Expansion of the SURFnet Intrusion Detection System

P.J. Siekerman en R. Buijs

02-02-2007

Universiteit van Amsterdam

SURFnet

Abstract

Over a period of one month we conducted research into the possibilities for expansion of the functionality of SURFnet IDS. SURFnet IDS is a distributed sensor-based intrusion detection system which simulates several vulnerable services to learn about malicious traffic on a network. It can detect more than twenty kinds of attacks and it makes information about these attacks accessible in a clear web-interface.

In this report we first looked at the current state of SURFnet IDS. Next we reviewed various intrusion detection applications and decided two might be suited for use in SURFnet IDS, namely Prelude and Snort. We then investigated how these two can be integrated into the current system and concluded that Prelude was not as promising as it initially looked and that Snort was the best candidate for actual implementation. A minor change to SURFnet IDS has to be made in order to read the Snort data, but after this change the amount and variety of malicious traffic which is detected, will increase dramatically.

Acknowledgments

During our project we were actively in contact with the project leader Rogier Spoor and the current SURFnet IDS developers Kees Trippelvitz and Jan van Lith. We would like to thank them for their dedication to our project. Their feedback and tips were very helpful to us.

Contents

Chapter 1

Introduction

Hackers, crackers, viruses, worms, exploits, vulnerabilities, trojans, spyware, adware ... the internet is a dangerous place. And yet we can hardly imagine life without it anymore. So how do we survive our daily life on the web? We secure our systems using firewalls, virusscanners and other tools. In this category of tools we also find the Intrusion Detection Systems (IDS).

Intrusion detection systems serve one primary purpose: gathering information about attacks against a system to strengthen the defense of the system. To defeat your enemy, you must know your enemy.

Various types of IDS applications exist. Host-based intrusion detection systems (HIDS) guard or simulate a host and analyze and report any attacks directed towards this host. Network intrusion detection systems (NIDS) are used to analyze network traffic directed to various hosts. While HIDS applications often actively simulate services or complete hosts, NIDS applications usually work passively by only inspecting traffic without actively modifying it.

When people speak of honeypots, they are usually referring to HIDS applications. There is a difference between a low-interaction honeypot and a high-interaction honeypot. A low-interaction honeypot only simulates services at the most basic level necessary to fool attackers into thinking they are dealing with a real system. A high-interaction honeypot is a real system running real services, often in a virtual machine, combined with intrusion detection software to analyze what attacks do to the system.

SURFnet is the organization that connects the various universities and other higher education institutions to the internet. In addition to internet connectivity it offers several other services to its customers. One of those services is a distributed intrusion detection system called SURFnet IDS [1].

In this report we examine the current state of SURFnet IDS and offer suggestions on how the service could be expanded by adding other intrusion detection technology. We start by taking a close look at SURFnet IDS as it is now in chapter 2. In chapter 3 we review various IDS applications to determine which of them offer the functionality and quality required for implementation in SURFnet IDS. Based on this review, we select two promising applications (Snort

and Prelude) and discuss in chapter 4 if and how they could be integrated into SURFnet IDS.

When IDS applications contain an active component which automatically responds to and defends against attacks, they are referred to as Intrusion Prevention Systems (IPS). We discuss why we decided not to consider such systems as possible extensions to SURFnet IDS in section 3.5.

In addition we decided to exclude all closed source and proprietary IDS solutions from our review. There are the obvious ideological and financial aspects of this choice which play a role, but the fact that it is impossible to examine the source of these programs to determine how exactly they work and the fact that it is impossible to modify them to suite the specific SURFnet IDS setting were the main reasons for making this decision.

Previous work

During the last two years, several other System and Network Engineering students have completed research projects in the area we are dealing with in this report. In 2005 a report was written about various IDS applications [2] and the original distributed sensor-based setup for SURFnet IDS was designed [3]. In 2006 the security of SURFnet IDS was reviewed [4] and the addition of Argos was advised [5], which is currently being implemented. We aim to build on these reports and suggest further improvements to SURFnet IDS.

Chapter 2

SURFnet IDS

In this section we'll describe the current SURFnet IDS architecture. We do this to get a clear picture of which functionality is offered by the current system. This overview will make it easier to determine what new applications in the structure would add in terms of functionality.

2.1 Current setup

The current SURFnet IDS is a distributed sensor-based intrusion detection system. The sensors are based on a modified and remastered Knoppix installation running on a USB-stick. When the sensor boots from the USB-stick it contacts the server and forwards all its data to the server using OpenVPN and ethernet bridging. The SURFnet IDS server collects the data from the clients. The server is running a honeypot called Nepenthes, which analyses the traffic to detect malicious data. The results are stored in a PostgreSQL database. The server has a web interface which shows the user the database contents in a user-friendly way. This web-interface shows which vulnerability attacks were detected by Nepenthes on the sensor. The web interface also shows a large variety of useful statistics, for instance about the ranking of the occurrence of attacks using specific vulnerabilities.

2.1.1 Nepenthes

Nepenthes is a low to medium-interaction honeypot [6]. Low-interaction means the honeypot simulates services and vulnerabilities only up to the minimum required level to fool attackers into thinking they are attacking a real system. It does not take any actions after the detection, except for logging and downloading the exploit file. Nepenthes is a modular application which can simulate several vulnerabilities. Because of this, it is easy to expand Nepenthes with new vulnerability modules. The application is being built and maintained by two German students. Nepenthes is able to recognize more than twenty known

Windows vulnerabilities. The application is not designed to detect zero day vulnerabilities.

2.2 Planned additions

Although the setup at the moment of writing this text only includes Nepenthes, an addition to the system is being worked on for future release. This update would include the addition of the program Argos as a second honeypot.

2.2.1 Argos

Argos is a full and secure system emulator designed for use as a honeypot [7] [8]. It is based on Qemu, an open source emulator that uses dynamic translation to achieve a fairly good emulation speed. Argos extends Qemu to enable it to detect remote attempts to compromise the emulated operating system. Using dynamic taint analysis it tracks network data throughout execution and detects any attempts to compromise the system. When an attack is detected the memory footprint of the attack is logged. Argos can be used to detect zero day attacks.

Argos and Nepenthes are linked in such a way that whenever traffic is received, it is initially routed to Nepenthes. As Nepenthes only emulates a limited number of vulnerabilities, there is a good chance that Nepenthes will not be able to classify the traffic. Based on a few criteria, such as how many times suspect traffic has been received from another host, a selection of this traffic is routed to Argos instead of Nepenthes. As Argos has a more flexible way of detecting attacks, by simulating a real system, allowing an attack to be executed on it and analyzing the behaviour of the attack, it is has a better chance of determining whether traffic Nepenthes can not analyse is really an attack or not.

2.3 Opportunities

The SURFnet IDS structure offers a lot of opportunities for expansion. The currently used honeypot Nepenthes covers only a small collection of exploits, and we think there should be a way to make SURFnet IDS detect more malicious traffic. An increase in the diversity of malicious traffic the IDS can detect and classify would make the IDS more interesting for customers.

Chapter 3

IDS software

In this chapter we will scrutinize several IDS programs to determine which of them would be able to add valuable features to the current SURFnet IDS setup. We will judge the various programs based on several aspects, such as their functionality, their current status, the level of documentation and how well they would integrate into the current situation.

As we described in chapter 2, the primary goal of SURFnet IDS is to supply information about attacks on a customer's network. The intention is to determine the following information for all network traffic the sensor receives:

- Is this traffic an attack?
- Where did it originate from?
- If this is an attack, how does the attack work? Which vulnerability or exploit does it use? What is it trying to achieve?

An IDS application is only useful if it helps us gain at least part of the information listed above. While some IDS applications might therefore not be valuable due to a lack of functionality, there are also applications that offer too much functionality. This extreme is found in the area of the Intrusion Prevention System (IPS) applications. These programs include both information gathering and automated active responses. We will cover IPS software further in section 3.5, where we will also explain why it is not possible to use such software in the current SURFnet IDS setup.

3.1 Filesystem integrity verification

A simple way of detecting when a system has been compromised, is through filesystem integrity verification. This technique creates a footprint of a system in its original state by creating signatures of essential files and gathering various other characteristics such as file access rights and file sizes and storing this footprint in a remote, safe location. Periodically new footprints of the system

are created and compared to the original footprint. Unexplainable differences between these footprints serve as indications of the system being compromised.

3.1.1 Tripwire

Tripwire is an open source host-based Intrusion Detection System (HIDS). Tripwire is used to detect changes in a host's filesystem. As most other filesystem integrity checkers, it works by making a baseline footprint of an original system and periodically comparing this footprint with footprints of the system as time progresses. These footprints consist of signatures of the essential files on the host, created using mainstream hashing techniques such as MD5 [9].

As we indicated before, SURFnet IDS is used to determine whether specific incoming traffic is an attack or not and to gather as much information about the attack as possible. Tripwire is only useful to determine whether or not an attack has occurred between the creation of two footprints. It has no way to determine any further details about the attack, such as when it occurred, where it originated from or how it worked. The basic determination whether an attack occurred or not, which is the only benefit of using Tripwire, is already offered by Argos (Section 2.2.1). Therefore adding Tripwire to the SURFnet IDS setup would not add any useful features.

3.1.2 Samhain

Samhain is an open source program which performs a service similar to Tripwire [10]. It creates a baseline footprint of a filesystem and compares newer footprints with this baseline to detect unwanted system modifications. It is possible to monitor a large number of clients with a single server by running client software on the various hosts and saving the footprints on the main server. Samhain can be configured to work as part of the Prelude framework (Section 3.4).

As indicated in our evaluation of Tripwire, a filesystem footprinting system would not add any significant new features to SURFnet IDS. Therefore we do not recommend using this program.

3.1.3 AIDE

AIDE is another open source filesystem integrity verification program [11] [12]. It is almost identical to Tripwire as it was specifically intended to offer a similar service. Because this program is very similar to Tripwire, the same analysis applies. Please refer to section 3.1.1 for more information.

3.2 Low-interaction honeypots

In section 2.1.1 we discussed the honeypot Nepenthes. Nepenthes is usually classified as a low-interaction honeypot. In contrast to high-interaction honeypots, which generally offer a completely functional system for an attacker to infect,

low-interaction honeypots only emulate the bare necessities of specific services needed to fool an attacker into believing he is dealing with a real service or system.

3.2.1 Honeyd

Honeyd is a rather nice honeypot. It serves multiple goals. First of all, it can be used to simulate a host with a vulnerable operating system or vulnerable services. Various services are available, such as FTP, HTTP, telnet and SMTP. Configuring honeyd to simulate such systems is as simple as editing a few configuration files. This is similar to the functionality most other honeypots supply as well. It can be used to record and analyze attacks on the dummy hosts and services.

The beautiful part of honeyd is its ability to simulate entire networks of hosts with a complete network structure and appropriate latency times and packet loss. It is possible to simulate quite complicated network structures, which will even show up correctly if the attacker runs a traceroute. When used in conjunction with the program arpd, it is able to claim all unused ip-addresses in a specific range and fill those with virtual dummy hosts which can serve to distract an attacker from the actual system.

Documentation

Honeyd documentation can be found at several locations [13] [14] [15]. There is quite a bit of documentation on the web about honeyd and it is a very well known program. Therefore the chance of finding a solution to any problems that might be encountered during installation or operation is relatively good. Unfortunately there do not seem to be many recent publications.

Status

Honeyd's developer is Niels Provos. He maintains the package single-handedly. Luckily various other people have submitted configuration files to simulate different services. The last major update (1.5) was in february 2006. Unfortunately there haven't been any big updates to the program for almost a year.

Conclusion

Researching honeyd initially resulted in a rather optimistic view of the program. There is a significant amount of information about the program and many people know it and appear to have used it at some point. Unfortunately further investigation showed a somewhat less optimistic picture. Development of the program appears to have paused or maybe even stopped. There have been no serious updates for almost a year. Although documentation and articles are in no short supply, none of it is recent.

The most interesting aspects of honeyd are its abilities to simulate large networks and claim all unused ip-addresses. Unfortunately neither of these

capabilities is directly applicable in the in the current SURFnet IDS setup, because SURFnet IDS's current goal is to monitor and analyze the traffic sent to a single sensor in a network. This might however be interesting as a part of a future expansion.

It seems that what was once a very promising and interesting program has since then collapsed and is no longer top of the range. Therefore we do not recommend using it in the SURFnet IDS setup.

3.2.2 Honeytrap

Honeytrap is a low-interaction honeypot that emulates TCP services. It can handle known and unknown attacks.

This honeypot is a very simple product. It does not run any services. The core principle is this: Whenever a request is received for an unused TCP port, honeytrap opens this port and accepts the incoming data. It saves this data for later analysis. When the data-transfer is done, the connection is closed and the port is released again. This way honeytrap is not limited to dealing with known attacks.

Although honeytrap normally does not emulate any services, it is possible to fake an attacker into believing that it does. This is done by defining a default response to requests on specific ports. For instance, it is possible to have honeytrap send a standard message in response to any requests sent to TCP port 135. If the attack includes a command to download a file, honeytrap includes several plugins to allow the program to download these malware files and store them for later analysis.

Another feature is the "mirror mode". This allows all traffic sent to honeytrap to be reflected back to the attacker. This results in the attacker attacking himself, allowing the traffic to be analysed without being at risk of infection.

Documentation

Honeytrap documentation can be found at several locations [16] [17]. Unfortunately the amount of documentation totals to 3 or 4 pages. There is no tutorial, no extensive manual, no faq, no help section. Basically the only way to get familiar with the program, is diving into the source code. Obviously this is not an option for normal users.

Status

Honeytrap's developer is Tillmann Werner. Unfortunately he appears to be the only developer, which results in relatively infrequent updates. Luckily the last official release (0.6.4) was very recent. The SVN repository shows that the developer is still working on the project. Updates have been submitted every two or three weeks during the last few months.

Conclusion

honeytrap was intended as a very simple honeypot, allowing unknown attacks to be processed, without fully simulating services. It does allow for custom responses on various ports, as a sort of "poor man's service emulator" as the developer calls it. These responses can be added by end-users.

Unfortunately it appears that this program is hardly being used by anyone. It seems that apart from the main developer there are no other people contributing to the project. Therefore development goes very slowly and addons are non-existent. There is no honeytrap community whatsoever.

Considering all the above we advise against using this honeypot.

3.3 Snort

Snort is an open source Network Intrusion Detection System (NIDS), capable of performing real-time traffic analysis and packet logging on IP networks [23]. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Snort can even detect evasion attacks.

Snort can run in four modes:

- *Sniffer mode* - reads the packets off of the network and displays them for the user in a continuous stream on the console (screen).
- *Packet Logger mode* - logs the packets to disk.
- *Network Intrusion Detection System (NIDS) mode* - the most complex and configurable configuration, which allows Snort to analyze network traffic for matches against a user-defined rule set and performs several actions based upon what it sees.
- *Inline mode* - obtains packets from iptables instead of from libpcap and then causes iptables to drop or pass packets based on Snort rules that use inline-specific rule types. When used in inline mode, Snort acts as an IPS system (Section 3.5).

The NIDS mode uses rules to detect malicious traffic and sends out alerts when it does [24]. The rules are frequently updated by a development group. The rules are free to download and use, just like the software. Installing a new ruleset only takes a few minutes. There is even an application available which automatically updates the rules [25].

The snort rules are divided in various categories. These categories include: backdoors, bad traffic, ddos, dns, exploits, ftp, imap, multimedia, mysql, p2p, pop, smtp and web.

All these categories together contain over 5000 rules to detect malicious traffic. The user can choose which rules he wants to use. Snort is able to alert the user according to these rules.

Documentation

Snort is a very well-documented program. On the Snort homepage you'll find a guide describing how Snort works and how it should be configured. This documentation is well build, and describes everything necessary. Besides the official documentation, there is a lot of information on the internet about how Snort can be configured and maintained.

Status

Snort is being actively developed, just like the Snort rulesets. In contrast to small programs like honeytrap (Section 3.2.2), Snort is one of the major open source programs. It has a large and active community of users. Program updates are released frequently. The official rulesets are updated often and there is a large group of users which releases custom rules for a variety of things.

Conclusion

Snort is a powerful application with a lot of nice features. Especially the NIDS function is suitable for our situation. Because Snort can detect lots more malicious traffic than Nepenthes can, we think this application deserves a place in SURFnet IDS. We will discuss how this program could be integrated into SURFnet IDS in detail in chapter 4.

3.4 Prelude

Prelude is not an IDS in the classic sense of the word. Instead of being a front-line HIDS or NIDS, it is intended as a framework which can be used to process and combine output from several other intrusion detection systems [18]. Prelude's core is an open source program. However, it is also available as a closed source commercial package which contains a few extra features and improvements [19].

Prelude has to be able to process input from a large number of very diverse programs. This is made possible by the choice to build the design of Prelude on the IDMEF (Intrusion Detection Message Exchange Format) standard, which is a standardized data model for intrusion detection related messages, which could for instance be implemented in XML.

The IDMEF standard was being developed by the Intrusion Detection Working Group of the IETF [20]. Unfortunately this working group ended before an official RFC could be produced. They did however release a final version of the document they were working on which apparently was supposed to become RFC 4765 [21]. Since the driving force behind the development of the standard has disappeared, it is uncertain what the future of the standard will look like.

Although the future of IDMEF is uncertain, there are currently quite a few programs that support it. A good example is the NIDS Snort, which has a plugin which allows the program to output alerts in IDMEF XML format [22].

Prelude is able to collect IDMEF structured information from a multitude of sources. Based on a collection of user-definable policies Prelude is able to combine the various pieces of information it receives into a single response. By combining the input from multiple sources, the number of false alerts should decrease and the accuracy of the system should increase.

If SURFnet IDS is to be extended by including other honeypots, a mechanism has to be created to combine information streams received from the various honeypots and other applications. This could be done using custom made scripts and programs, but a more elegant and robust solution would be the use of a framework such as Prelude. We will discuss if and how Prelude could be integrated into SURFnet IDS in section 4.2.

3.5 IPS software

Intrusion Prevention System (IPS) applications detect malicious traffic, just like an IDS. The difference between an IPS and an IDS is their behaviour. An IDS detects and reports malicious traffic on a certain network device. An IPS does the same, but in addition to an IDS it can also take action after detection. It can block the traffic for example. You could compare an IPS with a smart firewall.

In the current SURFnet IDS setup it would be very hard to implement an IPS, because SURFnet IDS is used by several different companies, with different unique networks, and different policies regarding security and firewall settings. The IPS would have to be setup in a unique way for each customer network, which would require a massive amount of work. The benefit of the current system, is the fact that it can be used quite effectively in its standard setup, without the need for a lot of configuration.

Though IPS applications are very interesting, they are beyond the scope of the current SURFnet IDS. If SURFnet were to run an IPS service, it would in effect end up being system administrator for a number of companies. Assuming such a role would create all sorts of new responsibilities and problems, such as users being upset when data is blocked incorrectly.

The current setup, which only gathers information and presents this information in an accessible format, is preferable. Because of the relative simplicity of this setup, one configuration can be used for all clients, which dramatically simplifies scaling the services. It also makes sure the responsibility for securing a network remains with the client's administrators. SURFnet only supplies information, the client's administrators are the ones who have to decide how to act on this information.

3.6 Conclusion

In this chapter we covered several programs, ranging from filesystem integrity checkers such as Tripwire, Samhain and Aide, to low-interaction honeypots such as honeyd and honeytrap. We looked at the IDS framework Prelude and the

NIDS Snort.

We concluded that filesystem integrity verification is not suitable for inclusion in SURFnet IDS and that the low-interaction honeypots honeyd and honeytrap were too outdated to be useful. Prelude and Snort however turned out to be very promising programs. Snort allows the detection of a large number of attacks simply by analyzing network traffic. Prelude offers a way to combine data from multiple IDS programs, such as Snort and Nepenthes. In the following chapter we will try to determine if and how Prelude and Snort could be integrated into the current SURFnet IDS setup.

Chapter 4

Implementation

In chapter 3 we looked at various IDS and honeypot applications. We concluded that although there are many such applications, only a few of them are actively being developed, are well documented and could add some unique functionality to SURFnet IDS. In this chapter we take a closer look at the two most interesting applications, Snort and Prelude, to determine if and how they can be added to SURFnet IDS.

4.1 Snort

As described in section 3.3, Snort is a valuable application because it can detect a lot of malicious traffic. Snort can run in four modes. One of them is intrusion detection mode, which is the one we choose to use for SURFnet IDS. Running in IDS mode, Snort will use its frequently updated rulesets to analyze the traffic it receives. If it detects malicious traffic it will log this and create an alert.

Snort can run next to Nepenthes without any modifications because Snort only sniffs the traffic, and does not need to respond to it like Nepenthes. So we suggest running Snort on the same machine as Nepenthes.

In consultation with the current SURFnet IDS developers we decided what information we would like to log. This includes:

- Timestamp
- Description of malicious traffic
- Source IP
- Destination IP

Since we only need the data described above, we can run Snort in the "Fast Alert" mode, which means it logs only a minimal amount of data. This mode does for example not log the complete datastream, but since we do not need this, it will save us space.

Snort can log in several ways: it can log to screen, log-file or database. Snort supports multiple databases, including PostgreSQL, which is the one SURFnet IDS already uses to store the data gathered by Nepenthes. Since the current SURFnet IDS developers would like to have the Snort logs in the same database as the Nepenthes database, we suggest using Snort's built-in PostgreSQL functionality.

Since Snort uses its own way of logging, it uses a few tables to log the data to the database. These tables differ from the tables used by Nepenthes, therefore the web interface of SURFnet IDS will have to be slightly modified in order to read the data from the appropriate Snort tables. This modification will include adding some simple SQL queries. The final situation will contain a web-interface similar to the current situation, with the Snort alerts combined with the Nepenthes alerts. This way SURFnet IDS will detect and report a lot more malicious traffic.

4.1.1 False-positives

Nepenthes does not produce any false-positives at all. If Nepenthes reports that an attack has occurred, it is certain that it truly was an attack. Unfortunately this certainty comes at a significant cost, which is the small number of attacks it is able to recognize. Nepenthes is limited to a small range of attacks, but is 100% accurate in that small range.

On the other hand, Snort allows for a much larger number of attacks to be recognized. It is based on a collection of customizable rules. This ruleset is updated frequently and includes rules to detect an enormously diverse range of attacks. While Snort offers a much broader functionality, this comes at a price. Snort has a reputation for creating a significant number of false positives.

As SURFnet deals with a large number of customers which have different skill-levels, some of their customers will not be able to determine which of these alerts are false and which are real. This could potentially lead to a lot of confusion and unnecessary panic.

It is possible to reduce this problem. The crucial factor here is which Snort rules are being used. Snort offers an enormous number of rules, but the choice of which rules to use is completely up to the person running the program. It should be possible to greatly reduce the number of false-positives Snort generates by selecting a very conservative collection of rules aimed at specific exploits instead of the default package.

4.1.2 Maintenance

Snort has another drawback related to the previous problem of false positives. Although the initial problem can be mitigated by carefully selecting which rules to use, the work does not stop there. Snort rules are updated very frequently, which makes keeping Snort up to date a process which requires frequent attention. Each time new rules are released, these have to be checked to determine whether or not they are suited for implementation in SURFnet IDS.

This makes including Snort a choice with a high maintenance cost. Nonetheless we believe this is a worthwhile investment of time and resources. We are convinced that vastly increasing the ability of SURFnet IDS to detect and report attacks outweighs the costs.

4.2 Prelude

In section 3.4 we looked at Prelude and determined that it might be a promising application. It supplies a framework which can be used to combine data gathered by a number of sensors. Prelude uses the IDMEF standard to communicate with applications such as Snort, Nepenthes and Nessus. It is possible to write Prelude policies which define how the data from the various sensors should be combined into one output.

Prelude offers a web-based interface called Prewikka. If Prelude were to be used, the most logical way to use it would be as a replacement of the current web-interface. Unfortunately Prewikka lacks several options which the current interface does include, such as the ability to display specific statistical data and the possibility of allowing various customers to log in and only see the data related to their own sensor. It might be possible to modify the Prewikka source to add such features, but this would require too much time and effort for it to be worth the switch.

A further drawback of Prewikka is the fact that this web-interface is rather slow, making it somewhat tedious to use. As SURFnet IDS is not just being used in a development environment, but also in production environments by a large number of customers, having a slow interface would have a significant negative effect on the popularity of the system.

Based on these observations, we think Prelude is not optimally suited for use in SURFnet IDS. Although the principle of using a standard framework to combine input from different intrusion detection systems is worth considering and Prelude supports an impressive number of programs as sensor, this does not trump the negative aspects, such as the limited options and the general slowness of the system.

Chapter 5

Conclusion

When we initiated this project, our main goal was to make SURFnet IDS detect more diverse malicious traffic. During our research we reviewed the most important of the currently available intrusion detection systems. We concluded most of these products aren't suitable for use at SURFnet IDS, and in some cases useless in any situation. Two products initially seemed promising as they appeared to fit our needs very well: Snort and Prelude. Unfortunately a closer inspection of Prelude made us decide against using this application.

Snort however is more promising. It allows detection of a lot more varied malicious traffic, which is exactly what we hoped to achieve. This application is actively being developed and maintained. Its rule based nature makes the program very flexible. Snort is fairly easy to implement in the current SURFnet IDS setup by running it next to Nepenthes and making use of its PostgreSQL database logging facilities.

We believe we found the best solution for improving SURFnet IDS by advising adding Snort to the current setup.

Future Work

As SURFnet IDS is a constantly evolving product, many more improvements could be considered. A possible topic might be the detection of technical problems on a network, such as hosts broadcasting large numbers of bogus messages causing the network to become slow.

During our research we reached the conclusion that there are very few high quality open source IDS related applications. We intentionally decided to exclude proprietary closed source products, for the reasons we discussed in our introduction. It might be worth investigating whether or not there are any high quality proprietary products on the market. This would require resources in the form of budget to acquire such products and a lot of time to rigorously test them.

Bibliography

[1] SURFnet IDS homepage,
http://ids.surfnet.nl/

[2] *Onderzoeksrapport RP1: IDS*, A. Dekker and C. Groen, February 2005,
http://staff.science.uva.nl/~delaat/snb-2004-2005/p16/report.pdf

[3] *SURFnet Intrusion Detection System*, K. Trippelvitz and H.J. Blok, July 2005,
http://staff.science.uva.nl/~delaat/snb-2004-2005/p30/report.pdf

[4] *Onderzoek veiligheid SURFnet IDS*, L. Bordewijk, J. Mace, March 2006,
http://staff.science.uva.nl/~delaat/snb-2005-2006/p11/report.pdf

[5] *Intrusion Detection System honeypots*, M. Meijerink, J. Spellen, February 2006,
http://staff.science.uva.nl/~delaat/snb-2005-2006/p29/report.pdf

[6] Nepenthes homepage,
http://nepenthes.mwcollect.org/

[7] Argos homepage,
http://www.few.vu.nl/argos

[8] *Argos: an Emulator for Fingerprinting Zero-Day Attacks*, G. Portokalidis, April 2006,
http://www.cs.kuleuven.ac.be/conference/EuroSys2006/papers/p15-portokalidis.pdf

[9] Tripwire homepage,
http://sourceforge.net/projects/tripwire/

[10] Samhain homepage,
http://www.la-samhna.de/samhain/

[11] Aide homepage,
http://www.sc.tut.fi/~rammer/aide.html

[12] Aide Sourceforge,
http://sourceforge.net/projects/aide/

[13] Honeyd homepage,
http://www.honeyd.org/

[14] *Honeypots: Tracking Hackers*, L. Spitzner, November 2002, ISBN: 0-321-10895-7

[15] *Simulating Networks with Honeyd*, R. Chandran and S. Pakala, December 2003,
http://paladion.net/papers/simulating_networks_with_honeyd.pdf

[16] Honeytrap homepage,
http://honeytrap.sourceforge.net/

[17] Honeytrap Sourceforge,
http://sourceforge.net/projects/honeytrap/

[18] Prelude homepage,
http://www.prelude-ids.org/

[19] Commercial Prelude homepage,
http://www.prelude-ids.com/

[20] IETF Intrusion Detection Working Group,
http://tools.ietf.org/wg/idwg/

[21] Experimental RFC 4765,
http://www.prelude-ids.org/IMG/txt/rfc4765.txt

[22] Snort IDMEF plugin,
http://sourceforge.net/projects/snort-idmef/

[23] Snort homepage,
http://www.snort.org/

[24] Bleeding Edge Snort signatures,
http://www.bleedingsnort.com/

[25] Oinkmaster homepage,
http://oinkmaster.sourceforge.net/