Experimental Studies Using SOARA: An Approach to Reduce Alarm Rates on Streams of Intrusion

Jorge Levera
jlevera@cs.uic.edu

Robert Grossman
grossman@uic.edu

University of Illinois at Chicago
Department of Computer Science
Chicago, IL, 60612, USA

and

Benjamin Barán
Universidad Nacional de Asunción
Centro Nacional de Computación
San Lorenzo, Paraguay
bbaran@cnc.una.py

Abstract

The overwhelming number of alarms generated by rule-based network intrusion detection systems makes the task of network security operators ineffective. Preliminary results on an approach called SOARA shows that false positive alarms can be reduced by detecting changes on streams of alarms using sketch-based time-decaying moving median. SOARA keeps a memory efficient sketch summary of the normal stream of alarms using relevant features. Sketches are updated according to established policies and a time-decaying moving median procedure is used on historical data to detect abnormal alarm rates on the stream. SOARA shows promising results on labeled and unlabeled test sets by focusing on exceptions on the normal stream of alarms, diverting the attention away from false positives.

Keywords: Data stream, intrusion detection, sketch summaries, time-decaying moving median.

1 Introduction

Given the proliferation of valuable assets on the Internet, it has become clear that network security operators are looking for robust intrusion detection systems (IDS) that are efficient, effective and easy to manage. Roughly speaking, IDS can be classified in host-based IDS (HBIDS) and network IDS (NIDS). HBIDS monitors users pattern of behavior on a particular host, and try to detect abnormal sequence of commands, or vulnerability exploits. NIDS monitors network packets in order to detect unusual or potentially hazardous traffic.

A NIDS can categorized as rule-based (RBIDS), statistical-based (SBIDS) or protocol based (PBIDS) intrusion detection system. RBIDS inspect network packets passing through the network and compare them with a set of rules. A match triggers an alarm. SBIDS build a statistical model of the network behavior, and trigger alarms for every deviation from the model. PBIDS trigger alarms for every departure from standard network protocols.

The most popular approach is RBIDS [9] because it is very effective against known attacks and efficient when the set of rules is kept to a reasonable size. Its main drawbacks are the impossibility of detecting unknown attacks and the overwhelming amount of alarms that could be generated [9, 10].

A great number of alarms generated by RBIDS are false positives [5, 7, 10] (i.e., attack did not actually take place). False positive alarms could be reduced by adding customized filters to the IDS or by eliminating the rules that causes the noise. Sometimes, it is difficult to apply any of those changes because either the IDS belongs to another organization (e.g., outsourcing, cooperative distributed IDS) or it is not safe to delete a specific rule. Then, the security operator is faced with the problem of detecting true positive alarms among a pile of false positives.

Several approaches have been proposed to reduce the number of alarms. Solutions were proposed from the realm of data mining [13, 14, 17], machine learning [20] and visualization [10, 26]. This paper presents a Stream Of Alarms Reduction Architecture (SOARA) to reduce intrusion alarms using change detection algorithms over streams of intrusion alarms.

In the context of SOARA, stream of alarms generated by IDS arrive to a SOARA trimmer which reduces them to simple and expressive time series. Those time series or signals are used by SOARA analyzer to build sketch-based summaries that model the behavior of the stream of alarms [11,16]. Changes on the stream of alarms are detected using a time-decaying moving median procedure [6] on signal values kept in the sketch. The signals that deviate significantly from the normal pattern of behavior are used to generate a predictable and manageable number of generalized alarms. Those trend alarms help network security operators focus on the most interesting data. Experimental results on labeled and unlabeled datasets show that SOARA can detect changes on the trend of alarms by focusing on exceptional data.

The rest of the paper is organized as follows: Section 2 surveys related work on alarm reduction. Section 3 presents the SOARA approach to alarm reduction. Section 4 shows experimental results. Section 5 discusses future work and concluding remarks.

2 Related work

Data mining techniques has been applied to intrusion detection data for several years [17, 21, 16, 28]. Lee and Stolfo [17] were the first ones to use clustering and association rule on system and network features in order to learn their normal behavior. Manganaris et al. [21] used alarm features to characterize the normal stream of alarms using association rules. Ye and Li [28] used clustering and classification on alarm features. Most of the previous approaches require a carefully selected training set in order to build a model of normal behavior and classify unseen data. Portnoy et al. [23] pioneered on unlabeled intrusion datasets and proposed a clustering technique to detect intrusions. Likewise, SOARA works on unlabeled data and requires no training.

Julisch and Dacier [14] presented a different approach to alarm reduction based on conceptual clustering techniques. In a later work, Julish [15] used clustering to find the root of most false positives. Shortcomings of their approaches include periodic tuning to adjust the model to changing network conditions, and the numerous parameters that are not trivial to determine [5]. Though SOARA requires a couple of tuning paramenters, they can be found and tuned easily requiring infrequent adjustments.

Lately, several authors have proposed techniques to correlate alarms [7, 22, 25]. Valdes and Skinner [25] provided a mathematical framework for alert correlation, extending ideas from multisensor data fusion. Cuppens and Miege [7] clustered, merged and correlated alarms in a cooperative IDS environment. Ning et al. [22] built attack scenarios. They correlated alarms by partial match of prerequisites and consequences of attacks. Correlation condenses alarms to a few groups and facilitates the distinction between false and true positives. However, prerequisites and consequences conditions are not trivial to find.

In the context of streaming data, several change detection algorithms for streams has been studied. Gilbert et al. [11] proposed sketch summaries base on wavelets to approximate aggregate data from streams. Barford et al. [5] reported results of signal analysis of four classes of network traffic anomalies using wavelet filters over streams. Krishnamurthy et al. [16] proposed a sketch-based summary of streams and applied several forecasting models. The authors in [5, 16] presented a general model that could be adapted to alarm reduction. Their experimental work is base on streams of network packets and flow information rather than streams of alarms. In [16] they mentioned false positive reduction as on going work based on the top *n* most abnormal streams. SOARA extends these ideas to the context of rule-based IDS where the input is a stream of alarms instead of network flow data. SOARA incorporates memory efficient implementation of their algorithm based on cache memory policies. Besides, sketch summaries used in SOARA store historical data at different time intervals in order to represent individual stream of alarms more accurately.

There are several other related work in the context of stream mining. Cohen and Strauss [6] formalized the problem of time-decaying aggregates and statistics over streams of data. Dong et al.[8] discussed mining of changes from data streams and proposed the expiration of old data based on data's distribution instead of arrival time. Yamanishi and Takeuchi [27] presented a framework for detecting outliers and change points on non-stationary time series data. Babcock et al. [3] presented techniques for maintaining variance and k-median clustering over data stream windows. SOARA brings together several of the previous mining techniques to the context of intrusion alarm reduction.

3 Reducing alarms with SOARA

This Section presents SOARA, a sketch-based time-decaying moving median approach to reduce alarm rates based on change detection over streams of intrusion alarms. First, the model used to summarize the stream is presented. Then, the time-decaying moving median approach is used to detect changes on the normal stream of alarms.

Throughout this paper, IDS alarms are assumed to be a set of *n* attributes $A=(a_0, a_1, ..., a_{n-1}) \in \Omega$ that describe a particular alarm and the network packet that triggered it. Common attributes are: time of the event, identifier of the rule that triggered the alarm, source IP address, target IP address, and target port number to name a few. For instance, Table 1 depicts a set of alarms generated by Snort IDS [24] where the first row defines an alarm $A_0 =(a^0_0=1;\ a^0_1=$"2004-03-12 10:24:35"$;\ a^0_2=$"SNMP trap tcp"$;\ a^0_3=58;\ a^0_4=40;\ a^0_5= 5912;\ a^0_6=162) \in \Omega$.; the second to fifth rows define A_1 to A_4 in a similar way.

Table 1. A set of intrusion alarms A_i from a Snort IDS and their corresponding attributes a^i_j

a_0 = rule Id	a_1 = Time Stamp	a_2 = Description	a_3 = Time-to-live	a_4 = IP length	a_5 = IP Id	a_6 = Target Port Number
1	2004-03-12 10:24:35	SNMP trap tcp	58	40	5912	162
2	2004-03-12 10:24:37	SCAN Proxy (8080) attempt	48	40	57215	8080
3	2004-03-12 10:24:39	SNMP request tcp	38	40	9835	161
4	2004-03-12 10:24:40	SCAN SOCKS Proxy attempt	38	40	59362	1080
2	2004-03-12 10:24:41	SCAN Proxy (8080) attempt	128	40	57310	8080

3.1 Modeling normal stream of intrusion alarms

There are several approaches to model streams of data [16]. In SOARA alarms arrive sequentially as a stream of events $A_0, A_1, A_2, \ldots$ to the SOARA trimmer. The goal of the trimmer is to reduce the number of attributes in A by combining them or discarding. The SOARA trimmer build streams of features $K = <k_0, k_1, .., k_L>$ where $K=F(A)$ is a function of subset of attributes in A. The stream $K_0, K_1, K_2, \ldots$ is used by SOARA analyzer in order to build sketch summaries of the stream at predefined time intervals. Figure 1 shows the architecture behind SOARA and its components. For simplicity, in this paper the knowledge database associated with SOARA is not used.

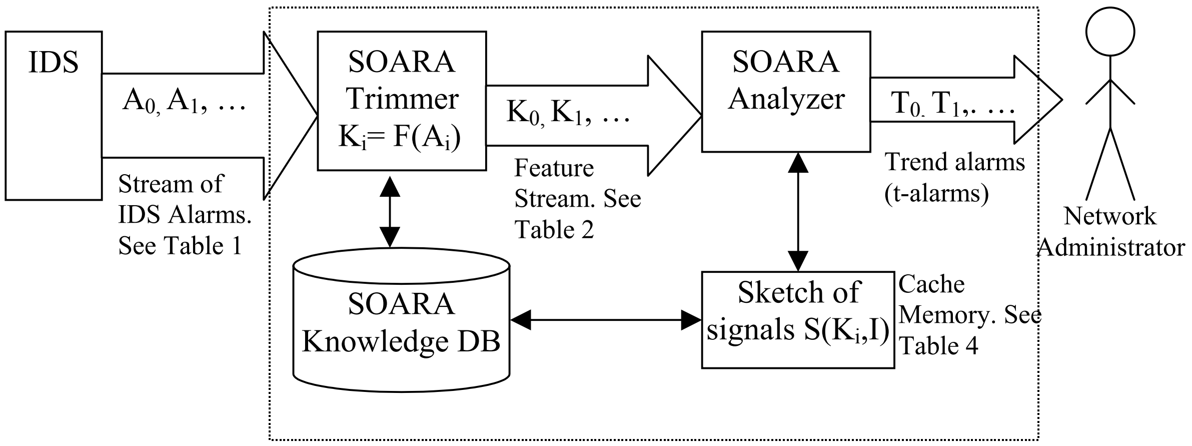
IDS
$A_0, A_1, \ldots$
Stream of IDS Alarms. See Table 1
SOARA Trimmer $K_i = F(A_i)$
$K_0, K_1, \ldots$
Feature Stream. See Table 2
SOARA Analyzer
$T_0, T_1, \ldots$
Trend alarms (t-alarms)
Network Administrator
SOARA Knowledge DB
Sketch of signals $S(K_i, I)$
Cache Memory. See Table 4

Figure 1. SOARA architecture

For the purpose of this paper, the attributes used in K are: (k_0 =identifier of rule that triggered alarm, k_1=destination port number, k_2=time-to-live attribute of network packet that triggered the alarm). The attributes in K were chosen based on previous experiments with sets of alarms [18]. Table 2 shows a stream of feature vectors $K_0, K_1, K_2, \ldots$ occurring at descrete time intervals t_i. The first row represents the feature vector $K_0 = < k^0_0 = 1; k^0_1 = 58; k^0_2 = 162 >$ that occurred at time $t=485$ minutes, the second row represents a feature vector $K_1 = < k^1_0 = 2; k^1_1 = 48; k^1_2 = 8080>$ that occurred at the descreate time $t=487$, and so on.

Table 2 Feature vectors K_i built by SOARA trimmer

Discrete Time	k_0**=Rule Id**	k_1**=Time-to-live**	k_2**=Target Port Number**
485	1	58	162
487	2	48	8080
489	3	38	161
490	4	38	1080
491	2	128	8080

During consecutive time intervals of size *t* called I_0, I_1, I_2... the SOARA analyzer receives streams of feature vectors $K_0, K_1, ...$ from the SOARA trimmer. The analyzer summarizes the alarm feature vectors received during each I_i, and builds signals or time series of the form $S(K, I_i) = u_i$ where u_i is an update value during the interval I_i. In this paper, the update value u_i correspond to the rate of the feature vector K_j during the interval I_i. Table 3 shows five instances of signals $S(K_i, I_0)$ where $I_0=[485, 500]$ is a 15-minute interval. The first row represents the signal $S(<485, 1, 58, 162>, I_0) = 0.27$, the second one signal $S(<487, 2, 48, 8080>, I_0) = 0.33$, and so on.

Consequently, for each value K_i there is a time varying signal $S(K_i, I)$. The arrival of a new alarm A_i during the interval I_j causes the signal $S(K_i, I_j)$ to be updated with $S'(K_i, I_j) = S(K_i, I_j) + u_i$. The goal of SOARA is to identify all those signals $S(K_i, I_j)$ with significant changes in their normal pattern of behavior. The fact that k_1 correspond to a rule identifier for this particular implementation of SOARA, relates signals $S(K, I)$ and intrusion alarms A (i.e., $k_0=a_0$). For this reason, in this paper we use the term signal rate and alarm rate interchangeably.

Table 3 Signals S(K_i, I) within a 15-minute interval I_i

Discrete Time	k_0**=Rule Id**	k_1**=Time-to-live**	k_2**=Target Port Number**	u_i**=Rate per minute**
485	1	58	162	0.27
487	2	48	8080	0.33
489	3	38	161	0.27
490	4	38	1080	0.27
491	2	128	8080	0.07

In SOARA, sketch summaries are implemented by storing a set of *m* distinct signals $S(K_i, I)$, $0\leq i\leq m$ that are identical in term of space and structure allocated in memory. Associate with each signal $S(K_i, I)$, SOARA stores the time interval I_j of last update to the signal, and historical data u_i.

The historical data field stores the last *w* update values $(u^i_0, u^i_1,... u^i_{w-1})$ of signal $S(K_i, I_j)$ during different intervals I_j. Note that the historical data $(u^i_0, u^i_1,... u^i_{w-1})$ correspond to observed values of signal $S(K_i, I_j)$ at different, possibly not consecutive, time intervals I_j. The only temporal restriction regarding two consecutive values u^i_j and u^i_{j+1} is that u^i_j was observed during an interval I_h prior to the interval I_n where u^i_{j+1} was observed. Table 4 shows sketch summaries of signals $S(K_i, I)$ as they are represented in SOARA. Each row correspond to a signal $S(K_i, I)$ for a specific value of K_i. The first column stores the last update interval time I_i for signal $S(K_i, I)$ represented in discrete time according to the value of interval I_i in minutes. For instance, the first row correspond to $K_0=(k^0_0=1, k^0_1=58, k^0_2=162)$ implying that signal $S(<1,58,162>, I)$ has been updated for the last time during interval *I=[1140, 1155]* minutes, and has five values of u^0_i (i.e., historical data) stored in chronological order (0.13, 0.07, 0.13, 0.07, 0.13, 0.13).

Table 4. A snapshot of sketch summaries used in SOARA. Rows represent signals *S(K,I)* and are identified by the combination of rule id, target port number and time-to-live of the network packet that triggered the alarm. For each signal, a windows of size *w* stores the signal's rate

Discrete Update Time I	***Rule id***	***Time-to-live***	***Target Port Nbr***	**Rate 0**	**Rate 1**	**Rate 2**	**Rate 3**	**Rate 4**	**Rate 5**
1155	*1*	*58*	*162*	0.13	0.07	0.13	0.07	0.13	0.13
1245	*2*	*48*	*8080*	0.47	0.40	0.20	0.20	0.40	0.07
1275	*3*	*38*	*161*	0.07					
1350	*4*	*38*	*1080*	0.20	0.40	0.20	0.20	0.20	0.80
1260	*2*	*128*	*8080*	0.7					

As mentioned before, SOARA keeps track of a maximum of *m* signals $S(K_i, I)$ at the same time. In order to use memory efficiently, signals $S(K_i, I)$ are treated like cache entries in a cache memory. At the beginning, the first $S(K_i, I)$ distinct signals are tracked and stored in memory. Their respective entries u_i are updated as new alarms arrive and time intervals I_i increase. When the *m+1* distinct signal arrives, a replacement algorithm is used to allocate a new entry according to a predefined policy.

In this paper, SOARA uses a replacement policy based on time of last update and priority assigned to cache entries. For instance, when the cache is full (i.e., there are m distinct signals been tracked), the signal that has not been updated for the longest time is removed. If there is more than one candidate entry for replacement, then the entry with the lowest priority is replaced.

Signal priorities are assigned according to the rule id related to the signal. Most IDS have a priority associated with the alarms generated by it. The priority identifies the level of importance associated with a particular type of alarm. In Snort IDS, each rule has a priority *p*, $1 \leq p \leq 5$ associated with it where $p=1$ rank the most dangerous alarm types. Consequently, every alarm generated by Snort due to a match with rule id *r* that has priority *p* will have a priority of *p*. In SOARA, every signal $S(K_i=<k_0, k_1, k_2> , I)$ where $k_0=r$ has a priority *p*. For example, if rule id four is an alarm of priority one according to the IDS, then any signal $S(K_i, I)$ where $k_0 = 4$ will have a priority of one to stay in the cache.

3.2 Change detection algorithm

In this Section, a simple approach to change detection is presented. In SOARA, a change detection algorithm computes the deviation of each sketch-summarized signal $S(K_i, I)$ with respect to a value predicted by a model. For simplicity, SOARA use moving median as the predictive model. Every *t* minutes, the value predicted by the moving median is compared against the new rate value for that particular combination of rule ID, TTL and target port number. The top *d* signals with the greatest deviation (i.e. error value) are considered trend alarms (t-alarm) because they deviate significantly from the rate values predicted.

Formally, the *w* observed values $(u^i_0, u^i_1,.., u^i_{w-1})$ of signal $S(K_i, I)$ are used as input to a predictive model $F(u^i_0, u^i_1,.., u^i_{w-1})$. The value $F(u^i_0, u^i_1,.., u^i_{w-1})$ is compared against the next new value u^i for $S(K_i, I)$. Trend alarms called t-alarms are generated for the top d values $e_i = u^i - F(u^i_0, u^i_1,.., u^i_{w-1})$. Intuitively, t-alarms represent the top d signals $S(K_i, I)$ that differ the most from the predicted value. Note that in SOARA, positive values of e_i denote an increase on the rate of alarms related to signal $S(K_i, I)$. SOARA focuses on rate increases only because they tend to indicate the beginning of an attack or the abnormal increase of a normal noise.

```
i = 0;   S[ ]=0;
F[ ]=0;  // initialize forecast value for signals S

Forever do
        {
        While ( size_of_time_Interval($I_i$) <= $t$ )do
                {
                A =Receive_alarm();
                // aggregate to vector K[]
                $K_j$[ $k_0$, $k_1$,.. , $k_L$ ] = f(A);
                // update signal S
                if ($K_j$[ ] already exist) then
                        S[$K_j$, $I_i$] = S[$K_j$, $I_i$] + $u_i$;    // update signal S
                else {
                        Allocate_space(S[ ]);  //remove oldest S[ K] if necessary
                                                // to keep track of at most $m$ distinct signals
                        Insert_New_Signal( S[$K_j$, $I_i$] );   // keep track of new signal
                    }
                // increase time
                Update($I_i$ , clock() );
                }
        // compute error in signals S[K, $I_i$]
        compute_error(S[ ], F[ ]);

        // trigger alarm for most deviated signal S[K, $I_i$]
        trigger_trend_alarm(S[ ]);

        // update forecast value for signals S[ ]
        predicted_value(F[K]);

        //move to next time interval $Ii$
        i = i + 1;
        }
```
Figure 2. Pseudo code for SOARA

It should be noted that if error is normalized, the relative importance of each entry is eliminated. Usually, the more often a type of alarm is triggered, the more important its deviation from the normal pattern is.

The error e_i associated with a signal $S(K_i, I)$ decreases with time. Say the current time interval I_j has just finished. If the signal $S(K_i, I)$ has last been updated during interval I_h, then the error e_i associated with signal $S(K_i, I)$ is reduced to $e_i = e_i{*}\ (I_h / I_j)$. The error is recalculated at the end of each time interval, before t-alarms are generated for the d most deviated signals. Figure 2 shows the pseudo code for SOARA.

4 Experimental results

SOARA was tested with four sets of alarms generated by Snort IDS version 1.9.1 with default set of rules. One of the IDS was located on a Public Sector Metropolitan Area Network of Asuncion, Paraguay [4]. Another set of alarms was generated by a Snort IDS located on the

Abilene network [1]. An additional set belongs to an IDS running at the University of Illinois at Chicago (UIC). The last set of alarms was generated using DARPA99 Intrusion Detection Set [19]. Table 5 gives an overview of the datasets used.

Table 5. Datasets used on experimental results. Some characteristics are shown to indicate their diverse nature and evaluate the results on each set

Dataset	**Network Type**	**Number of Alarms**	**Distinct Rule Ids**	**Time Span (days)**	**Distinct Source IP Addresses**	**Distinct Target IP Addresses**
UIC Network	LAN	592,121	52	180	20,161	2,429
Pub. Sect. MAN	MAN	430,794	67	7	514	272
Abilene Network	WAN	10,357,673	49,571	90	208,527	253,790
DARPA99	Synthetic	23,050	84	10	109	187

In order to compare SOARA's behavior over different datasets, we used the same tuning parameters for all datasets. In general, different networks require different tuning paramenters that depend on the amount of alarms generated by the IDS and received by the SOARA analyzer. Besides, the size *m* of the cache needed to hold signals S(K,I) depends on the diversity of the alarms generated by the IDS influence. The more diverse the origin, number and type of alarm the more likely it is that distinct signals will need to be tracked. The size of the network monitored by the IDS also affects the size of the cache.

The following parameters were used during the experiments. Time intervals I_i were generated every 15 minutes. SOARA kept track of a maximum of $m=30$ distinct signals $S(K,I)$. For each signal $S(K,I)$, SOARA kept a windows of size $w=7$ update values. Only $d=1$ t-alarm was generated for every interval I_i. The error value e_i decayed at a rate of *0.5* for every time interval I_i. In consequence, the moving median procedure ran every $t=15$ minutes over the historical data kept for every signal.

Table 6 shows the reduction obtained over the datasets. The reduction was computed as the number of RBIDS alarms to be examined on the exceptional signal pointed by t-alarms over the total number of RBIDS generated during the same time interval I_i.

Table 6. Average reduction on four datasets

Dataset	**Avg. Reduction**
Public Sector MAN	0.89
Abilene Network	0.78
UIC Network	0.59
DARPA 99	0.77

In general, UIC network dataset did not generated many raw alarms at $t=15$ minute intervals. In order to obtain greater reduction on these datasets, a larger *t* value is needed.

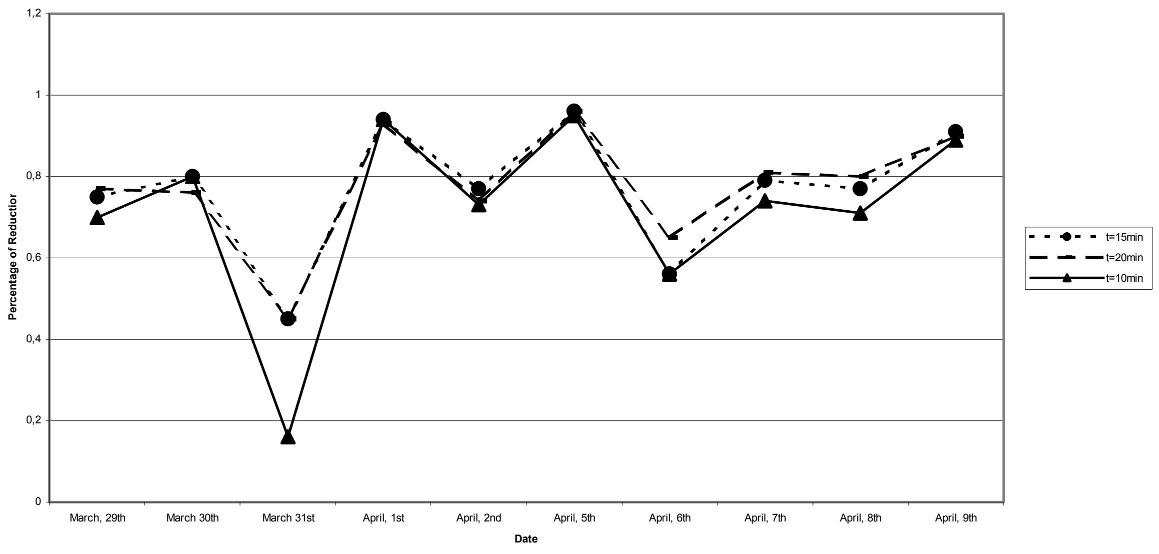

Figure 3. Reduction on the number of alarms detected on the labeled dataset DARPA99 using SOARA.

During experiments with DARPA99 dataset, SOARA was tested using different values of *t* (i.e. time intervals *Ii* of size *t*). SOARA ran with 10, 15 and 20 minute interval. Figure 3 shows the reduction on the number of alarms to be inspected over dataset DARPA 99. Figure 4 shows the effectiveness or percentage of true positives detected on the labeled dataset DARPA99 using SOARA. Most true positives detected were in the subset of raw alarms indicated by t-alarms (i.e., the signal with the biggest error value).

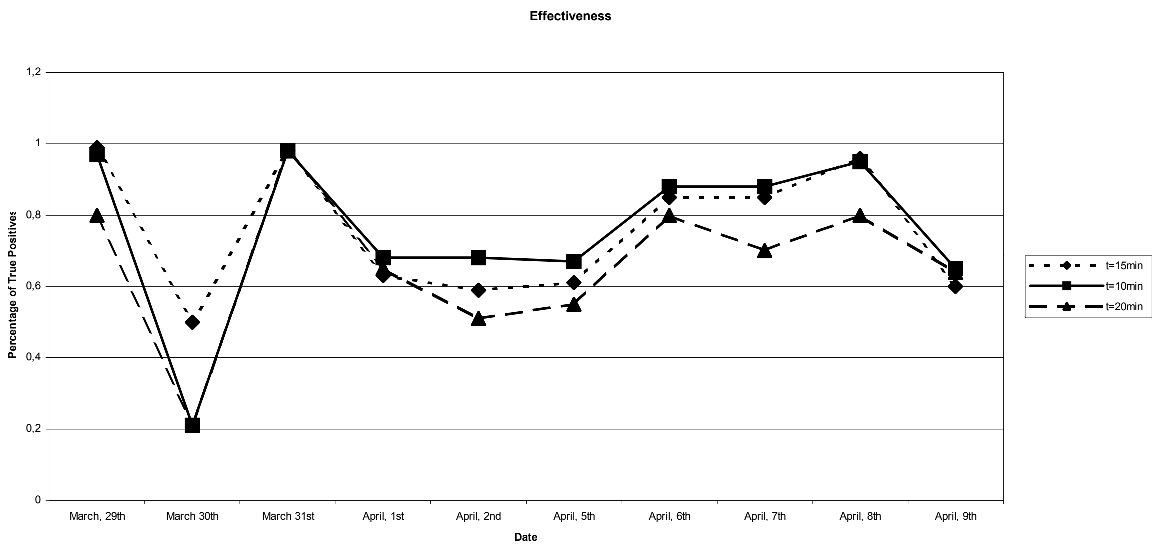

Figure 4. Effectiveness of SOARA (i.e., percentage of true positive alarms detected) on the labeled dataset DARPA99 using SOARA.

5 Conclusion

Experimental studies with SOARA aims at reducing the number of intrusion alarms to be examining by network security experts. Multidimensional signals aggregate alarms by several feature attributes using sketch summaries. A cache memory implementation of sketches allows efficient use of resources. The time-decaying moving median procedure finds exceptional values or changes on the stream of alarms by comparing the observed rate of alarms against the predicted one. Tests on several datasets show promising reduction on the number of alarms to be inspected. In particular, the labeled dataset DARPA99 showed that SOARA could improve the effectiveness of network security operators by focusing on the most interesting data.

In the future, SOARA will integrate additional network data to further reduce false positive alarms. Passive operating system fingerprints data could used to reduce the number of alarms triggered for the wrong operating system. This will reduce the number of t-alarms without significant computational demand. Besides, network flow data could be used to deduce open ports or service provided by distinct hosts on the network. The effectiveness of SOARA could be improved by keeping track of signals associated with services offered on the network only.

Further testing needs to be done on labeled datasets like DARPA99 to verify the reduction on the number of false positives versus the percentage of true positives detected by t-alarms.

In addition, a distributed SOARA system is being designed to exploit the fact that sketch summaries can be exchanged in a distributed IDS environment to improve the efficiency and cooperation between composing IDS.

References

[1] Advanced Network Management Lab, *The Abilene Project*, University of Indiana, Bloomington, Indiana, USA.

[2] Axelsson, Stefan, "The base-rate fallacy and the difficulty of intrusion detection," Journal of the ACM Transactions on Information Systems Security, 3(3), pp. 186-205, 2000.

[3] Babcock, B., Datar, M., Motwani, R. and O'Callaghan, L., "Maintaining variance and k-medians over data stream windows," In Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2003, pp. 234-243, San Diego, California.

[4] Barán, B., Aguayo, S., "Caracterización del Backbone ATM de la Red Metropolitana del Sector Publico Paraguayo", XXIV Conferencia Latinoamericana de Informática, Quito, Ecuador 1998.

[5] Barford, Paul, Kline, Jeffery, Plonka, David and Ron, Amos, "A signal analysis of network traffic anomalies," In Proceedings of the second ACM SIGCOMM Workshop on Internet measurment, pp. 71-82, Marseille, France, 2002.

[6] Cohen, Edith and Strauss, Martin, "Maintaining time-decaying stream aggregates," In Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2003, pp. 223-233, San Diego, California.

[7] Cuppens, Frederic and Miege, Alexandre, "Alert correlation in a cooperative intrusion detection framework," In Proceedings of the 2002 IEEE Symposium on Security and Privacy, May 2002.

[8] Dong, G., Han, J., Lakshmanan, L. V.S., Pei, J., Wang, H. and Yu, P. S., "Online mining of changes from data streams: Research problems and preliminary results, " In Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams, San Diego, CA, June 2003.

[9] Erbacher, R. F. and Sobylak, K., "Improving Intrusion Analysis Effectiveness," Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, George Mason University, September 24-26, 2003.

[10] Eskin, E., Arnold, A., Prerau, M., Portnoy, L. and Stolfo, S., "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data," Data Mining for Security Applications. Kluwer 2002.

[11] Gilbert, A. C., Kotidis, Y., Muthukrishnan, S. and Strauss, M., "Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries," In Proceedings of the 27th International Conference on Very Large Data Bases, pp 79-88, Rome, Italy, 2001.

[12] Hussain, Alefiya, Heidemann, John and Papadopoulos, Christos, "A framework for classifying denial of service attacks," In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp 99-110, Karlsruhe, Germany, August 25-29, 2003.

[13] Julisch, Klaus, “Mining alarm clusters to improve alarm handling efficiency,” In 17th Annual Computer Security Applications Conference (ACSAC), pp 12-21, December 2001.

[14] Julisch, Klaus and Dacier, Marc, “Mining Intrusion Alarms for Actionable Knowledge,” in SIGKDD’02, Edmonton, Alberta, Canada, 2002.

[15] Julisch, Klaus, "Clustering Intrusion Detection Alarms to Support Root Cause Analysis", in ACM Transactions on Information and System Security 6(4), November 2003.

[16] Krishnamurthy, B., Sen, S., and Zhang, Y. and Chen, Y., “Sketch-based change detection: methods, evaluation, and applications,” In Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement, pp 234-247, Miami Beach, FL, USA, 2003.

[17] Lee, Wenke and Stolfo, Sal. “Data Mining Approaches for Intrusion Detection" In Proceedings of the Seventh USENIX Security Symposium (SECURITY '98), San Antonio, TX, January 1998.

[18] Levera, J., Barán, B., and Grossman R., *“Experimental Studies Using Median Polish Procedures to Reduce Alarm Rates in Data Cubes of Intrusion Data”*, Workshop on Intelligence and Security Informatics for National and Homeland Security, Hsinchun Chen, Reagan Moore, Daniel Zeng, John Jeavitt, editors, LNCS 3073, pages 482-491, Springer Verlag, New York, 2004.

[19] Lincoln Laboratory, Massachussets Institute of Technology, DARPA 99 Intrusion Detection Data Set Attack Documentation. [Online] Available: http://www.ll.mit.edu/IST/ideval/docs/1999/attackDB.html.

[20] Maloof, M. A. and Michalski, R. S., “A Method for Partial-Memory Incremental Learning and its Application to Computer Intrusion Detection,” In Proceedings of the 7th International Conference on Tools with Artificial Intelligence, pp 392-397, Washington, DC, November 5-8, 1995.

[21] Manganaris, S., Christensen, M., Zerkle, D., and Hermiz, K., “A Data Mining Analysis of RTID Alarms,” Computer Networks, 34(4) , October 2000.

[22] Peng Ning, Yun Cui and Douglas S. Reeves, “Constructing attack scenarios through correlation of intrusion alerts, ”In Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA, 2002.

[23] Portnoy, L., Eskin, E. and Stolfo, S. J.. “Intrusion detection with unlabeled data using clustering" In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001). Philadelphia, PA: November 5-8, 2001.

[24] Roesch, M., "Snort - lightweight intrusion detection for networks," In Proceedings of Thirteenth Systems Administration Conference (LISA '99), pp. 229--238, The USENIX Association, Berkeley, California, 1999.

[25] Valdes, Alfonso and Skinner, Keith, “Probabilistic Alert Correlation,” Workshop on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, Number 2212, Springer-Verlag, 2001.

[26] Vert, G., Fricke, D. A., and McConnell, J. C., “A visual mathematical model for intrusion detection” In Proceedings of the 21st National Information Systems Security Conference, Crystal City, Arlington, VA, USA, October 5-8 1998.

[27] Yamanishi, Kenji and Takeuchi, Jun-ichi, “A unifying framework for detecting outliers and change points from non-stationary time series data,” In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 676-681, Edmonton, Alberta, Canada, 2002.

[28] Ye, N. and Li, X., “A Scalable Clustering Technique for Intrusion Signature Recognition,” In Proceedings of the 2001 IEEE, Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 5-6 June, 2001.