IST-153 Workshop on Cyber Resilience

Bayesian Attack Graphs for Security Risk Assessment

Luis Muñoz-González, Emil C. Lupu
Department of Computing, Imperial College London, 180 Queen's Gate, SW7 2AZ, London, UK.
Phone: +44 (0)20 7594 8249, Fax: +44 (0) 20 7594 8932
l.munoz@imperial.ac.uk, e.c.lupu@imperial.ac.uk

Abstract—Attack graphs offer a powerful framework for security risk assessment. They provide a compact representation of the attack paths that an attacker can follow to compromise network resources from the analysis of the network topology and vulnerabilities. The uncertainty about the attacker's behaviour makes Bayesian networks suitable to model attack graphs to perform static and dynamic security risk assessment. Thus, whilst static analysis of attack graphs considers the security posture at rest, dynamic analysis accounts for evidence of compromise at run-time, helping system administrators to react against potential threats. In this paper, we introduce a Bayesian attack graph model that allows to estimate the probabilities of an attacker compromising different resources of the network. We show how exact and approximate inference techniques can be efficiently applied on Bayesian attack graph models with thousands of nodes.

Index Terms—Attack Graphs, Security Risk Assessment, Bayesian Networks, Approximate Inference.

1 Introduction

The efforts to protect networks cannot cope with the sophistication of modern cyber attacks, as shown by the history of data-breaches that organizations have suffered recently [1]. Identify and patch vulnerabilities is not always possible, since lack of manpower or the impossibility of interrupting critical systems prevents from doing so. Thus, assessing and prioritizing the risks of the network is essential to optimize resources and the effort required to protect the network. However analysing the network risks in isolation offers a limited perspective of the network security, given the complex interdependencies between vulnerabilities. In this sense, Attack Graphs (AGs) [2], [3] provide a powerful framework to represent prior knowledge about vulnerabilities and network connectivity, depicting the 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 [4]. Two types of analysis can be performed: *Static analysis* determines the *a priori* risks of the network when we consider the security posture at rest. *Dynamic analysis* updates those risks in light of any indication of compromise at some of the networks components, e.g. from Security Information and Event Management (SIEM) and Intrusion Detection Systems (IDS). While the static analysis of AGs is useful for network hardening, the dynamic analysis allows system administrators to profile the attacker's paths and prioritize remediation strategies to mitigate the effects of ongoing attacks.

Given the uncertainty about the attackers' ability to exploit vulnerabilities, Bayesian Networks (BNs) provide an adequate framework to model AGs [5]–[9], since they depict causal relationships between random variables in a compact way. However, computing the unconditional and posterior probabilities that are needed to perform static and dynamic analysis on AGs is an NP-Hard problem. Therefore, the use of efficient inference techniques is of essence to reduce the time and computational resources required and improve the applicability of the approach to perform both static and dynamic analysis of Bayesian Attack Graphs (BAGs).

On the other side, given the typical clustered structure of corporate networks, the BAGs that are expected in real scenarios should also reflect this cluster structure. In this paper we show that this kind of cluster structures favours the scalability of exact and approximate inference algorithms to perform static and dynamic analysis of BAGs, scaling up to graphs with thousands of nodes [8], [9]. This modular structure of the networks and the BAGs can also facilitate analyses at different levels of granularity, which can allow a better scalability for Bayesian inference algorithms and, at the same time, produce risk assessments that can be more interpretable to system administrators, especially for large corporate networks.

The remainder of the paper is organized as follows: In Sect. 2 we describe BAG models and exact and approximate inference techniques to compute the probabilities needed for the static and dynamic analysis. In Sect. 3 we show experimental results comparing exact and approximate inference techniques for the static and dynamic analysis of synthetic BAGs of different sizes. Finally, in Sect. 4 we discuss the future research directions that, in our opinion, should be considered, to provide more efficient and scalable security risk assessment with AGs.

2 Bayesian Attack Graphs

AGs are graphical models that represent the knowledge about networks vulnerabilities and their interactions, showing the different paths an attacker can follow to compromise a given objective by exploiting a set of vulnerabilities. Along each attack path, vulnerabilities are exploited in sequence, so after each successful exploit the attacker acquires more privileges towards her goal. In the literature we can distinguish two main types of representations: *state-based* [10], [11] and *logical* [3], [12] AGs.

In line with most of the recent literature on AGs, in this paper, we only consider logical AGs, since state-based representations are known to scale exponentially with the number of nodes and vulnerabilities in the network, making them impractical even for small corporate networks. In contrast, logical AGs produce more compact representations that grow polinomially with the number of vulnerabilities and the number of connected pairs of hosts [13]. Logical AGs 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 [3].

Some of the literature on AG analysis assumes that monotonicity induces a Directed Acyclic Graph (DAG) structure [5], [7]. Although this is not completely true, and some directed cycles may be present, monotonicity helps to get rid of many directed cycles related to duplicate attack paths. However, [14] explain how to handle and eliminate remaining directed cycles without loss of integrity. For the remainder of the paper, we will consider AGs with a DAG structure.

The uncertainty about the attacker's behaviour and the DAG structure of the AG make Bayesian networks (BNs) a reasonable alternative to model and analyse AGs. Thus, BNs allow to calculate the probability of an attacker reaching a security condition (state) in the AG. More formally, a BN can be definedd as a directed acyclic graphical model where the nodes represent random variables and the directed edges represent the dependencies between these 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 particular context of the BAG, the nodes represent the different security states that an attacker can acquire. We model the behaviour of these states as Bernoulli random variables.

2.1 Conditional Probability Tables

The information available at each node in the BAG is the conditional probability distribution $p(X_i|\mathbf{pa}_i)$, i.e. the probability of a node to be compromised given the state of its parent nodes $\mathbf{pa}_i$. In oder words, $p(X_i|\mathbf{pa}_i)$ represent the probabilities of an attacker reaching a security state X_i given the observation of its preconditions $\mathbf{pa}_i$ and the vulnerabilities $\mathbf{v}_i$ that can be exploited to compromise X_i.

The probabilities of an attacker successfully exploiting vulnerabilities are parameters of the BAG model that are used to calculate the conditional probability tables. A common approach to estimate p_{v_i}, the probability of an attacker successfully exploiting a vulnerability v_i, is by means of CVSS [15], as proposed in [6]–[9]. More concretely the exploitability submetric of CVSS can be considered more appropriate to estimate these probabilities since it tries to measure the difficulty of exploiting a vulnerability.

To main types of conditional probability tables can be consider: *AND* and *OR*. In the first case all preconditions must be satisfied to be able to compromise node X_i. In contrast, in the OR case, only one precondition is needed to attempt to attack node X_i. Thus, the AND conditional probability table can be calculated as:

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

whereas for *OR* conditional probability we have:

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

The leak factor p_{l_i} models the non-perfect behaviour of the alert system and the possible presence of unknown zero-day vulnerabilities. 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.

2.2 Non-perfect Alert System and Zero-day Vulnerabilities

The leak factor p_{l_i} in (2) and (3) models the cases where, even when all the preconditions are in the *False* state, i.e. not achieved by the attacker, there is still some non-zero probability of X_i taking the *True* state. The reason for this is because the attacker can successfully exploit a zero-day vulnerability to compromise X_i or because the alert correlation system has triggered a false alarm.

Defining the error probability of the alert system as p_e, and the probabilities of an attacker successfully exploiting a zero-day vulnerability for a node in the BAG as p_{z_i}, the leak factor can be computed as:

$$p_{l_i} = 1 - (1 - p_e)(1 - p_{z_i}) \tag{4}$$

Estimating p_e, the error probability of the alert correlation system, is far from a trivial task. Although some *ad hoc* methodologies have been applied in the literature [16], [17], it still remains an open problem. The difficulty relies on the dynamic aspects of the system behaviour. However, even a rough estimate of p_e can be useful to provide better risk assessments with BAGs.

Estimating p_{z_i}, the probability of an attacker successfully exploiting a zero-day vulnerability to compromise node X_i is even more challenging. First, we need to estimate the probability of having zero-day vulnerabilities for the software running in each machine of the network. Second, it is not trivial to estimate the *easiness* of exploitation of these potential vulnerabilities. Finally, the preconditions needed

for the attacker to exploit a zero-day vulnerability can be the postconditions of longer attack paths. Thus, the severity of the potential zero-day vulnerabilities should be assessed according to the proximity of the current node with respect to the target node. As discussed in [9], similar to the estimation of the probability of successful exploitation of vulnerabilities with CVSS scores, in the case of zero-day vulnerabilities we can estimate the corresponding probabilities by means of the Common Weaknesses Scoring System (CWSS) [18]. Thus, CWSS scores provide a quantitative measure measure of the unfixed weaknesses present in a software application.

2.3 Static and Dynamic Analysis

For the static analysis of the BAGs, we are interested in calculating the unconditional probabilities $p(X_i)$ for all the nodes in the graph. Thus, $p(X_i)$ corresponds to the probability of an attacker reaching a given security condition X_i. Using Bayes rule we can compute this probability as:

$$p(X_i) = \sum_{\mathbf{X}-X_i} p(\mathbf{X}) = \sum_{\mathbf{X}-X_i} \prod_{j=1}^{n} p(X_j|\mathbf{pa}_j) \qquad (5)$$

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 harden the network or to apply static risk mitigation techniques.

For the dynamic analysis of the BAG, given evidence of attack on a set of nodes $\mathbf{X}_e$ (by means of the alert correlation system), we need to compute the posterior probability $p(X_i|\mathbf{X}_e)$ in all the nodes of the network, except for the set $\mathbf{X}_e$. Applying Bayes rule we can compute the posterior probability as:

$$p(X_i|\mathbf{X}_e) = \frac{p(X_i, \mathbf{X}_e)}{p(\mathbf{X}_e)} = \frac{\sum_{\mathbf{X}-\{X_i,\mathbf{X}_e\}} p(\mathbf{X})}{\sum_{\mathbf{X}-\mathbf{X}_e} p(\mathbf{X})} \qquad (6)$$

The posterior probabilities provide a re-estimation of the risk at run-time, which can help system administrators to plan and prioritize security measures to mitigate or contain an ongoing attack [7].

The exact calculation of (5) and (6) is an NP-Hard problem [19]. Thus, efficient algorithms such as Variable Elimination [20] or Junction Tree (JT) [21] are necessary even for small graphs. However, the applicability of these techniques can be limited in cases where the graphs are dense, demanding a lot of computational resources to compute (5) and (6). Even when the structure of BAGs is expected to have some special properties, given the typical clustered network structure and the limited number of attack paths to compromise a node, there are no guarantees about the computational complexity for these exact inference techniques. An experimental comparison between Variable Elimination and JT is presented in [8], showing that JT outperforms Variable Elimination both in terms of memory and time needed to compute the unconditional and posterior probabilities. Thus, JT can be applied to perform both static and dynamic analysis to graphs up to a few thousands of nodes.

Approximate inference in BNs is also known to be NP-Hard [19], but efficient techniques like Loopy Belief Propagation (LBP) [22] allow to efficiently estimate (5) and (6). Since for BAGs we only have Bernoulli random variables, LBP scales in time and memory as $\mathcal{O}(N2^s)$, where N is the number of nodes in the BAG, and s is the scope of the biggest factor, i.e. the maximum number of parent nodes that a node can have in the graph. Since we expect to have some security *in-place* we expect s to be small. Then, as shown in [9], LBP allows to scale-up to larger BAGs compared to exact inference techniques. Despite LBP do not provide the exact values for the unconditional and posterior probabilities, we should not be deterred about the estimates produced by LBP, since the probabilities of exploitation of vulnerabilities, modelled through CVSS scores, are already a rough estimate and, on the other hand, as shown in [9], the accuracy of LBP to estimate (5) and (6) is more than reasonable to help system administrators to harden the networks or to propose countermeasures to mitigate the effect of ongoing attacks.

3 Experiments

In this section we present an experimental evaluation comparing the time performance of LBP and JT for the static and dynamic analysis of BAGs, i.e. the time required to compute the unconditional and posterior probabilities respectively. We have used the Bayes Net toolbox for Matlab[1] as the core implementation for all the algorithms.

Following a similar methodology than in [8], [9] we have generated synthetic AGs for the experiments. Unfortunately, currently, there are no collections of AGs of similar variety obtained empirically from real systems. To the best of our knowledge no collections of empirically obtained AGs exist in the public domain at all. Then, to provide a comprehensive evaluation of the algorithms with AGs of different sizes and interdependencies we need to resort to synthetic AGs.

Since typical corporate networks are structure into subnetworks and contain several hosts with common software installations, 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, as routing and firewall rules between subnetworks usually hinder the progression of the attack. This is in line with the AG examples shown in [23], where only attacks across subnetworks are considered. To generate the synthetic graphs we have considered networks with clusters (subnetworks) of the same size n_c. For each cluster we have generated pseudo-random graphs with a DAG structure where we limit the maximum number of possible parents to each node to m. Finally, the dependencies across clusters are modelled by adding one edge from 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 the experiments we have generated synthetic graphs with $n_c = 20$ and 50, $m = 3$, varying the total number of nodes from 100 to $1,000$. The values of the probabilities of successful exploitation of vulnerabilities are drawn at random from the distribution of CVSS scores in [24]. For each graph size explored, we have generated 20 independent graphs for each value of n_c considered.

1. https://github.com/bayesnet/bnt

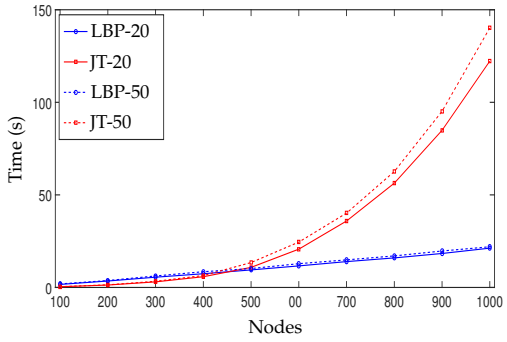

Fig. 1. Time to compute the unconditional probabilities for LBP and the JT algorithm for cluster networks with different cluster sizes ($n_c = 20$ and 50) and $m = 3$ for the static analysis of the BAGs.

In Fig. 1 we show the average time[2] required to compute the unconditional probabilities for the static analysis of the BAGs. We observe that JT scales exponentially with the number of nodes, whereas LBP[3] scales linearly and requires less time to perform the analysis. The size of the clusters does not have an impact in the time performance of LBP and, in the case of JT, the difference between the two cases is moderate.

In Fig. 2 we report the time required to perform the dynamic analysis, when we observe evidence of attack in 3 nodes (chosen at random). In Fig. 2(a) we can observe that both JT and LBP scale linearly with the number of nodes and that the cluster size, n_c, has a very small impact on the performance. However, in contrast to the static analysis, JT is much faster than LBP, computing all the posterior probabilities in less than 1 second for BAGs with $1,000$ nodes, as can be appreciated with more detail in Fig. 2(b).

The experimental results suggest that JT is more appealing to perform dynamic security risk assessment with BAGs. However, the exponential scalability for the static analysis, which in the case of JT is required before performing dynamic analysis, limits its applicability to large networks. In contrast, LBP scales linearly for both the static and dynamic analysis of the graph. Despite LBP is slower than JT for the dynamic analysis, as shown in [9], LBP allows to monitor the values of the posterior probabilities at each iteration of the algorithm. Therefore, we can also obtain accurate estimates for the posterior probabilities before the algorithm converges.

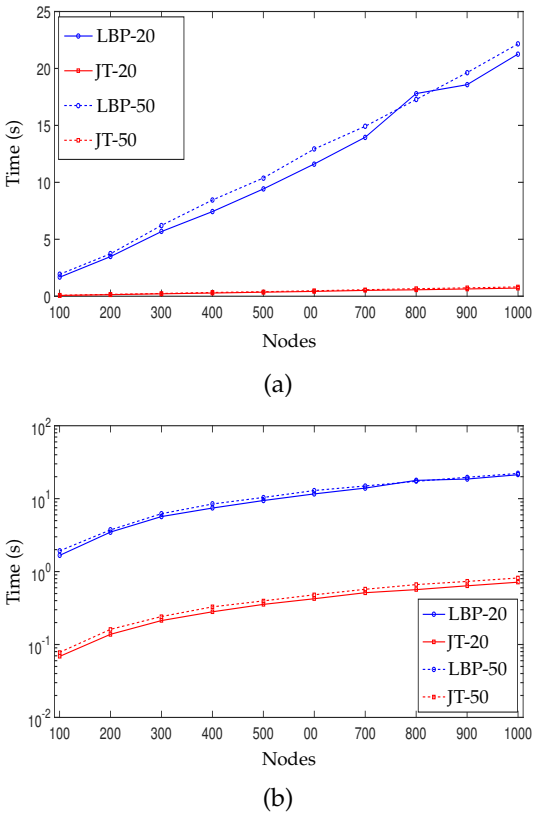

Fig. 2. Time to compute the posterior probabilities for LBP, and the JT algorithm for cluster networks with different cluster sizes ($n_c = 20$ and 50) and $m = 3$ for the dynamic analysis of the BAGs (when we observe evidence of attack at 3 random nodes): (a) Time in natural scale; (b) Time in logarithmic scale.

4 Discussion

Attack Graphs are a powerful tool for static and dynamic risk assessment of networks, since they take into account the interdependencies between vulnerabilities depicting the ways an attacker can compromise different network resources. Bayesian networks allow to measure the risk at the different nodes of the AG given the likelihood of successful exploitation of the vulnerabilities present in the networks and the relation between the preconditions needed to compromise a node. Although computing the probabilities in BNs is an NP-Hard problem, we have shown that the use of appropriate inference techniques such as Junction Tree and Loopy Belief Propagation, can allow to use BAGs for static and dynamic risk assessment, helping system administrators to enforce the security of the networks and mitigate the effect of ongoing attacks.

However, the heterogeneity of modern infrastructures and the dynamic aspect of the networks limits the application of traditional attack graph generation tools for these environments. Then, new methodologies and models are needed to cope with the requirements of modern networks. In the remainder of this section, we discuss some of the limitations and opportunities for the development of new attack graph models for both generation and analysis of AGs.

Although logical representations of AGs [3], [12] scale in a polynomial way with the number of nodes and vulnerabilities in the networks, their applicability is still limited to large scale scenarios, with thousands or millions of devices. Even if they can be computed, in these cases they offer a limited usability for system administrators. Thus, new mechanisms are needed to generate more scalable and interpretable AG models. This can be achieved by clustering *security domains* that are very similar. For example, within subnetworks, often we can find many machines that are configured similarly, so the vulnerabilities present on those machines should also be similar. Thus, in terms of

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

3. We have used the parallel version of LBP [19].

privilege escalation, compromising one or several of this machines can be considered equivalent.[4] On the other hand, the concept of "*security domain*" used in traditional AG methodologies is often restricted to the security privileges that the attacker can acquire on a single machine. This can be restrictive for modelling modern networks. For instance, if the attacker compromise a user account (for example with a *phishing* attack), she can have access to several machines or devices in the system.

Traditional AGs are built by considering only the network topology, reachability, and software vulnerabilities [12], not considering other security aspects present in the attack surface of modern infrastructures, such as IoT environments. In these scenarios we should also consider the physical and human vulnerabilities of the system. Thus, there is a need of new AG generation models capable of describing this extended and complex attack surface considering the interdependence between the cyber, human, and physical aspects of the network security.

Given the complexity and size of modern networks, different security perspectives can be considered. Thus, high level abstractions of the AG can help to produce more interpretable representations, as proposed in [25] to manage AG complexity for visualization. Similar approaches can be adopted for AG analysis: Thus, we can allow for tractable analysis of AGs in very large networks through lower resolution representations. This induces a trade-off between accuracy, the level of resolution, and the tractability of the analysis. For example, in large corporate networks, systems administrators may prioritize the contention of attacks to prevent their propagation at subnetwork level. Then, once the attack is contained within a subset of subnetworks, finer analyses may be required to mitigate the effect of the attacks at machine level in the affected subnetworks. To achieve this, we need to develop mechanisms to produce aggregate risk estimates at different levels, capable of summarizing the state of a given subnetwork or a set of machines. Then, Bayesian inference techniques can also be applied to perform static and dynamic analysis in such aggregate models.

Acknowledgments

This work has been supported by the UK government under EPSRC grant EP/L022729/1. The authors would like to thank British Telecom for their collaboration in this research.

References

[1] N. Lord, "The History of Data Breaches," https://digitalguardian.com/blog/history-data-breaches, 2015.
[2] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing, "Automated Generation and Analysis of Attack Graphs," in *Procs. of the IEEE Symp. on Security and Privacy*, 2002, pp. 273–284.
[3] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, Graph-Based Network Vulnerability Analysis," in *Procs. Conf. on Computer and Communications Security*, 2002, pp. 217–224.
[4] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," in *Conf. Computer Security Applications*, 2009, pp. 117–126.
[5] Y. Liu and H. Man, "Network vulnerability assessment using Bayesian networks," in *Data Mining, Intrusion Detection, Inform. Assurance, and Data Networks Security*, 2005, vol. 5812, pp. 61–71.
[6] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic Bayesian network," in *Procs. Workshop on Quality of protection*, 2008, pp. 23–30.
[7] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic Security Risk Management using Bayesian Attack Graphs," *IEEE Trans. on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2012.
[8] L. Muñoz-González, D. Sgandurra, M. Barrère, and E.C. Lupu, "Exact Inference Techniques for the Analysis of Bayesian Attack Graphs," *To appear in IEEE Trans. on Dependable and Secure Computing*, 2017.
[9] L. Muñoz-González, D. Sgandurra, A. Paudice, and E.C. Lupu, "Efficient Attack Graph Analysis through Approximate Inference," *To appear in ACM Trans. on Privacy and Security*, 2017.
[10] S. Jha, O. Sheyner, and J. Wing, "Two Formal Analyses of Attack Graphs," in *Procs. of the Workshop on Computer Security Foundations*, 2002, pp. 49–63.
[11] C Phillips and L.P. Swiler, "A graph-based system for network-vulnerability analysis," in *Procs. of the Workshop on New Security Paradigms*, 1998, pp. 71–79.
[12] S. Jajodia, S. Noel, and B. OBerry, "Topological analysis of network attack vulnerability," in *Managing Cyber Threats*, pp. 247–266. 2005.
[13] M. Albanese, S. Jajodia, and S. Noel, "Time-Efficient and Cost-Effective Network Hardening using Attack Graphs," in *Int. Conf. on Dependable Systems and Networks*, 2012, pp. 1–12.
[14] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *Procs. 22nd IFIP WG 11.3 Conf. on Data and Applications Security*, 2008, pp. 283–296.
[15] Common Vulnerability Scoring System, V3, "Development update," https://www.first.org/cvss, 2016.
[16] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B.D. Payne, "Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices," vol. 48, no. 1, 2015.
[17] B. Juba, C. Musco, F. Long, S. Sidiroglou-Douskos, and M.C. Rinard, "Principled Sampling for Anomaly Detection," in *Network and Distributed System Security Symposium*, 2015, pp. 1–14.
[18] Common Weaknesses Scoring System, ," https://cwe.mitre.org/cwss/cwss_v1.0.1.html, 2014.
[19] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, MIT press, Cambridge, MA, 2009.
[20] R. Dechter, "Bucket elimination: A unifying framework for probabilistic inference," in *Procs. Int. Conf. on Uncertainty in AI*, 1996, pp. 211–219.
[21] P.P. Shenoy and G.R. Shafer, "Axioms for probability and belief-function proagation," in *Procs. Conf. on Uncertainty in AI*, 1990, pp. 169–198.
[22] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.
[23] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams, "Cauldron mission-centric cyber situational awareness with defense in depth," in *Military Communications Conf.*, 2011, pp. 1339–1344.
[24] "CVE Details. The ultimate security vulnerability datasource," http://www.cvedetails.com, 2016.
[25] S. Noel and S. Jajodia, "Managing Attack Graph Complexity through Visual Hierarchical Aggregation," in *Procs. Workshop on Visualization and Data mining for Computer Security*, 2004, pp. 109–118.

4. Note that we are not considering data exfiltration scenarios, where compromising one or several machines can have a different impact.