A Hybrid Approach to Operating System Discovery Based on Diagnosis Theory

François Gagnon
School of Computer Science
Carleton University
Ottawa, Canada
fgagnon@sce.carleton.ca

Babak Esfandiari
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
babak@sce.carleton.ca

***Abstract*—Motivated by the increasing importance of knowing which operating systems are running in a given network, we evaluated operating system discovery (OSD) tools. The results indicated a serious lack of accuracy in current OSD tools.**

This thesis proposes a new approach to OS discovery which addresses the limitations of existing tools and leads to a more flexible, less intrusive, and much more accurate tool. Moreover, unlike existing OSD tools which are completely ad hoc, our approach is formal and follows the principles of diagnosis problem solving. This formalism allows us to:

- **characterize the complexity of OSD;**
- **use well-tested algorithms and**
- **benefit from numerous possible extensions.**

To fully address the needs of OSD, we generalize the theory of diagnosis with a query-based extension. This extension leads to a spectrum of test selection algorithms to solve each query.

I. Introduction

It has been recognized, by the research community [6], [9], [11], [16] as well as the industry [15], that contextual information surrounding a network attack, or more precisely an alarm from an intrusion detection system (IDS), is highly relevant for determining whether this attack is likely to succeed or not. Classifying IDS alarms is an important task for distinguishing the events that have to be handled immediately by the security team from those that can be postponed for a while.

One of the key piece of information needed to determine whether an attack will succeed or fail is the targeted host configuration, including the version of the operating system (OS) it is running and the version of the available services. In this thesis, we focussed on the impact of knowing which OS is running on a targeted host for the task of determining the likelihood of success of an attack directed towards that host.

This paper presents a brief summary of this thesis[1] and follows the chronological steps of the research process. We started with a theoretical analysis to determine the usefulness of OS information for classifying IDS alarms. From the results, showing that this information could allow us to filter out around 40% of the false positives alarms, we concluded that this was a promising direction. The second step was to evaluate current operating discovery (OSD) tools for this specific task. The results were less convincing here: the best tool was barely able to identify 13% of the false positives (approximately 1/3 of the potential for the approach). After an investigation to discover why current tools were not up for the task, we developed a new approach to OSD and implemented this approach. We then compared our tool with other OSD tools to measure the benefits of our new approach. The result were very impressive: our tool was able to identify 35% of the false alarms, nearly reaching the full potential of this piece of contextual information. Moreover, unlike existing OSD tools which are completely ad hoc, our approach follows a formal path through the use of the theory of diagnosis. The problem of OS discovery was so closely related to the general diagnosis framework that we've been able to use our application to propose a general extension to the diagnosis theory; namely, a query-based approach.

[1]full version available at: www.sce.carleton.ca/~fgagnon/PhDThesis.pdf

II. Motivation

Information about which operating system is running on a given computer is interesting in several situations. We've identified three different *questions*[2] regarding OS information that are useful in different contexts:

- Exact OS Query: ”Which OS is running on the computer?” This is the classical problem and is useful for network inventory and similar tasks.
- Group OS Query: ”Is the computer running an OS belonging to a given set of OSes *O*?” This is useful when trying to classify an IDS alarm, the set *O* is then the set of OSes that are vulnerable to the underlying attack.
- Single OS Query: ”Is the computer running the specific OS *o*?” This query is interesting for the task of finding out which computers need to be updated with a recently released patch.

Of course, from a knowledge point of view, those queries are redundant since the exact OS query allows us to answer the other queries as well. However, they prove to be quite interesting as they correspond to different computational problems,

[2]We call them queries to emphasize the fact that our approach relies on a knowledge base.

see Section III-B. Hence, if one is only interested in answering the single OS query, using the full exact OS query could mean longer computation or a more expensive solution.

A. *OSD for IDS Alarm Classification*

One of the main applications of OS discovery nowadays is the group OS query, used to determine if an IDS alarm is a false positive or not. To measure how good this approach actually is, we conducted an experiment to measure how many false positives we can identify assuming we know the exact OS version of the target. To conduct such a experiment, we used a publicly available dataset of attack scenarios produced at the Communication Research Center in Canada in 2006. The CRC dataset [10], [12] contains 6,656 traces. Each trace corresponds to one attack scenario, i.e., launching one of 92 attacks (covering several different vulnerabilities) against one of 95 targets (covering several different OS families). Each trace comes with the following information:

- the Security Focus Bugtraq ID [14] of the attack used;
- the outcome of the attack (success or failure);
- the configuration of the target;

From there, we can simulate a decision process in which we classify an alarm as a false positive if the target OS (taken from the target configuration) is not part of the systems vulnerable to this attack (based on the Security Focus vulnerability database). This corresponds to the best case scenario where we know the exact target OS, and it will give us an estimation about the relevance of using OS discovery to filter out false alarms from IDSes.

The results are quite interesting: 40.7% of the false alarms were identified as such (recall measure) simply by considering the target OS and only 0.7% of the real alarms were misclassified as false alarms (precision measure). After further investigation, it turned out that all the precision mistakes were caused by incomplete information in the Security Focus database (i.e., some vulnerable systems were not listed as such). Full description and results of this experiments can be found in [4], [5].

Those results indicate that using OS information is a promising way to classify IDS alarms. However, relying on the exact target configuration is unrealistic. Networks are too large and systems are too dynamic to manually maintain an up-to-date database of all the target configuration in a network. Instead, we need to rely on OS discovery tools to gather this information when needed. So the next step was to evaluate how effective current OSD tools are at this specific task.

B. *Existing OSD Tools*

Current OSD tools are all designed to answer the exact OS query.We can use that information to answer the group OS query needed for IDS alarm classification. For this experiment, we used nine OSD tools: Xprobe, Ettercap, Nmap, Siphon, SinFP and P0f in four different modes[3] (SynAck, RstAck, Syn, StrayAck). These tools get a recall percentage of 12.7, 8.5, 5.1, 2.4, 1.6, 8.7, 3.4, 2.2, and 0.6 respectively. The best tool, Xprobe, achieves 12.7% which is merely 1/3 of the potential (40.7%) of this approach. The precision is quite good for these tools. See [4], [5] for fully detailed results.

The question is: why are current OSD tools doing so poorly at that task? The answer is manifold. First, it is important to see that current OSD tools fall in two categories: passive and active.

Passive OSD tools rely on information that is naturally available on the network without any intervention on their part[4]. These tools must often deal with a very limited set of events. Moreover, existing passive tools are stateless; that is, they do not remember previous information when analyzing the current event (they can think a system might be running Linux even if a previous event should have discarded that possibility). Finally, existing passive tools are single packet-based; that is, they do not correlate information from multiple packets (which is interesting in many stimulus-response situations). All these features greatly limit their accuracy.

Active tools, on the other hand, send probes to initiate a response from the target. Thus they have access to the event they want, but not all events can be forced that way[5]. Moreover, a drawback of active tools is the generation of (sometime irregular [6]) network trafficto fingerprint the target. The tradeoff between obtaining good information and not injecting too much traffic on the network limits the number of available tests, thus the accuracy of active tools.

Every tool focuses on a specific set of events. For instance, Nmap and Xprobe, two active tools, do not consider the same types of probes at all. Obviously, the more events you consider, the more accurate you can be.

By addressing those limitations, we believed it was possible to improve on the current state of the art.

III. Hybrid OS Discovery

Our idea to improve state of the art OS discovery is to rely on a hybrid approach, much like diagnosis engines. As soon as the process is started, it gathers freely available information (passively) and starts making deductions about the possible explanations for the observed behavior (in our case the OS responsible for the observed traffic). When the user needs to know the final diagnosis, the engine can switch to active mode to fetch the possibly missing information. The active module relies extensively on the information gathered passively (only performing tests that will provide new information).

Since the diagnosis literature is mainly focused on finding the actual diagnosis, we've extended the classical diagnosis engine to include a query-based approach, see [3]. In this

[3]Since these modes are completely independent, we consider them as different tools.

[4]For instance, whenever a computer performs an ARP request, it has to fill the destination MAC field of the ARP packet without knowing the actual value. Some OSes will fill this field with 00:00:00:00:00:00, while other will fill it with FF:FF:FF:FF:FF:FF, and yet some others will use a seemingly random value

[5]Information gleaned from the DHCP protocol is an example.

[6]Nmap will send a TCP packet with no flags to see how the target reacts in that situation

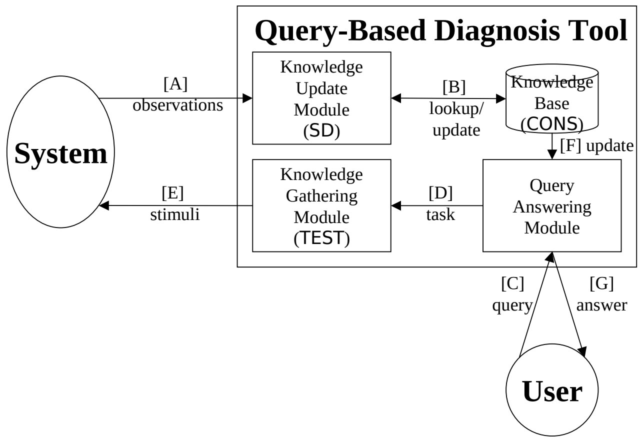

Fig. 1. Query-Based Diagnosis Tool

extension, the active module is guided by both the passively gathered information and the user query (defining the goal to reach). The query-based extension is interesting, as one query might be more expensive[7] to solve than another one in the same context.

A. Passive Module

The resulting diagnosis engine, see Figure 1, is built around a knowledge base in which we store possible explanations based on the information acquired so far. This will act as a memory to reuse previous deductions. Moreover, our implementation of the passive module gathers several network packets before analyzing them. This allows us to consider multi-packet phenomena (e.g., Syn-SynAck). Once a sufficient number[8] of packets have been gathered, they are analyzed and discarded; only the resulting deductions are kept[9]. The passive module algorithm is based on Reiter's Candidate Generation algorithm [13] (using conflict sets and hitting sets) and runs in polynomial time for single fault[10] diagnosis.

B. Active Module

We have implemented three queries in the active module (single candidate, group candidate, and exact candidate queries), each corresponding to one of the OSD queries mentioned earlier, see Section II. We've shown that these queries are meaningful in other diagnosis domains (medical or engineering diagnosis). We have also shown that these queries are useful: for instance, solving the single candidate query is never harder than solving the exact candidate query, but solving the exact candidate query can sometimes be much harder than solving the single candidate query, see [3].

[7]Either from a computational standpoint; to build a plan, or from a cost-based point of view; to execute the plan.

[8]The current threshold is set to 100; a tradeoff between execution time for larger sets of packets and the risk of separating a stimulus from its response for smaller sets.

[9]Keeping every packet would quickly slow down the engine.

[10]Assuming each IP address is associated with a single OS.

The active module relies on test selection strategies to propose which tests to execute in order to answer a given query. A direct implication is that we can encode as much tests as we want into our tool; in fact, the more tests we have, the better. This is an improvement over existing OSD tools which limit the number of available tests, since they always execute all of them.

For the single candidate query, we established that a greedy test selection, although computationally fast, provides a theoretically unbounded suboptimal solution (w.r.t., the number of tests executed). It is possible to obtain an optimal solution, but the single candidate query problem is NP-Hard (by a reduction to the Set Cover problem). However, an experiment we conducted, see [1], shows that in practice the greedy algorithm provide a very good approximation for the single candidate query in the OSD domain.

For the group candidate query and the exact candidate query, the task is more difficult. It is not easy to define a universal comparison metrics between two solutions. Hence, the definition of optimal solution is ambiguous. We believe this is a argument in favor of the usefulness of our query-based approach: the fact that the single candidate query is easily characterizable while the other two queries are not is a good indication that the underlying problems are quite different.

IV. Results

Using an implementation of our hybrid approach to OS discovery, we ran some experiments to compare with existing OSD tools. The comparison was made under two different angles. First, we ran the experiment described earlier to compare the tools in the context IDS alarm classification; that is, how good are they with the group OS query. We also compared the tools under the exact OS query. The second angle is interesting because it corresponds to the primary objective of all existing OSD tools. Hence they won't have a design disadvantage. Moreover, it will allow us to quantify the cost difference between directly answering the "easier" group OS query vs answering the full exact OS query in terms of the number of tests required, i.e., the number of packets injected.

When presenting the results, we consider three new OSD tools: posd, aosd, and hosd. posd is the passive only module of our hybrid approach discussed earlier, aosd is the active only module, while hosd is the full hybrid tool. Studying posd allows us to determine whether using a stateful approach to passive OSD has benefits. aosd helps us understand the advantage of using a test selection strategy for active OSD (instead of just running all tests). Finally, hosd will measure the impact of combining the passive and active strategies.

A. Group OS Query

To evaluate the group OS query for our tools, we used the same experiment as in section II-A. Let us recall that the potential of OS information was to filter out 40% of false alarms while the best current OSD tools achieved only 13%. Figure 2 shows the recall results.

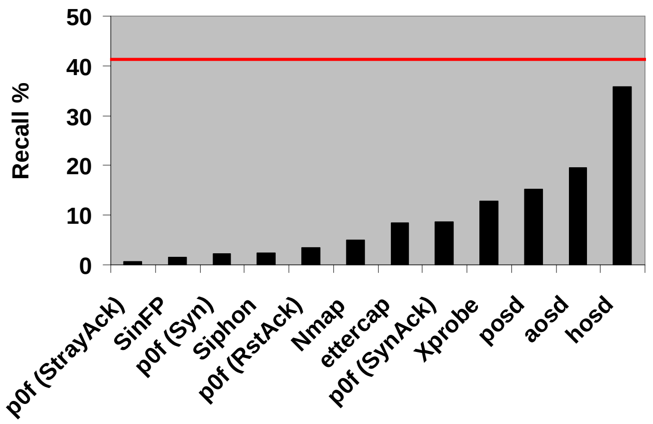

Fig. 2. OSD Tools Recall

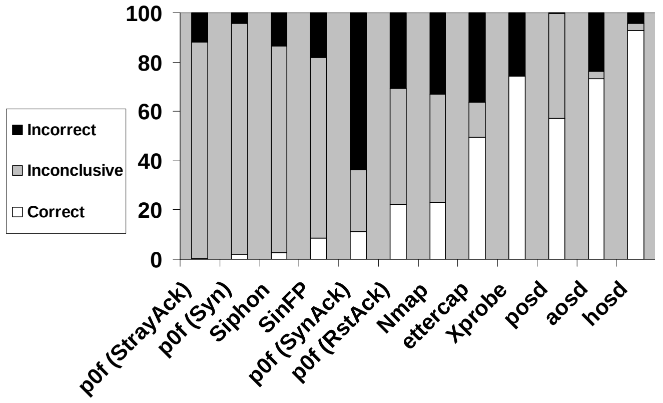

Fig. 3. Correctness for the Exact OS Query

We can see that our passive tool (posd) outperforms all other passive tools. This confirms our intuition that a knowledge-based approach is a key component here (allowing a stateful approach with multi-packet analysis). Our active tool is also more effective than all other active tools. This is mainly due to the fact that most active tools have a very limited set of tests (to avoid injecting too much traffic, since they always perform all their tests). aosd, on the other hand, relies on a test selection strategy to perform a minimal subset of tests.

But the most interesting conclusion is definitely the fact that combining the passive and active approach yields a significant improvement. The information gathered by the passive and active module is complementary.

B. Exact OS Query

The objective of the exact OS query is to identify the actual operating system of a computer. This is not a yes/no question, thus it is harder to evaluate. OSD tools will rarely identify a single possible OS; they provide a set of possible OSes (because different OSes sometimes behave in a similar way). We used the CRC dataset again, but this time looking for the OSD tools to identify the actual OS.

First, we established a *correctness* measure. For each test case (i.e., traffic trace) a tool either provides a *correct* answer (if the actual OS is among the set of possible OSes identified by the tool), an *incorrect* answer (if the actual OS is not among the set of possible OSes), or an *inconclusive* one (if the tool is unable to extract any information from the traffic). The intuition is that it is preferable to obtain an inconclusive answer than an incorrect one.

A limitation of the correctness measure is that it is easy for a tool to perform arbitrarily well by generating very large set of possible OSes. To circumvent this, we use a second measure, *imprecision*. Imprecision is the size of the possible OSes set given by a tool whenever it provides a correct answer. The lower the imprecision, the better. Below we compare the results of existing OSD tools with our own tool for the exact OS query based on the correctness and imprecision measures.

1) Correctness: Figure 3 presents the correctness for each tool. Below is our interpretation of these results:

- posd is again better than every other passive tool. This is another argument in favor of our knowledge-based approach to OSD.
- The combination of our passive and active modules provides excellent results and again we see how well posd and aosd complement each other.
- posd and hosd rarely provide the wrong answer. This is an effect of a safe strategy intrinsic to our hybrid approach: we do not need to guess aggressively; in case of doubt, we simply fetch additional information.

2) Imprecision: Figure **??** provides the average imprecision for each tool. The lower the imprecision, the better. Here are some interesting conclusions extracted from these results:

- hosd and aosd perform very well. The tools having better imprecision all have a highly inadequate correctness. hosd is arguably the best tradeoff between imprecision and correctness.
- posd does not have a very good imprecision by itself. Again, this confirms the importance of combining the passive and active approaches together.

C. Traffic Generated

Another interesting measure of evaluation for active OSD tools is the amount of traffic they inject on the network. Table I shows the number of packets injected by each active tool for the two experiments. We observe the following:

- Nmap injects a high volume of packets. This is mainly due to the port scan performed before starting the OS discovery module[11]. It is possible to disable this port scan, in which case Nmap still injects 30 packets on average, but the drawback is an important decrease in accuracy. It is fair to mention that Nmap was not designed specifically to be an OS discovery tool; however, it is worth noting that Nmap is currently one of the most, if not the most, popular tool for OS discovery.

[11]Note that using a hybrid approach, the information gleaned by a port scan can be constructed by the passive module

TABLE I
PACKET INJECTION SUMMARY FOR GROUP OS QUERY

	Nb packets sent					
	Group OS Query			Exact OS Query		
Tool	*Min*	*Mean*	*Max*	*Min*	*Mean*	*Max*
Nmap	882	1686	2186	882	1686	2186
Xprobe	7	7	7	7	7	7
aosd	1	7.9	16	3	13.3	21
hosd	0	2.1	8.4	0	3.9	13

- There is no difference for Nmap and Xprobe between the group OS query and the exact OS query. This was expected, since classical active OSD tools are not query-dependant. That is, they will send all their tests regardless of what the user wants to know.
- On the other hand, there is a difference for our tools, aosd and hosd, between the two queries. The exact OS query requires consistently more tests to be executed, both on average and in the worst case, than the group OS query. This confirms our intuition that a query-based approach is valuable for lowering the overall cost of active testing.
- There is a noticeable improvement between aosd and hosd. It is due to the combination of the passive and active modules (hosd needs to send fewer packets as it already possesses a lot of information from passive monitoring). In the best case, when the information gathered passively is sufficient to answer the query, hosd does not need to perform any test at all.

V. CONCLUSION AND FUTURE WORK

Based on our results, the following key points are worth mentioning.

Target configuration information is extremely relevant for the context of an attack. It can filter out a significant amount of non-critical alarms (40% when considering only the target OS). Current OSD tools, however, are not adequate for the task of IDS context gathering, as they achieve only 1/3 of their potential. This is a consequence of intrinsic limitations of the current approaches.

We were able to lift some of those limitations. A knowledge-oriented approach to OS discovery greatly improves the accuracy, starting with passive OSD. Combining the active and passive approaches into a hybrid one also increases the accuracy (by maximizing the number of phenomena that can be observed) while reducing the number of executed tests (by relying on test selection and passive information).

In our work, we modeled OSD as a diagnosis problem. This provided an intuitive framework for reasoning about the problem and it supplied algorithms for the different modules. We've shown that extending diagnosis theory with a query-based approach generally reduces the cost of extracting information, as some queries are "easier" to solve than others.

A. Future Work

Much interesting work remains to be done in the area of OS discovery.

Considering probabilities in the reasoning process is definitely the most important one. All OSes are not equally popular and this should be used to guide the decision process. Moreover, the reliability of OSD tests is not constant and this could be modeled through probabilities.

Another interesting aspect would be to switch to a multiple-fault diagnosis system [7] in order to capture network topologies such as network address translators (NAT). Every computer behind a NAT has its own behavior and a NAT may behave as if it was running several different OSes.

Distributing the diagnosis reasoning process [8] would help to deploy such tools in large network where we cannot gather all the relevant information from a single point.

Finally, performing an experiment to compare different ways (target configuration, vulnerability assessment, attack side-effects) of filtering IDS false alarms would give us a better idea about the big picture.

VI. PUBLICATIONS

The thesis results led to publications in different areas. Security and network management (with the improvement to OS discovery), artificial intelligence and knowledge representation (with the extension to the diagnosis theory), and vitualization (with the work on the VNEC platform, see Section VII-B). Following is a list of our publications.

A. Journals

- (2011) International Journal of Network Management. Gagnon F. and Esfandiari B. A hybrid approach to operating system discovery based on diagnosis, 21(2):106-119.
- (2009) Artificial Intelligence Review. Gagnon F. and Esfandiari B. A Query-Based Approach for Test Selection in Diagnosis, 29(3):249-263.

B. Refereed conferences and Workshops

- (2010) Gagnon F., Esfandiari B., and Dej T. - Network in a Box - Proceedings of the 2010 International Conference on Data Communication Networking (DCNET'10).
- (2009) Gagnon F. and Esfandiari B. - Using Answer Set Programming to Enhance Operating System Discovery - Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LP-NMR'09).
- (2009) Gagnon F., Esfandiari B. and Massicotte F. - Using Contextual Information for IDS Alarm Classification (Extended Abstract) - Proceedings of the 6th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'09), 147-156 (LNCS-5587).
- (2008) Gagnon F., Esfandiari B. and Dej T. VNEC: A Virtual Network Experiment Controller - Proceedings of the 2nd International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and New Technologies (SVM'08), 119-124.
- (2008) Gagnon F. and Esfandiari B. A Query-Based Approach for Test Selection in Diagnosis: Operating System Discovery as a Case Study - poster session of the

19th International Workshop on Principles of Diagnosis (DX'08).

- (2007) Gagnon F., Esfandiari B. and Bertossi L. A Hybrid Approach to Operating System Discovery using Answer Set Programming - Proceedings of the 10th IFIP/IEEE Symposium on Integrated Management (IM'07), 391-400.
- (2006) Massicotte F., Gagnon F., Couture M., Labiche Y., and Briand L. Automatic Evalaution of Intrusion Detection Systems - Proceedings of the 2006 Annual Computer Security Applications Conference (ACSAC'06).
- (2006) Massicotte F. and Gagnon F. A Publicly Available Data Set for the Evaluation of Signature-Based IDS - poster session of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06).

C. Misc

- (2010) Ph.D. Thesis, Carleton University. Gagnon F. A Hybrid Approach to Operating System Discovery Based on Diagnosis Theory.
www.sce.carleton.ca/~fgagnon/PhDThesis.pdf
- (2008) Book chapter in Emerging Artificial Intelligence Applications in Computer Engineering - Frontiers in AI and Applications Series, IOS Press. Gagnon F. and Esfandiari B. Using Artificial Intelligence for Intrusion Detection, pp. 295-306.

VII. Tools

Two open source tools were released as part of this research: hosd and VNEC.

A. HOSD

hosd is the implementation of our Hybrid approach to OS Discovery. It is available as an open source project through hosd.sourceforge.net. hosd is a Java tool with a prolog[12] reasoning engine.

B. VNEC

VNEC is a tool to facilitate the execution of repetitive network experiments using a virtual environment. VNEC stands for Virtual Network Experiment Controller. It emerged from our need to gather a large database of OS fingerprints. While collecting fingerprints on real computer is tedious and requires several machines, VNEC allows us to perform this task automatically using hundreds of virtual machines, see [2]. It is available as an open source project through vnec.sourceforge.net.

Although VNEC was developed with the objective of gathering OS fingerprints, it can be used for numerous network experiments such as the study of virus propagation patterns and software deployment testing.

[12]The first version of hosd came with an Answer Set Programming reasoning module.

References

[1] François Gagnon. *A hybrid Approach to Operating System Discovery Based on Diagnosis Theory*. Ph.d. thesis, Carleton University, 2010. www.sce.carleton.ca/~fgagnon/PhDThesis.pdf.

[2] François Gagnon, Tomas Dej, and Babak Esfandiari. Network in a Box. *(Submitted to) 2010 International Conference on Data Communication Networking (DCNET'10)*, 2010.

[3] François Gagnon and Babak Esfandiari. A query-based approach for test selection in diagnosis. *Artificial Intelligence Review*, 29(3):249–263, 2009.

[4] François Gagnon, Frédéric Massicotte, and Babak Esfandiari. On the Effectiveness of Target Configuration as Contextual Information for IDS Alarm Classification. Technical Report SCE-08-08, Department of Systems and Computer Engineering - Carleton University, 2008. http://www.sce.carleton.ca/~fgagnon/Publications/context.pdf.

[5] François Gagnon, Frédéric Massicotte, and Babak Esfandiari. Using Contextual Information for IDS Alarm Classification. *Proceedings of the 6th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'09)*, pages 147–156, 2009.

[6] Burak Dayioglu and Attila Ozgit. Use of Passive Network Mapping to Enhance Signature Quality of Misuse Network Intrusion Detection Systems. *Proceedings of the 16th International Symposium on Computer and Information Science (ISCIS'01)*, 2001.

[7] Johan de Kleer and Brian C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[8] Eric Fabre, Albert Benveniste, and Claude Jard. Distributed Diagnosis for Large Discrete Event Dynamic Systems. *Proceedings of the 15th IFAC World Congress on Automatic Control*, 2002.

[9] Christopher Kruegel and William Robertson. Alert Verification: Determining the Success of Intrusion Attempts. *Proceedings of the 1st Workshop on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA'04)*, 2004.

[10] Frédéric Massicotte and François Gagnon. A Publicly Available Data Set for the Evaluation of Signature-Based IDS. poster session of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06), 2006.

[11] Frédéric Massicotte, Mathieu Couture, Yvan Labiche, and Lionel Briand. Context-Based Intrusion Detection Using Snort, Nessus and Bugtraq Databases. *Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST'05)*, October 2005.

[12] Frédéric Massicotte, Annie De Montigny-Leboeuf, and Mathieu Couture. Using a VMware Network Infrastructure to Collect Traffic Traces for Intrusion Detection Evaluation. *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC'05)*, 2005.

[13] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[14] SecurityFocus. SecurityFocus Homepage. http://www.securityfocus.org/.

[15] Jerry Shenk and Dave Shackleford. Sourcefire Real-Time Network Awareness. SANS Analyst Program.

[16] Robin Sommer and Vern Paxson. Enhancing Byte-Level Network Intrusion Detection Signatures with Context. *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 262–271, 2003.