Communications Research Centre Canada
An Agency of Industry Canada
Centre de recherches sur les communications Canada
Un organisme d'Industrie Canada

Passive Network Monitoring Tool-eXtended (PNMT-X): Proof of Concept

Report on research progress to November 1, 2006

Frederic Massicotte, Research Engineer

CRC Technical Note no: CRC-TN-2007-0002

Ottawa, March 2007

Canada

CRC

Passive Network Monitoring Tool-eXtended (PNMT-X): Proof of Concept

Report on research progress to

November 1, 2006

Frederic Massicotte, Research Engineer

CRC Technical Note CRC-TN-2007- 0002
Ottawa, March 2007

Abstract

Network Intrusion Detection Systems (IDS) have the reputation of generating many false positives. Recent approaches, known as stateful IDS, utilize the state of communication sessions into account to address this issue. However, for IDS to be able to distinguish between a successful and failed attack attempt, it requires a correlation among the state of the multiple sessions, the reactions of the target system and other gathered of network context information. In this report, we present initial research that supports an IDS approach that attempts to confirm attack success or failure by collecting more network context and combining this information with the attack detected information provided by the IDS. The approach relies on capturing the related effects of an attack to be able to confirm the success or failure against a target system. This approach has been evaluated using existing attacks on real systems and the observed results are positive and further work is required to refine the algorithm.

Résumé

Les systèmes de détection d'intrusion réseau (SDI) ont la réputation de générer plusieurs faux positifs. Des approches récentes, connu sous le nom de SDI à états, utilisent l'état de la communication pour tenter de résoudre ce problème. Par contre, pour que ces SDI soient en mesure de distinguer entre une attaque réussite et une attaque qui a échoué, il faudrait aussi corréler l'état de plusieurs sessions, la réaction du système sous attaque ainsi que d'autres informations à propos de la situation réseautique. Dans ce rapport, nous présentons nos résultats de recherche initiaux sur un SDI qui est en mesure de confirmer la réussite ou l'échec d'une attaque en capturant plus d'information sur la situation réseautique tout en combinant cette information avec les attaques qui ont été détectées par un SDI. Cette approche est basée sur la capture des effets reliés à une attaque pour confirmer sa réussite ou son échec contre un système. Cette approche a été testée avec des attaques connues contre des vrais systèmes et les résultats sont très prometteurs et l'algorithme doit est raffiné dans un projet futur.

Table of Contents

1 Introduction

Our belief is that much remains to be done to improve attack detection. In [1], we reported the results of an experiment during which we evaluated the response of Snort 2.3.2 [2] (a stateful IDS) and Bro 0.9a9 [3] (another stateful IDS), which are two of the most advanced and widely used open source IDS, to well-known attacks. We used 92 vulnerability exploitation programs (VEP), which implemented attacks against 85 vulnerabilities in commonly used software systems and applications. Results showed that Snort, in particular, is not able to confirm attack attempts. Moreover, Snort is not able to distinguish between a successful and a failed attack attempt. This confirmed our belief that despite the adoption of stateful approaches, much remains to be done to improve attack detection.

Specifically, we identified that: (1) the false positives problem partly lies in the inability of current IDS to incorporate intrusion alarms in an even larger network context[1]; and (2) IDS mostly rely on a single packet within one session to identify attacks, whereas detecting modern attacks requires monitoring series of packets in multiple sessions to distinguish between successful and failed attacks based on the reaction of the target system.

The Passive Network Monitoring Tool[2] Extended (PNMT-X) approach described in this report attempts to address the second issue. PNMT-X was developed at the Communications Research Centre of Canada and it can monitor complex communication patterns that involve multiple packets in multiple sessions to capture the target system's reaction during an attack.

The rest of this report is structured as follows: Section 2 describes the measures defined to compare PNMT-X with Snort and Bro. Section 3 describes the attack scenarios identified in the traffic traces of our data set. Section 4 describes our approach. Section 5 presents the results we obtained and a comparative analysis with Snort and Bro. Conclusions are drawn in Section 6.

2 Metrics

In the literature, the techniques used to establish, compare and evaluate IDS accuracy are usually ad-hoc and the metrics that are used are poorly defined. Thus, we decided to develop our own IDS accuracy metrics to address this problem and be able to compare our PNMT-X accuracy with Snort and Bro. The measures we used a *confusion matrix* to measure the accuracy of IDS. Table 2.1 is an example of a confusion matrix for IDS testing.

[1] For instance, if a packet is recognized as being a threat to a Windows machine but is sent to a Unix machine (network context), a context-based IDS can be designed to be silent (if this is acceptable to the network administrator).

[2] PNMT is the result of previous related research at the Communications Research Centre of Canada, which was extended to include the proof of concept IDS

	Event	No Event
Attack	2	1
No Attack	3	4

Table 2.1 Confusion Matrix Example

The rows are associated with the known correct attribute of the data (e.g. Attack and No Attack) and the columns are associated with predicted value provided by the IDS (e.g. Event and No Event). Thus, the accuracy in a square confusion matrix is always obtained using the sum of the value in the diagonal from the top left side to the right bottom side divided by the sum of all values in the matrix. These values represent the number of occurrences of a predicted value in relation to its real value. For example, the value in position (No Attack, Event) represents the total number of instances that an IDS has generated an attack event when there is no attack. In the example of Figure 2.1 we used 10 test cases. There are 2 times when the IDS has provided an event when there is an attack, 1 time it provided no event when there is an attack, 3 times it provided an event when there is no attack and 4 times it provided no event when there is no attack. Equation 2.1 defines this 7measure.

$$Accuracy = \sum_{i=1}^{n} M[i,i] / \sum_{i=1}^{n} \sum_{j=j}^{n} M[i,j]$$

Equation 2.1

A value of zero (0) means that the IDS providing the result is never accurate and a value of one (1) means that the IDS is always accurate. In this example, the accuracy is 0.6 (e.g. (2+4)/(2+1+3+4)) for these test cases.

In our project, we observed that most IDS events can be classified into three types: attack events, related attack events, unrelated attack events. The attack events are the events generated by IDS when the IDS is convinced that there is an attack in progress. The related attack events are events likely generated by an attack. For example, during a buffer overflow attack, a resulting command prompt event is a related attack event. The unrelated attack events provided by the IDS are not related to any part of the attack, and can be classified as noise generated by the IDS. For this first part of the project, we did focus on the ability of IDS to detect attacks, meaning only attack events were needed. We determined that an IDS is able to identify that an attack is occurring if one of the events in its log file has an association with the vulnerability exploited by the attack attempt currently generated by the test case. Contrarily, it is not able to identify an attack if there is no event logged for the attack attempt.

All events that represent attacks are grouped into four classes: the *Attempt* (A) event, the *Likely-To-Succeed* (LS) event, the *Likely-To-Fail* (LF) event and the Silence (S) (absence of) event. The latter, (S), represents the case when the IDS does not provide an event related to an attack, regardless of reason. The *Attempt* event describes an attack attempt against a target system. The *Likely-To-Succeed* event confirms that an attack is likely to succeed. The *Likely-To-Fail* event

confirms that an attack attempt is likely to fail. All attack events that are raised by an IDS for a test case are classified into one of these four classes. The total number of events in each class is associated with its position in the matrix to determine the accuracy.

We developed three metrics to determine accuracy: the detection level, the classification level, and the confirmation level. Each of these metrics represents a level of precision achieved by the IDS.

2.1 Detection Level

The detection level measures the capability of the IDS to detect attacks. The known attributes are *Normal* traffic and *Attack* traffic and the predicted values are classified into *Detected* and *Not Detected.* Thus, the test cases that have provided *Attempt* event, *Likely-To-Succeed* event or the *Likely-To-Fail* event are grouped into *Detected* and the test cases in which we have no event *(Silence)* are grouped in *Not Detected.*

Under normal circumstances, a perfect IDS would only detect attacks when it sees attack traffic and would not detect attacks when it only sees normal traffic. In our test environment data set, all the traffic traces contain attack traffic, and we do not have "normal" traffic. Therefore, we are not able to completely evaluate this aspect of the IDS accuracy. However, the attack traffic is sufficient to evaluate the accuracy of IDS to detect attacks. The accuracy for this modified metric is simply the number of *Detected* traffic traces divided by the total of number of attack traffic traces. In this case, we state that an IDS is able to detect an attack if for at least one of the test cases related to an attack it is able to provide one event associated with this attack attempt.

2.2 Classification Level

The classification level measures the ability to distinguish between successful and failed attack attempts against a target. The known attributes are *Successful* attack and *Failed* attack and the predicted values are classified into *Positive* event and *Negative* event. Thus, the events provided by the IDS are classified into two classes: the *Positive* events, the *Attempt* event and the *Likely-To-Succeed* event and the *Negative* message, the *Likely-To-Fail* event and the *Silence* (absence of) event. The accuracy is measured by the sum of *Positive* events for *Successful* attacks and the sum of *Negative* events for the *Failed* attack attempts divided by the sum of all values.

2.3 Confirmation Level

The confirmation level measures the IDS ability to confirm the success or failure of attack attempts against a target. The known attributes are *Successful* attack and *Failed* attack and the predicted values are *Attempt*, *Likely-To-Succeed*, the *Likely-To-Fail* and *Silence*. This is not a square confusion matrix and the accuracy measures have to be modified. A perfect IDS is one able to confirm all successful attacks and confirm all failed attack attempts. Thus, the accuracy is the sum of *Successful* attacks that are *Likely-To-Succeed* and the number of *Failed* attack attempts that are *Likely-To-Fail* divided by the total of all attacks.

These three level metrics will have different results based on the unit of measurement used for the test case such as traffic traces, VEP and vulnerability. Our data set is a composition of traffic traces, in which each traffic trace is associated with a unique VEP. Each VEP is associated with a vulnerability or BID (Bugtraq identification number) in the Security Focus[3] database. Thus, the accuracy can be measured using traffic traces, VEP or vulnerability taken together as a unit or we can measure the accuracy by traffic traces, VEP or vulnerability taken individually. In the case of detection level, we measure the accuracy using a VEP as the unit of measurement for our data set. In this case, it is important to remember that some IDS such as Bro and Snort are silent when they detect failed attack attempts. Thus, it is not possible to measure the detection accuracy of IDS at the traffic traces level. It needs to be measured at the VEP level.

3 Multi-Session Attack Scenarios

In this section, we examine the attack scenarios contained in our data set and the capability needed by IDS to be better able to detect, classify and confirm intrusion situations. One of the reasons that IDS are not able to provide better accuracy is their inability to see all the information related to an attack. Assuming that the IDS is properly positioned to gather the requisite traffic, it is the IDS's native capabilities that must be considered and measured. For an attack to be successful against a computer system, two elements are required: a known (by the attacker) vulnerability and the corresponding mechanism that attacks this vulnerability.

In the case of our test data set, we need a vulnerable target system or service and the corresponding group of packets for exploiting this vulnerability for the attack to be successful. Through examination of the attacks included in our data set, we believe that two aspects of attacks could be used to improve IDS recognition of network based attacks: the attack context and the reaction of the target. It is important to note that the data set used in this project has been carefully constructed and examined to make the suppositions given in this section and draw the conclusions described later in this report.

First, the context of the attack can be used to improve accuracy of IDS because vulnerabilities are associated with software versions and knowing whether this particular software version is installed and active on a system could improve IDS accuracy.

Second, monitoring the reaction of a target system to an attack can also improve IDS accuracy for network attacks. Because a group of packets sent to a target system tries to exploit a particular vulnerability, the expected response from the target system can be different, depending on whether or not the attack is successful. Moreover, we can expect that a vulnerable system and a non-vulnerable system will behave differently when the same group of attack packets is sent to each target system.

For the research activities described in this report, the focus will be on the reaction of the target because attack context information is difficult to capture as the attack context is usually not given in the attack traffic traces. From a detailed examination of all of the VEP used in our data set we identified four abstract attack scenarios. These four attack scenarios are:

[3] www.securityfocus.com

1. Direct Shell Attack Scenario,
2. Reverse Shell Attack Scenario,
3. Denial of Service Attack Scenario and
4. Standard Message Reaction Attack Scenario.

The basic premise of this research project is that a small number of "scenarios" can be identified that represents most, if not all of the attack cases currently seen on networks. If this premise is correct, then attack detection can be made to more accurate, and be used to identify (predict) the onset of new attacks without previous detailed knowledge of the new attack.

Before giving more information on the definition of these attack scenarios, it is possible to make a few general observations from work to date. *Snort* and *Bro* do not have the capacity to detect the first three attack scenarios. These IDS are only able to correlate packets within a single session. They are not able to correlate packets in more than one session or to capture network context information such as port and host state. Snort and Bro have the capability or potential to be able to detect and classify attacks that are in the Standard Message Reaction Attack Scenario. However, based on our results presented in [1], only Bro has expressed this capability in its rule set. Even with this capability, we noticed that the verification of this scenario with *Bro* is often erroneous and incomplete. *Snort* may have the potential to detect this scenario by using *flowbit*, but it is not used to detect this scenario.

In the following sections, we will present each attack scenario. The attack scenarios are decomposed into a Likely-To-Succeed scenario and a Likely-To-Fail scenario, which distinguish between the successful attack and the failed attack attempt.

3.1 Direct Shell Attack Scenario

The Direct Shell Attack Scenario describes the case of an attacker trying to open a remote shell program on the target system. First, the attacker tries to attack the target system using a particular vulnerability in a network session. Usually the attacker adds code to be executed in the attack packet using that exploit buffer overflow vulnerability. The code to be executed contains a remote command shell server that the attacker can use to access the target system. After the attack, the attacker tries to connect to another selected port on the target system. If the attack failed, the port will be closed on the target system and there will be no session between the attacker and the target system on this port. This can be repeated 0 to N time until the attack is successful, thus that the predefined port is open and the attacker is able to connect to this port with the target system. Figure 3.1 represents this attack scenario. From this scenario, a successful attack and a failed attack can be identified.

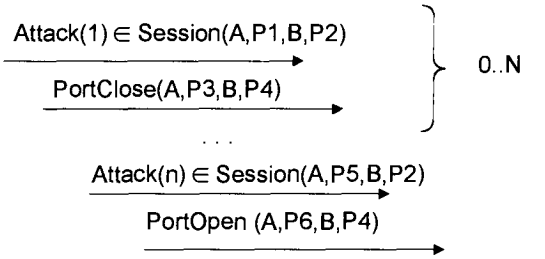

Figure 3.1 Direct Shell Attack Scenario

In the case of an attack that succeeds, we can derive two sub-scenarios. Figure 3.2 represents these attack scenarios. The *Strong* scenario specifies that the attacker has tried to connect to a predefined port but was not able to get a connection from the target system on that port until an attack attempt was tried. In the case of the *Weak* scenario, we were not able to see that the port was closed before the attack attempt. The hash line indicates optional information that can be used to increase the level of confidence that this attack was successful such as shell command or shell code identification.

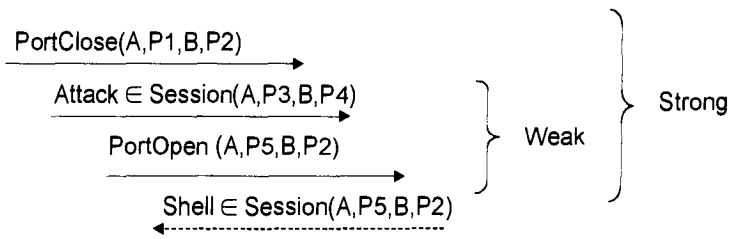

Figure 3.2 Successful Direct Shell Attack Scenario

In the case of a failure, presented in Figure 3.3 the attacker is never able to connect to this predefined port after the attack attempt.

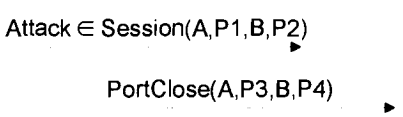

Figure 3.3 Failed Direct Shell Attack Scenario

3.2 Reverse Shell Attack Scenario

The Reverse Shell Attack Scenario describes an attacker trying to open a shell connection from the target system to the attacking system, or even a third system. First, the attacker tries to attack the

target system in a session. After the attack, it waits for a connection from the target system to connect to a predefined port on the attacking system. Usually, the attacker adds code to be executed in the attack packet that exploits a buffer overflow vulnerability in the target system. The code to be executed contains a remote command shell server that connects to the attacker and than can used to access the target system. The attack can be repeated 1 to N times. If the attack is successful, there will be a connection back from the target system to the attacker. Figure 3.4 represents this attack scenario. In this case, only a successful attack can be inferred. If the attack failed there will be no connection back from the target system to the attacker, thus a failure can not be observed by the IDS.

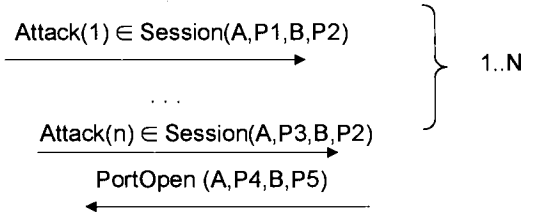

Figure 3.4 reverse Shell Attack Scenario

The success scenario, presented in Figure 3.5, shows that after the successful attack, there is connection back from the target to the attacker. The hash line indicates optional information that can be used to increase the level of confidence that this attack was successful such as shell command or shell code identification.

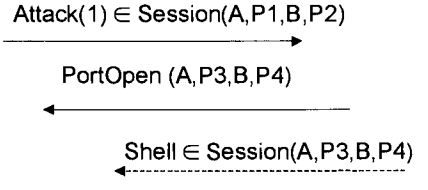

Figure 3.5 Successful Reverse Shell Attack Scenario

3.3 Denial of Service Attack Scenario

The Denial of Service Attack Scenario describes an attacker trying to cause a denial of service on a target system. First, the attacker initiates a denial of service attack on the target. If the service remains available to the attacker or other clients, the attack failed. The attacker can then try this attack from 1 to N times against the target system. If the attack is successful, the targeted system or particular service will no longer be available to the attacker and the other clients of the target system. Figure 3.6 represents this attack scenario. From this scenario a successful attack and a failed attack can be determined.

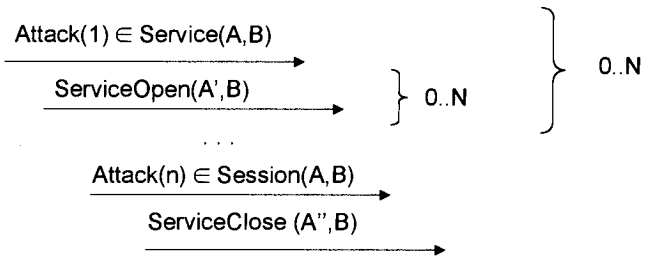

Figure 3.6 Denial of Service Attack Scenario

In the successful attack, presented in Figure 3.7, the service is no longer available to the attacker or any client of the target system after the denial of service attack.

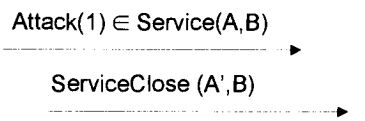

Figure 3.7 Successful Denial of Service Attack Scenario

In the failed attack, presented in Figure 3.8, the service is still available after the denial of service attack to the attacker and the other clients of the target system.

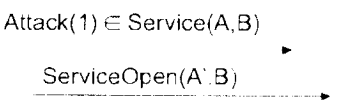

Figure 3.8 Failed Denial of Service Attack Scenario

3.4 Standard Message Reaction Attack Scenario

The Standard Message Reaction Attack Scenario describes the reaction of the attacked service to an attack attempt. First, the attacker sends an attack to the target system. In some situation such as HTTP, FTP and STMP the protocol provides standard success and/or error message to confirm the execution of a request by the client. Figure 3.9 represents this attack scenario. In many situations, this reaction from the server is a source of information concerning the success or failure of an attack. In the case of a failure of the attack, the target will respond with an error message. The attacker can try this attack from 1 to N times. If the attack is followed by a success message, we can say that the attacker has succeeded attacking the target system.

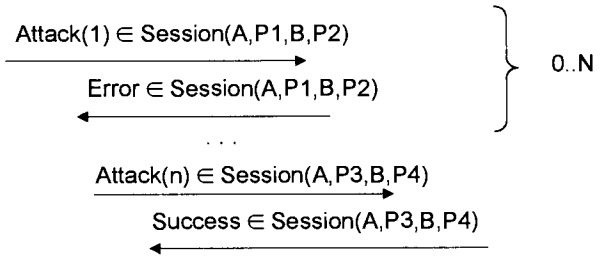

Figure 3.9 Standard Message Reaction Attack Scenario

The successful attack scenario, presented in Figure 3.10, arises when an attack is followed in the same session by a success message from the target system.

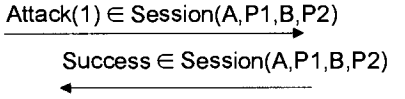

Figure 3.10 Successful Standard Message Reaction Attack Scenario

The failed attack, presented in Figure 3.11, arises when an attack is followed in the same session by an error message from the target system.

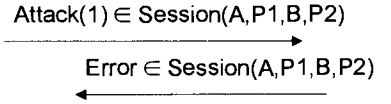

Figure 3.11 Failed Standard Message Reaction Attack Scenario

However, the situation concerning success or failure determination is not always straight forward. The error message can represents a successful attack and the success message can represent a failed attack. These situations will be further described in the next sections.

From the attacks contained in our data set, we developed a proof-of-concept[4] for an IDS that is able to detect, classify and confirm these attack attempts by using these four attack scenarios with our data set. The approach taken to implement the proof of concept is described in the next section.

[4] A proof-of-concept is a lab based system that embodies the basic research findings. It is the precursor to a prototype system, which will be field deployable.

4 Proof of Concept Design

The Passive Network Management Tool Extended (PNMT-X) is composed of two IDS engines and three IDS rule databases. Figure 4.1 represents the structure of the PNMT-X.

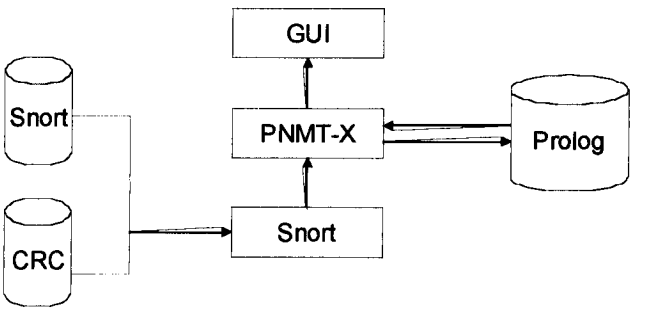

Figure 4.1 PNMT-X

There are three components: *Snort*[5], the PNMT-X engine and the user interface (GUI). In operation, the *Snort* engine provides events to the PNMT-X engine using two IDS rule databases: the default Snort rule database and the CRC Snort rule database. The default *Snort* rule database is the one that came with *Snort* 2.3.2.

The CRC *Snort* rule database was developed at CRC to allow *Snort* to provide attack context information to PNMT-X. The CRC *Snort* rule database is composed of two rule sets: the protocol rule set and the attack reaction rule set. The protocol rule set is used by the *Snort* engine to generate the events needed by the PNMT-X engine to monitor the sessions, the port states and the host states for the computer systems in the network. The attack reaction rule set is used by the *Snort* engine to generate the events needed by the PNMT-X engine to monitor the target's reaction to an attack. All the events generated by the *Snort* engine are sent to the PNMT-X engine.

In the proof-of-concept, the Snort generated events are stored in a file that is read after the entire traffic trace has been analyzed by the Snort engine. The PNMT-X engine reads the Snort events in order and verifies, using its rule database, whether or nit it can confirm that an attack attempt was successful or failed. The confirmed events are then sent to the GUI and stored in a file in XML format. The PNMT-X rule database is a Prolog rule set database that models two aspects of PNMT-X: the monitored objects and the attack scenario rules presented in the previous section.

To be able to identify the four types of attack scenarios, we need to model four objects: the *Session*, the *PortState*, the *HostState* and the *AttackAttempt* objects. We decided to use Prolog because it is a logic engine as well as a knowledge base. Thus, we could implement easily and it offer inference engine to verify if the Snort events with the gathered network context.

The Prolog database transforms each *Snort* event object using a set of rules into one of these objects. In this case, the Prolog database becomes a knowledge base that keeps track of the network

[5] Snort version is 2.3.2

context. The *Session* objects are used to monitor sessions between two computers. The *PortState* objects are used to monitor the state of a TCP/UDP port on a particular computer. The *Session* and *PortState* objects are used in the first three attack scenarios. The *HostState* objects are used to keep track of the host state and are used in the Denial of Service Attack Scenario. The *AttackAttempt* objects are used to describe the attack attempts tried against a target. These are the objects that the attack scenarios rules, defined in Prolog, are trying to confirm using the network context modeled by the Prolog database. To associate network context such as *Session*, *PortState* and *HostSate* with the *AttackAttempt*, we used temporal logic (TL). The temporal logic operators were written in Prolog to be used in the Prolog rules. To simplify the analysis we associated each Snort class with only one of the attack scenarios. In this case, the Direct Shell Attack Scenario and Reverse Shell Attack Scenario were used to confirm the *Administrative_Attack_Attempt* Snort class of events. The Denial of Service Attack Scenario is used to confirm *Denial_of_Service_Attempt Snort* class of events. The Standard Message Reaction Attack Scenario is used to confirm *Web_Attack_Attempt* Snort class of events.

For example, successful the Direct Shell Attack Scenario can written in Prolog such as

```
pnmtxEvent(Time, ID, "Direct Shell Attack", "LS", Classification, SIP, SP, DIP,
DP) :-
      snortEvent(Time, ID, Message, Classification, SIP, SP, DIP, DP),
      administrativeAttackAttempt(Classification),
      occur(previousConnection("closed", SIP, DIP, DP2), Time1, Time2),
      occur(connection("open", SIP, DIP, DP2), Time2, Time2),
      before(Time1, Time),
      before(Time, Time2).
```

This Prolog rule specifies that we have a PNMT-X *Likely-To-Succeed* Direct Shell Attack event when there is a Snort event in the knowledge that is classified as *Administrative-Attack-Attempt* and this event occurs after a port DP2 on DIP was identified to be closed for SIP and before the same port DP2 one DIP has been identified to be open for the same SIP. If this rule is true, PNMT-X identifies that the attack detected by Snort has caused this port opening on the target system and thus that the attack described by the Snort event is successful for this target system.

PNMT-X is not able to reclassify attack information when *Snort* is silent on test traffic traces because PNMT-X engine relies on *Snort* for events.

5 Results

In this section, we provide a comparative analysis of PNMT-X with *Snort* and *Bro*. *Snort* and *Bro* are not able to **confirm** any attack attempts when used with our data set. They only generate attack attempt events or they stay silent if they do not recognize any legitimate attack traffic. In the case of classification, *Snort* is able to distinguish between a successful attack and a failed attack attempt using its *flowbit* plug-in. This plug-in can be used to verify the reaction of the target to an attack. However, *Snort* did not provide useful results from our data set when compared to the approach used by *Bro* to classify attacks because Snort as opposed to Bro do not check for Standard Reaction Message to classify the attacks.

In this section we identify the accuracy improvements that PNMT-X provides by reclassifying the successful attacks, identified by *Snort* and *Bro* as attack attempts that are into *Likely-To-Succeed* and by reclassifying the failed attacks (attack attempts or by being silent) into *Likely-To-Fail*. We used 92 VEP in our data set for this analysis.

5.1 Snort Analysis

5.1.1 Complete Accuracy Maximization Class

This class regroups the VEP for which PNMT-X has provided perfect results to the limit of its design (limited to the *Snort* 2.3.2 engine detection level). In this case, there is no enhancement of PNMT-X rules possible or required. There are two cases, when this situation happens. First, no enhancement of PNMT-X rules is required, when the IDS has correctly classified all traffic traces. This situation happens when all successful attacks are classified as *Likely-To-Succeed* and when all failed attack attempts are classified as *Likely-To-Fail* by PNMT-X for all traffic traces generated by a VEP. Second, it is also not possible for PNMT-X to enhance *Snort*'s detection mechanism when *Snort* is not able to detect the attack attempt because it does not provide an appropriate event to the PNMT-X engine. In this situation, more rules have to be added to the *Snort* rule database to detect these attacks. The class *Complete Accuracy Maximization* is thus subdivided into three classes: the *Perfect Enhancement* sub-class, the *Perfect Restricted Enhancement* sub-class and the *Evasion* sub-class. There are 23 VEP in the *Perfect Enhancement* sub-class, 15 VEP in the *Perfect Restricted Enhancement* sub-class and 15 VEP in the Evasion sub-class. For 53 of the 92 VEP used in our data set, we were able to maximize accuracy with our first version of our proof-of-concept, PNMT-X.

The *Perfect Enhancement* sub-class is composed of all the VEP for which all successful attacks are classified as Likely-To-Succeed and for which all failed attack attempts are classified as Likely-To-Fail for all traffic traces generated by a VEP. Table 5.1 presents a summary of the VEP in this sub-class and the scenarios that were used to confirm all their attack attempts.

Exploit	BID	Attack Scenario			
		Direct Shell	Reverse Shell	Denial of Service	Standard Message Reaction
0x333hate.c	7294	X			
0x82-Remote.54AAb4.xpl.c	7294	X			
decodecheck.pl	2708				HTTP
execiis.c	2708				HTTP
fpse2000ex.c	2906				HTTP
IIS_escape_test.sh	2708				HTTP
iis50_printer_overflow.pm	2674				HTTP
Iisenc.zip	2708				HTTP
iisex.c	2708				HTTP
iisrules.pl	2708				HTTP
iisrulessh.pl	2708				HTTP
kod.c	514			IP	

Exploit	BID	Attack Scenario			
		Direct Shell	Reverse Shell	Denial of Service	Standard Message Reaction
kox.c	514			IP	
msadc.pl	529				HTTP
msdtc_dos.nasl	4006			TCP	
msftp_dos.pl	4482			TCP	
msftp_fuzz.pl	4482			TCP	
pimp.c	514			IP	
solaris_sadmind_exec.pm	8615	X			
solaris_snmpxdmid.pm	2417	X			
unicodecheck.pl	1806				HTTP
unicodexecute2.pl	1806				HTTP
windows_ssl_pct.pm	10116	X			

Table 5.1 Perfect Enhancement

The *Perfect Restricted Enhancement* sub-class is composed of all the VEP for which all successful attacks are classified as Likely-To-Succeed and for which all **detected** failed attack attempts are classified as Likely-To-Fail for all traffic traces generated by a VEP. The difference between this sub-class and the *Perfect Enhancement* sub-class resides in the fact that *Snort* is not able to detect some of the failed attack attempts. A thorough analysis has indicated that these undetected attacks are not legitimated attack attempts. *Snort* does not detect these VEP executions as attacks and PNMT-X behaves the same way because it does receive any event to correlate from the *Snort* engine. Table 5.2 presents the VEP in this sub-class and the scenarios that were used to confirm all possible attack attempts.

Exploit	BID	Attack Scenario			
		Direct Shell	Reverse Shell	Denial of Service	Standard Message Reaction
0x82-dcomrpc_usemgret.c	8205	X			
30.07.03.dcom.c	8205	X			
apache2.pl	2503				HTTP
dcom.c	8205	X			
iisuni.c	1806				HTTP
msasn1_ms04_007_killbill.pm	9633	X			
oc192-dcom.c	8205	X			
RFParalyze.c	1163			IP	
rfpoison.py	754			IP	
rpc!exec.c	8205	X			
samba_nttrans.pm	7106	X			

Exploit	BID	Attack Scenario			
		Direct Shell	Reverse Shell	Denial of Service	Standard Message Reaction
servu_mdtm_overflow.pm	9751	X			
smbnuke.c	5556			TCP	
warftpd_165_pass.pm	9751	X			
warftpd_165_user.pm	5556	X			

Table 5.2 Perfect Restricted Enhancement

The *Evasion* sub-class is composed of the VEP that were not detected by *Snort*. In this case, PNMT-X is not able to enhance detection accuracy. Table 5.3 presents a summary of the VEP in this sub-class and an estimated list of attacks that might have been used to increase the accuracy of *Snort* if it was able to detect these attack attempts.

Exploit	BID	Attack Scenario			
		Direct Shell	Reverse Shell	Denial of Service	Standard Message Reaction
DcomExpl_UnixWin32.zip	8205	X			
HOD-ms04011-lsasrv-expl.c	10108	X			
HOD-ms04031-expl.c	11372	X			
Lsass_ms04_011.pm	10108	X			
ms05_039_pnp.pm	14513	X			
msrpc_dcom_ms03_026.pm	8205	X			
mssql2000_preauthentication.pm	5411	X			
mssql2000_resolution.pm	5311	X			
MultiWinNuke.c	6005			IP	
Samba_exp2.tar.gz	7294	X			
THCIISSLame.c	10116	X			
Winnuke._eci.c	2010			IP	
winnuke.c	6005			IP	
winnuke.pl	2010			IP	
zp-exp-telnetd.c	3064	X			

Table 5.3 Evasion

5.1.2 Partial Accuracy Maximization Class

This class describes the VEP attacks that PNMT-X can partially confirm success or failure of the attack. There are two reasons why PNMT-X is not able to completely improve its confirmation accuracy when using *Snort* as a source of events: the attack classification and the missing reaction from the target.

Attack Classification by Snort

To simply our proof-of-concept, PNMT-X, we associated each *Snort* class of attack with only one of our attack scenarios. Although this is not ideal for our research purposes, it did allow us to avoid some problems with how *Snort* classifies attacks. For instance, one problem occurs when a particular technique that exploits the vulnerability that is associated with a *Snort* class of attack, but in fact produces a behavior detected by another scenario. This occurs in the case of all attacks associated with BID 2674 because the attacks are associated in *Snort* with the *Web_Attack_Attempt* class, and thus with our Standard Message Reaction Attack Scenario. However, its exploitation technique is to execute code on the target and thus these attacks behave as an attack by the Direct Shell Attack Scenario or the Reverse Shell Attack Scenario. In this case, this particular attack signature should have been classified in the *Administrative_Attack_Attempt Snort* class.

In a similar way, the *Snort Administrative_Attack_Attempt* class of attack could also be included in the Denial of Service Attempt class because some VEP exploit the same vulnerability to cause a denial of service. PNMT-X is not able to confirm the success or failure of these attacks because the IDS events provided by *Snort* are not in the proper classification or they should have been included into more than one *Snort* class.

Missing Reactions

The PNMT-X is not able to confirm the success or failure of an attack attempt when there is no reaction from the target to that attack. This situation arises in the case of the Reverse Shell scenario when the attack failed because the expected reverse shell session is not initiated by the attacked target. In this case, PNMT-X is not able to confirm the failure of the attack with the current version for the scenarios. This is the case for the traffic traces generated by the *wins.c* and the *wins_ms04_045.pm* where PNMT-X is able to confirm all successful attacks because we correlate the reverse shell with the attack attempt, but we are not able to confirm the failure when there is no shell session initiation from the target for these particular VEP that used a reverse shell code an their attack attempt.

It also happens in the case of the Standard Message Reaction Attack Scenario. Some of the attacks generated by some VEP do not trigger any respond message from the targeted server. In some of these cases, it is when the attack is successful, such as for *jill.c* and *sol2k.c* and in other of these cases it is when the attack failed. This missing information from the server does not trigger the Standard Message Reaction Scenario and PNMT-X is not able to confirm the success or failure. Table 5.4 presents a summary of the VEP and the scenarios that were used to confirm some of the attack attempt for this particular VEP in our data set.

		Attack Scenario

		Direct Shell	Reverse Shell	Denial of Service	Standard Message Reaction
ALL_UNIEXP.C	1806				HTTP
Apache_chunked_win32.pm	5033				HTTP
DDK-IIS.c	4485				HTTP
iis_nsiislog_post.pm	8035				HTTP
iis_w3who_overflow.pm	11820				HTTP
iis-zang.c	1806				HTTP
jill.c	2674				HTTP
lala.c	2708				HTTP
linux-wb.c	7116				HTTP
m00-apache-w00t.c	3335				HTTP
rs_iis.c	7116				HTTP
sambal.c	7294	X			
sambash.c	7106	X			
sol2k.c	2674				HTTP
win_msrpc_lsass_ms04-11_Ex.c	10108		X		
wins.c	11763		X		
wins_ms04_045.pm	11763		X		
Xnuxer.c	7116				HTTP

Table 5.4 Partial Accuracy Maximization

5.1.3 No Improvement Class

This class describes the VEP that PNMT-X is not able to confirm, for all their attack traffic traces, at least one successful attack or one failed attack attempt. Two reasons, the *attack classification* and the *missing reaction* from the target, described in the previous section also applies in this situation to explain the inability of PNMT-X to confirm successful attacks or failed attack attempts. In this case, these explanations can relate to all attack traffic traces of a VEP in this class. There is also two other explanations for this situation: unimplemented scenarios and unimplemented rules.

Unimplemented Scenarios

In the case of the unimplemented scenario, we decided, for this first version of the PNMT-X, to focus on the HTTP Standard Message Reaction alone to verify if the Standard Message Reaction Attack Scenario was possible to implement in our proof of concept IDS. Thus, FTP, POP, IMAP and STMP standard error message have not been implemented in the Standard Message Reaction Attack scenario. This explains why PNMT-X is not able to confirm the success or failure of attack that exploits FTP and SMTP vulnerabilities. In the case of POP and IMAP, we do not have any attacks that exploit a vulnerability related to these protocols.

Unimplemented Rules

Unimplemented rules only affects the Denial of Service Attack Scenario. In our data set, we noticed that a denial of service mainly affects three different levels of the communication stack: the IP level, the TCP level, the Application level. In the case of the IP level, the computer is no longer able to communication using its IP stack. In the case of the TCP level, the port where the attacked service is running in no longer opened for communication. The effect at the application level shows that the port is still open but the targeted application is no longer able to communicate using this port. In our proof of concept, we only implemented the IP and TCP communication effect of a denial of service. Thus, the attacks that affect only the application aspects of the communication are not confirmed by PNMT-X. Table 5.5 presents a list of the VEP in this class, with the scenario that should have been used to confirm the attack attempt.

Exploit	BID	Attack Scenario			
		Direct Shell	Reverse Shell	Denial of Service	Standard Message Reaction
0x82-WOOoou~Happy_new.c	8315				FTP
0x82-wu262.c	8315				FTP
7350oftpd.tar.gz	2124				FTP
bid3581.txt	3581				FTP
bysin2.c	7230				SMTP
crash_winlogon.c	1331			IP	
ftpglob.nasl	3581				FTP
iis_printer_bof.c	2674				HTTP
IIS5.0_SSL.c	10115			Application	
iis5hack.pl	2674				HTTP
iiswebexplt.pl	2674				FTP
MS03-039-linux.c	8459	X			
MS03-04.W2kFR.c	8459	X			
ms03-043.c	8826	X			
MS04-007-dos.c	9635			Application	
samba_trans2open.pm	7294	X			
sslbomb.c	10115			Application	
wd.pl	7116				HTTP

Table 5.5 No Improvement

5.1.4 Misclassification Class

This class describes the VEP that PNMT-X misclassified some successful attacks as Likely-To-Fail or some failed attack attempts as Likely-To-Succeed. There are two reasons for this situation: the sub-attack success and the misleading success message from the target.

Sub-attack Success

Sub-attack success arises when a VEP uses several different sub-attacks to exploit the same vulnerability within the course of the VEP attack. For example, *decodexecute.pl* first verifies whether or not the exploited vulnerability is present on the target by executing a directory listing. If the system is vulnerable it will respond. If the first attempt is successful the VEP will try to execute a command on the target system. In some situations, the first command is successful, but not the second one. In this case, the VEP is not successful, but part of the attack was successful. The PNMT-X system is able to confirm that the traffic trace contains a successful attack. However, our VEP traffic traces classification system that relies on the VEP output is not able to make the distinction that part of the attack was successful and this attack was classified as failed by our system.

Misleading Success Message

In the case of *iis_source_dumper.pm* and *iis40_htr.pm*, it seems that some target systems respond with a success message for the attack command sent by the attacking program. Thus, the type of standard message can not be used in these cases to distinguish between a successful and failed attack attempt. It could be possible to improve the accuracy in the case of *iis_source_dumper.pm* as the success message from the target contains information that the request was not executed successfully. This message contains contradictory information and this misclassification can be prevented by looking at the type of the message, and also at the part of the message containing the failed execution information. Table 5.6 presents a list of VEP in this class with the scenario that should be modified to accurately confirm the attack attempt.

Exploit	BID	Attack Scenario			
		Direct Shell	Reverse Shell	Denial of Service	Standard Message Reaction
decodexecute.pl	2708				HTTP
iis_source_dumper.pm	1578				HTTP
iis40_htr.pm	307				HTTP

Table 5.6 Misclassification

5.2 Bro Analysis

In this section we will provide a comparative analysis of PNMT-X with *Bro*. We know that *Bro* when compared to *Snort*, provides a better mechanism to improve detection accuracy especially in the case of the Standard Message Reaction Attack scenario. Because our previous analysis has shown that *Bro* is only better than *Snort* in the case of attacks that trigger standard protocol messages from the targets, we will only compare the results of PNMT-X for these VEP. This analysis includes 15 of the 92 VEP in our data set.

It is important to mention that for the three classes of results presented in this section PNMT-X always provides better results than *Bro* because *Bro* is not able to confirm attacks when compared to PNMT-X. However, *Bro* is able to classify the results by distinguishing successful attacks from failed attacks by providing an attack attempt message for a successful attack and being silent for a failed attack. This philosophy has some problems. It is not possible in this case to distinguish failed attack attempts from the attacks that could not be detected by the *Bro*. To further examine this issue, we did a manual analysis of the silent results provided by *Bro* to list the VEP traffic traces that *Bro* has identified as failed attack attempts and the attacks that it is not able to detect.

The accuracy improvement that PNMT-X provides is the reclassification of the successful attack identified by *Bro* as Likely-To-Succeed and the reclassification of the failed attack identified by *Bro* as a failure by being silent or the attack attempt as Likely-To-Fail. In this section, we try to determine if our Standard Message Reaction Attack Scenario is better than the one implemented by *Bro* for these VEP. We already know *Bro* has problems with its approach of looking at standard protocol messages from the target when it is attacked by a VEP. There are two main problems with *Bro's* approach for handling the VEP in this data set. First, they have assumed that for all the *Bro* rules using this functionality that if the attack sent to a target is successful, it will respond with a success message and if the attack failed it will respond with an error message. This assumption is not true because for some successful attacks the target will respond with an error message and for a failed attack attempt it will respond a success message. For *m00-apache-w00t.c* a response of 403 Access Forbidden is a success because with this message it knows that a user is on the target system. In some cases, the target will not respond at all. Because of this, modifications had to be made to this concept for PNMT-X and exception rules had to be written to take these situations into account. The results provided by PNMT-X on these VEP are classified into four classes: the Improvement class, the Perfect Enhancement class, the Partially Enhancement class and the No Improvement class.

5.2.1 Improvement Class

The Improvement class presents the VEP that PNMT-X provided better expectations than anticipated based on the *Bro* classification accuracy. These are all the VEP attacks that are not detected by *Bro* because of the problem with the standard protocol message discussed in the previous section. Table 5.7 presents a list of these VEP.

Exploit	Bid
apache2.pl	2503
iis_printer_bof.c	2674
iis40_htr.pm	307
iis50_printer_overflow.pm	2674
iis5hack.pl	2674
iiswebexplt.pl	2674
jill.c	2674
m00-apache-w00t.c	3335
sol2k.c	2674

Table 5.7 Improvement

5.2.2 Perfect Enhancement Class

The Perfect Enhancement class represents all the VEP attack traffic traces for which PNMT-X was able to confirm that all successful attacks detected as an attack attempt by *Bro* were Likely-To-Succeed and for which PNMT-X was able to confirm that when *Bro* is silent, these attacks were Likely-To-Fail. Table 5.8 presents the list of VEP that PNMT-X is able to confirm.

Exploit	Bid
ALL_UNIEXP.C	1806
Apache_chunked_win32.pm	5033
DDK-IIS.c	4485
decodecheck.pl	2708
decodexecute.pl	2708
execiis.c	2708
fpse2000ex.c	2906
iis_escape_test.sh	2708
iis_source_dumper.pm	1578
Iisenc.zip	2708
iisex.c	2708
iisrules.pl	2708
iisrulessh.pl	2708
iisuni.c	1806
iis-zang.c	1806
lala.c	2708
linux-wb.c	7116
msadc.pl	529
rs_iis.c	7116
unicodecheck.pl	1806
unicodexecute2.pl	1806

Table 5.8 Perfect Enhancement

5.2.3 Partial Enhancement Class

The Partial Enhancement class describes the VEP that PNMT-X can only partially confirm the success or failure of a VEP against a target. There is mainly one reason for which PNMT-X is not able to completely enhance its confirmation accuracy compared to *Bro*. In some cases, the attacks do not trigger any response, thus we are not able to use the standard protocol message to confirm the success or failure of the attack.

Furthermore, some *Bro* rules require the knowledge of the version and the product name of the target to raise an event to the IDS administrator. *Bro* captures this information in the banner sent by the target, but, it is not able to capture the banner and identify the version and the product name of the target when there is no response. This effect could favor *Bro*'s accuracy when the attack failed

if the target did not respond as *Bro* will be silent, which is the same behavior when it detects no attack or a failed attack attempt. However, when the attack is successful *Bro* also waits for the banner to make a decision, thus if there is no response it does not detect the attack attempt. Table 5.9 presents the list of VEP that PNMT-X is able to confirm their execution.

Exploit	Bid
iis_nsiislog_post.pm	8035
Xnuxer.c	7116

Table 5.9 Partial Enhancement

5.2.4 No Improvement Class

The No Improvement class describes the attacks that we were not able use to improve the accuracy of the PNMT-X compared to *Bro*. This result is associated with the absence of a reaction from the target for all traffic traces within a particular VEP. Table 5.10 presents the list of the VEP that cause the target to not generate a response for an attack.

Exploit	Bid
iis_w3who_overflow.pm	11820
wd.pl	7116

Table 5.10 No Improvement

6 Summary

Herein, we will present the results, in Table 5.11, of this analysis using the metric presented in Section 2. We are able to see that PNMT-X is more accurate than *Snort* and *Bro* for the classification and for confirmation of attacks by group by VEP or traces. The difference between VEP and traces in the table is the unit used. In the case of VEP, we grouped all traffic traces by VEP and we measure the accuracy for each VEP and then we average all VEP results to get the accuracy of each tested IDS. In the case of using trace units, we simply measure the accuracy by using all traffic traces. The table shows the same detection accuracy as *Snort* because we rely on *Snort* for providing detection information that it is able to find in the traffic traces. The other two IDS are not able to confirm any of the attacks in the data set. However, one interesting aspect is that *Snort* is better than *Bro* for detecting attacks, but *Bro* is better than *Snort* for classifying attacks. Thus, one would think that both *Snort* and *Bro* working together could lead to better surveillance of a network.

IDS	Detection Accuracy	Classification Accuracy VEP (trace)	Confirmation Accuracy VEP (trace)
Snort	84 %	27 % (29 %)	0 % (0 %)
Bro	71 %	60 % (50 %)	0 % (0 %)
PNMT-X	84 %	72 % (77 %)	50 % (54 %)

Table 5.11 IDS Accuracy Analysis

7 Conclusions

In this report, we presented a new set of metrics to compare results of IDS when used against our data set, a set of attack scenarios and a new IDS able to understand these scenarios. The metrics have been proven to be useful when comparing the accuracy of PNMT-X with Snort and Bro. We determined that *Snort* is better than *Bro* to detect attacks, but *Bro* is better than *Snort* to classify attacks. Moreover, *Snort* and *Bro* were not able to confirm attack attempts with our data set. However, the PNMT-X was able to use the attack scenarios to classify and confirm more attack attempt than *Snort* and *Bro*.

We have identified several problems with our approach: the Snort class type problem, the exception problems, the missing scenario problem and the scenario identification problem. The used of Snort event types to associate the events with our attack scenarios is only a one to one relationship. Thus, some attack attempts were not classified or confirmed. Thus, a one to many relationship has to integrated in our IDS for the Snort class type or simply associate each Snort event with many scenarios. In this case, it will be problematic to measure the classification and confirmation. One attack could be associated as *Likely-To-Fail* for the denial of service scenario and as an *Attempt* for the code execution scenario. We know that the attack as only one intention to cause a denial of service and not execute code, but PNMT-X will classify this attack attempt as *Attempt* and not *Likely-To-Fail* because it do not know the intention of the attack. Some attack exceptionally do not exactly match to the attack scenario, a solution need to be found to identify these attack attempts to be able to classify and confirm them. Some attack scenarios are missing such as the Standard FTP message Attack Scenario and the Application Denial of Service Scenario. These attack scenarios need to be implemented in PNMT-X. It has been very difficult to identify all the scenarios, we believe that an automatic way to identify the scenarios in the data set is needed. This automatic way to identify scenarios could also be used to resolve the exception problems and the missing scenario problems.

Based on the results obtained in this project, we plan to develop a prototype that will use these same attack scenarios. In addition, we will work to improve detection classification and confirmation rate for the attacks that were not able to classify and confirm with our data set.

8 References

[1] Massicotte, F., Gagnon F., Labiche, Y., Couture, M., Briand, L.: Automatic Evaluation of Intrusion Detection System *Proceedings of the Annual Computer Security Applications Conference (ACSAC)* Miami, Florida December 2006

[2] Green, C., Roesch M.: The Snort Project: version 2.3.2, User Manual, www.snort.org, 2005.

[3] Paxson, V.: The Bro Intrusion Detection System Project: version 0.9a9, User Manual, ww.bri-ids.org, 2005.