Effort estimates on web application vulnerability discovery

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

Teodor Sommestad
Swedish Defence Research Agency (FOI), Sweden
teodor.sommestad@foi.se

Abstract

Web application vulnerabilities are widely considered a serious concern. However, there are as of yet scarce data comparing the effectiveness of different security countermeasures or detailing the magnitude of the security issues associated with web applications. This paper studies the effort that is required by a professional penetration tester to find an input validation vulnerability in an enterprise web application that has been developed in the presence or absence of four security measures: (i) developer web application security training, (ii) type-safe API's, (iii) black box testing tools, or (iv) static code analyzers. The judgments of 21 experts are collected and combined using Cooke's classical method. The results show that 53 hours is enough to find a vulnerability with a certainty of 95% even though all measures have been employed during development. If no measure is employed 7 hours is enough to find a vulnerability with 95% certainty.

1. Introduction

Web Applications (WA) are widely used and often hold sensitive or important data. Attacks against them can thus be very costly due to losses of data integrity, confidentiality or availability. The arguably most problematic type of WA security issue is code injection due to poor input sanitizing. This category of vulnerabilities includes, for example, SQL injection (SQLi), Command Injection and cross site scripting (XSS). Of these, SQLi is the most critical type of WA vulnerability according to OWASP [1] and SANS 2011 Top 25 (that rank all known IT security vulnerabilities) [2]. Command Injection and XSS are not far behind, being ranked second and fourth by SANS [2]. Naturally, other WA flaws such as Cross Site Request Forgery (CSRF) are also bothersome.

In recent years, considerable effort has been performed to understand and manage this problem. For instance, organizations such as MITRE, SANS and OWASP have developed security awareness programs to help organizations to mitigate the issue. However, despite these efforts a recent study [3] shows that application developers still are unable to implement effective countermeasures for WA vulnerabilities.

One possible reason behind the frequent occurrence of WA vulnerabilities could be the lack of any useful overview regarding the effectiveness of different security measures. There are various tools available for the same purpose and it is difficult for practitioners to understand what security measures that are worth investing in for a particular WA. Data on the general effectiveness of different approaches would thus be valuable even if it come with a certain degree of uncertainty.

There are a few studies that have attempted to analyze the effectiveness of different countermeasures against WA attacks in laboratory environments [4–6]. Unfortunately, the difficulties of performing experiments on the topic in a representative manner have brought various constraints that delimit the value of their results for a decision maker in the industry. These constraints include, for example, that the effectiveness of countermeasures in combination is not investigated or that the type attacker which the data is valid for is not detailed. As a consequence, their results cannot be generalized to the domain at large or compared to studies by others.

Expert judgment is often used when quantitative data is difficult to obtain from experiments or studies of archival data. This study provides estimates by 21 WA security experts on the effort that is required to find an input validation vulnerability in WAs developed in 16 different ways. Since different experts can have different accuracy, Cooke's classical method [7], the best-practice method for weighting expert judgment, was employed to select the set of experts that perform best at estimates of this type.

The rest of the paper unfolds as follows. Section 2 describes the studied variables and the utilized assumptions. Section 3 describes Cooke's classical method. Section 4 describes the data collection method used during the study. Section 5 presents the observed results and Section 6 discusses these results. Finally, Section 7 concludes the paper.

2. Model and assumptions

Numerous variables can be assumed to influence the effort required to find vulnerabilities in WAs. Technical measures, process measures and organizational measures are all of relevance [8]. This study uses variables identified through a previous research [9] involving a literature review and expert judgment. This previous study is summarized in this section - the reader is referred to [9] for more comprehensive details.

2.1. Studied variables

The literature review identified four categories of variables of importance towards the effectiveness of countermeasures for WA vulnerabilities: (i) the type of attack (e.g., XSS or SQLi), (ii) whether the vulnerability is known to the public at large or not, (iii) the severity of the vulnerability (e.g., if successful exploitation can provide administrator-level privileges), and (iv) what type of security mechanism that is in question. According to the literature there are dependencies between all of these variables.

The hypothesized variables from the literature review were revised according to the archival data and judgment of six domain experts. The purpose of these revisions were threefold: (i) to revise the theoretical categorization and obtain variables on an abstraction level that is useful when the effectiveness of countermeasures used in practice is estimated, (ii) to determine what variable-dependencies that are important to study, and (iii) to obtain rough estimates on the general effectiveness of different defenses against WA attacks (both alone and in combination).

The expert judgment study resulted in several significant revisions of the hypothesized categorization, perhaps the most significant being the addition of a new category – the competence of the attacker in question. In addition, the type of vulnerability (e.g., XSS or SQLi), and the severity of a vulnerability, were not seen as significant towards the effectiveness of countermeasures. For example, the experts' believed that black box testing tools were equally effective for XSS and SQLi vulnerabilities, regardless of the vulnerability's severity. As a consequence, variables related to these two categories were discarded based on the reviewers' recommendations.

An important distinction was made between countermeasures that focus on finding, patching and removing vulnerabilities in the WA and countermeasures that are applied to secure an already deployed WA and do not require major changes of application source code (e.g., a WA firewall). This paper focuses on estimating the effectiveness of the first type of countermeasures, i.e., countermeasures that typically are employed during development. Four such countermeasures were identified during the pre-study: (i) black box testing, (ii) type-safe API's, (iii) static code analysis and (iv) the developers' WA security training.

Type-safe API's [10] involves using a development environment that is built to function in a secure and reliable fashion. In essence, a type-safe API defines a rule set for allowed code and how different parts of an application are allowed to exchange information. For instance, how a PHP application is allowed to communicate with an SQL database. If a developer writes code that does not comply with the rule set defined within the type-safe API an error is produced, notifying the developer of the proper syntax as defined by the API.

Developer security training [11] involves increasing the WA security awareness of the software developers. The aim is to make developers recognize what improper input and output sanitizing can result in and how such issues can be mitigated.

Black box testing [12] involves running automated scanners or fuzzers on deployed WAs without viewing server-side source code. The aim of a black box tests is to find vulnerabilities so that these can be removed before deployment.

Static code analysis [10], [12] involves white box testing for detecting vulnerabilities. They analyze the WA's source code and try to find vulnerabilities that would be exploitable in runtime by applying various checks.

Binary ranges were used for analyzing the effectiveness these four countermeasures. That is, a countermeasure is either employed or not employed during development of the WA. This gave a total of 16 different types of vulnerability discovery projects for which to produce effort estimates.

2.2. Assumptions

A number of assumptions are used for the effort estimates produced in this study. First, the competence of the attacker is expected to have a substantial impact on the effort required [13]. 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. Second, it is assumed that probing for the vulnerability is performed from an external network (e.g., the internet). Third, an input validation vulnerability refers to either SQLi, XSS, Command Injection or CSRF. This delimitation was made to focus only on vulnerability discovery in the actual WA and thus exclude, for example, buffer overflow vulnerabilities in the web server binary (a topic we have previously studied [14]). The final assumption is that all unspecified variables (e.g., the size of the WA's source code) have the value they typically have in an enterprise environment. The respondents were asked to consider the variation between enterprises and how this influenced the uncertainty of their estimates. Thus, any remaining uncertainty should be accounted for in the estimates.

3. Synthesizing expert judgment

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 [15]. The combination scheme used in this research is the classical model of Cooke [7]. Cooke's classical method attempts to identify the best set of experts and experience show that it outperforms both the best expert and the "equal weight" combination of experts' 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 [16].

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 often the respondent's estimated intervals cover 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' answers' 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. The reader is referred to [7] for a more detailed explanation.

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: $Cal(e) = 1 - \chi_3^2(2N\, I(s,p))$. The 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 an expert is the information score, i.e., how informative the expert's estimated intervals 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 expert's distribution with respect to the background measure b_i. It should be noted that the information score does not reflect calibration and does not depend on the realization of the seed questions. Thus, a respondent will receive a high information score if its estimates are substantially different from the background distributions even if the estimate is completely wrong.

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 the threshold value α 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 to 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, how the population and sample of experts that was chosen and how the elicitation instrument was developed and tested.

4.1. Seed questions

Since the experts' answers to the seed questions are used to weight them it is critical that the seeds are well validate. They need to be drawn from the respondents' domain of expertise, but need not necessarily be directly related to questions of the study [7].

Naturally, the robustness of the weights attributed to individual experts depends on the number of seeds used. This study used 8 seed questions. Experience shows this is enough to see substantial difference in calibration between experts [7].

In this study two types of seed questions were used (cf. Table 1): distributions of vulnerabilities according to typecasting (question 1-4) [3] and complexity (question 5-8) [17]. *Typecasting* involves what type of input variable that is vulnerable: either a Boolean, Free text (an arbitrary string), or Structured text (e.g., an URL or email). A more open ended typecast is often believed to be more difficult to sanitize than a more restricted one. *Complexity* involves whether the attack string is encoded or not. For example, a variable vulnerable to SQLi through a simple non-encoded single quote (i.e., `'`) is likely easier to spot than one that only is vulnerable to a specific encoded single quote (e.g., `&#x27`). These two types of questions are related to the respondents' domain of expertise as they gauge how well the expert can assess properties related to vulnerabilities that can be expected to be found. The realizations for question 5-8 were obtained from the authors of [17] directly.

Table 1. Seed questions and their realized values.

#	Seed	Value	Ref
1	Number of Boolean variables out of 100 variables vulnerable to SQLi.	4%	[3]
2	Number of Structured variables out of 100 Structured- or Free text variables vulnerable to SQLi.	77%	[3]
3	Number of Boolean variables out of 100 variables vulnerable to XSS.	4%	[3]
4	Number of Structured variables out of 100 Structured- or Free text variables vulnerable to XSS.	67%	[3]
5	Number of complex SQLi vulnerabilities out of 100 published during 2005.	7%	[17]
6	Number of complex SQLi vulnerabilities out of 100 published during 2009.	14%	[17]
7	Number of complex XSS vulnerabilities out of 100 published during 2005.	27%	[17]
8	Number of complex XSS vulnerabilities out of 100 published during 2009.	23%	[17]

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 [18] individuals that are expert judges are desirable. Good candidates for this are researchers and practitioners in the WA security field, such as professional WA penetration testers (the studied type of attacker). These can be expected to possess the required skills to evaluate the difficulty of finding vulnerabilities given different scenarios, and were thus chosen as the population of the study. There are multiple ways to reach out to respondents that are a part of this population. For example, they can be identified through authorship of publications, forums or email lists. This study involves invitations through large and relevant email lists (six public and one private) since this is a very simple way of reaching out to a very large sample of respondents.

The public lists include pen-test@securityfocus.com, security-basics@securityfocus.com, gpwn-list@lists.sans.org, owasp-dotnet@lists.owasp.org, owasp-testing@lists.owasp.org and owasp-sweden@lists.owasp.org. The private list is an international invitation-only list involving a sample of highly experienced security professionals (e.g., software penetration testers).

The potential issue of novices participating is handled by Cooke's classical method as this method scores respondents based on their performance on a set of test questions (the seeds). As recommended by [19], motivators were presented to the survey participants: (i) helping the WA security community as whole, (ii) the possibility to win a gift certificate on Amazon, and (iii) being able to compare their answers to other experts' answers after the survey was completed.

The survey was online between the 29th of February 2012 and the 22nd of March 2012. A total of 263 respondents opened it; of these, 52 fully answered the seeds, and 21 of these 52 respondents completed the entire survey. A completion rate of this magnitude can be expected of a more advanced survey such as the one utilized.

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 that explained its purpose and outline. In this introduction they also provided information about themselves, e.g., years of experience in the field of research.

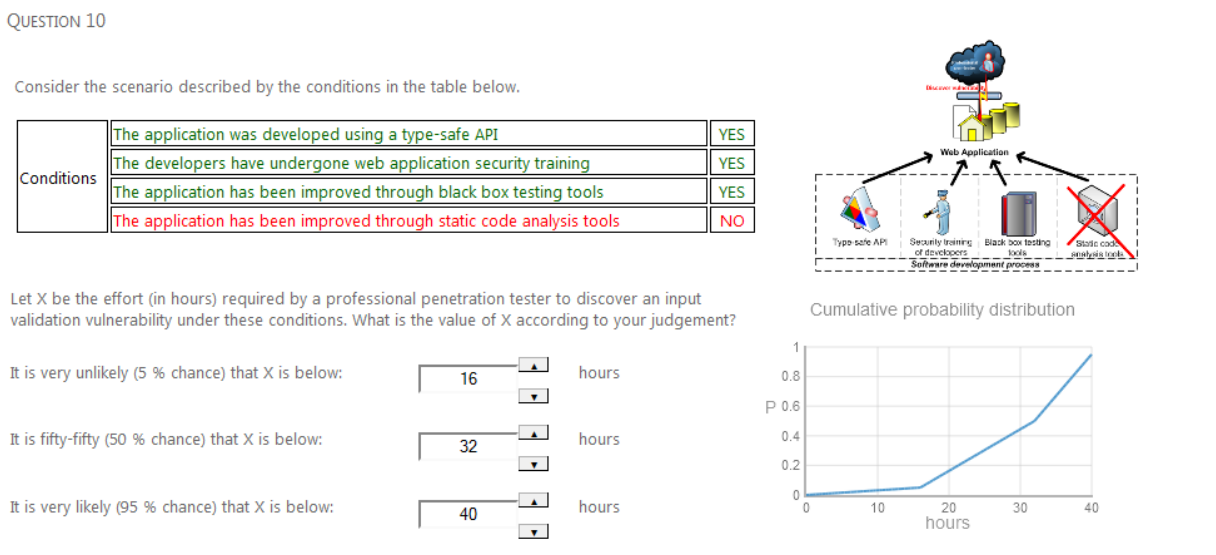
Second, the respondents received training regarding the answering format used in the survey. After confirming that this format was understood the respondents proceeded to its third part. In the third part both the seed questions and the questions regarding the studied variables were presented to the respondents. Each question was described through a scenario comprising its conditions. Scenarios and conditions for the seed questions can be found in Table 1; project types and conditions for the questions of interest in this study are described in Section 2. For these questions the respondents were asked to provide probability distributions that expressed their beliefs. As is customary in applications of Cooke's classical method each probability distribution was specified by setting the 5th percentile, the 50th percentile (the median), and the 95th percentile. 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] percent. The graphs displayed the probability density as a histogram, instantly updated upon change of the input values.

For the questions of interest, the respondents were asked to specify the number of hours required to discover an input validation vulnerability with a likelihood of 5 percent, 50 percent, and 95 percent (resulting in a probability density function). This is a common format to use for effort estimates [20] and in prediction in general [21]. As for the seeds the estimates on these questions dynamically updated graphs representing their answers. However, for these questions this graph showed the cumulative probability of finding a vulnerability as a function of hours spent.

Use of graphical formats is known to improve the accuracy of elicitation [22]. In this survey, figures and colors were used to complement the textual formulations and make the content easier to understand. In Figure 1 the format presented to respondents is exemplified.

In the fourth and final part of the survey the respondents were asked to detail any tested variable (cf. Section 2.1) that they would like to replace with another (more important) variable. This section also asked the respondents to describe how they pictured these variables. For example, if they depicted a specific static code analysis tool or a specific type-safe API (and if so, to detail this countermeasure). At the end of this section the respondents were finally asked to detail any perceived issues with the survey.

Elicitation of probability distributions is associated with a number of issues [22]. 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



QUESTION 10

Consider the scenario described by the conditions in the table below.

Conditions	The application was developed using a type-safe API	YES
	The developers have undergone web application security training	YES
	The application has been improved through black box testing tools	YES
	The application has been improved through static code analysis tools	NO

Let X be the effort (in hours) required by a professional penetration tester to discover an input validation vulnerability under these conditions. What is the value of X according to your judgement?

It is very unlikely (5 % chance) that X is below:	16	hours
It is fifty-fifty (50 % chance) that X is below:	32	hours
It is very likely (95 % chance) that X is below:	40	hours

Discover vulnerability
Web Application
Type-safe API
Security training of developers
Black box testing tools
Static code analysis tools
Software development process
Cumulative probability distribution
P
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
hours

Figure 1. Question and answering format used in the survey (for project 2).

representative of the population. This session contained two parts. First the respondent was given the task to fill in the survey without any help from the researchers. After this discussions followed regarding the instrument quality. These sessions resulted in several improvements. For example, a graphical figure describing one of the vulnerability discovery projects was revised.

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. Of these 34 completed the pilot during the week it was open. A reliability test using Cronbach's alpha [23], [24] 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 chapter presents the analysis performed on the judgment of the 21 respondents. Section 5.1 describes the respondents, Section 5.2 their performance on the seed questions and Section 5.3 the synthesized effort estimates produced using Cooke's method. Finally, Section 5.4 analyzes the effectiveness of the studied countermeasures.

5.1. Overall characteristics of respondents

The 21 participating respondents are associated with 9 countries. A majority (8) of the respondents were participating from the United States (all from different states) but a number of other countries were also observed [e.g., India (4 respondents), Colombia (3) and Sweden (1)]. The mean experience related to WA vulnerabilities was 6.9 years and the mean perceived competence 57% (from 1-100%, where 1% meant that the respondent perceived itself to be more knowledgeable on the topic than 99% of the community). The respondents generally positioned themselves towards practice rather than research (a mean of 37 on a scale from 1: only work with industry/practice to 100: only work with research/academia).

5.2. Performance on the seed questions

As in many other studies involving expert judgment some of the respondents were poorly calibrated. Their calibration score varied between 2.2×10^{-10} and 0.177 with a mean of 0.015. The respondents' information score varied between 0.577 and 3.99 with a mean of 1.82. Figure 2 shows the information score and calibration score of the 21 respondents.

Cooke's classical method aims to identify those respondents whose judgment is well calibrated and informative. The virtual decision maker was optimized at a calibration score of 0.0265. Consequently, the four rightmost respondents in Figure 2 received a weight higher than zero and the other 17 respondents received a weight of zero. As noted in Section 3.3 it is not uncommon that a substantial number of respondents receive the weight zero with this method. The four respondents that were sufficiently calibrated received the weights 0.4645, 0.2314, 0.2016 and 0.1025 after normalization.

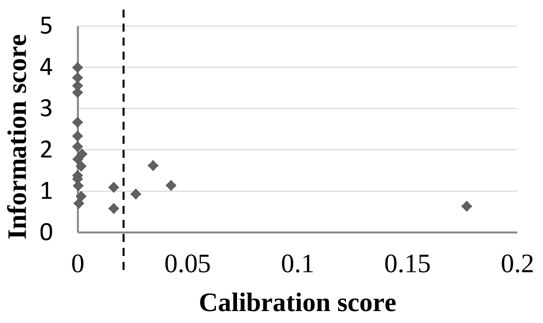


Figure 2. Information and calibration scores of the respondents.

The four respondents with highest calibration scores came from different countries, had a mean of 5.3 years experience, a mean perceived competence of 56% and positioned themselves towards practice rather than academia (a mean of 24).

5.3. Work effort in the project types

The respondents specified the effort (in hours) required to find a vulnerability with 5% certainty, 50% certainty and 95% certainty. As depicted in Table 2 the synthesized estimates show clear differences among the project types.

The median for the projects varies between 3 and 33 hours; the value at the 5th percentile varies between 1 and 17 hours; the value at the 95th percentile varies between 7 and 53 hours. For example, in vulnerability discovery project number four the WA was developed using a type-safe API, by developers that had undergone security training, but without the aid of black box testing tools or static code analyzers. In this project the expected number of hours to find a vulnerability is between 4 (5% certainty) and 15 hours (95% certainty), with a median (50% certainty) of 8 hours.

5.4. Effectiveness of countermeasures

As all combinations between the four tested countermeasures are studied it is possible to analyze their significance, both for when that they are employed alone and when they are employed in combination with others'. Significance is calculated as the mean difference in effort between when a countermeasure (or a combination of measures) is

Table 2. Different types of vulnerability discovery projects and the estimated hours to find a vulnerability with a certain degree of certainty.

Project	Type-safe API	Security training	Black box Testing	Static code analysis	Low (5%)	Medium (50%)	High (95%)
1	Yes	Yes	Yes	Yes	10	33	53
2	Yes	Yes	Yes	No	17	29	46
3	Yes	Yes	No	Yes	15	25	42
4	Yes	Yes	No	No	4	8	15
5	Yes	No	Yes	Yes	5	11	18
6	Yes	No	Yes	No	5	11	17
7	Yes	No	No	Yes	6	12	17
8	Yes	No	No	No	5	9	12
9	No	Yes	Yes	Yes	17	31	51
10	No	Yes	Yes	No	12	20	32
11	No	Yes	No	Yes	10	18	33
12	No	Yes	No	No	1	3	8
13	No	No	Yes	Yes	3	8	18
14	No	No	Yes	No	1	7	12
15	No	No	No	Yes	2	8	12
16	No	No	No	No	2	5	7

Table 3. Effectiveness of countermeasures, both alone and in combination.

Countermeasure	Increased effort (hours)		
	Low (5%)	Medium (50%)	High (95%)
Type-safe API (A)	2.7	4.7	7.4
Security training (B)	7.0	12.0	22.3
Black box testing (C)	3.3	7.9	11.2
Static code analysis (D)	2.7	6.7	13.5
AB	-0.8	0.8	0.7
AC	-1.2	-0.2	0.6
AD	-1.1	-0.5	-3
BC	3.4	6.8	9.8
BD	2.0	5.0	6.2
CD	-2.7	-2.7	-2.0
ABC	-1.2	0.1	-0.6
ABD	-1.1	-0.5	0.7
ACD	-2.3	-1.4	-3.9
BCD	-3.0	-1.3	-4.3
ABCD	-1.4	-1.1	0

employed compared to when the same combination of countermeasures is not employed. This is a customary calculation when studying a complete experimental design (i.e., when all possible state-space combinations have been tested) [25]. An overview of the results from this analysis can be seen in Table 3. For example, if a type-safe API is employed during development then the median additional effort required to find a vulnerability by a professional penetration tester is 4.7 hours.

The combined effects should be interpreted as follows: The effectiveness of countermeasures in combination is a sum of their effectiveness alone and their effectiveness in combination with others'. For example, black box testing (C) and static code analysis (D) provide an increased median effort of 7.9 hours and 6.7 hours if employed alone, and -2.7 hours if employed in combination; giving a total of 7.9 + 6.7 - 2.7 = 17.3 hours to find a vulnerability with 50% certainty. Thus, these tools are not perceived to "shine" together – but a combination of them still yields a more secure result than when only employing one of them.

6. Discussion

This chapter is divided into three sections. The first two sections discuss the most significant countermeasures alone and in combination with others'. The third section discusses the reliability and validity of the study.

6.1. Individual countermeasures

The most important countermeasure is believed to be developer security training. If the developers have undergone regular WA security training it is expected that the median time required to probe for an input validation vulnerability by a professional penetration tester will increase by 12 hours.

The second most important countermeasure is automated black box testing. Employment of this type of tool is expected to increase the effort with a median of 7.9 hours. A close runner up to the effectiveness of this countermeasure is static code analyzers: this type of tool is expected to increase the effort with a median of 6.7 hours.

The least effective tool is rather surprisingly believed to be type-safe API's. This type of countermeasure is expected to increase the effort with a median of 4.7 hours. A reason for this could be that they typically only help sanitize a subset of all input validation problems. For example, an API might cover SQLi but not XSS, only cover subset of all possible encodings for single quotes, or only cover a subset of all possible variable types.

6.2. Countermeasures in combination

Developer security training is believed to increase the effectiveness when employing black box testing tools or source code analyzers. The median effort is increased with 6.8 hours for black box testing (data for BC) and 5 hours for static code analyzers (data for BD). This is an expected result – developers that have received training should be more efficient with these tools.

Three strong negative joint effects are also present, all involving the combination of employing black box testing and static code analysis in combination. The joint effect of these two countermeasures (data for CD) is believed to decrease the effort for finding a vulnerability with a median of 2.7 hours. If the developers also have been security trained (data for BCD) the effect is slightly smaller but still negative (a median decreased effort of 1.3 hours). Replacing developer training for a type-safe API (data for ACD) is believed to have a similar effect on the median effort (1.4 hours). One reason behind these negative effects could be information overload: the results from black box testing and static code analysis are of similar type (e.g., listing unsanitized input variables) and their output (existing vulnerabilities) will sometimes overlap. Furthermore, more tools could mean more data to manually process and thus more difficulty when prioritizing different mitigation suggestions.

A final important remark is that none of the joint effects are larger than any of the lone effects. Consequently, adding one of the four countermeasures will increase the effort required to find an input validation vulnerability regardless of other countermeasures in place, but the size of this increase will differ.

6.3. Validity and reliability

There are two major topics in terms of validity and reliability that should be addressed: (i) whether the respondents estimates are representative for WAs at large, and (ii) whether the utilized data collection tool and methodology provides reliable results.

Regarding (i), it is difficult to estimate if the relatively small sample of respondents size can be said to be representative of the WAs in enterprises at large. On the other hand, the respondents were all experienced, originated from different geographical locations and had different backgrounds. This suggests that the results reflect a wide variety of conditions and that is not biased towards any particular cultural attributes or properties by a certain mindset (e.g., penetration testers that specialize in evaluating .NET applications). Furthermore, it is important to recognize that it is the first study made on the topic. Thus, even tentative results are valuable.

Regarding (ii), the choice of variables and their assumptions was made based on a pre-study involving both literature review and domain experts [9]. Concerning data collection methodology, Cooke [7] 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 [7]. 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 Cooke 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.

The respondents were also asked if they wanted to revise or replace any of the tested variables. None of the respondents suggested any revisions in this regard. Furthermore, the respondents did not share a single mindset regarding how they envisioned the studied variables. For example, various static code analyzers such as HP WebInspect and Fortify SCA were pictured by them.

7. Conclusions and future work

For researchers, the observations gained during this study denote that some combined effects are more important to include when deciding upon a certain research design. For example, if studying the usefulness of black box testing tools or static code analyzers there is a need to detail the expertise of the individuals that employ these and attempt to mitigate their discovered vulnerabilities. The observations also show clear evidence of that a fundamentally novel approach is required to fully secure a WA – nothing that is currently available will be enough to prevent a skilled attacker from finding security flaws.

For practitioners, the results show that all four analyzed countermeasures are important to employ when developing a WA. Furthermore, the results provide quantitative estimations that can be used to compare countermeasures; both alone, and in combination. It could be that a median of 33 hours required to find a vulnerability when all countermeasures have been employed is seen as inadequate security. However, the cost of having a professional spend a whole week to find a single vulnerability with 50% chance is surely out of scope for many vulnerability discovery projects. Furthermore, the perceived threat for most organizations is not professional penetration testers, but rather less competent attackers (e.g., script kiddies). A professional who spends 33 hours on a vulnerability discovery project is certainly more effective than a beginner who spends 33 hours. Thus, the results from this study need be seen in the light of a more competent attacker profile. On the other hand, for security sensitive sites such as online banks a security professional with one week at hand to probe for vulnerabilities might be a realistic threat.

Nonetheless, it would be interesting to redo the study in the scope of a less experienced attacker.

It would also be interesting to study the effectiveness of run-time countermeasures such as WA firewalls. A number of such countermeasures were identified during the pre-study [9], and we plan to study them utilizing the same approach as is presented in this paper.

Finally, it would be interesting to view the observations from this study in the light of data sources other than expert judgment. One such option could be to compare archival vulnerability data to the development processes of different products. Another option could be cyber defense competitions (e.g. as in [26]); to study the relative security of different countermeasures in controlled environments. In the future we aim to employ a mix of these methods to reexamine the estimates provided by this paper.

8. References

[1] OWASP, “2010 OWASP Top 10,” 2010.

[2] B. Martin, M. Brown, A. Paller, D. Kirby, and S. Christey, “2011 CWE/SANS Top 25 Most Dangerous Software Errors,” 2011.

[3] T. Scholte, D. Balzarotti, W. Robertson, and E. Kirda, “An Empirical Analysis of Input Validation Mechanisms in Web Applications and Languages,” in *The 27th Symposium On Applied Computing*, 2012, pp. 202-209.

[4] N. Antunes and M. Vieira, “Benchmarking Vulnerability Detection Tools for Web Services,” in *2010 IEEE International Conference on Web Services*, 2010, pp. 203-210.

[5] J. Fonseca, M. Vieira, and H. Madeira, “Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks,” in *13th Pacific Rim International Symposium on Dependable Computing*, 2007, pp. 365-372.

[6] I. A. Elia, J. Fonseca, and M. Vieira, “Comparing SQL Injection Detection Tools Using Attack Injection: An Experimental Study,” in *21st International Symposium onSoftware Reliability Engineering (ISSRE)*, 2010, pp. 289–298.

[7] R. M. Cooke, *Experts in uncertainty: opinion and subjective probability in science*. Oxford University Press, USA, 1991.

[8] 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, no. 7, pp. 1152-1171, Jul. 2009.

[9] H. Holm, M. Ekstedt, “A metamodel for web application injection attacks and countermeasures”, in *TEAR 2012 and PRET 2012*, LNBIP 131, pp. 198–217, 2012.

[10] M. D. Mitropoulos, V. Karakoidas, P. Louridas, and D. Spinellis, “Countering Code Injection Attacks: A Unified Approach,” *Information Management & Computer Security*, vol. 19, no. 3, pp. 3–3, 2011.

[11] R. L. Jones and A. Rastogi, “Secure coding: building security into the software development life cycle,” *Information Systems Security*, vol. 13, no. 5, pp. 29-39, 2004.

[12] Y. Shin and L. Williams, “Toward A Taxonomy of Techniques to Detect Cross-site Scripting and SQL Injection Vulnerabilities,” 2008.

[13] A. Ozment, “Improving vulnerability discovery models,” in *Proceedings of the 2007 ACM workshop on Quality of protection*, 2007, pp. 6–11.

[14] T. Sommestad, H. Holm, M. Ekstedt, ” Effort Estimates for Vulnerability Discovery Projects”, in *45th Hawaii International Conference on System Sciences*, pp. 5564—5573, 2012.

[15] R. T. Clemen and R. L. Winkler, “Combining probability distributions from experts in risk analysis,” *Risk Analysis*, vol. 19, pp. 187-204, 1999.

[16] R. Cooke, “TU Delft expert judgment data base,” *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 657-674, May 2008.

[17] T. Scholte, D. Balzarotti, and E. Kirda, “Have things changed now? An empirical study on input validation vulnerabilities in web applications,” *Computers and Security*, 2012.

[18] D. J. Weiss and J. Shanteau, “Empirical assessment of expertise,” *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45, no. 1, p. 104, 2003.

[19] 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, no. 11/12, pp. 1165-1192, 1998.

[20] H. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling*. Wiley, 2009.

[21] J. Armstrong, *Principles of forecasting: a handbook for researchers and practitioners*. Springer, 2001.

[22] P. H. Garthwaite, J. B. Kadane, and A. O’Hagan, “Statistical methods for eliciting probability distributions,” *Journal of the American Statistical Association*, vol. 100, no. 470, pp. 680-701, 2005.

[23] L. J. Cronbach, “Coefficient alpha and the internal structure of tests,” *Psychometrika*, vol. 16, no. 3, pp. 297-334, 1951.

[24] L. J. Cronbach and R. J. Shavelson, “My Current Thoughts on Coefficient Alpha and Successor Procedures,” *Educational and Psychological Measurement*, vol. 64, no. 3, pp. 391-418, Jun. 2004.

[25] D. C. Montgomery, *Design and analysis of experiments*. John Wiley & Sons Inc, 2008.

[26] H. Holm, T. Sommestad, U. Franke, and M. Ekstedt, “Success Rate of Remote Code Execution Attacks-Expert Assessments and Observations,” *Journal of Universal Computer Science*, vol. 18, no. 6, p. 732--749, 2012.