Efficient Attack Graph Analysis through Approximate Inference

LUIS MUÑOZ-GONZÁLEZ, Imperial College London
DANIELE SGANDURRA, Royal Holloway, University of London
ANDREA PAUDICE, Imperial College London
EMIL C. LUPU, Imperial College London

Attack graphs provide compact representations of the attack paths an attacker can follow to compromise network resources from the analysis of network vulnerabilities and topology. These representations are a powerful tool for security risk assessment. Bayesian inference on attack graphs enables the estimation of the risk of compromise to the system's components given their vulnerabilities and interconnections, and accounts for multi-step attacks spreading through the system. Whilst static analysis considers the risk posture at rest, dynamic analysis also accounts for evidence of compromise, e.g. from SIEM software or forensic investigation. However, in this context, exact Bayesian inference techniques do not scale well. In this paper we show how Loopy Belief Propagation - an approximate inference technique - can be applied to attack graphs, and that it scales linearly in the number of nodes for both static and dynamic analysis, making such analyses viable for larger networks. We experiment with different topologies and network clustering on synthetic Bayesian attack graphs with thousands of nodes to show that the algorithm's accuracy is acceptable and that it converges to a stable solution. We compare sequential and parallel versions of Loopy Belief Propagation with exact inference techniques for both static and dynamic analysis, showing the advantages and gains of approximate inference techniques when scaling to larger attack graphs.

CCS Concepts: •**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

General Terms: Security risk assessment, attack graphs, dynamic analysis

Additional Key Words and Phrases: Bayesian networks, probabilistic graphical models, approximate inference

ACM Reference Format:
Luis Muñoz-González, Daniele Sgandurra, Andrea Paudice, Emil C. Lupu, 2016. Efficient Attack Graph Analysis through Approximate Inference. *ACM Trans. Info. Syst. Sec.* 1, 1, Article 1 (January 2017), 30 pages.
DOI: http://dx.doi.org/10.1145/3105760

1. INTRODUCTION

Despite significant efforts to protect networks against cyber-attacks [Gartner, Inc. 2014], system administrators cannot cope with the sophistication of modern threats, as shown by the history of breaches that organizations have suffered in recent years [Lord 2015]. One of the most common strategies to protect networks is to identify and patch vulnerabilities. However, this is often not systematically done, either for lack of resources or because it requires interrupting critical systems. A risk-driven approach is therefore needed to optimise resources for network protection. Such an approach requires assessing the networks risks, prioritizing the most critical threats, and then estimating the *risk exposure*, given the likelihood of threats and the severity of the

This work has been supported by the UK government under EPSRC grant EP/L022729/1.
Authors' address: Department of Computing, Imperial College London, 180 Queen's Gate, SW7 2AZ, London, UK. E-mail: {l.munoz, a.paudice15, e.c.lupu}@imperial.ac.uk. Information Security Group, Royal Holloway, University of London, UK. E-mail: daniele.sgandurra@rhul.ac.uk.

DOI: http://dx.doi.org/10.1145/3105760

impacts [Wheeler 2011]. Finally, these values are used by administrators to select appropriate countermeasures. But often this analysis is carried out separately for each of the network components ignoring interdependencies between vulnerabilities, i.e. how successfully exploiting a vulnerability allows an attacker to exploit other vulnerabilities, thus moving across the network and acquiring privileges at every step.

These shortcomings can be addressed using Attack Graphs (AGs) [Sheyner et al. 2002; Ammann et al. 2002], a well-established technique to represent the possible paths of an attacker through the system by exploiting successive vulnerabilities. AGs allow system administrators to reason about threats and risks in a formal way to better select countermeasures [Ingols et al. 2009]. Two types of analysis can be performed. *Static analysis* determines the *a priori* risks to which network components are exposed. *Dynamic analysis* updates those risks in light of any indication that some of the network components may have been compromised, e.g. from Security Information and Event Management (SIEM) and Intrusion Detection Systems (IDS). Dynamic analysis also allows administrators to profile the attacker's paths, to determine the nodes that are more likely to be attacked in the next steps and evaluate the security risk when we observe evidence of an ongoing attack. As organizations are often under attack, dynamic analysis gives administrators important insights on the most vulnerable targets and where they should spend their efforts at run-time. Although several metrics have been proposed to perform security risk assessment using AGs [Wang et al. 2007a; 2007b], taking into consideration the length and the number of paths that let the attacker reach a goal, or the global impact of the existing vulnerabilities in the network, they fail to consider the difficulty of exploiting each vulnerability and the dependencies between the different attack paths. In this sense, it is easy to observe that both static and dynamic analysis of AGs have inherent probabilistic characteristics given the uncertainty about the attackers' ability to successfully exploit vulnerabilities. Therefore, considering the dependencies between vulnerabilities, Bayesian Networks (BNs) provide an appropriate framework to model AGs [Liu and Man 2005; Frigault et al. 2008]. They depict causal relationships between random variables in a compact way, so they can model the uncertainty about the attacker's behaviour and capabilities.

Bayesian Attack Graphs (BAGs) can be analysed through efficient algorithms, such as Variable Elimination or Junction Tree (JT), to make *exact inference*, i.e. to compute the unconditional probabilities of all the nodes in the BAG, i.e. the probabilities that an attacker can reach the different security states in the AG [Muñoz-González et al. 2016]. However, computing these probabilities is known to be an NP-Hard problem [Cooper 1990], and this limits the applicability of exact inference techniques to medium-size graphs (in the order of 100-1,000 nodes), especially when the structure of the graph is dense [Muñoz-González et al. 2016]. However, empirical investigations show that networks are highly complex: estimates of mean corporate network size are in the order of thousand of nodes [Sharma et al. 2011; Raftopoulos and Dimitropoulos 2013]. Moreover, each host may have somewhere between 2 and 11 vulnerabilities according to [WhiteHat Security 2015]. For example, Websites are reported to have an average of 6.5 vulnerabilities [Symantec 2015]. Thus, exact inference techniques can be very slow and computationally very expensive for such complex networks, especially for the dynamic analysis of BAGs. For this reason, simpler metrics, computationally less demanding, have been proposed for tractable analysis of AGs in real networks [Idika and Bhargava 2012; Noel and Jajodia 2014]. However, these metrics do not consider the dependencies between vulnerabilities.

To sidestep this limitation we propose to use *approximate inference* techniques to allow to analyse AGs in larger networks in a tractable way. Although approximate inference in BNs is also NP-Hard [Koller and Friedman 2009], inference algorithms, such as Loopy Belief Propagation (LBP) [Pearl 1988], have better scalability than ex-

act inference techniques. We should not be deterred by the "approximation" involved in this context for two reasons. First, because the probabilities are used for prioritising threats, so even if significant differences between them are meaningful, their absolute or accurate values are relatively less important. Second, because the probability of successful exploitation of a vulnerability is already a rough estimate, often based on the Common Vulnerability Scoring System (CVSS) [Common Vulnerability Scoring System, V3 2016; Frigault et al. 2008; Poolsappasit et al. 2012], which ignores other factors such as attacker's skills, knowledge and tooling and, hence, may not be an accurate indication of compromise *per se*. Therefore, notwithstanding their inherent approximation, the scalability of these algorithms makes them crucial to perform qualitative risk analysis of large networks. Although the validity of the results is limited by the errors in the estimates of the probabilities, in our experimental evaluation we show that the accuracy of the probability estimates provided by LBP is reasonable for static and dynamic risk analysis and mitigation using BAGs.

The main contributions of this work are as follows:

— We have proposed a revised BAG model from those described in [Liu and Man 2005; Frigault et al. 2008; Poolsappasit et al. 2012; Muñoz-González et al. 2016] considering a non-perfect behaviour in the IDS and the possible presence of zero-day vulnerabilities. We describe how to compute the conditional probability tables taking into account these two factors and show their effect on the unconditional and posterior probabilities compared to the ideal case.
— To the best of our knowledge, we are the first to propose the use of approximate inference techniques for the scalable analysis of AGs. Furthermore, we propose and compare both the sequential and parallel implementations of LBP to estimate the probabilities of compromise for the network nodes.
— We provide a comprehensive experimental evaluation using synthetic AGs which emulate the complexity of real scenarios. These experiments allow us to assess the applicability of sequential and parallel implementations of LBP for the analysis of AGs with thousands of nodes, and show the limitations of existing exact inference approaches, such as the JT algorithm [Shenoy and Shafer 1990].
— We show through experiments that LBP scales linearly in the number of nodes for both static and dynamic analysis of BAGs, which contrasts with the exponential scalability of exact inference (e.g. JT) in some cases, especially when the AG is dense. The experiments further show that the accuracy of LBP is sufficient for many practical needs exhibiting a rooted mean squared error smaller than 0.03.
— Finally, we show that it is possible to get accurate results before the LBP algorithm fully converges, by allowing administrators to monitor the probability estimates at each iteration, and enabling them to start planing risk mitigation strategies in advance. This is not possible with exact inference techniques.

The rest of the paper is organised as follows. In Sect. 2 we discuss the related work. In Sect. 3 we introduce our Bayesian AG model, including an example of a small typical corporate network as a use case. The use of Belief Propagation and LBP for the analysis of BAGs is described in Sect. 4. We discuss the Junction Tree algorithm proposed in [Muñoz-González et al. 2016] in Sect. 5. In Sect. 6 we present the experimental results for the static and dynamic analysis of synthetic AGs. Finally, in Sect. 7 we present the main conclusions and our plans for further work.

2. RELATED WORK

AGs are graphical models that represent the knowledge about network vulnerabilities and their interactions, showing the different paths an attacker can follow to reach a given goal by exploiting a set of vulnerabilities. Along each attack path, vulnerabilities

are exploited in sequence, so that each successful exploit gives the attacker more privileges towards his goal. In the literature of AGs we can distinguish two main types of representations, namely *state-based* and *logical* AGs.

State-based representations of AGs [Jha et al. 2002; Phillips and Swiler 1998; Sheyner et al. 2002; Sheyner and Wing 2004; Swiler et al. 2001] result in directed graphs, where each node represents the state of the whole network after a successful atomic attack. This approach suffers from a combinatorial explosion of the graph when increasing the number of nodes and vulnerabilities in the network. Moreover, these graphs contain duplicate attack paths that differ only in the order of the attack steps, which also increase the complexity of the graph, limiting the applicability of state-based representations to very small networks [Ammann et al. 2002; Jajodia et al. 2005; Ou et al. 2006].

The scalability problems of state-based representations are overcome with *logical AGs*, which are bipartite graphs that represent dependencies between exploits and security conditions [Ammann et al. 2002; Jajodia et al. 2005]. These representations rely on the monotonicity principle: the attacker never relinquishes privileges once obtained. Although not always applicable, this assumption is reasonable in most cases, as discussed in [Ammann et al. 2002]. Monotonicity allows to remove duplicated paths and to generate AGs that grow polynomially with the number of vulnerabilities and the number of connected pairs of hosts [Albanese et al. 2012].

AGs are a powerful tool to perform risk assessment and different metrics have been proposed in the literature. [Lippmann et al. 2006] propose to use the percentage of network assets an attacker has compromised. However, this metric is not goal-oriented, as it is the case of AGs. [Pamula et al. 2006] propose the weakest adversary metric, i.e. the measure of the risk according to the weakest attack path in the graph. Simpler approaches are used in [Phillips and Swiler 1998; Ortalo et al. 1999; Li and Vaughn 2006], where they propose to use the shortest path, the number of paths, and the average path length as metrics to measure risk. In a similar way, [Idika and Bhargava 2012] propose the normalized mean, standard deviation, mode, and median of the path lengths as a set of metrics to assess risk in AGs. [Noel and Jajodia 2014] propose a metrics suite for AGs that takes into account the CVSS scores of the vulnerabilities in the AGs as well as topological aspects of the graph, such as the connectivity, the number of cycles, and the depth. However, most of these metrics fail to account for the dependencies of the vulnerabilities and attack paths, as well as the difficulty to exploit the vulnerabilities. These limitations can be addressed with probabilistic models, which allow to compute the probabilities of each node in the graph to be compromised by an attacker when the network is at rest or under attack.

Other security metrics have also been proposed to evaluate the risk of *zero-day* vulnerabilities: [Wang et al. 2014] introduce *k-zero day safety*, a security metric to assess the impact of zero-day vulnerabilities taking into account how many vulnerabilities would be required for compromising network assets. Following a similar spirit, [Zhang et al. 2016] propose to use network diversity as a security metric to evaluate the resilience of networks w.r.t. zero-day attacks. The authors define different metrics based on the effective number of distinct resources and diversity metrics based on attacking efforts. From a different perspective, [Bilge and Dumitras 2012] present a empirical analysis of the impact of zero-day attacks based on the collection of real data.

Beyond standard risk assessment with AGs, [Albanese et al. 2011] introduce attack scenario graphs, which combine AGs with dependency graphs. The latter representations capture the dependencies across network components and services. Attack scenarios allows to provide more comprehensive risk assessment taking into account both the vulnerabilities present on the network and the services running.

Probabilistic models for AGs have been already proposed in the literature: For example, [Liu and Man 2005; Frigault et al. 2008; Wang et al. 2008] present mechanisms to calculate the conditional probability tables, which represent the probabilities of compromising a node given all possible states of the parent nodes (or preconditions). However, no mechanism is proposed to calculate the unconditional probabilities, i.e. the probabilities of compromising the nodes in the network regardless of the state of their corresponding preconditions. A more complete Bayesian model is described in [Xie et al. 2010], which takes into account non-perfect behaviour of the alert correlation system and the impact of zero-day vulnerabilities. However, no inference technique is proposed to calculate the unconditional probabilities and the experimental evaluation does not show the applicability of their model to networks of different sizes and the impact of having zero-day vulnerabilities and non-perfect IDS.

Several techniques have also been proposed in the literature for exact inference on BAGs. For example, forward-backward propagation is proposed in [Poolsappasit et al. 2012] to compute the unconditional probabilities. However, this procedure is only valid for chains [Rabiner and Juang 1986; Murphy 2012] and cannot be applied to general AGs. [Liu and Man 2005] propose to use Variable Elimination (VE) [Dechter 1996] for exact inference on BAGs. Although this is an efficient technique, it is highly dependent on the heuristic for the elimination ordering and no heuristic is suggested in [Liu and Man 2005]. Furthermore, none of these papers reports an experimental evaluation of the time and memory required by the techniques proposed to assess their suitability for static and dynamic analysis of AGs. More recently, the JT algorithm was proposed in [Muñoz-González et al. 2016] for exact inference in BAGs. This technique allows to efficiently compute the exact unconditional probabilities by using a probabilistic message passing scheme. The experimental evaluation shows the advantages of JT over VE in terms of the time and the memory required. However, the applicability of JT to large networks is limited, especially when the AGs are dense.

3. BAYESIAN ATTACK GRAPHS

In this section we propose a BAG model similar to the ones proposed in [Liu and Man 2005; Poolsappasit et al. 2012; Muñoz-González et al. 2016] but relaxing some of their assumptions to consider the effect of zero-day vulnerabilities and to model the non-perfect behaviour of the IDS. Thus, we first describe the general background on BAGs and the assumptions we are considering in our revised model. Then, we describe how to compute the conditional probability tables needed to make inference on the BAG model for the static and the dynamic analysis. We also discuss the impact of zero-day vulnerabilities and non-perfect IDS on the computation of the probabilities. Finally, we show an example of our proposed BAG model on a synthetic example representative of a small corporate network.

3.1. Background

Some of the literature on AG analysis assumes that monotonicity induces a Directed Acyclic Graph (DAG) structure of logical AGs [Liu and Man 2005; Poolsappasit et al. 2012; Muñoz-González et al. 2016]. Although monotonicity helps to get rid of many directed cycles related to duplicate attack paths (that appear in state-based representations) some directed cycles still remain. However, [Wang et al. 2008] explain how to handle and eliminate directed cycles without loss of integrity. In this paper, we consider AGs with a DAG structure. Where directed cycles appear, we refer to [Wang et al. 2008] to build the corresponding conditional probability tables in the nodes affected by

the cycle. The DAG structure[1] of logical AGs and the uncertainty about the attacker's behaviour and capabilities, make BNs a suitable alternative to model AGs and perform static and dynamic analysis. In particular, BNs allow to calculate the probability of an attacker reaching a security condition (state) in the AG.

Definition 3.1. A Bayesian Network (BN) is a directed acyclic graphical model where the nodes represent random variables and the directed edges represent the dependencies between random variables. Let $\mathbf{X} = \{X_1, ..., X_n\}$ be a set of (continuous or discrete) random variables. The joint probability distribution can be written as:

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_i|\mathbf{pa}_i) \tag{1}$$

Then, under the BN representation, for each node X_i there is a directed edge from each node in $\mathbf{pa}_i$, the set of parents nodes of X_i, pointing to X_i.

In the context of the BAG, the nodes represent the different security states that an attacker can reach. We model the behaviour of these states as Bernoulli random variables. Hence, the probability of an attacker compromising a node X_i is $\Pr(X_i = \mathrm{T}) = p$, whereas the probability of an attacker not compromising that node is $\Pr(X_i = \mathrm{F}) = 1 - p$, with $p \in [0, 1]$.[2] The probabilities of an attacker successfully exploiting a vulnerability, needed to compute the conditional probabilities $p(X_i|\mathbf{pa}_i)$ in (1), are represented as parameters of the model, since these values varies slowly across time (in the order of days or weeks).

3.2. Model Assumptions

In line with much of the related work [Liu and Man 2005; Frigault et al. 2008; Poolsappasit et al. 2012; Muñoz-González et al. 2016], we make some assumptions in our model, although we relax some constraints to model the non-perfect behaviour of the IDS and the presence of zero-day vulnerabilities:

— We consider that the probability of successfully exploiting a single vulnerability does not affect the probabilities that the attacker can successfully exploit other vulnerabilities. We also assume that these probabilities remain nearly constant in time. Although in [Frigault et al. 2008] the dynamic aspects of vulnerabilities are modelled with a Dynamic Bayesian Network, in practice changes to the probabilities typically occur over periods of days or weeks. Therefore, we argue that it is better to recompute the model when changes occur rather than to increase the complexity of the model to include the dynamic aspects of these probabilities.
— We assume that the probability of successfully exploiting a vulnerability is the same regardless of the attacker. However, these probabilities can be updated according to other models that take into account the attackers' capabilities and preferences. For example, we can use different sets of AG parameters corresponding to different attacker models identified as proposed in [Baiardi and Sgandurra 2013].
— We also assume that the topology of the network, host connectivity (including open ports) and the set of vulnerabilities do not change during the dynamic analysis of the BAG. This would require dynamic AG generation, which is out of the scope of this paper. However, if existing vulnerabilities are patched at run-time, our BAG representation can be easily updated by setting the probability of successful exploitation

[1]Note that, although we do not consider directed cycles, there are still (non-directed) loops in the AG for this logical representation of AGs.
[2]To simplify the mathematical notation we will refer to the unconditional probability of a node to be compromised as $\Pr(X_i)$ instead of $\Pr(X_i = \mathrm{T})$

of the patched vulnerabilities to zero. On the contrary, if new vulnerabilities are discovered or new nodes are added to the network, a new AG needs to be generated.

3.3. Conditional Probability Tables

The information available at each node X_i in a BAG is the conditional probability distribution $p(X_i|\mathbf{pa}_i)$, the probability of a node to be compromised given the state of its parent nodes $\mathbf{pa}_i$. Thus, these conditional probabilities represent the probabilities of an attacker reaching a security state X_i given the observations of its preconditions $\mathbf{pa}_i$ and the vulnerabilities $\mathbf{v}_i$ that can be exploited to compromise X_i. In this sense, we consider that the probabilities of successfully exploiting vulnerabilities are parameters of the BAG model that are used to calculate $p(X_i|\mathbf{pa}_i)$.

A common approach to estimate p_{v_i}, the probability of an attacker successfully exploiting a vulnerability v_i, is based upon CVSS [Common Vulnerability Scoring System, V3 2016]. Although CVSS scores are intended to estimate the impact of a vulnerability rather than its average probability of being successfully exploited, CVSS scores (or some of their submetrics) are often used in the literature to estimate p_{v_i} [Frigault et al. 2008; Poolsappasit et al. 2012; Muñoz-González et al. 2016]. In this sense, the exploitability submetric of CVSS can be considered more appropriate to estimate p_{v_i}, since it tries to measure the difficulty of exploiting a vulnerability.

Given an estimate of the probabilities of exploitation of the vulnerabilities that allow an attacker to reach a security state X_i from states $\mathbf{pa}_i$, we consider two possible cases to build the conditional probability tables [Poolsappasit et al. 2012; Muñoz-González et al. 2016]: *AND* and *OR* conditional probability tables. In the first case, all the preconditions need to be satisfied to be able to compromise X_i, i.e. the attacker needs to compromise all the nodes in $\mathbf{pa}_i$ before being able to perform an attack to compromise X_i. In the case of *OR* conditional probability tables, the attacker only needs to compromise one of the nodes in $\mathbf{pa}_i$ to attempt an attack to reach the security state X_i.

Considering that the alert system is not perfect and that there can be zero-day vulnerabilities, we introduce a leak factor, p_l, which we describe in Sect. 3.6, to consider these aspects. Then, *AND* conditional probability tables can be calculated as:

$$p(X_i|\mathbf{pa}_i) = \begin{cases} p_l, & \exists X_j \in \mathbf{pa}_i | X_j = F \\ 1 - (1 - p_{l_i})\,(1 - \prod_{j:X_j} p_{v_j}), & \text{otherwise} \end{cases} \tag{2}$$

whereas for *OR* conditional probability tables, using the noisy-OR formulation [Koller and Friedman 2009], we have:

$$p(X_i|\mathbf{pa}_i) = \begin{cases} p_l, & \forall X_j \in \mathbf{pa}_i | X_j = F \\ 1 - (1 - p_{l_i})\,\prod_{j:X_j}(1 - p_{v_j}), & \text{otherwise} \end{cases} \tag{3}$$

The leak factor p_{l_i} in (2) and (3) models the cases where, even when all the parent nodes are in the False state (those parent nodes have not been compromised), there is still some (non-zero) probability of X_i taking the True state. The reason for this is that the IDS system may have triggered a false alarm or X_i has been compromised because there is a zero-day vulnerability that allows the attacker to compromise X_i. In Sect. 3.6 we give more details about the effect of p_{l_i}. By combining (2) and (3) we can extend the construction of conditional probability tables to intermediate cases, where different subsets of preconditions need to be satisfied before trying to compromise X_i.

3.4. Prior on the Attacker's Initial State

In AG representations, there is usually a leaf node representing the initial state of the attacker when the attacker has not compromised any node in the network yet. Follow-

ing the model in [Muñoz-González et al. 2016], we consider that this node is not really a random variable, since it only represents that the attacker has full rights on his own machine. Under the Bayesian representation, we can consider that the Bernoulli random variable X_0 representing the initial state of the attacker has $\Pr(X_0) = 1$. Although [Poolsappasit et al. 2012] propose to use this initial node to reflect some subjective prior knowledge of the attacker capabilities (by letting the administrator set the value of $\Pr(X_0)$), this can lead to misleading conclusions, especially when reasoning using new evidence about the nodes that an attacker may have already compromised, as discussed in [Muñoz-González et al. 2016]. Finally, we can break loops in the BN by instantiating one initial node for each possible initial attack path. This does not affect the value of the unconditional probabilities of the rest of the nodes but can favour the convergence and the accuracy of LBP estimates [Murphy et al. 1999].

3.5. Static and Dynamic Analysis of the BAG

For the static analysis of AGs, i.e. considering the network at rest, we are interested in calculating the *unconditional probability distributions* $p(X_i)$, rather than $p(X_i|\mathbf{pa}_i)$. Thus, $p(X_i)$ corresponds to the probability of an attacker reaching a given security condition and, hence, is an indicator of the risk. Using Bayes rule, it is possible to calculate $p(X_i)$ from the product of conditional probability distributions:

$$p(X_i) = \sum_{\mathbf{X}-X_i} p(\mathbf{X}) = \sum_{\mathbf{X}-X_i} \prod_{j=1}^{n} p(X_j|\mathbf{pa}_j) \tag{4}$$

where $\mathbf{X} - X_i$ indicates that we sum over all the set of random variables $\mathbf{X}$ except X_i. These probabilities can be used as risk estimates to detect weak areas in the network and serve as an input for network hardening or static risk mitigation techniques [Noel et al. 2003; Albanese et al. 2012].

In contrast, for the dynamic analysis of AGs, given evidence of attacks on a set of nodes $\mathbf{X}_e$, e.g. through SIEM alerts, we need to compute the posterior probability $p(X_i|\mathbf{X}_e)$, i.e. the probability of an attacker compromising the node X_i given that we have observed evidence of attack at nodes $\mathbf{X}_e$. Again, using Bayes rule, we can compute this posterior distribution from the joint probability distribution:

$$p(X_i|\mathbf{X}_e) = \frac{p(X_i, \mathbf{X}_e)}{p(\mathbf{X}_e)} = \frac{\sum_{\mathbf{X}-\{X_i,\mathbf{X}_e\}} p(\mathbf{X})}{\sum_{\mathbf{X}-\mathbf{X}_e} p(\mathbf{X})} \tag{5}$$

These posterior probabilities provide a re-estimation of the risk at run-time and can help system administrators to apply security measures to contain an ongoing attack or to protect some valuable network resources. In this sense, [Poolsappasit et al. 2012] propose a model for dynamic risk mitigation with the probabilities provided by a BAGs by formulating a discrete reasoning problem solved using a genetic algorithm.

The exact calculation of (4) and (5) is an NP-Hard problem [Koller and Friedman 2009]. Thus, applying brute force and computing the joint probability distributions to make inference in probabilistic graphical models is not a reasonable approach in terms of computational time and memory. Efficient algorithms, such as Variable Elimination (VE) [Dechter 1996] or Junction Tree (JT) [Shenoy and Shafer 1990; Shafer and Shenoy 1990], are necessary even for small graphs. However, the applicability of these techniques is limited, especially when the graphs are dense, demanding a lot of memory to compute the unconditional probabilities. Even when the structure of BAGs is expected to have some special properties, given the network structure and the number of vulnerabilities, there are no guarantees about the computational complexity for exact inference techniques. In these cases, approximate inference is a reasonable alternative to enable a tractable estimation of the unconditional and posterior probabilities.

Although approximate inference in BNs is also NP-Hard [Koller and Friedman 2009], efficient techniques like LBP [Pearl 1988] allow to efficiently estimate the unconditional and the posterior probabilities in (4) and (5) for large networks. In Sect. 4 we describe how to use LBP for the analysis of BAG.

3.6. Zero-day Vulnerabilities and Non-perfect IDS

Estimating the error probability, p_e, of the alert correlation system is difficult due to the dynamic aspects of the system behaviour. In this sense, several approaches estimate this error probability based on *ad hoc* methodologies and test the system under particular conditions [Milenkoski et al. 2015]. More recently, [Juba et al. 2015] propose a novel approach to evaluate alert correlation systems providing statistical guarantees on the measured performances. Although this is still an open research problem, even a rough estimate of this error probability can be useful to provide better risk assessments with BAGs.

Estimating the probability of having a zero-day vulnerability being successfully exploited by an attacker, p_{z_i} (to gain privileges in node X_i), is even more challenging that the previous case and goes beyond the intention of this paper, since it clearly requires a much more detailed treatment and modelling. However, we think that an approach similar to the estimation of the probability of successful exploitation of vulnerabilities, by means of CVSS scores, can be achieved by using the Common Weakness Scoring System (CWSS) [Common Weaknesses Scoring System 2014] which provide a quantitative measure of the unfixed weaknesses present in a software application. For example, in the *Environmental* metric we have the *Likelihood of Discovery* and the *Likelihood of Exploit* which estimate, respectively, the likelihood of an attacker discovering the weakness in the software and the likelihood that, if discovered, the attacker can successfully exploit them. Though these submetrics can be useful for the estimation of the probability of successful exploitation of zero-day vulnerabilities, there are some aspects that should be further considered. CWSS scores are associated with a software application, whereas the nodes in the BAG model represent security conditions. Thus, we need a mechanism to aggregate the scores from the software running in the machines associated to a particular security condition in the BAG. Moreover, we should also consider the preconditions that should be met before the attacker can exploit some weaknesses. This can be modelled by considering the *Attack Surface* metric in the CWSS score through the *Required Privilege* and *Required Privilege Layer* submetrics, which identify the type of privileges and the operational layer to which the attacker must have privileges before he can exploit the software weakness.

The estimation of the probability of an attacker successfully exploiting a zero-day vulnerability should be something particular to each node in the BAG, rather than setting a unique value for all the nodes. This is motivated from the differences in the software running in the different nodes of the network and because of the preconditions the attacker should meet before exploiting a zero-day vulnerability. Therefore, we can expect that the preconditions needed to compromised a node in the BAG close to the target are more difficult to obtain than the preconditions to compromise one of the initial nodes in the BAG. The reason is that these preconditions of nodes close to the target are usually postconditions of longer attack paths, so they should reflect the accumulated efforts along the paths. On the other hand, we can think of an extension of the traditional AG model to consider potential attack paths that can be followed by an attacker exploiting some unknown vulnerabilities by combining the analysis of software vulnerabilities with the reachability information. Then, we can consider some of the submetrics of CWSS scores, such as the *Required Privilege* and the *Required Privilege Level* to model the preconditions, and the *Acquired Privilege* and the *Acquired Privilege Level* (from the *Base Finding* metric) to model the postconditions. The inclu-

sion of these potential attack paths in the AG representation, which are aligned with the framework proposed in [Wang et al. 2014], is a matter for further consideration.

For the sake of simplicity, in the remaining of this paper we will consider that we already know the error probability of the IDS, p_e, and the probabilities of an attacker successfully exploiting a zero-day vulnerability for all the nodes in the BAG, p_{z_i}. Then, we can compute the leak p_l, used in (2) and (3) to model the uncertainty introduced by the non-perfect IDS and the presence of zero-day vulnerabilities, as:

$$p_{l_i} = 1 - (1 - p_e)(1 - p_{z_i}) \tag{6}$$

In [Poolsappasit et al. 2012] the authors suggest to introduce a leak to consider zero-day vulnerabilities, letting the system administrator to model this parameter, but they do not describe how to recompute the conditional probability tables using the leak. Moreover, using a global parameter to estimate the effect of zero-day vulnerabilities is not the best strategy. A different approach is proposed in [Xie et al. 2010], where a leak is introduced in the calculation of the conditional probability tables to model zero-day vulnerabilities. The non-perfect behaviour of the IDS is modelled by introducing an extra node in the BAG to model the observation of the current state of the node, which depends on the actual state of the node. Although the proposed model considers the two sources of uncertainty, the addition of one extra node for each node in the graph increases the complexity of the graph, which can limit the application of their model for large networks. In contrast, our approach does not introduce extra complexity in the structure of the BAG, but only considers a correction factor p_{l_i} when computing the conditional probability tables, as shown in Sect. 3.3.

To illustrate the effect of the leak in the computation of the conditional probability tables and the unconditional probabilities, in Fig. 1 and 2 we show a simple example with three security states representing that the attacker has root privileges on hosts 1 and 2 (nodes A and B in the figures) and the goal of the attacker is to gain user privileges in host 3 (node C). We consider that the probabilities of successful exploitation for the vulnerabilities that can be exploited from A and B to compromise C are 0.8 and 0.4 respectively. In Fig. 1 we consider the OR case: only one of the preconditions (A or B) is needed to compromise C, whereas in Fig. 2 we show the AND case: the two preconditions should be met before compromising C. In Tables I and II we detail the conditional probability tables for the OR and the AND cases respectively computed in the ideal case (perfect IDS and no zero-day vulnerabilities) and in the case where $p_e = 0.10$ and $p_z = 0.05$. In Tables I and II we can observe that in the real case $\Pr(C = \mathrm{T})$ is higher for all the entries in the table.

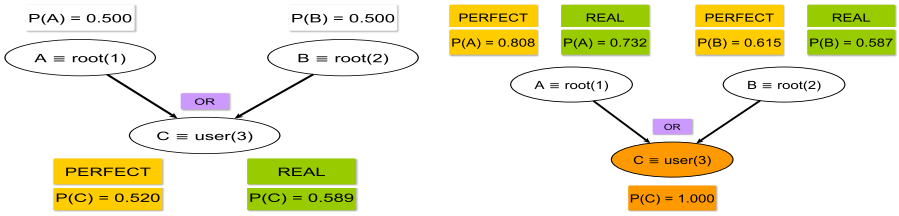
In the OR case, in Table I we appreciate that when both A and B are false (i.e. not compromised), there is still some probability that C can be compromised. This can be due to the presence of a zero-day vulnerability or because the IDS missed an intrusion. Similarly, in the AND case in Table II, when one of the preconditions is false, there is still some chance of having C compromised for the same reason.

In Fig. 1.(a) we show the unconditional probabilities for the perfect and the real case. As expected, the risk estimate in node C is higher in the real case. However, when we observe evidence of compromise in node C, as shown in Fig. 1.(b), the posterior probabilities on A and B for the perfect scenario are higher. This is due to the fact that there are two more possible causes for observing evidence of attack in C (apart of considering that A and B could have been compromised): an attacker exploited an unknown vulnerability or the IDS triggered a false alarm.

Similarly, for the AND case, the unconditional probability of C, shown in Fig. 2.(a), when no evidence of attack is observed, is higher for the real scenario, due to the potential presence of zero-day vulnerabilities or because the IDS missed some intrusions. In Fig. 2.(b), when evidence of compromise is observed in C, as in the OR case, the pos-

Table I. Conditional probability tables for and OR conditional probability table for the ideal scenario and a case where the error probability of the IDS is 0.10 and the probability of an attacker successfully exploiting a zero-day vulnerability is 0.05.

$\mathbf{Pa}_C$		Perfect case		Real case	
A	B	$\Pr(C = \mathrm{T})$	$\Pr(C = \mathrm{F})$	$\Pr(C = \mathrm{T})$	$\Pr(C = \mathrm{F})$
T	T	0.880	0.120	0.897	0.103
T	F	0.800	0.200	0.829	0.171
F	T	0.400	0.600	0.487	0.513
F	F	0	1	0.145	0.855



(a) (b)

Fig. 1. OR example: (a) Unconditional probabilities for the static analysis. (b) Posterior probabilities when there is evidence of attack at node C (the attacker gets user access at machine 3).

Table II. Conditional probability tables for and AND conditional probability table for the ideal scenario and a case where the error probability of the IDS is 0.10 and the probability of an attacker successfully exploiting a zero-day vulnerability is 0.05.

$\mathbf{Pa}_C$		Perfect case		Real case	
A	B	$\Pr(C = \mathrm{T})$	$\Pr(C = \mathrm{F})$	$\Pr(C = \mathrm{T})$	$\Pr(C = \mathrm{F})$
T	T	0.320	0.680	0.419	0.581
T	F	0	1	0.145	0.855
F	T	0	1	0.145	0.855
F	F	0	1	0.145	0.855

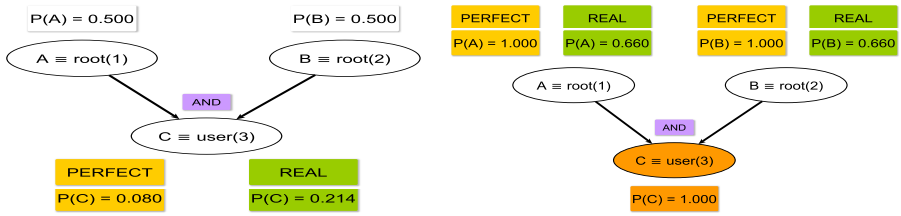
terior probabilities on A and B for the real scenario are lower for the same reasons than before.

Finally, in Fig. 3 we show the impact of the leak p_l computed in (6) from p_e and p_z on the unconditional probability on node C for the OR and AND cases. As we observe the probability of compromising C significantly increases in both cases when increasing p_e and p_z. Although in these simple examples, the increment of $\Pr(C)$ is nearly linear w.r.t. p_e and p_z, the combination of the effects produced by the leaks included in the calculation of the conditional probability tables can lead to a non-linear behaviour in more general BAGs, as we will show in Sect. 3.7. Despite this consideration, it is clear from Fig. 3 that the non-perfect behaviour of the IDS and the presence of zero-day vulnerabilities can have a significant impact on the assessment of the risk. Thus, the estimation of both p_e and p_z should be considered carefully, as we discussed before.

3.7. Example

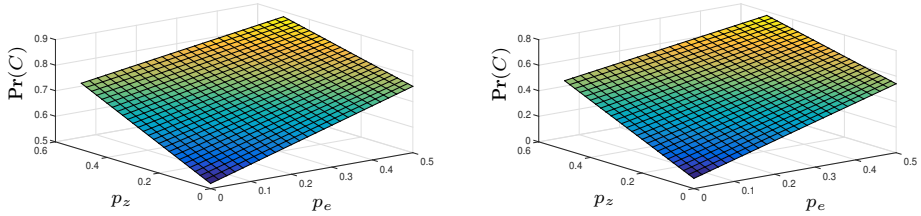
In this subsection we show an example of analysis of a BAG in the scenario depicted in Fig. 4, which represents a typical small corporate network. We will also use this example to illustrate and explain the inference algorithms described in Sect. 4 and 5.

In detail, for the network in Fig. 4 we have an internal LAN for corporate employees, and a DMZ hosting the company's servers, namely, a public Web server, a Mail server, and a local Database server (used to store public and private data). For each node,



(a) (b)

Fig. 2. AND example: (a) Unconditional probabilities for the static analysis. (b) Posterior probabilities when there is evidence of attack at node C (the attacker gets user access at machine 3).



(a) (b)

Fig. 3. Values for the unconditional probability in node C, $\Pr(C)$, as a function of the error probability of the IDS, p_e, and the probability of having zero-day vulnerabilities, p_z for (a) the OR case and (b) the AND case.

we have indicated the set of reachable ports, and from which other nodes they are reachable: this set includes those ports open/filtered by the firewall, as well as those open/closed by local firewalls, switches, routers, etc. In addition, we have highlighted some vulnerabilities that might be present on the network nodes. As an example, the Web server can be accessed on port 80 and port 43 by any other node (and also from the Internet), whereas it can be reached on port 22 (SSH server) only from the IP addresses belonging to the "Admin PC" node in the LAN[3]. Further, we suppose this node has a vulnerability affecting the SSH server (CVE-2015-6564). For each vulnerability, we show in Fig. 4 over which port it can be exploited (in case the vulnerability is a remote one), the CVE identifier, the type of vulnerability (DoS, elevation of privilege, etc.), and the likelihood of exploiting such a vulnerability. We have based this likelihood on the CVSS Exploitability Subscore, which we have divided by 10. In our particular example, when the corresponding score is 1.0, we have decided to lower the value to 0.95, since a probability of successful exploitation of 1.0 means that the attacker has already reached the next security state, without necessarily exploiting the vulnerability, which is not true. Finally, we suppose that a generic attacker exists that is willing to launch attacks from the Internet.

The BAG representation for this example is shown in Fig. 5. Following the guidelines described before, we have two nodes A_1 and A_2 that represent the initial attacker's state for the two possible initial attack paths, and the final objective of the attacker is to compromise the Database server (node F). With these settings, the joint probability for all the nodes in the BAG can be written as:

$$p(A_1, A_2, B, C, D, E, F) = p(A_1)\, p(A_2)\, p(B|A_1)\, p(C|A_2)\, p(D|B, C)\, p(E|C)\, p(F|D, E) \quad (7)$$

[3]If not explicitly indicated, it means that any other port is closed.

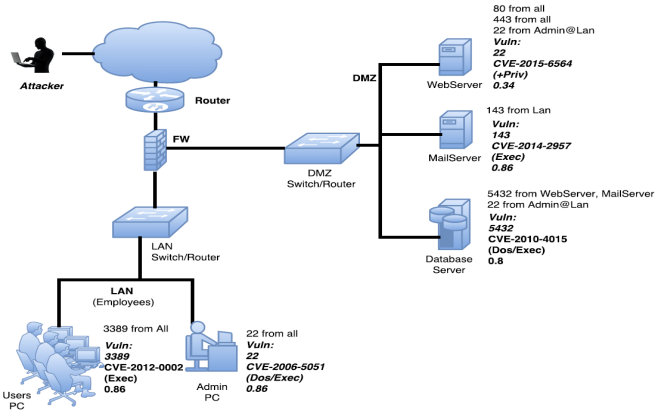


Fig. 4. Example of a Small Network with one LAN, a Mail server, a Web server, and a Database server.

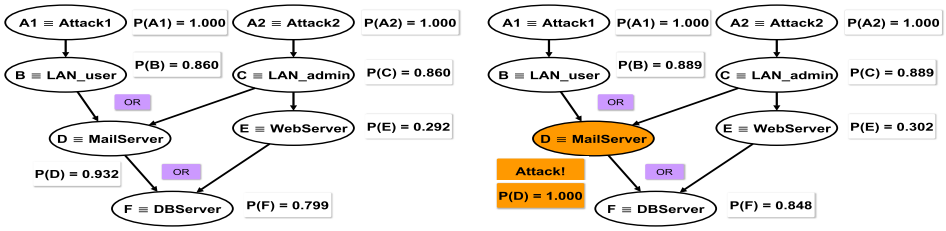


Fig. 5. BAG example for the network in Fig. 4: (a) Unconditional probabilities for the static analysis. (b) Posterior probabilities when there is evidence of attack at the Mail Server.

In Fig. 5.(a) we show the result of the static analysis, where we are interested in computing the unconditional probabilities when no evidence of attack is observed. In this case, we observe that the Database server can be compromised with a probability of 79.9% and that this high risk is due to the high probability of compromising the Mail server.

An example of dynamic analysis of the BAG is shown in Fig. 5.(b), where we consider that the alert correlation system triggers an alarm on the Mail server. For the sake of clarity, we have considered here a perfect behaviour of the alert correlation system. Fig. 5.(b) shows the posterior probabilities for all the network nodes given the evidence of compromise at node D. As expected, we observe that the risk of compromising the Database server increases to 84.8%. When reasoning about the potential attack path allowing the attacker to compromise the Mail server, the posterior probability of both nodes B and C is the same - in this example the paths are equally likely.

To illustrate the effect of the non-perfect behaviour of the IDS and the possibility of having zero-day vulnerabilities, in Fig. 6 we show the unconditional probabilities of compromising the Database server varying the error probability of the IDS, p_e, and the probability of an attacker successfully exploiting a zero-day vulnerability, p_z, in the interval $[0, 0.5]$. We have considered that the values of p_e and p_z are the same for all the nodes in the BAG. This simple example shows that both p_e and p_z can have a significant impact on the risk assessment. In our example the probability of compromising the Database server increases to 98.3% when $p_e = 0.5$ and $p_z = 0.5$, whereas the probability in a perfect scenario with $p_e = 0$ and $p_z = 0$ is 79.9%.

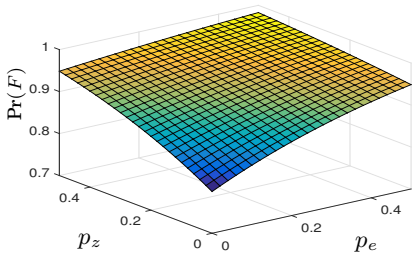
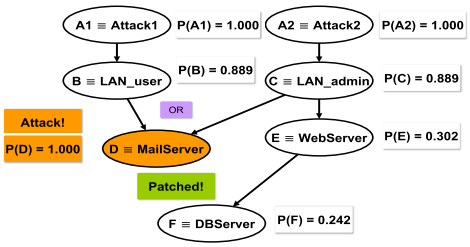


Fig. 6. Values for the unconditional probability in node F, $\Pr(F)$, as a function of the error probability of the IDS, p_e, and the probability of having zero-day vulnerabilities, p_z.



(a)

D	E	$\Pr(F = \mathrm{T})$	$\Pr(F = \mathrm{F})$
T	T	$(0.96 \rightarrow)\ 0.80$	$(0.04 \rightarrow)\ 0.20$
T	F	$(0.80 \rightarrow)\ 0.00$	$(0.20 \rightarrow)\ 1.00$
F	T	$(0.80 \rightarrow)\ 0.80$	$(0.20 \rightarrow)\ 0.20$
F	F	$(0.00 \rightarrow)\ 0.00$	$(1.00 \rightarrow)\ 1.00$

(b)

Fig. 7. (a) Updated posterior probabilities when the vulnerability from the Mail Server to the Database server has been patched in the BAG example in Fig. 5. (b) Updated conditional probability table for the Database server after patching the vulnerability (we show the previous values for each entry in the table in parentheses).

In Fig. 7.(a) we show the updated posterior probabilities when the system administrator has patched the vulnerability from the Mail Server to the Database Server. Thus, the corresponding attack path has been removed in the BAG representation. To recompute the posterior probabilities, we just need to update the conditional probability table for node F (the Database server) by considering that the probability of successful exploitation of the vulnerability from the Mail Server is zero. In Fig. 7.(b) we show the updated conditional probability table, which is equivalent to a conditional probability table where F depends only on node E (the Web server). Thus, D becomes a dummy variable for the computation of this conditional probability table, i.e. $p(F|D, E) = p(F|E)$. Finally, we recompute the posterior probabilities given the evidence of attack in the Mail Server with an exact or approximate inference algorithms, such as JT or LBP. As shown in Fig. 7, this countermeasure reduces significantly the risk of the Database server being compromised from 84.8% to 24.2%. In the next section we describe how these probabilities can be efficiently computed.

Although when patching a vulnerability we remove one or several attack paths from the AG, we suggest that, instead of recomputing the whole BAG model removing the corresponding attack paths, it is more efficient to set the probability of successful exploitation of the patched vulnerabilities to zero, which is equivalent to removing the attack paths from the BAG, since the conditional probability tables are equivalent. Thus, we do not recompute the whole model, but just update the corresponding parameters. This can be especially helpful when using JT for exact inference in BAGs, since we do not need to compute the clique tree again each time we patch a vulnerability, which in some cases, can be computationally expensive. On the other hand, our proposed approach copes with cases where a temporal countermeasure is applied,

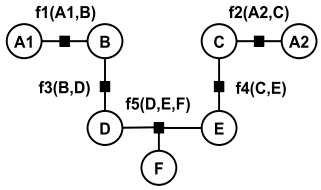


Fig. 8. Factor Graph representation for the BAG in Fig. 5 removing the attack path from the LAN admin to the Mail server.

e.g. the system administrator blocks part of the traffic or shut down some services temporarily to contain an ongoing attack.

4. SCALABLE INFERENCE ON BAYESIAN ATTACK GRAPHS

In this section we introduce two efficient techniques to make inference in BAGs: Belief Propagation (BP) and Loopy Belief Propagation (LBP). Both algorithms are based on message passing schemes that allow to compute the unconditional probabilities of the nodes in BNs or Markov Random Fields (MRFs). While BP is a method for exact inference, it is restricted to tree or polytree graph structures. On the contrary, LBP is an approximate inference technique that can be applied to graphs with loops.

4.1. Belief Propagation

As mentioned earlier, BP allows to compute the unconditional probabilities in a BN or a MRF when the graph is a tree or a polytree [Pearl 1982; 1988]. Although this structure is restrictive for AGs in general, BP can be applied for inference on Attack Trees [Schneier 1999]. Furthermore, for graphs with a general structure we can use the JT algorithm as an extension of BP, as we describe in Section 5. To describe the BP algorithm, we introduce the concept of *factor graphs* using a formulation similar to that given in [Bishop 2006]. BNs and MRFs allow to express the joint probability distribution of a set of random variables as a product of factors over subsets of those variables. Factor graphs make this factor decomposition explicit by introducing additional nodes for the factors themselves, in addition to those representing the variables, thus resulting in bipartite graphs.

Referring to the example shown in Fig. 5, if we remove the attack path from the LAN admin to the Mail server (for example, by patching the vulnerability), the resulting AG is a tree and we can use BP for inference. For this reduced BAG, the joint probability distribution can be factorised as:

$$p(A_1, A_2, B, C, D, E, F) = \prod_{i=1}^{5} f_i(\mathbf{X_i}) \tag{8}$$

where the corresponding factors $f_i(\mathbf{X_i})$ are:

$$\begin{aligned} f_1(A_1, B) &= p(A_1)\, p(B|A_1) & \qquad f_4(C, E) &= p(E|C) \\ f_2(A_2, C) &= p(A_2)\, p(C|A_2) & \qquad f_5(D, E, F) &= p(F|D, E) \\ f_3(B, D) &= p(D|B) \end{aligned} \tag{9}$$

Note that several factor graph representations may exist for a given BN (or MRF) [Bishop 2006]. However, the selection of a specific representation does not significantly impact the performance of BP. In our example, the corresponding factor graph according to (9) is shown in Fig. 8.

BP works by passing real valued functions, called messages, among the neighbouring nodes in the graph. Since factor graphs are bipartite, there are two possible types of messages: From variable to factor, and from factor to variable. The message from a variable X_i to a factor f_j in the neighbourhood of X_i is given by:

$$\mu_{X_i,f_j}(X_i) = \prod_{f_k \in \{\mathbf{F}_i - f_j\}} \mu_{f_k,X_i}(X_i) \tag{10}$$

where $\mu_{f_k,X_i}(X_i)$ are the messages from the factor nodes in the neighbourhood of X_i except f_j. On the other hand, the message from a factor node f_i to a variable node X_j in the neighbourhood of f_i is calculated as:

$$\mu_{f_i,X_j}(X_j) = \sum_{X_k \in \mathbf{X}_s} f_i(X_j, \mathbf{X}_s) \prod_{X_k \in \mathbf{X}_s} \mu_{X_k,f_j}(X_k) \tag{11}$$

where $\mathbf{X}_s$ is the set of variable nodes in the neighbourhood of f_i except X_j.

When a variable X_i is a leaf node, the corresponding messages to the factors in its neighbourhood are equal to one, i.e. $\mu_{X_i,f_j}(X_i) = 1$. On the contrary, if a factor f_i is a leaf node, the message to a variable node in its neighbourhood is given by $\mu_{f_i,X_j}(X_j) = \sum_{X_k \in \mathbf{X}_s} f_i(X_j, \mathbf{X}_s)$.

To compute the unconditional probabilities for all the nodes in the graph, BP needs to compute all the messages from all node variables to their corresponding factors and vice versa. BP proceeds starting from the leaf nodes (either variable or factor nodes) and propagates the messages across the graph such that a variable node X_i cannot send a message to a factor f_j until X_i receives all messages from its neighbouring factors except f_j. The same applies when sending messages from factors to variable nodes. For example, in the factor graph in (8), we cannot send a message from factor f_5 to variable F until f_5 receives a message from variables D and E.

Once all messages are computed, the unconditional probability for a node X_i, when the graph is a BN[4], can be calculated as:

$$p(X_i) = \prod_{f_j \in \mathbf{F}_i} \mu_{f_j,X_i}(X_i) \tag{12}$$

where $\mathbf{F}_i$ are the factor nodes in the neighbourhood of X_i. Therefore, BP can efficiently calculate all the marginal probabilities by computing all the messages once and storing them.

For the dynamic analysis, when we observe new evidence of compromise in some nodes, we only need to recompute the factors that depend on the variables that have changed in order to obtain the posterior probability on all the nodes of the network given the evidence of compromise. Further details are explained in [Koller and Friedman 2009]. Finally, the details about the computational complexity of BP will be discussed in Sect. 5 along with the corresponding discussion for the complexity of JT, as BP can be considered as a particular case of JT.

4.2. Loopy Belief Propagation

LBP is a simple extension of BP [Pearl 1988] applied to graphs (BNs or MRFs) that contain loops[5]. The difference is that, in the presence of loops, the results of LBP are approximate estimates of the unconditional probabilities of the nodes in the graph.

LBP uses the same factor graph representation as BP. In Fig. 9 we show the corresponding factor graph representation for the BAG depicted in Fig. 5, where we observe

[4]For MRFs we need to include a normalization factor.

[5]In this context a loop is an undirected cycle.

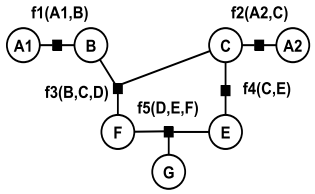


Fig. 9. Factor Graph representation for the BAG in Fig. 5.

that there is one loop due to the attack path from the LAN admin node to the Mail server. The corresponding factors $f_i(\mathbf{X_i})$ are the same as in (9) except for f_3, which in this case also depends on variable C, so that $f_3(B, C, D) = p(D|B, C)$.

There are two possible implementations of LBP according to how messages are computed, namely *Sequential LBP* (S-LBP) and *Parallel LBP* (P-LBP) [Murphy 2012] (they sometimes are also referred as asynchronous and synchronous LBP respectively). For S-LBP we iteratively compute the messages in (10) and (11) following some arbitrary schedule, until the unconditional probability estimates obtained with (12) converge or until a maximum number of iterations has been reached. Although there is no restriction on the order in which we update the messages (10) and (11), and the beliefs (12), depending on the structure of the graph, there are some scheduling techniques that can be applied to favour convergence and reduce the time to achieve it [Koller and Friedman 2009]. The detailed steps of the algorithm are shown in the Supplement.

In contrast, P-LBP updates all the messages for all factors and variable nodes at the same time, using the values of the messages at the previous iteration. Thus, at iteration t, we first update the messages from nodes to factors. The update equation for the message from a node X_i to a factor f_j can be written as:

$$\mu^{(t)}_{X_i,f_j}(X_i) = \prod_{f_k \in \{\mathbf{F}_i - f_j\}} \mu^{(t-1)}_{f_k,X_i}(X_i) \tag{13}$$

In a second step we update the messages from factors to variable nodes where the equation the message from factor f_i to node X_j is given by:

$$\mu^{(t)}_{f_i,X_j}(X_j) = \sum_{X_k \in \mathbf{X}_s} f_i(X_j, \mathbf{X}_s) \prod_{X_k \in \mathbf{X}_s} \mu^{(t)}_{X_k,f_j}(X_k) \tag{14}$$

Finally, we compute the new estimates of the marginal probabilities with (12) with the updated messages obtained using (14). As in the previous case, the algorithm is repeated until the unconditional probabilities converge or a maximum number of iterations is reached. The details of P-LBP are shown in the Supplement.

For the first iteration in both S-LBP and P-LBP we initialize the messages from nodes to factors to 1. The messages from a factor f_i to a node X_j are initialized as $\mu_{f_i,X_j}(X_j) = \sum_{X_k \in \mathbf{X}_s} f_i(X_j, \mathbf{X_s})$. The details of this procedure can also be found in the Supplement.

According to [Koller and Friedman 2009], S-LBP usually works better than P-LBP, although they require scheduling messages in a guided way. However, the experimental results on synthetic BAGs, presented in Sect. 6, show a similar behaviour for both implementations in terms of accuracy.

In Fig. 10 we show the estimates for the unconditional probabilities in the BAG depicted in Fig. 5.(a) when there is no evidence of attack. It can be observed that there is no difference (at least in the 3 first decimals) between the true and the estimated prob-

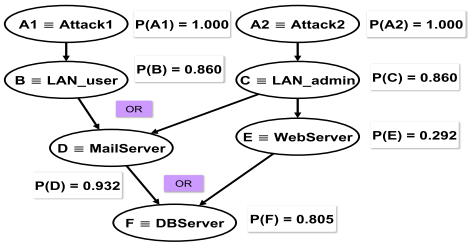


Fig. 10. Estimation of the unconditional probabilities provided by LBP for the BAG in Fig. 5.(a)

abilities for nodes A-E and that, for node F, the marginal probability estimated with LBP[6] is 0.805 while the true probability is 0.799. When evidence of attack is observed at the Mail server (node D), shown in Fig. 5.(b), the LBP posterior probability estimates match the exact probabilities. The reason for this exact result is that, given the evidence of attack, D can be considered as a deterministic node. This splits the graph into two tree structures with the remaining unobserved nodes: $\{A_1, B\}$ and $\{A_2, C, E, F\}$ and the message passing scheme produces exact results for the two trees.

One of the drawbacks of LBP is that, in general, convergence is not guaranteed. [Weiss 2000] show that LBP converges for graphs with a single loop and derives an analytical relationship between LBP probability estimates and the true unconditional probabilities. For more general graphs, [Mooij and Kappen 2005; Ihler et al. 2005] present sufficient conditions on the convergence of LBP based on the concept of α-contractions. However, applying the corresponding analysis is, in general, a difficult task [Koller and Friedman 2009]. It is also important to note that convergence does not mean correctness, i.e. LBP convergence does not imply that the unconditional probability estimates are accurate. However, the empirical study in [Murphy et al. 1999] shows that usually, when LBP converges, the approximate marginal probabilities are close to the exact values.

One simple way to favour convergence is to use *damping*. Hence, the update of the messages from a variable node X_i to a factor f_j have the form:

$$\hat{\mu}^{(t)}_{X_i,f_j}(X_i) = \alpha\mu^{(t)}_{X_i,f_j}(X_i) + (1-\alpha)\mu^{(t-1)}_{f_k,X_i}(X_i) \tag{15}$$

so that the damped message $\hat{\mu}^{(t)}_{X_i,f_j}(X_i)$ is a convex combination of the message update at iteration t and the message at iteration $t-1$, where the damping factor α is a positive value smaller than 1. The damped update for the messages from factors to variable nodes is analogous. This technique can be applied for both S-LBP and P-LBP. In Sect. 6, we analyse the effect of damping for the convergence and accuracy of LBP.

It is also possible to make LBP converge to a local minimum using double loop algorithms [Yuille 2001; Welling and Teh 2001]. Unfortunately, these techniques are slow and complicated and their accuracy is often worse than the standard LBP [Murphy 2012], since they are prone to converge to poor local minima. Similarly, other techniques, such as the mean field approximation, have been proved to converge but are usually less accurate than LBP, since the non-convexity of the mean field objective function leads to poor solutions [Weiss 2001].

5. EXACT INFERENCE IN BAYESIAN ATTACK GRAPHS: JUNCTION TREE

In this section we review the JT algorithm, the state-of-the-art technique for the static and dynamic analysis of BAGs scaling to graphs with hundreds of nodes [Muñoz-

[6]In this case both S-LBP and P-LBP provide the same result.

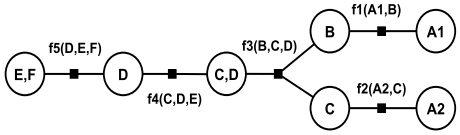


Fig. 11. Factor Graph representation for clique tree obtained from the BAG in Fig. 5.

González et al. 2016]. Although other exact inference techniques exist, as the Variable Elimination (VE) proposed in [Liu and Man 2005], JT is computationally more efficient than VE for the analysis of BAGs, as shown in [Muñoz-González et al. 2016]. In Sect. 6 we use JT as benchmark for the comparison with LBP.

The JT or clique tree algorithm is an extension of BP, for exact inference, that can be applied on BNs or MRFs with a general structure. In this case, BP's message scheme is applied to a tree structure where the nodes represent clusters of the random variables in the graph. Similar to [Muñoz-González et al. 2016], we will describe the Shenoy-Shafer method [Shenoy and Shafer 1990; Shafer and Shenoy 1990] method, which uses the same message passing scheme as BP.

The first step of JT is to create a cluster graph with a tree structure from the initial BN (or MRF). This cluster graph (or clique tree) can be considered an extension of factor graphs with clusters of several random variables between two factors. In this case, one random variable can appear in more than one cluster node. However, the cluster graph needs to satisfy the *running intersection property*: if a random variable X_i appears in two cluster nodes, $X_i \in C_j$ and $X_i \in C_k$, then X_i also appears in each cluster node in the unique path existing between C_j and C_k in the clique tree.

In the case of BNs, to create a clique tree we first need to moralize the graph, i.e. make the graph undirected and add a link between the nodes that have a common child (this step is not needed for MRFs). The moral graph is then triangulated to obtained a chordal graph, i.e. one in which every minimal loop in the graph is of length three. Each maximal clique in the chordal graph is a cluster node in the clique tree. As shown in [Koller and Friedman 2009], we can create the cluster graph with the VE algorithm.

For the BAG example in Fig. 5, we show in Fig. 11 a factor graph representation of the corresponding clique tree obtained using VE. For this factor graph representation, the assignment of the different terms in (7) to the factors in Fig. 11 is:

$$\begin{aligned} f_1(A_1, B) &= p(A_1)\, p(B|A_1) & \qquad f_4(C, D, E) &= p(E|C) \\ f_2(A_2, C) &= p(A_2)\, p(C|A_2) & \qquad f_5(D, E, F) &= p(F|D, E) \\ f_3(B, C, D) &= p(D|B) \end{aligned} \tag{16}$$

With the factor graph representation of the clique tree we can calculate the unconditional probabilities using the same message passing scheme as in BP. The difference is that, in this case, the scopes of the messages given in equations (10) and (11) depend on multiple random variables rather than just one, as in the case of BP.

Once all the messages are computed, the unconditional joint probability for the variables in a cluster node $\mathbf{X}_s$ (provided that the graph is a BN) is calculated as:

$$p(\mathbf{X}_s) = \prod_{f_j \in \mathbf{F}_s} \mu_{f_j, \mathbf{X}_s}(\mathbf{X}_s) \tag{17}$$

where $\mathbf{F}_s$ are the factor nodes in the neighbourhood of the cluster node $\mathbf{X}_s$. To calculate the marginal probability for one random variable X_i in the cluster $\mathbf{X}_s$, we just sum over

the rest of the variables in $\mathbf{X}_s$:

$$p(X_i) = \sum_{X_j \in \{\mathbf{X}_s - X_i\}} p(\mathbf{X}_s) \tag{18}$$

Evidence of compromise can be easily included when using JT, in the same way as in BP. Further details can be found in [Koller and Friedman 2009]. The computational complexity of JT is exponential in the scope of the biggest factor in the clique tree. Concretely, if all the variables in the graph are discrete and have K possible values each (in our case, $K = 2$), JT scales in time and space as $\mathcal{O}(|F|K^s)$, where $|F|$ is the number of factors and s is the size of the scope of the largest factor in the clique tree (3 in the example in Fig. 11). Moreover, the computational complexity of applying VE algorithm to build the clique tree is also exponential. This can limit the application of JT for large graphs, although it depends on the structure of the graph, as we will show in the experiments. The scalability for BP and LBP is similar to that of JT, i.e. they scale in time and space as $\mathcal{O}(|F|K^s)$. However, since BP and LBP do not cluster variables, the factors are usually smaller, so we expect to have smaller s.

6. EXPERIMENTS

In this section we present an experimental evaluation comparing the accuracy and performance of sequential and parallel implementations of LBP with that of the JT algorithm used in [Muñoz-González et al. 2016]. In detail, we first analyse the accuracy of LBP when computing the unconditional and posterior probabilities for the static and dynamic analysis of AGs, and then compare the time required to estimate these probabilities with that required by JT. This allows us to determine if LBP is a suitable alternative for tractable analysis of large AGs, and if the risk estimates are sufficiently accurate to help administrators propose risk mitigation strategies.

For JT we have used VE to build the clique tree, selecting the elimination ordering according to the *min-weight* heuristic [Koller and Friedman 2009]. As shown in [Muñoz-González et al. 2016], the elimination ordering has an impact on the performance of the algorithm, reducing the memory required and the time to compute the unconditional probabilities. For S-LBP and P-LBP we have used a tolerance threshold of 10^{-3} for the convergence of the algorithm, i.e. we assume that the algorithm has converged if the biggest change in the unconditional probabilities is less than 10^{-3}. We have used the Bayes Net toolbox for Matlab[7][8] for all the algorithms.

6.1. Synthetic Attack Graph Generation

To provide a comprehensive evaluation with different graph sizes, network topologies, and interdependencies, we have generated synthetic AGs in the experiments. Note that, currently, there are no collections of AGs of similar variety obtained empirically from real systems; in fact, no collections of empirically obtained AGs exist in the public domain at all. Furthermore, from the examples reported in the literature it is hard to determine the typical graph structure of AGs, e.g. for large corporate networks. We expect these graph structures to vary significantly since they depend on the network topology and the distribution and type of vulnerabilities across the network components. For these, reasons, we have used the structures proposed in [Muñoz-González et al. 2016] to generate the synthetic AGS: *pseudo-random graphs* and *cluster graphs*. Note that despite the dependency of the AG on the network topology, the use of Internet topology generators is not adequate to generate synthetic AGs, since they do

[7]https://github.com/bayesnet/bnt

[8]Our code implementation with the experiments for the BAG model can be found at http://rissgroup.org/

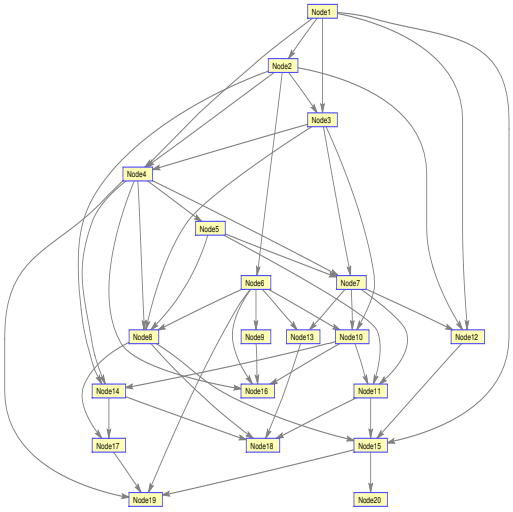


Fig. 12. Example of random BAG with 20 nodes and $m = 4$.

not consider the security aspects of the network but only the topology. On the other hand, Internet topologies can be fairly different to traditional corporate network deployments.

The *pseudo-random graphs* are random DAGs where we only limit the maximum number of parents that a node can have. This corresponds to restricting the maximum number of vulnerabilities that can lead an attacker to reach a certain security condition. Since in most real scenarios there is some security in place, we expect to have a reduced number of vulnerabilities that allows an attacker to compromise a network node [WhiteHat Security 2015; Symantec 2015]. We consider that this assumption is reasonable from a practical perspective. Thus, for each node in the graph X_i, we randomly select its number of parents by drawing a random integer n_p in the interval $[1, m]$ uniformly, where m is the maximum number of possible parents allowed. Then, we randomly select the n_p parent nodes for X_i from the set of nodes in the BAG for which X_i is not a parent node already. This avoids directed cycles and preserves the DAG structure. In Fig. 12 we show an example of a random BAG with 20 nodes and $m = 4$. Since we are not assuming anything about the structure of the graph other than the maximum in-degree of the nodes, there is no advantage for the inference algorithms used in the experiments to reduce the computational complexity, i.e. these graph structures can be representative of worse-case scenarios.

However, typical corporate networks are structured into subnetworks [Tan et al. 2003] and contain several hosts with common software installations so we can expect some form of cluster structure in the corresponding AG. Moreover, we expect a reduced number of vulnerabilities allowing the attacker to escalate privileges across different subnetworks. Routing and firewall rules between subnetworks usually hinder the progression of the attack. Then, the connectivity between subnetworks in the AG is expected to be weak. This argument is in line with the AGs examples shown in [Jajodia et al. 2011], where only attacks across subnetworks are considered. To generate this kind of graphs we have considered networks with clusters of the same size n_c. Then, for each cluster, following the same procedure as before, we have generated pseudo-random subgraphs, limiting the maximum number of parents for each node to m. Finally, to include the dependencies between clusters, we have added one edge from

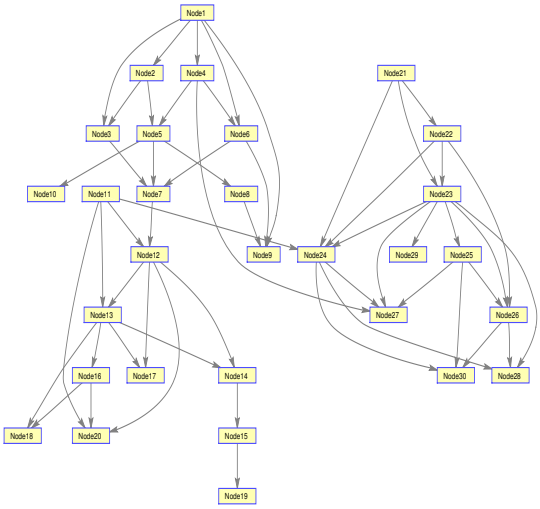


Fig. 13. Example of clustered BAG with 3 subnetworks and 10 nodes per subnetwork.

one node in each cluster to one node in each of the other clusters, provided that the DAG structure required for BNs is preserved. For our experiments we have generated synthetic clustered graphs with $n_c = 20$ and 50, varying the total number of network nodes from 100 to 1000. In Fig. 13 we show an example of a clustered BAG with 3 subnetworks, with 10 nodes per subnetwork, and $m = 3$.

The values for the probabilities of successful exploitation of vulnerabilities are drawn at random from the distribution of CVSS scores extracted from [cvs 2016]. We normalize the scores dividing them by 10. In Fig. 14 we show the distribution of CVSS scores. The value of these probabilities can have an impact in the accuracy of the unconditional probability estimates, and in the convergence of the algorithms. Although we think that the exploitability submetric of CVSS scores is a better indicator of the difficulty of exploiting a vulnerability, it is difficult to get the distribution of this subscore, and so we have used the distribution of the whole CVSS score instead. Finally, since we do not have data to estimate the error probability of the alert systems, and their accuracy changes in time and with the topology of the network, we have considered in our experiments that the alert system does not trigger false alarms. For the same reason, we do not also consider the possible presence of zero-day vulnerabilities. In any case, these factors does not affect the computational complexity of the algorithms used in the experiment, although the effect in the values of the unconditional and posterior probabilities can be affected, as shown in Sect. 3.6 and Sect. 3.7.

For the experimental evaluation we start analysing the accuracy, convergence, and scalability of S-LBP and P-LBP for the pseudo-random AGs. Then, following a similar treatment we present the experimental results on cluster AGs.

6.2. BAGs with pseudo-random structure

6.2.1. Accuracy and Convergence. In our first experiment we want to measure the accuracy and the convergence of both S-LBP and P-LBP for pseudo-random AGs. We have therefore generated synthetic AGs with 40 nodes and $m = 3$, where we have varied the proportion of OR and AND conditional probability tables. We have also explored different values for the damping factor α in the range $[0, 0.5]$. We have measured the accuracy using the Rooted Mean Squared Error (RMSE), comparing LBP estimates

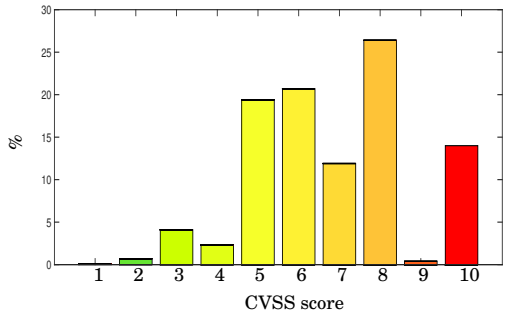


Fig. 14. Distribution of the CVSS scores [cvs 2016].

Table III. Average RMSE plus/minus one standard deviation for P-LBP on pseudo-random BAGs with $m = 3$ and 40 nodes varying the damping factor α and the probability of having AND-type conditional probability tables, p(AND).

p(AND)	$\alpha = 0.0$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
0.0	0.0315 ± 0.0223	0.0272 ± 0.0187	0.0291 ± 0.0194	0.0266 ± 0.0175
0.2	0.0218 ± 0.0141	0.0222 ± 0.0143	0.0266 ± 0.0206	0.0276 ± 0.0183
0.5	0.0182 ± 0.0103	0.0150 ± 0.0067	0.0165 ± 0.0123	0.0175 ± 0.0083
0.8	0.0081 ± 0.0069	0.0108 ± 0.0097	0.0131 ± 0.0100	0.0112 ± 0.0064
1.0	0.0081 ± 0.0058	0.0069 ± 0.0047	0.0107 ± 0.0096	0.0089 ± 0.0085

with the exact unconditional probabilities provided by JT. For each combination of parameters explored we have averaged the RMSE obtained for 20 independent BAGs. From the results in Tab. III we observe that the RMSE is less than 0.03 in most cases, which is a reasonable accuracy to estimate the risks of compromising the different nodes in the AG, especially when considering that the probabilities of successful exploitation of vulnerabilities are not accurate, since they are estimated with the CVSS scores. Thus, the accuracy of LBP is enough to allow system administrators to decide the actions that need to be taken (both for the static and the dynamic analysis of the network). It is also interesting to observe that a little bit of damping, i.e. small values of α, slightly improve the accuracy. Furthermore, it is remarkable that the RMSE is lower when the proportion of AND-type conditional probability tables is higher. This effect is due to the different coupling effect between the variables in the loops of the BAG depending on the type of conditional probability table. In Tab. III we only show the RMSE for P-LBP, since the differences w.r.t. S-LBP were negligible. Moreover, we have also observed that both implementations of LBP converged in all cases.

6.2.2. Accuracy with the Number of Iterations. In our second experiment we have analysed how the accuracy of the LBP probability estimates changes with the number of iterations. LBP allows to monitor the intermediate estimates of the unconditional probabilities, which is not possible with JT. This can help system administrators to reduce the response time to an attack, since they do not need to wait until the algorithm has completely converged.

In Fig. 15 we show the average RMSE of P-LBP and S-LBP as a function of the number of iterations for 25 pseudo-random BAGs with 100 nodes and $m = 3$. The probability of having AND-type conditional probability tables has been set to 0.5, and we have used a damping factor of $\alpha = 0.2$ in both LBP implementations. From the results in Fig. 15 we can observe that the algorithms converge on average in less than 15 iterations, although P-LBP seems to converge faster, and gets better estimates of the unconditional probabilities after the first iteration (although the final result is similar

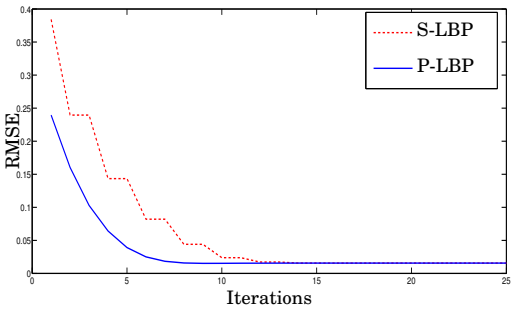


Fig. 15. Average RMSE of P-LBP and S-LBP with the number of iterations for 25 pseudo-random BAGs with 100 nodes and $m = 3$.

to S-LBP). From Fig. 15 it is important to note that, after 5 iterations, the RMSE for P-LBP is about 0.05. This accuracy can be considered reasonable to start planning risk mitigation strategies at run-time without waiting for LBP to converge.

6.2.3. Time Scalability. Our last experiment with pseudo-random BAGs is aimed at evaluating the time scalability of LBP and JT for the static and dynamic analysis of AGs. We have analysed networks with different sizes and different densities: For m we have explored the values 3 and 4, while for n, the number of nodes in the BAG, we have used values in the range $[20, 3000]$. However, for JT we have limited the value of n to 120 for $m = 3$ and to 80 for $m = 4$ because of physical memory limitations[9]. The probability of having AND-type conditional probability tables is set to 0.5. For each value of n and m we have generated 20 pseudo-random BAGs and, for each BAG, we compute the unconditional probabilities for all the nodes with both LBP and JT.

In Fig. 16 we show the average time required to compute the unconditional probabilities for all the nodes in the BAG for P-LBP, S-LBP, and JT. In the case of JT, the measured time includes the time required to build the clique tree, and compute all the messages and all the unconditional probabilities. For LBP, this time considers the computation of all the messages and the probability estimates. Therefore, we are, in essence, measuring the time required to perform the static analysis of the BAG. For both LBP variants we use $\alpha = 0.2$ and set the maximum number of iterations to two times the number of nodes in the graph. As in the previous experiment, both LBP implementations converged in all cases.

From the results shown in Fig. 16.(a) we can observe the exponential increase of JT with the number of nodes, whereas both P-LBP and S-LBP scale linearly. Although the time to compute the unconditional probabilities by JT is lower for small AGs (less than 100 nodes), it appears that LBP is a suitable alternative to make inference in large BAGs, where the exponential scalability of JT makes its use impractical. It is also interesting to note that P-LBP is faster than S-LBP. Moreover, when increasing the complexity of the network (by increasing m) the performance of P-LBP remains similar, whereas we can observe larger differences for S-LBP.

For the dynamic analysis of AGs, when we observe evidence of compromise in some nodes, we need to recompute all the messages taking into account the evidence as well as the posterior probabilities in all the graph nodes. This applies both to JT and LBP. However, for JT, we do not need to build the clique tree again. So, to analyse the performance of the algorithms for the dynamic analysis of AGs, we have randomly selected 3 nodes in each graph where we consider that evidence of attack has been observed.

[9]The experiments have been conducted in a 16 GB computer with an Intel Core i7 processor at 3.40 GHz.

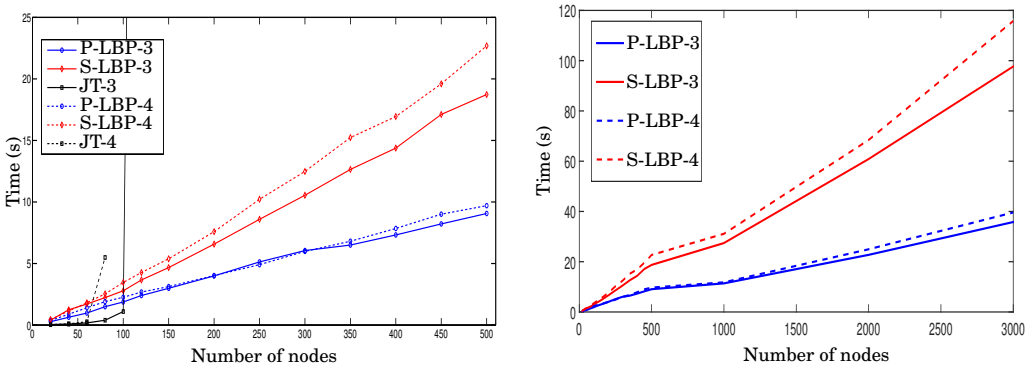


Fig. 16. (a) Average time to compute the unconditional probabilities for P-LBP, S-LBP, and JT for pseudo-random attack graphs. (b) Extended results for P-LBP and S-LBP for BAGs up to 3000 nodes. The notation P-LBP-m, S-LBP-m, and JT-m stands for the value of m, the maximum number of parents allowed for each node, used to generate the graphs in each case.

Then, we have measured the time required to recompute the posterior probabilities for P-LBP, S-LBP, and JT. We have omitted the resulting figure, since the results are very similar to those obtained for the static analysis (in Fig. 16). The measured times for the dynamic analysis are slightly lower for all the methods and the differences are not very significant. In the case of the JT, this means that for this kind of graphs the bottleneck of the algorithm is the computation of the messages rather than the time required to build the clique tree [Muñoz-González et al. 2016]. This is due to the strong interconnection of the nodes in the graph, which makes some clusters in the clique tree to have a high number of variables. For LBP implementations, the similarity of the results suggests that the number of iterations needed to converge are similar. In Fig. 16.(b) we show the extended experiments for LBP using pseudo-random BAGs up to 3000 nodes. We observe that for both LBP implementations linear scalability still holds. The linear scalability of LBP for this kind of graphs make it useful for both static and dynamic analysis of AGs, especially when the graphs are large. It is thus possible to analyse AGs with thousands of nodes, which can correspond to networks with tens or hundreds of thousands of nodes (depending on the number of vulnerabilities).

6.3. BAGs with cluster structure

6.3.1. Accuracy and Convergence. For this experiment we have generated synthetic cluster AGs with 5 clusters with $n_c = 20$ nodes per cluster and $m = 3$. As for the pseudo-random BAGs, we have varied the proportion of OR and AND conditional probability tables and we have explored values for the damping factor α in the range $[0, 0.5]$. For each combination of parameters explored we have averaged the RMSE obtained for 20 independent BAGs. The results are shown in Tab. IV, where we observe that the average RMSE is less than 0.04 in all cases, which means that the probability estimates provided by LBP are reasonable to perform risk assessment with BAGs. We can also appreciate that the proportion of AND/OR conditional probability tables has some impact on the accuracy of the probability estimates. Thus, having a bigger proportion of AND-type conditional probability tables results in more accurate estimates. In this case, in contrast to the results shown in Tab. III, damping does not provide any clear improvement in the accuracy. However, even in these cases, it is a good practice to include a little bit of damping in LBP updates to avoid potential instabilities in the

Table IV. Average RMSE plus/minus one standard deviation for P-LBP on clustered BAGs with $m = 3, 5$ subnetworks and $n_c = 20$ nodes per subnetwork, varying the damping factor α and the probability of having AND-type conditional probability tables, p(AND).

p(AND)	$\alpha = 0.0$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
0.0	0.0342 ± 0.0213	0.0376 ± 0.0213	0.0341 ± 0.0157	0.0337 ± 0.0186
0.2	0.0216 ± 0.0078	0.0247 ± 0.0105	0.0289 ± 0.0154	0.0229 ± 0.0123
0.5	0.0185 ± 0.0062	0.0219 ± 0.0128	0.0193 ± 0.0083	0.0224 ± 0.0091
0.8	0.0149 ± 0.0042	0.0136 ± 0.0035	0.0138 ± 0.0078	0.0127 ± 0.0053
1.0	0.0109 ± 0.0042	0.0115 ± 0.0039	0.0093 ± 0.0043	0.0107 ± 0.0038

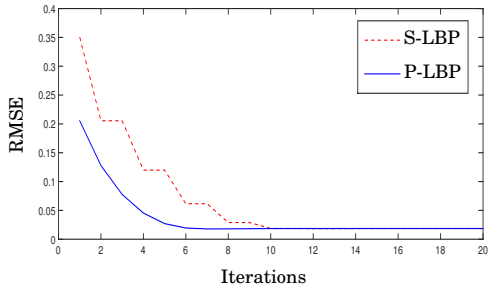


Fig. 17. Average RMSE of P-LBP and S-LBP with the number of iterations for 25 cluster BAGs with 100 nodes (5 subnetworks with 20 nodes per subnetwork) and $m = 3$.

algorithm. As in the case of the pseudo-random BAGs, we only show the RMSE for P-LBP, since the results obtained from S-LBP are very similar. Finally, we have also observed that both, P-LBP and S-LBP, converged in all cases.

6.3.2. Accuracy with the Number of Iterations. In Fig. 17 we show how the average RMSE of P-LBP and S-LBP decreases with the number of iterations. For these experiments we have generated 25 cluster BAGs with 5 clusters and $n_c = 20$ nodes per cluster, with $m = 3$. As in the case of the pseudo-random BAGs, we have set the probability of having AND-type conditional probability tables to 0.5 and we have used a damping factor of $\alpha = 0.2$.

The results in Fig. 17 are very similar to those shown in Fig. 15 for the case of the pseudo-random BAGs. Thus, we can observe that after 4 iterations, P-LBP produce estimates for the unconditional probabilities with an average RMSE less than 0.05. We also appreciate that, as in the previous case, P-LBP converges faster than S-LBP, although both techniques achieve a similar approximation error after convergence. The result of this experiment shows that, after a few iterations, the accuracy provided by LBP can be considered reasonable to start planning risk mitigation strategies at run-time before LBP converges.

6.3.3. Time Scalability. We report in Fig. 18.(a) the time required to perform the static analysis for P-LBP, S-LBP, and JT. As in the case of the pseudo-random networks we observe that JT scales exponentially with the number of nodes, although it is able to perform static analysis on larger networks (compared to the pseudo-random AGs). In contrast, both LBP implementations scale linearly with the number of nodes in the BAG, and require less time than JT to compute all the unconditional probabilities for graphs with more than 500 nodes. Also in line with the previous experiment, P-LBP shows a better performance than S-LBP, although the difference is not as significant as before. We can also appreciate that, for both LBP methods, the increment on the size of the clusters implies only a small difference in the time performance of the algorithms.

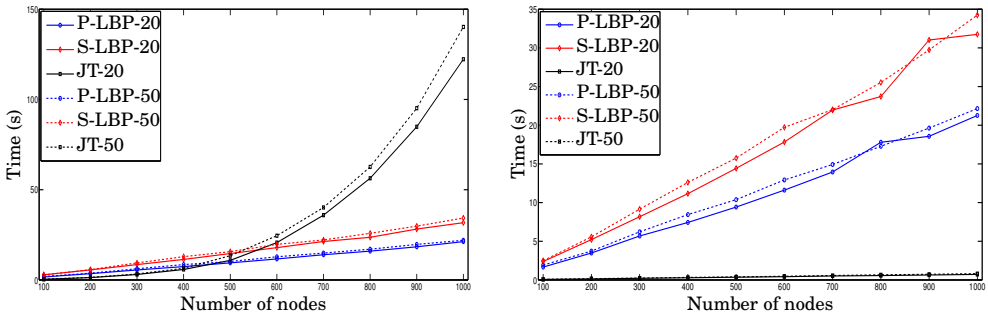


Fig. 18. Time to compute the unconditional probabilities for P-LBP, S-LBP, and the JT algorithm for cluster networks with different cluster sizes ($n_c = 20$ and 50) and $m = 3$ for: (a) the static analysis and (b) the dynamic analysis (when we observe evidence of attack at 3 random nodes).

In Fig. 18.(b) we show the time required to perform the dynamic analysis when we observe evidence of attack in 3 nodes chosen at random. We can observe again that the scalability for both LBP implementations is linear in the number of nodes, and that the time required to compute all the unconditional probabilities in the BAG is slightly lower than in the case of the static analysis in Fig. 18.(a). However, in this case the performance of JT is also linear in the number of nodes and the time required to compute the unconditional probabilities is much lower than in the case of two LBP methods.

These results indicate that, when JT is applied on clustered networks, the bottleneck is the generation of the clique tree, while the computation of the messages is simple. This suggests that the cluster structure of the BAGs produces clique trees with a reduced number of variables in each cluster of the tree. Hence, the messages sent across nodes involve a reduced number of random variables, allowing a fast calculation of the posterior probabilities for the dynamic analysis of the BAGs. In the case of LBP, the messages are simpler than in the case of JT. However, the time to compute the unconditional probabilities in the dynamic analysis is higher since, at each iteration, we need to compute all the messages and several iterations are needed to make the algorithm converge. Although JT is faster than LBP for the dynamic analysis of clustered BAGs, the exponential scalability for the static analysis, when the clique tree is generated, limits the tractability of the analysis for large networks. On the other hand, as mentioned before, LBP allows to monitor the values of the posterior probabilities at each iteration, so that we can obtain accurate estimates for the probabilities before the algorithm converges.

7. CONCLUSIONS

Bayesian Networks are a powerful tool for the static and dynamic analysis of AGs: the unconditional probabilities at each node in the graph (not only for the target) provide useful information to system administrators for security risk assessment and mitigation. The values of these probabilities take into account the dependencies between the different attack paths and the difficulty of exploiting each vulnerability, in contrast to other measures proposed in the literature to perform risk assessment in AGs. However, the scalability problems of the exact inference techniques proposed in the literature for the static and dynamic analysis of BAGs limit the applicability of these techniques to graphs with a few hundreds of nodes, far from the size of the AGs we can expect in large corporate networks.

In this paper we have shown that LBP, an approximate inference technique, can be used effectively for the static and dynamic analysis of large BAGs. We have verified through experiments that the reduction in the computational cost and memory requirements is significant. Moreover our experimental evaluation shows that *LBP scales linearly with the number of nodes for both the static and the dynamic analysis*. Overall these results show significantly better performance than the techniques proposed for the analysis of BAGs in the literature so far. We have experimented with synthetic AGs with a broad variety of topologies in an effort to ensure the applicability of the technique to many network deployments. The gains in scalability are obtained at the price of a loss in accuracy on the probabilities. We have however verified through experiments that this accuracy loss is very low with an average RMSE of less than 0.03, especially taking into account that the values for the probability of successful exploitation of the vulnerabilities, used to build the BAG models, are non-accurate estimates. We have also shown through the experiments that LBP compares favourably with the JT, the state of the art technique for exact inference in BAGs [Muñoz-González et al. 2016]. Although a significant amount of literature on the application of AG exists, few studies have considered the computational aspect of making inference on them. Furthermore, the lack of scalability has significantly hindered their application. Our results show that by using the right techniques, both static an dynamic analysis can be performed on AGs with thousands of nodes, even on a standard laptop computer. We have also evaluated the effect of clustering on the performance of the analysis and shown that this can lead to further significant gains in performance. Different approaches to allow for tractable analysis of AGs can include lower resolution representations of the AG, by for example representing similar security conditions (from similar network nodes) as a single node. High level abstractions of the original AG can also be considered, similar to the framework proposed by [Noel and Jajodia 2004] to manage AG complexity for visualization. These approaches applied to risk assessment induce a trade-off between the accuracy, the level of resolution, and the tractability of the analysis. However, the estimation of the risks associated to these aggregated nodes requires special consideration.

Our future research plans include modelling the attacker's capabilities to estimate the probability of successful exploitation of vulnerabilities, and the use of Bayesian inference techniques to help prioritizing forensic investigation using AGs.

ACKNOWLEDGMENTS

The authors would like to thank British Telecom for their collaboration in this research and our colleagues in our research group for their contribution to this work through many useful discussions. This research has been funded by the UK government under EPSRC project EP/L022729/1.

REFERENCES

2016. CVE Details. The ultimate security vulnerability datasource. http://www.cvedetails.com. (2016).

M. Albanese, S. Jajodia, and S. Noel. 2012. Time-Efficient and Cost-Effective Network Hardening using Attack Graphs. In *Int. Conf. on Dependable Systems and Networks*. 1–12.

M. Albanese, S. Jajodia, A. Pugliese, and V.S. Subrahmanian. 2011. Scalable Analysis of Attack Scenarios. In *European Symposium on Research in Computer Security*. 416–433.

P. Ammann, D. Wijesekera, and S. Kaushik. 2002. Scalable, Graph-Based Network Vulnerability Analysis. In *Procs. Conf. on Computer and Communications Security*. 217–224.

F. Baiardi and D. Sgandurra. 2013. Assessing ICT Risk through a Monte Carlo Method. *Environment Systems and Decisions* 33, 4 (2013), 486–499.

L. Bilge and T. Dumitras. 2012. Before We Knew It: An Empirical Study of Zero-Day Attacks in the Real World. In *Procs. Conf. on Computer and Communications Security*. 833–844.

C.M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer, New York, NY.

Common Vulnerability Scoring System, V3. 2016. Development update. https://www.first.org/cvss. (2016).

Common Weaknesses Scoring System. 2014. https://cwe.mitre.org/cwss/cwss_v1.0.1.html. (2014).

G.F. Cooper. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *J. of AI* 42, 2 (1990), 393–405.

R. Dechter. 1996. Bucket elimination: A unifying framework for probabilistic inference. In *Procs. Int. Conf. on Uncertainty in AI*. 211–219.

M. Frigault, L. Wang, A. Singhal, and S. Jajodia. 2008. Measuring network security using dynamic Bayesian network. In *Procs. Workshop on Quality of protection*. 23–30.

Gartner, Inc. 2014. Gartner Says Worldwide Information Security Spending Will Grow Almost 8 Percent in 2014 as Organizations Become More Threat-Aware. http://www.gartner.com/newsroom/id/2828722. (2014).

N. Idika and B. Bhargava. 2012. Extending Attack Graph-Based Security Metrics and Aggregating their Application. *IEEE Trans. on Dependable and Secure Computing* 9, 1 (2012), 75–85.

A.T. Ihler, J.W. Fisher, and A.S. Willsky. 2005. Loopy Belief Propagation: Convergence and Effects of Message Errors. *J. of Machine Learning Research* 6 (2005), 905–936.

K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer. 2009. Modeling modern network attacks and countermeasures using attack graphs. In *Conf. Computer Security Applications*. 117–126.

S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams. 2011. Cauldron mission-centric cyber situational awareness with defense in depth. In *Military Communications Conf.* 1339–1344.

S. Jajodia, S. Noel, and B. OBerry. 2005. Topological analysis of network attack vulnerability. In *Managing Cyber Threats*. 247–266.

S. Jha, O. Sheyner, and J. Wing. 2002. Two Formal Analyses of Attack Graphs. In *Procs. of the Workshop on Computer Security Foundations*. 49–63.

B. Juba, C. Musco, F. Long, S. Sidiroglou-Douskos, and M.C. Rinard. 2015. Principled Sampling for Anomaly Detection. In *Network and Distributed System Security Symposium*. 1–14.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, Cambridge, MA.

W. Li and R.B. Vaughn. 2006. Cluster Security Research Involving the Modeling of Network Exploitations Using Exploitation Graphs. In *Int. Symp. on Cluster Computing and the Grid*. 1–11.

R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R. Cunningham. 2006. Validating and Restoring Defense in Depth Using Attack Graphs. In *Procs of Military Communications Conf.* 1–10.

Y. Liu and H. Man. 2005. Network vulnerability assessment using Bayesian networks. In *Data Mining, Intrusion Detection, Inform. Assurance, and Data Networks Security*, Vol. 5812. 61–71.

N. Lord. 2015. The History of Data Breaches. https://digitalguardian.com/blog/history-data-breaches. (2015).

A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B.D. Payne. 2015. Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices. 48, 1 (2015).

J.M. Mooij and H.J. Kappen. 2005. Sufficient Conditions for Convergence of Loopy Belief Propagation. In *Procs. of the Conf. on Uncertainty in AI*. 396–403.

L. Muñoz-González, D. Sgandurra, M. Barrère, and E.C. Lupu. 2016. Exact Inference Techniques for the Analysis of Bayesian Attack Graphs. *To appear in IEEE Trans. on Dependable and Secure Computing* (2016).

K.P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. MIT press, Cambridge, MA.

K.P. Murphy, Y. Weiss, and M.I. Jordan. 1999. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Procs. of the Conf. on Uncertainty on AI*. 467–475.

S. Noel and S. Jajodia. 2004. Managing Attack Graph Complexity through Visual Hierarchical Aggregation. In *Procs. Workshop on Visualization and Data mining for Computer Security*. 109–118.

S. Noel and S. Jajodia. 2014. Metrics Suite for Network Attack Graph Analytics. In *Procs. of the 9th Annual Cyber and Information Security Research Conference*. 5–8.

S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. 2003. Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs. In *Procs of the 19th Computer Security Applications Conference, 2003*. 86–95.

R. Ortalo, Y. Deswarte, and M. Kaâniche. 1999. Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security. *IEEE Transactions on Software Engineering* 25, 5 (1999), 633–650.

X Ou, WF Boyer, and MA McQueen. 2006. A scalable approach to attack graph generation. In *Procs. Conf. on Computer and Communications Security*. 336–345.

J. Pamula, S. Jajodia, P. Ammann, and V. Swarup. 2006. A Weakest-Adversary Security Metric for Network Configuration Security Analysis. In *Workshop on Quality of Protection*. 31–38.

J. Pearl. 1982. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Procs. National Conf. on AI*. 133–136.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

C Phillips and L.P. Swiler. 1998. A graph-based system for network-vulnerability analysis. In *Procs. of the Workshop on New Security Paradigms*. 71–79.

N. Poolsappasit, R. Dewri, and I. Ray. 2012. Dynamic Security Risk Management using Bayesian Attack Graphs. *IEEE Trans. on Dependable and Secure Computing* 9, 1 (2012), 61–74.

L Rabiner and BH Juang. 1986. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine* 3, 1 (1986), 4–16.

E. Raftopoulos and X. Dimitropoulos. 2013. Understanding Network Forensics Analysis in an Operational Environment. In *Security and Privacy Workshops*. 111–118.

B. Schneier. 1999. Attack trees. *Dr. Dobbs journal* 24, 12 (1999), 21–29.

G.R. Shafer and P.P. Shenoy. 1990. Probability propagation. *Annals of Mathematics and AI* 2 (1990), 327–352.

A. Sharma, Z. Kalbarczyk, J. Barlow, and R. Iyer. 2011. Analysis of Security Data from a Large Computing Organization. In *Int. Conf. on Dependable Systems Networks*. 506–517.

P.P. Shenoy and G.R. Shafer. 1990. Axioms for probability and belief-function proagation. In *Procs. Conf. on Uncertainty in AI*. 169–198.

O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. 2002. Automated Generation and Analysis of Attack Graphs. In *Procs. of the IEEE Symp. on Security and Privacy*. 273–284.

O. Sheyner and J. Wing. 2004. Tools for Generating and Analyzing Attack Graphs. In *Formal Methods for Components and Objects*. 344–371.

L.P. Swiler, C Phillips, D Ellis, and S Chakerian. 2001. Computer-attack graph generation tool. In *Procs. of the DARPA Inform. Survivability Conf. and Exposition II*, Vol. 2. 307–321.

Symantec. 2015. Internet Security Threat Report - Volume 20 - Appendices. https://www4.symantec.com/mktginfo/whitepaper/ISTR/21347931_GA-internet-security-threat-report-volume-20-2015-appendices.pdf. (2015).

G. Tan, M. Poletto, J. Guttag, and F. Kaashoek. 2003. Role Classification of Hosts Within Enterprise Networks Based on Connection Patterns. In *USENIX, General Track*. 15–28.

L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. 2008. An attack graph-based probabilistic security metric. In *Procs. 22nd IFIP WG 11.3 Conf. on Data and Applications Security*. 283–296.

L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel. 2014. k-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown Vulnerabilities. *IEEE Trans. on Dependable and Secure Computing* 11, 1 (2014), 30–44.

L. Wang, A. Singhal, and S. Jajodia. 2007a. Measuring the Overall Security of Network Configurations Using Attack Graphs. In *IFIP Annual Conf. on Data and Applications Security and Privacy*. 98–112.

L. Wang, A. Singhal, and S. Jajodia. 2007b. Toward Measuring Network Security Using Attack Graphs. In *Procs. of the 2007 ACM Workshop on Quality of protection*. 49–54.

Y. Weiss. 2000. Correctness of Local Probability Propagation in Graphical Models with Loops. *Neural Computation* 12, 1 (2000), 1–41.

Y. Weiss. 2001. Comparing the Mean Field Method and Belief Propagation for Approximate Inference in MRFs. *Advanced Mean Field MethodsTheory and Practice* (2001), 229–240.

M. Welling and Y.W. Teh. 2001. Belief Optimization for Binary Networks: A Stable Alternative to Loopy Belief Propagation. In *Procs. of the Conf. on Uncertainty in AI*. 554–561.

E. Wheeler. 2011. *Security Risk Management: Building an Information Security Risk Management Program from the Ground Up*. Syngress Publishing.

WhiteHat Security. 2015. Website Security Statistics Report. https://info.whitehatsec.com/rs/whitehatsecurity/images/2015-Stats-Report.pdf. (2015).

P. Xie, J.H. Li, X. Ou, P. Liu, and R. Levy. 2010. Using Bayesian Networks for Cyber Security Analysis. In *Int. Conf. on Dependable Systems and Networks*. 211–220.

A.L. Yuille. 2001. CCCP Algorithms to Minimize the Bethe and Kikuchi Free Energies: Convergent Alternatives to Belief Propagation. *Neural Computation* 14 (2001), 1691–1722.

M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese. 2016. Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks. *IEEE Trans. on Information Forensics and Security* 11, 5 (2016), 1071–1086.

Received June 2016; revised November 2016; accepted May 2017