Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 8 (2013) pp. XXX-XXX

Quantitative Penetration Testing with Item Response Theory

Florian Arnold[1], **Wolter Pieters**[2] **and Mariëlle Stoelinga**[1]

[1]Formal Methods & Tools Group, Department of Computer Science
University of Twente, Enschede, The Netherlands
Email: florian_arnold@hotmail.de, m.i.a.stoelinga@utwente.nl

[2]Services, Cyber Security and Safety Group, Department of Computer Science, University of Twente &
Faculty of Technology, Policy and Management, Delft University of Technology, The Netherlands
Email: w.pieters@tudelft.nl

Abstract: **Existing penetration testing approaches assess the vulnerability of a system by determining whether certain attack paths are possible in practice. Thus, penetration testing has so far been used as a qualitative research method. To enable quantitative approaches to security risk management, including decision support based on the cost-effectiveness of countermeasures, one needs quantitative measures of the feasibility of an attack. Also, when physical or social attack steps are involved, the binary view on whether a vulnerability is present or not is insufficient, and one needs some viability metric. When penetration tests are performed anyway, it is very easy for the testers to keep track of, for example, the time they spend on each attack step. Therefore, this paper proposes the concept of quantitative penetration testing to determine the *difficulty* rather than the possibility of attacks based on such measurements. We do this by step-wise updates of expected time and probability of success for all steps in an attack scenario. In addition, we show how the skill of the testers can be included to improve the accuracy of the metrics, based on the framework of item response theory (Elo ratings). We prove the feasibility of the approach by means of simulations, and discuss application possibilities.**

Keywords: item response theory, penetration testing, quantitative security, security metrics, socio-technical security.

I. Introduction

Penetration testing is a method in which testers systematically try to reach a certain target asset in an organisation by discovering and exploiting vulnerabilities, in order to determine whether real attacks would be possible. Such vulnerabilities may exist in the IT architecture, but also in physical access controls or lack of awareness of employees, enabling social engineering attacks. The results of a penetration test enable an organisation to address identified attack opportunities by the implementation of countermeasures. This approach is particularly effective when automated tools can be employed to find standard vulnerabilities in remotely accessible machines.

However, the "patch everything" approach to information or cyber security has been controversial for a long time, especially when multi-step, targeted attacks are concerned. In such attacks, remote access may be combined with physical and even social attack steps, and a determined attacker often has a reasonable chance of getting in. A recent example is the Stuxnet attack, a sophisticated cyber attack which targeted industrial installations in 2010 and aroused great interest in media and among security experts [16]. This attack was carried out over a period of several months by using highly-complex malware in combination with physical infiltration, but it also relied on human error and eventually slowed down and damaged the production of centrifugal machines of Iranian nuclear enrichment facilities. The protection against this kind of complex attacks requires minimising existing security vulnerabilities.

However, economic concerns demand that countermeasures are cost-effective, and with a limited budget, risks need to be prioritised [4]. The mere existence of an attack possibility is not sufficient to provide decision support for countermeasure investment. To support decisions, quantitative metrics for security and security risks are needed [18]. Such metrics are not always easy to obtain, as data on attacks is often not available. Penetration testing, however, may constitute the ideal setting to provide the necessary data.

Existing penetration testing approaches assess the vulnerability of a system by determining whether certain attack paths are possible in practice. Therefore, penetration testing has thus far been used as a qualitative research method. But when complex, multi-domain penetration tests involving human testers are performed anyway, it is very easy for the testers to keep track of, for example, the time they spend on each attack step. Such measurements could be used as a basis for quantitative judgements on security. Therefore, this paper proposes the concept of quantitative penetration testing, and a method to determine the *difficulty* rather than the possibility of attacks from penetration testing results. Our method is based on the social science framework of item response theory, in particular Elo ratings. We apply it to both attack steps and the testers executing those.

More concretely, we derive estimates for both the expected time and the probability of success of attack steps from penetration testing results. This can be done either statically, i.e., by calculating estimates from a data set, or iteratively, i.e.,

by updating the expected values after each attack observation. In the latter case, it is also possible to take tester skill into account, by updating both the difficulty of attack steps as well as attacker skill ratings. The skill ratings of the testers can then be used to calculate more accurate difficulty levels for the steps. We provide simulations for expected time and probability of success estimates, and show that they converge reasonably well, making the approach feasible in practical applications.

This work extends [2] and provides an extensive formalisation of the algorithmic computations involved. We show how to estimate probability of success and expected time of attack steps in case underlying observation data is incomplete, which is a common problem in the analysis of attacks. Our estimates are obtained through a step-by-step update routine, and we demonstrate how to derive statistically significant results with this technique. Finally, we describe the algorithmic framework for updating the estimates for both the attacks steps and the attacker.

Organization of the paper. In section II, we discuss the state-of-the-art and related approaches. In section III, we define the requirements for quantitative penetration testing. We formalise our basic method for quantitative penetration testing in section IV, and use item response theory to include attacker skill in section V, including the corresponding algorithms. The results of simulations are shown in section VI. We end with application opportunities in section VII and conclusions in section VIII.

II. Related work

A. Penetration Testing

Penetration testing started as a hackers' art, involving long sessions to attempt to break into an organisation via the Internet. Many attempts have been made to move towards more scientific methods (see e.g. [20]), and several automated tools for online penetration testing are now available.

Next to online methods, penetration testing may also include physical access to the facilities of an organisation [1]. In addition, social engineering can be included to determine the human weaknesses that may provide access to assets [5, 6, 9, 10, 15]. Overviews of penetration testing methods are provided in [3, 8, 11].

Penetration testing may focus either on single vulnerabilities, or may involve multi-step attacks that would lead to the assets [21]. With multi-step attacks, attack trees [19, 27] or attack nets [20] can be used as a basis for these tests.

B. Security metrics

For a long time, it has been acknowledged that operational measures of computer, information and cyber security are important [18]. Such measures, or metrics, provide insight in the vulnerability of an organisation against attacks. This enables tests of the systems against imposed security policies [22], in particular when such policies are formulated quantitatively [24]. Metrics are also needed to integrate security into traditional (non-malicious) quantitative risk management approaches [23]. However, these metrics have typically not been associated with penetration testing.

C. Item Response Theory

Item response theory is a classical method to calibrate tests, such as intelligence tests, when the skill levels of the persons taking part in the calibration is unknown. From a set of correct and incorrect responses of a set of persons to a set of items, both the skill of the persons and the difficulty of the items are estimated. The simplest case are 1-parameter or Rasch models [26]. In the Math Garden project this idea was combined with dynamic updates of the ratings.[13] This system is similar to the Elo rating used to rank chess players [7].

In [25], it was proposed to apply the framework of item response theory to security metrics. The key idea is that the likelihood of success can be estimated from attack strength and defence strength (difficulty). In this paper, rather than considering single-event attacks, we focus on multi-step attacks, in which digital, physical, and social attack vectors can be combined. Also, the proposal in [25] did not include time as a separate variable, and this is an important contribution of the present paper. We separate probability of success (related to attacker skill and step difficulty) and time or effort spent (related to attacker speed and the labour intensity of the step) into different variables. Different possibilities for including response time (RT) in item response theory models are discussed in [29].

III. Requirements

A. Parameters

To enable quantitative penetration testing, at first one has to choose the quantitative variables to be taken into account. We consider the *time* that an attacker requires to perform an attack, and the *probability* that the attack is successful. Depending on the problem context, *time* can be replaced by other parameters, such as *resources* to answer the question 'How much money does an attacker have to invest?'

We consider multi-step attacks and, thus, we assume that complex attacks are composed of elementary steps. This assumption is widely used in attack modelling formalisms such as attack trees or attack nets [14, 19, 20, 27]. In an actual attack the steps are executed sequentially.

B. Distributions

We consider a random variable X that describes the time of a successful attack step execution. We are interested in the cumulative distribution function (CDF) that represents the probability that the attack step is executed successfully within t time units, that is the function $f(t) = \mathbb{P}[X \leq t]$ of X. We need to make an assumption for the underlying family of distributions and then estimate the corresponding parameters.

Experiments on the intrusion into computer systems showed that the intrusion process consists of different phases with an exponentially distributed execution time [12]. Especially in the context of complex analysis the exponential distribution has its merits: its shape is completely defined by one parameter, it is tractable and can easily be embedded in

complex calculations. Let X be an exponentially distributed random variable, then its CDF is given by

$$\mathbb{P}(X \leq t) = 1 - e^{-\lambda t}, \text{ for any } t \in \mathbb{R}^+.$$

Its expected value is given by $\mathbb{E}[X] = \frac{1}{\lambda}$.

C. Attack steps and attacks

Formally, an attack step is a step name associated with an execution time parameter and a success probability. The attacker needs to invest an exponentially distributed amount of time to successfully execute the attack step, while the success probability describes the chance to successfully complete it.

Definition 1 *An **attack step** a is an elementary, non-refinable step in the course of an attack. Its execution time is exponentially distributed with parameter $\lambda_a \in \mathbb{R}^+$. The probability that the attacker succeeds in the execution of the step is denoted by $p_a \in [0..1]$.*

Note that if one has information about the correlation between p_a and λ_a, then one could use this in the model [29], but for simplicity, we will not pursue that direction here. The parameter λ_a intrinsically reflects the labour intensity of attack step a. As the expected execution time of an attack step is $\frac{1}{\lambda_a}$, lower values for λ_a reflect a higher labour intensity for a.

To achieve his goal, the attacker has to execute a number of attack steps. We call this sequence of attacks steps an attack scenario.

Definition 2 *An **attack scenario** A is a sequence of attack steps $A = a_1, \ldots, a_n$. We denote with $n = |A|$ the number of attack steps. The attack step $a = A[i]$ at position i is in the following abbreviated with i, as A is always clear from the context.*

An example attack scenario is a 'laptop theft', composed of the steps 'get access to room', 'cut lock' and 'escape'. The attacker has to succeed in all three steps to finish the attack successfully. The model of this attack is presented in Fig. 1. The execution of each attack step i takes an exponentially distributed time with parameter λ_i. The attacker either fails with probability $1-p_i$ or succeeds with probability p_i, and if he fails one attack step, the whole attack is aborted, indicated by the black absorbing states. If the attacker succeeds in the execution of i, he immediately starts with the execution of attack step $i+1$. The attack is successful, if all 3 attack steps succeed.

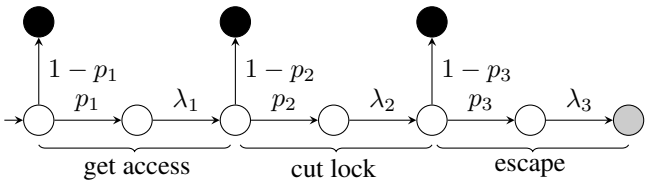


Figure. 1: The attack scenario 'laptop theft' composed of 3 attack steps, each defined by parameter λ_i and probability of success p_i, $i = 1, 2, 3$. States in which the attack fails are coloured in black. The state in which the attacker has reached his goal is coloured in grey.

We define an attacker's actual attempt as an attack execution. An attack execution consists of the attacker's plan in the form of an attack scenario and information about the duration and success for each involved attack step. If the attacker fails at a certain step, the attack is aborted and, thus, there is no information on subsequent steps.

Table 1: Four observations O_1, O_2, O_3 and O_4 of attack executions. The observation of one attack step i is denoted as tuple (a_i, r_i, t_i).

	r_E	t_E	1st step	2nd step	3rd step	4th step	5th step
O_1	1	54	$(a_3, 1, 9)$	$(a_2, 1, 14)$	$(a_7, 1, 2)$	$(a_1, 1, 20)$	$(a_2, 1, 9)$
O_2	0	17	$(a_7, 1, 3)$	$(a_2, 1, 1)$	$(a_1, 0, 13)$		
O_3	1	94	$(a_1, 1, 43)$	$(a_4, 1, \perp)$	$(a_3, 1, \perp)$	$(a_2, 1, \perp)$	
O_4	0	42	$(a_1, 1, 35)$	$(a_2, 1, \perp)$	$(a_4, \perp, \perp)$	$(a_6, \perp, \perp)$	$(a_5, \perp, \perp)$

Definition 3 *An **attack execution** $E = (A, t_1, \ldots, t_{m_1}, r_1, \ldots, r_{m_2})$ of attack scenario A consists of execution times $t_1, \ldots, t_{m_1} \in \mathbb{R}^{\geq 0}$ and results $r_1, \ldots, r_{m_2} \in \{0, 1\}$ for the involved attack steps, where t_i is the time needed to carry out step i; $r_i = 0$ indicates the failure and $r_i = 1$ the success of attack step i. The first attack step at which the attacker fails is denoted by f_E; if all attack steps are executed successfully, we define $f_E = n + 1$. We only consider execution times for successfully executed steps, so $m_1 = f_E - 1$, and results for all steps that the attacker worked on, so $m_2 = \min\{f_E, n\}$.*

An attack execution fails, if at least one attack step fails, so the result of E is defined by $r_E = \min\{r_i | i = 1, \ldots, m_2\}$. Similarly, the total execution time $t_E \in \mathbb{R}_{\geq 0}$ of E is denoted by $t_E = \sum_{j=1}^{f_E - 1} t_i$.

In practice, data about actual attacks is rare and mostly incomplete. Exact execution times of individual attack steps may not be retrievable, and it might not be possible to determine the exact point where an attack failed even though the total execution time and the result of the whole attack execution is known.

Definition 4 *An **attack observation** $O = (A, t_E, r_E, T, R)$ of an attack execution E of attack scenario A consists of the total execution time t_E, the result of the attack execution r_E, and functions $T : step \to \{t_{step}, \perp\}$ and $R : step \to \{r_{step}, \perp\}$ that define which execution times, respectively results, were observed. For $R(a) = \perp$ the result of attack step a is not known, $T(a) = \perp$ analogously. If $T(i) = t_i$ and $R(i) = r_i$ for all $i = 1, \ldots, n$, i.e. all results and times are observed, we say that the observation is complete.*

Note that the failed step f_E might not be known, in which case we set $m_1 = m_2 = n$. We assume that the attack scenario A as well as r_E and t_E are known. Note that this assumption does not imply that this information is always retrievable, since this is in general not the case. It rather suggests that it is of vital importance with respect to the analysis techniques described below.

Table 1 illustrates four example observations of attack executions. Some steps occur in more than one scenario, but not necessarily at the same index. In physical penetration testing, entering a building would typically occur often. O_1 is a complete observation of a successful attack; O_2 is a complete observation of an attack execution that failed at step a_1; O_3 is an incomplete observation of a successful attack execution in which only the execution time of the first attack step is known; O_4 is an incomplete observation of an unsuccessful attack execution in which neither all execution times nor the failed attack step is known.

D. Problem statement

On the basis of these definitions, we aim at solving the following problem:

> Under the assumption that the execution time of attack step a is exponentially distributed with parameter λ_a and its success probability governed by p_a, find good estimates $\overline{\lambda_a}$ and $\overline{p_a}$ on the basis of a number of attack observations.

IV. Basic parameter estimation

A. Static estimation

The most intuitive approach to derive estimates $\overline{\lambda_a}$ and $\overline{p_a}$ for an attack step a is the computation of the arithmetic mean, or simply mean, over a series of observations. Given a set of attack observations $\Omega = \{O_1, O_2 \dots\}$ in which an attack step a occurs multiple times, we can estimate the success probability of a by calculating the mean of our sample set. Let $r_1, \dots, r_k$ denote the observations of the results of attack step a within Ω, then

$$\overline{p_a} = \frac{1}{k} \sum_{j=1}^{k} r_j. \tag{1}$$

For example, the estimate for p_{a_1} on the basis of Table 1 is $\overline{p_{a_1}} = \frac{1+1+0+1}{4} = \frac{3}{4}$. The parameter $\overline{\lambda_a}$, which determines the shape of the CDF of the distribution of the execution time, can be derived in a similar fashion. Given observed execution times $t_1, \dots, t_k$ of attack step a within Ω, we can derive the mean of the execution time

$$\overline{t_a} = \frac{1}{k} \sum_{j=1}^{k} t_j \tag{2}$$

and use the fact that $1/\overline{\lambda_a}$ is the mean or expected value of the exponential distribution to derive $\overline{\lambda_a} = 1/\overline{t_a}$. For example, λ_{a_1} is estimated from the data in Table 1 as $\lambda_{a_1} = \frac{4}{20+13+43+35} = 0.009$. The advantage of these estimates is that they are consistent, i.e. on average we hit the true value, and unbiased, i.e. for $k \to \infty$ we hit the true value with arbitrary precision [17]. Moreover, the standard error of the mean, governed by $\frac{\sigma}{\sqrt{k}}$ with σ the standard deviation of one single observation, vanishes with increasing k. With these values we can derive confidence intervals to argue about the reliability of the estimate.

However, this approach has the following shortcomings:

- The values $r_i, \dots, r_n$ and $t_i, \dots, t_n$ have to be known. Thus, incomplete observations with either unknown execution times or unknown results have to be ignored when deriving the estimate. As we argued before, incomplete data is rather typical in attack observations and we need to find mechanisms which can deal with this;
- The number of observations k should be sufficiently large to achieve a reasonable accurate estimate. However, there is usually not sufficient amount of data about attacks available to yield good estimates. A possible solution to this dilemma is that in most cases a reasonable initial estimate for $\overline{p_a}^{init}$ and $\overline{\lambda_a}^{init}$ can be provided on the basis of expert opinion and previous experiences, and this initial estimate can then be updated when data becomes available. These updates are not possible with the static approach outlined above.

B. Dynamic estimation

To deal with the shortcomings above, we propose a technique that updates the estimates $\overline{\lambda_a}$ and $\overline{p_a}$ on a step-by-step basis. Starting from initial values $\overline{\lambda_a}^{init}$ and $\overline{p_a}^{init}$, we iteratively update these value with one observation at a time. The initial values $\overline{\lambda_a}^{init}$ and $\overline{p_a}^{init}$ can be chosen on the basis of expert opinions and previous experiences. As the quality of the observation varies from case to case, we provide update techniques for different observation scenarios.

Complete observation: all input data known Complete information about attacks can typically be derived from penetration tests. Assume we have a complete observation $O = (A, t_E, r_E, T, R)$ of an attack execution E and we want to estimate the parameters for some attack step i. Furthermore, we have initial estimates $\overline{\lambda_i}^{init}$ and $\overline{p_i}^{init}$ based on previous observations. Since $\frac{1}{\lambda_i}$ is the expected execution time, we use the observation t_i in O to obtain observation-based estimates $\overline{\lambda_i}^{O} = \frac{1}{t_i}$. We then update our initial estimates by performing a linear interpolation between $\overline{\lambda_i}^{init}$ and the observation based estimates $\overline{\lambda_i}^{O}$. If E was successful, we do so for each involved attack step; if E failed, only the steps $i = 1, \dots, f_E - 1$ that were successfully executed are updated. Note that we do not consider the execution times of attack steps that fail, because λ_i describes the execution time for successful steps. Formally, we have

$$\overline{\lambda_i} \leftarrow c_{\lambda_i} \overline{\lambda_i}^{init} + (1 - c_{\lambda_i}) \overline{\lambda_i}^{O} \ , \ i = 1, 2, \dots, f_E - 1. \tag{3}$$

The impact of the observation on the new estimate is determined by $c_{\lambda_i} \in [0,1]$. This value reflects the confidence in the previous estimate. The motivation for this parameter is that a higher confidence in the initial estimate should decrease the weight of observations on the update: $c_{\lambda_i} = 1$ expresses 100% confidence in the initial estimate, so that it is no longer updated. On the contrary, $c_{\lambda_i} = 0$ expresses absolute uncertainty. This parameter is discussed in more detail in the following paragraph.

The parameter $\overline{p_i}$ is updated analogously. From the observation we obtain the estimate $\overline{p_i}^{O} = r_i$ and derive the new estimate $\overline{p_i}$ as a linear interpolation between this value and the previous estimate $\overline{p_i}^{init}$ with confidence value c_{p_i}

$$\overline{p_i} \leftarrow c_{p_i} \overline{p_i}^{init} + (1 - c_{p_i}) r_i \ , \ i = 1, 2, \dots, f_E. \tag{4}$$

The choice of confidence values In principle, the iterative approach does not guarantee a relation with the mean values that would be obtained from a static estimation. However, below we explain how the confidence values can be choosen in such a way that it is possible to obtain the arithmetic mean with the dynamic estimation, and corresponding properties and guarantees for the accuracy of the obtained estimate hold.

Assume we want to estimate p_a from a set of observations $\Omega = \{O_1, O_2 \dots\}$. The dynamic estimation requires initial estimates $\overline{p_a}^{init}$ prior to the first update. Together with Ω they constitute the input of the dynamic estimation. The value $c_{p_a}^{init}$ expresses the, rather subjective, confidence in this initial estimate. Let $r_1, \dots, r_k$ denote the observed results of attack steps a in Ω and $c_{p_a}^j$ the confidence in $\overline{p_a}$ after the j-th update. The estimate $\overline{p_a}^1$ after the first update step is then

$$\overline{p_a}^1 = \overline{p_a}^{init} c_{p_a}^{init} + r_1(1 - c_{p_a}^{init}).$$

Recursively, the estimate $\overline{p_a}^j$ after each successive update $j = 2, \dots, k$ is

$$\overline{p_a}^j = \overline{p_a}^{j-1} c_{p_a}^{j-1} + r_j(1 - c_{p_a}^{j-1}).$$

Assume we have no information on $\overline{p_a}^{init}$, so $c_{p_a}^{init} = 0$. We want to find values $c_{p_a}^j$ for all $j = 1, \dots, k-1$ such that each result observation $r_1, \dots, r_k$ impacts the final estimate $\overline{p_a}^k$ with the same weight and we thus obtain the arithmetic mean. This is achieved by $c_{p_a}^j = \frac{1}{j}$, which yields

$$\overline{p_a}^k = (((r_1\frac{1}{2} + r_2\frac{1}{2})\frac{2}{3} + r_3\frac{1}{3})\frac{3}{4}\dots)\frac{k-1}{k} + r_k\frac{1}{k} = \frac{1}{k}\sum_{j=1}^{k} r_j.$$

We now consider the more general case that we have some knowledge about $\overline{p_a}^{init}$ with $c_{p_a}^{init} > 0$. We further want to obtain an update rule

$$c_{p_a}^j \leftarrow c_{p_a}^{j-1} + \rho, \tag{5}$$

such that $c_{p_a}^j \in [0, 1]$, and each result observation $r_1, \dots, r_k$ impacts the final estimate $\overline{p_a}^k$ with the same weight regardless of the initial confidence $c_{p_a}^{init}$, i.e.

$$\overline{p_a}^k = c_{p_a}^{init}\overline{p_a}^{init} + (1 - c_{p_a}^{init})\frac{1}{k}\sum_{j=1}^{k} r_j.$$

If we consider the last two update steps k and $k-1$, then the weight of r_k in $\overline{p_a}^k$ is $(1 - c_{p_a}^{k-1})$; while the weight of r_{k-1} is determined by $(1 - c_{p_a}^{k-2}) \cdot c_{p_a}^{k-1}$ as the product of the weights of r_{k-1} and $\overline{p_a}^{k-1}$. As this holds for all pairs $j, j-1$ with $j < k$ we obtain

$$(1 - c_{p_a}^j) = (1 - c_{p_a}^{j-1}) \cdot c_{p_a}^j.$$

We use (5) and set $c_{p_a}^j = c_{p_a}^{j-1} + \rho$. Solving the resulting equation with respect to ρ yields

$$\rho = \frac{1}{(\frac{1}{1 - c_{p_a}^{j-1}})(\frac{1}{1 - c_{p_a}^{j-1}} + 1)}. \tag{6}$$

By applying the update rule in (5) in the step-wise updates we now make sure that each prior observation has the same impact on the current estimate $\overline{p_a}^k$, independent of k. Obviously, the same holds for the estimate $\overline{\lambda_a}^k$.

Incomplete observation: unknown step times Assume we have an incomplete observation $O = (A, t_E, r_E, T, R)$ where not all execution times are observed, i.e. $T(i) = \perp$ for at least one attack step i. In the worst case we might even have $T(i) = \perp$ for all $i = 1, \dots, n$. In practice one faces this problem if information about the total execution time can be obtained but not broken down to the different attack steps. In the calculation of the mean we have to ignore all missing execution times. However, in the dynamic case we can use the current estimate $1/\overline{\lambda_i}$ for the execution time to retrieve the missing execution times.

We first consider the worst case scenario $T(i) = \perp$ for all $i = 1, \dots, n$, with t_E and f_E known. We assume that the proportion of the execution time of each attack step corresponds to the proportion on the basis of the previous estimates. In other words, we estimate

$$\overline{t_i} = t_E \cdot \frac{1/\overline{\lambda_i}}{\sum_{j=1}^{f_E - 1} 1/\overline{\lambda_j}} \;\; \forall i = 1, \dots, f_E - 1.$$

We can perform a similar estimation if any subset of the execution times of all steps is observed: Let t_E^* be the sum of all known execution times in the observation and J the set that contains the indexes of all attack steps for which the execution time is unknown, then

$$\overline{t_i} = (t_E - t_E^*) \cdot \frac{1/\overline{\lambda_i}}{\sum_{j \in J} 1/\lambda_j} \;\; \forall i \in J. \tag{7}$$

With these estimates one can then perform the updates according to (3).

Incomplete observation: unknown failed step Assume we have an incomplete observation $O = (A, t_E, r_E, T, R)$ in which some results are missing, i.e. $R(i) = \perp$ for at least one i. Since we assume that we always know the outcome r_E of the whole attack execution, this is only a problem if the attack failed, i.e. $r_E = 0$, but we do not know at which step f_E. This kind of observation occurs in practice when the attacker's goal and attack plan can be reconstructed from the reported information, but the reason for his failure remains unknown.

Assume that the last observed successful attack step in O is s and we know that the attack failed at some point after that, so that $r_i = 0$ for some $i = s+1, \dots, n$. Note that $s = 0$ if no attack step result is known. For steps $i = 1, \dots, s$ we can update with (4). For steps $i = s+1, \dots, n$ we can use our previous estimate $\overline{p_i}$ to estimate the probability $\overline{p_i^F}$ that the attack failed at this step. For the attack execution to fail at step i, the previous steps $s+1, \dots, i-1$ have to be executed successfully:

$$\overline{p_i^F}^* = (1 - \overline{p_i}) \prod_{j=s+1}^{i-1} \overline{p_j}\ ,\ i = s+1, \dots, n.$$

Since $\overline{p_i^F}$ represents the probability that the attacker fails at this step, we need $\sum_{j=s+1}^{n} \overline{p_j^F} = 1$ to hold. Thus, we normalise

$$\overline{p_i^F} = \overline{p_i^F}^* \frac{1}{\sum_{j=s+1}^{n} \overline{p_j^F}^*}\ ,\ i = s+1, \dots, n. \tag{8}$$

We can then update $\overline{p_i}$ by weighting the probability of failure $\overline{p_i^F}$ against the probability of success $\overline{p_i^S} = \sum_{j=i+1}^{n} \overline{p_j^F}$. Note that $\overline{p_i^S} + \overline{p_i^F} < 1$ if the probability that the attack failed

Table 2: Update procedures for $\overline{\lambda_i}$ and $\overline{p_i}$ on the basis of complete and incomplete observations.

Scenario	Input	Mechanism
Update of $\overline{\lambda_i}$ with complete observation	$A, t_i, f_E > i,$ $\overline{\lambda_i}, c_{\lambda_i}$	update with (3)
Update of $\overline{\lambda_i}$ with incomplete observation	$A, t_E, f_E > i,$ $\overline{\lambda_i}, c_{\lambda_i}$	find $\overline{t_i}$ with (7) update with (3)
Update of $\overline{p_i}$ with complete observation	$A, r_i,$ $\overline{p_i}, c_{p_i}$	update with (4)
Update of $\overline{p_i}$ with incomplete observation	$A, r_E = 0,$ $\overline{p_i}, c_{p_i}$	find $\overline{p_i^F}$ with (8) update with (4),(9)

before i is greater than zero. We address this uncertainty by multiplying the weight of the update by $1 - (\overline{p_i^S} + \overline{p_i^F})$:

$$\overline{p_i} \leftarrow (1-(1-c_{p_i})(\overline{p_i^S}+\overline{p_i^F}))\overline{p_i}+(1-c_{p_i})(\overline{p_i^S}\cdot 1+\overline{p_i^F}\cdot 0) \quad (9)$$

Table 2 summarises the above ideas by giving an overview on how to perform updates with either complete or incomplete observations. We remark that, in case one wants to perform a number of updates with complete and incomplete observations at the same time, one should perform updates on the complete observations first.

V. Item Response Theory Model

The more skilled and resourceful the attacker is, the more likely he will succeed in the execution of even difficult steps. It is therefore better to consider the properties of both defender and attacker in the estimations of attack step parameters. In this section we propose a model for the estimation of properties of attack steps for attacks in which the identity of the attacker is known. This is typically the case in a penetration testing setting. As both attacker skill and step difficulty are assessed, this model is very similar to what is called item response theory in social science.

The standard assumption in item response theory is that if the ratings of both "competing" actors are equal, then the probability of success is 0.5. For example, the probability of a person with skill 500 solving a problem with difficulty 500 is 0.5. A logistic distribution is typically used to relate the rating difference of the actors to the probability of success. We use a dynamic version of item response theory here, in which ratings are updated after each event. This is how the Elo rating for chess players works [7], as well as the adaptive math exercises in Math Garden [13]. Details can be found in [25].

In the original Elo framework, there is only one value to be updated. Here, like in [29], we take both probability of success and execution time into account. As we distinguish between the output parameters execution time and probability of success, we also define two parameters for each attacker, or rather attacker profile, and attack step to represent the individual impact on these two outputs. The attacker's influence upon the probability of success is called *skill*, and his *speed* impacts the execution time of an attack step. From the perspective of the attack step the probability of success is determined by its *difficulty* and the execution time by its *labour intensity*. These parameters form the basis of our analysis framework depicted in Figure 2. In the following, these parameters will be expressed by Elo ratings.

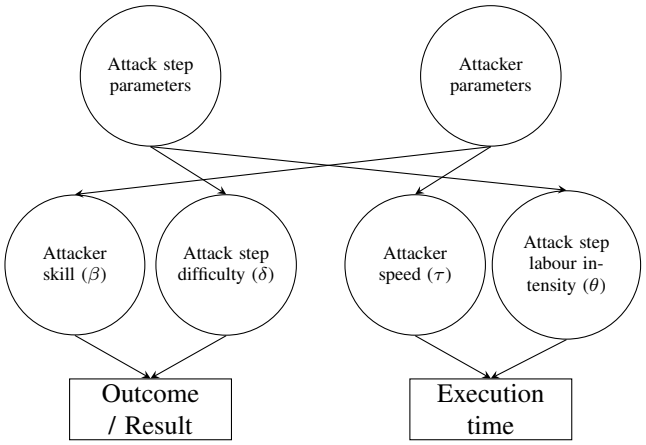


Figure. 2: Illustration of the hierachical framework for the modelling of execution times and attack outcome, in the style of [29].

In order to use the time parameter in combination with ratings, we need a definition that allows us to relate the speed and labour intensity ratings, just like the skill and difficulty ratings are related by the assumption that equality of the ratings gives 0.5 probability of success. Here, we assume that in case the ratings are equal, the expected duration is 1 time unit, or $\lambda = 1$. The dependencies of these parameters with respect to the formation of values for outcome and execution time are defined as follows.

A. Distribution of Execution Time

For an attack A with attacker j we denote with τ_j the speed of the attacker and with θ_i the labour intensity of attack step i. The relation between these two parameters is defined as in RT models [29]: the execution time t_{ij} of attack step i for attacker j can be derived by

$$t_{ij} = \frac{\theta_i}{\tau_j}. \quad (10)$$

Speed is thus defined by decomposing the execution time into two parameters, one for the speed of the attacker and one for the labour intensity of the attack step. We now want to obtain a distribution function for the execution time with respect to these two parameters. Remember that we assume the execution time of attacker j for attack step i to be exponentially distributed with parameter λ_{ij} and expected value $\frac{1}{\lambda_{ij}}$. We thus assume $\frac{1}{\lambda_{ij}} = t_{ij} = \frac{\theta_i}{\tau_j}$ and derive the CDF for the execution time X of attack step i

$$\mathbb{P}(X \leq t) = 1 - e^{-\frac{\tau_j}{\theta_i}t}, \text{ for any } t \in \mathbb{R}^+. \quad (11)$$

B. Probability for Attack Step Outcome

Similarly to above, we define β_j as the skill level of attacker j and δ_i as the difficulty of attack step i. The probability of the attacker succeeding in the attack step depends jointly on his skill and the difficulty of the attack step. We describe this probability by a logistic model (1PL or Rasch model, [26]), which expresses the probability to successfully execute

attack step i (denoted as $r_i = 1$) as

$$\mathbb{P}(r_i = 1) = \frac{e^{\beta_j - \delta_i}}{1 + e^{\beta_j - \delta_i}} = \frac{1}{1 + e^{\delta_i - \beta_j}}. \quad (12)$$

C. Updates of Elo ratings

In this section we present algorithms to systematically update the Elo ratings for θ, τ, δ and β on the basis of one single observation. In a penetration testing setting one can ask the testers to monitor the time they spent on the different attack steps precisely. Moreover, one knows the identity of the attackers, and can therefore maintain Elo ratings for each of them on the basis of past performance. In this case, for each attack step $i = 1, 2, \ldots$ one can estimate, store, and update the following information:

1. difficulty δ_i, expressed as Elo rating;
2. labour intensity θ_i expressed as Elo rating;
3. confidence c_{θ_i} of the labour intensity estimate.

Additionally, one can estimate information for each tester $j = 1, 2, \ldots$:

1. skill level β_j, expressed as Elo rating;
2. speed level τ_j, expressed as Elo rating;
3. confidence c_{τ_j} of the speed estimate.

Beyond the scope of penetration tests accurate data might not be available and we have to resort to the techniques described in section IV-B to fill the gaps.

Update algorithm for θ_i and τ_j Assume we have observed an attack execution E containing attack steps $a_1, \ldots, a_n$ and have identified attacker j. Furthermore, we have information on the timing of the attack in the form of the total execution time t_E and a subset T of the execution times of each involved attack step. The execution times are dependent on both τ_j and θ_i through equation (10), so that we update both parameters simultaneously on the basis of previous estimates. The update routine for θ_i is presented in Algorithm 1. We update τ_j and c_{τ_i} with a given observation O, with t_j being a vector containing all execution times t_{ij}. If the attack execution failed, we assume that we can only perform sensible updates if f_E is known; otherwise we are also missing execution times which we cannot estimate with (7) since it requires f_E. If the observation is incomplete and does not contain all execution times, we estimate missing data with ESTIMATEEXECUTIONTIMES by applying (7), where t_j is the vector containing all execution times. For each attack step up to $f_E - 1$, we then derive an estimate θ_i^O based on the single observation O through equation (10). We finally update θ_i with a linear interpolation between the previous estimate and θ_i^O as in (3). The impact of θ_i^O upon the update is determined by confidence value c_{θ_i}. Finally, the function UPDATECONFIDENCE updates the confidence values with (5) to make sure that each subsequent observation has the same impact upon the final estimate.

Algorithm 1 Update of $\theta_1, \ldots, \theta_{m_1}$

Require: $O, \theta_1, \ldots, \theta_{m_1}, c_{\theta_i}, \ldots, c_{\theta_{m_1}} \tau_j$
  if f_E known **then**
    if O is incomplete **then**
      $t_j \leftarrow$ ESTIMATEEXECUTIONTIMES$(T, t_E, \theta_1, \ldots, \theta_n)$
    end if
    for $i = 1, \ldots, f_E - 1$ **do**
      $\theta_i^O \leftarrow \tau_j t_{ij}$
      $\theta_i \leftarrow c_{\theta_i}\theta_i + (1 - c_{\theta_i})\theta_i^O$
      UPDATECONFIDENCE(c_{θ_i})
    end for
  end if

The update of τ_i is executed similarly. For each attack step we calculate the observation based estimate τ_{ij}^O and update by linear interpolation. We assume that the attacker's speed level does not evolve in the course of one attack and update τ_j only once on the basis of all execution times: with (10) we derive for each step an observation based estimate τ_{ij}^O and calculate the arithmetic mean τ_j^O of these values. The impact of τ_{ij}^O on the update is determined by confidence c_{τ_j}.

Algorithm 2 Update of τ_j

Require: $O, \theta_1, \ldots, \theta_{m_1}, \tau_j, c_{\tau_j}$
  if f_E known **then**
    if O is incomplete **then**
      $t_j \leftarrow$ ESTIMATEEXECUTIONTIMES$(T, t_E, \theta_1, \ldots, \theta_n)$
    end if
    $\tau_j^O \leftarrow 0$
    for $i = 1, \ldots, f_E - 1$ **do**
      $\tau_{ij}^O \leftarrow \frac{\theta_i}{t_{ij}}$
      $\tau_j^O \leftarrow \tau_j^O + \tau_{ij}^O$
    end for
    $\tau_j^O \leftarrow \frac{\tau_j^O}{f_E - 1}$
    $\tau_j \leftarrow c_{\tau_j}\tau_j + (1 - c_{\tau_j})\tau_j^O$
    UPDATECONFIDENCE(c_{τ_j})
  end if

Update algorithms for δ_i and β_j Given an attack observation O, we want to update the difficulty δ_i for each involved attack step i and the skill level β_j for attacker j. As above, we execute both updates simultaneously. The update routine for δ_i is presented in Algorithm 3. If O is incomplete and does not include the information at which step an unsuccessful attack execution failed, we first determine the last observed attack step s with GETLASTKNOWNSTEPRESULT. We then use the function ESTIMATEFAILPROB to determine for each step $i = s + 1, \ldots, m_2$ the failure probability $\overline{p_i^F}$ and store it in the vector $\overline{p^F}$. The function computes (8) by exploiting (12) to get $\overline{p_i} = \frac{1}{1+e^{\delta_i - \beta_j}}$. Note that if f_E is not known, we have $m_2 = n$. In contrast to the updates above, we cannot derive an observation based estimate, since $r_i \in \{0, 1\}$. So, instead of performing a linear interpolation, we update δ_i by calculating the difference between the expected probability p, computed with (12), and the observed result. We then add this value to the previous estimate, as in classic Elo models. Since this update routine already assures that each update equally impacts the final estimate, we do not need confidence

values in this context. However, an initial value δ_i^{init} is required prior to the first update. For steps $s+1,\ldots,m_2$ we update δ_i by using a slight adaptation of (9).

Algorithm 3 Update of $\delta_1,\ldots,\delta_{m_2}$

Require: $O, \delta_1,\ldots,\delta_{m_2},\beta_j$
  $s \leftarrow m_2$
  if O is incomplete **then**
    $s \leftarrow$ GETLASTKNOWNSTEPRESULT(O)
    $\overline{p^F} \leftarrow$ ESTIMATEFAILPROBS$(\delta_{s+1},\ldots,\delta_{m_2})$
    $\overline{p^S} \leftarrow$ ESTIMATESUCPROBS$(\overline{p^F})$
  end if
  for $i = 1,\ldots,s$ **do**
    $p = \frac{1}{1+e^{\delta_i-\beta_j}}$
    $\delta_i \leftarrow \delta_i + (p - r_i)$
  end for
  for $i = s+1,\ldots,m_2$ **do**
    $p = \frac{1}{1+e^{\delta_i-\beta_j}}$
    $\delta_i \leftarrow \delta_i + \overline{p^F}_i(p-0) + \overline{p^S}_i(p-1)$
  end for

The update procedure for attacker skill β_j of attacker j is executed analogously to Algorithm 3. In this case we update by subtracting the expected probability p from the result r_i, since a successful execution of an attack step should increase the Elo value.

Algorithm 4 Update of β_j

Require: $O, \delta_1,\ldots,\delta_{m_2},\beta_j$
  if O is incomplete **then**
    $s \leftarrow$ GETLASTKNOWNSTEPRESULT(O)
    $\overline{p^F} \leftarrow$ ESTIMATEFAILPROBS$(\delta_{s+1},\ldots,\delta_{m_2})$
    $\overline{p^S} \leftarrow$ ESTIMATESUCPROBS$(\overline{p^F})$
  end if
  for $i = 1,\ldots,s$ **do**
    $p = \frac{1}{1+e^{\delta_i-\beta_j}}$
    $\beta_j \leftarrow \beta_j + (r_i - p)$
  end for
  for $i = s+1,\ldots,m_2$ **do**
    $p = \frac{1}{1+e^{\delta_i-\beta_j}}$
    $\beta_j \leftarrow \beta_j + \overline{p^F}_i(0-p) + \overline{p^S}_i(1-p)$
  end for

VI. Simulation

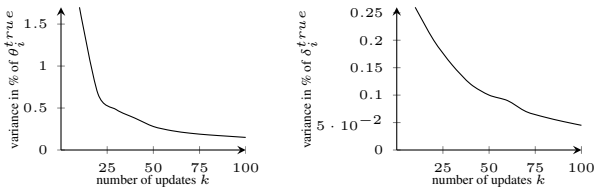
We implemented the framework in a simulation program as a proof-of-concept. Each simulation run consists of a test set that contains k observations of attacks. Each attack is randomly synthesised from a pool of attack steps. Further, each attack step in this pool has a true value θ_i^{true} for its labour intensity and a true value δ_i^{true} for its difficulty.

We want to investigate how fast the quality of the rating θ_i improves with a growing number of observations, and execute several simulation runs with varying k. In each simulation run we randomly generate k attacks; the execution times of all involved attack steps are generated randomly with the CDF in (11). The attacker speed τ_j is randomly generated for each observation.

Initially we set $c_{\theta_i} = 0$, so we assume there is no initial estimate, and use all k observations to perform step-wise updates on θ_i. To measure the accuracy of the result, we computed the sample variance in percentage of the true value for $N = 5000$ simulation runs, i.e.

$$\sigma^2_{N-1} = \frac{1}{(N-1)\theta_i^{true}} \sum_{j=1}^{N} (\theta_i^j - \theta_i^{true})^2.$$

We conducted similar experiments for the attack step difficulty δ_i. Here, the outcome of each attack step is randomly generated by using a Bernoulli distribution where the parameter is chosen according to (12). The results are shown in Figure 3.



(a) Simulation result for θ_i (b) Simulation result for δ_i

Figure. 3: The sample variance σ^2_{N-1} (in percentage) of simulation runs on θ_i and δ_i.

The step-wise update algorithms iteratively improve the quality of the estimates, especially significantly for the labour intensity. Quite accurate results with a sample variance below 0.5 can be achieved after about 25 updates.

VII. Application

The framework outlined above paves the way towards obtaining quantitative results from penetration tests. The practial applicability depends to a large extent on the goal of the measurements. If one wants to obtain statistically significant results, one would need to set up a large-scale experiment with many penetration testers. Testers need to try the same attacks in order to be able to update their ratings. This can be done for research purposes, and it has been shown for qualitative penetration testing using social engineering [6]. Based on the ideas developed in this paper, we are planning similar experiments to obtain quantitative data. However, such experiments would most likely be unrealistic in a corporate risk management setting.

Still, as our results show, one can obtain reasonable estimates with only a few attempts and a few penetration testers. A reasonable strategy for practical testing could be to let 2 or 3 penetration testers execute the same scenarios, monitor the variance of the outcomes, and hire more penetration testers only if the variance is high.

Quantitative penetration testing has the advantage that improvements in security can also be quantified. If the test is repeated after improvements have been made, the newly measured difficulty of attack steps can be compared against the previous value. With item response theory, this is possible *even if different penetration testers are involved*, assuming the ratings of the testers are sufficiently accurate.

Apart from yielding quantitative results, our proposal has another advantage: ratings may motivate penetration testers

to perform well. Our hypothesis is that penetration testers would be keen on obtaining high ratings, and therefore would be incentivised to do a good job. To test this hypothesis, one would need to run two parallel penetration tests, one with and one without the rating incentive, and evaluate differences in the time needed to succeed. Obviously, ratings could also be an incentive to cheat in reporting results, for example by reporting a shorter time than actually needed, or sharing ideas with other penetration testers, in order to increase one's rating. If this turns out to be a real problem, reporting would have to be done by an independent actor. On the other hand, security officers might be tempted to simplify attack steps hoping to increase their budget.

Finally, if there is not enough data to support difficulty ratings for steps, one can also rate organisations instead of attack steps. For each attack observation, the ratings of the attacker/tester and the rating of the organisation would then be updated. In this case, one would not obtain quantitative results of steps, but one would still have the advantage of being able to say how likely it would be that an attacker with a low rating would succeed in attacking the organisation.

The main application of quantitative penetration testing is foreseen in quantitative security risk management, requiring quantitative security metrics. Based on the Risk Taxonomy of The Open Group [28], which we use in our project, quantitative penetration testing provides a metric for the vulnerability of the organisation to attacks. However, in order to fully estimate risk, metrics for the expected frequency of attacks and the impact of attacks are needed as well. These are not trivial, and especially a suitable model of the (real) attackers is required to estimate their behaviour in response to the perceived gain, effort (time), and probability of success. We address such questions in other papers. Here, the claim is that our new proposal for quantitative penetration testing provides an important step towards fully quantitative security risk management, and in particular decision support for investment in countermeasures.

VIII. Conclusions and Discussion

In this paper, we have presented a model-based framework for quantitative penetration testing, which is the first such framework as far as we are aware of. The approach features the registration of the time taken in testing, and the calculation of the difficulty of attack steps based on the time and the skill of the tester. The skill of the tester is also updated based on the performance in the tests.

Beyond the scope of penetration tests the approach can as well be used with real attack data, but in a more limited sense, since the identity of the attacker is unknown, and the time for the individual steps may not be available either.

The main limitation of the approach lies in the amount of data required to obtain statistically significant results. However, as we have discussed, in many practical settings it may be sufficient to gain reasonable confidence in the estimates by repeating the test scenarios a few times, and monitoring the variance in the outcomes. In any case, the simple addition of time metrics to penetration testing already improves upon the existing situation in terms of the information provided for security risk management purposes.

One possible extension would be separating the time spent by the attacker, and the total time elapsed before success. This would for example be relevant in a phishing attack, in which the time spent per attempt is negligible, but the time until success may be much longer. Another extension involves multiple skill ratings for the testers, for example separating their hacking, physical access, and social engineering skills. The rating to be updated is then dependent on the nature of the attack step.

In the future, we plan to use this approach for gathering data on the difficulty of attack steps, including social engineering, to be used in the case studies in our current project. We expect the case studies to provide insights for further extensions of the framework.

Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement ICT-318003 (TRE$_S$-PASS). This publication reflects only the authors' views and the Union is not liable for any use that may be made of the information contained herein.

References

[1] W. Allsopp. *Unauthorised Access: Physical Penetration Testing For IT Security Teams*. Wiley, Chichester, 2009.

[2] F. Arnold, W. Pieters, and M.I.A. Stoelinga. Quantitative penetration testing with item response theory. In *9th Int. Conf. on Information Assurance and Security*. IEEE, 2013.

[3] M. Bishop. About penetration testing. *Security & Privacy, IEEE*, 5(6):84–87, 2007.

[4] Bob Blakley, Ellen McDermott, and Dan Geer. Information security is information risk management. In *Proceedings of the 2001 Workshop on New Security Paradigms, NSPW '01*, pages 97–104. ACM, 2001.

[5] J. P. Ceraolo. Penetration testing through social engineering. *Information systems security*, 4(4):37–48, 1996.

[6] T. Dimkov, W. Pieters, and P. H. Hartel. Two methodologies for physical penetration testing using social engineering. In *Proceedings of ACSAC*, pages 399–408. Centre for Telematics and Information Technology University of Twente, 2010.

[7] A. Elo. *The rating of Chessplayers, Past and present*. Arco Publishers, New York, 1978.

[8] S. Furnell and M. Papadaki. Testing our defences or defending our tests: the obstacles to performing security assessment references. *Computer Fraud & Security*, 2008(5):8–12, 2008.

[9] R. Gula. Broadening the scope of penetration testing techniques. *Enterasys Networks*, 1999.

[10] H. Hasle, Y. Kristiansen, K. Kintel, and E. Snekkenes. Measuring resistance to social engineering. In *Information Security Practice and Experience*, pages 132–143. Springer, 2005.

[11] A. Hudic, L. Zechner, S. Islam, C. Krieg, E.R. Weippl, S. Winkler, and R. Hable. Towards a unified penetration testing taxonomy. In *2012 International Conference on Privacy, Security, Risk and Trust (PASSAT), and 2012 International Confernece on Social Computing (SocialCom)*, pages 811–812. IEEE, 2012.

[12] E. Jonsson and T. Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *IEEE Transactions on Software Engineering*, 23(4):235–245, 1997.

[13] S. Klinkenberg, M. Straatemeier, and H. L. J. van der Maas. Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation. *Comput. Educ.*, 57:1813–1824, 2011.

[14] B. Kordy, L. Pietre-Cambacedes, and P. Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 2014.

[15] I. Kotenko, M. Stepashkin, and E. Doynikova. Security analysis of information systems taking into account social engineering attacks. In *22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 611–618. IEEE, 2011.

[16] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE*, 9(3):49 –51, 2011.

[17] E.L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer Texts in Statistics. Springer, 1998.

[18] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2(2–3):211–229, 1993.

[19] S. Mauw and M. Oostdijk. Foundations of attack trees. In *International Conference on Information Security and Cryptology, ICISC 2005. LNCS 3935*, pages 186–198. Springer, 2006.

[20] J. P. McDermott. Attack net penetration testing. In *Proceedings of the 2000 workshop on New security paradigms*, pages 15–21. ACM, 2001.

[21] V. Nunes Leal Franqueira, R.H.C. Lopes, and P.A.T. van Eck. Multi-step attack modelling and simulation (MsAMS) framework based on mobile ambients. In *Proceeding of the 24th Annual ACM Symposium on Applied Computing, SAC'2009*, pages 66–73, New York, 2009. ACM.

[22] W. Pieters, T. Dimkov, and D. Pavlovic. Security policy alignment: A formal approach. *Systems Journal, IEEE*, 7(2):275–287, 2013.

[23] W. Pieters, Z. Lukszo, D. Hadžiosmanović, and J. van den Berg. Reconciling malicious and accidental risk in cyber security. *Journal of Internet Services and Information Security*, 4(2):4–26, 2014.

[24] W. Pieters, J. Padget, F. Dechesne, V. Dignum, and H. Aldewereld. Effectiveness of qualitative and quantitative security obligations. *Journal of Information Security and Applications*, 2014.

[25] W. Pieters, S. H. G. Van der Ven, and C. W. Probst. A move in the security measurement stalemate: elo-style ratings to quantify vulnerability. In *Proceedings of the 2012 Workshop on New Security Paradigms*, pages 1–14. ACM, 2012.

[26] G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests.* MESA Press, 1960.

[27] B. Schneier. Attack trees: Modeling security threats. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[28] The Open Group. Risk taxonomy. Technical Report C081, The Open Group, 2009.

[29] W. J. van Der Linden. Conceptual issues in response-time modeling. *Journal of Educational Measurement*, 46(3):247–272, 2009.

Author Biographies

Florian Arnold was born in Bergisch Gladbach, on the 29th of June 1986. He obtained the Master's degree in Operations Research from the Clausthal University of Technology in Germany (2012) and continued his research as PhD student in the Formal Methods and Tools Group of the University of Twente, the Netherlands. In the course of the European TRESPASS project he developed stochastic model checking techniques to enable quantified risk-analysis of socio-technical systems .

Wolter Pieters has Master degrees in both computer science (2002) and philosophy of technology (2003) from the University of Twente, and a PhD degree in information security from Radboud University Nijmegen, the Netherlands (2008). Currently he is technical leader of the TRESPASS project at the University of Twente, and assistant professor cyber risk at Delft University of Technology. In the TRESPASS project, he addresses cyber security risk management in socio-technical systems through the concept of attack navigators, including research on security policies and security metrics. He also published on electronic voting, verification of security properties, and philosophy and ethics of cyber security.

Mariëlle Stoelinga is an associate professor in ICT risk management at the university of Twente, the Netherlands. She holds an MSc and a PhD degree from Radboud University Nijmegen, the Netherlands, and has been a postdoctoral scholar at the University of California at Santa Cruz, USA. Now, she leads a research team that is involved in serveral EU and Dutch projects, including TRESPASS project on cyber security risks.