Intrusion Detection Systems with Correlation Capabilities

Daniel Johansson
danjo133@student.liu.se

Pär Andersson
paran213@student.liu.se

Abstract

Alert correlation in network intrusion detection systems is a method of correlating generated alerts to both reduce the information that network administrators have to read, and help detect patterns that would be missed without the correlation. In this paper we introduce the concepts of network intrusion detection systems and the possible usages of correlation. We cover some of the current research in the area and present some of the more popular available applications. We have also conducted experimental research with our own basic correlation system. The result of the experiments shows that correlation can both reduce the amount of information and help detect attacks.

1. Introduction

One of the main problems with Network Intrusion Detection Systems(NIDS) is the quantity of alerts that are generated. In highly loaded networks the amount of reports often overwhelms a human operator. This problem increases when more sensors are added to the system. To remedy this situation alerts and/or events from the sensors can be correlated, so that information concerning the same attack is grouped together and not reported more than once. Correlation can also enable the detection of attacks that are undetectable if data from only one sensor is considered. In this paper we will discuss current research regarding correlation in NIDS. We will also set up a NIDS installation with more than one sensor and demonstrate how correlation works.

2. Background

2.1. Why NIDS

With the increase in importance to keep more information available over computer networks comes the need for Network Intrusion Detection Systems(NIDS), this is because there will always be faults in systems, both through bugs in software and in system setups. Because of this, a system is never completely safe. In the real world this has been known since the dawn of time and guards have been hired to protect important things, in software security the Intrusion Detection Systems(IDS) have taken their role. One of the major problems with NIDS is how to get useful information from it, the sheer amount of information produced by the systems can often overwhelm a human. This is where correlation of events in order to find attacks comes as a solution and this is what this report will discuss. Another problem comes when deploying NIDS and that is: many organizations underestimate the amount of traffic generated by a NIDS on a large scale network. They go from testing to deployment without a pilot phase, and as a result they get poorly maintained systems that don't work as intended.[15]

In a complete IDS you protect not only the network but also the hosts and servers on the private network. A Host Based Intrusion Detection system run on a single host and can monitor for example the local file system, running processes, memory usage etc. Correlation can be efficiently used in a HIDS as well as in a NIDS, or data between HIDS and NIDS can be correlated. However we will only discuss NIDS.

2.2. Anomaly and Misuse Detection

The goal of a NIDS is to listen to traffic on the network and detect malicious behavior based on this generate alerts. There are two common ways of doing this. The first method try to create a model of what is normal usage of the network and then events that deviate from the specified behavior can be detected, this is known as anomaly detection. One obvious benefit of anomaly detection is that previously unknown attacks can be detected. Unfortunately anomaly detection system often have a high rate of false positives, i.e. normal traffic being classified as attacks.

The other method is misuse detection. This method use rules that specify known bad behavior, and creates alerts when events match a rule. This kind of system is easier to implement, gives a low amount of false positives, but can only detect things that has been specified in the rules. There is always a delay between for example a new type of network attack is discovered and new rules can be written

and integrated into systems, attacks missed because of this is known as false negatives.

2.3. Correlation

The most basic setup of a NIDS is to use one sensor that then e-mails its generated alerts to the network administrator. In bigger networks there is often a need for more sensors to cover the entire network. If all these sensors use the basic setup mentioned before this results in a lot of e-mail messages for the administrator.

A common solution is that all the sensors send their generated alerts to a central server where they can be stored, often using a database. The central server can then send status reports to the administrators, but this will still generate to much information.

This is where alert correlation can help. By correlating the alerts from the different sensor the alerts can be merged together to reduce the information. The correlation can also help detect attacks that would otherwise be missed. Sensors distributed across the network can catch attacks against different subnets or when you want to catch attacks against several nodes. Alerts from sensors placed at different depth in the network can be used to find attacks that might use TTL/fragmentation attacks.[16]

2.4. Related work

There exist lots of research in the area of Intrusion Detection System and Network Intrusion Detection Systems. However there are not so much that specifically target the area of Alert Correlation, and the research that exist is mainly from 2001-2003. One notable exception is the research done by Baleur et. al [18] that have created a sophisticated system that performs alert correlation as a process with many phases. One other relevant paper is by Cuppens and Miège [8] that cover many different ways of performing correlation.

Much research is also about combining NIDS system with traditional host based systems, to improve security.[6][14]

Many of the researchers[8][14][6] use the Intrusion Detection Message Exchange Format (IDMEF)[10] which is a protocol for sending Intrusion Detection Messages that is currently being standardized by the Internet Engineering Task Force (IETF)

Decentralization of the correlation has also been researched. Krügel et. al [13] have proposed a language for specifying distributed attacks. This can be used to ease the writing of pattern matching rules that matches correctly against a greater number of attacks.

3. Available Software

In this section we will discuss some of the NIDS software that exists. We will also cover some tools for security auditing that we will later use in our practical NIDS study.

There exists a number of commercial NIDS applications that have some correlation abilities. For this project we are unable to test any of them as we have no funding. Therefore we are dependent on either reading others reviews or reading the software developers information. We have not found a good source of reviews, and the software companies of course only highlight the good features of their products. Based on this we have chosen to only investigate free NIDS software and not any commercial software.

3.1. Intrusion Detection

3.1.1. Snort

"Snort is the de facto standard for intrusion prevention. Snort® is an open source network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods"[17]

Snort is a very popular NIDS software developed by a company named SOURCEfire and released as free software under the GNU General Public License(GPL).[3] The software is available on most platforms, including Microsoft Windows, Mac OSX, Linux. SOURCEfire provides official updates to the rule set using a subscription model where paying subscribers get the new rules first, later registered users and last unregistered users. There also exist community created rules that are available free for all.

Since Snort is rule based it is mainly a misuse detection system, but it also has some anomaly detection capabilities. Snort supports logging events to either log files or a database.

3.1.2. ACID - Analysis Console for Intrusion Databases

ACID[9] is a free web based program written in PHP that analyzes Snort alerts stored in a database. It features include advanced searching capabilities, displaying of raw alert data, creation of charts and statistics. It also has some basic correlation capabilities. However the project has been discontinued and is no longer developed.

3.1.3. BASE - Basic Analysis and Security Engine

BASE[1] is directly based on code from the ACID project. BASE is much improved, has less bugs, more

features, improved interface and graphics. It is actively maintained and developed by a group of volunteers.

3.1.4. QuIDScor

QuIDScor™is an open source tool for correlating IDS events with vulnerabilities detected by QualysGuard.[4]

QuIDScor is an IDS manager that categorizes alerts from snort into "Validated Alerts", "Unknown Alerts", "Invalidated Alerts", based on vulnerability information gained through QualysGuard when doing your regular security audits. This decreases the amount of data the network manager needs to watch.

3.1.5. IDEA - Intrusion Detection Exchange Architecture

"IDEA is an architecture for implementing a distributed intrusion detection system on a computer network. It provides a way to incorporate many different IDS sensors into an architecture, and have them report to a central IDS server. This server collects, aggregates, and correlates data from the sensors, providing a unified view of network activity. "[11] IDEA has been discontinued since 2003, the web page doesn't mention what it tries to correlate and it only supports MySQL as back-end.

3.2. Security auditing tools

3.2.1. IDSwakeup

IDSwakeup[7] is a software that can be used to test NIDS installations, it tries to mimic several well known attacks to make the NIDS generate false positives.

3.2.2. nmap

Nmap ("Network Mapper")[12] is used for network exploration and/or security auditing. It supports many different methods of scanning for open ports and online hosts. It also have support for trying to identifying what operating system the scanned target is running.

4. Practical research

For this part we set up a test network with three private subnets and one common subnet, with NIDS running on two of the gateways separating the private subnets from the common subnet. We then performed some simple attacks to demonstrate how correlation can be used on the resulting alerts.

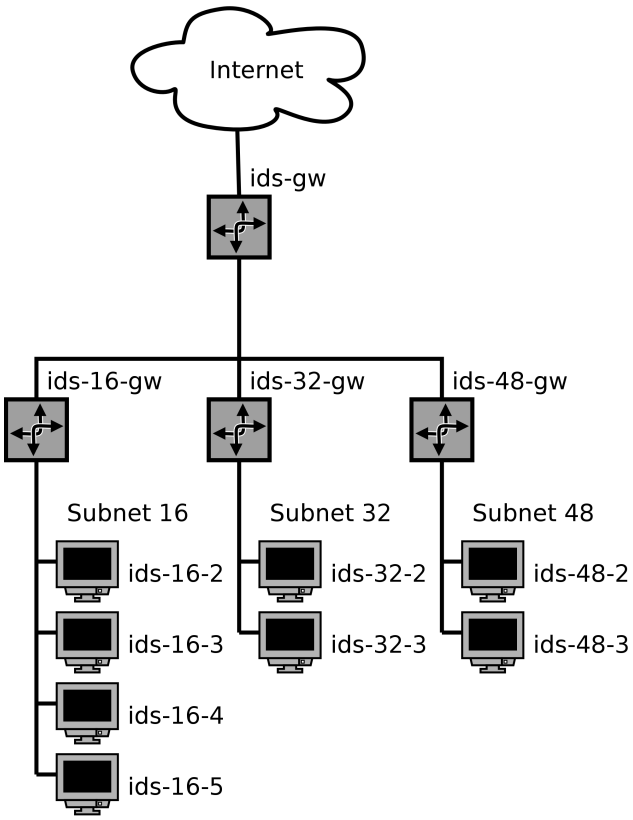

Figure 1. Test system network topology

4.1. User Mode Linux

We have set up our test system using User Mode Linux (UML)[5]. UML is a type of virtual machine that works by running instances of the Linux kernel as user-space processes on a host Linux system. Support for virtual networking is also provided.

4.2. Operating system

All the UML instances used run the operating system Debian GNU/Linux version 3.1[2]. Debian is a free operating system developed by a group of individuals.

4.3. Test system

Our test system uses 12 UML instances, where eight are ordinary hosts and four are routers. The UML instances are connected by a virtual network, as shown in figure 1.

- *ids-gw*: One gateway that connects the virtual network to the rest of the world. This node also runs the centralized database server used for storing alerts and doing correlation.

- *ids-16-gw, ids-32-gw, ids-48-gw*: Three gateway nodes, that each has one interface connected to a virtual internal network, and one interface connected to the other gateways by the virtual interconnection network.
- *ids-16-2, ids-16-3, ids-16-4, ids-16-5*: Four nodes on the first internal network (Subnet 16), connected to ids-16-gw.
- *ids-32-2, ids-32-3*: Two nodes on the second internal network (Subnet 32), connected to ids-32-gw.
- *ids-48-2, ids-48-3*: Two nodes on the third internal network (Subnet 48), connected to ids-48-gw.

4.3.1. NIDS Sensors

We use Snort as the NIDS for our tests. The Version of Snort used is 2.3.2 as available in Debian 3.1, but with manually updated rule set to the latest publicly available.

Snort was installed on two of the gateways, ids-16-gw and ids-32-gw listening on the interface connected to the internal networks. We use Snort version 2.3.2 and it is configured to send the generated alerts over the network to be stored in a database running on ids-gw.

4.3.2. Correlation Server

Alerts from the Snort instances are stored in a PostgreSQL database running on ids-gw. On the gateway we also run a web server with BASE and our correlation software installed. While IDEA incorporates the concepts that we want to illustrate in this report, it had too many drawbacks that would have taken us too much time to sort out given the time-frame of the project.

4.4. Attacks

One attack that is easy to detect using correlation is the "ping sweep", especially if the targeted addresses are in subnets covered by different sensors, this will be the first attack that we will perform and try to detect. This attack will be performed using Nmap.

Another attack that will be a little more difficult will be a "sparse port scan", i.e. a port scan that touches several nodes but does so slowly, thus trying to make the NIDS believe that there is no connection between the attacks.

We will also use the program IDSwakeup, as this will hopefully result in lots of alerts, with the same pattern each time. By running IDSwakeup against several nodes we will simulate for example a worm trying to infect those hosts, or a malicious user trying the same series of attacks against them. Due to the number of alerts that each execution of IDSwakeup will produce from the NIDS this will be very good to illustrate the information reduction that comes from correlation.

We will also use nmap to port scan various hosts, the results from this should be similar as when attacking with IDSwakeup. This should however result in far less alerts.

Other attacks that might illustrate the need for distributed NIDS are DoS and TTL based attacks, however this is not the focus of this report so these attacks will not be performed. DoS attacks might be easier for the NIDS systems to handle, since no single sensor have to take the burden of the entire data stream. TTL-based attacks make the NIDS create a different packet than the end node by sending packets with different TTL.[16] Having two sensors on a serial line would then see that traffic have changed, this particular attack can also be avoided by using the snort frag3 preprocessor, but attacks based on the same principle might still be valid.

4.5. Alert Correlation

BASE has a nice user interface and is useful to see for example what alerts are generated. However it has very limited correlation abilities, it is only possible to group events, eg. you can us it to watch unique alerts and see that an alert has been caught by several sensors. We wanted to be able to detect the attacks described. We first started to extend BASE with our own correlation functions, mainly because it already had a suitable user interface to present the data. However we realized that the code is quite complex, and not very well documented, and decided that extending BASE would not be possible considering this projects time frame. Instead of extending BASE we created our own basic correlation software.

Our correlation software is implemented in PHP, does not have a nice user interface but is good enough to serve as a proof of concept for using alert correlation. The program is not run in real time, and does not take any input parameters. It connects to the database on the server, reads the data of all alerts stored there by the Snort sensors and then performs different types of correlation on the alerts. The results are presented as a web page that list the resulting alerts from correlation as well as the original Snort alerts.

The data our program looks at when performing correlation comes from the alerts generated by Snort and has the following interesting fields: The ID of the generating sensor(*sid*), the name of the Snort signature that matched and caused the alert to be generated(*sig_name*), the source address of the matching packet(*ip_src*), the destination address of the matching packet(*ip_dst*) and the timestamp of when the alert was generated (*timestamp*).

The program have three different ways to correlate the alerts:

- *Burst correlation*: This mode simply correlates the alerts from all sensors based on *timestamp* and detects bursts of alerts. As soon as one alert is found a new burst is started and subsequent alerts are then added into this burst. A burst is ended when there is a specified time interval with no new alerts. Using this distributed attacks can be detected. Of course this method will be very infeasible on a larger scale network where alerts are generated continuously. This method could scale much better if the type of the alerts were also used, instead of detecting bursts of alerts it could for example detect bursts of ping scan alerts.

- *Pattern correlation*: Here we look at alerts from all sensors and then try to find similar patterns of alerts that have traversed different sensors and creates an alert-sequence. This is done by searching through the alerts ordered by *timestamp* and group possible attacks together based on common sequences with the same *sig_name*. This type of correlation will catch both attacks against distributed nodes separated by time and attacks such as those indicating a worm.

- *Ping Sweep*: This is specifically created for finding ping sweeps. It looks at alert data from all sensors and detects the "ICMP PING" *sig_name*, if many of these have the same *ip_src*, and happen within a short time of each other they are grouped together and classified as a ping sweep.

4.6. Results

We will now discuss the results using our software when performing the described attacks. A summary of how many Snort alerts that was generated from each attack, and how many alerts our correlators generated from these is shown in table 1.

Attack	Alerts	Bursts	Patterns	Ping sweeps
Ping Sweep	12	1	1	1
IDSwakeup	150	1	10	-
Port scan	36	6	1	-

Table 1. Correlation software results

We first performed a simple ping sweep using nmap scanning all IP addresses in subnet 16 and 32. The scanned subnets have two and four nodes respectively. The ping sweep resulted in two alerts per scanned node so the total number of generated alerts was 12. All those alerts were grouped together by our ping sweep detector as being one attack. Our other two correlation methods also produced interesting results, the burst correlator reported the entire ping sweep as one burst as expected. The pattern matcher found one pattern containing four alerts, this is because one sensor reported eight alerts while the other reported four so the smallest common sequence was detected.

Our second test was to simulate an attacker simultaneously attacking two different nodes in the network. These attacks were performed using IDSwakeup, and produced a total of 150 sensor alerts. Our program showed this as one burst, and 10 patterns. The reason that the entire IDSwakeup attack was not correlated into one pattern is likely because of delays in the network and the operating systems TCP stacks so that some of the packets pass through the sensors in different order. This problem could be solved by using more advanced pattern matching rules. Also IDSwakeup is not a very realistic attack simulation due to the number of attacks that it perform, ordinary attacks will result in fewer and shorter patterns. We still consider this a good result for illustrating correlation, as reviewing 10 patterns is much better than 150 alerts.

After this we did port scan attacks using nmap. The correlation of alerts produced when executing the scans worked very well. Each scan results in an average of six alerts, which our software correlated into one burst. Our pattern correlator also detected the scans as using the same patterns and grouped them together. When scanning the same nodes that was targeted in the previous ping sweep, one at a time, this generated 36 alerts correlated into six bursts and one pattern.

5. Conclusions

In this paper we have covered the usages of alert correlation in intrusion detection systems and why it is necessary to help network administrators handle a high number of alerts and to help them detect otherwise unnoticed attacks. There have been much research in the area, especially around 2002-2003. Unfortunately much research is about the broader subject of network intrusion detection, but we have not found many studies that focus especially on the alert correlation of alerts from distributed sensors.

There is not so much free software available to do advanced correlation. Some projects do exist but they don't seem to be very mature yet.

Using our test system we have seen that implementing some simple correlating methods in a program is easy. Using our program it was also easy to illustrate the benefits of using correlation, both for information reduction and attack identification. However, our simple software can only detect a small number of attacks and therefore more advanced correlation algorithms are necessary to be useful in a real life scenario.

References

[1] *Basic Analysis and Security Engine*. http://secureideas.sourceforge.net/.

[2] *Debian GNU/Linux 3.1*. The Debian Project, http://www.debian.org/News/2005/20050606.

[3] *GNU General Public License*. Free Software Foundation, Inc., http://www.gnu.org/licenses/gpl.html.

[4] *Qualys IDS Correlation Daemon*. Qualys, Inc, http://quidscor.sourceforge.net/.

[5] *User-mode Linux Kernel*. http://user-mode-linux.sourceforge.net/.

[6] D. Andersson, M. Fong, and A. Valdes. Heterogeneous sensor correlation: A case study of live traffic analysis. In *Third Ann. IEEE Information Assurance Workshop*, June 2002.

[7] S. Aubert. *IDSwakeup*. http://www.hsc.fr/ressources/outils/idswakeup/.

[8] F. Cuppens and A. Miège. Alert correlation in a cooperative intrusion detection framework. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 202, Washington, DC, USA, 2002. IEEE Computer Society.

[9] R. Danyliw. *Analysis Console for Intrusion Databases*. http://acidlab.sourceforge.net/.

[10] H. Debar, D. Curry, and B. Feinstein. *Intrusion Detection Exchange Format, Internet-Draft*. http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-16.txt, March 2006.

[11] I. Duffy. *Intrusion Detection Exchange Architetcure*. http://idea-arch.sourceforge.net/.

[12] Fyodor. *Nmap, Network Mapper*. http://www.insecure.org/nmap/.

[13] C. Krügel, T. Toth, and C. Kerer. Decentralized event correlation for intrusion detection. In *ICISC '01: Proceedings of the 4th International Conference Seoul on Information Security and Cryptology*, pages 114–131, London, UK, 2002. Springer-Verlag.

[14] P. Porras, M. Fong, and A. Valdes. A mission impact based approach to infosec alarm correlation. In *In Proc. of RAID*, pages 95–114, 2002.

[15] G. Shipley. *Event Correlation*. Network Computing, http://www.networkcomputing.com/1401/1401f25.html.

[16] S. Siddharth. Evading nids. *http://www.securityfocus.com/infocus/1852*, dec 2005.

[17] SOURCEfire. *Snort® network intrusion prevention and detection system*. http://www.snort.org.

[18] F. Valeur. A comprehensive approach to intrusion detection alert correlation. *IEEE Trans. Dependable Secur. Comput.*, 1(3):146–169, 2004. Member-Giovanni Vigna and Member-Christopher Kruegel and Fellow-Richard A. Kemmerer.