

Advanced Reaction using Risk Assessment in Intrusion Detection Systems

Wael Kanoun¹, Nora Cuppens-Boulahia¹, Frédéric Cuppens¹, and Fabien Autrel²

¹ ENST-Bretagne, Cesson Sévigné 35576, France.

wael.kanoun, nora.cuppens, frederic.cuppens@enst-bretagne.fr

² SWID, Cesson Sévigné 35512, France.

autrel.fabien@gmail.com

Abstract. Current intrusion detection systems go beyond the detection of attacks and provide reaction mechanisms to cope with detected attacks or at least reduce their effect. Previous research works have proposed methods to automatically select possible countermeasures capable of ending the detected attack. But actually, countermeasures have side effects and can be as harmful as the detected attack. In this paper, we propose to improve the reaction selection process by giving means to quantify the effectiveness and select the countermeasure that has the minimum negative side effect on the information system. To achieve this goal, we adopt a risk assessment and analysis approach.

Keywords: Intrusion detection system, attack scenario, countermeasure, risk analysis, potentiality, impact.

1 Introduction

In intrusion detection approach [1], several security monitoring modules exist. A module gathers and correlates the generated alerts to recognize the current attack and ask the administrator to take action to prevent the damage of the attacks[2]. After all, in the intrusion detection approach, it is almost useless to recognize the attack without having the means to stop it.

There are two different approaches for the reaction perspective: Hot reaction and policy based reaction. The first aims to launch a local action on the target machine to end a process, or on target network component to block a traffic, that are the cause of the launched alerts. For example *Kill process*, *Reset connection*, *Quarantine* can be used to react against an attack. The second acts on more general scope; it considers not only the threats reported in the alerts, but also constraints and objectives of the organization operating the information system and this by modifying the access policy. Therefore a trade-off can be established between security objectives, operation objectives and constraints.

Whatever the adopted approach, each countermeasure can have negative or positive side effects. The same countermeasure that was activated to end an attack can make the information system more vulnerable, expose it to other

attacks, or even have an impact more disastrous than the attack itself. For example *Firewall reconfiguration* is effective against a DOS attack, but can be very harmful if valuable connections will be lost, therefore many questions emerge: Is it better to stand still? Or is the attack harmful enough to react? In this case, which countermeasure must be selected with minimum negative side effects?

To answer these questions, we adopt a risk analysis approach. Risk analysis is a known method to analyze and evaluate the risks that threaten organization assets. The first step of a risk analysis method is to collect data that describes the system state; the second step is analyze them and find the potential threats and their severity; and the final step is to study the countermeasure effectiveness to eliminate these threats or reduce their severity. The existing methods are used to manage system assets and evaluate the risk that threatens these assets: they are unfortunately abstract, informal and not fully compatible with intrusion detection and computer systems. In section 2, related works are presented. The model is presented in section 3, and an implementation is showed in section 4. Finally section 5 concludes this paper.

2 Related Works

In intrusion detection approach, the final objective is to detect intrusions and then block them to prevent the attacker to achieve his or her objective. First, to detect and recognize the current attack, an alerts correlation procedure is needed. The correlation procedure recognizes relationships between alerts in order to associate these alerts into a more global intrusion scenario, and the intrusion objectives that violates the predefined organization security policies. There are many approaches that can be used for this purpose: implicit [4], explicit [5, 6] and semi-explicit [7, 8] correlations. The semi-explicit approach is based on the description of the elementary intrusions corresponding to the alerts. This approach then finds causal relationships between these elementary alerts and connects these elementary alerts when such a relationship exists. The correlation procedure then consists in building a scenario that corresponds to an attack graph of steps corresponding to the elementary intrusions. This approach is more generic and flexible because only the elementary steps are defined as entities and not the whole attacks scenarii. Regarding reaction, it is also the most interesting because it provides a precise diagnosis of the ongoing intrusion scenario. Using an approach similar to the one used to describe elementary intrusions, elementary countermeasures can be specified. In this case, anti-correlation [9] can be used to find the countermeasures capable of ending a detected scenario. Anti-correlation approach is based upon finding the appropriate countermeasure that turn an elementary future step of an attack inexecutable due to preconditions value modifications. Therefore, using anti-correlation approach, the administrator knows which countermeasures from a predefined library those who are capable of blocking the threat.

There are two types of reaction: Hot Reactions and Policy Based Reactions [10]. In the first case, simple countermeasures are activated to end the detected

attack. The advantage is fast reaction guaranteed by activating a simple countermeasure; therefore the threat is instantaneously terminated. In other hand, hot reactions do not prevent the occurrence of the attack in the future, therefore a countermeasure is activated each time the attack occurs. Policy based reactions consists of modifying or creating new rules in the access policy to prevent an attack in the future, therefore it corresponds to a long term reaction.

Whatever the adopted type of reaction, a countermeasure could have negative impact on the information system. For example, in some situations, an administrator prefers not to react because the risk of the detected attack is smaller than the risk resulting from triggering off a candidate countermeasure. The goal is not always to block the attack, but to minimize the risk incurred by target information system. Therefore a risk assessment method is needed to evaluate and quantify the risk of an attack and its countermeasures. The method is useful to decide when it is preferable to react, and which countermeasure should be activated. There are several Risk Assessment methods like EBIOS [11, 12], MARION [13], MEHARI [14], etc. These methods are used to manage system assets and evaluate the risk that threatens these assets; they are unfortunately abstract, informal and incompatible with intrusion detection and computer systems: Many elements and parameters are related to physical and nature disasters (fire, earthquake, failure, etc.). Besides, there are many essentials factors that exists in intrusion detection systems and even in computer systems (networks, firewall, software, etc.) that does not exist in these methods. There are also elements that need redefinition to be compatible with the intrusion systems like potentiality and impact of a threat.

3 Risk Assessment Model

MEHARI method is the latest, more accurate and flexible risk analysis method, and that is why we decided to adapt this method to detection intrusion systems. Therefore, we propose a new risk analysis method compatible with intrusion detection systems inspired from MEHARI, by adapting, redefining and adding new parameters and functions.

The risk is defined as a potential exploitation of an existing vulnerability; this exploitation has an impact on the affected assets. Therefore, the gravity of risk *Grav* of an attack scenario is the combination of two major factors: Potentiality *Pot* and Impact *Imp*. Each of the major factors depends on minor factors. In turn, these minor factors are evaluated using audit clusters. In MEHARI method, an audit cluster is a group of general questions to determine the general system state. In our approach, an audit cluster contains multiple coherent questions or tests, therefore the value of an audit cluster is the arithmetic mean value of all the questions-tests. These questions-tests aim to evaluate a specific service state (antivirus, firewall, vulnerabilities, etc.) in real time when an attack occurs.

In our method, we choose to evaluate the risk gravity, the major and minor factors and the audit clusters with a scale that ranges from 0 to 4. The value 0 is used when the studied element does not exist, and the value 4 in the maximum

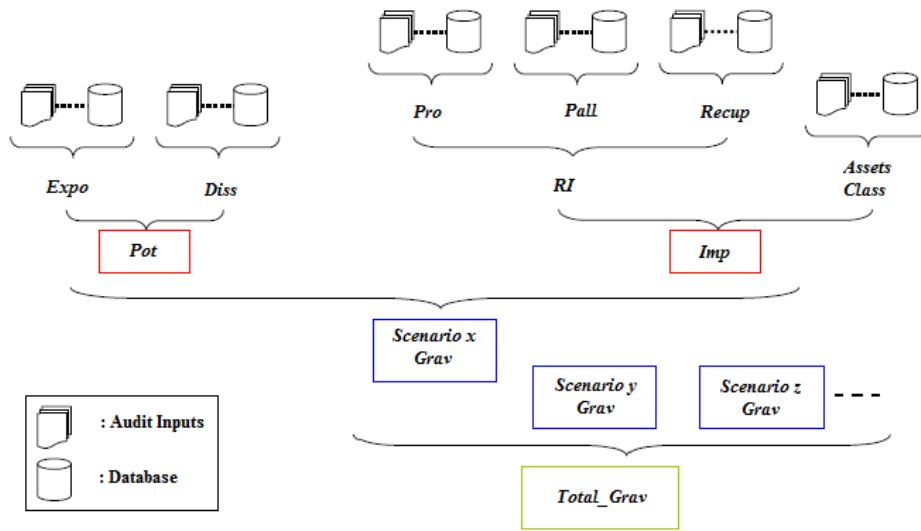


Fig. 1. Risk Assessment structure

value that an element can have. This scale is sufficient to evaluate these factors, and a larger scale would be more confusing for the system administrators.

The structure of the model is shown in Fig.1. The total risk *Total_Grav* is the combination of the candidate scenarii risk gravities. In the following sections we detail each of the risk gravity's factors.

3.1 Potentiality *Pot*

The major factor Potentiality *Pot* measures the probability of a given scenario to take place and achieve its objective with success. To evaluate *Pot*, we must first evaluate its minor factors: natural exposition *Expo* and dissuasive measures *Diss* and we have to take into account classification of the attack also. The minor factors can be evaluated after the appropriate audit clusters are calculated. These questions-tests aim to evaluate the system state (active services, existent vulnerabilities, etc.). As we aforementioned, all these elements can have values between 0 and 4. The value 0 indicates that the studied scenario is impossible, and the value 4 indicates that the occurrence and the successful execution of the scenario are inevitable.

Natural Exposure *Expo* This minor factor measures the natural exposure of the target system faced to the detected attack, and by adopting a defensive centric view. It reflects the system state contribution (established connections, acquired privileges, existing vulnerabilities, time, previous incidents, etc.). We propose the following audit clusters that contribute in the evaluation of *Expo*:

- *LAMBDA_Prediction*: This is the most important cluster for the *Expo* evaluation. It estimates the probability of occurrences of the studied scenario. This is possibly done by the analysis of the predicates and the facts that describes the pre and post conditions of an elementary attack step. LAMBDA language [15] is an example to describe these elementary steps by defining the pre and post conditions. The value of this cluster increases each time the attacker gets closer to his or her intrusion objective. This cluster is estimated in real time.
- *History*: This cluster indicates the number of scenario incidents achieved with success in the past. If *History* increases, EXPO increases as well. This cluster is most useful in the situation where an elementary step in the attacks graph is common to two or more scenarios. This cluster is very similar to the concept of Natural Exposure in MEHARI method, and it is estimated offline.
- *Global_Time*: Some attacks occur in special hours or dates (at night, weekends or holidays), and the same behavior could be normal during a time, but it is an evidence of an attack during another time. For example, a high level of network traffic is normal during the day, but suspicious during the night. If the scenario is independent of time, this cluster will have the mid scale value (thus 2).

Using these audit clusters, *Expo* can now be calculated:

$$EXPO = \frac{\alpha * LAMBDA_Predict + \beta * History + \gamma * Global_Time}{\alpha + \beta + \gamma} \quad (1)$$

where α, β and γ are three coefficients that depend on the system.

Dissuasive Measures *Diss* To reduce the probability of an attacker to progress with his or her attack (and thus to decrease the potentiality *Pot*), dissuasive measures *Diss* can be enforced. They aim particularly to reduce the attackers' aggression. They are useful against human attackers, not automated attacks or accidents. If *Diss* increases, *Pot* normally will decrease because *Diss* makes the attacks riskier and more difficult. We propose three clusters to evaluate these measures:

- *Logging*: This cluster joins all the questions-tests that check the installation and the state of all logging mechanisms. This cluster is evaluated in real time to verify that the attack actions are logged.
- *Send_Warning*: This cluster indicates if it is possible to send warning to an attacker (knowing his or her IP, his or her username, etc.). In fact, warnings play a significant role to reduce scenarios *Pot* because the attacker knows that he or she was caught red handed and it is risky to let the attack go on. This cluster is evaluated in real time.
- *Laws*: This cluster checks if the current attack can be considered a violation of the local, national or international laws. The fact that a law forbids a particular attack reduces its probability because it is risky for the attacker. This cluster is evaluated offline and in real time, because we should first verify the laws in the attacker's country and the target country as well.

The minor factor *Diss* can now be calculated, using the three audit clusters presented above. These clusters have the same weight effect on the attacker aggression level:

$$DISS = \frac{LOGGING + SEND_WARNING + LAWS}{3} \quad (2)$$

Evaluation of Potentiality *Pot* After we had estimated *Expo* and *Diss*, we can now estimate the major factor *Pot*. However, the attacks scenarii are too far to have the same proprieties regarding their potentiality. Therefore, we will propose a classification, and associate each class with a specific function to calculate *Pot*. This classification provides means to evaluate POT more accurately and realistically. Many taxonomies were proposed in [16, 17]. In our approach, we will consider two classes: Malicious Actions and Accidents. These two classes allow us to consider if the human factor is involved, and therefore the effectiveness of *Diss*.

For Malicious Actions, having *Expo* and *Diss*, we propose to use a predefined 2D matrix to calculate *Pot*. For Accidents and Non-Malicious Actions, *Diss* is useless and therefore:

$$Pot = Expo \quad (3)$$

3.2 Impact *Imp*

The second major factor to evaluate Risk Gravity of an attack scenario is Impact *Imp*. \vec{Imp} is defined as a vector with three cells that correspond to the three fundamental security principles: Availability *Avail*, Confidentiality *Conf* and Integrity *Integ*. Therefore, with each Intrusion Objective, a vector \vec{Imp} is associated and should be evaluated. Actually, it is not possible to statically evaluate \vec{Imp} of a scenario (or more precisely the \vec{Imp} of the scenario's intrusion objective) directly because it depends on several dynamic elements. The impact depends on the importance of the target assets \vec{Class} , and the impact reduction measures level \vec{IR} that are deployed on the system to reduce and limit the impact once the attack was successful.

Assets Classification *Class* For each attack, the attacker seeks to achieve it by successfully executing the last node in the scenario graph: The final intrusion objective that violates the system security policy. Each intrusion objective is of course associated with a sensitive asset. However, the assets have different levels of classification that depend on their importance for system functionalities and survivability. Therefore, for each asset, we will associate the vector \vec{Class} with 3 cells that represent three types of classification relative to the three cells of \vec{Imp} (*Avail*, *Conf* and *Integ*). \vec{Class} will be the sum of two components: $\vec{Static_Class}$ that is evaluated offline and reflects the intrinsic value of the asset, and $\vec{Dynamic_Class}$ that can be evaluated online using the following audit clusters:

- *Global_Time*: The assets value can depend on time. For instance, confidentiality level of information decreases over the time, or the availability of a server is less required during holidays, etc.
- *Conn_Nbr*: Connection number is most useful to calculate *Dynamic_Class* that increases if the number of connections increases.

Impact Reduction \vec{IR} To face attacks, many measures are used to reduce their impact. Vector \vec{IR} aims to evaluate the measures levels that reduce the impact relative to *Avail*, *Conf* and *Integ*. Therefore *vector* \vec{IR} contains three cells: IR_{Avail} , IR_{Conf} and IR_{Integ} . There are three sets of measures: Protective Measures *PRO* to reduce direct consequences, Palliative Measures *Pall* to reduce indirect consequences and Recuperative Measures *Recup* to reduce the final losses. Once *Pro*, *Pall* and *Recup* are calculated, we can evaluate the three cells of \vec{IR} . For each cell, we define a 3D matrix to calculate the component IR_x in function of the three types of measures. In the following sections, we present the three sets of impact reduction measures and then how to calculate them using a specific taxonomy:

Protective Measures Pro: The Protective Measures or *Pro* aim, once the attack was executed successfully, to limit and contain direct negative consequences. Therefore, the goal of *Pro* is not to prevent the attack itself, but to confine the attack damage and prevent its propagation and therefore the infection of other assets of the system. For instance, a firewall prevents worms from propagating through the network. Thus, *Pro* is a major factor to reduce the impact of an intrusion. To evaluate *Pro*, we propose the use of the following audit clusters:

- *Antivirus*: The role of this cluster is to check if an antivirus is installed. It also checks if its signature base is up to date.
- *Antispyware*: Similar to the previous Cluster, it checks the antispyware deployed in the system.
- *Quarantine*: This cluster checks if quarantine mechanisms are installed and if they are effective against the detected scenario.
- *Firewall*: This cluster checks if filtering components, like firewalls, are installed and the effectiveness of their configuration against the detected attack.
- *Global_Time*: The systems level of protection depends on time. For instance, many machines are turned off during night and therefore they are protected. It checks the system clock to verify if the target assets are more vulnerable or exposed in the attack occurrence time.
- *Admin_Avail*: Actually, the attacks can occur anytime and anywhere. This cluster aims to determine if the administrator is available when the attack is detected. In other hand, the administrator works in specific period and he or she is not always present when an attack occurs. Therefore, when the administrator is absent, *Prot* and the three cells of \vec{IR} decrease and those of \vec{Imp} increase.

Palliative Measures Pall: Palliative Measures *Pall* verify the system capability of reducing indirect consequences of attacks. In other words, *Pall* is used to maintain the system functionalities running as normally as possible. These measures are essential to reduce the impact and the risk gravity of the detected attacks. To evaluate *Pall*, we propose the following audit clusters:

- *Backup_Ready*: This cluster checks if the target assets (especially if they were victim of an attack that violates their integrity or availability) have ready-to-use backups (or already in use as it is the case for load distribution). These backups reduce significantly the impact of an attack on the system. For example, the impact of a DOS attack on a given server has low impact if an existing backup server is ready to be used.
- *Admin_Avail*: If the administrators, who are capable to properly react and limit the indirect consequences and to assure the good operation of the system, are not available, *Pall* will dramatically decrease and the impact of the attack increases.

Recuperative Measures Recup: After every attack, the system will suffer from losses: confidentiality, integrity or availability of hardware or software assets that can ultimately lead to financial losses. To reduce these kind of losses, Recuperative Measures *Recup* can be used. The recuperation process is generally complex, but it reduces the final losses of the system and the company who owns that system. To evaluate *Recup*, we propose the following audit clusters:

- *Backup_Exist*: This cluster checks if the target assets have backups. In general, these backups are not ready-to-use (offline backups), and a specific procedure is required to put them into service. In spite of that, they are much useful to reduce final losses, for instance: Stored hard disks containing redundant data, or a router waiting to be configured and activated.
- *Third_Party*: In some special cases, we cannot limit the losses, but it is possible to subscribe to a third party like an assurance company that takes charge of these losses. Thus, the final losses and the impact of an attack are reduced. Therefore, this cluster verifies if the target assets are assured, or if the cost and the losses are partially or totally undertaken by a third party.

Evaluation of Pro, Pall and Recup: As we explained, the impact gravity depends on the attack's intrusion objective. These objectives are not similar: An impact reduction measure that can be effective against an objective *A* can be useless against another objective *B*. Therefore we need "attack centric" taxonomy to classify these objectives and associate each class with the functions to calculate *Pro*, *Pall* and *Recup* using their audit clusters. We propose to define five categories: (1) User to Root, (2) Remote to Local, (3) Denial of Service, (4) Probe and (5) System Access.

Evaluation of Impact After having calculated the \overrightarrow{Class} and \overrightarrow{IR} vectors, we can evaluate \overrightarrow{Imp} . Like \overrightarrow{Class} and \overrightarrow{IR} , \overrightarrow{Imp} is a vector with three components:

Imp_{Avail} , Imp_{Conf} and Imp_{Integ} . To calculate the component Imp_x , we can use a predefined 2D matrix to combine IR_x and $Class_x$. The IR_x has the effect to reduce Imp_x whereas $Class_x$ has the opposing effect. After that the three components of \vec{Imp} are calculated, we keep the component that has the highest value therefore:

$$Imp = \max(Imp_x); x \in \{Avail, Conf, Integ\} \quad (4)$$

3.3 Risk Gravity of an Attack Scenario $Grav$

For each detected attack, the risk gravity must be evaluated to estimate the danger level of this attack. The risk is the combination of Potentiality and Impact. An attack that occurs frequently with little impact may have the same risk level as another rare attack that have significant impact. In our approach, we use a 2D matrix to calculate the scenario's gravity of risk. If a scenario has Pot or Imp equal to zero, the scenario's gravity risk $Grav$ will be null. $Grav$ of a scenario u can be calculated using the function f defined as a 2D matrix:

$$Grav^u = f(Pot^u; Imp^u) \quad (5)$$

Table 1. Example of function f for $Grav^u$ evaluation

$POT^u \backslash Imp^u$	0	1	2	3	4
0	0	0	0	0	0
1	0	0	1	2	3
2	0	0	1	3	4
3	0	1	2	3	4
4	0	2	3	4	4

If a scenario u has $Imp^u = 3$ but $Pot^u = 1$, therefore $Grav^u$ will be lower (= 2) than Imp^u considering the fact that scenario u has a low potentiality.

3.4 Total Risk Gravity $Total_Grav$

In most situations, the correlation and reaction module do not deal with one specific scenario. Instead, the module have to take into account many candidate and even simultaneous scenarios. Therefore, before estimating the total gravity of risk, we must evaluate the gravity of risk of each scenario separately as mentioned in the sections 3.1, 3.2 and 3.3. Then we define the Total gravity as an ordered vector containing the values of gravity risk of each candidate scenario (See Fig.2). An order relation can be defined between the different instances of $Total_Grav$ using the lexicographic comparison. Therefore we are able to judge which graph has the highest risk gravity.

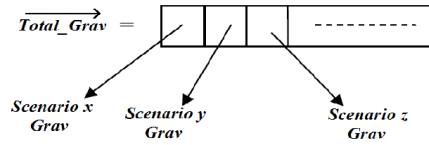


Fig. 2. Construction of $\overrightarrow{Total_Grav}$

3.5 Reaction and Countermeasure Selection

The correlation and reaction modules suggest for each detected attack a set of countermeasures using anti-correlation [9] approach. This set is called *Anticorrelation_CM*. These countermeasures are capable of stopping the detected attack. In other hand, the reaction is based on the assessment of the countermeasure negative and/or positive side effects on the information system. These side effects can be evaluated using the previously introduced method. Therefore, to judge if a countermeasure u is acceptable, a comparison must be done between:

- Situation "Before" : This is the state of the Information system before the execution of a countermeasure u . The correspondent risk is $\overrightarrow{Total_Grav}$
- Situation "After" : This is the state of the Information system after the simulated execution of a countermeasure u . The correspondent risk is $\overrightarrow{Total_Grav_CM_u}$ that depends on two elements: (1) countermeasure intrinsic impact G_{CM_u} and (2) information system future state $\overrightarrow{Total_Grav_u}$ that we define below.

Countermeasure Intrinsic Impact and Risk Gravity A countermeasure can have negative impact on the information system due to its intrinsic impact. We propose to associate with the countermeasure description a field called *Impact*. This field is an integer between 0 and 4. To calculate the Risk Gravity G_{CM_u} of the countermeasure u , the function f is used, the *Pot* parameter value is 4 because the execution of the studied countermeasure is guaranteed once it was selected, and the *Imp* parameter value is the one that exists in the countermeasure description.

$$G_{CM_u} = f(Pot = 4, Imp = CM.Impact) \quad (6)$$

System Information Risk Gravity after Reaction A countermeasure u modifies *Pot* or *Imp*. Therefore, we must evaluate the new $\overrightarrow{Total_Grav_u}$ after the selection of the countermeasure u . The same method is used to calculate $\overrightarrow{Total_Grav_u}$ as $\overrightarrow{Total_Grav}$, but using the new graph attack and the information system state after the simulated execution of the countermeasure u . The new graph attack is due to the modification caused by the simulated countermeasure execution in the relations found by the correlation module.

Countermeasure Selection Procedure Once for each countermeasure u , G_{CM_u} and $\overrightarrow{Total_Grav}_u$ are evaluated, $\overrightarrow{Total_Grav_CM_u}$ can be evaluated :

$$\overrightarrow{Total_Grav_CM_u} = \overrightarrow{Total_Grav}_u \cup G_{CM_u} \quad (7)$$

Now, only the countermeasures from *Anticorrelated_CM* that decrease the total gravity risk are kept and a new set *Risk_Eff_CM* is defined that contains only risk efficient countermeasures:

$$\begin{aligned} \forall CM_u \in Anticorrelated_CM; \overrightarrow{Total_Grav_CM_u} \leq \overrightarrow{Total_Grav} \quad (8) \\ \Rightarrow CM_u \in Risk_Eff_CM \end{aligned}$$

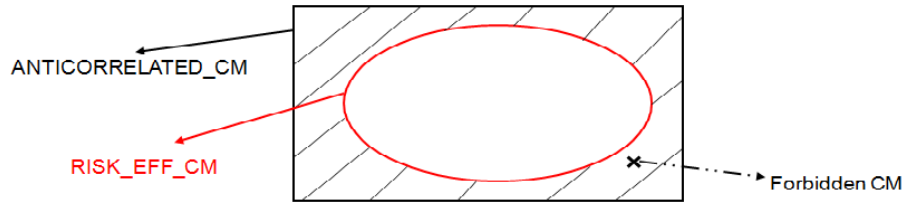


Fig. 3. *Anticorrelated_CM* and *Risk_Eff_CM* sets

Now, the procedure used to judge if a reaction is necessary and which countermeasure to select is the following:

```

If (( $CM_u \in Risk\_Eff\_CM$ ) and
    ( $Total\_Grav\_CM_u = \min_{CM_i \in Risk\_Eff\_CM}(CM_i)$ ))
 $CM_u$  is selected
Else
No Reaction

```

4 Deployment Application

To verify the utility and the effectiveness of our model, we seek in this section to evaluate the risk of the Mitnick attack and the candidate countermeasure to judge if the countermeasure is effective. The Mitnick attack graph generated by CRIM [18] using the LAMBDA language [15] is composed of four elementary steps (See Fig.4). We suppose that the attacker was capable to execute successfully the first three steps. Therefore one final step remains before the attacker achieves his or her intrusion objective *Illegal Remote Shell* on a critical machine, and we suppose the correspondent Impact is maximum (= 4).

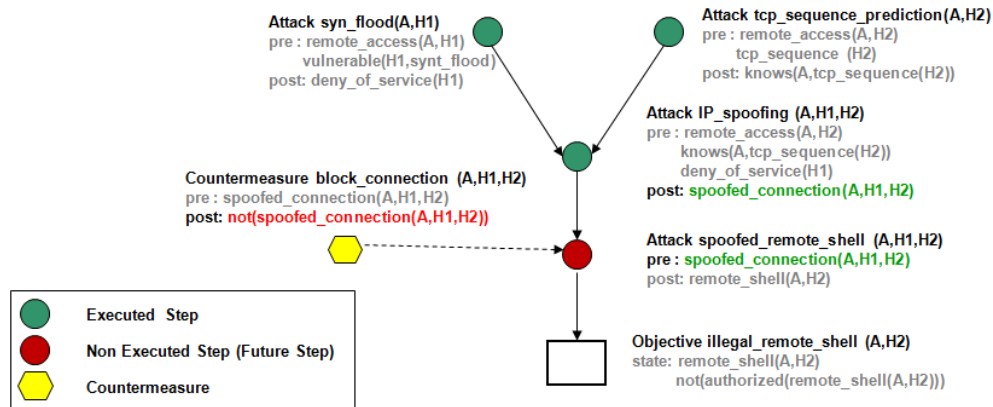


Fig. 4. Mitnick Attack graph generated by CRIM using LAMBDA language

Before Reaction The *Attack spoofed_remote_shell* precondition is *true*. The attacker is very close to his or her Intrusion objective and $LAMBDA_Pred = 4 \Rightarrow Pot = 4$. Therefore $GRAV_{MITNICK} = f(Pot = 4, Imp = 4) = 4$. Mitnick is the only candidate scenario, so $\overrightarrow{Total_Grav} = 4$

After Reaction Simulation The *Attack spoofed_remote_shell* precondition becomes *false*, so $LAMBDA_Pred$ decreases and $Pot = 1$.

Therefore $GRAV'_{mitnick} = f(Pot = 1, Imp = 4) = 3 \Rightarrow \overrightarrow{Total_Grav'_{block_conn}} = 3$. We suppose that $G_{block_conn} = 1 \Rightarrow \overrightarrow{Total_Grav_block_conn} = 3, 1$.

It is clear that $\overrightarrow{Total_Grav_block_conn} < \overrightarrow{Total_Grav}$, therefore the countermeasure *block_conn* can be selected.

5 Conclusion

In this paper, a risk assessment model is presented to improve the reaction in the detection intrusion system. This model is used to assess and quantify the risk of attacks and countermeasures. Therefore, a clear procedure can be used to judge if a reaction is necessary, and which countermeasure must be chosen. Using the risk analysis approach, the reaction against an attack is more efficient, and harmful countermeasures may be avoided. There are several perspectives to this work: First, the audit clusters are so far considered as inputs; we will propose the evaluation method of each cluster. Second, the functions used can be more precise by using advanced ponderable mean functions, and use more sophisticated taxonomies.

Acknowledgments

This work was partially supported by the European CELTIC RED project.

References

1. R. BACE: Intrusion Detection. McMillan Technical Publishing, 2000.
2. F. Cuppens: Managing Alerts in a Multi-Intrusion Detection Environment . 17th Annual Computer Security Applications Conference New-Orleans, Decembre 2001.
3. F. Cuppens, F. Autrel, A. Mieke et S. Benferhat: Correlation in an intrusion detection process . Internet Security Communication Workshop (SECI'02), Tunis, Septembre 2002.
4. R. Lippmann: Using Key String and Neural Networks to Reduce False Alarms and Detect New Attacks with Sniffer-Based Intrusion Detection Systems. Proceedings of the Second International Workshop on the Recent Advances in Intrusion Detection (RAID'99), Purdue, USA, October 1999.
5. M. Huang: A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis. Louvain-La-Neuve, Belgium,1998.
6. B. Morin, H. Debar: Correlation of Intrusion Symptoms: an Application of Chronicles. Proceedings of the Sixth International Symposium on the Recent Advances in Intrusion Detection (RAID'02), Pittsburg, USA, September 2003.
7. F. Cuppens, F. Autrel, A. Mieke et S. Benferhat: Recognizing Malicious Intention in an Intrusion Detection Process . Second International Conference on Hybrid Intelligent Systems, Santiago, Chili, Decembre 2002.
8. Peng Ning, Yun Cui, Douglas S. Reeves: Constructing attack scenarios through correlation of intrusion alerts. ACM Conference on Computer and Communications Security 2002.
9. F. Cuppens, F. Autrel, Y. Bouzida, J. Garcia, S. Gombault, and T. Sans: Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework . Annals of Telecommunications. Vol. 61, no. 1-2. January-February 2006.
10. H. Debar, Y. Thomas, N. Boulahia-Cuppens, F. Cuppens: Using contextual security policies for threat response . Third GI International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA). Germany. July 2006.
11. www.ssi.gouv.fr/fr/confiance/documents/methodes/ebiosv2-memento-2004-02-04.pdf
12. www.cases.public.lu/publications/recherche/these_jph/NMA-JPH_MISC27.pdf
13. http://i-a.ch/docs/CLUSIF_Marion.pdf
14. www.clusif.asso.fr/fr/production/ouvrages/type.asp?id=METHODES
15. F. Cuppens, R. Ortalo: LAMBDA: A Language to Model a Database for Detection of Attacks. Third International Workshop on Recent Advances in Intrusion Detection (RAID'2000). Toulouse, October 2000.
16. K. Kendall: A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Massachusetts Institute of Technology, June 1999.
17. J. Mirkivich, J. Martin, P. Reiher: Towards a Taxonomy of Intrusion Detection Systems and Attacks. Project IST-1999-11583, MAFTIA deliverable D3, September 2001.
18. F. Autrel, F. Cuppens: CRIM : un module de corrlation d'alertes et de raction aux attaques . Annals of Telecommunications. Vol. 61, no. 9-10. September-October 2006.