Effort estimates for vulnerability discovery projects

Teodor Sommestad
The Royal Institute of Technology (KTH), Sweden
teodors@ics.kth.se

Hannes Holm
The Royal Institute of Technology (KTH), Sweden
hannesh@ics.kth.se

Mathias Ekstedt
The Royal Institute of Technology (KTH), Sweden
mathiase@ics.kth.se

Abstract

Security vulnerabilities continue to be an issue in the software field and new severe vulnerabilities are discovered in software products each month. This paper analyzes estimates from domain experts on the amount of effort required for a penetration tester to find a zero-day vulnerability in a software product. Estimates are developed using Cooke's classical method for 16 types of vulnerability discovery projects – each corresponding to a configuration of four security measures. The estimates indicate that, regardless of project type, two weeks of testing are enough to discover a software vulnerability of high severity with fifty percent chance. In some project types an eight-to-five-week is enough to find a zero-day vulnerability with 95 percent probability. While all studied measures increase the effort required for the penetration tester none of them have a striking impact on the effort required to find a vulnerability.

1. Introduction

A substantial share of the security problems encountered in enterprises today arises because software products have security vulnerabilities. New vulnerabilities are discovered on a continuous basis. During 2010 alone, a total of 2096 new software vulnerabilities of high severity were publicly announced [1]. Many factors influence the number of vulnerabilities that are found in a software product. The effort invested into searching for vulnerabilities in a software product is one important variable [2,3]. Another important variable is the difficulty associated with finding vulnerabilities in the software product, i.e. how much effort that is required to find a vulnerability in it.

Secure software development practice (see [4] for an overview) suggests a wide range of measures to increase the security of a software product's source code and thus increase the effort required to find a vulnerability, e.g. testing during the development phase. A natural question to ask is how much effort that is required to find a vulnerability in a software product given that different security enhancing measures have been used. Unfortunately, there are no studies available which answer this question, or even provide rough estimates of it. Ideally, this would be tested in experiments or derived from representative archival data on projects that attempted to discover vulnerabilities. However, constructing experiments of this kind are associated with substantial cost, and reliable archival data on efforts made not available to the community [5].

Expert judgment is often used when quantitative data is difficult to obtain from experiments or studies of archival data. This paper presents expert estimates on vulnerability discovery effort that are constructed using Cooke's classical method. This method assigns weights to experts based on how correct and certain they are on a set of questions related to the issue investigated, and for which the true answer is known at the time of analysis. It has been used to assess uncertain quantities in a wide range of domains and in general outperforms other methods that synthesize or aggregate domain experts' judgment [6].

The effort estimates presented in this paper quantify the effort associated with finding a zero-day vulnerability in a software product. That is, finding a vulnerability in deployed software product which is not already publicly announced or patched [7]. The experts in this study are researchers in the software vulnerability field. They used their domain knowledge to assess the work effort it takes for a professional penetration tester taking on 16 hypothetical vulnerability discovery projects, all with the goal to find a zero-day vulnerability of high severity. The resulting estimates show the probability that a vulnerability is found as a function of the work days spent on the project.

The paper is structured as follows. Section 2 presents the variables used in the effort estimation model. In section 3 Cooke's classical method is

explained. Section 4 presents the method and section 5 presents the results. In section 6 these results are discussed and in section 7 conclusions are drawn.

2. Model and assumptions

This paper estimates the effort that is required to discover a zero-day vulnerability in a software product given that different security measures are used. Both the software security field and effort estimation field are well explored. However, no previous work has been found on the work-effort required to find zero-day vulnerabilities. This section presents the variables assessed in this study and the assumptions it is based upon.

2.1. Variables impacting discovery effort

A countless number of variables can be assumed to influence the effort required to find vulnerabilities in it. Technical measures, process measures and organizational measures are all of relevance [4].

Naturally, the scope of this research does not include all variables that could have an impact on the effort required to find a zero-day vulnerability. To identify a manageable set of variables to include a panel consisting of three security experts were consulted. All experts in this panel had practical experience of penetration testing and worked with security testing on a regular basis. They prioritized a list of candidate variables drawn from literature such as [4,8-11]. They were also given the option to suggest variables not included in the list presented to them. Table 1 shows the variables that came out of this process and are included in this study. All these variables were expected to have an impact on the effort required to find a new vulnerability in a software product.

Table 1. Variables studied.

Variable	Description
Scrutinized	The targeted software has been scrutinized before.
SourceCode	The professional penetration tester has access to the source code.
SafeLanguage	The software is written in a safe language (e.g. C#, Java) or a safe dialect (e.g. Cyclone).
CodeAnalyzers	The software has been analyzed by static code analyzers and improved based on the result.

All variables described in Table 1 have support in literature. Software which has been *scrutinized* and tested in practice will be more difficult to find vulnerabilities in. This type of effect is often assumed in software reliability models [5] and data on vulnerabilities found in software products imply that a saturation level for vulnerabilities discovered in a product is reached after a certain time on the market [12]. Access to the *source code*, i.e. the uncompiled code, is also considered a relevant factor [13]. Access to the uncompiled source code will enable white box testing and is likely to decrease the effort required to find a vulnerability. If the programming language used to create the software product is a *safe language* [14] many potential programming flaws leading to vulnerabilities can be avoided. Finally, the use of *code analyzers* is often a recommended practice in software development to identify vulnerabilities in the code [15-17] .

2.2. Assumptions

A number of assumptions are used for the effort estimates produced in this study and are kept constant in this study. First, the competence of the actual performer of the vulnerability discovery project can be expected to have a substantial impact on the effort required [5]. To eliminate variations caused by this variable it is assumed that the person who carries out the vulnerability discovery project is a professional penetration tester. Secondly, it would be extremely difficult to find vulnerabilities in a product which is completely inaccessible to the person carrying out the project. Therefore, it is assumed that the person searching for vulnerabilities has access to the compiled code (the binary) even if the source code (*SourceCode*) is unavailable to him/her. Third, a work day was set to eight hours of work. This was specified to avoid confusion about what quantity that should be estimated (calendar, budget or effort) [18]. Fourth, the vulnerability that should be discovered needs to qualify as a high severity-vulnerability according to the Common Vulnerability Scoring System [19]. Since such vulnerabilities are more severe than other vulnerabilities it is more interesting to obtain knowledge about them. The final assumption used, and presented to those who estimated effort, was that all unspecified variables (e.g. the size of the source code) assume the state they typically have in an enterprise environment. Thus, any uncertainty remaining after the variables and assumptions are specified should be accounted for in the estimates. That is, variation between software not covered by the assumptions or variables will introduce uncertainty and variation to the effort required. The respondents were asked to consider this uncertainty onto the estimates the made.

3. Synthesizing expert judgments

There is much research on how to combine, or synthesize, the judgment of multiple experts to increase the calibration of the estimate used. Research has shown that a group of individuals assess an uncertain quantity better than the average expert, but the best individuals in the group are often better calibrated than the group as a whole [20]. The combination scheme used in this research is the classical model of Cooke [21]. Experience shows that Cooke's classical method outperforms both the best expert and the "equal weight" combination of estimates. In an evaluation involving 45 studies it performs significantly better than both in 27 studies and performs equally as well as the best expert in 15 of them [6].

In Cooke's classical method *calibration* and *information* scores are calculated for the experts based on their answers on a set of seed questions, i.e. questions for which the true answer is known at the time of analysis. The calibration score shows how well the respondent's answers represent the true value; the information score show how precise the respondent's answers are. These two scores are used to define a *decision maker* which assigns weights to the experts based on their performance. The weights defined by this decision maker are used to weight the respondents answer's to the questions of interest – in this case the effort estimates for vulnerability discovery projects. In sections 3.1, 3.2 and in 3.3 Cooke's classical method is explained. For a more detailed explanation the reader is referred to [21].

3.1. Calibration score

In the elicitation phase the experts provide individual answers to the seed questions. The seed questions request the respondents to specify a probability distribution for an uncertain continuous variable. This distribution is typically specified by stating its 5th, 50th, and 95th percentile values. These percentiles yield four intervals over the percentiles *[0-5, 5-50, 50-95, 95-100]* with probabilities of p= *[0.05, 0.45, 0.45, 0.05]*. As the seeds are realizations of these uncertain variables the well calibrated expert will have approximately 5% of the realizations in the first interval, 45 % of the realizations in the second interval, 45 % of the realizations in the third interval and 5% of the realizations in the fourth interval. If s is the distribution of the seeds over the intervals the relative information of s with respect to p is: $I(s,p) = \sum_{i=1}^{4} \ln(s_i / p_i)$.

This value indicates how surprised someone would be if one believed that the distribution was p and then learnt that it was s.

If N is the number of samples (seeds) the statistic of $2NI(s, p)$ is asymptotically Chi-square distribution with three degrees of freedom. This is asymptotic behavior is used to calculate the calibration Cal of expert e as: $Cal(e) = 1 - \chi_3^2(2N\, I(s,p))$. Calibration measures the statistical likelihood of a hypothesis. The hypothesis tested is that realizations of the seeds (s) are sampled independently from a distribution agreeing with the expert's assessments (p).

3.2. Information score

The second score used to weight experts is the information score, i.e. how informative the expert's distributions are. This score is calculated as the deviation of the expert's distribution to some meaningful background measure. In this study the background measure is a uniform distribution over [0,1].

If b_i is the background density for seed $i \in \{1,\ldots,N\}$ and $d_{e,i}$ is the density of expert e on seed i the information score for expert e is calculated as: $inf(e) = \frac{1}{N}\sum_{i=1}^{N} I(d_{e,i}, b_i)$, i.e. as the relative information of the experts distribution with respect to the background measure. It should be noted that the information score does not reflect calibration and does not depend on the realization of the seed questions. So, regardless of what the correct answer is to a seed question a respondent will receive a low information score for an answer which is similar to the background measure, i.e. the answer is distributed evenly over the variable's range. Conversely, an answer which is more certain and has focused the probability density over few possible outcomes will yield high information scores.

3.3. Constructing a decision maker

The classical method rewards experts who produce answers with high calibration (high statistical likelihood) and high information value (low entropy). A strictly proper scoring rule is used to calculate the weights the decision maker should use. If the calibration score of the expert e is equal or greater than a threshold value the expert's weight is obtained as $w(e)=Cal(e)*Inf(e)$. If the expert's calibration is less than α the expert's weight is set to zero, a situation which is common to happen a substantial number of experts in practical applications.

The threshold value α corresponds to the significance level for rejection of the hypothesis that the expert is well calibrated. The value of α is identified by resolving the value that would optimize a virtual decision maker. This virtual decision maker combines the experts' answers (probability distributions) based on the weights they obtain at the chosen threshold value (α). The optimal level for α is where this virtual expert would receive the highest possible weight if it was added to the expert pool and had its calibration and information scored as the actual experts.

When α has been resolved the normalized value of the experts weights *w(e)* are used to combine their estimates of the uncertain quantities of interest.

4. Data collection method

This section presents how the data was collected in terms of: how seed questions for Cooke's classical method were constructed, the population and sample of experts that was chosen and how the elicitation instrument was developed and tested.

4.1. Seed questions

As the experts performance on answering the seed questions are used to weight them, it is critical that the seeds are well validated and also that they lie in the same domain as the studied variables. They need to be drawn from the respondents' domain of expertise, but need not necessarily be directly related to questions of the study [21].

Naturally, the robustness of the weights attributed to individual experts depends on the number of seeds used. This study used 11 seed questions. Experience shows this is more than enough to see substantial difference in calibration [21] between experts.

For this study two types of seed questions were used (cf. Table 2). All of these were constructed using information from the national vulnerability database and concerned characteristics of existing vulnerabilities in software products. Questions 1-5 concerned different types of vulnerabilities and under what conditions they could be exploited; questions 6-11 concerned how often publicly known vulnerabilities in different products was due to input validation or buffer errors and to authentication or authorization errors (cf. Table 3). Both these two types of questions are related to the topic as they gauge how well the expert can assess properties related to vulnerabilities that can be expected to be found.

Table 2. Seed questions used in abbreviated format and their realized value.

#	Question	Real
1	What portion of vulnerabilities published during 2010 of high severity has a complete impact on CIA	57 %
2	What portion of vulnerabilities published during 2010 of medium severity has a complete impact on CIA.	6 %
3	What portion of vulnerabilities published during 2010 that are remotely exploitable (does not require LAN access) will require that the attacker can authenticate itself before succeeding with an exploit?	9 %
4	What portion of vulnerabilities published in 2010 that are remotely exploitable (does not require LAN access) and requires that the attacker can authenticate itself before the exploit is of high severity?	15 %
5	What portion of vulnerabilities published in 2010 that are remotely exploitable (does not require LAN access) is of high severity?	52 %
6	What portion of vulnerabilities publicly announced in 2010 with high severity is due to input validation or buffer errors?	53 %
7	What portion of vulnerabilities publicly announced with high severity for Windows 7 is due to input validation or buffer errors?	36 %
8	What portion of vulnerabilities publicly announced with high severity for Apple's products is due to input validation or buffer errors?	31 %
9	What portion of vulnerabilities publicly announced with high severity for the .NET framework is due to authentication or authorization errors?	10 %
1	What portion of vulnerabilities publicly announced with high severity for the Microsoft's Internet Information Services is due to authentication or authorization errors?	13 %
1	What portion of vulnerabilities publicly announced with high severity for Cisco's products is due to authentication or authorization errors?	11 %

Table 3. Error types from NVD used.

Input validation/buffer errors	Authentication or authorization errors
CWE 20: Improper Input Validation CWE 89: SQL Injection CWE 119: Failure to Constrain Operations within the Bounds of a Memory Buffer CWE 134: Uncontrolled Format String CWE 189: Numeric Errors	CWE 255: Credentials Management CWE 264: Permissions, Privileges, and Access Controls CWE 287: Improper Authentication CWE 310: Cryptographic Issues

4.2 The domain experts

As this research aims to identify quantities related to discovery effort the respondents needed both the ability to evaluate aspects in the domain and the ability to reason in terms of probabilities. In terms of the expert categories described in [22] individuals that are expert judges are desirable.

Good candidates for this are researchers in the software security field. These can be expected to both understand how to reason with probabilities and to possess the required skills to evaluate the effectiveness difficulty of finding vulnerabilities in software. Software security researchers were therefore chosen as the population to survey. To identify suitable respondents, articles published in the SCOPUS database [23], INSPEC or Compendex [24] between January 2005 and September 2010 were reviewed. Authors was considered if they had written articles in the information technology field with any of the following phrases in the title, abstract or keywords: "software vulnerability", "software vulnerabilities", "software exploit", "software exploits", "exploit development", "develop exploits",, "develop an exploit" ,"exploit writing", "writing exploits", "vulnerability research", or "exploit code". If their contact information could be found they were added to the sample of respondents. After reviewing and screening respondents and their contact information a sample of 384 individuals was assessed. The contact information for approximately 80 turned out to be incorrect or outdated.

As recommended by [25] , motivators were presented to the respondents invited to the survey: i) helping the research community as whole, ii) the possibility to win a gift certificate on literature, and iii) being able to compare their answers to other experts after the survey was completed. Out of approximately 300 researchers invited to the survey 92 opened the survey and 17 submitted answers to the survey's questions. A response rate of this magnitude is logically to be expected of a more advanced survey of this type.

4.3. Elicitation instrument

A web survey was used to collect the probability distributions from the invited respondents. The survey was structured into four parts, each beginning with a short introduction to the section. First, the respondents were given an introduction to the survey as such that explained the purpose of the survey and its outline. In this introduction they also confirmed that they were the person who had been invited and provided information about themselves, e.g. years of experience in the field of research. Second, the respondents received training regarding the answering format used in the survey. After confirming that this format was understood the respondents proceeded to its third part. In the third part both the seed questions and the questions of the study were presented to the respondents. Finally, the respondents were asked to provide qualitative feedback on the survey and the variables covered by it.

Questions in section 3 were each described through a scenario entailing a number of conditions. *Scenarios* and conditions for the seed questions can be found in Table 2; project types and conditions for the questions of interest in this study is described in section 2.1.

In the seed questions the respondent was asked to provide a probability distribution that expressed the respondent's belief. As is custom in applications of Cooke's classical method this probability distribution was specified by setting the 5^{th} percentile, the 50^{th} percentile (the median), and the 95^{th} percentile for the probability distribution. In the survey the respondents specified their distribution by adjusting sliders or entering values to draw a dynamically updated graph over their probability distribution. The three points specified by the respondents defines four intervals over the range [0, 100]. The graphs displayed the probability density as a histogram, instantly updated upon change of the input values.

In the question of interest, the respondent specified probability distributions for work days required to find a zero-day vulnerability. The respondents were asked to specify the number of work days that would be needed to find a zero-day vulnerability with a probability of 5 percent, 50 percent and 95 percent. This is a common format to use for effort estimates [26] and in prediction in general [27]. As before the estimates dynamically updated a graph representing the answer. However, for these questions this graph showed the cumulative probability of finding a zero-day vulnerability as a function of work days spent. This graph was plotted using linear interpolation between the three values specified by the respondent.

Use of graphical formats is known to improve the accuracy of elicitation [28]. Figures and colors were also used to complement the textual formulations and make the content easier to understand. In Figure 1 the format presented to respondents is exemplified.

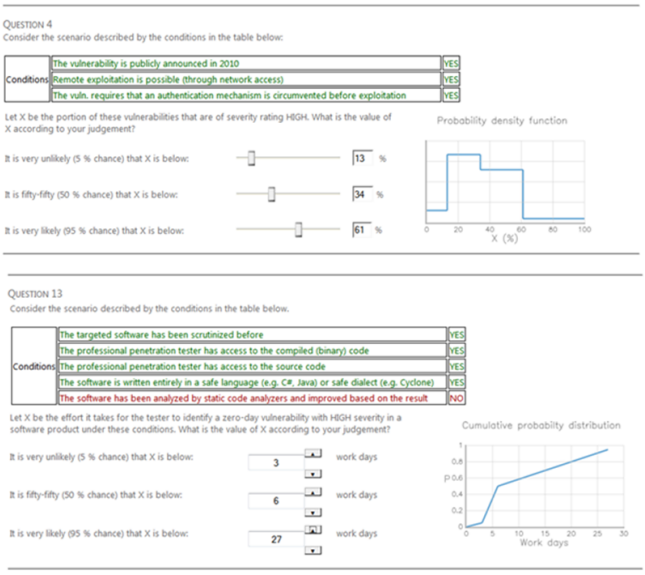

Figure 1. Examples of question and answering format in the survey (seed 4 and project type 2).

Elicitation of probability distributions is associated with a number of issues [28]. Effort was therefore spent on ensuring that the measurement instrument held sufficient quality. After careful construction the survey was qualitatively reviewed during personal sessions with an external respondent representative of the population. This session contained two parts. First the respondent was given the task to fill in the survey, given the same amount of information as someone doing it remotely. After this discussions followed regarding the instrument quality. These sessions resulted in several improvements.

Before this qualitative review the question format as such had been tested in a pilot study on other security parameters. In that pilot study a randomized sample of 500 respondents was invited; 34 of these completed the pilot during the week it was open. The questions in this pilot survey were presented in the same way as in the present survey. A reliability test using Cronbach's alpha [29,30] was carried out using four different ways to phrase questions for one variable. Results from this test showed $\alpha = 0.817$, which indicates good internal consistency of the instrument.

5. Results

This section presents the result of the analysis performed on the judgment of the 17 researchers. In section 5.1 the overall performance of the respondents on the seed questions is presented. In section 5.2 the synthesized estimates of those respondents who were assigned weight are presented. In section 5.3 the influence that each of the four individual variable have on the effectiveness is described.

5.1. Respondents' performance

As in many other studies involving expert judgment some of the respondents were poorly calibrated. Their calibration score varied between 0.540×10^{-3} and 0.615 with a mean of 0.305. The respondents' information score varied between 0.0770 and 1.009 with a mean of 0.324. Figure 2 shows the information score and calibration score of the 17 respondents.

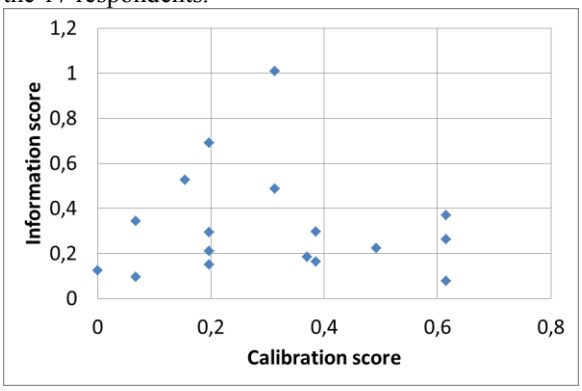

Figure 2. Information and calibration scores of the respondents.

Cooke's classical method aims is to identify those respondents whose judgment is well calibrated and informative. The virtual decision maker was optimized at a significance level (α) of 0.615. Consequently, the three rightmost respondents in Figure 2 received a weight higher than zero and the other 14 respondents received a weight of zero. As noted in 0 above it is not uncommon that a substantial number of respondents receive the weight zero with this method.

The twelve respondents who received a positive weight all had the same calibration score (0.615). Their weights are therefore directly proportional with their information score (cf. section 3.2). They received weights 0.1086, 0.3711 and 0.5203 after normalization.

5.2. Work effort in the project types

To identify the probability distribution which the virtual decision maker assigns to the 16 types of vulnerability discovery projects examined the respondents' individual estimates were combined

based on the respondent's weights. The estimated distributions were assumed to be distributed in the same way as they were presented to the respondents (c.f. section 4.3), i.e. as depicted in the linearly interpolated cumulative probability distributions for the finding of a zero-day vulnerability when work effort is increased.

Table 1. Different types of vulnerability discovery projects and the estimated effort to find a vulnerability with a certain degree of certainty. Values have been rounded off to closest number of full days.

Project	Scrutinized	SourceCode	SafeLanguage	CodeAnalyzers	Low (5%)	Median(50%)	High95%)
1	Yes	Yes	Yes	Yes	3	13	74
2	Yes	Yes	Yes	No	1	3	26
3	Yes	Yes	No	Yes	0	13	26
4	Yes	Yes	No	No	0	1	7
5	Yes	No	Yes	Yes	1	12	855
6	Yes	No	Yes	No	0	10	27
7	Yes	No	No	Yes	2	9	855
8	Yes	No	No	No	1	4	257
9	No	Yes	Yes	Yes	1	6	27
10	No	Yes	Yes	No	0	4	9
11	No	Yes	No	Yes	0	3	17
12	No	Yes	No	No	1	3	8
13	No	No	Yes	Yes	1	14	344
14	No	No	Yes	No	1	7	27
15	No	No	No	Yes	1	6	18
16	No	No	No	No	0	3	9

The respondents specified the cumulative probability distribution through its 5th, 50th and 95th percentile. As depicted in Table 4 and the synthesized estimates show clear differences among the project types. The median for the projects varies between 1 and 14 work days; the value at the 5th percentile varies between 0 and 3 work days; the value at the 95th percentile varies between 7 and 855 work days. As could be expected is project type 5 the one with highest expected effort, closely followed by project type 7. For these two project types a time budget of more than 2 years and 4 months is needed to find a vulnerability with 95 percent certainty. In other project types this certainty can be obtained with a time-budget of just a week or a month. Project type 4, 10, 12 and 16 are associated with lowest work effort.

5.3 Variables influence on the effectiveness

Four variables are varied to specify the 16 project types. The variation over scenarios supports this hypothesis that they influence effort. A relevant question is then how important these variables are for the effort required by the attacker. Table 5 shows the mean influence that the four variables have on the probability distribution. These values are the mean difference obtained when comparing scenarios where the variable is in the state true with those scenarios where the variable is in the state false, and all other variables remain in the same state. For instance, the values for *Scrutinized* are obtained as the mean value of the difference between scenarios 1 and 9, 2 and 10, 3 and 11 and so on.

All variables have a positive impact on the effort required to find a zero day vulnerability given a number of work days. As can be seen from Table 5 the most influential variables on the 95th percentile are *Scrutinized* (if the software has been searched for vulnerabilities before), *SourceCode* (if the attacker can get access to the source code) and *CodeAnalyzers* (if the software product has been improved with static code analyzers). The impact of these variables on the high extreme value, where a zero-day vulnerability is found with 95 percent probability, is substantial. Such sizeable difference cannot be found for the variable *SafeLanguage.* As a consequence this variable has a meager influence on the expected work effort in comparison to the other variables.

Table 5. Mean influence in work days of the variables under the assumptions used in the study.

Variable	Low (5%)	Median (50%)	High (95%)
Scrutinized	+0.4	+1.1	+208.5
SourceCode	+0.1	+3.6	+274.8
SafeLanguage	+0.4	+4.6	+24.0
CodeAnalyzers	+0.6	+3.9	+230.8

6. Discussion

Software insecurity is a serious problem in today's society. Decision makers can certainly make use of data on the effectiveness of measures that

make vulnerability discovery projects more cumbersome. Most decision makers probably would prefer reliable empirical data to base their decisions on. However, such data is not available today. It is difficult to obtain such data from archival studies as no such archives are available and as indicated from the result of this study it would also be costly to collect this data from repeated experiments.

The use of expert judgment can be motivated in absence of reliable data. This study extracts and synthesizes data from domain experts. The method used to analyze the experts' judgments and combine these is described in section 6.1 below. The elicitation instrument used is discussed in section 6.2. The result as such and the importance variables included in the study are discussed in section 6.3.

6.1. Expert judgment analysis

In this study Cooke's classical method [21] was used to synthesize expert judgments. This performance based method aims to select the experts that are well calibrated and combine their judgments in an optimal way. The track record of this method [6] positions it as the best-practice when it comes to combining experts' judgment of uncertain quantities.

Eleven seed questions were used to evaluate calibration and information scores. These seed questions are drawn from a vulnerability database. A concern to the validity is that this source also is available to the respondents who could have used them to identify the answers to the seed questions. If they would do so these seeds would not work well as a gauge for how well calibrated and informative the expert's own judgment is. However, it appears unlikely that anyone did so. None of the respondents answering the survey has given comments that indicate that they have realized that the correct answer can be found in online databases. Neither did the qualitative reviewer realize this during the qualitative reviews. Furthermore, inspections of the answers received do not indicate any answers were based on these sources.

The use of these seed questions shows that calibration varies among experts. This can be seen through the calibration scores to the seed questions used in this study (c.f. Figure 2). The three best calibrated experts were assigned weight when the virtual decision maker was optimized. The synthesized probability distributions created based on their judgment involve a great deal of uncertainty. In some cases the 95 percent confidence interval spans over 886 work days. As can be seen from Figure 2 , the estimates provided by the three respondents who obtained weight are not the most informative ones. This should not be seen as surprising. Overconfidence is a well-known cause for poor calibration in expert judgments [31]. Cooke's methods only assign weights to experts with a calibration score that exceeds a threshold value. However, these experts' weight is calculated with the information score as one of two factors. This avoids domination of uninformative experts in the synthesis of judgments.

When using this method it is appropriate to perform robustness test with respect to the seed variables and the experts by removing one expert and investigating the impact of this removal [21]. Such tests were performed and indicate that the solution is robust to changes in both seed questions and experts.

6.2. Validity and reliability of the elicitation instrument

Cooke [21] suggests that seven guidelines should be followed when data is elicited from experts. How these have been addressed in the present study is described below.

Cooke states that questions must be clear and unambiguous and that a dry run should be carried out before the actual study. In this study the clarity of questions were tested in qualitative reviews with a strategically selected respondent representative of the population. The comments received from this person helped improve the understandability of the instrument and remove ambiguity. Also, a quantitative test was performed on a survey with a similar structure and a similar way of phrasing questions. This quantitative test was made through a pilot survey answered by 34 respondents. It indicated good reliability of the survey instrument.

It is also suggested that an attractive graphical format and a brief explanation of the elicitation format should be prepared [21]. The answering format used in this study was supported by graphical illustrations – the answers were entered by entering a probability function on the screen. This format was also carefully explained in an introductory training section in the survey. Also, background information introduced each new section.

Cooke further recommends that the elicitation should not exceed one hour and that coaching should be avoided. None of the respondents who completed the survey spent more than one hour to do so and efforts were made to ensure that the questions were formulated in a neutral way.

The last recommendation given in [21] is that an analyst should be present when respondents answer the questions. The respondents were given contact information to the research group when invited to the

survey and they were encouraged to use these any if questions arose. It is possible that analysts' physical absence from the elicitation suppressed some potential questions from being asked. In the survey the respondents were asked to comment the clarity of the questions and the question format used. Based on the comment received it appears as if the questions and the assumptions were understandable. Two respondents did however comment that the questions perhaps should be directed towards practitioners ("hackers") rather than researchers. While practitioners probably need more guidance in specifying answers through probability distributions this recommendation gives input to future research efforts in this track

6.3. Variables importance to zero-day discovery projects

Two weeks of work is enough to have a fifty-fifty chance of finding a zero-day vulnerability in all projects types assessed. In some cases two weeks of work is enough to give more than 95 percent chance of discovering a zero-day vulnerability and the fifty percent chance is reached after just a couple of days. While these estimates give dismaying results they are not in conflict with already known data. The rate with which vulnerabilities are publically announced hints that effort required to find them is modest. We also tested this prediction model using the PERT formula [32] on a number of software products which have been scrutinized. The estimates appear reasonable when compared to the publicly disclosed vulnerabilities in SecurityFocus [33] during 2010. For example, the estimates says that during all days of 2010 there would be the equivalent of approximately 7 professional 8 hour work-day on finding and disclosing vulnerabilities in Firefox, and that 17 professional penetration testers working each work-day on Internet Explorer 8.

No radical impact can be made using the measures included in this study, but they all help to increase the security of software products. Their impact on the median value is similar for all measures except making sure that products have been scrutinized (this has less impact on the median). The use of safe languages does not impact the extreme value (95^{th} percentile) as much as the other ones. As a consequence, it does not influence the expected effort as much as the other three countermeasures do, and could be seen as less effective.

In the survey the respondents were asked to indicate if there were important variables missing. Only three out of 17 respondents suggested other priorities than used in the survey. All three suggested different things: fuzzers combined with static code analysis as one variable, if static code analysis was performed on a regular basis (not just performed), and variables indicating the security expertise of the developer and or development process (not specified which). All these suggestion were considered in the discussion with the panel of experts who prioritized variables to include in the survey, but were intentionally excluded from the survey. This, together with the survey-respondents' opinions indicates that the most important variables for estimating vulnerability discovery are included in this study.

While the most important variables seems to be included in our model the estimates indicate that the effort required to discover a new vulnerability can be as high as man-years even if the compiled code is available to the attacker. This study does not reveal which these conditions are, i.e. when the penetration tester will have to spend years searching for a vulnerability. The expert panel and the respondents of the survey indicated that the most important variables are included in the model used here. It is therefore likely that a number of favorable conditions must apply in these cases. In order to obtain better and more detailed knowledge in this area further work could explore what set of measures that causes this effect and how to achieve such secure software products.

7. Conclusion

It appears difficult to achieve a high level of security assurance in today's software intensive environment. The probability that a professional penetration tester will find a previously unknown vulnerability in software product used today is disturbingly high. Under most conditions a few days appears enough to find a zero-day vulnerability with a fifty percent chance. Countermeasures do increase work effort required, but none of them seem to have a striking impact on the effort required to find a vulnerability, at least not in the general case. The estimates made by experts included in this study are associated with a great deal of uncertainty. Under some conditions the professional penetration tester will need man years of effort required to find a zero-day vulnerability, i.e. the 95^{th} percentile spans man-years. This study does not reveal which these conditions are, but since no crucial variables seem to be omitted from this study it is likely that a number of favorable conditions must apply in these cases.

8. References

[1] U.S.D. of C. NIST Computer Security Resource Center, "National Vulnerability Database," 2011.
[2] O.H. Alhazmi and Y.K. Malaiya, "Quantitative vulnerability assessment of systems software," *Proceedings of Annual Reliability and Maintainability Symposium*, Ieee, 2005, pp. 615-620.
[3] S.-W. Woo, H. Joh, O.H. Alhazmi, and Y.K. Malaiya, "Modeling vulnerability discovery process in Apache and IIS HTTP servers," *Computers & Security*, vol. 30, Jan. 2011, pp. 50-62.
[4] B. De Win, R. Scandariato, K. Buyens, J. Grégoire, and W. Joosen, "On the secure software development process: CLASP, SDL and Touchpoints compared," *Information and Software Technology*, vol. 51, Jul. 2009, pp. 1152-1171.
[5] A. Ozment, "Improving vulnerability discovery models," *Proceedings of the 2007 ACM workshop on Quality of protection*, ACM, 2007, p. 6–11.
[6] R. Cooke, "TU Delft expert judgment data base," *Reliability Engineering & System Safety*, vol. 93, May. 2008, pp. 657-674.
[7] M.A. McQueen, T.A. McQueen, W.F. Boyer, and M.R. Chaffin, "Empirical estimates and observations of 0day vulnerabilities," *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, IEEE, 2009, p. 1–12.
[8] C. Cowan, "Software security for open-source systems," *Security & Privacy, IEEE*, vol. 1, 2003, p. 38–45.
[9] M. Howard and D.C. LeBlanc, *Writing Secure Code*, Redmond, WA, USA: Microsoft Press, 2002.
[10] Y. Younan, "Efficient countermeasures for software vulnerabilities due to memory management errors," Katholieke Universiteit Leuven, 2008.
[11] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller, "Predicting vulnerable software components," *Proceedings of the 14th ACM conference on Computer and communications security*, New York, New York, USA: ACM, 2007, p. 529–540.
[12] S. Sridhar and K. Altinkemer, "Software Vulnerabilities: Open Source versus Proprietary Software Security," *AMCIS 2005 Proceedings*, 2005.
[13] C. Payne, "On the security of open source software," *Information Systems Journal*, vol. 12, Jan. 2002, pp. 61-78.
[14] T. Jim, G. Morrisett, D. Grossman, M. Hicks, J. Cheney, and Y. Wang, "Cyclone: A safe dialect of C," *USENIX*, Monterrey, CA, USA: 2002, pp. 275-288.
[15] M.D. Penta, L. Cerulo, and L. Aversano, "The life and death of statically detected vulnerabilities: An empirical study," *Information and Software Technology*, vol. 51, Oct. 2009, pp. 1469-1484.
[16] S. Heckman and L. Williams, "A systematic literature review of actionable alert identification techniques for automated static code analysis," *Information and Software Technology*, 2010.
[17] Y. Kim, J. Lee, H. Han, and K.-M. Choe, "Filtering false alarms of buffer overflow analysis using SMT solvers," *Information and Software Technology*, vol. 52, Feb. 2010, pp. 210-219.
[18] S. Grimstad, M. Jorgensen, and K. Molokken-Ostvold, "Software effort estimation terminology: The tower of Babel," *Information and Software Technology*, vol. 48, Apr. 2006, pp. 302-310.
[19] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," *Published by FIRST-Forum of Incident Response and Security Teams*, 2007, pp. 1-23.
[20] R.T. Clemen and R.L. Winkler, "Combining probability distributions from experts in risk analysis," *Risk Analysis*, vol. 19, 1999, pp. 187-204.
[21] R. Cooke, "Experts in uncertainty: opinion and subjective probability in science," 1991.
[22] D.J. Weiss and J. Shanteau, "Empirical Assessment of Expertise," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45, 2003, pp. 104-116.
[23] Elsevier B.V., "Scopus," 2011.
[24] Elsevier Inc, "Engineering Village," 2011.
[25] S.T. Cavusgil and L.A. Elvey-Kirk, "Mail survey response behavior: A conceptualization of motivating factors and an empirical study," *European Journal of Marketing*, vol. 32, 1998, p. 1165–1192.
[26] H. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling*, New York, NY, USA: John Wiley & Sons, 2001.
[27] J. Armstrong, *Principles of forecasting – A Handbook for Researchers and Practitioners*, Netherlands: Kluwer Academic Publishers Group, 2001.
[28] P.H. Garthwaite, J.B. Kadane, and A. O'Hagan, "Statistical methods for eliciting probability distributions," *Journal of the American Statistical Association*, vol. 100, 2005, pp. 680-701.
[29] L.J. Cronbach and R.J. Shavelson, "My Current Thoughts on Coefficient Alpha and Successor Procedures," *Educational and Psychological Measurement*, vol. 64, Jun. 2004, pp. 391-418.
[30] L.J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, 1951, p. 297–334.
[31] S. Lin, "A study of expert overconfidence," *Reliability Engineering & System Safety*, vol. 93, May. 2008, pp. 711-721.
[32] H. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling*, New York, NY, USA: John Wiley & Sons, 2001.
[33] SecurityFocus, "SecurityFocus," 2011.