Fraunhofer
FIRST

Ingmar Schuster, Tammo Krueger, Christian Gehl, Konrad Rieck, Pavel Laskov

FIPS: FIRST Intrusion Prevention System

FIRST Reports 1/2010

FIRST Reports
Herausgegeben von
Prof. Dr.-Ing. Stefan Jähnichen

ISSN 1613-5024

FIRST Reports 1/2010

Die FIRST Reports können bezogen werden über:

Fraunhofer-Institut für Rechnerarchitektur
und Softwaretechnik FIRST
Kekuléstraße 7
12489 Berlin

Tel.: ++49 (0)30 6392 18 00
Fax: ++49 (0)30 6392 18 05
E-Mail: first@first.fraunhofer.de
Internet: www.first.fraunhofer.de

FIPS: FIRST Intrusion Prevention System

Ingmar Schuster, Tammo Krueger, Christian Gehl, Konrad Rieck, Pavel Laskov

Fraunhofer Institute FIRST.IDA,
Kekuléstr. 7, 12489 Berlin, Germany,

E-mail: {sching,krutam,cgehl,rieck,laskov}@first.fraunhofer.de.

Abstract

Intrusion Prevention Systems try to actively disarm attacks on computer systems and networks. In this work, we introduce the network based FIRST Intrusion Prevention System (FIPS) which is capable of detecting novel attacks and contain them effectively. This inline device operates by redirecting anomalous packets to a specially hardened shadow system or logging them to a so-called forensic sink for further examination. Both the offline and real life evaluation of the implementation shows that the system yields very high accuracy rates and is faster than comparable standard solutions. Efficient retraining procedures are introduced to readjust the anomaly detectors after some time of deployment to further boost the accuracy for real life tasks.

1 Introduction

As organizations rely on computer systems to carry out important business tasks, computer security is constantly rising in importance. Computers are attacked over the internet as well as by local users who misuse their privileges or attempt to gain additional privileges they are not supposed to have. Successful attacks, also called *intrusions*, can compromise confidentiality, availability and integrity of computers and networks. Intrusion Prevention Systems (IPSs) monitor events on a computer or network, automatically analyzing them for possible attacks and trying to counteract. Having evolved from Intrusion Detection Systems (IDSs), they enable organizations to protect their systems from known or even unknown threats, depending on the attack detection techniques used.

This work describes the FIRST Intrusion Prevention System (FIPS), a network based IPS using anomaly detection. Network based IPSs work by examining network traffic for possible attacks. If network traffic is classified as an attack, an automatic procedure is triggered that tries to prevent the attack from being successful. In this work, mechanisms are presented for attack detection and a reasonable response. The attack detection method works by first modeling legitimate packet payloads and afterwards comparing new payloads with this model. If the payloads do not fit the model, they are classified as anomalous. This approach has the advantage that attacks that are yet unknown can potentially be detected, simply because they do not fit the model of normality. The automatic response is two-part: if the suspected attack occurs early enough in the connection, it is transparently redirected to a hardened shadow system that takes special care to prevent attacks but does so at a computational cost. If the suspected attack occurs later in the connection, all packets from the attacker are logged but not delivered.

The goals of this work are to implement and evaluate an industrial strength implementation of an IPS in order to test previous work [5] in a real world environment. Therefore after introducing the basic concepts and implementation details we focus on both the offline and real life evaluation of FIPS. We give detailed accuracy results both for data collected offline and online during a period of 1.5 months. Furthermore concepts for retraining the anomaly detector using data collected in the forensic sink are evaluated. Runtime measurements of the system and an in-depth evaluation of the real life performance of FIPS complete the experimental section. Related work and future directions of FIPS conclude the paper.

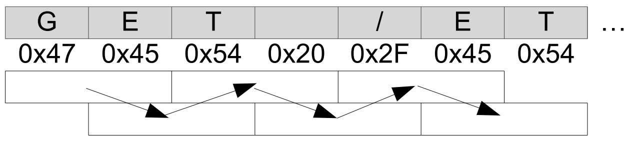

Figure 1: Sliding window procedure for payload "`GET /ET`" using 2-grams.

2 Implementation and Deployment

In this Section we introduce the foundations of the FIRST Intrusion Prevention System (FIPS): Based on a solid anomaly detection framework described in Section 2.1 we give details about the integration into the network infrastructure in Section 2.2. By introducing a so-called shadow system which mirrors the monitored system but is specifically prepared to detect malicious behavior, we can alleviate the effects of anomaly based detection mechanisms, namely the false positive rate. After discussing network connection bookkeeping and packet redirection we conclude this section with details about our FIPS deployment in a real life environment in Section 2.3.

2.1 Anomaly Detection

The anomaly detection method used is based on a model developed in [7]. The basic premise is that attacks can be detected as deviations from a model of normal behavior. To build a normality model, payloads are mapped to vectors by simply treating them as sequences of bytes. A sliding window of length n is moved over the byte sequence, obtaining a so called n-gram at each position. An n-gram is simply a subsequence of length n, where the set of all n-grams is

$$S = \{0, \dots, 255\}^n.$$

A payload x can be embedded into vector space by simply determining for every n-gram $s \in S$ whether it is contained in the sequence x:

$$\phi(x) = (contains(x, s))_{s \in S}\,,$$

where $contains(x, s)$ is a function returning 1 iff s is contained in x, 0 otherwise. With this embedding it is possible to normalize the vector to cancel out possible length dependencies. For example, in a short payload most n-grams are very frequent and the resulting vector becomes long, while in a long payload frequencies might be lower and the resulting vector short. A way to circumvent the effect is by normalizing the vector to a length of one using a 2-norm:

$$\phi_{norm}(x) = \phi(x)/||\phi(x)||.$$

Index in vector	0x202f	0x2f45	0x4554	0x4745	0x5420
Value	$1/\sqrt{5}$	$1/\sqrt{5}$	$1/\sqrt{5}$	$1/\sqrt{5}$	$1/\sqrt{5}$

Table 1: Embedding for payload "`GET /ET`" using 2-grams and $\phi_{norm}()$.

As an example, we embed the payload "`GET /ET`" using 2-grams and $\phi_{norm}()$ as the embedding function. From the sliding window procedure in Figure 1 we obtain the normalized vector in Table 1. The 2-grams from the sliding window procedure are reused as indices for the corresponding component in the embedded vector. All other components are set to 0.

Distance function	$d(x,z)$
Euclidean	$\sqrt{\sum_{s \in S} \lvert \phi_s(x) - \phi_s(z) \rvert^2}$
Manhattan	$\sum_{s \in S} \lvert \phi_s(x) - \phi_s(z) \rvert$
χ-Squared	$\sum_{s \in S} \frac{\lvert \phi_s(x) - \phi_s(z) \rvert^2}{\phi_s(x) + \phi_s(z)}$
Canberra	$\sum_{s \in S} \frac{\lvert \phi_s(x) - \phi_s(z) \rvert}{\phi_s(x) + \phi_s(z)}$

Table 2: Distance functions for payloads.ϕ_s returns the value of the embedding vector for gram s.

In a vector space common vectorial distance functions can be applied. Packets with similar payloads will share most n-grams and will be close to each other in vector space. Packets with dissimilar payloads will share few n-grams and be far from each other in vector space, thus yielding high distance values. It is not viable however to compute vectors and distances explicitly, because for $n = 4$ the vector space already has over 4.2 billion dimensions. Taking advantage of the fact that the number of n-grams contained in a single payload is linear in length, efficient methods for extraction and comparison of vectors have been developed in [9]. These methods are used with the distance measures in Table 2.1 throughout the paper.

We use the model of normality developed in [5] which in turn is based on earlier work on the usage of anomaly detection for intrusion prevention [8, 12, 13]. This model, called the centroid model, assumes normal data to be close to a center of normal payload embeddings. The model is computed using a simple algorithm:

1. *Training:* collect samples of normal data packets $X = \{x_1, \ldots, x_n\}$ and compute their mean $\mu = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)$. Now μ can be thought of as the center of a sphere in a $|S|$-dimensional vector space.

2. *Validation:* determine the anomaly threshold t_{an} from an independently collected sample $Y = \{y_1, \ldots, y_m\}$ of normal data packets. Set t_{an} so that the ratio of packets y_i for which the distances $d(\mu, y_i)$ with $d(\mu, y_i) > t_{an}$ is less or equal to a predefined anomaly rate. t_{an} corresponds to the radius of a sphere in an $|S|$-dimensional vector space.

3. *Detection:* We define the detection function

$$\mathrm{anomaly}(y) = \begin{cases} true & \text{if } d(\mu, y) > t_{an} \\ false & \text{otherwise} \end{cases}$$

for a payload y. This function returns *true* if a packets vector space embedding lies in the sphere and is considered normal or *false* otherwise.

2.2 Network Infrastructure

One of the main ideas used in our network IPS is to compensate false alarms in anomaly detection with an automatic response which results in almost no limitations for legitimate users. This approach was first developed in [5]. It works by combining a network based information source, an analysis engine and a response device on a system acting as a firewall-like router. Traversing this router is the only way a client can send requests to a server which is to be protected from attacks. When the IPS detects an anomaly in the first payload of a connection, it suspects an attack and redirects the connection to a hardened shadow server (Figure 3). The shadow server provides the same services as the production server but runs additional software for intrusion prevention that slows it down. If the first payload is

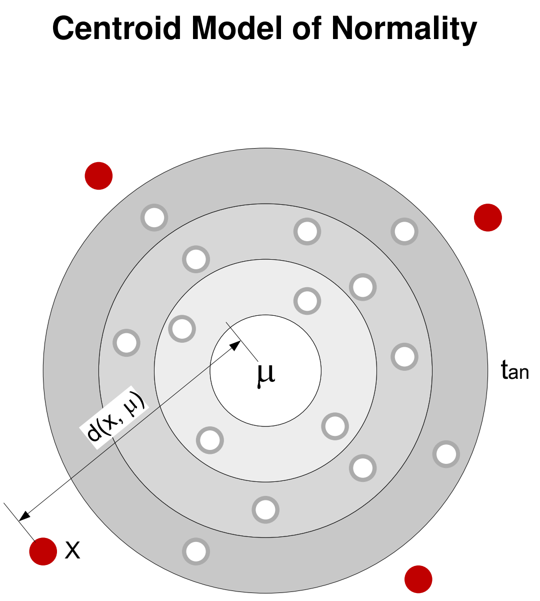

Figure 2: The centroid model of normality can be seen as a sphere in the vector space that packet payloads are embedded in. Normal packets lie in the sphere, anomalous packets outside of it.

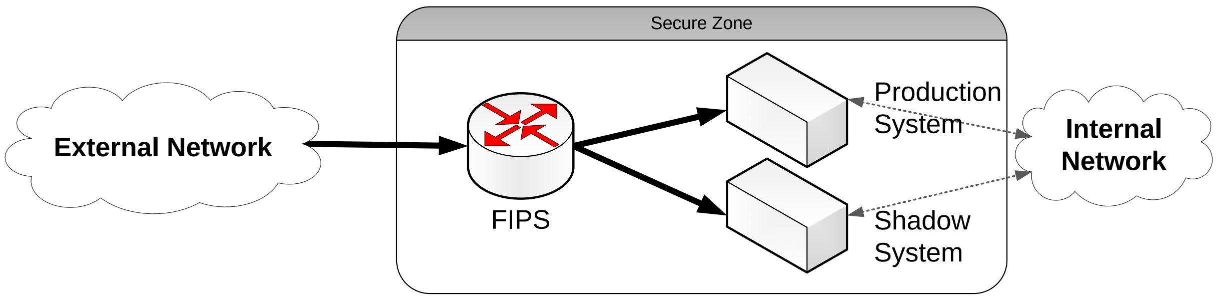

Figure 3: Network setup with the IPS acting as a firewall-like router. Normal connections are routed to the production system, most anomalous connections to the shadow system.

not anomalous, but a subsequent one is, the TCP flow is logged to a *forensic sink* from that point on. Packets are still acknowledged at the transport layer so that as much malicious traffic as possible can be collected, but packets are not forwarded anymore. The forensic sink can later be used for fine-tuning anomaly detection, for example by retraining the centroid model from Section 2.1 with additional data.

This approach has several advantages. Because the IPS is the only router through which servers are reachable, it can contain attacks directed at many different servers. The current implementation supports many combinations of production and shadow server, limited only by restrictions in the operating system kernel[1]. The configuration of the IPS is trivial and does not interfere with the workings of a server (however session synchronization is an issue, if we monitor stateful applications). The deployment of the IPS results in no additional load for the protected servers. Also, as the IPS acts solely as a router, an attacker will have difficulty detecting its existence. Because even connections regarded as anomalous in

[1]These stem from the necessity to queue packets into user space, where each queue has to be uniquely identifiable. For this purpose the Linux kernel uses an unsigned 16 bit integer, resulting in a maximum of 65 535 combinations of production and shadow server.

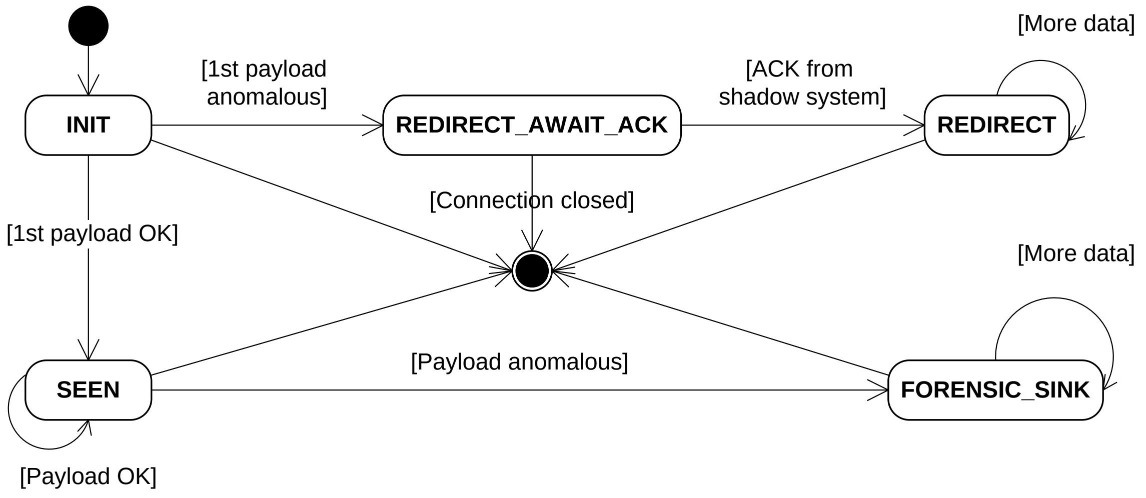

Figure 4: Routing state diagram as developed in [5].

many cases are served by a shadow server, the drawback of anomaly detection based analysis, namely producing false positives, is highly alleviated. Even if a legitimate connection is seen as anomalous, the client will not notice this other than its connection being slower than normal.

Each connection is associated with a *routing state* which specifies the actions that are taken for packets belonging to the connection. The routing state machine is given in Figure 4. A connection starts in the `INIT` state. This state is altered when seeing the first packet with payload. The $\text{anomaly}()$ function from Section 2.1 is used to decide how a packet is handled. If it is anomalous, the packet is redirected to the shadow server. If the packet is normal, the connection status is changed to `SEEN` and the IPS delivers the packet to the production system.

If the IPS later encounters an anomalous packet in a connection in `SEEN` state, the state is changed to `FORENSIC_SINK` and all packets are logged to the forensic sink without being forwarded. Packets are acknowledged however in order to collect more data of the suspected attack. While in principle it would be possible to redirect the connection to the shadow server at this point, it would increase the memory usage of the IPS immensely. Because a client request could be distributed over several TCP packets, every packet would have to be stored before forwarding it to the production server. Only this way packets could be replayed when a redirection to the shadow server becomes necessary because of an anomalous packet. Also some protocols make redirection from a certain point on very difficult. Because the case of an anomalous packet in the middle of a connection is rare the effort would be unjustified.

If the first packet payload is anomalous, redirection can be done much cheaper. The packet is dropped and the TCP connection to the production server is teared down by the IPS by injecting a `RST` packet. The IPS then injects a `SYN` packet to the shadow server and sets the connection state to `REDIRECT_AWAIT_ACK` . This state is kept until the shadow server sends a `SYN/ACK` packet. When the client retries to send the first packet with payload because it never received an `ACK` , the packet is redirected to the shadow server. A sequence diagram of the process is given in Figure 5. To make the redirection transparent, acknowledgment numbers have to be translated from the old connection that the client still thinks is active to the new connection and vice versa.

Let y be the old, z be the new acknowledgment number. Then the IPS stores $d = z - y$ and translates client packets to the new connection by adding d to their acknowledgment number, while subtracting d from the acknowledgment number of server packets to translate them to the old connection.

Because packet redirection has to be transparent to both client and shadow server, each of the following packets has to be rewritten. The production servers IP address and port are rewritten to those of the shadow server. In addition the sequence and acknowledgment numbers are rewritten to match those

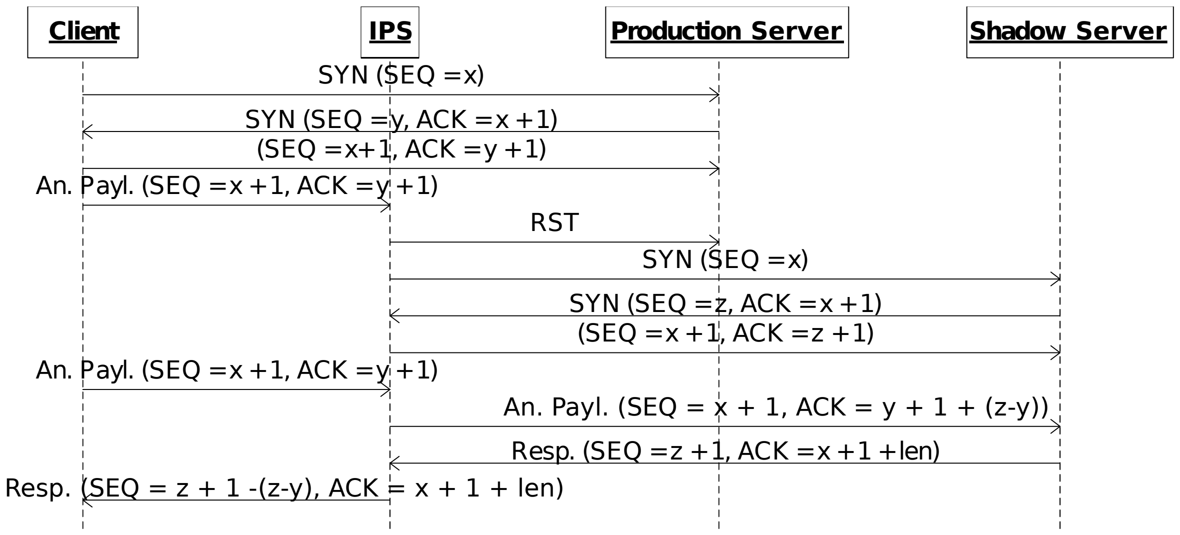
Client
IPS
Production Server
Shadow Server
SYN (SEQ =x)
SYN (SEQ =y, ACK =x +1)
(SEQ =x+1, ACK =y +1)
An. Payl. (SEQ =x +1, ACK =y +1)
RST
SYN (SEQ =x)
SYN (SEQ =z, ACK =x +1)
(SEQ =x +1, ACK =z +1)
An. Payl. (SEQ =x +1, ACK =y +1)
An. Payl. (SEQ = x + 1, ACK = y + 1 + (z-y))
Resp. (SEQ =z +1, ACK =x +1 +len)
Resp. (SEQ = z + 1 -(z-y), ACK = x + 1 + len)

Figure 5: Sequence diagram for the redirection process [5]. If the first packet with a payload is classified as anomalous (in the diagram: *An. Payl.*), the connection to the production server is reset by the IPS and a new connection to the shadow server is established.

of the new connection (for a packet coming from the client) or those of the old connection (for a packet coming from the shadow server), respectively. When the connection is closed, the routing state machine reaches its final state. All information belonging to the closed connection is freed.

2.3 Real Life Deployment

For the integration in our real live environment we have to adjust the setup as introduced in Figure 3: Since the DMZ of our institute is not equipped with an internal router or firewall, we incorporate both the routing and destination NAT inside our FIPS system as depicted in Figure 6. The rationale here is to do Network Address Translation (NAT) from a given externally visible IP address to a newly created, internal network, in which both the production and the shadow server reside.

The FIPS system is the default router for the internal network and also in charge for the NATing of the addresses. We implement a "software" NATing mechanism inside FIPS, since the standard NATing of netfilter interferes with our redirection mechanism. Because NATing with netfilter tracks connections inside the Linux kernel, once we start our redirection mechanism the kernel is sending RST-packets to the shadow system because the connection is not known to the kernel. While it is theoretical possible to add these connections to the Linux connection tracking table via the libconntrack, we opted for implementing NAT in FIPS itself for the sake of simplicity.

We use a Squid Proxy as production system and a fully equipped ModSecurity system installed in reverse proxy mode as shadow system. Both systems mirror the web content of servers located inside the internal network. The request from production and shadow system to the internal servers are also routed by FIPS (as depicted by the dashed lines), which adds an additional load to FIPS but allows for a seamless and easy integration into the existing infrastructure. Results of the evaluation in this real environment are discussed in Section 3.4.

3 Evaluation

In the following section we evaluate FIPS first on pre-collected datasets and examine how the data logged to the forensic sink can be used to efficiently update the anomaly detector. Furthermore we give details

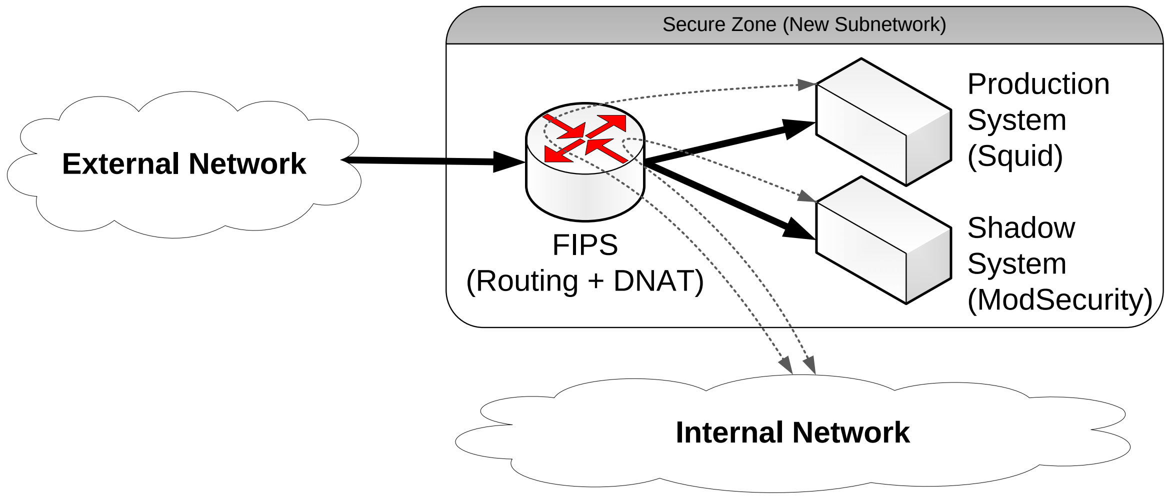
Secure Zone (New Subnetwork)
External Network
Production System (Squid)
Shadow System (ModSecurity)
FIPS (Routing + DNAT)
Internal Network

Figure 6: Network setup used in the real life evaluation. Certain adjustments compared to the general setup had to be applied to seamlessly integrate FIPS into the existing infrastructure. See text for details.

Dataset	Selected Model	anomaly rate	real FP	TP
FIRST 2007	4-grams bin. Canberra	0.1	0.0037	0.9583
Blog 2009	3-grams bin. χ-Squared	0.1	0.0151	0.7361

Table 3: Accuracy rates for the complete datasets.

about the runtime performance of FIPS and compare it to the prominent open-source IPS SNORT. This sections concludes with the evaluation of the real life deployment of FIPS.

3.1 Offline Evaluation

For our first evaluation of FIPS we collected two datasets: The *FIRST 2007* dataset contains roughly 500k packets gathered during 21 consecutive days at our institute's web server. The server contains both static and dynamic content hosted by the web content management system OpenWorx. The *Blog 2009* dataset contains roughly 1500k packets gathered during 35 consecutive days at a domain running several WordPress blog instances. In addition we assembled a pool of recent attacks as detailed in the appendix in Table 5. Each of these datasets are splitted in 3 equally sized parts, which form the training, validation and testing pool. The training and validation pool are used to perform model selection: For each of the distance of Table 2.1 and all n-grams with $n \in \{2, 3, 4\}$ and the predefined anomaly rates of $\{0.01, 0.05, 0.1\}$ we calculate the respective centroid and thresholds as described in Section 2.1 and choose the best model in terms of area under the Receiver Operating Characteristic (ROC) curve. This model is then evaluated on the testing pool: Each legitimate packet redirected to the forensic sink is counted as a false positive while each attack entirely delivered to the production system is counted as a false negative.

The results are denoted in Table 3. While the FIRST 2007 dataset is accurately handled by FIPS, both in terms of false positives and true positives, the methods seems to break down on the Blog 2009 dataset: the relatively high number of false positives of roughly 1% is accompanied by a surprisingly low number of true positives compared to the FIRST 2007 dataset.

As Table 3.1 reveals, these findings are due to the mixture of several subdomains inside the blog dataset: If we split the dataset according to their respective subdomains, we get a much more diverse picture: While most of the domains are now accurately captured by FIPS, some are entirely broken. If

Domain	Selected Model	anomaly rate	real FP	TP	# packets
doXXX	4-grams bin. norm. Euclidean	0.1	0.00652	1.00000	3836
fiXXX	2-grams bin. χ-Squared	0.1	0.00305	0.58333	5894
frXXX	3-grams bin. norm. Manhattan	0.1	0.01749	0.93056	96973
huXXX	4-grams bin. Canberra	0.1	0.02340	0.90278	8204
jaXXX	4-grams bin. Canberra	0.1	0.15953	0.75000	17639
niXXX	4-grams bin. Canberra	0.1	0.04057	0.81944	10402
riXXX	2-grams bin. Euclidean	0.1	0.05299	0.87500	6700
saXXX	4-grams bin. Euclidean	0.1	0.04883	0.76389	8090
svXXX	4-grams bin. Canberra	0.1	0.05294	0.83333	21554
ttXXX	4-grams bin. Canberra	0.1	0.00133	0.98611	2251
tiXXX	2-grams bin. χ-Squared	0.1	0.01181	0.90278	269402

Table 4: Accuracy rates for the individual subdomains of the Blog 2009 dataset.

we look at the number of packets we have collected for each subdomain, we see that several datasets are below 100k packets. The two datasets containing enough packets for stable estimates (frXXX and tiXX) show almost the same performance, which is a good indicator for the real performance of FIPS on this dataset. In an additional analysis, we focus on the tiXXX subdomain, since it has the highest number of packets. Note that for all datasets the models with a predefined anomaly rate of 0.1 have been selected, but almost all of them exhibit a much smaller real false positive due to the redirection mechanism employed in FIPS.

3.2 Retraining with Sink-data

While FIPS is capable of providing a sufficient protection against a wide range of attacks, it also generates a some false positives. Each of these result in a corrupted or even totally broken connection for a legitimate user request. Therefore we need a simple, yet efficient method for improving the anomaly detector during the life time of a FIPS installation. Since FIPS collects all the packets resulting in false positives inside the forensic sink, it is natural to exploit this data pool for the task at hand. After cleaning the forensic sink data from real attacks, we propose a weighted mean scheme formalized in the following equation:

$$\begin{aligned}
\mu_\alpha &= \frac{(1-\alpha)N\mu_{\text{train}} + \alpha M \mu_{\text{sink}}}{(1-\alpha)N + \alpha M}, \text{with} \\
\mu_{\text{train}} &= \frac{1}{N}\sum_{i=1}^{N} \phi(x_{\text{train},i}) \text{ and} \\
\mu_{sink} &= \frac{1}{M}\sum_{i=1}^{M} \phi(x_{\text{sink},i}),
\end{aligned}$$

where μ_{sink} and μ_{train} are the means when using only the train or sink data respectively. The parameter α controls how much weight the new, formerly dropped packets are given in updating of the centroid: $\alpha = 0$ just returns the normal centroid based on the train pool, while $\alpha = 1.0$ results in a sink-only centroid. The case $\alpha = 0.5$ corresponds to the overall centroid on the merged train and sink pool. Every setting in between these gradually gives more weight to the train ($\alpha < 0.5$) or the sink data ($\alpha > 0.5$).

For the evaluation of this retraining procedure we update the original centroid with the sink data given a specific value of α. We then test the performance of this updated centroid on the remaining test data. To get reliable estimates and error bars we repeat this procedure 10 times with different slices of the data. The resulting absolute *decrease* of the false positives rates for different values of α is depicted in Figure

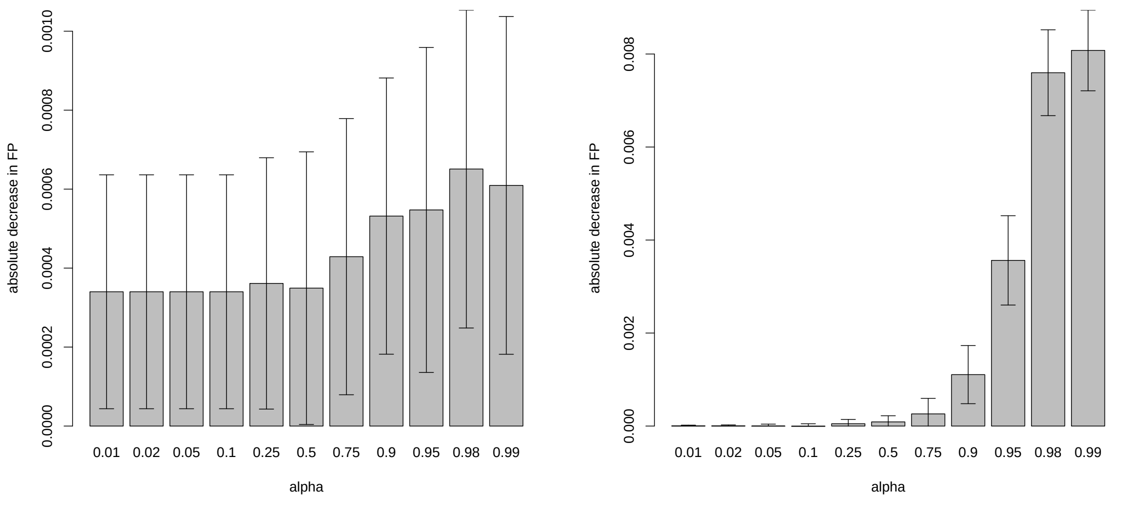

Figure 7: Decrease of false positive rates of the normal centroid versus the sink-adjusted centroid for the FIRST 2007 and tiXXX subdomain of the Blog 2009 dataset with varying α. Error bars give the 95% confidence interval based on the standard error.

3.2. For both datasets we see a continuous improvement with increasing α of up to 0.8% for the tiXXX subdomain of the Blog 2009 dataset. But at least for the tiXXX subdomain this comes at a price: As Figure 3.2 reveals, the true positive rates also decrease which results in more succesfull attacks. Fixing α to a value of 0.95 gives a good compromise between decrease of false positive rate and decrease of true positives for both datasets. However an attacker could try to exploit this update procedure to manipulate the centroid. This matter is further discussed and analyzed in [3], where bounds are given for how much of the traffic the attacker has to control, to really exploit the update procedure.

3.3 Runtime Evaluation

For the runtime evaluation we use Tcpreplay. With Tcpreplay collected real traffic can be replayed over a given network device. This enables very realistic performance measurements. In order to use Tcpreplay, incoming and outgoing HTTP traffic was collected at a web server at Fraunhofer FIRST during 25h in 2007. For SNORT and binary embedding with differing n-gram size for anomaly detection the traffic was replayed and points of the characteristic curves recorded, which is shown in Figure 3.3. The plots show forwarded traffic speed as a function of incoming traffic speed. With decreasing n, the point at which packet loss occurs is delayed. After the system loses packets the forwarding speed stays more or less constant.

For FIPS we define the maximum forwarding rate using $f : \mathbb{N} \to \mathbb{N}, i \mapsto f(i)$ as the forwarding function which maps incoming speed to forwarding speed. For measurement points $\{i_1, \ldots, i_n\}$, $f(i_l)$ is the maximum forwarding rate iff $f(i_k) = i_k$ for all $k \leq l$ and $f(i_{l+1}) < i_{l+1}$. In other words, the maximum forwarding rate is defined as the last measured forwarding rate before packet loss occurs. SNORT shows packet loss at every nominal replay speed. Packet loss was not constant even for low replay speeds, which makes it unlikely that the reason was truncated connections or a setup error on our part. Because of the packet loss the maximum forwarding speed for SNORT was defined as the forwarding speed at the first local maximum point (last measured forwarding speed before the speed decreases for the first time with increasing input speed). This definition sometimes could give higher maximum forwarding speed then the more rigid definition used for FIPS. Thus it could overestimate

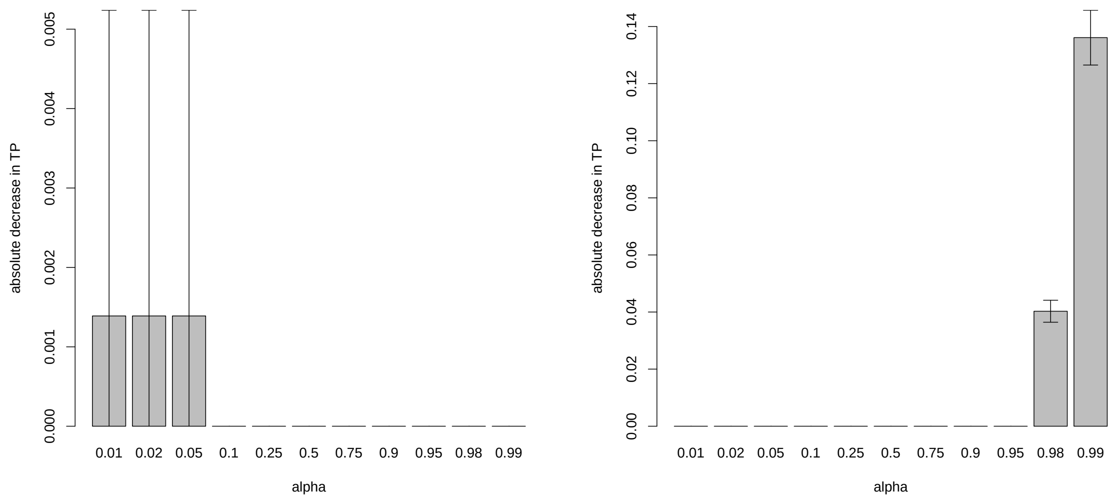

Figure 8: Decrease of true positive rates of the normal centroid versus the sink-adjusted centroid for the FIRST 2007 and tiXXX subdomain of the Blog 2009 dataset with varying α. Error bars give the 95% confidence interval based on the standard error.

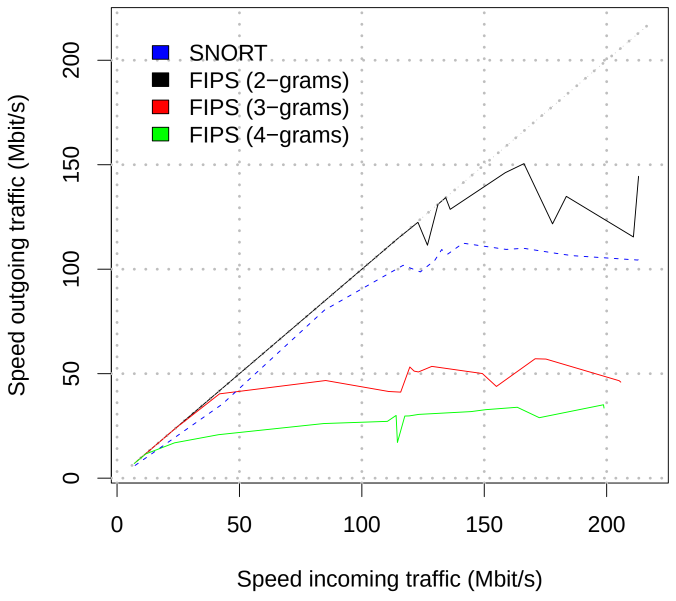

Figure 9: Characteristic curves Snort vs. FIPS

	# (All)	% (All)	# (Sink)	% (Sink)
Packets (Cookie)	807815	30.71	9756	81.52
Packets (non-Cookie)	1822731	69.29	2211	18.48

Table 5: Absolute and relative numbers of packets inside the overall and the sink pool, which originate from a HTTP request containing a cookie. Cookie-related packets are overrepresented inside the sink.

SNORTs performance. Still, for instance the optimal parameter combination of the FIRST 2007 dataset, binary embedding with 2-grams, the anomaly detection based IPS performs at a maximum forwarding speed of 117 Mbit/s, while SNORTs maximum forwarding speed is only 102 Mbit/s.

3.4 Online Evaluation

After setting up FIPS as described in 2.3, we evaluated the system for a period of 1.5 month. The system was up and running during the complete period without any software failures. We captured roughly 2,500k packets in total, of which 68.10% were delivered to the production system, 31.45% were delivered to the shadow system and 0.45% were logged to the forensic sink.

The shadow system captured roughly 2,000 packets of malicious communication mainly originating from scanners. None of these traffic patterns could be observed on the production system, showing that FIPS efficiently encapsulated the attempted attacks.

While the number of packets delivered to the forensic sink is quite low and we already introduced a method in Section 3.2 to lower this value even more, closer inspection of this data might give further insight. Table 3.4 shows the absolute and relative numbers of packets inside the overall and the sink pool, which originate from a request containing a specific cookie from the content management system OpenWorx. We clearly see that the cookie-related packets are overrepresented inside the forensic sink compared to the overall ratio. This might hint at a slight deficiency of the anomaly detector if the request at hand contains a cookie which often involves large parts of random structure.

To conclude the analysis we have a further look at the length distribution of packets inside the forensic sink, which are not cookie-related. The comparison to the overall length distribution of non-cookie-related packets are shown in Figure 3.4. The distribution of the sink data clearly shows a peak in the lower and upper range of the length, which cannot be observed in the overall distribution. This might indicate a weakness of the anomaly detector for overly small or big packet payloads. We discuss possible remedies in the conclusions.

4 Related Work

The most prominent open-source intrusion prevention system is SNORT which has a built-in inline mode. By using the SNORT engine on packets queued to the user space with the `libipq` library, SNORT inline decides per packet whether it should be dropped or delivered to the network. Another well known open-source intrusion prevention system specialized on monitoring HTTP traffic is the ModSecurity platform. Operated as a reverse proxy ModSecurity is capable of blocking traffic on a signature based negative security model and also on a positive security model, which basically enumerates allowed traffic. Both systems however lack an anomaly-driven decision mechanism apart from a simple dropping procedure.

A crucial prerequisite for inline intrusion prevention is the ability to perform the intrusion detection part extremely fast. Therefore the SafeCard IPS [2] was implemented directly on a network card. The SafeCard architecture performs deep packet inspection doing full TCP re-assembly with a cascade of test modules of increasing complexity. Having signature based payload inspectors the architecture however lacks an anomaly detection component for the detection of novel, unknown attack vectors. Other approaches exploiting modern GPU architectures for fast packet processing are the Gnort system [11],

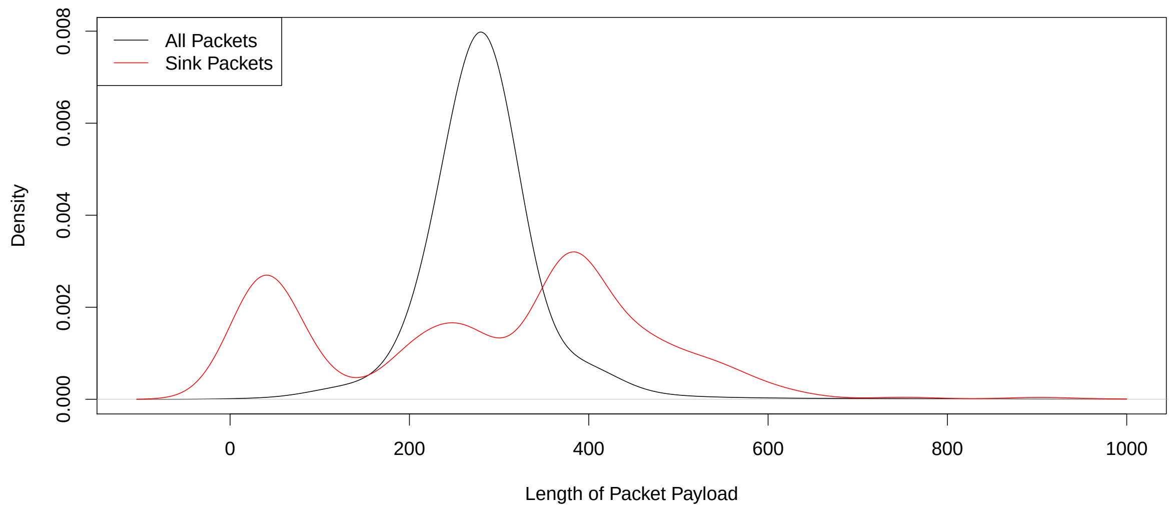

Figure 10: Length Distribution of non-cookie-related packets of all packets compared to the packets logged to the forensic sink.

which shifts performance critical parts of the signature matching algorithm to the GPU by performing multiple matching processes in parallel.

Valeur et al. [10] applied anomaly detection techniques combined with intrusion prevention mechanisms in the specific context of web applications and clients: A reverse HTTP proxy architecture is used to forward requests to servers running under different privileges by exploiting anomaly scores obtained from a WebAnomaly sensor [4]. In line with this work the "shadow honeypot" architecture proposed by [1] redirects suspicious traffic to a specially adjusted application. This instrumentation is achieved using a source-to-source translator and is therefore limited to applications, where the actual source code is available.

The concept of shadow systems have been introduced in [12]. In [6] this is applied concept to the virtualization domain and presents a system which could handle both the production and the shadow system in one virtual environment. Primarily developed as a secure system to actively monitor systems, it could be used in our setup to lower the cost induced by the redirection process. If both the production and the shadow system are managed by the same hypervisor, special shortcuts for the redirection process could be implemented.

5 Conclusion and Further Work

In this paper we have introduced the anomaly-based network intrusion system FIPS. Our offline and online evaluation shows that our system is ready for real life deployment. An efficient update procedure for an already deployed system demonstrates the ability of FIPS to further adapt to errors and improve the overall performance of the system.

Results of our real life installation are promising but also show room for improvement: While the overall false positive rate is in an acceptable range, packets logged to the forensic sink show some distinctive features which could be exploited in an enhanced version of FIPS. For instance one could imagine a specifically crafted model for requests with cookies to eliminate the relative high number of cookie-related packets inside the forensic sink. Furthermore special heuristics could be employed when dealing with very small packets, to eliminate some more packets erroneously logged to the forensic sink.

While there are still open problems in an all-day use this paper has shown that anomaly based detection combined with clever redirection mechanisms and retraining methods can yield fast, efficient and up-to-date protection for inline network protection.

References

[1] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. Detecting targeted attacks using shadow honeypots. In *Proc. of USENIX Security Symposium*, pages 129–144, 2005.

[2] W. de Bruijn, A. Slowinska, K. van Reeuwijk, T. Hruby, L. Xu, and H. Bos. Safecard: a gigabit IPS on the network card. In *Recent Adances in Intrusion Detection (RAID)*, pages 311–330, 2006.

[3] M. Kloft and P. Laskov. A poisoning attack against online anomaly detection. In *NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security*, 2007.

[4] C. Kruegel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *Proc. of ACM Symposium on Applied Computing*, pages 201–208, 2002.

[5] T. Krueger, C. Gehl, K. Rieck, and P. Laskov. An architecture for inline anomaly detection. In *Proc. of European Conference on Computer Network Defense (EC2ND)*, pages 11–18, 2008.

[6] B. D. Payne, M. Carbone, M. Sharif, and W. Lee. Lares: An architecture for secure active monitoring using virtualization. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 233–247, 2008.

[7] K. Rieck and P. Laskov. Detecting unknown network attacks using language models. In *Detection of Intrusions and Malware, and Vulnerability Assessment, Proc. of 3rd DIMVA Conference*, LNCS, pages 74–90, July 2006.

[8] K. Rieck and P. Laskov. Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology*, 2(4):243–256, 2007.

[9] K. Rieck and P. Laskov. Linear-time computation of similarity measures for sequential data. *Journal of Machine Learning Research*, 9(Jan):23–48, 2008.

[10] F. Valeur, G. Vigna, C. Kruegel, and E. Kirda. An anomaly-driven reverse proxy for web applications. In *Proc. of the 2006 ACM symposium on Applied computing*, pages 361–368, 2006.

[11] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. Markatos, and S. Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In *Recent Adances in Intrusion Detection (RAID)*, 2008.

[12] K. Wang, J. Parekh, and S. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *RAID*, pages 226–248, 2006.

[13] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection. In *RAID*, pages 203–222, 2004.

Name	Attack type	CVE	Published
Buffer overflow attacks			
iis_htr	Buffer overflow	1999-0874	June 1999
iis_printer	Buffer overflow	2001-0241	May 2001
idq_isapi	Buffer overflow	2001-0500	June 2001
apache_chunked	Buffer overflow	2002-0392	June 2002
altn_webadmin	Buffer overflow	2003-0471	June 2003
ia_webmail	Buffer overflow	2003-1192	November 2003
icecast_header	Buffer overflow	2004-1561	September 2004
iis_w3who	Buffer overflow	2004-1134	December 2004
iis_rsa_webagent	Buffer overflow	2005-4734	October 2005
peercast_url	Buffer overflow	2006-1148	March 2006
novell_messenger	Buffer overflow	2006-0992	April 2006
shttpd_post	Buffer overflow	2006-5216	October 2006
novell_edirectory	Buffer overflow	2006-5478	October 2006
Code injection attacks			
awstats	Shell code injection	2005-0116	January 2005
php_vbullentin	PHP code injection	2005-0511	February 2005
php_xmlrpc	PHP code injection	2005-1921	June 2005
barracuda	Perl code injection	2005-2847	September 2005
php_pajax	PHP code injection	2006-1551	April 2006
apache_modjk	Binary code injection	2007-0774	March 2007
php_inject	PHP code injection	—	*
sql_inject	SQL code injection	—	*
WordPress attacks			
crlf_injection	response_splitting	2004-1584	December 2004
cat_id_variable	SQL injection	2005-1810	June 2005
cache_lastpostdate_cookie	PHP code injection	2005-2612	August 2005
redirect_to	Cross-site scripting	2007-1599	March 2007
DMSGuestbook	Directory traversal	2008-0615	February 2008
Spreadsheet	SQL injection	2008-1982	April 2008
media_holder	SQL injection	6842$^{\text{milwOrm}}$	October 2008
Page_Flip_Image_Gallery	Directory traversal	2008-5752	December 2008
fMoblog	SQL injection	2009-0968	March 2009
wordpress_mu	Cross-site scripting	2009-1030	March 2009
Miscellaneous attacks			
httptunnel	HTTP tunnel	—	March 1999
shoutcast	Format string	2004-1373	December 2004
php_unserialize	Integer overflow	—	March 2007
xss	Cross-site scripting	—	*

Table 6: Table of 35 exploits. Each attack is executed in different variants. A description of the attacks is available at the Common Vulnerabilities and Exposures (CVE) web site. Attacks marked by * are artificial and have been specifically created for the data set.