

Quantitative Penetration Testing with Item Response Theory (extended version)

Florian Arnold*, Wolter Pieters† and Mariëlle Stoeltinga*

**Formal Methods & Tools Group, Department of Computer Science
University of Twente, Enschede, The Netherlands*

Email: f.arnold@utwente.nl, m.i.a.stoeltinga@utwente.nl

†*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 implementing 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. Economic

concerns demand that countermeasures are cost-effective, and with a limited budget, risks need to be prioritised [1]. Statements like “there is a problem” are not sufficient to provide decision support for countermeasure investment. In order to support decisions, quantitative metrics for security and security risks are needed [2]. Such measures are not always easy to obtain, as data on attacks is often not shared. 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.

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, calculating estimates from a data set, or iteratively, 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 success probability estimates, and show that they converge reasonably well, making the approach feasible in practical applications.

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. 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. [3]), and several automated tools for online testing are now available.

Next to online methods, penetration testing may also include physical access to the facilities of an organisation[4]. In addition, social engineering can be included to determine the human weaknesses that may provide access to assets [5], [6], [7], [8], [9]. Overviews of penetration testing methods are provided in [10], [11], [12].

Penetration testing may focus on single vulnerabilities, or may involve multi-step attacks that would lead to the assets [13]. With multi-step attacks, attack trees [14], [15] or attack nets [3] may be used as a basis for the tests. Attack trees can be annotated with the expected time needed for the attack steps, so if attack trees are used, they can be annotated with the time taken in the tests as a basis for further calculations on the trees.

B. Item Response Theory

Item response theory is typically used to calibrate tests, such as intelligence tests, when the skill levels of the persons taking part in the calibration is not known. 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 can be estimated. The simplest case are 1-parameter or Rasch models [16]. In the Math Garden project, this idea was combined with dynamic updates of the ratings, rather than a separate calibration phase [17]. This system is similar to the one used to rank players in chess [18].

In [19], 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 defense 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 [19] 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 intensiveness of the step) into different variables. Different possibilities for including response time (RT) in Item Response Theory models are discussed in [20].

III. REQUIREMENTS

A. Parameters

In order to enable quantitative penetration testing, the first thing to do is 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 many resources does an attacker have to invest?'. We can also replace *time* by *knowledge* to determine 'How much does the attacker have to know about the attacked system?'.

In the following section, we will focus on the parameter *time* for the simplicity of its measurement. We consider multi-step attacks and, thus, we assume that complex attacks are synthesized by elementary steps which are sequentially executed one after another. This assumption is widely used in attack modelling formalisms like attack trees or attack nets [14], [15], [3].

B. Distributions

We consider a random variable X that describes the time of an successful attack execution. We are interested in the cumulative distribution function (CDF) that the system is still not compromised at time t , that is the function $f(t) = \mathbb{P}[X \leq t]$ of X . Since any CDF always approaches 1 for increasing t , this implies that an attacker always succeeds if he is given sufficient time.

However, for most attacks there is a chance that the attacker is caught or aborts the attack attempt after a certain time. We consider this possibility of failure by deriving (1) the probability that the attack step is executed successfully, independent of time t , and (2) the CDF which describes the execution time of an attack step, on the condition that the attack step is successfully executed.

For (2) we need to make an assumption for the underlying family of distributions and then estimate the corresponding parameters. Many time-dependent modelling formalisms make use of the exponential distribution to express the evolution of the attacker's success as time progresses. Empirical underpinning for this modelling choice is given in [21] where experiments on the intrusion into computer systems yielded the result, that the intrusion process consists of different phases with an exponentially distributed execution time. Let X be an exponentially distributed random variable, then it is defined by its cumulative distribution function (CDF)

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

Especially in the context of complex analysis the exponential distribution has its merits: Its shape is completely defined by only one parameter, it is tractable and can easily be embedded in complex calculations. Its expected value is given by

$$\mathbb{E}[X] = \frac{1}{\lambda}.$$

One can argue that the simplicity of the exponential distribution does not accurately reflect the nature of human

behaviour as it is usually characterized by fluctuations. RT models try to repair this flaw by using Weibull distributions or log normal distribution [20]. However, the possibly higher precision of these distributions comes at the expense of more input parameters. In other words, the exploitation of the full potential of these families of distribution requires sufficient data to estimate the parameters. As in the security context data is a good in short supply, this is the main reason to resort to the exponential distribution.

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 [20], 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 $A = a_1, \dots, a_n$. 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 attack scenario is illustrated in Figure 1. The scenario consists of n attacks steps which are carried out one after another. 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 in 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 n attack steps have been successfully compromised. 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 there is thus no information on subsequent steps.

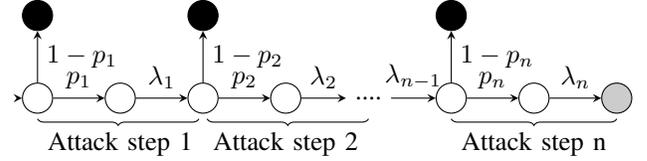


Figure 1. An attack scenario with n attack steps, each defined by parameter λ_i and success probability p_i , $i = 1, \dots, n$. States in which the attack fails are colored in black. The state in which the attacker has reached his goal is colored in gray.

Definition 3: An attack execution

$E = (A, t_1, \dots, t_{m_1}, r_1, \dots, r_{m_2})$ of attack scenario A consists of execution times $t_1, \dots, t_{m_1} \in \mathbb{R}^{\geq 0}$ and results $t_1, \dots, t_{m_2} \in \{0, 1\}$ 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 $r_E = \min\{r_i | i = 1, \dots, m_2\}$. Similarly, the total execution time of E is denoted by $t_E = \sum_{j=1}^{f_E-1} t_j$.

In practice, data about actual attacks is rare and mostly incomplete. Exact execution times of individual attack steps may not be retrievable, it might not be possible to determine the exact point where an attack failed, but 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 $T : \text{step} \rightarrow \{t_{\text{step}}, \perp\}$ and $R : \text{step} \rightarrow \{r_{\text{step}}, \perp\}$ 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 $i = 1, \dots, n$, 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. This assumption should not imply that these information are always retrievable, as this is in general not the case, but rather suggest that they are of vital importance with respect to the analysis techniques described below.

Table I 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

Table I
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)

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 $\bar{\lambda}_a$ and \bar{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 $\bar{\lambda}_a$ and \bar{p}_a for an attack step a is the computation of the mean average 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 average of our sample set. Let r_1, \dots, r_k denote the observations of the results of attack step a within Ω , then

$$\bar{p}_a = \frac{1}{k} \sum_{j=1}^k r_j. \quad (1)$$

For example, the estimate for p_{a_1} on the basis of Table I is $\bar{p}_{a_1} = \frac{1+1+0+1}{4} = \frac{3}{4}$. The parameter $\bar{\lambda}_a$, 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 average of the execution time

$$\bar{t}_a = \frac{1}{k} \sum_{j=1}^k t_j \quad (2)$$

and use the fact that $1/\bar{\lambda}_a$ is the expected value of the exponential distribution to derive $\bar{\lambda}_a = 1/\bar{t}_a$. For example, λ_{a_1} is estimated from the data in Table I 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 $k \rightarrow \infty$ we hit the true value

with arbitrary precision. Moreover, the standard error of the average 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 for instance 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. Unknown execution times and results in incomplete observations have to be ignored in 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 \bar{p}_a^{init} and $\bar{\lambda}_a^{init}$ can be provided on the basis of expert opinion and previous experiences, and this initial estimate can then be updated when data become 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 $\bar{\lambda}_a$ and \bar{p}_a on a step-by-step basis. Starting from initial values $\bar{\lambda}_a^{init}$ and \bar{p}_a^{init} , we iteratively update these value with one observation at a time. The initial values $\bar{\lambda}_a^{init}$ and \bar{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 $\bar{\lambda}_i^{init}$ and \bar{p}_i^{init} based on previous observations. Since $\frac{1}{\bar{\lambda}_i}$ is the expected execution time, we use the observation t_i in O to obtain observation based estimates $\bar{\lambda}_i^O = \frac{1}{t_i}$. We then update our initial estimates by performing a linear interpolation between $\bar{\lambda}_i^{init}$ and the observation based estimates $\bar{\lambda}_i^O$. 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

$$\bar{\lambda}_i \leftarrow c_{\lambda_i} \bar{\lambda}_i^{init} + (1 - c_{\lambda_i}) \bar{\lambda}_i^O, \quad i = 1, 2, \dots, f_E - 1. \quad (3)$$

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 \bar{p}_i is updated analogously. From the observation we obtain the estimate $\bar{p}_i^O = r_i$ and derive the new estimate \bar{p}_i as a linear interpolation between this value and the previous estimate \bar{p}_i^{init} on the basis of confidence value c_{p_i}

$$\bar{p}_i \leftarrow c_{p_i} \bar{p}_i^{init} + (1 - c_{p_i}) r_i, \quad i = 1, 2, \dots, f_E. \quad (4)$$

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, the confidence values can be chosen in such a way that it is possible to obtain a mean average 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 \bar{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 \bar{p}_a after the j -th update. The estimate \bar{p}_a^1 after the first update step is then

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

Recursively, the estimate \bar{p}_a^j after each successive update $j = 2, \dots, k$ is

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

Assume we have no information on \bar{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 \bar{p}_a^k with the same weight and we thus obtain a mean average. This is achieved by $c_{p_a}^j = \frac{1}{j}$, which yields

$$\bar{p}_a^k = \left(\left(\left(r_1 \frac{1}{2} + r_2 \frac{1}{2} \right) \frac{2}{3} + r_3 \frac{1}{3} \right) \frac{3}{4} \dots \right) \frac{k-1}{k} + r_k \frac{1}{k} = \frac{1}{k} \sum_{j=1}^k r_j.$$

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

$$c_{p_a}^j \leftarrow c_{p_a}^{j-1} + \rho, \quad (5)$$

such that $c_{p_a}^j \in [0, 1]$, and each result observation r_1, \dots, r_k impacts the final estimate \bar{p}_a^k with the same weight regard-

less of the initial confidence $c_{p_a}^{init}$, i.e.

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

If we consider the last two update steps k and $k - 1$, then the weight of r_k in \bar{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 \bar{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

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

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 \bar{p}_a^k , independent of k . Obviously, the same holds for the estimate $\bar{\lambda}_a^k$.

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 average we have to ignore all missing execution times. However, in the dynamic case we can use the current estimate $1/\bar{\lambda}_i$ 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

$$\bar{t}_i = t_E \cdot \frac{1/\bar{\lambda}_i}{\sum_{j=1}^{f_E-1} 1/\bar{\lambda}_j} \quad \forall i = 1, \dots, f_E - 1.$$

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

$$\bar{t}_i = (t_E - t_E^*) \cdot \frac{1/\bar{\lambda}_i}{\sum_{j \in J} 1/\bar{\lambda}_j} \quad \forall i \in J. \quad (7)$$

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

Incomplete observation: unknown failed step: Assume we have an incomplete observation

$O = (A, t_E, r_E, T, R)$ where not all results are observed, 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 \bar{p}_i to estimate the probability \bar{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:

$$\bar{p}_i^{F*} = (1 - \bar{p}_i) \prod_{j=s+1}^{i-1} \bar{p}_j, \quad i = s + 1, \dots, n.$$

Since \bar{p}_i^F represents the probability that the attacker fails at this step, we need $\sum_{j=s+1}^n \bar{p}_j^F = 1$ to hold. Thus, we normalize

$$\bar{p}_i^F = \bar{p}_i^{F*} \frac{1}{\sum_{j=s+1}^n \bar{p}_j^{F*}}, \quad i = s + 1, \dots, n. \quad (8)$$

We can then update \bar{p}_i by weighting the failure probability \bar{p}_i^F against the success probability $\bar{p}_i^S = \sum_{j=i+1}^n \bar{p}_j^F$. Note that $\bar{p}_i^S + \bar{p}_i^F < 0$ if the probability that the attack failed before i is greater than zero. We address this uncertainty by decreasing the weight of the update by $1 - (\bar{p}_i^S + \bar{p}_i^F)$:

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

Table II summarizes the above ideas by giving an overview on how to perform updates with either complete and 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. INCLUDING ATTACKER SKILL

A. Attacker model

In the previous section, we assumed that one typically does not know the identity of the attacker. In this case, one basically assumes that the attacks launched against the system come from a single “attacker”, who is representative for the total population of attackers.

In this section, we propose a model for situations in which the identity of the attacker is known. This is typically the case in a penetration testing setting, and therefore the

Table II
UPDATE PROCEDURES FOR $\bar{\lambda}_i$ AND \bar{p}_i ON THE BASIS OF COMPLETE AND INCOMPLETE OBSERVATIONS.

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

approach presented here is the main one to be used in quantitative penetration testing. As both attacker skill and step difficulty are assessed, this model is very similar to what is called Item Response Theory in social science.

B. Item Response Theory

Item Response Theory is a means to gather data on skill and difficulty levels without having a priori information. Typically, it can be used to calibrate tests that measure a certain ability. A group of persons is asked to solve test items, and based on the results, both the ability of the persons and the difficulty of the items is estimated. This static calibration approach is reflected in Rasch models [16], as well as multi-parameter variants.

Next to the static setting, Item Response Theory can also be applied iteratively, by updating ratings after each event. A famous application of such dynamic models occurs in chess, where so-called Elo ratings are used to rank players [18]. Players start with a default value, and gain or lose points based on the result of the matches they play. In particular, the difference between the actual result and the expected result determines how the ratings are updated. There is only one type of entity here (players), with associated skill levels.

In the Math Garden project [17], dynamic ratings are applied to both persons (skill) and test items (difficulty), in order to offer children math problems tailored to their skill level. Again, both ratings are updated based on the difference between actual and expected result.

The similarity of the latter approach to security, where attackers play “matches” against systems, was noted in [19]. The paper focused on a general risk management setting, and mentioned that for malicious settings, data could be hard to obtain. The authors suggested the application to penetration testing, for reasons of data availability, but did not discuss the details.

For the dynamic setting, initial (default) ratings are necessary to bootstrap the system. In chess, for example, all players start with the same default rating. When additional information is available, one could decide to adjust the initial ratings to previous knowledge about skill and difficulty. In the security field, initial values for dynamic ratings may be set based on expert judgement, or results of earlier scientific

experiments, when such data are already available from earlier risk assessments.

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.

In the original Elo framework, there is only one value to be updated. Here, we take both probability of success and time into account. As we distinguish between the output parameters execution time and success probability, we also define two parameters for each person and attack step to represent the individual impact on these two outputs. The attacker’s influence upon the success probability is called *skill*, and his *speed* impacts the execution time of an attack step. From the perspective of the attack step, the success probability 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. The idea of parameter separation is also found in Item

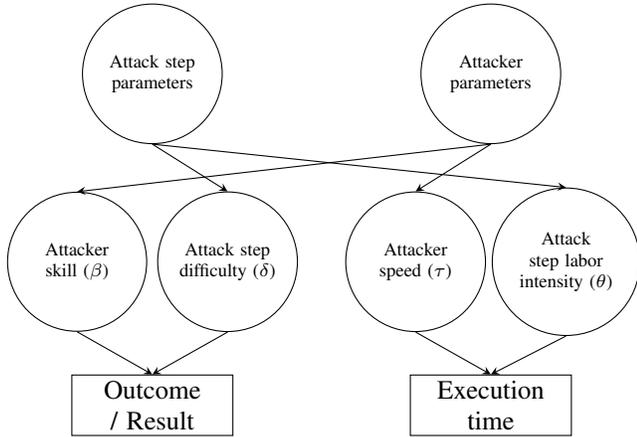


Figure 2. Illustration of the hierarchical framework for the modelling of execution times and attack outcome.

Response Theory: “RT [response time] models with speed as a person parameter should also have an item parameter for their time intensity” [20].

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 unit of time, or $\lambda = 1$. The dependencies of these parameters with respect to the formation of values for outcome and execution time are defined as follows.

Distribution of Execution time: For an observation of 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 .

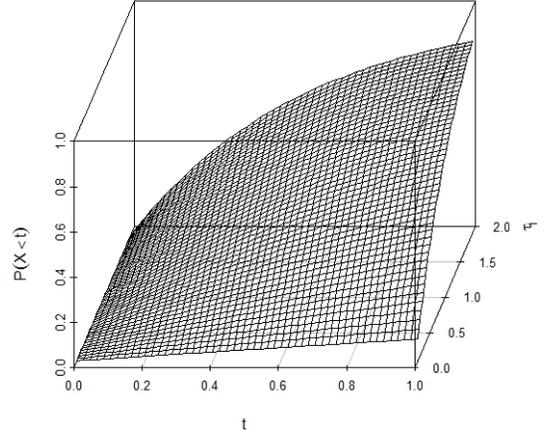


Figure 3. The CDF $\mathbb{P}(X \leq t)$ for the execution time of attack step i in dependence of attacker speed τ_j . The labour intensity is fixed as $\theta_i = 1$

The relation between these two parameters is defined as in RT models [20]: 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 can thus assume that $\frac{1}{\lambda_{ij}} = t_{ij} = \frac{\theta_i}{\tau_j}$ and derive the CDF for the execution time X of attack step i by attacker j

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

This CDF in dependence of τ_j is visualized in Figure 3.

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. The standard model is the Rasch model, or one parameter logistic model (1PL model) [16], which expresses the probability to successfully penetrate attack step i 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, based on past performance. In this case, for each attack step $i = 1, 2, \dots$ one can estimate, store, and update the following information:

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

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

- 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 4.2 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, \dots, 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

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

Require: $O, \theta_1, \dots, \theta_{m_1}, c_{\theta_1}, \dots, c_{\theta_{m_1}}, \tau_j$

if f_E known **then**

if O is incomplete **then**

$t_j \leftarrow \text{ESTIMATEEXECUTIONTIMES}(T, t_E, \theta_1, \dots, \theta_n)$

end if

for $i = 1, \dots, f_E - 1$ **do**

$\theta_i^O \leftarrow \tau_j t_{ij}$

$\theta_i \leftarrow c_{\theta_i} \theta_i + (1 - c_{\theta_i}) \theta_i^O$

$\text{UPDATECONFIDENCE}(c_{\theta_i})$

end for

end if

presented in Algorithm 1. We update τ_j and c_{τ_j} 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 f_E is unknown as well. If the observation is incomplete and does not contain all execution time, we estimate the missing execution times with $\text{ESTIMATEEXECUTIONTIMES}$ 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 the confidence c_{θ_i} . Finally, the function UPDATECONFIDENCE updates the confidence values with (5) to make sure that each observation has the same impact upon the final estimate.

The update of τ_j 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 mean average 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: $O, \theta_1, \dots, \theta_{m_1}, \tau_j, c_{\tau_j}$

if f_E known **then**

if O is incomplete **then**

$t_j \leftarrow \text{ESTIMATEEXECUTIONTIMES}(T, t_E, \theta_1, \dots, \theta_n)$

end if

$\tau_j^O \leftarrow 0$

for $i = 1, \dots, f_E - 1$ **do**

$\tau_{ij}^O \leftarrow \frac{\theta_i}{t_{ij}}$

$\tau_j^O \leftarrow \tau_j^O + \tau_{ij}^O$

end for

$\tau_j^O \leftarrow \frac{\tau_j^O}{f_E - 1}$

$\tau_j \leftarrow c_{\tau_j} \tau_j + (1 - c_{\tau_j}) \tau_j^O$

$\text{UPDATECONFIDENCE}(c_{\tau_j})$

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 $\text{GETLASTKNOWNSTEPRESULT}$. We then use the function ESTIMATEFAILPROB to determine for each step $i = s + 1, \dots, m_2$ the failure probability \bar{p}_i^F and store it in the vector \bar{p}^F . The function computes (8) by exploiting (12) to get $\bar{p}_i = \frac{1}{1 + e^{\delta_i - \beta_j}}$. 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 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 $s + 1, \dots, m_2$ we update δ_i by using a slight adaptation of

(9).

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

Require: $O, \delta_1, \dots, \delta_{m_2}, \beta_j$

$s \leftarrow m_2$

if O is incomplete **then**

$s \leftarrow \text{GETLASTKNOWNSTEPRESULT}(O)$

$\overline{p^F} \leftarrow \text{ESTIMATEFAILPROBS}(\delta_{s+1}, \dots, \delta_{m_2})$

$\overline{p^S} \leftarrow \text{ESTIMATESUCPROBS}(\overline{p^F})$

end if

for $i = 1, \dots, s$ **do**

$p = \frac{1}{1+e^{\delta_i-\beta_j}}$

$\delta_i \leftarrow \delta_i + (p - r_i)$

end for

for $i = s + 1, \dots, m_2$ **do**

$p = \frac{1}{1+e^{\delta_i-\beta_j}}$

$\delta_i \leftarrow \delta_i + \overline{p^F}(p - 0) + \overline{p^S}(p - 1)$

end for

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

Algorithm 4 Update of β_j

Require: $O, \delta_1, \dots, \delta_{m_2}, \beta_j$

if O is incomplete **then**

$s \leftarrow \text{GETLASTKNOWNSTEPRESULT}(O)$

$\overline{p^F} \leftarrow \text{ESTIMATEFAILPROBS}(\delta_{s+1}, \dots, \delta_{m_2})$

$\overline{p^S} \leftarrow \text{ESTIMATESUCPROBS}(\overline{p^F})$

end if

for $i = 1, \dots, s$ **do**

$p = \frac{1}{1+e^{\delta_i-\beta_j}}$

$\beta_j \leftarrow \beta_j + (r_i - p)$

end for

for $i = s + 1, \dots, m_2$ **do**

$p = \frac{1}{1+e^{\delta_i-\beta_j}}$

$\beta_j \leftarrow \beta_j + \overline{p^F}(0 - p) + \overline{p^S}(1 - p)$

end for

VI. SIMULATION

We implemented the above algorithms in a simulation program as a proof-of-concept. Each simulation run consists of a test set containing a number k of attack observations. Each attack is randomly synthesized from a pool of 10 attack steps. Each attack step has a 50% chance to end up in an attack observation; thus, on average one attack consists of 5 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} .

Simulation runs on θ_i : We want to investigate the impact of the number k of observations and the impact of the quality of observations on the accuracy of the resulting estimate of the labour intensity θ_i . For this, we execute simulation runs with varying k . In each simulation run, we generate k observations, where observed execution times t_1, \dots, t_{f_E-1} are generated randomly according to (11) with θ_i^{true} and τ_j as inputs. The attacker strength b_i is randomly generated for each observation. Furthermore, the probability that an observation is incomplete is determined by $p_I \in [0, 1]$. All incomplete observations have $T(i) = \perp$ for each involved attack step i , so only the total execution time of the whole attack is known.

We set $c_{\theta_i}^{init} = 0$, so we assume there is no initial estimate, and use all k observations to perform stepwise updates on θ_i according to Algorithm 1. To measure the accuracy of the result, we computed the sample variance, in percentage of the true value, for $N > 1000$ simulation runs for different pairs of parameters k, p_I , i.e.

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

The results are shown in Figure 4.

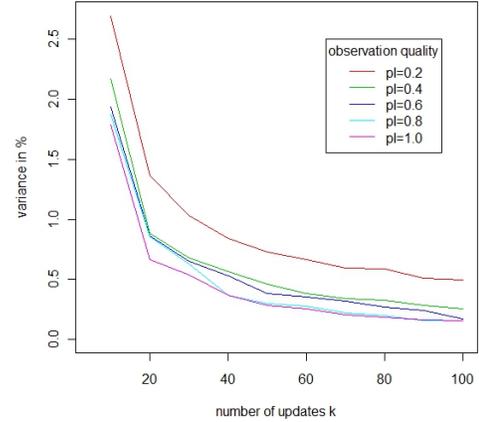


Figure 4. The sample variance σ_{N-1}^2 (in percentage) of simulation runs on θ_i^k for different parameter pairs k and p_I .

Simulation runs on δ_i : We conducted similar experiments to determine the impact of the number k of observations and the impact of the quality of observations on the accuracy of the estimate of the attack step difficulty δ_i^k . We choose a similar set-up as above: the results r_1, \dots, r_{m_2} of each observation are generated with (12), with δ_i^{true} and β_j as inputs. The probability that an observation is incomplete is determined by p_I . All incomplete observations have $R(i) = \perp$ for each attack step i , so we do not know the outcome of any involved attack step, but only the outcome of the whole attack. We further have $c_{\delta_i}^{init} = 0$. The results are shown in Figure 5.

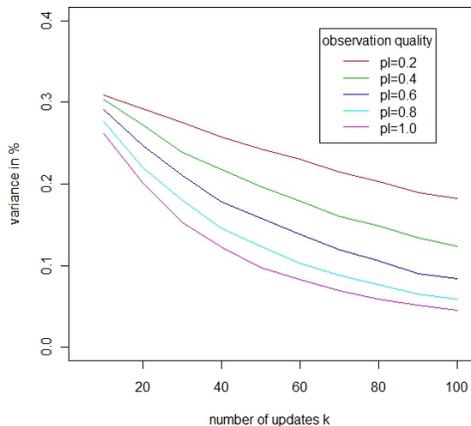


Figure 5. The sample variance σ_{N-1}^2 (in percentage) of simulation runs on δ_i^k for different parameter pairs k and p_I .

Conclusions from simulations: The stepwise update algorithms iteratively improve the quality of the estimates and yields quite accurate results after about 20 to 40 updates. Performing updates on the complete observations first and providing a good initial estimate can improve the results further. The consideration of incomplete observations does not seem to improve the accuracy of δ_i significantly, since one obtains similar results when ignoring them. However, for θ_i they visibly improve the accuracy. Thus, incomplete observations should be considered in the estimation process.

VII. APPLICATION

The framework outlined above paves the way towards obtaining quantitative results from penetration tests. The practical 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 in the outcomes, and hire more pentesters 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 can be compared against the previous value. With Item Reponse 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, with and 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, 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.

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, which we use in our project, quantitative penetration testing provides a metric for the vulnerability of the organisation to attack steps and 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 presented a framework for quantitative penetration testing, which is the first such framework as far as we are aware. 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.

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. We have provided methods for such settings as well, and these are somewhat simpler, as they do not

rely on Item Response Theory for taking attacker skill into account.

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 (effort) spent by the attacker, and the real time elapsed before success. This would for example be relevant in a phishing attack, in which the effort (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 attack step category.

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 (TRES-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] B. Blakley, E. McDermott, and D. Geer, "Information security is information risk management," in *Proceedings of NSPW*. New York, NY, USA: ACM, 2001, pp. 97–104.
- [2] 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*, vol. 2, no. 2–3, pp. 211–229, 1993.
- [3] J. P. McDermott, "Attack net penetration testing," in *Proceedings of the 2000 workshop on New security paradigms*. ACM, 2001, pp. 15–21.
- [4] W. Allsopp, *Unauthorised Access: Physical Penetration Testing For IT Security Teams*. Wiley, 2010.
- [5] J. P. Ceraolo, "Penetration testing through social engineering," *Information systems security*, vol. 4, no. 4, pp. 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*, December 2010, Technical Report, pp. 399–408.
- [7] R. Gula, "Broadening the scope of penetration testing techniques," *Enterasys Networks*, 1999.
- [8] H. Hasle, Y. Kristiansen, K. Kintel, and E. Snekkenes, "Measuring resistance to social engineering," in *Information Security Practice and Experience*. Springer, 2005, pp. 132–143.
- [9] I. Kotenko, M. Stepashkin, and E. Doynikova, "Security analysis of information systems taking into account social engineering attacks," in *Proc. of PDP*. IEEE, 2011, pp. 611–618.
- [10] M. Bishop, "About penetration testing," *Security & Privacy, IEEE*, vol. 5, no. 6, pp. 84–87, 2007.
- [11] S. Furnell and M. Papadaki, "Testing our defences or defending our tests: the obstacles to performing security assessment references," *Computer Fraud & Security*, vol. 2008, no. 5, pp. 8–12, 2008.
- [12] A. Hudic, L. Zechner, S. Islam, C. Krieg, E. Weippl, S. Winkler, and R. Hable, "Towards a unified penetration testing taxonomy," in *Proceedings of PASSAT*, 2012, pp. 811–812.
- [13] V. Nunes Leal Franqueira, R. Lopes, and P. van Eck, "Multi-step attack modelling and simulation (MsAMS) framework based on mobile ambients," in *Proceedings of SAC*. New York: ACM, 2009, pp. 66–73.
- [14] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *Proceedings of ICISC*, ser. LNCS, D. Won and S. Kim, Eds., vol. 3935. Springer, 2006, pp. 186–198.
- [15] B. Schneier, "Attack trees: Modeling security threats," *Dr. Dobbs's journal*, vol. 24, no. 12, pp. 21–29, 1999.
- [16] G. Rasch, *Probabilistic Models for Some Intelligence and Attainment Tests*. MESA Press, 1960.
- [17] 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.*, vol. 57, pp. 1813–1824, September 2011.
- [18] A. Elo, *The rating of Chessplayers, Past and present*. New York: Arco Publishers, 1978.
- [19] 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 NSPW*. New York, NY, USA: ACM, 2012, pp. 1–14.
- [20] W. J. Van Der Linden, "Conceptual issues in response-time modeling," *Journal of Educational Measurement*, vol. 46, no. 3, pp. 247–272, 2009.
- [21] E. Jonsson and T. Olovsson, "A quantitative model of the security intrusion process based on attacker behavior," *Software Engineering, IEEE Transactions on*, vol. 23, no. 4, pp. 235–245, 1997.