Efficient Distributed Signature Analysis

Michael Vogel, Sebastian Schmerl, Hartmut König
Brandenburg University of Technology, Cottbus
Computer Science Department
{mv, sbs, koenig}@informatik.tu-cottbus.de

Abstract. Intrusion Detection Systems (IDS) have proven as valuable measure to cope reactively with attacks in the Internet. The growing complexity of IT-systems, however, increases rapidly the audit data volumes and the size of the signature bases. This forces IDS to drop audit data in high load situations thus offering attackers chances to act undetected. To tackle this issue we propose an efficient and adaptive analysis approach for multi-step signatures that is based on a dynamic distribution of analyses. We propose different optimization strategies for an efficient analysis distribution. The strengths and weaknesses of each strategy are evaluated based on a prototype implementation.

1 Introduction

Intrusion detection systems (IDS) have been proven as an important instrument for the protection of computer systems and networks. Nowadays, most IDS use a centralized approach and apply misuse detection (signature analysis). They mainly use single-step signatures to identify harmful behavior in a stream of audit data or network packets, respectively. This will change in next years, since multi-step signatures, in contrast to single step signatures, allow it to model attacks much more precisely, in particular the specific characteristics of the attack traces, existing dependencies, and the chronological order of the attack steps. This will reduce significantly the false alarm rate. Multi-step signatures can store state information to track the attack through its different stages. This supports in particular the attack detection in application layer protocols as well as in Web 2.0 applications which are of increasing importance. Many semantic aspects of today's attacks cannot be modeled by single-step signatures at all or only insufficiently. Therefore, we focus our work on multi-step signatures.

A challenge that all intrusion detection systems are facing is the increasing performance of networks and end systems. This leads to a rapid growth of audit data volumes to be analyzed. On the other hand, the growing complexity of IT systems creates novel vulnerabilities and offers new possibilities for running attacks so that the number of signatures to be analyzed increases as well. Already now intrusion detection systems are forced to reject audit data in high load situations or to delay the analysis of security violations significantly. Thus, counter-measures become impossible or lose their impact. As consequence, systems become unprotected, when they are intensively used.

To cope with this situation several approaches have been proposed, e.g. the detection of intrusions based on an analysis of more compact, less detailed audit data [2, 3] and network flows [1] as well as various optimizing analysis methods for signature and network based single-step intrusion detection systems. For example, [6] describes an approach for the IDS SNORT that transforms signatures into a decision tree to reduce the number of redundant comparisons during analysis. Optimized string matching algorithms are proposed in [5]. These approaches aim at optimizing the non-distributed, single threaded signature analyses. A distributed approach like GNORT [8] utilizes the massive parallel computing capabilities of graphic processors (GPUs), but it only doubles the analysis throughput compared to the sequential Snort. So far, network-based IDS only apply primitive means to parallelize analyses by load balancing [10, 11]. There are also almost no approaches to parallelize host-based IDS analyses [9].

On the other hand, free computing resources are available in any network. CPU technology will provide only slightly more computation power per core in the future. In this paper we present a distributed signature analysis approach to use free resources in networks to overcome this issue. In Section 2 we introduce a generic model for multi-step signatures to discuss distribution strategies. Next, in Section 3 we introduce several distribution strategies and outline their benefits and constraints for multi-step IDS. In Section 4 we evaluate the strategies based on measurements of a prototype implementation and discuss their applicability. Some final remarks conclude the paper.

2 Modeling Multi-Step Signatures

We first introduce a generic model for multi-step signatures which covers all existing multi-step-signature languages and related intrusion detection systems. The model confines to typical characteristics of multi-step signatures and their analysis. The core concept of a multi-step signature language is its ability to store information about attacks, system states, and related changes. Accordingly, a multi-step signature can be defined as a directed, labeled graph $MS = \{V, E, EvT, SI, f, state, sens, cond, mark, trans\}$ with:

- V – set of state nodes, representing the stages of an attack,
- $E \subseteq V \times V$ – set of edges representing valid state transitions,
- EvT – finite set of types of security relevant events (audit events), e.g. different types of system calls or network events,
- SI – set of state information representing the state and the stage of a certain attack,
- $state: V \rightarrow \wp(SI)$ – a function which labels state nodes with a set $si \subseteq SI$ of state information,
- $f \in V$ – a final node indicating a detected attack as soon as labeled by $state(f)$,
- $sens: E \rightarrow EvT$ – a function which labels each graph edge with an event type, whereby the occurrence of an event of the specified type can trigger a state transition represented by the edge,
- $cond: E \rightarrow B$ – a function labeling each edge $(a, b) \in E$ with a Boolean condition $B: SI \times Ev \rightarrow \{0, 1\}$ which specifies arbitrary expressions (arithmetic, string matching, …) between features of state information $si \in \text{state}(a)$ of node a and the occurring event $ev \in Ev$, whereby a state transition requires a fulfilled condition,
- $mark: V \times SI \rightarrow V$ – a labeling function which adds state information si to node v, whereby $mark(v, si) = v'$, with $state(v') = state(v) \cup si$,
- $trans: E \times Ev \rightarrow V$ – transition function that evaluates for each occurring event ev of type $evType \in EvT$ whether edge $e = (a, b) \in E$ is sensitive to this event type ($sens(e) = evType$) and whether its condition $cond(e)$ is fulfilled. In this case, the

transition (a, b) is executed by reading state information *si* of node *a* by *state*(a), modifying *si* (its features) to *si*', and moving *si*' to node *b* by applying *mark*(b, *si*').

The detection of a multi-step attack can be outlined as follows. For each occurring audit event *ev* and for each edge $e = (a, b) \in E$ of the signature, the function *trans*(e, *ev*) is executed. This function evaluates by *sens*(e) whether edge *e* is sensitive to the type of event *ev* which may triggers a state transition. In this case, the edge condition *cond*(e) is evaluated by correlating features of event *ev* with state information $si \in \text{state}(a)$ of node *a* which represents the stage of the attack and contains aggregated information of former state transitions. If the edge condition *cond*(e) is fulfilled the state transition (a, b) is executed in two steps: (1) State information $si \in state(a)$ is read and updated or modified with information from the current event *ev*. (2) Next, the successor node *b* is labeled with the modified state information *si*' by *mark*(b, *si*'). This evaluation process is executed for each edge of the signature and all signatures. It must be repeated for each occurring audit event. An attack is detected if the final node *f* of a signature is reached and labeled with state information.

Now we extend the model for the needs of a distributed analysis by defining functions for statistical data collection to derive optimal distribution decisions.

- *C* – a cluster representing a virtual analysis unit with limited computation capacities to assign signature fragments (state nodes) to,
- $cl\colon V \to C$ – a function which assigns each state node to a cluster, whereby initially each node is assigned to a unique cluster,
- $compC\colon E \to \mathrm{N}$ – a function labeling each edge $e \in E$ with the value of the computation effort (e.g. #cpu cycles), which was consumed in the previous time frame to evaluate the edge condition of *e*.
- commC$\colon E \to \mathrm{N}$ – a function labeling each edge $(a, b) \in E$ with the value of the communication effort (e.g. #bytes) which was consumed in the previous time frame to transmit state information from *a* to *b*.
- $evC\colon E \to \mathrm{N}$ – a function labeling each edge $(a, \mathrm{b}) \in E$ with the value of the communication effort which was consumed in the previous time frame to transmit audit events of type *sens*(a, b) from a sensor to node *a*.

3 Distributed Audit Data Analysis Strategies

In signature based intrusion detection systems the analysis of audit data requires considerable computation efforts. Signature bases that cover all currently known vulnerabilities of the protected systems may easily consist of thousands of signatures. In high load situations, when large amounts of audit data are recorded, the resource consumption of the analysis system grows rapidly and exceeds frequently the available resources. For very short time periods (seconds), buffering can be an appropriate measure, but as soon as the buffer capacities are exhausted the analysis system has to drop audit data and becomes useless and blind for attacks.

In order to reliably prevent such overload situations the analysis efforts of a signature based IDS should be kept continuously on a reasonable low level. This can be achieved by distributing and balancing the required analysis efforts among free resources in the protected domain. So it is more appropriate to utilize five analysis units with a load of 20 % each, instead of only two systems with 50 % load each. This allows it to keep sufficient, free resources needed for analysis efforts in high load situations. As known, analysis distribution causes additional communication overhead which burdens the network and may lead to transmission delays. To ensure that the transmission of audit data in high load situations does not delay the analysis, sufficient bandwidth has to be provided. These demands, however, are conflictive and cannot be achieved at the same time.

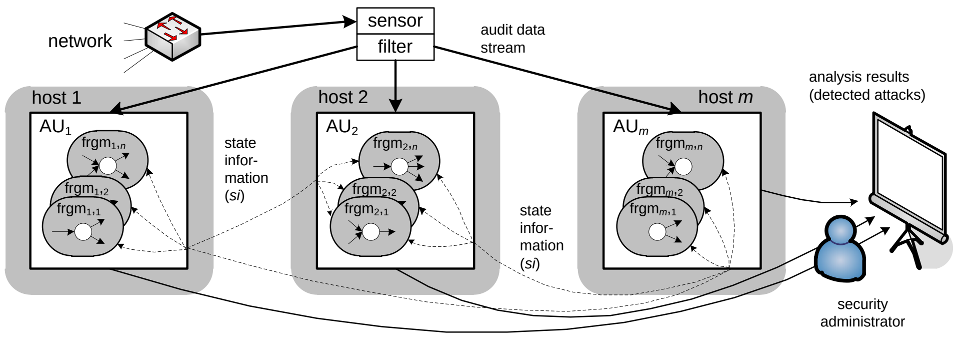

Fig. 1. Elements of a distributed signature analysis

Fig. 1 shows the elements of a distributed audit data analysis. A *sensor* logs audit events which are forwarded to *analysis units*, in our example AU_1 to AU_3. Each analysis unit AU_i evaluates a subset of the signatures. The audit events are classified into different event types *EvT*. To reduce the network utilization a configurable *filter* discards non-relevant audit events of types which are not needed by the respective analysis unit. For example, only events of type *EvT* in $\cup_{a,b \in V_1}$ *sens*(a, b) are forwarded to AU_1, where V_1 is the set of state nodes assigned to AU_1. The analysis units transmit state information $si \in SI$ between each other if needed. Analysis results are reported to a central component. Based on this concept we present below five distribution strategies.

A) Distributing Complete Signatures

The first approach simply distributes complete signatures to different analysis units AU_i. Hereby, a simple optimization can be performed balancing the number of assigned signatures to each analysis unit. Additionally, a finer-grained optimization can be achieved if statistics on resource consumption are gathered continuously for each signature. So CPU hardware counters can be used [7] to determine the number of clock cycles utilized to evaluate audit events and edge conditions for each signature. For this, the signatures' edges e are labeled with these values by function *compC*(e) to estimate the resource consumption of each signature. An example for a respective signature distribution is depicted in Fig. 2. Edges are labeled with required computation effort and examined event types as defined by *compC*(e) and *sens*(e), respectively.

This simple strategy causes, however, two major problems. (1) If an overload situation is mainly caused by a certain signature the performance problem will be solely moved to another analysis unit (another CPU or host) and the strategy to distribute whole signatures fails. (2) A typical signature *MS* usually correlates different types of audit events. Particularly, *MS* analyzes all audit events of types in $\cup_{a,b \in V}$ *sens*(a, b), where V is the set of state nodes of MS. If whole signatures are distributed nearly all audit events captured by a sensor have to be sent to the related analysis units. This multiplies the communication effort by n (number of used analysis units). This is unacceptable for most IT-infrastructures. Fig. 2 depicts these problems exemplarily.

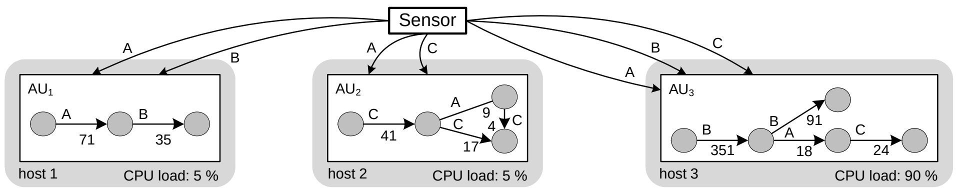

Fig. 2. Example of a simple, non-fragmented signature distribution

First, the three signatures MS_1 to MS_3 are assigned to AU_1 to AU_3. An optimal distribution would strive for a balanced load on all three systems of 33 % each. But, the analysis of signature MS_3 utilizes 90 % of the available resources of the respective analysis unit, while the analyses of MS_1 and MS_2 require only 5 % each. Secondly, additional communication is caused through event duplication. So the sensor has to triplicate type A events because they are analyzed by all three signatures. Analogously, type B and C events have to be duplicated for two analysis units. Therefore, further distribution options are required to balance the computation load finer-grained and to reduce the communication overhead.

B) Fragmenting Signatures

A more fine-grained signature distribution can be achieved by distributing signature fragments. This allows it to minimize audit event duplication. One or many state nodes (fragments) of a signature can be assigned to various analysis units, as depicted in Fig. **1**. Now the distribution strategy can pool nodes having outgoing state transition edges which are sensitive to the same audit event types (label $sens(a, b)$) onto the same analysis units. So, audit events only have to be duplicated for few analysis units. Fragmentation also supports a better balance of the analysis efforts among the units. Signatures that require the majority of available computation resources of an analysis unit, as discussed above, can be split up now. An optimization strategy though which only aims at pooling signature fragments that analyze the same audit event types on the same analysis unit is not desirable. If two state nodes $a, b \in V$ are connected by an edge (a, b) and a and b have been assigned to different analysis units AU_1 and AU_2 ($cl(a) \neq cl(b)$), then state information $si \in SI$ must be transferred from AU_1 to AU_2, whenever transition condition $cond(a, b)$ is fulfilled. The transfer of state information has to be minimized as well. To sum up, signature fragmentation allows a better balance of the computation load among analysis units and reduces audit event duplication. Fragmentation though may cause additional communication to transfer state information between analysis units.

C) Additional Reduction of the Communication Costs

Our next optimization does not primarily aim for an optimal balance of the computation efforts among the analysis units. Instead it takes the required communication effort to transmit audit events from the sensor to the analysis units into account as well as the transfer of state information between the distributed units.

To achieve this, the sensor has to gather a statistics that logs how often audit events of different types occur. The average data amount of an audit event is almost the same (*evSize* = 100 … 1000 bytes). The communication costs to transmit events of different types from the sensor to each analysis unit are labeled to the event types by function *evC*(*evType*) = *#events* * *evSize*. Additionally, each analysis unit continuously maintains a statistics that logs the number and size of transferred state information separately for each edge of the multi-step signature. The statistical values are labeled at each edge (*a*, *b*) of the signature by function *commC*(*a*, *b*). They allow determining the communication effort for audit event duplication and state information transfers for arbitrary signature distributions as an optimization criteria. For a given signature distribution, the event duplication effort can be determined according to equation (1).

$$\sum_{i=1}^{\#AU} \sum_{evT \in S_i} evC(evT), \text{where } S_i = \bigcup_{a \in V_i, b \in V} sens(a,b) \qquad \textbf{(1)} \qquad \sum_{(a,b)\in E, cl(a) \neq cl(b)} commC(a,b) \qquad \textbf{(2)}$$

Here, S_i is the set of event types examined by the subset V_i of signature fragments ($V_i \subset V$) that is assigned to AU_i. Similarly, the statistics permits to calculate the communication costs for state information transfers among different AUs by eq. (2). We apply the cluster component of our signature model (cf. Sect. 2). Each state node *a* is mapped onto a cluster by function *cl*(*a*), virtually representing an analysis unit. Only state information transfers between nodes *a* and *b* assigned to different AUs (clusters) cause real communication costs. Transfers between nodes on the same AU use shared memory.

Based on the statistical data, we can determine a communication efficient distribution of signature fragments by using a Greedy clustering algorithm. For this, we initially assign each state node of the signatures to an exclusive cluster. This initial state represents a virtual signature distribution for a maximum number of analysis units. In this case, all communication costs labeled to clusters and the signature edges are relevant because the audit events have to be duplicated for various analysis units and all state information has to be transferred between nodes assigned to different analysis units. Therefore, the initial stage represents the worst case communication scenario. The communication overhead, as defined in equations (1) and (2), can now be minimized by an iterative merging of clusters. We merge stepwise two clusters C_i and C_j that possess the maximum cumulated communication costs ($\Sigma_{(a,\, b)\in E}$ *commC*(*a*, *b*), with *cl*(*a*) = C_i, *cl*(*b*) = C_j) on edges leading from nodes of C_i to nodes of C_j or vice versa. This merging process is repeated until the desired number of analysis units (e.g. 3) is reached. Since the clustering algorithm only optimizes the communication costs, it is necessary to limit the number of assigned state nodes and thus the computation effort required to evaluate the transition conditions for each cluster. As a result, our clustering algorithm creates signature distributions with a minimal communication overhead and an acceptable balance of analysis effort.

D) Detection of Repeated Dependencies between Audit Events

A fine-grained signature distribution, as described above, induces new challenges. When fragmentation is applied and the nodes of a signature are assigned to different analysis units, state information may arrive delayed at the successor nodes due to network latencies. An attack, however, can be only detected in the audit event stream if all preceding attack steps, described by the state information, have been recognized before. This is not possible, when the state information arrives too late. There is a simple solution for this problem. Since audit events from the same sensor can be ordered chronologically, state information can be related to the audit events. This can be achieved by enumerating all emitted audit events consecutively with a unique ID in the sensor. Each state information *si* is labeled with the ID of the related audit event. Thus, the analysis units can easily identify delayed state information by comparing their IDs. Fig. 3 illustrates the problem.

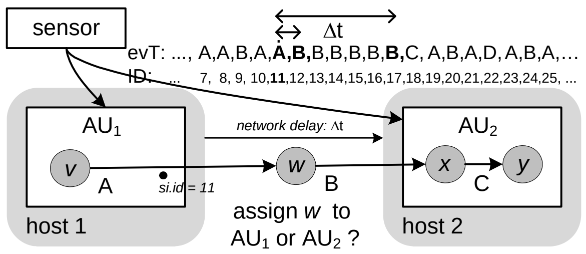
Δt
sensor
evT: ..., A,A,B,A,**A,B,**B,B,B,B,**B,**C, A,B,A,D, A,B,A,...
ID: ... 7, 8, 9, 10,**11**,12,13,14,15,16,17,18,19,20,21,22,23,24,25, ...
AU₁
network delay: Δt
AU₂
v
A
si.id = 11
w
B
x
C
y
assign w to AU₁ or AU₂ ?
host 1
host 2

Fig. 3. Unfavorable sequence of audit events

Event type sequence within Δt	Absolute frequency
A→B	18
B→C	6

Table 1. Event type sequences

It shows a fragmented signature consisting of four nodes. The nodes *v*, *x* and *y* are assigned to the analysis units AU_1 and AU_2, respectively. Which is the optimal assignment of node *w*? The assumed sequence of audit events emitted by the sensor is indicated above the AUs. Since type *B* events often occur after type *A*, while type *C* events rarely occur after *B*, node *w* should be assigned together with *v* to AU_1 to prevent delayed state information arrivals at node *w*. If delayed state information arrives the related audit event has to be re-evaluated. This requires that the audit events are buffered in the analysis units for a short time. When new state information arrives all audit events that are older than the received one can be removed from the buffer. Therefore, the required buffer memory is assumed to be constant and is not discussed further here.

The repeated evaluation of audit events though may lead to a significant additional computation effort. We demonstrate this with the example above and assume now that node *w* has been assigned to AU_2. If a type *A* event triggers a transition the respective state information has to be placed on node *w*. We assume that the information is delayed by Δt due to network latencies. The absolute frequency of type *B* and *C* events occurring after type *A* and *B* events within Δt is listed in Table 1. When events of type *B* occur after *A* within a time window of Δt, they have to be buffered in AU_2 for repeated evaluation. An event sequence in the audit data stream containing events of type *i* and *j* within Δt is called *critical* if it may cause repeated evaluations. This is the case, if an applied signature contains edges which are sensitive to *i* and *j*, respectively, and a type *j* event is expected after analyzing a type *i* event. Now, the sensor and the analysis units can dynamically update the statistics how often such critical sequences occur in the audit event stream**.** Each analysis unit also continuously logs number and types of repeatedly evaluated audit events. Based on this statistics, the responsible transition edges of a signature can be assigned to the same unit to avoid repeated analyses.

E) Iterative Adaptation of Signature Distributions

The characteristics of the audit data stream can change frequently. Therefore, the statistics on critical event sequences has to be updated continuously and the signature distribution has to be adapted accordingly. All optimization strategies described above determine an entirely new signature distribution. This requires high reorganization efforts because usually many state nodes have to be moved to other analysis units. Therefore, these strategies should only be applied to get an optimal initial signature distribution. They may be repeated perhaps 2–6 times per day. For the normal analysis process, an iterative adaptation strategy should be preferred which adapts the signature distribution with minimal reorganization continuously. If the resource consumption of the distributed analysis systems runs out of balance signature fragments from high loaded analysis units should be reassigned to less occupied ones. The fragments to be moved can be selected accordingly to one of the following procedures. (a) A fragment from the most occupied analysis unit is reassigned to the least occupied unit. (b) Like (a), but the most suitable fragment is selected whose reassignment rebalances the computation load best. This can be achieved by evaluating the computational loads of the analysis units using the analysis statistics based on equation (2). (c) That fragment is chosen from the highest loaded unit which mostly requires repeated event analysis due to delayed state information. Again, the analysis statistics have to be evaluated to select the responsible signature fragment. We consider again nodes v and w of the above example. Let node v be the source of the delayed information and $(v, w) \in E$, then node w is reassigned to the same analysis unit as node v. No additional effort for repeated event analysis will be induced by state information transfers between v and w during further analysis.

After finishing a reorganization step the efficiency gain has to be evaluated. If the adapted analysis distribution is not better than the previous one the reorganization step can be easily taken back due to the low reorganization effort required for single fragments. Thereafter an alternative fragment can be chosen for reassignment.

4. Evaluation

The distribution strategies described in the last section have been implemented and evaluated using the distributed intrusion detection system DSAM (distributed signature analysis module) that utilizes EDL multi-step signatures [4]. DSAM sensors are configurable and forward audit events only to analysis units where they are required. The DSAM analysis units are configurable as well. A set of signature fragments (state nodes) is assigned to each of them. The analysis units transfer automatically state information via the network if needed. For the performance measurements, we applied three typical signatures and examined all possible partitions (assignments) of the contained signature fragments to three analysis units. We evaluated each of the 965 possible partitions. These partitions include very efficient distributions as well as completely inefficient ones which cause an unfavorable computation or network load or both. Because of limited space we only give a brief overview about two of the applied signature examples (see Fig. 4).

The first example (Fig. 4 (a)) describes a link shell attack which exploits a vulnerability of a specific shell function and the SUID (set user ID) mechanism of the Solaris OS. If a link refers to a shell script and the scripts file name starts with a hyphen “-” an attacker can get an interactive shell with the access rights of the script owner (e.g. root) by executing the link. The second signature – the SUID script attack (Fig. 4 (b)) – describes how to gain administrative privileges in Solaris by exploiting a vulnerability of the extended file access rights.

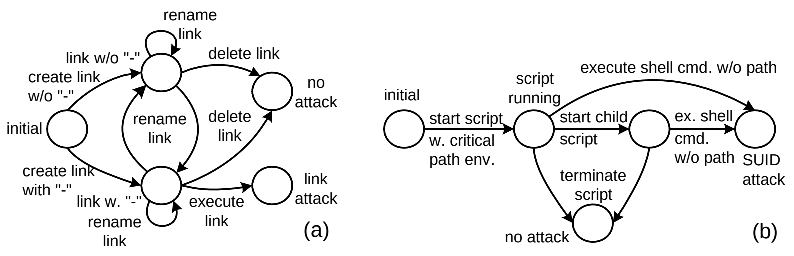

Fig. 4. (a) Shell link and (b) SUID script attack

To evaluate the analysis efficiency of DSAM in a high load situation we used a generic set of audit data. This set was created by capturing system calls of a host, while the described attacks were executed. All logged system calls that did not belong to the attacks were discarded manually. Thus, the audit data set only contains relevant attack traces of the applied signatures. Concerning the required analysis effort, this represents a stress test and a worst scenario. Additionally, the captured attack traces have been duplicated to create a sufficiently large audit data file of 6,000 events (system calls). The experiments were conducted on four machines (Intel Xeon "Prestonia", 2.66 GHz, 512 KB L2 cache, 2 GB RAM) connected by 1 GE links. One machine executed the sensor; the others run an analysis unit. First we applied the generic audit data set and evaluated the strategy A of Section 3 by assigning the three example signatures to different AUs without fragmenting them. Then we applied the strategy B to fragment and to assign fine-grained signature parts to the analysis units. We evaluated all 965 possible distributions of the signature fragments on the three AUs and measured the runtime separately for each unit. Table 2 contains the values for some selected distributions.

distribution ID		0	96	302	626
sensor	real [s]	47.95	29.63	110.64	19.72
AU_1	real [s]	47.63	37.72	110.66	19.75
	user [s]	46.89	28.67	108.47	17.95
AU_2	real [s]		29.33	116.34	25.44
	user [s]		9.83	33.89	18.08
AU_3	real [s]		32.39	113.72	23.45
	user [s]		7.84	17.97	17.97

Table 2. Runtimes of selected distributions

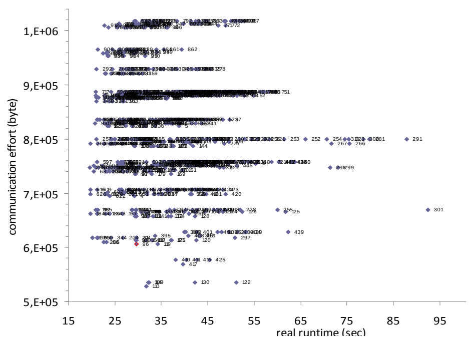

Fig. 5. Runtime and communication costs for different signature distributions

The sensor runtime is related to the slowest AU, as the sensor terminates after transmitting the last audit event to the slowest AU. The third column (ID 0) represents the non-distributed case, where only one AU is used that receives all signatures. This is the benchmark for any optimizations of the described strategies. The fourth column (ID 96) represents the simple, non-fragmented distribution for three AUs. The runtime shows a relevant improvement for the distributed case. The other columns represent the least and most efficient distribution. The diagram in Fig. 5 shows for each of the 965 signature distributions the sensor runtime and the required communication effort (bytes) for transmitting state information and audit events. Each point in the diagram represents a signature distribution and is enumerated with a unique ID. The diagram shows many analysis distributions (at the bottom) that require a low computation effort. Furthermore, the step-wise increase of the communication effort for audit event duplication can be seen. The figure indicates that there are many distributions with reasonable computation and communication effort (lower left corner) that should be selected for efficient analyses.

In order to select a suitable analysis distribution for a given network and CPU utilization a metrics based approach can be used. The metrics shall determine a suitability degree for each signature distribution and resource utilization. We defined a metrics that maps features of the audit event characteristics monitored by the sensor as well as the statistics maintained by the analysis units for a specific signature distribution onto a metrics value *M*. Simplified, *M* is defined by equation (3).

$$M = \alpha \sum_{i=1}^{\#AU} \sum_{evT \in S_i} evC(evT) + \beta \sum_{\substack{(a,b)\in E, \\ cl(a)\neq cl(b)}} commC(a,b) + \gamma \sum_{e\in E} compC(e) + \delta \sum_{c\in C} bal(c) \qquad \textbf{(3)}$$

The metrics combines and weights four features of a signature distribution by weight factors (α, β, …): (a) the communication cost for audit event duplication from equation (1), (b) the state information transfers between analysis units from equation (2), (c) the computation effort for transition condition evaluation (IDS core functionality), and (d) a load balance function *bal*(*c*) which determines for each analysis unit (cluster *c*) the deviation of the actual load from the average load of all units. Nevertheless, our experimental evaluation shows that the applied worst case scenario (the events contain a large number of attack traces) is misleading, as we were not able to find a reasonable metrics. The problem results from the additional computation effort for repeated event analysis for delayed state information. This effort changes significantly if the distances between critical event sequences change only slightly. Therefore, the computation overhead for repeated event evaluations dominates the overall runtime of unfavorable signature distributions and cannot be predicted by metrics *M*. Fig. 6 depicts the logged runtime (sec) together with the metrics prognosis (normalized to range [0,1]) for all 965 distributions. We tried different weight factors as well as additional distribution features to find a reasonable metrics, but we could not identify a common relation between metrics prediction and real runtime (see Fig. 6).

Hence, this kind of metrics cannot be used to predict optimal analysis distributions for the worst case, since they do not take all important computation dependencies between the signature fragments into account. Moreover, a further enhancement by additional statistical information is not desirable because CPU and network consumption change continuously and even minor changes would lead to entirely different distributions. Such a metrics can only discover an initial distribution with a satisfying analysis performance that must be adapted periodically. So, we applied next the iterative optimization strategy E for the worst case. When the dynamically updated sensor statistics and the analysis units suddenly indicate significant changes in the audit data characteristics the current distribution has to be iteratively adapted. In typical IT infrastructures this happens inevitable by changing user and system activities (e.g. daily schedules).

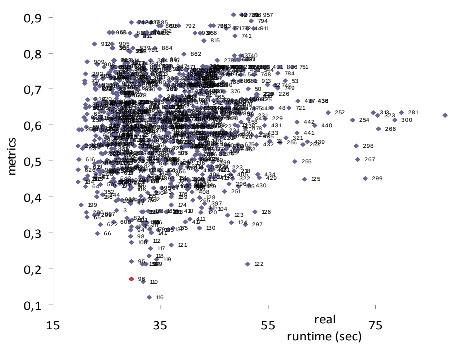

Fig. 6. Metrics prognosis and real runtime

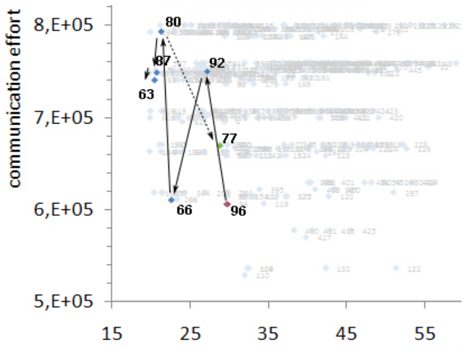

Fig. 7. Iterative optimization of distributions

Our iterative optimization strategy minimizes first the effort for repeated event evaluations by reassigning the responsible signature fragments. For this, fragments are iteratively selected and moved according to selection procedure (c) of strategy E until no computation effort is spent for repeated event evaluations. Then, as a second objective, the load balance of the analysis units is improved iteratively by selecting signature fragments according to procedure (b). For this, a suitable fragment is chosen from the most loaded analysis unit and reassigned to the least loaded unit. If the subsequent performance evaluation turns out that the new distribution performs worse than the previous one the previous distribution is restored. This causes only minor reorganization effort, as only a single signature fragment is reassigned. Fig. 7 depicts the iterative adaptation of the signature distributions by an annotated fragment of Fig. **5**.

Starting from the simple non-fragmented distribution (ID 96), where each of the three signatures is assigned to a different analysis unit, the distribution was adapted by consecutively reassigning single signature fragments generating the distributions with the IDs 92, 66, 80, 77, 80, 87, and 63. Distribution 77 was found to perform worse compared to the previous one after some analysis time. Therefore, it was reverted to ID 80 and another suitable fragment was reassigned (ID 87). The iterative adaptation of the signature distribution stops with distribution 63, when no suitable fragment for a further reassignment step can be found. This happens, when a pretty good balanced (and efficient) signature distribution is reached, such that also the “lightest” fragment from the most loaded analysis unit that requires least computation resources will impair the overall load balance, even if it will be reassigned.

To sum up, in worst case situations we always can apply the iterative optimization strategy starting with a sufficiently efficient non-fragmented signature distribution. Minor changes in the distribution can be easily taken back if they turn out to be misleading. The communication overhead can be estimated in advance. Thus, we can prevent high network utilizations. This strategy may only find semi-optimal distributions, but even these perform better than the distributed analysis of complete signatures.

5. Summary

The increasing performance of IT systems leads to a rapid growth of audit data volumes. This represents a major challenge for signature based intrusion detection systems, which already have to cope with growing signature databases. In this paper we presented various optimization strategies which aim at balancing the analysis load of complex signature based IDS over distributed analysis units. We focused on the analysis of multi-step signatures because complex attacks, i.e. on application level protocols (Web 2.0), will be much more important in the future, while simply structured attacks can be successfully prevented by today's proactive measures (e.g. address space layout randomization, stack protection). A prototype implementation was used to evaluate the achievable performance improvements by the various proposed optimization strategies. Our results indicate that relevant improvements in the efficiency of audit event analyses can be only obtained by a fine-grained assignment of fragmented signatures. But, the results also indicate that poorly chosen distributions require much more computation effort than the non-distributed baseline. Our results also show that a dynamic approach which iteratively adapts the signature distribution to the current analysis situation during runtime should be favored over a statically determined optimal signature distribution. As future work, we plan to integrate our prototype into a multi-agent platform to build up a distributed intrusion detection system responsible for securing an IT infrastructure (e.g. a company network). The system will analyze audit events from many widely distributed sensors and adapt dynamically to changes of the audit stream characteristics, analysis load, available free computation resources, and network bandwidth.

References

1. Cisco Systems Inc.: NetFlow Services and Applications. White Paper. (2002) http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm
2. McHugh, J.: Set, Bags and Rock and Roll – Analyzing Large Datasets of Network Data. In: 9th European Symposium on Research in Computer Security (ESORICS), LNCS 3193, pp. 407–422, Springer, (2004)
3. Sommer, R. and Feldmann, A.: NetFlow: Information Loss or Win? In: 2nd ACM SIGCOMM and USENIX Internet Measurement Workshop (IMW 2002), Marseille, France, 2002
4. Meier, M.: A Model for the Semantics of Attack Signatures in Misuse Detection Systems. In: 7th Information Security Conference (ISC), LNCS 3225, pp. 158–169, Springer, (2004)
5. Anagnostakis, K.G., Markatos, E.P., Antonatos, S., Polychronakis, M.: E2xB: A Domain Specific String Matching Algorithm for Intrusion Detection. In 18th IFIP International Information Security Conference (SEC 2003), pp. 217–228, Kluwer Academic Publishing, (2003)
6. Yang, L., Karim, R., Ganapathy, V., Smith, R.: Improving NFA-based Signature Matching Using Ordered Binary Decision Diagrams. In: 13th International Symposium on Recent Advances in Intrusion Detection (RAID'10), Ottawa, Canada, Sept. 2010. LNCS 6307, pp. 58–78, Springer (2010)
7. Shemitz, J.: Using RDTSC for Pentium Benchmarking. Visual Developer Magazine. Jun./Jul. (1996), Coriolis Group, Scottsdale, AZ, USA. http://www.midnightbeach.com/jon/pubs/rdtsc.htm
8. Vasiliadis, G., Antonatos, S., Polychronakis, M., Markatos, E.P., Ioannidis, S.: Gnort: High Performance Network Intrusion Detection Using Graphics Processors. In: 11th International Symposium on Recent Advances in Intrusion Detection (RAID'08), LNCS 5230, pp. 116–134, Springer, (2008)
9. Kruegel, C., Toth, T., Kerer, C.: Decentralized Event Correlation for Intrusion Detection. In: Intl. Conference on Information Security and Cryptology (ICISC), LNCS 2288, pp. 114–131, Springer (2001)
10. Colajanni, M., Marchetti, M.: A Parallel Architecture for Stateful Intrusion Detection in High Trac Networks. In: IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation, IEEE Press, (2006)
11. Schaelicke, L., Wheeler, K., Freeland, C.: SPANIDS: A Scalable Network Intrusion Detection Loadbalancer. In: 2nd Conference on Computing Frontiers (CCF 2005), pp. 315–322, ACM, (2005)