Context-Based Intrusion Detection Using Snort, Nessus and Bugtraq Databases

Frédéric Massicotte[1,2]
Mathieu Couture[1]
[1] Communications Research Centre
3701 Carling Avenue
P.O. Box 11490, Stn. H
Ottawa, ON K2H 8S2
Canada
http://www.crc.ca

Lionel Briand[2],
Yvan Labiche[2]
[2] Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, ON K1S 5B6
Canada
http://www.sce.carleton.ca/squall

E-mail: {frederic.massicotte,mathieu.couture}@crc.ca

Abstract

Intrusion Detection Systems (IDS) use different techniques to reduce the number of false positives they generate. Simple network context information such as the communication session state has been added in IDS signatures to only raise alarms in the proper context. However, this is often not sufficient and more network context information needs to be added to these Stateful IDS (SIDS) signatures to reduce the number of false positives. IDS are also used with other network monitoring systems such as Vulnerability Detection Systems (VDS) and vulnerability databases in centralized correlation systems to determine the importance of an alarm. The correlation mechanism relies on the accuracy of a standardized relationship between IDS signatures, VDS signatures and the vulnerability databases. In this paper, we study the strength of the relationships between Snort signatures, Nessus scripts and the Bugtraq vulnerability database, as well as their potential for information correlation and for deriving network context that could be incorporated in intrusion detection signatures.

1 Introduction

Intrusion Detection Systems (IDS) have the reputation of generating many false positives. To address the issue, IDS vendors have provided a number of options. First, they have proposed to add more context to their intrusion detection signatures by using a stateful approach to intrusion detection for communication sessions. However, a closer examination shows that SIDS also need to be able to acquire more network context to correlate it with potential intrusions. Information such as the configuration of a system, its operating system, its role in the network, its active services are also needed by IDS to derive the context of an intrusion.

Correlation systems represent a second approach proposed by vendors. These systems are used with IDS and other network monitoring tools, such as passive and active Vulnerability Detection System (VDS), to correlate IDS alarms with vulnerabilities discovered by the VDS. Correlation systems gather events from different sources, IDS, VDS, etc. and they correlate IDS alarms with a larger network context than the one inferred by an IDS alone.

Despite the adoption of correlation systems and of stateful approaches, the problem of false positives generated by IDS still lies in their inability to incorporate intrusion alarms in a larger network context. Typically, the monitoring engine and signature language are unable to fully incorporate the context and the state of the network in their intrusion signatures.

In [14], we present a passive information gathering language based on the Object Constraint Language (OCL) [16] that infers, from a sequence of packets, the network context and that incorporates this network context in intrusion detection signatures. We use this language in our Passive Network Monitoring Tool

(PNMT) [8, 9, 14][1], which, based on its multi-packets engine and network context acquisition capability, is able to identify intrusions within a larger network context than IDS.

A lot of the network context that needs to be incorporated in the IDS signatures of PNMT can be inferred from the same packet sequence used to identify intrusions. Thus, an IDS with network context acquisition capability would be able to derive the context of an intrusion, correctly place the attack in the context and thus reduce the number of false positives by using minimal information from other sources. However, other information sources than IDS signatures are needed to incorporate the network context in the IDS signatures and correlate network context with static information such as vulnerabilities.

To develop an IDS rule database with network context for PNMT and prove the concept, information from different vulnerability databases is needed. Relationship information from IDS signatures, VDS scripts and information from vulnerability databases needed have to be analyzed and incorporated into the IDS.

In this article, we present the results of a survey conducted at the Communications Research Centre (CRC)[2] on the strength of the relationships between IDS signatures, VDS scripts and vulnerability databases. These results are used to determine how this information can help reduce IDS false positives by either populating the network context with static information, adding network context information to the IDS signatures or correlating IDS alarms with VDS. This survey complements the ongoing development of PNMT at CRC.

Section 2 describes the work conducted on the different models used for network context gathering and event correlation between IDS, VDS and vulnerability databases. Section 3 describes the network model that is used by PNMT to model the network context. Section 4 provides a brief description of the different sources of information used in this analysis, a description of their relationships and how this information can be used to provide IDS rules with network context. Section 5 describes the results of the analysis of sample IDS rules and their references to Nessus and Bugtraq databases. Section 6 describes intrusion signatures with network context. The last section concludes this article by outlining future work that will follow this project.

[1]This has been tested using a proof of concept version of PNMT to validate the prototype version still under development.

[2]The Communication Research Centre is an Agency of Industry Canada.

2 Related Work

2.1 Network Model

As far as can be determined from related research products, few methods have been proposed to model a computer network from a perspective useful to our research activities. The most important contributions come from Vigna [19], Goldman et al. [11] and M2D2 [15]. M2D2 is probably the most formal and complete of these three network models. It is mostly inspired from the two other network models and provides a network model for information correlation with the full network context. In particular, the Vigna model was used in [18] to develop a correlation system and in [20] for use in an intrusion detection system. However, those models do not provide all the flexibility needed for use in a real-time monitoring system such as PNMT. They are designed for ideal situations where the network context is entirely known or where the network model is only used in a correlation system. Thus, they were not developed to add network context to an IDS signature language.

For our needs, M2D2 is the most appropriate network model. However, it must be modified to model the different characteristics of the information we have to handle. In particular, it does not currently provide a way to capture static and dynamic information. Information such as the relationship between a known vulnerability and a particular software is static, while other information such as the relationship between a network interface and an IP address is dynamic. A particular vulnerability will always be associated with a particular version of a product, but a network interface could change its IP address.

2.2 Network Model Requirements

Based on our review of the literature on network models that are used for network information management systems and event correlation systems, we have defined a set of properties that reflects what a network model needs to provide to enable network context gathering for IDS. We believe that the most relevant requirements for a network model are the following:

- A network model should be expressed formally and independently from the information gathering engine.

- A network model should be used with a rule specification language.

- A network model should be easily extendable and scalable to add new network entities in the model to derive new information.
- A network model should be able to model the different types of information required in a real-time system such as static and dynamic associations between elements of the network model.

3 Network Model

To model the network, we incorporated some of M2D2's relationships as Unified Model Language (UML) [17] associations between classes. This UML network model, combined with the use of OCL [16], meets all the requirements proposed in the previous section. While the M2D2 model is designed to express relationships between exploits, vulnerabilities and the different systems located in the network, our proposed model was designed to provide a framework for event correlation in the case of intrusion detection. Hence, modifications to the M2D2 model had to be made. In this section, we present our network model and we show how it can be populated to capture the network context.

3.1 Network Model Overview

As already mentioned, the M2D2 model is able to express static relationship between exploits, vulnerabilities and products such as operating systems and services. However, the M2D2 model does not completely meet our needs for a number of reasons. First, the model does not use an object-oriented concept to model the network. We think that networks are more appropriately modeled as objects for two reasons: because network components are objects; and because the inheritance relationship between objects is well-suited to expressing different sub-categories of network components with the same behavior. Second, the representation of dynamic information in M2D2 is not strong enough to allow real-time passive network information gathering. As a result, we developed our own network model based on concepts derived from M2D2. Figure 1 presents our network model. This model is used to capture the network context and to express the information gathering rules using OCL with the packet model defined in [14].

3.2 Dynamic Information

Figure 1 defines the relationships between network system configurations and their potential vulnerabilities and exploits. In particular, based on information about the configuration of a system, such as its installed products and its active services, it is possible to know the potential exploits and vulnerabilities that could affect it.

The *IPStack*, *Session*, *Port*, *Interface*, *Host* and the *Alarm* classes and the associations between them are dynamic information. The associations between these classes and the *Exploit* and *Product* classes are also dynamic information. The *Host* class models each component communicating on the network. This class includes infrastructure equipment such as routers, switches, etc. as well as workstations and servers. It is composed of a set of *Interface* objects that allows hosts to communicate on the network. All interfaces include a set of *IPStack* objects. The *IPStack* class models the configuration of the IP communication stack. This class is composed of IP stack configuration information as well as of a set of *Product* objects that are installed on the system using this IP stack. The *IPStack* class is also composed of a set of *Port* objects that represent, with the *IPStack* objects, *Session* objects. The *Port* class is also composed of a *Product* object to capture information such as the fact that a port is associated with a product (e.g., a Microsoft IIS FTP server running on port 21 of a particular host).

To enable the monitoring system to infer this dynamic information and to populate the model to reflect the current network context, information-gathering rules are needed. A description of the information-gathering rules used by PNMT is provided in [14].

3.3 Static Information

The other classes of our model (*Exploit*, *Vulnerability*, *Product*, *Reference* and *Vendor*) represent static information. For instance, the relationship between a product version, a known exploit and a known vulnerability is static. A product will always be vulnerable with a particular exploit if the network administrator does not update the product.

The *Exploit* class describes the characteristics of an exploit. The *Exploit* objects affect different vulnerabilities. In fact, those *Vulnerability* objects can have *Reference* to other vulnerabilities and they can also be associated with several *Product* objects. In particular, the recursive relationship between products reflects the fact that products are associated with one another. For example, such an association is used to model the fact that Internet Explorer runs on Windows operating systems. The relationship between *Product* and *Vendor* models the static relation between a product, such as Windows, and its vendor, Microsoft.

To use dynamic and static information properly,

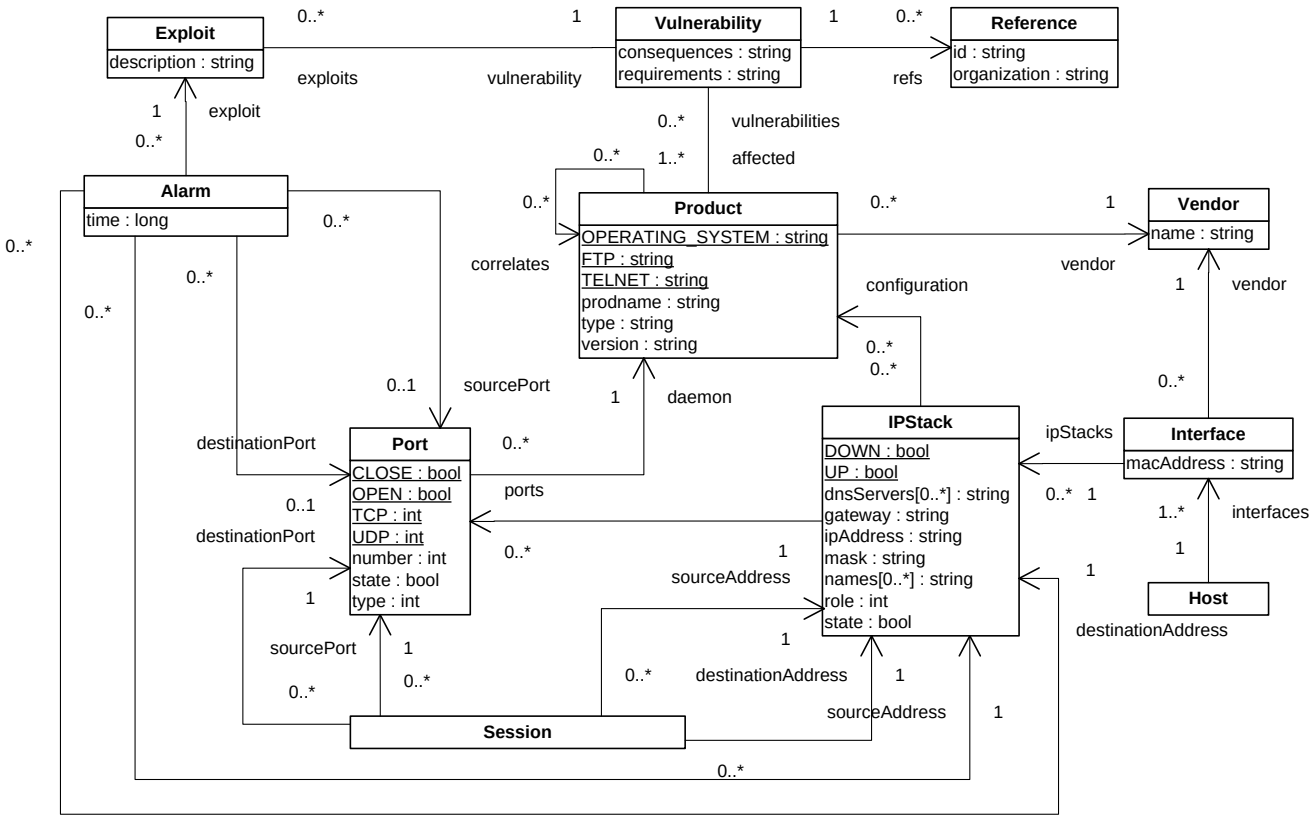


Figure 1. Network Model.

PNMT requires dynamic information, such as IDS signatures with network context, to be defined and static information to be contained in the network. To provide it with these data, we analyzed IDS signatures, VDS signatures and vulnerability databases from different sources.

4 Information Sources

To develop PNMT and to be able to include more network context in IDS signatures, the relationships between exploits, vulnerabilities and products are needed. Intrusions are implementations of known vulnerability exploitations. These vulnerabilities are usually associated with a product, such as an operating system, a service, equipment (router, switch), a particular configuration, etc.

For example, we assume that a given sequence of packets could exploit a particular vulnerability. We also assume that this attack could only be successful if these packets are sent to a target that has the proper operating system, given there is an active session between the intruder and the target and given the target is running a particular service.

Information gathering techniques and rules to populate the dynamic part of the network model are discussed in [8, 9, 14]. However, two key pieces of information are still needed: Information sources to add network context to standard IDS rules and information sources to populate the static part of the network model.

There are few open IDS signature databases and vulnerability databases that provide the information needed for this survey. On the other hand, efforts to create standardized information sources for intrusion rules and vulnerability descriptions, such as Snort [5], Nessus [3], Bugtraq [1] and CVE [2], make the correlation of events between those systems possible to a certain extent. In this section, we briefly describe these information sources as well as the information needed to integrate network context in IDS signatures and to populate the static network context information in the network model. Table 1 provides a brief description of each source of information.

4.1 Snort Rules

Snort [7, 12] is the foundation stone of open source IDS software. It offers a lightweight, efficient and flex-

Information Source	Entries	References to	Information Provided	Web site
Snort 2.3.2	2648	Nessus Bugtraq CVE ArarchnIDS McAfee	Exploit characteristics References to vulnerabilities	`http://www.snort.org/rules`
Nessus (NeWT 2.1)	5955	Bugtraq CVE	How to discover vulnerabilities How to identify products Vulnerability and product associations	`http://www.nessus.org/plugins`
Bugtraq	13092	CVE	Vulnerability descriptions Vulnerability and product associations	`http://www.securityfocus.com/bid`
CVE	9826	Bugtraq ISS FreeBSD Cisco ...	Vulnerability descriptions	`http://www.cve.mitre.org`

Table 1. Intrusion and Vulnerability Database Descriptions Summary

ible intrusion detection engine with a complete intrusion signature rule set. The Snort 2.3.2 (which is the version we used in this study) signature database provides more than 2648 signatures. Since version 1.8, Snort uses references to other vulnerability databases and IDS signature databases such as Nessus, Bugtraq, CVE, ArarchnIDS and McAfee. Because the ArarchnIDS database is old and contains only 555 vulnerabilities and since there are few references from Snort to McAfee, it was decided to only use the Nessus, Bugtraq and CVE databases in this survey. The IDS signatures included in these information sources will provide the static information on exploit characteristics needed for our model. Thus, information from Snort signatures and its references will provide us with the relationships between exploits and vulnerabilities.

4.2 Nessus Scripts

Nessus [6] is an open source Active Vulnerability Detection System maintained by Tenable. Tenable also offers a commercial Passive Vulnerability Detection System, NeVO [10, 13, 4], but it is not freely available. In this study, we used the current Windows version of Nessus: NeWT 2.1. The Nessus script database contains about 5955 scripts that are used to identify vulnerabilities. The Nessus scripts offer references to Bugtraq and CVE. These scripts can be used to define how to discover vulnerabilities and product versions. They could also be used to define relationships between vulnerabilities and products.

4.3 Bugtraq Vulnerability Database

Bugtraq [1] is a database of known vulnerabilities. It contains detailed information about the description of the vulnerability, the products they affect, their known exploits and the ways to prevent these vulnerabilities. The Bugtraq database contains more that 13 092 vulnerabilities. However, Bugtraq does not provide information on how to discover the vulnerability and it only references CVE.

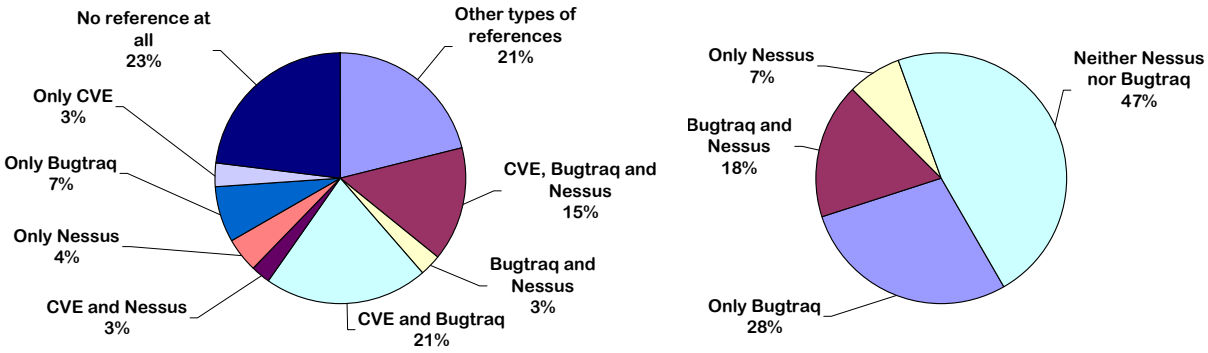
4.4 CVE Dictionary

CVE [2] is a dictionary that assigns to each vulnerability a unique number and a description. The vulnerability number is separated in three parts. The first part is the prefix CAN or CVE for candidate and confirmed vulnerabilities respectively. The second part is the year that the vulnerability was discovered. The last part is a sequential number that identifies vulnerabilities listed during a year. The CVE dictionary does provide reference numbers to several other vulnerability databases, but no associations between vulnerabilities and products or between exploits and vulnerabilities. Thus, even if CVE is recognized as a standard dictionary in terms of vulnerability, it does not provide relationships between information within our network model.

It is important to note that all these information sources are not formally databases, but in order to keep their explanation simple, as most of them could be standardize into database systems, we will refer to them as *databases*.

4.5 Information Sources used with the Network Model

As explained in the previous section, these databases can be used for two purposes in PNMT: to populate static information in the network model and to specify



(a) Distribution for Nessus, Bugtraq and CVE

(b) Distribution for Nessus and Bugtraq

Figure 2. Snort Rules Reference Distribution

enhanced intrusion detection rules with network context to reduce false positives generated by IDS.

The static network information for the static classes can be provided by using exploit descriptions and vulnerabilities available in the Snort [5], Nessus [3], CVE [2] and Bugtraq [1] databases. The exploits described in Snort as IDS signatures can be linked to vulnerabilities in the Nessus, CVE and Bugtraq databases. The *Vulnerability* objects can be gathered from the Nessus, CVE and Bugtraq databases. The relationships between vulnerabilities, products and vendors can be gathered from the Bugtraq database. The relationships between the different types of products and their relationships with vendors can also be partially gathered from the Bugtraq database. Thus, if we are able to specify rules to capture the dynamic relationship in the network model between *IPStack* objects and *Product* objects, we will be able to define the potential vulnerabilities for each system in the network.

Intrusion detection rules that infer the relationships between *Port* objects, *IPStack* objects and *Exploit* objects to define an *Alarm* object could also be generated in part from those databases. The Snort intrusion rules provide the exploit characteristics and its references to vulnerabilities provide the network context. With these references, we are able to integrate in the intrusion rules which products are affected by a particular exploit. Thus, the products can be added in the intrusion signatures to capture the fact that this attack can only occur if the system is using this particular product. The relationships between *Exploit*, *Vulnerability*, *Product* and *Vendor* provide a more precise context when raising IDS alarms and thus may help reduce the number of false positives. This can be achieved using once again Snort, Nessus and Bugtraq databases.

5 Database Reference Analysis

To include more network context in intrusion rules and to populate the static part of the network model, the accuracy of the relationships between Snort exploit descriptions, Nessus and Bugtraq needs to be analyzed carefully. First, we will analyze the relationships between Snort rules and Nessus, Bugtraq and CVE, to verify how many rules in a given database actually refer to other databases and to which databases these rules are referencing most often.

5.1 Snort Reference Analysis

We used a custom parser to transfer the rule references so they could be more easily analyzed with a query language. From the statistical analysis presented in Figure 2(a) and Figure 2(b), we show that nearly half of the references (44%) refer to other databases than Nessus, Bugtraq and CVE or to no database at all. However, more than half of the references (53%) refer to Nessus or Bugtraq and can thus be used to infer intrusion rules with network context or to populate the static objects of the network context. It is also important to note that only 15% of the Snort rules actually refer to all three databases and that most of the rules (respectively 42% and 46%) have references to CVE or to Bugtraq.

5.2 Snort Relationships Analysis

As a result of this preliminary analysis, a more precise examination of Snort rules was warranted to verify the relationships of Snort rules to their Bugtraq references and to verify the relationships of Snort rules to their Bugtraq references inferred through Snort rules references to Nessus. In particular, Figure 3 represents the relationships between these databases. These relationships are important to create accurate intrusion detection rules with network context using OCL. This way, we are able to assess their effectiveness in reducing false positives when they are used with PNMT and IDS.

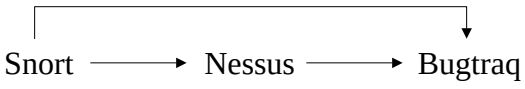


Figure 3. Snort, Nessus and Bugtraq Relationships Diagram

For this analysis we decided to drop the CVE references. These references are not useful to infer relationships between Snort, Nessus and Bugtraq because CVE entries do not provide more information that can be used in the network model than the information available in these databases. However, missing references between those databases can be inferred using the CVE number.

To analyze and identify the strengths of the relationships between Snort, Nessus and Bugtraq, we classified the Snort rules into four disjoint categories, as presented in Figure 4(a). The results of this analysis are presented in Figure 4.

Some categories had to be split into subcategories to identify more precisely the accuracy of the relationships. The subcategory descriptions are provided in Appendix A. Here is a description of these four categories:

1. Snort rules with direct and indirect relationships to Bugtraq: These Snort rules are the rules for which all relationships are defined in Figure 3. These rules have references to Nessus and Bugtraq and their Nessus references have references to Bugtraq. Thus, the direct relationship is the Snort to Bugtraq relationship and the indirect relationship is the Snort to Nessus to Bugtraq relationship. To validate the accuracy of the relationships leading to Bugtraq using Snort or using the Snort to Nessus to Bugtraq references, subcategories had to be defined. The results of this analysis are presented in Figure 4(b). This figure presents the distribution of the rules in this category, which are defined using the five subcategories described in Appendix A.

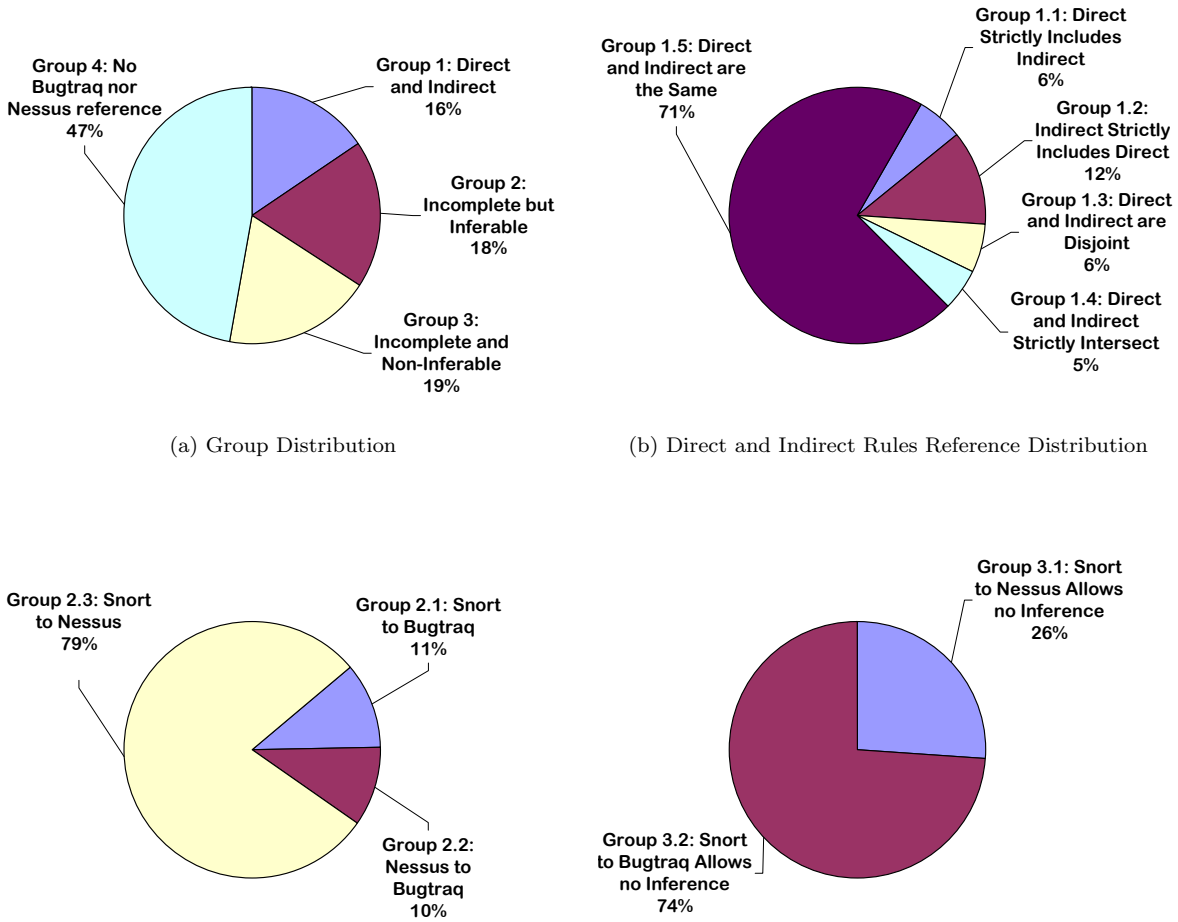
2. Snort rules with incomplete relationships that can be inferred: These Snort rules are the rules that are missing one of the relationships defined in Figure 3. Thus, the missing relationship has to be inferred using the two other relationships. The results for this analysis are presented in Figure 4(c). This figure presents the distribution of the rules in this category which are defined using the three subcategories described in Appendix A.

3. Snort rules with incomplete relationships that cannot be inferred: These Snort rules are missing two of the relationships defined in Figure 3. As a result, they have either a reference to Bugtraq or to Nessus, but the missing relationships cannot be inferred using the only relationship they have. The results of this analysis are presented in Figure 4(d). This figure presents the distribution of the rules in this category which are defined using the two subcategories described in Appendix A.

4. Snort rules without any relationship to Nessus and Bugtraq: These Snort rules are the rules that do not have any reference to Nessus or Bugtraq. In fact, they are missing the two relationships from Snort defined in Figure 3. Thus, these rules cannot be improved by adding more network context or be correlated with VDS.

It is important to note that in Figure 4(a), 47% of the Snort rules do not have any references to Nessus or Bugtraq. Thus, network context information cannot be added to those rules using information provided by Nessus and Bugtraq. Moreover, the alarms from these rules cannot be correlated with VDS events such as Nessus, nor with vulnerability information provided by Bugtraq. This implies that correlation systems cannot correlate these Snort alarms with Nessus events using the references provided by Snort and Nessus.

Figure 4(a) shows that only 16% of the Snort rules have references to Bugtraq and Nessus. Of this 16%, 71% have the same set of Bugtraq references whether we use the Snort to Bugtraq references or the Snort to Nessus to Bugtraq references. Thus, only about 11.4% of the Snort rules are suitable to be used for adding network context or in a correlation with Nessus and Bugtraq. It is important to note that 29% of the Snort rules in Figure 4(b) present discrepancies, depending on whether we use the direct or indirect relationship to



(a) Group Distribution

(b) Direct and Indirect Rules Reference Distribution

(c) Inferable Rules Reference Distribution

(d) Non-inferable Rules References Distribution

Figure 4. Snort Rule Group Reference Distribution

Bugtraq. In fact, 6% of them seem to refer to different Bugtraq vulnerabilities.

However, Figure 4(c) shows that 79% of the missing relationships between Snort and Nessus could probably be inferred using the Snort to Bugtraq and Nessus to Bugtraq relationships. It is difficult to assess whether those inferred relationships are accurate since 29% of the time there seems to be disagreements between the three databases. Nevertheless, from the sample we analyzed, the inferred relationships seemed to be coherent. A more in-depth analysis would be needed to support this conclusion. It should noted however that the inferred relationships from Nessus to Bugtraq would not be useful to us for adding network context to intrusion rules. We decided to raise this issue only to provide a complete analysis of the Snort rules distribution.

From Figure 4(d), we find that most of the missing relationships between Snort and Nessus cannot be inferred using the Snort to Bugtraq relationships. One could think that CVE references could be used to infer the missing relationships between Snort rules to Nessus and to Bugtraq entries. Indeed, CVE could be used to infer relationships between Snort and Nessus and between Snort and Bugtraq. However, as described in Figure 2(a), only 24% of the Snort rules have CVE references that have either a Nessus or Bugtraq reference. Thus, at most 24% of the 19% (i.e., 4%) of the rule references could be inferred using the CVE references.

The same reasoning applies for the Snort rules without Nessus and Bugtraq references. There are only 3%

of the Snort rules without references to Nessus and to Bugtraq that have a reference to CVE. Thus, at most 3% of the 47% (i.e., 1.4%) of the rule reference in this situation could be inferred using the CVE references. Thus, to add network context to intrusion detection rules and to perform event correlation using Snort, Nessus and Bugtraq, the CVE references do not provide much more relationships between these three databases.

In the next section, we use some of the inferred relationships between Snort, Nessus and Bugtraq to write context-based vulnerability and intrusion rules in OCL that are included in PNMT.

6 Vulnerability and Intrusion Rules

In this section, we describe some of the context-based vulnerability and intrusion rules that have been used with PNMT.

6.1 Context-Based Vulnerability Detection

As explained in the previous section, potential vulnerabilities can be identified using dynamic information inferred from information gathering rules specified in OCL. We developed a parser that automatically populates the static information classes of the network model presented in Figure 1. In particular, the *Vulnerability*, *Product*, *Vendor* and *Reference* classes are populated based on the vulnerability information provided in the Bugtraq database with the vulnerability relevant to intrusion detection rules of Snort. Using this information, dynamic network context information can be associated with products to identify potential vulnerabilities. Figure 5 describes a simplified version of an OCL dynamic information gathering rule to express how to identify that a server is using Microsoft IIS FTP server. By using the information inferred from this rule in combination with static information acquired from Bugtraq, it is possible to identify that this computer is opened to several vulnerabilities. The OCL rule specifies that a server is using Microsoft IIS FTP server if we see in the payload of a packet $p1$ the string "Microsoft FTP Service" is contained in an open session. Thus, this OCL dynamic information gathering rule enables PNMT to discover a subset of the relationships define in Figure 1 between *IPStack* and *Product* objects.

6.2 Context-Based Intrusion Detection

We also use the information provided by Snort, Nessus and Bugtraq to populate the *Exploit* class as well as establishing its relationship with the *Vulnerability* class. We have successfully tested several intrusion detection rules with network context in comparison with the original Snort version. In particular, Figure 6 describes the Snort rule 1408 that is used to identify an attack on the Microsoft Distributed Transaction Service.

```
Context Packet

inv: allIntances()->forAll( p1 |
     p1.data.contains("Microsoft FTP Service.") and
     sessionOpen(p1.ip.destinationAddress,
     p1.ip.sourceAddress, p1.tcp.destinationPort,
     p1.tcp.sourcePort)
     implies
     IPStack:allInstances()->exists(ipAddress =
     p1.ip.sourceAddress and
     ports->exists(number = p1.tcp.sourcePort and
     state = Port.OPEN and
     type = Port.TCP and
     deamon.prodname = "ISS" and
     deamon.type = Product.FTP and
     deamon.vendor.name = "Microsoft"))
```

Figure 5. Microsoft FTP Server Detection Rule

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 3372
(msg: "DOS MSDTC attempt"; flow:to_server,established;
dszie:>1023; reference:bugtraq,4006; reference:cve,2002-0224;
reference:nessus,10939; classtype:attempted-dos;
sid:1408; rev:10;)
```

Figure 6. Snort "DOS MSDTC attempt" Rule

This attack creates a buffer overflow that may trigger a denial of service for this service. The rule states that if we have an open TCP session from the client to the service on port 3372 and that the size of one of the packets sent to the server is greater than 1023 bytes we have a "DOS MSDTC attempt" (a denial of service attempt on the Microsoft distributed transaction service). As described in this rule, it has a reference to Nessus script number 10939 and Bugtraq vulnerability 4006. Using these references, we are able to add more network context to this rule and reduce the number of false positives. From the Bugtraq vulnerability database we are able to find that only certain version of Windows such as Windows NT and Windows 2000 are vulnerable.

A simplified version of this Snort rule with network context written in OCL is provided in Figure 7. This OCL rule describes that an alarm is added to the network model if the payload of a packet $p1$ is bigger than

```
Context: Packet

inv: allInstances()->forAll(p1 |
     p1.data.size() > 1023 and
     p1.tcp.destionationPort = 3372 and
     sessionOpen(p1.ip.destinationAddress,
     p1.ip.sourceAddress, p1.tcp.destinationPort,
     p1.tcp.sourcePort) and
     ((IPStack:hasOS(p1.ip.destinationAddress,
     "Windows", "NT") or
     IPStack:hasOS(p1.ip.destinationAddress,
     "Windows", "2000"))
     implies
     Alarm:logAttack(p1.ip.sourceAddress,
     p1.ip.destinationAddress,
     p1.tcp.sourcePort, p1.tcp.destinationPort,
     p1.time, "DOS MSDTC attempt")
```

Figure 7. PNMT OCL "DOS MSDTC attempt" Rule

1023 bytes, if there is an active session between the source IP address and the destination IP address on port 3372 of the packet $p1$ and if the operating system of the destination IP address of $p1$ is either Windows 2000 or Windows NT. The difference between this rule and the one specified in Snort is that this rule will only raise an alarm if it is able to confirm that the destination of the attack has a Windows 2000 or Windows NT operating system.

We have tested this vulnerability in two different scenarios. First, we have tested this scenario on a vulnerable Windows 2000 operating system and then we used the same exploit on a non-vulnerable Windows XP operating system. In both cases, Snort raised an alarm. However, the attack may only have been successful when this exploit was used against a vulnerable Windows 2000 operating system. On the other hand, PNMT using this OCL rule, it only raises an alarm in the first case.

The OCL rule could be further improved to confirm this alarm by verifying if another computer after this event is able to communicate on this port. If not, it means that the denial of service was successful. In fact, all denial of service intrusion signatures could be completed using this information.

7 Future Work and Conclusion

From the survey presented in this article, we know that the information to link the Snort, Nessus and Bugtraq databases together is available. Despite the efforts invested to develop event correlation techniques and tools, it appears that the way the information from the databases is formatted and reference each other does not allow efficient correlation of security events between IDS and VDS. As a result, we believe that considerable improvements in terms of information formatting and references have to be made to be able to automatically derive intrusion detection rules with network context from these databases or to use them with event correlation systems.

To complete the dynamic rule set and the static information provided by Snort, Nessus and Bugtraq databases, we derived the missing dynamic information needed to passively gather the network context. As a result of our work, a library of approximately 25 network information inference rules is under development. These rules allow PNMT to gather information on different aspects of network components. These rules are classified into Host Discovery, Operating System Identification, Protocol Discovery, Service Discovery, Switch and Router Discovery, and IP Host Configuration. In the case of Operating System Identification, we have so far implemented 2 of the 12 rules proposed in [8]. The report lists 12 rule types for operating system fingerprinting, capturing over 200 different operating systems.

At the beginning of this project, we were optimistic with regard to the additional relationship information we could obtain from intrusion rules, vulnerabilities and products. Our optimism was justified by the claims made by the vendors of different correlation tools that they could correlate events generated by security systems such as Snort and Nessus. Furthermore, we investigated the possibility of developing a tool that could automatically generate all the possible intrusion rules with their appropriate network context, based on the information provided by Snort, Nessus and Bugtraq. However, our initial approach did not yield the expected results. Based on our analysis, and since the Nessus scripts are not as well formatted as the Snort rules and Bugtraq vulnerability entries, we had to consider other options. For now, we decided to manually derive a subset of intrusion rules with network context from these databases before designing an automated tool. This allows us to ensure that the rules derived from these databases are appropriate to prove that adding more network context to intrusion detection rules will reduce the number of false positives found by IDS.

The accuracy of the references between Snort and Nessus on one hand, and Snort and Bugtraq on the other hand, seems to have increased since the introduction of references in Snort rules (first introduced in Snort 1.8). As a matter of fact, their accuracy has con-

siderably improved since Snort 2.0. Moreover, for the past few years, companies have been maintaining these rules in collaboration with the open source community.

The development of PNMT is still in progress, but plans are being made to test it on a large network when the development is completed. We also believe that network context monitoring constitutes a key element that could be integrated in commercial and open source IDS to reduce the number of false positives. Furthermore, PNMT provides a formal approach to incorporate network context into network monitoring systems.

References

[1] Bugtraq vulnerability database. `http://www.securityfocus.org/bid`.

[2] CVE dictionary. `http://www.cve.mitre.org/`.

[3] Nessus scripts. `http://www.nessus.org/plugins/`.

[4] NeVO homepage. `http://www.tenablesecurity.com/nevo.html`.

[5] Snort rules. `http://cvs.sourceforge.net/viewcvs.py/snort/snort/doc/signatures/`.

[6] H. Anderson. Introduction to nessus. `http://www.securityfocus.com/infocus/1741`, october 2003.

[7] J. Beale and J. C. Foster. *Snort 2.0 Intrusion Detection*. Syngress Publishing, 2003.

[8] A. DeMontigny-Leboeuf. A multi-packet signature approach to passive operating system detection. CRC/DRDC joint Technical Report CRC-TN-2005-001 / DRDC-Ottawa-TM-2005-018, December 2004.

[9] A. DeMontigny-Leboeuf and F. Massicotte. Passive network discovery for real time situation awareness. In *NATO/RTO Adaptive Defence in Unclassified Networks*, Toulouse, France, April 2004.

[10] R. Deraison, R. Gula, and T. Hayton. Passive vulnerability scanning- an introduction to nevo. `http://www.tenablesecurity.com/papers.html`, august 2003.

[11] R. Goldman, W. Heimerdinger, S. Harp, C. Geib, V. Thomas, and R. Carter. Information modeling for intrusion report aggregation. In *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX II)*, pages 329–342. DARPA, 2001.

[12] C. Green and M. Roesch. Snort users manual 2.1.0 - the snort project, Dec. 2003.

[13] R. Gula. Correlating ids alerts with vulnerability information. `http://www.tenablesecurity.com/papers.html`, december 2002.

[14] F. Massicotte. Using object-oriented modeling for specifying and designing a network-context sensitive intrusion detection system. Master's thesis, Department of Systems and Computer Engineering, Carleton University, September 2005.

[15] B. Morin, L. Mé, H. Debar, and M. Ducassé. M2d2 : A formal data model for ids alert correlation. In *5th International Conference on Recent Advances in Intrusion Detection (RAID 2002)*, volume 2516 of *LNCS*, pages 177–198, Zurich, October 2002. Springer.

[16] OMG. UML 2.0 OCL specification. `http://www.omg.org/docs/ptc/03-10-14.pdf`, 2003. Draft Adopted Specification ptc/03-08-08.

[17] OMG. UML 2.0 superstructure specification, 2003. Final Adopted Specification ptc/03-08-02.

[18] F. Valeur, G. Vigna, C.Kruegel, and R. Kemmerer. A Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, July-September 2004.

[19] G. Vigna. A topological characterization of tcp/ip security. Technical Report TR-96.156, Politecnico di Milano, 1996.

[20] G. Vigna and R. A. Kemmerer. Netstat: A network-based intrusion detection approach. In *ACSAC*, pages 25–, 1998.

A Subcategory Descriptions

1. Snort rules with direct and indirect relationships to Bugtraq.

   1.1 The direct references to Bugtraq that strictly include the indirect references to Bugtraq: These are the Snort rules where their direct Bugtraq references strictly contain all their Snort to Nessus to Bugtraq references.

   1.2 The indirect references to Bugtraq that strictly include the direct references to Bugtraq: These are the Snort rules where their Snort to Nessus to Bugtraq references strictly contain all their direct Bugtraq references.

   1.3 The direct and indirect references to Bugtraq that are disjoint: These are the Snort rules where their direct Bugtraq references are totally different from their indirect relationships to Bugtraq.

   1.4 The direct and indirect references to Bugtraq that strictly intersect: These are the Snort rules where their direct Bugtraq references have some references that are the same as when using their indirect Snort to Nessus to Bugtraq references.

   1.5 The direct and indirect references to Bugtraq that are the same: These are the Snort rules where their direct Bugtraq references are equal to their indirect Snort to Nessus to Bugtraq references.

2. Snort rules with incomplete relationships that can be inferred.

   2.1 The Snort to Bugtraq relationships that can be inferred: These are the Snort rules that do

not have Bugtraq references, but that have Nessus references that can in turn have references to Bugtraq. Thus, their Bugtraq references could probably be inferred using information provided by the Snort to Nessus and the Nessus to Bugtraq known references.

2.2 The Nessus to Bugtraq relationships that can be inferred: These are the Snort rules that have Nessus and Bugtraq references, but where their Nessus references do not have any references to Bugtraq. Thus, the Nessus to Bugtraq references could probably be inferred using information provided by Snort to Nessus and Snort to Bugtraq known references.

2.3 The Snort to Nessus relationships that can be inferred: These are the Snort rules that have Bugtraq references, that do not have any Nessus references, but where there are Nessus scripts that refer to the same Bugtraq references. Thus, their Nessus references could probably be inferred using information provided by Snort to Bugtraq and Nessus to Bugtraq known references.

3. Snort rules with incomplete relationships that cannot be inferred.

3.1 The Snort to Bugtraq relationships that cannot be inferred: These are the Snort rules that do not have Bugtraq references, that have Nessus references, but where their Nessus references do not have any references to Bugtraq. Thus, their Bugtraq references cannot be inferred because there is no relationship from Snort to Bugtraq and from Nessus to Bugtraq.

3.2 The Snort to Nessus relationships that cannot be inferred: These are the Snort rules that do not have Nessus references, that have Bugtraq references, but where there are no Nessus scripts that have the same references to Bugtraq. Thus, their Nessus references cannot be inferred because there is no relationship from Snort to Nessus and from Nessus to Bugtraq.