

# An Assessment of the DARPA IDS Evaluation Dataset Using Snort

S Terry Brugger\*      Jedadiah Chow†

November 8, 2005

## Abstract

One of the many criticisms of the DARPA IDS evaluation is that it did not evaluate traditional, signature based, off-the-shelf intrusion detection systems. We performed such an evaluation on the 1998 dataset using Snort to determine the usefulness of the DARPA dataset, and found that overall detection performance was low and false positive rates were unacceptable. We present these results in greater depth, which indicate that the dataset does consist primarily of attacks that are difficult for signature based detectors to find; however, we find no support that the false positive rate is realistic.

**Keywords:** Snort, NIDS, Intrusion Detection, DARPA IDS Evaluation Dataset

## 1 Introduction

In 1998 DARPA recognized the need to be able to perform quantitative evaluations on intrusion detection systems. MIT's Lincoln Labs was contracted to work with the Air Force Research Laboratory in Rome, NY to build an evaluation dataset and perform an evaluation of the then current IDS research being funded by DARPA. They built a small network intending to simulate an Air Force base connected to the Internet, producing background activity with scripts, and injecting attacks at well defined points, and gathered tcpdump, Sun BSM, process and filesystem information. The DARPA funded IDS researchers were given seven weeks of this data with a list of when and where the attacks occurred, to train their systems. Once they trained their systems, two weeks of data were used to test the detection performance of the intrusion detection systems. The results of this evaluation were published in (Lippmann, Fried,

---

UCRL-CONF-214731

\*Department of Computer Science, University of California, Davis, One Shields Ave, Davis CA 95616, <[zow@acm.org](mailto:zow@acm.org)>, WWW home page: <http://bruggerink.com/~zow>

†Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 387 Soda Hall, Berkeley CA 94720

Graf, Haines, Kendall, McClung, Weber, Webster, Wyschogrod, Cunningham, and Zissman 2000; Lippmann, Haines, Fried, Korba, and Das 2000).

After the evaluation, the researchers whose systems were evaluated, and others in the community provided feedback on the evaluation. This resulted in changes for the 1999 evaluation, such as using more stealthy attacks, including a Windows NT target (and its audit logs), defining a security policy for the target network, and testing more recent attacks.

While the 1999 evaluation addressed some of the concerns voiced in the IDS research community, there remained serious reservations regarding its applicability. McHugh published an in-depth criticism of the evaluation based solely on the procedures used in building the dataset and performing the evaluation (McHugh 2000). While Lincoln Labs made the datasets available after the evaluations and many researchers used this data as the basis of evaluating their systems, after McHugh's criticism was published, the use of these datasets was more closely scrutinized by reviewers. In 2003 Malhony and Chan decided to look more closely at the data itself. They discovered that the data included irregularities, such as differences in the TTL<sup>1</sup> for attacks versus normal traffic, that even a simplistic IDS could identify and achieve better performance than would never be achieved in the real world (Mahoney and Chan 2003). Unfortunately, despite these criticisms, the DARPA dataset remains of interest as other efforts to provide datasets for IDS researchers, such as PREDICT (RTI International 2005) focus on the use of real traffic, which lacks the ground truth of what is and is not an attack.

McHugh's primary criticism of the evaluation was the failure to verify that the network realistically simulated a real-world network. Malhony and Chan's work proved that, in fact, the data did not authentically simulate real world conditions. Another criticism made by McHugh was that, while the evaluation was performed to test the performance of advanced intrusion detection systems, traditional, signature-based IDSs were not run against the data to see how they would fare. If existing IDSs performed satisfactorily, either more advanced systems are not necessary, or the dataset does not model sufficiently advanced attacks that require more advanced systems to detect. Such an evaluation should serve as a baseline against which all other systems are evaluated. Lincoln Labs provided such a baseline though the use of a naive keyword-based detector, which they claimed had a 30% intrusion detection rate for anywhere from zero to 100% false positives. In fact, Lippmann et. al. published a paper on how the performance of this keyword detector could be improved (Lippmann and Cunningham 1999).

Despite the warnings about the DARPA IDS evaluation dataset, we thought that it may still be useful for a first-order IDS evaluation. Given Chan and Malhony's work, it appeared that if an advanced IDS could not perform well on the DARPA dataset, it would not perform acceptably on realistic data. In the process of testing this hypothesis, we thought it would be interesting to baseline the dataset against a contemporary version of Snort, the open-source, signature-

---

<sup>1</sup>IP packet Time To Live

based intrusion detection system (Caswell and Roesch 2004). We chose Snort as it's readily available to anyone who wishes to validate our results<sup>2</sup>. While other off the shelf ID systems may provide incremental improvements over Snort in certain areas, our interest is in the general behavior of the DARPA dataset, not in performing an intrusion detector shootout. We expected that given the six years between the dataset creation and the Snort ruleset that we are using, that we could achieve excellent detection performance and could minimize the false positive rate with some fine tuning. Put another way, given the time span between the dataset and the ruleset, we expect that any attacks in the dataset that could be detected by a signature based system would be detected by the ruleset we used.

We were not familiar with the work of Pickering until after we concluded our study. He performed a similar evaluation for his undergraduate thesis. After reviewing his work, we found that his premise was different from ours, which affected his conclusions. Further, he considered the fact that Snort could achieve better performance by tuning it to the dataset, to be a failing of the dataset; an opinion we do not share. Finally, he doesn't correlate the alert to the attack, which we found demonstrates that most of the attacks are only detected through the generation of some artifact that is atypical on most networks (Pickering 2002). What we found instead, though, was that detection performance was very low, even as the false positive rate soared past the 50% guess rate line. We initially thought this may be due to a failure in the DARPA dataset to properly model attacks, or perhaps due to old rules being removed from the Snort distribution. Instead, our analysis indicates that the dataset does in fact model attacks that Snort has trouble detecting. More to the point, Snort may detect only a small portion of the connections in many types of attacks, skewing the performance figures against it. Further, many of the rules that detect these portions of attacks are policy based and produce an unacceptable rate of false positives.

This leads to the conclusion that the DARPA IDS evaluation dataset is still useful for testing intrusion detection systems in that good performance against it is a *necessary but not sufficient* condition to demonstrating the capabilities of an advanced IDS. One should not conclude based on these findings that Snort is in any way poor – it does what it is designed to do (signature based detection) and does it well. It is not a complete intrusion detection solution however – we consider an “advanced IDS” to be any system that detects attacks by means other than signature based detection. Besides some newer and additional attacks (a few of which were addressed by the 1999 dataset), the intrusion detection community needs datasets to accurately test the false positive rate of systems. We will explain our methodology, and then examine the results in detail, broken down by attack type.

---

<sup>2</sup>The same argument could be made for Bro, however we have found that while Bro is useful as a research IDS, it is not commonly used in production environments, and hence its signature base is not as actively maintained.

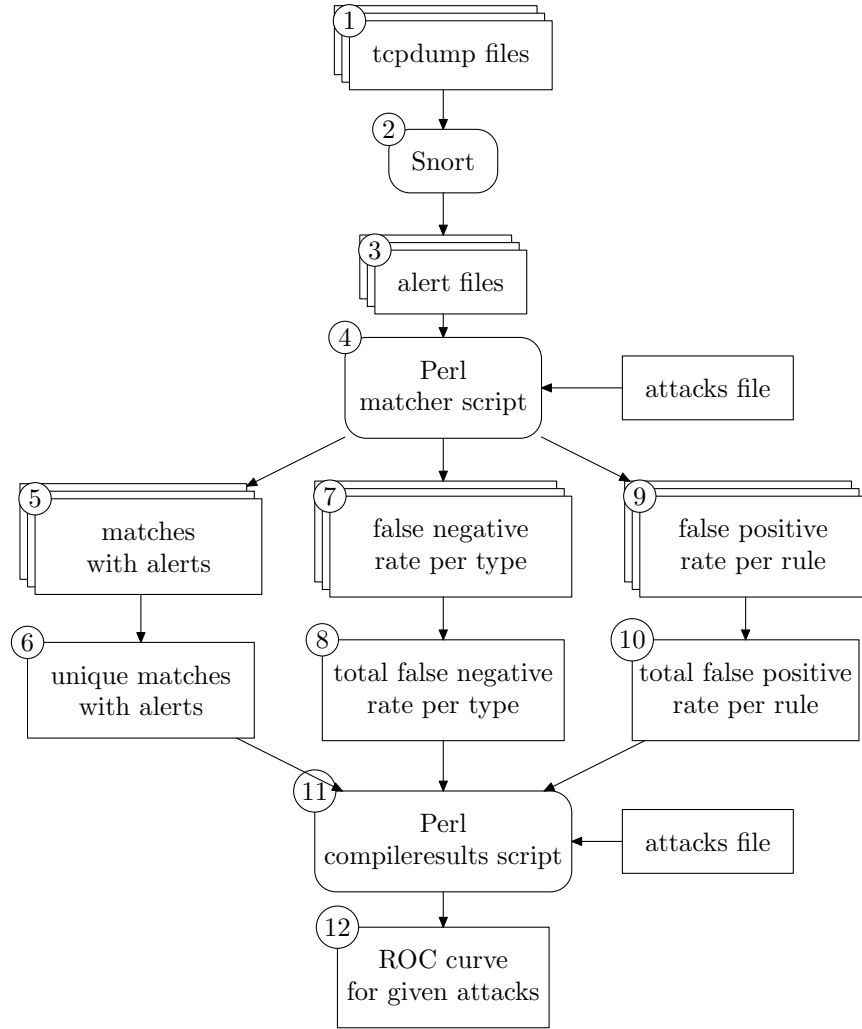


Figure 1: Basic process flow for our testing methodology. See text for detailed description.

## 2 Methodology

Figure 1 illustrates our process to run Snort against the DARPA dataset, culminating in the generation of ROC curves (which appear in the results section).

We used the tcpdump (pcap) files in the seven weeks of training data from the 1998 data set. Each file was input into Snort version 2.1.3 (Caswell and Roesch 2004) configured with all rules and intrusion detection preprocessors enabled. An alert file was generated for each tcpdump file. Using a custom built Perl

script, this alert file was matched against the list of attacks in the tcpdump file. A match was defined as the same IPs, and port numbers or ICMP types/codes. Originally we also used the timestamp; however, because the attacks are labeled by the time the malicious connection started and Snort identifies the time the packet with the attack occurs, this missed numerous matches. Not including the time corrected this problem, and while it could allow an alert and an attack to be matched erroneously, this is highly unlikely and did not occur in the large amount of data we spot checked. Further, such a case would (falsely) improve Snort's performance, and the final results would indicate that did not happen.

The matching script also reports the false negative rates per attack type and the false positive rate on a per rule basis. Once all the attack and alert files are matched, a basic Perl script counts the false positives per rule, and the false negatives per attack type. The lists of positive matches is then filtered to eliminate duplicates, where the same rule generated multiple alerts for the same attack.

Finally, another Perl script was used to generate the receiver operating characteristic (ROC) curves for a given set of attacks. By taking the lists of attacks correlated with Snort alerts, the script calculated the true positive rate for each rule on the given set of attacks, and divided it by the false positive rate for that rule. It then outputted the rule with the highest ratio (ordering those with zero false positives from highest to lowest true positive rate), and eliminated all attacks (and matches for those attacks) detected by that rule. This process was repeated until all rules were accounted for. We didn't include attacks detected by higher ranked rules in the true positive count because those detections don't provide the user with any additional benefit even though they are still saddled with the false positives that rule produces. The order that the rules are selected will follow the ROC curve from the origin (0,0) to the point at which everything is identified as an intrusion (1,1), and the selected ordering will raise the height of the curve as quickly as possible, hence maximizing the area under the curve. For example, if rule A detected attacks 1, 3, 6, 7, and 8 with 2 false positives, rule B detected attacks 1, 2, 4, and 5 with 1 false positive, and rule C detected attacks 4 and 5 with no false positives, then rule C would be output first, followed by rule A (with a 5:2 ratio beating B's 2:1 ratio since B's detection of 4 and 5 no longer count), and finally rule B. If a rule only detected attacks that higher ranked rules also detected, it was dropped, as it would be if we were tuning Snort for a real environment.

We plotted an overall ROC curve, inclusive of all attacks in the dataset, as well as ROC curves for the four standard categories of attacks: denial of service (DoS), probe activity, remote to local (R2L), and user to root (U2R). Since we could not find the categorization used by Lincoln Labs for which attacks were in which categories, we categorized each attack based on the descriptions provided by Lincoln Labs and our domain knowledge. We excluded the "anomaly" attack from any of the four categories as it appears it was scored separately from the other four categories; nevertheless, it is included in the overall results. Some attacks were difficult to categorize, for example we included the "multihop" and "warez" attacks as R2L attacks, as root access did not appear to be necessary,

and "imap" as a U2R attack as it results in a root shell, although our domain knowledge tells us that the attack can be launched remotely.

Unfortunately, the labels for the 1998 testing data have been lost with time (Lincoln Labs no longer has them, and they were not part of the data distributed to evaluation participants), so we could not use the rule ordering generated at this stage to run against the testing data and generate the ROC curves based on the performance of the rules on the training data (essentially duplicating the process used in the evaluation). Lincoln Labs recommendation is to use the 1999 testing data, as that addressed some of the problems with the 1998 data. Unfortunately, the format of the attack labels changed in the 1999 data, which would have meant rewriting our scripts. Given that our interest is in analyzing the dataset, and not comparing Snort to the other systems evaluated, and that none of the changes made in 1999 should have affected our general findings, we did not pursue the matter further.

### 3 Results

In this section we present our actual results of running Snort against the DARPA dataset. Figure 2 shows the receiver operating characteristic curves for the results overall, and for each of the four types of malicious activity. It also shows the "Random Guess" line, which represents the probabilistic performance if we guess that a given percentage of connections are malicious. Since the bulk of activity is near the origin, we use a logarithmic scale for legibility. As a side effect, the origin of the graphs is not (0,0). Also note that we use real receiver operating characteristic curves, not the "ROC" plots used by Lincoln Labs, which used a number of false positives per unit time (typically per day) for the X-scale. Finally, we base our analysis on the number of malicious connections detected, as opposed to the number of actual attacks detected because many – especially in the DoS and Probe categories – only had a small number of their connections detected, and then only by rules not specifically designed to detect that activity. This would be particularly problematic in an IPS (Intrusion Prevention System) which only blocks those packets seen as malicious. The following subsections will describe these cases in greater detail.

#### 3.1 Overall

Of the roughly 1.8 million connections containing attacks in the dataset, 1.7 million of those are from the denial of service attacks, which is to be expected given the nature of such attacks. As such, the overall results are highly skewed by Snort's performance on the DoS attacks and we will keep this section brief in preference to examining the individual categories which appear below. A visual examination of the ROC curve (line with "star" points in figure 2) shows three distinct segments: those rules which have an excellent detection rate with barely any false positives resulting the initial spike, followed by a slow climb for the large number of rules that provided some detections with many false positives,

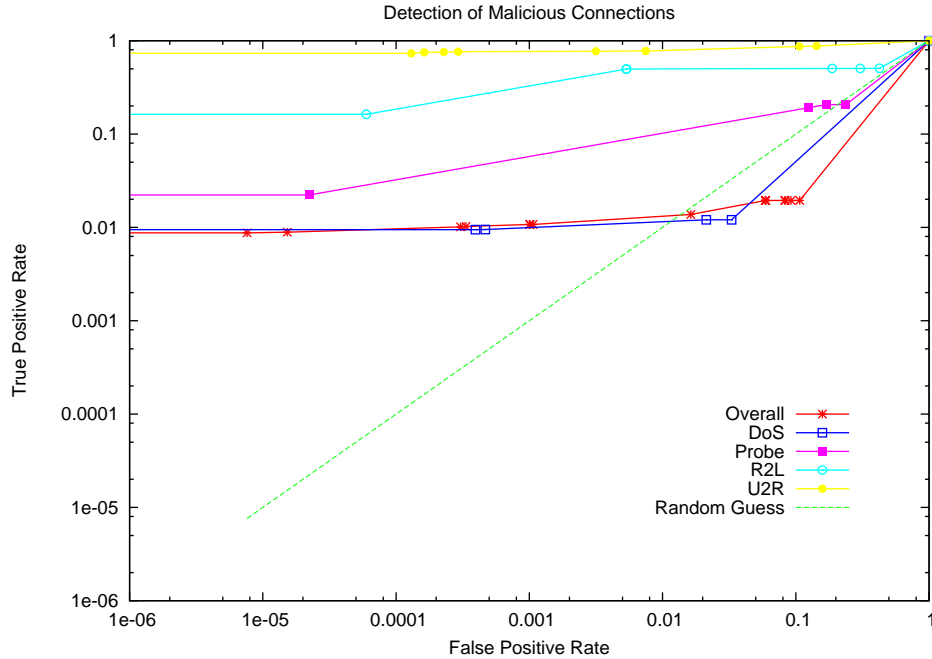


Figure 2: Receiver Operating Characteristic Curves in log,log scale. Y-axis is percentage of malicious connections detected. X-axis is percentage of alerts produced not corresponding to an attack. See text for explanation of connection versus attack detections.

and finally our extrapolation from the last rule to (1,1).

### 3.2 Denial of Service

These days, the majority of DoS attacks are distributed denial of service attacks designed to consume all network bandwidth and be almost indistinguishable from legitimate traffic. Such was not the case in 1998, however. At the time, many denial of service attacks exploited weaknesses in network stacks or services, each form of which had a unique signature, hence we expect Snort to do well on these. Still, there were a limited number of flood style attacks, the most popular of which being the SYN-flood. While a Snort preprocessor could probably be built to detect such an attack, Snort does not come with one by default, and only includes rules to detect SYN-floods from clients (zombies) with a particular packet signature. As such, the bulk of the neptune (SYN-flood) attack packets that Snort does detect in the dataset are those targeted towards ports typically associated with services that may provide information leakage, such as SNMP.

The degree of true versus false positive is related to the amount of legitimate usage of these services in the dataset.

Two denial of service attacks that Snort detects perfectly are the “back” attack against the Apache webserver, and the “land” attack against various TCP/IP stacks. Both are perfect examples of non-resource exhaustion DoS attacks that were more common in 1998 than they are today. Snort also detects almost 97% of ping of death (pod) packets with its “spp\_frag2” preprocessor, with no false positives. This accounts for the bulk of the initial detection spike on DoS attacks in figure 2 (line with empty squares).

Snort did not fare as well on the smurf, syslog, and teardrop attacks. Only five of the 250133 smurf attacks were detected, and those were only detected because they generated an ICMP port unreachable response. Snort alerts on ICMP port unreachable packets because such activity can be indicative of probing activity; yet it also fires due to network outages, and accounts for a large number of the false positives observed. Generally, however, smurf is a resource exhaustion attack utilizing ICMP packets instead of SYN packets, so we expect Snort’s detection performance on it to be similar to the neptune attack described above, for the same reasons. The second, syslog, sends syslog messages from unresolvable IP addresses, which would be difficult to write a rule for (it would actually require a preprocessor with fairly high overhead). The failure to detect the teardrop attack is a little more difficult to explain. Supposedly, the teardrop attack, which misuses fragmentation in a UDP packet should be detected by Snort using the rule “1:270”, only that rule never fired in our tests with the DARPA dataset. We couldn’t determine whether this was a problem with the dataset or Snort. The best way to determine that would be to launch the actual attack against a vulnerable machine (to verify that the attack is correct), and see if Snort detects it. Given that we didn’t have either the actual attack or a vulnerable system on hand, we did not pursue the matter further.

### 3.3 Probes

The probe activity in the dataset is all performed over the network, and most of that activity is noisy and undetectable by Snort, given the correct rules. In our experience, the default Snort rules have a very low threshold for detecting probe activity (for instance triggering on almost all ICMP activity), hence we expect it to do well in this category. The performance is shown by the line with the solid boxes in figure 2.

Overall, 16 rules detected over 20% of the probe connections. Eleven of the rules detected many of the probes from the portsweep, nmap, and satan scans with a negligible number of false positives. Two of the rules were specifically designed to detect probing activity, and the other nine were each designed to detect information leakage from a particular service. The next rule detected all ICMP ping activity, which allowed the majority of ipsweep connections to be detected (as well as a large amount of nmap activity), and also accounts for the sharp increase in false positives. Likewise, the following rule, designed to detect the attempted evasion of ID systems, finds some of the satan, portsweep, and

ipsweep activity with a few thousand false positives. The remaining three rules all find a small number of the probe connections with a number of false positives due to legitimate use of the services they are designed to detect activity against.

While Snort's performance does not look particularly impressive when graphed out, it succeeds in detecting part of every probe, which is typically sufficient. In fact, logging every connection associated with a probe will likely only serve to fill up a system's log files. What was troubling is that it only detected the ipsweep scans using a rule that fires on all ping activity, which generated a huge number of false positives on the DARPA data, and which we would expect to also happen in a real environment.

### 3.4 Remote To Local

The primary intent of Snort is to detect intrusions that happen over the network. As such, we would hope that it does well in the Remote To Local (R2L) category, especially given the time elapsed between the evaluation and the ruleset used, which should have allowed plenty of time to fine-tune the detection signatures. The results are shown by the line with empty circles in figure 2.

Indeed, we see that with just six rules we detect well over 10% of the malicious connections with a negligible number of false positives, and another five rules detect well over half the attacks; however, the false positive rate is unacceptably high. Notably, unlike the previous categories, the true detection versus false positive rate did not cumulatively exceed the 50% guess rate, although it came close.

Only one attack, phf, was perfectly detected with no false positives. Two of the attacks, dict and guest, were almost perfectly detected by rules designed to detect failed login attempts; yet these also produced a reasonable number of false positives, as we would expect them to in a real environment. We wouldn't expect Snort to do better on the dict attacks without a preprocessor, and while a rule could be written to detect the usage of guest accounts, this is probably better handled by a host-based detector. Both the spy and warez attacks were perfectly detected by rules designed to detect policy violations: the use of telnet and anonymous ftp servers, and given their legitimate use in the dataset, both rules produced a large number of false positives.

The attacks that were not detected well (had a low true positive rate) were ftp-write, warezclient, and warezmaster. Some of these instances were detected by the afore mentioned policy rules; however most of the connections that were detected, were detected by rules specifically written for these attacks, hence they had a low number of false positives. Generally, these attacks are difficult to write rules for as they don't break anything per se, but rather abuse legitimate services and use them in unintentional ways. The remaining R2L attack, multihop, was detected twice by a rule designed to detect many different types of attacks with only a single false positive, and once by the general rule for telnet access with the very high number of false positives, leaving six of its instances undetected.

### 3.5 User To Root

Generally, a User To Root (U2R) attack entails actions that happen solely on a local machine to elevate a user's privileges to that of the superuser. As such, the best way to detect them is via a host-based detector which may use inputs such as the syscall trace, changes to the filesystem, and process table. While such data was included as part of the eval, since Snort is a network-based IDS, it did not take advantage of these resources. Such attacks may be launched via a remote terminal session, as all apparently were for the eval (meaning it wasn't concerned with insider threats). If such a session is encrypted, then our primary hope of flagging it as intrusive comes from machine learning recognizing that the attributes of the connection (such as the source, time, packet interarrival times, etc.) is anomalous. If the connection is not encrypted, such as the Telnet sessions in the DARPA data, then we have some hope of detecting the attack based on a signature, such as the commands typed, or the output produced. Such signatures tend to be fragile, and we did not have high expectations for Snort's performance in this area.

As shown by the line with solid circles in figure 2, a small number of rules (five) were able to detect some effects of some of the U2R attacks, particularly rootkit, imap, ffb, and loadmodule with only a few false positives. Interestingly, the bulk of this initial spike is from a rule to detect BSD-style ICMP ping traffic, which detected 89% of the 254 rootkit attacks with only four false positives. This indicates that the particular rootkit used in the eval has its own ping packaged with it that was typically used in the execution of those attacks. Most of the rest of this initial spike was from rules designed explicitly to detect attack code and responses in terminal traffic, which interestingly produced five false positives. At the beginning of the slope into higher false positives, a couple of rules detected odd network traffic (failed logins and SYN-FIN packets) during a couple of the attacks, with a significant level of false positives. Most of the remaining attacks were detected with the policy rule to detect all telnet traffic which has significantly more false positives on the test network than the number of attacks detected. Finally, a few remaining rootkit instances were detected with the afore mentioned (in the section on DoS attacks) rule that fires on all "Port Unreachable" packets. With a well tuned Snort ruleset (meaning one that has had rules that produce high rates of false positives removed) on a contemporary network (where most terminal sessions are encrypted), we would only expect to detect imap – which could also be classified as a remote to local attack – with any reliability.

## 4 Conclusion

While we initially thought that Snort would perform well on the DARPA IDS evaluation dataset, we found that it did not. Upon discovering this empirical result, we thought that it may be due to a failure in the dataset to model the attacks correctly, or perhaps because the attacks were so old that Snort no

longer included the rules to detect them.

What we found instead was that the dataset only includes a very limited number of attacks that are detectable with a fixed signature. The majority of malicious connections in the DARPA dataset come from denial of service attacks and probing activity. While Snort has some capability to detect such attacks, such detections have not been the primary focus of its design. Interestingly, many of the advanced ID systems that were evaluated either as part of the DARPA evaluation, or that evaluated themselves using that data, fared very well on the DoS and probe attacks, whereas many had trouble with the low connection rate remote to local, and user to root attacks, where Snort performs best. As such, we would not endorse changing Snort to detect some of these other attacks, but rather use Snort in conjunction with another NID designed for such purposes. Snort still had trouble detecting misuse of resources, such as the various warez attacks, where we believe an anomaly based detector may fare better. While Snort performed impressively on the user to root attacks (although one must consider that many were detected solely because a telnet connection was used), we suspect that, given the widespread adoption of encrypted connections, that the days of NIDs detecting such attacks are numbered, and that a host-based intrusion detection approach is necessary.

Given that Snort detected most of the attacks that we would expect a signature-based detector to, and that many other attacks were detected through more erroneous means – policy rule violations, for example – we feel more comfortable now that the DARPA IDS evaluation dataset properly models attacks, and can serve to help evaluate the true positive performance of a network intrusion detector for the types of attacks that it includes. We did not, however, get the same impression of its usefulness in evaluating the false positive performance. As McHugh observed, no evidence has been provided that the DARPA dataset accurately models a real network. In fact, casual observation shows that the datarate of the provided network data is far below that of even a trivial network used for business (as opposed to casual home users). This was actually an issue in our experiment, as apparently all, or at least a very significant number, of the images transferred as part of their artificial HTTP activity were 0x0 sized images, which Snort kept alerting about, because these are typically used as “web-bugs” to track the surfing habits of users across the WWW. We discarded that rule as it didn’t provide any true positives; yet it illustrates how the artificial nature of the dataset prevents an objective evaluation of the false positives produced by an IDS. Conversely, we wonder how accurate the low-number of false positives with the BSD-style ping rule that detected most of the rootkit instances was. Generally speaking, we do not believe that the false positive rates reported herein have any bearing on Snort’s false positive performance in a real environment, which may be much worse, or much better.

In conclusion, we believe that any sufficiently advanced IDS should be able to achieve good true positive detection performance on the DARPA IDS evaluation dataset. Demonstrating such performance, however, is only necessary to show the capabilities of such a detector, *it is not sufficient*. In particular the intrusion detection community needs a more realistic dataset to evaluate the false positive

rate of such systems. Further, there are many new threats that have emerged in the five years since the last evaluation (botnets, spyware, flash worms, etc), that a contemporary IDS needs to be able to detect, and which should be tested for in such a dataset. Additionally, a better dataset should include a realistic number of attacks which could be easily detected by a signature-based detector. How such a dataset can be produced and validated is an open research problem. We believe that even when these more advanced network ID systems are deployed, signature-based NIDs, such as Snort, will continue to play an important role in enterprise security architectures.

## 5 Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

## 6 Acknowledgments

Some of the work in this paper was performed under the auspices of the United States Department of Energy by Lawrence Livermore National Lab under contract number W-7405-ENG-48. The authors wish to thank Dr. Berger and the DOE Office of University Partnerships, who provided funding for Mr. Chow through the College Cyber Defenders portion of the Laboratory Critical Skills Development Program. We would also like to thank Dr. Matt Bishop for his feedback on this paper.

## References

- Caswell, B. and M. Roesch (2004, 16 May). Snort: The open source network intrusion detection system. <http://www.snort.org/>.
- Lippmann, R. P. and R. K. Cunningham (1999, 7–9 September). Improving intrusion detection performance using keyword selection and neural networks. In *Proc. of the Conf. on Recent Advances in Intrusion Detection (RAID) '99*, West Lafayette, IN.

- Lippmann, R. P., D. J. Fried, I. Graf, J. W. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. K. Cunningham, and M. Zissman (2000, January). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *Proc. of the DARPA Information Survivability Conference and Exposition*, Los Alamitos, CA. IEEE Computer Society Press.
- Lippmann, R. P., J. W. Haines, D. J. Fried, J. Korba, and K. J. Das (2000, October). The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks* 34, 579–595.
- Mahoney, M. V. and P. K. Chan (2003, 8–10 September). An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In G. Vigna, E. Jonsson, and C. Krügel (Eds.), *Proc. 6th Intl. Symp. on Recent Advances in Intrusion Detection (RAID 2003)*, Volume 2820 of *Lecture Notes in Computer Science*, Pittsburgh, PA, pp. 220–237. Springer.
- McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans. Information System Security* 3(4), 262–294.
- Pickering, K. J. (2002, March). Evaluating the viability of intrusion detection system benchmarking. University of Virginia Undergraduate Thesis.
- RTI International (2005). PREDICT: Protected REpository for the Defense of Infrastructure against Cyber Threats. <http://www.predict.org/>.

## Appendix: Snort rule performance

The following five tables show the cumulative true and false positive counts for the Snort rules that detected at least one attack that no other rule did. The rules are ordered based on the true positive to false positive ratio for each rule, eliminating true positives for attacks which were already matched. The five tables present the rule performance for all attacks, and for each of the four types of attacks.

Table 1: Cumulative results by Snort rule. Each attack is detected at most once. Rules are sorted in descending order of TP:FP ratio.

Snort Rule	True Positives	False Positives
[113:1:1] (spp_frag2) Oversized fragment, probable DoS	10138	0
[1:1420:11] SNMP trap tcp	11643	0
[1:1418:11] SNMP request tcp	13147	0
[1:1421:11] SNMP AgentX/tcp request	14648	0
[1:1445:5] POLICY FTP file_id.diz access possible warez site	14792	0
[1:613:6] SCAN myscan	14852	0
[1:504:7] MISC source port 53 to <1024	14872	0
[1:598:12] RPC portmap listing TCP 111	14878	0
[1:332:8] FINGER 0 query	14883	0
[1:323:5] FINGER root query	14888	0
[1:1147:7] WEB-MISC cat access	14893	0
[1:330:9] FINGER redirection attempt	14898	0
[1:584:11] RPC portmap rusers request UDP	14902	0
[1:503:7] MISC Source Port 20 to <1024	14906	0
[1:612:6] RPC rusers query UDP	14910	0
[1:335:5] FTP .rhosts	14912	0
[1:359:5] FTP satan scan	14913	0
[1:489:7] INFO FTP no password	14914	0
[1:547:6] POLICY FTP 'MKD ' possible warez site	14915	0
[1:621:7] SCAN FIN	16223	1
[1:546:6] POLICY FTP 'CWD ' possible warez site	16512	2
[1:1156:9] WEB-MISC apache directory disclosure attempt	18793	40
[1:368:6] ICMP PING BSDtype	19019	44
[1:1251:6] INFO TELNET Bad Login	19937	132
[1:1882:10] ATTACK-RESPONSES id check returned userid	19946	133
[1:527:8] BAD-TRAFFIC same SRC/DST	19981	140
[111:2:1] (spp_stream4) possible EVASIVE RST detection	25526	2142
[1:384:5] ICMP PING	36030	7780
[1:652:9] SHELLCODE Linux shellcode	36032	7782
[1:498:6] ATTACK-RESPONSES id check returned root	36034	7784
[1:1417:9] SNMP request udp	36035	7787
[1:1419:9] SNMP trap udp	36036	7790
[1:716:12] TELNET access	36097	10829
[1:624:7] SCAN SYN FIN	36099	10962
[1:402:7] ICMP Destination Unreachable Port Unreachable	36107	12089
[1:553:7] POLICY FTP anonymous login attempt	36108	14057

Table 2: Cumulative results on denial of service attacks by Snort rule. Each attack is detected at most once. Rules are sorted in descending order of TP:FP ratio.

Snort Rule	True Positives	False Positives
[113:1:1] (spp_frag2) Oversized fragment, probable DoS	10133	0
[1:1420:11] SNMP trap tcp	11624	0
[1:1418:11] SNMP request tcp	13114	0
[1:1421:11] SNMP AgentX/tcp request	14604	0
[1:613:6] SCAN myscan	14664	0
[1:504:7] MISC source port 53 to <1024	14684	0
[1:503:7] MISC Source Port 20 to <1024	14688	0
[1:1156:9] WEB-MISC apache directory disclosure attempt	16969	38
[1:527:8] BAD-TRAFFIC same SRC/DST	17004	45
[111:2:1] (spp_stream4) possible EVASIVE RST detection	21604	2047
[1:402:7] ICMP Destination Unreachable Port Unreachable	21609	3174

Table 3: Cumulative results on probes by Snort rule. Each attack is detected at most once. Rules are sorted in descending order of TP:FP ratio.

Snort Rule	True Positives	False Positives
[1:1418:11] SNMP request tcp	14	0
[1:1420:11] SNMP trap tcp	28	0
[1:1421:11] SNMP AgentX/tcp request	39	0
[1:598:12] RPC portmap listing TCP 111	45	0
[1:323:5] FINGER root query	50	0
[1:330:9] FINGER redirection attempt	55	0
[1:332:8] FINGER 0 query	60	0
[1:584:11] RPC portmap rusers request UDP	64	0
[1:612:6] RPC rusers query UDP	68	0
[1:359:5] FTP satan scan	69	0
[1:621:7] SCAN FIN	1377	1
[1:384:5] ICMP PING	11881	5639
[111:2:1] (spp_stream4) possible EVASIVE RST detection	12819	7641
[1:1417:9] SNMP request udp	12820	7644
[1:1419:9] SNMP trap udp	12821	7647
[1:716:12] TELNET access	12832	10686

Table 4: Cumulative results on remote to local attacks by Snort rule. Each attack is detected at most once. Rules are sorted in descending order of TP:FP ratio.

Snort Rule	True Positives	False Positives
[1:1445:5] POLICY FTP file_id.diz access possible warez site	144	0
[1:1122:5] WEB-MISC /etc/passwd	149	0
[113:1:1] (spp_frag2) Oversized fragment, probable DoS	154	0
[1:335:5] FTP .rhosts	156	0
[1:547:6] POLICY FTP 'MKD ' possible warez site	157	0
[1:546:6] POLICY FTP 'CWD ' possible warez site	446	1
[1:1251:6] INFO TELNET Bad Login	1362	89
[1:1882:10] ATTACK-RESPONSES id check returned userid	1366	90
[1:716:12] TELNET access	1385	3129
[1:553:7] POLICY FTP anonymous login attempt	1386	5097
[111:2:1] (spp_stream4) possible EVASIVE RST detection	1387	7099

Table 5: Cumulative results on user to root attacks by Snort rule. Each attack is detected at most once. Rules are sorted in descending order of TP:FP ratio.

Snort Rule	True Positives	False Positives
[1:489:7] INFO FTP no password	1	0
[1:368:6] ICMP PING BSDtype	227	4
[1:1882:10] ATTACK-RESPONSES id check returned userid	232	5
[1:652:9] SHELLCODE Linux shellcode	234	7
[1:498:6] ATTACK-RESPONSES id check returned root	236	9
[1:1251:6] INFO TELNET Bad Login	238	97
[1:624:7] SCAN SYN FIN	240	230
[1:716:12] TELNET access	268	3269
[1:402:7] ICMP Destination Unreachable Port Unreachable	271	4396

## Appendix: Attack detection performance

The following table summarizes the true positive and false negative performance for all of the connections that were part of an attack. The true positives are the number of attack connections that Snort alerted on with at least one of the above rules. The false negatives are the number of attack connections that Snort failed to alert on.

Table 6: True positive and false negative performance per attack type

Attack Type	True Positives	False Negatives
anomaly	9	0
back	2281	0
dict	881	0
dict_simple	1	0
eject	11	0
eject-fail	1	0
ffb	5	5
ffb_clear	1	0
format	1	5
format-fail	1	0
format_clear	1	0
ftp-write	4	4
guest	50	0
imap	5	3
ipsweep	9405	6931
land	35	0
loadmodule	3	8
multihop	3	6
neptune	9155	1517488
nmap	2100	257
perl_clear	1	0
perlmagic	4	0
phf	5	0
pod	10133	365
portsweep	1147	9470
rootkit	237	17
satan	180	32452
smurf	5	250128
spy	2	0
warez	1	0
warezclient	439	1327
warezmaster	1	18