Success rate of remote code execution attacks – expert assessments and observations

Hannes Holm
(Royal Institute of Technology, 100 44 Stockholm, Sweden
hannesh@ics.kth.se)

Teodor Sommestad
(Royal Institute of Technology, 100 44 Stockholm, Sweden
teodors@ics.kth.se)

Ulrik Franke
(Royal Institute of Technology, 100 44 Stockholm, Sweden
ulrikf@ics.kth.se)

Mathias Ekstedt
(Royal Institute of Technology, 100 44 Stockholm, Sweden
mathiase@ics.kth.se)

Abstract: This paper describes a study on how cyber security experts assess the importance of three variables related to the probability of successful remote code execution attacks: (i) non-executable memory, (ii) access and (iii) exploits for High or Medium vulnerabilities as defined by the Common Vulnerability Scoring System. The rest of the relevant variables were fixed by the environment of a cyber defense exercise where the respondents participated. The questionnaire was fully completed by fifteen experts. These experts perceived access as the most important variable and availability of exploits for High vulnerabilities as more important than Medium vulnerabilities. Non-executable memory was not seen as significant. Estimates by the experts are compared to observations of actual attacks carried out during the cyber defense exercise. These comparisons show that experts' in general provide fairly inaccurate advice on an abstraction level such as in the present study. However, results also show a prediction model constructed through expert judgment likely is of better quality if the experts' estimates are weighted according to their expertise.

Keywords: Cyber security, Remote code execution, Software vulnerabilities
Categories: K.6.5, K.6.3, D.2.9

1 Introduction

Exploits which can provide administrator privileges of systems are attractive to attackers, particularly if they can be executed remotely. Various types of vulnerabilities can be used to achieve this. One example is to take advantage of vulnerabilities in the memory management of a system in order to execute the attacker's own programming instructions.

Many conditions influence whether an attacker can manage to successfully execute remote code on a target. A few examples include the type of vulnerabilities

present in the targeted machine, the competence of the attacker and the quality of any protective measures that are in place. Clearly, it is valuable to know under which circumstances such attacks are likely to succeed and under which circumstances they are not. Data regarding this would be useful when prioritizing protective measures, e.g. mitigation options, or as input to security analysis frameworks, e.g. [Homer, 09] and [Sommestad, 10]. This is especially true for enterprise decision makers - decision support which does not require extensive data collection would be valuable even if it is only approximate.

However, little such observational data currently exist in the domain. One explanation for this is that most enterprises do not want to share their cyber security incidents, often due to the potential adverse economic effects [Campbell, 03][Cavusoglu, 04]. It is also a field where measurement can be difficult. For instance, it is virtually impossible to assert that confidential business data have not been read by unauthorized individuals.

Use of expert judgment, i.e. to elicit knowledge from domain experts based on their experience, is a commonly applied technique to obtain data when observations are difficult to perform. Many security researchers have turned to this methodology as a viable option (e.g. [Haimes, 03][Madan, 02][Taylor, 02]). Expert judgment has for instance been used to assess the importance of attributes that are related to critical infrastructure risks [Cooke, 04], and to quantify parameters in security risk models [Ryan, 10].

This paper presents judgments made by 15 cyber security experts participating in an international cyber defense exercise. The experts estimated the probability of successful remote exploitation of software vulnerabilities given different scenarios, all from a viewpoint of use towards practical enterprise decision making.

In addition to surveying expert assessments actual remote code executions carried out during the cyber defense exercise were observed. These results are compared to the estimations by the experts to measure the quality of their predictions. A limitation with this comparison is that only one out of eight scenarios estimated by the experts could be observed.

The rest of the paper unfolds as follows: Sect. 2 describes related work. Sect. 3 explains the variables studied in this paper. Sect. 4 describes the methodology of the expert assessment study. Sect. 5 presents the results and analysis of the expert assessments. Sect. 6 describes how the observations of the actual attacks were collected and compares these to the estimates by the experts. Sect. 7 present a critical discussion of the results of the study and Sect. 8 concludes the paper.

2 Related work

This paper focuses on vulnerabilities that enable attackers to execute arbitrary code on a targeted machine from a remote location. A common class of vulnerabilities that make this possible is buffer overflows [Cowan, 03][One, 96].

Since buffer overflow vulnerabilities are severe security problems, a number of techniques and tools have been developed to eliminate them or make exploitation of them more difficult. Common countermeasures include authorization, authentication control, vulnerability management (removing vulnerabilities due to software flaws), and measures focusing explicitly on hardening computers or source code against

arbitrary code attacks [Mell, 07]. Younan [Younan, 08] divides measures that hardens computers or source code into ten different types. Seven of these focus on the software product itself and how to decrease its vulnerability, viz.: (i) safe languages (e.g. Cyclone [Jim, 02]), (ii) bound checkers (e.g. Cash [Chiueh, 05]), (iii) hardened libraries (e.g. FormatGuard [Cowan, 01]), (iv) separation and replication of information countermeasures (e.g. Libverify [Baratloo, 00]), (v) runtime taint trackers (e.g. TaintCheck [Newsome, 05]), (vi) dynamic analysis and testing (e.g. Purify [Joyce, 92]), and (vii) static analysis such as [Dor, 01]. These measures will help removing vulnerabilities in the code or decrease their severity. However, they are not designed to influence the difficulty of exploiting the high-severity vulnerabilities that may remain in the code. As this study focuses on the latter, the following three types of measures are of particular relevance:

(i) Probabilistic countermeasures. This category includes address space layout randomization (ASLR) [PaX, 03] which changes the machines memory's layout between executions or user sessions to make memory referencing difficult for an attacker during buffer overflows; instruction set randomization [Kc, 03] which obscures the execution language to make it difficult for the attacker to invoke meaningful machine instructions in a buffer overflow; StackGuard which places a "canary" in memory which must be correctly guessed by the attacker during a buffer overflow [Cowan, 98].

(ii) Paging-based countermeasures. This category includes non-executable memory [Younan, 08] which flag parts of memory containing data to make the machine ignore machine instructions placed there during a buffer overflow; guard-page-based [Perens, 11] which places data next to pages which will terminate the program if they are changed.

(iii) Execution monitors. This category includes policy enforcers, such as SASI [Erlingsson, 99], that regulates an executable's access to resources on the machine, for instance, to make it difficult for attackers to invoke system calls which are not needed in normal operation; fault-isolation measures, such as MiSFIT [Small, 97], that limit the consequences of a successful attack, for instance, by separating executable's address spaces; anomaly detectors, such as work by Forrest et al. [Forrest, 96], that detect unusual behavior if this is created during an code execution attack.

More detailed descriptions of these three types of measures can be found in [Erlingsson, 99][Frykholm, 00][Xenitellis, 03][Younan, 08]. These references contain discussions on the effectiveness of different measures. Also, the descriptions of the techniques and tools come with a qualitative or theoretic evaluation of their effectiveness.

The number of empirical evaluations, however, is small. The study performed by [Wilander, 03] is an exception. This study focuses on measures that remove or degrade the software product's vulnerabilities and tests seven different techniques for thwarting buffer overflow attacks. It accurately describes the exploits these measures work against. However, it does not capture the competence of attackers or show the exploits they apply in practice. Therefore, it cannot be used to infer probabilities of success in general. This also applies to studies focusing on the effectiveness of specific run-time techniques, for example [Shacham, 04]. In [Shacham, 04] the effectiveness of address space layout randomization is tested under different

conditions and weaknesses that are exploitable under specific condition, but how often these conditions apply in practice is not discussed (or known to the community at large).

3 Studied variables

As can be seen in Sect. 2, there are many variables which influence the possibility of successfully perform arbitrary code execution attacks. However, most defense mechanisms are not widely used in practice. Because the aim of this research is to construct a model that is useful for enterprise decision makers, such as network administrators, the focus is placed on variables that are common in practice. This also means that any countermeasure that is deployed during the development phase of software is ruled out. As such, three types of countermeasures discussed in Sect. 2 are of importance: probabilistic countermeasures, paging-based countermeasures and execution monitors (cf. Sect. 2). Of these, probabilistic countermeasures are by some thought not to prevent, but rather delay code execution attacks [Shacham, 04] (as the exploit does not need to be rewritten, but rather tested enough times). Execution monitors are not as common as the other two in practice. The topic of this paper is thus to study the significance of perhaps the most commonly used paging-based countermeasure (and defense mechanism in general) against code execution attacks, namely, non-executable memory. Non-executable memory is an easily implemented countermeasure which is available for most operating systems (e.g. all Windows operating systems since XP). It is important to study the effectiveness of this countermeasure in the context of any significant conditions related to the success rate of remote code executions in practice. The two arguably most important conditions for this purpose is the severity of the vulnerability in question and if the attacker has access to the targeted service as a legitimate user or not [Mell, 07]. As such, these two conditions were included in the study. Naturally, there are numerous other variables that influence the probability of succeeding with remote code execution attacks – even on abstraction level such as the present study. Some common examples of other protective measures were provided in Sect. 2. In this study, only the security measures depicted in Fig. 1 were evaluated. However, the states of all other relevant measures were fixed by the environment of a cyber defense exercise. Sect. 4 describes this in more detail. How non-executable memory, vulnerability severity and service access were handled in the present study is detailed below.

Non-executable memory (NX). This is a paging-based countermeasure [Younan, 08] that marks the data segment of a computer memory as non-executable. This is done to make it more difficult for an attacker to execute code that has been injected into the data segment. NX is a feature that is available as an option for most operating systems and has implementations without costs in computational power or memory usage [Younan, 08]. It is for example used in products such as Windows XP Service Pack 2, Windows 7, and a number of Linux distributions.

Access. In all scenarios considered in this study it is assumed that the attacker can connect to the vulnerable service. The access variable involves whether the attacker can access the service and its resources as a user of it. Some vulnerabilities which exist in the inner-logic of a service could require access as an authorized user to the service to be exploited. As such, access to the targeted service should generally

decrease the effort required for successful injection . For example, if the service is the SMB service on a machine and access has state *true*, it would mean that the attacker is a part of the Windows domain. If it has the state *false* the attacker is not in the domain but can still send requests to the service as an external entity. Vulnerabilities that are only exploitable from within the Windows domain would be impossible to exploit is the state us *false*.

Severity of the exploited vulnerability. In all scenarios considered there is an *exploit* available to the attacker corresponding to a *vulnerability* on the targeted machine. In this investigation, exploits for High and Medium severity vulnerabilities as defined by the Common Vulnerability Scoring System (CVSS) [Mell, 07] are distinguished. The CVSS is a set of metrics used to quantitatively compare and describe different vulnerabilities using attributes such as ease of exploitation and the consequence of successful exploitation. The characteristics of a *High* vulnerability were informally translated to "There is full impact on confidentiality, integrity and availability". The characteristics of a *Medium* vulnerability were informally translated to "There is partial impact on confidentiality, integrity". As such, a High severity vulnerability should in general provide a greater opportunity for successful code injection attacks then a Medium severity vulnerability.

NX, access and the severity of the vulnerability exploited are all believed to affect the probability that **successful arbitrary code execution** is in the state *yes***,** i.e. how probable it is to succeed with executing code remotely on a machine with the chosen defense mechanisms.

An overview of the chosen variables and their possible states is shown in Fig. 1.

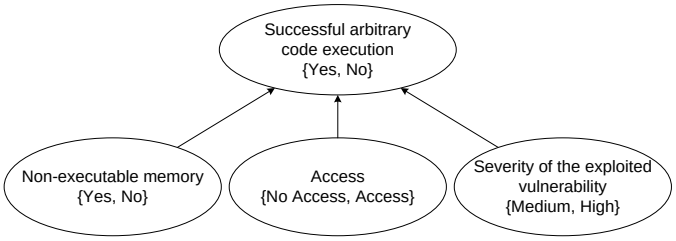

Figure 1. Studied variables and their possible states.

4 Expert assessment methodology

This section describes the respondents of the study, the cyber defense exercise, and the carried out questionnaire.

4.1 The respondents and the cyber defense exercise

The preferred population for assessing this type of data is IT security practitioners (such as professional pen-testers) or security researchers. Respondents of this type should be able to give assessments regarding how important the studied variables are

as they have practical experience from performing the cyber attack in question under different conditions.

This study describes a survey which was handed out before the start of a cyber defense exercise named Baltic Cyber Shield. The exercise was managed at the Cooperative Cyber Defense Centre of Excellence in Tallinn, Estonia. Its virtual battlefield was designed and hosted by the Swedish Defence Research Agency with the support of the Swedish National Defence College. The environment was set to mimic a typical critical information infrastructure with elements of supervisory control and data acquisition. The exercise included a sixteen person strong red team (i.e. attackers), six blue teams (i.e. defenders) of 6-10 people per team, a white team (i.e. game management), a green team (i.e. technical infrastructure management) and one white observer per team. The majority of the exercise's participants were computer security specialists and some were computer security researchers. They were affiliated with various northern European governments, military, private sector and academic institutions. The interested reader is referred to [Geers, 10][Holm, 11] for more thorough descriptions of the exercise.

The survey was distributed to all blue team and red team members. A sample of fifteen respondents from the cyber defense exercise fully completed the survey and thus provides the empirical data for this study. Members from all teams except one blue team and the red team answered the survey. Respondent ages ranged from 26 to 53 years, with a mean of 33 years. The respondents were asked to grade their general expertise in the IT-security domain from 1 = no expertise to 5 = very high expertise. The respondents' self-assessed expertise ranged from no expertise (1 respondent) to very high expertise (3 respondents), with a mean of 3.33.

4.2 The questionnaire

The respondents were asked to specify the probability of carrying out successful arbitrary code execution attacks given all possible combinations of the three examined variables. The probability that *successful arbitrary code execution* is in the state *yes* was assessed through a discrete scale of 0-100 percent. Hence, the answer 0 percent means that there is no possibility of a successful attack and the answer 100 percent means that the attack is certain to succeed. Due to the complexity of the question format each question was, as recommended by [Garthwaite, 05], accompanied by a figure describing the scenario. Respondents were also required to state the confidence of their answers on a scale from 1 (very low confidence) to 5 (very high confidence).

The survey was composed of five main parts: (i) introductory letter, (ii) respondent background, (iii) respondent training, (iv) probability assessments, and (v) importance of other variables. In the final part of the survey, (v), the respondents were asked whether there were any additional important factors, not in the study. Survey training is something of particular importance when trying to assess information for complex questions, to ensure that respondents have correctly interpreted the topic of interest [Garthwaite, 05]. This survey included a training section, familiarizing the respondents with several concepts: the question format and the scales used, the overall scenario and the CVSS.

4.3 Implicitly defined variables

As stated in Sect. 3, there are numerous variables that are of importance to the probability of successful remote code execution attacks. The survey only explicitly defined the state of three variables. However, the questions were formulated in a way that implicitly fixed other relevant variables. The exact formulation was *"For the coming questions, assume that you are to carry out an arbitrary code execution attack against a service running on a machine in one of the blue team's networks. Also assume that you can connect the machine and the service in question, i.e. you have access to the service's LAN."*

As all respondents had been given previous access to the network architecture, had equal knowledge of the scenario and therefore should have pictured the same general network and defense mechanisms, this overall description effectively fixed the states of all variables of importance.

4.4 Method for analysis of expert assessments

A designed experiment in the shape of a complete factorial design [Montgomery, 08] was employed as main means of gathering data. Designed experiments concern extraction of a maximum amount of unbiased information regarding the factors affecting a process from as few observations as possible [Montgomery, 08]. This is a method that is traditionally employed in observational studies but which is equally potent when eliciting information through questionnaires [Gable, 94].

A complete experimental design means that every possible variable-state combination is tested. In the context of the present study with three variables, each with two states, there are 2^3 possible combinations to consider (cf. Figure 1). Studying all possible combinations enables analysis on how different combinations of variables and states affect the outcome. Such analysis is very important as the outcome in most cases depend on combinations of factors. For example, in the context of detecting SQL injection vulnerabilities [Antunes, 10]: If white-box testing in general detects 23% and black-box testing detects a mean of 63% of the existing vulnerabilities then the combination of these approaches are not likely to detect a mean of 89% of the existing vulnerabilities. It is more likely that running two such static based detection techniques finds roughly the same security issues, or perhaps, that one detects a subset of the other approach (detection rate by black-box tools is a subset of detection rate by white-box tools). As such it is naturally very valuable to study the importance of a variable when it is coupled with other variables.

When analyzing results gained through designed experiments it is important to use sound methodology and statistical techniques [Montgomery, 08]. That is, to analyze the results critically in terms of e.g. distribution assumptions (potential lack of fit) and strength of formulated hypotheses. Recommended such statistical tools include QQ-plots, ANOVA and regression analysis [Montgomery, 08]. These tools are described below.

QQ-plots are used to assess the distribution of data sets, e.g. to make sure that statistical methods used are applicable to the distributions at hand [Warner, 08].

Analysis of variance, ANOVA, is a collection of tools used for statistical tests to assess the significance of relations between variables. The null hypothesis of an ANOVA, H0, is true if the differences between observed groups of data can be

described by chance and false if there are systematic differences large enough to justify rejection of H0. The boundary associated with rejecting a null hypothesis is generally described using a probability, *p*. Datasets containing groups of observed data with differences large enough to reject the null hypothesis are statistically significant. A commonly used probabilistic boundary is $p < 0.05$, which implies that there is less than 5% probability for H0 to be true [Warner, 08].

A regression analysis provides an equation that models the expected outcome on a quantitative variable Y from data on one or more variables X_1, $X_2 \ldots X_n$ [Warner, 08]. A key task in regression analysis is to determine how well the identified equation actually models the variation in a dataset. The measure of the spread of points around the regression line can be presented using the coefficient of determination, R^2, where $0 < R^2 \leq 1$ [Warner, 08] . In other words, R^2 measures how well the regression model explains the variation in the dataset. An R^2 of 1 (100%) means the model explains all variation. The adjusted R^2 is a version of R^2 which also compensates for the number of degrees of freedom [Montgomery, 08], and is therefore generally to be preferred over the traditional R^2.

5 Results and analysis of expert assessments

An overview of the survey responses regarding the probability of successful remote code execution is given in Table 1. As can be seen, access seems to be the most influential variable, while presence of non-executable memory (NX) seems to be of minor importance. The normal distribution of the eight studied scenarios was determined using QQ-plots.

Table 1. Studied scenarios and their results (in probability of success (%)). LCI: 95% Lower Confidence Interval. UCI: 95% Upper Confidence Interval.

Scenario	NX	Access	Exploit	Mean	Stdev	LCI	UCI	Samples
1	Yes	Yes	High	85.6	8.3	81.4	89.8	15
2	Yes	Yes	Medium	74.6	11.5	68.6	80.7	15
3	No	Yes	High	81.2	14.5	73.9	88.6	15
4	No	Yes	Medium	65.7	14.3	58.5	72.3	15
5	Yes	No	High	54.7	25.4	41.8	67.5	15
6	Yes	No	Medium	42.9	21.7	31.9	53.9	15
7	No	No	High	52.3	30.2	37.0	67.6	15
8	No	No	Medium	43.7	28.5	29.2	58.1	15

As all possible combinations (2^3) between the studied variables (non-executable memory (NX), access and available exploits) were evaluated it was possible to analyze the results using traditional experimental design techniques [Montgomery, 08] (cf. Sect. 4.4), treating the respondents' answers as the experiment's outcome.

The experimental design was as described in Sect. 4.4 analyzed through QQ-plots, ANOVA, regression analysis and evaluation of model assumptions through an analysis of residuals. No problems regarding residual lack of fit (e.g. faulty distribution assumptions) were found, and the model assumptions should thus not be rejected.

A statistical analysis of the experts' assessments (cf. Table 2) shows that both access ($p < 0.0001$) and the severity of the exploited vulnerability ($p = 0.055$) are of importance for successful remote code executions value. However, non-executable memory is not seen as important by the respondents ($p = 0.32$). Furthermore, there are no seemingly important relations between any of the studied variables (NX, access and exploit), suggesting that the influence of these variables on successful remote code execution are in fact independent from one another. Equation 1 describes how the most important variables; Access (A) and the Severity of the exploited vulnerability (E), relate to the probability of successful remote code execution (P).

$$P = 62.59 + 14.21 * A + 3.69 * E \quad (\mathbf{1})$$

The value of A is {No access, Access} = {-1, 1} and the value of E is {Medium, High} = {-1, 1}. The chosen regression model (cf. Equation 1) has an adjusted R^2 of 0.32 which can be considered decent but not great. The regression model suggests that it will be quite likely to succeed with a remote code execution, even if there only is an exploit for a Medium vulnerability available and the attacker does not have access (44.7% probability), and highly probable if there is an exploit for a High vulnerability available and the attacker has access (80.5% probability).

Table 2. Results from the designed experiment.

Source	p-value
NX	0.32
Access	< 0.0001
Exploit	0.055
NX-Access	0.44
NX-Exploit	0.32
Access-Exploit	0.13
NX-Access-Exploit	0.11

Table 3 describes how certain the experts were for the different scenarios (on a scale from 1 to 5). The experts are overall fairly certain: no confidence interval goes below 2.5 (out of 5). Furthermore, all confidence intervals in Table 3 largely overlap and there is thus no reason to believe that the importance of any scenario or variable is more difficult to assess than another.

Table 3. How certain the experts were for the different scenarios (on a scale from 1 to 5). LCI: 95% Lower Confidence Interval. UCI: 95% Upper Confidence Interval.

Scenario	NX	Access	Exploit	Mean	Stdev	LCI	UCI	Samples
1	Yes	Yes	High	3.5	1.5	2.8	4.3	15
2	Yes	Yes	Medium	3.4	1.3	2.7	4.1	15
3	No	Yes	High	3.6	1.2	3.0	4.2	15
4	No	Yes	Medium	3.2	1.1	2.7	3.8	15
5	Yes	No	High	3.1	1.2	2.5	3.7	15
6	Yes	No	Medium	3.1	1.1	2.5	3.6	15
7	No	No	High	3.3	1.0	2.8	3.9	15
8	No	No	Medium	3.3	1.3	2.6	4.0	15

6 Observations of actual attacks

An important aspect when using expert judgment is to evaluate the reliability of the provided information. One such method is to use the validity-based approach, to compare an actual outcome to an expert's assessment [Hoenig, 85]. This method is appealing in its simplicity. However, this approach is often impractical as experts are needed in situations where correct answers seldom exist [Gigerenzer, 99].

The exercise featured observations of actual attacks against the systems which were the focus of the expert study. However, the systems in the exercise were identical in terms of the variable states in the prediction model. As such, it is possible to use the validity based approach to study the accuracy of the expert prediction model – however, only a single scenario (scenario 3, cf. Table 1) could be studied.

6.1 Methodology and data collection for observations of attacks

Actual observations of attacks were collected through three sources of data captured during the exercise: network data, red team attack logs and white team observer logs. This chapter describes these sources of data.

Only successful compromises in one of network zone types, the demilitarized zone (DMZ), were captured. This was done due to that the DMZ environment was static in terms of what services (and versions of them) that were required to be operated. Furthermore, services on systems in the DMZ were directly reachable by attackers, making these systems good candidates for the present study. Table 4 describes the seven systems in the DMZ in terms of operating systems and which services that were needed to be operational and externally accessible

Also, one of the blue teams configured their firewall to block all traffic to and from the DMZ (due to a misunderstanding of the exercise reward system). These data points are excluded to increase the overall data quality.

Table 4. Studied systems in terms of observations of actual code injection attacks.

System	Operating System	Services
External Firewall	Debian 5.04 (lenny/stable)	HTTP, SSH
DNS + NTP	Debian 4.0r1	DNS, FTP, NTP, SSH
E-mail	Debian 4.0r1	HTTP, IMAP, POP3, SMTP, SSH
Customer Portal	Debian 4.0 etch/oldstable	HTTP, HTTPS, SMTP, SSH
Public Website	Win Server 2003	FTP, HTTP, HTTPS, RFB, SSH
Historian	Win Server 2000 SP4	FTP, HTTP, RFB, SMTP, SSH
News	CentOS 5.4	HTTP, SSH

TCP-dump sniffers were placed in all blue team networks in such a fashion that all network traffic in and out of all the different network zones was captured. The resulting .pcap files were then run through the intrusion detection system Snort [Sourcefire, 11] to assess all malicious traffic, resulting in approximately 3,000,000 alarms. Attackers used dynamic IPs to hide their identities but as all utilized IPs were uniquely mapped to the attackers it was possible to identify the attackers through their corresponding static MAC addresses. Due to the possibility of false alarms generated by network traffic of the blue teams' only traffic originating from attacker MAC addresses was studied. Thus, the possibility of assessing attacks from compromised machines was lost in order not to compromise the data quality. However, as the large majority of attacks originated from attacker MACs this is estimated to be a minor issue. Of the 3,000,000 original alarms approximately 100,000 corresponded to traffic from attacker MAC addresses. Of these 100,000 alarms approximately 19,000 corresponded to Snort signatures for arbitrary code execution attempts. As Snort generally give multiple alarms for a single exploit an algorithm was used to elicit unique arbitrary code execution attacks. This algorithm involved checking if there were multiple alarms originating from a specific MAC address against a specific destination IP and a specific destination port (e.g. 21 - FTP) within less than 30 seconds. If yes, these alarms were aggregated to unique attacks. This resulted in a set of 169 attempted code injection attacks.

While probing and attempted intrusions could be derived from the network data, successful attacks could not. This due to that it is very difficult to differentiate attempted intrusions from successful intrusions through Snort. For example, one reason it that successful attacks often carried payloads (e.g. terminals) that used encrypted means of communication. Successful attacks were instead identified from logging by the red team and observers. As part of the exercise an important objective for the red team was to take a screen capture every time they managed to compromise a system. Additionally, observers with the objective to log successful attacks were present at the location of the red team to log any successful intrusions. Both of these datasets were used to identify successful attacks. Also, a single system could be compromised several times as the blue teams often were able to regain control over previously compromised systems. The data set includes a total of 56 successful

intrusions in the DMZ, not counting the successful intrusions against the blue team with the misconfigured firewall.

Unfortunately, as the red team and observers sometimes did not note which particular service and vulnerability that was exploited observations could only be assessed on a system level rather than a service level as was the viewpoint presented in the survey to the respondents (cf. Sect. 4). If each exploited service and vulnerability had identical properties this would not be an issue – the probability of success against a system would simply correspond to a mean of a set of representative attacks against services and vulnerabilities on a particular system.

While all services in the DMZ had CVSS high severity vulnerabilities, not all were accessible by the attackers as regular users of them. As such, there could be slight reliability issues if one would simply regard the observed success rates as strictly representative of the topic of the present study; any interpretation of the comparisons between observations and estimations by respondents need take this fact into account. However, we believe that while this clearly is an issue, the systems were fairly similar in terms of the studied variables and any reliability problems regarding comparisons between actual and perceived attack rates should as such be limited.

6.2 Observed attacks

An overview of the results regarding variable states for the systems in the DMZ, and the observed success rates of code injection attacks against these systems, can be seen in Table 5. The success rates for attacks against the systems were highly varied, ranging from 5.6% (DNS + NTP) to 77.8% (External Firewall). This suggests that there are variables of great importance that are not modeled in Table 5.

Table 5. Variable states and observed success rates of code injection attacks against the systems in the DMZ.

System	NX	Access*	Exploit	Attempted attacks	Successful attacks	Probability of success
External Firewall	No	Yes/No	High	9	7	77.8%
DNS + NTP	No	Yes/No	High	18	1	5.6%
E-mail	No	Yes/No	High	23	8	34.8%
Customer Portal	No	Yes/No	High	16	9	56.3%
Public Website	No	Yes/No	High	40	17	42.5%
Historian	No	Yes/No	High	42	9	21.4%
News	No	Yes/No	High	21	5	23.8%

*Variable state not known.

6.3 A comparison of expert assessments and observations

The regression model elicited by the respondents of the study (cf. Equation 1) predicts an 80.49% probability of success given the variable states in Table 5, a number out of bounds for the range of the observed success rates. There are many possible reasons

behind this, with four likely being: (i) The experts had another view of what is comprised by an 'attack'. (ii) The experts in general pictured the External Firewall as the target in question. Given such a scenario, the prediction model is of high quality – at least for this particular scenario. (iii) Some would-be experts were not experts after all. (iv) There are other variables than the three evaluated in this study that are of great importance towards the success rate of code injection attacks.

The scope of the attempted arbitrary code execution attack was not thoroughly detailed in the questionnaire (cf. Sect. 4.3). As such, some respondents might have viewed an attack as a set of exploits (or a single exploit which is tested several times with different tunings) rather than an individual run of a single exploit. Given such a viewpoint, the actual success rate against each system in the DMZ was in fact 100% (i.e., all systems were compromised at least once).

Assuming that the respondents did view an attack as a single run of an exploit; the estimates by the 15 respondents for the tested scenario show that their approximations range from 50% to 99%. As such, it is likely that each expert had different experiences from code injection attacks given systems with properties such as those in the DMZ. This supports the simple logic that the experts most likely did not all picture the External Firewall as the target of the attack. More likely, the experts pictured a more general system representative of the exercise as a whole, as depicted in the questionnaire (cf. Sect. 4.3) Furthermore, some experts were closer to the mean of the actual success rates, 33.1%, than others. The most accurate expert, which perceived a success rate of 50% for the compared scenario, had 13 years of professional experience from working with IT security and had a self-assessed competence score of 3 out of 5. There were many respondents that were better experts according to these simple metrics – something which perhaps says something of their usefulness. This hint towards that some experts indeed are more accurate than others and as such, that the best result is gained through the usage of more potent expert scoring methods.

The most significant result of the study however lies in the variance given by both the experts' and the observational data. It is clear that several significant variables were not modeled in this study. In other words, there is a need to detail more variables in order to get a satisfactory prediction model. This is elaborated in Sect. 7.

7 Discussion

This section involves a critical discussion of the results obtain in this study. The first section includes a discussion of the variables tested in the study; the second section features a discussion on the topic of reliability and validity.

7.1 Variables and their importance

Both access and the severity of the exploited vulnerability are seen as important by the experts. However, non-executable memory is not seen as relevant. This can be due to respondents being able to tune the exploits to counter this defense mechanism. That is, a stack-smashing exploit can quite easily be tuned into an arc-injection (i.e. return-into-libc) [Pincus, 04] exploit, and if this is the case any non-executable memory defense is rendered next to useless. However, it is frequently pointed out that non-

executable memory is a potent defense if combined with other countermeasures, such as address space layout randomization [PaX, 03][Shacham, 04] or canaries such as Stackguard [Cowan, 98]. Thus, the reason why non-executable memory did not turn out as significant during this study could be that the states of the numerous other important variables in place during the cyber defense exercise were fixed in such a way that enabling non-executable memory would not make any difference.

The respondents were also asked to detail any missing variables they perceived as important to the probability of successful remote code execution, both in the survey (cf. Table 6), and through informal discussions after the cyber defense exercise. It is notable that no respondents listed the same missing variables. However, the discussions and variance in estimations made it clear that several variables of great practical importance should be evaluated further, for example address spaced layout randomization. This is also contingent to the findings gained from observations of actual attacks (cf. Sect. 6) – there is a need to detail more variables in order to reach satisfactory data quality.

Table 6. Missing variables.

Variable	Frequency
System monitoring	1
System updates	1
Application sandboxing	1
Unknown vulnerability	1
Service misconfiguration	1

As such a very important result gained through this study is that the variables studied are perceived to be fairly independent from each other. This suggests that it might be possible to reduce the number of studied scenarios without decreasing the data quality of the resulting prediction model. In other words, it might not be necessary to study the dependencies between each and every variable – especially not those in the present study. It could be useful to qualitatively estimate any dependencies beforehand and use such knowledge when gathering quantitative data; giving a means to collect information on significantly more variables and/or variable states.

7.2 Validity and reliability

While there were no formal quantitative ways of capturing the reliability of the questionnaire tool, no respondent had any major issue answering any survey question. However, one thing that came up during the informal discussions was that respondents requested the possibility to answer through probability distributions instead of point estimates. Also, the survey was based on a network which the respondents had practical experience from. Numerous variable states were therefore not specified in the survey but were instead known through the respondents' familiarity with the scenario. While this might increase the reliability of the study (by

giving all respondents a common understanding of the scenario) it has negative impact on the external validity. Several important variables were static and it is therefore difficult to generalize the results to a more general case, e.g. another enterprise environment. Thus, while the results might be valid and reliable for a cyber defense exercise of this type and the computer network used in it they are not likely to maintain validity for a more general case. We believe, however, that these results should give a clear hint about the relative importance of the studied variables, even for a more general case.

The comparisons between estimations and observations for the scenario represented by access to the service, an exploit for a high vulnerability available, and without NX, show that the prediction model's estimation differs significantly from the mean of the observations (80.49% compared to 33.1%). As-is, the prediction quality is as such likely not very reliable. However, results also show that a better prediction model could be gained if one would select the best expert(s) or weight the quality of their judgment according to some well-defined scoring system. Two such models which could provide adequate results are Cochran-Weiss-Shanteau [Weiss, 03] and Cooke's classical method [Cooke, 91]. Both methods however require more details than what is available in this study, and are thus not applicable to this particular case. It would however be very interesting to conduct future studies according to such a scheme and evaluate if greater prediction model data quality is obtained.

Additionally, the comparisons show that there likely is a need to detail more variables in order to reach satisfactory data quality. One such variable could be ASLR, as discussed in Sect. 7.1. Another such variable is a more thorough definition of an attack (cf. Sect. 6.2).

8 Conclusions and future work

Among the variables addressed in this study, experts perceive user access to the service that is to be exploited as the most important studied factor for successful remote code execution. A readily available exploit for a High severity vulnerability as defined by the CVSS gives a greater success rate than a readily available exploit for a Medium severity vulnerability. The presence of non-executable memory is not perceived as important by the experts, presumably due to lack of other related countermeasures such as address space layout randomization.

This study shows that it is possible to obtain approximate estimates from security experts on issues that are difficult to evaluate through experiments. The experts tend to agree with each other, indicating that they are experts in the domain [Einhorn, 74]. Comparisons between estimates and observations of actual code injection attacks however show that there is a need to value the judgment of more competent experts higher than those with lower competence, perhaps through the usage of a proper scoring function such as Cooke's classical method [Cooke, 91]. The results also show that there is a need to detail more variables then the ones studied in this paper to assess sufficient prediction quality.

Furthermore, the studied variables are perceived to have independent effect on the likelihood of successful code injection attacks which could enable analysis of a greater number of variables without significantly increased data collection costs or decreased quality of the resulting prediction model. As such, future studies involving

expert judgment should apply a proper scoring system and include more variables in the study, and dependencies do perhaps not need to be studied.

Finally, we believe that there is a great need for more quantitative studies assessing the importance of different countermeasures, not only for arbitrary code execution attacks, but for any cyber attack.

References

[Antunes, 10] Antunes, N. and Vieira, M. Benchmarking Vulnerability Detection Tools for Web Services. IEEE, 2010.

[Baratloo, 00] Baratloo, A. and Singh, N. Transparent run-time defense against stack smashing attacks. Proceedings of the annual conference on USENIX, (2000).

[Campbell, 03] Campbell, K., Gordon, L.A., Loeb, M.P., and Zhou, L. The economic cost of publicly announced information security breaches: empirical evidence from the stock market. Journal of Computer Security 11, 3 (2003), 431–448.

[Cavusoglu, 04] Cavusoglu, H., Mishra, B., and Raghunathan, S. The effect of internet security breach announcements on market value: Capital market reactions for breached firms and internet security developers. International Journal of Electronic Commerce 9, 1 (2004), 70–104.

[Chiueh, 05] Chiueh, L.-chung L.T.-cker. Checking Array Bound Violation Using Segmentation Hardware. 2005 International Conference on Dependable Systems and Networks (DSN'05), (2005), 388-397.

[Cooke, 91] Cooke, R.M. Experts in uncertainty: opinion and subjective probability in science. Oxford University Press, USA, 1991.

[Cooke, 04] Cooke, R. and Goossens, L. Expert judgement elicitation for risk assessments of critical infrastructures. Journal of Risk Research 7, 6 (2004), 643–656.

[Cowan, 98] Cowan, C., Pu, C., Maier, D., et al. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. Proceedings of the 7th conference on USENIX Security Symposium-Volume 7, Usenix Association (1998), 5–5.

[Cowan, 01] Cowan, C., Barringer, M., Beattie, S., Kroah-Hartman, G., Frantzen, M., and Lokier, J. FormatGuard: Automatic protection from printf format string vulnerabilities. Proceedings of the 10th conference on USENIX Security Symposium-Volume 10, USENIX Association (2001), 15–15.

[Cowan, 03] Cowan, C., Wagle, P., Pu, C., Beattie, S., and Walpole, J. Buffer Overflows : Attacks and Defenses for the Vulnerability of the Decade. Foundations of Intrusion Tolerant Systems, 2003 [Organically Assured and Survivable Information Systems], (2003), 227-237.

[Dor, 01] Dor, N. Cleanness checking of string manipulations in C programs via integer analysis. Static Analysis, (2001).

[Einhorn, 74] Einhorn, H. Expert judgment: Some necessary conditions and an example. Journal of Applied Psychology, (1974).

[Erlingsson, 99] Erlingsson, U. and Schneider, F.B. SASI enforcement of security policies: A retrospective. Proceedings of the 1999 workshop on New security paradigms, ACM (1999), 87–95.

[Erlingsson, 07] Erlingsson, U. Low-level Software Security : Attacks and Defenses Low-level Software Security : Attacks and Defenses. Redmond, WA, USA, 2007.

[Forrest, 96] Forrest, S., Hofmeyr, S.A., Somayaji, A., and Longstaff, T.A. A sense of self for unix processes. In Proceedings of the IEEE Symposium on Security and Privacy, Published by the IEEE Computer Society (1996), 120-128.

[Frykholm, 00] Frykholm, N. Countermeasures against buffer overflow attacks. RSA Tech Note, (2000), 1-9.

[Gable, 94] Gable, G.G. Integrating case study and survey research methods: an example in information systems. European Journal of Information Systems 3, 2 (1994), 112–126.

[Garthwaite, 05] Garthwaite, P.H., Kadane, J.B., and O'Hagan, A. Statistical methods for eliciting probability distributions. Journal of the American Statistical Association 100, 470 (2005), 680-701.

[Geers, 10] Geers, K. Live Fire Exercise: Preparing for Cyber War. (2010).

[Gigerenzer, 99] Gigerenzer, G. and Todd, P.M. Simple heuristics that make us smart. Oxford University Press, USA, 1999.

[Haimes, 2011] Haimes, Y.Y. Accident Precursors, Terrorist Attacks, and Systems Engineering. NAE Workshop, (2003).

[Hoenig, 85] Hoenig, M. Drawing the Line on Expert Opinions. J. Prod. Liab. 8, (1985), 335–336.

[Holm, 11] Holm, H., Sommestad, T., Almroth, J., and Persson, M. A quantitative evaluation of vulnerability scanning. Information Management & Computer Security, 19, 4 (2011).

[Homer, 09] Homer, J., Manhattan, K., Ou, X., and Schmidt, D. A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks. Citeseer, Citeseer (2009).

[Jim, 02] Jim, T., Morrisett, G., Grossman, D., Hicks, M., Cheney, J., and Wang, Y. Cyclone: A safe dialect of C. USENIX, (2002), 275-288.

[Joyce, 92] Joyce., R.H. and B. Purify: Fast detection of memory leaks and access errors. Winter USENIX Conferenc, (1992), 125--136.

[Kc, 03] Kc, G., Keromytis, A., and Prevelakis, V. Countering code-injection attacks with instruction-set randomization. Proceedings of the 10th ACM conference on Computer and communications security, (2003), 280.

[Madan, 02] Madan, B.B., Gogeva-Popstojanova, K., Vaidyanathan, K., and Trivedi, K.S. Modeling and quantification of security attributes of software systems. International Conference on Dependable Systems and Networks, IEEE Computer Society (2002).

[Mell, 07] Mell, P., Scarfone, K., and Romanosky, S. A complete guide to the common vulnerability scoring system version 2.0. Published by FIRST-Forum of Incident Response and Security Teams, (2007), 1-23.

[Montgomery, 08] Montgomery, D.C. Design and analysis of experiments. John Wiley & Sons Inc, 2008.

[Newsome, 05] Newsome, J. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. Network and Distributed System Security, May 2004 (2005).

[One, 96] One, A. Smashing the stack for fun and profit. Phrack magazine 7, 49 (1996).

[PaX, 03] PaX Team. PaX address space layout randomization (ASLR). 2003.

[Perens, 11] Perens, B. Electric fence 2.0.5. 2011. http://perens.com/FreeSoftware/.

[Pincus, 04] Pincus, J. and Baker, B. Beyond stack smashing: Recent advances in exploiting buffer overruns. Security & Privacy, IEEE 2, 4 (2004), 20–27.

[Ryan, 10] Ryan, J.J.C.H., Mazzuchi, T.A., Ryan, D.J., Lopez de la Cruz, J., and Cooke, R. Quantifying information security risks using expert judgment elicitation. Computers & Operations Research, (2010).

[Shacham, 04] Shacham, H., Page, M., Pfaff, B., and Goh, E. On the effectiveness of address-space randomization. ACM conference on, (2004), 298.

[Small, 97] Small, C. A tool for constructing safe extensible C++ systems. Proceedings of the 3rd conference on USENIX Conference on Object-Oriented Technologies, USENIX Association (1997), 175-184.

[Sommestad, 10] Sommestad, T., Ekstedt, M., and Johnson, P. A probabilistic relational model for security risk analysis. Computers & Security 29, 6 (2010), 659–679.

[Sourcefire, 11] Sourcefire. Snort. 2011. An intrusion-detection model.

[Taylor, 02] Taylor, C., Krings, A., and Alves-Foss, J. Risk analysis and probabilistic survivability assessment (RAPSA): An assessment approach for power substation hardening. Proc. ACM Workshop on Scientific Aspects of Cyber Terrorism,(SACT), Washington DC, Citeseer (2002).

[Warner, 08]Warner, R.M. Applied statistics: From bivariate through multivariate techniques. Sage Publications, Inc, 2008.

[Weiss, 03] Weiss, D.J. and Shanteau, J. Empirical assessment of expertise. Human Factors: The Journal of the Human Factors and Ergonomics Society 45, 1 (2003), 104.

[Wilander, 03] Wilander, J. and Kamkar, M. A comparison of publicly available tools for dynamic buffer overflow prevention. Proceedings of the 10th Network and Distributed System Security Symposium, Citeseer (2003), 149–162.

[Xenitellis, 03] Xenitellis, S.D. Identifying security vulnerabilities through input flow tracing and analysis. Information Management & Computer Security 11, 4 (2003), 195-199.

[Younan, 08] Younan, Y. Efficient countermeasures for software vulnerabilities due to memory management errors. status: published, 2008.