Athens Journal of Technology and Engineering - Volume 6, Issue 4 – Pages 211-222

Estimating the Success of IT Security Measures in Industry 4.0 Environments using Monte Carlo Simulation on Attack Defense Trees

By Till Hänisch[*] & Christoph Karg[†]

The choice of defense strategies in IT-security is often guided by qualitative methods only. For common scenarios like securing desktop computers, web servers, or extranets, there are well accepted best practices for establishing a secure environment. For other scenarios like computers in production environments (often referred as "Industry 4.0") this is not the case. To secure such systems, there are a number of options, but their relevance for a certain application is less clear and is specific for the situation. Especially, for small and medium enterprises it is often unclear, which security measures to apply in their production. This paper describes a method based on attack defense trees, which allows to assess the value of defense measures based on simulated attacks.

Keywords: *IT-security, Industry 4.0, Attack Tree.*

Introduction

Examples like the ransomware attacks of the last years, show that high profile cyber-attacks like advanced persistent threats (APTs) are not only targeting large multinational enterprises or governments, but also small and medium enterprises. Unfortunately, there is no clear consensus on how small and medium enterprises should protect their IT systems in production environments, for example in "Industry 4.0" scenarios. The general best practices published by institutions like the BSI in Germany or the CPA (Axelsen 2018) are not targeted to production environments and therefore of little help. The most prominent security measure, which is separating the production systems from the internet ("air gap"), is no longer realistic. Current trends like using Big Data, Industrial Internet of Things, and especially remote maintenance need a network connection between production IT and the outside world. Because of this, the choice of adequate security measures is increasingly important. To address this question, we conducted a survey among a number of larger companies, aiming to identify best practices for IT security in production (Hänisch and Rogge 2017). But it was unclear, how to prioritize them for each different small or medium company.

While there is little doubt that common techniques like network segmentation, firewalls, virus scanners, intrusion detection systems, or enhanced employee awareness against social attacks make sense, and should be used by any means, allocation of budget or assigning priorities to more sophisticated techniques is not so obvious.

[*]Professor, DHBW Heidenheim, Germany.
[†]Professor, Hochschule Aalen, Germany.

https://doi.org/10.30958/ajte.6-4-1 doi=10.30958/ajte.6-4-1

To decide, if adding a certain countermeasure makes sense in a given environment, a tool is needed, which allows to answer the question "Does countermeasure X stop the threat Y?" To decide, if the countermeasures currently in use in a given environment make sense, the question "Which countermeasure stops which threats?" also has to be addressed.

Attack Defense Trees (ADT) are a well-established way to structure the complex interdependencies of threat steps and countermeasures (Roy et al. 2010). But since there are many possible paths through the Attack Defense Tree, the questions given above can't be answered directly. A Monte Carlo simulation is used: a (large) number of attacks is simulated as a random walk through the tree. By counting, which countermeasure stops which attack, the above questions can be answered.

One problem remains: For every node in the ATD, the success probability has to be defined. Since there are many nodes, it is too much effort to build a new individual tree for every environment; for example a new tree for every company. If a common standardized tree is used, it has to be defined without knowing the details of the actual (difficulty of the) attack: standard attack trees neither compensate for local specialties of the target, like the awareness level of employees, the degree of standardization or the response times of a CERT team, nor do they incorporate knowledge about the attacker. While the latter is hard to specify (usually you don’t know the attacker in advance) it might be possible to define attacker groups by giving an upper limit of the knowledge (or for example the resources available) of an attacker.

The contribution of this paper is the definition of a method that allows specifying the assumptions about the success probabilities of an attack given a set of countermeasures in a way that is accessible to humans and allows adapting to a certain situation. The most important adaption is the analysis of a specific company without having to build a new tree. Intuition needed for finding the relevant defense measures can be replaced by interviews; selection of relevant attacks can be based on published data and/or attack trees and might be extended by case specific expertise. With the described method, the specification of success probabilities is simplified by splitting the probabilities in two parts; one that is common to all scenarios and can be reused, and a second, which might be different for each company.

In summary: This paper describes a method based on attack defense trees which allows to assess the value of defense measures based on simulated attacks. The goal is not to provide an automatic strategy generator or tool selector, but to make the importance of specific measures transparent and help guide decisions by supporting human security specialists.

Related Work

While attack trees are normally used to find out which way an attacker will choose, we assume that we already know about the possible threats, but want to find the best measures to stop them.

One problem[1] of prioritizing countermeasures is to assess the probability of a successful attack, without the countermeasure in question. Because intuition is a questionable way of computing probabilities of rare events (Taleb 2007), a quantitative approach is advantageous. Methods for calculating probabilities of success for an attack are quite old: Attack trees were developed in the sixties (Ericson 1999), and have been used in IT security for more than twenty years (Amoroso 1994, Schneier 2000).

Roy et al. (2010) propose an extension of Attack-Defense-Trees called Attack-Countermeasure-Trees, which includes defense nodes at any level in the tree. Our formalism could be applied to this type of tree too, but we put our focus on trees as simple as possible to keep the effort of adaption to a special case like a company as low as possible. They also use ROI-type calculations to find an optimal set of countermeasures. While cost is a well-accepted way of measuring the value of things, real costs are often not that easy to define for all events. To do that, it is necessary to rely on assumptions about possible threats, possible attackers, and their motivation and the effectiveness of the defense measures deployed. While assumptions are certainly necessary to a certain extent, one has to be aware that assumptions are no facts, and have to be treated accordingly. In our opinion, a good method is one that needs as few assumptions as possible.

Probabilities, required in many methods, especially in attack trees, can either be based on the past (that means counting), or on assumptions. Extrapolating the past into the future is risky, especially with rare events and impossible for events never seen before (Taleb 2007). Guessing probabilities is hard too, especially for the same kind of events as above: while humans have a good intuitive understanding of the probability of common events, the human brain is particularly bad assessing rare events. That makes sense from an evolutionary point of view, but is very bad for IT-security, where especially the rare and/or new attacks might be very dangerous in terms of damage. It is necessary to use a method for specifying the possible damage in a way that is accessible to humans.

There are attempts to generate Attack Trees automatically, see for example (Paul 2014). But these are not adapted to a specific environment, company specific aspects like the awareness of employees are not considered.

Attacks can be modeled in other forms like UML diagrams, which allow to model not only a static state space but can also show dynamic interactions, which are especially useful to analyze the behaviour in case of failed defense measures. But, the complexity of these models limits their use to general cases where their use is to understand general attack and defense mechanisms and not to analyze a specific environment for example a company (Löhner 2018).

The separation of the probability in two parameters can be used for other purposes, like (Fen et al. 2012) proposing a two part definition of the possibility of an attack using a difficulty of an attack and the attack detection possibility to define the success probability but doesn't use the difficulty parameter to adapt to

[1]The one investigated in this paper. There are more, like estimating the damage of a successful attack.

special situations but to select the path an attacker chooses to select the attack steps using multi-attribute utility theory.

A common approach in IT security management is the assessment of risk by qualitative means. For example, the OWASP risk rating methodology (OWASP 2019) assesses the risk of a threat by correlating its likelihood and its impact. Both the likelihood and the impact of the threat are rated with the value LOW, MEDIUM, or HIGH, respectively. The likelihood rating is derived by analyzing factors of the threat agent such as his or her skill level and motive and analyzing factors of the vulnerability such as the ease of its discoverability and the ease of its exploitation. Factors relevant for the assessment of the impact are technical impacts such as the loss of availability or confidentiality and business impacts such as financial damage or non-compliance. Depending on the rating of the likelihood and the impact, the overall risk severity of the threat is rated with None, Low, Medium, High, or Critical, respectively.

Another popular framework is the Common Vulnerability Scoring Framework (CVSS) (FIRST 2019). The goal of CVSS is the communication of characteristics and severity of software vulnerabilities. Similarly to the OWASP approach, CVSS rates a vulnerability with the value None, Low, Medium, High or Critical, respectively. The rating is derived by assessing the vulnerability with three metric groups. The Base metrics group addresses intrinsic properties of the vulnerability which do not change over time and are not dependent on the user's environment. Metrics of the group rate the exploitability of the vulnerability and the respective impact in terms of confidentiality, availability, and integrity. The Temporal metrics group considers aspects of the vulnerability which might change over time such as the existence of exploitation tools or security patches. The Environment metrics group allows the customization of the score with respect to the requirements of a particular company or organization. While the assessment of the base group is mandatory to determine the vulnerability's score, the assessment of the temporal and environment is optional. An advantage of CVSS over other risk assessment frameworks is the world wide support by certs and organizations. For example, the U.S. National Institute of Standards and Technology runs the National Vulnerability Database[2] which provides a comprehensive and up-to-date list of vulnerabilities with CVSS scores.

Methodology

ADTs are an extension of attack trees, which include countermeasures as leaf nodes (Kordy et al. 2014). While these trees and especially their evaluation can be made complex, see for example (Bistarelli 2007), a very simple version is used in our proposed method.

[2]Website: https://nvd.nist.gov.

Figure 1. *a) Simple Attack Tree with Different Node Types. To Achieve the Goal (N0009) there are Three Alternatives (N0005, N0006, N0007) from which the Attacker might choose one. b) The Corresponding Attack Defense Tree Shows N0005 having one Countermeasure N0001 which might stop this Attack. N0006 Needs Two Measures N0002 and N0003 to be Successful to Stop the Attack*

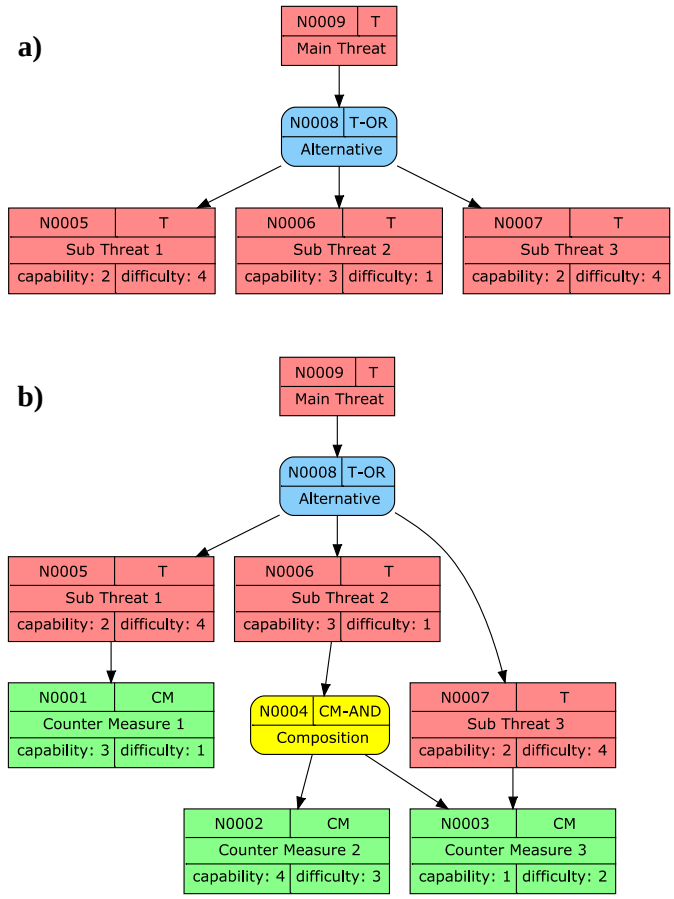


Figure 1a shows a simple attack tree. The attack is modeled as a hierarchy of simpler steps. Threat nodes in Figure 1 may have groups of children (Roy et al. 2010) where either all have to be successful (Composition, AND), or where it is sufficient that one is successful (Alternative, OR). To keep the design of the tree simple, we do not use selections out of a group of events. If this way of modeling is needed, it can be converted into nested structures of AND and OR.

In addition to these combinations, we suggest a group of actions, which have to be successful in a given sequence (SAND); for an overview see (Kordy et al. 2015). This is interesting for example in cases, where security incidents can be detected in an early stage of an attack and, if detected, be used to prevent following steps to be successful. Using a sequence allows to clarify, which attack steps shall be blocked and which can be ignored due to blocking of measures in earlier steps. This does not change the overall probability of success, but is important information for the prioritization of defense measures.

A second use of a sequence is to make the intent of the modeler explicit, when in reality the order of steps is not arbitrary. Assume we want to model the following attack steps:

1. Spear phishing Office-IT
2. Put Drive-by Malware in place
3. Infect control panel app

In step 2, maybe an XSS attack of a trusted website could be used. When specifying the difficulty of this step, it will highly depend on the website, which has to be used. In general, it is not difficult to infect a random site with an XSS attack vector. But, if for example we want to attack an office application in the controlling department of a large company with good employee awareness, it will be very difficult to exploit one of the few websites, an employee will use during his work. Because of this, the actual difficulty for this attack step cannot be defined without knowing in which context (and the order of the steps is part of this context) the steps will be executed. But since the usefulness of this additional modeling tool has not been validated in our practical experiments, in this paper we will focus on the more common grouping methods of "AND" and "OR".

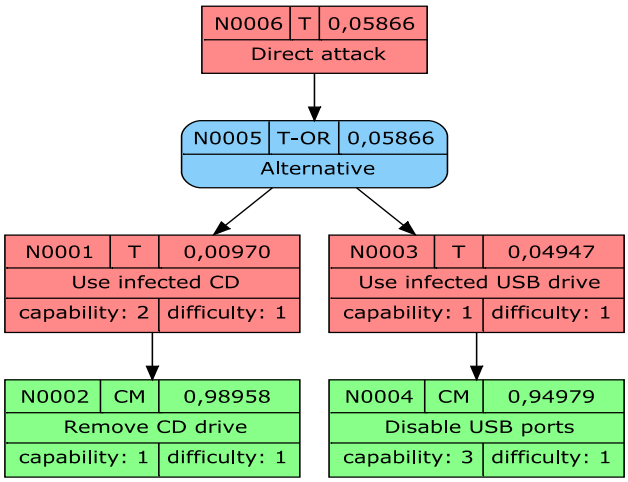
Figure 1b shows the same tree extended by countermeasures shown as green nodes. Countermeasures can also be modeled as subtrees containing logical groupings, exactly like attack steps.

The construction of trees can be based on real examples found in the wild, for example from interviews. The initial attack tree we used for our work was based on interviews with a number of companies about their defense measures used to secure their production sites (Hänisch and Rogge 2017). Additionally, it was used to understand implications on security architecture for industry appliances (Hänisch 2018).

The success probabilities of the individual nodes are defined by an empirical equation. The success probability of an attack step like “Use infected CD" to place malware is defined in terms of the capabilities the attacker must have and the actual difficulty of making the attack successful in a certain situation. Required capabilities in this example are “Be able to produce an innocent looking but interesting DVD” and “Physically put the disc to a place where it is taken and put into the machine to be attacked”. Both of these are simple, if the attacker has physical access, and are pretty difficult, if not. How hard it is in a certain situation, e.g. for a certain company facility, to guess, what looks innocent and interesting to the people in the company to be attacked, and to get physical access to a relevant place depends heavily on the company[3]. That means, the capability level is at least in a first approximation common to all targets but the difficulty is different for every situation. That means, the capability part has to be defined only once, while the difficulty part must be defined specifically for every situation.

[3]And maybe on the type of attacker, like insider, motivated kid or member of an intelligence agency.

Figure 2. *Part of a larger Attack Defense Tree. To Attack a Computer without going over the Network, an Infected CD or an Infected USB Stick could be used. The Attack via the CD could be prevented by Removing the Drive, the Attack via USB by Disabling All (Accessible) USB ports. The Success Probabilities of the Different Nodes are given by their Respective Capability and Difficulty Value. The Resulting Success Probability is shown in the Top Right Corner of Each Node*



The probability of success for an attack step is therefore separated in two parameters

$$p = p(c, d)$$

If c and d are assumed to be independent sub-probabilities, p is the product of both, so our implementation uses

$$p = 1 - c * d$$

as a starting point. This can be modified by a frequency component providing (empirical) information about how "popular" this attack (step) is:

$$p = f * (1 - c * d)$$

From an economic perspective it makes more sense to deploy measures, which help against common attacks, instead of rarely used ones. But, this might depend on the attacker. An intelligence agency with unlimited resources might choose a rare and complex way to attack a target hoping that no-one will focus on preventing rare attack vectors. In this paper, we will only discuss the simple form with $f \equiv 1$.

Since we need probabilities and it is difficult to select values, especially for very rare and very common cases, we use a scale from 1 to 6, borrowing and simplifying[4] ideas from the OWASP risk rating model (OWASP).

[4]From OWASP Threat Agent Forces we borrow "Skill Level" and "Opportunity" as somewhat similar to our "capability" and "difficulty". Instead of the OWASP likelihood level scale with three groups of three, we use 3 groups of 2, in total 6 levels.

Table 1. *Definition of Probability Scale Values. The Same Values are used for Attacks and Countermeasures*

Scale value	Definition	Probability value
1	very low/very easy (e.g. everyone)	0.1
2	low/easy (e.g. some skills required/some problems)	0.3
3	middle (well, middle)	0.5
4	high (e.g. expert level)	0.8
5	very high/very difficult (e.g. highly motivated hacker/ criminal)	0.9
6	extremly high/extremly difficult (e.g. government agency level)	0.99

To rank the measures, a Monte Carlo simulation is applied. This is a random walk over the state space of the tree: We walk down the tree till we are at the leaf. While going back, we check how things work: if a measure catches, the attack has failed. If the measure succeeds this is logged. If the measure succeeds, the threats on the way back have failed as a consequence.

If the measure has not succeeded (or we have no measure), we make random attempts with the threats on our way up. If for example in Figure 1b "countermeasure 1" fails, we try, if "Sub Threat 1" is successful.

If a threat fails, everything up to the next "OR" on our way back has also failed. The probability for choosing an (OR) alternative is weighted by the required capability. As a first approximation we chose the associated probabilities from the Capability - Probability table defined above (see Table 1). Then we shuffle the alternatives to avoid dependence on the (artificial) sequence of the definition. Then we add the probabilities of the alternatives for normalization. Next, we generate a random number and add up the probabilities until we reach this number. The corresponding alternative is the one we choose. This is basically a roulette wheel selection as used in genetic optimization; see for example (Lipowski and Lipowska 2012) or (Goldberg 1989). More details on the final implementation can be found in (Karg and Hänisch 2019).

This process is repeated a (large) number of times. Since we logged, which measures were successful in stopping an attack, we can sort the measures by this number. The ones which prevented many attacks are the more important ones. This is the central assumption in our model

Findings/Results

To evaluate our method, we created an example tree, part of which is shown in the following Figure 3. The complete tree is available on github[5], along with the source code. Other examples for the construction of Attack Defense Trees can be found in (Edge 2007).

[5]https://github.com/TillHaenisch/ATD.git.

Figure 3. *Part of the Example Tree, here Placing Malware. The Lower Graphic Shows the Complete Tree with the Rectangle Defining the Border of the Graphic Above*

The tree was constructed from interviews with a number of companies to find out about the countermeasures used in their production IT. These were matched with a model-threat that tries to compromise a machine in a production

environment. The respective success probabilities were assigned using the scheme described above. This tree consists of 70 attack steps and 54 countermeasures. Even for this limited attack scope, the tree is pretty big and hard to oversee and understand. In our experience, a much larger tree is not manageable. This makes the use of very general attack trees rather difficult to say the least.

To find out, if the results of the Monte-Carlo-Simulation of the attack-defense-mechanisms are consistent with real-life strategies from our interviews, we simulated 10.000 attacks and analyzed, which threat steps were successful, and which measures successfully caught attacks. Table 2 shows some of the results.

Table 2. *Simulation Results: The Most and Least Successful Countermeasures and Attack Steps; there are shown the Type of Node (Threat or Countermeasure), the Actual Countermeasure or Threat Step and the Number of Successful Events in the Simulation in Absolute Numbers for 10000 Simulated Attacks*

ID	type of node	name	# successful
1	measure	spamfilter	1020
2	measure	increase employee awareness	970
3	measure	anti-malware-application	501
4	measure	remove Device	911
6	measure	uninstall nonessential application	374
7	measure	L2 segmentation	366
8	measure	no default gateway configured	266
		…	
48	measure	endpoint authentication via certificates	0
49	measure	End-to-End encryption	0
50	measure	use cryptographic techniques	0
1	threat	identify IoT with autoupdate from internet	1347
2	threat	poison DNS cache from target site	474
3	threat	rare component	388
4	threat	MitM redirected router traffic	333
5	threat	identify Person with access to target	227
		…	
32	threat	USB available	0
55	threat	L3 routing	0
57	threat	first target IoT device	0
65	threat	place malware directly	0

Since most of the attacks require the unintentional cooperation of a human, it is reasonable that a spam filter catches a large number of attacks by blocking the

required phishing mails. The same holds for increasing the awareness of employees.

Measures like “endpoint authentication via certificates” or “L2 segmentation” didn’t stop a single attack in this experiment.

The intended use of the described method is to check, if certain measures might be superfluous, because other measures catch all relevant attack. In Table 2 for example, the measure "End-to-End encryption" does not catch a single attempt. Of course, that does not mean, that this measure is worthless, but only that other measures seem to be more important for the specific threat under consideration.

A second benefit is to check, if a measure considered to be implemented is useful. If it does not catch (many) attacks, it might be superfluous. These checks are not meant to be absolute in the way, that they are used to decide which measures should be implemented or not, but should be considered as indicators that further investigation is needed.

In addition, looking at the successful measures, we also log successful threats. Every successful threat step, meaning it is not blocked by a measure and was found to be successful in a random walk through our tree, is logged too. With this information we can again sort by this number to get an idea which ways into our system are easy to take. For example, the threat step "poison DNS cache from target site" shown in Table 2 was successful in a large number of cases. That does not necessarily mean, that this is a real problem, because the related problems might be caught by measures not modeled in the tree, but that has to be evaluated.

If it is a real problem, meaning this threat step is not caught by other measures, action is necessary. To reduce the success probability of a critical threat step, either additional measures can be deployed or the difficulty of the threat step can be increased by techniques not modeled in the tree.

Conclusions

Based on interviews with companies, an attack defense tree for a machine in a production environment was built. The success probabilities for its threat steps and countermeasures were assigned according to the scheme described in the article at hand. The Monte Carlo simulation of the overall success probability of attacking a machine gives results for the effectiveness of common countermeasures, which are compatible with the best practices found in the interviews. It seems reasonable, that the described model for specifying probabilities can be used in practice to assess the effectiveness of countermeasures. The model described should be tested with a number of specific practical problems to identify possible issues.

Acknowledgments

Special thanks to Stephan Rogge from certerius for helpful discussions.

References

Amoroso EG (1994) *Fundamentals of Computer Security Technology*. USA, Upper Saddle River, NJ: Prentice-Hall, Inc.

Axelsen M (2018) *It Checklist for Small Business, CPA Australia.* Retrieved form https://bit.ly/2tzKEaV.

Bistarelli SM, Aglio D, Peretti P (2007) Strategic Games on Defense Trees. *LNCS* 4691: 1-15.

Edge K (2007) *A Framework for Analyzing and Mitigating the Vulnerabilities of Complex Systems via Attack and Protection Trees.* Ph.D. Dissertation, Air Force Institute of Technology. Retrieved from https://bit.ly/2lMxMNW.

Ericson II, Clifton A (1999) *Fault Tree Analysis – A History.* Proceedings of the 17th International System Safety Conference, 1999.

Fen Y, Xinchun Y, Hao H (2012) A Network Attack Modeling Method Based on MLL-AT. *Physics Procedia* 24(C): 1765-1772.

Goldberg DE (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston, MA: Addison-Wesley Longman Publishing.

Hänisch T, Rogge S (2017) IT-Sicherheit in der Industrie 4.0. [IT Security in the Industry 4.0]. In VP Andelfinger, T Hänisch (Eds.), *IT-Sicherheit in der Industrie 4.0*. Germany, Wiesbaden: Springer.

Hänisch T (2018) An Architecture for Reliable Industry 4.0 Appliances. *Athens Journal of Technology and Engineering* 5(1): 7-18.

Karg C, Hänisch T (2019) *Using an Extended Attack Defense Graph Model to Estimate the Risk of a Successful Attack on an IT Infrastructure*. Presented at 15th Annual International Conference on Information Technology & Computer Science, 20-23 May 2019, Athens, Greece.

Kordy B, Mauw S, Radomirovi´c S, Schweitzer P (2014) Attack–Defense Trees. *Journal of Logic and Computation* 24(1).

Kordy B, Jhawar R, Mauw S, Radomirovic S, Trujillo-Rasua R (2015) *Attack Trees with Sequential Conjunction*, Presented at the 30th IFIP International Information Security Conference (SEC).

Lipowski A, Lipowska D (2012) Roulette-Wheel Selection via Stochastic Acceptance. *Physica A: Statistical Mechanics and its Applications* 391(6): 2193-2196.

Löhner B (2018) *Attack-Defense-Trees and Other Security Modeling Tools, Network Architectures and Services*. Network Architectures and Services, September 2018.

OWASP Risk Rating Methodology. Retrieved from https://bit.ly/1BJAUe8.

Paul S (2014) *Towards Automating the Construction & Maintenance of Attack Trees: A Feasibility Study.* First International Workshop on Graphical Models for Security (GraMSec 2014).

Roy A, Kim D, Trivedi KS (2010) *Cyber Security Analysis using Attack Countermeasure Trees.* CSIIRW '10 Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research Article No. 28, 2010.

Schneier B (2000) *Secrets and Lies: Digital Security in a Networked World*. USA, NY: John Wiley and Sons, Inc.,

Taleb N (2007) *The Black Swan: The Impact of the Highly Improbable*. USA, NY: Random House.