Automatic Evaluation of Intrusion Detection Systems

Frédéric Massicotte
Canada Communication Research Center
3701 Carling
Ottawa, Canada

François Gagnon, Yvan Labiche,
Lionel Briand and Mathieu Couture
Carleton University
1125 Colonel By
Ottawa, Canada

Abstract

An Intrusion Detection System (IDS) is a crucial element of a network security posture. Although there are many IDS products available, it is rather difficult to find information about their accuracy. Only a few organizations evaluate these products. Furthermore, the data used to test and evaluate these IDS is usually proprietary. Thus, the research community cannot easily evaluate the next generation of IDS. Toward this end, DARPA provided in 1998, 1999 and 2000 an Intrusion Detection Evaluation Data Set. However, no new data set has been released by DARPA since 2000, in part because of the cumbersomeness of the task. In this paper, we propose a strategy to address certain aspects of generating a publicly available documented data set for testing and evaluating intrusion detection systems. We also present a tool that automatically analyzes and evaluates IDS using our proposed data set.

1 Introduction

Since the DARPA Intrusion Detection Evaluation Data Set [1] was made available in 1998 and then updated in 1999 and 2000, no other significant publicly available data set has been provided to benchmark Intrusion Detection Systems (IDS).

Other organizations [2, 3] also provide data sets of traffic traces with attacks and intrusions such as worms and denials of service. However, these data sets are mainly used for the statistical and traffic behavior analysis of large traffic traces (e.g., studying the infection evolution of worms). These data sets are very useful to the security research community, but they are not sufficiently documented for automated IDS testing and evaluation. Moreover, these data sets contain traffic from only 4 different worms: Nimda, Slammer, W32.Mydoom and Witty; and from only a few denials of service. Thus, these data sets contain an insufficient variety of attack instances and behaviors to properly test and evaluate IDS.

This lack of properly documented data sets for IDS testing and evaluation was mentioned in a NIST report [4], which concludes with recommendations for IDS testing research. In particular, the authors insist that data sets should contain realistic data and be shared freely among multiple organizations. They also state that there is a need to provide the security community with a large set of attack traces. Such information could be easily added to and would greatly augment existing vulnerability databases. The resulting vulnerability/attack trace databases would aid IDS testing researchers and would provide valuable data for IDS developers.

Data sets used to test IDS can be described by two main characteristics: the type of intrusion detection technology used (signature-based or anomaly-based) and the location of the IDS (host-based or network-based). The test cases needed to evaluate a signature network-based IDS are significantly different from those needed by an anomaly host-based IDS.

In this paper, we present both a traffic trace generation technique and an IDS evaluation framework. The former, referred to a Virtual Network Infrastructure (VNI), allows us to automatically generate properly documented data sets of attack scenarios. The latter, referred to as Intrusion Detection System Evaluation Framework (IDSEF), allows us to automatically test and evaluate IDS using these traffic traces.

The data set generated by our framework, though extensible, is currently specific to signature-based, network intrusion detection systems. It currently contains only well-known attacks, without background traffic. The main reason for these two restrictions is that we wanted to be convinced of the feasibility of our approach to devise a thorough experimental evaluation of existing IDS. Also, the current goal of the data set is not to check whether IDS raise alarms on normal traffic (which will be the focus of future work), but rather to test and evaluate the detection accuracy of IDS in the case of successful and failed attack attempts.

This paper also reports on an initial evaluation of our framework on two well-known IDS namely Snort [5] and Bro [6]. The experiment showed that properly documented data sets such as ours can be used to automatically test and evaluate IDS. Results are encouraging as we are able to automatically generate a large, properly documented data

set, which is publicly available to the research community, and use this data set to perform an initial evaluation of Snort and Bro. This evaluation also identified many problems with Snort and Bro.

This paper is organized as follows. Section 2 describes related work on IDS testing and evaluation. Section 3 describes the VNI as well as the automatic collection process of the data set and the automatic documentation process. It also summarizes the current contents of our Intrusion Detection Evaluation data sets. Section 4 describes the IDSEF and discusses the automatic analysis of the results. Section 5 presents the results of the evaluation of Snort and Bro using our data set. The last section concludes this article by outlining future work.

2 Related Work

The relevant literature shows that many different techniques are used to test IDS. A classification of testing techniques can be found in [7]. There are two main techniques for testing IDS detection accuracy: the IDS stimulator approach and the vulnerability exploitation program approach.

2.1 IDS Stimulators

Descriptions of the most popular IDS stimulators can be found in [8, 9, 10, 11]. They are used for many testing purposes: To generate false alarms by sending packets that resemble well-known intrusions [9, 11] (These false attacks are launched in combination with real attacks to test whether IDS are still able to detect real attacks when flooded by false alarms [12]); for cross-testing network-based IDS signatures and for testing the IDS engine [8] (In particular, test cases are generated from the Snort signature database and launched against different IDS); and to generate the appropriate traffic by using the IDS signature [8, 9, 11]. Thus, these tools rely on publicly available signature databases to generate the test cases. Unfortunately, in many situations, the needed signatures are undisclosed or not available from vendors.

2.2 Vulnerability Exploitation Programs

To overcome this limitation imposed by the IDS vendors, vulnerability exploitation programs can be used to generate test cases. IDS evasion techniques such as packet fragmentation can also be applied to these vulnerability exploitation programs to further test the accuracy of the IDS. The most popular IDS evasion techniques used by hackers can be found in [13–19]: [13] provides a classification of such IDS evasion techniques.

The use of vulnerability exploitation programs for IDS testing and evaluation usually implies building a test bed where the attacks are launched using these vulnerability exploitation programs. The attack traffic can be combined with real or emulated normal traffic as background. The traffic is either recorded for off-line IDS testing or the IDS are tested in real-time on the test bed network.

A number of organizations and projects such as [1, 12-13, 16, 20-24] have developed such test beds and techniques. However, we found three major problems with the data sets they used for IDS testing and evaluation: their availability, the documentation of their traffic traces and their generation processes.

With the exception of those provided by DARPA, most of the data sets used for evaluating and testing IDS are not publicly available. Since the DARPA traffic traces represent the only significant, publicly available data, they are still used by the security research community, even if they contain no recent attacks and the techniques used to create normal traffic has been criticized [25].

Documentation is one of the main problems with the available traffic traces from [2, 3] and the DARPA data sets. To test and evaluate IDS, it is essential to use a properly documented data set. For each attack in the data set, it is important to know key information such as the targeted system configuration (operating system, targeted service) and the attack specification (vulnerability exploitation program used, its configuration and targeted vulnerability). As presented in Section 4 such information allows the automation of IDS testing and evaluation.

In a number of cases, the generation of traffic traces and their use during actual IDS testing and evaluation is manual or semi-automated. Manual intervention takes time and restricts the diversity and updatability of the data set. Manual or semi-automated IDS evaluation limits the number of test cases the testing techniques are able to use. For instance, according to the authors of [13], one of the most recent IDS evaluations by NSS (4th edition) [12] was done manually. In addition, some of the tests conducted in [24] were done by hand. In [16, 20], the authors used test beds with real systems. Incidentally, resetting the targeted system to the initial conditions in place before the attack affected the system is either slow [20] (reloading Ghost images of the unaffected system) or not automatic. In fact, the number of vulnerability exploitation programs used in the data sets discussed in this section is often small and the variety of the targeted systems is limited (Section 3.4).

2.3 Proposed Solution

The techniques proposed in this paper attempt to overcome these limitations. Our contribution is to propose a virtual network infrastructure (VNI) that is able to generate a data set that can be shared with the research community and that can be rapidly and automatically updated, documented and used for IDS testing and evaluation. This system is completely automated and can generate a data set in a matter of hours. This allows us to generate real attacks and then quickly revert our test network back to its initial state before each attack (see

Section 3). Our VNI is therefore able to rapidly and efficiently generate a large data set with hundreds of vulnerability exploitation programs launched against a large variety of target systems (Section 3.4). Our approach is also flexible as we can choose whether to apply or not IDS evasion techniques. It can be used to generate data sets for other purposes: An older version of the VNI was used to fingerprint 208 operating systems according to 12 passive fingerprint identification techniques [26]. The current version is used to provide data to the LEURRE [27] and SCRIPTGEN [28] projects of Eurecom. It is also updatable: new attack scenarios, IDS evasion techniques, and target systems can easily be added.

3 Virtual Network Infrastructure Overview

In this section, we first summarize the requirements for an infrastructure supporting our approach and discuss our design choices (Section 3.1). We then describe our infrastructure for collecting traffic traces and discuss our traffic trace documentation (Sections 3.2 and 3.3). Section 3.4 then summarizes the current contents of our data set.

3.1 Infrastructure Requirements

To create a large scale data set, a controlled network infrastructure had to be developed to allow: (1) recording of all network traffic; (2) control of network traffic noise; (3) control of attack propagation; (4) usage of real and heterogeneous system configurations; and (5) fast attack recovery to initial condition state.

To meet most of these requirements emulators or virtual machines are often used. For instance, they are used by the security community to construct virtual networks as they provide traffic propagation control and reduce computer resources (e.g., [29, 30]).

We developed a controlled virtual network using VMware[1] 5.0 [31]. We selected VMware, among others (e.g., [32, 33]), as we already had a positive experience with it [26] and it fulfilled the aforementioned requirements. It provides a virtual broadcasting network environment that allows the capture of all communications, in our case generated by the attack within a single traffic trace. These traffic traces can then be used to study attack behaviors (req. 1). This virtual network also allows us to control the network traffic to create clean traces that only contain network traffic relevant to the attack scenarios (req. 2). With VMware, the attack propagation is confined, thus preventing infection of the physical machines running the virtual network (req. 3). VMware facilitates the creation of template virtual machines having different software configurations (operating system, services, etc). Thus, it allows creation of a database of virtual machine templates that can be used to rapidly deploy custom network configurations within a single physical machine (req. 4). Also, VMware snapshot allows restoration of the test network to the state it was in before each attack attempt. All attack scenarios can then be performed under the same initial conditions (req. 5).

[1] VMware is a trademark of VMware, inc.

3.2 Collection Process

The virtual network we use to collect traffic traces is shown in Figure 1. It contains attack systems (Attacker), target systems (Target) and network infrastructure services and equipment such as DNS and mail servers. The attack systems are used to launch attacks against the target systems by using vulnerability exploitation programs either with or without IDS evasion techniques. The attack systems are also used to capture the packets that are generated by the execution of the vulnerability exploitation programs. The network infrastructure services and equipment ensure the network communications needed while the attack is in progress.

Each step of our collection process is indicated in Figure 1. A link between the involved entities, steps 2 to 5 being repeated for each attack.

1. Script Generation. This process chooses which vulnerability exploitation program (VEP) will be run against a given target system and how it should be configured. For the current data set, we decided to run every VEP against every target system offering a service on the same port as the VEP targeted service. To automate script generation, we built a database containing the complete system configuration for each target template, as well as the ports targeted by the VEP we downloaded.

2. Virtual Network Setup. A different virtual network is built for every script. Each contains the target virtual machine, the attacking virtual machine and some other machines offering the network services needed for the execution of the attack (e.g., DNS server). The coordinator opens the virtual network and locks the resources (virtual machines) it uses. Many virtual networks can be setup in parallel on different physical machines and a coordinator controls each of them.

3. Current Attack Script Setup. Once the virtual network is ready, the coordinator provides the attacking virtual machine with the proper attack configuration. To communicate with the attacker, the coordinator (i.e., the physical machine) uses a hard drive shared via VMware. This shared drive is the only way the coordinator can communicate with the virtual network. This enables us to isolate the virtual network (from a networking point of view) while keeping some communication capabilities.

4. Attack Execution. The attack machine performs the attacks while generated traffic is recorded. The attack system is composed of four layers (Figure 2): the Control Layer is composed of a Java module called ExploitRunner that controls and executes the attacks based on the configuration provided by the coordinator; the VEP are executed at the Attack Layer and then provided to the

Mutation Layer where evasion techniques can be applied; and the traffic generated by the attacks is captured (using *tcpdump*) and documented by the Recording Layer.

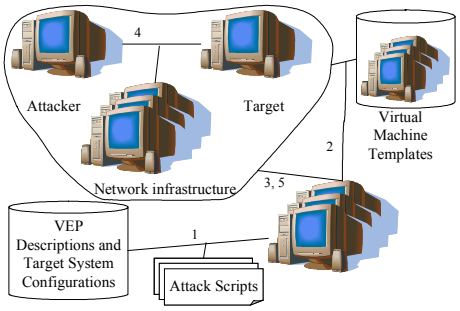

Figure 1. Virtual network infrastructure

5. Tear Down. This includes saving the traffic traces (VEP output and the recorded traffic) on the same shared drive used in step 3. Then, the coordinator stores the attack trace in the data set and restores the attacker and target virtual machines to their initial state to avoid side effects (e.g., impact on the next attack).

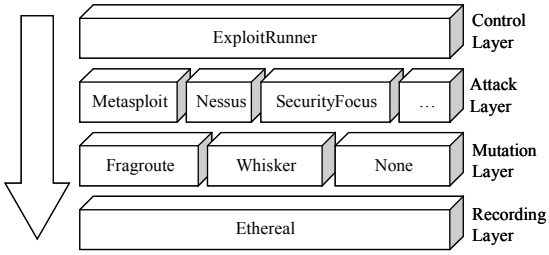

Figure 2. Attack system

3.3 Documenting Traffic Traces

In our data set, each traffic trace is documented by four characteristics: the target system configuration, the VEP configuration, whether or not the target system has the vulnerability exploited by that program and the success of the attack: see an example in Figure 3.

```
System Configuration
  IP: 10.92.39.14
  Name: VMWin2000ServerTarget
  Operating System: Windows 2000 Server
  Ports:
    21/tcp Microsoft IIS FTP Server 5.0
    25/tcp Microsoft IIS SMTP Service 5.0
    80/tcp Microsoft IIS Web Server 5.0
Vulnerability Exploitation Program Configuration
  name: jill.c
  reference: Bugtraq,2674
  command: jill 10.92.39.14 80 10.92.39.3 30
  Vulnerable: yes
  Success: yes
```

Figure 3. Traffic trace label example

The target system configuration description includes the list of installed software (e.g. the operating system, the different daemons and their versions), as well as its IP configuration. The VEP configuration defines the options used to launch the attack. The vulnerability of the target system is decided automatically on the basis of: (1) its configuration; (2) the VEP being used in this attack; and (3) the vulnerability information available in the Security-Focus database [34]. We will see in Section 4 that this last piece of information is paramount as it allows automated IDS testing. As mentioned in the previous section, an analysis determines whether the attacks have been successful or not and automatically classifies the attack outputs. The traffic traces are labeled according to three categories: one for those that succeed in exploiting the vulnerability (Yes); one for those that fail (No); and one for those for which we were not able to determine whether they were successful (Unclassified). This classification is automatically made by looking at the program outputs (hacker point of view) and at the effect on the targeted system (victim point of view).

3.4 Data Set Summary

The current version of our data set was only developed to test and evaluate network-based and signature-based intrusion detection systems for attack scenario recognition. It is composed of two different data sets: the standard IDS evaluation data set (StdSet) and the IDS evasion data set (EvaSet). The former contains traffic traces of attack scenarios derived from the execution of well-known VEP. Our VEP are mainly taken from the ones available in [34, 35, 36, 37]. The latter contains the IDS evasion techniques applied to the successful attacks contained in the standard data set. This data set is used to verify whether IDS are able to detect modified attacks.

The two sets are collections of *tcpdump* traffic traces, each one containing one attack scenario. To generate StdSet, we used 124 VEP (covering a total of 92 vulnerabilities) and 108 different target system configurations. Each VEP was launched against vulnerable and non-vulnerable systems (among the 108 target system configurations) that offered a service on the port targeted by the VEP. Every combination (VEP + configuration + target) produces a traffic trace in the set. This resulted in 10446 traffic traces in StdSet of which 420 succeeded and 10026 failed. EvaSet contains 3549 attack traces generated by applying IDS evasion techniques to the successful attacks from StdSet. Figure 4 shows the VEP and the corresponding attack scenarios in StdSet, classified according to the port number they target. For example, in StdSet, 45 VEP target TCP port number 80 and this corresponds to 2801 attack scenarios. Note that the number of VEP does not reflect directly the number of attacks on a particular port since each VEP is used multiple times (for each possible configuration and against each possible target system).

These numbers are much higher than has been reported in literature: [22], [20], and [21] used respectively 9, 66

and 27 VEP against only 3 different target systems based on [4]; [23], [13] and [24] used respectively 50, 10 and 60 VEP ([23] and [13] only used 9 and 5 different target systems).

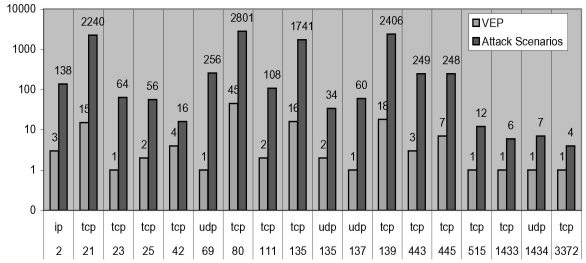

Figure 4. Standard data set port distribution

4 IDS Evaluation Framework Overview

In this section, we first describe the evaluation framework and the corresponding data collection process, and then discuss our result classification.

4.1 Collection Process

To demonstrate that our documented data sets can be useful to automatically evaluate signature-based network IDS and can provide interesting analysis even with only well-known VEP, we developed an IDS evaluation framework, written in Java, that consists of four components (Figure 5). First, the IDSEvaluator takes each test case (documented traffic trace) from the data set and provides it to each tested IDS. The alarms generated by the IDS are collected and provided to the IDSResultsAnalyzer, which is responsible for automatically analyzing them. The IDSResultsAnalyzer relies on the vulnerability reference information (SecurityFocus, CVE, etc.) provided by IDS alarms to determine whether the IDS had detected the attack scenario. Because each VEP used in the data set is related to a specific documented vulnerability (e.g., SecurityFocus), the IDS alarms can be associated with a particular VEP. In other words, with the information from our documented data set and the vulnerability reference information, our IDS evaluation framework is able to automatically verify the detection or not of each attack.

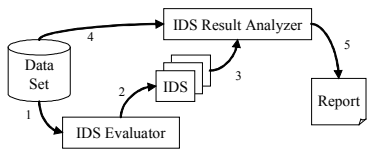

Figure 5. IDS evaluation framework

4.2 IDS Alarms Analysis Process

After analyzing the IDS alarms, the IDSResultAnalyzer automatically classifies the results using two parameters: the actual success (Success) or failure (¬Success) of the attack and the detection (Detection) or not (¬Detection) of this particular attack by the IDS. Recall that the attack success can be determined from our documented data set.

A proper classification should refine the notion of detection and consider at least four classes of detection: (Attempt), (Successful), (Failed) and (Silent). The Attempt detection class specifies when an attack attempt is detected or when the IDS knows nothing about the probability of the success of the attack. The Successful (resp. Failed) detection class specifies when an attack occurs and the IDS has some evidences that it may succeed (resp. Fail). The Silent detection class represents the absence of messages when nothing suspicious is happening. Unfortunately, Snort 2.3.2 and Bro 0.9a9 do not provide detailed enough messages to allow us to use this refined classification. They do not provide alarms when a failed attempt is detected and they do not distinguish properly between attempts and successful attempts. Therefore, in the current analysis the (Detection) and (¬Detection) messages are used. This provides four possible results for each traffic trace analyzed by the IDS:

True Positive (TP): when the attack succeeded and the IDS was able to detect it (Success ∧ Detection)

True Negative (TN): when the attack failed and the IDS did not report on it (¬Success ∧ ¬Detection)

False Positive (FP): when the attack failed and the IDS reported on it (¬Success ∧ Detection)

False Negative (FN): when the attack succeeded and the IDS was not able to detect it (Success ∧ ¬Detection)

The above classification only provides fine-grained information, specifically information at the level of the attack scenario. Each of the four classifications needs to be incorporated together to form a more precise analysis. It is essential to know how many TPs, TNs, FPs, and FNs we have for a given group of attack scenarios. By using this view, we are able to analyze if the IDS is able to properly distinguish between failed and successful attack attempts for a group of attack scenarios. Therefore, we combine the four classes above and suggest a classification of fifteen classes presented in Table 1.

In this classification, an IDS is said to be Alarmist for a group of attack scenarios when the IDS emits an alarm on all failed attack scenarios in this group: TN=No and FP=Yes. An IDS is said to be Quiet for a given group of attack scenarios when the IDS does not report on any failed attack scenario in this group: TN=Yes and FP=No. If the IDS emits an alarm for only some of the failed attack scenarios on a group, then we say that the IDS is Partially Alarmist for this group: TN= Yes and FP=Yes. Another dimension is used in Table 1. There is Complete Detection for a group of attack scenarios by an IDS when all successful attacks scenarios are correctly detected by an IDS: TP=Yes and FN=No. On the other hand, there is Complete Evasion when none of the successful attacks are detected by the IDS: TP=No and FN=Yes. Otherwise, the group is Partial Evasion: TP=Yes and FN=Yes.

The fifteen classes are further grouped into three sub-groups; one when the analysis contains successful and failed attack scenarios; one when there are only failed attack scenarios; and one when there are only successful attack scenarios.

Class Name	TP	TN	FP	FN
Success and Failed Attempts				
Quiet and Complete Detection	Yes	Yes	No	No
Partially Alarmist and Complete Detection	Yes	Yes	Yes	No
Alarmist and Complete Detection	Yes	No	Yes	No
Quiet and Partial Evasion	Yes	Yes	No	Yes
Partially Alarmist and Partial Evasion	Yes	Yes	Yes	Yes
Alarmist and Partial Evasion	Yes	No	Yes	Yes
Partially Alarmist and Complete Evasion	No	Yes	Yes	Yes
Alarmist and Complete Evasion	No	No	Yes	Yes
Quiet and Complete Evasion	No	Yes	No	Yes
Failed Attempts Only				
Alarmist	No	No	Yes	No
Partially Alarmist	No	Yes	Yes	No
Quiet	No	Yes	No	No
Successful Attempts Only				
Complete Detection	Yes	No	No	No
Partial Evasion	Yes	No	No	Yes
Complete Evasion	No	No	No	Yes

Table 1. Proposed classification

5 IDS Evaluation Results

We have selected Snort 2.3.2 (released 10/03/2005) and Bro 0.9a9 (released 19/05/2005) for our initial evaluation of the efficiency of our data set and evaluation framework since they are two well-known and widely used IDS.

In the case of Snort, we used the set of rules included in the release. Bro comes with a Snort rule set translated into the Bro signature language. The translated Snort rules are however older than the rules available in Snort 2.3.2. Bro provides a rule converter (*s2b*) that not only translates Snort rules to a Bro format, but also enhances Snort rules to reduce false positives. To get a fair comparison between Snort and Bro, we thus used *s2b* to update the Bro rule set from Snort 2.3.2 rules. One difficulty we encountered was that the *s2b* converter was not able to convert all Snort plug-ins such as *byte test*, *byte jump*, *isdataat*, *pcre*, *window* and *flowbits*. In these cases, we manually translated into the Bro language the Snort rules that are used to monitor the vulnerability contained in our data set. In the results we report next, we used a subset of our data sets. We only used the VEP associated to the most recent vulnerabilities that have been released before or around the release of Snort and Bro to provide a fair analysis for each IDS. As a result, StdSet and EvaSet used in this analysis include 102 VEP. The results are grouped by VEP and classified as proposed in the previous section. From StdSet, 5 groups of VEP for which we have similar observations are discussed (Sections 5.1 to 5.5). In the case of EvaSet, we mainly focus on the results obtained from the packet fragmentation and HTTP evasion techniques (Section 5.6). Section 5.7 summarizes the results.

5.1 Inconclusive Group

The first inconclusive group contains VEP that did not generate any successful attack against any targeted system. In this case, 18 VEP were undetected by both Snort and Bro when the corresponding traffic traces were submitted to them. Thus, the analysis is inconclusive since we are not able to determine whether the IDS did not provide any alarm because they know that the VEP failed or because they do not have any signature in their database to detect those attacks. Those VEP, are therefore removed from the rest of the discussion on results.

5.2 Complete Evasion Group

Even though the VEP used to generate this data set are well-known to the IDS community, some of them are missed by the IDS. For 15 of the VEP used in this data set, both Snort and Bro seem to be "blind". Fourteen are classified in the Quiet and Complete Evasion class, and one is in the Partial Alarmist and Complete Evasion class. This is a reminder that the IDS signature database needs to be updated constantly to keep up with new attack variations.

A more in-depth analysis provides interesting results. First, VEP such as samba_exp2.tar.gz, THCIISSLame.c, lsass_ms0411.pm, DcomExpl_unixwin32.zip, and HOD-ms04011-lsarv-epx.c evade intrusion detection even though VEP related to the same Bugtraq ID are detected by both IDS.

For some VEP, only one of the two IDS was "blind" and the other was able to detect the attacks. In particular, VEP kod.c, kox.c, and pimp.c are detected by Snort but not by Bro because Bro does not have any modules to analyze IGMP (which is used in these attacks). Only one VEP, msrpc_dcom_ms03_26.pm, is not detected by Snort and detected by Bro.

5.3 Alarmist Group

One of the main results that emerged from this experiment is that both Snort and Bro are alarmist. In many situations, they raise alarms regardless of the vulnerabilities of the targeted systems and the actual success of the attack. The Alarmist group is the list of VEP that have generated IDS alarms regardless of their success or failure. In this case, a network administrator does not have any idea if the attack succeeded or not. In fact, only one of the VEP has been classified as Quiet and Complete Detection out of the 84 VEP for both Snort and Bro. Table 2 reports[2] on these VEP for which both Snort and Bro are Alarmist/Partially Alarmist. Other results (not shown here) indicate that Bro (resp. Snort) is Alarmist/Partially Alarmist for a total of 49 (resp. 68) of the 84 conclusive VEP.

[2]In Tables 2-4, we omitted the FN columns since they were filled with 0.

Some results are partially alarmists for two reasons. First, in some cases, the attack did not complete. This is what happened with smbnuke.c that does not completely execute the attack against Samba servers. Second, some available configurations of the VEP are not considered attacks. In some cases, the VEP offer to attack ports that are not checked by the IDS or in other situations, the VEP configurations are not considered attacks by both IDS.

	Snort 2.3.2			Bro 0.9a9		
VEP	**TP**	**TN**	**FP**	**TP**	**TN**	**FP**
Alarmist and Complete Detection						
0x333hate.c	6	0	52	6	0	52
0x82-Remote.54AAb4.xpl.c	4	0	112	4	0	112
msftp_fuzz.pl	4	0	77	4	0	77
msftp_dos.pl	4	0	77	4	0	77
win_msrpc_lsass_ms04-11_ex.c	13	0	59	13	0	59
wins.c	2	0	2	2	0	2
Partially Alarmist and Complete Detection						
ftpglob_nasl	8	2	68	8	2	68
msasn1_ms04_007_killbill.pm	7	24	18	7	24	18
rfparalyze	2	7	49	2	7	49
sambal.c	12	6	211	12	6	211
smbnuke.c	14	77	23	14	77	23
sslbomb.c	3	14	7	3	14	7
Alarmist (Failed Only)						
0x82-w0000u_happy_new.c	0	0	242	0	0	242
0x82-wu262.c	0	0	238	0	0	238
ms03-04.W2kFR.c	0	0	16	0	0	16
ms03-043.c	0	0	16	0	0	16
msdtc_dos_nasl.c	0	0	4	0	0	4
Partially Alarmist (Failed Only)						
ms04-007-dos.c	0	18	83	0	18	83
rfpoison.py	0	5	53	0	5	53

Table 2. Alarmist results for Snort and Bro[2]

5.4 Bro Enhancement Group

Bro provides enhancement to Snort rules when they are translated into the Bro signature language. Bro mainly provides two types of improvement to Snort rules: the error reply management and the attack server configuration context management. The error reply management is based on the hypothesis that if an attack succeeds, the server replies with a positive response such as message code 200 OK for HTTP and if the attack fails, we get an error message back from the server such as 403 Access Forbidden in the case of HTTP. The server configuration rule enhancement context is based on the hypothesis that network configuration context information such as the type and version of the attacked server could reduce false positives. In this case, experience has shown [38] that IDS can reduce the number of false positives they generate and/or prioritize alarms based on their knowledge of the network context when attacks are identified.

These improvements to the Snort rules by Bro are very effective. In fact, 30 of the results for the VEP have totally or partially moved from being classified as false positives for Snort to true negative for Bro: Snort provides results in the Alarmist / Partially Alarmist group, but Bro is able to detect that all or some attacks have failed based on the two enhancements previously discussed. Table 3 presents these results. However, some of these enhancements prevented Bro from detecting successful attacks generated by five VEP in the data set. For instance, the attack scenarios from sol2k.c, iis50_printer_overflow.pm, iiswebexplt.pl, and jill.c evade Bro detection. The enhanced version of this rule by Bro requires no error message from the server when the server is IIS. However, in the case of this particular group of attacks, no information is provided by the server to identify it as IIS when the attack is successful. Thus, the rule is never triggered and we have false negatives when the attacks are successful

m00-apache-w00t.c also evades detection when Bro is used, but for a different reason. Once again the enhanced version of the rule for detecting this attack requires no error message as a reply by the server. In this particular case, it is the type of error message given by the server that represents the exploitation of this vulnerability. This VEP is used to find the name of users allowed on the server. If a request is made with a particular user name, the error message 403 Forbidden Access specifies to the VEP that

	Snort 2.3.2			Bro 0.9a9		
VEP	**TP**	**TN**	**FP**	**TP**	**TN**	**FP**
Alarm. & Compl. Det. to Part. Alarm. & Compl. Det.						
all_uniexp.c	6	0	29	6	1	28
decodecheck.pl	7	0	31	7	28	3
decodexecute.pl	3	0	35	3	29	6
iisenc.zip	5	0	33	5	30	3
iis_nsiislog_post.pm	4	0	136	4	130	6
iis_zang.c	4	0	31	4	11	20
Lala.c	6	0	29	2	9	20
Alarm. & Compl. Det. to Quiet & Compl. Det.						
execiis.c	6	0	29	6	29	0
iis_escape_test	6	0	29	6	29	0
iisex.c	6	0	29	6	29	0
iisrules.pl	6	0	29	6	29	0
iisrulessh.pl	6	0	29	6	29	0
unicodecheck.pl	4	0	31	4	31	0
unicodexecute2.pl	4	0	31	4	31	0
Part. Alarm. & Compl Det. to Quit & Compl Det.						
iisuni.c	6	35	99	6	134	0
Part. Alarm. & Compl. Det.						
iis_source_dumper.pm	2	23	10	2	30	3
Alarm. (Failed Only) to Part. Alarm. (Failed Only)						
apache chunked win32.pm	0	0	75	0	45	30
ddk-iis.c	0	0	152	0	136	16
iis40_htr.pm	0	0	38	0	15	23
linux-wb.c	0	0	38	0	34	4
xnuser.c	0	0	38	0	27	11
Alarm. (Failed Only) to Quiet (Failed Only)						
fpse2000ex.c	0	0	38	0	38	0
wd.pl	0	0	16	0	16	0
iis_printer.bof.c	0	0	38	0	38	0
iis_w3who_overflow.pm	0	0	70	0	70	0
iis5.0_ssl.c	0	0	38	0	38	0
iis5hack.pl	0	0	38	0	38	0
windows_ssl_pct.pm	0	0	152	0	152	0
msadc.pl	0	0	31	0	31	0
rs_iis.c	0	0	35	0	26	9

Table 3. Bro enhancement over Snort[2]

this user name is valid on this server. Bro once again failed to detect successful attempts of this attack.

5.5 Snort Enhancement Group

Snort also provides a mechanism to compare server responses with the client requests. This mechanism can be achieved using the flowbits plug-ins. The *s2b* tool was not able to translate flowbits. This enhancement by Snort is also effective because 6 of the results for the VEP used to generate attacks have partially moved from being classified as false positives for Bro to true negative for Snort. Moreover, for four VEP, Snort provides better results than Bro because it was not possible to translate, even by hand, the corresponding Snort rules into Bro language. Even if Snort is able to detect that a small number of attacks generated by the VEP have failed (only against Windows NT) the number of false positives is still large. Table 4 presents these results for the flowbits enhancement. As we can see, the flowbit enhancement is minimal compared to Bro enhancement.

5.6 IDS Evasion Results

The EvaSet is used to test the accuracy of IDS in the case of attacks hidden using standard IDS evasion techniques. The objective is not to develop new IDS evasion techniques, but to show that our virtual network infrastructure is able to automatically produce attack traces used in combination with IDS evasion techniques.

The list of papers related to IDS evasion techniques described in Section 2, refers to a variety of tools [39–42] that can be used to manipulate attacks to evade detection by IDS. These tools provide an explosion of possible mutations when applied to every successful attack scenarios in our standard data. The EvaSet contains two groups of IDS evasion techniques: packet fragmentation

	Snort 2.3.2			Bro 0.9a9		
VEP	**TP**	**TN**	**FP**	**TP**	**TN**	**FP**
Part. Alarm. & Compl. Det. to Alarm & Compl. Det.						
0x82-dcomrpc usemgret.c	45	5	40	45	0	45
30.07.03.dcom.c	13	12	173	13	0	185
dcom.c	13	8	105	13	0	113
ms03-039-linux	0	2	23	0	0	25
oc192-dcom.c	13	2	21	13	0	23
rpcexec.c	4	2	30	4	0	32

Table 4. Snort enhancement over Bro[2]

[19] and HTTP evasion techniques [14, 15, 18]. They are respectively implemented using Fragroute [42] and a proxy version of Whisker [40]. These tools are proxy applications, meaning that attack traffic is captured by these tools and manipulated to apply an IDS evasion technique. They provide an efficient way for launching the same attacks from the original data set while applying IDS evasion techniques. Table 5 provides a summary of results from the analysis of Snort and Bro against EvaSet. To provide a fair analysis, only the results of IDS evasion techniques that are applied to successful attacks that have been detected by both IDS are presented.

In both cases, the IDS were able to detect the attacks even with the IDS evasion techniques, except for Unicode URI, Unicode URI (no /) and Null Method in the case of Bro. Bro does not detect both Unicode URI encoding methods since it does not have a Unicode decoder for HTTP URI. In the case of Null Method, Bro seems unable to properly decode the mutated URI as opposed to the targeted server. It is interesting to note that some VEP were not successful when they were used with some IDS evasion techniques. This provides another way to test and evaluate the detection accuracy of IDS in the case of successful and failed attack attempts. The results that are classified as true negative in the case of Bro are related to enhancements provided by Bro over the Snort rules. In this case, when the mutated attack failed it is able, in some situations, to detect it. This explains why Bro is Quiet/Partially Alarmist for this data set. The packet fragmentation problem seems to be properly handled by both IDS even in the case of the newer (frag-7-unix) and older (frag-7-windows) fragmentation problem [42]. However, for both IDS even when fragments are overlapping and the attack only affects a Windows system that favors older fragments over newer fragments, the IDS still raise an alarm. The IDS are able to properly detect the attack but are not able to know the technique used to reassemble the fragments by the targeted systems. Even if the IDS are able to properly decode the modifications made by the IDS evasion techniques, many of the results provided by Snort and Bro are again in the Alarmist/Partially Alarmist group for this set.

5.7 Results Discussion

Based on the data sets we propose, it seems that the Snort rules enhancement by Bro provide very good results to reduce false positives. It is clear by looking at the results from our data sets that Snort detects attempts rather than intrusions. In fact, Table 2 to Table 4 show that even if the attack failed, Snort still raises an alarm for 68 of the 84 VEP that provided results.

A statistical analysis of the Snort rule set seems to confirm this hypothesis: Only 379 of the 3202 rules use flowbits. 2428 of these rules are client to server rules and 167 are server to client rules (specific reaction from the server when attacked). Thus, a significant part of the Snort rule set only looks at attacks from the client to the server (mostly attempts detection) and only a few of them use flowbits or look at an effect from the server.

Figure 6 presents the comparative analysis of Snort and Bro detection rates in the case of successful and failed attacks. From Figure 6 (a), we can see that Snort has a better detection rate than Bro for successful attacks (Bro is missing the IGMP attacks and those checked using

	Snort 2.3.2	Bro 0.9a9
Fragmentation		
frag-1	Alarm. & Compl. Det.	Part. Alarm. & Compl. Det.
frag-2	Alarm. & Compl. Det.	Part. Alarm. & Compl. Det.
frag-3	Alarm. & Compl. Det.	Part. Alarm. & Compl. Det.
frag-4	Alarm. & Compl. Det.	Part. Alarm. & Compl. Det.
frag-5	Alarm. & Compl. Det.	Part. Alarm. & Compl. Det.
frag-6	Alarm. & Compl. Det.	Part. Alarm. & Compl. Det.
frag-7-unix	Alarm. & Compl. Det.	Part. Alarm. & Compl. Det.
frag-7-win32	Alarm. & Compl. Det.	Part. Alarm. & Compl. Det.
tcp-5	Part. Alarm. & Compl. Det	Part. Alarm. & Compl. Det.
tcp-7	Part. Alarm. & Compl. Det	Part. Alarm. & Compl. Det.
tcp-9	Part. Alarm. & Compl. Det	Part. Alarm. & Compl. Det.
HTTP Encoding		
Unicode URI	Alarm. & Compl. Det.	Quiet & **Compl. Eva.**
Unicode URI (no /)	Alarm. & Compl. Det.	Quiet & **Compl. Eva.**
Null Method	Alarm. & Compl. Det.	Quiet & **Part. Eva.**
Fake Parameter	Alarm. & Compl. Det.	Quiet & Compl. Det.
Session Slicing	Alarm. & Compl. Det.	Quiet & Compl. Det.
Prepend long string	Alarm. & Compl. Det.	Quiet & Compl. Det.
Premature Ending	Alarm. & Compl. Det.	Quiet & Compl. Det.
Self Reference	Alarm. & Compl. Det.	Quiet & Compl. Det.
ReverseTransversal	Alarm. & Compl. Det.	Quiet & Compl. Det.
Case Sensitive	Alarm. & Compl. Det.	Quiet & Compl. Det.

Table 5. IDS evasion results

incorrect enhanced rules). However, from Figure 6 (b) it is clear that Bro raised fewer false alarms than Snort.

We conclude that it is important for IDS to inform the administrator, even if the attack is likely to fail. One problem of automatically evaluating IDS is to make a distinction between true negative and no detection at all. When the IDS does not provide us with any indication that an attack attempt was made against a system, it is difficult to know if the IDS did not provide us any message because it found that the attempt failed or because it did not recognized an attempt had been tried.

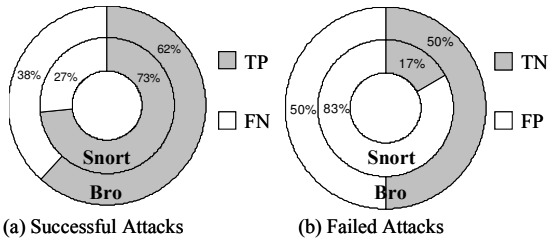

Figure 6. Detection rate analysis

6 Conclusion

We have described a Virtual Network Infrastructure to create large scale, clearly documented data sets for Intrusion Detection Systems (IDS) testing and evaluation. It allows us to record network traffic produced during attacks, control the network (e.g., traffic noise), control the attack propagation (confinement), use various heterogeneous target system configurations, and quickly recover from attacks. It is flexible (it can easily include IDS evasion techniques on attacks), updateable (it can easily incorporate new target configurations and new attacks) and completely automated. We also showed that properly documented traffic traces can be used with our intrusion detection system evaluation framework to automatically test and evaluate IDS.

The current strategies used to generate the IDS data sets have limitations. For instance, the explosion of all possible evasions applicable on an attack is infinite, it is difficult to know if our attacks are representative of attacks commonly used on the Internet and if the variation we used to exploit a vulnerability is sufficient to test and evaluate an IDS even if we produce a larger data set than the other proposed solutions. We are currently working to address these issues.

It is also difficult to properly address the performance issues of IDS with the current data set because the infrastructure is virtual. Moreover, a problem is often seen with off-line analysis of IDS using recorded traffic traces when you evaluate reactive IDS (or IDS that respond to attack). In both cases, we believe that our data set could still be used as a building block to resolve IDS testing problems. The recorded attacks could be used with tools such as TCPReplay and Opera to increase the throughput. Then, this stream could be inserted into normal traffic or traffic generated by IDS stimulators to test IDS performance under stress. In the case of reactive IDS, a system such as Tomahawk [43] could be used in combination with our recorded data set to partially address this problem.

Finally, by periodically updating and sharing the attack traces we generated with the research community in network security we could provide a common reference to evaluate intrusion detection systems.

To conclude, to the best knowledge of the authors, this is the first attempt to automatically and systematically produce attack traces and make the results publicly available to the research community since the DARPA data sets.

To obtain a version of the data set, send an e-mail to networksystem-security@crc.ca.

References

1. Lincoln Laboratory Massachusetts Institute of Technology: DARPA Intrusion Detection Evaluation (2006)
2. CAIDA: Cooperative association for internet data analysis (2006)
3. NLANR: National laboratory of network applied research, passive measurement analysis project (2006)
4. Mell, P., Hu, V., Lipmann, R., Haines, J., Zissman, M.: An Overview of Issues in Testing Intrusion Detection Systems. Technical Report NIST IR 7007, NIST (2006)

5. Beale, J., Foster, J.C.: Snort 2.0 Intrusion Detection. Syngress Publishing (2003)
6. Paxson, V.: Bro: A System for Detecting Network Intruders in Real-Time. Computer Networks 31 (1999) 2435–2463
7. Athanasiades, N., Abler, R., Levine, J., Owen, H., Riley, G.: Intrusion Detection Testing and Benchmarking Methodologies. Proc. IEEE International Workshop on Information Assurance (IWIA'03) (2003)
8. Mutz, D., Vigna, G., Kemmerer, R.A.: An Experience Developing an IDS Stimulator for the Black-Box Testing of Network Intrusion Detection Systems. Proc. Annual Computer Security Applications Conference (2003)
9. Sniph: Snot. www.securityfocus.com/tools/1983 (2006)
10. Aubert, S.: IDSWakeup. www.hsc.fr/ressources/outils/idswakeup/ (2006)
11. Giovanni, C.: Fun with Packets: Designing a Stick. packetstormsecurity.nl/distributed/stick.htm (2006)
12. The NSS Group: Intrusion Detection Systems Group Test (Edition 4) (2003)
13. Vigna, G., Robertson, W., Balzarotti, D.: Testing network-based intrusion detection signatures using mutant exploits. Proc. ACM conference on Computer and communications security (2004) 21 – 30
14. Timm, K.: IDS Evasion Techniques and Tactics. www.securityfocus.com/infocus/1577 (2002)
15. Hacker, E.: IDS Evasion with Unicode. www.securityfocus.com/infocus/1232 (2001)
16. Marty, R.: THOR - a tool to test intrusion detection systems by variations of attacks. Master's thesis, ETH Zurich (2002)
17. Handley, M., Kreibich, C., Paxson, V.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics (HTML). Proc. USENIX Security Symposium 2001 (2001)
18. Roelker, D.: HTTP IDS Evasions Revisited. docs.idsresearch.org/http_ids_evasions.pdf (2006)
19. Ptacek, T., Newsham, T.: Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Secure Networks (1998)
20. The NSS Group: Intrusion Detection Systems Group Test (Edition 3). (2002)
21. Yocom, B., Brown, K.: Intrusion battleground evolves. Network World Fusion (2001) 53–62
22. Mueller, P., Shipley, G.: Cover story: dragon claws its way to the top. Netw. Comput. 12 (2001) 45–67
23. Rossey, L.M., Cunningham, R.K., Fried, D.J., Rabek, J.C., Lippmann, R.P., Haines, J.W., Zissman, M.A.: LARIAT: Lincoln Adaptable Real-time Information Assurance Testbed. Proc. IEEE Aerospace Conference (2002)
24. Debar, H., Morin, B.: Evaluation of the Diagnostic Capabilities of Commercial Intrusion Detection Systems. Proc. RAID (2002).
25. McHugh, J.: 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 (2000)
26. De Montigny-Leboeuf, A.: A Multi-Packet Signature Approach to Passive Operating System Detection. CRC/DRDC Technical Report CRC-TN-2005-001 / DRDC-Ottawa-TM-2005-018 (2004)
27. Project, LEURRE.: *Eurecom*. http://www.leurrecom.org (2006)
28. Leita, C., Mermoud, K., Dacier, M.: ScriptGen: an automated script generation tool for honeyd. In: ACSAC 2005, 21st Annual Computer Security Applications Conference, December 5-9, 2005, Tucson, USA. (2005)
29. Vrable, M., Ma, J., Chen, J., Moore, D., VandeKieft, E., Snoeren, A.C., Voelker, G.M., Savage, S.: Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm. Proc. ACM Symposium on Operating System Principles (SOSP). (2005)
30. Jiang, X., Xu, D., Wang, H.J., Spafford, E.H.: Virtual Playgrounds for Worm Behavior Investigation. Proc. RAID (2005)
31. VMWare Inc: Vmware. http://www.vmware.com (2005)
32. Dike, J.: The User-mode Linux Kernel Home Page. user-mode-linux.sourceforge.net/ (2005)
33. Bochs: Bochs homepage. bochs.sourceforge.net/ (2006)
34. SecurityFocus: SecurityFocus Homepage. www.securityfocus.org/ (2005)
35. Project, M.: Metasploit. http://www.metasploit.com (2006)
36. Anderson, H.: Introduction to nessus. www.securityfocus.com/infocus/1741 (2003)
37. Barber, J.J.: Operator. ussysadmin.com/operator/ (2006)
38. Massicotte, F.: Using Object-Oriented Modeling for Specifying and Designing a Network-Context sensitive Intrusion Detection System. Master's thesis, Department of Systems and Computer Eng., Carleton University (2005)
39. K2: ADMmutate. www.ktwo.ca/security.html (2006)
40. Rain Forest Puppy: Whisker. www.wiretrip.net/rfp/txt/whiskerids.html (2006)
41. CIRT.net: Nikto. www.cirt.net/code/nikto.shtml (2006)
42. Song, D.: Fragroute 1.2. www.monkey.org/_dugsong/fragroute/ (2006)
43. Smith, B.: Tomahawk. tomahawk.sourceforge.net/ (2006)