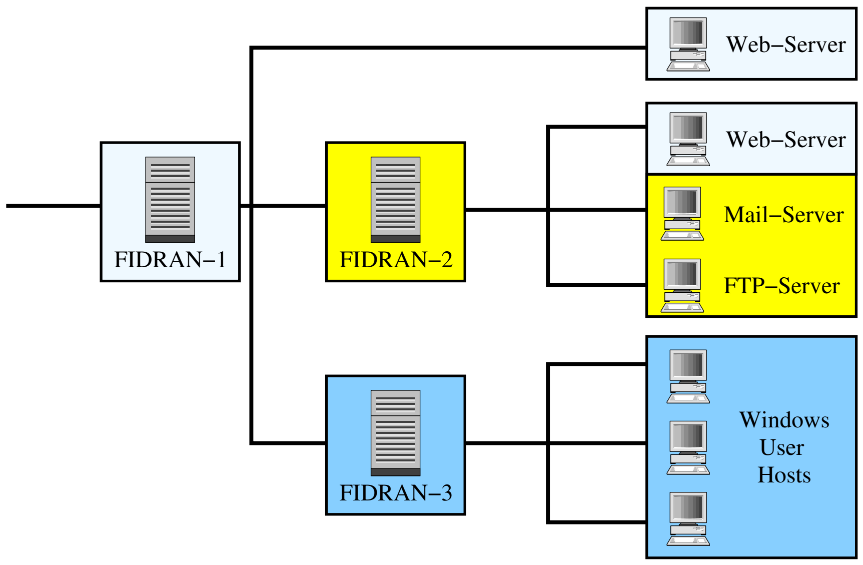


Figure 2: Distribution of Security Tasks

Another benefit of our approach on top of an active networking environment is the simplification of management tasks. At first, the framework allows to dynamically integrate and remove op-modules on a FIDRAN-host. Consequently, the placement of new security functions takes place in an automated manner. For example, an op-modules that contains the detection intelligence for a new attack can be placed on the service repository. Next, all FIDRAN hosts can be triggered to contact the service repository in order to check the new op-module according its operation area. If it is the case that the security policy of a FIDRAN-host orders the integration of the op-module, then the FIDRAN-host does.

3 The FIDRAN Architecture

In the design of our framework the following requirements are taken into account: Firstly, the framework should allow to distribute the intrusion detection and response tasks over multiple systems of a network in order to realize a subnet/node-specific protection, to scale the amount of security tasks to be performed per FIDRAN-host (e.g. limiting the amount of attack signatures per FIDRAN-host), and to be able to also detect insider attacks that would remain undetected in networks with one centralized IDS. Secondly, the framework should be realized following a modular concept, allowing to combine different security technologies, to dynamically extend functionality through the integration of new modules, and to facilitate and accelerate maintenance and configuration of the intrusion detection and response infrastructure. Finally, the underlying base mechanisms have to be highly efficient in order to be able to cope with today's high traffic volume.

The FIDRAN architecture which is depicted in figure 3 consists of a management module, a control module, a security policy, an alarm evaluation module and a varying set of kernel and user space op-modules (anomaly detection, signature detection, bandwidth monitor, etc.).

The FIDRAN management module is a resistant user space process, it performs an initial check on each newly arrived FIDRAN op-module. If the op-module passes this check, then the management module integrates it into the system. Furthermore, in the event of detecting a specific attack, it triggers the corresponding response mechanism which is specified in the security policy.

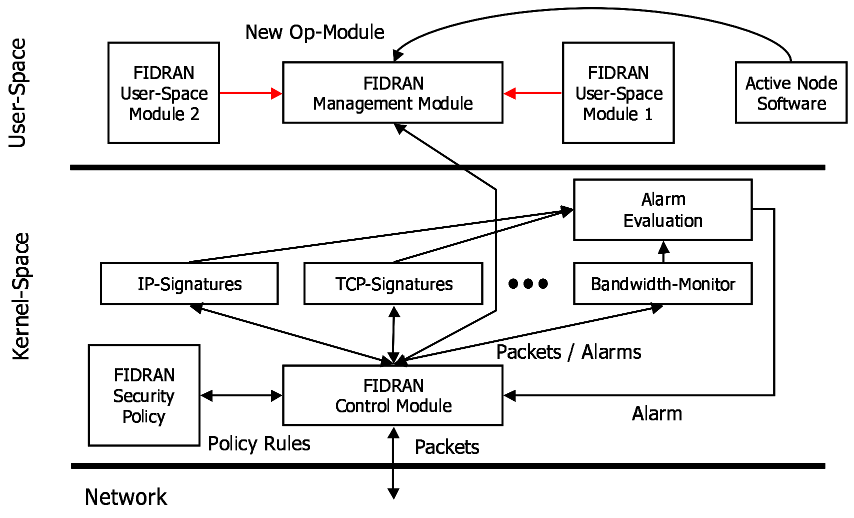


Figure 3: The FIDRAN Architecture

The FIDRAN control module is the central unit of the system in kernel space. It registers / unregisters the op-modules which are loaded / unloaded into kernel space by the management module. Additionally, it observes the network traffic and distributes the packets according to the security policy among the op-modules. Further on, the control module is able to analyze and to forward the alarms that are triggered by the op-modules.

A FIDRAN op-module performs its individual set of operations on the traffic specified in the security policy and sends the result to the FIDRAN control module. An op-module can for example include signature-detection functionality, anomaly detection algorithms or any kind of security intelligence.

The FIDRAN security policy [HS03] defines the set of FIDRAN modules that can be and must be loaded on the system. Furthermore, it specifies the security operations which must be performed by the system. Finally, the response mechanism to specific attacks are set in the security policy.

Third party modules are deployed by the active networking software like a normal active service in user space. Furthermore, these modules are supervised by an integrated access control and resource monitoring mechanism [HSS+02] such that the FIDRAN system can not be harmed by them.

3.1 Distribution of Security Operations

Basing on the possibility to distribute security operations we intend to realize a subnet respectively node specific protection, which is depicted in figure 2. FIDRAN could either be running on a single node, e.g. on the gateway between a subnet and the Internet or it could be running on several hosts of a subnet. In the former case, the FIDRAN-gateway is performing all security tasks on its own, whereas in the latter case, the security operations to be performed by FIDRAN are distributed among several hosts. The specification which security operations to execute on what traffic is specified in the security policy.

With the aid of the security policy the network administrator is able to specify and to distribute security duties among several FIDRAN-hosts. Regarding figure 2, the FIDRAN-1 node scans the traffic which is addressed to the web servers. Consequently only web-server specific op-modules are running on that node. Further on, FIDRAN-2 inspects the traffic which is either destined to the mail server or to the ftp-server and finally, FIDRAN-3 protects the user hosts running a Windows OS.

Summarizing, security responsibilities and the thereto appertaining work are distributed among the involved FIDRAN-hosts. This feature allows to individually scale the amount of network traffic to be analyzed per FIDRAN-host which again results in a more efficient supervision. Thereby, the FIDRAN-system becomes more resistant against evasion techniques.

3.2 A FIDRAN Op-Module

In this section we present a simple TCP null scan op-module. An attacker uses the TCP null scan technique for OS specification. If a TCP probe packet with no flags set is sent to a closed port, a RST/ACK packet is recieved, whereas if the port is open no reponse is received.

```
#include "op.c"

struct op_t op = {
   version:     1.0,
   description: "tcp null scan",
   author:      "fidran",
   ostype:      "",
   protocol:    F_TCP,
   call:        F_ANYHOOK,
   priority:    50
};
```

```
int opLoad() { return 0; }

void opUnload() { }

int opApply() {
      if (!F_tcp->syn && !F_tcp->fin && !F_tcp->rst &&
          !F_tcp->ack && !F_tcp->psh && !F_tcp->urg) {
         F_alarm("tcp null scan from %d.%d.%d.%d:%d to %d.%d.%d.%d:%d",
                    NIPQUAD(F_ip->saddr), F_tcp->source,
                           NIPQUAD(F_ip->daddr), F_tcp->dest);
         return F_DROP;
      }
   return F_ACCEPT;
}
```

The struct 'op' contains the following module-specific information: version, a short description, author, OS, protocol, hooks and finally the priority. Most of the fields are self-explanatory and are not further described.

The OS field lists the operating systems which are protected by the op-module (blank OS = all). This means that we exploit the fact that vulnerabilities and attacks can be categorized with reference to their potential victims (e.g. OS, application software) and which allows the realization of a demand driven intrusion detection and response infrastructure, as it usually makes little sense to scan traffic for attacks targeted at a Windows OS, while the subnet to be protected consists solely of Linux hosts. However, if a network administrator intends to scan for vulnerabilities of multiple operating systems, this is, of course, possible.

The field 'call' contains the events to which the op-module registers itself. Generally, there are two events, either the arrival of a packet or the resounding of an alarm which has been triggered by an op-module. Thus, the author of an op-module has the possibility to register his module to certain events. The benefit of the two events (alarm or packet arrival) is that there are two different types of op-modules. The first type, which we call layer-1 op-modules, processes and evaluates network packets. In contrast the second type so-called layer-2 op-modules processes and evaluates the alarms, which are launched by layer-1 op-modules. In section 3.3 we discuss in detail the benefit of this approach.

Furthermore, our approach provides the possibility to label an op-module with a priority. The control module uses linked lists to handle the individual execution sequences of packets (a TCP-packet addressed to a Windows host traverses a different set of op-modules than an UDP-packet addressed to a Linux machine) and the priority determines the position of an op-module inside these linked lists.

3.3 The Alarm Evaluation Module

A particularity of our approach is the possibility to integrate an alarm evaluation module, either in user-space or in kernel space, as a second-layer op-module. This allows to evaluate the collectivity or a specified subset of all alarms in a new manner.

A fundamental drawback of signature-based IDSes is the vulnerability to high false positive rates. Existing tools like Stick [Gio] or PCP [PYD] are designed to exploit this vulnerability through the creation of packets which trigger a false positive on the IDS. A false positive occurs in the case that 'normal' activity triggers the IDS. The ability to craft such packets can be exploited to deactivate the according alarm on the IDS. Thus it appears that one big question is how to react to false alarms! Even further, Axelsson argues in [Axe99] that the false alarm rate is the limiting factor for the performance of an IDS.

Hence we follow the approach of an alarm evaluation module. The alarm evaluation module keeps track of the triggered alarms. Now, if the alarm rate increases in an uncommon way, the alarm evaluation module is also able to launch an alarm which invokes specified countermeasures. One possibility would be the installation of a further op-module which evaluates the packets in a different manner. The success of artificially crafted 'false positive' packets strongly depends on the detection algorithm of the IDS. Thus, our approach provides the possibility to include new algorithms in order to minimize the drawback of a static analysis engine.

4 Measurements

In order to obtain a preliminary performance assessment of our first prototype we carried out the following experiment: A file of of length 1GByte was transmitted via FTP from a server to a client interconnected with a Fast Ethernet network (100 MBit/s). The packets exchanged during this transmission were routed through and analyzed by a FIDRAN system running on a separate host. All hosts were Pentium III machines running at 800MHz frequency under the Linux 2.4 operating system. The FIDRAN system receiving the packets via the netfilter mechanism of Linux performed 10.000 times a simple filter lookup, that checks for every packet if it is a TCP segment and, if so, whether at least one of the TCP-flags SYN, ACK, FIN, RST, URG, or PSH is set.

Conducting this experiment we obtained the following results: When directly forwarding the packets without involvement of FIDRAN but including netfilter processing the transfer of the file took 98.31 s to complete. Additionally routing each packet through the FIDRAN control module lead to a total of 99.12 s and performing the aforementioned 10.000 filter lookups resulted in a total transfer time of 215.3 s. This test was conducted in order to get an impression of the impact of FIDRAN on a real application as a FTP-transfer.

Next, we measured, with the help of Iperf [ipe], the impact of FIDRAN on the network performance. IPerf is a tool to measure the maximum TCP bandwidth between two communication endpoints. Figure 4 depicts the outcome and table 2 presents the corresponding figures of the experiment. Again we used the above described op-modules which perform a check of the TCP-flgs. We started the experiment with no op-module installed and ended it with 1000 active op-modules. According to table 2 the throughput for a number of 100 installed op-modules is 96.8 % of its maximum.

5 Related Work

Snort [snoa] is able to sniff network traffic (passive protocol analysis) and compares it with known attack signatures. However, if the attack is sufficently distributed (spatially / temporally), then in most cases Snort will not detect the attack. Further on, Snort-Inline [snob] is a modified version of Snort, which is also able to drop or modify packets. Both Snort version do not provide the possibility to integrate dynamically new detection algorithms.

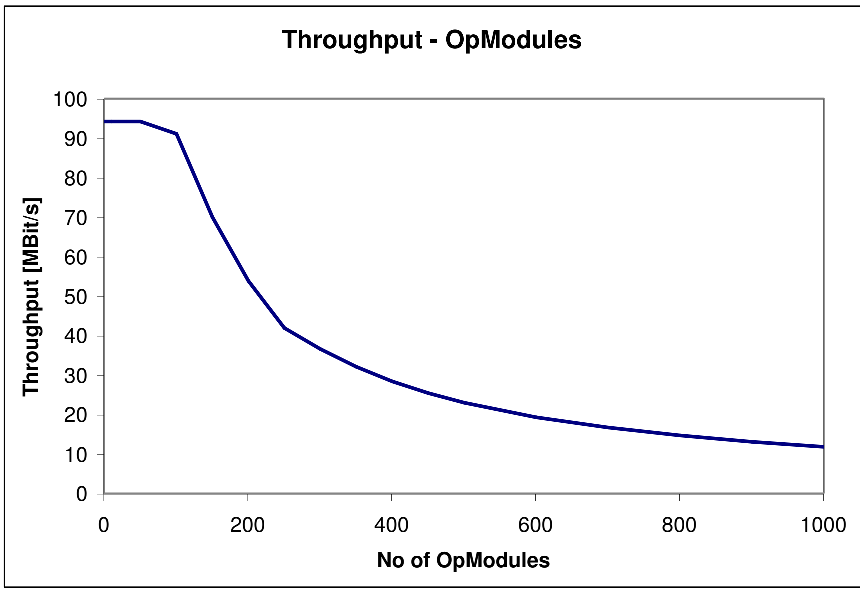


Figure 4: The Influence of Op-Modules on the Throughput

Table 2: The Influence of OpModules on the Throughput

	0	**50**	**100**	**150**	**200**	**250**	**500**	**1000**
Throughput [Mbit/s]	94.1	94.1	91.0	69.9	53.8	41.8	22.9	11.7
Rel. Throughput [%]	100	100	96.8	74.3	57.2	44.2	24.3	12.4

The paper Active Network Based DDoS Defense [ea02] describes how active networking technology can be used for DDoS protection. The presented approach consists of a sensor which remarks a rapid increase of network traffic and a mobile traffic rate limiter which clones itself. The rate limiter migrates upstream along the attack path in order to stem the attack.

The Intrusion Blocker based on Active Networks - IBAN [ea] consists of a management station, mobile vulnerabilities scanners, and mobile intrusion blockers. A mobile scanner is an application designed to detect one particular vulnerability by looking at system fingerprints. If the scanner has found a vulnerable service an intrusion blocker is placed close to the corresponding system which inspects the traffic for the vulnerable service and blocks the traffic if it detects an attack attempt. IBAN focuses on the detection of automated known attacks. A scanner and a blocker are designed for one particular vulnerability. A mobile application is designed for a particular vulnerability. Consequently, numerous mobile applications could exist in an average network. Further on, each application observes the traffic for a specific traffic pattern, thus each mobile application performs a set of identical operations

The FLAME project [AIM$^+$02] allows users to install kernel modules for real-time packet monitoring. The code is written in Cyclone and is processed by a trusted compiler. A set of credentials is used at compile time to verify that the module is authorized to perform the requested actions. Even if the compiler could be trusted it is still dangerous to install user modules in kernel space. In contrast FIDRAN allows the loading of kernel modules which originates from trusted sources. Further on, a control module coordinates the kernel modules which improves efficiency.

Summarizing, we state that few projects exist which exploit the possibilities provided by an active networking environment or which allow the integration of different security technologies.

6 Conclusion and Future Work

In this paper we described a *Flexible Intrusion Detection and Response Framework based on Active Networking (FIDRAN)* that allows to combine multiple intrusion detection and response technologies in order to join their individual strengths and to overcome their specific weaknesses. The design of FIDRAN allows to dynamically add new functionality and to reconfigure the system at runtime. The distribution of security operations among FIDRAN hosts helps both to make FIDRAN itself more resistant against attacks and to scale the load on a per host basis. Additionally, the active networking infrastructure allows to dynamically relocate op-modules in order to keep the load on each FIDRAN host under a certain upper limit. First experiments with a prototype show, that the overhead caused by the screening for simple attack patterns is within an acceptable range.

Future work will go towards a fully automated placement of FIDRAN op-modules. Currently, the network administrator specifies manually which op-modules can be placed on which FIDRAN-host. Actually, we are working on a network scanner, which analyzes specified network ranges in terms of operating systems and running services and which will inherit the task of placing FIDRAN op-modules. Certainly, the network administrator will still have the right to overrule the network scanner and to specify his own placement strategy, but the goal of FIDRAN is a self-regulating and distributed intrusion detection and response framework.

References

[AIM$^+$02] K. G. Anagnostakis, S. Ioannidis, S. Miltchev, J. Ioannidis, Michael B. Greenwald, and J. M. Smith. Efficient Packet Monitoring for Network Management. In *Proceedings of IFIP/IEEE Network Operations and Management Symposium (NOMS) 2002*, April 2002.

[Amo99] Edward Amoroso. *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response*. Intrusion Net Books, 1999.

[Axe99] Stefan Axelsson. The Base-Rate Fallacy and Its Implications for the Difficulty of Intrusion Detection. In *ACM Conference on Computer and Communications Security*, pages 1–7, 1999.

[BAMF01] H. K. Browne, W. A. Arbaugh, J. McHugh, and W. L. Fithen. A trend analysis of exploitations. In Francis M. Titsworth, editor, *Proceedings of the 2001 IEEE Symposium* *on Security and Privacy*, pages 214–231, Los Alamitos, CA, May 14–16 2001. IEEE Computer Society.

[ea] W. La Cholter et al. IBAN: Intrusion Blocker based on Active Networks. In *Proc. of Dance 2002*.

[ea02] Dan Sterne et al. Active Network Based DDoS Defense. In *Proc. of Dance 2002*, 2002.

[Gio] Coretez Giovanni. Fun with Packets: Designing a Stick. http://www.eurocompton.net/stick/papers/Peopledos.pdf.

[HJS03] A. Hess, M. Jung, and G. Schäfer. FIDRAN: A flexible Intrusion Detection and Response Framework for Active Networks. In *Symposium on Computers and Communications (ISCC'2003)*, 2003. accepted for publication.

[HS03] A. Hess and G. Schaefer. A Flexible and Dynamic Access Control Policy Framework for an Active Networking Environment. In *Proc. of Kommunikation in Verteilten Systemen (KiVS 2003)*, Leipzig, Germany, February 2003. accepted for publication.

[HSS$^+$02] A. Hess, M. Schoeller, G. Schaefer, M. Zitterbart, and A. Wolisz. A dynamic and flexible Access Control and Resource Monitoring Mechanism for Active Nodes. In *Short Paper Proc. of OpenArch 2002*, pages 11–16, New York, USA, June 2002.

[ipe] IPerf. http://dast.nlanr.net/Projects/Iperf/.

[LFG$^+$00] Richard Lippmann, David Fried, Isaac Graf, Joshua Haines, Kristopher Kendall, David McClung, Dan Weber, Seth Webster, Dan Wyschogrod, Robert Cunningham, and Marc Zissman. Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, Los Alamitos, CA, 2000. IEEE Computer Society Press.

[nim] Nimda Attack. http://www.cert.org/advisories/CA-2001-26.html.

[NN01] D. Northcutt and J. Novak. *Network Intrusion Detection – An Analyst's Handbook*. New Riders, 2001.

[Pro01] The Honeynet Project. *Know Your Enemy*. Addison Wesley, 2001.

[PYD] Samuel Patton, William Yurcik, and David Doss. An Achilles' Heel in Signature-Based IDS: Squealing False Positives in SNORT.

[SC] S. Staniford-Chen. Common Intrusion Detection Framework. http://www.isi.edu/gost/cidf/.

[SJ] Grim G. Staniford, S. and R. Jonkman. Flash Worms: Thirty Seconds to Infect the Internet. http://www.silicondefense.com/flash/.

[SN02] Scott Winters Karen Kent Frederick Ronald W. Ritchey Stephen Northcutt, Lenny Zeltser. *Network Perimeter Security*. New Riders, 2002.

[snoa] Snort. http://www.snort.org.

[snob] Snort-Inline. http://www.honeynet.org/papers/honeynet/tools/.

[Spi02] Lance Spitzner. *Tracking Hackers*. Addison Wesley, 2002.

[Wea] Nicholas C Weaver. Warhol Worms: The Potential for Very Fast Internet Plagues. http://www.cs.berkeley.edu/ nweaver/warhol.html.