

# Alert verification through alert correlation – an empirical test of SnIPS

Teodor Sommestad and Hannes Holm, Swedish Defence Research Agency FOI, Linköping, Sweden

**Abstract:** A significant problem with today's intrusion detection systems is the high number of alerts they produce for events that are regarded as benign or non-critical by system administrators. A large number of solutions has been proposed to deal with this issue. This paper tests SnIPS, a tool that correlates alerts from the intrusion detection system Snort and assign beliefs host has been compromised at different occasions. The tests are performed against data collected from a cyber security exercise where 51 compromises of monitored machines occurred. The beliefs assigned by SnIPS are not calibrated in the sense that they reflect the probability that a host has been compromised. However, a compromise is more likely when alerts have a high belief. Alerts from SnIPS's with high beliefs also have a better precision than the high-priority alerts from Snort, even if static network information is used to verify these alerts. However, the recall of SnIPS is lower than if high-priority alerts from Snort are used.

**Keywords:** Intrusion detection; Snort; alert verification; alert correlation; empirical test

## 1 Introduction

An intrusion detection systems (IDS) is a valuable security technology for practitioners that has undergone extensive scientific research. An IDS can serve both as a real-time tool to prevent ongoing attacks and as a support to forensic analyses or incident handling performed after an intrusion has occurred.

Overall, IDSs can be categorized as either anomaly based or signature based Axelsson (2000). Anomaly based IDS concerns profiling the "normal" state of a system in operation and identifying malicious events based on deviation from this profile. Signature based IDS, on the other hand, build on rules that operate the same regardless of the "normal" background traffic. Most scientific research has focused on anomaly based IDS (Catania & García, 2012). As opposed to the research literature, the market is dominated by signature based network IDSs. For instance, the technology analysis firm Gartner notes that signature quality remains the primary selection factor on the market for IDSs with preventive capabilities (i.e., intrusion prevention systems) (Young & Pescatore, 2009). This discrepancy could be because reading and writing complex rules for most signature based IDS is a relatively straightforward process; most employ well documented high-level domain specific languages. This straightforwardness also makes it rather easily test the reliability of rules and understand the limitations of the IDS. Furthermore, as most rules are general (i.e., not specialized to a certain system in operation), they can be shared with others'. A good example of a signature based system is network IDS Snort, which has a vivid community and tens of thousands of rules available for public usage.

However, while signature based IDS are dominant in practice and relatively straightforward to use, they do not come without problems. In particular, qualitative analyses by Werlinger et al. (2008) and Goodall et al. (2009) found that, in practical applications, considerable system administrator expertise is required to configure filters in order to increase the effectiveness of the IDS. This problem is clearly illustrated by the many studies that focus on false positives produced by signature based IDS. For example, Tjhai et al. (2008) found that 96% of the alerts provided by Snort for traffic corresponding to their university's web server were false positives.

There is a number of ideas on how to increase the effectiveness of signature based IDS. In a recent review, Hubballi and Suryanarayanan (2014) identified nine different techniques. This paper focuses on one of these: alert verification. The aim of alert verification is to identify alerts that are associated with real intrusions in machines and remove alerts that did not lead to a compromised machine. There are multiple proposals for how this can be accomplished. This paper focuses on the specific solution called SnIPS and aims at answering the three research questions:

- 1) How calibrated are the beliefs assigned by SnIPS on host compromise?
- 2) How does SnIPS compare to the high-priority alerts of Snort?
- 3) How does SnIPS compare to verification based on static network information?

An overview of different approaches to alert verification is given in section 2. Section 3 provides a more in-depth description on SnIPS, the specific proposal addressed in the present paper. Section 4 describes the test of SnIPS. Section 5 presents the results of this test. Section 6 discusses these results and concludes the paper.

## **2 Alert verification approaches**

Alert verification aims at determining if an alert was associated with a successful attack or not. The idea rests on the notion that it is more relevant for system administrators to know when a machine has been compromised by an attacker than to know when the attackers failed to compromise a machine. Some of these correlate alerts with static information such as vulnerability data; other correlate alerts to dynamic information such as system events.

### ***2.1 Alert verification using static information***

Many proposals within alert verification rely on an analysis performed before the alert was received using static information. Often information on software vulnerabilities or network topology. Many of these solutions simply assume that a successful attack must target a machine and software with a known vulnerability that can be exploited (see (Gula, 2002) (Njogu & Jiawei, 2010) (Neelakantan & Rao, 2008) (Gupta, Joshi, Bhattacharjee, & Mundada, 2012) (Gagnon, Massicotte, & Esfandiari, 2009) (Njogu, Jiawei, Kiere, & Hanyurwimfura, 2013) (Haukeli, 2012) (Xiao & Xiao, 2007)). Other ideas use static information to prioritize alerts based on the potential impact of the vulnerability (Gupta et al., 2012) or asses if potential attackers have the connectivity required to exploit the known vulnerabilities (Meng & Li, 2012)(Haukeli, 2012)(Xiao & Xiao, 2007).

Sommestad and Franke (2015) tested a number of filtering procedures based on the passive approach against the same data used for tests in this paper. These tests assessed the correspondence to attacker actions, regardless if they were successful or not. Even though their tests did not investigate alert verification specifically, they indicate the efficacy of filters for alert verification. The most successful filter in terms of increased the portion of alerts that arise because of attacker activity simply removed alerts did not involve an external host. This doubled the precision of the alerts, but at the cost of reducing the number of detected attacker actions by 60 percent. This filter is compared to SnIPS in the present study.

### ***2.2 Alert verification using dynamic information***

There are a number of proposal that use dynamic information generated before, during, or after the attack. Some of these are active in the sense that they probe the monitored machine or network for some additional information that could verify the compromise. This work include proposals that verify that alerts lead to a compromise by looking for anomalies in the output from a network service (e.g. a web server) after the alert was produced (Carlson & Bishop,

2005)(Bolzoni, Crispo, & Etalle, 2007) and proposals that inspect processes or files after an alert to see if some modification expected from a compromise has occurred (Kruegel, Robertson, & Vigna, 2004). Little recent work has been performed along these lines, and to the authors' knowledge none of these proposals have been subjected to rigorous tests and none of them are implemented in publicly available prototypes.

An alternative to additional probes is to assess if multiple alerts have been generated from security sensors that collectively indicate a compromised machine. In this approach, dynamic information from sensors are mapped to a signature for a compromised machine. Despite the simplicity of this approach, which requires no additional sensors to operate, there is little research along these lines that focus on alert verification. This paper tests SnIPS, the most widely referenced solution that binds alerts together to assess if a host has been compromised. SnIPS is described further below.

### **3 SnIPS**

For a complete description of SnIPS (Snort Intrusion Analysis using Proof Strengthening) and its internals the reader is referred to descriptions by its originators (Sundaramurthy, Zomlot, & Ou, 2011)(Ou, Rajagopalan, & Sakhivelmurugan, 2009). The overall purpose of SnIPS is to aggregate many uncertain pieces of information provided by the IDS Snort to predicates that correspond to more certain issues that should be conveyed to system administrators. This is accomplished by classifying Snort alerts according to a set of predicates and correlating observations of these predicates through Dempster–Shafer theory. These internal models include rules concerning a) probes against the host, b) exploits sent to the host, c) exploits sent from the host, and d) command-and-control-traffic exchanged with other compromised hosts. The primary output of the tool is the belief (scored 0-1) that a host has been compromised during a certain period of time. Thus, the alerts from Snort are expected to follow a certain pattern before, during, and after a host is compromised.

SnIPS has been evaluated against the Darpa dataset, a dataset from the Honeynet project, and a dataset from the Treasure Hunt event held as a part of a graduate course at University of California at Santa Barbara (Ou et al., 2009). Only brief descriptions of these tests are available, but the results seem to be promising. For instance, SnIPS assigned beliefs to 90 of 100 attacks in the Darpa dataset, despite a gap between the date of the rules used by SnIPS and the date of the attacks (performed in 1998); SnIPS confirmed that the dataset from the Honeynet project was compromised as the “truth file” indicates; and two of three stages in the attack of the Treasure Hunt event was reported by SnIPS. Unfortunately, none of these tests offers quantitative analyses of how accurate the analysis of SnIPS is or if it successfully aggregates Snort's alerts to high quality information of value to a system administrator. For instance, while it is clear that SnIPS reports compromises when there are Snort alerts concerning them, it is unclear if SnIPS reduces the number of false alarms significantly and if the assigned beliefs are useful.

### **4 Materials and methods**

This paper evaluates SnIPS using a dataset from a cyber security exercise held in 2012 to answer the three research questions:

- 1) How calibrated are the beliefs assigned by SnIPS on host compromise?
- 2) How does SnIPS compare to the high-priority alerts of Snort?
- 3) How does SnIPS compare to verification based on static network information?

The cyber security exercise is called SMIA 2012, and has a publically available dataset<sup>1</sup> that has been previously described in (Sommestad et al., 2015; Sommestad & Sandström, 2015). The exercise was carried out within the Cyber Range And Training Environment (CRATE), which is a cyber range managed by the Swedish Defence Research Agency (FOI), and was a capture the flag exercise (extract data from hosts) that involved two red teams (i.e., attackers), but no blue teams (i.e., defenders). Section 4.1 describes the preparation of the monitored system environment. Section 4.2 describes the injection of benign events. Section 4.3 describes the injection of malicious events. Section 4.4 describes the configuration of detection sensors. Finally, section 4.5 describes the process of coding of events and alerts.

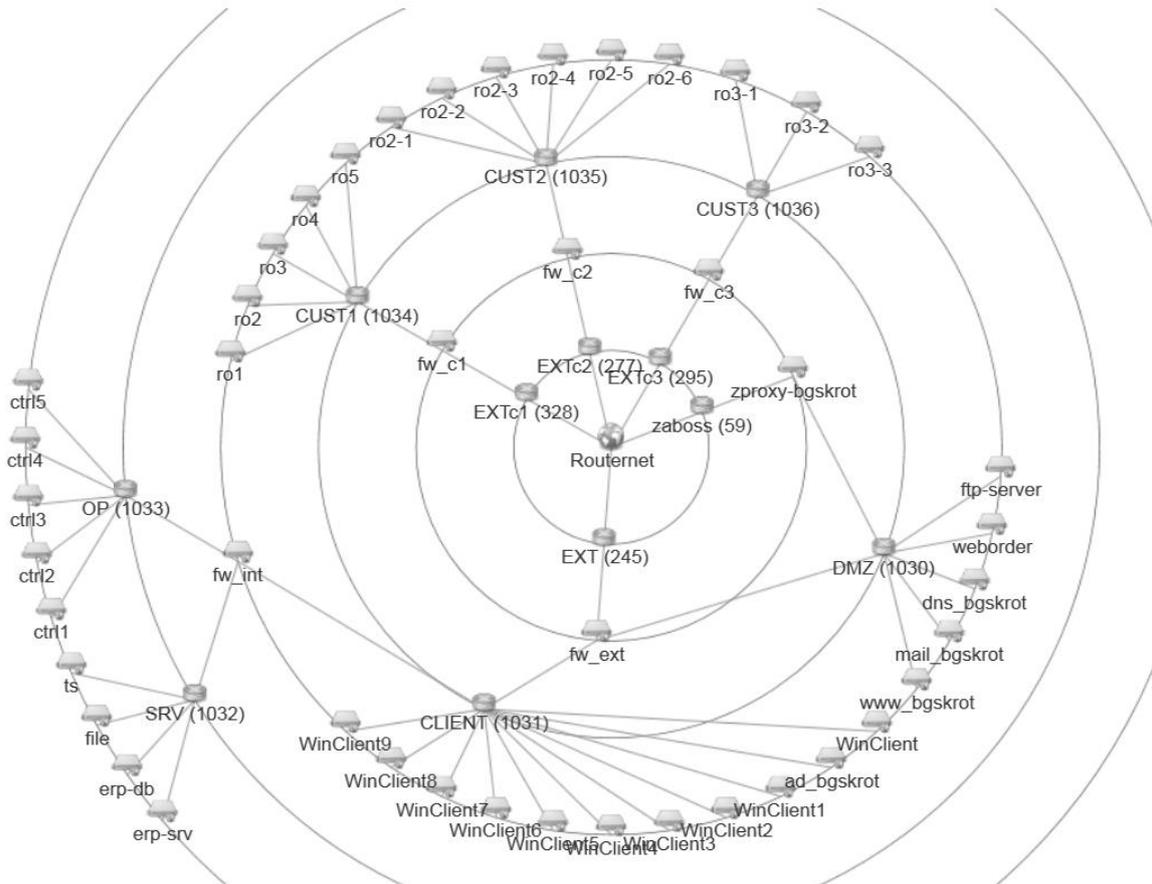
#### ***4.1 Preparation of the monitored system environment***

Over a thousand virtual machines were deployed in CRATE, together forming computer networks of various size and complexity. Of these, five fictitious organizations containing a total of 138 hosts were targeted and monitored. The number of hosts in the organizations varied (4, 15, 36, 39 and 44 hosts), with the smaller ones representing small organizations with a few servers and the larger ones representing organizations with several network zones and firewalls limiting access possibilities between them. Figure 1 illustrates one of the monitored computer networks.

A number of different operating systems and applications were instantiated in these networks. Different patch levels and versions of Windows (2000, XP, 2003, 7, 2008) and a number versions of Linux-based distributions (e.g., CentOS, Gentoo, Debian, Ubuntu) were used. These ran a number of desktop applications and server applications. Among others', the client-side applications included different versions of Adobe Reader, software development tools like Visual Studio, web browsers like Internet Explorer or Firefox, and applications of the Microsoft Office suite or Open Office. Server-side applications included different versions of Wordpress, phpMyAdmin, IIS, Domain Controllers, network infrastructure services (e.g., DNS and DHCP), and FTP servers. The aim was that the deployed applications should be representative of the standard software found in most enterprise computer networks. However, custom built applications (e.g., interconnected spreadsheet applications) and larger enterprise systems (e.g., ERP systems) were not present. Furthermore, to enable a meaningful exercise for the attacking teams, and to produce enough data for the test, these applications were more vulnerable than the ones found in the typical enterprise (i.e., they had not been updated and patched recently).

---

<sup>1</sup> <ftp://download.iwlab.foi.se/dataset/smia2012/>



**Figure 1. Example of an organization monitored and targeted in the test.**

## 4.2 Injection of benign events

An essential component in a test addressing precision and false positives of an IDS is the benign background traffic. Without background traffic, detection of attacks is trivial, because every event is an attack and every network request indicates a compromised host. Two main alternatives are available to produce background traffic: recording from real networks or simulating synthetic events (Athanasiaades, Abler, Levine, Owen, & Riley, 2003). Both these alternatives have their pros and cons.

Apart from the various traffic generated autonomously by the installed systems and software, many kinds of events were produced synthetically with scripts implemented in Auto IT (Jonathan Bennett AutoIt Consulting Ltd, 2015). These scripts emulated user actions through sending and receiving emails to one another, surfing websites, opening emails/attachments and accessing files on local machines as well as shared network folders. The synthetic user agents performed these behaviors according to predefined instructions created based on historical actions of seventeen real users working in three different office environments. Each work week of these seventeen users was used as a template for a synthetic user agent. Thus, the activities performed by the scripted user agents followed the same sequence and had the same intensity as real users do during a work week. However, there are some limitations to the realism provided by the user agents.

First, the cyber range is a synthetic environment of limited scope. Because of this it did not contain all the email accounts that the users had emailed, all the files they had used or all the websites they had visited. For example, did not host all the websites that real users had visited.

Consequently, many of the actions of the real users' lacked meaning in the isolated environment of this test, and a mapping between real actions to simulation had to be made. Second, the scripted users only performed standard behavior of users such as surfing, accessing files and sending emails. More sporadic tasks, like installation of custom software applications or changes in system configurations, were not performed.

### ***4.3 Injection of malicious events***

In this test, two independent teams attacked the computer networks in order to find secret keys hidden in them. One team consisted of security researchers from FOI, the other team consisted of security specialists from the Swedish Armed Forces Network and Telecommunications Unit. Both teams restricted themselves to using only publicly available tools and publicly known exploits, e.g., the tools and exploits packed with the operating system Backtrack 5.

The experiment started at noon a Tuesday and continued until Thursday noon the same week. The attackers had no prior knowledge about which machines the secret keys were hidden in, but were told that they were in some of the nine monitored networks. As a result, a mix of reconnaissance and penetration activities was conducted. According to their own activity logs the two teams conducted 197 penetration attempts and 192 network scans (or other reconnaissance related activities) during the experiment. These activities led to 51 unique machines of the monitored machines.

### ***4.4 Configuration of detection sensors***

Nine networks were monitored in this test. In these networks, all traffic was monitored using Snort 2.9.0.5 running a snapshot of the full official Sourcefire rule set dated March 8, 2011. The sensors were set to ignore alerts of priority 0 ("None") and priority 4 ("No priority"). In addition, the following rules were ignored because of the numerous false (low-priority) alerts they produced in the test environment: SID 129-4, SID 129-12 on by small packets, SID 399, and SID 3218.

A total of 380 041 alerts were generated by Snort concerning the monitored networks; 6756 priority-1 alerts, 79567 priority-2 alerts, and 293 718 priority-3 alerts. A priority of 1 signifies "dangerous and harmful attacks", for example, SID 2103 (a buffer overflow of Samba). A priority 2 alarm signifies "suspicious signatures potentially preparing attacks", for example, SID 3677 (denial of service attack against Ethereal). A priority 3 alarm signifies "unusual traffic not identified as dangerous", for example, SID 2465 (access to resources using Samba)(Riebach, Rathgeb, & Toedtmann, 2005). Naturally, a less severe alarm is likely to provide more false alarms, and require greater rule customization to be useful. In this test we compare SnIPS to the priority-1 alerts produced by Snort. These alerts include, among other things, attacks that involve executable code and attempts to gain privileges in a system and is a straightforward operationalization indication of a potentially compromised host. Another reasonable operationalization would be to only include alerts that were of classes "Successful Administrator Privilege Gain" and "Successful User Privilege Gain", which are two classes of priority-1 alerts. However, none of the 6756 priority-1 alerts belonged to these classes. The vast majority concerned "Attempted User Privilege Gain" (5387 alerts) and "Executable code was detected" (1297 alerts).

The latest available version of SnIPS was used. This version, dated April 2012, required some modifications to work with available versions of MySQL and Apache. The creators of SnIPS kindly provided a working version. During the test SnIPS was set to analyze the entire time range the experiment for each organization at a time.

As SnIPS predicates correspond to all three kinds of Snort priorities, it was provided with all alerts. This resulted in 1852 statements concerning compromises from SnIPS, with mean belief of 0.19. However, for the comparison of the accuracy for SnIPS and Snort (section 5.2), priority 2 and 3 alerts were discarded as they are meant to indicate preparation of intrusions, rather than the intrusions themselves.

#### ***4.5 Coding of events and alerts***

The relationship between alerts and recorded compromises were used to assess the accuracy of SnIPS and compare it to the accuracy of Snort. Accuracy was assessed as precision and recall, where precision is the portion of alerts associated with a logged compromise and recall is the portion of logged compromises that gave rise to an alert. In addition, the number of alerts was used to estimate of the workload required to manually inspect them. As will be described below, the belief assigned to the alerts from SnIPS was considered in the analysis in terms of deciles.

Two issues were identified through these inspections of the logs, screen recordings, and interviews with the attackers.

First, the attackers were not as adamant about logging unsuccessful attempts and scans. They often tried several exploits but only logged a few of these, e.g. the ones they were successful with, and did not record all their reconnaissance activity. While this makes the attack log incomplete, it does not pose a threat to the present study as it only deals with successful compromises (just as SnIPS). Because, according to the attackers, successful compromises were logged in all cases. This claim was verified by the researchers using screen recordings available on a subset of the attackers' machines.

Second, the times logged for the events are not exact. In many cases, the attackers logged their actions post-hoc, e.g. after they had compromised a machine, elevated their privileges, or extracted some information. For this reason, they often estimated the time of compromise after the fact. Discussions with the attackers as well as comparisons to Snort-alerts suggests that the vast majority of these estimates are within 15 minutes of the time that an exploit was executed. However, to be on the safe side, all Snort priority-1 alerts within one hour of the recorded time are considered correct indicators of compromise in this test. SnIPS, which also take actions performed after the exploit into account in its assessment, was given an even more generous time frame for its indicators of compromise. All SnIPS alerts that stated a time interval within one hour before the logged compromise and three hours after the logged compromise was considered as correct indicators of a compromise.

## **5 Results**

This section addresses the three research questions. First, the calibration of the beliefs assigned to SnIPS's alerts are addressed. In other words, it is tested whether statements are more correct when higher beliefs are assigned. Second, the output of SnIPS is compared to the output of Snort's built-in prioritization scheme. Third, Snort's built-in prioritization scheme combined with the best verification filter in the evaluation by Sommestad and Franke (2015) is compared to SnIPS. The comparisons focus on recall, precision, number of alerts and precision.

### ***5.1 SnIPS's belief system***

The beliefs in SnIPS are calculated from supporting observations using the evidence theory of Dempster-Shafer. As such, they do not necessarily hold the mathematical properties of probabilities, but are calculated from subjective probabilities (Shafer, 1992). The creators of SnIPS do not explicitly state what the beliefs produced by SnIPS express. However, a reasonable interpretation is that the belief reflects the precision (i.e. the probability that the host

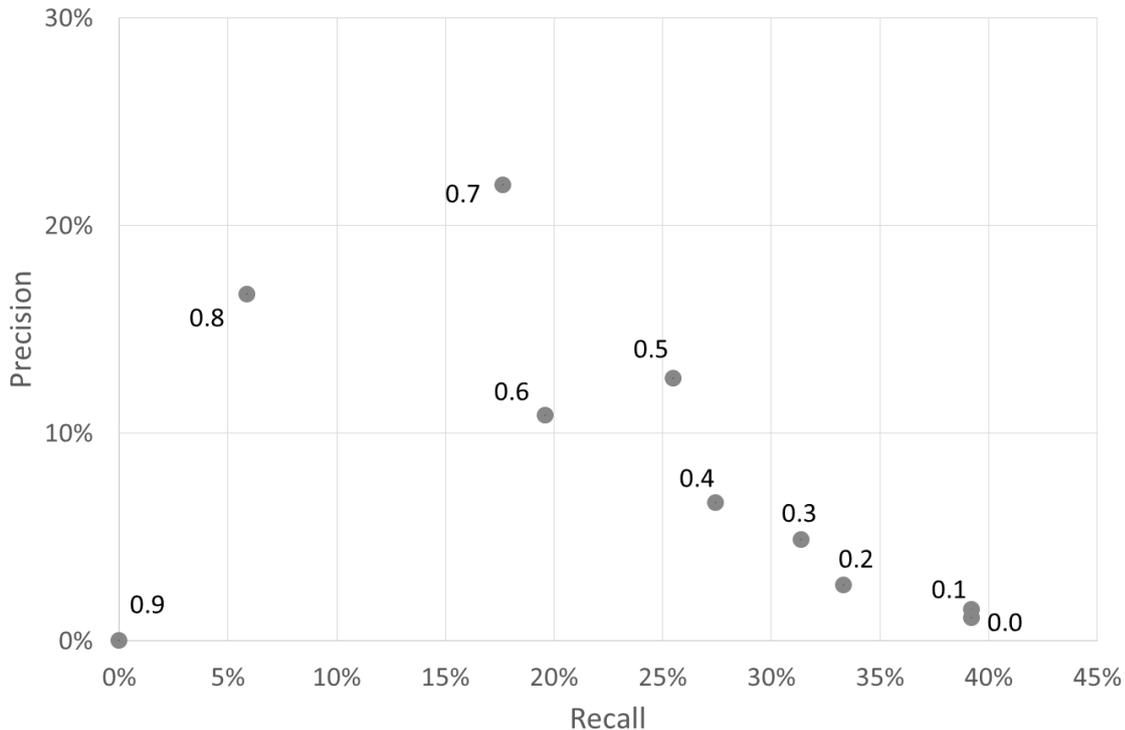
has been compromised). Table 1 describes the precision of SnIPS at ten different belief-levels. As can be seen, the beliefs are far too high to represent precision. For instance, when SnIPS assigns the belief of 0.80 to 0.90, precision is 0.27; when SnIPS assigns the belief 0.50 to 0.60, the precision is 0.18; only three of SnIPS's 702 alerts assigns with a belief of 0.1 to 0.2 could be linked to a logged compromise.

**Table 1. Precision and recall at different belief-intervals.**

<b>Belief interval</b>	<b>Number of alerts</b>	<b>Observed compromises</b>	<b>Precision</b>	<b>Recall</b>
0.9-1.0	1	0	0.00	0.00
0.8-0.9	17	3	0.18	0.06
0.7-0.8	23	6	0.26	0.12
0.6-0.7	51	1	0.02	0.02
0.5-0.6	11	3	0.27	0.06
0.4-0.5	108	1	0.01	0.02
0.3-0.4	119	2	0.02	0.04
0.2-0.3	309	1	0.00	0.02
0.1-0.2	702	3	0.00	0.06
0.0-0.1	511	0	0.00	0.00

While the beliefs' are uncalibrated, they do have a relationship to precision. The Pearson correlation between the mean of these belief intervals (e.g. 0.75 for the interval 0.7-0.8) and the precision is 0.45. Thus, a higher belief is associated with a higher precision. Furthermore, there is a clear relationship between the number of alerts produced by SnIPS and the chosen belief-thresholds. Just one alert had a belief higher than 0.9; 103 alerts were produced with a belief higher than 0.5; 1852 alerts were produced with any belief. The Pearson correlation between the number of alerts and the threshold is -0.84.

Another reasonable usage of SnIPS's beliefs is to set a threshold for how high the belief should be for a system administrator to care about the alert. Figure 2 illustrates how ten different belief-thresholds relate to precision and recall. As one would expect from a reasonable belief system, and as noted above, a higher belief-threshold is associated with a higher precision. Thus, it is more likely that alerts with a high belief is a due to a real compromise. However, in this (rather limited sample) this relationship is not monotonic: a belief-threshold of 0.7 yields a higher precision than those of 0.9 and 0.8, and a belief-threshold of 0.5 yields a higher precision than 0.6.



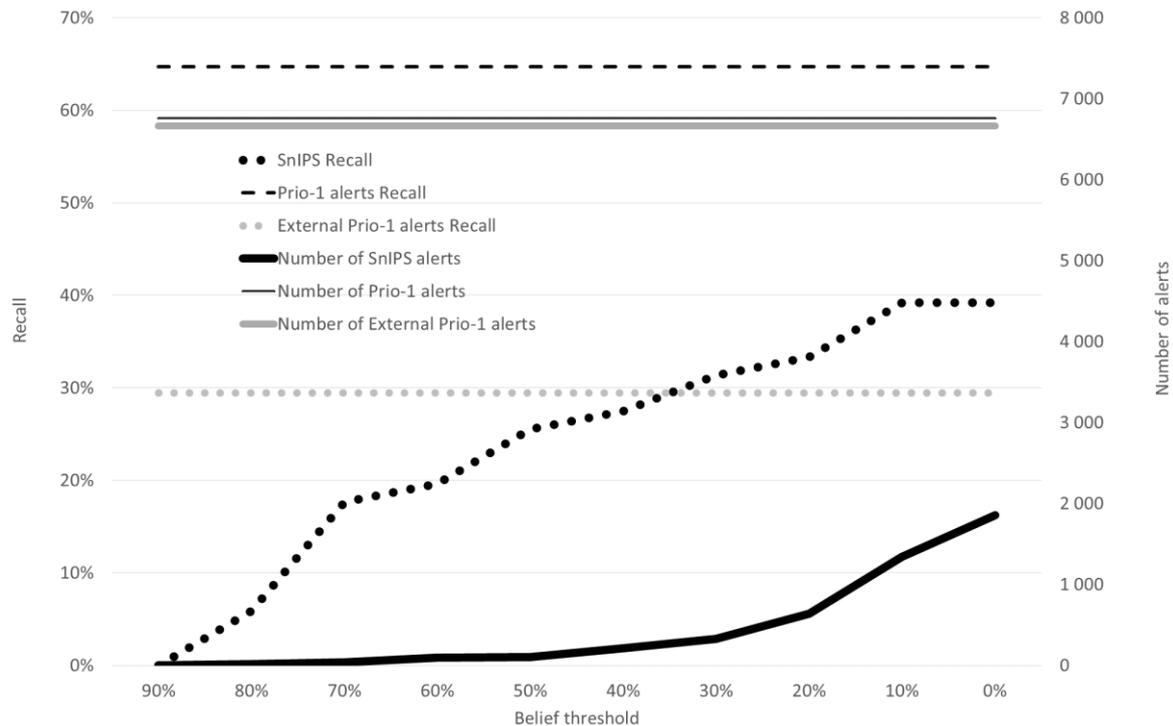
**Figure 2. The precision and recall of SnIPS alerts on different belief-thresholds**

Figure 2 also illustrates that higher beliefs are associated with lower recall. While this is undesirable, as thresholds with both high recall and precision is ideal, it is reasonable that more compromises are detected when the information quality requirement (i.e. the belief-threshold) is reduced.

## 5.2 SnIPS compared to Snort's priority 1-alerts

SnIPS aims to augment the intrusion detection system Snort by correlating its alerts to assess if a host has been compromised. Thus, one would expect that SnIPS provides better indicators of compromise than Snort. Especially since Snort make no aspiration to assess if a malicious activity was successful or not; Snort simply reports potential threats to security. Still, a reasonable comparison can be provided by only including Snort priority-1 alerts, as these are meant to indicate “dangerous and harmful attacks” (see section 4.4).

Of the 6756 priority-1 alerts produced by Snort, 132 arrived within one hour of a logged successful compromise of the machine (i.e. a precision of 0.02). If all of these 6756 would have been inspected or reacted upon, a system administrator would have identified or handled 32 of the 51 logged compromises (i.e. a recall of 0.65). As illustrated in Figure 3, SnIPS do not reach the same recall at any belief threshold, but offer a considerably fewer alerts than Snort's priority-1 alerts.



**Figure 3. The recall and number of Snort’s priority-1 alerts and SnIPS at different alerts on different belief-thresholds.**

### 5.3 SnIPS compared to Snort’s priority 1-alerts filtered with network information

When Sommestad and Franke (Somme stad et al, 2015) tested filters using static information, the best alternative was to only look at alerts that involved external hosts. This verification technique does not offer improvements of the same magnitude when it is applied in conjunction with priority-1 alerts to identify compromised machines in this test. It reduces the recall to less than a half, to 0.29, and only reduce the number of priority-1 alerts to inspect by 95, to 6661. The precision resulting from this verification technique is only 0.9%.

## 6 Discussion and conclusions

This experiment was performed using synthetic user agents in a synthetic environment. While the attackers were security professionals, neither the attacks nor scenario can be expected to be fully representative the typical cases that system administrators face in contemporary enterprises. For instance, all exploits used in the experiment was publicly known and the amount of attacks during these two days was considerable. Furthermore, the logging of attacks was not perfect, especially not with respect to the actual time of compromise. This required the present study to use an inclusive strategy when classifying alerts, accepting alerts with hours of logged compromises as true positives. Despite these limitations this experiment provides some tentative answers to the research questions.

### 6.1 The calibration of SnIPS’s alerts

Research question one concerned the belief-system used by SnIPS. The experiment shows that the beliefs of SnIPS seem to follow a logical pattern where high beliefs are associated with high precision, i.e. a higher likelihood of the host being compromised. However, in absolute terms,

the beliefs are too high in all deciles investigated in this test and a general reduction of the belief-strength of SnIPS seems to be appropriate. Thus, they are not calibrated with respect to precision and should not be interpreted as the actual probability that a host has been compromised.

## ***6.2 SnIPS versus Snort and filters on network information***

Research question two and three concerned how SnIPS compared to the priority-1 alerts of Snort and a verification technique for Snort's priority-1 alerts based on network information. Compared to these alternatives, the beliefs offer more granular priorities that can be used to filter out alerts of higher precision. Perhaps more importantly, SnIPS do not produce an overwhelming number of alerts, scores its alerts, and enables an analyst to drill down from a predicate to the individual predicates and Snort rules that provided its belief. For instance, there are 103 SnIPS alerts with a belief above 0.5. If these 103 alerts would be all the alerts a system administrator have the time to process, 13 of the logged compromises would be handled, yielding a recall of 0.27. If the alternative was to randomly select Snort's priority-1 alerts until a recall of 0.27 was obtained, the system administrator would have to go through some 40% of them. This would mean that the system administrator would have to go through some 2722 Snort's priority-1 alerts identify the same number of compromises that can be identified with the 103 alerts with the highest beliefs produced by SnIPS.

The utility of higher precision depends on the costs associated with missed compromises and the cost associated with spending time on handling alerts (or alternatively, the time available for handling alerts). If the resources available for manually processing alerts is constrained, SnIPS alerts is a better alternative than randomly chosen Snort alerts of priority-1, and much better than Snort alerts of priority-1 that involve external machines. However, it is possible, and perhaps even likely, that system administrators use some other clues or heuristics to prioritize their work efforts. For example, if clusters/bursts of three priority-1 alerts that come in sequence are required for the administrator to look at a host, there would be some 292 cases to inspect, and 14 compromises would be dealt with. The same number of compromised machines would have been dealt with if the 201 SnIPS-alerts of belief 0.4 and higher had been inspected. Thus, simple heuristics can give improvement of the almost the same magnitude as SnIPS. However, as the results show here, simple verification techniques based on network information are not likely to provide any improvements that compare with SnIPS.

## ***6.3 Further work***

SnIPS is not only a tool for clustering and prioritizing alerts, it also formalize how a system administrators typically correlate and reason about alerts. This test did not identify any apparent issues associated with the type of reasoning embedded in SnIPS. On the contrary, high beliefs from SnIPS are associated to a likely compromise. It is possible, perhaps even likely, that a reasoning system such as SnIPS can improve intrusion detection systems significantly by correlating alerts. Especially if additional sensors would be included in the reasoning framework and the beliefs would be calibrated with actual compromises. Research that investigate this type of reasoning about alerts with additional sensors would be valuable.

In addition, because SnIPS is supposed to help a system administrator to analyze network traffic corresponding to systems in operation, it would make sense if further research on SnIPS evaluates the value of SnIPS in experiments involving such system administrators. Tests would enable straightforward comparisons to various heuristics used by system administrators, could also investigate if the clustering of alerts provided together with a SnIPS-alert helps a system administrator further, and could investigate at what workload a support system like SnIPS offers value.

## 7 Acknowledgements

The authors would like to thank Xinming Ou and Loai Zomlot for their help with setting up SnIPS for this test.

## 8 References

- Athanasiaades, N., Abler, R., Levine, J., Owen, H., & Riley, G. (2003). Intrusion detection testing and benchmarking methodologies. In *Proceedings of Information Assurance, 2003*, pp. 63–72, Darmstadt, Germany, IEEE.
- Axelsson, S. (2000). *Intrusion detection systems: A survey and taxonomy. Technical Report* (Vol. 99). Göteborg, Sweden.
- Bolzoni, D., Crispo, B., & Etalle, S. (2007). ATLANTIDES: An architecture for alert verification in network intrusion detection systems. In *Proceedings of the 21st Large Installation System Administration Conference (LISA '07)*, pp. 141–152, Dallas, TX, USENIX.
- Carlson, J., & Bishop, M. (2005). Verify Results of Network Intrusion Alerts Using Lightweight Protocol Analysis. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC'05)*, pp. 117–126, Tucson, AZ, IEEE.
- Catania, C. A., & García, C. (2012). Automatic network intrusion detection : Current techniques and open issues. *Computers and Electrical Engineering*, 38(5), 1062–1072.
- Gagnon, F., Massicotte, F., & Esfandiari, B. (2009). Using Contextual Information for IDS Alarm Classification. In *Proceedings of International Conference, DIMVA*, pp. 147–156. 2009, Como, Italy, Springer Berlin Heidelberg.
- Goodall, J. R., Lutters, W. G., & Komlodi, A. (2009). Developing expertise for network intrusion detection. *Information Technology & People*, 22(2), 92–108.
- Gula, R. (2002, May 11). Correlating ids alerts with vulnerability information, 2011 (Revision 4). Retrieved December 12, 2016 from Tenable's website: <https://www.tenable.com/sites/drupal.dmz.tenablesecurity.com/files/uploads/documents/whitepapers/va-ids-new.pdf>
- Gupta, D., Joshi, P. S., Bhattacharjee, A. K., & Mundada, R. S. (2012). IDS alerts classification using knowledge-based evaluation. In *Proceedings of the Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pp. 1–8, Bangalore, India, IEEE.
- Haukeli, J. (2012). *False positive reduction through IDS network awareness*. Master thesis, Oslo University College.
- Hubballi, N., & Suryanarayanan, V. (2014). False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*, 49, 1–17.
- Jonathan Bennett AutoIt Consulting Ltd. (2015). AutoIt Script Editor. Retrieved July 1, 2016, from AutoIts website: <http://www.autoitscript.com/site/autoit/>
- Kruegel, C., Robertson, W., & Vigna, G. (2004). Using alert verification to identify successful intrusion attempts. *Journal of Practice in Information Processing and Communication (PIK)*, 27(4), 219–227.
- Meng, Y., & Li, W. (2012). Constructing Context-based Non-Critical Alarm Filter in Intrusion Detection. In *Proceedings of the The Seventh International Conference on*

- Internet Monitoring and Protection*, pp. 75–81, Stuttgart, Germany, IEEE.
- Neelakantan, S., & Rao, S. (2008). A Threat-Aware Signature Based Intrusion-Detection Approach for Obtaining Network-Specific Useful Alarms. In *2008 The Third International Conference on Internet Monitoring and Protection*, pp. 80–85, Bucharest, Romania, IEEE.
- Njogu, H. W., & Jiawei, L. (2010). Using Alert Cluster to reduce IDS alerts. In *Proceedings of the 3rd International Conference on Computer Science and Information Technology*, pp. 467–471, Geneva, Switzerland, IEEE.
- Njogu, H. W., Jiawei, L., Kiere, J. N., & Hanyurwimfura, D. (2013). A comprehensive vulnerability based alert management approach for large networks. *Future Generation Computer Systems*, 29(1), 27–45.
- Ou, X., Rajagopalan, S. R., & Sakthivelmurugan, S. (2009). An Empirical Approach to Modeling Uncertainty in Intrusion Analysis. In *Proceedings of the 2009 Annual Computer Security Applications Conference*, pp. 494–503. Honolulu, Hawaii, IEEE.
- Riebach, S., Rathgeb, E. P., & Toedtman, B. (2005). Efficient Deployment of Honeynets for Statistical and Forensic Analysis of Attacks from the Internet. In *Proceedings of the 4th IFIP-TC6 International Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems*, pp. 756–767, Waterloo, Canada, Springer Berlin Heidelberg.
- Shafer, G. (1992). The Dempster-Shafer Theory. In S. C. Shapiro (Ed.), *Encyclopedia of Artificial Intelligence Second Edition* (Second Edition), pp. 330–331. Wiley.
- Sommestad, T., & Franke, U. (2015). A test of intrusion alert filtering based on network information. *Security and Communication Networks*, 8(3), 2291–2301.
- Sommestad, T., & Sandström, F. (2015). An empirical test of the accuracy of an attack graph analysis tool. *Information and Computer Security*, 23(5), 516–531.
- Sundaramurthy, S. C., Zomlot, L., & Ou, X. (2011). Practical IDS alert correlation in the face of dynamic threats. In *Proceedings of the 2011 International Conference on Security and Management*, Las Vegas, Nevada, ACM.
- Tjhai, G., Papadaki, M., Furnell, S. M., & Clarke, N. L. (2008). Investigating the problem of IDS false alarms: An experimental study using Snort. In *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*, pp. 253–267, Milano, Italy, Springer Berlin Heidelberg.
- Werlinger, R., Hawkey, K., Muldner, K., Jaferian, P., & Beznosov, K. (2008). The challenges of using an intrusion detection system: is it worth the effort? In *Proceedings of SOUPS '08 Proceedings of the 4th symposium on Usable privacy and security*, pp. 107–118, Pittsburgh, PA, ACM.
- Xiao, M., & Xiao, D. (2007). Alert Verification Based on Attack Classification in Collaborative Intrusion Detection. In *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, pp. 739–744, Qingdao, China, IEEE.
- Young, G., & Pescatore, J. (2009, April 14). *Magic quadrant for network intrusion prevention system appliances*. Retrieved December 11 2016, from <http://www.adexsus.com/v2/pdf/Detectores de Intrusos/Gartner/Cuadrante Magico.pdf>