

A Rule-Based Temporal Alert Correlation System

Peyman Kabiri¹ and Ali A. Ghorbani²

(Corresponding author: Peyman Kabiri)

Department of Computer Engineering, Iran University of Science and Technology¹

Narmak, Tehran, Iran (Email: Peyman.Kabiri@iust.ac.ir)

Faculty of Computer Science, University of New Brunswick²

Fredericton, NB, E3B 5A3, Canada (Email: ghorbani@unb.ca)

(Received Nov. 10, 2005; revised and accepted Jan. 7 & Apr. 25, 2006)

Abstract

This paper reports a research work to address the problem of the large number of alerts generated by the detectors in an intrusion detection system. Some of these alerts are redundant and have to be aggregated; others may follow a certain attack pattern that should be correlated. Generally, this operation is referred to as alert correlation. A more detailed explanation of the alert correlation is presented in the paper. Paper proposes a rule-based approach to solve this problem. In the reported work, an inference engine is implemented to derive the correlation between the alerts using a scenario-based knowledge base and to aggregate redundant alerts. Experimental results based on sample alerts and scenarios are reported in this paper.

Keywords: alert aggregation, alert correlation, intrusion detection, temporal alert correlation

1 Introduction

Currently, overwhelming number of alerts that are generated in the Intrusion Detection System (IDS) by its detectors is a source of confusion rather than help for the network security officer. There are two tasks, which should be performed on the generated alerts. One is to reduce its redundancies, i.e. to aggregate the redundant alerts generated by the detectors. The other one is to analyze the relationships between the sequences of events that are occurring in the system and to extract the possible attack strategies/patterns. Since this latter part has to be performed based on the sequence and the occurrence time of the alerts, it is referred to as the temporal alert correlation.

In other words, temporal alert correlator intends to connect the dots between the alerts along the time line where they have been encountered. Goal of the temporal alert correlation is to reduce the number of alerts gener-

ated by the detection unit and to convert them into sensible high-level alerts (alert fusion, alert merging can be a prerequisite for the alert correlation). This correlation will lead to a reduction in the number of false positive alarms. It will also provide a kind of broad and overall view on the activities in the network. This information can be a valuable strategic information for any network administrator.

New alerts will provide information regarding the overall situation of the ongoing attack(s). This information may include the current state of the attack progress; list of correlated alerts that have led the system to the current situation and if the attack is not yet completed. Later on, based on the available scenarios, system can predict goal(s) and targets of the attack. This requires an adaptive/learning system. After a survey on current literature in this area, it was decided to proceed with the implementation of a rule-based inference engine to address both the alert correlation and alert aggregation problem areas.

Before going any further, it seems necessary to provide a definition for alert correlation. Kim et al. [4] refer to Aromoso definition of the alert correlation and say: "Literally, intrusion correlation is defined that it refers to the interpretation, combination, and analysis of information from all available source about target system activity for the purposes of intrusion detection and response".

A brief description of some reported works in this area are given in the following section.

2 Previous Works

Cuppens et al. [3] report a work that is part of MIRADOR project. They have designed a cooperative module for the intrusion detection system that is called CRIM. Cuppens et al. have defined two types of correlation of the events: explicit correlation and implicit correlation. As it is clear from their name, explicit correlation is a type of correlation where it is possible for the network ad-

ministrator to find some kind of correlation in the events. However, the implicit correlation of the events requires a kind of adaptive and machine learning based system to find the hidden correlation between the events.

In Cuppens et al. approach, scenarios can be coded into knowledge and stored in the expert system (rule-based system) knowledge-base. The rule-based inference engine or the expert system is responsible for tracking the progress of the network status within the available scenarios and producing the appropriate response. Extracting scenarios is a major task in this approach and requires an in depth knowledge of different attack methods.

In another paper, Cuppens et al. [2] focus on the intrusion scenarios. Their paper considers the intrusion scenario to consist of a number of correlated events that their sequence and collaboration will lead to the act of intrusion. Work provides detailed description of scenarios for several attacks including the Mitnick attack and describes them in detail. Providing sample rules, in [1] they emphasize on the rule-based presentation of the illegal file access scenario. They defined alert correlation as: “Action1 is correlated with Action2, if Action1 may enable the intruder to perform Action2.” They also provide a definition for fusion process (the same as aggregation process): “The process of merging the simple alerts generated by different IDS detecting for the same attack is called fusion process.”

Ning et al. [6, 7] report a work in rule-based alert correlation (and the graph theory) that uses prerequisite and consequence of the attacks to correlate them. Once the prerequisite of a certain attack is detected by the system, it would be possible to predict the consequence of the attack. In the same way, it might be able to merge them and consequently detect a higher-level correlated attack. Ning et al. [6, 7] use SQL commands for processing the rules such as this one:

```
“SELECT DISTINCT c.HyperAlertID, p.HyperAlertID
FROM
PrereqSet p, ExpandedConseqSet c
WHERE
p.EncodedPredicate D c.EncodedPredicate
AND c.end time < p.begin time (Ning et al. [6, 7])”
```

Valdes et al. [8] have reported a work in probabilistic alert correlation, where they calculate the similarity value between the alerts. Both Valdes et al. [8] and Valeur et al. [9] have followed the multi-phase analysis of the alert system scheme. This approach is a fusion using a graph-theoretic approach and it is in two phases. During the first phase, low-level alerts are aggregated and converted into threads. In the second phase using the fused alerts from the phase one, threads are correlated to provide a higher-level view of the security state of the system. The current work is inspired by this approach. Mathew et al. [5] report a work for alert fusion using the graph-based approach.

2.1 Proposed Approach

Based on majority of the papers that authors of this paper found and studied, the proposed method of approach for the alert correlation is mainly a rule-based approach (using inference engine and working memory that constitutes an Expert System). Major contribution of the reported works is about using scenarios to represent the attack pattern and the connection between the alerts. **Extracting scenarios is primarily a heuristic-based task that has to be performed by the human expert.**

Statistical methods are also used to analyze and correlate network features. These methods require a training dataset for extracting the correlation information. In this case, properties of the training dataset, e.g. the way dataset is collected and the validity of it, are very important. In other words, training with these datasets is similar to extracting scenarios, but with a difference. Scenarios are more suitable in modelling the attack patterns and easier to prepare and maintain. However, this approach requires a great deal of expertise and experience to extract the scenarios. Statistical approach, however, is less dependent on the expertise or experience of the professional in charge of training the system. Instead, statistical approach is sensitive to the completeness and the way training datasets are collected. It is also very difficult to maintain the statistical-based systems.

In the case of using an statistical approach, it is difficult to keep such a system up to date, because adding new attack patterns requires the system to be trained again. After training the statistical-based alert correlator, results have to be verified to make sure that the alert correlator operates correctly. Preparing the test dataset is an additional hard task for the system manager in this approach.

Although rule-based systems requires expert knowledge to operate, they are easier to implement and maintain. At the same time, rules can be in a text format understandable by the operator, and therefore, operator can add new rules of his/her own to the system. Preparing a comprehensive and up to date dataset can be a nightmare for any company that produces statistical-based alert correlator system. Therefore, a rule-based (expert system-based) approach is selected for this work.

3 Structure of the System

The first field of each record in the knowledge-base is the **FactName** that corresponds to the name of the fact/rule or the name of the scenario or the term ‘Scenario’. The **FactValue** field describes the value of the ‘FactName’ as it is required in the definition of the fact clause. In the case of defining a scenario rule, FactName/FactValue will represent the consequence part of the rule. *Parameter_x* and *Operator_x* are the parameter(s) and the operator(s) of the rule, respectively. Depending on the form of the rule, some of them might be NULL or “*” (Star character acts as a wildcard and means any value).

The Certainty Factor (**CF**) field of the record will determine the certainty over the correctness of that rule or fact. The Certainty Factor field can be a numerical value between 0 and 1, in the case of $CF=0$ this can be considered to be the same as NULL for the Certainty Factor (implementation of this part is left for future). Finally, **ClauseTime** field is a timestamp field that denotes time of the event. Timestamp can be used to extract sequence of the events.

$Operator_x$ can be either one of the AND or OR logical operations. Using database tables provides the benefit of being able to use any database system to perform queries. Queries can be easily performed using the SQL commands by which it is possible to perform the Forward/Backward chaining process. This method of implementation will significantly reduce the programming time required for completing the design.

Some may object to the performance of the inference engine. Actually, this is a valid point with respect to its performance. However, considering the low priority of the correlation in comparison with other parts of the IDS this drawback can be justified. In some occasions, the correlation engine might be even used for offline queries. Nevertheless, this approach is mainly a rapid prototyping approach and many things such as converting the string-based knowledge to a numerical format can improve its performance. Once the string comparison is eliminated from the code, the execution speed of the code will significantly increase. This issue will be discussed in more detail in Section 6.

Block diagram of the proposed expert system engine is depicted in Figure 1. Working memory and knowledge-base tables can be indexed on the RuleFact, Fact and either of the $Parameter_x$ fields. During the inference process, indexing will help us with forward and backward chaining through the knowledge-base. The aim of adding index to the tables is to increase the speed of the inference process. However, system has not gone under a heavy load yet and needs to be tested with a large dataset to be certain of its performance.

Receiving a new alert, the inference engine will check it with the current alerts stored in working memory (Fact-Name/FactValue). If necessary, the alert will be aggregated, otherwise it will be compared against the currently loaded scenarios. If needed, new matching scenarios will be loaded into the working memory. Later on, using the recently updated working memory, the inference engine will backtrack the included alerts and will follow the sequence of the events encountered with respect to different loaded scenarios. Using the available scenarios, system will produce a sequence of steps each corresponding to a different received alert. Following this sequence, user can track progress of the possible attacks.

Temporal alert correlation module is a part of a larger IDS system that is being developed in the faculty of Computer Science at the University of New Brunswick. Currently, this module is working in an offline manner waiting for the rest of the IDS blocks to get ready. If decided

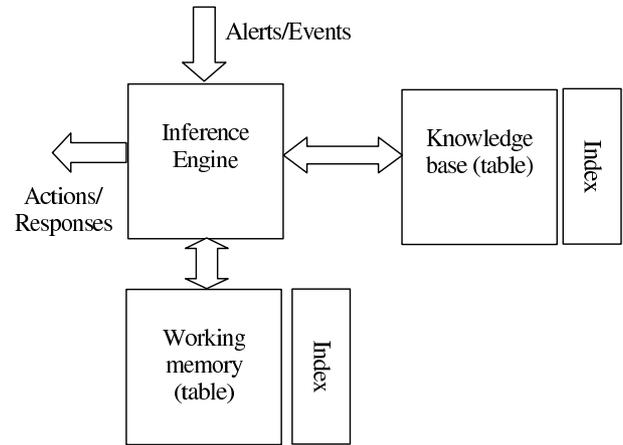


Figure 1: Block diagram of the expert system engine

not to add the contents of the working memory to the knowledge-base then the working memory should be removed (emptied). To do so, at the beginning of processing each new event, system will first check the current time. If the difference between the current time and the last marked time was greater than a threshold (time window), system will erase all the older facts and scenarios in the system. Then it will mark the current time as the last marked time.

As it was emphasized earlier in the text, using database engines will ease the coding of the expert system engine. This is because there are many cases where multiple conditions are going to be applied on different number of fields and/or a group of fields that have to be sorted with respect to different constraints/orders. In all of these cases, a simple database query can replace a complicated procedure that otherwise had to be hard coded. It might not look very efficient with respect to the processing time required, however, for the first prototype it is the best solution. At the same time, alert correlation is not a frequent event in the system and it is not expected to occur frequently. Unless its output is required for a quick response, it will not become a bottleneck for the system. In case of using SQL scripts for the queries, Borland SQL system, MS-SQL and MySQL database system are among the available options.

In order to improve the execution time, instead of loading the scripts from a file, system is using hard coded C commands to send the SQL commands to the server. It should be noted that the format of the rules in the knowledge-base should be kept simple. Each rule should not contain more than a few components (two or three components only) otherwise system will become unnecessarily complex.

Presently, for presenting the facts, only Fact-Name/FactValue fields are used. For the simplicity in the process, formats used for the Facts and Scenarios are

FactName	FactValue	Parameter1	ParameterValue1	Operator1	Parameter2	ParameterValue2	Operator2	Parameter3	ParameterValue3	CF	ClauseTime
Scenario	General	OverFlow	*	*	PortScan	80	*	*	*	0	1
Scenario	General	Transient	Intrusion	*	OverFlow	TCP	*	*	*	0	2
Scenario	General	BackDoor	*	*	Transient	Intrusion	*	*	*	0	3
Scenario	General1	OverFlow	*	*	PortScan	80	*	*	*	0	3
Scenario	General1	DoS	*	*	OverFlow	*	*	*	*	0	4
Scenario	General1	DDos	*	*	DoS	*	*	*	*	0	5
Scenario	General1	SynFlood	*	*	DDoS	*	*	*	*	0	6
Scenario	General1	Worm	*	*	DoS	*	*	DDos	80	0	7
Scenario	General1	DoS	80	*	PortScan	80	*	*	*	0	1
Scenario	General2	DDoS	80	*	DoS	80	AND	PortScan	80	0	2
Scenario	General2	SpecialO...	80	*	DoS	80	OR	SpecialOR	80	0	3
Scenario	General2	SpecialAN...	80	*	DoS	80	AND	SpecialAnd	80	0	4

Figure 2: Contents of the knowledge-base table

the same. The Scenario format is as follows:

'Scenario', < ScenarioName >
Parameter1 ← Parameter2 ⊗ Parameter3,

where FactName field holds the term 'Scenario', ScenarioName field holds the name for this scenario and Parameter 1 represents the consequence part of the rule. For now operator '⊗' can only be replaced by either 'AND' or 'OR' operators.

System is currently working in a simulated environment where events are saved in a text file and they will be loaded into a table before the execution starts. Later on, the result of the inference that is a sequence of events will be extracted from the scenarios that is already saved in a table called 'Scenario'. This table represents output of the system. Transient actions or responses are added to the WorkingMemory table in the form of new facts. Later on, adding new facts can be converted into an actual action. Introduction of the transient rules to the system enables the system to follow a sequence of rules within one clock cycle (one inference cycle).

4 Experimental Results

The general form of the knowledge-base records is in the form of if-then-else clauses. Using wildcards ('*') can be very helpful in making general rules. In order to provide a better understanding of the way inference engine works, the following example for the eventbase, knowledge-base and the scenario tables are provided.

Considering the knowledge presented in Figure 2, this knowledge contains three different scenarios General, General1, and General2, respectively. Any given event can make one or more of these scenarios to fire. As an example, the first line of the General scenario in the text file can be stated as follows:

Scenario,General,OverFlow,*,*,PortScan,80,*,*,*,1.0

The first item 'Scenario', indicates that this is a scenario rule and its name is 'General' (FactName). Parameter1 represents the consequence and its value is a wildcard that means OverFlow on all ports/any port. Operator1 contains a wildcard but it is going to be ignored by the

system. Parameter2 indicates a PortScan alert on port 80 (ParameterValue1). Since Parameter3 is a wildcard, Parameter3 together with its value and operator2 will be ignored by the system. CF is currently inactive so this field will be ignored as well. The final field is the ClauseTime that indicates the timestamp of the rule. Contents of the eventbase are presented in Table 1. Contents of the text file will look like this:

```
PortScan,80,*,*,*,*,*,*,*,0.0
PortScan,80,*,*,*,*,*,*,*,0.0
OverFlow,TCP,*,*,*,*,*,*,*,0.0
PortScan,80,*,*,*,*,*,*,*,0.0
PortScan,80,*,*,*,*,*,*,*,1.0
DoS,*,*,*,*,*,*,*,*,0.0
PortScan,80,*,*,*,*,*,*,*,0.0
```

Specifically for the knowledge-base, the ClauseTime field shows the sequence of the events in the scenario. In the first line of the Table 1 for example, Fact/FactName = PortScan/80 means that this alert is generated to report a PortScan on port 80. Finally, result of the inference process that is saved in the scenario table, is presented in Figure 3.

As it is clear from Figure 3, only those steps in the scenarios are executed that their prerequisite is present. The final consequence is each scenario presents a prediction for attackers next move. New timestamps indicate the selection time during the inference process. WorkingMemory table is presented in Figure 4.

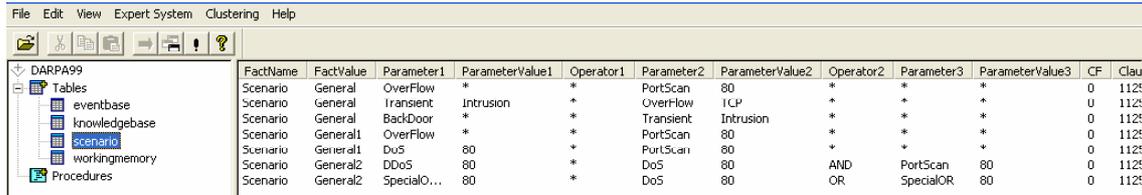
From the contents of the working memory it is seen that the system can successfully aggregate the redundant PortScan, 80 alerts. The two copies of the alert are due to the difference in their ClauseTime (timestamp) that makes them individually unique.

The reader should note that both of these scenarios are examples to show the execution of the inference engine and not necessarily effective and correct rules/scenarios for the real attack.

One of the major benefits in using the rule-based systems is the format of the rules. The human readable format of the knowledge-base or the rule-base will help us with maintaining and debugging the knowledge-base. Due

Table 1: Events that are presented to the EventBase table. Where FName=FactName, FVal=FactValue, P=Parameter, PV=ParameterValue, Op=Operator and CT=ClauseTime.

FName	FVal	P1	PV1	Op1	P2	PV2	Op2	P3	PV3	CF	CT
PortScan	80	*	*	*	*	*	*	*	*	*	0.0
PortScan	80	*	*	*	*	*	*	*	*	*	0.0
OverFlow	TCP	*	*	*	*	*	*	*	*	*	0.0
PortScan	80	*	*	*	*	*	*	*	*	*	0.0
PortScan	80	*	*	*	*	*	*	*	*	*	1.0
DOS	*	*	*	*	*	*	*	*	*	*	0.0
PortScan	80	*	*	*	*	*	*	*	*	*	0.0



FactName	FactValue	Parameter1	ParameterValue1	Operator1	Parameter2	ParameterValue2	Operator2	Parameter3	ParameterValue3	CF	Clau
Scenario	General	OverFlow	*	*	PortScan	80	*	*	*	0	112z
Scenario	General	Transient	Intrusion	*	OverFlow	ICP	*	*	*	0	112z
Scenario	General	BackDoor	*	*	Transient	Intrusion	*	*	*	0	112z
Scenario	General1	OverFlow	*	*	PortScan	80	*	*	*	0	112z
Scenario	General1	DoS	80	*	PortScan	80	*	*	*	0	112z
Scenario	General2	DDoS	80	*	DoS	80	AND	PortScan	80	0	112z
Scenario	General2	SpecialO...	80	*	DoS	80	OR	SpecialOR	80	0	112z

Figure 3: A sample instance of the scenario table

to its string-based rule definition, using the rule-based system will slow down the execution of the code. Text file processing, string parsing and the volume of data transfer will affect the performance of the system and will reduce its speed. This problem is going to slowdown search for the appropriate rules and facts within the knowledge-base. In order to eliminate aforementioned problem, it is intended to find an appropriate data structure that can address the problem and prevent the slowdown of the search process.

5 Conclusions

The proposed approach is to use tables to accommodate rules. Here, every field of the table can represent a part of the rule, e.g. name, operation, operands, etc. Using the table will enable the system to speed up the search process and to avoid the overhead that is caused by the text processing. At the same time, names and other string parts of the rule can be presented in a numerical form. This part will require a preprocessing to parse and convert the general form of the rules e.g. name (param1, param2) into a numerical form. As another way for implementing this conversion, it can be considered to have rules stored in the text format in a table and later on, the preprocessing can convert them into the numerical format.

This preprocessing will both increase the search speed and will reduce the size of the information stored in the tables. Since the preprocessing is required only when the knowledge-base is updated and can be an offline process, it will not enforce any processing load over the system. For the time being, this part is not yet implemented.

There is an additional factor influencing the perfor-

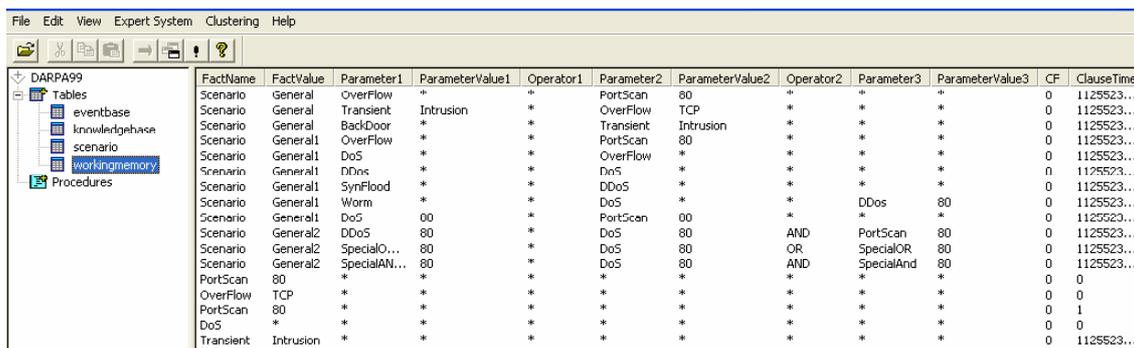
mance and execution time of the system. This factor is the length of the rule chain in the inference process. It is important to avoid using long sequences in the knowledge presentation within the knowledge-base. The longer is the knowledge chain in the knowledge-base, longer will be the execution time for the inference process.

Finally, as stated in Section 2.1, similar to any other expert system, this approach can only deal with the known attack scenarios. In order to make it operational it requires a number of attack scenarios stored in its knowledge-base to become operational. Extracting these scenarios (knowledge) is the responsibility of the network security experts. Knowledge engineer will prepare these scenarios in an appropriate format suitable for the knowledge-base. It is required that a human expert in network security or a hacker to study the current strategies for intrusion and to prepare them as intrusion scenarios. The goal for the system is to connect the dots even if they are implicitly connected.

6 Future Work

The Certainty Factor (CF) is a numerical value that presents our certainty over the rule or the fact associated with it. CF is introduced as a parameter to the system to add the uncertainty into the inference process. Implementing the CF in the inference system, the certainty of the results over the inferred decision can be calculated. However, currently this feature is not yet used in the inference process.

It would be very effective if the concept of ‘Demon’ can be implemented in our expert system engine. Demons can provide us with a vital and effective feature in the



FactName	FactValue	Parameter1	ParameterValue1	Operator1	Parameter2	ParameterValue2	Operator2	Parameter3	ParameterValue3	CF	ClauseTime
Scenario	General	Overflow	*	*	PortScan	80	*	*	*	0	1125523...
Scenario	General	Transient	Intrusion	*	Overflow	TCP	*	*	*	0	1125523...
Scenario	General	BackDoor	*	*	Transient	Intrusion	*	*	*	0	1125523...
Scenario	General1	Overflow	*	*	PortScan	80	*	*	*	0	1125523...
Scenario	General1	DoS	*	*	Overflow	*	*	*	*	0	1125523...
Scenario	General1	Dfnc	*	*	DoS	*	*	*	*	0	1125523...
Scenario	General1	SynFlood	*	*	DDoS	*	*	*	*	0	1125523...
Scenario	General1	Worm	*	*	DoS	*	*	DDos	80	0	1125523...
Scenario	General1	DoS	00	*	PortScan	00	*	*	*	0	1125523...
Scenario	General2	DDoS	80	*	DoS	80	AND	PortScan	80	0	1125523...
Scenario	General2	SpecialO...	80	*	DoS	80	OR	SpecialOR	80	0	1125523...
Scenario	General2	SpecialAN...	80	*	DoS	80	AND	SpecialAnd	80	0	1125523...
PortScan	80	*	*	*	*	*	*	*	*	0	0
Overflow	TCP	*	*	*	*	*	*	*	*	0	0
PortScan	80	*	*	*	*	*	*	*	*	0	1
DoS	*	*	*	*	*	*	*	*	*	0	0
Transient	Intrusion	*	*	*	*	*	*	*	*	0	1125523...

Figure 4: A sample instance of the WorkingMemory table

system that is asynchronous response to new and unexpected events in the system. In other words, Demons can provide an appropriate response for exceptions in the inference process in a timely manner.

Actions/Responses are outputs from the inference engine. Once the inference process reaches to a conclusion or a demon fires, the generated fact in the working memory can also become an output for the system. This output can be a hyper-alert, urgent response command or any other action that might be needed. These actions/responses can themselves be a rule or a fact stored in the knowledge-base.

Intension is to start extracting/collecting attack scenarios to fill the knowledge-base of the system. Scenario extraction is the next phase of the work that requires extensive research about the attack scenarios.

Acknowledgements

This work was funded by the Atlantic Canada Opportunity Agency (ACOA) through the Atlantic Innovation Fund (AIF) to Dr. Ali A. Ghorbani.

References

- [1] F. Cuppens, F. Autrel, A. Mige, and S. Benferhat, "Correlation in an intrusion detection process," in *Proceedings of the Internet Security Communication Workshop (SECI'02) (Tunisia)*, pp. 153-172, Sept. 2002.
- [2] F. Cuppens, F. Autrel, A. Mige, and S. Benferhat, "Recognizing malicious intention in an intrusion detection process," in *Proceedings of Second International Conference on Hybrid Intelligent Systems*, pp. 806-817, Dec. 2002.
- [3] F. Cuppens and A. Mige, "Alert correlation in a cooperative intrusion detection framework," in *Proceedings of 2002 IEEE Symposium on Security and Privacy*, pp. 202-215, 2002.
- [4] J. Kim, K. Kang, J. Na, I. Kim, K. Kim, J. Jang, and S. Sohn, "Practical network attack situation analysis using sliding window cache scheme," in *The 9th*

Asia-Pacific Conference on Communications (APCC 2003), vol. 3, pp. 1038-1041, Sept. 2003.

- [5] S. Mathew, S. Chintan, and S. Upadhyaya, "An alert fusion framework for situation awareness of coordinated multistage attacks," in *Proceedings of the Third IEEE International Workshop on Information Assurance (IWIA '05)*, pp. 95-104, 2005.
- [6] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *Proceedings of the 9th ACM conference on Computer and communication security (Washington D.C., USA)*, pp. 245-254, Nov. 2002.
- [7] P. Ning, Y. Cui, D. S. Reeves, and D. Xu, "Techniques and tools for analyzing intrusion alerts," *ACM Transactions on Information and System Security*, vol. 7, no. 2, pp. 274-318, 2004.
- [8] A. Valdes and K. Skinner, "Probabilistic alert correlation," in *Proceedings of 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, pp. 54-68, CA, USA, Oct. 2001.
- [9] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "Comprehensive approach to intrusion detection alert correlation," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 3, pp. 146-169, 2004.



Peyman Kabiri received his PhD in Computing (Robotics and Machine Learning) and MSc in Real time Systems from the Nottingham Trent University, Nottingham-UK in years 2000 and 1996 respectively. He received his B.Eng. in Computer Hardware Engineering from Iran University of Science and Technology, Tehran-Iran in 1992. He is currently Assistant professor in Department of Computer Engineering Iran University of Science and Technology. His previous academic positions were as follows: He was with the Faculty of Computer Science/ University of New Brunswick as project coordinator acting as a research associate from 7th September 2004 till 30th September 2005. Assistant professor in Department of Computer Engineering Iran

University of Science and Technology and Assistant Professor in Azad University - central branch - Faculty of Engineering both in Tehran. He was a reviewer in several conferences. His research interests include Machine Learning, Robotics and Network Intrusion Detection.



Ali A. Ghorbani received his PhD and Master in Computer Science from the University of New Brunswick, and the George Washington University, Washington D.C., USA, respectively. He was on the faculty of the Department of Computer Engineering, Iran University of Science and Technology,

Tehran, Iran, from 1987 to 1998. Since 1999, he has been at the faculty of Computer Science, University of New Brunswick (UNB), Fredericton, Canada, where he is currently a Professor of Computer Science. He is also a member of the Privacy, Security and Trust (PST) network, University of New Brunswick.

He has held a variety of positions in academia for the past 26 years. His research originated in software development, where he designed and developed a number of large-scale systems. His current research focus is Neural Networks, Web intelligence, agent systems, complex adaptive systems, and trust and security. He established the Intelligent and Adaptive Systems (IAS) and Network Security research groups in 2002 and 2004, respectively, at the faculty of Computer Science, UNB. The IAS group (<http://ias.cs.unb.ca>) pursues research on machine and statistical learning, data mining, intelligent agents and multiagent systems and Web intelligence. The NSL group (<http://nsl.cs.unb.ca>) is home to R&D in computer and network security.

He authored more than 100 research papers in journals and conference proceedings and has edited 6 volumes. He is on the editorial board of the Computational Intelligence (CI), an international journal. He is also associate editor of International Journal of Information Technology and Web Engineering.

Dr. Ghorbani a member of ACM, IEEE, IEEE Computer Society, and ANNS.