Flaws and frauds in the evaluation of IDS/IPS technologies

Stefano Zanero
DEI - Politecnico di Milano
via Ponzio 34/5
20133 Milano Italy

Abstract

In this paper, we will briefly review the problem of IDS and IPS performance evaluation, demonstrating how flawed current approaches to evaluation are, both in academia and in the industry. We will show all the issues in current testing methodologies, as well as key reasons to distrust claimed performance rates of current IDS systems. We will also show how practical testing architectures can be created to compare systems, and how they can be used in academic and industrial evaluations.

1 Introduction

A well known problem in the art of war is the fact that the defender needs to plan for everything, while the attacker just needs to hit one weak spot. A reformulation in terms of computer security is in [1]:

> [the] philosophy of protection [...] depends upon systems to: behave predictably (they should do what we think they will do); be available when we need them; be safe (they should not do what we don't want them to do); be capable of protecting our data from unwanted disclosure,modification,and destruction; respond quickly. In other words, systems should be trustworthy".

Significantly, the title of the cited work is "Fortresses built upon sand". Almost none of these conditions are respected by real world system. This means that we must realistically consider information systems as being inherently insecure: software and hardware are not trustworthy, and people are willingly or unwillingly violating security policies. Furthermore, policy specifications can be incorrect, or incomplete, or incorrectly implemented.

As one of Murphy's laws would have it: "The only difference between systems that can fail and systems that cannot possibly fail is that, when the latter actually fail, they fail in a totally devastating and unforeseen manner that is usually also impossible to repair". The lesson that everybody learns through practice, if any, is that every defensive system will, at some time, fail, so we must plan for failure. As we plan for disaster recovery and continuity, because disasters will happen at some point, we must design systems to withstand attacks, and fail gracefully. We must design them in a way which makes it possible to recover them from attacks without losing data.

But even more importantly, any secure information system must be designed for being tamper-evident, because when it will be broken into, we want to be able to detect the intrusion attempt, in order to react to it. Since tamper evidence is all but a natural property of a computer system, we call Intrusion Detection Systems all the systems that can detect intrusion attempts, and possibly assist in post-attack forensics and recovery. The concept of a system capable of detecting intrusions was introduced in 1980 by J.P. Anderson [2].

The idea behind any type of IDS is that any information system is designed to serve some goals, and the three properties of security are aimed to ensure that the information system is not abused to do something else. Thus, when someone willingly violates the security paradigm of an information system, his behavior and/or the behavior of the system will somehow differ from the "normal" behavior. Ideally an IDS would detect these behavioral anomalies and tag them as suspicious.

Testing such systems is intrinsically difficult, as we will show in the following. After establishing in Section 2 the basic motivations for performing such testing, in Section 3 we will try to define various possible metrics for evaluating the performance of an IDS, and to show their pitfalls. Section 4 will outline the difficulties in generating proper workloads for such tests. Section 5 will instead deal with the even more complex issue of testing distributed IDS systems or correlation engines. In Section 6 we will use the well known DARPA IDEVAL dataset as a case study in how even the best and most motivated efforts can lead to results of a questionable quality. Finally, in Section 7 we will draw our conclusions and outline future work in this area.

2 Establishing the Motivations for Testing

Testing is a word which means different things to different people. Basically, testing aims to answer two different questions:

1. Does it work ?
2. How well does it work ?

The first question is the most important and crucial one in any engineering task. As Bob Colwell would have it, "if you didn't test it, it doesn't work" [3]. However, the depths entailed by proper testing are often forgotten, and invariably lead to disastrous results. If you test something, but you don't test it accurately, it still doesn't work, it's just pretending to.

On the other hand, as scientists and engineers we are also interested in seeing how well something works, either objectively, or subjectively ("how well does it fit my own perceived needs ?"). Also, the answer may be comparative (among n different systems), or absolute (on a specific scale, usually a metric of some sort). This is the point where the idea of testing that a customer has diverges from the conception a scientist has. Whereas a customer wants a subjective answer to the "how well" question, most often a comparative one (especially when concerned with buy decisions). Scientists, on the other hand, should answer the "how well" question in an objective, repeatable way. This may be relative or absolute (even if most scientific tests aim for an absolute outcome, in order to allow future studies to easily compare), but definitely should be standardized and unbiased as much as possible.

Misuse Based	**Anomaly Based**
Require continuous updates	Do not require updates
No initial training	Long and complex training
Need tuning	Tuning included in training
Cannot detect new attacks	Can detect new attacks
Precise alerts	Vague alerts
Almost no false positives	Huge numbers of false positives
Lots of non contextual alerts	No non contextual alerts
Easier to design	More difficult to design

Table 1: Comparison between strengths and weaknesses of anomaly based and misuse based IDSs

So, scientific and industrial testing should be based on rationally motivated, open, disclosed, unbiased standards. And an unbiased testing standard needs to be grounded in the definition of requirements and metrics for the evaluation subjects. In particular, requirements usually pertain to the "does it work" assessment, while metrics are usually what is needed for comparison and performance assessment, which is the core subject of this paper.

3 IDS requirements, performance metrics and fallacies thereof

It sounds strange to be writing a section like this for a class of applications that has been devised almost 30 years ago. However, it is really noteworthy that requirements and metrics for IDS are not very well defined, neither in literature, nor on the market (and this may very well be the core reason for all the confusion surrounding the performance evaluation issue).

First of all, we need to recall that Intrusion Detection Systems can be broadly divided in two main categories, based on two different approaches: anomaly detection or misuse detection. An anomaly detection IDS tries to create a model of normal behavior for the monitored system(s) or for their users, and flags as suspicious any deviation from this "normal" behavior which exceeds carefully tuned thresholds. A misuse detection IDS, symmetrically, uses a knowledge base (often called a set of *signatures*) in order to recognize directly the intrusion attempts, which means that instead of trying to describe the normal behavior of a system it tries to describe the anomalous behaviors. We will not, due to space constraints, analyze the differences between the two approaches. A summary can be found in Table 1.

Another distinction can be drawn on the base of the source of data being audited by the intrusion detection system, between *network* based and *host* based systems. A host based IDS controls a single machine, sometimes even a single application, and depends on data which can be traced by the operating system of the monitored host, e.g. system calls, resources usage, privilege escalations, and/or system logs. A network based IDS is connected to a network segment and tries to analyze all the traffic which flows through the segment (usually, by the means of a network sniffer), trying to detect packets which could be part of an attack.

For the sake of simplicity, in the following we will use terminology and examples related to Network IDSs, but everything can be translated to apply to host-based systems as well. We will, instead, carefully make distinctions between testing methodologies that apply to misuse based systems, to host based systems, or to both.

3.1 False positives and negatives

The first, naïve requirement for an intrusion detector is that it should alert on intrusion attempts. Some researchers contend that this should be extended as "successful" intrusion attempts, but we will stick to the simplified version throughout this text. We define the following quantities:

TP True Positives, alerts raised for real intrusion attempts;

FP False Positives, alerts raised on non-intrusive behaviors;

TN True Negatives, no alerts raised and no intrusion attempts present;

FN False Negatives, no alerts raised when real intrusion attempts present.

False positives are the bane of intrusion detection systems, because after a while an error-prone system is just ignored and not used anymore. Anomaly detection systems are particularly prone to false positives, while signature based systems usually do not have any meaningful rate of false positives (provided that properly tested rules are deployed). They rather have non-contextual alerts, which means true positives on attacks that are not deemed interesting, since they are targeting a non-vulnerable platform. The so called target-based architectures [4] try to reduce this problem, but this gets back to the idea of "successful or unsuccessful attacks" we cited above, so we omit it from our evaluation. In some cases it is difficult however to define what a "false positive" really is. For instance, if you send a sufficiently strange, but not malicious, packet to a good anomaly detector, it will likely be flagged. Is this really a "false positive" for such a system? True, it is not an attack, but surely it is anomalous.

False negatives are obviously also a problem. In particular, for misuse based systems, most new attacks will generate false negatives, unless they are very similar to an existing attack.

Typically, two metrics are then defined. Detection Rate, measuring how many attacks are detected overall:

$$DR = \frac{TP}{TP + FN}$$

and the False Positive Rate, measuring how many alerts are false:

$$FR = \frac{FP}{TN + FP}$$

It is easy to see that the DR is equivalent to the "recall" rate in information retrieval systems, while the FP rate is somehow the inverse of the concept of "precision".

Intuitively, in an anomaly detector, these two variables are bound by a trade off: the more sensitive a system is, the more false positives it generates, but the

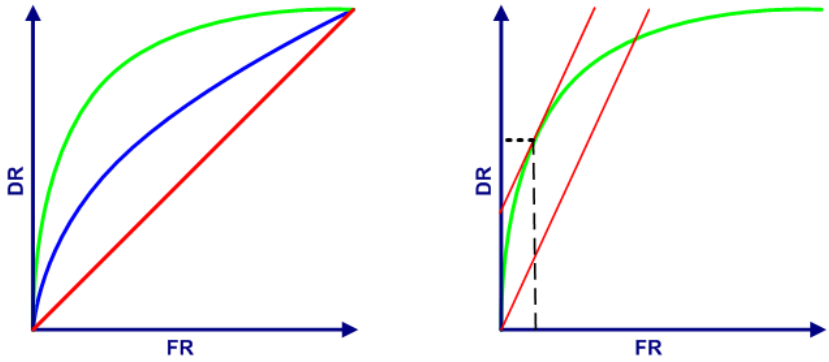


Figure 1: Examples of ROC curves

higher the detection rate is. If sensitivity is a variable, s, then $FR = FR(s)$ and $DR = DR(s)$. Therefore we can represent the two quantities as a parametric curve, which is named ROC, Receiver Operating Characteristic, in radar and signal analysis literature.

In Figure 1 we have traced some imaginary ROC curves. Axes are obviously scaled from 0 to 1. An IDS which does not generate any alert has $DR = 0$ and $FR = 0$ since $TP = FP = 0$, while an IDS which flags anything has $DR = 1$ and $FR = 1$ (since $FN = TN = 0$). Among these extremes, any behavior can happen.

In general ROC curves are monotonous non decreasing and above the bisectrix. Intuitively, the larger the area below the curve, the better the detection to false alert ratio is. But this definition is scarcely operative (needing a point-by-point analysis to trace and interpolate the curve). Additionally, this global dominance criterion is not always valid: the costs we associate to a false positive or a false negative is generally different, and subjective (depending on the network size, number of analysts, and so on). Let us call α the cost of a false positive, and β the cost of a false negative, and p the ratio of positive events on the total ($\frac{FP+TP}{FP+TP+FN+TN}$). We can write the cost function:

$$C = FR\alpha(1-p) + (1-DR)\beta p$$

The gradient of this cost function is a line with coefficient:

$$\frac{\alpha(1-p)}{\beta p}$$

If we trace it in the ROC diagram (Figure 1 on the right), we can determine the minimum cost point on the ROC curve, and thus a satisfying sensitivity value. In this locality, an algorithm that globally performs worse could be a better choice. This type of analysis, of course, is extremely subjective (since the weights depend on the type of organization, on its security objectives and so on).

Another huge problem is that this model basically makes no sense for misuse detectors. In this case, most of the results depend on the ruleset. So, are we benchmarking the ruleset, or the engine? For most commercial IDSs this may make little sense, but for open source projects and academic evaluation the

difference is subtle but important. Also, do False Negatives actually make sense for these systems? If we use a new attack, or an attack not covered by the rules, this will almost surely result in a false negative. But this is expected, so what content of information do we gain from this?

Basically, we suggest that *evaluating misuse detectors on the base of false negatives and false positives is impossible*, in a general sense. It is just not fair, neither indicative of their properties.

3.2 Evaluating Resistance to Polymorphism

Computer attacks are polymorph. We are not referring to the same meaning of polymorphism as in "polymorphic virus" or "polymorphic shellcode". Viruses (and shellcode) are polymorph if they can somehow mutate themselves in order to avoid or delay detection [5–7]. But in this case we are talking of self-mutating algorithms, which can be (in most cases) still captured and detected [8].

Computer attacks are as mutable as the attacker is ingenuous. They can be performed in a number of different ways. Signatures should strive to catch the core exploitation steps of a vulnerability, but this is not granted: in fact, many products base their signatures on the currently available exploits. There is also a significant range of well known evasion techniques [9–11] which should be either directly detected, or avoided through proper canonicalization and reconstruction of the input. In order to evaluate the resistance of an IDS to such techniques, in [12] the authors propose a clever tool (named "Sploit") to generate mutants of an exploit by applying a number of mutation operators and evasion techniques at the network, session and application protocol layers. The mutated exploits are then ran against a target machine to verify they are still effective, and used to verify the resistance of two NIDS against the mutation operators employed. Sploit is an excellent idea and much more complete than earlier efforts (Thor (R. Marty), much more limited AGENT (Rubin, Jha, and Miller): formal, based on logical induction for mutations), but still has some limitations: firstly, it still tests engine and signatures together; in second place, it's qualitative more than quantitative, and as of now, comparative rather than absolute.This makes it suitable, in order to strike the point that IDSs still suffer from combinations of evasion attacks. On the other hand, using it for real comparative evaluations of IDS systems is not viable.

3.3 Evaluating Coverage

A possible metric, particularly attractive for misuse based systems, is the coverage of the rulebase. There are evaluations (such as the one of ICSA Labs) which measure the coverage of anti-virus software, so it seems reasonable to extend them to the coverage of misuse based IDS [13]. However, this meets two very hard problems: firstly, while we have rather good zoo and wild virus lists and samples, we do not have any good vulnerability list, let alone a reliable wild exploits list or database.

This means that (once more) we cannot deliver an absolute measure of coverage, but we can still perform relative coverage analysis, in two different ways: we can do it offline, by simply counting the signatures of each system after a normalization step (because some systems may use a different number of signatures to recognize the same attack). This normalization step is not simple to do,

nor vendors are usually keen on disclosing their signature details. This process is also unsuitable for anomaly detectors. If on the other hand we choose to run a set of attacks and test the systems live, we will still meet all the difficulties we discuss in the other parts of this paper.

3.4 Evaluating Performance

In a stricter meaning of "performance evaluation" we are also concerned with measuring "how fast" an IDS is. Once again, this metric opens up a real can of worms. In fact, we can measure at least two different values: the *throughput* of the system, or the *latency* the system inserts. We need not recall the basics of queueing theory here, but it is sufficient to remind that while linked, these two values are not the same thing, nor they express the same characteristic for a system. In the case of an IDS system, we are more interested in the first one, while in the case of an IPS latency of course takes prevalence. Also, it may be noteworthy that there's a heated debate [14] on whether packets per second or bytes per second are the appropriate metric unit for throughput of such systems.

In most simple performance evaluations, systems are loaded by implicitly supposing they will behave like an M/M/1 system, that is, they will be subject to a Poisson arrival process, with an exponentially distributed service time. In such conditions it is simple to measure the latency and the throughput. Fact is, real systems, and particularly network IDSs, rarely behave as *infinite* queues. Rather, they act like systems with a finite capacity queue [15], which means they have a buffer which can store a finite number of packets c, and when this buffer fills up, further packets are discarded rather than processed. In evaluating a Network Intrusion Detector which works as a sniffer, the rejection rate is extremely important. Under the hypothesis that the system behaves as M/M/1/c, this rate can be statistically computed.

In fact, we ran a number of tests on Snort and on the Cisco IDS systems, using a stateless generator (a small tool named "blabla", which can generate up to 50kps and 100Mbps of traffic following various distributions with great accuracy). Both systems behave on stateless packets as M/M/1/c systems, and as expected the c parameter turns out to be a trade-off handle between the number of discarded packets and congestion (i.e. waiting time for processing). This is an important consideration to keep in mind when moving, e.g., a Snort box in-line for intrusion prevention purposes.

However, we must remind that modeling a NIDS in this way *still is a simplification*. In fact, the arrival process on a real network can be very different from a Poisson one, but most importantly the system could be *load dependent*. In fact, since modern NIDSs are *stateful* systems, we expected them to behave differently depending e.g. on the number of concurrent connections, hosts and protocols they are handling. Using stateful traffic generation, and keeping dangling and open connections, clearly shows a load dependency, which however we have not yet studied in depth and which will be the subject of future extensions of this work.

Another important lessons learned from queueing theory is that traffic distribution matters, and it particularly impacts the rejection rate, i.e. different distributions of packet arrivals with the same *average* arrival rate can lead to dramatically different results in terms of dropped packets. We have also verified that types of packets and their matching with rules heavily impacts on service

times. This means that all these things should be carefully documented in tests, and that the alert reader should beware of test results that do not clearly state these parameters.

3.5 Peculiarities of Intrusion Prevention Systems

Evaluating Intrusion Prevention System is not a simple extension of the evaluation of IDS. While it is true that IPS are "nothing more than IDS that can block actions in addition to flag them", this change in aims and goals creates very different and specific evaluation metrics [16].

As we already noted, for instance, while a NIDS just needed to have a sufficient throughput which avoided to drop packets, a NIPS, acting as a gateway very much in the same way a firewall would, must be designed not to introduce unnecessary latency: therefore, the *response time* becomes important.

Denial of service is also an issue with these systems: if a reactive network IDS blocks services based on detected attacks, a spoofed attack packet could be enough to block legitimate connections. False positives may also create denial of service conditions against legitimate users. Because of this, even in the case of systems that can act both as IPS and as IDS (see Snort) the configuration will be extremely different: in a reactive system we will try to lower the FPR as much as possible, not really caring about the DR.

3.6 Unmeasurable or qualitative indicators

A number of interesting indicators of the qualities of an IDS are not measurable, or qualitative. For instance, the *comprehensiveness* of the model used by the IDS indicates how likely it is that a new class of attack will be completely outside the scope of what the IDS looks at. For instance, a sensor operating at layer 3 of the ISO/OSI stack will never detect attacks at layer 2. The engine itself must be rewritten or patched to go below the original layers. Anomaly detection and misuse detection systems suffer of this problem in a similar manner. If an attack shows up only in the variables that an anomaly detection system does not measure, then the IDS is blind to it. It is easy to imagine forms of attack specifically studied to find and exploit these "dead spots". An interesting example is in [17].

An IDS should be designed to be as secure as possible, and also designed for survivability, since it has a similar function as an aircraft emergency flight recorder. Since the IDS logs can quickly become the only valuable source of information on a security breach, it is important that the IDS system itself is not compromised. Subversion of an IDS can disable the intrusion alerts, generating a false sense of security, and can lead to irreversible alterations in logs and traces. A distributed IDS is also vulnerable to common attacks against communication between components, and must adopt all the techniques to ensure end-to-end communication security and confidentiality. However, the intrinsic security of a system cannot be measured meaningfully, and in fact it is the subject of a whole, separate research area.

Other typical indicators of quality that cannot be measured easily are usability, flexibility, ease of deployment.

4 Generating realistic workloads for IDS evaluation

It is extremely difficult to generate a reliable, realistic workload suitable for IDS evaluation, and this should already be partially clear, given all the considerations in the previous Section.

The basic idea seems deceivingly simple: it is enough to generate a *realistic background traffic* and then superimpose to this traffic a set of *attacks*. Then we can feed the generated test data to any IDS, compare the alerts with the attacks, that are perfectly known (i.e. with a *truth file*), and simply count false positives and false negatives. We will now carefully explore both components of the workload to outline their quirks and most common fallacies in their generation.

4.1 Background traffic

We have basically two options for generating background traffic. Either we use real data, or we artificially generate some. Real data grants realism, obviously, but it is nor repeatable (because for privacy issues, usually, such data cannot be disclosed publicly), nor standard (real data coming from a university network is so different from data coming from a corporate network that it is wise to doubt that one can be used as a simulation of the other). In other words, this data is excellent for relative, in-house evaluations; but useless for absolute or scientific evaluations. Also, nobody can grant that this traffic is attack-free: this means that our truth file will not be complete.

The only way to make them suitable for publication is sanitization, which however may introduce statistical biases (e.g. it may change character distribution in sanitized packets). This can bias anomaly detectors evaluation. On the other hand, the less we sanitize, the riskier everything gets for privacy. Also, network peculiarities (e.g. a reduced number of protocols, or limited variability in contents) could overfit training of anomaly detectors and positively bias detection rate.

Using a TCP replayer (such as the "tcpreplay" utility) is not as simple as it may seem. At high network speeds, buffer size issues require the use of more than one replay interface, with problems of synchronization.Also, packet timestamps may not be accurate enough to fake a realistic stateful communication.

Artificially generated traffic is a likewise difficult path. Noise generators can be used: stateless load generators that create more or less random packets. However, since intrusion detection algorithms depend heavily on content of the packets, concurrent session impact, and so on (as we saw in Section 3.4), this approach is unlikely to lead to sensible results.

A more complex approach is to use artificially generated data that can simulate the interaction of users over a testbed network. This approach was taken, for instance, by DARPA, as we will see more in detail in the following 6. This is a good way to create a repeatable, scientific test on solid ground. In that order, at the Performance Evaluation Lab of the Politecnico di Milano we are working on a set of scriptable, GPL'ed traffic generators that can emulate users and generate traffic with statistically sound distributions; they are designed to be distributed in order to use multiple machines to generate a higher load, and we are also integrating them with scriptable stub servers. The idea is, following

Puketza et al. [18], that a FOSS testbed can be distributed and (provided that the parameters and experimental setup are described) it will create a repeatable experiment. This type of approach also solves other well-known shortcomings correlated with the use of datasets: the high dimensions of datasets for high speed networks and the problems of timestamp accuracy we already described above.

On the other hand, we have an hard time defining what "normal background traffic" would be composed of. Various types of measurements have been performed [19, 20], and they have similar results: TCP is predominating (up to 95% of the total bytes, 85% to 90% of the total number of packets); UDP is the rest, with ICMP being a residual 1-2%. HTTP is the dominant application layer protocol (looking at the packets 70% for CAIDA, over 60% for Cisco, and 65% in our environment), but in a slightly decreasing trend, while DNS and SMTP (5-8% each) account for most of the rest of Internet traffic. on "general" networks, gaming and peer-to-peer traffic can reach 10% of the total packets. Average packet size is 570 byte, many full-size packets and many small packets are present. This gave origin to the so-called IMIX (Internet packet mix, or 7:4:1 distribution): seven 64B packets, four 570B packets and one 1518B packet, tipically used in load generators. But we don't just need a mix of packets of the correct sizes (as if we were testing a router): we also have established that there is a dependency against the number of open connections.

Also the network infrastructure used for traffic generation is important. for instance, usually traffic generators, attack network and victim network are all connected to a switch, or to multiple switches. The port capacity of the sniffing port could limit the IDS: on a gigabit Ethernet port, inter-packet arrival gap is 96 ns. If multiple Fast Ethernet ports, generating traffic at their peak (80Mbps), are used, multiple frames will surely happen in a 96 ns bucket: this means that the port buffer will fill up to the point that the switch itself will begin to drop packets.

Also, if we generate traffic, we will end up with "perfect" traffic which contains no anomalous or broken packets, so common in the Internet nowadays [21].

The quality of background traffic, already important for proper evaluation of misuse detectors, is of paramount importance in the case of evaluation of anomaly detectors. The realism of the training traffic (which will be constituted purely by "background" traffic) will strongly influence and bias the measured detection rate and false positive rate. Basically, the more similar the training traffic is to the test traffic, the easier the job for an anomaly detector: we must strive to ensure that this is still a fair test, compared with real world conditions.

4.2 Attack generation

Generating attacks is by no means simpler than generating the background traffic. It is not enough to gather a handful of attack scripts and run them. Careful consideration must be given to how many scripts will be used, and how they will be chosen, because this would obviously bias the test (particularly so in the case of misuse detectors). Attacks them must be ran against both vulnerable and not vulnerable machines, and they must also be perfectly blended with background traffic. In fact, replaying datasets, mixing them, superimposing attacks can create artifacts that are easy to detect (an example will be shown in Section 6). Another important question here is whether or not we will mutate

or obfuscate the attacks using evasion techniques (as discussed briefly in Section 3.2).

5 Evaluating alert correlation systems

Even if, for simplicity, in this work we talked of IDS systems as monolithic entities, they are often composed of distributed components, sometimes of different types. Network and host based probes are complementary, and also anomaly and misuse based systems should be combined. For a distributed monitoring network, an appropriate collaboration infrastructure is needed. Moreover, in complex network infrastructures where log data and data from security scanners are also available, there is a problem of correlating and aggregating this data in a usable form. A class of products called SIM (Security Information Management) has been developed to try to address this problem.

The main goal of a correlation systems should be to reduce the amount of alerts and data a security analyst has to check. In doing this, the DR should ideally not decrease, while the FPR should be reduced as much as possible. This suggests that a measurable sub-goal of correlation systems is, somehow, the global *reduction* of FPR (by reducing the total number of alerts fired by source IDS through a rejection or discarding mechanism). Since alert correlation is a relatively new problem, evaluation techniques are limited to a few approaches [22]. Thus the development of solid testing methodologies is needed from both the theoretical and the practical points of view. To take into account the above considerations, a first, simple metric can be constructed as follows: let be $\mathbb{A}$ the alert set of the original alert stream (union of all alert streams reported by all IDS), and $\mathbb{A}'$ the set of alerts in output of the fusion system. We define also $DR_{\mathbb{A}}$ and $FPR_{\mathbb{A}}$ as, respectively, the detection rate and the false positive rate in the overall detection environment; in the same way $DR_{\mathbb{A}'}$ and $FPR_{\mathbb{A}'}$ will be the DR and the FPR measured on the fused output.

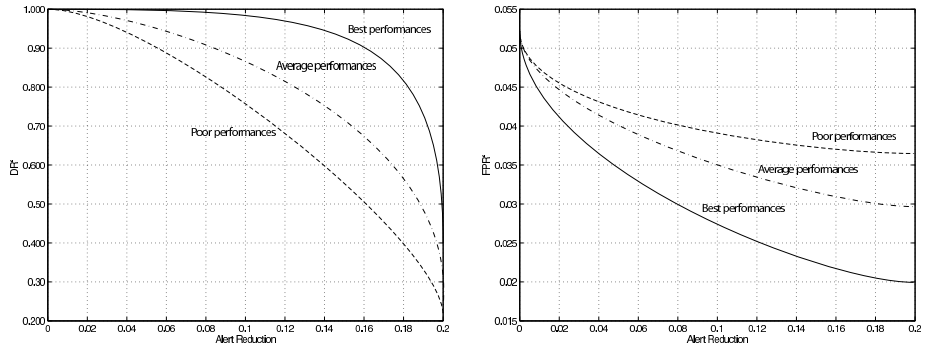
Implementing the ideal correlation system means minimizing $AR = |\mathbb{A}'|/|\mathbb{A}|$ (the alert reduction) while maximizing both

- $1 - (DR_{\mathbb{A}'}/DR_{\mathbb{A}})$
- $FPR_{\mathbb{A}'}/FPR_{\mathbb{A}}$

Therefore, a correlation system is better than another if it has both a greater $DR_{\mathbb{A}'}/DR_{\mathbb{A}}$ and a lower $FPR_{\mathbb{A}'}/FPR_{\mathbb{A}}$ rate than the latter. This criterion, however, by no means has to be considered as complete or exhaustive. It is useful to compare $DR_{\mathbb{A}'}$ and $FPR_{\mathbb{A}'}$ plots, vs. AR, of different correlation systems obtaining diagrams like the one exemplified in Fig. 2; this gives a graphical idea of which correlation algorithm is better: for instance, Fig. 2 show that the algorithm labeled as "Best performances" is better than the others, because it shows higher $FPR_{\mathbb{A}'}$ reduction while $DR_{\mathbb{A}'}$ does not significantly decrease.

6 Case study: the validity of the DARPA dataset

The lack of reliable datasets for testing IDS is a serious problem for research. IDS researchers need objective evaluation datasets that can be replicated and



(a) (b)

Figure 2: Hypothetical plot of the $FPR_{\mathbb{A}'}$ (a) and $DR_{\mathbb{A}'}$ vs. AR.

openly shared. We also need reliable truth files for those datasets. This rules out any use of real world traffic, and also unfortunately rules out most of the dumps from the DEFCON and other CTFs [23]. The dataset created by the Lincoln Laboratory at M.I.T., also known as "DARPA IDS Evaluation dataset" [24], is basically the only dataset freely available along with truth files, and it has been extensively (almost exclusively) used for research in the field. This is a crucial factor: any bias or error in the DARPA dataset has influenced, and will influence in the future, the very basic research on this topic.

These data have been collected in order to evaluate detection rates and false positives rates of IDS. Both the background traffic and the attack traffic are artificially generated. There are two datasets: 1998 and 1999. The 1999 dataset for instance [25], spans over 5 weeks, and contains the packet dumps in `tcpdump` format of 5 weeks, over 2 sniffers, one placed between the gateway and 5 "target" machines (thus emulating an "internal" sniffer), and the other placed beyond the gateway, recording packets flowing between the simulated LAN and the simulated Internet. Both attack-free data and clearly labeled attack traces are present. Also, BSM auditing data for Solaris systems, NT auditing data for Windows systems, a directory tree snapshots of each system, the content of sensitive directories, and inode data are made available as part of the dataset. The 1998 dataset is similar, and it is described by a master's thesis [26].

6.1 Network Data Shortcomings

Let us examine briefly how realistic these data are. In fact, the network data of the 1999 dataset have already been extensively criticized. In [27] it is noted that no detail is available on the generation methods, that there is no evidence that the traffic is actually realistic, and that spurious packets, so common on the Internet today, are not taken into account. The same can be said for checksum errors, fragmented packets, and similars. The simulated network is flat, and therefore unrealistic.

In [28] it is additionally noticed that the synthetic packets share strange regularities that are not present in real world traffic:

- SYN packets use always a 4-byte set of options, while in the real world this value ranges from 0 to 28 bytes.

- The TCP window size varies among seven fixed values ranging from 512 and 32120.
- There are just 29 distinct IP source addresses, and half of these account for over 99.9% of the traffic; in real world data, for a similar network with similar characteristics, over 24.000 unique addresses were counted.
- TTL and TOS fields are unrealistically similar for most packets. For instance, in the dataset 9 values of TTL out of 256 are used, while in real world data 177 different values can be seen; similarly, just 4 different TOS fields were observed in the dataset, against over 40.
- There are no packets with checksum errors in the IDEVAL dataset, while in real data a small but not null percentage of packets exhibits checksum errors; similarly, the dataset lacks fragmented packets, flag anomalies, etc.
- HTTP requests are all of the form `GET url HTTP/1.0` with 6 different keywords and 5 different User-Agent. Real traffic shows different commands, over 70 different keywords and over 800 different user agents; in real traffic commands and keywords are sometimes malformed, while in the dataset this is not present. Similar consideration apply to SMTP and SSH traffic.

The authors even propose a simple IDS system based on a single byte of the IP header (the third byte of the IP address, in particular), which achieves a 45% Detection Rate with just a bunch of false positives.

These characteristics make it difficult to understand whether IDSs tested and developed on DARPA traffic are capable of detecting true anomalies, or they are just capable of detecting the irregularities in the synthetic DARPA traffic. For instance, attacks *back*, *dosnuke*, *neptune*, *neptbus*, *netcat*, *ntinfoscan* and *quaeso* can be easily spotted, even by human eye, because they use TTL values that never appear into the training set. SMTP attacks are recognizable by the fact that they do not begin with a regular `HELO` o `EHLO` command; most attacks come from IP addresses that are not present in the training files; and so on.

6.2 Host Data Shortcomings

The works cited above, however, did not address the host based auditing data contained in the dataset. As we showed in [29], there are a number of issues also in this part of the IDEVAL data.

The first problem is that in the training datasets there are too few execution instances for each software, in order to representatively model its behavior, as can be seen in Table 2. Of just 6 programs present, for two (`fdformat` and `eject`), only a handful of executions is available, making training unrealistically simple.

The number of system calls used is also extremely limited, making execution flows very similars. Additionally, most of these executions are similars, not covering the full range of possible execution paths of the programs (thus causing overfitting of any anomaly model).

Program name	Number of executions
`fdformat`	5
`eject`	7
`ps`	105
`ftpd`	65
`telnetd`	1082
`sendmail`	827

Table 2: Number of instances of execution in the IDEVAL dataset

The arguments show the same lack of variability. In all the training dataset, all the arguments of the system calls related to `telnetd` belong to the following set:

```
fork, .so.1, utmp, wtmp, initpipe, exec, netconfig,
service_door, :zero, logindmux, pts
```

Just to give another example, the FTP operations (30 sessions on the whole) use a very limited subset of file (on average 2 per session), and are performed always by the same users on the same files, for a limitation of the synthetic generator of these operations. In addition, during training, no uploads or idle sessions were performed.

Local process names are often crafted, or meaningless. Sometimes there are duplicate executions of programs with identical PID and timestamps.

Finally, also in this case we were able to create a "detector" which finds all the host-based attacks without any false positive. A simple script which flags as anomalous any argument longer than 500 characters can do this. In other words: the only meaningful indicator of attacks in the IDEVAL dataset is the length of strings.

6.3 Outdated software and attacks

The last dataset in the IDEVAL series was created in 1999. Obviously, since then, everything changed: the usage of network protocols, the protocols themselves, the operating systems and applications used. For instance, all the machines involved are Solaris version 2.5.1 hosts, which are evidently ancient nowadays.

The attacks are similarly outdated. The only attack technique used are buffer overflows, and all the instances are detectable in the `execve` system call arguments. Nowadays attackers and attack type are much more complex than this, operating at various layers of the network and application stack, with a wide range of techniques and scenarios that were just not imaginable in 1999.

Finally, the whole IDEVAL dataset is evidently outdated (computing is extremely different than today in 1999, of course!). The usage of network protocols, the protocols themselves, the operating systems and applications are representative of a world that does not exist anymore. Attacks are similarly outdated. The most used attack technique is buffer overflow, and intrusion scenarios are extremely simple. Nowadays attackers and attack types are much

more complex than this, operating at various layers of the network and application stack, with a wide range of techniques that were just not imaginable in 1999.

6.4 Unrealistic attack scenarios

As we already stated, the IDEVAL dataset contains both host and network auditing data. However, it must be noted that many attacks are not directly detectable in both systems. The only such attacks are the ones in which an attacker exploits a vulnerability in a local or remote service to allow an intruder to obtain or escalate privileges. In particular, we use the BSM audit logs from the system named `pascal.eyrie.af.mil`, which runs a Solaris 2.5.1 operating system.

7 Conclusions

In this paper, we briefly reviewed all the issues in testing intrusion detection and prevention systems, and how they can lead to misleading results, if not to outright frauds. We have repeatedly shown how many differences are there in testing intrusion detection and prevention systems, and in testing misuse based systems or anomaly based ones. We have also reviewed a number of performance metrics, and their fallacies. False positives and detection rate are deceivingly simple and do not adequately account for misuse based detectors. Resistance to polymorphism is important and difficult to quantify. Coverage is meaningful for misuse detectors, more than anomaly detectors, and still difficult to measure. Performance indexes are easier to measure, but still hide complexities, for instance in building the appropriate model for the relationship between throughput, latency and discarded packets. We then moved on to the difficulties in generating a reliable, realistic workload suitable for IDS evaluation, both for the generation of background traffic and for the generation of the attacks.

We have also outlined a number of shortcomings in the IDEVAL dataset, which is still the only standard dataset for the validation and evaluation of Intrusion Detection Systems widely available. The network data suffer of various well known problems, regularities, and characteristic flaws. The execution traces for system call analysis are also flawed, as we demonstrated in our earlier works: they are too simple and predictable. In addition, the dataset is now hopelessly outdated, because the protocols, applications and operating systems used are not representative any more of normal network usage; and also because the attack types are not representative of the modern threat scenario.

We can conclude that we are still very far away from designing a complete, scientific testing methodology for IDS and IPS systems. However, we know a lot of things about wrong methodologies and common fallacies. Enough to debunk most claims found in both marketing literature and the technical press.

Acknowledgments

The presentation of this work at the FIRST conference was made possible by a supporting grant from the Programme Committee, which we gratefully acknowledge; in particular, we need to warmly thank Mr. Arjen de Langraaf.

This work was partially carried out in the framework of the FIRB-Perf project funded by the Italian Ministry of Education, Universities and Research (MIUR), in the research unit led by prof. Giuseppe Serazzi, whose continuing support we gratefully acknowledge. An early version of this work was shown first at the Black Hat conference series [30]. We also need to thank, for their helpful comments and support, Richard Bejtlich, Renaud Bidou, Cedric Blancher, Federico Maggi and Thomas Ptacek.

References

[1] Dixie B. Baker. Fortresses built upon sand. In *Proc. of the 1996 Workshop on New Security Paradigms*, pages 148–153. ACM Press, 1996.

[2] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, J. P. Anderson Co., Ft. Washington, Pennsylvania, Apr 1980.

[3] Bob Colwell. If you didn't test it, it doesn't work. *Computer*, 35(5):11–13, 2002.

[4] J. Snyder. Taking aim: Target-based IDSes squelch network noise to pinpoint the alerts you really care about. *Information Security Magazine*, January 2004.

[5] D.M. Chess and S.R. White. An undetectable computer virus. In *Proceedings of the 2000 Virus Bulletin Conference (VB2000)*, 2000.

[6] Carey Nachenberg. Computer virus-antivirus coevolution. *Commun. ACM*, 40(1):46–51, 1997.

[7] E. Filiol, M. Helenius, and S. Zanero. Open Problems in Computer Virology. *Journal in Computer Virology*, 1(3):55–66, 2006.

[8] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Polymorphic worm detection using structural information of executables. 2005.

[9] Thomas H. Ptacek and Timothy N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical Report T2R-0Y6, Secure Networks, Calgary, Canada, 1998.

[10] M. Handley, V. Paxson, and C. Kreibich. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. USENIX Security Symposium*, 2001.

[11] K-2. ADMmutate. In *CanSecWest 2001 Conference*, March 2001.

[12] G. Vigna, W. Robertson, and D. Balzarotti. Testing Network-based Intrusion Detection Signatures Using Mutant Exploits. In *Proceedings of the ACM Conference on Computer and Communication Security (ACM CCS)*, pages 21–30, Washington, DC, October 2004.

[13] Icsa ids testing methodology. available online at `https://www.icsalabs.com/icsa/main.php?pid=jgh475fg`.

[14] Marcus J. Ranum. Experiences benchmarking intrusion detection systems. `http://www.snort.org/docs/Benchmarking-IDS-NFR.pdf`, 2001.

[15] Simonetta Balsamo, Raif O. Onvural, and Vittoria De Nitto Persone. *Analysis of Queueing Networks with Blocking*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[16] Renaud Bidou. Ips shortcomings. In *Black Hat Briefings 2006 USA*. available online at `https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Bidou.pdf`.

[17] Kymie M. C. Tan, Kevin S. Killourhy, and Roy A Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In Giovanni Vigna Andreas Wespi and Luca Deri, editors, *Fifth International Symposium on Recent Advances in Intrusion Detection (RAID-2002)*, volume 2516 of *Lecture Notes in Computer Science*, pages 54–73, Zurich, Switzerland, October 2002. Springer-Verlag.

[18] Nicholas J. Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee, and Ronald A. Olsson. A methodology for testing intrusion detection systems. *IEEE Transactions on Software Engineering*, 22(10):719–729, 1996.

[19] K. Claffy, G. Miller, and K. Thompson. The nature of the beast: Recent traffic measurements from an internet backbone. 1998.

[20] S. McCreary and K. Claffy. Trends in wide area ip traffic patterns: A view from ames internet exchange. available online at `http://www.caida.org/outreach/papers/2000/-AIX0005/`, 2000.

[21] Museum of broken packets. available online at `http://lcamtuf.coredump.cx/mobp/`.

[22] Joshua Haines, Dorene Kewley Ryder, Laura Tinnel, and Stephen Taylor. Validation of sensor alert correlators. *IEEE Security and Privacy*, 01(1):46–56, 2003.

[23] The Shmooo Group Capture the CTF project. available online at `http://cctf.shmoo.com`.

[24] Darpa intrusion detection evaluation datasets. available online at `http://www.ll.mit.edu/IST/ideval/data/dataindex.html`.

[25] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. In *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, pages 162–182, London, UK, 2000. Springer-Verlag.

[26] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachussets Institute of Technology, 1999.

[27] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. on Information and System Security*, 3(4):262–294, 2000.

[28] M. V. Mahoney and P. K. Chan. An analysis of the 1999 DARPA / Lincoln laboratory evaluation data for network anomaly detection. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, pages 220–237, Pittsburgh, PA, USA, September 2003.

[29] Stefano Zanero. *Unsupervised Learning Algorithms for Intrusion Detection.* PhD thesis, Politecnico di Milano T.U., Milano, Italy, May 2006.

[30] Stefano Zanero. My ids is better than yours... or is it ? In *Blackhat Federal 2006 Briefings*, 2006.