Towards a Unifying Approach in Understanding Security Problems

Prasanth Anbalagan
Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA
panbala@ncsu.edu

Mladen Vouk
Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA
vouk@ncsu.edu

Abstract

To evaluate security in the context of software reliability engineering, it is necessary to analyse security problems, actual exploits, and their relationship with an understanding of the operational behaviour of the system. That can be done in terms of the effort involved in security exploits, through classic reliability factors such as calendar and in-service time, etc. Existing studies focus primarily on security problems and security exploits. Less attention has been given to the study of the relationship between security problems and security exploits.

We present an analysis and classification of 43,710 vulnerabilities from the Open Source National Vulnerability Database and vulnerabilities for two specific products - Bugzilla and FEDORA. About 35% of the published vulnerabilities have been exploited. 34% of the vulnerabilities are disclosed as a result of an exploit and only 1.3% have been exploited after being publicly disclosed.

We investigate a unifying approach, to understand security as a component of reliability. We focus on the disclosure and exploits of security problems with respect to calendar time and in-service time, and the impact of such exploits on the process of correcting the security problems, and discuss our approach using the collected data.

1 Related Work

Security problems (or vulnerabilities or faults) and security exploits (or attacks or failures) are a subset of the general category of software faults and failures [25]. Existing studies discuss security problems and exploits from the viewpoints of types of attacks (e.g. [24, 30, 28, 5]), types of security problems (e.g. [22, 9, 32, 33, 23]), security patches (e.g. [12]), attacker behaviour (e.g. [10, 19, 1]), a developer (e.g. [15, 6]), a system (e.g. [16, 21]), prediction (e.g. [27, 2, 4, 3]), policies and qualitative discussions (e.g. [11, 17, 8, 18]), comparison with reliability (e.g. [20, 31]), etc.

Alhazmi et al. [2, 3, 4] present a vulnerability discovery model and predict the long term and short term number of vulnerabilities for several major operating systems including RedHat Linux. Thomas et al. [26] study RedHat packages and correlate vulnerabilities with dependencies between packages. Thomas et al. [27] map past vulnerabilities to components, and build a model to predict vulnerabilities based on past history and function calls. Fengmin et al. [16] propose a model to analyse the behaviour of an intrusion detection system and its response. Bharat et al. [21] present a semi Markov model to quantify the security attributes of the model proposed by Fengmin et al.

Schniedwind [31] provides a study on the conditional probability of security failures given that the reliability failures have occurred. Littlewood et al. [20] provide an overview of similarities between reliability and security from the perspective of operational behaviour of a system rather than just the safeguards used in the design and implementation phases of the system. Stefan [15] provides a comparison on the total number of exploits experienced and the number of patches released after disclosure of vulnerabilities. This study provides an overall view of exploits and patches available, but does not study the effect of exploits on correcting the vulnerabilities. Anbalagan et al. [6] show that security problems have different characteristics from non-security problems in terms of reporting and correcting them. Huseyin et al. [12] discuss when vendors should release patches and when customers should apply the patches.

Lawrence et al. [10] provide a cost benefit analysis in attackers trying to exploit vulnerabilities. Jonsson et al. [19] discuss that attackers follow a learning, attack and innovative phase in carrying out attacks. Ashish et al. [1] study the frequency of attacks from attackers based on the disclosure status of vulnerabilities. William et al. [22] classify security flaws in operating system based on features like error handling, parallelism, etc. Bellowin et al. [9] classify the different types of security faults observed in TCP-IP protocol suite. Mathur et al. [33] classify vulnerabilities based on the cause of the attack, its impact, and fix required for recovery. Tsipenyuk et al. [32] discuss the classification of vulnerabilities based on code quality and the environment.
in which such vulnerabilities are exploited. Miles et al. [23] discuss a specific type of vulnerability - those that are hidden and then exploited.


1.1 Objectives

While it is necessary to study security problems and exploits individually, it is also necessary to understand the effect of one on another. We present a model of relationships between security problems and their exploits in the field. Using our model, we evaluate how promptly a project team fixes security problems, i.e., whether there is any backlog in fixing security problems or not, a project team’s response to security problems that have experienced failures (exploits) and those that remain unexploited in the field. Existing studies capture the operational behaviour of security [20] mostly in terms of the effort. We capture the operational behaviour of security in terms of calendar time and inservice time [25, 13]. Our objectives are to:

1. Using vulnerabilities from the Open Source Vulnerability Database\(^1\) and the National Vulnerability Database\(^2\), discuss a model that captures the states through which a system may go based on the the type of the vulnerabilities, their disclosure status, exploit status, and their correction status.

2. Map vulnerabilities for Bugzilla and FEDORA products to the model. We then estimate model parameters in terms of rates of disclosure and exploit of security problems [29], and in terms of correction rates, all with respect to calendar time as well as inservice time.

3. Discuss the results of our study.

The paper is organized as follows. In Section 2, we discuss characteristics of the vulnerability data from the various vulnerability databases. In Section 3, we discuss a model based on our analysis of the vulnerability data. In Section 4, we discuss the metrics in detail. In Section 5, we discuss estimation of security problem rates. In Section 6, we further discuss our results and limitations of our approach. Section 7 concludes the paper. This work is an extension of [7] which describes how promptly high, medium, and low severity security problem reports for FEDORA, Ubuntu, Suse and RedHat Linux are corrected. The model presented in this paper attempts to capture user involvement in actual security exploits and the effect of exploits on the process of correcting the security problems. This study is based on 43710 vulnerabilities from the OSVDB and vulnerabilities specific to Bugzilla from the OSVDB, and FEDORA product from the RedHat bug tracking system\(^3\). Our study is completely based on publicly available data and does not include any insider information like information on security process within RedHat.

2 Vulnerabilities

A much discussed issue is the disclosure of vulnerabilities. Information about vulnerabilities is either kept secret until the problem is resolved and is then released when a fix is available, or the information about vulnerabilities is made available before the problem has been resolved and a fix has been released. An argument in favour of non-disclosure is that releasing information about a security problem may increase the risk for end-users since it makes a larger number of potential hackers aware of it, and until a fix is available, this increases the exposure of end-users to danger. A counter-argument may be that disclosure of security problems helps alert end-users in taking precautions against exploits. In this section, we study various characteristics of the collected vulnerabilities.

2.1 Status of vulnerabilities

A vulnerability exploit is either known or not. Vulnerabilities in the OSVDB contain details on whether a vulnerability has been exploited (we call them “exploit known” problems), or an exploit of the vulnerability is unknown. The category of “exploit unknown” problems tells us that an exploit is not known for a security problem. However, it is possible that the security problem may have been exploited in the field, but the exploit was not reported. We still are interested in observing the process of correcting security problems knowing that the problems have been exploited in the field. Table 1 shows the proportion of vulnerabilities with exploits known and unknown. From the table, we find that exploited vulnerabilities constitute about 35% of the vulnerabilities found in the OSVDB. Existing studies capture the process of correcting the overall set of security

\(^1\)http://osvdb.org
\(^2\)http://nvd.nist.gov/
\(^3\)https://bugzilla.redhat.com/
problems [6]. Since the exploited problems constitute a significant fraction, it is worthwhile to study the process of correcting these problems.

Knowing that a security problem has been exploited is likely to prompt the developers to fix problems faster. For example, consider the RedHat vulnerability #12727\(^4\) from OSVDB. This problem was disclosed on Jan 06, 2005 and exploited on Feb 12, 2005. A problem report corresponding to this vulnerability is maintained in the RedHat’s bug database with the bug id #144099\(^5\). Soon after the security problem was exploited on Feb 12, 2005, the security response team closed the bug with a fix and issued a security update or patch on Feb 15, 2005. Although the developers may have been working on the problems, it appears that an actual exploit of the security problem prompted the security response team in completing the correction activity and releasing the patch. In this paper, we study quantitatively how often security problems are disclosed, how many are exploited, and how quickly they are corrected. We distinguish the process of correcting such problems from that of the security problems for which an exploit is unknown.

### 2.2 Interaction with Attack Mechanism

One facet of how a vulnerability is exploited is whether a user is deceived into interacting with the attack mechanism to get exploited (we call these voluntary exploits). An example is a recent security alert from Adobe\(^6\). Adobe alerted of the presence of a critical security problem in Adobe Reader and Acrobat, on May 1, 2009, even before releasing the fix on May 12, 2009. The alert says that when a user opens a malicious PDF file from an untrusted source, the application would crash and the malicious PDF file can execute arbitrary code and take control of the affected system\(^7\). Adobe cautioned users from opening files from untrusted sources, and provided tentative solutions in preventing applications from automatically opening PDF files. One thing to note is that an exploit of this security problem is possible when users voluntarily interact with the attack mechanism by opening a malicious file. As already mentioned, we call such exploits “voluntary” exploits. On the other hand, we call the exploits that do not require any voluntary interaction from users as “involuntary exploits”.

Information on whether a vulnerability requires user to voluntarily interact with an attack mechanism or not, is available in the National Vulnerability Database, a U.S. government repository devised to manage vulnerability data. Table 2 shows the total number of vulnerabilities along with the number of “voluntary” type vulnerabilities. On the average 24% of the vulnerabilities are of “voluntary” type. In projects like Suse Linux (7.14%), voluntary exposure may not be significant. But for projects like Firefox, Internet Explorer, SeaMonkey, etc., it is significant.

<table>
<thead>
<tr>
<th>Project</th>
<th>Total</th>
<th>Voluntary</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora</td>
<td>908</td>
<td>192</td>
<td>21.1</td>
</tr>
<tr>
<td>Firefox</td>
<td>225</td>
<td>76</td>
<td>33.8</td>
</tr>
<tr>
<td>SeaMonkey</td>
<td>167</td>
<td>78</td>
<td>44.1</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>1105</td>
<td>244</td>
<td>22.1</td>
</tr>
<tr>
<td>RedHat</td>
<td>822</td>
<td>79</td>
<td>9.61</td>
</tr>
<tr>
<td>Thunderbird</td>
<td>165</td>
<td>61</td>
<td>37.0</td>
</tr>
<tr>
<td>OpenOffice</td>
<td>28</td>
<td>23</td>
<td>82.1</td>
</tr>
<tr>
<td>Suse</td>
<td>56</td>
<td>4</td>
<td>7.14</td>
</tr>
<tr>
<td>Apache Server</td>
<td>127</td>
<td>13</td>
<td>10.2</td>
</tr>
<tr>
<td>WindowsXP</td>
<td>440</td>
<td>126</td>
<td>28.6</td>
</tr>
<tr>
<td>IE 5.0</td>
<td>119</td>
<td>69</td>
<td>58.0</td>
</tr>
<tr>
<td>IE 6.0</td>
<td>352</td>
<td>115</td>
<td>32.7</td>
</tr>
<tr>
<td>Overall</td>
<td>4524</td>
<td>1070</td>
<td>23.7</td>
</tr>
</tbody>
</table>

Table 3 shows the data for exploited problems - a) those that were exploited and disclosed immediately, i.e., an exploit resulted in the disclosure the problem, b) those that were exploited but disclosed after waiting for a certain period of time, and c) those that were exploited after disclosure. The first category of problem reports - exploited and disclosed immediately tells us that these security problems were hidden initially and an exploit occurred in the field which eventually led to the disclosure of the security problem. Similarly the second category let to disclosure after a certain time period from the exploit date. The third category indicates that the security problem reports were exploited after disclosure but before they were fixed.

We can observe that the proportion of security problems exploited is about 35% (inclusive of voluntary and involuntary type security problems). About 65% (55.3% and 9.2%) of the security problems remain with exploit unknown. Compared to the proportion of vulnerabilities observed for individual projects in Table 2, the overall percentage of voluntary type security problems in the OSVDB database is about 12.8%. The data from OSVDB is a pool of security problems for a large number of products. While studying such a collection of security problems may give

---

\(^4\)http://osvdb.org/show/osvdb/12727
\(^5\)https://bugzilla.redhat.com/show_activity.cgi?id = 144099
\(^6\)http://www.adobe.com
\(^7\)http://www.adobe.com/support/security/advisories/apsa09-02.html

### Table 1. Data on exploits known/unknown

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of vulnerabilities</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploit known</td>
<td>15508</td>
<td>35.48%</td>
</tr>
<tr>
<td>Exploit unknown</td>
<td>28202</td>
<td>64.52%</td>
</tr>
<tr>
<td>Total</td>
<td>43710</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

### Table 2. Vulnerabilities from the NVD
Table 3. Voluntary/Involuntary

<table>
<thead>
<tr>
<th>Category</th>
<th>Involuntary</th>
<th>Voluntary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploited and then disclosed immediately</td>
<td>13306 (30.4%)</td>
<td>1489 (3.4%)</td>
</tr>
<tr>
<td>Exploited after disclosure but before fix</td>
<td>168 (0.4%)</td>
<td>20 (0.1%)</td>
</tr>
<tr>
<td>Exploited after a certain time period</td>
<td>463 (1.1%)</td>
<td>62 (0.2%)</td>
</tr>
<tr>
<td>Exploited (total)</td>
<td>13937 (31.9%)</td>
<td>1571 (3.6%)</td>
</tr>
<tr>
<td>Exploit status unknown</td>
<td>24166 (55.3%)</td>
<td>4036 (9.2%)</td>
</tr>
<tr>
<td>Total</td>
<td>38103 (87.2%)</td>
<td>5607 (12.8%)</td>
</tr>
</tbody>
</table>

Table 5. Bugzilla security problems

<table>
<thead>
<tr>
<th>Category</th>
<th>Voluntary</th>
<th>Involuntary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploited and then disclosed immediately</td>
<td>0 (0%)</td>
<td>28 (26.17%)</td>
</tr>
<tr>
<td>Exploited after disclosure but before fix</td>
<td>0 (0%)</td>
<td>1 (0.93%)</td>
</tr>
<tr>
<td>Exploited (total)</td>
<td>0 (0%)</td>
<td>29 (27.1%)</td>
</tr>
<tr>
<td>Exploit status unknown</td>
<td>9 (8.41%)</td>
<td>69 (64.49%)</td>
</tr>
<tr>
<td>Total</td>
<td>9 (8.41%)</td>
<td>98 (91.59%)</td>
</tr>
</tbody>
</table>

Involuntary

5607 (12.8%)

168 (0.4%)

9 (8.41%)

28 (26.17%)

Voluntary

13937 (31.9%)

0 (0%)

9 (8.41%)

Voluntary

38103 (87.2%)

62 (0.2%)

4036 (9.2%)

20 (0.1%)

69 (64.49%)

1 (0.93%)

98 (91.59%)

1489 (3.4%)

3 Model

In this section, we present a model based on our analysis of vulnerability data in Tables 2, 3, 4 and 5. The model captures the states through which a system may go based on the type of the security problems (voluntary or involuntary exploit security problems), disclosure status (publicly disclosed or hidden), exploit status (exploited or exploit unknown), and their correction status (fix known or unknown). Problem disclosure rates and the exploit rates are relatively slowly changing functions of time [21]. In this context, we discuss a generic multi-state model (Figure 2) that captures security transitions over time. Later we discuss the model based on estimated rates from the OSVDB data in Section 6. Since the problem disclosure rates vary with time [21], a non-homogeneous Markov chain model can be appropriate. Consider a continuous-time stochastic process X(t) (X(t),t ∈ R+) X(t) represents the system being in one of the states {Good,\( \text{disclosed}_{(v)} \), \( \text{disclosed}_{(nv)} \), \( \text{disclosed}_{(snv)} \), \( \text{exploited}_{(v)} \), \( \text{exploited}_{(nv)} \), \( \text{exploited}_{(snv)} \), \( \text{fixed} \}).

- \text{good or hidden}\ represents the start state where the system may contain undisclosed and unknown problems. A transition to the same state indicates that such problems, if present, remain with the exploit and fix status unknown.
- \text{disclosed}_{(v)}\ represents the disclosure of unexploited “voluntary” security problems. The disclosure is primarily through means other than actual exploits in the field. For example, a vendor could identify the problems during testing. A transition to the same state indicates that such problems remain with the exploit and fix status unknown. Similarly \text{disclosed}_{(nv)}\ represents the disclosure of unexploited “involuntary” security problems. \text{disclosed}_{(snv)}\ and \text{disclosed}_{(snv)}\ represent the disclosure of ‘voluntary’ and “involuntary” security problems after their exploits in the field. The data collected reveals that a major proportion of vulnerabilities exploited were disclosed immediately as a result of experiencing an exploit.
- \text{exploited}_{(v)}\ represents the exploit of disclosed “voluntary” security problems in the field. Similarly \text{exploited}_{(nv)}\ represents the exploit of disclosed “involuntary” security problems. \text{exploited}_{(sv)}\ and \text{exploited}_{(snv)}\ represent the exploits of hidden “voluntary” and “involuntary” security problems.
- \text{fixed}_{(s)}\ represents the system state where the security problems are fixed. From the data, we observed that for those problem reports that were fixed, the patch release occurred before the problem reports were updated in the bug tracking system as fixed. Hence the state \text{fixed}\ implies that the security problem is fixed and the patch has been released. For the transition to occur, the user must apply the patch.

Figure 2 shows the model with transition rates. The basic assumption is that in the very small period \( \text{d}t \) not more than
Table 4. Fedora security problem statistics [7]

<table>
<thead>
<tr>
<th>Releases</th>
<th>F 1</th>
<th>F 2</th>
<th>F 3</th>
<th>F 4</th>
<th>F 5</th>
<th>F 6</th>
<th>F 7</th>
<th>F 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>31</td>
<td>128</td>
<td>82</td>
<td>154</td>
<td>201</td>
<td>194</td>
<td>92</td>
<td>26</td>
</tr>
<tr>
<td>Voluntary</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
<td>15 (18%)</td>
<td>21 (13.6%)</td>
<td>53 (26.4%)</td>
<td>62 (32%)</td>
<td>29 (31.5%)</td>
<td>12 (46%)</td>
</tr>
<tr>
<td>Involuntary</td>
<td>31 (100%)</td>
<td>128 (100%)</td>
<td>67 (82%)</td>
<td>133 (86.4%)</td>
<td>148 (73.6%)</td>
<td>132 (68%)</td>
<td>63 (68.5%)</td>
<td>14 (54%)</td>
</tr>
<tr>
<td>Downloads</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2864875</td>
<td>1920667</td>
<td>2152583</td>
</tr>
<tr>
<td>From</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10/24/06</td>
<td>04/23/07</td>
<td>11/08/07</td>
</tr>
<tr>
<td>To</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>05/31/07</td>
<td>11/07/07</td>
<td>02/22/08</td>
</tr>
</tbody>
</table>

Figure 1. Security Model

one transition takes place. The transition rates from one state to the other are as follows:

- $\lambda_v$ (good/hidden to disclosed$_v$) - rate of disclosure of “voluntary” security problems. Similarly $\lambda_{nv}$ represents the disclosure of “involuntary” problems.

- $\lambda_{ve}$ (disclosed$_v$ to exploited$_v$) - rate of exploit of disclosed “voluntary” security problems. Similarly $\lambda_{nve}$, $\lambda_{sve}$, and $\lambda_{snve}$ represent the exploits of disclosed “involuntary” security problems, hidden “voluntary” and “involuntary” security problems.

- $\lambda_{sved}$ (exploited$_v$ to disclosed$_sv$) - rate of disclosure of hidden and exploited “voluntary” security problems. Similarly $\lambda_{snved}$ represents the disclosure of hidden and exploited “involuntary” security problems. Note that we are interested in only the first exploit and not subsequent exploits. Hence no transition to exploits from the states disclosed$_{sv}$ and disclosed$_{snv}$). This is because we are interested in observing the influence of the information that an exploit for a problem exists, on the process of correcting the problem. Not the failures.

- $\mu_v$ (disclosed$_v$ to fixed) - rate of fix of disclosed and unexploited “voluntary” security problems. Similarly $\mu_{nv}$ represent fixing of disclosed and unexploited “involuntary” security problems.

- $\mu_{ve}$ (exploited$_v$ to fixed) - rate of fix of disclosed and exploited “voluntary” security problems. Similarly $\mu_{sve}$, and $\mu_{snve}$ represent fixing of hidden and exploited “voluntary” and “involuntary” security problems.
• $\mu_{\text{good}}$ (disclosed to fixed) - rate of fix of exploited and disclosed “voluntary” security problems. Similarly $\mu_{\text{good}}$ represent fixing of exploited and disclosed “involuntary” problems.

• $\mu_{\text{fix}}$ (good to fixed) - rate of fix of hidden security problems before being disclosed or exploited in the field. Similar to $\mu_{\text{good}}$ and $\mu_{\text{involuntary}}$, we include $\mu_{\text{fix}}$ for completeness of the model.

• $\mu_{\text{fix}}$ (fixed to good) - rate of release of fixes for the security problems.

The rates $\lambda_{dv}$, $\lambda_{ev}$, $\lambda_{g}$, $\lambda_{esv}$, $\lambda_{dsnv}$, $\lambda_{dv}$, $\lambda_{esnv}$ for the states $\{\text{disclosed}^{(v)}, \text{exploited}^{(v)}, \text{Good}, \text{exploited}^{(sv)}, \text{disclosed}^{(sv)}, \text{exploited}^{(nv)}, \text{disclosed}^{(nv)}, \text{exploited}^{(nv)}\}$ represent transitions to the same state itself. We base our analysis on information such as the discovery date, disclosure date, exploit date, and fix date of security problems, project download statistics, etc. Using the data, we compute rates of disclosure, exploit, and correction of security problems with respect to in-service time as well as calendar time, and discuss how promptly problems already exploited are resolved compared to those for which an exploit is still unknown.

4 Metrics

Figures 2 shows a sample window of three periods encompassing five problem reports. The problem reports are marked with the time when the problems were reported ($R_i$), exploited ($E_i$), fixed ($F_i$). In Figure 2, $X_i$ gives the time to problem report (TTPR), the amount of time system is operational without a problem being reported, $Y_i$ gives the time to problem correction (TTPE), the amount of time taken to fix a security problem for which an exploit is unknown. In Figure 2, $Z_i$ gives the time to problem exploit (TTEP), the amount of time system is operational without a problem being exploited, and $W_i$ gives the time to problem correction (TTPCE), the amount of time taken to fix an exploited security problem.

4.1 MTTPR

Mean Time To Problem Report (MTTPR) is the mean or average of all TTPRs during a particular interval. For period i-1, the MTTPR is given by the sum of $X1$ and $X2$ divided by two (since only problems 1 and 2 were reported in that period).

4.2 MTTPE

Mean Time To Problem Exploit (MTTPE) is the mean or average of all TTPEs during a particular interval. This

4.3 MTTPCE

Mean Time To Problem Correction (MTTPCE) is defined as the average of all TTPCEs during a particular interval. For period i+1 in Figure 2, Y3 and Y4 indicate the time spent to fix problem reports 4 and 5. For period i+1, the MTTPCE is the total correction time (sum of Y3 and Y4) divided by two (since only problems 4 and 5 were reported in that period).

4.4 MTTPCE

Mean Time To Problem Correction (MTTPCE) is defined as the average of all TTTPCEs in the time interval. For period i in Figure 2, W1 and W2 indicate the time spent to fix exploited problem reports 2 and 3. MTTPCE is the sum of W1 and W2 divided by two (since only problems 2 and 3 were exploited in that period). For security problems that were exploited and disclosed after a certain time period (about 0.5%), the MTTPCE captures the total time spent in correcting the problem knowing that an exploit for the security problem is present. For security problems that were exploited after disclosure (about 1% of the total security problems), the MTTPCE captures only the correction time spent in correcting the problem after it was exploited. In the event one is interested in including the time spent in correction from disclosure to fix date for such problems, then TTPCE can be computed from the point of disclosure until the problem is corrected. Further, one can study the difference between the time spent from disclosure to exploit and from exploit to fix.
4.5 System Usage

The system usage consists of information such as how many systems are in operation during a particular time period and how release of FEDORA do they run, how many have been attacked, what is the downtime, etc. We used the download/installations statistics (available from FEDORA release 6 onwards) to account for the total number of systems operational during a particular period. Ideally, not all the downloads or installations may be in operation during a particular time period. Further, the total number of systems operational in the field may be higher than just the number of downloads. But FEDORA project affirms that these numbers provide a reasonable estimate of the actual number of downloads or installations may be in operation during a particular period. Ideally, not all systems operational during a particular period. 

The inservice time is defined as the cumulative number of systems and the time they have been operational. Hence we use this statistic in approximating the system usage for FEDORA. Based on this we can calculate the inservice time i.e., product of the total number of systems and the time they have been operational. The inservice time is defined as the cumulative sum of the execution time of software [25, 13]. In our case, the inservice time is estimated as

\[
\text{Inservice time}(t) = \sum_{i=1}^{t} \Delta t_i \times n_i
\]  

(1)

\[
\Delta t_i = t_i - t_{i-1}
\]  

(2)

where \(\Delta t_i\) is time interval at \(i\), and \(n_i\) is the total number of systems in usage in the time interval \(i\). In our analysis, we use the number of installations(registrations) for \(n_i\). The inservice time approximates FEDORA usage as a whole, e.g., the usage of all the \(n_i\) systems during a particular week. We use inservice time to analyse the system as a whole.

5 Rates

We show how to compute disclosure rates, exploit rates, and correction rates with respect to inservice time as well as calendar time. We compute calendar time based rates for vulnerabilities from the OSVDB, Bugzilla and FEDORA, and inservice time based rates for FEDORA i.e., considering system usage. We show the general procedures to compute the rates. For specific category of security problems, the rates can be computed using the general procedure but including only problems from the respective categories.

5.1 Problem disclosure rates

We define problem disclosure rate as the number of unique security problems reported in the time period of interest [6, 29]. We focus on hidden security problems and publicly disclosed security problem reports. Considering inservice time, the problem disclosure rate \(\lambda_d\) in the time interval \(i\) can be estimated as,

\[
\lambda_d = \frac{(n_f)_i}{(n_i \times \Delta t_i)}
\]  

(3)

where \((n_f)_i\) represents number of problems reported in time interval \(\Delta t_i\) (e.g., given in minutes). Typically, the number of users \((n_i)\) in time interval \(\Delta t_i\) is very large compared to the total number of problem reports considered \((\min (n_f))\) [Table 4]. Considering calendar time, the problem disclosure rate \(\lambda_d\) in the time interval \(i\) be estimated as,

\[
\lambda_d = \frac{1}{\text{MTTPR}}
\]  

(4)

We may wish to use inservice time based \(\lambda_d\) when discussing the security of the system as a whole, and MTTPR in the context of calendar-based problem correction rate.

5.2 Problem correction rates

We define problem correction rate as the number of security problems resolved in the time period of interest [6]. The exploit unknown problem correction rate can be estimated through the inverse of the Mean Time To Problem Correction (MTTPC\(_{EU}\)), where this time refers only to the specific category of security problem reports in time frame, i.e.,

\[
\mu_d = \frac{1}{\text{MTTPC}_{EU}}
\]  

(5)

Similarly, the exploited problems correction rate can be estimated through the inverse of the Mean Time To Problem Correction (MTTPC\(_E\)).

\[
\mu_d = \frac{1}{\text{MTTPC}_{E}}
\]  

(6)

5.3 Problem exploit rates

We define problem exploit rate as the number of security problem reports exploited in the field in the time period of interest [29]. In practice, data such as the actual field exploits, operational profile of the system, etc., are required to estimate system exploit rate accurately. If we assume that \(P_e\) is the probability that exposure to an open problem report results in a field exploit, then based on the inservice time, the problem exploit rate can be estimated using [29] as,

\[
\lambda_d = \frac{1}{n_i} \times \frac{(n_f)_i}{(n_i \times \Delta t_i)} \times P_e
\]  

(7)

\[
\lambda_{de} = \lambda_d \times P_e
\]  

(8)

where \(\lambda_{de}\) is an estimate of security problem exploit in the field. Considering the calendar time, the problem exploit rate can be estimated using,
\[ \lambda_{de} = \frac{1}{MTTPE_i} \]  

where MTTPE is the Mean Time to Problem Exploit.

6 Numerical results

Table 6 shows the rates computed for FEDORA as a whole using inservice time and from the perspective of a single system using calendar time. Considering inservice time, the proportion of unique voluntary and involuntary problems exploited during the usage of about 2 million systems in a particular week, is very low. Further, the correction rate is very high compared to the exploit rate with respect to inservice time that even the very low percentage of security problems exploited gets patched quickly. Considering calendar time, we find that the correction rate is enough to patch the exploited involuntary type security problems but not the voluntary type problems. This may be because developers may focus more on problems where user does not have control over in exploits.

Table 7 shows the rates computed using the vulnerabilities collected from the OSVDB database. The table shows the average rates of transition from one state to the other. The rates in this table present an overall view of the general pool of vulnerabilities and cannot be attributed to the general behaviour of a single product. On the average, unexploited “voluntary” security problems are disclosed at the rate of 5.5 problems per week. The disclosed “voluntary” security problems are fixed, before being exploited, at the rate of 1.59 problems per week and exploited before being fixed at the rate of 0.15 problems per week. This shows a 28.9% chance that “voluntary” security problems disclosed per week are fixed before being exploited and only a 2.72% chance of being exploited before they are fixed. The disclosed “voluntary” security problems remain in the disclosed state, i.e., without the exploit status or fix status known at the rate of 3.86 problems per week. This leaves about 70.18% chance that “voluntary” security problems disclosed per week remain without the exploit status or fix status known. For “voluntary” security problems exploited after disclosure, the correction rate is high enough that no backlogs are observed.

On the average, unexploited “involuntary” security problems are disclosed at the rate of 17.7 problems per week. The disclosed “involuntary” security problems are fixed, before being exploited, at the rate of 2.19 problems per week and exploited, before being fixed, at the rate of 0.48 problems per week. This shows a 12.37% chance that “involuntary” security problems disclosed per week are fixed before being exploited and only a 2.71% chance of being exploited before they are fixed. The disclosed “involuntary” security problems remain in the disclosed state, i.e., without the exploit status or fix status known at the rate of 15.03 problems per week. This leaves about 84.75% chance that “involuntary” security problems disclosed per week remain without the exploit or the fix status known. For the “involuntary” security problems exploited after disclosure, the correction rate is high enough that no backlogs are observed.

On the average, hidden “voluntary” security problems are disclosed as a result of an exploit at the rate of 9.9 problems per week. Exploited and disclosed “voluntary” security problems are fixed at the rate of 2.65 problems per week and remain in the exploited state, i.e., without the fix status known at the rate of 0.35 problems per week. This shows a 88.3% chance that “voluntary” security problems disclosed per week as a result of exploit are fixed and 11.35% chance of remaining without the fix status known.

On the average, hidden “involuntary” security problems are disclosed as a result of an exploit at the rate of 3 problems per week. Exploited and disclosed “involuntary” security problems are fixed at the rate of 5.4 problems per week and remain in the exploited state, i.e., without the fix status known at the rate of 4.5 problems per week. This shows a 55.6% chance that “involuntary” security problems disclosed per week as a result of exploit are fixed and 44.4% chance of remaining without the fix status known.

During a particular week, there is a high chance of security problems remain with exploit and fix status unknown after being disclosed. This only implies that the solution status is unknown but not necessarily that the problems remain unfixed. Since OSVDB updates the problem reports with the workarounds available with references back to the vendor or third party references, the problem reports may not be updated when the final patch is available from the vendor. In order to identify the availability of the actual patch one may need to refer back to the actual vendor or product. While collecting this data for all vulnerabilities may be tedious, we collect the details for Bugzilla product with the help of associated problem report information in the Bugzilla bug tracking system and discuss the model below.

Table 8 shows the rates for a single product - Bugzilla, computed based on the data presented in Table 5. Since there are no voluntary type problems that have been exploited, transitions rates involving exploited voluntary type problems are not present in the table. Also, the involuntary type security problems are disclosed immediately upon exploit. From Table 8, we find that in general the correction rate for Bugzilla is high enough to keep with the disclosure and exploit rates without any backlogs. We can also observe that exploited voluntary type security problems are corrected at a much faster rate compared to the unexploited involuntary type problems. Based on this and presence of no backlogs, one can assess how promptly a project reacts to a security problem knowing whether it has been exploited in the field or not.
Table 6. FEDORA - Rates

<table>
<thead>
<tr>
<th>Rates</th>
<th>Fedora (whole)</th>
<th>Fedora (single)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security problem disclosure rates in security problems/min (per week)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voluntary</td>
<td>2.5E-10</td>
<td>8.0E-05 (1)</td>
</tr>
<tr>
<td>Involuntary</td>
<td>3.8E-10</td>
<td>2.8E-04 (3)</td>
</tr>
<tr>
<td>Security problem exploit rates in security problems exploited/min (per week)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voluntary</td>
<td>5.6E-11</td>
<td>4.3E-05 (0.6)</td>
</tr>
<tr>
<td>Involuntary</td>
<td>8.4E-11</td>
<td>1.2E-04 (1.4)</td>
</tr>
<tr>
<td>Security problem correction rates in security problems/min (per week)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voluntary</td>
<td>4.1E-05 (0.4)</td>
<td></td>
</tr>
<tr>
<td>Involuntary</td>
<td>2.0E-04 (2.0)</td>
<td></td>
</tr>
</tbody>
</table>

7 Summary

In this paper, we have presented an analysis and classification of large sets of published vulnerabilities from the OSVDB and RedHat bug database. Our analysis revealed that 35.5% of the published vulnerabilities have been exploited with 1.3% of the vulnerabilities exploited after disclosure and 34.2% disclosed as a result of experiencing an exploit, and 64.5% of the published vulnerabilities disclosed through means other than actual exploits like testing activities. The proportion of voluntary exploit problems is about 12.8% with values ranging from 7% to as high as 82% for individual projects. We have quantified security problems disclosure rate, exploitation and correction rates using the available data. We have discussed an approach in understanding security problem-exploit relationship and the impact of such exploits in correcting the security problems. Using our approach one can model a project team’s response to fix vulnerabilities which have been exploited in the field and vulnerabilities for which an exploit is unknown. Also, we have shown the use of inservice time and calendar time, and their use in security analysis from the perspective of the system as a whole and a single system.

References


Table 8. Rates - Number of security problems per minute (per week) - Bugzilla

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Mean per minute (per week)</th>
</tr>
</thead>
<tbody>
<tr>
<td>good/hidden to disclosed&lt;v&gt;</td>
<td>disclosed&lt;v&gt;</td>
<td>3.1E-006 (0.03)</td>
</tr>
<tr>
<td>disclosed&lt;v&gt;</td>
<td>to fixed</td>
<td>4.6E-004 (4.63)</td>
</tr>
<tr>
<td>good/hidden to exploited&lt;snv&gt;</td>
<td>disclosed&lt;snv&gt;</td>
<td>1.1E-005 (0.11)</td>
</tr>
<tr>
<td>disclosed&lt;snv&gt;</td>
<td>disclosed&lt;snv&gt;</td>
<td>disclosed immediately</td>
</tr>
<tr>
<td>good/hidden to</td>
<td>disclosed&lt;nv&gt;</td>
<td>1.5E-005 (0.15)</td>
</tr>
<tr>
<td>disclosed&lt;nv&gt;</td>
<td>fixed</td>
<td>1.5E-003 (14.6)</td>
</tr>
</tbody>
</table>


