Bayesian Attack Model for Dynamic Risk Assessment

François-Xavier Aguessy[1,2], Olivier Bettan[1], Gregory Blanc[2], Vania Conan[1], and Hervé Debar[2]
francois-xavier.aguessy@telecom-sudparis.eu

[1] Thales Group, 4 avenue des Louvresses, 92622 Gennevilliers, France
[2] SAMOVAR, Télécom SudParis, Université Paris Saclay, 9 rue Charles Fourier, 91011 Evry, France

Abstract. Because of the threat of advanced multi-step attacks, it is often difficult for security operators to completely cover all vulnerabilities when deploying remediations. Deploying sensors to monitor attacks exploiting residual vulnerabilities is not sufficient and new tools are needed to assess the risk associated to the security events produced by these sensors. Although attack graphs were proposed to represent known multi-step attacks occurring in an information system, they are not directly suited for dynamic risk assessment. In this paper, we present the Bayesian Attack Model (BAM), a Bayesian network-based extension to topological attack graphs, capable of handling topological cycles, making it fit for any information system. Evaluation is performed on realistic topologies to study the sensitivity of its probabilistic parameters.

Keywords: Bayesian Attack Model, dynamic risk assessment, topological attack graph, cycle management, Bayesian network

1 Introduction

Managing the security of Information Systems (IS) is increasingly complex, due to the numerous security mechanisms that are implemented, and the significant amount of dynamic data produced by security enforcement points. In critical environments, security operators generally know most of the vulnerabilities of their IS thanks to regular vulnerability scans. Unfortunately, many vulnerabilities are not patched, either because patching may disrupt critical services, or because they are not a priority for system administrators. As a second line of defence, security operators deploy sensors (*e.g.*, Host or Network Intrusion Detection Systems) generating alerts when an attacker attempts to exploit such vulnerabilities. As these security events are produced, operators need to evaluate the risk brought by ongoing attacks in their system, to respond appropriately: this process is called dynamic risk assessment (DRA) [14].

The most impacting attacks are composed of several successive exploitation steps. Several models have been proposed to formalize such multi-steps attacks.

An attack graph is a model regrouping all the paths an attacker may follow in an information system. It has been first introduced by Phillips and Swiler in [18]. A study of the state of the art about attack graphs compiled from early literature on the subject has been carried out by Lippmann and Ingols [12], while a more recent one was made available by Kordy *et al.* [10]. Topological attack graphs are based on directed graphs. Their nodes are topological assets (hosts, IP addresses, etc.) and their edges represent possible attack steps between such nodes [8]. Attack graphs are generated with attack graph engines. There are three main attack graph engines: (1) *MulVAL*, the Multi-host, Multi-stage Vulnerability Analysis Language tool created by Ou *et al.* [15], (2) the Topological Vulnerability Analysis tool (*TVA*) presented by Jajodia *et al.* in [8,9] (commercialized under the name *Cauldron*) and (3) Artz's *NetSPA* [2]. Attack graphs are attractive because they leverage readily available information (vulnerability scans and network topology). However, they are not adapted for ongoing attacks, because they can not represent the progression of an attacker nor be triggered by alerts. Thus, they must be enriched to provide the functionalities needed to perform Dynamic Risk Assessment, for example using Bayesian networks.

A Bayesian network is a probabilistic graphical model introduced by Judea Pearl [16]. It is based on a Directed Acyclic Graph, where nodes represent random variables, and edges represent probabilistic dependencies between variables [3]. For discrete random variables, these dependencies can be specified using a Conditional Probability Table associated with each child node. Bayesian networks are particularly interesting for computing inference, *i.e.* calculating the probability of each state of all nodes of the network, given some evidences, *i.e.* nodes that have been set to a specific state. Inference can be done efficiently using the algorithm of Lauritzen and Spiegelhalter [11]. A Bayesian attack graph, introduced by Liu and Man in [13] is an extension of an attack graph based on a Bayesian network, constituted of nodes representing a host in a specific system state (a true state means that the host is compromised) and edges representing possible exploits that can be instantiated from a source host to a target host. The major concern of building such a Bayesian network from an attack graph is due to the structure of a Bayesian network that must be acyclic, while attack graphs almost always contain cycles. To avoid cycles, Liu and Man assume that an attacker will never backtrack once reaching a compromised state, but do not detail how such assumption is used to build the model. In [6], Frigault and Wang use Bayesian inference in Bayesian Attack Graphs to calculate security metrics in an information system. Xie *et al.* present in [21] a Bayesian network used to model the uncertainty of occurring attacks. The Bayesian attack graph is enhanced with three new properties: separation of the types of uncertainty, automatic computation of its parameters and insensitivity to perturbation in the parameters choice. This model also adds nodes dedicated to dynamic security modelling: an *attack action node* models whether or not an attacker action has been performed, a *local observation node* models the inaccuracy of observations.

In this paper, we propose a new model combining attack graphs and Bayesian networks for DRA. It is built from the knowledge security operators have about their IS: network topology, known vulnerabilities and detection sensors. Then, we change the states of the sensor nodes according to the security events received. This model is capable of representing the attacks that may occur (vulnerabilities) and the ones ongoing (alerts). It outputs probabilities that attacks have succeeded and that assets of the IS may have been compromised. With respect to the current state of the art, our contributions are twofold. First, we provide an explicit model and process for handling cycles. This process is supported by a clear definition of a set of model parameters. The sensitivity of the model toward these parameters is studied in the validation. Second, we provide a significant performance improvement in terms of number of nodes and vulnerabilities over the existing state of the art. While classic Bayesian attack graph models are usually demonstrated over a few nodes and vulnerabilities, we show that our model can be realistically computed at the scale of an enterprise IS.

This paper is organised as follows: in Section 2, we formally define the structure and the conditional probability tables of our Bayesian Attack Model built from a topological attack graph. Section 3 validates the results of the Bayesian Attack Model on a realistic use case and analyses its sensitivity toward the probabilistic parameters. Section 4 compares our work with the related work, before concluding and presenting future work, in Section 5.

2 The Bayesian Attack Model

Given the advantages brought by Bayesian Attack Graphs, they provide a strong foundation for dynamic security modelling. Our proposal extends Bayesian Networks to be used for DRA with real-scale IS.

The Bayesian Attack Model (BAM) described all along this section is built from a Topological Attack Graph, which is described in section 2.1, and a set of detection alerts. The BAM is composed of submodels called Bayesian Attack Trees (BAT). BAT and BAM are described in section 2.3. Each BAT is composed of a sequence of attack steps, typed nodes linked together. They are described in section 2.2. The probabilistic relations between nodes of a BAT are described in conditional probability tables whose content is detailed in section 2.4.

2.1 Topological Attack Graph

The BAM is built from a topological attack graph.

Definition 1. *A **topological attack graph** is a directed graph* TAG(TN,AS)*:*

- $TN = \{\mathrm{TN}_i, i \in \{1..N\}\}$ *is a set of* N ***topological nodes**: the assets of an information system,*
- AS *is a set of **attack steps**, the edges that represent the fact that an attack allows the attacker to move from the parent topological node to the child topological node.*
  - *Each attack step has a **type** of attack, describing how the attacker can move between nodes (exploitation of a vulnerability, credential theft, etc.).*

- *Depending on the type of attack, each attack step is associated with a set of **conditions*** [c]*.*
- *Some attack steps are associated with a **sensor** that may raise an alert indicating that this attack has been detected.*

A TAG can be generated with an attack graph engine such as MulVAL [15] or TVA [9]. Topological nodes represent, for example, an IP address or a computer cluster. Attack steps are, for example, the exploitation of a vulnerability.

Definition 2. *A **condition*** c *is a fact that needs to be verified, for an attack step to be possible. It is associated with a probability of success* P(c)*.*

The condition fact is, for example, "a vulnerability is exploited on the destination host". For such conditions, in our experiments, we use an approximation of the probability of successful exploitation using information coming from the Exploitability Metrics of the Common Vulnerability Scoring System (CVSS) [5]. It is deduced from (1) the Attack Complexity (AC), (2) Privileges Required (PR), (3) and User Interaction (UI) values, as well as the Attack Vector (AV), which is taken into account when constructing the topological attack graph.

Definition 3. *A **sensor*** s *of an attack step is an oracle issuing an alert when the attack step has been detected. It is associated with a false negative and a false positive rates.*

A sensor represents, for example, an Intrusion Detection System, a System Event Management, or a human report.

Grouping attack steps In topological attack graphs, there may exist many attack steps between two topological nodes. Attack steps can be of different types, depending on the attack (*cf.* Definition 1). Generally, there are very few possible types of attack steps (*e.g.*, the remote exploitation of a vulnerability on a server). In order to reduce the size of the model, while preserving information, we group all attack steps of the same type between two topological nodes into a single vertex with (1) a new condition: a multivariable boolean function (usually, an OR) of all conditions applying to the grouped attack steps; (2) an attached sensor node activated only when the boolean function of grouped sensors is true.

When several conditions c_i of an attack step as are grouped in one condition c, we define the probability of successful exploitation associated with this new condition. For example, when grouping several conditions c_i "a vulnerability is exploited on the destination host" into one new condition c "at least one vulnerability of the list is exploited on the destination host", we assume that the exploitation of each vulnerability is independent, to compute its probability of exploitation $P(c)$. This is an acceptable approximation since we consider all the existing vulnerabilities between two topological nodes. Thus, the probability of exploitation $P(c)$ becomes:

$$P(c) = P(\bigvee_{i \in \{\text{vulnerabilities of } as\}} c_i) = 1 - \prod_{i \in \{\text{vulnerabilities of } as\}} (1 - P(c_i))$$

Breaking cycles in topological attack graphs A TAG is a model defined globally for a system, containing all potential attacks that can happen. It thus almost always contains cycles, especially inside local networks in which any host can attack any other one. For example, a host tn_1 may be able to attack another host tn_2 that can also attack tn_1 (directly or in several steps). A common assumption to break cycles in attack graphs is that an attacker will not backtrack, *i.e.*, come back on a node he has already successfully exploited. This is reasonable because backtracking does not bring new information about attack paths. It has been properly justified by Ammann *et al.* in [1] and by Liu and Man in [13]. However, the solutions of the state of the art for Bayesian modelling of an attack graph such as the ones of Liu and Man [13] and Poolsappasit *et al.* [19] use this assumption to delete arbitrary possible attack steps. In reality, it is impossible to know a priori which path the attacker can choose. Deleting paths in the Bayesian model thus suppresses actually possible attacker actions. The only way to break cycles, while keeping all possible paths, is to enumerate all paths, starting from every possible attack source, keeping in the nodes a memory of the path of the attacker. So, using this memory, we build an acyclic TAG by ensuring that the paths do not backtrack on already exploited nodes. For example, a node $tn_1tn_2tn_3$ means that the attacker controls the node tn_3, having first compromised tn_1, then tn_2, finally tn_3. Unfortunately, this cycle breaking process causes a combinatorial explosion in the number of nodes of the model. We discuss in Section 2.6 how we mitigate such limitation.

2.2 Representation of an attack step in BAM

An attack step in the TAG is an edge which is associated with several conditions and can be related to a detection sensor. In the BAM, we detail the attack steps, the conditions, and sensors as nodes, in order to model the probabilistic interactions between such elements, using the nodes detailed below. Each node represents a boolean random variable with two mutually exclusive states.

Definition 4. *A* ***Bayesian topological node*** $\mathrm{btn}\ (\mathrm{tn}_1, \cdots, \mathrm{tn}_n)$*, with* $\forall i, \mathrm{tn}_i \in \mathrm{TN}$ *(cf. Def. 1), is a node of the BAM representing the random variable describing the state of compromise of* tn_n *using the path of the topological attack graph* $\mathrm{tn}_1 \to \cdots \to \mathrm{tn}_n$ *(i.e.,* Compromised *or* NotCompromised*).*

Definition 5. *A* ***Bayesian attack step node*** $\mathrm{basn(as)}$*, with* $\mathrm{as} \in \mathrm{AS}$ *(cf. Def. 1), is a node of the BAM representing the random variable describing the attack success of* as *(i.e.,* Succeeded *or* Failed*).*

Definition 6. *A* ***Bayesian condition node*** $\mathrm{bcn(c)}$*, with* c *a condition (cf. Def. 2), is a node of the BAM representing the random variable describing that the condition* c *is fulfilled (i.e.,* Succeeded *or* Failed*).*

Definition 7. *A* ***Bayesian sensor node*** $\mathrm{bsen(s)}$*, with* s *a sensor (cf. Def. 3), is a node of the BAM representing the random variable describing the state of the sensor* s *(i.e.,* Alert *or* NoAlert*).*

These nodes are linked with edges, indicating that the child node has a conditional dependency to the state of its parents. For example, a Bayesian attack step node has a dependency toward its condition(s) and the topological node from which it may be accomplished. Thus it is the child of the nodes representing the conditions and the topological node. In the same way, a Bayesian sensor node is the child of a Bayesian attack step, and a Bayesian topological node is the child of a Bayesian attack step.

Definition 8. *A **Bayesian edge*** e*, is a link from a parent node to a child node that represents a conditional dependency of the child toward its parent.*

Appendix A Figure 3 shows the details of the representation of an attack step from tn_n (source) to tn_{n+1} (target).

2.3 Complete Bayesian Attack Model

Bayesian Attack Tree and Global Model The complete BAM is composed of a family of Bayesian Attack Trees (BAT), as defined below, each one issued from one attack source.

Definition 9. *A **Bayesian Attack Tree** is a Bayesian network represented by* BAT(AS, DAG,P) *where:*

- *AS is a special Bayesian Topological Node, the attack source of this BAT.*
- DAG(BN,E) *is a polytree structure, constituted of*
  - *BN, the Bayesian nodes* BN=[btn],[basn],[bcn],[bsen] *(cf. Defs. 4-7)*
  - *E, the set of edges* $E = \{e\}$ *representing a conditional dependency between the nodes (cf. Def. 8).*
- P *is a set of local probability distributions, associated with each node of* DAG. *As all nodes are discrete random variables, the local probability distributions can be specified within a Conditional Probability Table.*

To build the whole structure of one BAT of the BAM, we start from a potential attack source of the acyclic TAG. It is the Attack Source and the root of the BAT. Then, we recursively add the attack steps contained in the acyclic TAG with the nodes described in Subsection 2.2. To avoid cycles, each attack step is added, as soon as its target has not been already compromised during the currently followed path. This can be achieved thanks to the memory of past topological nodes in Bayesian topological nodes. This building process also ensures that the graph structure of each BAT is a polytree: a Directed Acyclic Graph for which there are no undirected cycles either. This allows to use very efficient exact inference algorithms in the Bayesian network such as Pearl's algorithm [17].

The complete BAM is constituted of the set of all BATs. As we consider that each topological node may be a source of attack, the BAM contains exactly N BAT (*i.e.*, the number of topological nodes in the TAG).

Definition 10. *The **Bayesian Attack Model*** BAM($\{BAT_i\}$)*, is a family of* N *Bayesian networks where, for all* i *in* $\{1..N\}, \mathrm{BAT}_i$ *is a BAT, whose attack source is node* i *in the topological attack graph.*

Figure 1 summarises the global architecture of the BAM. In this example, it is built from a TAG containing 3 nodes and thus is composed of 3 BATs.

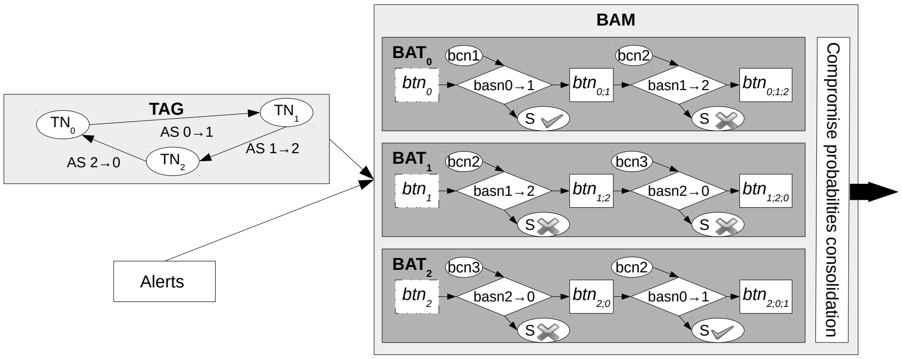


Fig. 1. Bayesian Attack Model Architecture

Consolidation of probabilities As each Bayesian topological node contains the history of the attack that can lead to this node, many Bayesian topological nodes can represent the same topological node in several BATs, when the attacker used a different path to reach it (e.g. the Bayesian topological node $tn_1 \rightarrow tn_2 \rightarrow tn_4$ is different from $tn_2 \rightarrow tn_3 \rightarrow tn_4$, even if the attacker has the control of the same topological node tn_4 at the end.).

In the complete BAM, we thus have many Bayesian topological nodes representing the same asset of the IS. However, what most interests a security operator is the attacks that are the most likely to compromise his assets. Thus, as output of the consolidation of probabilities, we assign to a physical asset a probability of compromise that is the maximum of the probabilities of Bayesian topological nodes targeting the same asset.

$$P(TN_k) = \max_{i \in \{1..N\}} P_{BAT_i}(TN_k)) = \max_{i \in \{1..N\}} (\max_{\{\forall TN_1..TN_{k-1}\}} P_{BAT_i}(TN_1, ..., TN_k))$$

2.4 Conditional Probability Tables

We now specify the local probability distribution associated with each node, describing the probability dependencies of a node toward his parents. As the nodes are discrete random variables, we can describe the probability dependencies using conditional probability tables (CPT).

A Bayesian Topological node has one parent for each type of attack that can be used to compromise it. Its probability table represents a *noisy-OR*. At least one *succeeded* attack step is needed to *compromise* this node. Even if no known attack step has *succeeded*, there is still a little chance that an attack of this topological node may be an unknown one (e.g. a 0-day). We denote it by *pua*. Such a CPT is described in Appendix B Table 1.

An attack step node has two types of parents: (1) one Bayesian topological node, the source of the attack, required to perform the attack; (2) one or more Bayesian condition nodes. Depending on the type of attack modelled, the condition nodes may not exist for the attack node. The $probabilityNewAttackStep$ parameter represents the fact that an attacker may have reached his objective. Even if he has compromised the topological node and conditions are verified, it is not certain that he will attempt to propagate through the execution of a new exploit. We describe in Appendix B Table 2 the CPT of a Bayesian attack node, for the exploitation of a vulnerability.

A Sensor node has only one parent, the attack node related to the sensor. Its CPT thus contains only two values and their complement representing the $falsePositive$ and $falseNegative$ rates attached to the sensor. The CPT of a Bayesian sensor node is described in Appendix B Table 3.

The attack source of a BAT is a Bayesian topological node without parents. As such, it does not have a complete CPT, but only a prior probability value and its complementary. This *attackSourceProbability* parameter represents the *a priori* probability of having an attack issued from this node. It thus has to be set by the operators, knowing the risk that an attack starts from a topological node. It can be deduced from a risk evaluation methodology (*e.g.*, ISO 27005 [7]). In a typical system, a high probability can be set to the Internet (*e.g.*, 0.7), a medium one to servers in a demilitarised zone (internal subnetwork protected by a firewall exposing external-facing services on the Internet) (*e.g.*, 0.4), and a small one for production database servers (*e.g.*, 0.1).

The Attack conditions also do not have any parents. Their probability is the probability of successful exploitation $P(c)$ associated with the condition. It highly depends on the type of condition modelled by this node. For example, for a condition describing the successful exploitation of at least one vulnerability of a list on a host. The estimation of this probability of successful exploitation follows the process detailed in Section 2.1, with values for each vulnerability, coming from the Exploitability Metrics of the CVSS, as explained in Section 2.1.

2.5 Bayesian Attack Model usage

We build our Bayesian Attack Model from the knowledge that the security operators have about the information system: network topology, known vulnerabilities and deployed detection sensors. Then, we change the state of the Bayesian sensor or topological nodes according to the security events received from the sensors:

Sensor Nodes: If the sensor of an attack step exists and is deployed in the network, as long as it has not issued any alert, all related sensor nodes of the BAM (that may appear in several BATs) are set to the *no alert* state. When the sensor raises an alert corresponding to this attack step, the Bayesian sensor nodes are set to the *alert* state. If the sensor also gives an alert confidence probability, it is possible to set the state *alert* to this probability.

Topological Nodes: As soon as a compromise information is known for a topological node, all related Bayesian topological nodes are set to the corresponding state. For example, if a Host Intrusion Detection System (HIDS) says that a host is healthy, the related Bayesian topological nodes in all BATs are set to the *not compromised* state. Conversely, if the HIDS says that a host is compromised, the related Bayesian topological nodes are set to the *compromised* state. If the HIDS also gives a compromise probability, the *compromised* state is set to this probability.

The Bayesian nodes for which there is no compromise information (no deployed sensor, Bayesian attack step nodes and Bayesian condition nodes) are not set in any state and their probability are updated by the Bayesian inference.

Each time the BAM changes state (when we fix nodes in a different state), we use a Bayesian network belief propagation algorithm (Lauritzen or Pearl's inference algorithm) to update the probabilities of each state at all the nodes. Then, for each topological node of the topological attack graph, the maximum probability of the state *compromised* of all related Bayesian topological nodes, provides security operators with the probability of the asset being compromised, as described in Subsection 2.3.

2.6 Model size limitation

Use of a nbSteps parameter to prevent performance issues: The main limitation when implementing this model is the combinatorial explosion of the number of nodes, due to the redundancy introduced by the cycle breaking process. In order to improve the performance and prevent this combinatorial explosion, we limit the number of successive attack steps added to each BAT, according to a *nbSteps* parameter. Thus, we can contain the number of nodes to process in the BAM, as detailed in Section 3.1.

Impact of the nbSteps parameter on the outputs of the BAM: Thanks to the redundancy of the model, and as each topological node is an attack source of a BAT, if some attack steps are discarded in a BAT, they will be in another BAT, closer to the BAT attack source. The probabilities of Bayesian topological nodes in a BAT represent the probability of the attacker exploiting this node *starting from the attack source*. As long as no attack has been detected on a path, the probability of a node compromise decreases rapidly as a function of the length of the path between the attack source and the node. During initial probability computation, the probabilities of nodes far from the attack sources are very low. These probabilities are below the maximum used during the probability consolidation detailed in Section 2.3 and do not have any effect on final compromise probabilities. In that case, the *nbSteps* parameter has no impact on final results.

The key limitation this parameter introduces is when attacks start being detected and introduced in a path. More precisely, the limitation arises when more than two detections are injected in the model. For example, to compute the combined impact of two detections relative to each other, they need to appear in the same BAT. The maximum compromise probability of the topological node related to the first detection will be in the BAT in which it is the attack source. If the second detection is attached to a node that is more than *nbSteps* away (*i.e.*, separated with more than $nbSteps - 1$ missed detections), it will not be in the same BAT and these two attacks will be taken into account separately. This will prevent the increase of probabilities of the nodes between the two detections. Detections may be separated by other nodes without detections for two reasons: if there are not enough sensors or if there are false negatives, both undesired cases. As a summary, the only case when the impact of the limitation of the BAT depth to *nbSteps* is significant is when there are more missed detections than $nbSteps - 1$ between two successive detections for the same attack. These assumptions are validated by the experimental validation of Section 3.2.

3 Validation

3.1 Complexity evaluation

The main computation done on each Bayesian Attack Tree of the BAM is the execution of the belief propagation algorithm (probability inference), computing the probability of all nodes, according to evidences, nodes set to a specific state. The complexity of the inference in a Bayesian network is directly linked to the number of nodes and structure of the network. We estimate the number of nodes M of a BAT, depending on N, the number of topological nodes in the attack graph, and k the maximum number of *consolidated* attack steps between two topological nodes in the topological attack graph (*i.e.*, the maximum number of different types of attacks). M is also strongly depending on the existence of attack steps between the topological nodes. An attack step needs the existence of at least a vulnerability and of an authorised network access, which depends totally on the monitored information system. Thus for this complexity evaluation, we consider the worst case: there are k attack steps between each pair of topological nodes. For each attack step, we add ≈ 4 nodes to the BAM (sometimes few more, according to the number of conditions). Thus, in the worst case, for each BAT, starting from an attack source, the number of nodes to add is

$$M \sim 4 \times k \times (N \times \cdots \times (N - nbSteps - 1)) = 4 \times k \times \frac{N!}{(N - nbSteps)!} = \mathcal{O}(N^{nbSteps})$$

The degree of the polynomial curves of the number of nodes in the BAM increases with the parameter *nbSteps*. However, even if the number of nodes in each BAT is high, the Bayesian inference can be done efficiently. Indeed, as the structure is a *polytree*, some efficient inference algorithms can be used. For example, Pearl's belief propagation algorithm is linear in the number of nodes [17].

Thus, for each BAT, in the worst case, the complexity of the construction and probability inference $\mathcal{C}(BAT)$ is $\mathcal{C}(BAT) = \mathcal{O}(N^{nbSteps})$. Finally, for the whole

BAM, as there are at most N attack sources, in the worst case, the complexity of the inference in the whole model $\mathcal{C}(BAM)$ is

$$\mathcal{C}(BAM) = N.\mathcal{C}(BAT) = \mathcal{O}(N^{nbSteps+1})$$

The calculations on each BAT are independent. So, they may be easily done in parallel, which gives in practice, $\mathcal{C}(BAM) = \mathcal{O}(N^{nbSteps})$ with N processors.

3.2 Experimental use-case-based validation

We will first present a use-case and the scenarios that have been chosen to do the experimental validation of the BAM, then discuss the results obtained.

Validation scenarios In order to validate the accuracy of the results, while keeping the scenarios simple for explanations, we implemented a real infrastructure of 11 virtual machines, for a total of a hundred vulnerabilities. A host (that will be called host A, thereafter) can be attacked from the Internet, and can attack the other hosts G to J of its subnetwork. The latter hosts can attack hosts A, C and D. This network topology is representative of a real information system, where an ingress firewall (host K) protects the LAN (E to J), and where publicly accessible servers are put in a demilitarised zone (A to D). The topological attack graph used to populate the BAM has been generated from a report of the vulnerability scanner Nessus, done on this infrastructure.

We apply 6 attack scenarios on this network topology, as summarised in Appendix C Table 4. The attack is carried out through three attack steps. In the first scenario, no step is detected; it represents the basic risk of the IT system. In scenarios 2 to 4, steps are detected and alerts are generated. Scenarios 5 and 6 represent detection anomalies. These scenarios represent the dynamic evolution of a system with different possible situations:

- Scenarios 1, 2, 3, then 4: Normal evolution of an attack during the time.
- Scenarios 1, 2, then 5: Evolution of an attack in which an attack step cannot be detected (no sensor for this step).
- Scenarios 1, 2, then 6: Evolution of an attack in which an attack step has not been detected while there was a sensor for this step.

We assume in these scenarios that the alerts given by the sensors are binary (*alert*, *noalert*), *i.e.*, we do not have alert confidence.

Parameters default values This use-case represents a typical critical IS. It is managed by a security operator who often uses a vulnerability scanner. Most vulnerabilities are known, but there is still a chance (*e.g.*, 0.1%) that a very motivated attacker knows a non-public vulnerability. As the system contains known unpatched vulnerabilities, sensors are deployed to raise an alert when one of the vulnerabilities is exploited. These sensors have a medium chance (*e.g.*, 5%) to raise false positives, when an attack do not succeed while being detected.

However, for the vulnerabilities for which a detection sensor is deployed, the probability of having a false negative is lower (*e.g.*, 1%). The operator knows that his system is quite well protected, so it is very unlikely that an attack occurs with more than 2 undetected steps (*e.g.*, *nbSteps* can be set to 3). Most attacks may come from the Internet (*e.g.*, probability of the Internet being a source of attack of 70%), even if internal hosts may also be a new source of attacks (undetected phishing, malicious employee, *etc.*) with a lower probability (*e.g.*, 10%)). Finally, as valuable machines are not deeply protected (they can be reached in 3 steps from the Internet), the probability that the attacker propagates through a new attack step is medium (*e.g.*, 30%). After an attack, he may have already found what he was looking for. Default values of the parameters used for this use case are summarised in Appendix D Table 5.

Results and analysis The Bayesian Attack Model was implemented in Java, using the SMILE Bayesian Network library [4]. The results of the compromise probabilities of each topological node calculated by the BAM, for each scenario, are shown in Appendix E Figure 4. The first scenario is the basic risk. The only host that has a *medium* risk is the Internet. The other hosts have a *not-significant* risk. In the scenarios 2, 3 and 4, the sensors corresponding to the 3 steps attack are set progressively. Each new sensor set as *detected* confirms the attack that is currently happening and increases the compromise probability of the previous and future states. For example, in scenario 4, the Internet, and the 3 victim hosts are in the *high*-risk zone. In scenario 5, and scenario 6, when there is a missing detection or a false negative / false positive, the probabilities of an ongoing attack are lower, but higher than the basic risk, and the probabilities of scenario 2, that should precede this state. So, a security operator may investigate the appropriate machines to confirm or disprove the attack.

Parameter sensitivity analysis Several parameters can be customised in the BAM (*cf.* Section 3.2). We summarise in Appendix F, Table 6 the results of the sensitivity analysis of these parameters with the range of variation that we find appropriate for the given parameters (range of values that may occur in real-life). The false positives and negatives rates vary from 0 to 30%, because beyond, their values are meaning-less (*e.g.*, a vulnerability signature with more 30% false positive is useless). The number of successive steps varies from 1 (its minimum) to 4 (the maximum possible number of successive attack steps for this use-case). Sources probabilities vary from 0 to 1, as, according to the context, all values may be possible. The probability of having an exploitation of an unknown vulnerability is low (15% is a far upper bound). The probability of the attacker making a new attack step is difficult to estimate. We thus need to study the impact of this parameter on its whole possible variation interval (0 to 100%).

The most interesting result of this analysis is the *ranking influence* describing the impact of the variation of a parameter on the rank of topological nodes probabilities (on the whole parameter variation range, for the 6 scenarios). This rank will determine the priorities of security operators in their IS. The *probability* *influence* describes the effect of the variation of the parameters on the absolute value of the topological nodes probability. The only parameter that has an impact on the rank of the topological nodes probabilities is $probabilityOtherHosts$. However, this parameter can be estimated quite accurately with a risk analysis methodology, which gives the security risk of each topological node, according to its position in the information system. All other parameters do not have any effect on the ranking on their whole variation range, which is a comforting result. Four parameters have a medium impact on the absolute value of the compromise probabilities of topological nodes. With a medium uncertainty on such parameters (*e.g.*, 0.2), the variation of the absolute value of the probabilities is medium (*e.g.*, up to 0.2). Other parameters have a low impact on absolute values of probabilities. With a medium uncertainty on such parameters (*e.g.*, 0.2), the variation of the absolute value of the probabilities is low (*e.g.*, up to 0.02). So, absolute value of probabilities may be a little impacted by uncertainty on parameters, but rank is mostly not impacted by the variation of the parameters.

3.3 Performance evaluation

In order to dynamically assess the risk of a system, the BAM has to be evaluated each time a correlated alert, or a set of correlated alerts is received: the sensors and topological nodes are set in their new states, then the probabilities are updated. The duration of such a process needs to be quite fast (around 1 minute is good), for the operator to properly understand the risk in operational time. We simulate random network topologies with different parameters (number of hosts, subnets, vulnerabilities and network services and connectivity between subnets) to evaluate the performance of the BAM. We generate the TAGs related to the topologies. Then, we generate random attack scenarios with seven successive attack steps. Finally, we evaluate the BAM on the different scenarios.

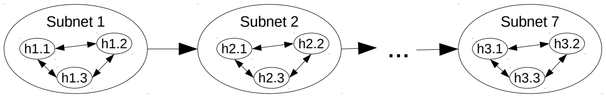


Fig. 2. Network topology for simulations

We generate random topologies, as depicted in Figure 2, containing from 1 to 70 hosts, in 7 subnets. These topologies are representative of a real network in which defense in depth is implemented: all the hosts of a subnet have access to all the hosts of a deeper subnet. In each subnet, all accesses between hosts are authorized. Each host has 30 random vulnerabilities for a maximum total of around 2000 vulnerabilities. The results of the duration in seconds of the BAM generation and the inference after the evaluation of one scenario of 7 successive attack steps, on these topologies, is displayed in Appendix G Figure 5. The parameters of the BAM are in the default values detailed in Section 5. This simulation shows that for medium-sized topologies (up to 70 hosts) the duration of the Bayesian Attack Model generation and of the inference remains acceptable ($<$ 1 minute 30 seconds) on a laptop-class computer.

Even if the number of topological nodes of these simulations is limited (70 hosts), it could be extended to much bigger IS, by clustering together identical templates of servers or of client machines in one topological node, as they possess the same vulnerabilities and authorised accesses and thus behave in a similar way in the BAM. Even with 60 assets in the topological attack graph with, for example, 10 templates of client machines, 30 of network servers, and 20 of business application servers, it is possible to model a usual big-sized IS.

3.4 Accuracy evaluation

To evaluate the accuracy of the results (*i.e.*, how close the probabilities are to the truth), we simulate attack scenarios on the random topologies presented in Section 3.3 and compare the theoretical results with the outputs of the BAM. The results are shown in Appendix G Figure 6. We compare the theoretical results known in the scenarios with the results of the BAM. In each scenario, we know the nodes that are compromised and healthy, *i.e.*, nodes with a theoretical probability of respectively 1 and 0. Then, we assess if the BAM probabilities of compromised nodes are close to 1, and if the BAM probabilities of healthy nodes are close to 0. The plot shows the maximum errors (in terms of distance to the theoretical values 1 and 0) of compromised and healthy nodes. This figure shows a large free space between the errors on compromised hosts and the errors on healthy hosts. This means that if there are no false-positives nor false-negatives in the detection inputs of the BAM, it allows to distinguish exactly healthy and compromised hosts, for example with a boundary at the probability of 0.5. So, there are no false negatives nor false positives introduced by the BAM. The graphical difference of the results between the values for a low number of hosts and high number of hosts is probably due to the random attack scenarios that may be shorter when there are not enough hosts.

4 Related Work

Many people proposed enhancements to improve attack graphs with Bayesian networks, to use them for dynamic risk assessment [20,13,21]. However, they do not describe how they manage cycles that are inherent to attack graphs. In [21], Xie *et al.* present an extension of MulVAL attack graphs using Bayesian networks, but they do not mention how to manage the cycle problem, while MulVAL attack graphs frequently contain cycles. In the same way, in [6], Frigault and Wang do not mention how they deal with the cycle problem constructing Bayesian attack graphs. In [13], Liu and Man assert that to delete cycles, they assume that an attacker will never backtrack. The same assumption is used by Poolsappasit *et al.* in [19]. However, they both do not present how they deal with this assumption to keep all possible paths in the graph, while deleting cycles. We propose here a novel model that explodes cycles in the building process, keeping all possible paths while deleting the cycles, to compute the Bayesian inference.

The Bayesian model presented by Xie *et al.* in [21] is based on logical attack graphs. It is thus very verbose and can be huge for real information systems. In [13], Liu and Man's model is a topological graph, in which are added violation states. It is thus quite compact, but does not detail the attacks, their conditions and, mainly, the sensors that can change state. Thus, the only observations that can be set on this model are observations on topological nodes. The model we present is a topological model. So, it is much more compact than those based on logical attack graphs. However, it contains the logical conditions necessary to carry out the attacks, in order to keep all information important to model attacks, and add sensor nodes that can be activated with detections. Moreover, we also add several improvements (attack nodes gathering, polytree structure of BAT, etc.) that either reduce the size of the graph structure or improve the performance of the inference. We thus constrain the size of the graph in which we do Bayesian inference, while conserving all paths by linearising cycles.

The experimental validation we did on the Bayesian Attack Model is on a real topology of a complexity similar or superior to what was done in the literature and on simulated topologies that are far bigger than the state of the art. For example, Xie *et al.* assess their model on a topology of 3 hosts and 3 vulnerabilities [21], Liu and Man on a topology of 4 hosts and 8 vulnerabilities [13]. The real world examples used by Frigault and Wang in [6] contain at most 8 vulnerabilities on 4 hosts. The test network used by Poolsappasit *et al.* in [19] contains 8 hosts in 2 subnets, but with only 13 vulnerabilities. Thanks to our polytree model, we successfully run our Bayesian Attack Model efficiently on simulated topologies with up to 70 hosts for a total of more than 2000 vulnerabilities.

5 Conclusion and Future Work

We present in this paper a new Bayesian Attack Model (BAM), representing all the possible attacks in an information system. This model enables dynamic risk assessment. It is built from a topological attack graph, using already available information. Sensor nodes can be activated by dynamic security events to update the compromise probabilities of topological assets, which rank the risk level of ongoing attacks. This model handles the cycles that are inherent to attack graphs and thus is applicable to any information system, with multiple potential attack sources. The cycle breaking process significantly increases the number of nodes in the model, but thanks to the polytree structure of the Bayesian networks we build, the inference remains efficient, for medium information systems. In order to be able to use the Bayesian Attack Model for bigger information systems, future work will investigate how the usage of a hierarchical topological attack graph can be appropriate to build the Bayesian Attack Model.

References

1. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. pp. 217–224. ACM (2002)
2. Artz, M.L.: Netspa: A network security planning architecture. Ph.D. thesis, Massachusetts Institute of Technology (2002)
3. Ben-Gal, I., Ruggeri, F., Faltin, F., Kenett, R.: Bayesian networks, encyclopedia of statistics in quality and reliability (2007)
4. Druzdzel, M.J.: Smile: Structural modeling, inference, and learning engine and genie: a development environment for graphical decision-theoretic models (1999)
5. Forum of Incident Response and Security Teams: Common vulnerability scoring system v3.0: Specification document. pp. 1–21 (2015)
6. Frigault, M., Wang, L.: Measuring network security using bayesian network-based attack graphs pp. 698–703 (July 2008)
7. ISO, I., Std, I.: Iso 27005: 2011. Information technology–Security techniques–Information security risk management. ISO (2011)
8. Jajodia, S., Noel, S., O'Berry, B.: Topological analysis of network attack vulnerability. Managing Cyber Threats pp. 247–266 (2005)
9. Jajodia, S., Noel, S., Kalapa, P., Albanese, M., Williams, J.: Cauldron mission-centric cyber situational awareness with defense in depth. In: Military Communications Conference. pp. 1339–1344. IEEE (2011)
10. Kordy, B., Piètre-Cambacédès, L., Schweitzer, P.: DAG-based attack and defense modeling: Dont miss the forest for the attack trees. Computer science review 13, 1–38 (2014)
11. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society (1988)
12. Lippmann, R.P., Ingols, K.W.: An annotated review of past papers on attack graphs. Tech. rep., DTIC Document (2005)
13. Liu, Y., Man, H.: Network vulnerability assessment using bayesian networks. In: Defense and Security. pp. 61–71. International Society for Optics and Photonics (2005)
14. López, D., Pastor, O., García Villalba, L.: Dynamic risk assessment in information systems: state-of-the-art. In: Proceedings of the 6th International Conference on Information Technology, Amman. pp. 8–10 (2013)
15. Ou, X., Govindavajhala, S., Appel, A.W.: Mulval: A logic-based network security analyzer. In: USENIX Security Symposium (2005)
16. Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial intelligence 29(3), 241–288 (1986)
17. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann (1988)
18. Phillips, C., Swiler, L.P.: A graph-based system for network-vulnerability analysis (1998)
19. Poolsappasit, N., Dewri, R., Ray, I.: Dynamic security risk management using bayesian attack graphs. Dependable and Secure Computing (2012)
20. Qin, X., Lee, W.: Attack plan recognition and prediction using causal networks. In: Computer Security Applications Conference. pp. 370–379 (Dec 2004)
21. Xie, P., Li, J.H., Ou, X., Liu, P., Levy, R.: Using bayesian networks for cyber security analysis. In: IEEE/IFIP International Conference on Dependable Systems and Networks. pp. 211–220. IEEE (2010)

A Appendix: Detail of a Bayesian attack step

Figure 3 shows the details of the representation of an attack step from tn_n (source) to tn_{n+1} (target). It is composed of a Bayesian attack step node that binds a Bayesian topological node to another one. This Bayesian attack step has two conditions (bcn_1 and bcn_2) and a sensor ($bsen$).

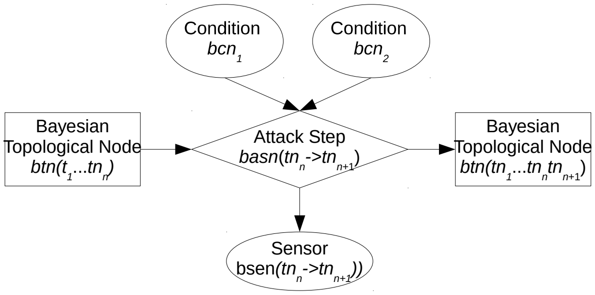


Fig. 3. Bayesian attack step

B Appendix: Conditional Probability Tables

In this appendix, we detail the conditional probability tables (CPTs) associated with the nodes of the Bayesian Attack Model. Each node with at least one parent is associated with a CPT which depends on its type of node. In these tables, the first lines represent all possible states of the parents. The last lines contain the probabilities of each state of the child node according to the states of its parents.

Table 1 shows the CPT of a Bayesian topological node, according to the states of its parents: Bayesian attack step nodes. It represents a noisy OR: an OR with a small residual probability (the $probabilityUnknownAttack$ parameter).

Bayesian attack step node 1	*Succeeded*	*Failed*	*Succeeded*	*Failed*
Bayesian attack step node 2	*Succeeded*		*Failed*	
Bayesian topological node				
Compromised	1	1	1	pua
NotCompromised	0	0	0	$1 - pua$

with $pua = probabilityUnknownAttack$.

Table 1. CPT of a Bayesian topological node

Table 2 shows the CPT of a Bayesian attack step node, for the exploitation of a vulnerability, according to the states of its parents: a Bayesian topological node and a Bayesian condition node. It represents an AND on the parents with the $probabilityNewAttackStep$ parameter, when all conditions are fulfilled.

Bayesian topological node	$Comp$	$NotComp$	$Comp$	$NotComp$
Bayesian condition node	*Succeeded*		*Failed*	
Bayesian attack step node				
Succeeded	$pnas$	0	0	0
Failed	$1 - pnas$	1	1	1

with $Comp$ = State $Compromised$; $NotComp$ = State $NotCompromised$; $pnas = probabilityNewAttackStep$.

Table 2. CPT of a Bayesian attack step node ”exploitation of a vulnerability”

Finally, Table 3 shows the CPT of a Bayesian sensor node, according to the state of its parent: a Bayesian attack step node. It represents the potential false positive and false negative rates of the sensor.

Bayesian attack step node	*Succeeded*	*Failed*
Bayesian sensor node		
Alert	$1 - falseNegative$	$falsePositive$
NoAlert	$falseNegative$	$1 - falsePositive$

Table 3. CPT of a Bayesian sensor node

C Appendix: Simulation scenarios

Table 4 shows the detection scenarios applied on the use-case. In the first scenario, no step is detected; it represents the basic risk of the IT system. In scenarios 2 to 4, steps $I \rightarrow A$, $A \rightarrow G$ and $G \rightarrow D$ are progressively detected and alerts are generated. Scenarios 5 and 6 represent detection anomalies on $A \rightarrow G$ (no sensor information or false negative).

Scenario	$I \to A$	$A \to G$	$G \to D$	**Comment**
1	×	×	×	Basic risk
2	✓	×	×	First alert
3	✓	✓	×	Second alert
4	✓	✓	✓	Third alert
5	✓	O	✓	No information available for the second step (=no sensor)
6	✓	×	✓	No detection for the second step

Caption $I \to A$: Attack from the Internet to host A; $A \to G$: Attack from host A to host G; $G \to D$: Attack from host G to host D; O: No values set (=no sensor); ✓: Sensor node set to *alert*; ×: Sensor node has been set to *noalert*.

Table 4. Simulation scenarios

D Appendix: Default values of the parameters

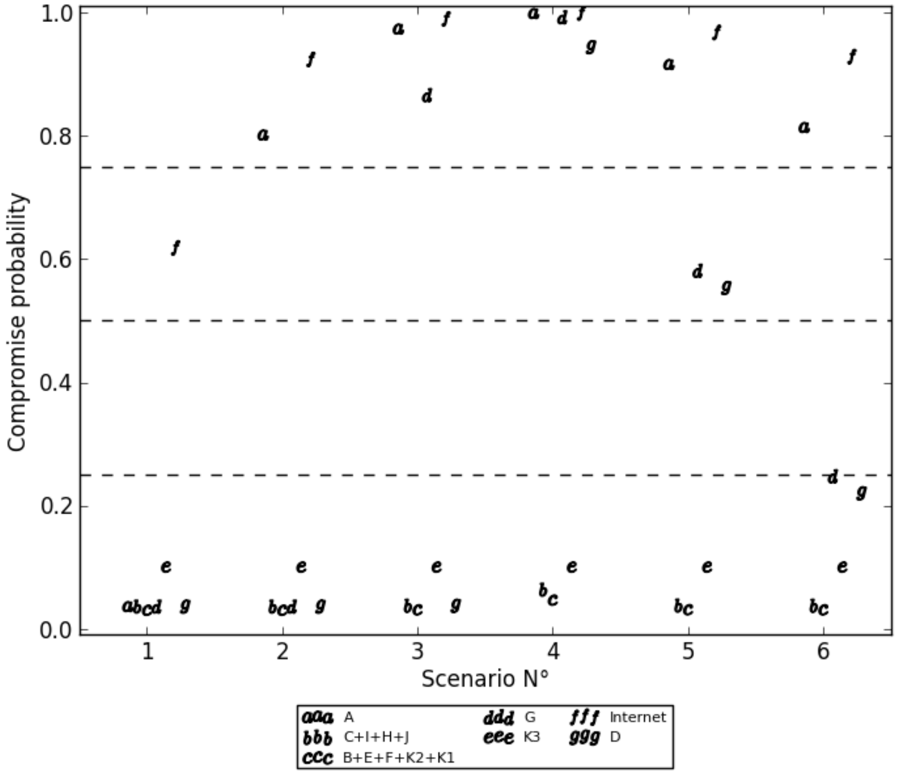
Table 5 presents all the parameters of the Bayesian Attack Model: *probabilityUnknownAttack*, *falsePositive*, *falseNegative*, *nbSteps*, *probabilityInternet*, *probabilityOtherHosts*, and *probabilityNewAttackStep*. Each parameter is associated with its description and the default value that was chosen for the use-cases.

Parameter name	**Default value**	**Meanings**	**Default value explanation**
probability-UnknownAttack	0.001	Probability that an unknown attack occurs.	Very small probability of having a 0-day, a unknown vulnerability.
falsePositive	0.05	False positive rate of each sensor.	Sensors may raise an alert, even if the attack has not succeeded.
falseNegative	0.01	False negative rate of each sensor.	This value is smaller as it only concerns vulnerabilities for which a sensor has been deployed.
nbSteps	3	Number of successive attack steps to keep.	Allow to recognise multi-step attacks with at most 2 missing alerts. *C.f.* Subsection 2.6 for full explanation.
probability-Internet	0.7	A priori probability of an attack coming from the Internet.	The internet is the main source of attacks. Thus, 70% of chances of being a source of attack, 30% not to be a source.
probability-OtherHosts	0.1	A priori probability of an attack issued from an internal host.	An internal host may issue an attack. Thus, 10% of chances of being a source of attack, 90% not to be a source.
probability-NewAttackStep	0.3	Probability that the attack propagates through a new attack step.	70% of chance that the attacker does not continue his attack. He may have already found what he was looking for.

Table 5. Default values of the parameters used in the BAM

E Appendix: Validation results

Figure 4 shows the results of the BAM for the six scenarios of the use case presented in Subsection 3.2. Markers represent hosts of the topology, and the ordinate is their compromise probability in the abscissa scenario. The horizontal lines give some idea of the threshold that could be taken to define the compromise risk level of the hosts. For example, the hosts under the lowest line (*probability* $\leq$ 0.25) have a *not-significant* risk of being compromised, above the lowest line ($0.25 <$ *probability* ≤ 0.50) have a *low* risk, above the second line ($0.50 <$ *probability* ≤ 0.75) have a *medium* risk, and above the upper line ($0.75 <$ *probability*) have a *high* risk of being compromised.



For readability of the results, the hosts having the same value (more or less 10^{-10}) for all the scenarios have been grouped on one point, and the points are spread around the scenario number.

Fig. 4. Results for each scenario

F Appendix: Parameters sensitivity analysis results

Table 5 summarises the sensitivity analysis of the parameters of the Bayesian Attack Model. We give in this table the range of variation that we find appropriate for the parameters. Then, we study the influence of each parameter in its whole variation range on the ranking between the compromise probability of the hosts, and on the value of the probabilities.

Name	**Variation range**	**Ranking influence**	**Proba influence**	**Comment**
falseNegative	$[0.0 - 0.3]$	No impact	Almost no impact	
falsePositive	$[0.0 - 0.3]$	No impact	Medium impact in scenarios 3, 4 and 5 (decrease)	Impact for low values (between 0 and 0.05). Effect amplified by the number of detections set.
nbSteps	$[[1 - 4]]$	No impact	Medium impact in scenarios 4, 5 and 6	Parameter sensitive particularly for scenarios that contain attacks longer than *nbSteps*.
probability Internet	$[0.0 - 1.0]$	No impact (except the Internet)	Little impact (increase)	Probability of hosts attacked directly from the Internet increases with the increase of this parameter.
probability OtherHosts	$[0.0 - 1.0]$	Medium Impact (different probability growth curve)	Medium impact (increase)	Probability of all hosts (except the Internet) increase with the increase of such parameter. Stronger increase for hosts attackable from many hosts.
probability NewAttack	$[0.0 - 0.15]$	No impact (except the Internet)	Low impact (increase)	With the increase of this parameter, the probability of attackable hosts increases slowly, as it is more probable that they are attacked, using attacks that are not known and cannot be detected.
probability NewAttack-Step	$[0.0 - 1.0]$	No impact	Medium impact (increase then decrease, maximum around 0.3)	When this parameter is small (increase from 0 to 0.3), the parameter represents that even if an attack is possible, it may not happen. Then (decrease from 0.3 to 1) it represents that even if an attacker has compromised a host, he may not do another attack.

Table 6. Sensitivity analysis of the parameters of the BAM

G Appendix: Performance and evaluation results

Figure 5 presents the results of the duration in seconds of the Bayesian Attack Model generation and the inference after the evaluation of one scenario of 7 successive attack steps, on random simulated topologies from 1 to 70 hosts.

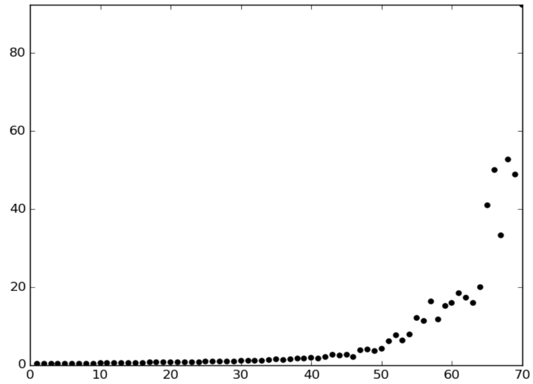


Fig. 5. Duration in seconds, according to the number of hosts

Figure 6 presents the results of an accuracy evaluation of the BAM on random simulated topologies. The curve with triangles represent the mean and standard deviation, during 10 simulations, of the minimum probability of the hosts known as compromised. The curve with circles represent the mean and standard deviation, during 10 simulations, of the maximum probability of the hosts known as healthy. In other words, this graph shows the maximum errors (in terms of distance to the theoretical values 1 and 0) of compromised and healthy nodes.

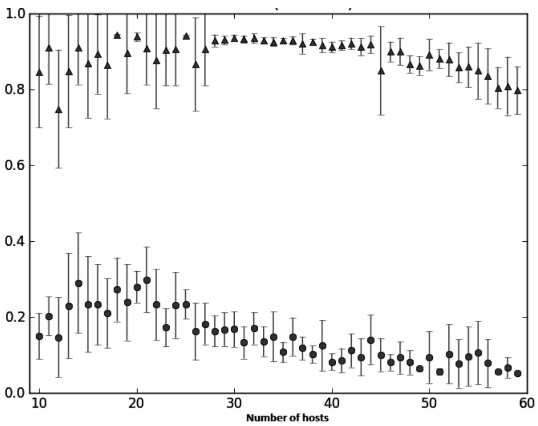


Fig. 6. Accuracy of the results of the BAM according to the number of hosts