

Baseline Analysis of Security Data

by: Ophir Rachman, Ph.D, 02/18/2005

<http://www.securitydocs.com/library/3018>

Abstract

In the past 10 years we have witnessed many theoretical and semi-practical attempts to use data mining technology in the area of intrusion detection. The governing approach was to use data mining algorithms to detect anomalies that represent intrusions, or attacks. There was an attempt to use data mining and anomaly detection as a replacement for existing intrusion detection techniques. However, except for specific areas (such as wide spread worms) there is no practical implementation of this approach. The main reason being that existing technologies are suffice for detecting intrusions, and, that data mining does not provide enough context to serve as an intrusion detection tool.

Regardless of the development in the SIM (Security Information Management) area, there is still a huge problem with existing detection tools. For both signature based and behavioral rules approach, most of the rules are too generic. The consequence is that although these tools detect all the intrusions, they detect much more than that. This problem, known as false positives, is a big barrier for intrusion detection tools to cross before their deployment can be practical. To date, intrusion detection vendors, or more precisely security experts, are struggling with an inherent conflict and are sometimes forced to write less adequate detection rules just to reduce the number of false positives.

In this paper we suggest a different approach for using data mining technology in the intrusion detection area. We claim that the best positioning for a data mining technology within an intrusion detection system is not as a detection engine, but rather as an analysis layer that will filter out the false positives. The ability of data mining technology to build behavioral models representing 'normal' behavior of data is most suitable to model the data generated by the intrusion detection engines. The approach presented in this paper is being developed by [Securimine Software Inc.](#), located in Mountain View, CA, USA. A product called Securimine for Snort (SFS) is distributed as a freeware for analysis of Snort alerts.

Introduction

The computer security technology has developed greatly over the past decade. This is the result of the massive exposure of computers on the Internet which led to an evolution of security threats from old fashion viruses to cyber attacks such as worms, distributed denial of service attacks and remote attacks using software vulnerabilities to intrude computer systems.

The evolution of cyber attacks created new defensive challenges, and consequently, new protection technologies emerged. Today we witness a wide range of products that protect against a wide spectrum of possible attacks. Of these products one can mention Routers, Firewalls, Antivirus, Network/Host Intrusion Detection/prevention (NIDS/NIPS, HIDS/HIPS), Honeypot Virtual Networks, Virtual Private Networks (VPN), Distributed Denial of Service (DDOS), Personal Firewalls, Anti Spam, Anti Malware, Web Application Protection, Vulnerabilities Scanners, Cryptographic solutions, and more.

The security market has developed and matured greatly in recent years, especially if we consider the basic network security elements such as routers, firewalls and intrusion detection systems. These components are widely deployed and the deployment rate is expected to grow rapidly in the next years especially with the introduction of UTM (unified threat management) appliances that include router,

firewall and IDS functionality. Other areas, like the desktop and portable PC's protection are not yet matured and are expected to consolidate and mature slowly in the next years.

Still, building a high quality security system is a complex and costly task. A security system is built from multiple security devices and requires special talent and qualified personnel to make the system useful. Some of the challenges of deploying a security system are: aggregation of data from multiple devices, normalizing and displaying data from multiple formats, correlation between data of different devices, handling the large volumes of data, data analysis and forensics, etc.

These challenges created a need for sophisticated management tools and led to the development of Security Information Management (SIM) solutions. The focus of SIM products is to consolidate the endless stream of information produced by security products and to provide platform for viewing and managing this data.

Although SIM products are a step in the right direction, existing SIM users are still not satisfied. The systems are not automated and require intensive human labor for monitoring and analyzing the huge amounts of data generated by the security devices. This highlights an inherent problem in security devices that was not solved by current SIM products - false positives.

Security devices can not always distinguish between real attacks and normal behavior of the system. Often, legal network or users activity is detected by security devices as suspicious or malicious and an alert is generated. These alerts, known as "false positives", are one of the major problems in SIM deployments today. The amount of security information generated by security products is overwhelming. Depends on the size of the system, the number of alerts generated by security products per day may amount to hundreds of thousands if not millions even if there is no real attack involved.

The problem of false positives is inherent in security products. All the attempts made to develop a more accurate detection technology failed. The problem is that attacks indeed do look like normal traffic. Additionally, normal traffic is not something that can be formalized as it varies from site to site. The false positives problem introduces a conflict that is not yet resolved: on one side, the designer of the detection system would like to detect everything that may be related to an attack. On the other hand, if the detection rules are set too wide, the flow of false positives will make the rule useless. Consequently, security products designers are extra careful when writing rules and they try to reduce false positives. However, this comes at the cost of ignoring data that may be meaningful.

This paper proposes to use baseline techniques to monitor security data. Baseline monitoring is a rather simple technique: at some point in time the current status of the system is recorded and referred to as the baseline status for the system. From that moment on, any deviation from the baseline is reported.

Applying baseline techniques for security data is far from trivial. The major problem is to define the baseline itself. The baseline can not be just the set of alerts that are currently recorded in the SIM repository since the behavior is not systematic and there are no formats, rules or algorithms to describe the data. So, what can be done to describe the data and define the baseline?

The area in computer science that is targeting the problem of modeling and describing huge amounts of data is Data Mining. This paper describes a way to use data mining techniques to represent the baseline of a security system deployment and to analyze new data against this baseline. We call this baseline the 'Behavioral Model' of the system and it includes behavioral groups of alerts, and behavioral patterns. This approach was experimentally implemented as a tool to analyze Snort alerts (Snort is a widely deployed open source NIDS system). The implementation proves that baseline monitoring is invaluable for monitoring security data. Instead of analyzing hundreds of thousands of alerts, our methodology

enabled the administrator to focus on alerts that deviate from the baseline behavioral model. In most cases it reduces the number of 'elements' to monitor by 99.9%.

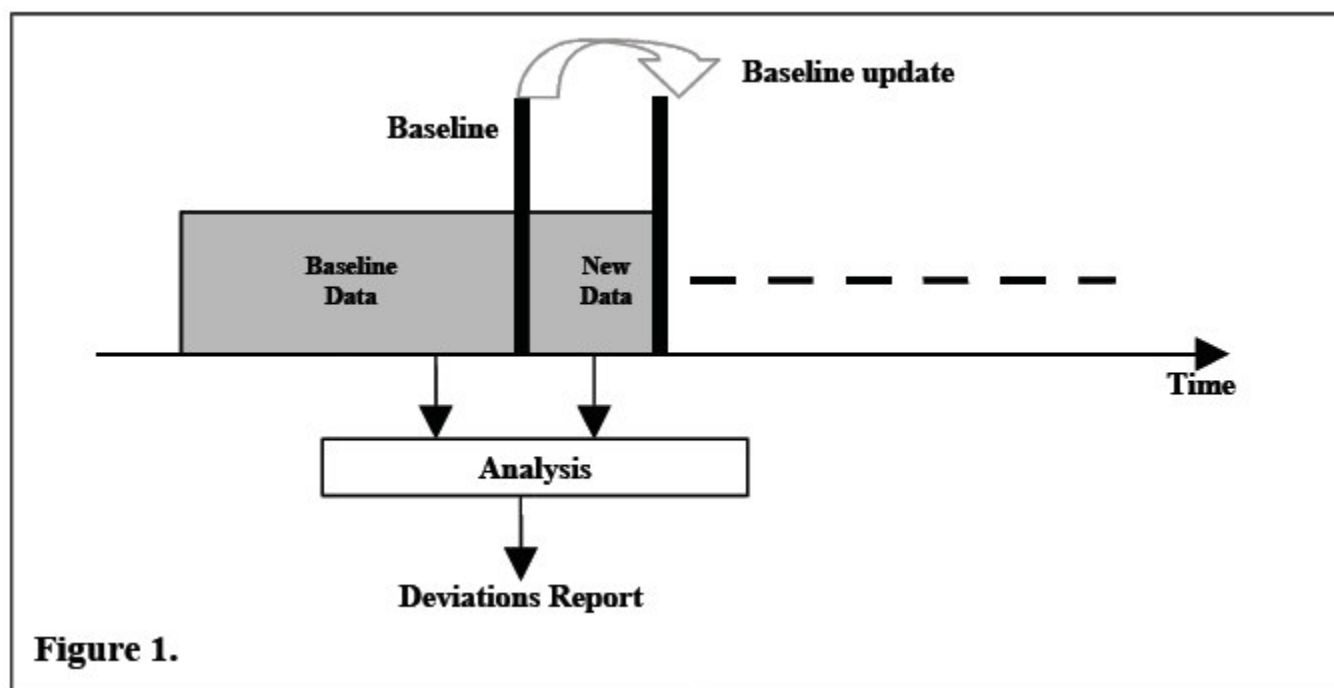
Baseline Monitoring

Baseline monitoring is a simple concept that applies to many areas where there is a need to monitor and analyze large amounts of data. Reviewing the data as a whole and time after time is not effective. Baseline monitoring is useful when analyzing changes in the data rather than analyzing the whole data.

Baseline monitoring consists of the following stages:

1. Building the baseline: collecting any relevant data and building a description of the current status.
2. Analysis: collect new data and compare it to the baseline while alerting any deviations from the baseline.
3. Action: based on the analysis results, any deviations from the baseline must be investigated and justified or fixed.
4. Updating the baseline: moving forward collect new data and update or rebuild the baseline.

The above stages are described in the following Figure:



Baseline monitoring is especially effective:

- When the data contains large percentage of repetitions with or without variations.
- When the data contains patterns and can be formatted so it is possible to compare new data to the baseline data.
- When the interesting analysis result is not semantic but rather differential, in other words, “what is different ...” is the interesting question.

Examples of baseline monitoring are monitoring file system operations, monitoring access to resources, etc. In order to efficiently use baseline monitoring methodology, one must define the baseline in the right way. The baseline is a description of the existing status and it has to reflect that. In the next section we describe how to define the baseline for a security system.

Security System Baseline

In this paper we introduce a methodology for baseline monitoring of security data. The crucial decision is what information should be included in the baseline. One may think of many ways to define this information: total number of alerts, number of alerts per alert type, times and frequency of alerts, behavior of attacked hosts and attacking hosts, etc. The problem is that there is no magic formula that fits our needs of a security data baseline.

It is important to note that if the security data was random, there was no good way to represent it. However, security data is far from random. Data from each security deployment follows certain rules and patterns that although vary from site to site, can be identified and described. Our suggestion for security data baseline consists of two types of entities:

- ***Behavioral groups of alerts***

An alerts group is a set of alerts that are close to one another and represent the same type of activity in variations.

For example, consider a backup software that generates specific types of alerts, or a network synchronization utility that generates the same types of alerts, or an internal application that triggers some alerts in a certain way, etc.

The alerts in alerts groups are not identical, but they have some similarities that are different from group to group and from deployment to deployment. We claim that security data is mostly combined from such groups.

- ***Behavioral patterns***

Although intuitively anyone understands the concept of patterns, this concept is hard to define formally. Roughly speaking, a pattern is a logical statement that defines characteristic behavior of the data.

The following are examples for such patterns:

1. "If the source IP is 10.1.123.2 then the destination IP is either 12.2.143.2 or 12.111.0.232".
2. "Alerts of type X and Y happen only on weekends and always from the same source IP address".
3. "Alerts of type X are triggered only with a specific payload data Y".
4. "Alert X is triggered only after alert Y is triggered".

The behavioral groups are an attempt to generalize, consolidate and classify the data, while Behavioral patterns are an attempt to isolate specific behaviors that are characteristic in the system.

Building the Baseline

Behavioral patterns are not easy to identify as one may think. It is a complicated task to try and identify what is characteristic to the system and what is merely noise. For example, consider an alert X and

denote by Y the set of all the source addresses that this alert is triggered from in the baseline data. The logical statement “Alert X is triggered from source IP’s Y” is true, but, does it represent a characteristic behavior of the system? Probably not!

Calculating the real patterns is not trivial and involves some heuristics and randomized algorithms. We are not going to discuss the patterns identification algorithms in this paper mainly since these algorithms are not generic and are specific to the exact type of pattern.

The behavioral groups identification is based on data mining technology known as Clustering. The purpose of clustering is to identify groups within a give set of data objects. Our approach to generating the behavioral groups is as follows:

1. First, identify groups, or clusters, within the set of values of each parameter. For example, consider ‘destination IP’ as a parameter and the set of all the possible values of this parameter in the baseline data. In order to classify this set, one needs to define a ‘distance’ for every two values in the set. One can think of many distance functions such as
 - a. based on syntactic similarity of the IP address, or
 - b. based on relations to other parameters, for example, if two IP addresses appear in events with similar alerts, then these IP addresses are close to one another, etc.

Using one or more distance functions, the set of values is classified into clusters of similar parameters values. Using the right distance functions results in interesting classifications of the parameters. For example, one can identify groups of local hosts with the same functionality, or groups of payloads that are related to the same application, etc.

2. Then, once we created classifications of the various parameters, one can define a distance function between events (based on the similarities of their parameters) and follow the same clustering algorithm to create groups of alerts. These are the behavioral groups considered for the baseline.

Deviations Analysis

Once a baseline is defined, one can compare new data to the baseline data and report deviations from the baseline. The suggested analysis is performed in two levels:

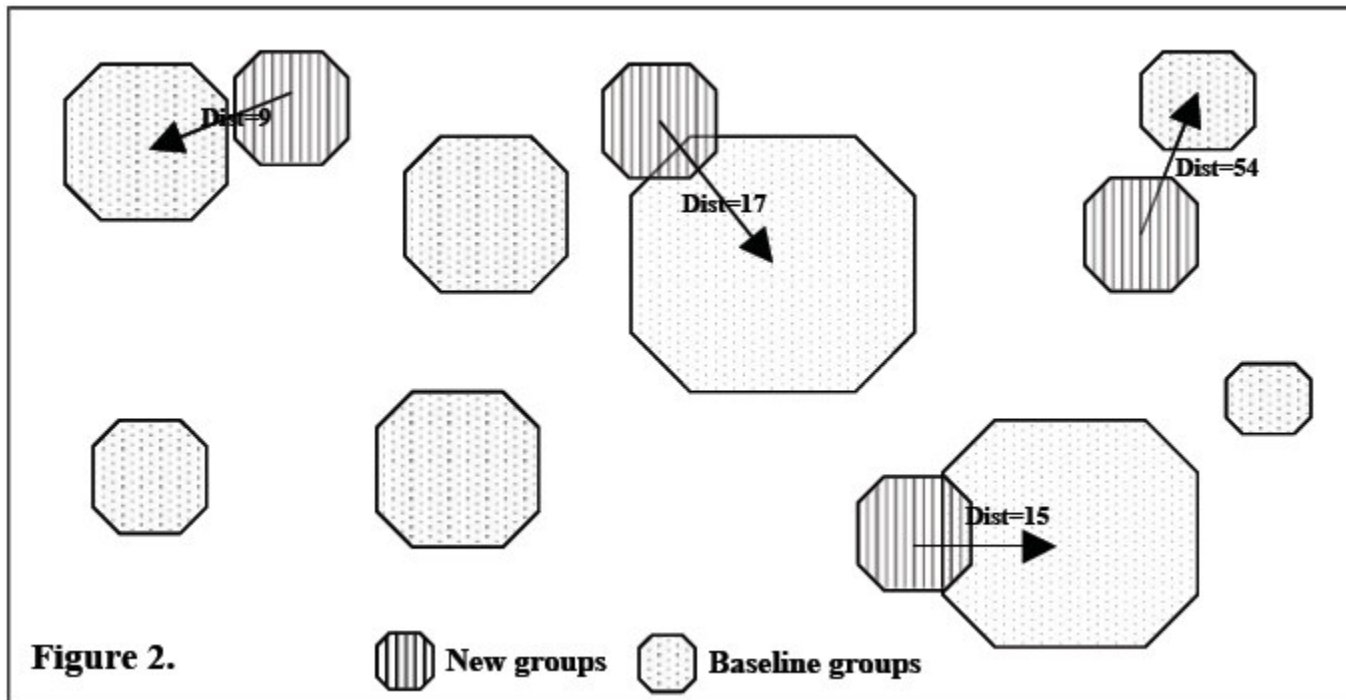
1. Behavioral groups comparison: the purpose of this step is mainly to identify new data that is expected and can be categorized as part of known behavioral groups. This is done by first grouping the new data in a similar way done for the baseline data, and then, for each group in the new data, search for matching groups in the baseline.

It is possible to ‘measure’ how close a new group is to existing groups by applying the same distance function created between alerts. Using this technique each group in the new data can get a ‘score’ based on how close it is to the baseline data. This score can be used to prioritize groups of alerts and highlight groups of alerts that are different from the baseline and therefore require further investigation.

Figure 2 below describes this process. For each of the new groups the best matching model group is identified (indicated by an arrow in the diagram), and, the distance of the new groups from its best matching group (indicated in the diagram by $\text{dist}=\text{xxx}$) which may be interpreted as the ‘deviation score’ for this group; the larger the distance is, the higher the deviation score is.

2. Behavioral patterns violations: another interesting analysis is to identify specific alerts that violate

one or more of the behavioral patterns that were identified in the baseline. This analysis highlights specific alerts that represent behavior that deviates from known characteristic behavior and may need extra attention.



Experiments

To test the baseline analysis theory we used a tool called SFS. This tool was developed by Securimine Software Inc., and is targeted for baseline monitoring and analysis of Snort alerts.

In our experiments with real life data, we built a baseline based on 30 consecutive days of data from our test deployment. We then ran an analysis of new alerts on a daily basis. For each day, we had about 10,000 alerts and the analysis results were:

- The number of groups identified in the new data was ~200.
- The number of groups which got a high score indicating deviation from the model were ~10.
- The amount of violated patterns was minimal (~2-3) per day.

The above results show that on a day to day basis, one can reduce the amount of ‘objects’ to view from 10,000 to ~10. This is the expected behavior in normal environments where ‘nothing new’; is happening.

On the other hand, when we tried to launch attacks against our system these attacks were highlighted as deviations from the baseline and we had to invest no time trying to identify them in the day to day data.

Summary

In this paper we outlined a baseline monitoring and analysis system for security data. The experiments

we did with real life data indicate that this technique is useful in the following areas:

- Quick and automatic identification of real (or new) attacks.
- Providing tools to compare new behaviors to known behaviors and emphasizing the differences thus providing initial investigation tool for suspicious data.

More than that, this technique will remove the conflict rules developers are facing. It will enable the rules to be wider and perhaps trigger more alerts, however, the baseline analysis level will automatically discard all the alerts that are redundant and represent normal behavior, while highlighting the important ones.

All rights reserved
Securimine Software Inc. (www.securimine.com)